****

**BILKENT UNIVERSITY**

**CS224: COMPUTER ORGANISATION**

**LAB7 PRELIMINARY REPORT**

**BERAT BİÇER**

**21503050**

**SECTION 5**

TRIS Registers

* A TRIS bit set = 1 configures it as input
* A TRIS bit set = 0 configures it as output
* A read from TRIS reads the last value written to TRIS register
* On reset, all devices are set to input

PORT Registers

* A write to PORT register writes to the corresponding LAT register. If any pins configured as output, they are corrected to input.
* A write to a PORT register is the same as a write to a LAT register
* A read from a PORT register reads the value in the module (such as button matrix or leds) the I/O connection is connected to.

LAT Registers

* Holds data written to the module pins are connected to.
* A write to a LAT register transfers the data to the I/O device. Device set to output is updated to input.
* Read from a LAT register reads the data in LAT register, not from the I/O device.

SET, CLR, INV Registers

* Allows atomic bit manipulations
* Operates on the corresponding registers: if set is ‘1’, corresponding value is changed to user-specified value. If set is ‘0’, corresponding value is not changed.
* Ex: 0x0001 to TRISASET sets only bit 0 in TRISA; 0x0020 to PORTDCLR clears only bit 5 in PORTD; 0x9000 to LATCINV inverts only (bit 15 and bit 12) in LATC.
* To see the effect of the operation, corresponding register must be read.

Typical method to toggle an I/O pin requires read/modify/write operation performed in software. For example, a read from a PORTx register, mask and modify the desired bits and writing the value back to PORTx register.

A more efficient way uses PORTxINV register. A write to PORTxINV register effectively performs a read-modify-write operation on the target register, equivalent to the software operation above. However, it’s done in hardware. To toggle an I/O pin, a ‘1’ is written to the corresponding bit in PORTxINV register. This operation will read PORTx register; invert only the specified bits before as ‘1’ and write the resulting value to LATx register, thus toggling the corresponding I/O pins in a single instruction cycle: PORTAINV = 0x0001;

TRISx SET, CLR, INV Register Behavior

* A value written to a TRISxSET register reads the TRISx register, sets specified bits with a value ‘1’, writes modified value back to TRISx register.
* A value written to a TRISxCLR register reads the TRISx register, clears specified bits with a value ‘1’, writes modified value back to TRISx register.
* A value written to a TRISxINV register reads the TRISx register, inverts specified bits with a value ‘1’, writes modified value back to TRISx register.
* Bits specified with a value ‘0’ are not affected.

LATX SET, CLR, INV Register Behavior

* A value written to a PORTxSET register reads the PORTx register, sets specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* A value written to a PORTxCLR register reads the PORTx register, clears specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* A value written to a PORTxINV register reads the PORTx register, inverts specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* Bits specified with a value ‘0’ are not affected.

LATX SET, CLR, INV Register Behavior

* A value written to a LATxSET register reads the LATx register, sets specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* A value written to a LATxCLR register reads the LATx register, clears specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* A value written to a LATxINV register reads the LATx register, inverts specified bits with a value ‘1’, writes modified value back to LATx register. I/O pin is configured as input.
* Bits specified with a value ‘0’ are not affected.

ODC Registers

If the ODC bit for an I/O pin is ‘1’, that pin acts as an open-drain output. Else, it’s configured as a normal digital output. ODC bit is only valid for output pins. After a reset, all bits in ODCx register is set to ‘0’.

Open-drain feature allows generation of outputs higher than VDD on any desired digital only pins by using external-pull-up resistors. Maximum open-drain voltage allowed is the same as the maximum VIH specifications. ODC register settings takes effect in all I/O modes, allowing the output to behave as an open-drain even if a peripheral is controlling the pin. Although user could achieve the same result by manipulating the corresponding LAT and TRIS bits, this procedure will not allow the peripheral to operate in open-drain mode. Since I2C pins are already open-drain pins, ODCx settings don’t affect the I2C pins. Also, ODCx settings don’t affect the JTAG output characteristics as the JTAG scan cells are inserted between ODCx logic and I/O device.

// Rule of thumb: Always read inputs from PORTx and write outputs to LATx.

// If you need to read what you set an output to, read LATx.

void main(){

int value = 0b00000000;

JTAGEN\_bit = 0; // Mandatory

// Set port A as output, Port B as input

TRISA = 0xFF00;

TRISB = 0xFFFF;

// Initially every led is on

PORTASET = 0b11111111;

// Led shift

while(true){

Delay\_ms(250);

value += 1;

value = value << 1;

if(value == 0b00000000)

PORTASET = 0b11111111;

else if(PORTBbits.RF0 == 0)

PORTACLR = 0b11111111;

else if(PORTBbits.RF1 == 0){

PORTACLN = 0b11111111;

PORTASET = 0b10101010;

}

else if(PORTBbits.RF2 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b010101010;

}

else if(PORTBbits.RF3 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b10111101;

}

else if(PORTBbits.RF4 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b11101101;

}

else if(PORTBbits.RF5 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b10010001;

}

else if(PORTBbits.RF6 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b11111100;

}

else if(PORTBbits.RF7 == 0){

PORTACLR = 0b11111111;

PORTASET = 0b11100001;

}

else {

PORTACLR = 0b11111111;

PORTASET = value;

}

}

}