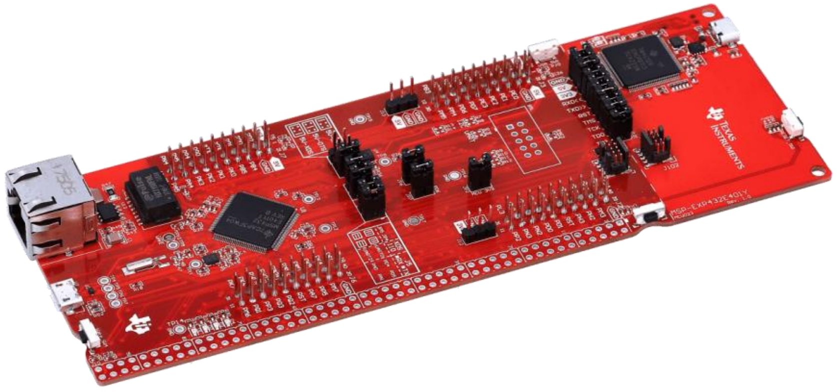
EE673: Experiment-4
Introduction to CCS and DSP Coding

Microcontrollers

- Control card and development is a printed circuit board with microcontroller mounted on them with few other hardware components
- Docking station provides power to the control card and has a bread-board area for prototyping
- Development kit has provision for providing power to the microcontroller
- Access to the key device signals are available using a series of header pins
- Board power can be provided by the provided USB cable or a 5V barrel supply

Docking station and control card



Development kit

Microcontroller

Code Composer Studio (CCS)

- Generally industries and academics uses microcontroller made by Texas instruments (TI)
- CCS – software which provides an interface that will support all TI microcontroller and microprocessor
- CCS provides a platform to edit our code
- If there is an error it allows us to debug

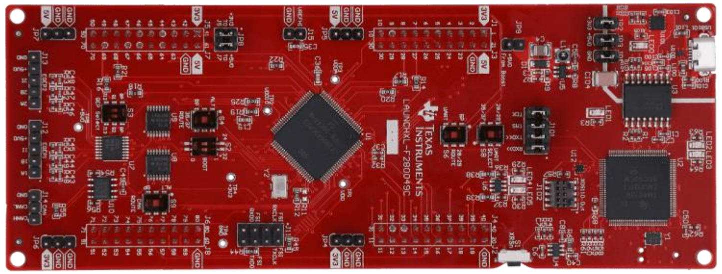
Control Suite:

- Used to enhance speed of CCS
- Compilation of few instructions which will help minimize system development time

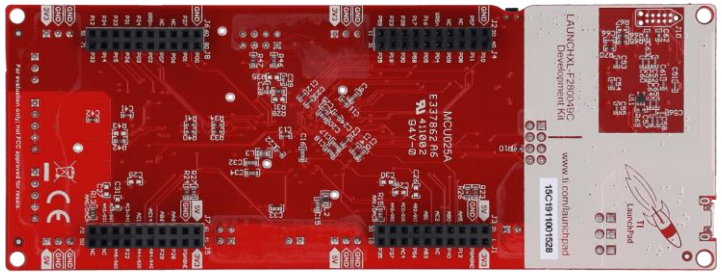
F280049C Launchpad

Details: https://www.ti.com/tool/LAUNCHXL-F280049C#tech-docs

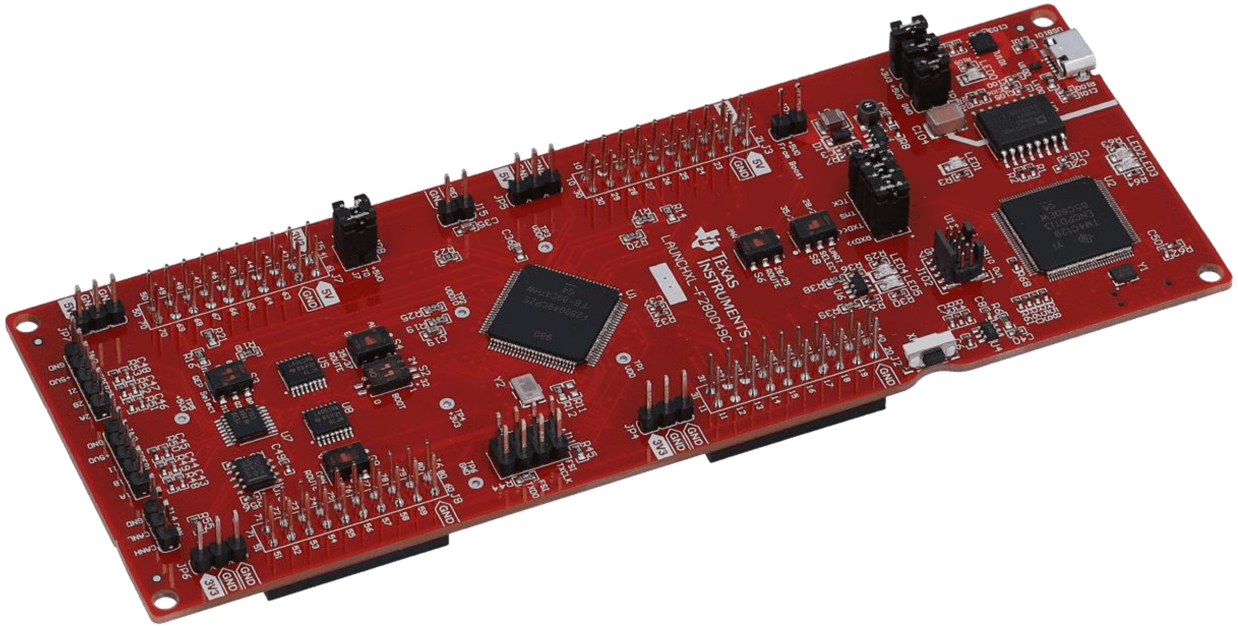
Technical reference manual: https://www.ti.com/lit/ug/sprui33f/sprui33f.pdf?ts=1692781443282



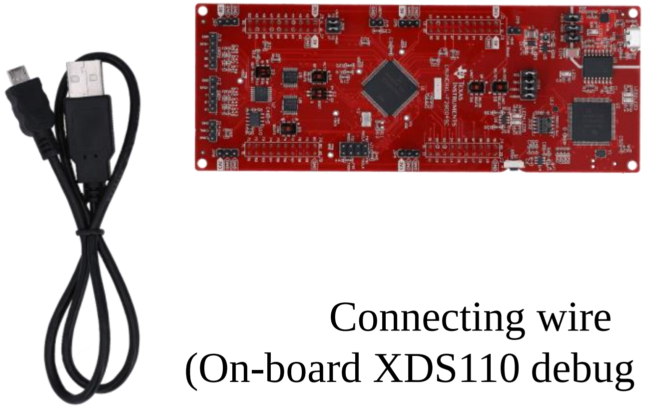
Front side



Back side

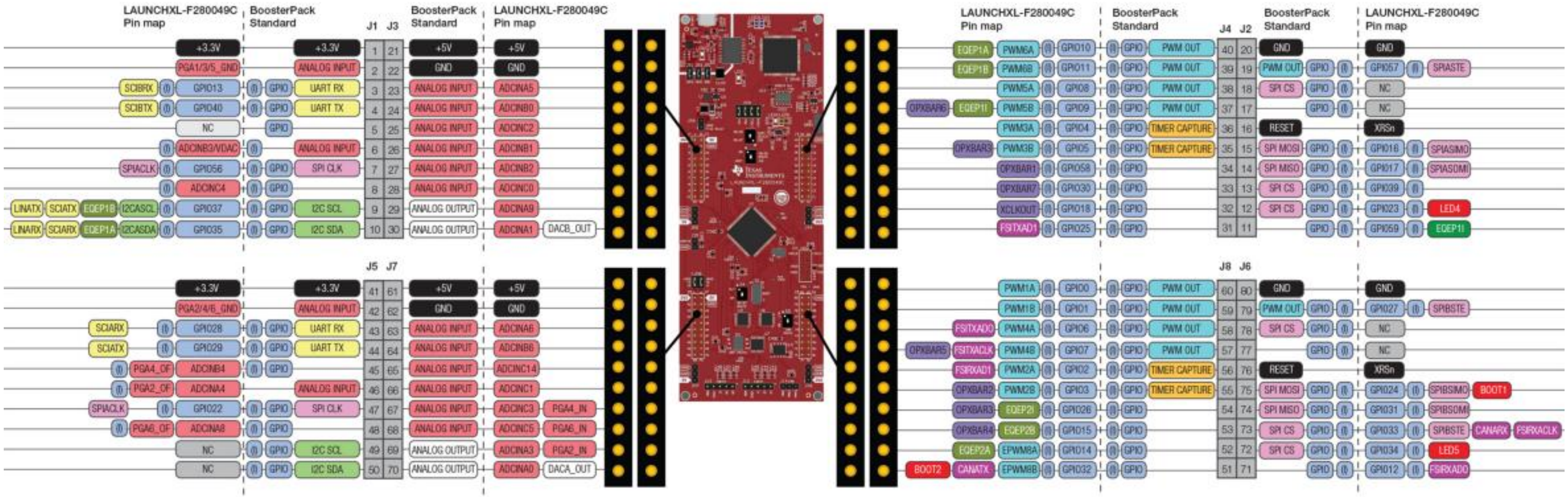


Development kit



Connecting wire
(On-board XDS110 debug probe)

F280049C Launchpad



F280049C Pin Map

GPIO Initialization

- GPIO module controls the device's digital and analog I/O multiplexing
- Device uses shared pins to maximize application flexibility
- Pins are named by their general-purpose I/O name (for example, GPIO0, GPIO25, GPIO58)
- These pins can be individually selected to operate as digital I/O (also called GPIO mode), or connected to one of several peripheral I/O signals
- Pin positions – datasheet (page 20)

GPIO Initialization

workspace_v12 - EE673_GPIO_toggle/EE673_GPIO_toggle_code.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Project Explorer

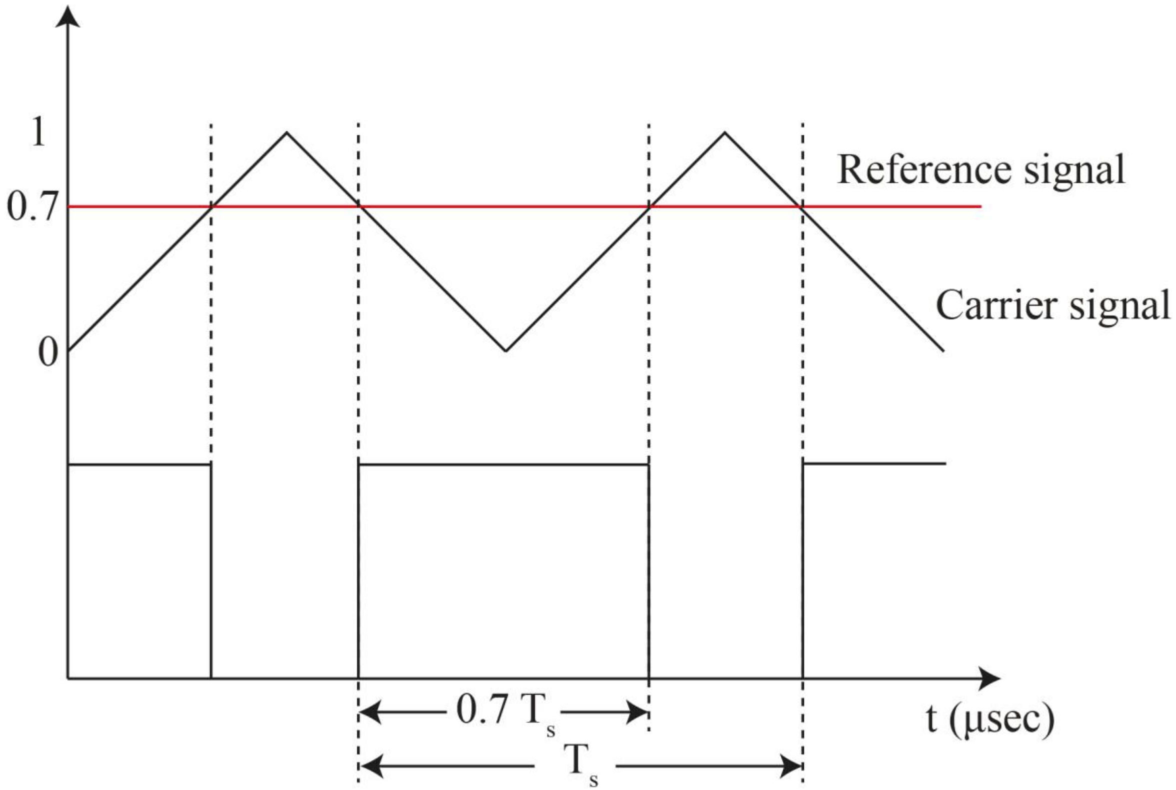
- EE673_GPIO_toggle [Active - CPU1_RAM]
- EE673_PWM_code

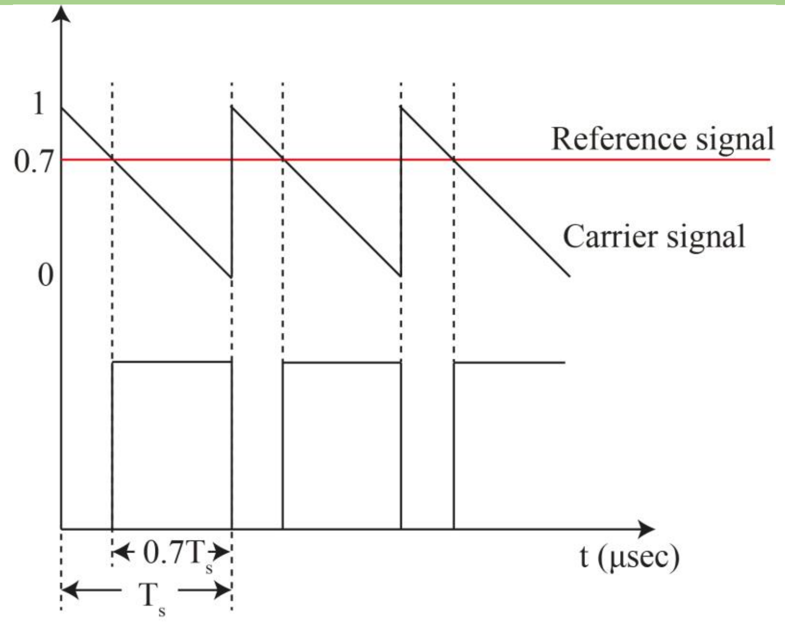
EE673_GPIO_toggle_code.c | EE673_PWM_code.c

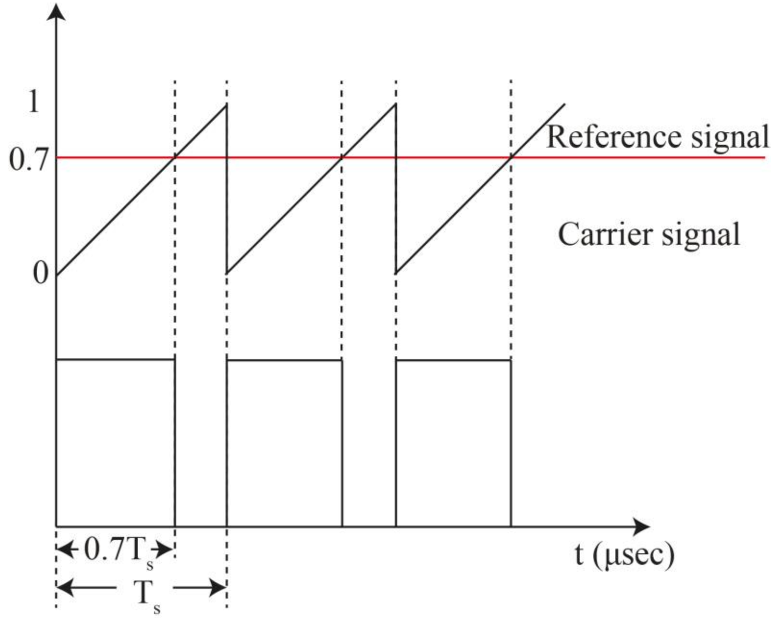
```c
#include "F28x_Project.h"
// user defined functions
void init_gpio(void);
// Main
void main(void)
{

    InitSysCtrl();// Initialize device clock and peripherals

    InitGpio(); // Initialize GPIO

    init_gpio();

    while(1)
    {
        // toggle the LED
        GpioDataRegs.GPATOGGLE.bit.GPIO23 = 1;       // toggle LED4
        F28x_usDelay(10000000);

    }
}

void init_gpio(void)
{
    // initialize the LED
    EALLOW;
    GpioCtrlRegs.GPADIR.bit.GPIO23 = 1;          // make the pin (GPIO23) as output pin
    GpioDataRegs.GPACLEAR.bit.GPIO23 = 1;            // initially in off condition
//    GpioCtrlRegs.GPAMUX2.bit.GPIO23 = 0;       // only one function of this pin no need to define
    EDIS;

}
```

Datasheet: Page No. 20
TRM: Page No. 904, 905, 906, 964, 969, 971

PWM Generation







Sample code

workspace_v12 - EE673_PWM_code/EE673_PWM_code.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Project Explorer
- EE673_GPIO_toggle
- **EE673_PWM_code** [Active - CPU1_RAM]

*EE673_PWM_code.c | *EE673_GPIO_toggle_code.c

Initialization part

```c
// Included Files
#include "f28x_project.h"

void init_gpio(void);
void init_epwm(void);

void main(void)
{
    // Initialize device clock and peripherals
    InitSysCtrl();
    // Initialize GPIO
    InitGpio();
    //initialize flash
    InitFlash();

    init_gpio();
    init_epwm();
}
//gipo initialization code
void init_gpio(void)
{
    EALLOW;
    //setup EPWM PINS
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;   // Configure GPIO0 as EPWM1A

    EDIS;
}

void init_epwm(void)
{
    EALLOW;

    EPwm1Regs.TBCTL.bit.CTRMODE = 3;     //stop the counter (Freeze)
    EPwm1Regs.TBCTL.bit.CLKDIV = 0;      // div by 1 prescaler  =100MHZ
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;      // div by 1 prescaler = 100MHZ
     EPwm1Regs.TBCTR = 0;                   //reset counter value
     EPwm1Regs.TBPRD = 5000; //time period
     EPwm1Regs.CMPA.bit.CMPA = 2500 ;      //set duty cycle

     //action qualifier group
     EPwm1Regs.AQCTLA.bit.ZRO = 0;    //do nothing to EPWM1A when TBCTR == 0
     EPwm1Regs.AQCTLA.bit.CAU = 2; //When TBCTR = CMPA on Up Count--EPWM1A force high
     EPwm1Regs.AQCTLA.bit.CAD = 1; // When TBCTR = CMPA on Down Count --EPWM1A force low
     EPwm1Regs.TBCTL.bit.CTRMODE = 2;     //start counter

     EDIS;
}
```

GPIO initialization part

PWM Generation part

Datasheet: Page No. 20
TRM Page No. 1834, 1837, 1847, 1853

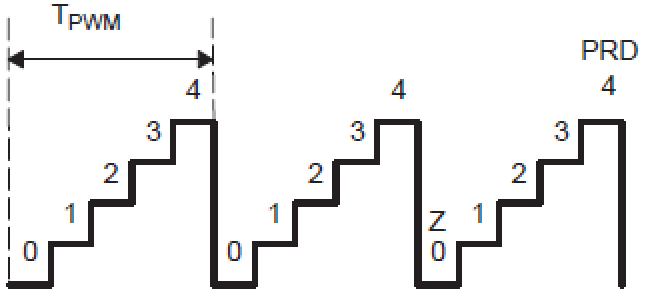
Enhanced Pulse Width Modulation (ePWM)

- Triangular waveform generation and dc waveform generation is done by ePWM module
- One ePWM module will have two signals → EPWMxA, EPWMxB
- 7 sub-modules:

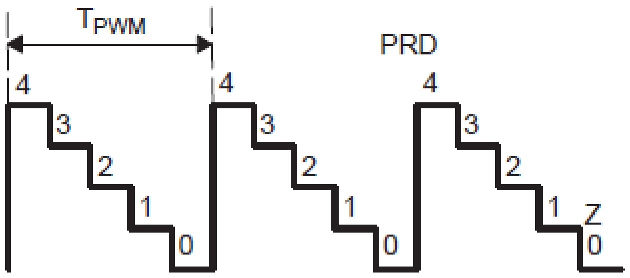
1. Time-base (TB)
2. Counter Compare (CC)
3. Action Qualifier (AQ)
4. Dead-band (DB) generator
5. PWM – chopper (PC)
6. Trip-zone (TZ)
7. Event trigger and interrupt (ET)

Time-Base Sub-module

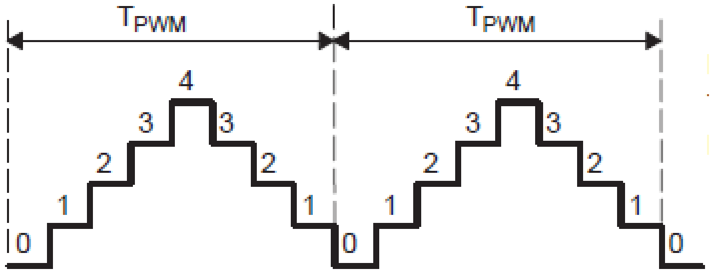
- Generates triangular waveform
- In DSP, everything is discrete. Triangular waveforms generated in steps
- Registers used for triangular waveform generation:
  - TBCTR: Time-base counter
  - TBPRD: Time-base period register
  - TBCTL: Time-base control register
    - HSCLKDIV, CLKDIV, CTRMODE, SYNCOSEL
  - TBPHS: Time-Base Phase Register
- frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter



Up Count Mode



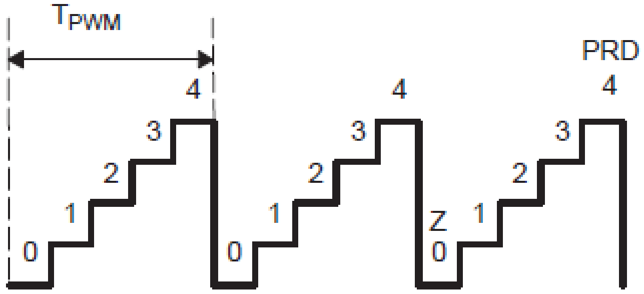
Down Count Mode



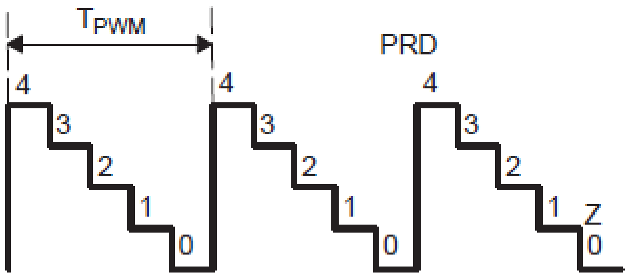
Up Down Count Mode

Time-Base Sub-module

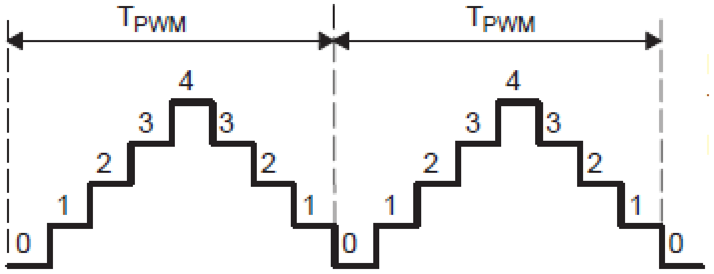
- Time increment for each step is defined by the time-base clock (TBCLK)
- TBCTR increments its value from 0 up to the value is stored in TBPRD
- Microcontroller operates at a particular crystal frequency and corresponding time period is stored in SYSCLKOUT
- TBCLK - Time-base clock (determines the rate at which time-base counter increments or decrements)
- $TBCLK = \frac{SYSCLKOUT}{HSCLKDIV * CLKDIV}$
- Values could be added to these two registers to vary TBCLK



Up Count Mode

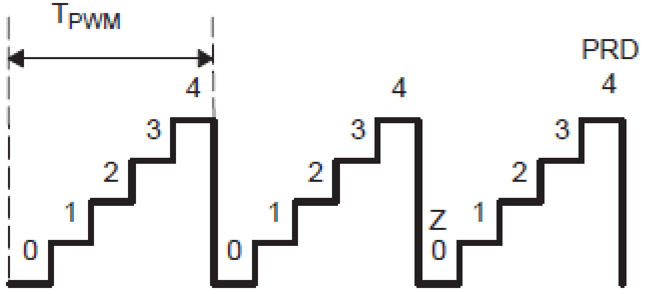


Down Count Mode

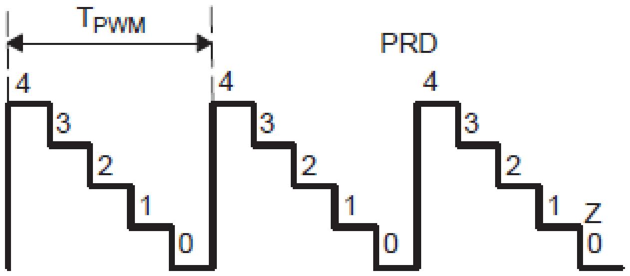


Up Down Count Mode

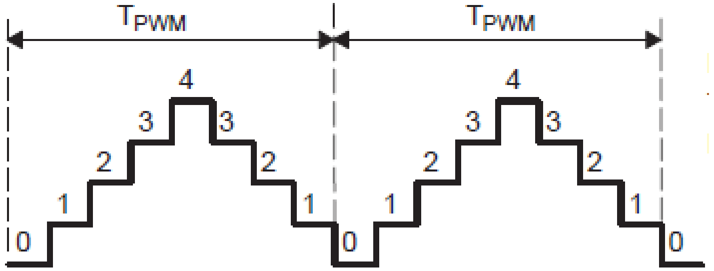
Time-Base Sub-module



Up Count Mode



Down Count Mode



Up Down Count Mode

$$T_{PWM} = (TBPRD + 1) * T_{TBCLK}$$
$$F_{PWM} = 1/T_{PWM}$$

$$T_{PWM} = (TBPRD + 1) * T_{TBCLK}$$
$$F_{PWM} = 1/T_{PWM}$$

$$T_{PWM} = 2 * TBPRD * T_{TBCLK}$$
$$F_{PWM} = 1/T_{PWM}$$

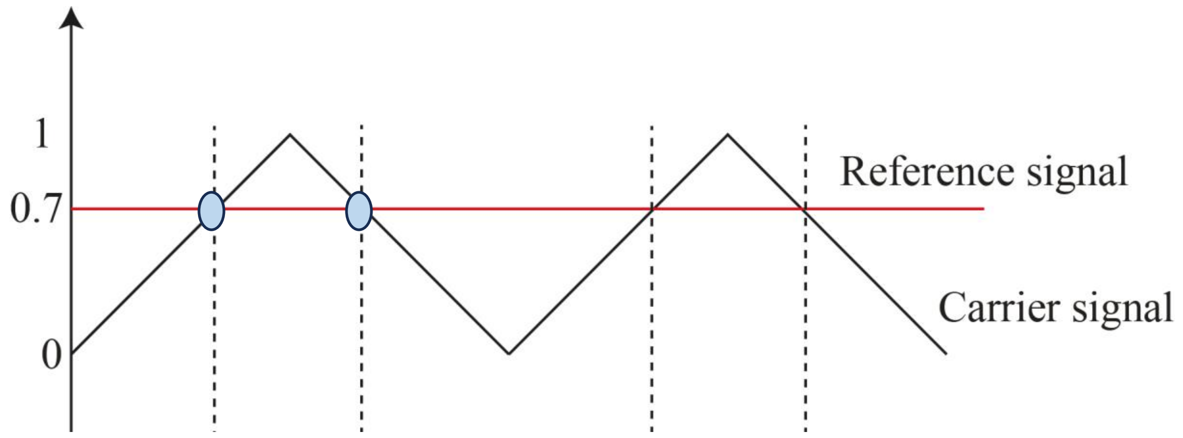
- Counting begins from zero to the value stored in TBPRD register and is repeated
- T_{PWM} – Time period of PWM or time required for repetition
- With same PRD value – higher switching frequency is achieved using up & down counter

Time-Base Sub-module

- TBCTL: Control register of time-base sub-module
- CTRMODE – counter mode: helps to select between 3 counter modes
- SYNCOSEL: helps maintain same phase for different carrier waves generated
- TBPHS – Time-base phase register: used to add phase difference between carrier waves
- HSPCLKDIV – High speed prescalar clock division
- CLKDIV – Clock Division
- HSPCLKDIV & CLKDIV – 3 bit registers. Values could be added to this to vary TBCLK

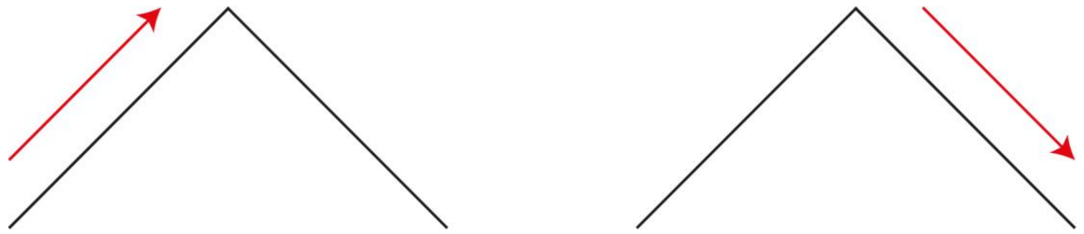
Counter - Compare Sub-module

- Used for reference signal generation
- CMPA: Compare for EPWMxA
- CMPB: Compare for EPWMxB
- The value in the active CMP register is continuously compared to the time-base counter (TBCTR)
- When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare" event.
- This event is sent to the action-qualifier submodule



Action-Qualifier Sub-module

- Whenever event is created, action qualifier (AQ) will decide what action to take and when to take action
- Event is created when counter value is equal to zero or PRD
- Event is also generated when CMPA or CMPB values equals some values of triangular waveform



- Event is generated when counter is up counting or down counting
- CMPA can compare itself with counter associated with ePWMxA and ePWMxB

Action-Qualifier Sub-module

- AQCTLA: Action-Qualifier Output A Control Register
  - When to take action:
    - CBD: time-base counter equals the active CMPB register while decrementing
    - CBU: time-base counter equals the active CMPB register while incrementing
    - CAD: time-base counter equals the active CMPA register while decrementing
    - CAU: time-base counter equals the active CMPA register while incrementing
    - PRD: time-base counter equals the period value
    - ZRO: time-base counter equals the period value
  - What action?
    - 00: Do nothing
    - 01: Force ePWMxA output low
    - 10: Force ePWMxA output high
    - 11: Toggle EPWMxA output

Figure 4-10. Action-Qualifier Output A Control Register (AQCTLA)

15–12	11–10	9–8
Reserved	CBD	CBU
R-0	R/W-0	R/W-0

7–6	5–4	3–2	1–0
CAD	CAU	PRD	ZRO
R/W-0	R/W-0	R/W-0	R/W-0

Sample code

workspace_v12 - EE673_PWM_code/EE673_PWM_code.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Project Explorer
- EE673_GPIO_toggle
- **EE673_PWM_code** [Active - CPU1_RAM]

*EE673_PWM_code.c | *EE673_GPIO_toggle_code.c

Initialization part

```c
// Included Files
#include "f28x_project.h"

void init_gpio(void);
void init_epwm(void);

void main(void)
{
    // Initialize device clock and peripherals
    InitSysCtrl();
    // Initialize GPIO
    InitGpio();
    //initialize flash
    InitFlash();

    init_gpio();
    init_epwm();
}
```

GPIO initialization part

TRM: Page No. 904, 905

```c
//gipo initialization code
void init_gpio(void)
{
    EALLOW;
    //setup EPWM PINS
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;   // Configure GPIO0 as EPWM1A

    EDIS;
}
```

PWM Generation part

```c
void init_epwm(void)
{
    EALLOW;

    EPwm1Regs.TBCTL.bit.CTRMODE = 3;    //stop the counter (Freeze)
    EPwm1Regs.TBCTL.bit.CLKDIV = 0;     // div by 1 prescaler  =100MHZ
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;     // div by 1 prescaler = 100MHZ
     EPwm1Regs.TBCTR = 0;                  //reset counter value
     EPwm1Regs.TBPRD = 5000; //time period
     EPwm1Regs.CMPA.bit.CMPA = 2500 ;      //set duty cycle

     //action qualifier group
     EPwm1Regs.AQCTLA.bit.ZRO = 0;    //do nothing to EPWM1A when TBCTR == 0
     EPwm1Regs.AQCTLA.bit.CAU = 2; //When TBCTR = CMPA on Up Count--EPWM1A force high
     EPwm1Regs.AQCTLA.bit.CAD = 1; // When TBCTR = CMPA on Down Count --EPWM1A force low
     EPwm1Regs.TBCTL.bit.CTRMODE = 2;     //start counter

     EDIS;
}
```

Thank You