4 DCE-BUS SPECIFICATIONS

4.1 DCE-BUS CONCEPT

The DCE-BUS provides a common transfer medium for the exchange of data and control information between DCE-BUS compatible modules and one or more processor modules. It allows data devices or processor modules to be directly plugged into the bus in any physical position. It carries two interrupt request signals, a reset signal, D. C. power lines and control signals for device selection, addressing and data transfer.

The DCE-BUS may be controlled via the GIC on the master processor either in a simple software controlled mode, or in a fast-bus mode with the Large System Adapter (DCE-LSA) in DCE-X based systems The DCE-BUS can also be driven via the DCE-LSA, to provide a very fast memory-mapped I/O mode of operation.

4.1.1 General Description

Figure 4. 1 illustrates the implementation of the DCE-BUS via the GIC on the master DCE processor module. GIC Port 0 is used as a bi-directional data path for data transfer between the master DCE processor and other modules. GIC Port 1 is used to specify the address of the connecting module, and GIC Port 2 issues the control signals to complete the transfer logic.

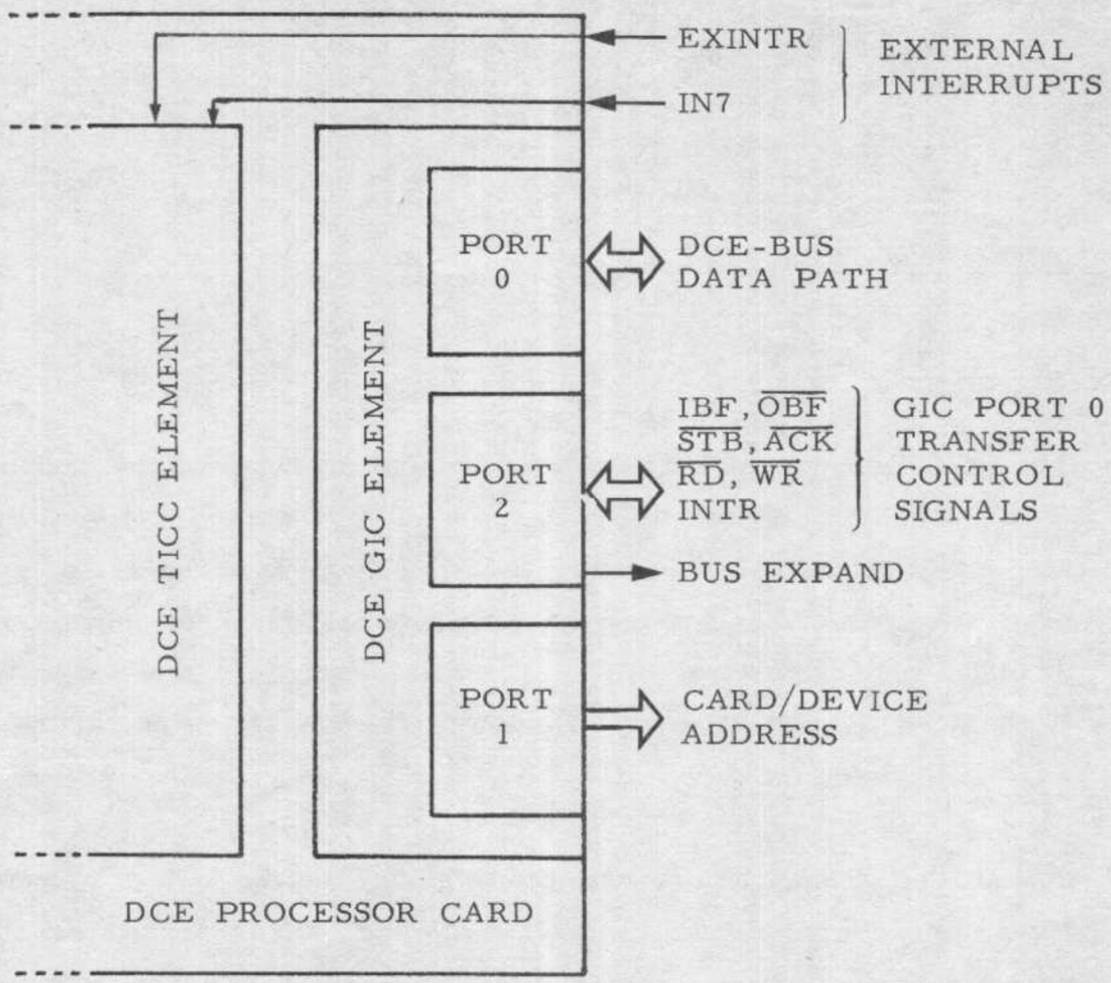


Figure 4.1 : DCE-BUS GIC Port Allocation

In the software controlled transfer modes, Port 0 must be alternated between input and output by software command. The fast-bus mode with DCE-LSA requires Port 0 to be set up as bi-directional. In both modes Port 1 carries the card/device address, and Port 2 carries control signals. Software controlled transfer operations require software activation of the $\overline{RD}$ and $\overline{WR}$ signals with appropriate timing. Fast-bus configured systems generate these $\overline{RD}$ and $\overline{WR}$ signals automatically via the DCE-LSA. The DCE-BUS also enables several DCE processors to operate in master-slave mode under software control.

The memory-mapped I/O mode does not use the GIC at all. The DCE-LSA uses the address, data, read and write signals from the CPU (via the X-BUS) to drive the DCE-BUS.

Two external interrupt request signals are provided on the DCE-BUS. Section 3.2 describes the processing of these interrupts in the 8-bit DCE systems. More interrupt vectors are possible by special logic on the external card to supply interrupt source indentification data.

4.1.2 Software Controlled READ Operation

The Read operation transfers data from the DCE-BUS data path (GIC Port 0) to the selected CPU register of the DCE processor (the accumulator when using the DCE GIC I/O macro commands). Figure 4.2 illustrates the DCE-BUS configured for software controlled read operations.

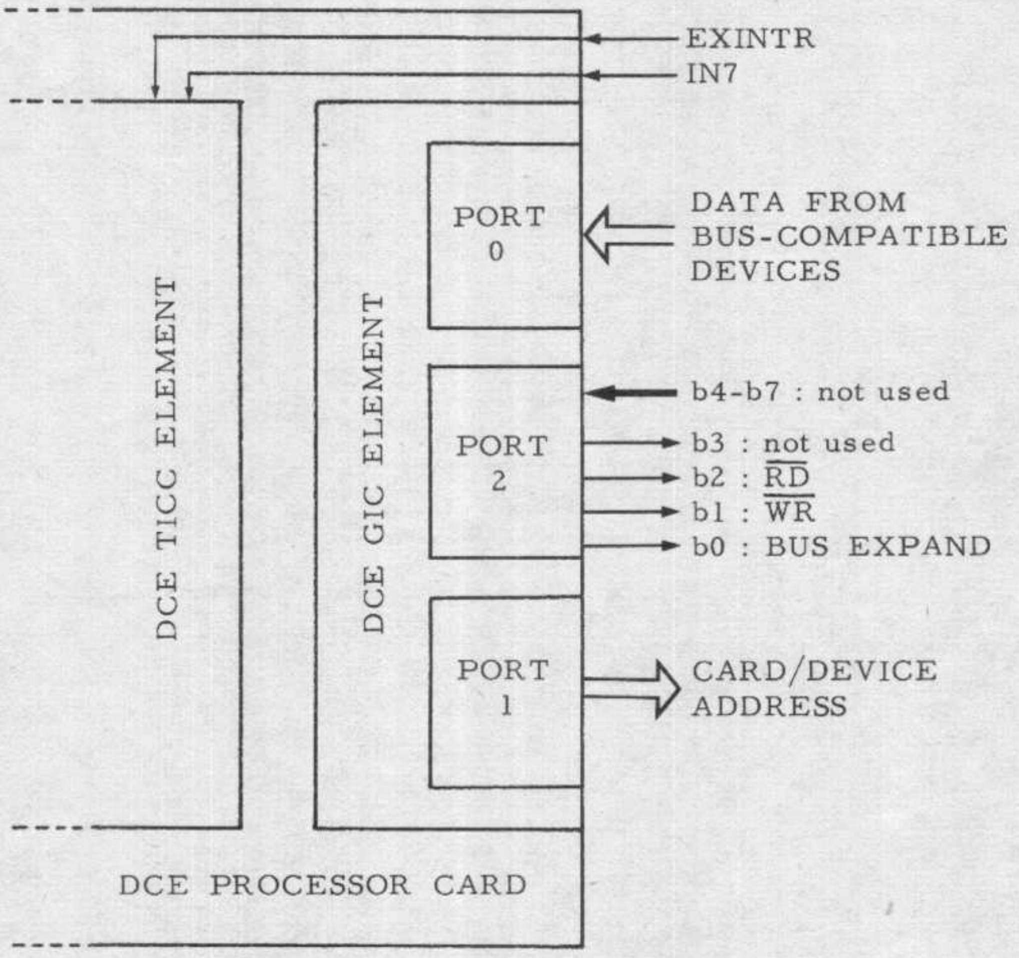
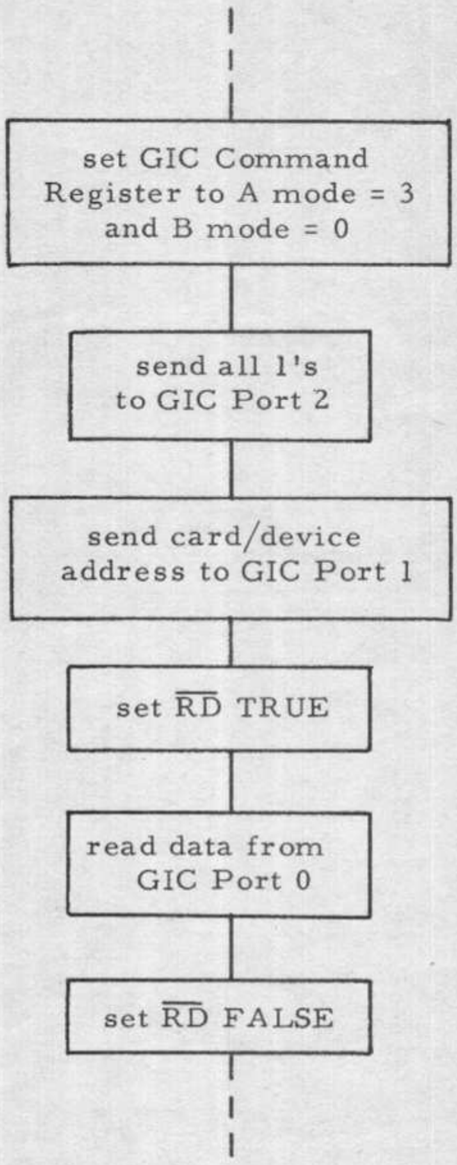


Figure 4.2 : Software Controlled READ Configuration

To place the DCE-BUS in the software controlled Read mode, the GIC should be set up with the GIC configuration command 'GICC 3,0' and GIC Port 2 should be set to all 1's. The complete software controlled read sequence is illustrated below. The stabilization times are accounted for by the execution times of the read command instructions.


set GIC Command Register to A mode = 3 and B mode = 0
send all 1's to GIC Port 2
send card/device address to GIC Port 1
set $\overline{\text{RD}}$ TRUE
read data from GIC Port 0
set $\overline{\text{RD}}$ FALSE

High-speed read control signal generation by hardware is explained in Section 4.1.4.

The following example realizes the software controlled transfer of an 8-bit data word from a device connected to the DCE-BUS into the DCE processor CPU accumulator.

```
GICC    3,0        :  port 0 - input,  port 2 (4-7) - input,
                      port 1 - output, port 2 (0-3) - output.
MVI     A,FF
STGI    2          :  all 1's to GIC port 2
MVI     A,adr      :  load card/device address
STGI    1          :  send to GIC port 1
MVI     A,FB
STGI    2          :  set RD true
LDGI    0          :  read data from DCE-BUS into Accumulator
STA     data area  :  store data
MVI     A,FE
STGI    2          :  set RD false
```

|| * Note: interrupts should be disabled during the software read sequence

1010
1011

BUS EXPAND Signal

This signal is provided to allow the realization of large systems by doubling the number of bus-compatible modules that may be driven by a single DCE processor module. It can be used as an extra addressing line to switch between two banks of bus-compatible modules, each of which responds similarly to the Card/Device addresses.

In normal small system usage, this signal may be used to disable the address decode logic on all the bus-compatible modules on the DCE-BUS while they are inactive. Such an arrangement will prevent the accidental activation of modules.

The above example assumes that this signal when high enables all the address decode logic. It is therefore set at the start of the sequence and cleared at the end.

4.1.3 Software Controlled WRITE Operation

The Write operation transfers data from a CPU register of the DCE processor to the DCE-BUS Data Path (GIC Port 0). The accumulator is used by the DCE GIC I/O macro commands. Figure 4.3 illustrates the DCE-BUS configured for software controlled Write operations.

To place the DCE-BUS in the software controlled Write mode, the GIC should be set up with the GIC Configuration Command 'GICC 1,0' and GIC Port 2 should be set to all 1's. The complete software controlled Write sequence is illustrated below. The stabilization times are accounted for by the execution times of the Write command instructions.

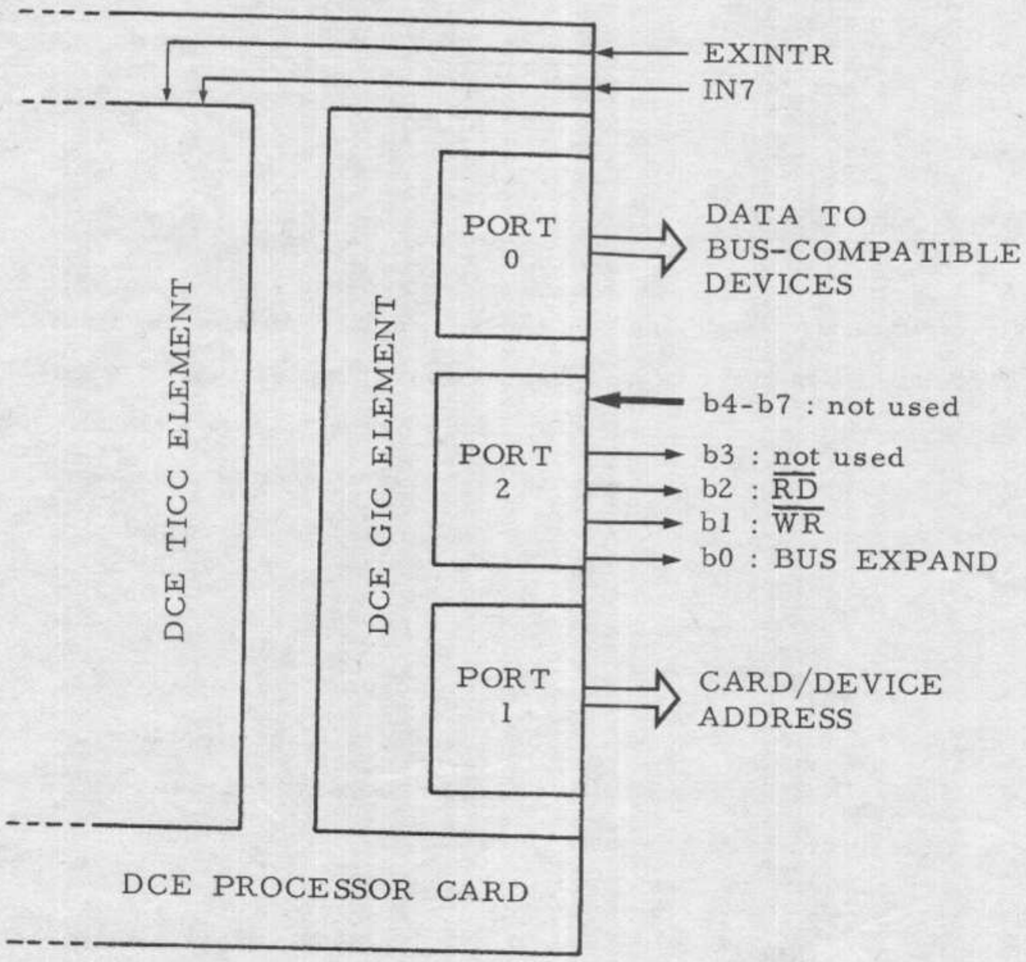
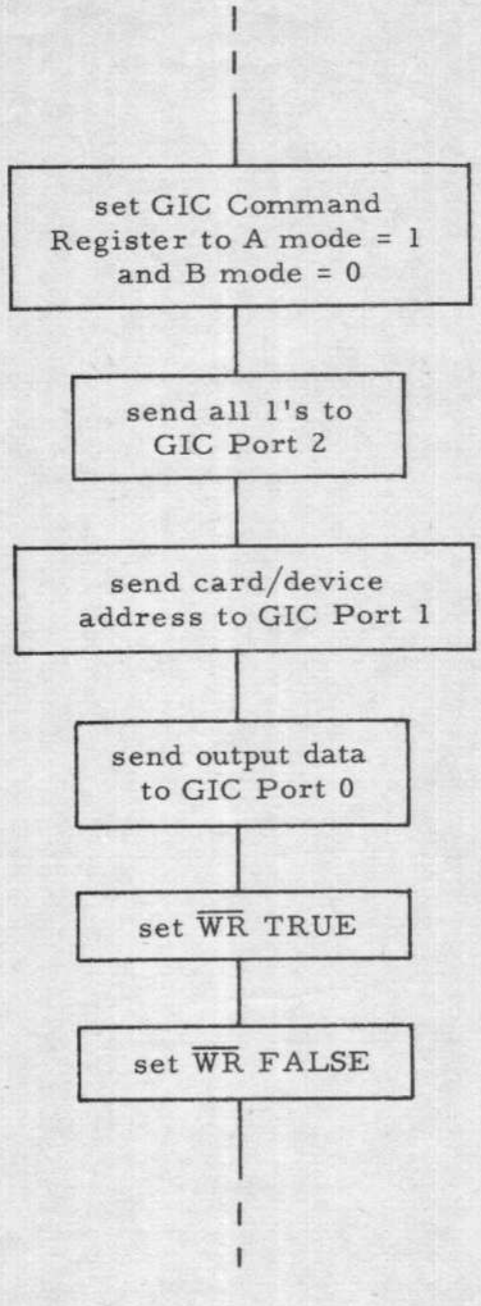


Figure 4.3 : Software Controlled WRITE Configuration


set GIC Command Register to A mode = 1 and B mode = 0
send all 1's to GIC Port 2
send card/device address to GIC Port 1
send output data to GIC Port 0
set $\overline{WR}$ TRUE
set $\overline{WR}$ FALSE

H 80

High-speed write control signal generation by hardware is explained in Section 4. 1. 4.

The following example realizes the software controlled transfer of an 8-bit data word from the DCE processor accumulator to a device connected to the DCE-BUS.

```
GICC    1,0      :  port 0 - output, port 2 (4-7) - input
                    port 1 - output, port 2 (0-3) - output
MVI     A,FF
STGI    2        :  all 1's to GIC Port 2
MVI     A,adr    :  load card/device address
STGI    1        :  send to GIC Port 1
MVI     A,data   :  load data to be output
STGI    0        :  send to GIC Port 0
MVI     A,FD
STGI    2        :  set WR true
MVI     A,FE
STGI    2        :  set WR false
```

!! * Note: interrupts should be disabled during the software write sequence. !!

4.1.4 Hardware Controlled READ/WRITE Operation (FAST-BUS Mode)

The Fast-Bus mode of operation assumes that the system is implemented together with the DCE-LSA card (see Section 6.9). The Fast-Bus mode of operation uses the GIC Port 0 in bi-directional mode. Data is moved between the DCE-BUS and the DCE processor under control of the handshake signals from Port 2. The DCE-LSA generates the $\overline{RD}$ and $\overline{WR}$ signals in response to these input and output handshake control signals which are defined in Section 2.4

Figure 4.4 illustrates the DCE-BUS configured for the FAST-BUS mode of operation. To place the DCE-BUS in the FAST-BUS mode the GIC should be set up with the GIC configuration command 'GICC 8,1'. A complete read and write sequence is illustrated below. The stabilization and transfer control timing is accounted for by the DCE-LSA card logic.

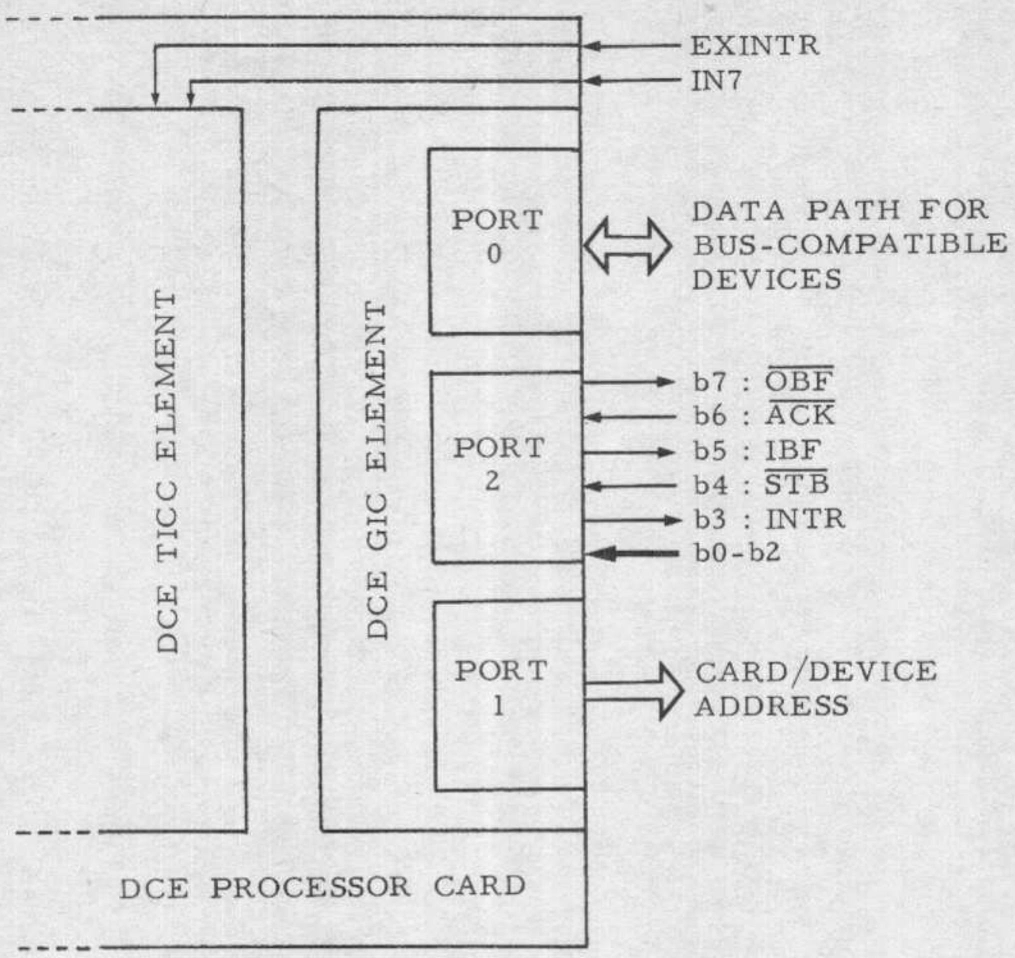
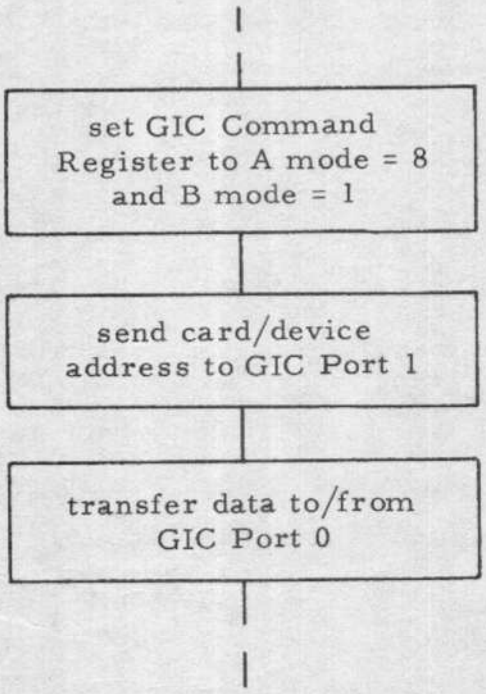

EXINTR
IN7
DCE TICC ELEMENT
DCE GIC ELEMENT
PORT 0
DATA PATH FOR BUS-COMPATIBLE DEVICES
PORT 2
b7 : $\overline{\text{OBF}}$
b6 : $\overline{\text{ACK}}$
b5 : IBF
b4 : $\overline{\text{STB}}$
b3 : INTR
b0-b2
PORT 1
CARD/DEVICE ADDRESS
DCE PROCESSOR CARD

Figure 4.4 : Fast-Bus Configuration


set GIC Command Register to A mode = 8 and B mode = 1
send card/device address to GIC Port 1
transfer data to/from GIC Port 0

Fast-Bus Read/Write Sequence

The following example realizes the hardware controlled transfer of data between the DCE processor accumulator and the device connected to the DCE-BUS

```
GICC      8,1         :  port 0 - bi-directional, port 2 (3-7) - H.S.C
                         port 1 - output, port 2 (0-2) - input
MVI       A,adr       :  load card/device address
STGI      1           :  send to GIC Port 1
MVI       A,data
STGI      0           :  write data to Port 0
or
(LDGI     0           :  read data from Port 0)
```

Note: interrupts should be disabled during these read/write sequences.

To realize interrupt controlled transfers using the Fast-Bus mode, connect the INTR signal of the GIC to one of the interrupt lines. In this case each time the external device has data available for input, or is ready to receive data, an interrupt request will be automatically generated.

4.2 DCE-BUS INTERRUPT FACILITIES

The DCE-BUS carries the two available external vectored interrupt requests to the DCE processor. Connecting modules may drive either of these interrupt request lines asynchronously. See Section 3.2 for a full description of interrupt handling by the DCE processor.

The number of available external interrupt requests may be expanded beyond two by the insertion of priority encoding logic on the bus compatible modules. This should gate interrupt source identification data onto the input data path, upon a read command. This data would then be decoded by software to determine the interrupting device. DAI Real-World cards with multiple-level interrupts may also be used in this respect. See Section 4. 3. 2 for details.

4. 3 DCE-BUS COMPATIBLE MODULES

Figure 4. 5 illustrates the basic logical subsets of a DCE compatible module including the logical definitions necessary for multiple interrupt vectors. The DCE-BUS allows some functions to be implemented using software or hardware logic. A good example of this is the realization of Read/Write Control and the multiple interrupt definition vectors. DAI Real-World cards are logically realized for best economic and format compatibility.

4. 3. 1 Data Devices

The DCE-BUS allocates 8 bits for card and device addressing. Generally, four of the bits are allocated for the specific card address and the other four are allocated for addressing data devices (registers) on that card. Data devices are usually logic elements that have read and write capabilities and are connectable to the DCE-BUS data path. Typically (as in DAI real-world cards) these devices are logical controllers for a particular type of external device. Data devices can require interrupt servicing. It is quite practical and economic to consider a direct link to a data device with dedicated device address lines, that eliminates the necessity for device select decoding. In this case the number of data devices cannot exceed eight. Designers constructing DCE-BUS compatible modules must trade-off the above approach against the added hardware necessary for device address decoding, to fulfill their specific requirements.

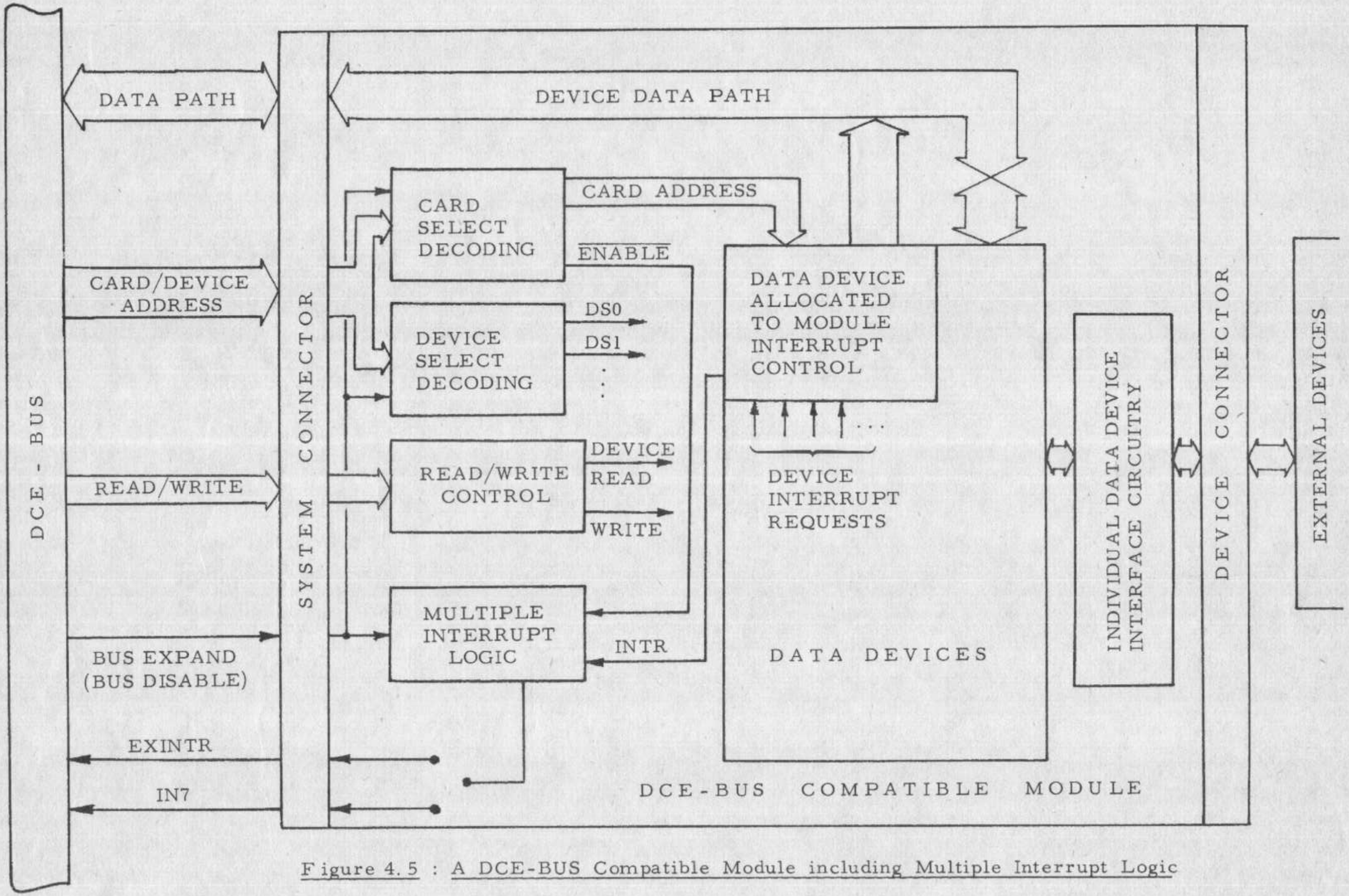


Figure 4.5 : A DCE-BUS Compatible Module including Multiple Interrupt Logic

4.3.2 Multiple Interrupt Control

It can be quite interesting to allow several data devices to interrupt the DCE processor. An effective technique for realizing this would be to allocate one of the data devices as a read-only device that places on the DCE-BUS data path the number of the interrupting device, when addressed by the card/device address lines.

A typical set of logic elements to realize this function would be a priority encoder and some tri-state buffers. Figure 4.6 illustrates this arrangement.

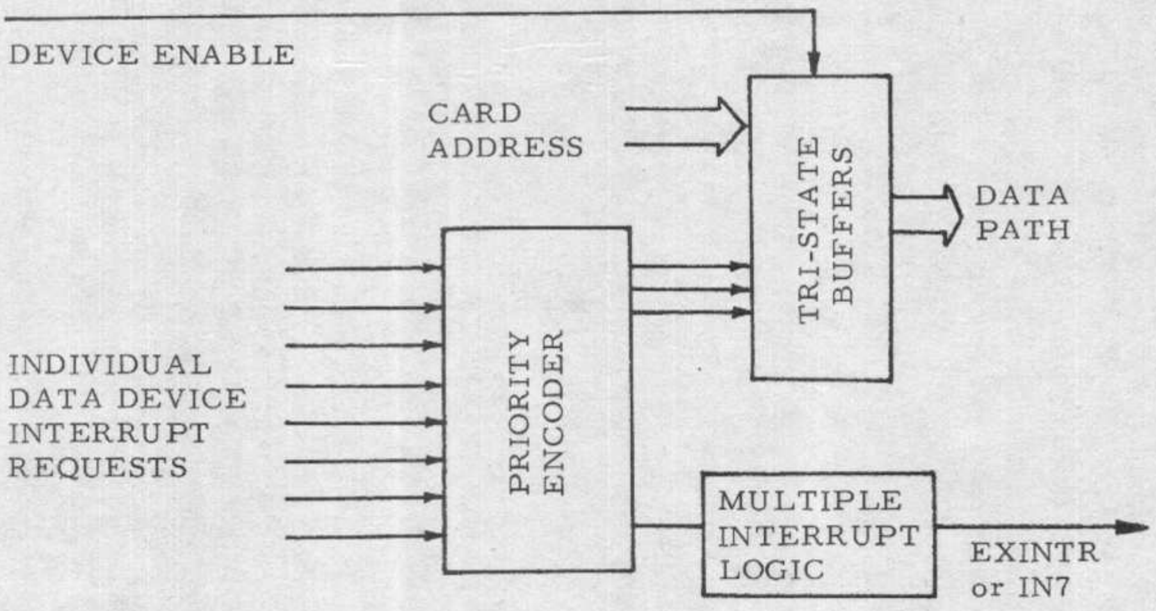


Figure 4.6 : Typical Module Interrupt Controller

To allow more than two bus compatible modules to interrupt the DCE processor, the modules can be equipped with the multi-module interrupt logic. Figures 4.7 and 4.8 illustrate the necessary logic functions to realize multiple vectored interrupts from bus compatible modules. Notice should be taken of the combination of software and hardware logic functions.

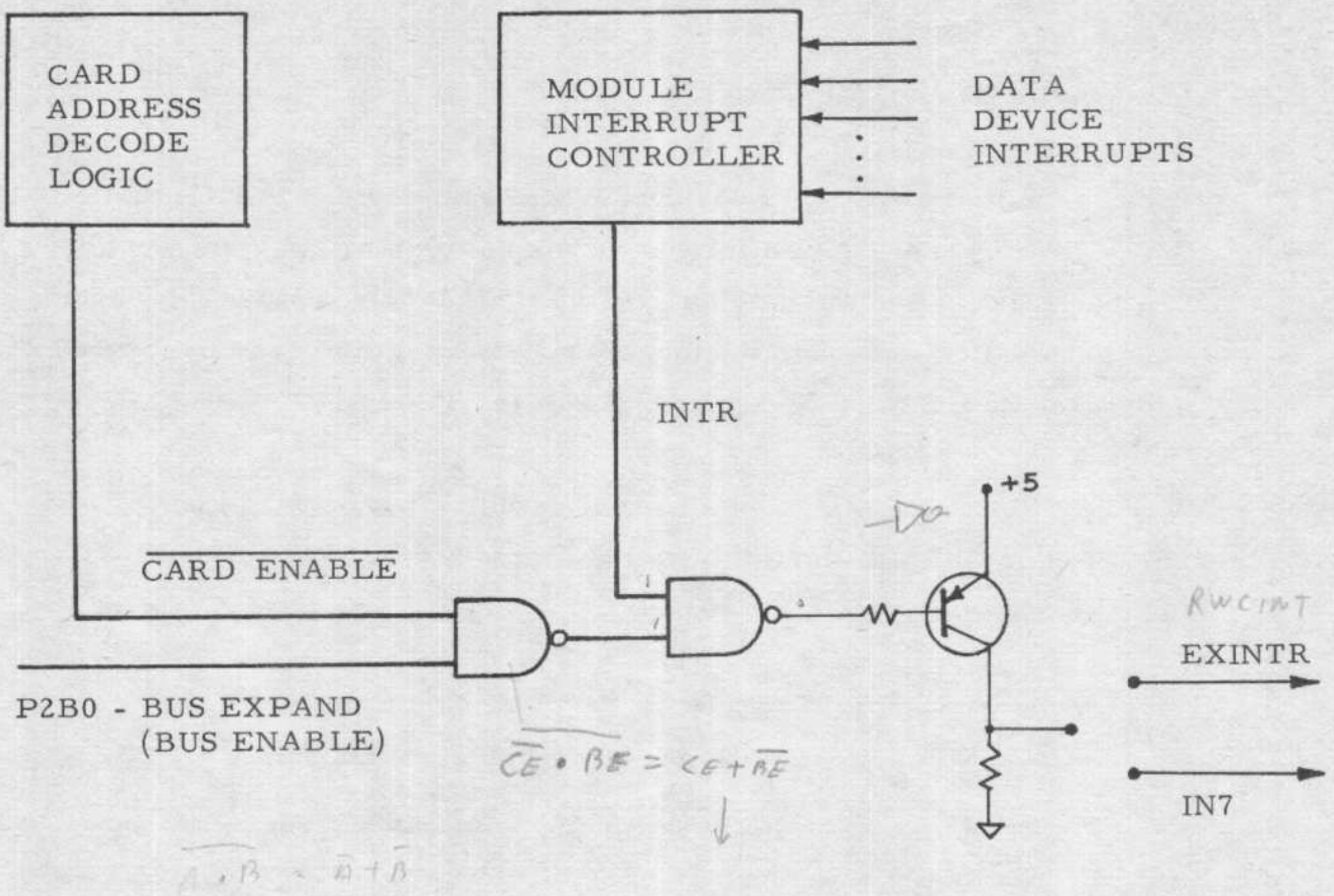

CARD ADDRESS DECODE LOGIC
MODULE INTERRUPT CONTROLLER
DATA DEVICE INTERRUPTS
INTR
+5
$\overline{\text{CARD ENABLE}}$
RWCINT
EXINTR
P2B0 - BUS EXPAND (BUS ENABLE)
$\overline{CE} \cdot BE = CE + \overline{BE}$
IN7
$\overline{A \cdot B} = \bar{A} + \bar{B}$

Figure 4.7 : Hardware Logic Necessary For Multiple Module Interrupt Vectoring

By using this technique on each participating bus compatible module, a large number of different interrupt vectors can be connected to each DCE processor. Note the program logic functions that form a part of this scheme. The starting address of this software logic sequence must be at location 0010H or 0038H of the DCE program memory, depending on the usage of interrupt request lines EXINTR or IN7 respectively.

$RWC\,INT = INTR \cdot (CE + \overline{BE})$

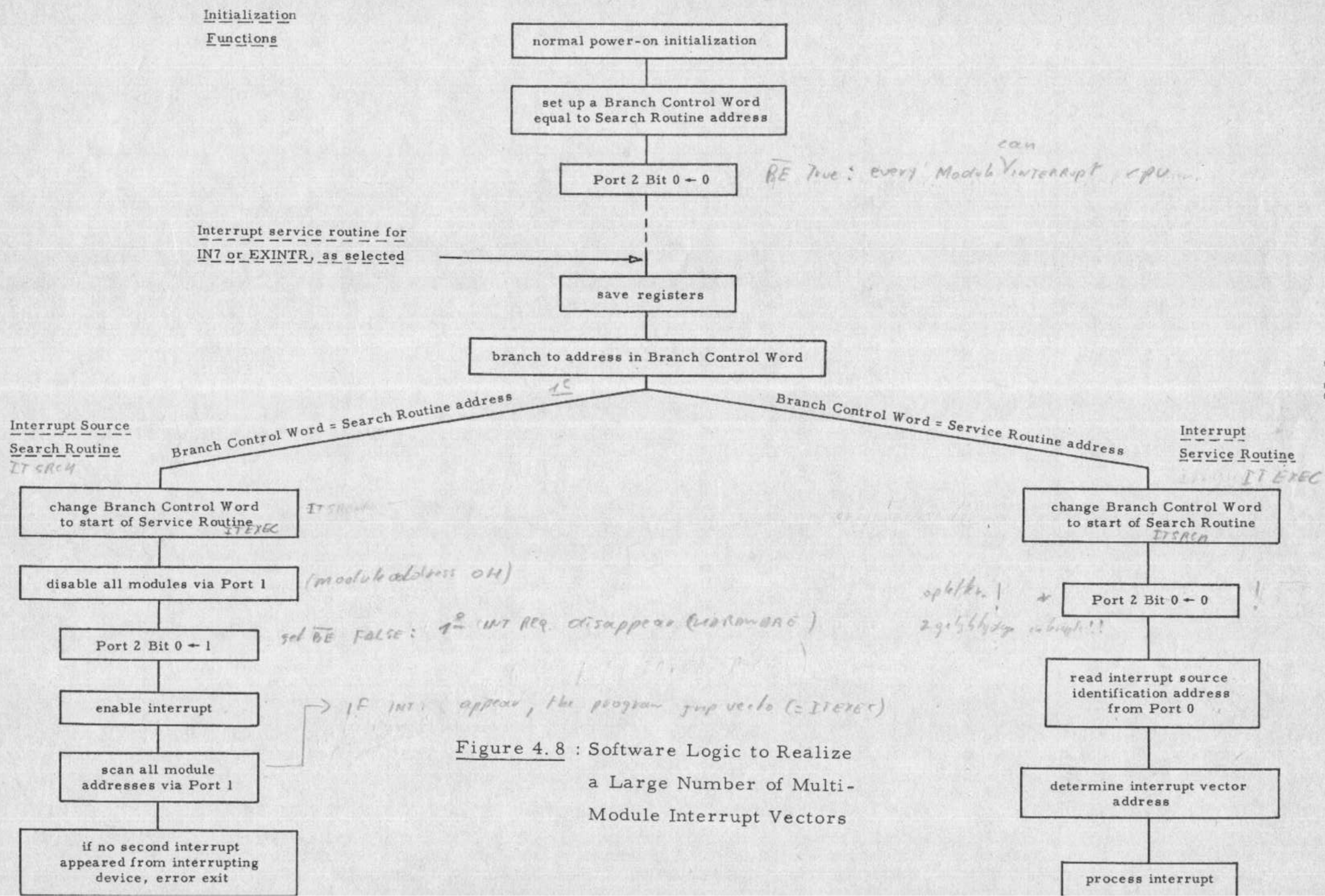

Initialization
Functions
normal power-on initialization
set up a Branch Control Word equal to Search Routine address
Port 2 Bit 0 ← 0
can
$\overline{BE}$ true: every Module can interrupt cpu.
Interrupt service routine for IN7 or EXINTR, as selected
save registers
branch to address in Branch Control Word
Branch Control Word = Search Routine address
Branch Control Word = Service Routine address
Interrupt Source Search Routine
IT SRCH
Interrupt Service Routine
IT EXEC
change Branch Control Word to start of Service Routine
IT EXEC
IT SRCH
disable all modules via Port 1
(module address 0H)
Port 2 Bit 0 ← 1
set $\overline{BE}$ FALSE: 1st INT REQ. disappear
enable interrupt
IF INT appear, the program jump vector (= ITEXEC)
scan all module addresses via Port 1
if no second interrupt appeared from interrupting device, error exit
change Branch Control Word to start of Search Routine
ITSRCH
Port 2 Bit 0 ← 0
read interrupt source identification address from Port 0
determine interrupt vector address
process interrupt

Figure 4.8 : Software Logic to Realize a Large Number of Multi-Module Interrupt Vectors

4.3.3 Read/Write Control

The I/O devices within the DCE processor architecture are structured to make the addressing and data transfer protocol between them and the 8080 CPU logically similar to those for memory locations. This logical order is extended to the bus compatible data devices. Therefore, in addition to the card/device address logic, they also need a read pulse ($\overline{RD}$) and a write pulse ($\overline{WR}$).

Bus compatible modules can be constructed that generate these $\overline{RD}$ and $\overline{WR}$ signals automatically with hardware; or they can be generated simply by software sequences. Figures 3.9 and 3.10 illustrate the basic Read and Write timing sequences.

Systems designed for software controlled Read and Write operations can be enacted as described in Sections 4.1.2 and 4.1.3.

Hardware Read/Write logic must be constructed to satisfy the timing diagrams below. The DCE-LSA card will perform this function.
DAI Real-World cards are designed to operate within systems that are structured for hardware controlled I/O transfer via the DCE-LSA, or for software controlled I/O transfer. Figures 4.11 and 4.13 illustrate the timing sequences for generating $\overline{RD}$ and $\overline{WR}$ pulses from the GIC handshake control signals. Figures 4.12 and 4.14 illustrate the hardware logic elements for realizing such a scheme, for bus compatible modules containing LSI peripheral devices standard to the 8080 (e.g. 8255).
DAI Real-World cards are constructed in this manner, using the 8255 as the RIC (Real-World Interface Control). Section 7 of this manual fully describes all the bus compatible Real-World modules.

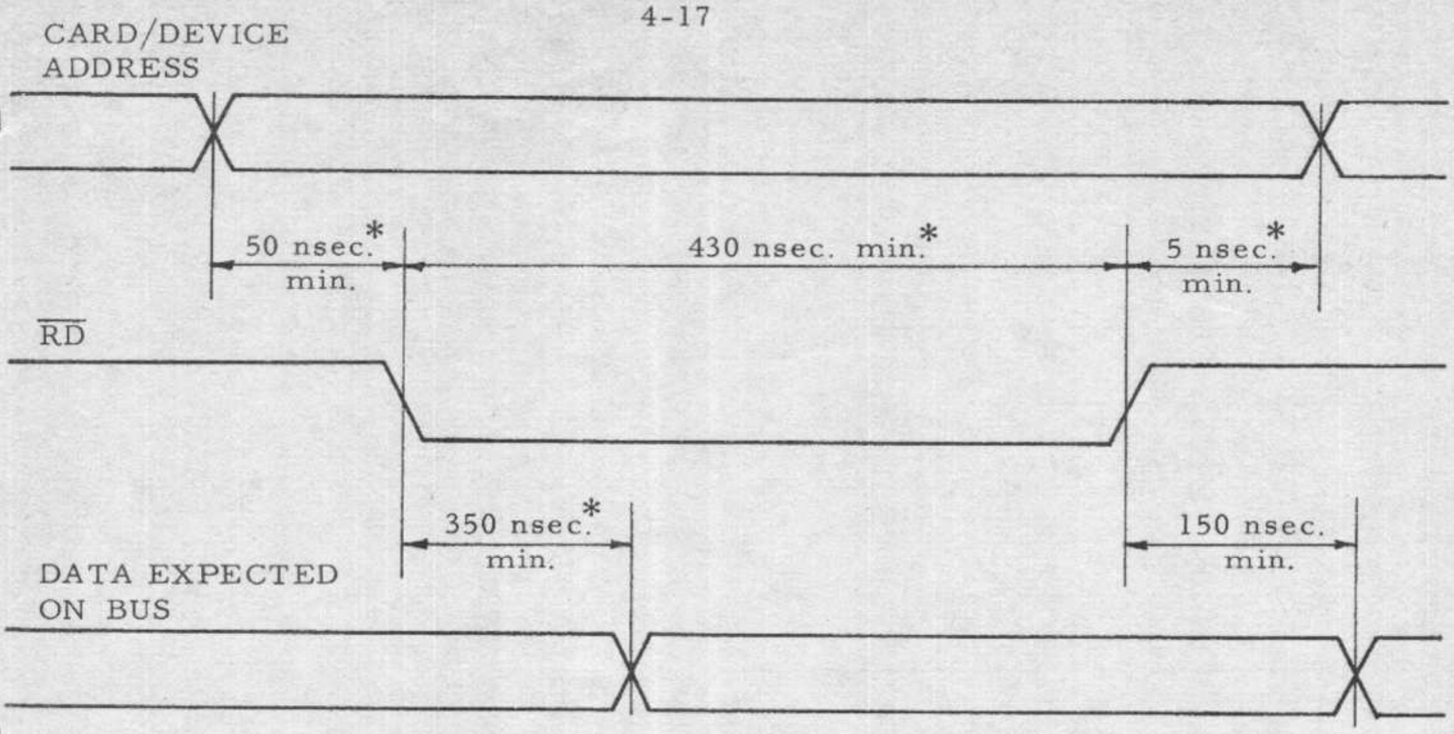


Figure 4.9 : DCE-BUS Basic READ Timing Sequence

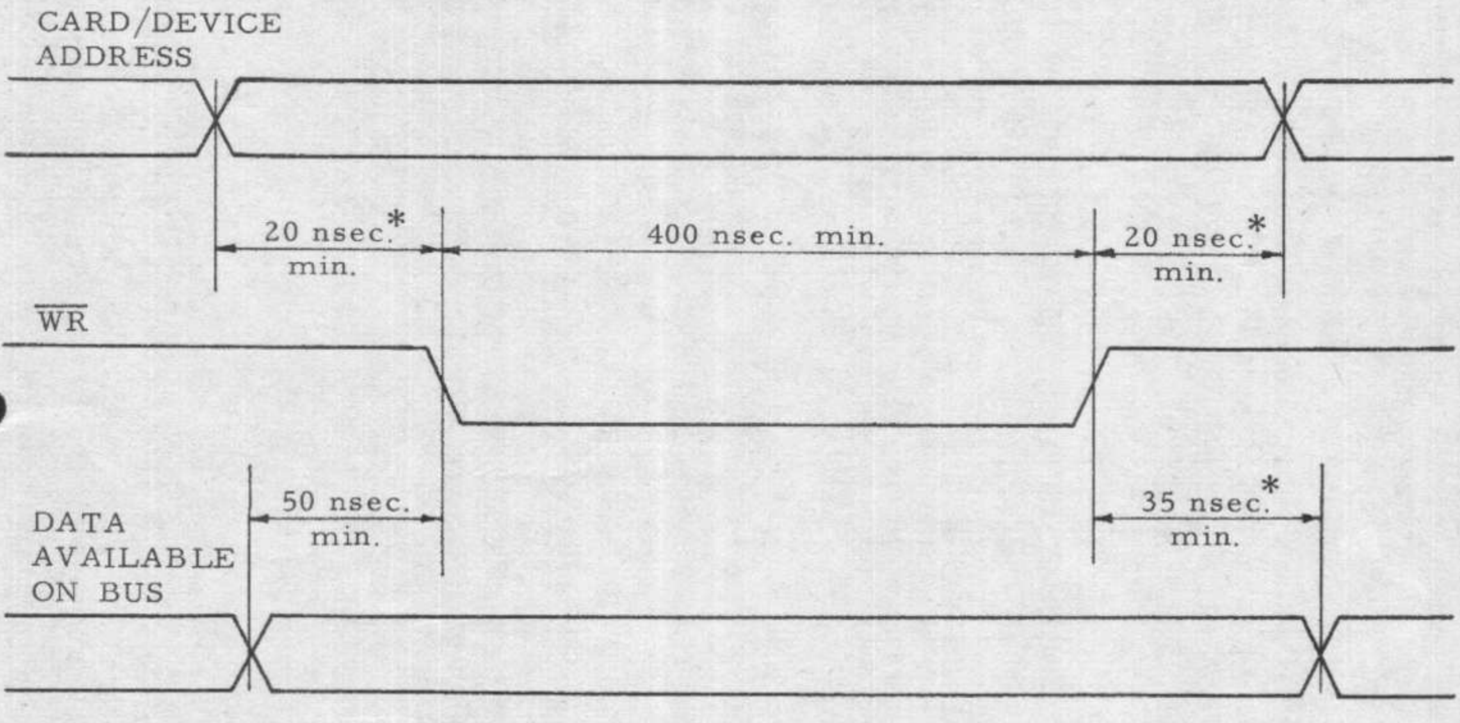
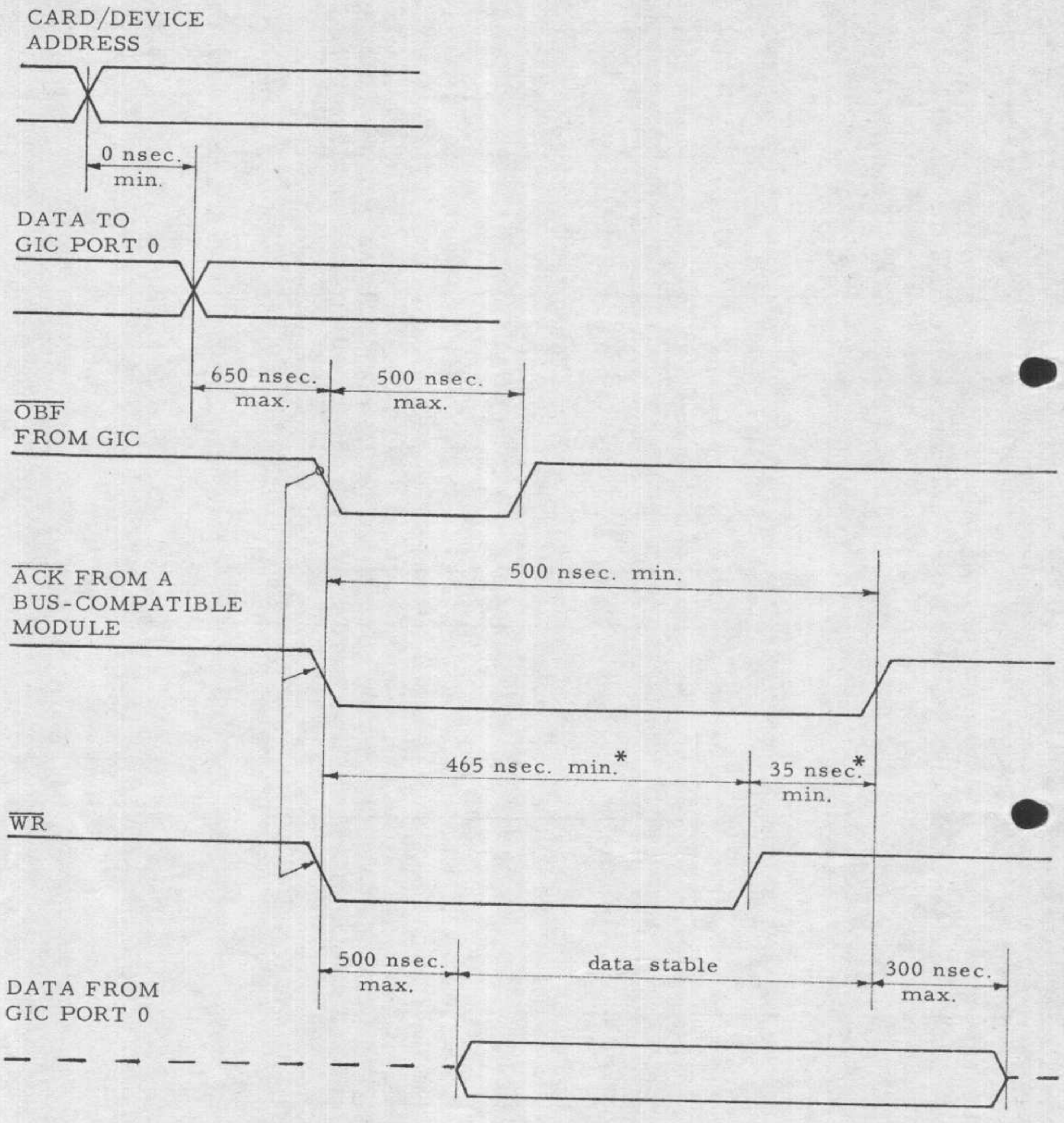


Figure 4.10 : DCE-BUS Basic WRITE Timing Sequence

Time between consecutive READs and/or WRITEs = 850 nsec. min.

* for 8255 type devices as Module Interface Controller



* for 8255 type devices as Module Interface Controller

Figure 4.11 : Using GIC Handshake Signals to Generate $\overline{\text{WR}}$ (Fast-Bus Mode)

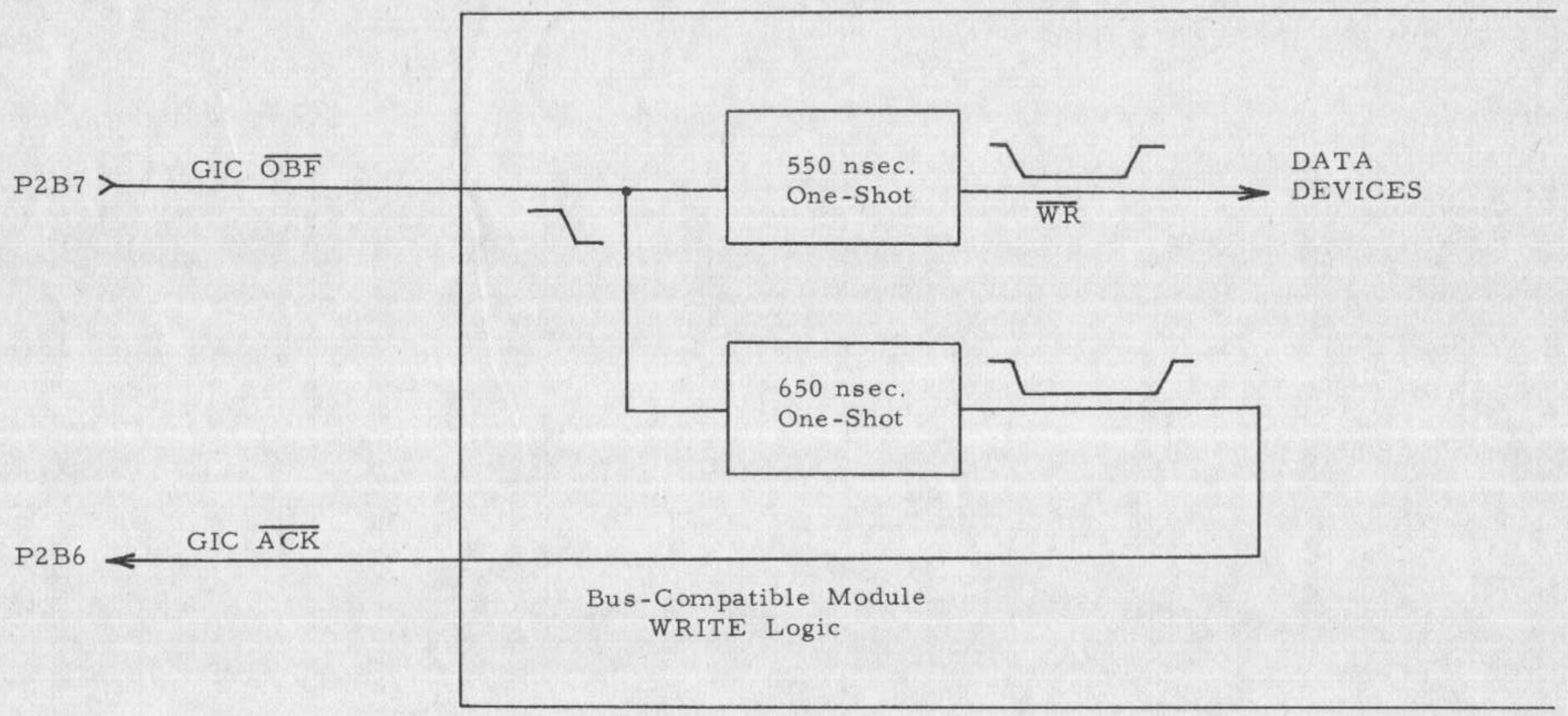
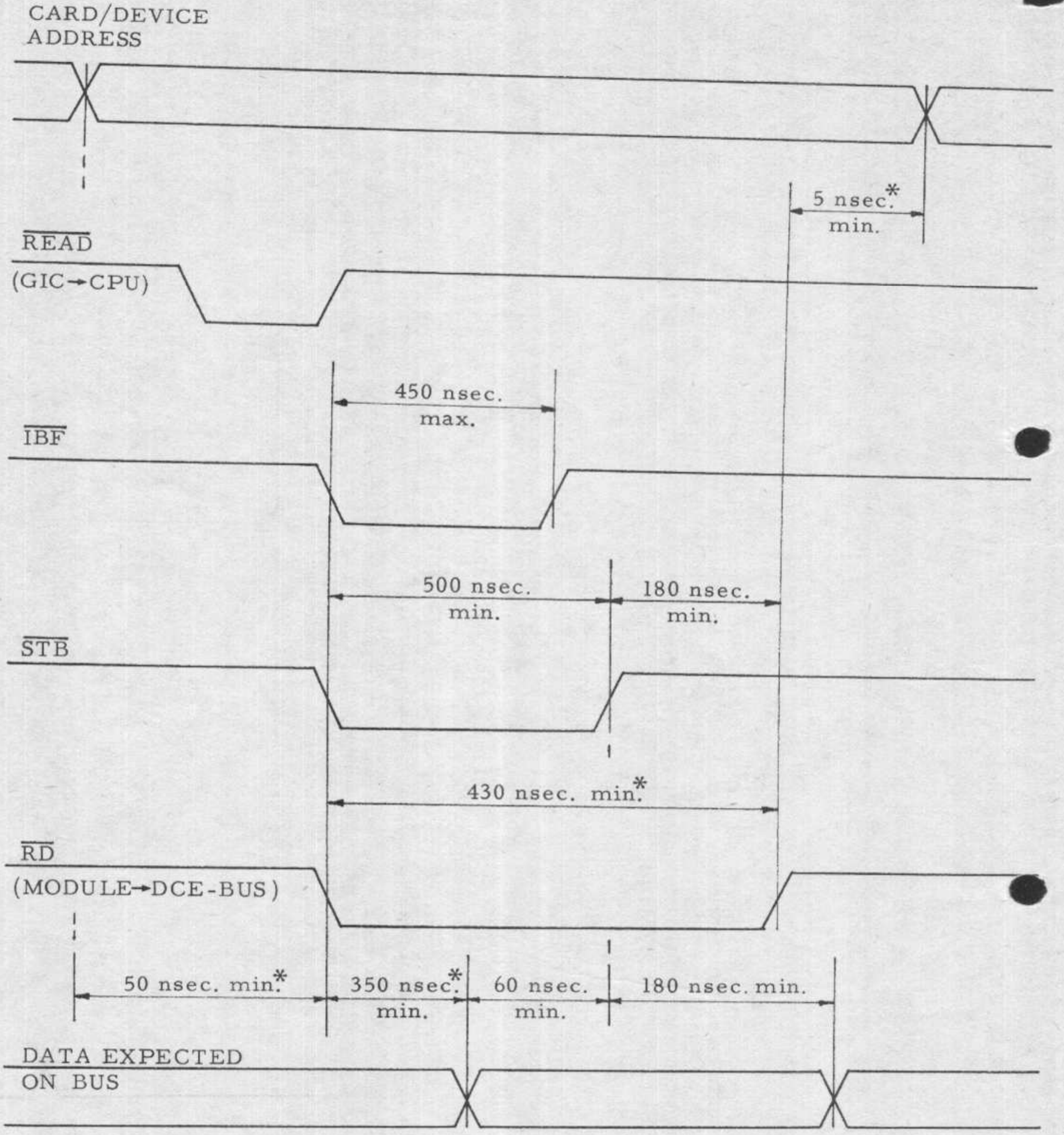

P2B7
GIC $\overline{\text{OBF}}$
550 nsec. One-Shot
$\overline{\text{WR}}$
DATA DEVICES
650 nsec. One-Shot
P2B6
GIC $\overline{\text{ACK}}$
Bus-Compatible Module WRITE Logic

Figure 4. 12 : A Hardware Logic Configuration to Generate $\overline{\text{WR}}$ on Bus-Compatible Modules that use 8255 type Devices (Fast-Bus Mode)



* for 8255 type devices as Module Interface Controller

Figure 4.13 : Using GIC Handshake Signals to Generate $\overline{\text{RD}}$ (Fast-Bus Mode)

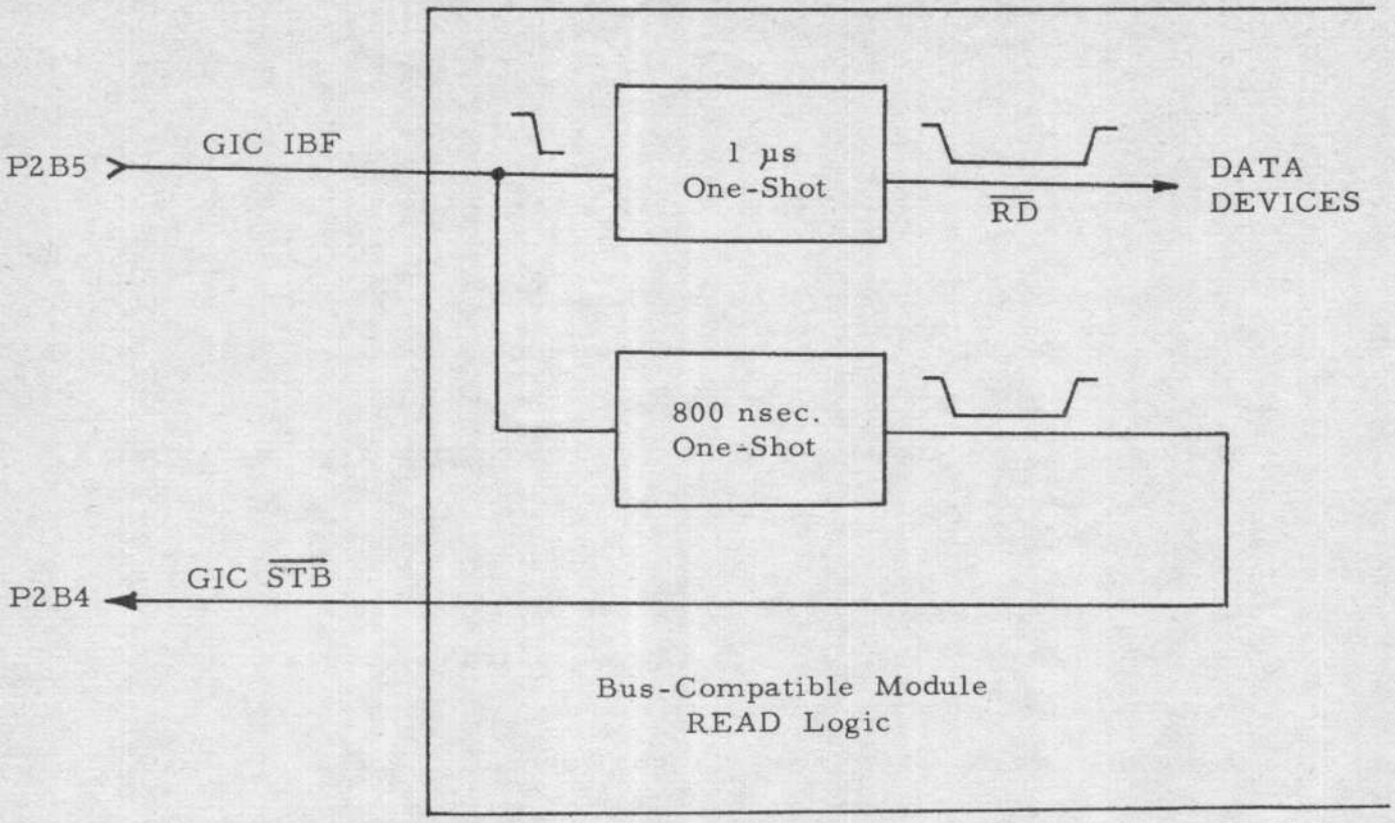


Figure 4.14 : A Hardware Logic Configuration to Generate $\overline{RD}$ on Bus-Compatible Modules that use 8255 type Devices (Fast-Bus Mode)

Note: Using this logic results in a dual transfer of data:

(i) transfer from GIC Input Buffer to DCE processor CPU Register,

(ii) transfer from Peripheral Device to GIC Input Buffer.

Shifting peripheral data into the DCE processor CPU requires two Read operations.

4.3.4 Card/Device Address Decoding

The DCE-BUS provides eight latched bits via GIC Port 1, intended for use as card/device addresses for bus compatible modules. Address decode logic should be provided as needed to select a particular card (module) and then a particular data device on that card.

It is possible to construct systems where bus compatible modules and data devices on them are directly addressed by the bits from GIC Port 1, thus eliminating the need for hardware address decode logic. However, with the insertion of hardware decoding as shown in Figure 4.5, the number of modules connectable to the DCE-BUS, and the number of data devices on each such module can be made large.

DAI Real-World cards use 4 out of the 8 address bits as card select address. The other 4 bits are used for device selection within the selected card.

4.4 DCE-BUS LOADING

The DCE-BUS is driven by the GIC device on a DCE processor module. Each GIC I/O line has a sink capability of 1.6mA at 0.4V.

The family of Real-World modules present a standard interface to the DCE-BUS. This is usually implemented by a Real-World Interface Controller (RIC), which is the same type of device as the GIC on DCE processor modules. Real-World modules with such an interface (eg. RWC-F, RWC-T24) present 1 unit load to the DCE-BUS. Modules which do not use a single device RIC interface to the DCE-BUS may present the equivalent of several unit loads to the DCE-BUS (eg. RWC-CCE).

The DCE-BUS can support at least 32 such unit loads without any further buffering.