uOttawa

L'Université canadienne
Canada's university

CEG3136 Computer Architecture II

Module 7 – Computer Buses and Parallel I/O

Notes for

Dr. Voicu Groza

Université d'Ottawa | University of Ottawa

www.uOttawa.ca

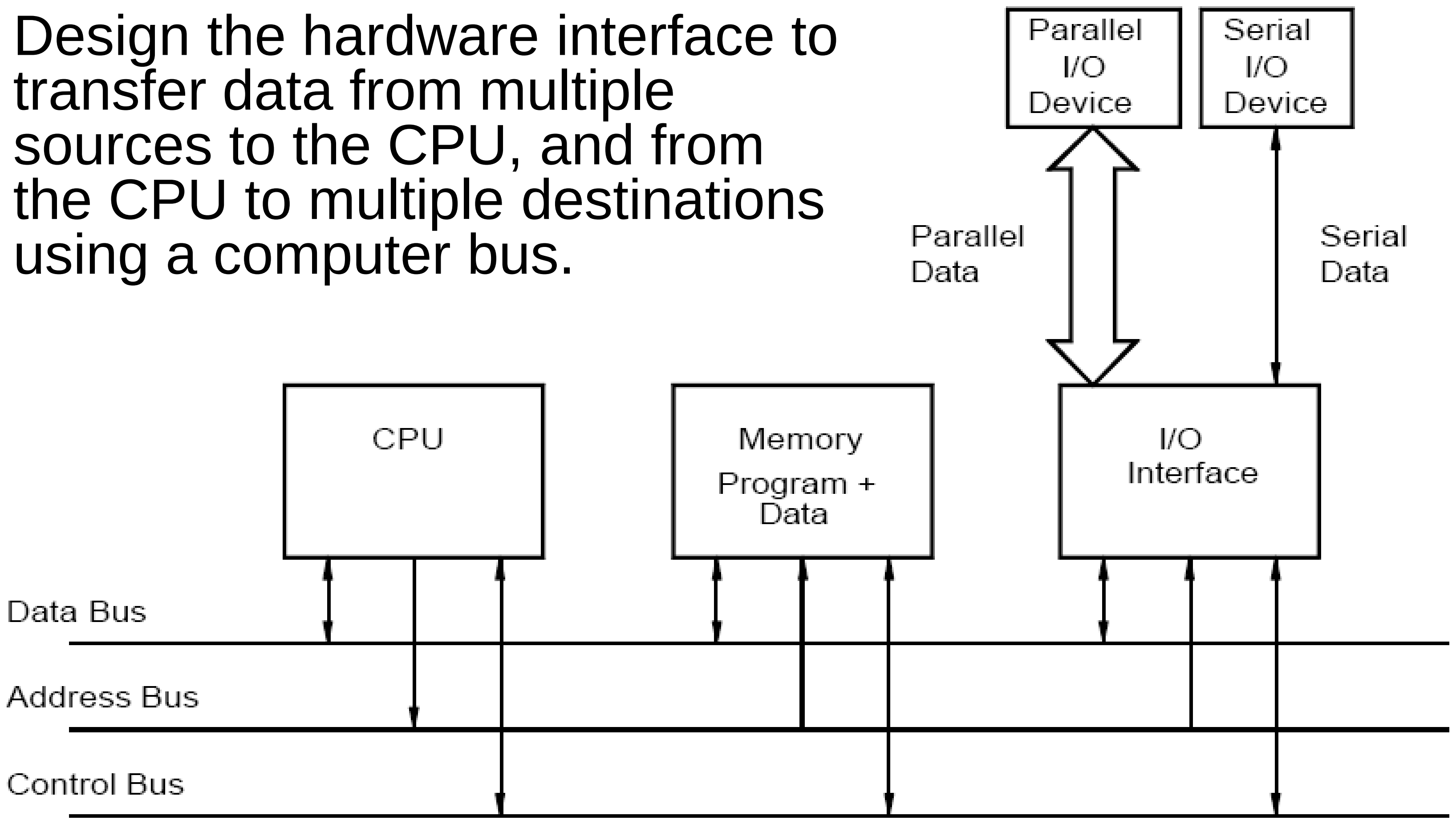
Topics of discussion

- Parallel Bus Architecture
- Memory Map versus Separate I/O
- I/O Synchronization
- Interfacing to Devices
- MC68HC9S12 Parallel Ports

- Reading: Chapter 11

Introduction

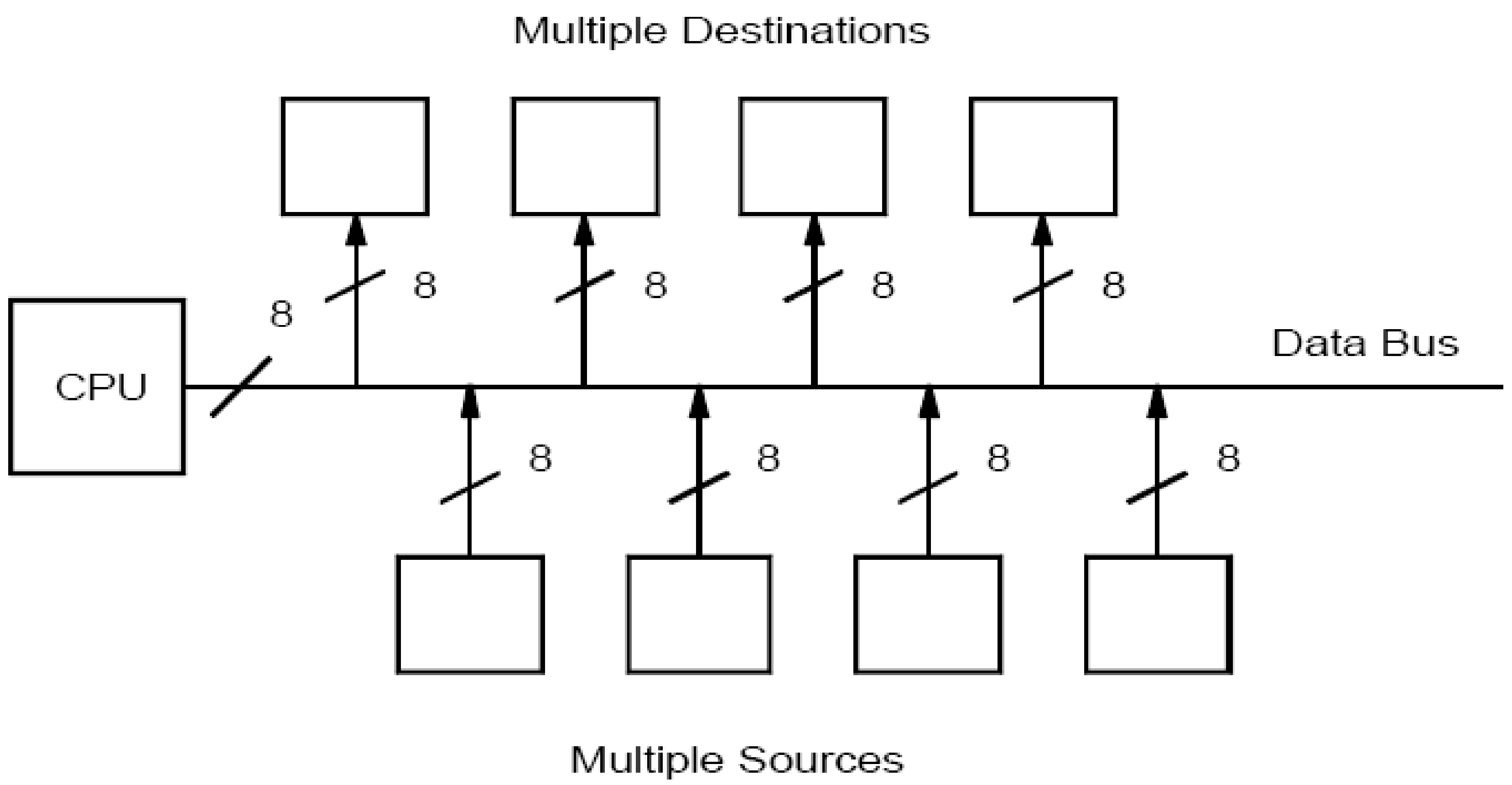
- Design the hardware interface to transfer data from multiple sources to the CPU, and from the CPU to multiple destinations using a computer bus.



The Computer Bus

- Data Bus
  - Bi-directional to carry data between CPU and I/O devices (including memory)
- Address Bus
  - Most often unidirectional with CPU as the source
- Control Bus
  - Carries other signals to control operation

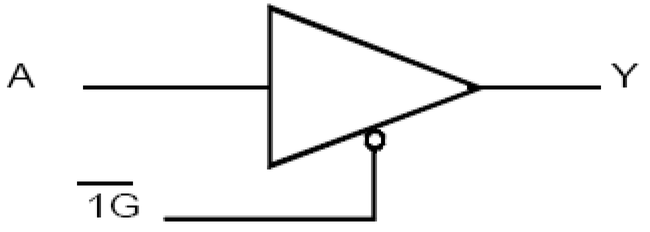
The Data Bus


Multiple Destinations
8
8
8
8
8
CPU
Data Bus
8
8
8
8
Multiple Sources

Circuits for Input Interface

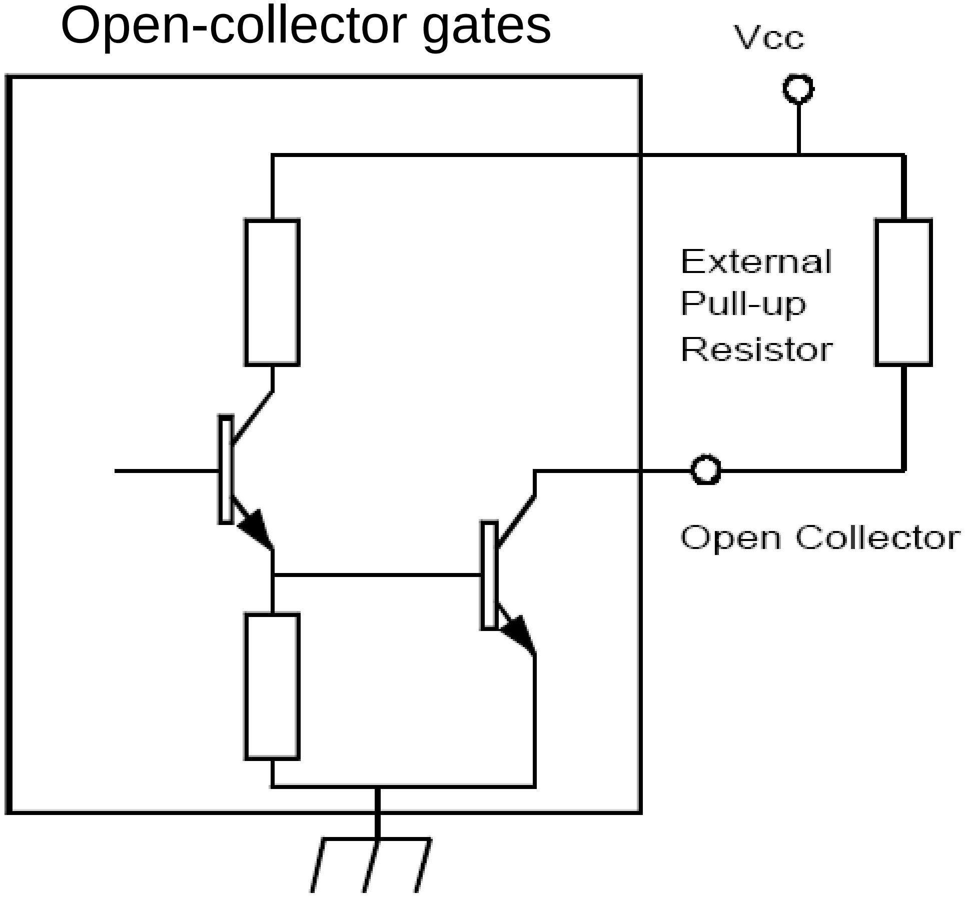
- Various sources of data use tri-state or open-collector gates

Tri-State gates



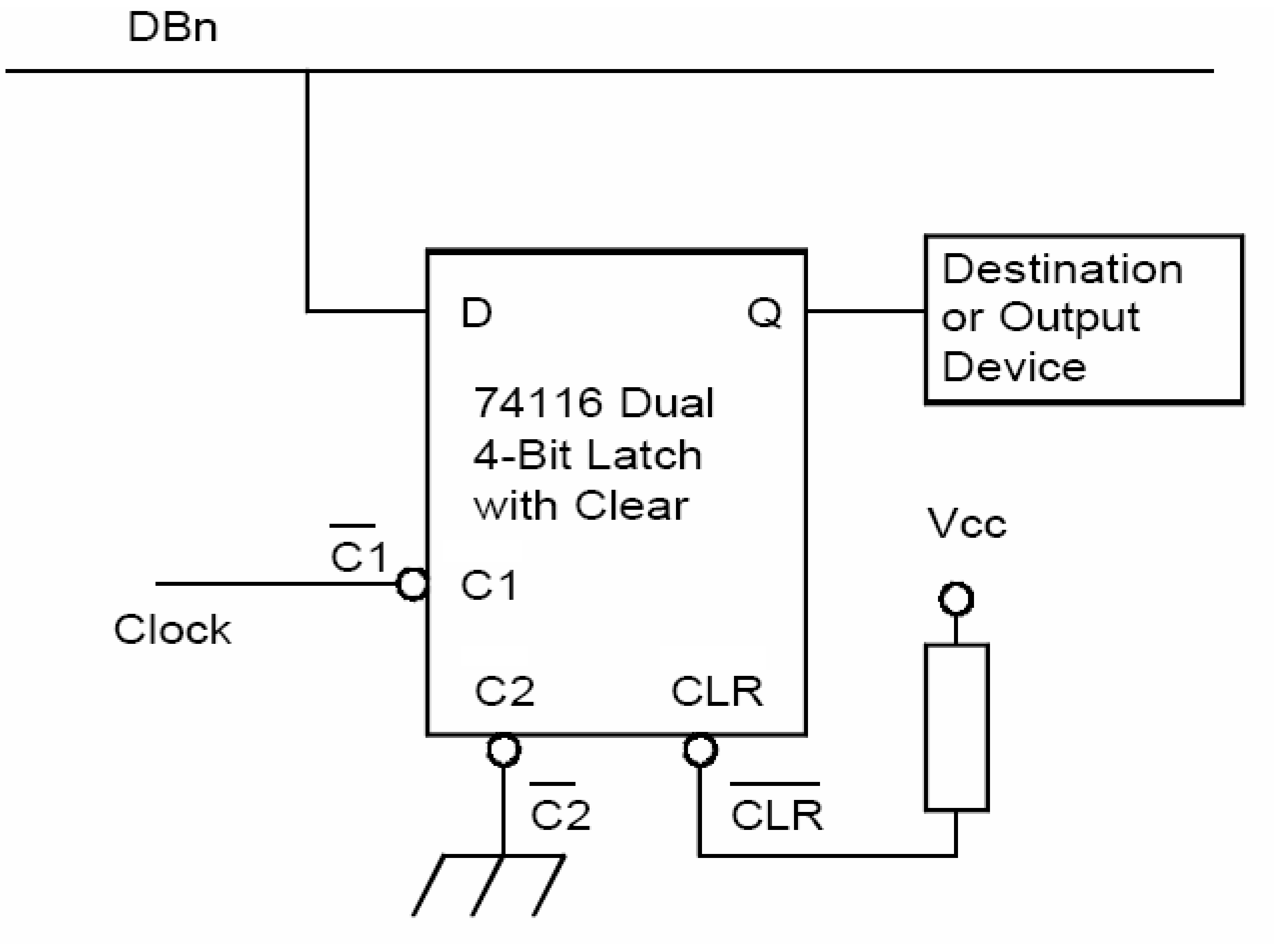
$\overline{1G}$	A	Y
0	0	0
0	1	1
1	0	High Impedance
1	1	High Impedance

Open-collector gates



Circuits for Output Interface

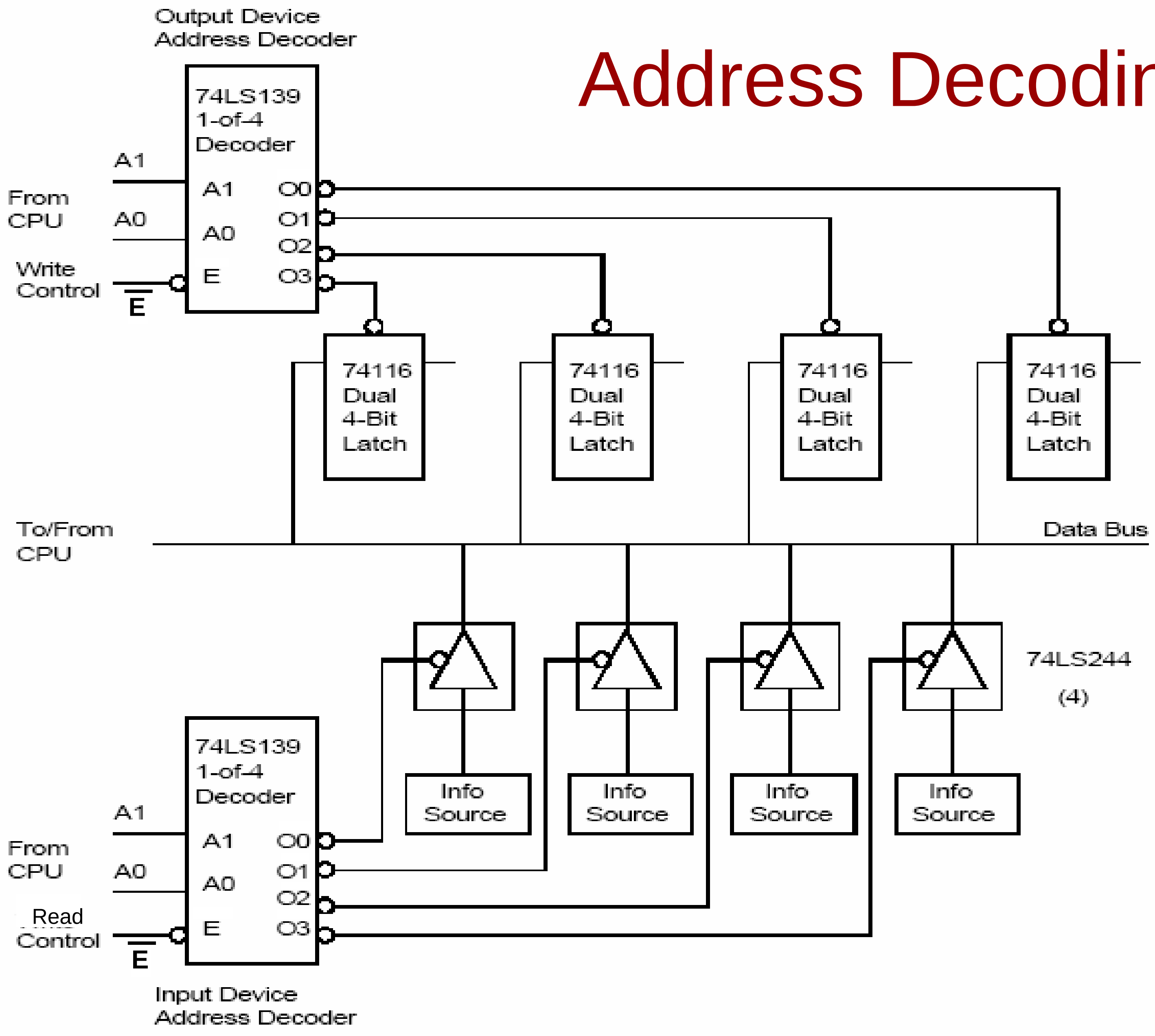
- Output interface is the latch (to capture data from the data bus)



Multiple Sources and Destinations

- Want to provide the ability for the CPU to select one of many sources and destinations.
- Address decoding
  - Use an address decoder to select devices
  - Two control lines
    - WRITE_CONTROL – enables output
    - READ_CONTROL – enables input

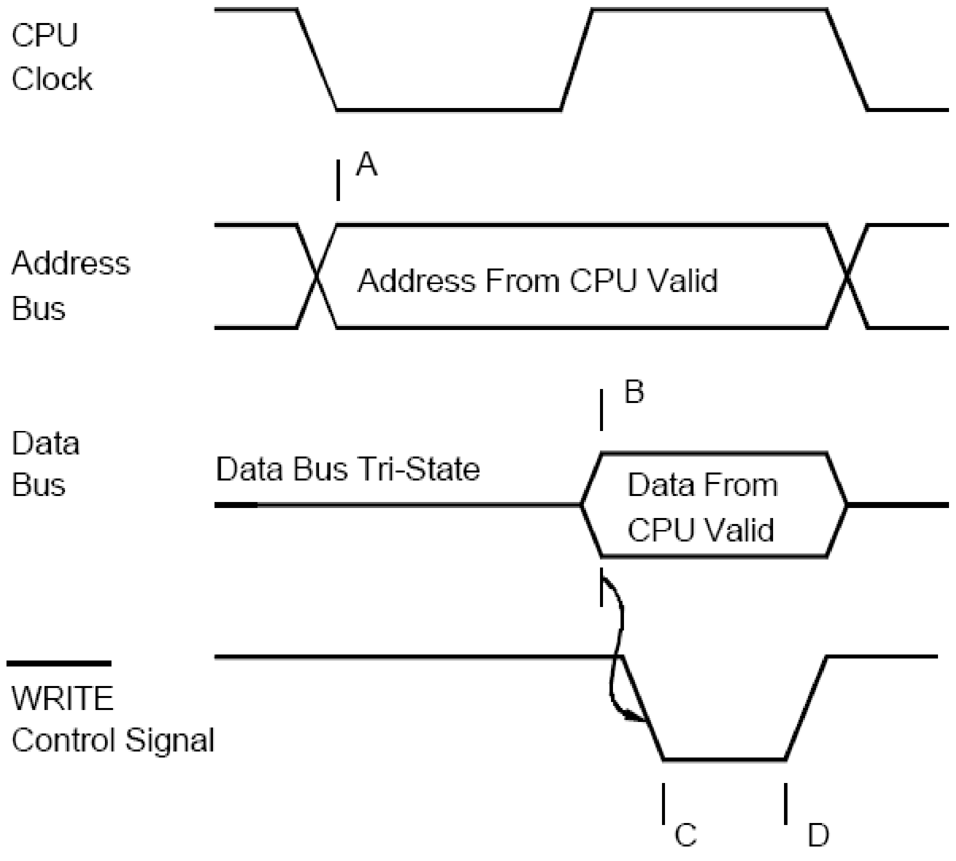
Address Decoding



Timing Signals

- Require timing and synchronization.
- Write Cycle
  - Point A – Place address on address bus
  - Point B – Place data on the data bus
  - Point C – Assert the WRITE signal
  - Point D – Disable the WRITE signal
  - Data can be captured on rising or falling edge of WRITE signal
- Read Cycle
  - Point A – Place address on address bus
  - Point B – Assert the READ signal
  - Point C – Data read to be read by CPU
  - Some synchronization may be required

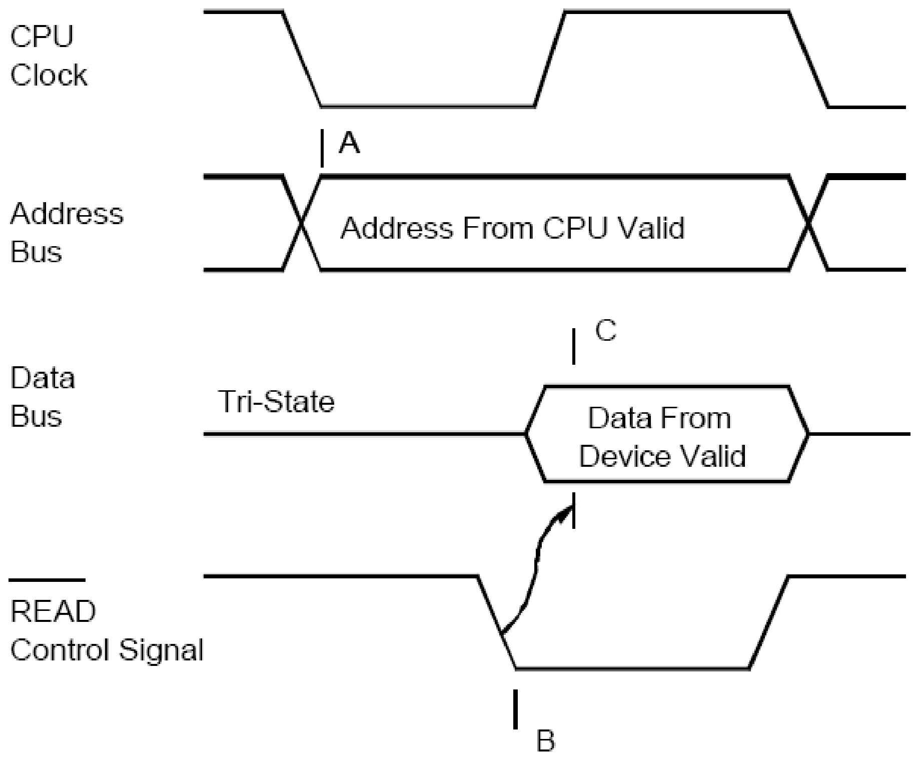
Timing Signals: Typical CPU Write Cycle



- Point A – Place address on address bus
- Point B – Place data on the data bus
- Point C – Assert the WRITE signal
- Point D – Disable the WRITE signal

Data can be captured at destination on rising or falling edge of WRITE signal

Timing Signals: Typical CPU Read Cycle

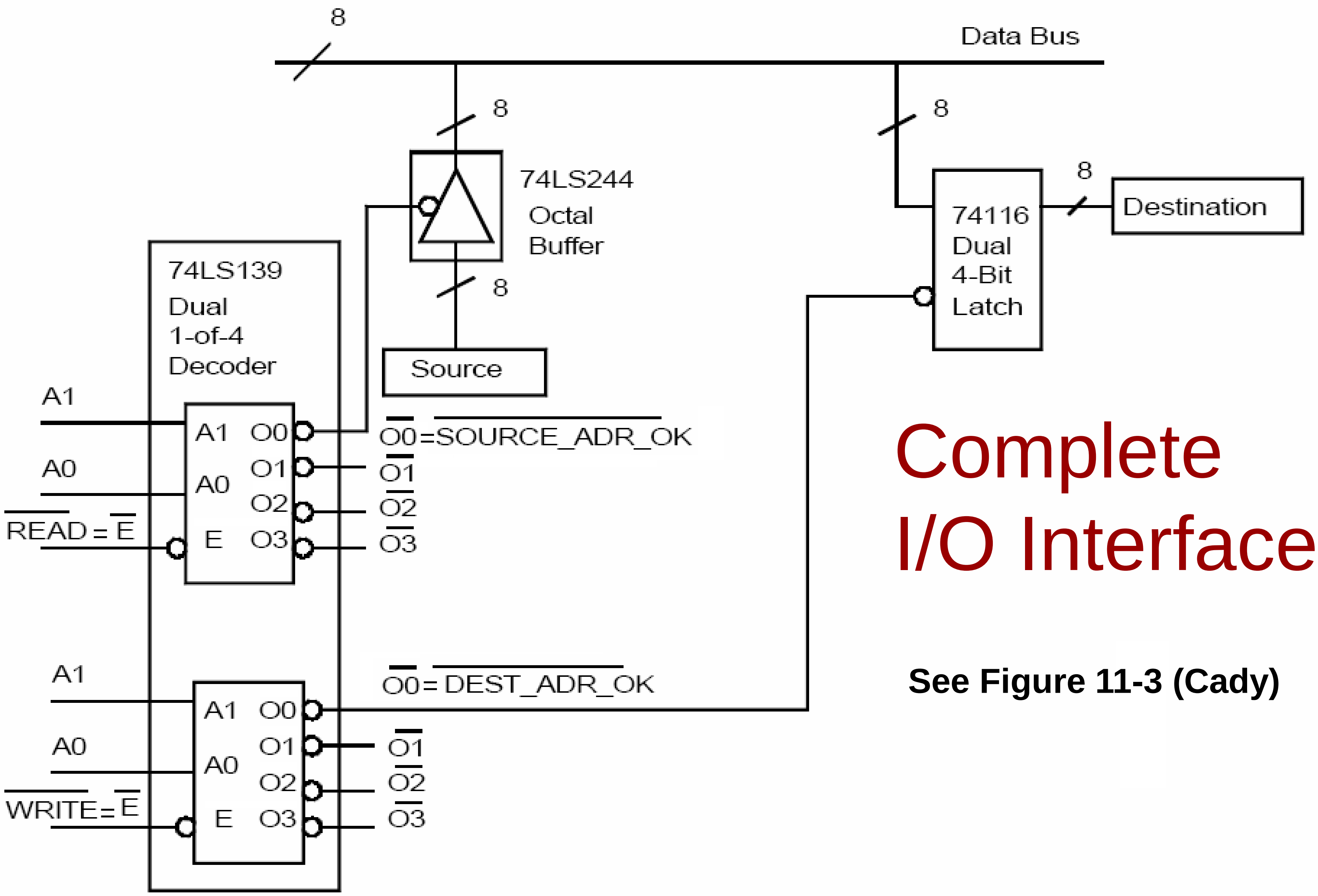


Point A – Place address on address bus

Point C – Data to be read by CPU
Some synchronization may be required

Point B – Assert the READ signal

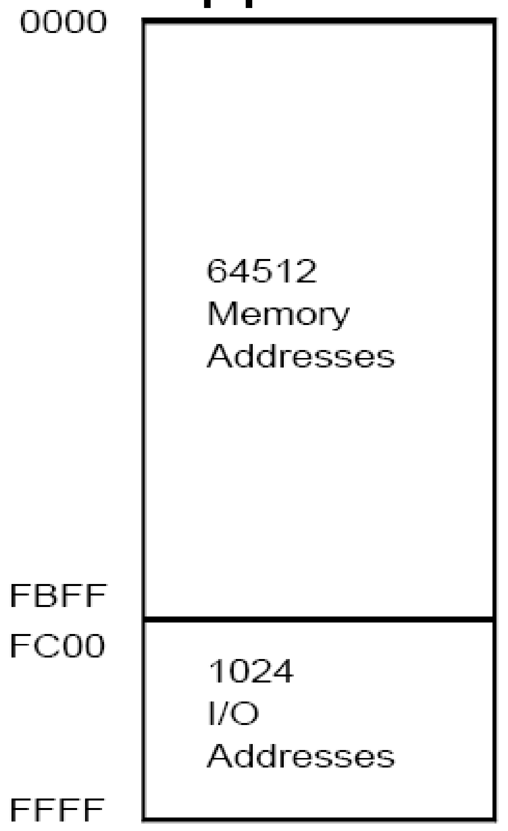
Complete I/O Interface



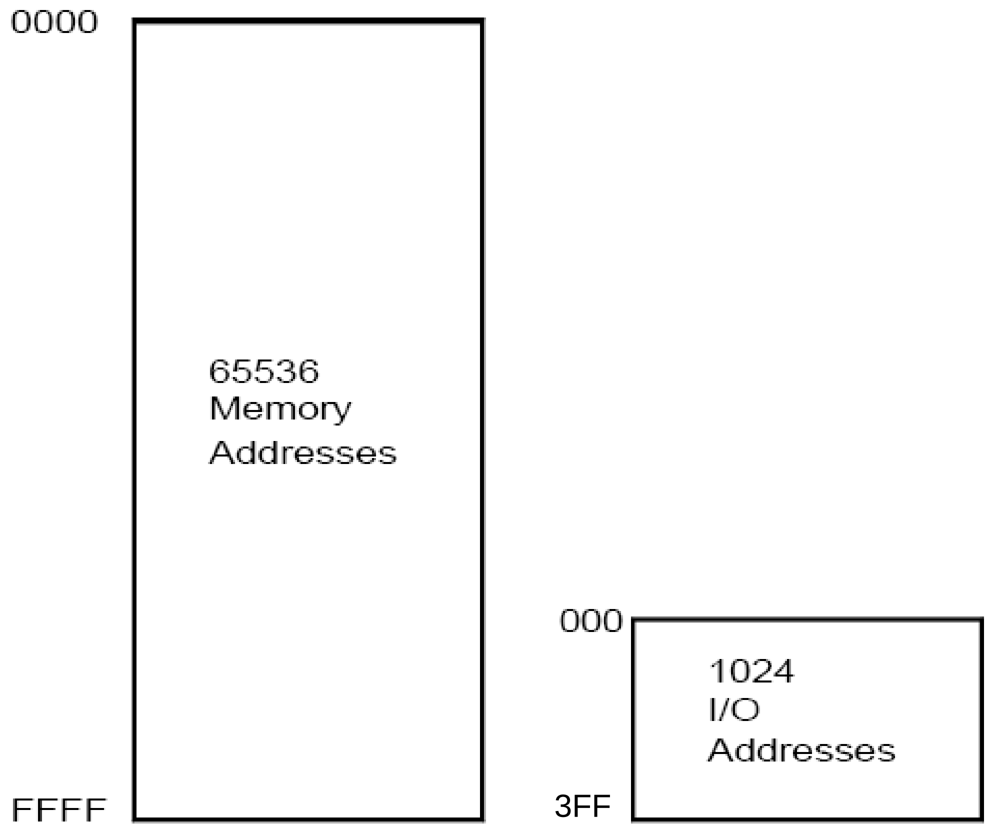
See Figure 11-3 (Cady)

I/O Addressing

Memory-mapped I/O



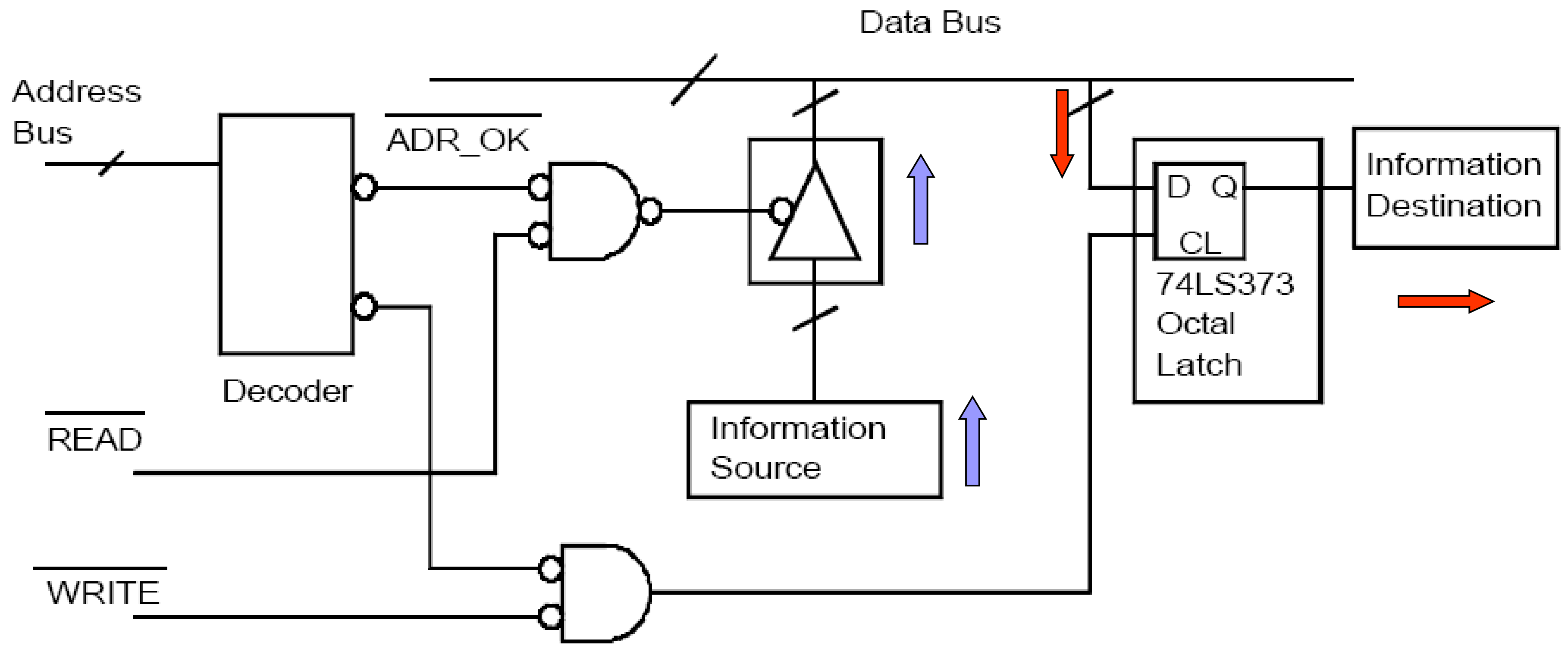
Separate or Isolated I/O



Memory Mapped I/O

- May require full address decoding
  - Allows all instructions to access I/O registers
  - Special I/O instructions not required
- Popular Design
  - With DEC computers (PDP-8)
  - In Motorola microcomputers
- Advantage:
  - Reduces amount of memory for programs
- Disadvantages:
  - Full address decoders required to prevent conflict with memory
  - Requires more complex circuitry in peripheral hardware

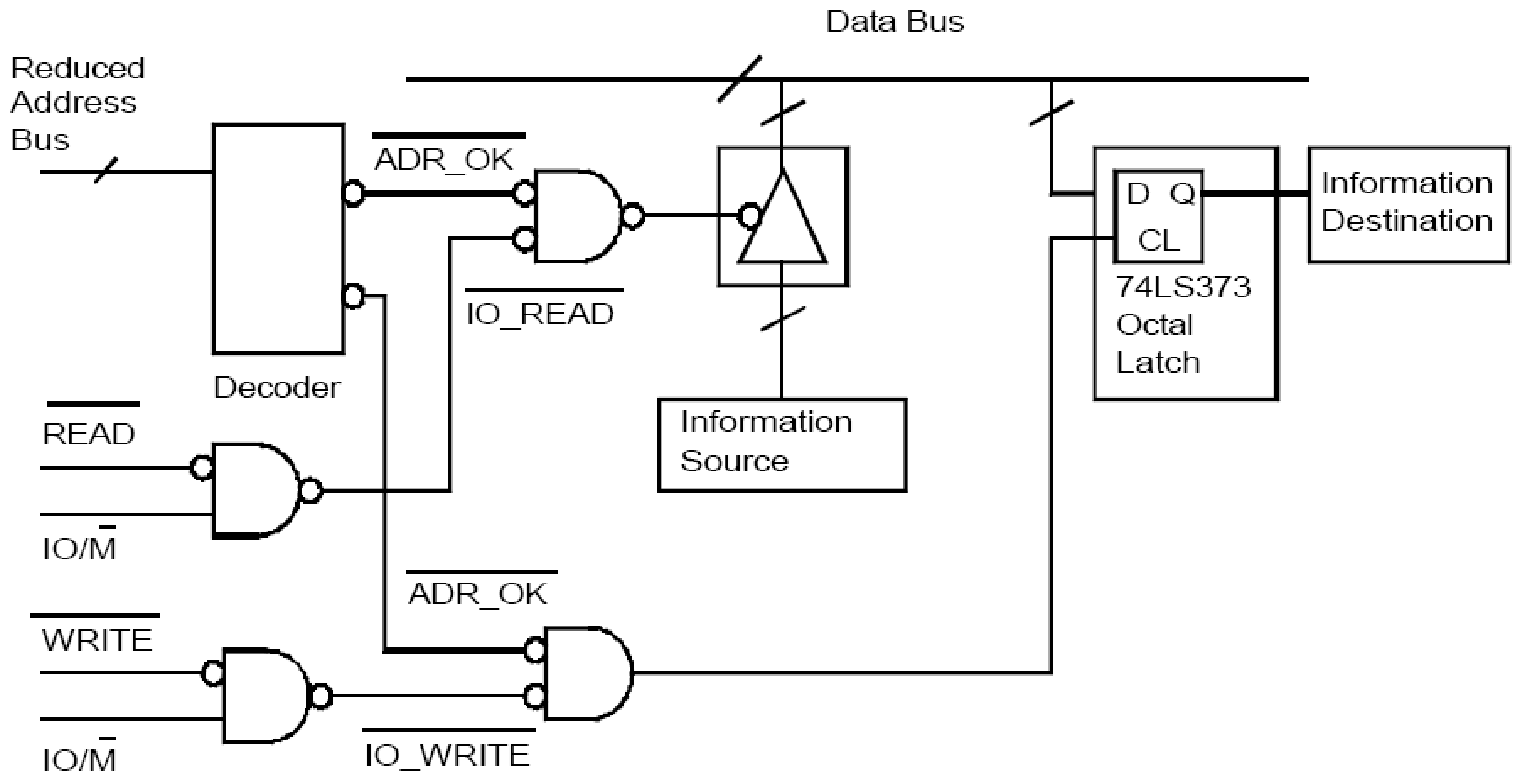
Address decoding for Memory Mapped I/O



Isolated or Separate I/O

- Separate address space for I/O
  - I/O map smaller than memory map
- Requires additional hardware in CPU
  - Uses same address bus
  - But signal indicates if memory or I/O being accessed (IO/$\overline{\text{M}}$)
  - Instruction decoder and sequence controller must process special instructions for I/O (input and output instructions)

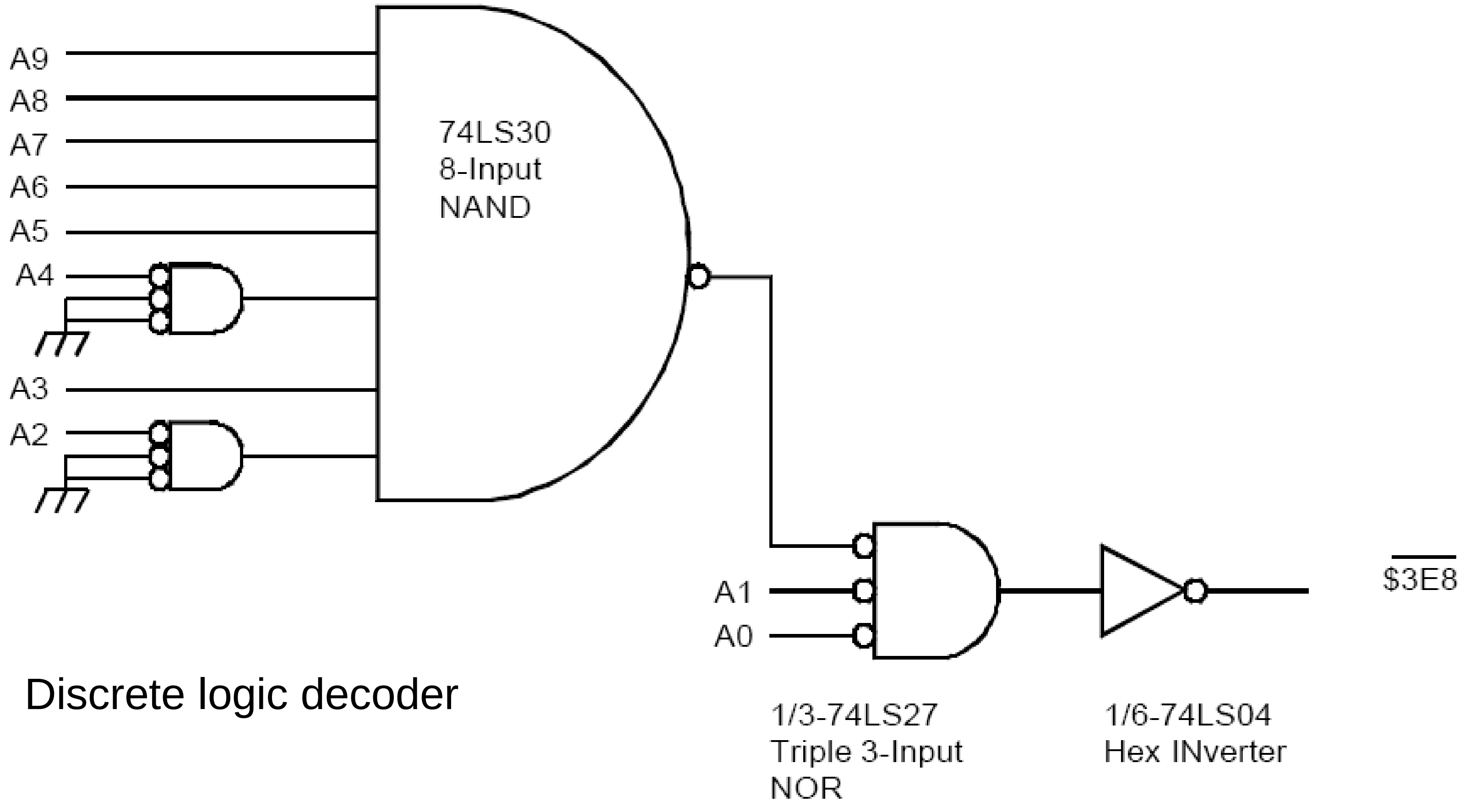
Address Decoding for Separate I/O



Address Decoding

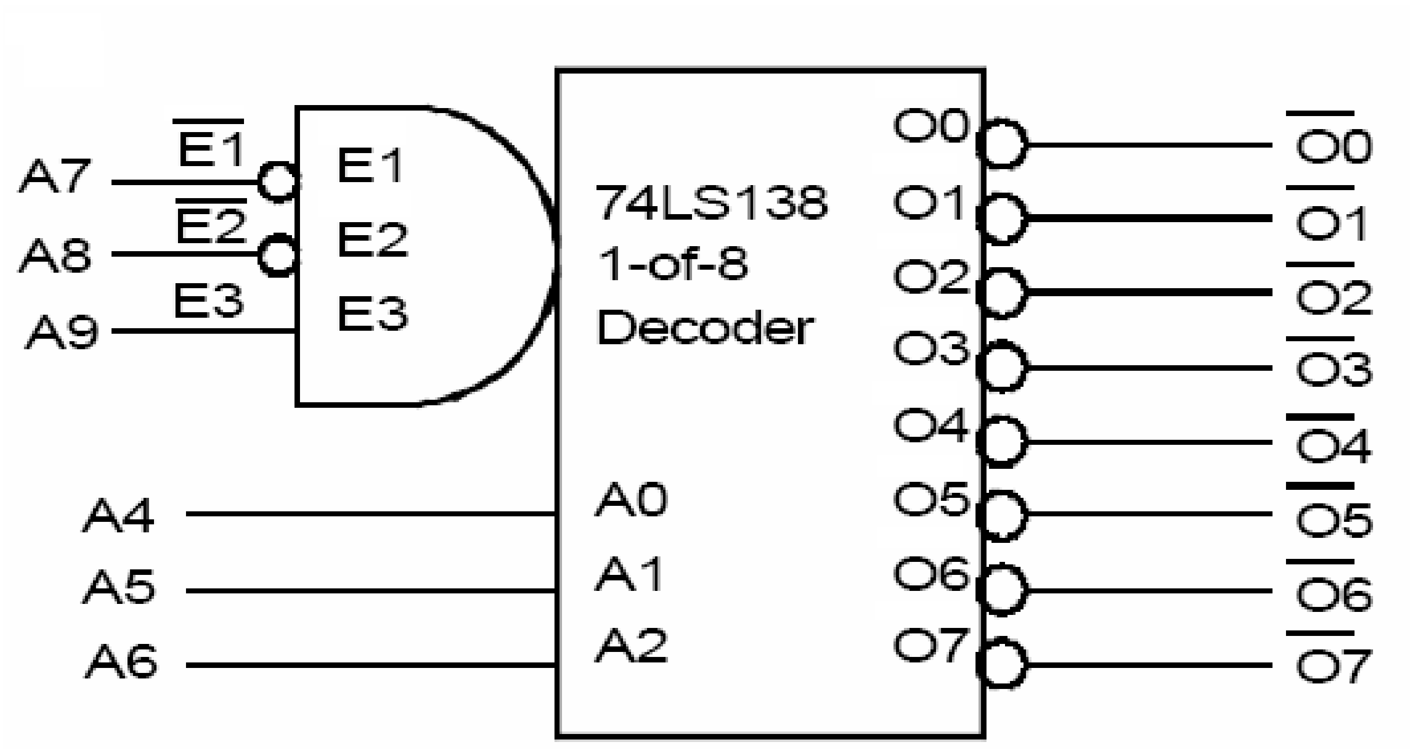
- Full Address Decoding
  - All address bits are decoded
  - Required when many I/O devices are present or with memory mapped I/O
- Partial Address Decoding
  - Only higher-order bits on bus are decoded
  - Most useful with isolated I/O

Full Address Decoding



Discrete logic decoder

Partial Address Decoding



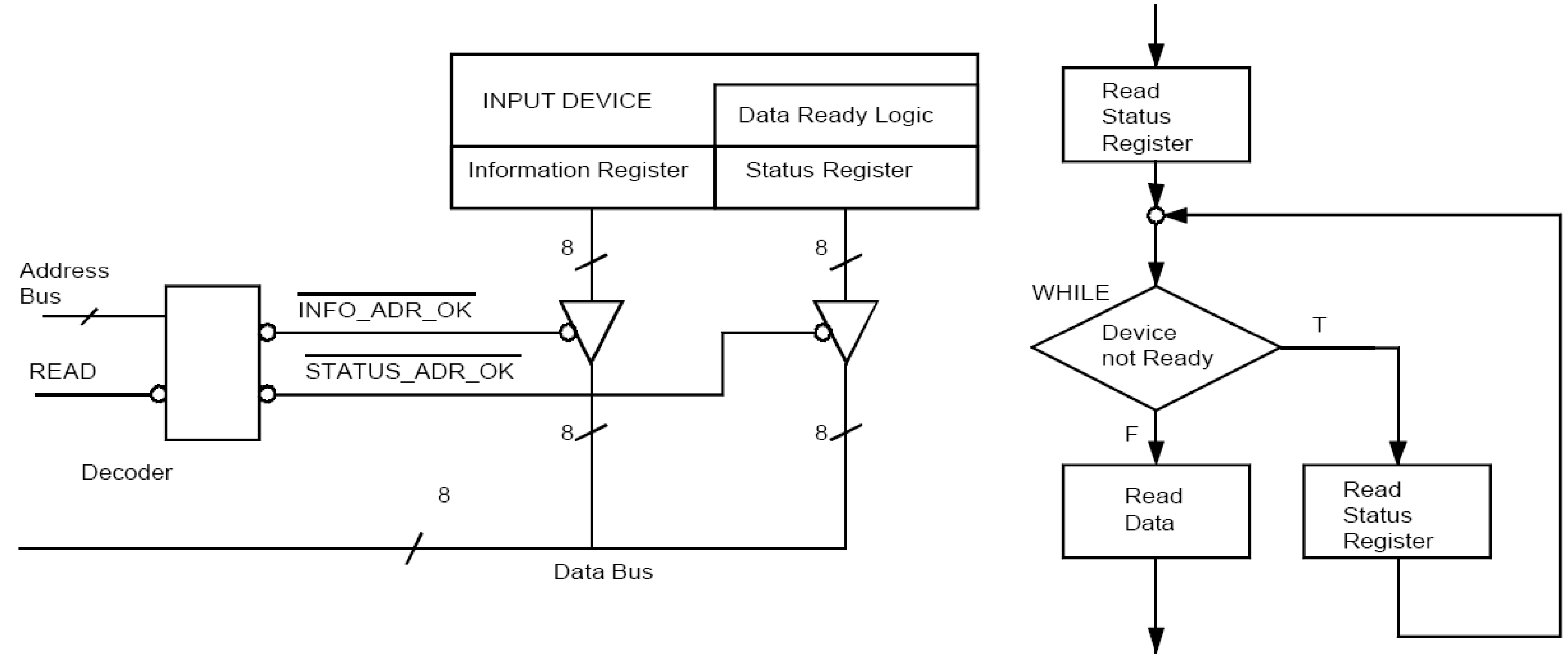
I/O Synchronization

- Three problems to deal with for data transfer between CPU and I/O devices
  - CPU is faster than some I/O devices
  - I/O device transfers data at unpredictable intervals
  - CPU is slower than some other I/O devices

Software I/O Synchronization

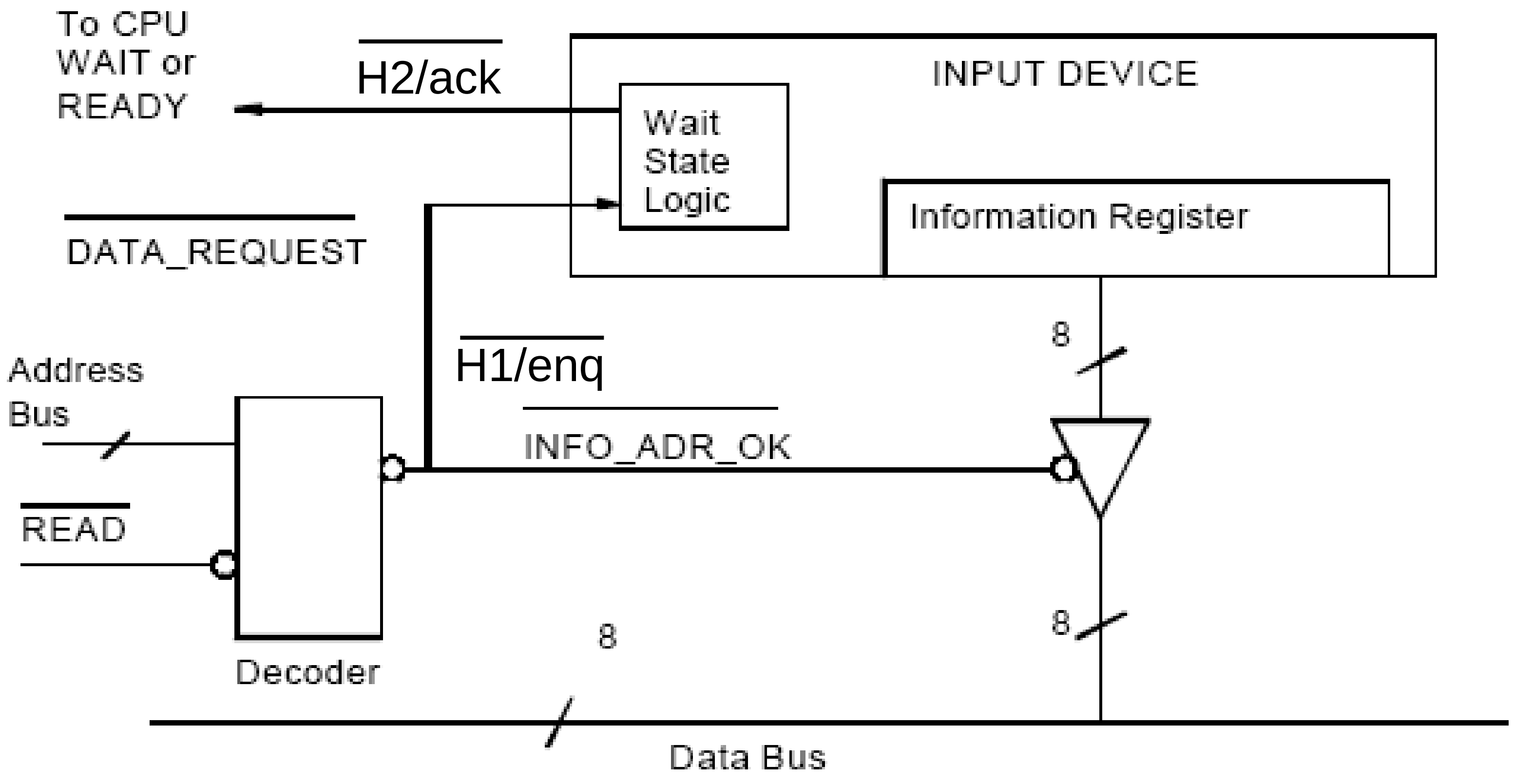
- Hardware provides an interface to the I/O device
  - Contains a status register for control
  - Contains a data register for data transfer
- Polling
  - CPU monitors the status register
  - When data is ready, CPU reads or writes from/to the data register
- Interrupts
  - I/O device can interrupt the CPU when data can be read or written (will study later)

Software Polling

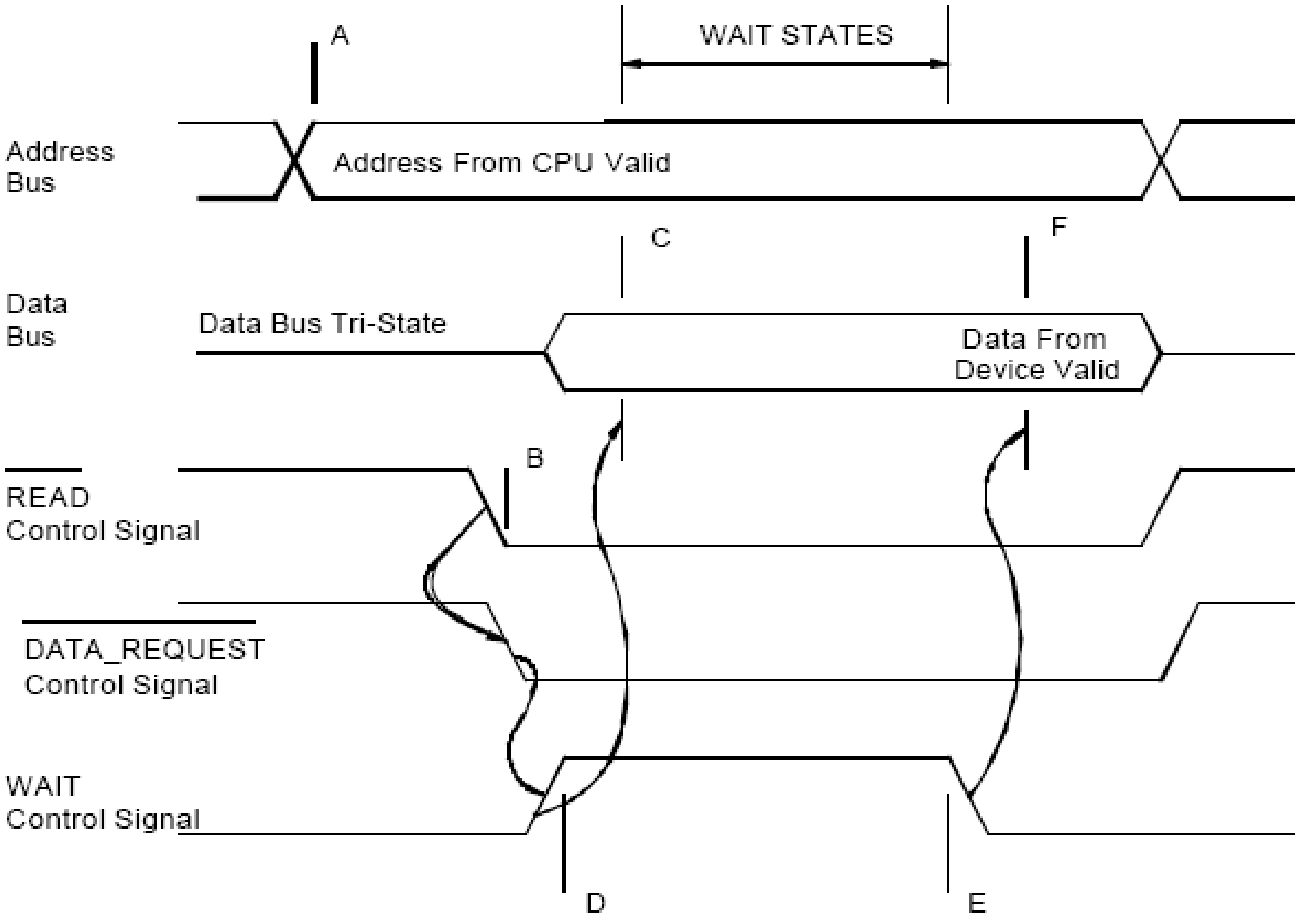


Handshaking I/O

- Hardware method to place CPU in Wait state
  - READY or WAIT – asserted by external device until data is ready


To CPU
WAIT or
READY
$\overline{\text{H2/ack}}$
INPUT DEVICE
Wait
State
Logic
Information Register
$\overline{\text{DATA\_REQUEST}}$
$\overline{\text{H1/enq}}$
8
Address
Bus
$\overline{\text{INFO\_ADR\_OK}}$
$\overline{\text{READ}}$
8
8
Decoder
Data Bus

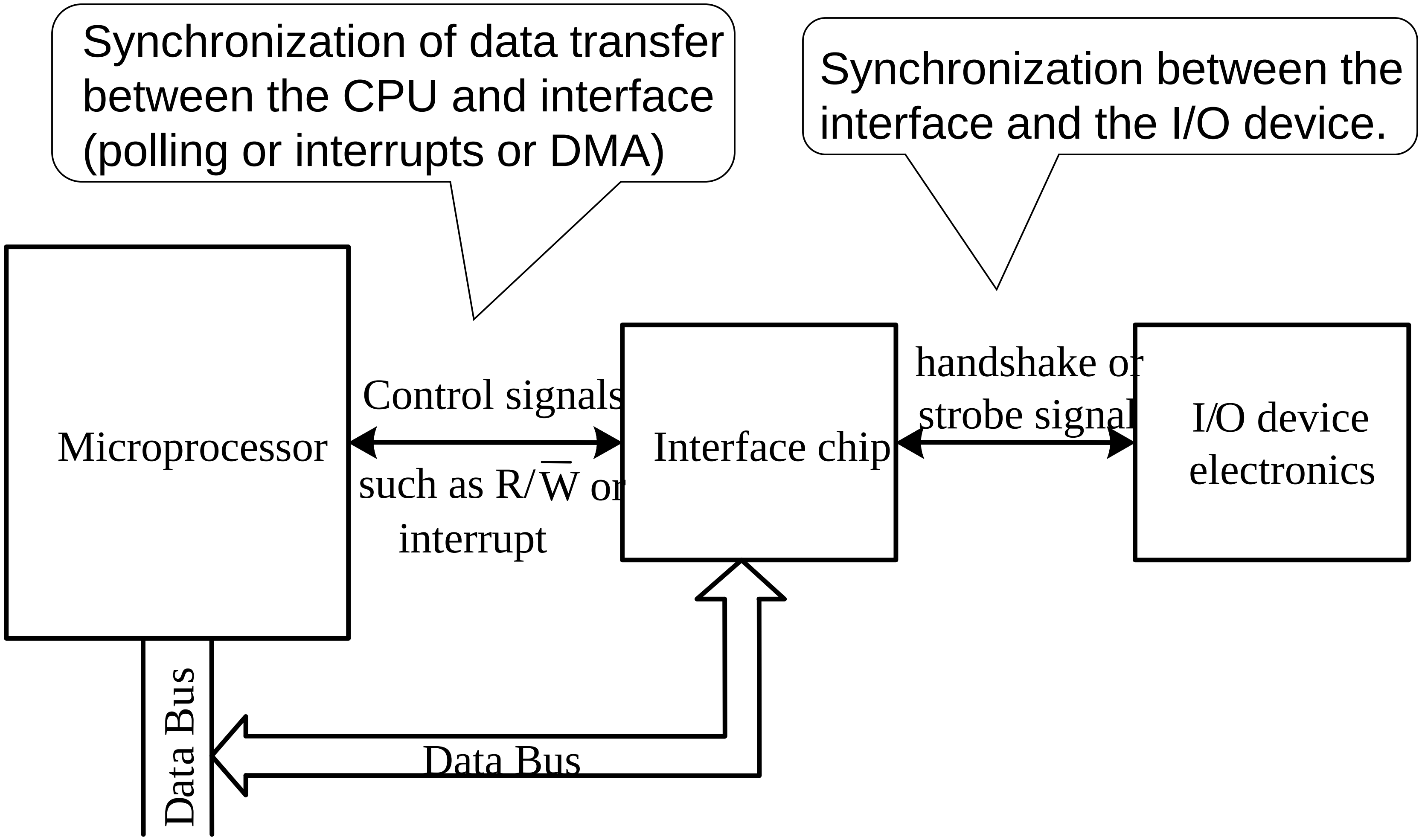
Reading using Handshaking I/O



Software Synchronization versus Handshaking

- Handshaking I/O: Advantages and Disadvantages
  - Potentially faster
  - But ties up the CPU in wait states
- Software I/O: Advantages and Disadvantages
  - CPU can do other things when polling
  - CPU design simpler (no need for wait states or WAIT signal)
  - Logic in I/O devices more complex (see the interface chip)

The Interface Chip


Synchronization of data transfer between the CPU and interface (polling or interrupts or DMA)
Synchronization between the interface and the I/O device.
Microprocessor
Control signals
such as R/$\overline{W}$ or interrupt
Interface chip
handshake or strobe signal
I/O device electronics
Data Bus
Data Bus

The role of an interface chip

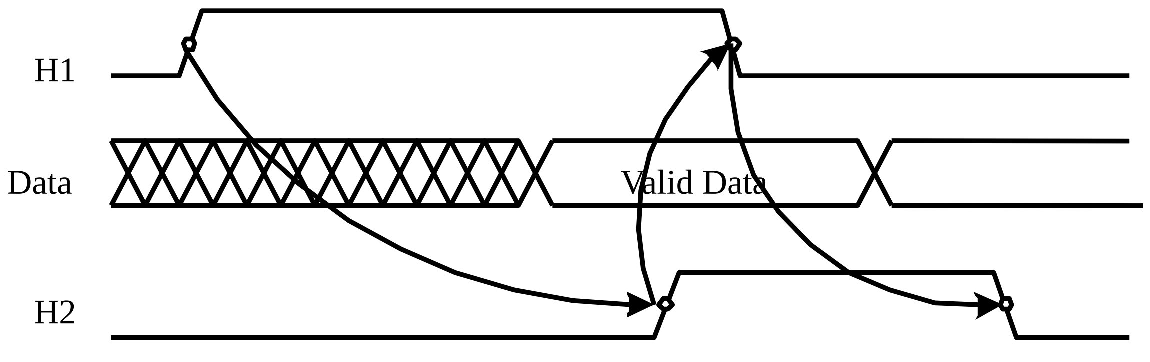
Synchronization Between Interface Chip and I/O Device

- **Brute force** method – synchronization is not critical
  - For input – the interface chip reads levels on input pins and makes them available to the CPU in a data register.
  - For output – the chip places levels on the output pins according to the values in the data register updated by the CPU.
- **Strobe method** – indicates data values on bus are stable.
  - For input: interface chip updates the data register with pins levels when strobe signal is asserted.
  - For output: the interface chip places the data on output pins and then asserts a strobe signal. Device must capture data with the assertion of the strobe signal

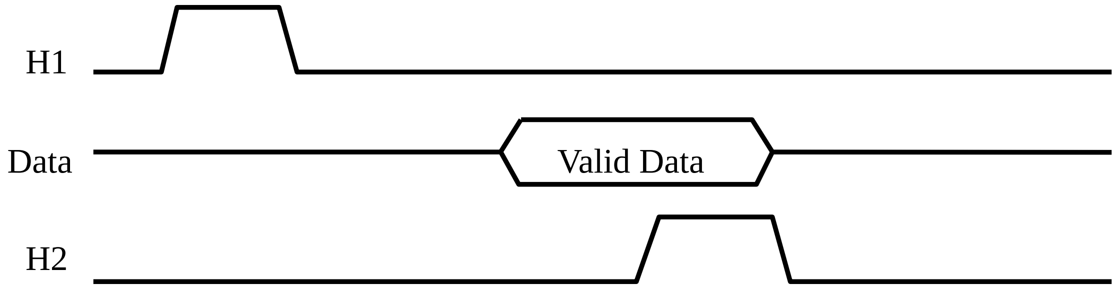
Synchronization between Interface Chip and I/O Device

- **Hardware Handshaking** – Used when timing is critical
  - Two handshake signals are used to transfer the data
  - One signal, say H1, is activated by the interface chip while the second, say H2, is activated by the I/O device.
  - Two handshake modes are available
    - Interlocked handshake mode
    - Pulse-mode handshake

Input Handshake Protocol



(a) Interlocked



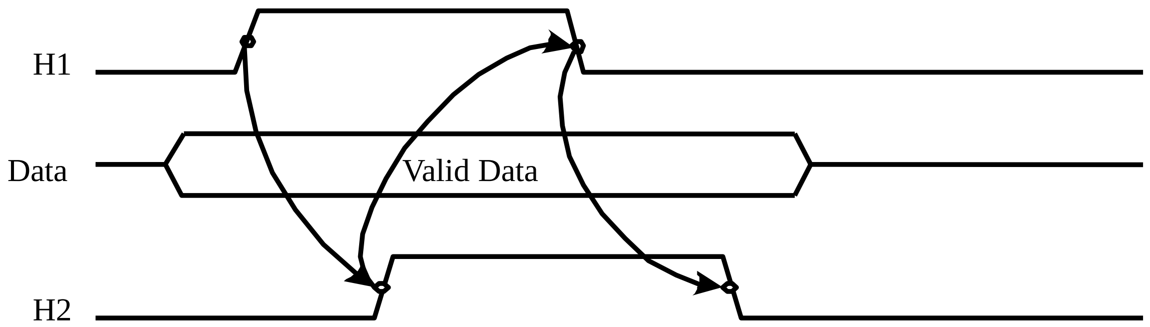
(b) Pulse mode

Step 1. The interface chip asserts (possibly with a pulse) H1 to indicate its intention to input

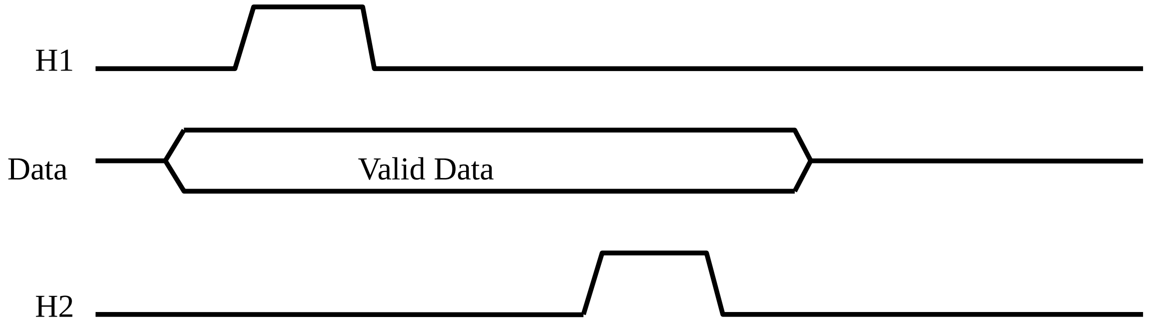
Step 2. The device places the data on the data pins and asserts (possibly with a pulse) H2

Step 3. The interface chip latches the data and de-activates H1. H2 is then deactivated after some delay.

Output Handshake Protocol



(a) Interlocked



(b) Pulse Mode

Step 1. The interface chip places the data on the output pins and asserts (possibly with a pulse) H1 to indicate that it has valid data to be output.

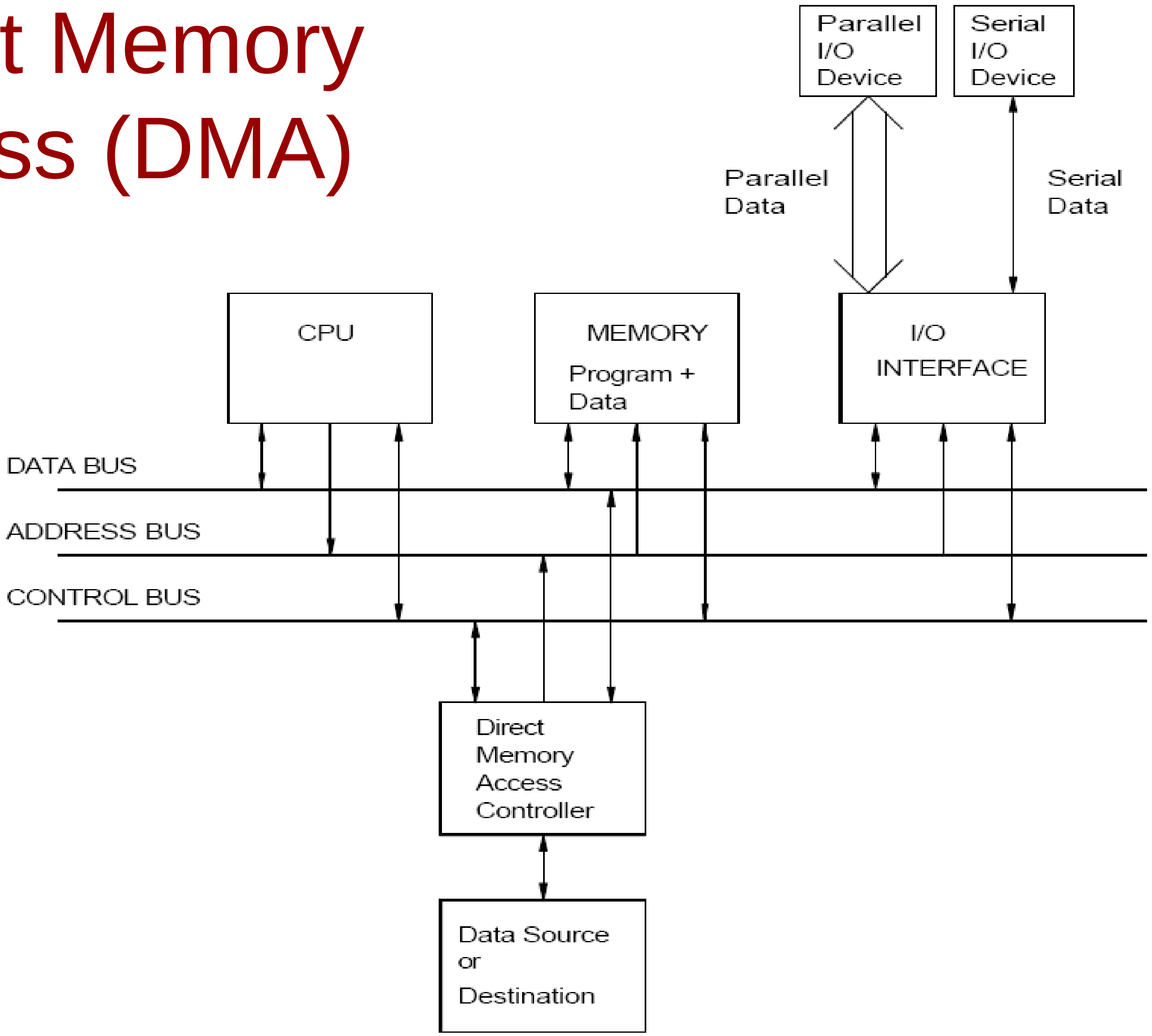
Step 2. The output device latches the data and activates (possibly with a pulse) H2 to acknowledge the reception of the data.

Step 3. The interface chip deactivates H1 after the assertion of H2. The device will then deactivate H2.

More Bus Ideas

- Multiplexed Bus
  - Using the same pins for different functions, for example multiplexing 16 bit address on 8 bit bus
- Bidirectional bus transceiver
  - External chip to drive more devices on data bus
- Bus Masters and Slaves
  - So far, CPU has been the only master of a BUS
  - Other devices can be masters – for example the DMA
  - Signals for bus arbitration are required

Direct Memory Access (DMA)

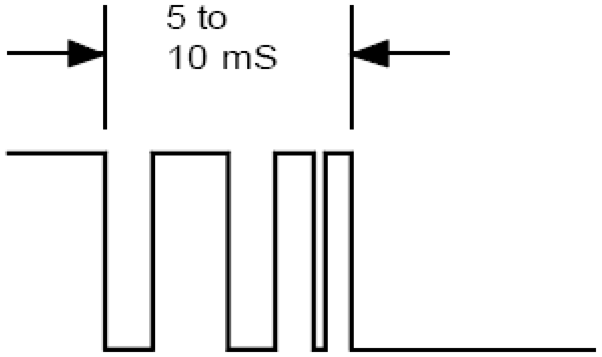


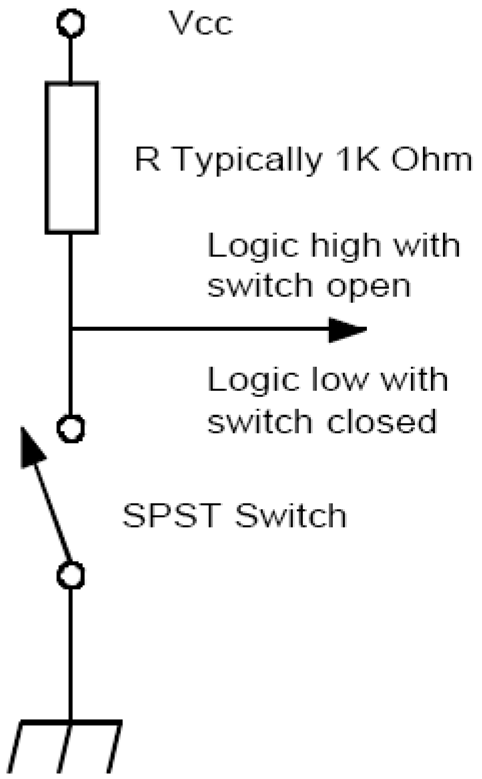
Simple I/O Devices

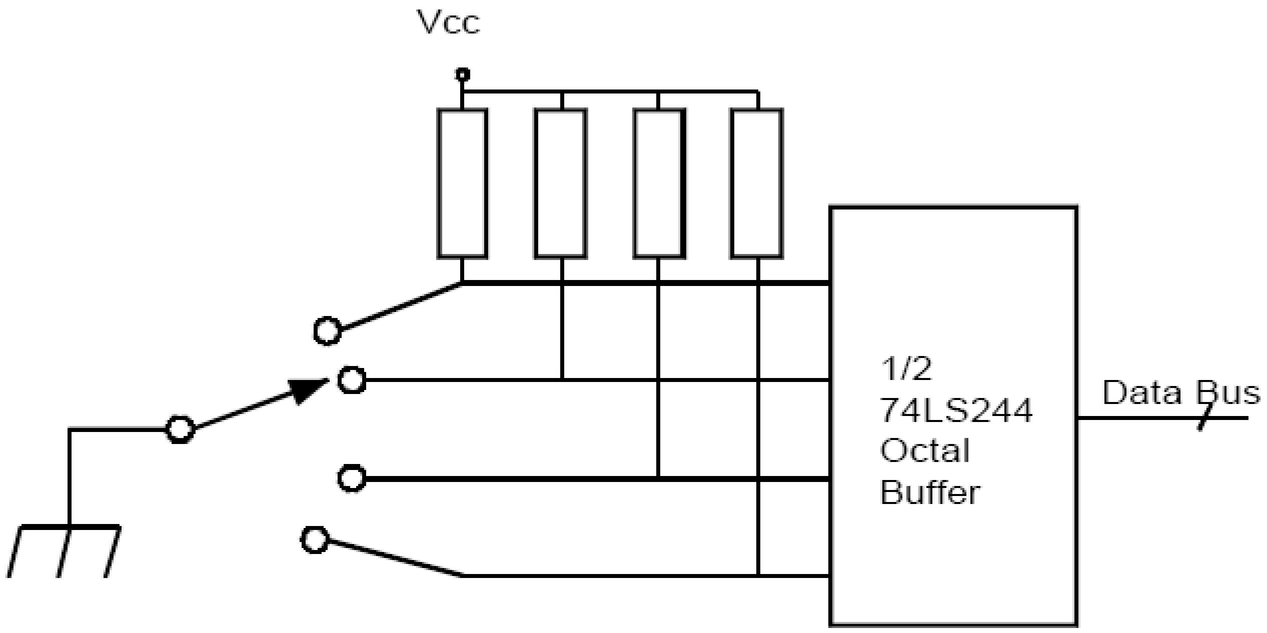
- Switches
- Keypads
- LEDs
- Displays
- Programmable I/O Devices

Switches

- Pullup resistors used to set output high (switch open) or low (switch closed)
- Switches have a bounce problem



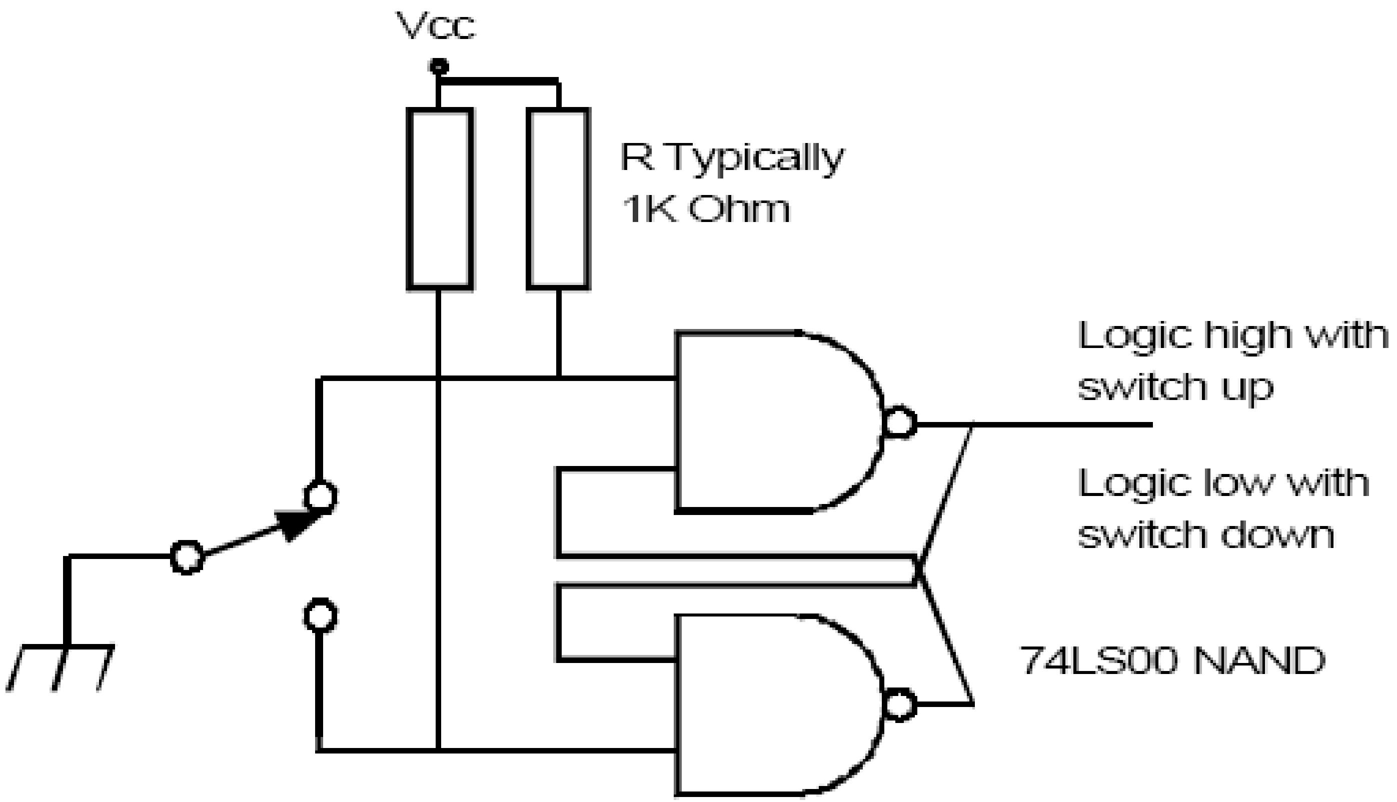




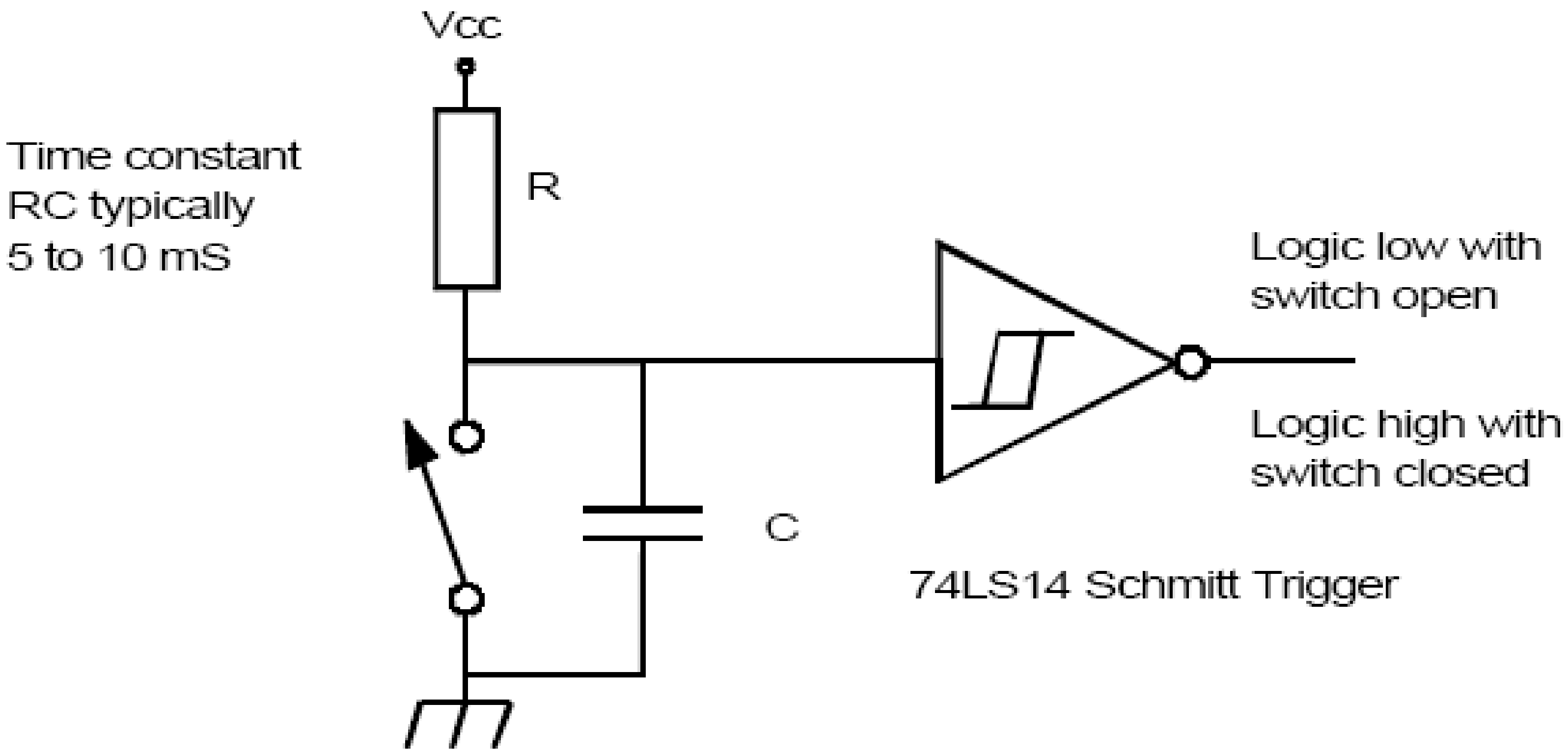
Software Debouncing

- Bouncing will last 5-10 milliseconds
- Debouncing – method 1
  - When a change is detected
    - Wait 10 milliseconds, and test to confirm change
- Debouncing – method 2
  - When a change is detected (say 0), set a counter to 10
  - Every millisecond – test input
    - If change still present (0) – decrement counter
    - If change not present (1) – increment counter
  - If counter reaches 0, then change is confirmed (0)
  - If counter reaches 20, then change is not confirmed (1)

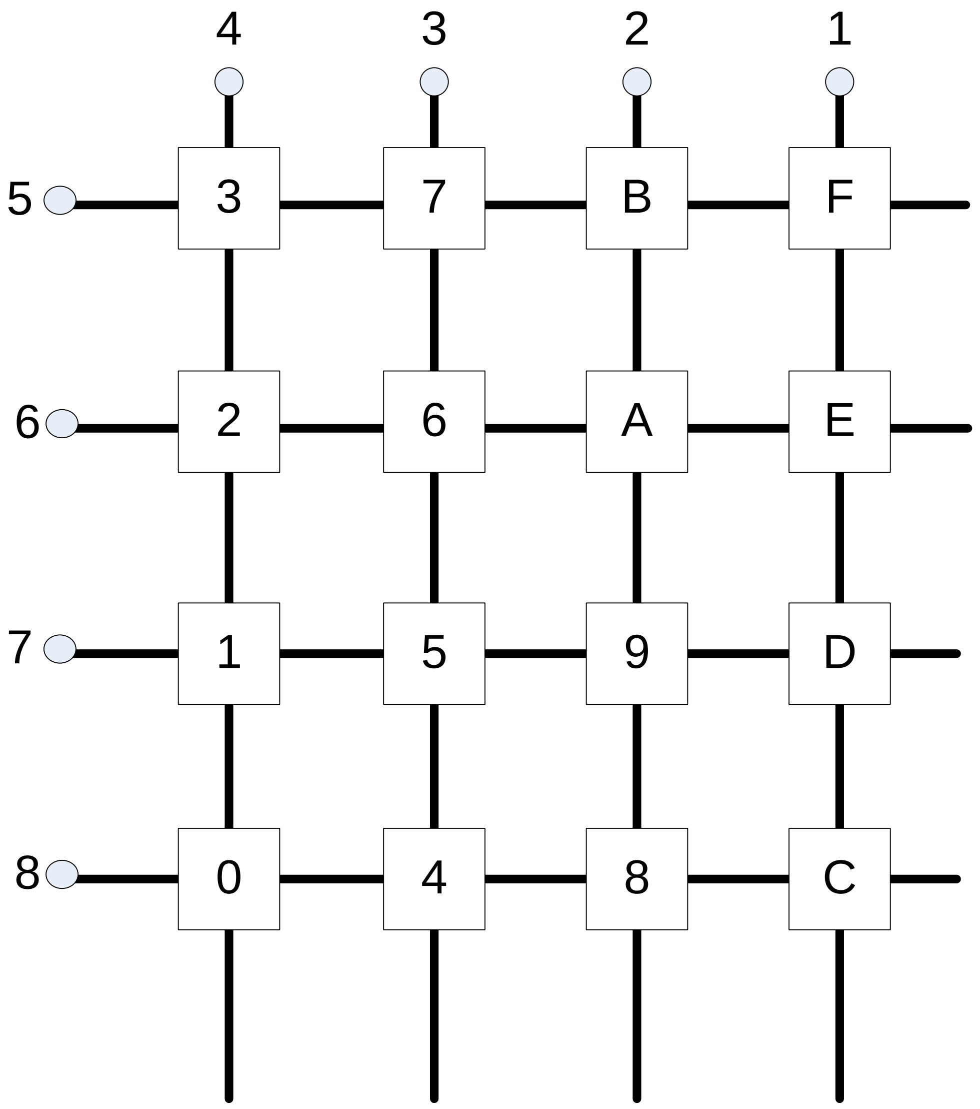
Hardware Debouncing 1



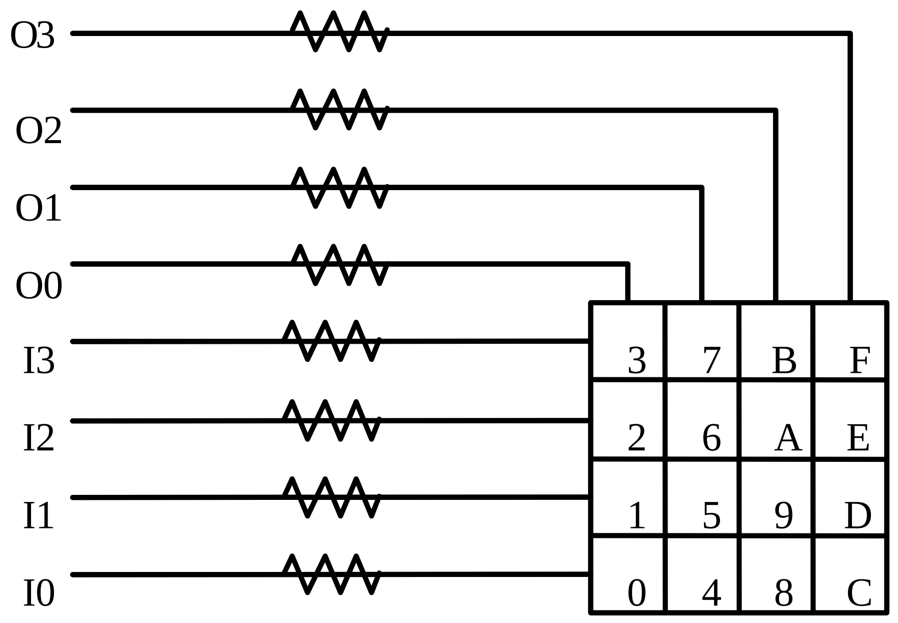
Hardware Debouncing 2



Matrix of switches – key pads



Interfacing to Keypads

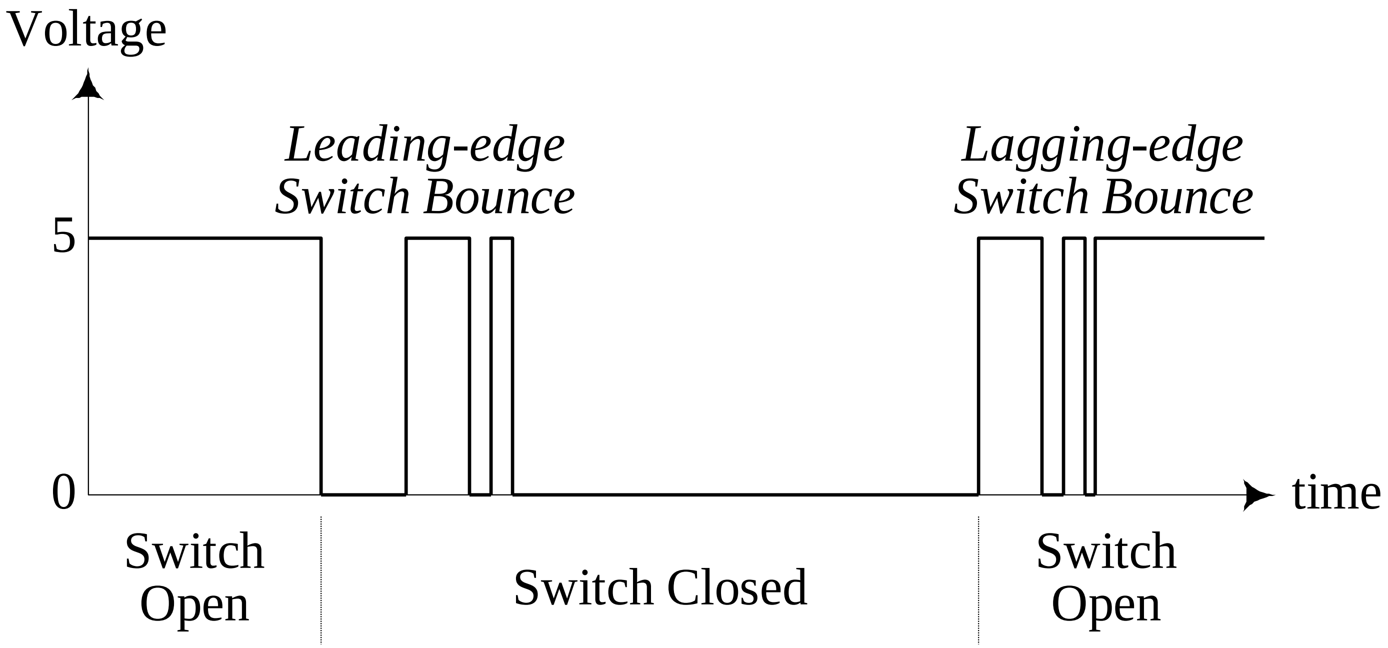


O3	O2	O1	O0	Key Hit
1	1	1	0	0, 1, 2, and 3
1	1	0	1	4, 5, 6, and 7
1	0	1	1	8, 9, A, and B
0	1	1	1	C, D, E, and F

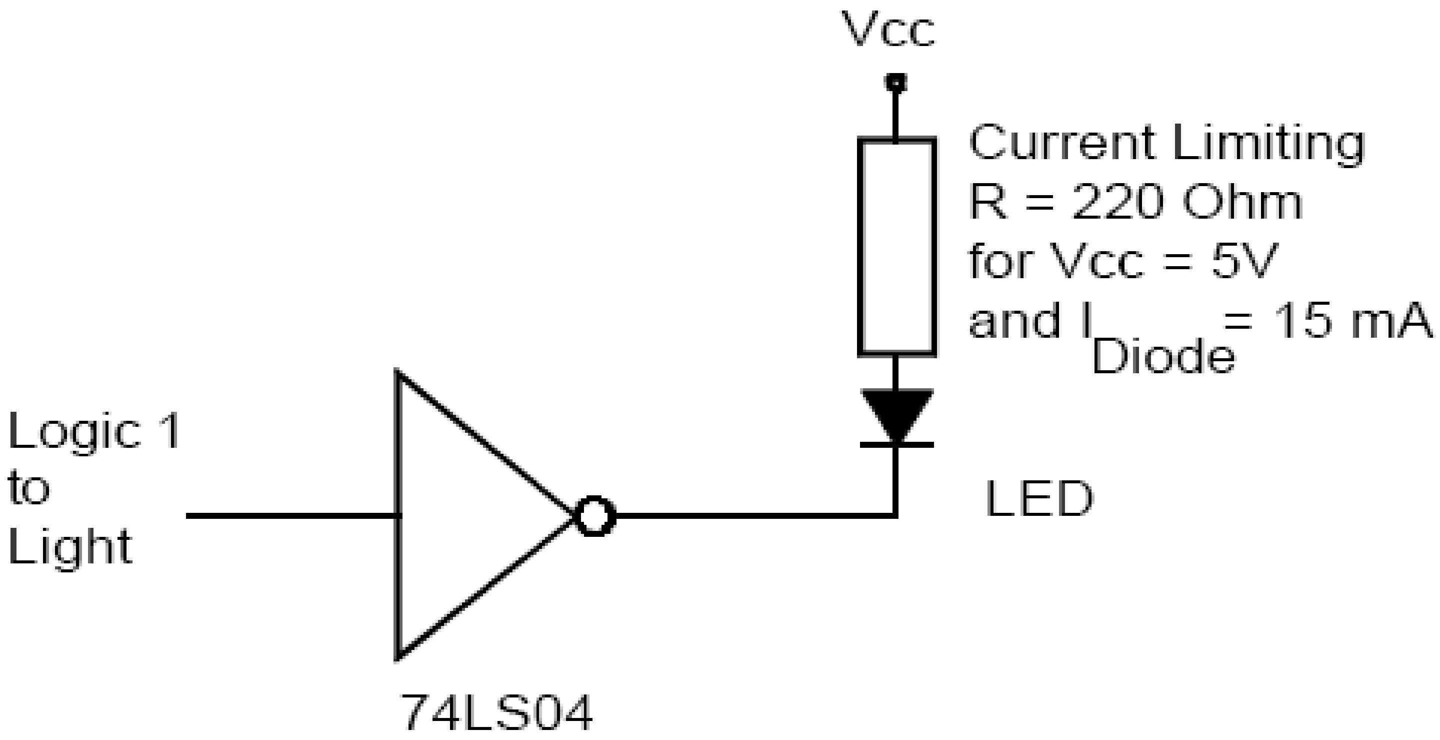
Key	O3	O2	O1	O0	I3	I2	I1	I0
0	1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0	1
2	1	1	1	0	1	0	1	1
3	1	1	1	0	0	1	1	1
4	1	1	0	1	1	1	1	0
5	1	1	0	1	1	1	0	1
6	1	1	0	1	1	0	1	1
7	1	1	0	1	0	1	1	1
8	1	0	1	1	1	1	1	0
9	1	0	1	1	1	1	0	1
A	1	0	1	1	1	0	1	1
B	1	0	1	1	0	1	1	1
C	0	1	1	1	1	1	1	0
D	0	1	1	1	1	1	0	1
E	0	1	1	1	1	0	1	1
F	0	1	1	1	0	1	1	1

Debouncing with Keypads

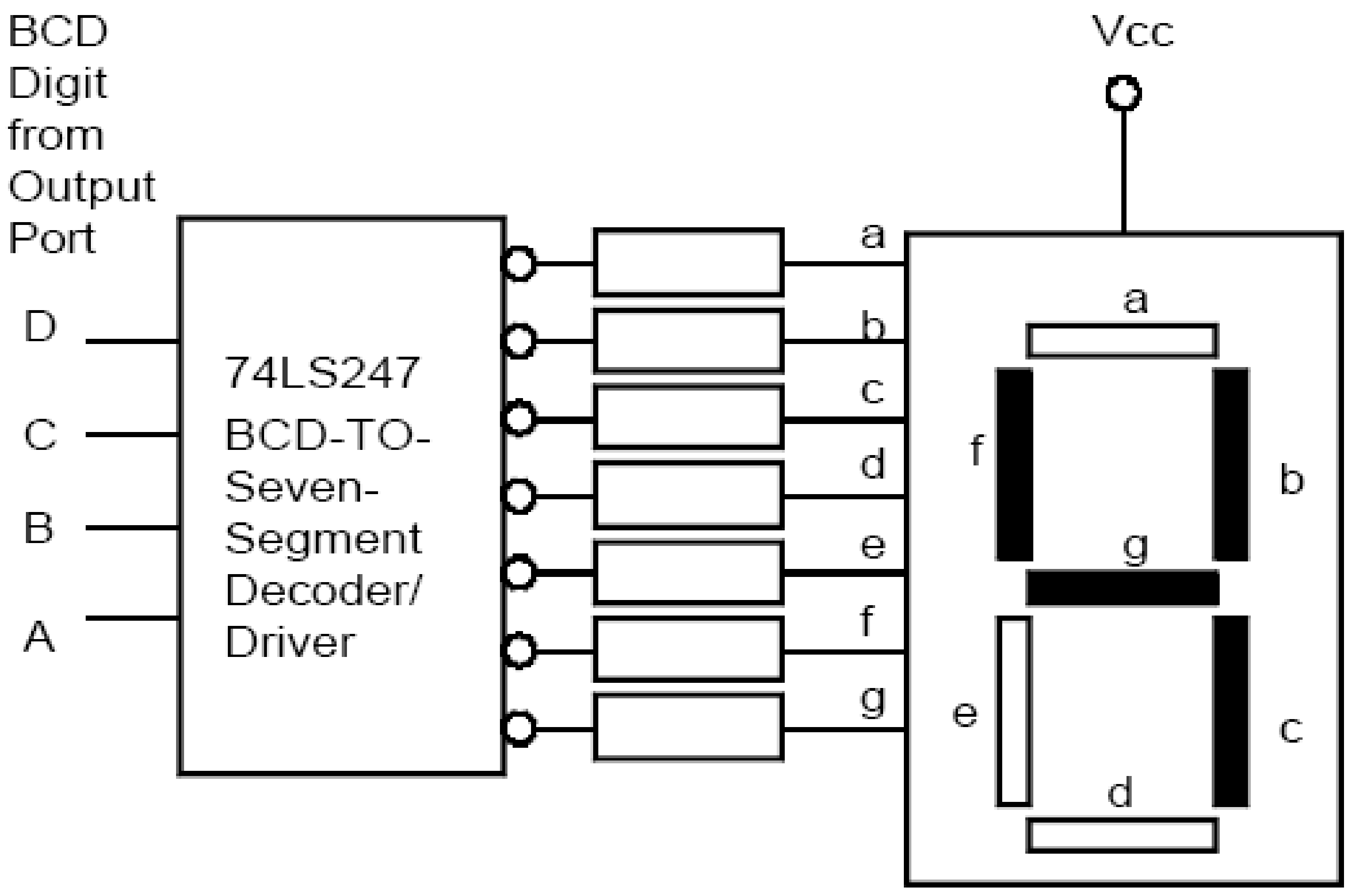
- Key press involves two switch bounces
  - Debounce leading edge
  - Time out for ignoring lagging-edge



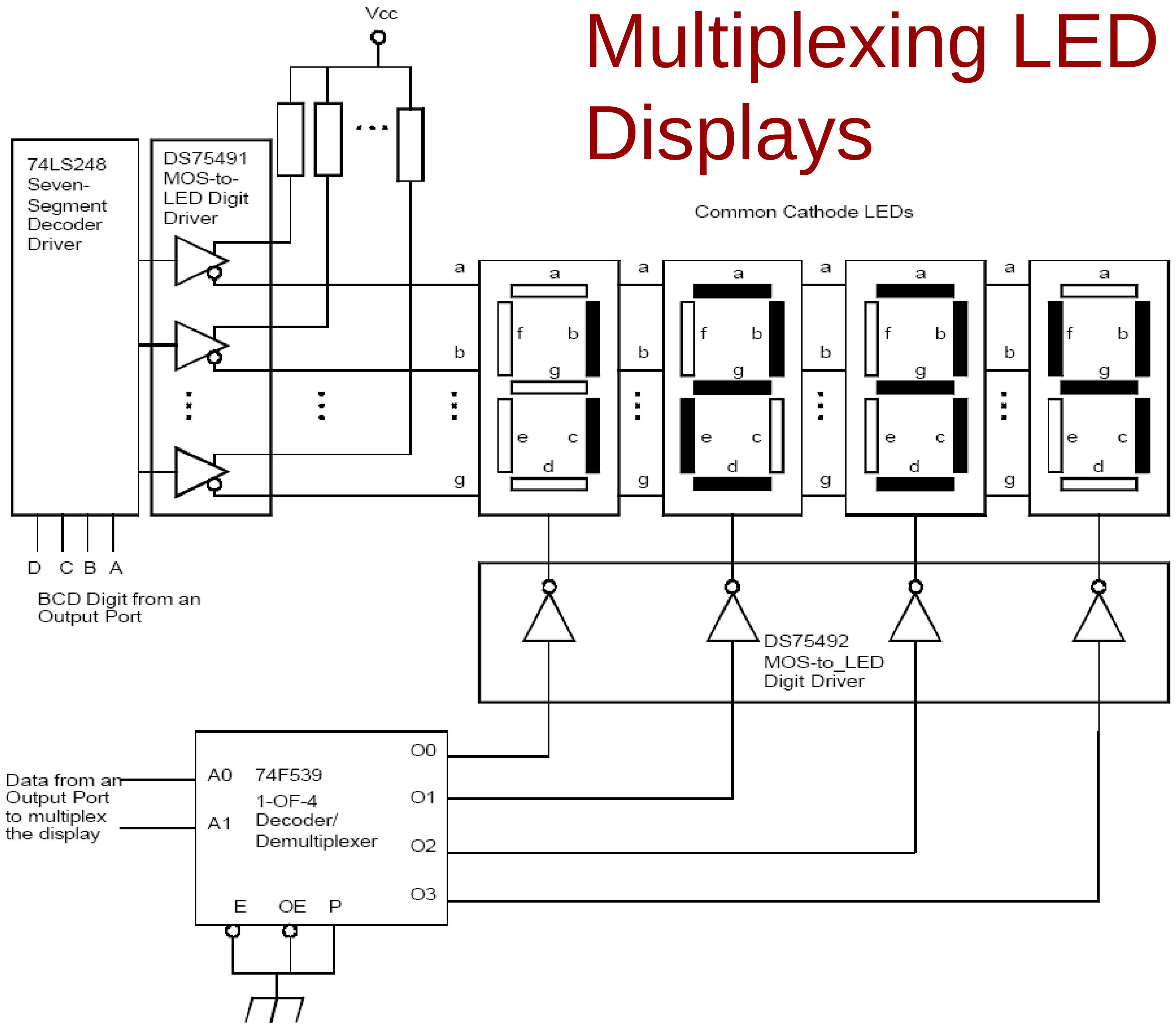
Driving LEDs



7-Segment Display



Multiplexing LED Displays



Vcc
74LS248 Seven-Segment Decoder Driver
DS75491 MOS-to-LED Digit Driver
Common Cathode LEDs
a
b
g
f
c
d
e
D C B A
BCD Digit from an Output Port
DS75492 MOS-to_LED Digit Driver
Data from an Output Port to multiplex the display
A0
A1
74F539 1-OF-4 Decoder/ Demultiplexer
O0
O1
O2
O3
E
OE
P

Programmable I/O Devices

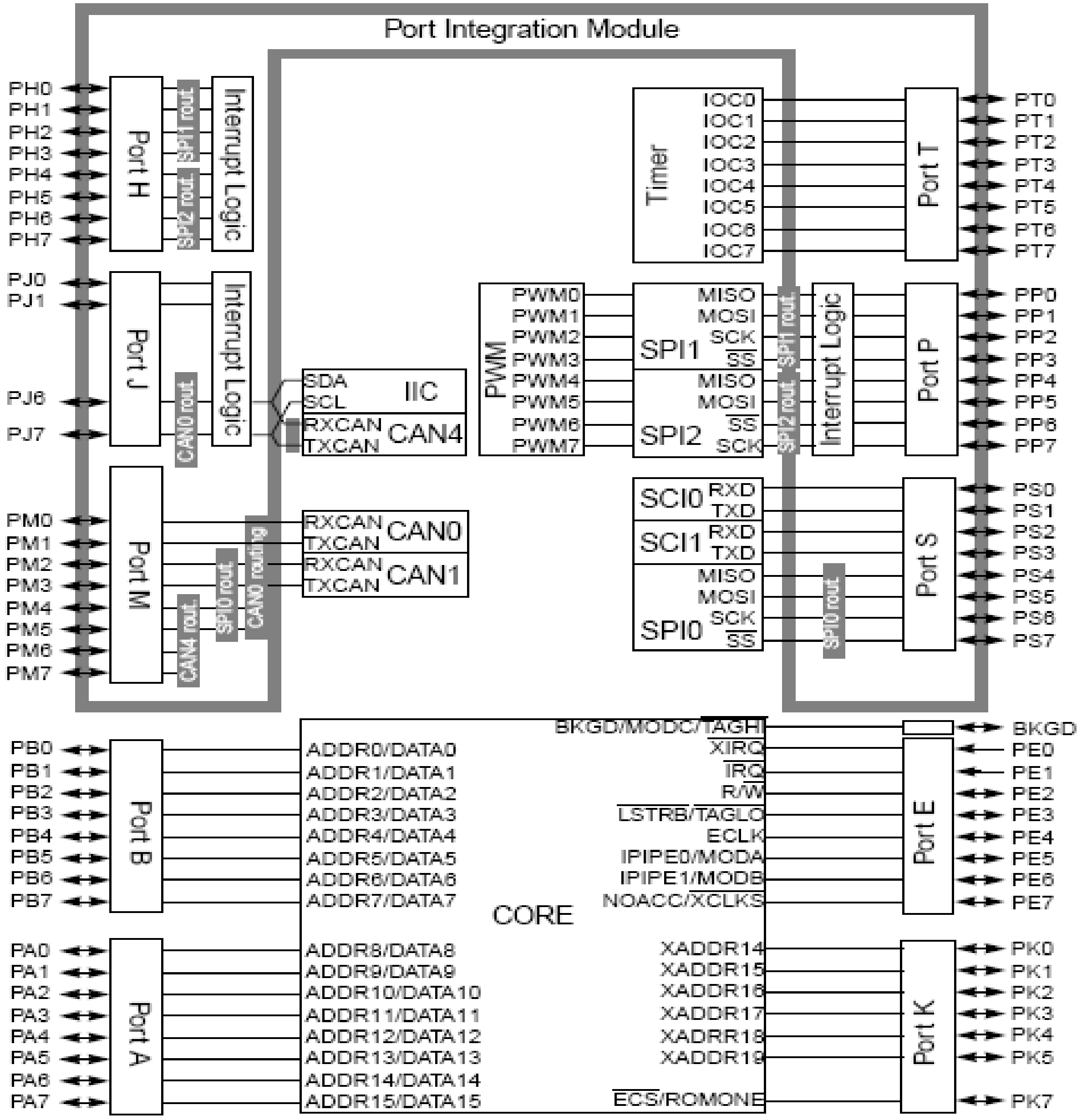
- Introduced by Motorola in 1974
- Interface chips dedicated to I/O tasks
  - Parallel ports
  - Serial ports
- Example – Motorola 6821 Peripheral Interface Adaptor
  - Control Register
  - Data Direction Register
  - Data Register

M68HC9S12 Parallel I/O

- MC68HC9S12
  - Many ports allow parallel I/O
  - For example, Ports A and B can be used for creating an external address/data bus or as I/O ports
- Operating modes
  - Single chip mode
  - Normal expanded mode
    - Allows addressing of external memory

Po
M

- R
  bo
  m
  m
  - [



Source: MC9S12DT256 Port Integration Module (PIM) Block Guide, V03.04

Operating Modes

- Normal Single-Chip Mode
  - Ports A, B, E, and K are used as parallel ports (but 2 bits in port E are reserved for interrupt signal input).
- Normal Expanded Mode
  - Ports A and B are used as multiplexed bus for addresses and data.
  - Port E provides control signals
  - Port K provides paging signals and essentially extends the address bus.

External address and data bus

- Narrow mode
  - Only 8 bit data bus is used
  - Data bus multiplexed on Port A with address
- Wide mode
  - 16 bit data bus
  - Multiplexed with address on both Ports A and B

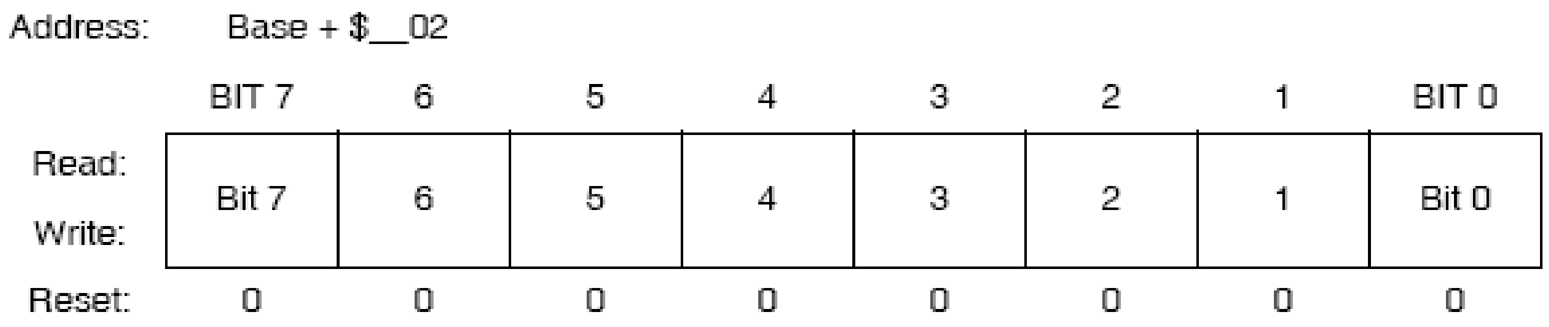
Overview of the 9S12DG256 Parallel Ports

- Different members of the 68HCS12 family contains from 80 to 112 pins
- Pins can be used for different functions
- The MC9S12DG256 offers a number of parallel ports in single chip mode: A, B, E, and K
  - Other ports can also be used for parallel I/O (e.g. Port P)

Ports A, B, E and K

- Parallel port registers
  - Data Direction Registers: DDRA, DDRB, DDRE, DDRK
    - Defines pins as input or output
  - Data Registers: PORTA, PORTB, PORTE, PORTK
    - Used to determine values of input pins or set the levels of the output pins.
  - PUCR Register
    - On bit per port to activate the pull up registers

Data Direction Register: Port A



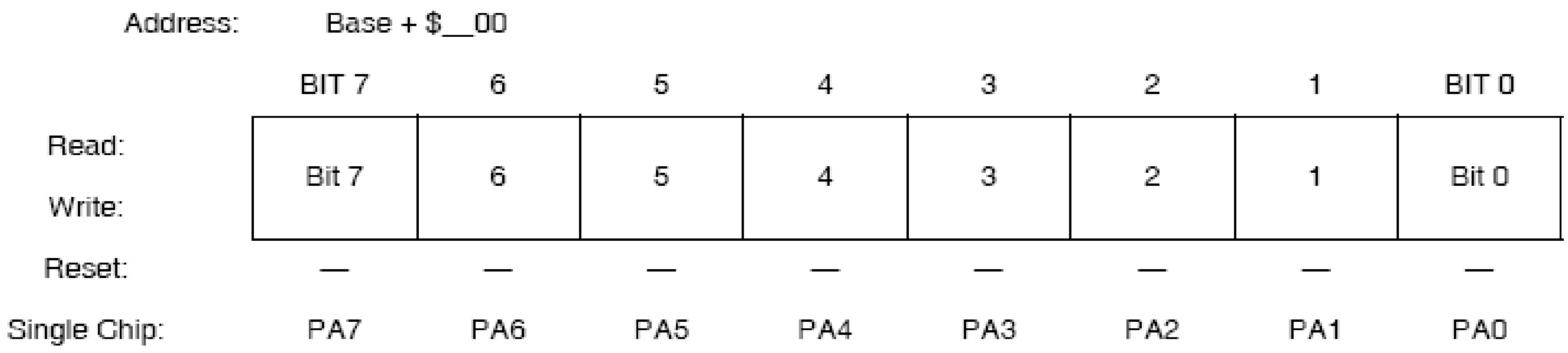
Address: Base + $__02

	BIT 7	6	5	4	3	2	1	BIT 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-4 Data Direction Register A (DDRA)

- Each bit corresponds to a pin of the port
- Configure the port pins
  - 0 ⇒ the pin corresponding to the bit is configures as an input pin
  - 1 ⇒ the pin corresponding to the bit is configured as an output pin

Data Register: Port A



Address: Base + $__00

	BIT 7	6	5	4	3	2	1	BIT 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	—	—	—	—	—	—	—	—
Single Chip:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

- Each bit corresponds to a pin of the port
- Reading:
  - Gives the state of the pin, 0V $\Rightarrow$ 0, 5V $\Rightarrow$ 1
- Writing:
  - For output pins, sets the level on the pins: 0 $\Rightarrow$ 0V, 1 $\Rightarrow$ 5V
  - No effect on input pins

HCS12 PARALLEL PORTS

DATA REGISTERS

CONTROL REGISTERS

PORTA	$0000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PORTB	$0001	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
DDRA	$0002	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
DDRB	$0003	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
	…..	…..	…..	…..	…..	…..	…..	…..	…..
PORTE	'$0008	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
DDRE	$0009	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
	…..	…..	…..	…..	…..	…..	…..	…..	…..
PUCR	$000C	PUPKE	0	0	PUPEE	0	0	PUPBE	PUPAE
	…..	…..	…..	…..	…..	…..	…..	…..	…..
PORTK	$0032	PK7	PK6	PK5	PK4	PK3	PK2	PK1	PK0
DDRK	$0033	DDRK7	DDRK6	DDRK5	DDRK4	DDRK3	DDRK2	DDRK1	DDRK0

HCS12 PARALLEL PORTS

DATA REGISTERS

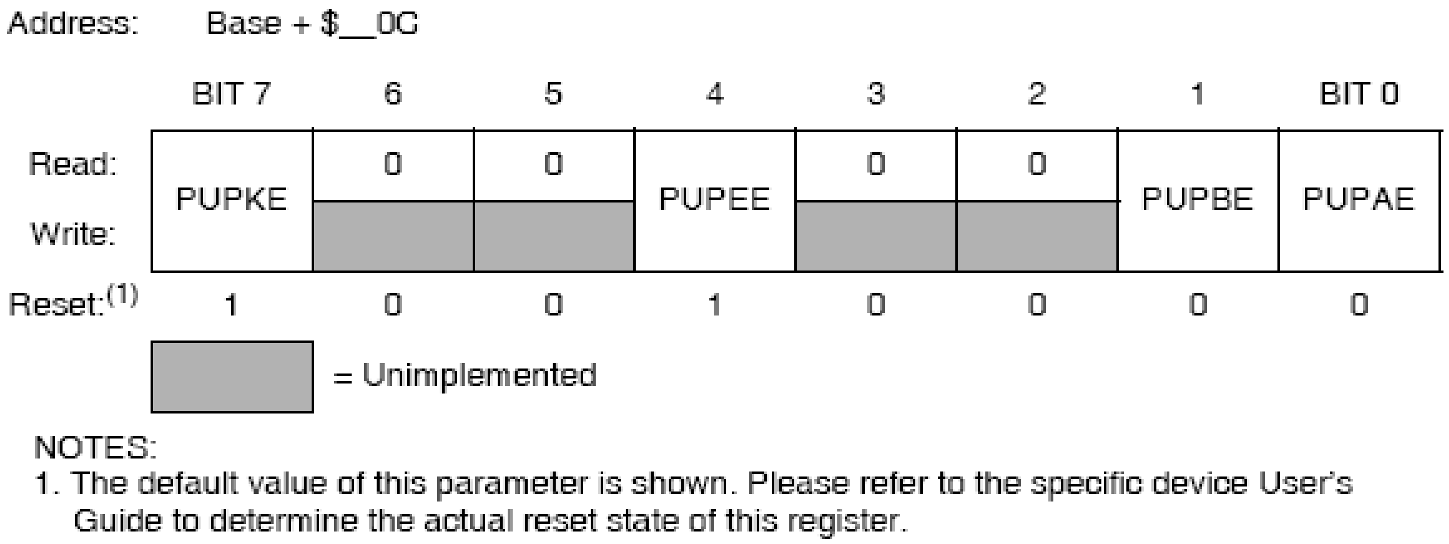
PORTA	$0000	**PORTB**	$0001	**PORTE**	$0008	**PORTK**	$32
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PK7	Pk6	PK5	PK4	PK3	PK2	PK1	PK0

CONTROL REGISTERS (Data Direction Registers - DDR)

DDRA	$0002	**DDRB**	$0003	**DDRE**	$0009	**DDRK**	$0033
DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
DDRK7	DDRK6	DDRK5	DDRK4	DDRK3	DDRK2	DDRK1	DDRK0

PUCR	$000C						
PUPKE	0	0	PUPEE	0	0	PUPBE	PUPAE

The PUCR



Address: Base + $__0G

	BIT 7	6	5	4	3	2	1	BIT 0
Read:	PUPKE	0	0	PUPEE	0	0	PUPBE	PUPAE
Write:		▨	▨		▨	▨		
Reset:(1)	1	0	0	1	0	0	0	0

▨ = Unimplemented

NOTES:
1. The default value of this parameter is shown. Please refer to the specific device User's Guide to determine the actual reset state of this register.

Figure 3-11 Pullup Control Register (PUCR)

-When the PUCR bit is set, pullup resistors will be activated for the corresponding port (for input)

-No effect on output pins

Port A Configuration

- The M68HC9S12DP256 is running in single chip mode. What instructions configures Port A as an output port and writes the contents of accumulator A on the Port A pins.

```
PORTA        equu   $00      ; data register – note address
DDRA         equ    $02      ; data direction register
             movb   #$FF,DDRA  ; Port A for output
             staa   PORTA      ; write acc A to Port A
```