CpE 390: Microprocessor Systems
Lecture 7

Parallel Ports

Basic Concepts of I/O

- I/O devices (peripheral devices) are pieces of equipment that an embedded system used to exchange data with the external world

- The speed and electrical characteristics of I/O devices are different from CPU, therefore, it is impossible to connect them directly to CPU

- I/O devices are usually attached to the data bus via interface chips

Interface Chips

- It consists of control registers, status registers, data direction latches, data registers, and control circuitry.
- It has data pins that are connected to the CPU and I/O port pins that are connected to the I/O devices.
- Each interface chip has a chip enable signal input or inputs which, when asserted, allow the interface chip to react to the data transfer request.
- Data transfer between an I/O device and the interface chip can be proceeded bit-by-bit (serial) or in multiple bits (parallel).

Interface Chip

- In input operation, the input device places data in the data register on the IC, which holds data until they are read by the processor
- In output operation, the processor writes data into the data register on the IC, and the data register holds them until they are fetched by the output device
- Address decoder makes sure that each time one and only one peripheral device respond to the CPU's I/O request.

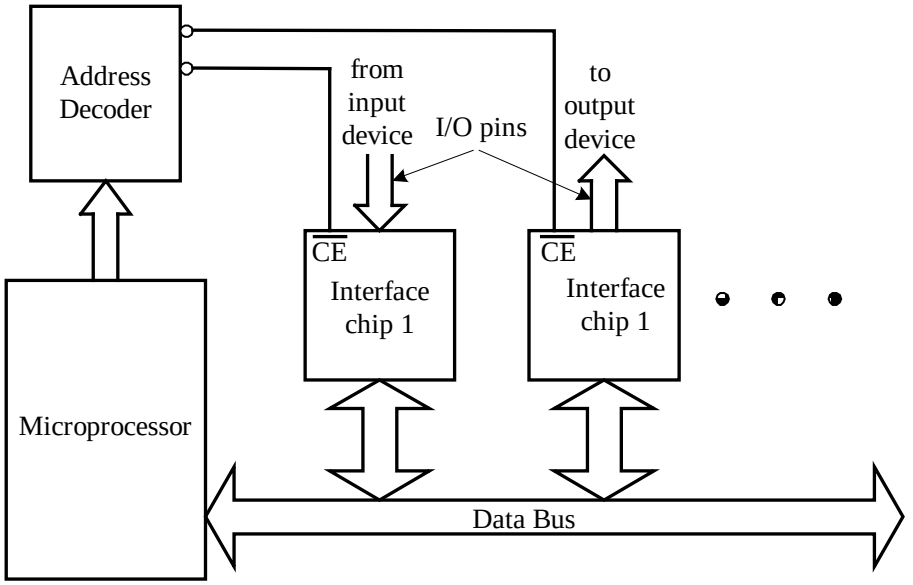


Figure 7.1 Interface chip, I/O devices, and microprocessor

I/O Schemes

Isolated I/O scheme

- Dedicated instructions for I/O operations.
- A separate address space for I/O devices.
- Example: Intel x86

Advantages:

- Not as susceptible to software errors as the memory mapped I/O because different instructions are used to access memory and I/O devices
- I/O devices do not occupy the limited memory space
- The I/O address decoder can be smaller because the I/O address space is much smaller

Disadvantages:

- inflexible I/O instructions
- inflexible I/O addressing modes

I/O Schemes

Memory-mapped I/O scheme

- Use the same instruction set to perform memory accesses and I/O operations.
- The I/O devices and memory components are resident in the same memory space.
- Example: 68hc12

Advantages:

- All instructions and addressing modes are available for I/O operations, so I/O programming is very flexible

Disadvantages:

- More susceptible to software errors
- A smaller memory space is available for main memory

Speed difference between CPU & I/O

- Problem: I/O devices are typically slower than the CPU and may require many cycles to complete an operation

- If the CPU is performing multiple operations on a single device, then it must wait for one operation to complete before starting the next one
  - Example 1: CPU writes several characters to an output device
  - Example 2: Keyboard interface
  - Solution ?

I/O Transfer Synchronization

- Synchronizing data transfer between the CPU and the interface chip.
- Synchronizing data transfer between the interface chip and the I/O device.

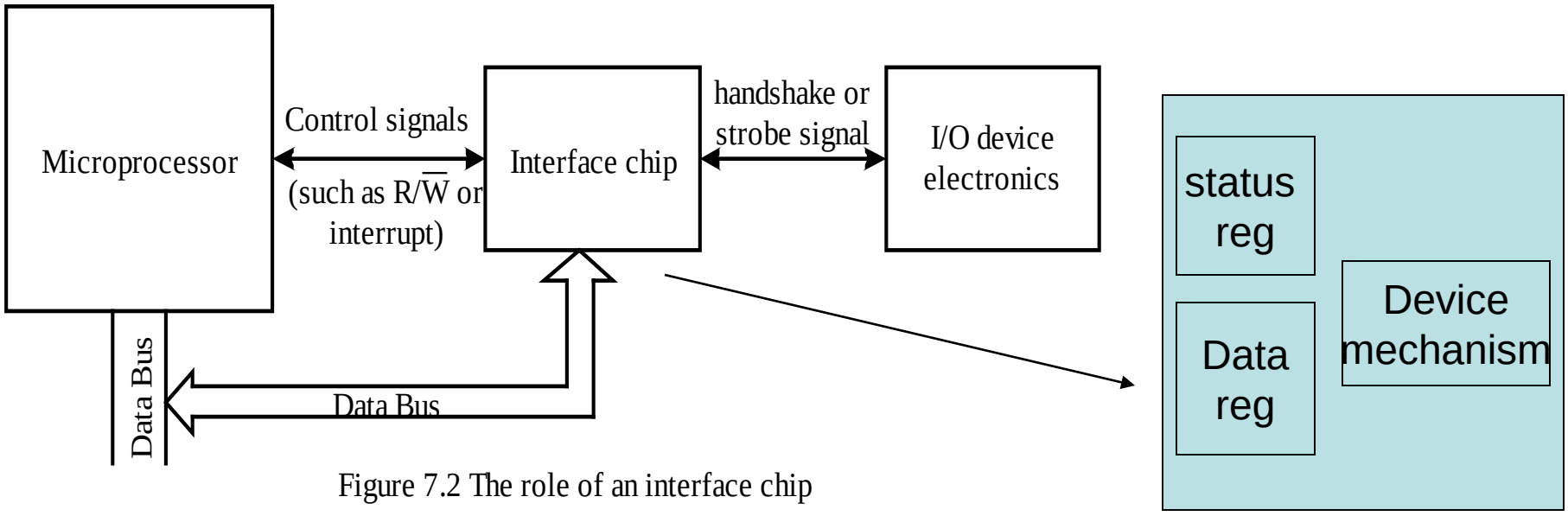


Figure 7.2 The role of an interface chip

Synchronizing CPU & IC

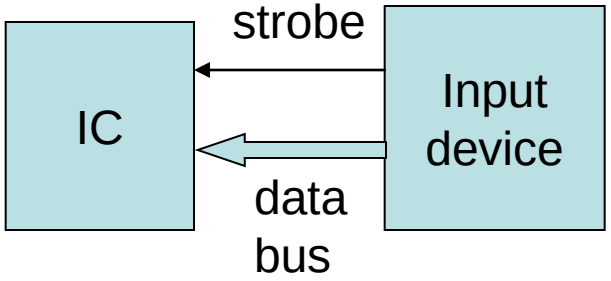
- *Polling method*
  - The microprocessor checks a status bit of the interface chip to find out if the IC has received new data from the input device if it can send new data to the IC.
  - Advantage: simple to implement
  - Disadvantage: CPU is tied up

- *Interrupt-driven method*
  - The IC interrupts the microprocessor whenever it has received new data from the input device it can accept new data from the microprocessor.
  - Advantage: CPU utilization is improved, it can do non I/O related computation or control of other I/O devices
  - Disadvantage: overhead of interrupts

Synchronizing IC and I/O Devices

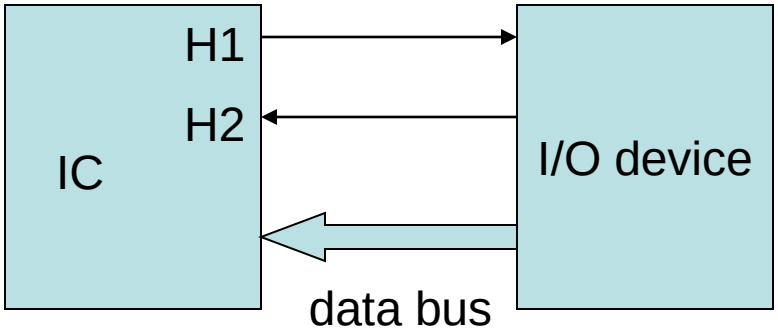
- ***Brute-force method*** -- useful when the data timing is unimportant
  - The data will be read from or written to the IC directly without checking.

- ***The strobe method*** -- a strobe signal is used to indicate that data are stable on I/O port pins
  - The data can only be latched on I/O port pins on IC when a strobe signal is asserted



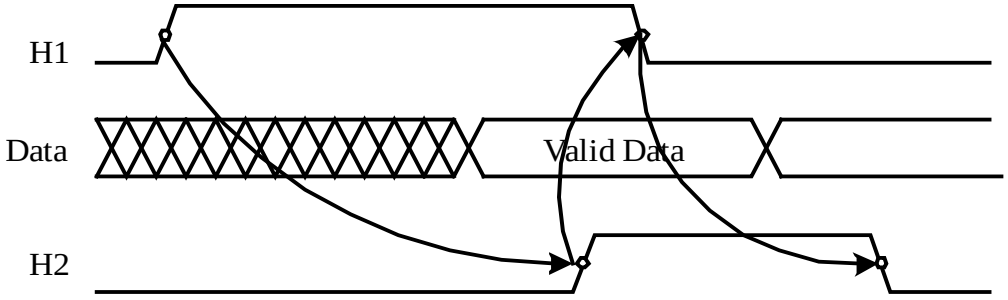
Synchronizing IC and I/O Devices

- ***The handshake method*** -- used when timing is crucial
  - Two handshake signals are used to synchronize the data transfer. One signal, call it H1, is asserted by the IC. The other signal, call it H2, is asserted by the I/O device.
  - Two handshake modes are available -- pulse mode and interlocked mode.

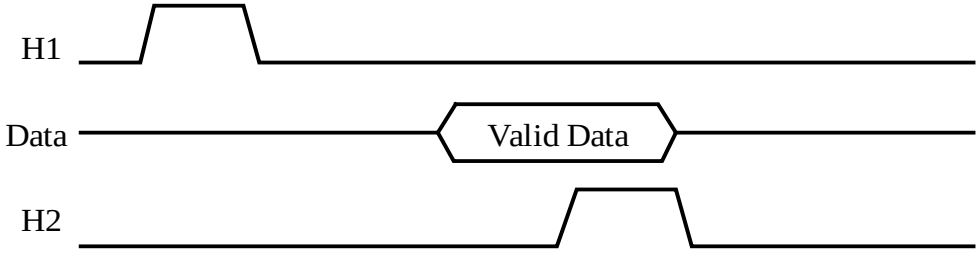


Input Handshake Protocol

Step 1. The interface chip asserts (or pulses) H1 to indicate its intention to input data.
Step 2. The input device puts data on the data port pins and also asserts (or pulses) the handshake signal H2.
Step 3. The interface chip latches the data and de-asserts H1. After some delay, the input device also de-asserts H2.



(a) Interlocked



(b) Pulse mode

Figure 7.3 Input Handshakes

Output Handshake Protocol

Step 1. The interface chip places data on the port pins and asserts (or pulses) H1 to indicate that it has valid data to be output.

Step 2. The output device latches the data and asserts (or pulses) H2 to acknowledge the receipt of data.

Step 3. The interface chip de-asserts H1 following the assertion of H2. The output device then de-asserts H2.

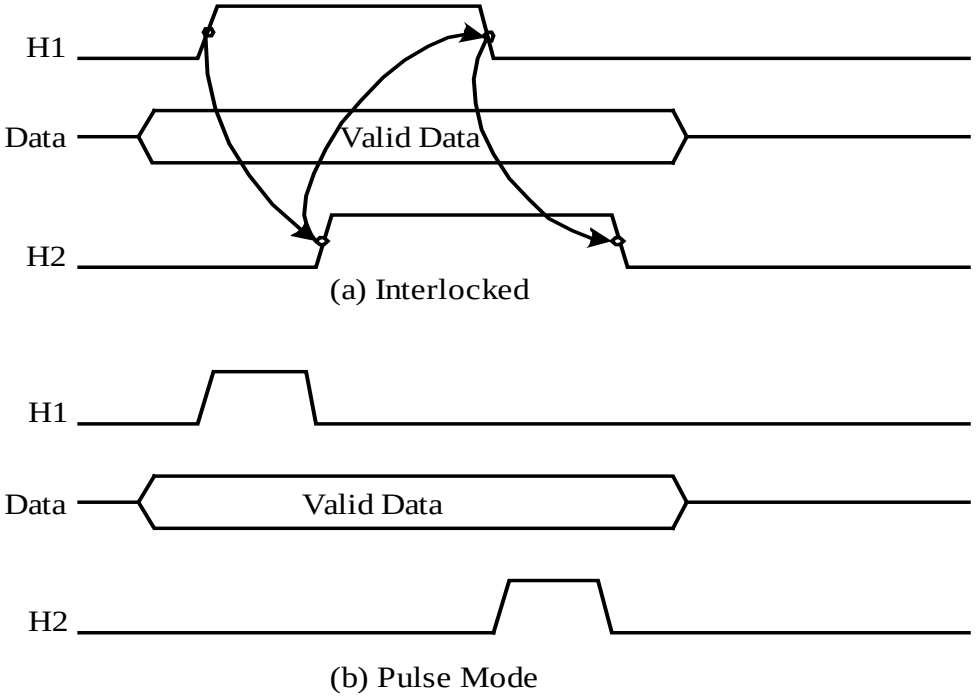


Figure 7.4 Output Handshaking

Overview of 68HC12 Parallel Ports

- In general, a 68HC12 I/O port has
  - I/O pins
  - A data register
  - A data direction register: set a bit to 1 (0) will configure the corresponding pin for output (input).
- All I/O pins serve multiple functions, general-purpose I/O or special functions.
- All members except the 812A4 multiplexed address and data pins on port A and B pins.
- All ports direction need to be configured before they are used
  - Default: input ports

Port A & B

- In expanded mode, these two ports are used multiplexed address/data bus
- In single chip mode, these two ports are used for general I/O ports.
- The direction of each port A or port B pin is configured by setting or clearing the associated bit in the DDRA or DDRB register: 1 (output)/0 (input).

Example. Assuming that the 68HC12 is operating in single-chip mode, write/read operations.

Solution:

We need to configure port A for output before outputting the contents of accumulator A to it.

```
PORTA  equ     $00       ; check Appendix C register blocks for register address
PORTB  equ     $01
DDRA   equ     $02
DDRB   equ     $03
       movb    #$FF,DDRA       ; configure port A for output
       staa    PORTA           ; output accumulator A to port A
```

Port A &B

```
movb #$00, DDRB
ldab  PORTB

bset   DDRA, $40    ; config pin6 of PORTA as output pin

bclr    DDRB, $80   ;  config pin7 of PORTB as input pin

ldaa  PORTB
bpl    wait             ;  branch if N=0
jsr     routine 1


movw #$FF, DDRA ; config both A and B as output
std    PORTA
```

Port C & D

- These two ports are available only in the 68HC812A4.
- In expanded wide mode, Port C and Port D are used for the upper and lower bytes of the 16-bit data bus.
- In expanded narrow mode, Port C is used to carry the upper and lower bytes of data whereas Port D is used as a general-purpose I/O port.
- In single-chip mode, the pin directions of Port C and Port D are configured via the DDRC and DDRD registers.

Example 7.2 Write an instruction sequence to read the current signal levels of Port C and Port D into accumulator A and B.

Solution:

```
PORTC  equ     $04
PORTD  equ     $05
DDRC   equ     $06
DDRD   equ     $07
       …
       movw    #$00,DDRC; configure Port C and Port D for input
       ldd     PORTC  ; read in 16 bits in one operation
```

Port E

- Used for bus control and interrupt service request signals.
- Port E pins 1 and 0 can only be used for input.
- Pins 6 and 5 are used to set up the operating mode when the CPU exits the reset mode.
- The PEAR register (see following slides) determines pin functions and the DDRE register determines whether a pin is input or output.
- The PEAR settings override the DDRE settings.

	7	6	5	4	3	2	1	0	
value after	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	$0008
reset:	0	0	0	0	0	0	0	0	
Alternate pin	ARST or DBE/ CAL	IPIPE1/ MODB	IPIPE0/ MODA	ECLK	LSTRB/ TAGLO	R/$\overline{W}$	$\overline{IRQ}$ /V_{PP}	$\overline{XIRQ}$	

Figure 7.7 PORTE register

Port E

	7	6	5	4	3	2	1	0	
value after reset:	NDBE	0	PIPOE	NECLK	LSTRE	RDWE	0	0	$000A
normal expanded	0	-	0	0	0	0	-	-	
special expanded	0	-	1	0	1	1	-	-	
peripheral	1	-	0	1	0	0	-	-	
normal single-chip	1	-	0	1	0	0	-	-	
special single-chip	0	-	1	0	1	1	-	-	

NDBE: *No data bus enable*. Can be read/written any time.
When set to 0, PE7 is used for external control of data enables on memories. When set to 1, PE7 is used for general-purpose I/O

PIPOE: *Pipe signal output enable*.
In normal mode: write once. Special mode: write anytime except the first time. This bit has no effect in single chip modes.
0 = PE[6:5] are general-purpose I/O
1 = PE[6:5] are outputs and indicate the state of the instruction queue.

NECLK: *No external E clock.* Can be read anytime.
In expanded mode, writes to this bit has no effect. E clock is required for de-multiplexing the external address. NECLK can be written once in normal single chip mode and can be written anytime in special single chip mode.
0 = PE4 is the external E-clock. 1 = PE4 is a general-purpose I/O pin.

LSTRE: *Low strobe ($\overline{LSTRB}$) enable*. Can be read anytime.
In normal modes: write once; special modes: write anytime except the first time. This bit has no effect in single-chip modes or normal expanded narrow mode.
0 = PE3 is a general-purpose I/O pin.
1 = PE3 is configured as the $\overline{LSTRB}$ bus-control output, provided the 68HC12 is not in single chip or normal expanded narrow modes.

RDWE: *Read/write enable*. Can be read anytime.
In normal modes: write once; special modes: write anytime except the first time. This bit has no effect in single-chip modes.
0 = PE2 is a general-purpose I/O pin
1 = PE2 is configured as the R/$\overline{W}$ pin. In single-chip mode, RDWE has no effect and PE2 is a general-purpose I/O pin.
R/W is used for external writes. After reset in normal expanded mode, it is disabled. If needed it should be enabled before any external writes.

Figure 7.8a PEAR register (common to all members)

Port P & Port S

- Port P
  - When the PWM function is enabled, the lower four Port P pins are used for PWM outputs.
  - The upper four pins are used for general I/O.
  - The PORTP register and DDRP register are Port P data register and data direction register.
- Port S
  - Port S is the 8-bit interface to the standard serial interface consisting of serial communication interface (one channel SCI in B family, two channels in other 68HC12 members) and serial peripheral interface (SPI).
  - Port S pins are available for general-purpose I/O when serial interfaces are not enabled.

Port T & Port AD

- Port T
  - This port is used either as the input capture and output compare functions pins or general purpose I/O pins.
  - The TEN bit in the TSCR register enables the timer function.
  - The PORTT register and DDRT are Port T data register and data direction register.
  - Pin 7 is also used as the pulse accumulator input when the PA function is enabled.
- Port AD
  - Pins of this port are the input to the analog-to-digital converter system.
  - When A/D converter is disabled, these pins are available for general I/O. The ADPU bit in the ATDCTL2 register enables the A/D converter.
  - Port AD pins are input only. PORTAD is the data register of this port.

Port AD0 and AD1

- These two ports are analog input interface to the analog-to-digital subsystem.
- When A/D subsystem is not enabled, these pins are available for input.
- The ADPU bit in the ATD0CTL2 and ATD1CTL2 registers enables the AD0 and AD1.
- PORTAD0 and PORTAD1 are data registers of the AD0 and AD1 subsystem, respectively.

Interface with Simple I/O Devices

LEDs, switches, keypads, seven-segment displays, ADC, DCA, motors, relays, etc.

Example 7.3 Use the 68HC12 Port P to drive green, yellow, red, and blue LEDs. Light each of them for half of a second in turn and repeat. The 68HC12 uses a 16-MHz crystal oscillator to generate internal clock signals.

Solution:

The upper four pins of the Port P can be used for this purpose. The circuit connection is shown in Figure 7.10.

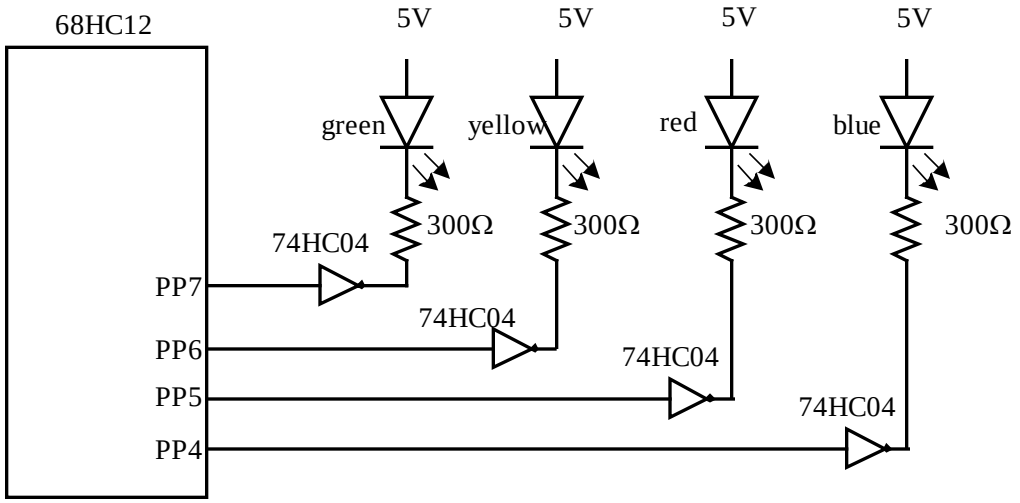


Figure 7.10 Circuit connection for example 7.3

```
PORTP   equ     $56         ; register address
DDRP    equ     $57

        org     $1000
        ldaa    #$FF        ; configure PORTP for output
        staa    DDRP        ;           “
forever ldaa    #$80        ; turn on green LED and turn off other LEDs
        staa    PORTP       ;           “
        jsr     delay_hs    ; wait for half of a second
        ldaa    #$40        ; turn on yellow LED and turn off other LEDs
        staa    PORTP       ;           “
        jsr     delay_hs    ; wait for half of a second
        ldaa    #$20        ; turn on red LED and turn off other LEDs
        staa    PORTP       ;           “
        jsr     delay_hs    ; wait for half of a second
        ldaa    #$10        ; turn on blue LED and turn off other LEDs
        staa    PORTP       ;           “
        jsr     delay_hs    ; wait for half of a second
        jmp     forever     ; repeat
        swi
```

```
delay_hs   ldab    #5              ; 1 E cycle
out_loop   ldx     #20000          ; 2 E cycles
inner_loop psha                    ; 2 E cycles
           pula                    ; 3 E cycles
           psha
           pula
           psha
           pula
           psha
           pula
           psha
           pula
           psha
           pula
           psha
           pula
           nop                     ; 1 E cycle
           nop
           dbne    x,inner_loop    ; 3 E cycles
           dbne    b,out_loop
           rts                     ; 5 E cycles
           end
```

Driving a Single Seven-Segment Display

- A common cathode seven-segment display is driven by the 74HC244 via resistors.
- The output high voltage of the 74HC244 is close to 5V with a 5V power supply.
- The segment patterns for 0 to 9 are shown in Table 7.3.

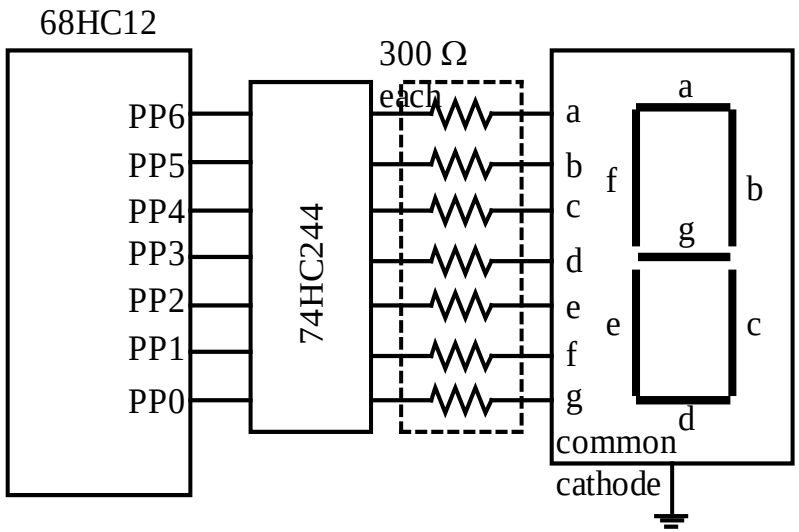


Figure 7.12 Driving a single seven-segment display

Table 7.3 BCD to seven-segment decoder

BCD digit	Segments							Corresponding Hex Number
	a	b	c	d	e	f	g	
0	1	1	1	1	1	1	0	$7E
1	0	1	1	0	0	0	0	$30
2	1	1	0	1	1	0	1	$6D
3	1	1	1	1	0	0	1	$79
4	0	1	1	0	0	1	1	$33
5	1	0	1	1	0	1	1	$5B
6	1	0	1	1	1	1	1	$5F
7	1	1	1	0	0	0	0	$70
8	1	1	1	1	1	1	1	$7F
9	1	1	1	1	0	1	1	$7B

Driving Multiple Seven-Segment Displays

- Time-multiplexing technique is often used to drive multiple displays in order to save I/O pins.
- One parallel port is used to drive the segment pattern and the other port turns on one display at a time. Each display is turned on and then off many times within a second. The **persistence of vision** makes us feel that all displays are turned on simultaneously.

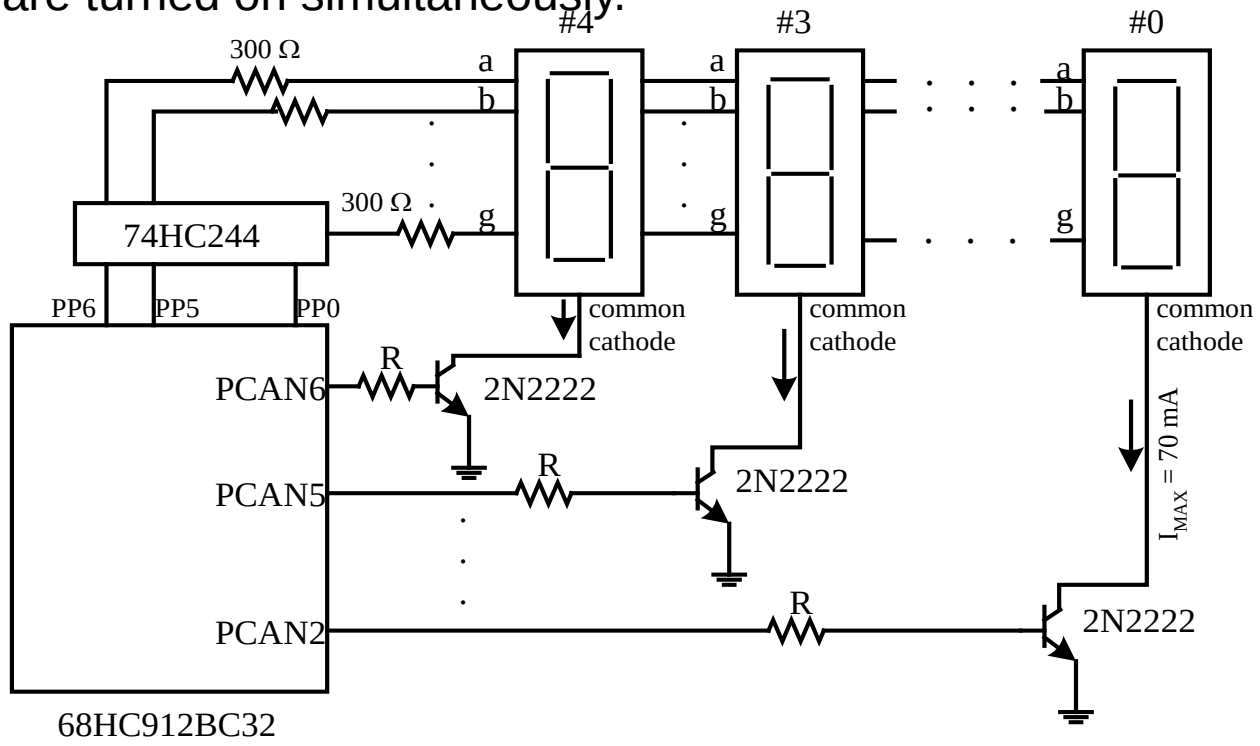


Figure 7.13 Port P and Port CAN together drive five seven-segment displays (HC912BC32)

Example 7.4 Write a sequence of instructions to display 4 on the seven-segment display #4 in Figure 7.13.

Solution:

To display 4 on the seven-segment no. 4 in Figure 7.13, we need to

1. output the hex value $33 to Port P.
2. set the PCAN6 pin to 1
3. clear pins PCAN5..PCAN2 to 0

The instruction sequence is as follows:

```
PORTP    equ      $56
DDRP     equ      $57
PORTCAN           equ          $13E
DDRCAN equ        $13F
four     equ      $33          ; seven-segment pattern of digit 4
         …
         movb     #$7C,DDRCAN      ; configure PORT CAN for output
         movb     #$FF,DDRP  ; configure PORT P for output
         ldaa     PORTCAN    ; set PCAN6 to 1 and clear PCAN5..2 to 0
         ora      #$40       ;        “
         anda     #$C3       ;        “
         staa     PORTCAN    ;        “
         movb     #four,PORTP      ; output the seven-segment pattern to PORTP
```

Using Include File to Save Time

- All the equate directives that define the address of I/O registers have collected in one file **hc12.inc** and place under the **miniide** installation directory.

- By adding the line **#include "..\miniide\hc12.inc"** as the first line of your assembly program, you don't need to type any of those **equate** directives. Where **..** refers to the installation directory of the **miniide** software.

Example 7.5 Write a program to display 12345 on the five seven-segment displays shown in Figure 7.13.

Solution:

- We will display 12345 on display #4, #3, #2, #1, and #0, respectively.
- To selectively light these displays, we will clear pins PCAN6..PCAN2 and then set one of them to 1.
- This can be done by using the table-lookup technique. Table 7.4 will be used.

Table 7.4 Table of display patterns for Example 7.5

seven-segment display	displayed BCD digit	Port P	Port CAN
#4	1	$30	$40
#3	2	$6D	$20
#2	3	$79	$10
#1	4	$33	$08
#0	5	$5B	$04

- The algorithm that utilizes the principle of persistence of vision is shown in Figure 7.14.

```
display    db $30,$40
           db $6D,$20
           db $79,$10
           db $33,$08
           db $5B,$04
```

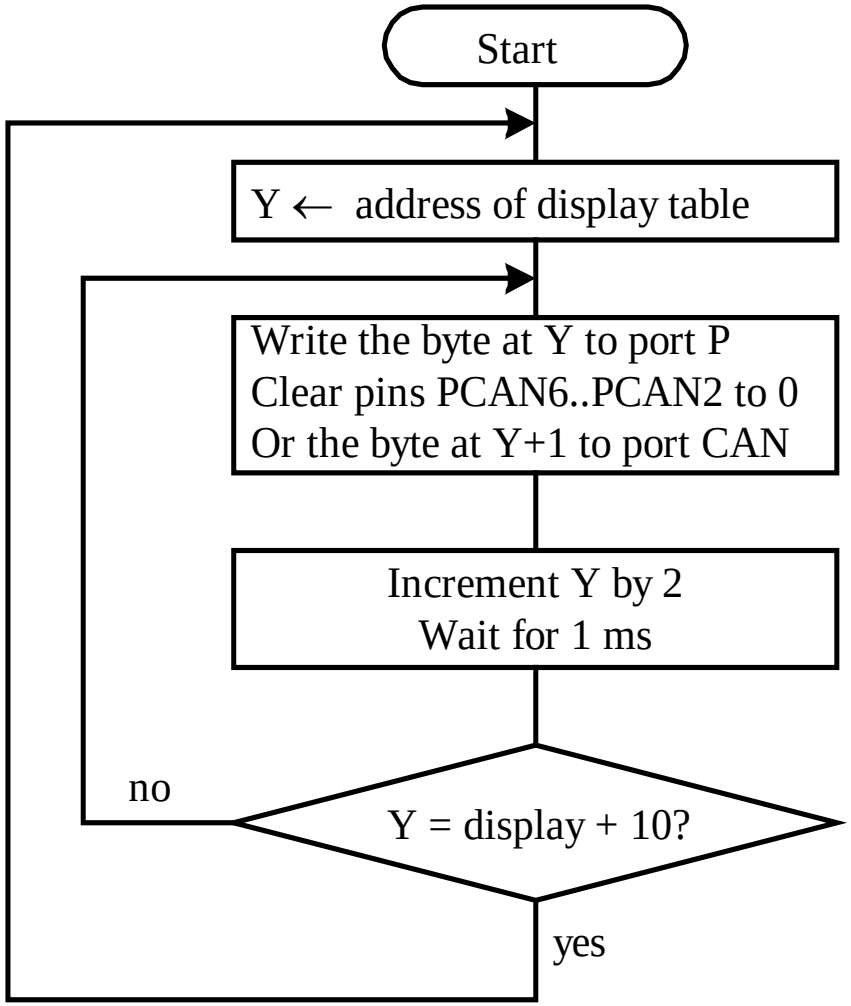


Figure 7.14 Time-multiplexed seven-segment display algorithm

```
#include  "d:\miniide\hc12.inc"
          org     $1000
          ldaa    #$FF
          staa    DDRP        ; configure PORT P for output
          ldaa    #$7C
          staa    DDRCAN      ; configure PORT CAN for output
forever   ldy     #display    ; use Y as a pointer to the display table
next      ldaa    0,y         ; get the digit pattern
          staa    PORTP       ; output the digit pattern
          ldaa    PORTCAN
          anda    #$83
          staa    PORTCAN     ; clear PORTCAN6..PORTCAN2 pins
          ldaa    1,y
          oraa    PORTCAN
          staa    PORTCAN     ; turn on the appropriate seven-segment display
          ; the folloing instructions create a delay of 1 ms for 16MHz oscillator
          iny
          iny
          ldx     #200
again     psha                          ; 3 E cycles
          pula                          ; 2 E cycles
          psha
          pula
```

```
            psha
            pula
            psha
            pula
            psha
            pula
            psha
            pula
            psha
            pula
            nop
            nop
            dbne    x,again         ; 3 E cycles
            cpy     #display+10             ; reach the end of the display table?
            lbeq    forever         ; if yes, then start over again
            jmp     next
display     db      $30,$40
            db      $6D,$20
            db      $79,$10
            db      $33,$08
            db      $5B,$04
            end
```

The AD7302 D/A Converter

-A dual-channel 8-bit D/A converter made by Analog Devices

-The AD7302 converts an 8-bit digital value into an analog voltage.

-The block diagram is shown in Figure 7.24. The AD7302 is designed to be a memory-mapped device. The CS signal must be low in order for this chip to work.

-The AD7302 needs a reference voltage (5V) to operate. The reference voltage could be external one (from the REFIN pin) or the internal VDD.

- Each conversion takes about 2 ms to complete.

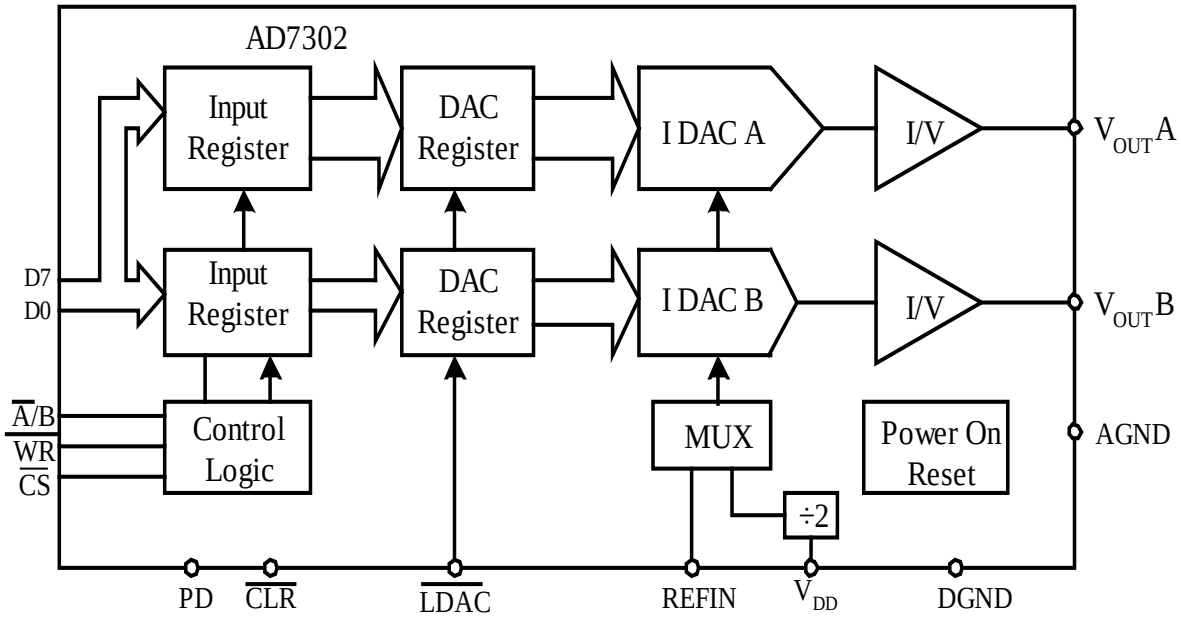


Figure 7.24 Functional block diagram of the AD7302

- The output from either DAC is given by

$$V_{OUT}A/B = 2 \times V_{REF} \times (N/256)$$

where N is the digital value to be converted.

Using the AD7302 to Generate a Waveform

- Configure PP7..PP0, PCAN6..PCAN5 for output.
- Output the digital value from 0 to 255 and repeat. For each value, pull the PCAN6 to low and then to high so that the value on pins PP7..PP0 can be transferred to the AD7302. Pull the signal PCAN5 to low during the process.

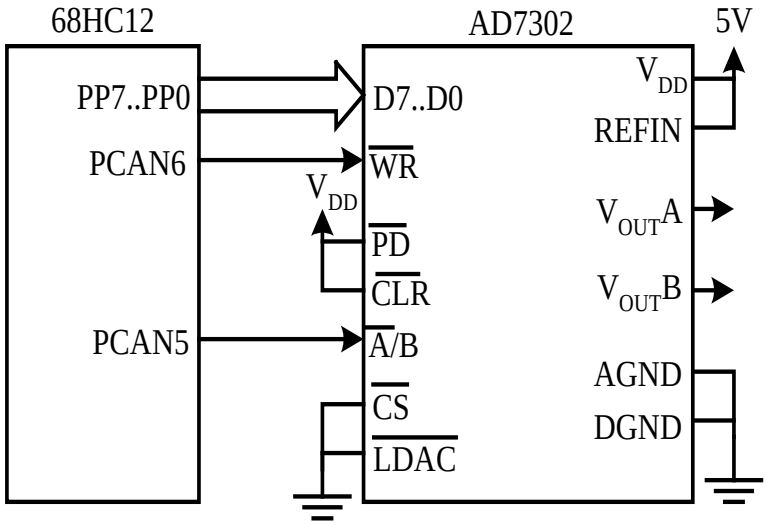


Figure 7.25 Circuit connection between the AD7302 and the 68HC12

Example 7.10 Write a program to generate a sawtooth waveform from V_{OUT}A pin.

The assembly program is as follows:

```
#include "D:\miniide\hc12.inc"
            org     $1000
            ldaa    #$FF
            staa    DDRP            ; configure PORTP for output
            ldaa    #$6F
            staa    DDRCAN          ; configure PCAN6..PCAN5 for output
            bclr    PORTCAN,$20 ; select VOUTA output
            clr      PORTP           ; initialize PORTP
loop        inc     PORTP           ; increase the output by one step
            bclr    PORTCAN,$40 ; generate a rising edge on the PORTCAN6 pin
            bset    PORTCAN,$40 ;                       "
            nop                     ; add NOP so that each step has 2 microsecond
            bra     loop            ; to complete the D/A conversion
            end
```

Homework #6

- See course website: http://390.revan.us
  - click homework tab
- Please submit a hard copy

Homework #5

Study for the Midterm Exam