# 마이크로컨트롤러 기초(#514112 )

# #.7 Basic Timer1 기초

한림대학교
전자공학과 이선우

# Contents

- Digital Counter Basics
- MSP430x4xx Timers
  - Overview
  - Basic Timer 1
  - Example program

# Digital Counter
# &
# Timer

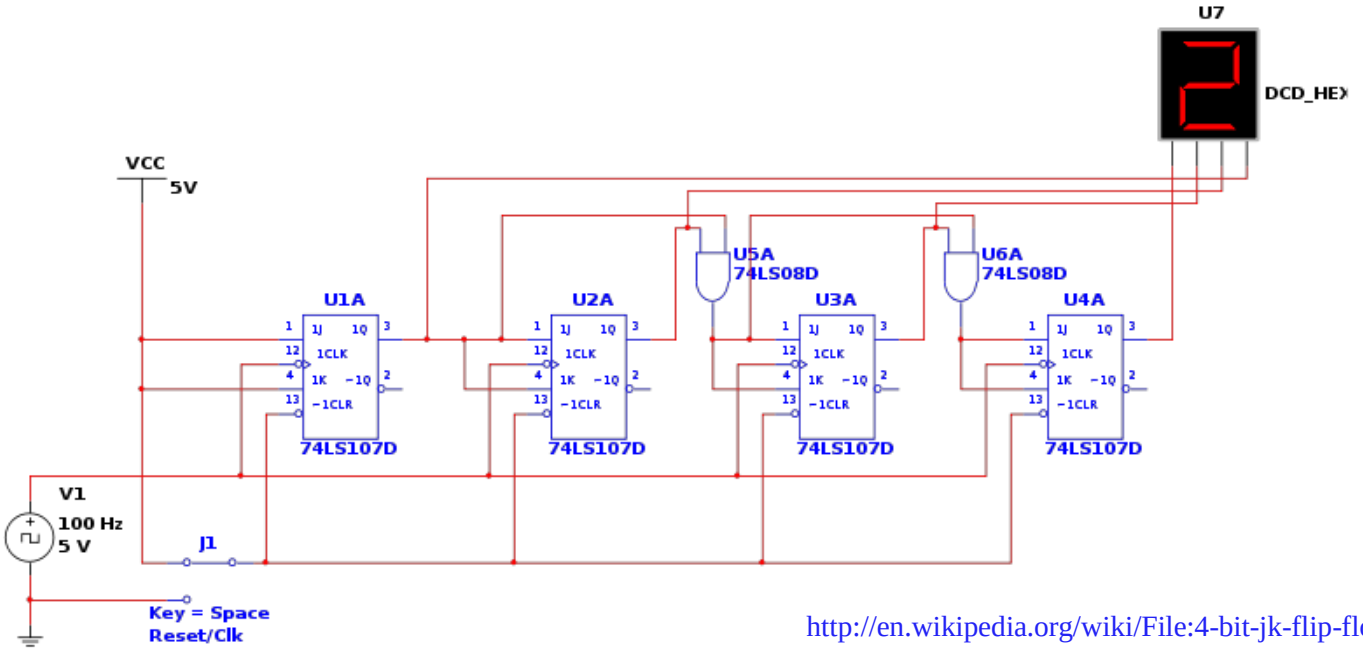
# Counter Basics

- Digital counter (계수기)
  - A device which stores the number of times a particular event or process has occurred. (http://en.wikipedia.org/wiki/Digital_counter)
  - 클럭 펄스를 세어서 수치를 처리하기 위한 논리 회로 (디지털 회로) (위키피디아 한글)
  - 작동 원리
    - 기계식 카운터 입력 (스위치 누름) = 클럭 펄스 (rising or falling edge)
    - 자리 수 = 카운터 레지스터 크기
      8bit counter의 셀 수 있는 범위: 0~255,
      10 bit → 0~ 1023 (1024개)


http://en.wikipedia.org/wiki/File:CountersMechanical.agr.jpg

# Digital counter 종류

- Asynchronous (ripple) counter- 변화하는 상태 비트가 다음 단의 클럭 입력으로 사용
- Synchronous counter-모든 상태 비트가 하나의 클럭 신호에 의해 변화함.
- Up-down counter-증가/감소 양 방향으로 셀 수 있는 카운터




http://en.wikipedia.org/wiki/File:4-bit-jk-flip-flop_V1.1.svg

# Timer (타이머)

- MCU의 기본적인 주변기기, 따라서 모든 MCU가 내장
- 기본 회로는 클럭 신호를 카운트하는 카운터 (counter), 즉, 입력 클럭에 따라 내부 레지스터의 값을 증가/감소.
- 이러한 카운터에 정확한 주기의 클럭 신호를 입력하면, 필요한 시간 간격을 측정할 수 있다.➔Timer
  - 예:1Khz clock (한 주기 시간=1msec), 1000개 카운트 ➔1sec
- 외부 신호 혹은 이벤트의 발생 개수를 카운트할 필요가 있는 경우, 원래의 카운터로 이용 가능
- 타이머(카운터)의 사양(능력)
  - 셀 수 있는 개수(표시 가능한 수의 범위) → 저장할 수 있는 비트의 개수
  - Ex. 8-bit timer → 0~255 (256개),
    16-bit timer→0~65535(64K=64*1024),
    32bit-timer→0~4G(4*1024*1024개)

http://en.wikipedia.org/wiki/File:Lux_Products_Long_Ring_Timer.jpg

# MSP430
# Basic Timer1
# (ch. 13)

# MSP430x4xx 타이머 종류

- MSP430x4xx series는 다음과 같은 3종류의 타이머 내장
  - Basic Timer1
    - Two independent, cascadable 8-bit timers
    - Selectable clock source
    - Interrupt capability
    - LCD control signal generation
  - Timer A, Timer B
    - 16-bit timer/counter with 3 or 5 capture/compare registers
    - Multiple capture/compares, PWM outputs, interval timing

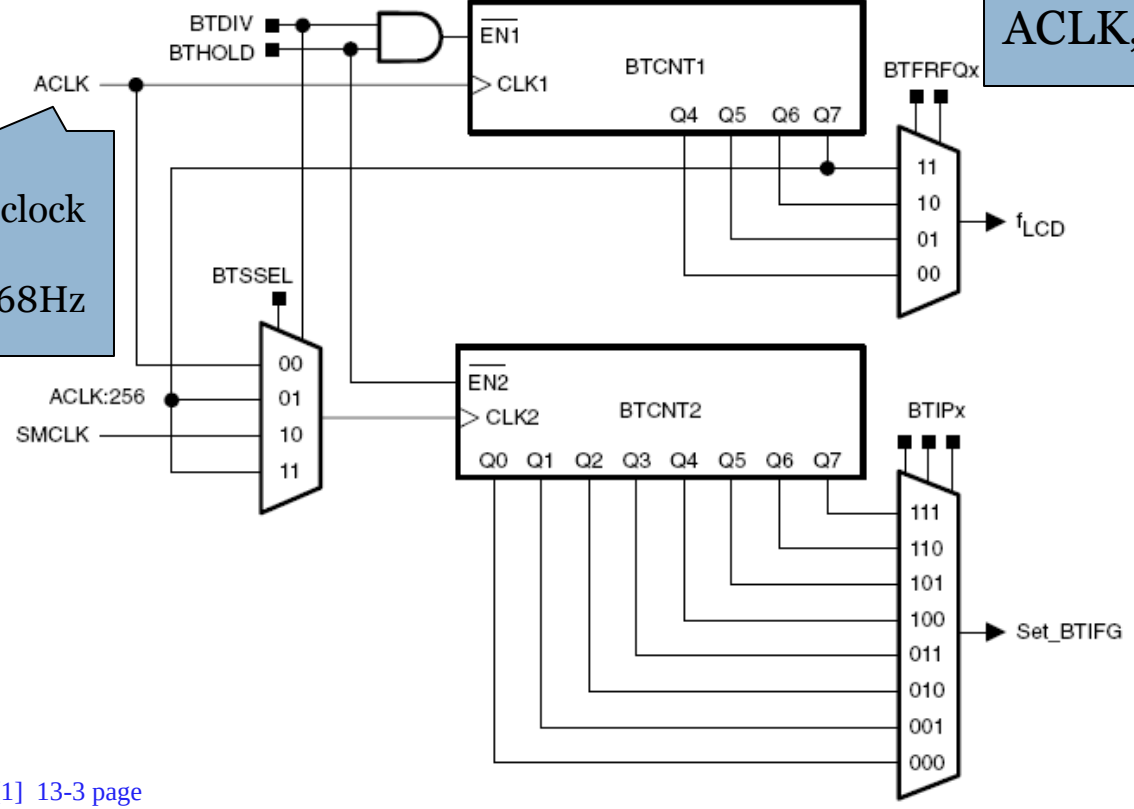
# Basic Timer 1

- Block diagram

2개의 별도 카운터 (1&2)
• BTCNT1: incremented with ACLK, LCD controller의 다이나믹 구동을 위한 주파수를 제공함.
• BTCNT2: source ACLK,SMCLK, ACLK/256

ACLK: auxiliary clock → 보드:32768Hz

Figure 13–1. Basic Timer1 Block Diagram




[1] 13-3 page

# Clock signals in MCU

- 모든 Sequential digital logic devices(순차 논리 회로 장치)가 동작하기 위해서는 clock 신호가 필요. → 이 신호에 의해 모든 내부 논리 회로들이 동기화되어 동작함 (synchronized)
- MCU = typical sequential logic device
  - 모든 내부 장치가 클럭 신호에 의해 동기화
  - 만약 클럭 신호가 멈추면 MCU의 동작도 멈춤 (sleep/suspend/low-power mode)
- 클럭 신호 발생 방법
  - 대부분의 경우: 외부 crystal oscillator + 내부 발진/증폭 회로
    - 장점: 정확한 주기 확보, 단점: 고가
  - 내부 RC oscillator 회로 이용: 저가(장점), 부정확한 주기(단점)

# Clocks in MSP430

- 대개의 MCU의 경우 2개 이상의 클럭 신호 이용
  - Main clock
    - CPU와 내부 관련 레지스터 동작 용, 따라서 매우 빠름
    - 외부 oscillator의 주파수를 내부 발진 회로를 이용하여 배증(X)하여 사용하는 것이 일반적.
    - MCLK in MSP430FG4618
      - 내부 clock module(ch.5) 회로 내장
      - 외부 Osc. (=32768Hz) * x = MCLK (master clock)
      - Default: $f_{crystal}$ * 32, 타겟보드: 32768*32 =1048576Hz
  - Peripheral (auxiliary) clock : 주변 장치 동작용으로 대개 Main clock에 비해 속도가 느리나 더 빠른 경우도 있음.
    - MSP430 경우
      → ACLK (auxiliary clock): 32.768KHz for real-time clock module
      →SMCLK (sub-main clock): MCLK과 동일

# 타이머와 클럭 관계

- MCU 내장 타이머 (=카운터)
  - 32768Hz clock을 몇 개 세느냐를 이용하여 원하는 시간 간격(interval)을 결정
  - BT1 이용 경우:
    - BTCNT1 (8bit)
      - ACLK 입력에 따라 0~255까지 표시 가능 (각 비트는 Qx로 표시)
      - Q7의 경우 주기=ACLK/256인 클럭 신호가 됨. (왜??)
      - Overflow: 현재 값 255 + 1 (클럭 입력)→0 (reset)
    - BTCNT2 (8bit)
      - 3가지 종류의 클럭 소스: ACLK, ACLK/256 (by using BTCNT1), SMCLK
      - 입력되는 클럭에 따라 0,1,2, …, 254, 255, 0, 1, 2, …로 BNCNT2 값 변화
      - 각 자리 수 비트가 rising/falling edge 발생할 때 인터럽트 발생시킴.
      - ➔ 즉, 입력 클럭($f_{CLK2}$)을 2, 4, 8, 16, …, 128, 256 개를 셀 때의 인터벌을 측정(발생)할 수 있음.

# 기본적인 이용 방법

- BT1의 경우 일반적인 이용방법
  - LCD 컨트롤러용 클럭 생성 (MSP430FG4618은 해당 없음)
  - 기본적인 시간 간격 타이머로 이용
- Ex.
  - 타겟보드 default 설정: ACLK=32768Hz, SMCLK=ACLK*32=1048576Hz (≈1MHz)
  - 긴 간격 생성은?
    - ACLK/256=128Hz
    - 128Hz/2,4,8,16,31,64,128,256
      → 64Hz, 32Hz, 16Hz,8Hz,4Hz,2Hz,1Hz, 0.5Hz 시간간격마다 인터럽트 발생 (이를 이용)
  - 짧은 시간 간격은?
    - ACLK을 선택하면 위보다 256배 짧은 간격 생성 가능
  - 더 짧은 간격은? SMCLK 사용함!

# 타이머 활용 방법 (개념 이해)

- Task
  - 0.5초의 인터벌을 만들어(관측하여) 이 시간 간격으로 LED를 깜박이자.
- 요구사항
  - 0.5초의 인터벌 관측
  - LED 1개 ON/OFF
- 타겟 보드에서는..
  - ACLK=32768Hz 이므로 T=1/32768 sec, 따라서 이 클럭을 16384개를 세면 0.5초가 됨. (카운트값=$f_{CLK2}/f_{desired}$)
  - 결정한 인터벌 마다 LED를 toggle시키면 됨.

# BT1 Control Registers

*Table 13–1. Basic Timer1 Registers*

| Register | Short Form | Register Type | Address | Initial State |
|---|---|---|---|---|
| Basic Timer1 Control | BTCTL | Read/write | 040h | Unchanged |
| Basic Timer1 Counter 1 | BTCNT1 | Read/write | 046h | Unchanged |
| Basic Timer1 Counter 2 | BTCNT2 | Read/write | 047h | Unchanged |
| SFR interrupt enable register 2 | IE2 | Read/write | 001h | Reset with PUC |
| SFR interrupt flag register 2 | IFG2 | Read/write | 003h | Reset with PUC |

**BTCTL, Basic Timer1 Control Register**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BTSSEL | BTHOLD | BTDIV | BTFRFQx | | BTIPx | | |
| rw | rw | rw | rw | rw | rw | rw | rw |

**IE2, Interrupt Enable Register 2**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BTIE | | | | | | | |
| rw–0 | | | | | | | |

**IFG2, Interrupt Flag Register 2**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BTIFG | | | | | | | |
| rw–0 | | | | | | | |

**BTSSEL** Bit 7 BTCNT2 clock select. This bit, together with the BTDIV bit, selects the clock source for BTCNT2. See the description for BTDIV.

**BTHOLD** Bit 6 Basic Timer1 Hold.
0 BTCNT1 and BTCNT2 are operational
1 BTCNT1 is held if BTDIV=1
BTCNT2 is held

**BTDIV** Bit 5 Basic Timer1 clock divide. This bit together with the BTSSEL bit, selects the clock source for BTCNT2.

| BTSSEL | BTDIV | BTCNT2 Clock Source |
|---|---|---|
| 0 | 0 | ACLK |
| 0 | 1 | ACLK/256 |
| 1 | 0 | SMCLK |
| 1 | 1 | ACLK/256 |

**BTFRFQx** Bits 4–3 $f_{LCD}$ frequency. These bits control the LCD update frequency.
00 $f_{ACLK}/32$
01 $f_{ACLK}/64$
10 $f_{ACLK}/128$
11 $f_{ACLK}/256$

**BTIPx** Bits 2–0 Basic Timer1 Interrupt Interval.
000 $f_{CLK2}/2$
001 $f_{CLK2}/4$
010 $f_{CLK2}/8$
011 $f_{CLK2}/16$
100 $f_{CLK2}/32$
101 $f_{CLK2}/64$
110 $f_{CLK2}/128$
111 $f_{CLK2}/256$

[1] 13-7 page

# C program for LED blinking using polling

```
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  P2DIR |= 0x02;
  P2OUT = 0x02;

  //setup Basic Timer1
  // BTCNT2 <- slow interval 2Hz
  BTCNT1 = 0;
  BTCNT2 = 0;
  BTCTL = 0x25; //001 00 101
  while(1) {
    //정해진 간격 체크?
    if( BTCNT2 == 255 )
        //LED toggle
        P2OUT ^= 0x02;
   }
}
```

**BT1 설정**

•BTCNT1/2 reg. 초기화
(대개 reset에 의해 0이 되나, 0으로의 초기화 필요)

•BTCTL 설정

-가장 중요! 전체 동작 설정

* 클럭 소스 결정: BTSEL=0 & BTDIV=1 ➔ ACLK/256

* 인터럽트 발생 인터벌: /64 →101

Polling 방법

• BTCNT2는 언제나 읽기 가능, 따라서 이 값을 비교하여 특정 간격 측정(관측) 가능

• 255가 되면 LED toggle

• 255 다음 값은?

# C program for LED blinking using Interrupt

```
#pragma vector=BASICTIMER_VECTOR
__interrupt void bt1_handler(void)
{
  P2OUT ^= 0x02;  //toggle LED1
}

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  P2DIR |= 0x02;
  P2OUT = 0x02;

  //setup Basic Timer1
  // BTCNT2 <- slow interval 2Hz
  BTCNT1 = 0;
  BTCNT2 = 0;
  BTCTL = 0x25; //001 00 101
  IE2 |= 0x80;
  __enable_interrupt();

  while(1);
}
```

**BT1 Irq. Serive Routine(ISR)**

- 지정한 인터벌마다 BT1이 발생시킴.
- 이 때마다 자동으로 이 루틴 실행 ➔ LED toggle

**BT1 control reg.설정**

- 앞과 동일
- 클럭소스=ACLK/256, BT1 Interrupt Interval = fclk2/64

**Interrupt enable**

- IE2.7 (BTIE bit) = 1 해야만 IRQ. 동작 실행.
- GIE=1 (by __enable..())