마이크로컨트롤러 기초(#514112)

#.8 Basic Timer1 응용

한림대학교
전자공학과 이선우

Contents

- Low-Power Mode
  - Low-power mode?
  - LPM in MSP430
  - Example program for LPM4
- MSP430x4xx Timers
  - Basic clock generation
  - LPM3 and Awakening by BT1
  - Example programs

Low-Power Mode

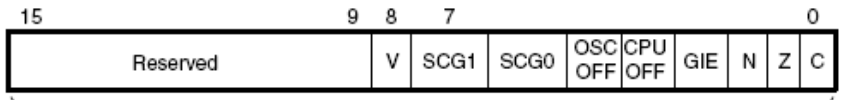
Low Power Mode란?

- 모든 휴대용 임베디드 시스템은 주 전원으로 배터리를 이용하므로 가능한한 적은 소비 전력을 사용하도록 설계, 구현되어야 한다. (배터리 용량→ 사용 시간(편리성, 유효성)을 결정하는 가장 중요한 인자)
- MSP430은 에너지를 적게 또 효과적으로 사용할 수 있도록 설계됨→ 시장 호평 이유!
  - Active mode (정상 동작 모드): Vcc=3V, I=600uA
  - Low power mode: $I_{(LPM3)}$=1.9uA , $I_{(LPM4)}$=0.3uA
- 에너지 절약 방법
  - Power OFF (전원 차단)
    - 장치 전체의 전원 자체를 ON/OFF 하는 의미
    - 장치를 동작시키는 클럭 신호를 ON/OFF
  - 처리 속도를 낮춤(Fosc ↓)
    - Main/auxiliary clock speed ↓

MSP430x4xx series LPM

모두 5개의 Low Power Mode 존재

Figure 3–6. Status Register Bits



SR에 있는 4bit에 의해 결정

SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled (41x/42x peripheral M remains on) SMCLK , ACLK are active
0	1	0	1	LPM1	CPU, MCLK, DCO osc. are disabled (41x/42x peripheral MCLK remains on) DC generator is disabled if the DCO is not used f MCLK or SMCLK in active mode SMCLK , ACLK are active
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator remains enabled ACLK is active
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator disabled ACLK is active
1	1	1	1	LPM4	CPU and all clocks disabled

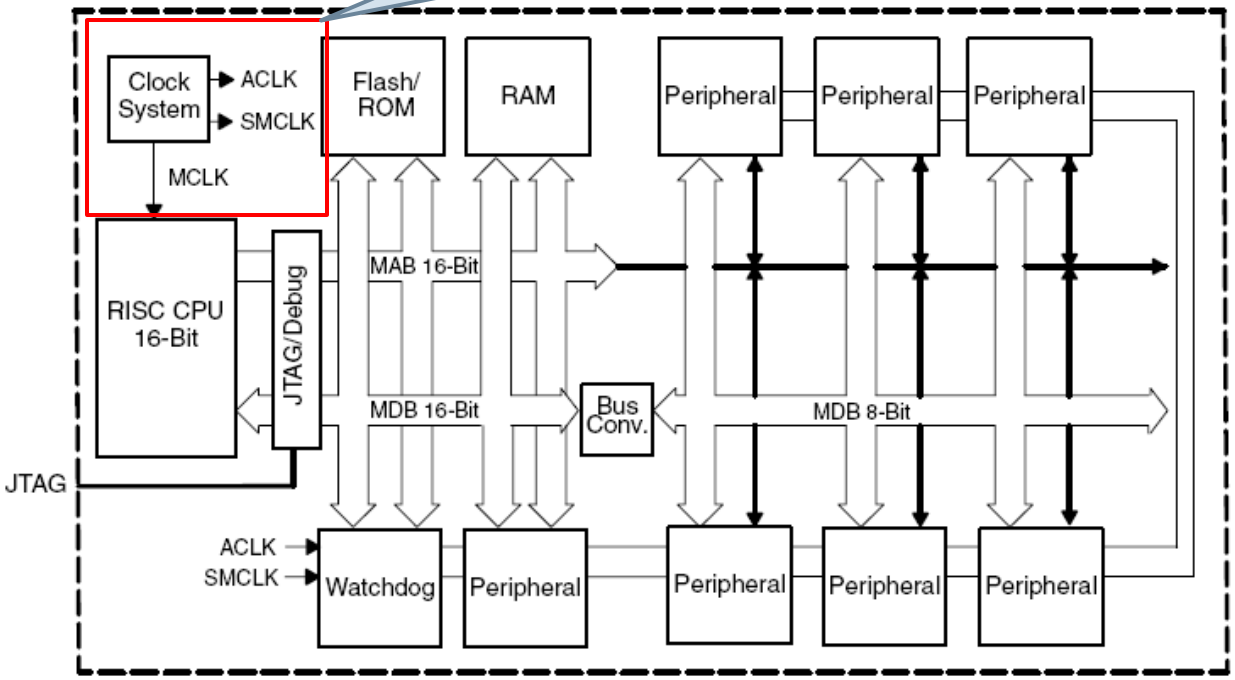
LPM3: 가장 많이 사용!!
(기존 SLEEP 모드 동일)
*ACLK에 의해 동작하는 다른 장치에 의해.
깨어남(Wake-up)

-정상동작 모드: I=600uA
-LPM0: CPU,MCLK만 OFF
Vcc=3V, I=80uA (1/8)
-LPM1: LPM0와 동일(461x 경우)
-LPM2: ACLK만 유효, I=17uA (1/40)
-LPM3: I=1.9uA (1/360)
-LPM4: I=0.3uA (1/2000)

MSP430 Architecture & Clock signals

* MCU 내의 모든 장치(CPU포함)는 반드시 클럭 신호를 필요로 하며 이 신호에 동기되어 동작함.
* 4개의 클럭 신호 존재:
  - MCLK: Master clock, CPU 구동, 1~8 MHz
  - SMCLK: Sub-Main clock, 빠른 클럭을 필요로 하는 주변 장치, MCLK과 같은 빠르기
  - ACLK: Auxiliary clock, 대부분의 주변장치에서 사용, 32KHz

Figure 1–1. MSP430 Architecture



한림대학교 HALLYM UNIVERSITY
senu@hallym.ac.kr

LP mode로 들어가는 방법

- 대개의 MCU에서 저전력소비 상태(LP mode)로 변경되는 것을 'sleep상태'로, 정상 동작 상태(active)로 변경되는 것을 '깨어남(waking)' 이라고 표현한다.
- 다른 MCU들도 저전력상태를 지원하고 이를 위해 별도 명령어 (sleep, stop)등이 존재하나, MSP430의 경우는 별도 명령어가 없고 SR의 관련 4bit(SCG0, SCG1, CPUOFF, OSCOFF)의 변경으로 모드가 변경된다.
  - Assembly: bis.w #GIE|LPM3, SR
  - IAR C: __low_power_mode_3[0/1/2/4]();

Waking from a LPM

- MSP430을 깨우기 위해서는 인터럽트가 필요. 즉, 어떤 종류의 IRQ. 발생도 MCU 깨울 수 있다.
- 정상상태와의 차이점: LPM2 이상에서는 ACLK만이 유효하므로 대개 CPU 동작을 시키기 위해서 MCLK을 먼저 유효하게 만드는 작업이 더 필요함 ➔ DCO 작동을 위한 별도 지연(delay) 발생 (x461x series의 경우 max. 6us)

LPM4 특징/이용법

- LPM4의 경우는 ACLK도 disable시키므로 BT1 등의 타이머도 동작하지 않는다.
- 따라서 LPM4로 변경했을 때 waking시키는 유일한 방법은 P1/2의 입력 핀의 상태 변화(외부 인터럽트,external interrupt)만이 유일하다.
- 이용 예

```
void main(void)
{
//SW1--P1.0 H→L IRQ. 설정
 P1DIR &= ~(BIT0);
 P1IES |= BIT0;
 P1IE |= BIT0;
  while(1) {
      __low_power_mode_4();
   }
}

#pragma vector=PORT1_VECTOR
__interrupt void P1_ISR(void)
{
     P1IFG &= ~BIT0;
     P2OUT ^= LED1;
}
```

Basic Timer1 Applications

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
  - 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 #2: #1과 조금 다른 방법

- 1번 이용법과 동일한 작업 (즉 LED 점멸)
- 차이점: while(1) loop 사용
  ISR을 끝내고 되돌아 갈때 LPM 관련 비트만 클리어시킴.

```
#define   LED1      BIT1

void main(void)
{
//port and BT1 초기화
// BT1 irq. Enable
  while(1) {
     __low_power_mode_3();
          P2OUT ^= LED1;
  }
}

#pragma vector=BASICTIMER_VECTOR
__interrupt void BT1_ISR(void)
{
     __low_power_mode_off_on_exit();
}
```

이 명령어 수행되면 이 지점에서 sleep 상태로 됨.
IRQ. 발생에 따라 깨어난 후 ISR이 실행, 즉 active 상태로 변경되어 돌아와 다음 명령어 (P2OUT..) 실행되어 LED1 토글.

이 명령어는 단순하게 SR의 LPM관련 비트를 모두 clear 시켜 **active상태로 변경하는 역할만 함.**

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
  }
```

Example code #4: 디지털 시계

- 1초 인터벌 설정, 인터럽트마다 초,분,시 변수 update!

```
unsigned char sec, min, hour;
char sectick;
void main(void)
{
// BT1 설정: 1sec 인터벌
// BT1 irq. Enable 설정
//초기 시각 설정, 예
  hour=14;
  min=32;
  sec=0;
  sectick=1;
  while(1) {
    if(sectick)
    {
     //4개 세그먼트에 시,분표시하는 루틴
        display_clock();
        sectick=0;
    }
  }
}
```

```
#pragma vector=BASICTIMER_VECTOR
__interrupt void BT1_ISR(void)
{
 //초 변수 update
  sec++;
  if(sec==60) {
    sec = 0;
    min++;
  }
  if(min==60) {
    min = 0;
    hour++;
  }
  if(hour==24) {
    hour = 0;
  }
  sectick = 1;
}
```