Lecture 8
Timers (part 2)

Embedded Systems

DCSE, UET Peshawar

Bilal Habib

Structure of a pin

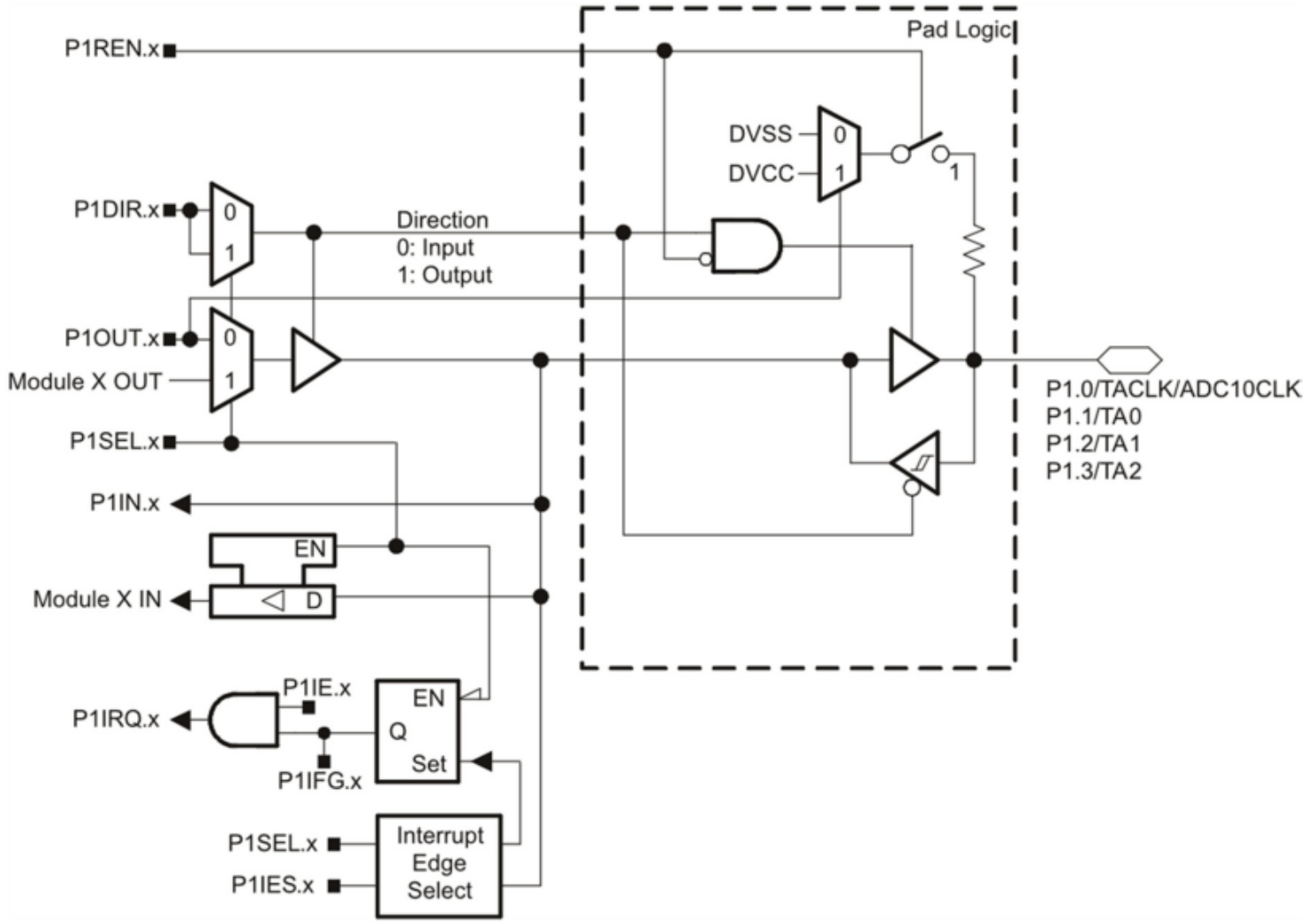


Fig. 8.6 Structure of pin P1.0 in an MSP430F2274 MCU (*Courtesy of Texas Instruments, Inc.*)

Port Configuration

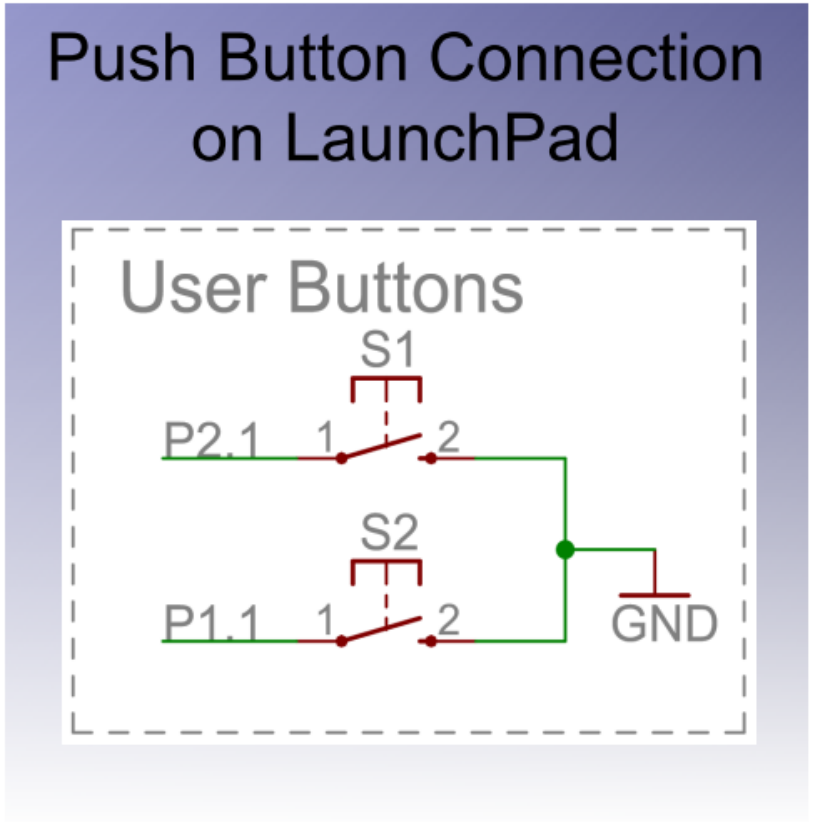
- **Port P1 input, P1IN**: reading returns the logical values on the inputs if they are configured for digital input/output. This register is read-only. It does not need to be initialized because its contents are determined by the external signals.
- **Port P1 output, P1OUT**: writing sends the value to be driven to each pin if it is configured as a digital output.
- **Port P1 direction, P1DIR**: clearing a bit to 0 configures a pin as an input, which is the default in most cases. Writing a 1 switches the pin to become an output.
- **Port P1 resistor enable, P1REN**: setting a bit to 1 activates a pull-up or pull-down resistor on a pin. Pull-ups are often used to connect a switch to an input.
- **Port P1 selection, P1SEL**: selects either digital input/output (0, default) or an alternative function (1). Further registers may be needed to choose the particular function.

Pin Configuration (contd..)

- **Port P1 interrupt enable, P1IE**: enables interrupts when the value on an input pin changes. This feature is activated by setting appropriate bits of P1IE to 1. Interrupts are off (0) by default.
- **Port P1 interrupt edge select, P1IES**: can generate interrupts either on a positive edge (0), when the input goes from low to high, or on a negative edge from high to low (1).
- **Port P1 interrupt flag, P1IFG**: a bit is set when the selected transition has been detected on the input. In addition, an interrupt is requested if it has been enabled. These bits can also be set by software, which provides a mechanism for generating a software interrupt (SWI).

Internal Pull-up

- The internal pull up resistor is a pull up/down resistor.
- It has to be enabled using PxREN
- It pulls the input towards the logic level of PxOUT for the same pin.
  - → Pull down if PxOUT is 0 (default)
  - → Pull up if PxOUT is 1



Output Mode

Output Modes

Table 7.5: Output unit modes *(Courtesy of Texas Instruments, Inc.)*

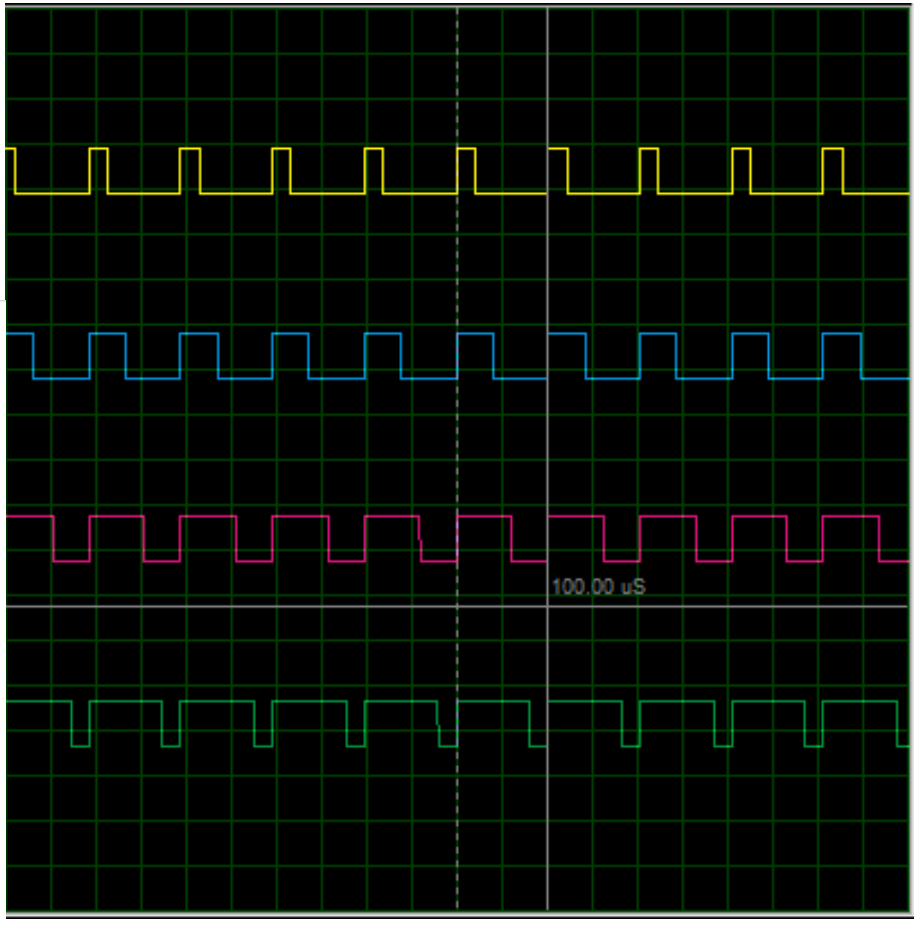
MODx	Mode	Description
000	Output	The output signal OUTx is defined by the OUTx bit. The OUTx signal updates immediately when OUTx is updated
001	Set	Output set when timer reaches the TACCRx value. It remains set until a timer reset, or until another mode affecting the output is selected
010	Toggle/Reset	The output is toggled when the timer counts to the TACCRx value. It is reset when the timer counts to the TACCR0 value
011	Set/Reset	The output is set when the timer counts to the TACCRx value. It is reset when the timer counts to the TACCR0 value
100	Toggle	The output is toggled when the timer counts to the TACCRx value. The output period is double the timer period
101	Reset	Output reset when timer reaches the TACCRx value. It remains reset until another output mode is selected and affects the output
110	Toggle/Set	The output is toggled when the timer counts to the TACCRx value. It is set when the timer counts to the TACCR0 value
111	Reset/Set	The output is reset when the timer counts to the TACCRx value. It is set when the timer counts to the TACCR0 value

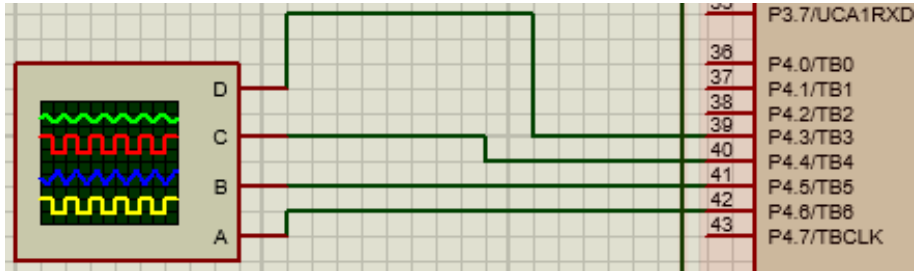
PWM Example: Output Modes

```
#include <MSP430F2418.h>
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer
    BCSCTL1 = CALBC1_1MHZ;         // Set range to calibrated 1MHz
    DCOCTL  = CALDCO_1MHZ;         // Set DCO step and modulation to calibrated 1MHz
    // Init PWM outputs: P4.{3-6} -> TB0.{3-6}
    // Try looking at these on an oscilloscope to see what the output looks like
    P4DIR |= 0x78;                 // make pins P4.{3-6} outputs
    P4SEL |= 0x78;                 // select module 1 of 3 (module 0 is GPIO)

    TB0CCR0 = 100;                 // (1 / 1) / 100 ticks = 10K Hz
    TB0CCR3 = 80;                  // 80 / 100 = 80% duty cycle
    TB0CCR4 = 60;                  // 60 / 100 = 60% duty cycle
    TB0CCR5 = 40;                  // 40 / 100 = 40% duty cycle
    TB0CCR6 = 20;                  // 20 / 100 = 20% duty cycle

    // set output mode to reset/set (see page 459 in user's guide - slau367f)
    TB0CCTL3 = TB0CCTL4 = TB0CCTL5 = TB0CCTL6 = OUTMOD_7;
    // clock source = SMCLK divided by 1, put timer in UP mode, clear timer register
    TB0CTL = TASSEL_2 | ID_0 | MC_1 | TBCLR;
    while(1)                                // Enter low power mode
        __bis_SR_register(LPM4_bits);    // SMCLK stays on in LPM3
}
```





Capture Mode

Capture Mode

```
#include <MSP430F2418.h> // Specific device
#include <stdint.h> // Integers of defined sizes
uint16_t last_time = 0; // Last time captured
uint16_t cap_diff, new_time=0;
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD ; // Stop watchdog timer
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;

    P1OUT = 0;
    P1DIR &=  ~BIT2; // P1.2 ,1 input , others output
    P1SEL |= BIT2; // P1.2 = S2 to Timer_A CCI0A

/* TA0CCTL1=Timer A Capture Control 1,  CM_3=both edges, CCIS_0= Capture input select: 0 - CCIxA*/
// SCS = Capture sychronize
// CAP = Capture Mode, CCIE = Capture/compare interrupt enable
TA0CCTL1 |= CM_3 | CCIS_0 | SCS | CAP | CCIE;
// Start timer: SMCLK , no prescale , continuous mode , no ints , clear
TA0CTL = TASSEL_2 | MC_2 | TACLR ;
for (;;) { // Loop forever with interrupts
    __bis_SR_register(LPM4_bits);   // send controller into low power mode
    }
return 0;
}
// -------------------------------------------------------------------------
// Interrupt service routine for TACCR0 .CCIFG , called on capture
// -------------------------------------------------------------------------
#pragma vector = TIMERA1_VECTOR
__interrupt void port_1 (void) // Flag cleared automatically
{
    new_time = TA0CCR1 ; // Save time for next capture
    cap_diff = new_time-last_time;
    last_time = new_time;
    TACCTL1 &= ~CCIFG;
   __bic_SR_register_on_exit(LPM4_bits);
}
```