임베디드시스템 기초(#514115)

#4. BT1 Review & Real Time Clock

한림대학교
전자공학과 이선우

한림대학교
HALLYM UNIVERSITY

MSP430x4xx 타이머 종류

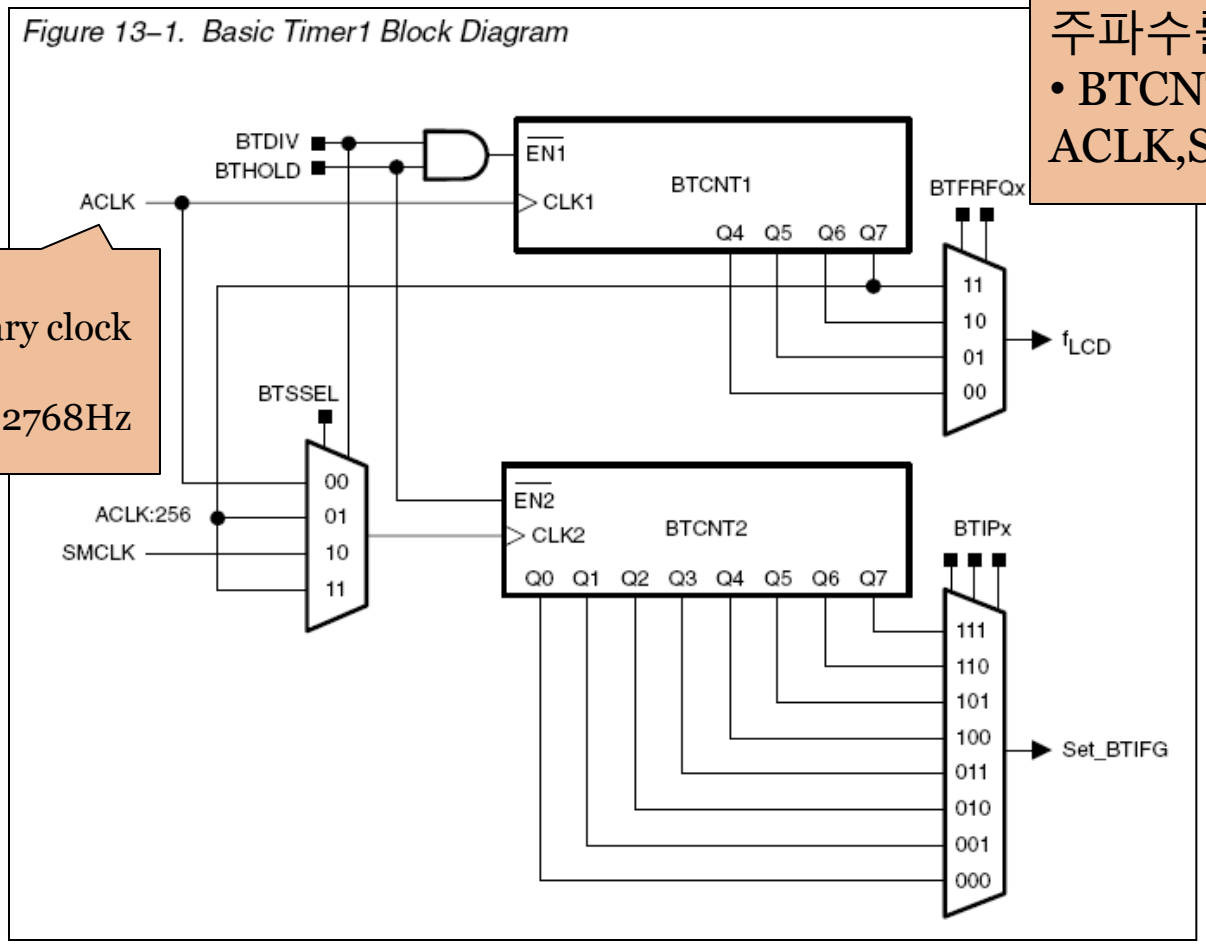
- MSP430x4xx series는 다음과 같은 3종의 타이머 내장
  - Basic Timer1
    - Two independent, cascadable 8-bit timers
    - Selectable clock source
    - Interrupt capability
    - LCD control signal generation
  - Timer A, Timer B
    - 16-bit timer/counter with 3 or 5 capture/compare registers
    - Multiple capture/compares,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



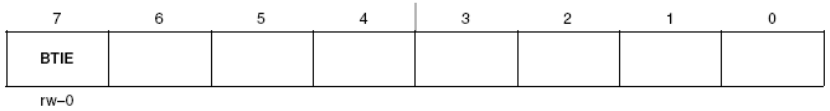
BT1 Registers

Table 13–1. Basic Timer1 Registers

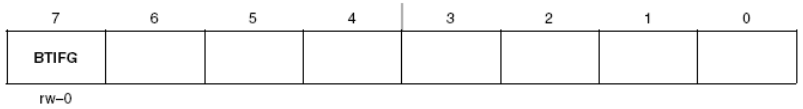
Register	Short Form	Register Type	Address	Initial State
Basic Timer1 Control	BTCTL	Read/write	040h	Unchanged
Basic Timer1 Counter 1	BTCNT1	Read/write	046h	Unchanged
Basic Timer1 Counter 2	BTCNT2	Read/write	047h	Unchanged
SFR interrupt enable register 2	IE2	Read/write	001h	Reset with PUC
SFR interrupt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

BTCTL, Basic Timer1 Control Register

IE2, Interrupt Enable Register 2



IFG2, Interrupt Flag Register 2



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 $f_{CLK2}/256$

BTCNT1 operation: Signal F_{Lcd}

- 적당한 LCD용 클럭 신호 생성 방법
- BTCNT1 제공 가능 클럭: ACLK/256,128,64,32 4종류
- 알맞은 신호 주파수: $f_{LCD} = 2 \times mux \times f_{Frame}$
  - 일례, 4-mux LCD 사용, frame freq.가 30~100Hz라면:
  - f_{LCD} = 2*4*30 (100) = 240Hz(min), 800Hz(max)
  - 따라서 f_{LCD} = 32768/128(64)=256 (512)Hz

- LCD_A 모듈을 내장하고 있는 모델의 경우는 별도 F_{LCD} 내장 회로를 가지므로 이 기능이 필요 없음.

BTCNT2 operation

- 일반적인 8bit up counter로 작동
- 클럭 소스가 CNT1이 ACLK에 비해 3종류(ACLK, ACLK/256, SMCLK) 선택 가능
  - ACLK/256을 선택하면 16bit counter로 사용 가능
- BTIPx bit 설정을 통해 8종류의 시간 간격 지난 후에 인터럽트 발생 (BT Timeout event: BTIFG)
  - f_{CLK2}/(2,4,8,16,32,64,128,256)

기본적인 이용 방법

- BT1의 경우 일반적인 이용방법
  - LCD 컨트롤러용 클럭 생성
  - 기본적인 시간 간격 타이머로 이용
- Ex.
  - 평가보드 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!

C 언어 이용 설정 방법

```
#pragma vector=BASICTIMER_VECTOR
__interrupt void bt1_handler(void)
{

  P2OUT ^= 0x02;  //toggle LED1

}

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;     // Stop
     watchdog timer

  P2DIR |= 0x06;
  P2OUT = 0x02;

  //setup Basic Timer1
  // BTCNT1 <- f_LCD = 512Hz
  // BTCNT2 <- slow interval 2Hz
  BTCTL = 0x2d; //; // 001 01 101 -> 0x2D
  BTCNT1 = 0;
  BTCNT2 = 0;
  IE2 |= 0x80;
  __enable_interrupt(); //intrinsic ftn.

  while(1);

}
```

BTCTL 설정

- 가장 중요! 전체 동작 설정
- BTSEL=0, BTHOLD=0, BTDIV=1
- f_{LCD} freq. = fACLK/64 <= 0 1
- BT Irq. Interval = /64 <= 101

CNT reg. 초기화

- 대개 RESET되면 0이 되나 초기화 할 필요가 있다.

Interrupt enable

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

BT1 활용 방법

- 기본 기능: 특정 인터벌마다 인터럽트 발생시킴
- 활용: 기본 소프트웨어 타이머로 활용
  - 모든 OS에서 필요한 system tick을 만드는데 사용. 즉 일정한 인터벌에서 특정 작업 수행 가능
  - 예1: 스위치 입력 기능
    - 기존 무한루프에 딜레이 사용 방법: 불필요한 자원(전력)을 낭비
    - BT1 이용 일정 간격마다 스위치 상태 읽어 반응하게 함.
    - Matrix keypad scanning 할 때 이용.
  - 예2: 디지털 시계 구현
    - 1초 인터벌 설정, 이를 적절하게 count하여 시계 구현함.
- 부가 기능: LPM3 이용 저전력 동작 가능하게 함
  - LPM3는 ACLK 사용 가능하므로 타이머 동작하고, BT1에 의해 발생하는 인터럽트는 sleep상태의 MCU를 깨워(awaking) 어떤 작업을 할 수 있다.

Example code #1: LPM & BT1

- 일정 시간 간격으로 BT1이 인터럽트를 발생시키고 이를 이용하여 LED를 점멸시키는 응용 예

```
#define   LED1      BIT1

void main(void)
{
//port and BT1 초기화
// BT1 irq. Enable

  __low_power_mode_3(); //LPM3; ACLK만 유효
}

#pragma vector=BASICTIMER_VECTOR
__interrupt void BT1_ISR(void)
{
     P2OUT ^= LED1;
}
```

LP 모드(LPM3)로 진입
초기 설정 이후 이 명령을 사용하여 LPM3 로 설정 ➔ ACLK이외 다른 클럭 신호 OFF, CPU는 현재 상태에서 멈춤(sleep상태)
*IRQ. 발생하면 깨어남.

정해진 시간 간격이 지나 BT1이 IRQ.를 발생시키면 sleep상태에서 깨어나 ISR을 실행함.
즉, LED 토글하고 main 루틴으로 돌아감.
*되돌아갈 때 스택에 저장했던 SR 값을 복원 ➔ LPM 모드 세팅이 적용(즉 LMP3 상태 유지)

Example code #3: LPM & BT1 & SW1

- 저전력 상태에서 일정 시간마다 SW1 상태 읽어(debouncing 적용) 이 값이 변할 때 마다 LED를 toggle시키는 프로그램

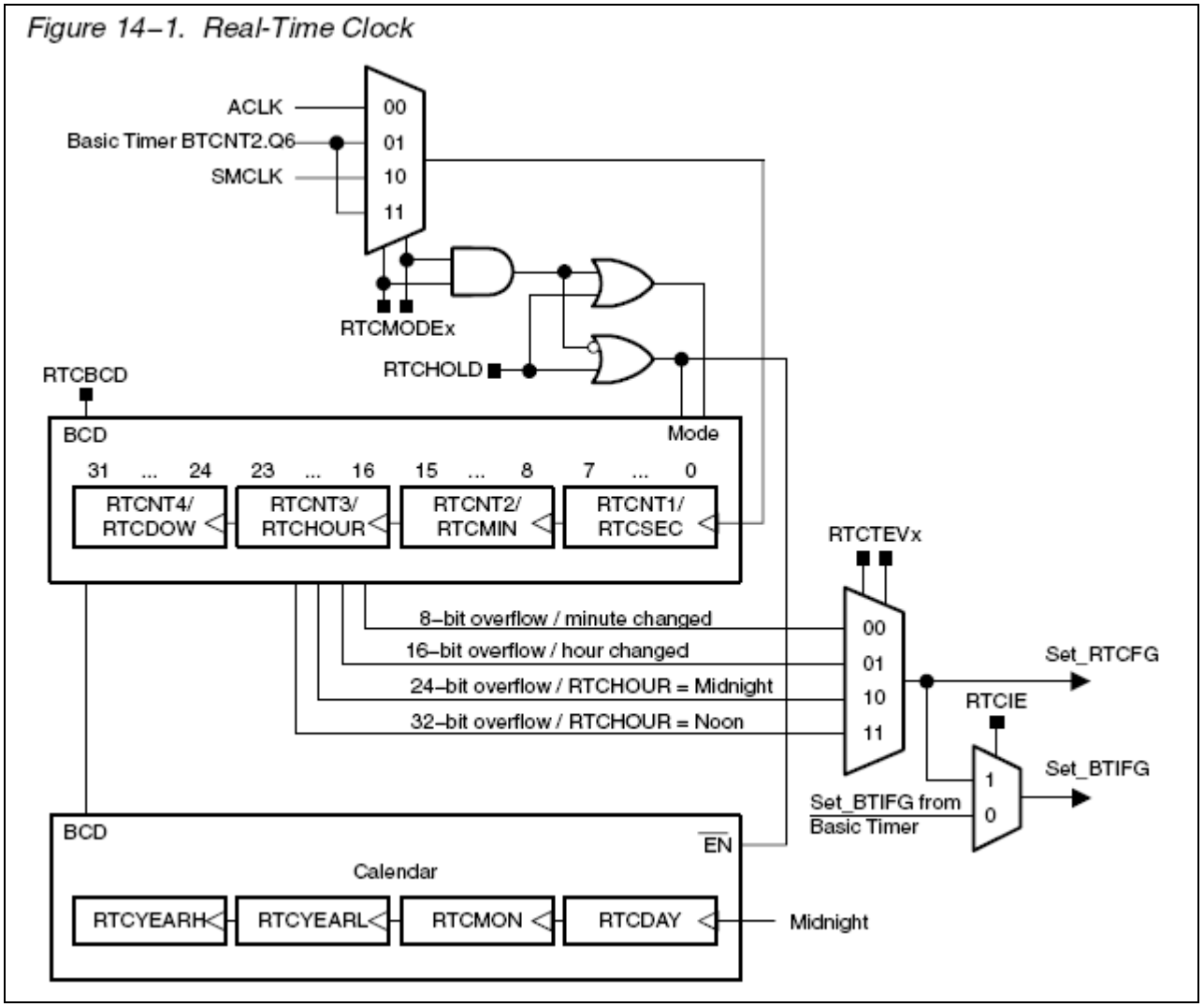
```
unsigned char firstread, swpush;
void main(void)
{
// GPIO port 설정: SW1 입력, LED 출력
설정
// BT1 설정: 약 20~40msec 정도 인터벌
// BT1 irq. Enable 설정
  firstread = 0;
  __low_power_mode_3(); //LPM3;
ACLK만 유효
}

#pragma vector=BASICTIMER_VECTOR
__interrupt void BT1_ISR(void)
{
  //SW1 읽음
  swpush = PxIN & SW1BIT;
  if(!swpush) //SW1 눌렸나?
  {
    if(firstread) //이번이 두번째 누름?
    {
        P2OUT ^= LED1; //Toggling
        firstread = 0;
    }
    else
        firstread = 1; //첫번째 눌림
  }
```

Real Time Clock

- 시계란?
  - 1Hz clock을 분(60)/시(60)/날(24) 단위로 개수를 세어 표시하는 장치
- RTC module
  - Calendar Mode: 캘린터/시계 기능
    - 자동적인 시/분/초 및 날짜(년/월/일/요일) 계산 기능
    - BCD format 가능
  - Counter Mode: 일반적인 32-bit 타이머/카운터 기능
    - 3개 클럭 소스(ACLK, SMCLK, BTCNT2.Q6) 선택 가능
    - 4개 8-bit 카운터 레지스터(RTCNT1~4)의 개별 R/W 가능, 연속 연결하여 32-bit 카운터로도 동작
    - 8/16/24/32-bit overflow 발생 시 인터럽트 발생

RTC block diagram



RTC 동작: calendar mode

- 일반적인 시계 정보 (년, 달, 일, 요일, 시, 분 초 제공)
  - 윤년 계산 알고리즘 내장 (1901~2099)
  - BCD (Binary Coded Decimal) format 제공
    - 10진수의 각 자리를 4bit의 2진수로 별도로 표시하는 방법. Ex) 10 → 0x0A (X), 0x10(O), 32→ 0x10(X), 0x32(O)
    - 10진수 숫자의 디스플레이 장치(7세그먼트 등)를 이용한 표시에 사용됨.
- BT1과의 관계
  - Calendar모드 설정되면 자동으로 ACLK이 BT1의 소스로, 2개 8-bit counter가 연결(cascaded)되어 1Hz 클럭 생성하도록 함 (전제조건:ACLK=32768Hz)

RTC 동작: calendar mode

- Interrupts 관련 사항
  - RTC의 여러 사건들은 BT1 인터럽트 관련 하드웨어를 공통으로 사용함. (별도 인터럽트 벡터가 없음)
    - RTIE=1이면 RTC Event 발생에 의해 BTIFG(BT1 irq. Flag) set (블럭 다이어그램 참조!)
    - RTCEVx bit에 의해 카운터/캘린터 모드에 따라 각각 4개씩 이벤트 정의되어 있음.
  - 따라서 코드 상에서는 BT1에 대한 ISR을 만드는 것과 동일하게 ISR 작성해야 함!
    즉, #pragma vector=BASICTIMER_VECTOR 사용

RTC registers

RTCCTL, Real-Time Clock Control Register

7	6	5	4	3	2	1	0
RTCBCD	RTCHOLD	RTCMODEx		RTCTEVx		RTCIE	RTCFG
rw-(0)	rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0

RTCBCD Bit 7 BCD format select. This bit selects BCD format for the calendar registers when RTCMODEx = 11.
0 Hexadecimal format
1 BCD format

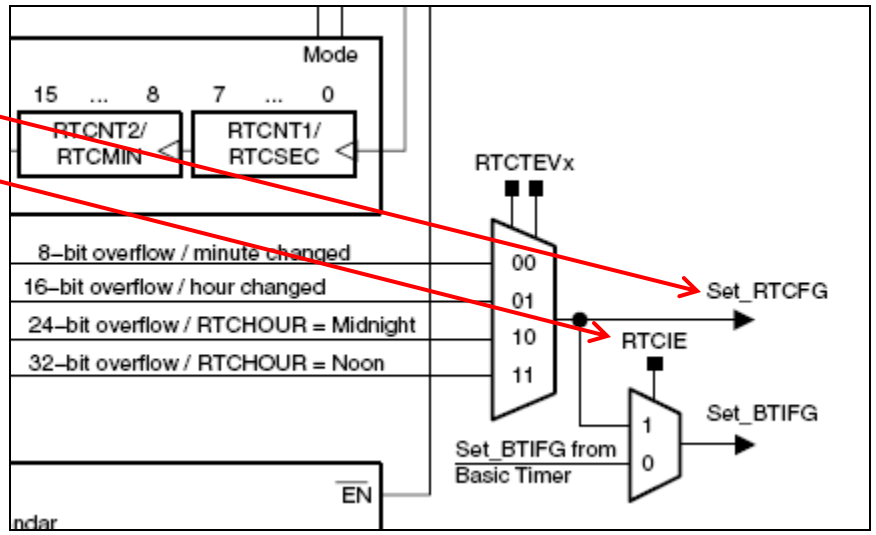
RTCHOLD Bit 6 Real-Time Clock hold
0 Real-Time Clock is operational
1 RTCMODEx < 11: The RTC module is stopped
RTCMODEx = 11: The RTC and the Basic Timer1 are stopped

RTCMODEx Bits 5-4 Real-Time Clock mode and clock source select

RTCMODEx	Counter Mode	Clock Source
00	32-bit counter	ACLK
01	32-bit counter	BTCNT2.Q6
10	32-bit counter	SMCLK
11	Calendar mode	BTCNT2.Q6

RTCTEVx Bits 3-2 Real-Time Clock interrupt event. These bits select the event for setting RTCFG.

RTC Mode	RTCTEVx	Interrupt Interval
Counter Mode	00	8-bit overflow
	01	16-bit overflow
	10	24-bit overflow
	11	32-bit overflow
Calendar Mode	00	Minute changed
	01	Hour changed
	10	Every day at midnight (00:00)
	11	Every day at noon (12:00)



RTCSEC, RTC Seconds Register, Calendar Mode with Hexadecimal Format

7	6	5	4	3	2	1	0
0	0	Seconds (0...59)					
r-0	r-0	rw	rw	rw	rw	rw	rw

RTCSEC, RTC Seconds Register, Calendar Mode with BCD Format

7	6	5	4	3	2	1	0
0	Seconds - high digit (0...5)			Seconds - low digit (0...9)			
r-0	rw	rw	rw	rw	rw	rw	rw

RTC module Coding 방법

```
#pragma vector=BASICTIMER_VECTOR
__interrupt void rtc_handler(void)
{
  P2OUT ^= 0x02;  //toggle LED1
}
void main(void)
{
  P2DIR |= 0x06;
  P2OUT = 0x02;

  //setup Real-Time Clcok
  RTCCTL = 0xF2; // 0b 1 1 11 00 10 -> 0xF2
  //init time
  RTCSEC =  0x00;       // Set Seconds
  RTCMIN =  0x59;       // Set Minutes
  RTCHOUR = 0x12;       // Set Hours
                        // Init date
  RTCDOW =  0x01;       // Set DOW
  RTCDAY =  0x21;       // Set Day
  RTCMON =  0x09;       // Set Month
  RTCYEAR = 0x2009;     // Set Year
  RTCCTL &= ~RTCHOLD;   // Enable RTC
  __enable_interrupt(); //GIE=1
  while(1);

}
```

RTCCTL 설정

- 가장 중요! 전체 동작 설정
- RTCBCD=1 (BCD format사용)
- RTCHOLD =1 (stop)
- RTCMODEx = 11 (calendar mode)
- RTCTEVx= 00 (분 변화시 인터럽트 발생)
- RTCIE = 1 Irq. enable

시간/날짜 레지스터 초기화

- 대개 RESET되면 0(or 1) 이 되나 시작 날짜/시간 초기화 할 필요가 있다.

C Programming TIP #1

- 거의 모든 응용 프로그램들은 복수개의 소스 파일들로 구성
  - 이유: 각 기능별로 구분하여 프로그래밍하는 것이 작성/디버깅에 편리하기 때문.
  - 방법: 기능별로 별도 파일로 작성하고 main() 함수에서 사용.
- 예: LCD_A, GPIO, BT 모듈 별로 별도 파일 작성
  - 작성순서
    - lcd.c : 관련된 함수에 대한 코드 작성. lcd_init(), lcd_num_dis() 등의 함수를 만듬.
    - lcd.h: 다른 파일에서 lcd 관련 함수를 call할 수 있도록 정의된 함수를 선언(declaration)함.
    - main_clock.c: main() 함수가 있는 파일
      - #include “lcd.h” 를 이용 lcd 관련 함수가 무엇이 있는지 안다.
      - 코드 중에 lcd 관련 함수를 CALL

C Programming TIP #1

```
/* lcd.c file: LCD_A 제어기 관련 함수들
*/
#include "lcd.h"

void lcd_init(void)
{
  //port setup
  //LCDA setup
}
void lcd_nu_dis(int digit, int num)
{
//digit: 7개 숫자판 중 하나 지정
//num: 표시하는 숫자
}
```

```
/* main_clock.c file */

#include "lcd.h"

void main(void)
{
   //초기화 관련..
  led1_init();
  lcd_init();

  //LCD 2번 숫자판에 2표시
  lcd_num_dis(2,2);
  lcd_num_dis(3,3);
…
}
```

```
/* lcd.h file*/

void lcd_init(void);
void lcd_nu_dis(int digit, int num);
```