CE320 : Microcomputers I

Lab6:
Output Device

Jung Me Park, Ph.D

ECE Department, Kettering University

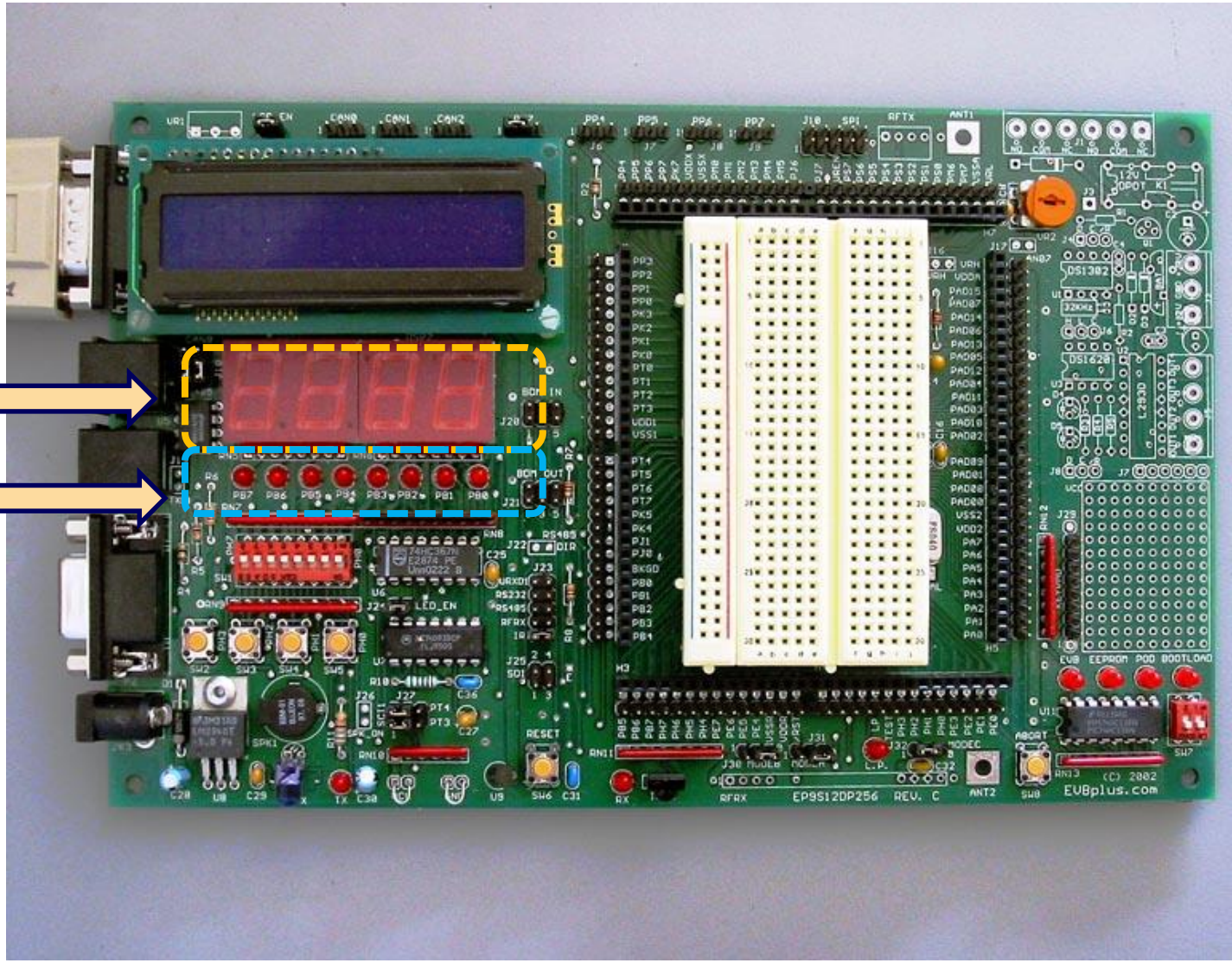
jpark@kettering.edu

Kettering UNIVERSITY | College of Engineering

Dragon 12 plus: Interfacing with LEDS

4 7-segment LEDs

8 Flashing LEDs



jpark@kettering.edu Kettering UNIVERSITY College of Engineering

Example Codes: 7 Segment Display

- Write an assembly program to **display the number 1234 on the four 7-segment LEDs**
  - 7-segment LEDs are mainly used to display **decimal digits** and a small set of letters.
  - To display 1234 on the four seven-segment LEDs

0	0	0	0	0	1	1	0

=$06

0	1	0	1	1	0	1	1

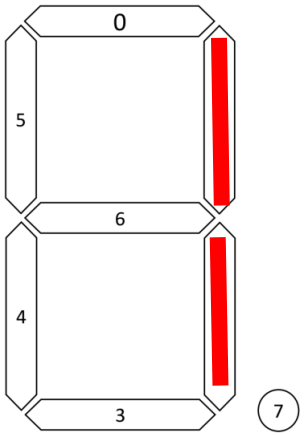
=$5B

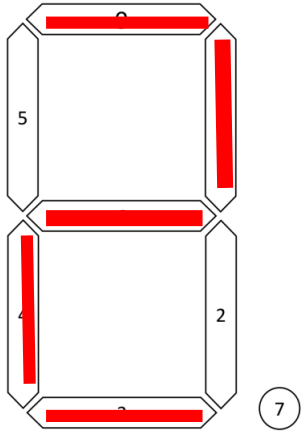
0	1	0	0	1	1	1	1

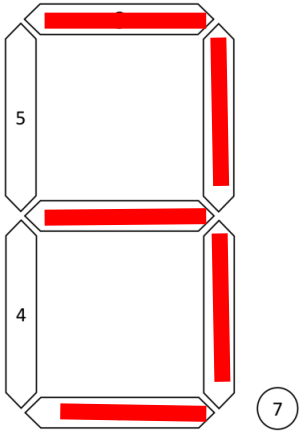
=$4F

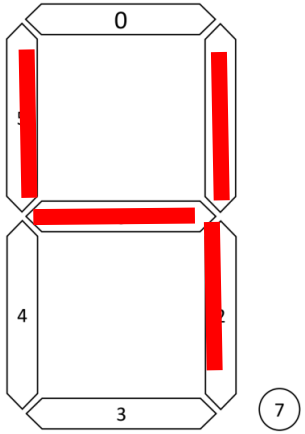
0	1	1	0	0	1	1	0

=$66





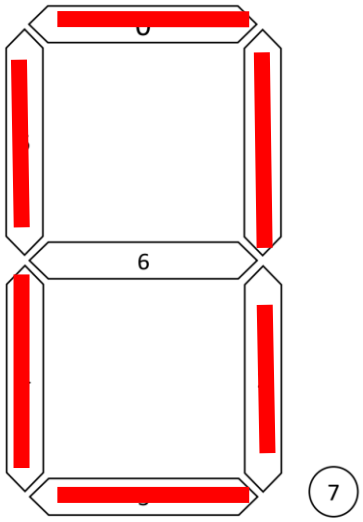


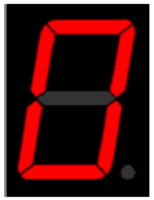


Interfacing with 7-Segments Display LEDs

- **8 Bits** in **Port B ($0001)** are used as a data for LEDs

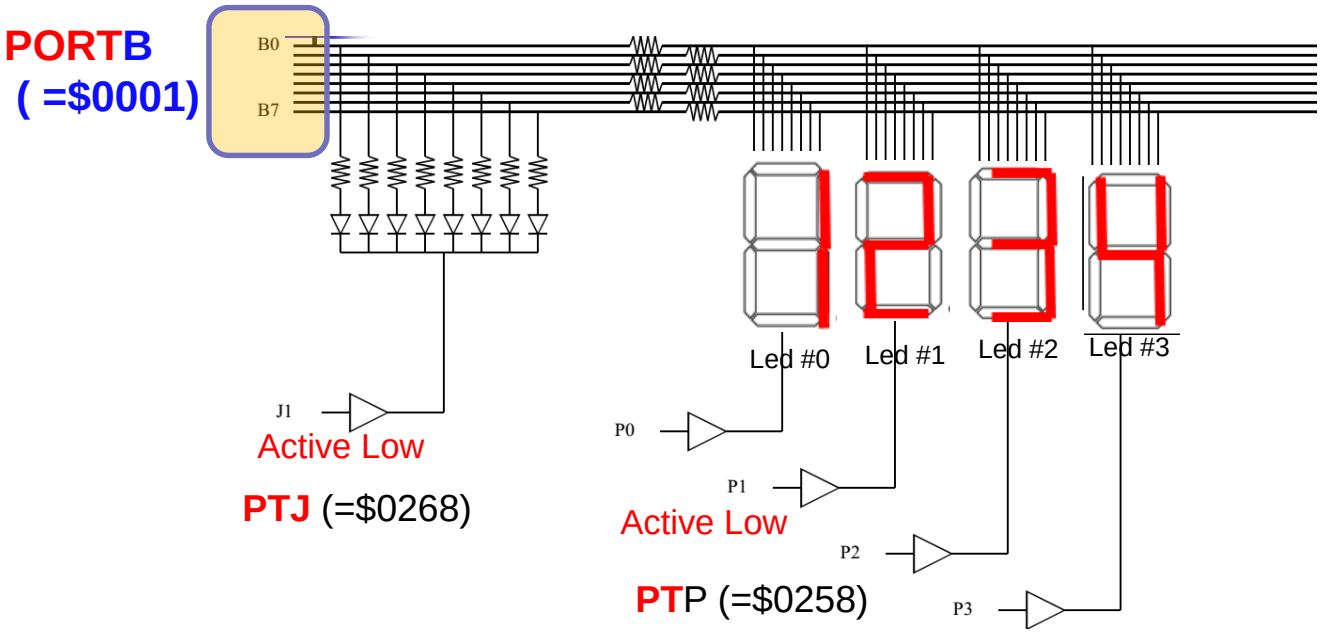
Decimal digit	Segments 6 5 4 3 2 1 0	Value to send to port B
0	0 1 1 1 1 1 1	$3F
1	0 0 0 0 1 1 0	$06
2	1 0 1 1 0 1 1	$5B
3	1 0 0 1 1 1 1	$4F
4	1 1 0 0 1 1 0	$66
5	1 1 0 1 1 0 1	$6D
6	1 1 1 1 1 0 1	$7D
7	0 0 0 0 1 1 1	$07
8	1 1 1 1 1 1 1	$7F
9	1 1 0 1 1 1 1	$6F





Example Code: 7 Segment LED Display

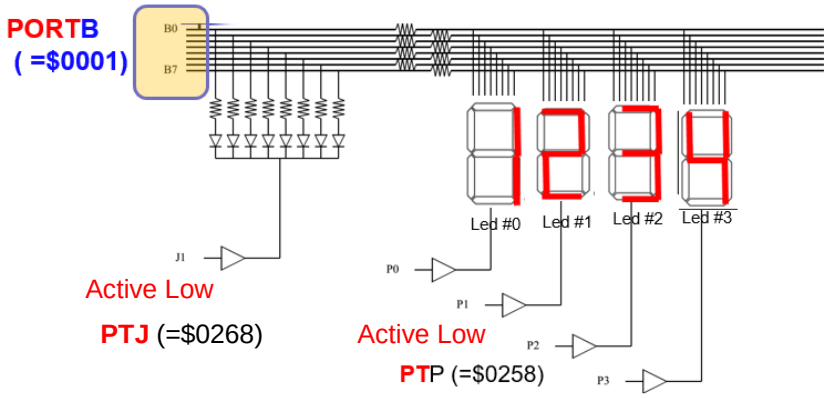
- Task: Write a program to display 1234 on the four seven-segment LEDs



Example Code: Data Declaration (I)

- Data declaration to display 1234 on the four seven-segment LEDs
- Need a set of data
  1. **PORT B data :** the data displayed on the selected LED

Decimal digit	Segments g	f	e	d	c	b	a	Value to send to port B
0	0	1	1	1	1	1	1	$3F
1	0	0	0	0	1	1	0	$06
2	1	0	1	1	0	1	1	$5B
3	1	0	0	1	1	1	1	$4F
4	1	1	0	0	1	1	0	$66
5	1	1	0	1	1	0	1	$6D
6	1	1	1	1	1	0	1	$7D
7	0	0	0	0	1	1	1	$07
8	1	1	1	1	1	1	1	$7F
9	1	1	0	1	1	1	1	$6F

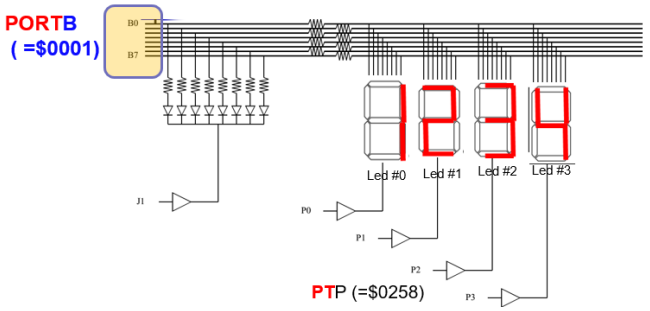


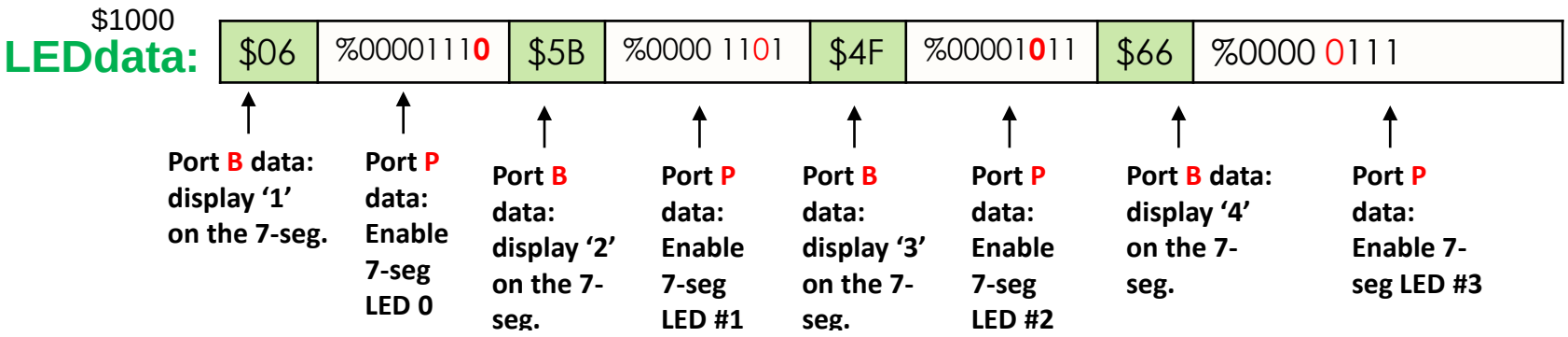
  2. **Port P data** : select the LED to display. Enable (active low) one 7-segment LED one at a time.

Example Code: Data Declaration (II)

- Data declaration to display 1234 on the four seven-segment LEDs
- Need a set of data
  - **PORT B data :** the data displayed on the selected LED
  - **Port P data :** Select the LED to display. Enable (active low) the selected LED

```
        ORG $1000
LEDdata: DC.b  $06,  %00001110
         DC.b  $5B,  %00001101
         DC.b  $4F,  %00001011
         DC.b  $66,  %00000111
```





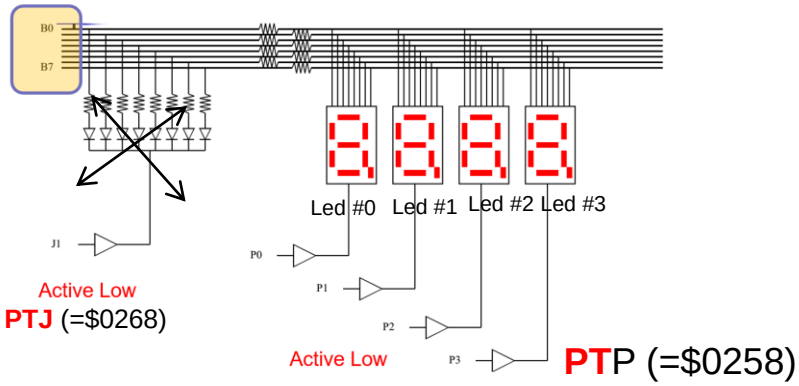
Example Codes: Configuration

2. Configure Port B , Port P and Port J as output ports.

```
BSET    DDRB,   %11111111      ;configure 8bits in Port B as output pins
BSET    DDRJ,   %0000 0010      ;configure  bit 1 in Port J  as a output pin
BSET    DDRP,   %0000 1111      ; configure lower 4 bits in Port P as output pins
```

3. Disabling PJ1 by setting the value 1

```
BSET    PTJ,    % 0000 0010     ; Disable flashing LEDs
```



LEDdata
= $1000

$1000	$1001	$1002	$1003	$1004	$1005	$1006	$1007
$06	%00001110	$5B	%0000 1101	$4F	%00001011	$66	%0000 0111

X

```
Forever:
        LDX   # LEDdata
Loop:
        MOVB   0,X,  PORTB
        INX
        MOVB   0,X,  PTP
        INX
        LDY   #10
        JSR   Delay_yms
        CPX  # LEDdata +8
        BNE   Loop
        BRA   Forever
```

1st iteration

X=$1000

PORT B= $06 (display digit ' 1' on the led)

X=$1001

PORT P= %0000 1110 ; LED#0 is activated

X=$1002

Y= 10 ms

Call subroutine for y ms delay

Compare X with the end of the LEDdata

If X is not the end of the LED Data, then go to Loop

LEDdata:
=$1000

$1000	$1001	$1002	$1003	$1004	$1005	$1006	$1007
$06	%00001110	$5B	%0000 1101	$4F	%00001011	$66	%0000 0111

Previous Iteration X=$1002

```
Loop:
        MOVB    0,X,  PORTB
        INX
        MOVB    0,X,  PTP
        INX
        LDY     #10
        JSR     Delay_yms
        CPX     # LEDdata +8
        BNE     Loop
```

2nd Iteration

PORT B= $5B (display digit ' 2' on the led)

X=$1003

PORT P= %0000 1101 ; LED#1 is activated

X=$1004

Y= 10 ms

Call subroutine for y ms delay

Compare X with the end of the LEDdata

If X is not the end of the LED Data, then go to Loop

LEDdata:

$1000	$1001	$1002	$1003	$1004	$1005	$1006	$1007
$06	%00001110	$5B	%0000 1101	$4F	%00001011	$66	%0000 0111

$1000

Previous Iteration X=$1004

Loop:	
MOVB 0,X, PORT**B**	PORT B= $4F (display '3' on the LED)
INX	X=$1005
MOVB 0,X, PT**P**	PORT P= %0000 1**0**11 ; LED#2 is activated
INX	X=$1006
LDY #10	Y= 10 ms
JSR Delay_yms	Call subroutine for y ms delay
CPX # **LEDdata** +8	Compare X with the end of the LEDdata
BNE Loop	If X is not the end of the LED Data, then go to Loop

3rd Iteration

LEDdata: ($1000)

$1000	$1001	$1002	$1003	$1004	$1005	$1006	$1007
$06	%00001110	$5B	%0000 1101	$4F	%00001011	$66	%0000 0111

Previous Iteration X=$1006

Forever:

```
        LDX   # LEDdata
Loop:
        MOVB    0,X,  PORTB
        INX
        MOVB    0,X,  PTP
        INX
        LDY   #10
        JSR   Delay_yms
        CPX  # LEDdata +8
        BNE   Loop
        BRA   Forever
```

4th Iteration
PORT B=$66 (display '4' on the LED)
X=$1007
PORT P= %0000 0111 ; LED#3 is activated
X=$1008
Y= 10 ms
Call subroutine for y ms delay
Compare X with the end of the LEDdata
X == $1008, so BNE is not taken.
Branch To Forever - Start again.

Stack application: Delay Program

- Clock speed of Dragon12+:
  - 24 MHz (24,000,000 Hz) means 24 million clock cycles / sec
  - 24,000 clock cycles / 1 ms
- The following instruction sequence creates a delay of 1 ms.

```
            LDX #1000

Innerloop:  PSHA          ; 2 E cycles
            PULA          ; 3 E cycles
            PSHA          ; 2 E cycles
            PULA          ; 3 E cycles
            PSHA          ; 2 E cycles
            PULA          ; 3 E cycles
            PSHA          ; 2 E cycles
            PULA          ; 3 E cycles
            nop           ; 1 E cycle

            DBNE  X, Innerloop   ; 3 E cycles
```

24 E clock cycles

= 24 * 1000
(24,000 cycles)
= 1 ms

Y ms Delay Program

- Given E-clock = 24MHz, to make y ms delay (approximately)

```
Delay_yms:                          ; subroutine to make a delay of Y ms
            PSHX                    ; save X register in the stack

Outerloop:
            LDX #1000

Innerloop:  PSHA                    ; 2 E cycles
            PULA                    ; 3 E cycles
            PSHA                    ; 2 E cycles
            PULA                    ; 3 E cycles
            PSHA                    ; 2 E cycles
            PULA                    ; 3 E cycles
            PSHA                    ; 2 E cycles
            PULA                    ; 3 E cycles
            nop                     ; 1 E cycle

            DBNE   X, Innerloop     ; 3 E cycles

            DBNE   Y, Outerloop
            PULX                    ; restore X register from the stack
            RTS
```

= 24 * 1000
(24,000 cycles)
= 1 ms