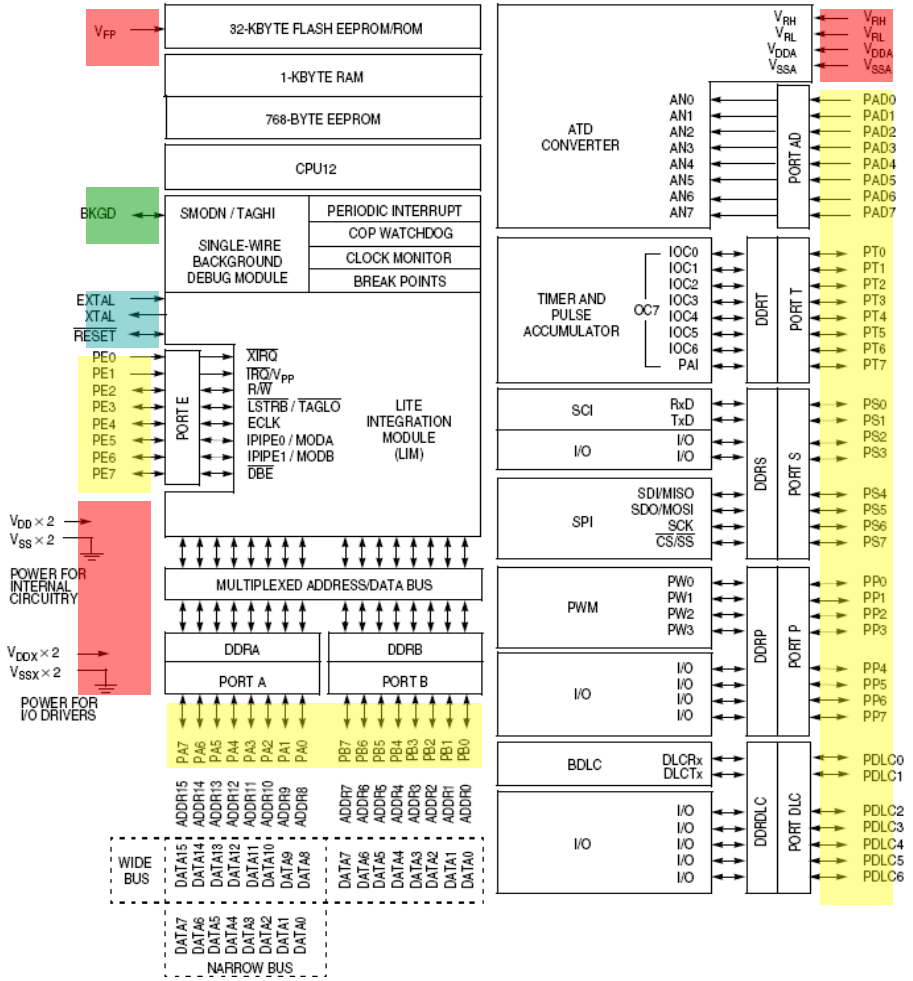
Block Diagram for MC68HC12BE32

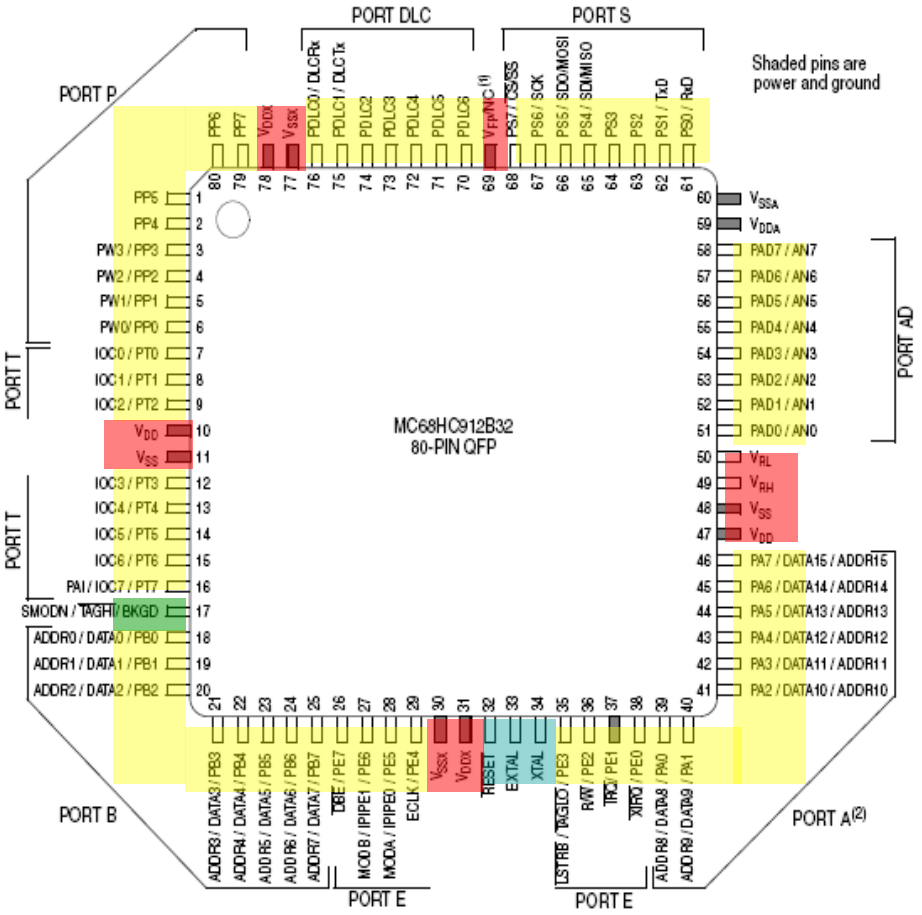
- Power Supply……
- Clock and RESET
- Mode……..
- I/O Ports (8x8)…..
  - PORT A
  - PORT B
  - PORT E
  - PORT DLC
  - PORT P
  - PORT S
  - PORT T
  - PORT AD



MC68HC12BE32 80-PIN QFP

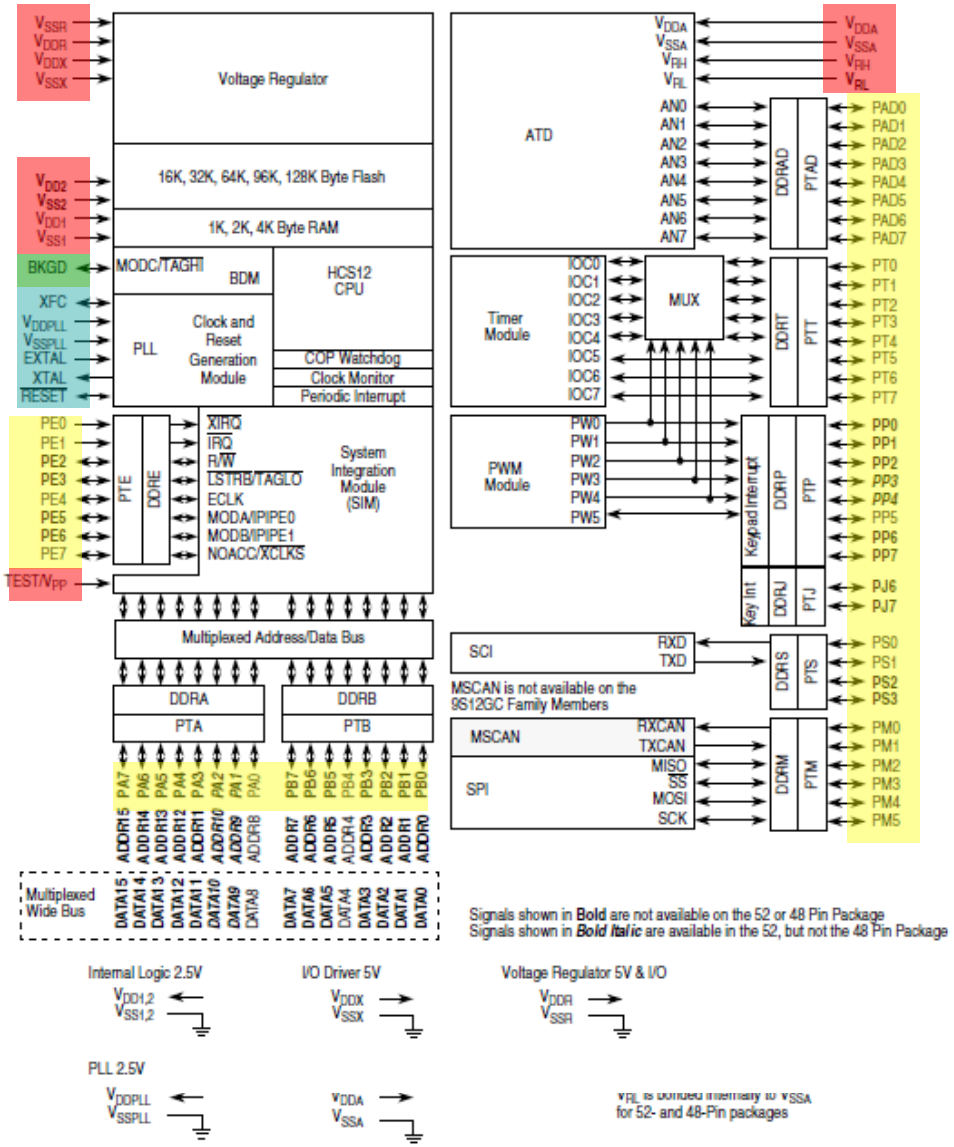
- Power Supply……13
- Clock & RESET……3
- Mode……1
- PORT A……8
- PORT B……8
- PORT E……8
- PORT DLC……7
- PORT P……8
- PORT S……8
- PORT T……8
- PORT AD……8

Total……80



Block Diagram for MC9S12C Family

- Power Supply……
- Clock and RESET
- Mode……..
- I/O Pins…..
  - PORT A 8
  - PORT B 8
  - PORT E 8
  - PORT J 2
  - PORT M 6
  - PORT P 8
  - PORT S 4
  - PORT T 8
  - PORT AD 8



MC9S12C 80-PIN QFP

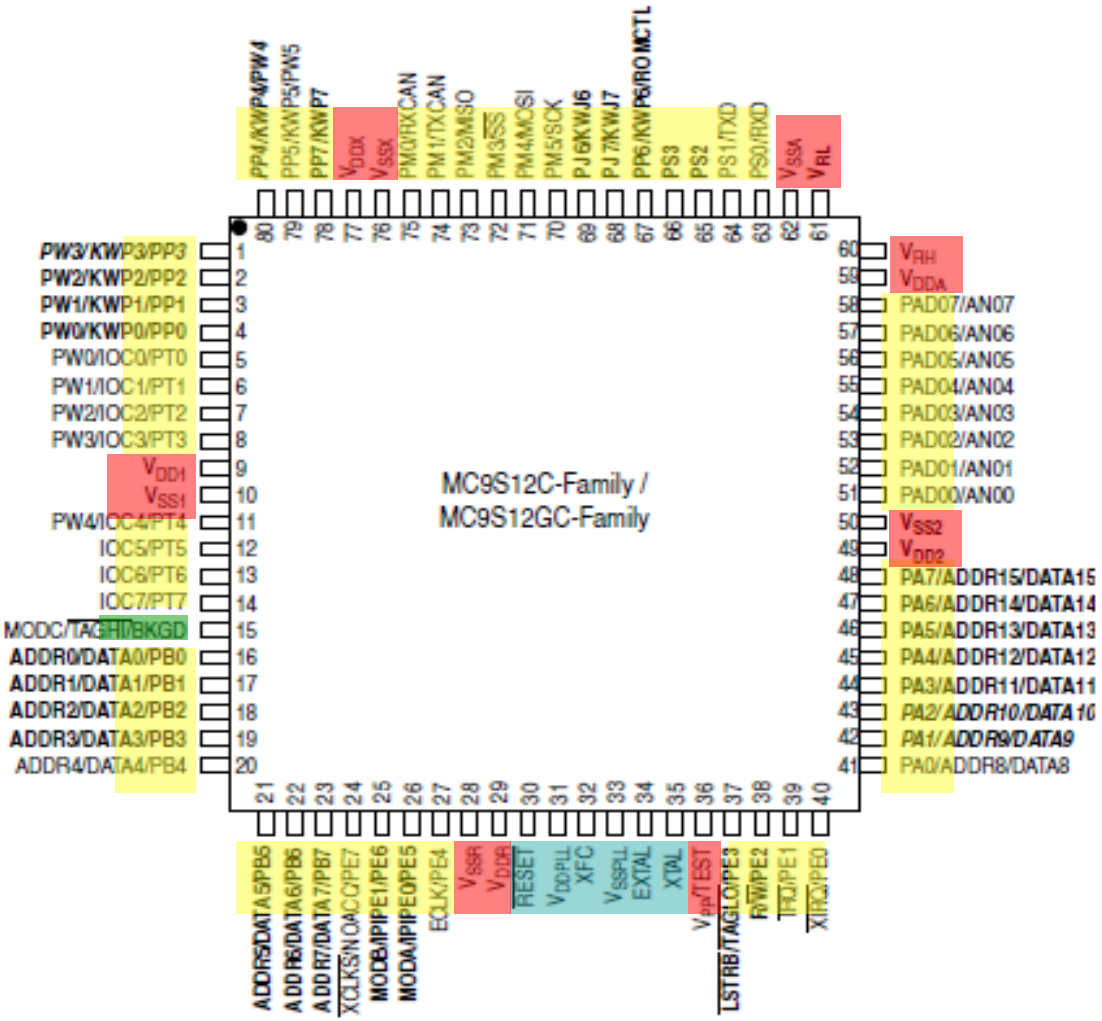
- Power Supply…….13
- Clock & RESET……6
- Mode……..1

I/O

- PORT A……..8
- PORT B……..8
- PORT E……..8
- PORT J……..2
- PORT M……..6
- PORT P……..8
- PORT S……..4
- PORT T……..8
- PORT AD……..8

Total I/O 60

Total……80



GPIO Ports

Digital I/O Ports:

- All microcontrollers have ports that may be used to input or output digital (binary) information.
- In some cases a port may be restricted to be either an input or output port exclusively.
- For most ports in modern microcontrollers you may choose the direction of information flow and program the port to operate in either mode.
- In some cases, this choice can be made for individual bits within the port.
- When the microcontroller is reset, all bidirectional ports are initialized to be in the input mode for circuit safety. After reset your program must choose the operating mode for the ports.

M68HC12 Parallel I/O

Register Block:

- Size ½K (512 bytes) (1K for other types)
- Mappable to any 2-Kbyte boundary
  - by manipulating bits in the register-initialization register (INITRG).
  - INITRG15:11 establishes the upper five bits of the register block's 16-bit address.
- Occupies the first 512 bytes of the 2-Kbyte block.

Almost all I/O features must be programmed or initialized before use by setting or resetting bits in control registers.

M68HC12 Parallel I/O

The M68HC12 incorporates eight ports:

Port A, B, AD, DLC, E, P, S, and T which are used to control and access the various device subsystems.

When not used for these purposes, port pins may be used for general-purpose I/O.

Each port consists of:

- A data register which can be read and written at any time
- A data direction register (DDR) which controls the direction of each pin (except port AD and PE1–PE0)

Setting a bit in the DDR makes the corresponding bit an output

Resetting a bit in the DDR makes the corresponding bit an input

After reset, all port pins are configured as input.

Port Description

Port name	Data direction	Data/DD register addresses	Description (GPIO or…)
PORTA	In/Out	$0000/$0002	External Address/Data
PORTB	In/Out	$0001/$0003	External Address/Data
PORTAD	In	$006F	A/D converter inputs
PORTDLC	In/Out	$00FE/$00FF	Byte data link comm subsystem
PORTE	PE7-PE2 Out PE1-PE0 In	$0008/$0009	Bus control signals
PORTP	In/Out	$0056/$ $0057	4 Pulse Width Modulator
PORTS	In/Out	$00D6/$ $00D7	8-bit interface to Serial comm/Serial peripheral
PORTT	In/Out	$00AE/$ $00AF	Input capture/Output compare in Timer and pulse accumulator

Addressing Modes for Accessing I/O ports

```
    ldx   #0
;
    ldaa  1          ; Direct
;
    ldaa  1,x        ; Indexed
;
    ldx   vector     ; Indexed indirect
    ldaa  [0,x]
;
vector: dw       1
```

```
regs:   equ      0
portb:  equ      1
;
    ldx   #regs
;
    ldaa  PORTB              ; Direct
;
    ldaa  PORTB,x            ; Indexed
;
    ldx   vector   ; Indexed indirect
    ldaa  [0,x]
;
vector: dw       PORTB
```

Setting the Direction of I/O Port Bits

```
PORTB: equ     1                ; Address of port B
DDRB:  equ     3                ; Address of its DD register
OBITS: equ     %11110000        ; Bits to be output
;
       bset    DDRB,OBITS       ; Set DD register
;
; Output data to bits 7-4
;      ldaa    #%10101010
;      staa    PORTB
;
; Read data to bits 3-0
       ldaa    PORTB
```

If a port has a mixture of input and output bits:

- Writing to a port affects only those bits that are configured for output
- Reading the port returns the values on the input bits as well as the last values output to the output bits

Real Time I/O Synchronization

Typically microprocessors are much faster than the I/O devices they serve.
They must be synchronized using SOFTWARE and HARDWARE techniques

Software synchronization

- Delay loops (Blind cycles)
- Polling
- Handshaking

Interrupts

- I/O device causes software to execute upon request
- Periodic interrupts check the I/O status (Periodic polling)

Direct Memory Access (DMA)

I/O device directly transfers data to/from memory

I/O Using Delay Loops

```
        ldab    #Count          ; Load B with count
Delay:  dbne    b, Delay        ; Decrement B, if B ≠ 0 repeat
```

DelayTime = t(ldab) + t(dbne) × Count

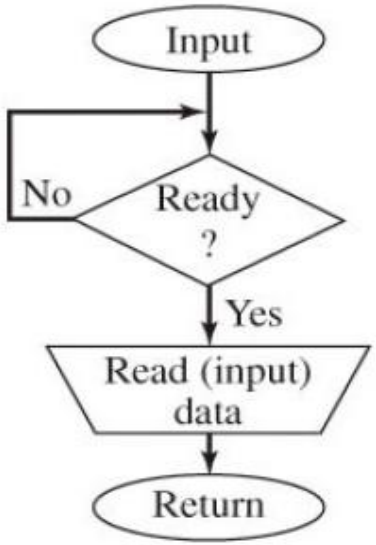
For longer time delays use nested loops:

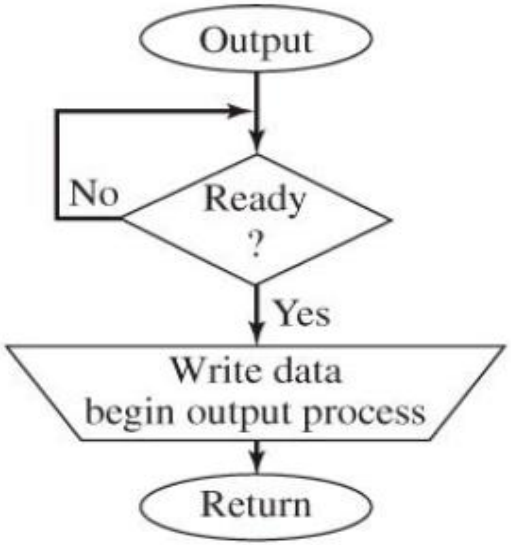
```
        ldab    #Count          ; Load B with count
Delay:  ldx     #cnt            ; Load X with cnt
inDelay: dbne   x,inDelay       ; Decrement X, if X ≠ 0 repeat
        dbne    b, Delay        ; Decrement B, if B ≠ 0 GOTO Delay
```

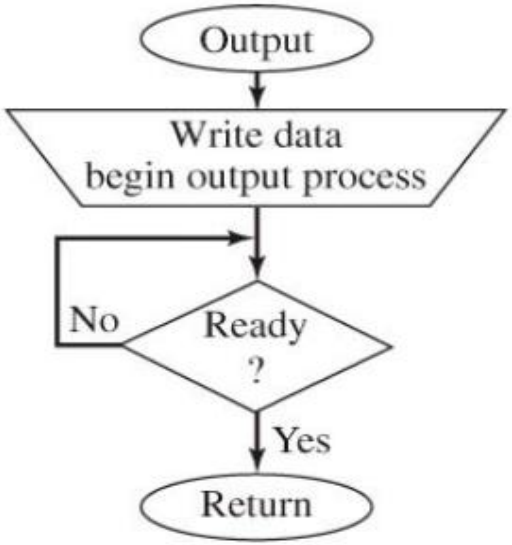
Pros	Cons
•Simple	•Inflexible
•Predictable	•Inefficient if the delay is long

Works well for simple, high-speed devices.

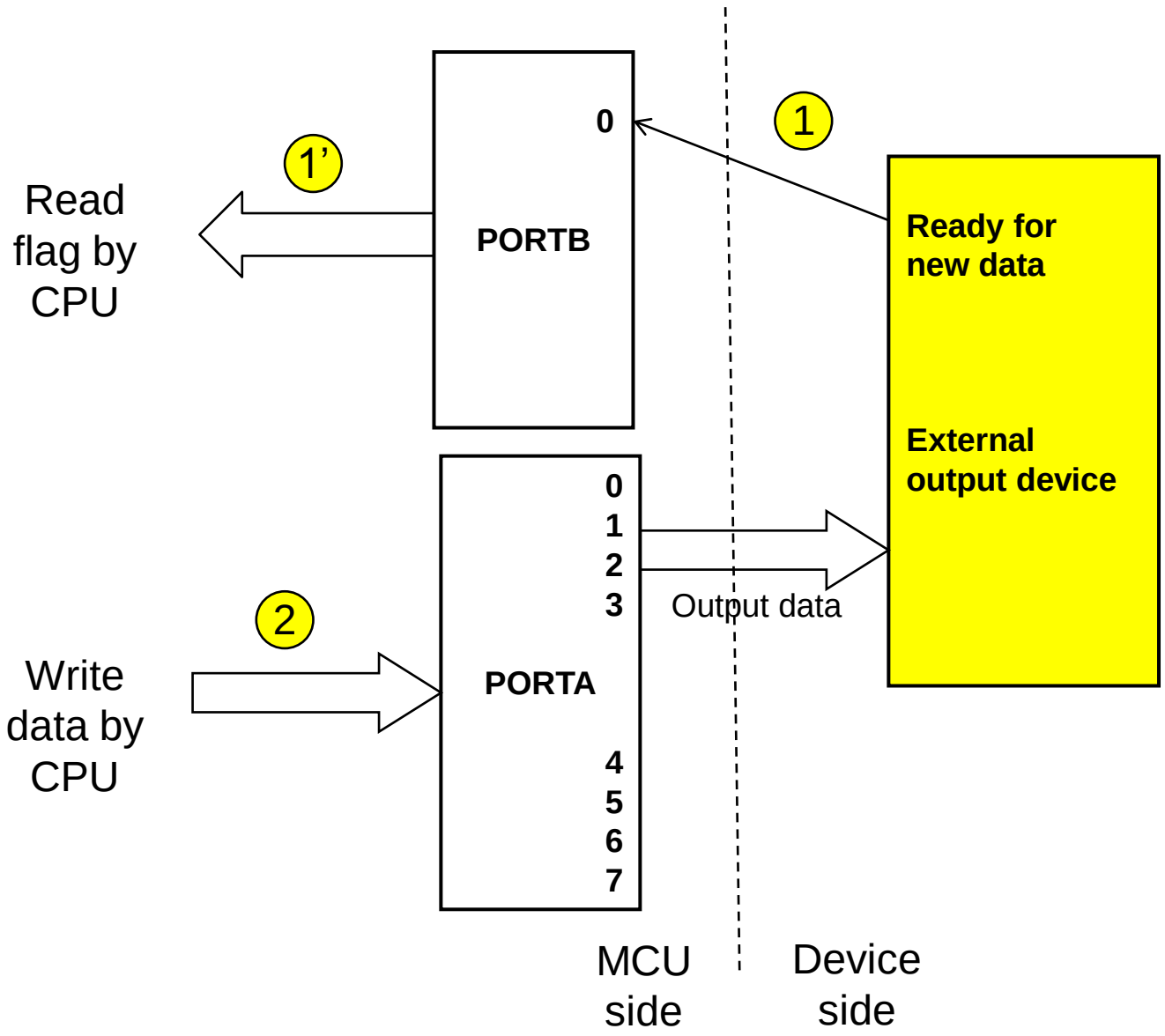
I/O Using Polling



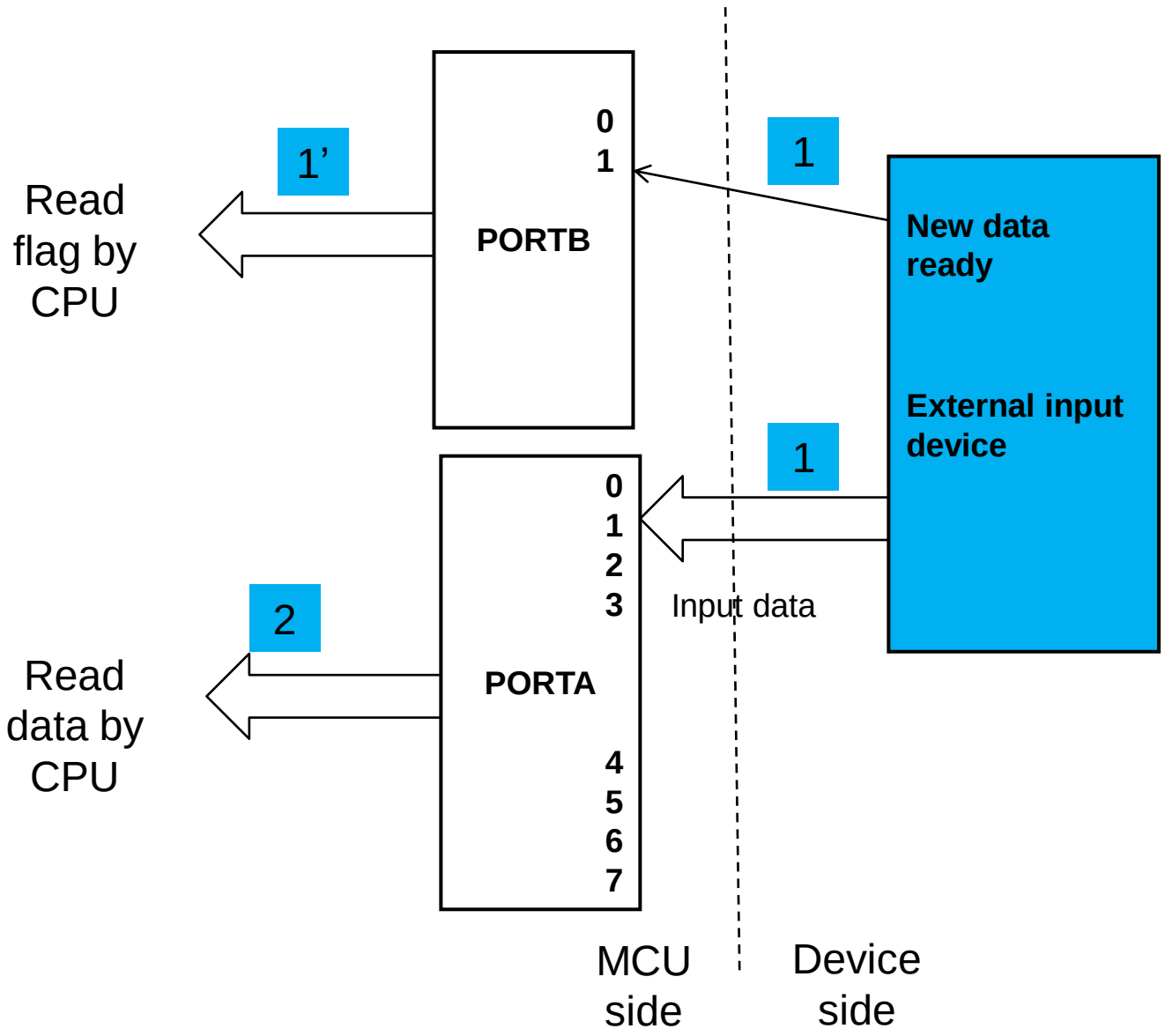




Output Using Polling



Input Using Polling



I/O Using Polling (Code)

```
PORTA   equ     0
PORTB   equ     1
DDRA    equ     2
Bit0    equ     %00000001
Bit1    equ     %00000010
O_bits  equ     %00001111
;
; Set up PORTA[3:0] for output
        bset    DDRA,O_bits
;
; Output data to PORTA: takes 2 steps
; 1- wait until the status bit PORTB[0]=1
Spin1:  brclr   PORTB,Bit0,Spin1
; 2- then output the data
        ldaa    data1
        staa    PORTA
;
```

```
; Input data from PORTA: takes 2 steps
; 1- wait until the status bit PORTB[1]=1
Spin2:  brclr   PORTB,Bit1,Spin2
; 2- then input the data
        ldaa    PORTA
        staa    data2

data1   ds.b    1
data2   ds.b    1
```