EK-DIA20-TM-002

DIA20
IBUS ADAPTER
UNIT DESCRIPTION

digital equipment corporation • marlborough, massachusetts

1st Edition, March 1975
2nd Edition (Rev), July 1976

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

ILLUSTRATIONS

TABLES

PREFACE

The DIA20 theory manual contains three levels (chapters) of descriptions:

1. Overview
2. Functional Description
3. Logic Description

The overview chapter describes basic operation and identifies the major logic elements. The functional description chapter first describes the operation of the EBus and the IBus (the DIA20 is a bus adapter interfacing one bus to the other). It then describes DIA20 operation in relation to the various bus signals and commands. The level of detail in this chapter is limited to a functional perspective; it does not provide specific details.

The third chapter, logic description, contains a comprehensive description of the basic logic elements introduced in the overview. DIA20 operation is again described for the various bus operations; the description is more detailed and at the logic level, however. By using print prefixes, this chapter provides a direct index into the logic print set.

CHAPTER 1
OVERVIEW

1.1 GENERAL INFORMATION

The DIA20 IBus Adapter Control (IBC) is a KL10 controller which allows devices that operate on the KA10/KI10 I/O Bus (IBUS) to be connected to the KL10 system. The IBC interfaces directly to the IBus and connects to the KL10 via the EBus. The IBC serves mainly to extend EBus operation to the KA10/KI10 devices, translating EBus signals and commands to IBus signals and commands. It also provides voltage level conversion between the two buses.

The IBC consists of three double height hex-boards (2-M8550 and 1-M8551) that are mounted in the system I/O cabinet. TTL integrated circuitry is used throughout except at the IBus interface where a mixture of discrete components and integrated circuitry are used to drive and receive signals on the negative I/O Bus. Two IBus outputs are provided allowing two separate chains of I/O devices to be connected. Quicklatch connectors are employed and each IBus can have a maximum length of 100 feet.

1.2 BASIC OPERATION

The IBC connects between the EBus and the IBus as shown in Figure 1-1. It performs the following major functions and operations.

a. IBus Command by Default
b. Data Transfer
c. PI Control
d. Bus Level Conversion and Discharge

1.2.1 IBus Command by Default

When the IBC detects the start of EBus commands that select a controller by means of device code, it generates these same commands on the IBus (simulating I/O Bus control and timing) provided that no other KL10 controller responds to (is addressed by) the EBus command. Commands directed to I/O devices by default, when no KL10 device is addressed, are DATA OUT (CONO/DATAO) and DATA IN (CONI/DATAI).

1.2.2 Data Transfer

The IBC provides a 36-line bidirectional data path between buses. Data is gated directly from the EBus to the IBus for DATA OUT operations. For DATA IN operations, IBus data is stored in a data buffer and the buffer outputs gated to the EBus.

1.2.3 PI Control

The IBC relays priority interrupt requests generated by KA10/KI10 devices from the IBus to the EBus. It then responds to the PI SERVED command, like other KL10 controllers, by comparing the interrupting channel number(s) with the channel number being served. When a match occurs, it places its physical number on the EBus. In response to the PI ADR command, the IBC differs from other KL10 controllers in that it must relay the channel number being served to the IBus. Then it must simulate I/O Bus PI control signals and timing to collect any API response information. This information, which consists of an interrupt function code and address, is then transmitted on the EBus. If there is no API response, the IBC transmits a standard interrupt function code.

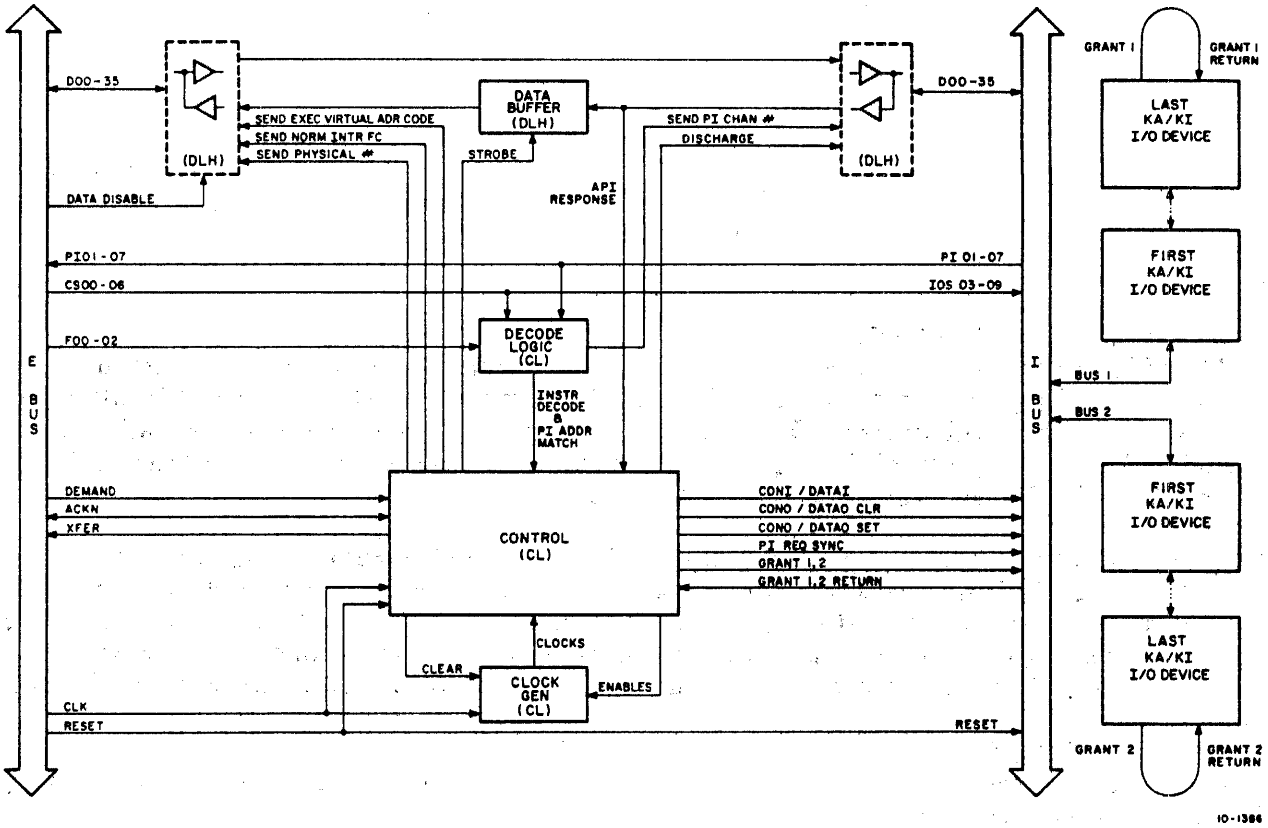


Figure 1-1 DIA20 Simplified Block Diagram

1.2.4 Bus Level Conversion and Discharge

The IBC provides voltage level conversion between the two buses. EBus signals are nominally 0 V and +3 V. IBus signals are 0 V and -3 V. The adapter also discharges the IBus data lines at the completion of all commands except PI SERVED. (It is not necessary for this command since no IBus action takes place.) Discharge is accomplished by returning the data lines to a high negative voltage for a short time to drive the lines rapidly back to -3 V. Discharge time is four EBus clock periods.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 BUS OPERATION

The EBus connects all KL10 controllers to the EBox; the IBus connects all KA10/KI10 I/O controllers to the IBC. Information flow on the two buses is shown in Figure 1-1.

2.1.1 Controller Selection

All KL10 controllers connect to and share the EBus. Similarly, all KA10/KI10 controllers connect to and share the IBus. Because of the common connections, each bus has controller select lines so that commands can be addressed to a specific device. EBus devices are addressed by either device code designation, physical number, or PI channel number being served. IBus select lines address KA10/KI10 controllers by device code only.

All controllers, with one important exception, are assigned a specific address or device code. Codes assigned to KL10 controllers do not conflict with those previously assigned to KA10/KI10 (DECsystem-10) controllers. However, some PDP-6 device codes have been supplanted by KL10 assignments. For commands utilizing device code selection, each controller decodes the select lines and compares them with its hard-wired device code designation. If a match occurs, it responds to the command.

The DIA20 adapter is the exception in that it has no device code assignment. It responds to EBus commands, not because it was selected directly, but by default, because no other KL10 controller was selected. There can be only one device on a bus addressed in this manner; consequently, only one IBC can be connected in a single-processor system.

Both buses provide for seven bits of device code. Lines CS 00–06 are used on the EBus and lines IOS 03–09 are used on the IBus. There are actually fourteen IOS lines as a pair of complementary lines are used for each bit of device code address. The double-railed signals are provided for ease in decoding. It should be noted that some KA10/KI10 devices can have more than one device code if a large amount of control and status information is to be handled by the IBus commands.

In its capacity of extending EBus operation to KA10/KI10 controllers, the IBC relays the information on the CS lines directly to the IOS lines. EBox commands selecting by device code can then address IBus devices through the adapter. These commands are CONO, DATAO, CONI, and DATAI.

In addition to a device code, KL10 controllers have a physical address, or physical number. When generating a PI ADR IN command, the EBox addresses a controller by placing its physical number, encoded in binary, on the four high-order controller select lines CS 00–03. Each controller must decode the select lines and compare them with its hard-wired physical number. The controller responds if a match occurs. The IBC has a physical number of 15_{10} and is addressed just as any other KL10 device. It does not respond by default as for device code addressing.

KL10 controllers are also addressed by the number of the priority interrupt channel being serviced by the EBox. This occurs for the PI SERVED command when the channel number, encoded in binary, is placed on the three low-order controller select lines CS 04–06. Each EBus controller, when interrupting, compares this number with the interrupting channel number and, if a match occurs, sends its physical address to the EBox via the data lines. The IBC, while having no interrupt capability of its own, monitors interrupt requests on the IBus, performs the comparison, and responds like other KL10 controllers.

Since more than one controller can respond to the PI SERVED command, the physical address must be placed on the data lines so as to not interfere with the response from other devices. This is accomplished by having each controller assert only the data line corresponding to its physical number. For example, the IBC asserts line 15.

2.1.2 Priority Interrupt Requests

Part of the control information sent to a controller (usually by a CONO command) is a 3-bit interrupt channel number encoded in binary. Both EBus and IBus controllers store this number. When conditions are met for initiating an interrupt request, a controller generates a signal on one of seven request lines PI 01–07. The request line asserted depends on the stored channel number. A zero channel number causes no request and provides a means for the programmer to inhibit interrupt activity.

The IBC has no interrupt capability of its own and does not store a channel number. Instead, in its capacity of allowing KA10/KI10 devices to operate on a KL10 system, the IBC relays lines PI 01–07 from the IBus directly to the EBus.

2.1.3 Command Control

Function lines F 00–02 are used on the EBus to specify which EBox operation is to be performed. Depending on the operation, the lines are binary encoded as shown in Table 2-1.

Table 2-1 EBus Function Codes

Command	Function Lines			Function Code
	F00	F01	F02	
CONO	0	0	0	0
CONI	0	0	1	1
DATAO	0	1	0	2
DATAI	0	1	1	3
API 4 PI SERVED	1	0	0	4
API 5 PI ADR IN	1	0	1	5
–	1	1	0	6
–	1	1	1	7

The EBox begins an operation by asserting both the controller select and function lines. It then raises the DEMAND line after a fixed delay. The delay is to ensure that the select and function lines are valid when DEMAND is received by the controllers, even for worst-case skew conditions on the bus cable and between bus transmitters and receivers. Each controller decodes the function lines to determine what command is to be performed. The controllers also decode the select lines. If selected by device code or physical number, that is, if the select lines correspond to its hard-wired address, the controller responds to DEMAND by starting the operation and by asserting ACKN. There should be no more than one such response on the bus and ACKN must be generated within 150 ns of receiving DEMAND. After the selected controller has performed the appropriate action, depending on the function code, it signals the EBox by asserting XFER. The EBox then drops DEMAND ending the operation. ACKN and XFER are cleared in the controller on the trailing edge of DEMAND. If XFER is not received within a specific time interval after the assertion of DEMAND, a time-out sequence is provided in the EBox that ends the operation by clearing DEMAND.

The IBC has a physical address and responds to physical number selection as described above. Because it has no device code designation, the IBC cannot respond to device code selection in the same manner as other controllers. Instead of responding directly by generating ACKN and starting an operation immediately, it responds indirectly by receiving or monitoring ACKN and delaying any action until at least 250 ns after the start of the command. Then, if no other device on the bus has generated ACKN, the IBC responds by generating the equivalent command on the IBus. Consequently, EBus commands are directed to the IBus when a KL10 controller has not been addressed. It should be noted that with a DIA20 in a system, time-outs by the EBox (those caused by a device code not being recognized by any controller) will not occur because the IBC responds by default for this case and generates XFER. This is true even if the addressed device does not exist on the IBus.

Whereas command control on the EBus is asynchronous and follows a handshaking sequence, commands on the IBus (with the exception of API control) are synchronous, with the duration and timing of the control signals determining the duration of an operation independent of bus length. The IBus control lines are asserted by the IBC, closely simulating the I/O Bus outputs from a KI10 CPU. As does the KI10, the IBC provides two output connections and a fast or slow mode of operation. The slow mode generates the correct timing for controllers designed to operate with the KA10 CPU. The fast mode accommodates controllers designed for the faster KI10. The mode of operation is not program selectable and is controlled by a jumper wire (W1) on the DIA20 control board (M8551). The control lines asserted by the IBC for the various EBox commands are listed in Table 2-2.

Table 2-2 IBus Control Signals

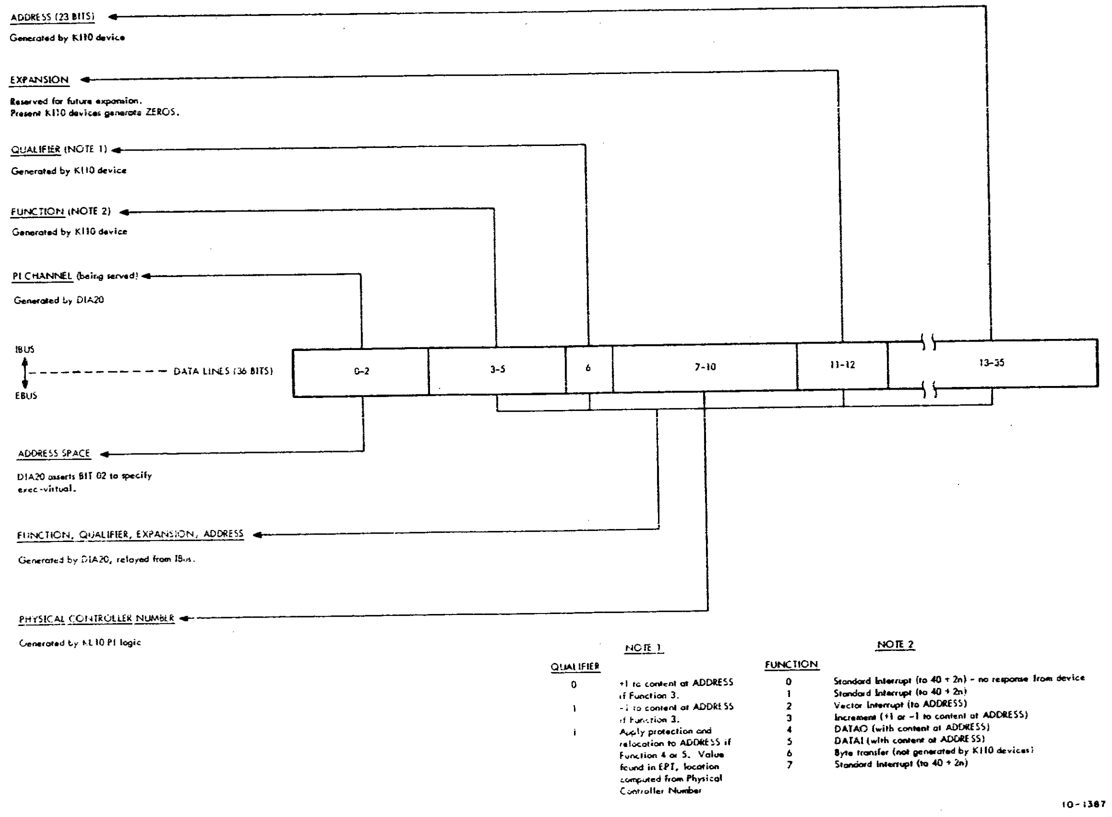
EBus Command	IBus Control Signals
CONO	CONO CLR, CONO SET
CONI	CONI
DATAO	DATAO CLR, DATAO SET
DATAI	DATAI
PI SERVED	
PI ADR IN	PI REQ SYNC, GRANT 1, GRANT 2 (*GRANT 1 RETURN, *GRANT 2 RETURN)

*GRANT signals returned from end of bus.

There is a corresponding control line (or pair of lines) for each of the EBus operations: CONI, DATAI, CONO, and DATAO. The IBC asserts these lines on both IBus outputs simultaneously. There is a single control line for each of the DATA IN (CONI and DATAI) commands; the duration of these signals determines when and for how long the device places information on the IBus data lines. The duration changes with fast and slow modes of operation. To perform each of the DATA OUT (CONO and DATAO) commands, two lines are asserted. The CONO/DATAO CLR pulse is the first signal generated and its normal function is to prepare the device to receive information. The CONO/DATAO SET pulse is generated next. It is normally used by the device to strobe the outgoing data off the IBus data lines. Both the width and spacing of the CLR and SET pulses change with the mode of operation.

Of the two PI control commands generated by the EBox, only PI ADR IN initiates any IBus action. This command causes the PI REQ SYNC and GRANT lines to be asserted on the IBus. These signals are used by the KI10 controllers which utilize an automatic priority interrupt (API) system. API response is also used by KL10 controllers; it allows the interrupting device to specify an interrupt address and function. The various interrupt functions are listed in Figure 2-1.

The PI REQ SYNC level is generated first on both IBus outputs. Its function is to prepare the KI10 device for receiving a GRANT level. At the same time, the channel number being served is asserted on data lines 0–2. GRANT 1 is then generated on bus 1. A GRANT signal is bused through KA10 devices and it is relayed through KI10 controllers that are not interrupting or that are interrupting on a channel other than the one being served. The first KI10 controller on the bus that is interrupting on the specified channel responds by placing a function code and an address (if any) on the IBus data lines.


ADDRESS (23 BITS)
Generated by KI10 device
EXPANSION
Reserved for future expansion.
Present KI10 devices generate ZEROS.
QUALIFIER (NOTE 1)
Generated by KI10 device
FUNCTION (NOTE 2)
Generated by KI10 device
PI CHANNEL (being served)
Generated by DIA20
IBUS
DATA LINES (36 BITS)
EBUS
0-2
3-5
6
7-10
11-12
13-35
ADDRESS SPACE
DIA20 asserts BIT 02 to specify exec-virtual.
FUNCTION, QUALIFIER, EXPANSION, ADDRESS
Generated by DIA20, relayed from IBus.
PHYSICAL CONTROLLER NUMBER
Generated by KL10 PI logic
10-1387

NOTE 1

QUALIFIER	
0	+1 to content at ADDRESS if Function 3.
1	-1 to content at ADDRESS if Function 3.
i	Apply protection and relocation to ADDRESS if Function 4 or 5. Value found in EPT, location computed from Physical Controller Number

NOTE 2

FUNCTION	
0	Standard Interrupt (to 40 + 2n) - no response from device
1	Standard Interrupt (to 40 + 2n)
2	Vector Interrupt (to ADDRESS)
3	Increment (+1 or -1 to content at ADDRESS)
4	DATAO (with content at ADDRESS)
5	DATAI (with content at ADDRESS)
6	Byte transfer (not generated by KI10 devices)
7	Standard Interrupt (to 40 + 2n)

Figure 2-1 API Response Bit Format

This information is gated to the EBox by the IBC. The responding controller also blocks the GRANT level from proceeding down the bus ending the IBus operation. With no API response or if there are only KA10 devices connected, GRANT 1 is passed through all devices and is returned to the IBC on another line by a terminator connection at the end of the bus. This signal is called GRANT 1 RETURN. The return on bus 1 then causes the IBC to generate a GRANT 2 level on bus 2. Operation is the same as for GRANT 1 and an API response ends the operation. If the IBC receives a GRANT 2 RETURN, indicating that a KA10 device initiated the interrupt, it generates and sends a standard KA10 interrupt function code (FC = 1) to the EBox.

2.1.4 Data Lines

Both buses have 36 data lines for transferring information to and from controllers. During the CONO and DATAO commands, EBus data is gated through the DIA20 to the KA10/KI10 controllers on the IBus. During the CONI and DATAI commands, IBus data is stored by the IBC in a 36-bit buffer register and the buffer outputs transmitted to the EBox. The IBus data is stored because it must be asserted on the EBus after the completion of the IBus command; that is, after the IBus data lines have been negated by the selected I/O device.

The data lines are also used to transfer the API response from the controllers to the EBox during the PI ADR IN command. As for CONI and DATAI operations, the information from the I/O devices is stored in the IBC data buffer.

2.1.5 Miscellaneous Signals

The CLK line on the EBus provides a continous clock train that can be used by KL10 controllers to sequence logic and to synchronize its operation with the KL10 system. The IBC uses the CLOCK signal to sequence its clock generators. There is no corresponding clock signal on the IBus.

RESET is used by all controllers on both buses to clear control logic and any status indicators to some initial state. It specifically should clear any conditions that cause interrupts. It is generated by the EBox on power-up by CONO APR 200000, or by a diagnostic function via the 10/11 interface. The IBC uses RESET to clear its own control logic (it has no error or status flags) and it relays the signal from the EBus to the IBus to clear all KA10/KI10 devices.

The DATA DISABLE signal is asserted on the EBus during certain diagnostic operations. It is used by the IBC and its function is to disable a KL10 controller's EBus data transmitters, usually between clocks, when the controller is single-stepped through a diagnostic I/O operation. This is necessary because the otherwise asserted DATA lines (during a CONI and DATAI, for example) would not allow dialogue over the EBus between the DTE20 and the rest of the system, thus preventing a diagnostic from checking the execution of the I/O instruction. There is no signal comparable to DATA DISABLE on the IBus.

NOTE

An IBus transmitter for RDI PULSE and an IBus receiver for RDI DATA and DRUM SPLIT are provided in the DIA20 (to preserve electrical integrity), but these signals are not used on the EBus. The IBus read-in (RDI) signals are not required because the KL10 is bootstrapped under control of the Master Front End Processor. The DRUM SPLIT signal is not required because the particular RMW memory references this signal inhibited in the KA10/KI10 CPU (to minimize data overruns in fast I/O devices time-sharing memory with the processor via the DF10) are not implemented by the KL10 CPU.

2.1.6 Bus Signal Summary

Table 2-3 summarizes the functions of various EBus and IBus signals.

Table 2-3 IBC Bus Interface Summary

EBus			IBus	
Function	Name		Name	Function
Select KA10/KI10 I/O devices by device code (by default). Select KL10 devices by device code, physical no., and PI channel no. being served.	CS 00–06 →		IOS 03–09 →	Select KA10/KI10 I/O device by device code.
Specify function to be performed (DATAO, DATAI, etc.).	F 00–02 →			
Executes function in selected KA10/KI10 device if no ACKN from KL10 device.	DEMAND →		CONI/DATAI →	Executes DATA IN operation in selected device.
Received–Clears IBC Transmitted–IBC acknowledges PI ADR IN command.	ACKN ↔ [illegible]	D I A 2 0	CONO/DATAO CLR →	Readies selected device for DATA OUT operation.
			CONO/DATAO SET →	Executes DATA OUT operation in selected device.
			PI REQ SYNC →	Synchronizes KI10 devices for API response enable.
Signals function is executed.	XFER ←		GRANT 1, 2 →	Enables API response in KI10 devices.
			GRANT 1, 2 RETURN ←	Signals no API response on bus.
Priority interrupt requests.	PI 01–07 ←		PI 01–07 ←	Priority interrupt requests.
Data lines. Transfer data and control information to selected device. Transfer data, status, physical no., and API response from selected device.	D 00–35 ↔		DATA 00–35 ↔	Data lines. Transfer data, control information, and PI channel no. being served to selected device. Transfer data, status, and API response from selected device.
CPU clock. Sequences control logic.	CLK →			
Resets devices to initial state.	RESET →		RESET →	Resets device to initial state.
Disables EBus data transmitters.	DATA DISABLE →		[illegible] FREEW	

2.2 ADAPTER OPERATION

Following is a brief description of DIA20 operation for the various EBox commands. A detailed logic description, together with timing diagrams, is provided in Chapter 3.

2.2.1 CONO/DATAO

During a CONO/DATAO, the EBox asserts the controller select lines and function lines and then asserts the DEMAND line, after a deskewing delay. The device code information on the select lines is relayed through the IBC to the KA10/KI10 I/O devices. If, after receiving DEMAND, no other KL10 controller responds to the CONO or DATAO command (function code = 0 or 2) by asserting ACKN within two EBus clock periods, the IBC generates the command on the IBus by first gating the outgoing data from the EBox to the IBus data lines. The IBC, with the data lines still asserted, then generates the CONO or DATAO CLR signal followed by the CONO or DATAO SET signal. A selected I/O device uses the CLR and SET pulses to gate or strobe the outgoing data. Shortly after the trailing edge of the SET signal, the IBus data lines are gated off and discharged by the IBC. This ends the operation on the IBus. The IBC then asserts XFER, causing the EBox to drop DEMAND and end EBus operation. When DEMAND goes false on the bus, the IBC shuts down clearing XFER.

2.2.2 CONI/DATAI

During a CONI/DATAI, the EBox asserts the controller select lines and function lines and, after a deskewing delay, raises the DEMAND line. The device code information on the select lines is relayed through the IBC to the KA10/KI10 I/O devices. If, after receiving DEMAND, no other KL10 controller responds to the CONI or DATAI command (function code = 1 or 3) by asserting ACKN within three EBus clock periods, the IBC generates the command on the IBus by asserting either the CONI or DATAI control signal. When a selected I/O device receives the control signal, it gates data onto the IBus data lines for as long as the signal is asserted. At the trailing edge of the CONI or DATAI level, the IBC strobes the information on the data lines into a 36-bit buffer. It then discharges the lines ending the IBus operation. With the information stored in the data buffer asserted on the EBus data lines, the IBC generates XFER, which causes the EBox to collect the incoming data and to end the operation by clearing DEMAND after a deskewing delay. When DEMAND goes false on the bus, the IBC shuts down clearing the data lines and XFER.

2.2.3 PI SERVED/PI ADR IN

Each controller stores a priority interrupt channel number which is loaded by a DATA OUT operation (Paragraph 2.2.1). When an interrupt condition occurs, one of seven request lines can be raised by a device. The line that is asserted depends on the stored channel number. The IBC relays the request lines from the KA10/KI10 I/O devices to the EBox. More than one request can be asserted at any time and more than one device can be asserting the same request line. The EBox detects all interrupts and resolves the interrupt priority on a channel number basis (lowest channel number has highest priority). When it is ready to serve a particular channel number, it generates the PI SERVED command. It first asserts the function lines and, at the same time, places the channel number on the controller select lines. It then asserts DEMAND after a fixed delay. The IBC responds to the PI SERVED command (function code = 4) and the DEMAND signal by determining if an I/O device is interrupting on the channel number being serviced. If so, it asserts data line 15 on the EBus. This line corresponds to the IBC physical number address of 15. If there is no interrupting I/O device for the specified channel, no action is taken by the IBC. More than one KL10 controller can respond to the PI SERVED operation (each asserting the appropriate data line) and, after waiting at least 400 ns for all responses, the EBox strobes the data lines and clears DEMAND. On the trailing edge of DEMAND, the IBC shuts down clearing data line 15 if it had been asserted.

After issuing the PI SERVED command to determine which controller(s) is requesting on a particular PI channel, the EBox resolves interrupt priority on a physical number basis (lowest physical number has highest priority) and issues the PI ADR IN command. It asserts the function lines and, if the IBC is to be serviced, places a physical number of 15 on the controller select lines. It also places the channel number on the select lines since the I/O devices could be interrupting on more than one channel. The EBox then asserts DEMAND after a deskewing delay. The IBC responds to the PI ADR IN command (function code = 5) and to DEMAND by gating the binary encoded channel number on the select lines to IBus data lines 00–02 and by asserting ACKN on the EBus. It then generates PI REQ SYNC to all KA10/KI10 devices on both IBus outputs and, shortly after, asserts GRANT 1 on port 1. The SYNC and GRANT signals are bused through KA10 type controllers. KI10 controllers implement an API system and use PI REQ SYNC to synchronize the response to the GRANT levels that follow.

Each KI10 controller that is interrupting sets a synchronizing flip-flop when it receives the PI REQ SYNC level. Also, the controller compares the channel number being served (encoded on the data lines) to the PI channel number stored in its PI register. If there is no match (device interrupting on another channel), the device uses the synchronizing flip-flop to gate the GRANT level on to the next I/O device. The first API device detecting a match responds by blocking the path of the GRANT level down the bus and by asserting an interrupt function code on the IBus data lines 03–05. It can also assert a 23-bit interrupt address on lines 13–35 and a qualifier bit on line 06. Format for the API response on the data lines is shown in Figure 2-1.

The IBC detects an API response by monitoring data lines 03–05. If one or more of these lines go true, indicating an interrupt function code has been generated, it strobes the data lines into its 36-bit data buffer and transmits the API response to the EBox on the EBus data lines. The IBC then clears GRANT and PI REQ SYNC. The trailing edge of the SYNC signal causes the KI10 device to remove the API response from the data lines. The IBC then discharges the lines and generates XFER ending EBus operation.

If there is no API response on IBus port 1, the GRANT 1 signal is relayed through KI10 devices and bused through KA10 devices until it reaches the end of the bus. The terminator module plugged into the last device then returns the GRANT 1 level on another line. After receiving GRANT 1 RETURN, the IBC clears GRANT 1 and generates GRANT 2 on port 2. An API response on port 2 causes the information to be sent to the EBox and the operation to end, just as described for port 1. Operation on both ports is identical and if a GRANT 2 RETURN signal is received, it again indicates that an API device did not interrupt on the channel being serviced. With no API response on either port, the asserted interrupt request must be from a KA10 device. The IBC then transmits a standard KA10 interrupt function code (FC = 1) to the EBox by asserting EBus data line 5. It also clears GRANT 2 and PI REQ SYNC. XFER is then generated ending EBus operation.

NOTE

If no IBus is connected to a port, GRANT and GRANT RETURN must be jumpered together at the IBus connector. With no jumper installed, PI ADR IN operation is unspecified.

CHAPTER 3
LOGIC DESCRIPTION

This chapter gives a detailed description of DIA20 (IBC) operation at the logic level. Reference should be made to the DIA20 print set and to Figure 3-1. The IBC consists of the following major logic elements.

a. Data Buffer
b. Decode Logic
c. Clock Generators
d. Control Logic

3.1 DATA BUFFER

The data buffer consists of 36 D-type flip-flops used to store the incoming data from the KA10/KI10 I/O devices. The buffer outputs, DLH1 EBUS D00–35 OUT, drive the 36 EBus data line transmitters directly. The incoming data, which is the 36 IBus data receiver outputs, DLH2/3 KI D00–35 IN, is clocked into the buffer by control flip-flop CL3 EBUS STROBE. The buffer is cleared after every operation by CL1 CLEAR, remains cleared during DATA OUT and PI SERVED operations, and contains information only when clocked during the CONI, DATAI, and PI ADR IN operations.

3.2 DECODE LOGIC

The IBC contains logic for decoding the function, controller select, and priority interrupt request lines. The function lines are decoded by a four to ten line decoder, with only six of the outputs having to be used to specify the six EBox operations. The fourth input to the decoder, in addition to the three function line inputs (F 00–02), is DEMAND. It is used to prevent glitching on decoder outputs 0–7 while the function lines are changing. In addition, it delays assertion of the outputs until CL1 CLEAR goes false (function of DEMAND). This allows the outputs to be used to edge-trigger flip-flops held off by CLEAR.

The binary encoded high-order select lines (CS 00–03) are decoded to assert CL1 PHY COMPARE when the IBC physical number address of 15 appears on the lines. The binary encoded low-order select lines (CS 04–06) are used by a data selector/mixer circuit to select one of the seven PI request lines (PI 01–07) from the I/O devices. The line selected depends on the binary value of CS 04–06. During the PI SERVED and PI ADR IN commands, the output of the mixer, CL1 INTR COMPARE, will be true if the channel being serviced (encoded on CS 04–06) selects an asserted request line. The INTR COMPARE level causes the IBC to execute the PI SERVED command. Together with PHY COMPARE, it also causes execution of the PI ADR IN command by asserting CL1 PI ADR IN COMPARED.

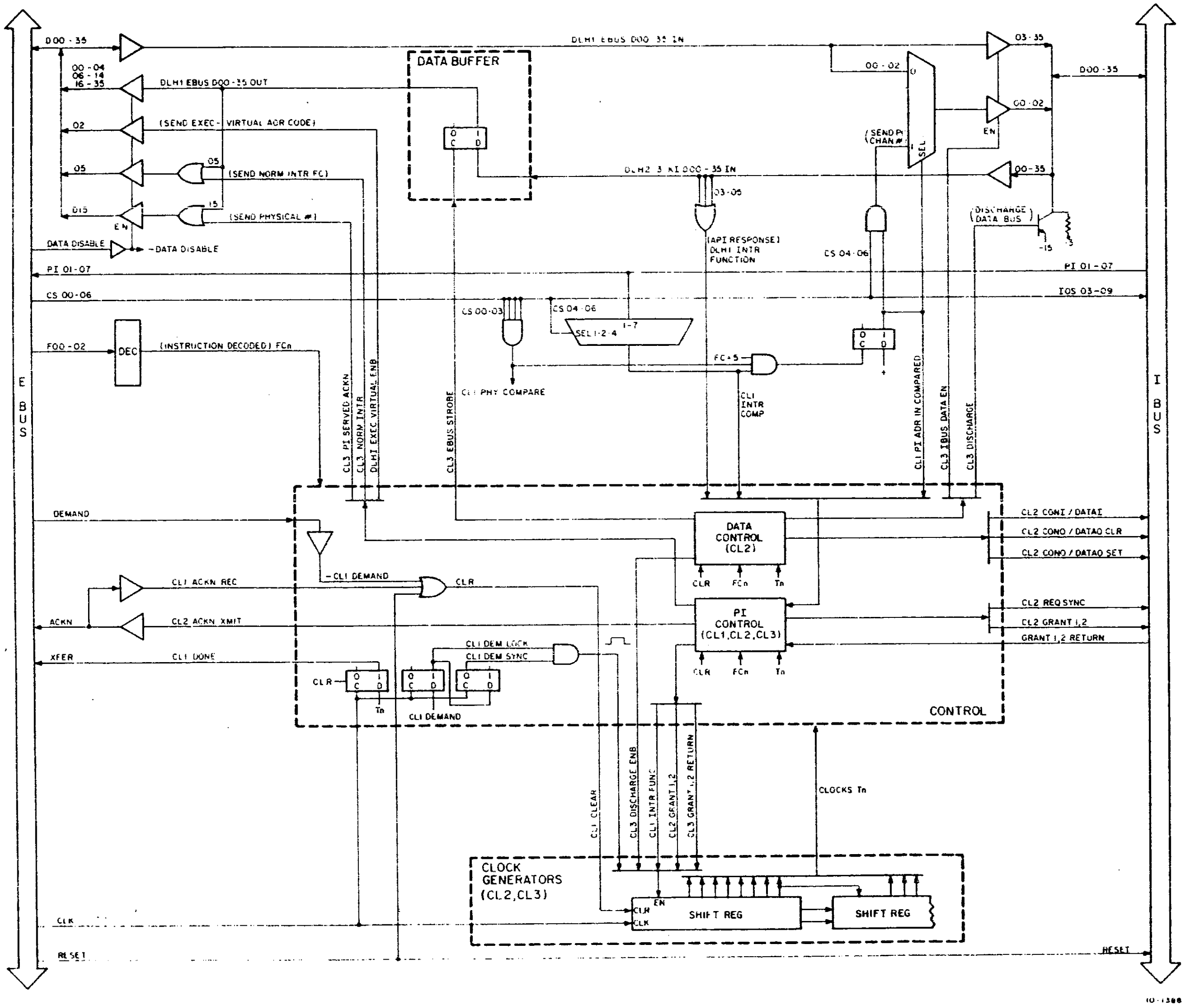


Figure 3-1 DIA20 Detailed Block Diagram

3.3 CLOCK GENERATORS

A series of shift registers are used to generate clocks and time states in the DIA20. The registers are clocked by EBus CLK (received as CL1 CLOCK and hereafter called IBC clock). The register outputs, asserted one after the other, provide a clock train to sequence the logic through all commands.

The shift registers can be divided into four groups depending on the function performed. Clocks CL2 T00–T27, the outputs from a 24-bit register, sequence the logic generating the IBus commands. Clocks CL2 GRANT 1 T00–T17 and GRANT 2 T00–T17 are generated by two 16-bit registers that serve in lieu of one-shots to provide a time-out interval when an IBus port is not connected during the PI ADR IN command. CL3 GRANT RETURN T000–T375, a 4-bit register, provides a logical delay in clearing the GRANT logic during the same command. The last group, CL2 DISCHARGE T00–T17, generates the timing signals required to discharge the IBus data lines and to terminate IBC operation.

3.4 CONTROL LOGIC

A description of control logic operation follows. Timing diagrams are included to show logic signal sequence for each bus command.

3.4.1 Start-up and Clear

The IBC is held in the cleared state by the negation of DEMAND whenever an EBus command is not in progress. It is also cleared during an operation if RESET is asserted on the bus or if another KL10 controller is generating an ACKN signal. CL1 CLEAR performs the clearing function by direct clearing the clock generator shift registers, the control flip-flops, and the 36-bit data buffer.

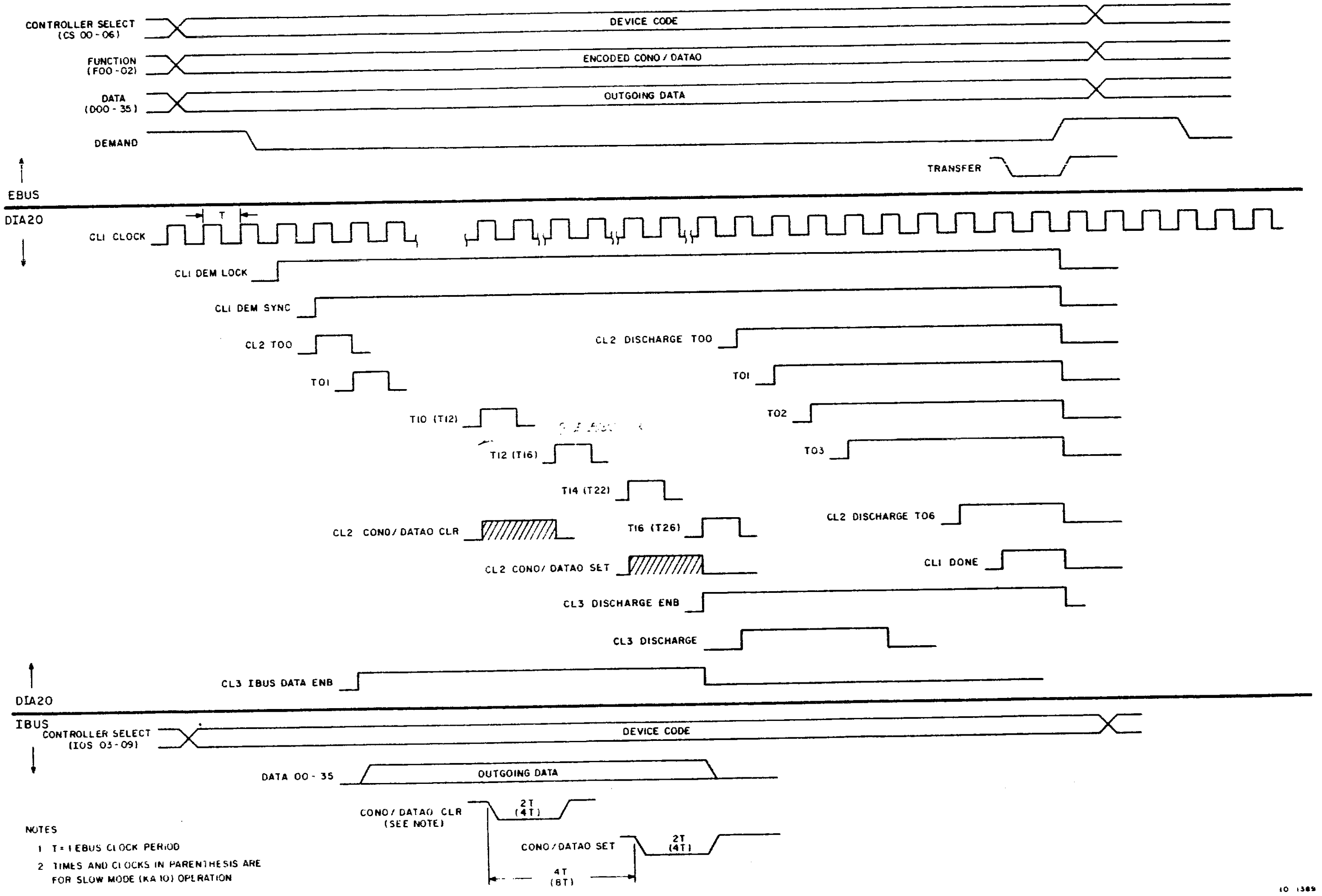
At the start of an operation DEMAND is asserted by the EBox, CLEAR goes false, and the IBC is released to respond to the command. The first IBC clock with DEMAND true sets the synchronizing flip-flop CL1 DEMAND LOCK, asserting the data input to the CL1 DEMAND SYNC flip-flop and to the shift register CL2 T00–T07. Input to the shift register is also conditioned by DEMAND SYNC (0). When the next IBC clock occurs the first clock generator output T00 will be asserted and DEMAND SYNC will set, cutting off the input to the shift register. Removing the input one clock period after DEMAND LOCK goes true causes T00 and subsequent shift register outputs to have a duration of one IBC clock period.

In addition to acting with DEMAND SYNC to define clock duration, DEMAND LOCK also serves a synchronizing function for the case where DEMAND and the leading edge of the IBC clock race and cause DEMAND LOCK to dither (i.e., to be clocked to an unresolved state). This can occur when the data input has been present for only a short time before the flip-flop is clocked and minimal energy is coupled into the circuit. If T00 was generated directly by DEMAND and the first occurring clock, dithering might occur in the clock generator output T00 instead of the synchronizing flip-flop, possibly causing the IBC to malfunction.

With T00 asserted, a clock train will be generated by the shift register as it is shifted by the IBC clock. The EBox command will then be executed by the IBC unless another controller responds to DEMAND by raising ACKN on the bus. ACKN generates CLEAR which clears the clock generator, immediately shutting down the IBC and holding it cleared while the other controller executes the command. Assuming no ACKN response, IBC operation for the various commands is as follows.

3.4.2 CONO/DATAO Operation

With reference to Figure 3-2, the EBox first asserts the select, function, and data lines. It then asserts DEMAND, starting the clock generator as described in Paragraph 3.4.1. At time T01, CL3 IBUS DATA ENB flip-flop is set generating DLH1 IBus DATA ENB A and B, which enable the inputs to the IBus data transmitters. This causes the outgoing EBus data to be gated directly to the IBus.



NOTES

1 T = 1 EBUS CLOCK PERIOD

2 TIMES AND CLOCKS IN PARENTHESIS ARE FOR SLOW MODE (KA10) OPERATION

Figure 3-2 CONO/DATAO Timing Diagram

If the IBC is jumpered for fast mode operation, the first of the IBus control signals is generated by T10. At this time, either CL2 CONO CLR or CL2 DATAO CLR (depending on the function code) is clocked on asserting the corresponding bus control line. The flip-flop is then direct cleared at time T12, giving a CONO/DATAO CLR signal equal to two clock periods. In slow mode, times T12 and T16 are gated to set and clear the flip-flop to give a CLR pulse duration of four clock periods.

The CONO/DATAO SET signal, which has the same duration as the CLR signal, is asserted next. Times T14 and T16 set and clear CL2 CONO SET or CL2 DATAO SET in fast mode. T22 and T26 are used in slow mode. When the SET flip-flop clears, CL3 DISCHARGE ENB is edge-triggered to the ONE state. DISCHARGE ENB clears the IBUS DATA ENB levels, removing the outgoing data from the IBus data lines. The next IBC clock then sets CL3 DISCHARGE and clock generator CL2 DISCHARGE T00–T07 is started with T00 going true. DISCHARGE also drives four bus reset circuits (DLH prints), which rapidly return asserted data lines to the clamp voltage of -3 V. The DISCHARGE clock generator continues to shift and the data input is removed from the DISCHARGE flip-flop by DISCHARGE T03. The next IBC clocks then clock the flip-flop off, causing a bus discharge interval of four clock periods. This ends IBus operation. Following the discharge interval, DISCHARGE T06 enables the DONE flip-flop, causing it to set three clock periods later. DONE then asserts XFER on the EBus, which causes the EBox to drop DEMAND. The trailing edge of DEMAND asserts CL1 CLEAR to end DIA20 operation.

3.4.3 CONI/DATAI Operation

Timing for the DATA IN operation is shown on Figure 3-3. The EBox asserts the select lines and function lines and then raises DEMAND, which starts the clock generator as described in Paragraph 3.4.1. At time T02, either the CL2 CONI or CL2 DATAI flip-flop (depending on function code) is clocked on, asserting the CONI or DATAI control line on the IBus. The selected KA10/KI10 I/O device uses this signal to gate the incoming data on the data lines. At time T11 (fast mode) or T23 (slow mode), the CONI or DATAI flip-flop is direct cleared giving a CONI/DATAI signal duration of either 7 or 17 clock periods, depending on the mode of operation. The trailing edge of this signal defines the end of the IBus command and, after a cable delay, the I/O device negates the IBus data lines.

When the CONI or DATAI flip-flop is cleared, before the data lines go false, both CL3 EBUS STROBE and CL3 DISCHARGE ENB are edge-triggered to the ONE state. EBUS STROBE clocks the incoming data into the 36-bit data buffer where it is immediately transmitted on the EBus. DISCHARGE ENB enables the DISCHARGE clock generator. The DIA20 then discharges the IBus data lines, sets DONE to generate XFER, and shuts down as previously described for the DATA OUT operation (Paragraph 3.4.2).

3.4.4 PI SERVED Operation

Figure 3-4 shows the timing for the PI SERVED operation. The EBox asserts the function lines, places the PI channel number on select lines CS 04–06, and asserts DEMAND. The clock generator goes active with DEMAND (Paragraph 3.4.1) and at time T00, CL3 PI SERVED ACKN will be clocked on if CL1 INTR COMPARE is true (Paragraph 3.4.2). PI SERVED ACKN immediately asserts EBus data line D15, signaling the EBox that the IBC (physical address = 15) has responded to the PI SERVED command. The EBox holds DEMAND true for at least 400 ns, allowing time for all responding KL10 controllers to assert the appropriate data line. At the trailing edge of DEMAND, CL1 CLEAR is generated and the DIA20 shuts down ending the operation.

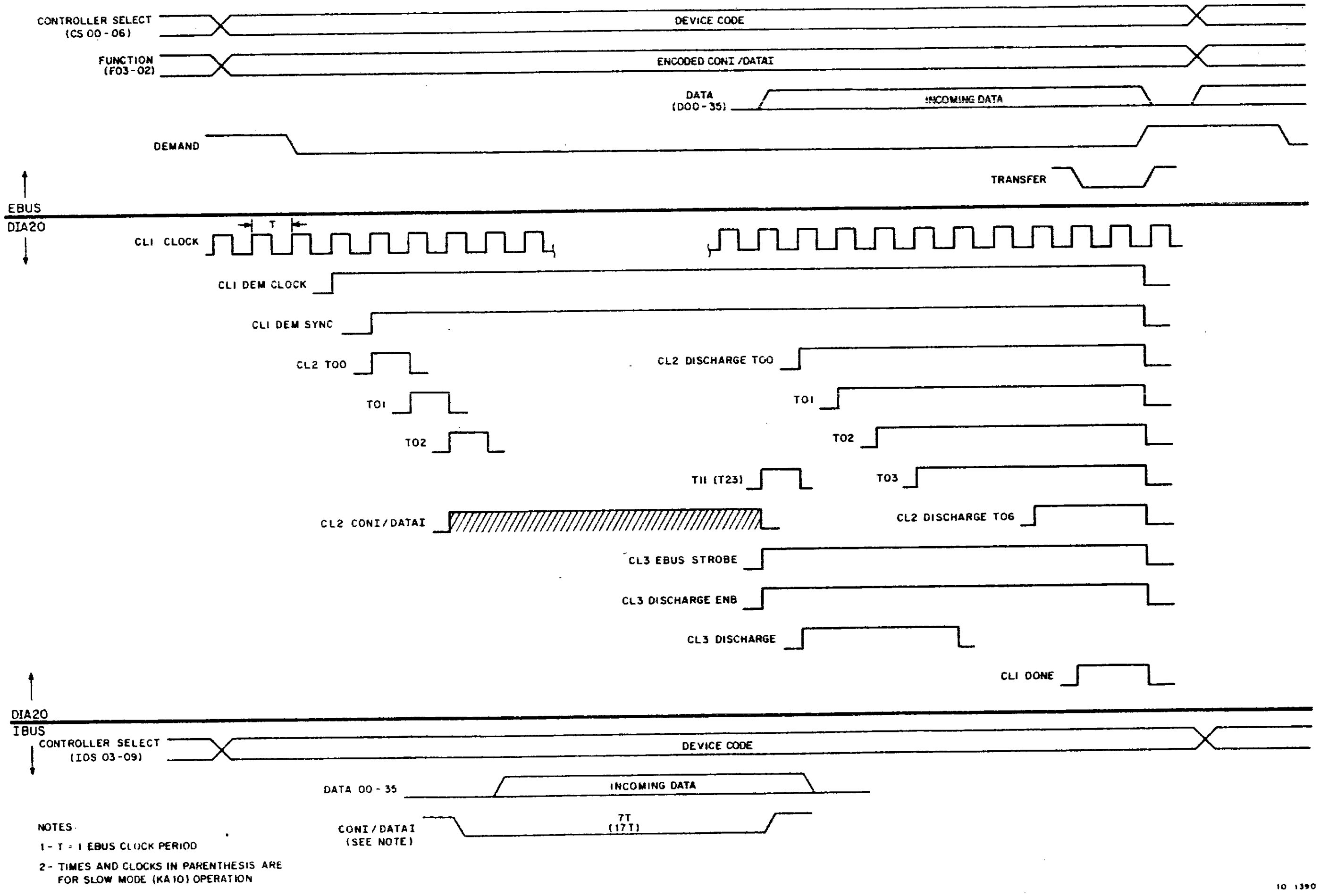


Figure 3-3 CONI/DATAI Timing Diagram

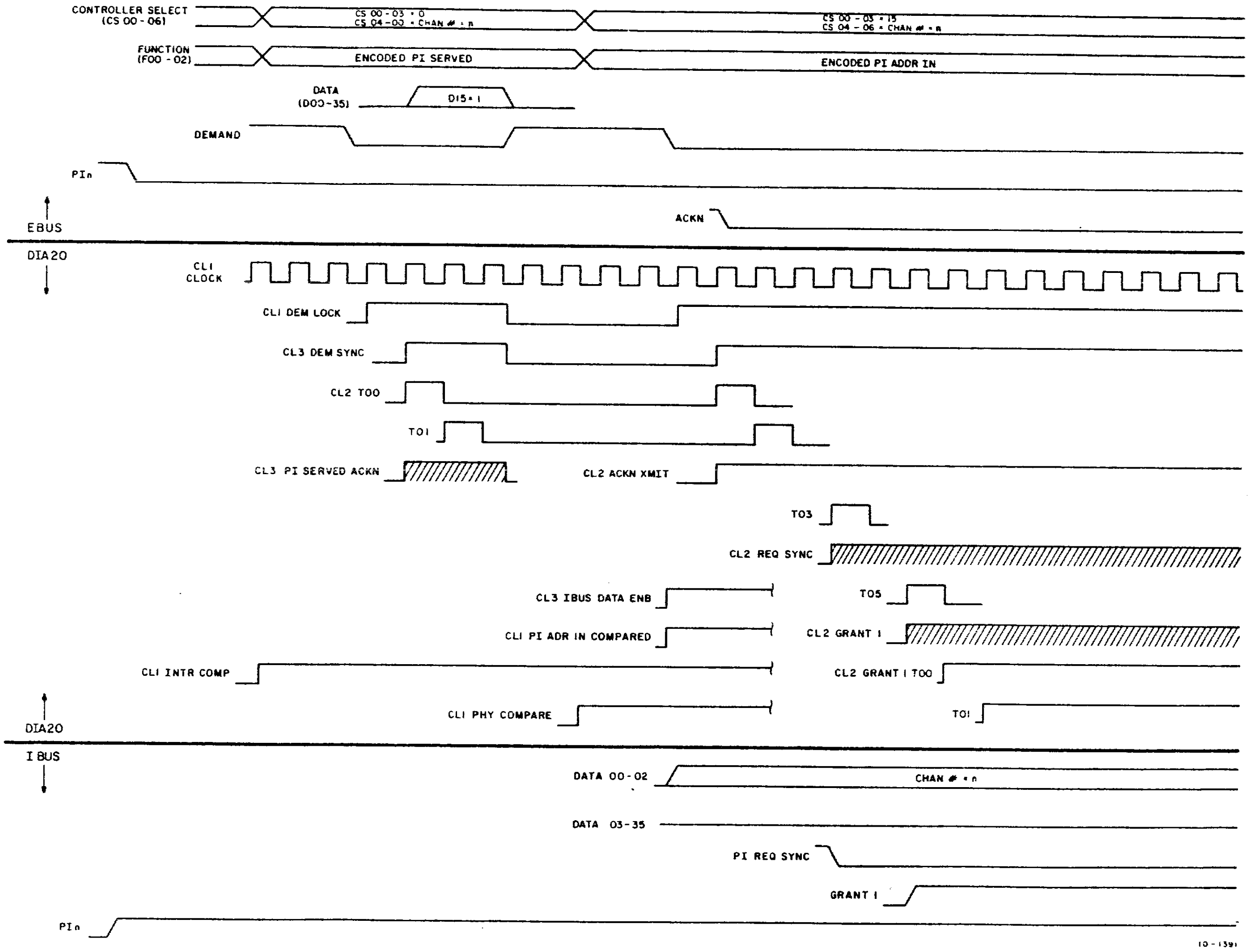


Figure 3-4 PI SERVED and PI ADR IN (Start-up) Timing Diagram

3.4.5 PI ADDRESS IN Operation

After the PI SERVED command and after resolving physical number priority, the EBox issues the PI ADR IN command to collect an interrupt function code and address (if any) from the interrupting controller. With the function and select lines asserted, the EBox raises DEMAND starting the clock generator as described in Paragraph 3.4.1. If a KA10/KI10 I/O device is interrupting on the channel number being serviced and if the IBC is being addressed (physical number = 15), flip-flop CL1 PI ADR IN COMPARED (Paragraph 3.4.2) sets CL3 IBUS DATA ENB to enable the IBus transmitters and cause the channel number on the low-order select lines to be gated out to the IBus on data lines 00–02 via the mixer outputs DLH1 EBUS BROADCAST D 00–02. PI ADR IN COMPARED also causes the IBC to respond to the command by setting CL2 ACKN XMIT at time T00. Start-up timing is shown in Figure 3-4.

ACKN XMIT asserts ACKN on the EBus. It also blocks CL1 ACKN REC at the input to CL1 CLEAR. This gating is required because ACKN REC, a bus receiver output, goes true when ACKN is transmitted. It would otherwise shut down the IBC as it normally does when another controller responds on the bus. At time T03, CL2 REQ SYNC sets and asserts PI REQ SYNC on the IBus. This is followed by the GRANT 1 control signal when CL2 GRANT 1 sets at time T05. PI REQ SYNC and GRANT are used by KI10 devices to synchronize and generate an API response as discussed in Paragraph 2.2.3.

Figure 3-5 illustrates IBC operation after the assertion of the GRANT line for bus 1. The generation of a standard interrupt by a KA10 I/O device is indicated. Also shown is IBC operation after an API response from a KI10 I/O device. The API response is shown as occurring on bus 2 (preceded by a GRANT RETURN on bus 1). A response can occur on either bus, however.

After receiving a GRANT level, a KI10 device on either bus generates an API response by asserting an interrupt function code on data lines 03–05. This asserts DLH1 INTR FUNCTION, the OR of the three lines, which causes the next IBC clock to set CL1 INTR FUNC and thus enable the clock generator CL2 T10–T27. DLH1 EXEC VIRTUAL ENB then goes to ONE at time T12 and asserts EBus data line D02 to specify an executive virtual address space for the forthcoming API information (Figure 2-1). At time T13, CL3 EBUS STROBE is direct set and the API response information on the IBus data lines is clocked into the data buffer and transmitted on the EBus. The GRANT flip-flop is also cleared at this time. At time T23, CL3 PI ADR IN DONE is direct set and the REQ SYNC flip-flop is direct cleared. PI REQ SYNC then goes false on the bus and the responding controller removes the API information from the data lines ending the IBus operation. PI ADR IN DONE sets CL3 DISCHARGE ENB enabling the DISCHARGE clock generator. The DIA20 then discharges the data lines, sets DONE generating XFER, and shuts down as previously described for DATA OUT and DATA IN operations.

DIA20 operation following a GRANT RETURN differs depending on the IBus port. When the GRANT signal is returned from the end of bus 1, it indicates that the interrupting device is not on bus 1 or that it is on bus 1 but it is a KA10 device. (GRANT is bused through KA10 devices.) When received, the GRANT 1 RETURN signal sets flip-flop CL3 GRANT 1 RETURN to enable clock generator CL3 GRANT RETURN T000–T375. At time T375, CL3 GRANT 1 RETURN DLY is asserted to clear CL2 GRANT 1 and negate the GRANT level on bus 1. The clearing of CL2 GRANT 1 also sets CL2 GRANT 2 to assert the GRANT line on bus 2. The clock generator output is used to clear the GRANT flip-flop so that a minimum duration (three clock periods) of the GRANT signal is ensured for short bus lengths.

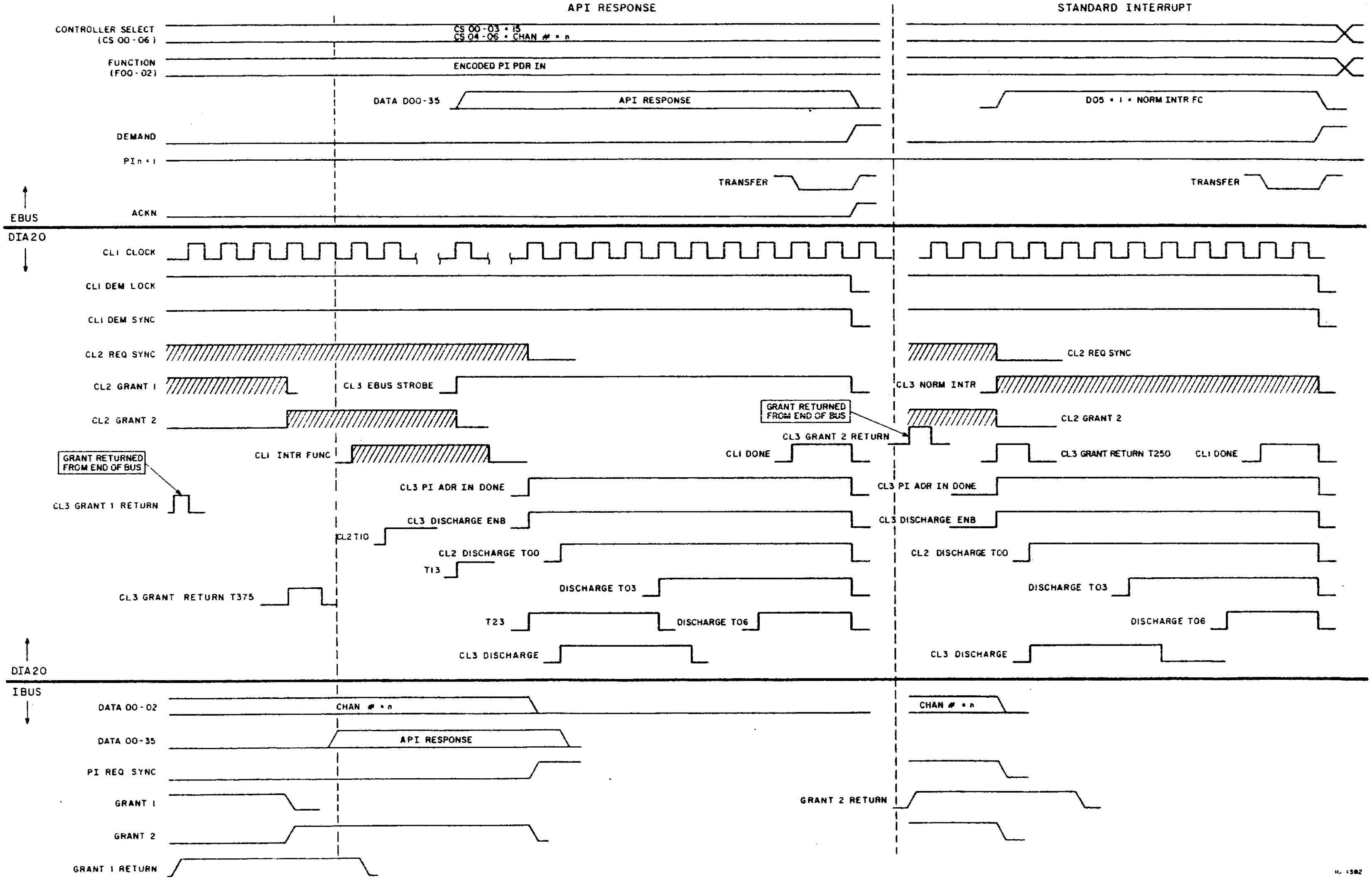


Figure 3-5 PI ADR IN (API RESPONSE and STANDARD INTERRUPT) Timing Diagram

If the GRANT level is returned from bus 2, it indicates that the interrupting device is a KA10 device (no API response from either bus). Thus, the DIA20 must send a standard interrupt function code to the EBox. As for bus 1, the GRANT RETURN level for bus 2 sets a flip-flop (CL3 GRANT 2 RETURN) that enables clock generator CL3 GRANT RETURN T000–T375. Bus 2 operation differs in that output T250 from the clock generator, which asserts CL3 GRANT 2 RETURN DLY, is used to clear the GRANT flip-flop (CL2 GRANT 2) instead of output T375. This is because T375 could still be true if there is a GRANT RETURN on both buses and bus 2 has a short cable length. GRANT 2 would then be cleared too soon. As stated previously, the purpose of the GRANT RETURN delay is to ensure a minimum duration for the GRANT level.

To generate the standard interrupt, GRANT 2 RETURN DLY sets CL3 NORM INTR. This flip-flop asserts EBus data line 05 to send the appropriate interrupt function code (FC = 1) to the EBox. GRANT 2 RETURN DLY also sets PI ADR IN DONE. As occurs for an API response, PI ADR IN DONE sets CL3 DISCHARGE ENB to enable the DISCHARGE clock generator. The DIA20 then discharges the data lines, sets DONE generating XFER, and shuts down as previously described for DATA OUT and DATA IN operations.

APPENDIX A
ABBREVIATIONS AND MNEMONICS

ACKN	Acknowledge
ADR	Address
CL	Control
CLK	Clock
CLR	Clear
CONI	Conditions In
CONO	Conditions Out
CS	Controller Select
D	Data
DATAI	Data In
DATAO	Data Out
DEM	Demand
DLH	Data
DLY	Delay
EBUS	Execution Bus
ENB	Enable
EPT	Executive Process Table
EXEC	Executive
F	Function
FUNC	Function
GND	Ground
IBC	IBus Controller/Adapter (DIA20)
IBUS	KA10/KI10 I/O Bus
INTR	Interrupt
I/O	Input/Output
IOS	I/O Select
KA	KA10
KI	KI10
PI	Priority Interrupt/Priority Interrupt Request
PHY	Physical (address)
RDI	Read-In
REC	Received
REQ	Request
SYNC	Synchronize/Synchronization/Synchronizer
T	Time/Clock Period (EBus)
XFER	Transfer
XMIT	Transmit

INDEX
DIA20

DIA20 IBUS ADAPTER
UNIT DESCRIPTION
EK-DIA20-TM-002

Reader's Comments

CUT OUT ON DOTTED LINE

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? ____________________

____

____

____

What features are most useful? ____________________

____

____

____

What faults do you find with the manual? ____________________

____

____

____

Does this manual satisfy the need you think it was intended to satisfy? ____________________

Does it satisfy *your* needs? ____________________ Why? ____________________

____

____

____

Would you please indicate any factual errors you have found. ____________________

____

____

____

Please describe your position. ____________________

Name ____________________ Organization ____________________

Street ____________________ Department ____________________

City ____________________ State ____________________ Zip or Country ____________________

— — — — — — — — — — — Fold Here — — — — — — — — — — —

— — — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
146 Main Street
Maynard, Massachusetts 01754