D. Nagle

PDP-X Technical Memorandum # 26

Title: IO Bus Sequences

Author(s): H. Burkhardt

Index Keys: Bus
IO
Sequences

Distribution Keys: A

Obsolete: None

Revision: None

Date: October 5, 1967

Contents

1. Introduction

The PDP-X IO Bus provides many facilities that are available to both standard and customer-designed peripheral devices. Some of these facilities are:

1. Multiplexor Channel input and output
2. Data Packing
3. Priority Interrupt synchronization
4. ability to specify arbitrary memory locations for data transfer operations
5. addition of external data to the contents of main memory

The data input/output and packing operations are described in PDP-X Technical Memorandum #29 (sections 3.1, 3.2, 3.3). This document describes additional features of the IO Bus and provides detailed information on the sequences of events on the fully interlocked bus. Details concerning the electrical and mechanical characteristics of the IO Bus are beyond the scope of this document. It is sufficient to say, however, that the IO Bus uses a twisted pair, push/pull current system. The reader is first referred to Technical Memorandum #29 in the sections mentioned above, as well as to section 3.7.

2. The IO Bus

The IO Bus is the connection between IO device control units and the central processor. All communication lines to and from the processor are common to all control units except for the priority request and grant lines which are common only to devices on a given priority level. At any one instant, however, only one control unit is logically connected to the processor (selected). This connection is maintained until the processor re-selects a different device. Since all standard devices can operate with a comparitively small amount of processor intervention, many devices may be operational simultaneously.

The IO Bus consists of 8 bi-directional data lines and 32 control lines. The control lines may be broken into four groups:

a. 8 outbound control lines (Command Out) (from the processor to the device)

b. 8 inbound control lines (Response In) (from the device to the processor)

c. 8 inbound priority request lines (from the devices to the processor, 1 line per each priority level)

d. 8 outbound priority grant lines (from the processor to the devices, 1 line per priority level)

In addition, power and ground return are available on the bus.

The priority request and grant lines are used to synchronize device service requests with processor activity. The bi-directional data lines and the outbound and inbound control lines are used to control the actual data transfers and to indicate special functions or exceptional conditions.

For purposes of further discussion, it is meaningful to distinguish between two types of operation on the bus:

1. Processor initiated activity

2. Device initiated activity

(It must be remembered that both types of activity are directly or indirectly initiated by the processor. The distinction is made here between specific tasks performed by the program controlling the processor such as reading data or changing device status and those processor functions performed without the intervention of the program because of requests generated by a device. Of course, the program had to initialize the processor and the device to generate and respond to these requests. It might further be noted that the IO Bus is, quite literally, merely a collection of wires. The functional properties ascribed to these wires arise only because there is hardware in the processor controlling the signals transmitted by these wires.)

3. Processor Initiated Activity

The processor (or to more exact, the program controlling the processor) may initiate activity on the IO Bus by the execution of an IO class instruction. These instructions are:

IOR/IOW/IORC/IOWC	Read or write data
IOS/IOC	Read or write status
RIO	Reset IO system

One additional instruction, IOT, is provided for the convenience of the programmer. IOT appears to the IO Bus (and hence the selected device) as an IOS (read status). All of these instructions (with the exception of RIO) contain a Device Address (device selection code, device number, etc.) .

When an IO class instruction is executed, this Device Address is transmitted on the Data Lines. The outbound control lines are set to indicate that an Address is on Data Lines and a SYNC level is transmitted (one of the outbound control lines). Every device on the IO Bus decodes the command lines and the Data Lines, but only one device recognizes the Device Address code. This device becomes selected (sets a Flip-Flop in itself) and signals the processor that it has received and decoded the address by setting the inbound control lines and raising RTN (one of the inbound control lines). The processor may now disconnect the Device Address from the Data Lines and drop SYNC. The selected device then drops RTN and the other inbound control lines and the bus becomes free. (Automatic time-out circuitry in the processor will disconnect the Device Address from the bus, etc. and allow the processor program to continue if no device responds.)

The processor next initiates a DATI END sequence (IOR, IORC), DATO sequence (IOW, IOWC), CONI sequence (IOS) or CONO sequence (IOC). Figure 1 depicts a one-byte data input sequence (IOR, IORC). The status input sequence (IOS, IOT) would be identical except that the processor would set the command lines to CONI instead of DATI END. Figure 2 depicts a one-byte data output sequence (IOW, IOWC). The status output sequence (IOC) would be identical except that the processor would set the command lines to CONO instead of DATO.

Figures 3 and 4 depict two-byte input and output respectively. When the first data transfer between processor and device is accomplished, the device responds and indicates that it requires (or has) an additional byte of data (or status).

When an RIO instruction is executed (or when the corresponding console switch is closed), the processor sets the outbound control lines to indicate a Reset operation and raises SYNC (one of the outbound control lines). It then waits for all devices to receive this signal (the same time-out that occurs during the address-selection sequence) and then proceeds.

4. Device-Initiated Activity

When a device finishes a task previously requested by the processor, it will set the REQ bit in its status register. The processor may test this bit (with an IOS or IOT instruction) to determine if the device has finished its task. Since the processor would have no way of knowing when the device finished unless it continually tested this bit (precluding any other computational activity), an interrupt system is made available to allow the device to signal its service request to the processor. Every device has an ENABLE bit in its status register which allows it to be connected or disconnected from this interrupt system. If this ENABLE bit is set and the device has either finished its task or has detected an unusual condition (REQ or UNUSUAL), the device will signal the central processor that it requires service by raising the REQ_i line corresponding to the priority level to which it is assigned (i). (Note that this does not affect any operations on the data, outbound control and inbound control lines that may be occurring.) When the processor is able to honor this service request, it raises the corresponding Priority $(grant)_i$ line. This line passes through all devices whose priority is i. The first such device that is also requesting (has REQ_i up), does not re-transmit this signal to the next device, but instead, connects its Device Address code and the HIGH bit from its status register onto the Data Lines and raises Address In (one of the inbound control lines). When the processor detects the rise of Address In, it reads in the Device Address and HIGH from the Data Lines. The processor then determines if a channel operation is indicated or not by examining the contents of the main memory location indicated by the Device Address and HIGH.

Figure 5 depicts the sequence that occurs when a channel operation is not indicated. Figure 6 depicts the sequence that occurs when a channel operation is indicated. In a channel operation, the device is selected. The processor does so by setting the outbound control lines to indicate an address-selection sequence and raises SYNC. Since the Device Address is already on the Data Lines, the device becomes selected and sets the inbound control lines including RTN. The proces-

sor then examines the inbound control lines and drops Priority, and SYNC. The inbound control lines 4, 5 indicate one of four initial types of operations:

0 Normal, the Byte Counter and Byte Address are updated. Response In 6-7 are then examined.

1 Inhibit BA Count. Only the Byte Counter is updated. The Byte Address is left undisturbed. Response In 6-7 are then examined.

2 Force Last Cycle After Reading in 16-bit Byte Address. The IO Bus performs 2 DATI cycles to read in a 16-bit Byte Address. The Byte Counter and Byte Address are undisturbed. Response In 6-7 are then examined.

3 Permit only BC Count. The Byte Counter is incremented and the result is re-transmitted to the device. The interrupt is then dismissed by the processor.

Modes 2 and 3 are depicted in Figures 7 and 8 respectively. Figure 8a depicts Mode 3 for a 2-byte device. After Modes 0, 1, 2, Response In 6-7 are examined to determine the type of data transfer as follows:

0 OUT
1 ILLEGAL
2 IN
3 ADD-IN

These types of transfer are depicted for a single-byte device in Figures 9, 10, 11. If the device is a multiple-byte device, channel overflow is indicated while transferring the byte in which the overflow occurred. In the case of ADD-IN, which reads in data bytes, does the addition, stores the sum in memory and transmits the sum; two types of overflow may occur:

a. channel overflow

b. add overflow

Channel overflow is indicated while transforming the byte in which the overflow occurred (DATI LAST) and add overflow is indicated during each

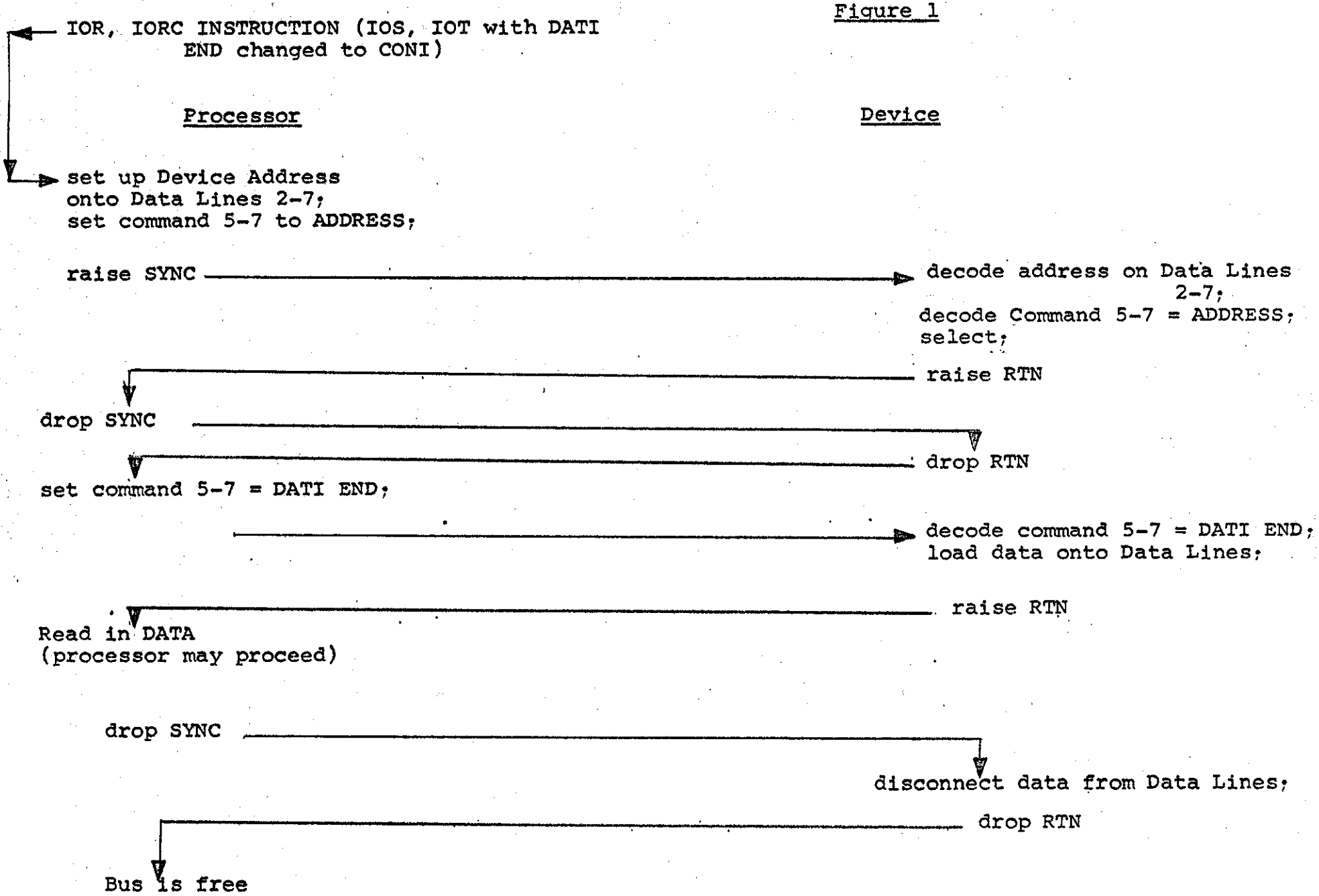
each output transfer. Only one or two byte transfers will be made during an ADD-IN operation. Additional bytes will not be received. Double-byte additions across word boundaries may produce unpredictable sum and overflow results.

5. Appendix

Bus Sequences

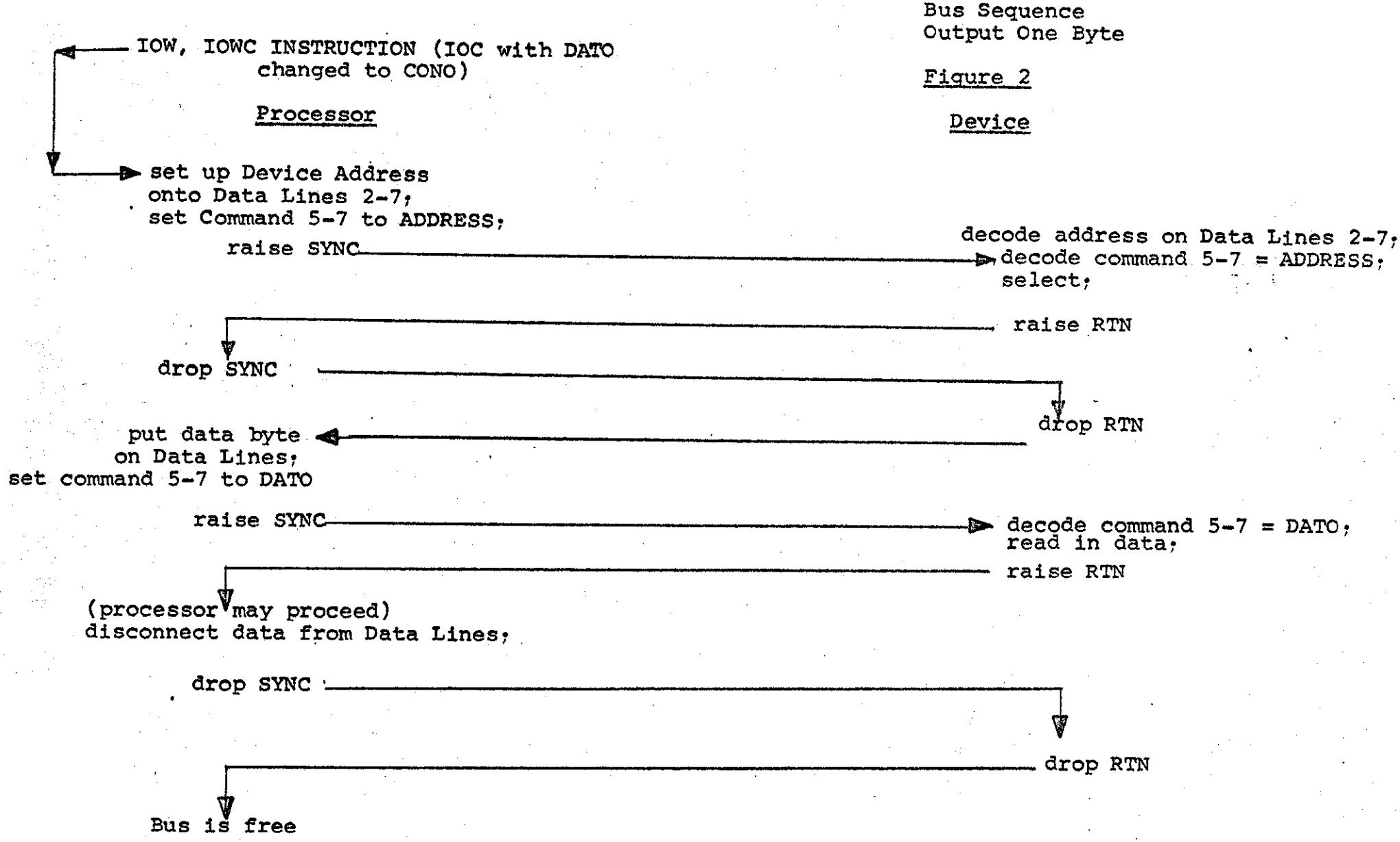
Bus Sequence
Input One Byte

Figure 1



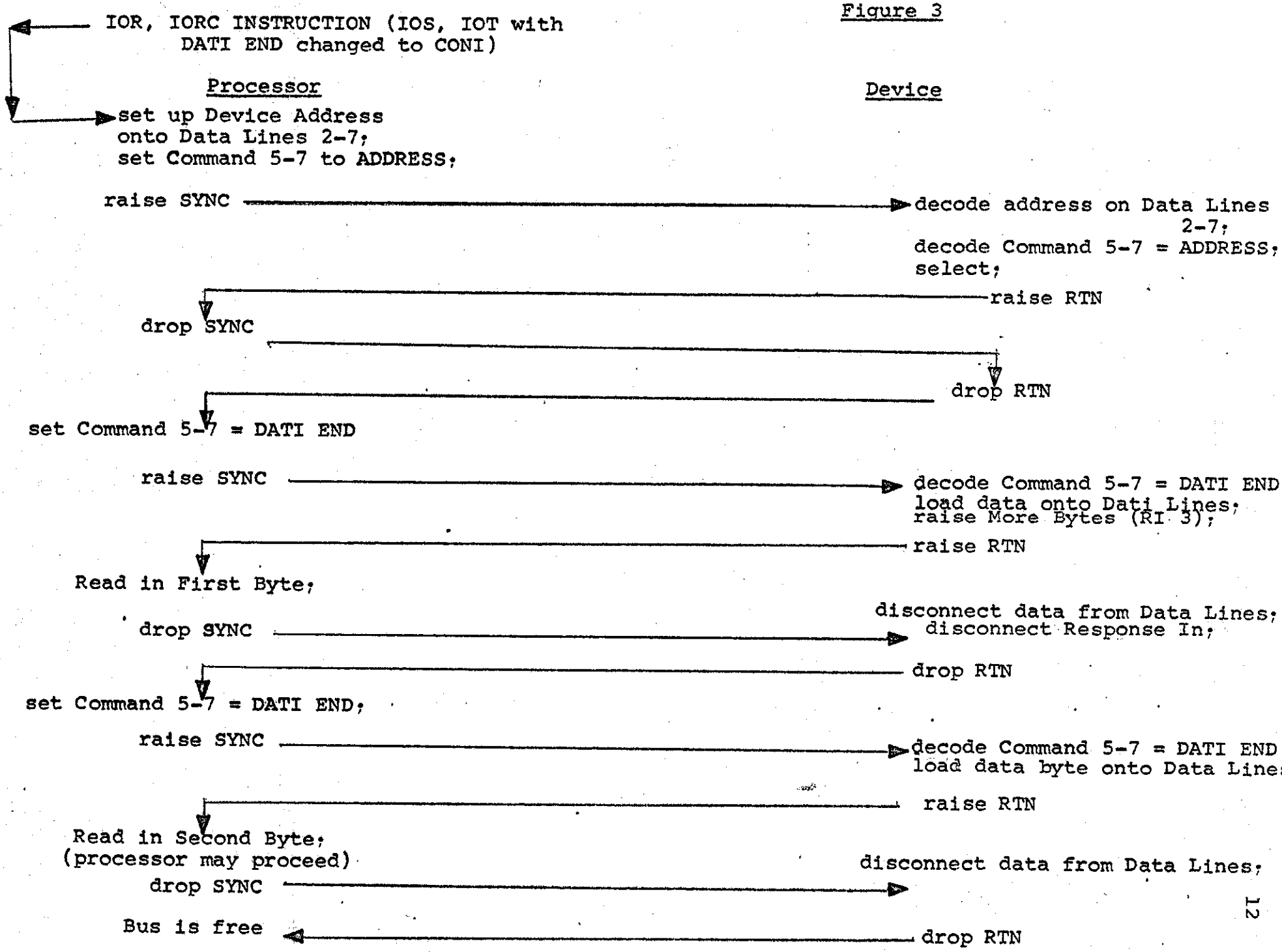
Bus Sequence
Output One Byte

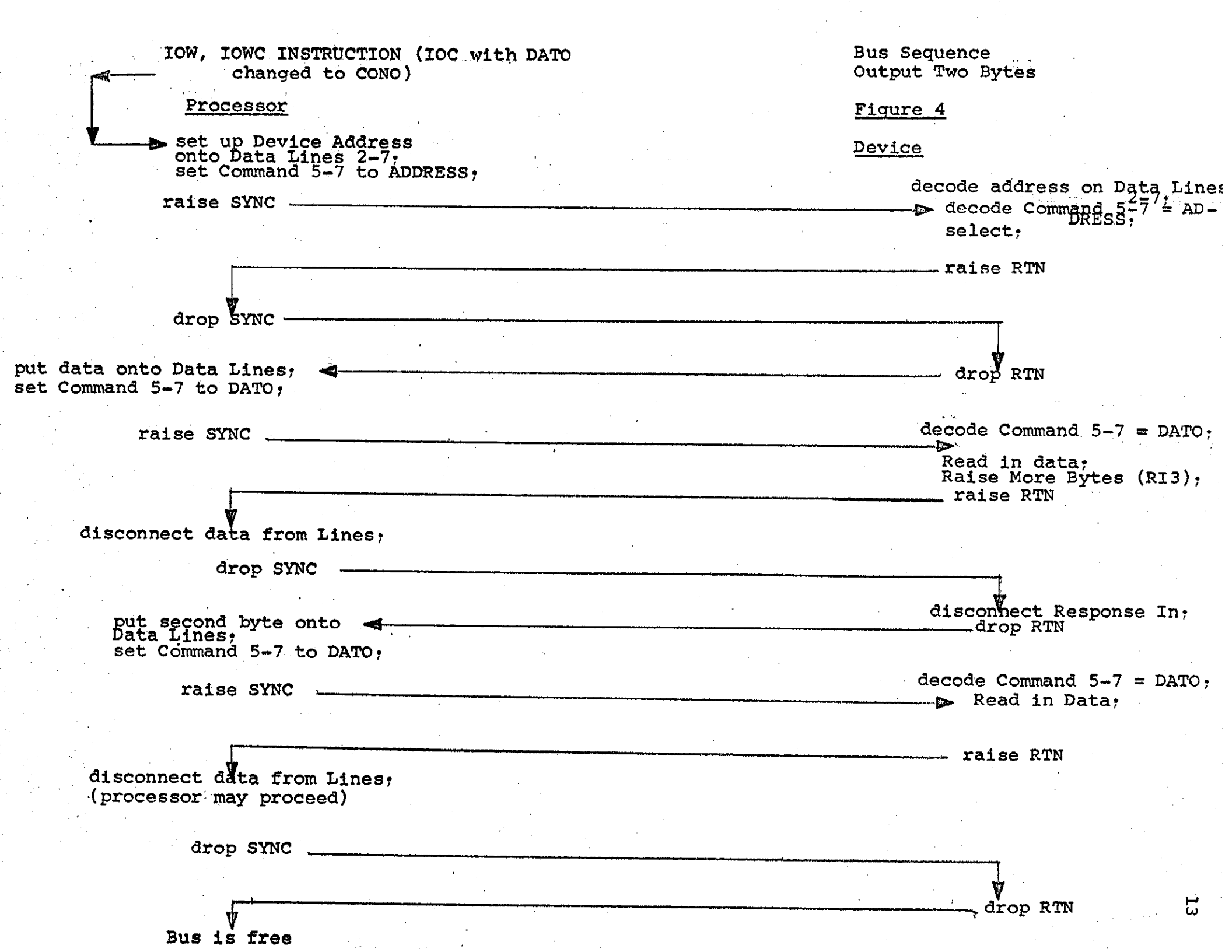
Figure 2



Bus Sequence
Input Two Bytes

Figure 3



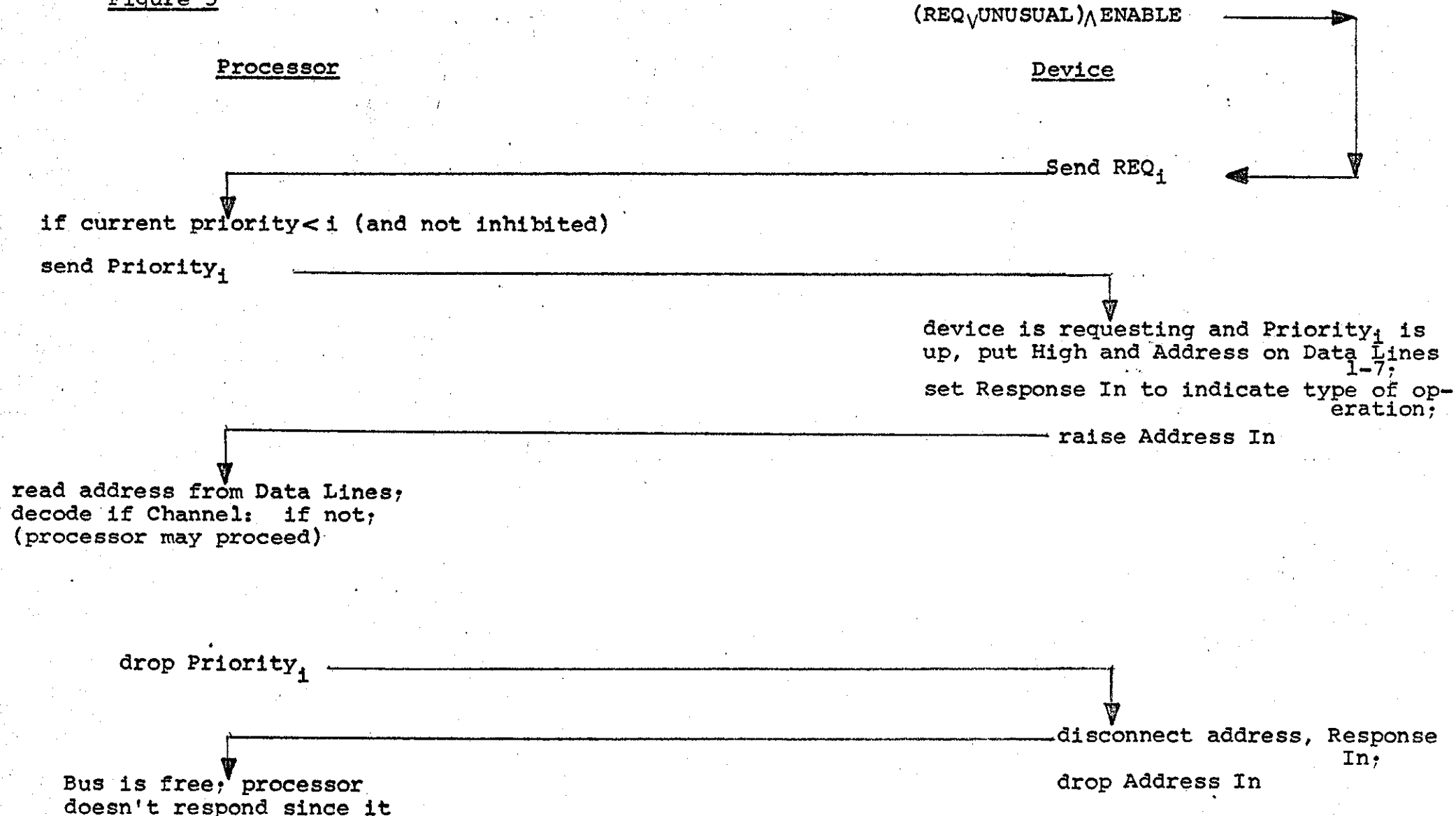


Bus Sequence
Output Two Bytes

Figure 4

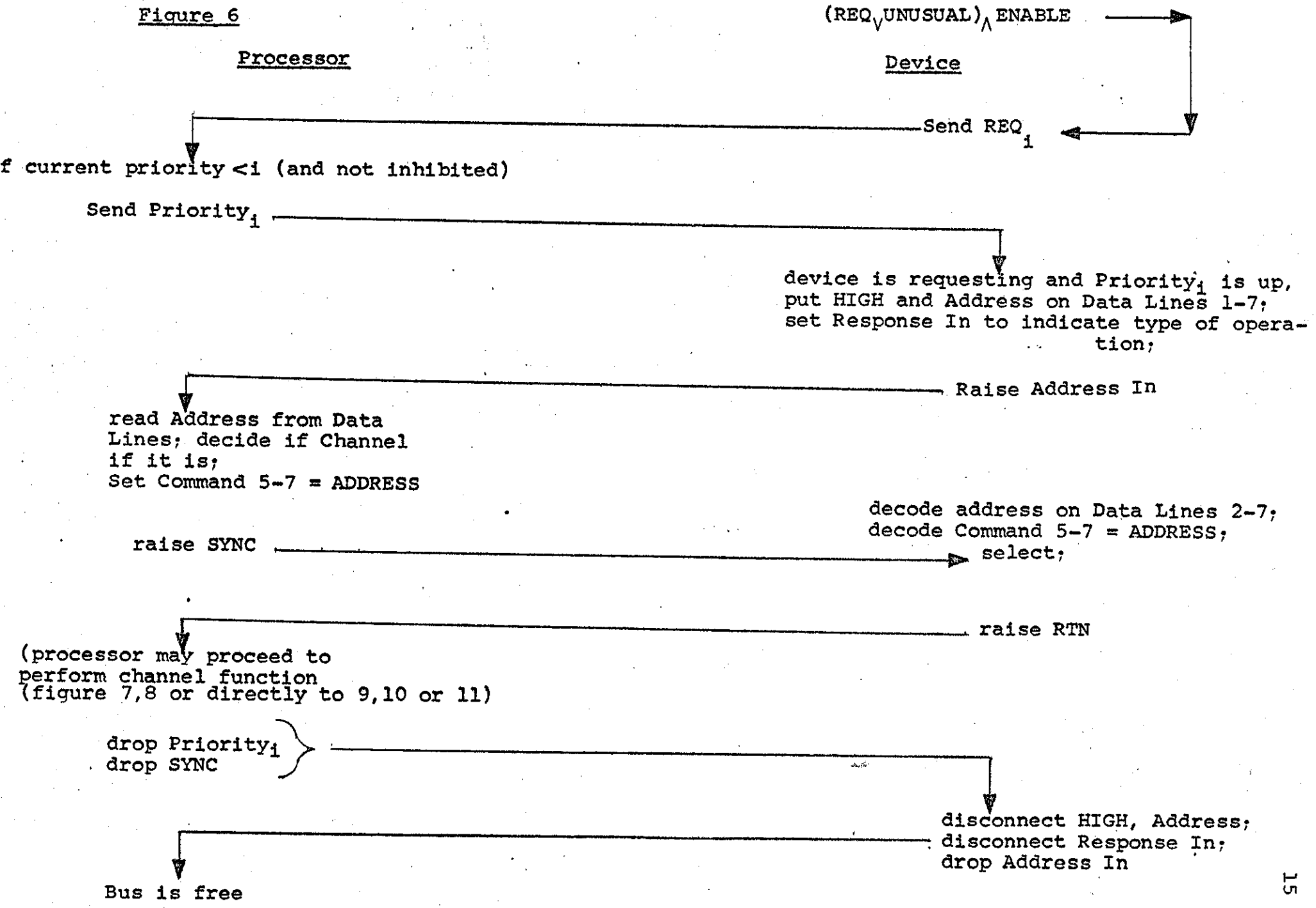
Bus Sequence
Interrupt - No Channel

Figure 5



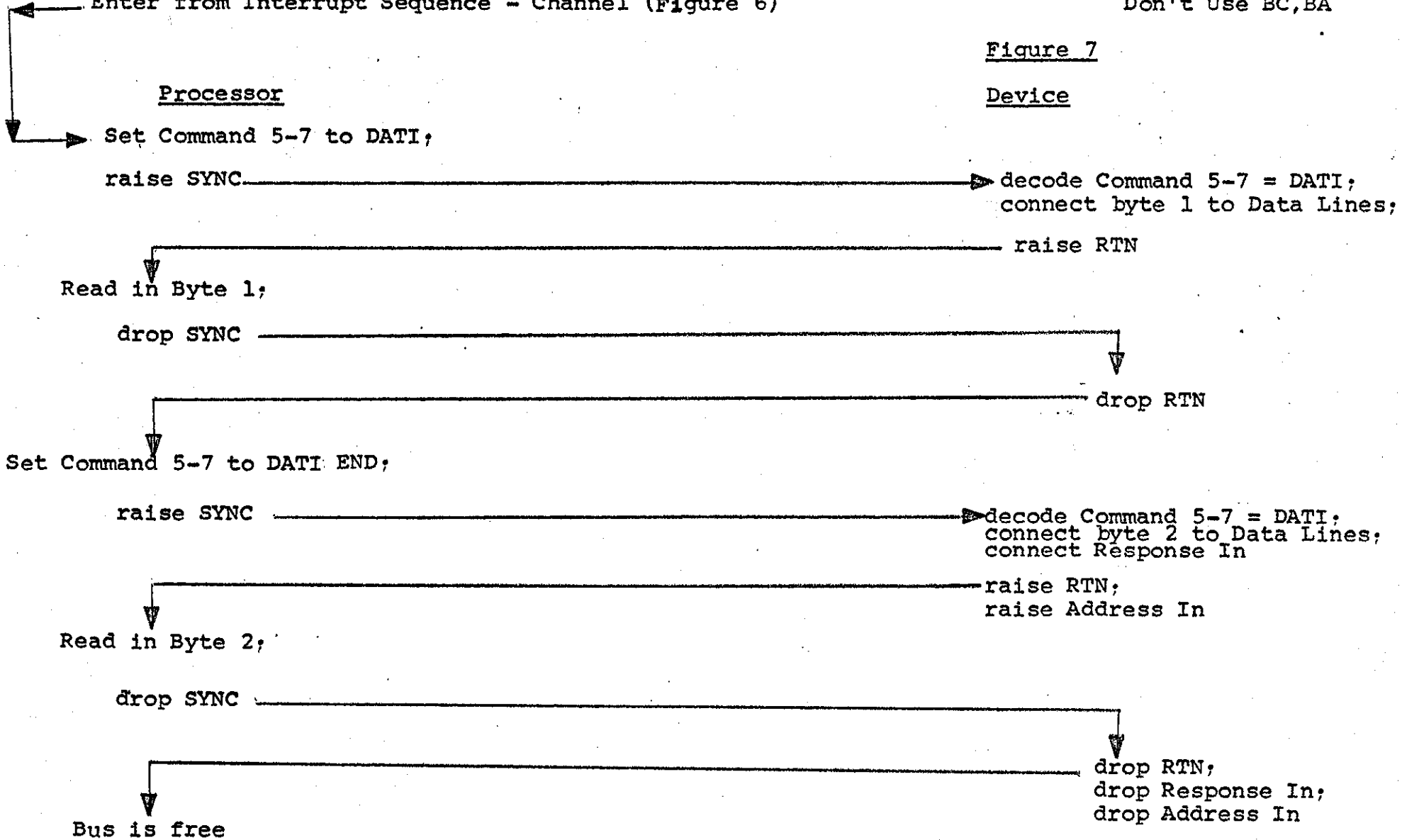
Bus Sequence
Interrupt - Channel

Figure 6

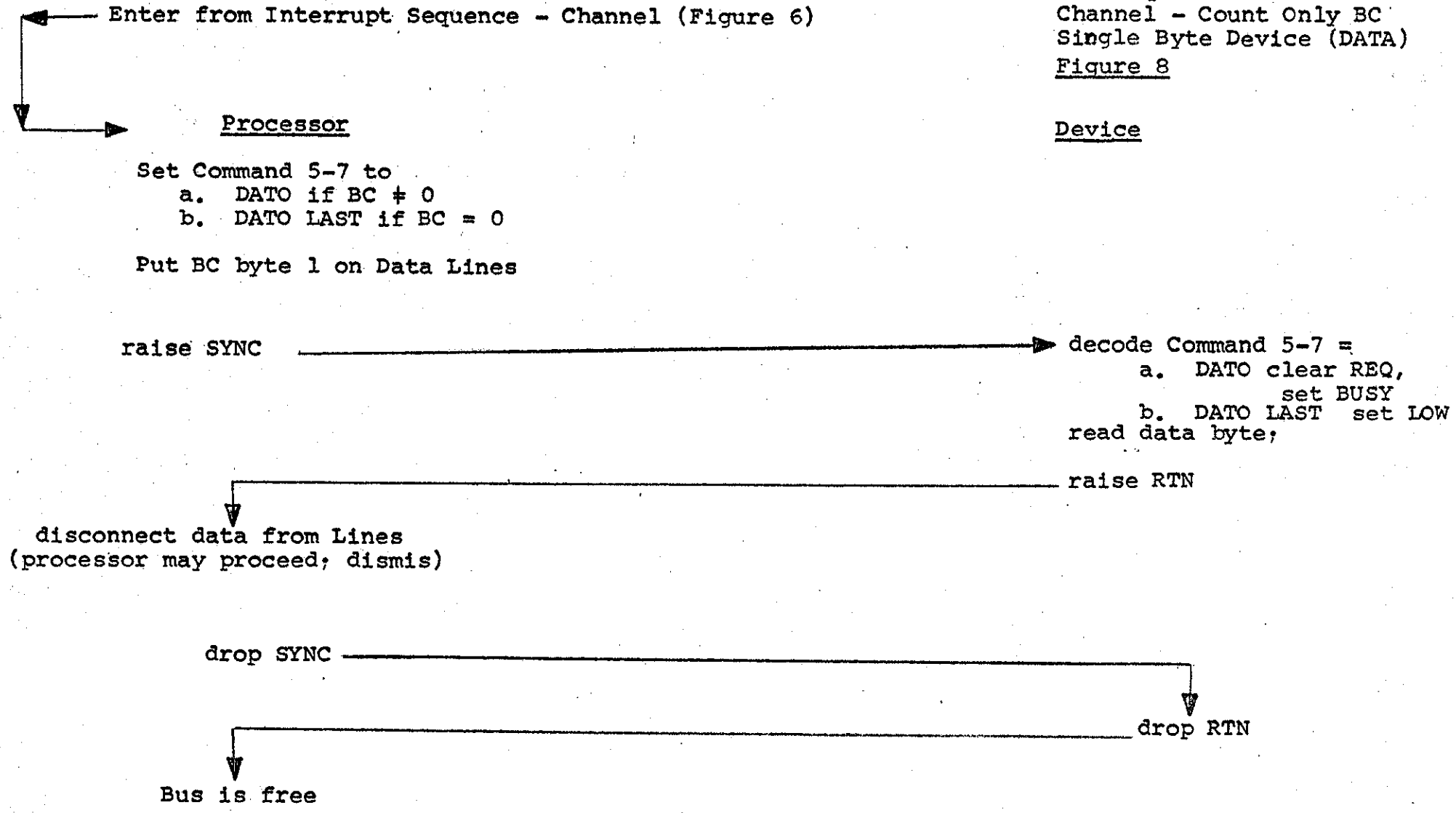


Bus Sequence
Channel - Read 16-bit Address
Don't Use BC,BA

Figure 7

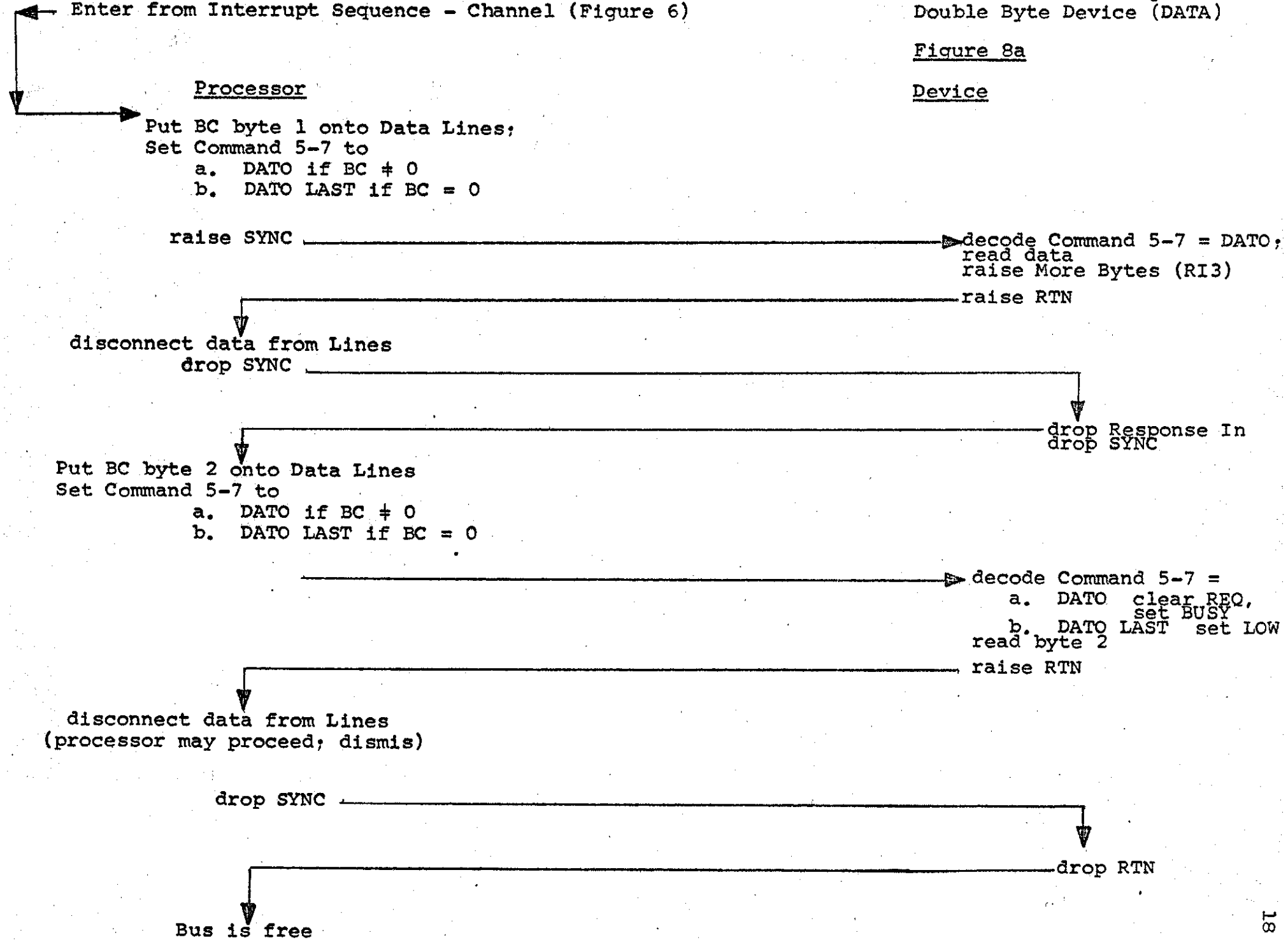


Bus Sequence
Channel - Count Only BC
Single Byte Device (DATA)
Figure 8



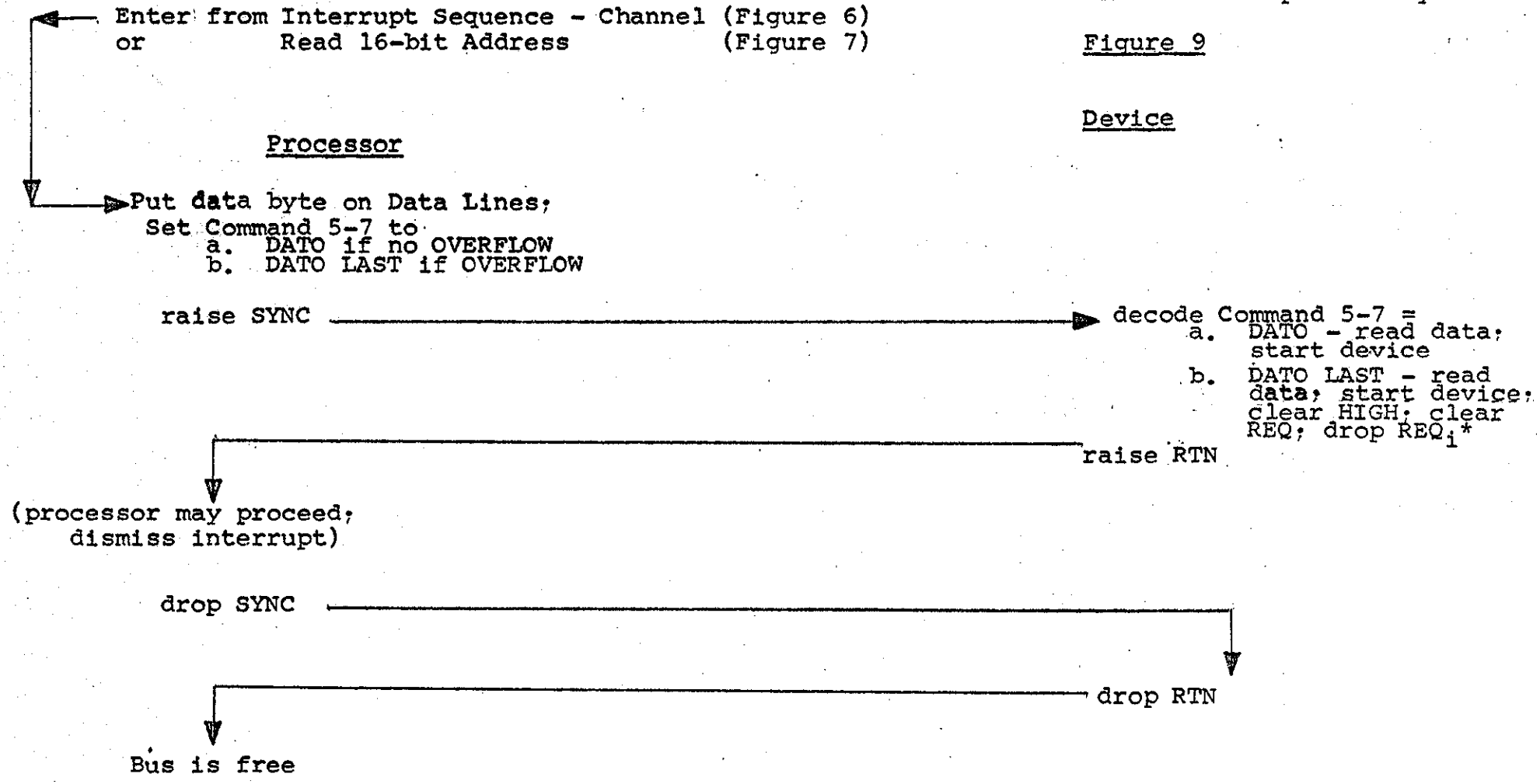
Bus Sequence
Channel - Count Only BC
Double Byte Device (DATA)

Figure 8a



Bus Sequence
Channel - Output One Byte

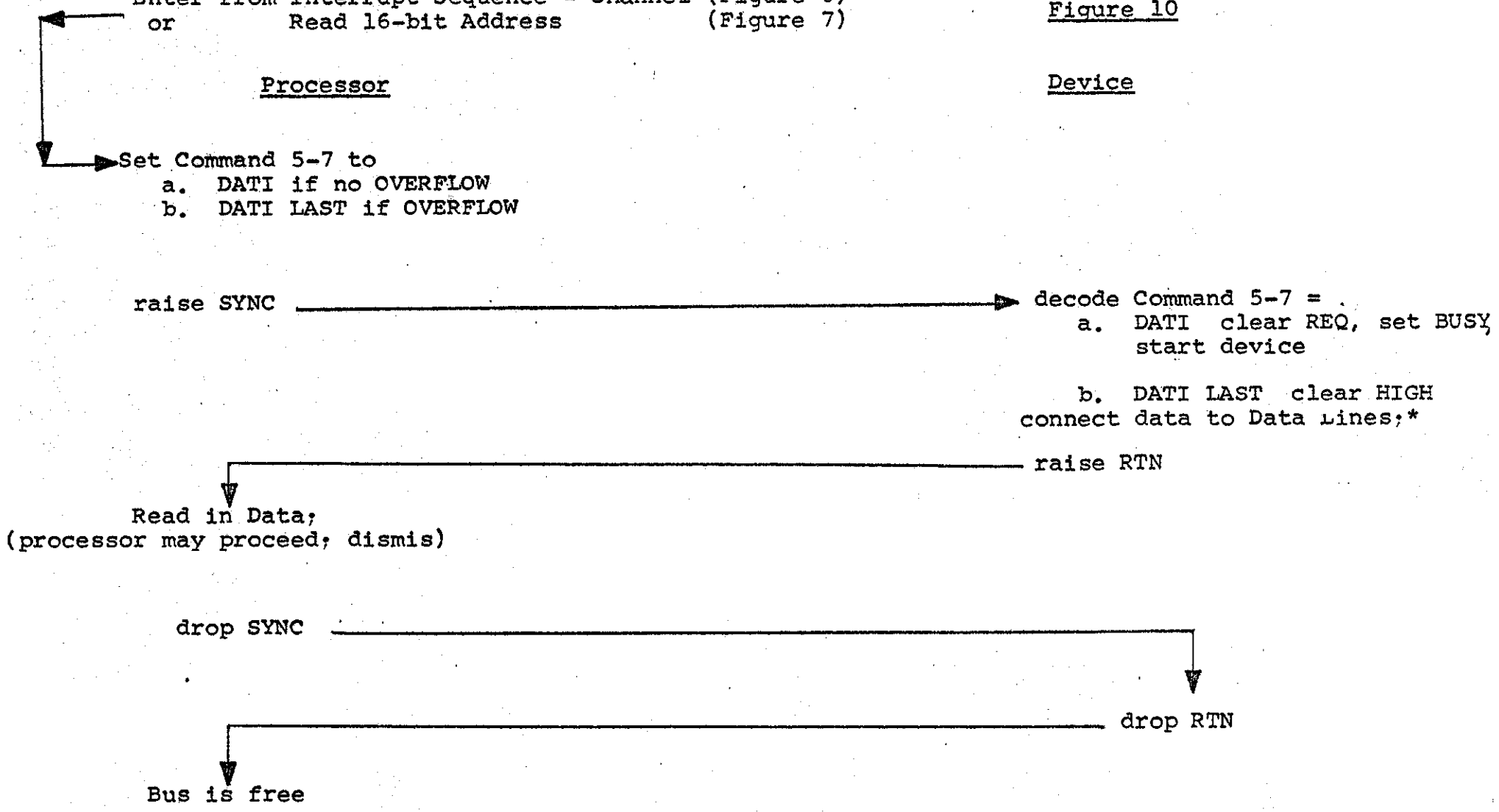
Figure 9



*Device can signal more bytes by raising More Bytes (Response In 3). This sequence is the same as the second sequence in Figure 2 (i.e., after the device Selection Sequence).

Bus Sequence
Channel - Input One Byte

Figure 10



*Device can signal more bytes by raising More Bytes (Response In 3). This sequence is the same as the second sequence in Figure 7 (i.e., after the device selection sequence).

Bus Sequence
Channel - Add In One Byte

Figure 11