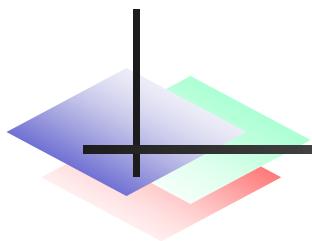


MCU 응용

- DEMO9S12XEP100을 이용한 -



2014년 1월

동양미래대학교
로봇자동화공학부
정준

다루는 내용

- ❖ MCU 학습을 위한 기본 지식
- ❖ 예제로 학습하는 MCU 응용
 - 개발환경 사용법 익히기
 - Bus Frequency 적용하기
 - LED와 Switch 사용하기
 - Timer를 이용해서 시간 지연 및 런타임 측정하기
 - Interrupt 제대로 사용하기
 - Debugging 용도로 PC와 UART 통신하기
 - 블루투스 모듈을 이용해서 PC/스마트폰과 UART 통신하기
 - LCD에 정보 출력하기
 - ADC 함수를 만들어 거리 측정 센서로 물체까지의 거리 측정하기
 - 타이머로 초음파 센서 모듈의 펄스 폭을 측정하여 거리 측정하기
 - 로타리 엔코더의 회전각을 외부 인터럽트를 이용해서 측정하기
 - PWM 출력과 모터 드라이버를 이용해서 RC Servo 모터와 DC 모터 구동하기
- ➔ 이 모든 기능을 동시에 사용할 수 있는 F/W 만들기
- ❖ 교육에 사용되는 회로부 만들기

학습 목표

- ❖ MC9S12XEP100 MCU용 응용 Firmware를 개발할 수 있다.
- ❖ MC9S12XEP100 MCU와 관련해서
 - CodeWarrior를 사용할 수 있다.
 - Digital I/O Port를 이용해 LED나 스위치를 연결해 사용할 수 있다.
 - 일정시간 간격으로 인터럽트를 발생시켜 사용할 수 있다.
 - 인터럽트 개념을 정확히 이해하고 문제 없이 제대로 사용할 수 있다.
 - UART 통신으로 PC에 연결해 사용할 수 있다.
 - Character LCD에 문자를 출력할 수 있다.
 - A/D Convert로 아날로그 전압을 측정할 수 있다.
 - 적외센서를 연결해 거리를 측정할 수 있다.
 - 초음파 센서를 연결해 거리를 측정할 수 있다.
 - 증감형 로타리 엔코더를 연결해 회전각을 검출할 수 있다.
 - 모터 드라이버 전용 칩에 맞는 PWM 파형을 출력할 수 있다.
 - RC용 서보 모터에 적합한 PWM 파형을 출력할 수 있다.
 - Event Driven Programming을 할 수 있다.

학습 방법

❖ MCU 학습에 좋은 방법

- 도제식이 좋음
 - ◆ 문제를 내주고 풀게 함
 - ◆ 틀린 점만 지적해 줌
- 그러나 교육 시간이 제한되어 있음

• MCU 학습을 어학에 비유하면
문법 공부가 필요한 것이 아니라
작문 공부가 필요하다!

❖ 책만 보면서 학습하는 것이 좋지 않은 이유

- 빠른 학습은 가능하나
- 개발자의 '생각의 흐름'을 알 수 없고, 중간 과정도 알 수 없고, 결과물만 알 수 있음
- 또한 MCU 관련 자료는 프로그래밍 기법 보다는 기능 설명 위주

❖ 이 강좌의 학습 방법

- 되도록이면 소스 코드에 대한 예습을 하지 말 것
 - ◆ 예습을 하면 생각할 문제의 답을 미리 보게 됨
 - ◆ 예습을 하려거든 문제만 보고 직접 소스 코드를 만들 것
- 강의 후 그날 배운 내용을 숙지할 것
 - ◆ 계속 연관이 되므로
- 강의 자료에 필요한 내용을 필기할 것!

• 계단형 학습 효과

- 학습 효과가 바로 나타나지 않음
- 꾸준히 하다 보면 어느 순간 학습 효과

• 벤다이어그램 학습 효과

- 여기 저기 겹치는 부분이 존재

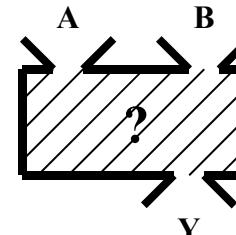
• 50% 이상과 미만의 차이

- 아는 게 별로 없으면 모르는 것이 나올 때 모르는 것이 더 많아짐
- 50% 이상을 알면, 모르는 것이 나올 때 이를 바탕으로 알게 되어 더 많이 알게 됨

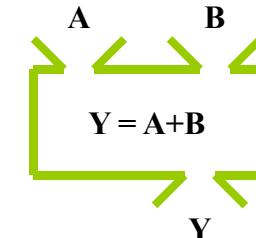
Function을 사용하는 개발자의 자세...

❖ Black Function v.s. White Function

- 정의: Function 내부를 모르면 Black Function,
Function 내부를 알면 White Function
- 예: $A = 1 \text{ or } 2, B = 3 \text{ or } 4$ 일 때,
Black Box: $(A=1, B=3) \rightarrow Y=4, (1, 4) \rightarrow 5, (2, 3) \rightarrow 5, (2, 4) \rightarrow 6$
White Box: $Y = A+B$



Black Function



White Function

❖ Black Function의 사용 방법

- 어떤 기능을 하는 함수(블록, 모듈, 칩 등)가 있을 때, 입력에 대한 출력의 결과를 안다면, 그 함수 또는 블록 내부에 대해 자세히 알려 하지 말자.(정 알고 싶으면 남는 시간에...)
 - ◆ 왜? 연구에는 시간이 중요하지 않으나 개발에는 시간도 비용이다!
- Black Function의 입출력에 대한 문서를 잘 읽어 이해하도록 한다.
 - ◆ Black Function이 핸드폰이라면, 핸드폰 매뉴얼
 - ◆ Black Function이 IC 칩이라면, IC Datasheet
 - ◆ Black Function이 C Function이라면, Library Manual
 - ※ 개발의 대가와 그렇지 않은 사람의 차이 중 하나: **Datasheet나 Spec. Sheet의 내용을 아느냐 모르느냐!**
 - ※ 슬픈 현실: Datasheet나 Manual을 너무 안 읽어요!, 찾아보는 것을 너무 안 해요!
→ 개발 관련 문서를 안 만들거나 대충 만들어요 → Know-how가 잘 안 쌓여요 → 모래성일 가능성이 커요
- Black Function의 입력 범위를 벗어나지 않도록 한다. 설사 그 입력에 대한 출력이 나오더라도.
- Black Function의 출력이 예상치 않는 결과를 가져온다면 반드시 짚고 넘어간다. → F/A
- 어떤 Function이라도 입출력을 아는 Black Function 으로 만들면 시스템의 일부로 사용할 수 있다.

❖ Interface 란?

- 서로 다른 Function 또는 블록들을 연결시켜주는 고리!

개발환경 설치

❖ CodeWarrior Development Studio for HCS12(X) Microcontrollers 설치

- 목적: C/C++로 Firmware 구현
- 반드시 V5.1 Special Edition을 설치할 것
 - ◆ Evaluation Edition은 기능 제한은 없는 대신 날짜 제한이 있음
 - ◆ Special Edition은 날짜 제한이 없는 대신 C code 사이즈 제한 (32KB for S12X and 512 Byte for XGATE)
 - 참고: 32KB면 여러분이 작성하는 웬만한 프로그램은 다 구현 가능
- 위치1: 제공되는 CD
- 위치2: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-HCS12X&fpfsp=1&tab=Design Tools Tab

❖ SofTec 제공 System Software 설치

- 목적: USB 인터페이스에 의한 Firmware 다운로딩 및 Debugging 환경 제공
- 설치할 파일: demo9s12xep100_setup.exe (다른 파일은 설치할 필요 없음)
- 위치: 제공되는 CD_Driver: \DEMO9S12XEP100\Progs
- 불행히도 이 파일은 64bit Windows에 설치 안 됨
 - ◆ 64bit Windows를 사용하려면 VirtualBox를 이용 (시연 참조)

❖ WinMerge를 인터넷에서 검색해서 설치

- 목적: 두 개의 파일/폴더의 내용을 비교/편집할 때 사용하는 유틸리티
- CodeWarrior에도 이 기능이 있기는 함

❖ Tera Term를 인터넷에서 검색해서 설치

- 목적: Serial 통신
- Windows XP에서는 하이퍼터미널 사용 가능

참조할 파일 준비

- ❖ MC9S12XEP100 Reference Manual: MC9S12XEP100RMV1.pdf
 - 위치:
http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DEMO9S12XEP100&fsp=1&tab=Documentation_Tab
 - 참고: CodeWarrior 설치 폴더에도 존재하나 위 웹에서 최신 버전 (Rev. 1.24)을 받을 것.
- ❖ 보드 회로도: demo9s12xep100_schematic.pdf
 - 위치1: 시작→모든 프로그램→SofTec Microsystems→DEMO9S12XEP100→User's Manuals
 - 위치2: C:\Program Files\SofTec Microsystems\DEMO9S12XEP100
- ❖ 보드 매뉴얼: demo9s12xep100_manual.pdf
 - 위치: 보드 회로도와 같은 위치
 - 점퍼 세팅 참조
- ❖ 기타 datasheet
 - 카페 게시판 참조!

소스 코드 관리 및 프로그램 설계

❖ 소스 코드 관리(버전 관리) 툴 예

- SourceSafe
- CVS
- ClearCase

※ 소스 코드는 한번에 완성되는 것이 아니라 여러 사람에 의해 완성을 향해 발전해 나가기 때문에
이력이 중요!

❖ 설계

- 잘 설계된 프로그램은
 - ◆ 설계하는 데 오랜 시간이 걸려 소스 코드 완성까지 시간이 많이 걸리지만
 - ◆ 오류가 적어 디버깅 시간이 짧고, 추후 기능 추가 시에도 수정을 쉽게 할 수 있음
- 대충 설계된 프로그램은
 - ◆ 소스 코드 완성까지 시간이 짧게 걸리지만
 - ◆ 오류가 많이 디버깅 시간이 길게 되고, 추후 수정이 어렵고, 수정해도 **누더기 코드**가 되어 점점 복잡해짐

❖ 실습: 다른 버전의 소스 코드를 비교할 수 있는 WinMerge

- 파일명이 같은 두 개의 파일을 비교 편집할 수 있는 유틸리티
- 여러분들은 소스코드 관리 툴을 사용 안 하는 대신 폴더로 이력을 관리하고 이 유틸리티를 이용해서 이전 버전과 현 버전의 비교 편집에 활용하세요!

MCU 학습을 위한 참고 사항

❖ 어셈블리어 = 기계어

- Registers: Accumulator, 범용 Register, PC, SP, FLAG(Status, Condition, CCR 등)
- Memory (Data Memory, Code Memory, I/O Memory 등)
- 기계어: Register와 Memory의 연산(산술, 논리, 비교), 분기, 이동
- CPU는 Clock 주파수에 맞춰 PC 레지스터의 명령어를 실행!
- 어셈블리어: 숫자로 된 기계어를 문자로 거의 1:1 치환해 놓은 것
- 참고: Label, Mnemonic, Operand, Addressing
- 어셈블리어로 프로그래밍은 못해도 해석은 할 줄 알아야 한다!

❖ MCU란?

- CPU + Peripherals
- DSP란? → Sum($A_n \cdot X_n$) 연산에 적합한 기계어 명령어 및 구조가 있는 MCU로 보면 됨
 - ◆ 왜 그럴까?
- 참고: RISC와 CISC, Harvard Architecture(Pipe Line)

❖ X bit MCU

- X bit는 보통 기계어 한 명령어로 처리할 수 있는 최대 bit를 말함
 - ◆ 8bit MCU라도 16bit code memory를 사용하는 경우 있음
 - ◆ 16bit MCU지만 연산만 40bit로 하는 DSP도 있음
- 일반적으로 x bit MCU면, register의 연산 및 데이터 이동을 한 명령어로 x bit 만큼 할 수 있음

MCU 응용하기에 앞서

- ❖ Datasheet를 언제나 참조할 수 있도록 준비
 - MCU Datasheet를 처음 보는 사람들을 위한 조언
 - ◆ MCU 응용이 주업무라면, 목차를 훑어보고 처음부터 끝까지 한번쯤은 다 읽어볼 것을 추천 (양이 방대)
 - ◆ 어떤 MCU든 Datasheet를 한번 통독하고 나면 어떤 MCU를 사용하더라도 쉽게 적응이 가능
 - Compiler Manual을 언제나 참조할 수 있도록 준비
 - ◆ MCU 응용이 주업무가 될 것이라고 생각 Compiler Manual도 한번쯤 통독 추천 (양이 방대)
- ❖ 개발 도구 활용을 극대화하자
 - 도구를 잘 사용하는 것도 능력
 - 에디터, 소스 코드 관리, 기타 개발에 사용되는 다양한 S/W 및 H/W 활용 능력
- ❖ 다른 사람이 작성한 소스 코드를 분석하자
 - 기존 개발자들이 작성한 소스 코드를 반드시 분석해 볼 것
 - 소스 코드의 단편적인 부분만 보지 말고 전체 구조도 반드시 익힐 것
- ❖ 어셈블리어로 프로그램은 못 짜도 이해는 할 수 있도록 하자
 - MCU의 Register, Memory Map, Stack, Interrupt, I/O Register 등을 이해할 수 있다
 - C 언어가 어떻게 어셈블리어로 번역되어 어떤 구조로 실행되는지 알 수 있다
- ❖ 이 모든 것을 열심히 하다 보면, 내공이 쌓이고 그러면 필요한 부분을 그때 그 때 찾아 보고 적용할 수 있는 능력이 생기게 된다!

교육에 활용하는 소스 코드 작성 원칙 및 제한

❖ 원칙

- 되도록이면 CodeWarrior 스타일을 따름
 - ◆ 회사마다 나름대로의 코드 작성 규칙이 있으며, 반드시 따를 것!
 - ◆ 소스 코드 작성의 개성은 보이는 것에 두는 것이 아님. 개성은 알고리즘 및 구조 설계에 둘 것!
 - ◆ 변수명/함수명 작명법, 보이는 Style, MISRA, 등
 - ◆ 과제: MISRA가 무엇인지 조사
- CodeWarrior 기본 제공 파일을 이용
 - ◆ 기본 제공 헤더 파일을 적극 사용

❖ 시간상, 수준상 제한

- BANK, PAGE 등의 개념은 사용하지 않음
 - ◆ 참고: 제한된 구조 속에서 더 많은 용량의 메모리를 사용하는 기법
- MCS12XEP100의 XGATE는 사용하지 않음
 - ◆ MCS12XEP100에는 두 개의 CPU가 있으며, XGATE는 이 중 인터럽트 처리용으로 보면 됨.
- 자동 코드 생성(Processor Expert)은 사용하지 않음
 - ◆ 기본을 알아야 하기에...
 - ◆ 자동은 처음에는 편하나 나중에는 복잡해지고, 내용을 잘 모르고 사용하다가는 반드시 문제를 발생시킴
 - ◆ 하지만 자동으로 생성된 코드의 구조와 내용을 이해하는 것은 학습에 큰 도움
- Power Saving은 다루지 않음
 - ◆ 저전력용 제품의 F/W에서는 매우 중요할 수 있으나
- Exception 처리는 다루지 않음
 - ◆ 매우 중요함. 제품의 F/W는 절대 무한 루프에 빠지면 안 된다.
→ 문제가 있으면 빠져 나오고 오류를 보고해야 함

이 교육에서 다루는 예제 중심의 학습 안내

❖ 예제 구성

- 프로젝트 이름이 숫자로 시작됨 00.어쩌구, 01.저쩌구...
- 각 예제에서 다룬 핵심 소스 코드가 다음 예제에 포함되는 방식
 - ◆ 예) LED 관련 소스 코드는 뒤에 나오는 모든 예제에 포함됨
- 예제를 다 하고 나면 얻게 되는 것
 - ◆ 모형차를 이용한 프로젝트에 필요한 기본 소스 코드가 완성
- 프로젝트 명이 MN.0. 형태로 시작되는 경우는 학습자가 직접 빈 부분을 채워 넣는 과제
- 프로젝트 명 끝이 **Prj로 끝나는** 경우도 학습자가 직접 소스 코드를 만드는 과제
- 30개가 넘는 예제
 - ◆ 한 예제에 1시간만 잡아도 무려 30시간!

❖ 이전 교육에서의 일부 의견

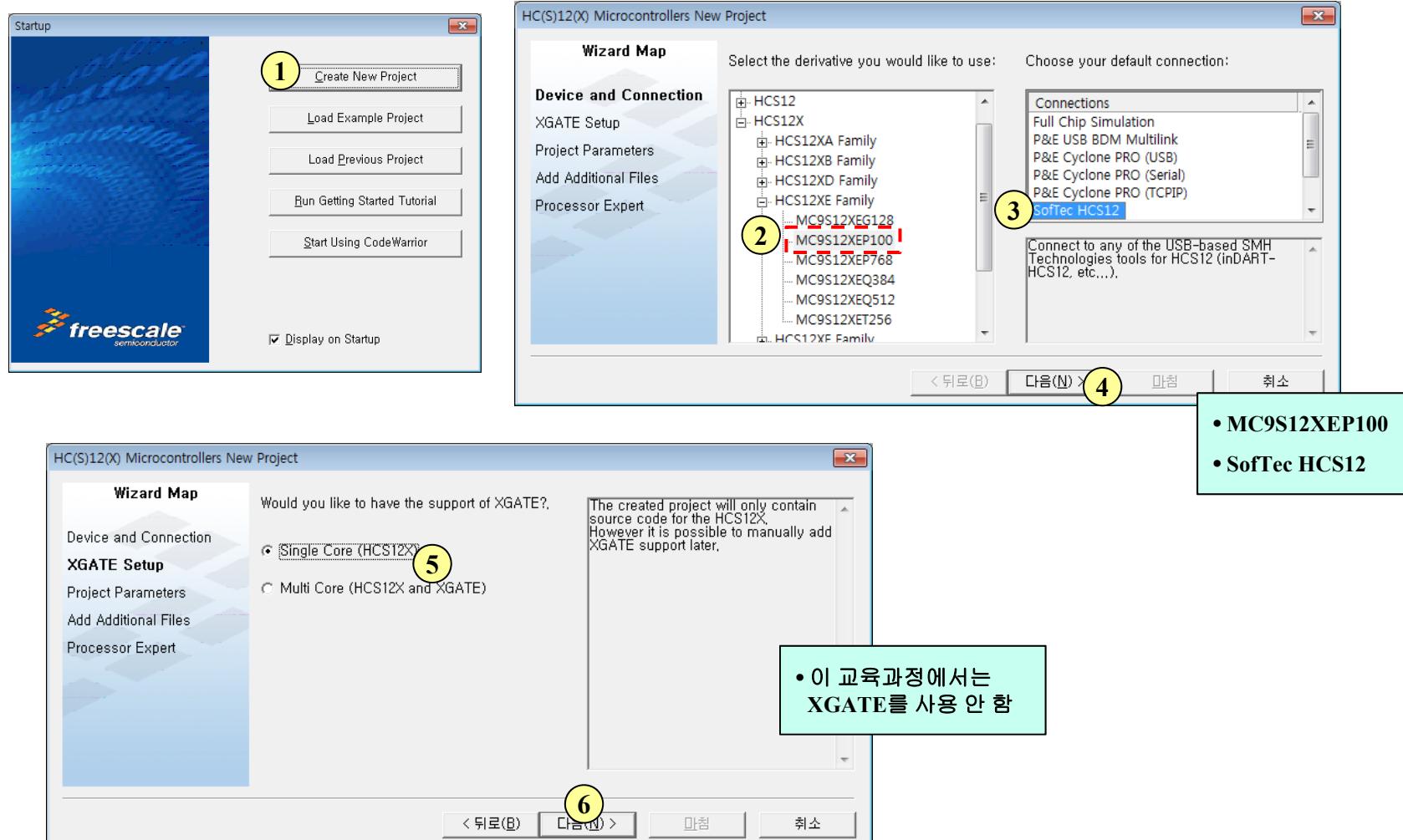
- 내가 직접 예제를 작성해 보고 싶었으나 이미 다 되어 있어서 아쉬웠다!
 - ◆ 설명만 듣고 직접 작성해 보세요!
 - ◆ **공통 과제의 핵심 부분을 여기 다룬 예제와 같은 형태로 후회 없이 소스 코드를 작성해 보세요!**
- 따라가기에 힘들었다!
 - ◆ 교육 시간이 제한되어 있어 짧은 시간에 꼭 필요한 내용을 다루기 위해 어쩔 수 없습니다.
 - ◆ 여기 있는 예제가 소스 코드 상으로는 얼마 안 되지만, 몇십 줄을 작성하기 위해 며칠이 걸리는 경우도 많았습니다. 또한, 버그 하나 잡기 위해 몇 시간을 소비한 적도 많았습니다.
 - ◆ 소스 코드 작성은 시간 가는 줄 모르는 작업! 시간이 많이 필요합니다!

00.Start – 학습 목표

- ❖ CodeWarrior를 이용해 Project를 만들 수 있다.
- ❖ 비트 연산을 이용해 특정 비트 자리를 액세스 할 수 있다.
 - 특정 비트 자리만 0 또는 1로 변경시킬 수 있다. 또한 반전시킬 수 있다.
 - 특정 비트 자리가 0인지 1인지 판단할 수 있다.
- ❖ MC9S12XEP100.h에 선언된 레지스터와 비트 매크로를 사용할 수 있다.
 - 데이터시트에 나와 있는 레지스터가 어떻게 선언되어 있는지 찾을 수 있다.
 - 선언된 레지스터와 비트 매크로를 이용해 레지스터를 액세스 할 수 있다.
- ❖ CodeWarrior의 Project 구조를 설명할 수 있다.
- ❖ COP (=WDT)가 무엇인지 설명할 수 있다.

00.Start – 프로젝트 만들기 (1)

❖ Create New Project로 Project를 만들어 보자



00.Start – 프로젝트 만들기 (2)



저장 위치

• C는 코드 제한 32KB

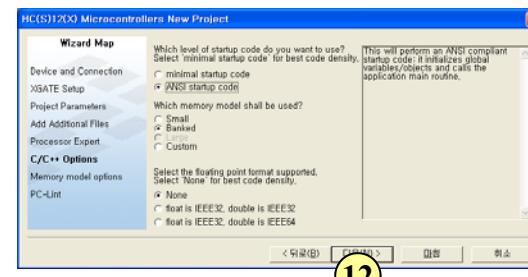
• C++는 코드 제한 1KB



10



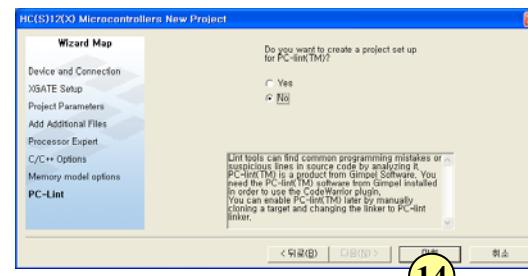
11



12



13



14

00.Start – 참고: C 언어로 특정 비트 값 변경하기

- ❖ **unsigned char data** 변수에 대해 **data**의 비트가 표와 같이 매크로 상수로 선언되어 있다면, (B_n 에서 n 은 비트 번호)

- **data** 변수의 B_1 자리만 0으로 변경

```
data &=~B1_M; #define data &=~(1<<B1_N)
```

- **data** 변수의 B_1 자리만 1으로 변경

```
data |=B1_M;
```

- **data** 변수의 B_1 자리만 반전

```
data ^=B1_M;
```

- **data** 변수의 B_0, B_3 자리만 0으로 변경

```
data &=~(B3_M|B0_M);
```

- **data** 변수의 B_0, B_3 자리만 1로 변경

```
data |=B3_M|B0_M;
```

- **data** 변수의 B_0, B_3 자리만 반전

```
data ^=B3_M|B0_M;
```

- **data** 변수의 B_0, B_1, B_2, B_3 자리에 각각 0, 1, 0, 1을 한 줄 명령어로 넣기

```
data = (data&0xf0)|0x05;
```

- **data** 변수의 B_2 자리가 1인지? 0인지?

```
if ((data & B2_M) == 0x0)
```

```
if ((data & B2_M) != 0x0)
```

- **data** 변수의 B_0, B_1 자리가 하나라도 1인지? 전부 1인지?

```
if ((data & (B1_M|B0_M)) == (B1_M|B0_M))
```

```
if ((data & (B1_M|B0_M)) != (B1_M|B0_M))
```

00.Start – I/O 레지스터 매크로 상수 선언

❖ I/O 레지스터란? (줄여서 레지스터라고도 함)

- 일반적으로 레지스터란? CPU 내부에 존재하는 메모리 (범용 Reg., PC, Flag, SP 등)
- I/O 레지스터는 MCU 내부 장치(Peripherals)를 제어할 때 사용하는 특수 메모리

❖ 레지스터 매크로 상수 선언은 어디에?

- MC9S12XEP100.h 에...

- ◆ 위치는 CodeWarrior 설치 폴더 ...
- ◆ C:\Program Files\Freescale\WCS12v5.1\lib\hc12c\include
- ◆ 여기에 선언된 매크로를 적극 이용하자!

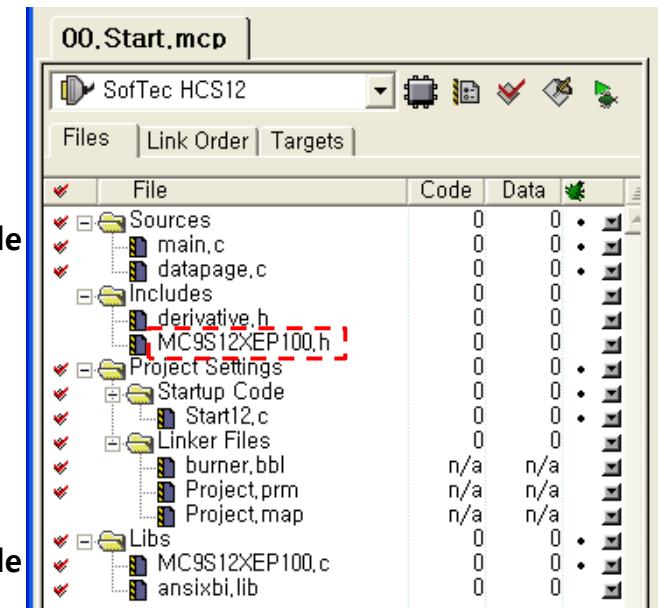
- 공통이므로...

- ◆ 이용하는 방법은?

- Reference Manual에서 사용할 Register의 이름을 이 파일에서 '찾기'로 찾아 사용

- 이 외에 hidef.h 도...

- ◆ C:\Program Files\Freescale\WCS12v5.1\lib\hc12c\include



❖ I/O 레지스터 사용시 주의점!

- I/O 레지스터는 전역변수나 다름 없다

- ◆ 전역변수 사용에 의해 발생할 수 있는 문제가 똑같이 발생
 - ◆ 여러 부분에서 동시에 Access 될 수 있다 → 특히 여러 사람이 프로그램을 함께 작성할 때 주의 필요

- H/W와 연결되어 있다

- ◆ 예) 1을 써야 0이 기록되는 레지스터도 있고, 읽기만 했는데 동작하는 레지스터도 있고... 등등

00.Start – I/O 레지스터 매크로 상수 이용 예

❖ REFDV Reg.에 0x3f 대입

- REFDV = 0x3f;

❖ REFDIV1 비트만 0으로 변경

- REFDV_REFDIV1 = 0;

❖ REFDIV5..0 비트만 19로 변경

- REFDV_REFDIV = 19;

❖ REFDIV 중 REFFRQ1..0 값만 취함

■ 방법1: REFDV_REFFRQ

- ◆ 비트 위치를 오른쪽으로 정렬한 값
- ◆ 예) REFFRQ1..0이 10b라면 10b

■ 방법2: REFDV & REFDV_REFFRQ_MASK

- ◆ 원래의 비트 위치에서의 값
- ◆ 예) REFFRQ1..0이 10b라면 10000000b

■ 방법3: (REFDV & REFDV_REFFRQ_MASK) >> REFDV_REFFRQ_BITNUM

- ◆ REFDV_REFFRQ와 같은 값

```
/* *** REFDV - S12XECRG Reference Divider Register; 0x00000035 ***/
typedef union {
    byte Byte;
    struct {
        byte REFDIV0 : 1;
        byte REFDIV1 : 1;
        byte REFDIV2 : 1;
        byte REFDIV3 : 1;
        byte REFDIV4 : 1;
        byte REFDIV5 : 1;
        byte REFFRQ0 : 1;
        byte REFFRQ1 : 1;
    } Bits;
    struct {
        byte grpREFDIV : 6;
        byte grpREFFRQ : 2;
    } MergedBits;
} REFDVSTR;
extern volatile REFDVSTR _REFDV @(REG_BASE + 0x00000035UL);
#define REFDV           _REFDV.Byte
#define REFDV_REFDIV0   _REFDV.Bits.REFDIV0
#define REFDV_REFDIV1   _REFDV.Bits.REFDIV1
#define REFDV_REFDIV2   _REFDV.Bits.REFDIV2
#define REFDV_REFDIV3   _REFDV.Bits.REFDIV3
#define REFDV_REFDIV4   _REFDV.Bits.REFDIV4
#define REFDV_REFDIV5   _REFDV.Bits.REFDIV5
#define REFDV_REFFRQ0   _REFDV.Bits.REFFRQ0
#define REFDV_REFFRQ1   _REFDV.Bits.REFFRQ1
#define REFDV_REFDIV    _REFDV.MergedBits.grpREFDIV
#define REFDV_REFFRQ    _REFDV.MergedBits.grpREFFRQ

#define REFDV_REFDIV0_MASK 1U
#define REFDV_REFDIV1_MASK 2U
#define REFDV_REFDIV2_MASK 4U
#define REFDV_REFDIV3_MASK 8U
#define REFDV_REFDIV4_MASK 16U
#define REFDV_REFDIV5_MASK 32U
#define REFDV_REFFRQ0_MASK 64U
#define REFDV_REFFRQ1_MASK 128U
#define REFDV_REFDIV_BITNUM 63U
#define REFDV_REFFRQ_BITNUM 0U
#define REFDV_REFFRQ0_MASK 192U
#define REFDV_REFFRQ1_MASK 6U
```

00.Start – 기본 제공 파일

❖ main.c

- 프로그램의 실질적 시작점인 main 함수가 있는 파일

❖ Datapage.c

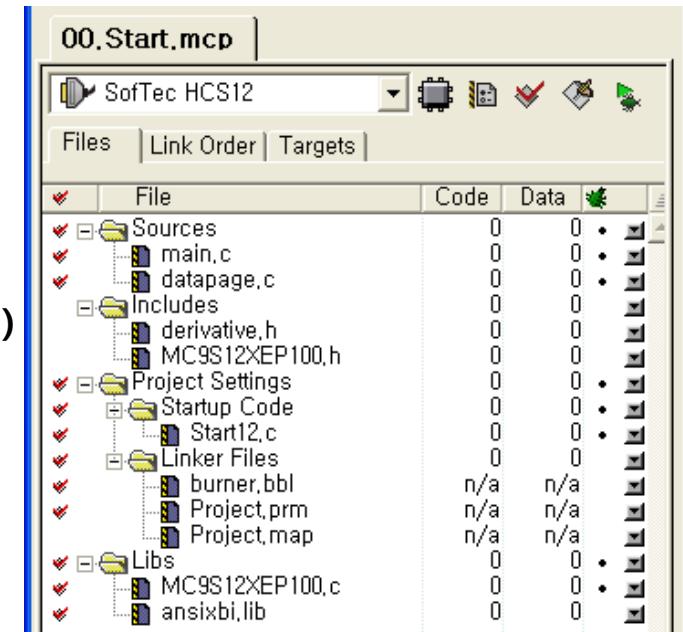
- 적은 Memory Address로 많은 Memory 이용 시 사용
- 참고: Widow와 Page 개념 (Ref. Manual p.35 그림 참조)
- 우리는 사용 안 하므로 설명 생략

❖ derivative.h

- #include <MC9S12XEP100.h> 만 있음

❖ Start12.c (보통 Startup 파일이라고 함)

- 제일 먼저 실행되는 프로그램
- main 함수보다 먼저 실행됨
- main 함수를 실행하기 전 MCU를 초기화
 - ◆ 메모리 초기화
 - Global Variables, Stack, Interrupt Vector Table, Page 등
 - ◆ 기타 여러 초기화를 담당
 - COP(Computer Operating Properly) WDT(Watch Dog Timer)



00.Start – main.c

```
#include <hidef.h>      /* common defines and macros */  
#include "derivative.h"    /* derivative-specific definitions */  
  
void main(void) {  
    /* put your own code here */  
  
    EnableInterrupts;  
  
    for(;;){  
        _FEED_COP(); /* feeds the dog */  
    } /* loop forever */  
    /* please make sure that you never leave main */  
}
```

• Interrupt 허용
• 어디에 선언되어 있을까?
hidef.h

• COP 건드려 주기
• 어디에 선언되어 있을까?
• 여기서는 의미 없음
• 왜?
hidef.h

• 일반적으로 EnableInterrupts 이전에 MCU 초기화가 실행되도록 할 것!
• EnableInterrupts 이후에 실질적인 프로그램 동작이 이루어지도록!

• 왜?

• WDT(Watch Dog Timer) 란?
- 일정 시간 동안 Watch Dog를 건드려 주지 않으면 MCU를 Reset 시켜주는 타이머
- MCU가 Hang 된 경우 MCU를 다시 시작시키기 위한 안전 장치라고 보면 됨!
- 대부분의 MCU에 이 기능 존재
- Clock이 필요하며, 어떤 MCU는 Clock이 멈추는 것 조차 대비해 별도 Clock을 이용하기도 함

PIM – 개요

❖ General Purpose I/O (GPIO)

- Reference Manual에서는 Port Integration Module (PIM)이라 함
- GPIO란 디지털 입출력 포트를 말함
 - ◆ 디지털 신호 (H/L)를 입력 받거나 출력할 때 사용
 - ◆ 내부 Pull-up 저항, Pull-down 저항을 연결해 사용하기도 함
 - ◆ 이 MCU는 reduce drive output과 interrupt 기능도 있음
- 레지스터 사용법은 Reference Manual Chapter 2 참조
 - ◆ 소스 코드 작성시 참조해야 함
 - ◆ Table 2-103, Figure 2-107, Table 2-3으로 전반적인 기능 파악
- 전기적 특성은 Reference Manual A.1 참조
 - ◆ 회로 설계할 때 반드시 참조해야 함
 - ◆ 찾아보기1: I/O 핀 하나 당 최대 전류는?
 - ◆ 찾아보기2: 디지털 출력 H의 최소 전압은?
 - ◆ 찾아보기3: Pull-up 저항의 저항값 범위는?

전류 25mA, 5V 전압 허용 3.25V, 25~50kΩ

PIM – Pin Configuration

❖ 방향: DDR이 1이면 출력, 0이면 입력

- 입력인 경우, Pull-Up 저항 또는 Pull-Down 저항 연결 가능
- 입력인 경우, Falling Edge나 Rising Edge에서 인터럽트 가능
- 출력인 경우, Full Drive와 Reduced Drive 가능

Table 2-3. Pin Configuration Summary

DDR	IO	RDR	PE	PS ⁽¹⁾	IE ⁽²⁾	Function	Pull Device	Interrupt
0	x	x	0	x	0	Input	Disabled	Disabled
0	x	x	1	0	0	Input	Pull Up	Disabled
0	x	x	1	1	0	Input	Pull Down	Disabled
0	x	x	0	0	1	Input	Disabled	Falling edge
0	x	x	0	1	1	Input	Disabled	Rising edge
0	x	x	1	0	1	Input	Pull Up	Falling edge
0	x	x	1	1	1	Input	Pull Down	Rising edge
1	0	0	x	x	0	Output, full drive to 0	Disabled	Disabled
1	1	0	x	x	0	Output, full drive to 1	Disabled	Disabled
1	0	1	x	x	0	Output, reduced drive to 0	Disabled	Disabled
1	1	1	x	x	0	Output, reduced drive to 1	Disabled	Disabled
1	0	0	x	0	1	Output, full drive to 0	Disabled	Falling edge
1	1	0	x	1	1	Output, full drive to 1	Disabled	Rising edge
1	0	1	x	0	1	Output, reduced drive to 0	Disabled	Falling edge
1	1	1	x	1	1	Output, reduced drive to 1	Disabled	Rising edge

1. Always "0" on Port A, B, C, D, E, K, AD0, and AD1.

2. Applicable only on Port P, H, and J.

- DDR: Data Direction Register
- IO은 Output Level
- RDR: Reduced Drive
- PE: Pull Enable
- PS: Pull Select
- IE: Interrupt Enable

PIM – Availability

- ❖ 포트마다 기능의 차이가 있으니 용도에 맞게 골라 사용!

Table 2-103. Register availability per port⁽¹⁾

Port	Data	Input	Data Direction	Reduced Drive	Pull Enable	Polarity Select	Wired-Or Mode	Interrupt Enable	Interrupt Flag	Routing
A	yes	-	yes	yes	yes	-	-	-	-	-
B	yes	-	yes			-	-	-	-	-
C	yes	-	yes			-	-	-	-	-
D	yes	-	yes			-	-	-	-	-
E	yes	-	yes			-	-	-	-	-
K	yes	-	yes			-	-	-	-	-
T	yes	yes	yes	yes	yes	yes	-	-	-	-
S	yes	yes	yes	yes	yes	yes	yes	-	-	yes
M	yes	yes	yes	yes	yes	yes	yes	-	-	yes
P	yes	yes	yes	yes	yes	yes	-	yes	yes	-
H	yes	yes	yes	yes	yes	yes	-	yes	yes	-
J	yes	yes	yes	yes	yes	yes	-	yes	yes	-
AD0	yes	-	yes	yes	yes	-	-	-	-	-
AD1	yes	-	yes	yes	yes	-	-	-	-	-
R	yes	yes	yes	yes	yes	yes	-	-	-	-
L	yes	yes	yes	yes	yes	yes	yes	-	-	yes
F	yes	yes	yes	yes	yes	yes	-	-	-	yes

1. Each cell represents one register with individual configuration bits

PIM – Register Name

- ❖ Data Register: PORTx PTx
- ❖ Input Register: PTIx
- ❖ Data Direction Register: DDRx
- ❖ Reduced Drive Register: RDRx
- ❖ Pull Device Enable Register: PERx
- ❖ Polarity Select Register: PPSx
- ❖ Wired-or Mode Register: WOMx
- ❖ Interrupt Enable Register: PIE_x
- ❖ Interrupt Flag Register: PIE_x

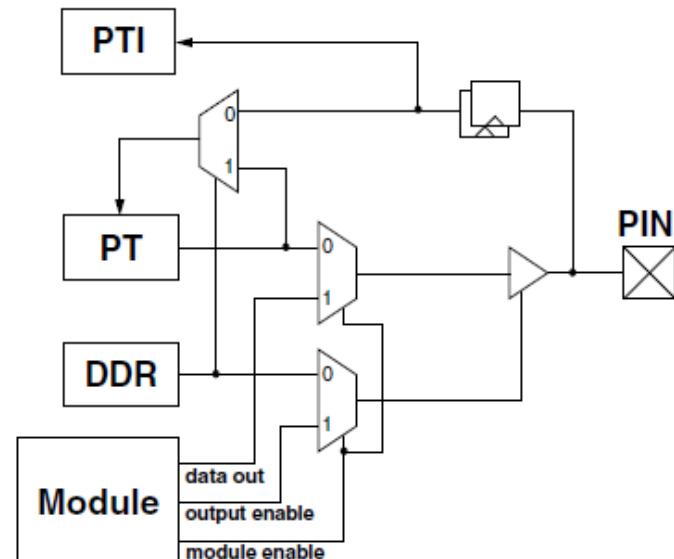


Figure 2-107. Illustration of I/O pin functionality

- 앞 페이지의 Availability를 참고하여 포트를 정하고,
- 정한 포트에 대한 레지스터는 이름은 이 페이지를 참고하며,
- 정확한 사용법은 Reference Manual을 참고!

01.LedBlinking – 학습 목표 및 문제 정의, 회로도

❖ 학습 목표

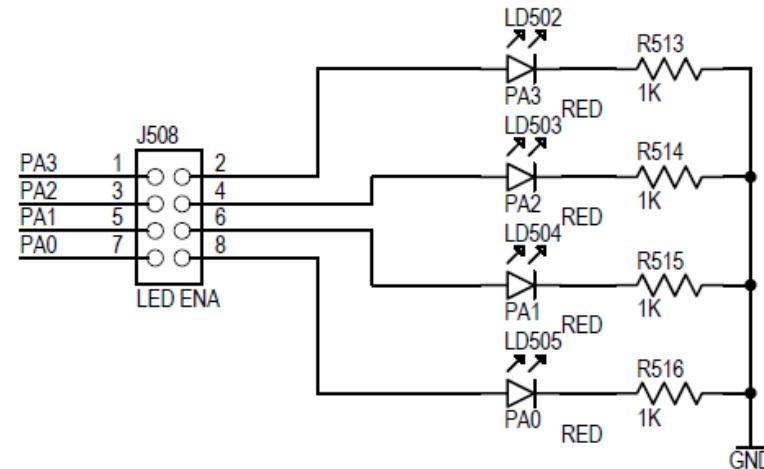
- GPIO를 통해 디지털 전압을 출력할 수 있다. → 출력 전압을 이용해 LED ON/OFF
- CodeWarrior를 이용해 컴파일, 다운로딩, 디버깅을 할 수 있다.

❖ 문제 정의: PA3~PA0에 연결된 LED를 다음과 같이 동작시킴

1. PA3~PA0 LED 4개 전부 0.25초 동안 켜 후, 0.25초 동안 끄
2. PA3 LED만 0.25초 켜 후, 0.25초 끄
3. 1~2 과정을 무한 반복

❖ 회로도

- PA0~PA3을 LED ON/OFF에 안 쓰려면?
- Digital High 전압이 5V이고, LED forward voltage가 2.2V라면 LED ON시 흐르는 전류는?



전류 = 5V / 2.2V = 2.27mA

(5-2.2)/1 = 2.8mA

01.LedBlinking – 시간 지연 함수

❖ 시간 지연 함수를 만들어 보자

```
// ms millisecond 동안 시간을 지연시키는 함수  
void delay_ms(unsigned int ms) {  
    volatile int i;  
  
    while (ms--) {  
        for (i=0; i<332; i++) {  
        }  
    }  
}
```

• 이 숫자를 적당히 조절하여 while 한 루프가 1msec가 되도록 정함

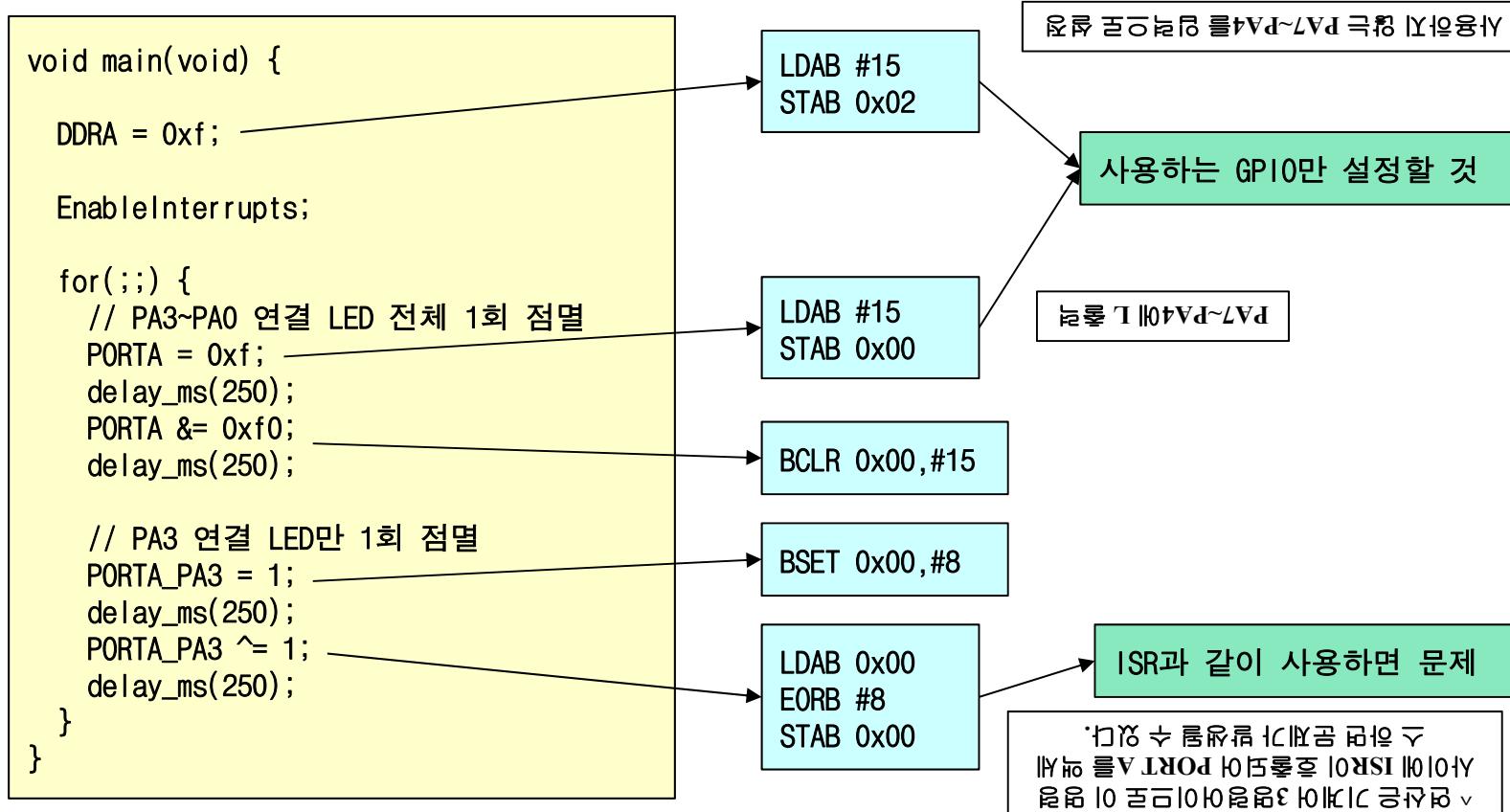
❖ 문제가 있다면?

- Clock이 변경되면(= bus frequency가 변경되면) for 루프의 반복 횟수도 변경시켜야 한다
- while 루프 중에 ISR(Interrupt Service Routine)이 실행되면 지연시간이 ISR 실행시간 만큼 길어진다

01.LedBlinking – 구현

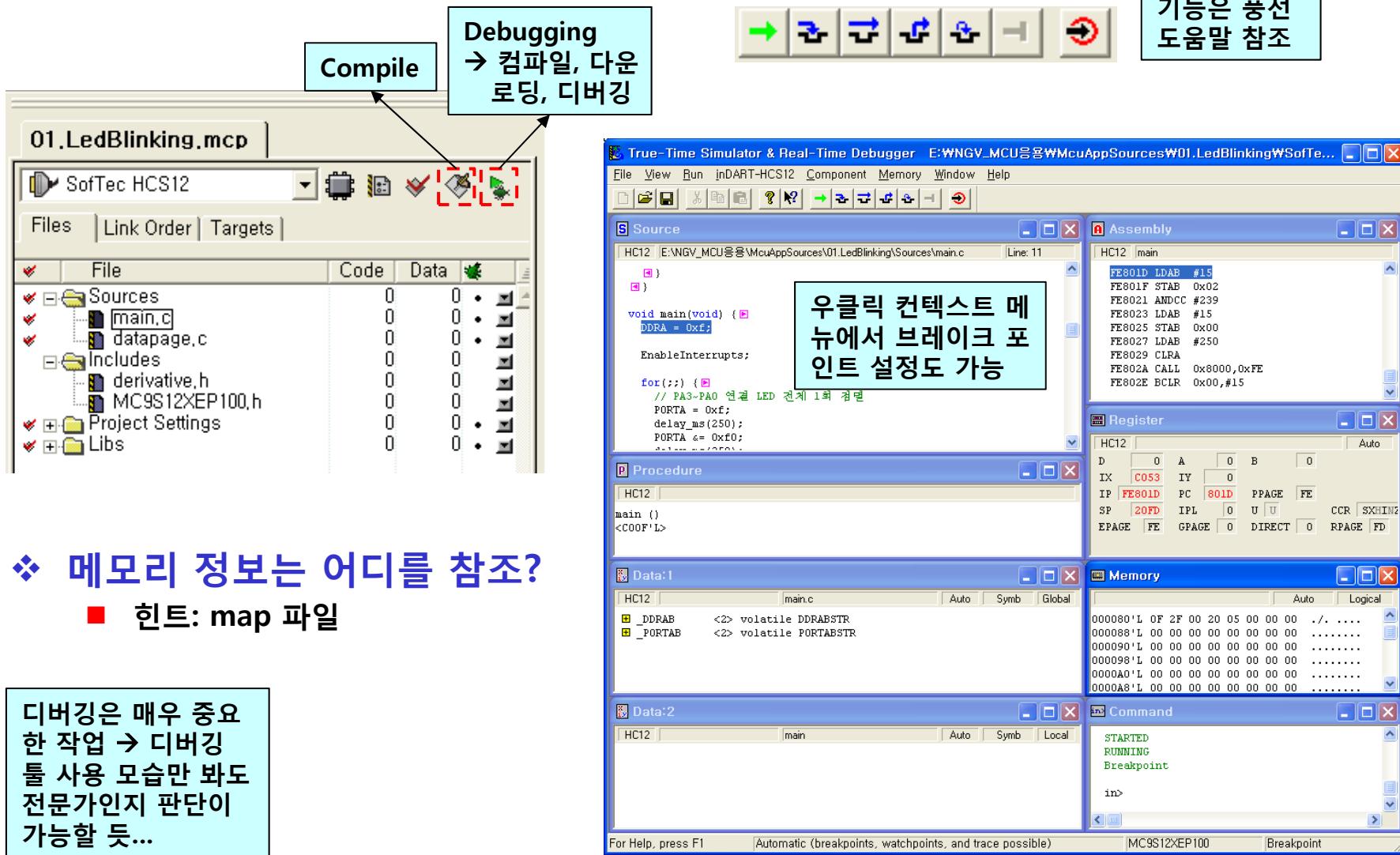
- ❖ 소스 코드
- ❖ 문제가 있다면?
 - LED 점멸 목적으로는 전혀 문제가 없으나
 - 이런 식의 코드는 실전 코드에서 문제를 야기할 수 있다. 왜?

어셈블리로 번역된 결과는 C 파일을 우 클릭해서 팝업되는 콘텍스트 메뉴에서 Disassemble을 선택하면 볼 수 있다!



01.LedBlinking – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기



❖ 메모리 정보는 어디를 참조?

■ 힌트: map 파일

디버깅은 매우 중요한 작업 → 디버깅 툴 사용 모습만 봐도 전문가인지 판단이 가능할 듯...

02.LedBlinkingPrj0 – 과제

❖ 문제

- PA3~PA0 LED 4개 전부 0.25초 동안 켜 후, 0.25초 동안 끄
 - PAn LED만 0.25초 켜 후, 0.25초 끄 (n은 0, 1, 2, 3을 반복)
 - 1~2 과정을 무한 반복

필수

- ## ■ 디버깅 기능을 활용해 보기

풀이

```

void main(void) {
    int n=0;
    DDRA |= 0xf;
    EnableInterrupts();
    // PA3~PA0 が LED を制御する
    // PA3~PA0 の LED 制御
    PORTA |= 0xf0;
    delay_ms(250);
    PORTA |= 0x10;
    delay_ms(250);
    PORTA |= ~((1<<n));
    delay_ms(250);
    PORTA |= ((1<<n));
    delay_ms(250);
    PORTA |= ~((1<<n));
    delay_ms(250);
    PORTA |= ((1<<n));
    delay_ms(250);
}

```

03.BusFrequency – 개요

❖ 학습 목표

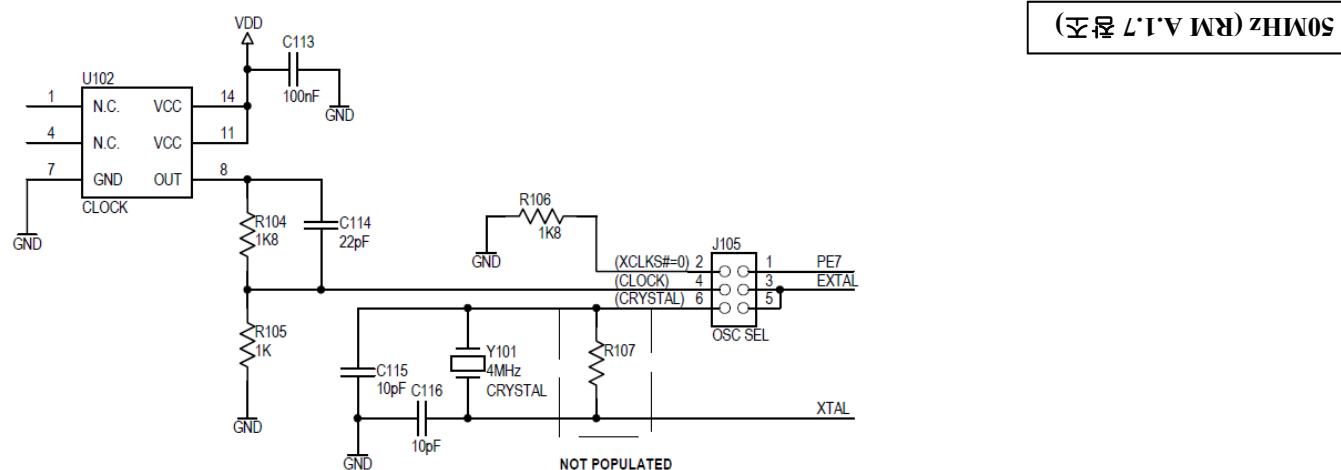
- Bus Frequency를 설정하고 사용할 수 있다.

❖ CPU Clock

- 예전 CPU는 CPU에 공급되는 Clock 속도로 동작했으나
- 최근 CPU는 내부에서 PLL로 공급 Clock 이상의 주파수로 CPU를 동작시킬 수 있음

❖ DEMO9S12XEP100 보드

- 4MHz Crystal Oscillator를 사용
 - ◆ 기본적으로 Bus Frequency는 이 주파수의 반 = 2MHz
- 문제: MC9S12XEP100의 최대 CPU Bus Frequency(=Core Freq./2 = 1/M.C.)는?



03. Bus Frequency – 설정

❖ 공식: Reference Manual Chapter 11 참조

- $f_{VCO} = 2 \times f_{OSC} \times (\text{SYNDIV} + 1) / (\text{REFDIV} + 1)$
 - ◆ SYNDIV: 0~63, REFDIV: 0~63, f_{VCO} : 32~120MHz
- $f_{PLL} = f_{VCO} / (2 \times \text{POSTDIV})$
 - ◆ POSDIV: 0~31 (IF POSDIV==0 THEN $f_{PLL} = f_{VCO}$)
- $f_{BUS} = f_{PLL}/2$
- $f_{REF} = f_{OSC}/(\text{REFDIV} + 1)$

❖ 그렇다면, (Reference Manual Table 11-14 참조)

조립 회로 설계 공학

- 20MHz는?
- 40MHz는?

Table 11-14. Examples of IPLL Divider Settings⁽¹⁾

f_{osc}	REFDIV[5:0]	f_{REF}	REFFRQ[1:0]	SYNDIV[5:0]	f_{vco}	VCOFRQ[1:0]	POSTDIV[4:0]	f_{PLL}	f_{BUS}
4MHz	\$01	2MHz	01	\$18	100MHz	11	\$00	100MHz	50 MHz
8MHz	\$03	2MHz	01	\$18	100MHz	11	\$00	100MHz	50 MHz
4MHz	\$00	4MHz	01	\$09	80MHz	01	\$00	80MHz	40MHz
8MHz	\$00	8MHz	10	\$04	80MHz	01	\$00	80MHz	40MHz
4MHz	\$00	4MHz	01	\$03	32MHz	00	\$01	16MHz	8MHz
4MHz	\$01	2MHz	01	\$18	100MHz	11	\$01	50MHz	25MHz

1. f_{PLL} and f_{BUS} values in this table may exceed maximum allowed frequencies for some devices.
Refer to device information for maximum values.

03.BusFrequency – 선행처리기 #if 의 활용

- ❖ 선행처리기를 잘 활용하면 효과적인 소스 코드 관리가 가능
 - 그러나 선행처리기를 이용한 소스 코드 관리의 지나친 사용은 소스 코드를 복잡하게 만들어 버그를 유발할 수도...
- ❖ #if ~ #endif
- ❖ #if ~ #else ~ #endif
- ❖ #if ~ #elif ~ #elif ~ ... #endif
- ❖ #ifdef ~ #endif
- ❖ #ifndef ~ #endif

03.BusFrequency – 문제 정의 및 풀이 (1)

- ❖ 문제 정의: Bus Frequency를 2MHz, 20MHz, 40MHz로 선택해서 사용할 수 있도록 하고 PA3 연결 LED를 0.25초 간격으로 토글

```
// 사용하려는 Bus Frequency에 따라 BFRQ_OPTION을 편집
#define BFRQ_2MHZ    0 // f_BUS = f_OSC/2
#define BFRQ_20MHZ   1 // f_BUS = f_PLL/2
#define BFRQ_40MHZ   2 // f_BUS = f_PLL/2
#define BFRQ_OPTION BFRQ_20MHZ // 추후 이 부분만 변경하면 됨

#if BFRQ_OPTION == BFRQ_20MHZ // f_VCO=80M, f_PLL=40M, f_BUS=20MHz
#define BFRQ_SYNDIV
#define BFRQ_REFDIV
#define BFRQ_POSTDIV
#define BFRQ_VCOFRQ
#define BFRQ_REFFRQ
#elif BFRQ_OPTION == BFRQ_40MHZ // f_VCO=80M, f_PLL=80M, f_BUS=40MHz
#define BFRQ_SYNDIV
#define BFRQ_REFDIV
#define BFRQ_POSTDIV
#define BFRQ_VCOFRQ
#define BFRQ_REFFRQ
#endif
```

```
void BFRQ_Init(void) {
    #if BFRQ_OPTION != BFRQ_2MHZ
        CLKSEL_PLLSEL = 0;
        PLLCTL = PLLCTL_CME_MASK|PLLCTL_SCME_MASK;
        SYNR_SYNDIV = BFRQ_SYNDIV;
        SYNR_VCOFRQ = BFRQ_VCOFRQ;
        REFDV_REFDIV = BFRQ_REFDIV;
        REFDV_REFFRQ = BFRQ_REFFRQ;
        POSTDIV = BFRQ_POSTDIV;
        PLLCTL = PLLCTL_CME_MASK|PLLCTL_PLLON_MASK
            |PLLCTL_SCME_MASK;
        while (!CRGFLG_LOCK) {
        }
        CRGFLG = CRGFLG_LOCKIF_MASK;
        CLKSEL_PLLSEL = 1;
    #endif
}
```

실습: 점선 사각형 안에 들어갈 값을 Reference Manual을 참고하여 제공되는 “03.0.BusFrequency” 프로젝트의 main.c에 채워 넣기. (제대로 설정해야 BFRQ_OPTION 값이 상관 없이 PA3 LED가 2회/초 점멸)

03.BusFrequency – 문제 정의 및 풀이 (2)

- ❖ Bus Frequency를 2MHz, 20MHz, 40MHz로 선택해서 사용할 수 있도록 하고 PA3 연결 LED를 0.25초 간격으로 토글

```
// ms millisec. 동안 시간을 지연시키는 함수
void delay_ms(unsigned int ms) {
    int i;

    while (ms--) {
#if BFRQ_OPTION == BFRQ_2MHZ
        for (i=0; i<332; i++) {
#elif BFRQ_OPTION == BFRQ_20MHZ
        for (i=0; i<332; i++) {
#elif BFRQ_OPTION == BFRQ_40MHZ
        for (i=0; i<6665; i++) {
#endif
    }
}
```

```
void main(void) {

    BFRQ_Init();

    DDRA |= 0xf;

    EnableInterrupts;

    for(;;) {
        PORTA_PA3 ^= 1;
        delay_ms(250);
    }
}
```

- 이후 모든 실습 코드는 20MHz를 사용하나 Bus Frequency를 변경해도 시간 관련 부분이 정상 동작하도록 함!

03. Bus Frequency – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- Bus Frequency를 변경하고, delay_ms 함수 내부의 for 루프 횟수는 그대로 두어 Bus Frequency의 영향을 살펴보기
- Crystal Oscillator의 파형을 오실로스코프로 측정해 보기
- Bus Frequency와 관련된 파형을 오실로스코프로 측정해 보기
 - ◆ 어느 핀을 측정하면 될까?
 - ◆ 측정을 위해서 Register 설정이 필요할까?
 - ◆ Reference Manual을 참고하여 측정해 보세요!

04.BFRQSplitIntoFiles – 개요, project.h, project.c

❖ 학습 목표

- 프로그램을 여러 파일로 나누어 작성할 수 있다.
- 즉, 기능별로 .c와 .h를 만들어 사용할 수 있다.

❖ 문제 정의

- 03.BusFrequency에서 Bus Frequency와 관련된 부분을 busfreq.c 와 busfreq.h 파일로 구현

❖ project.h

- 이 헤더 파일에 공통으로 사용되는 매크로를 선언!

- project.h

```
#ifndef _PROJECT_H_
#define _PROJECT_H_

#ifndef __cplusplus
extern "C" {
#endif

#include <hidef.h>
#include <stdtypes.h>
#include "derivative.h"

// 인터럽트 처리를 위한 매크로 함수
extern volatile Byte CCR_reg;
#define EnterCriticalSection() {__asm(pshc); __asm(sei); __asm(movb 1,SP+,CCR_reg); }
#define LeaveCriticalSection() {__asm(movb CCR_reg, 1,-SP); __asm(pulc); }

// 사용하려는 Bus Frequency에 따라 BFRQ_OPTION을 편집
#define BFRQ_2MHZ 0 // f_BUS = f_OSC/2
#define BFRQ_20MHZ 1 // f_BUS = f_PLL/2
#define BFRQ_40MHZ 2 // f_BUS = f_PLL/2
#define BFRQ_OPTION BFRQ_20MHZ

#ifndef __cplusplus
}
#endif

#endif
```

필수 헤더 파일

- project.c

```
#include "project.h"
volatile Byte CCR_reg;
```

이 부분은 인터
럽트에서 설명

Bus Frequency 관련해서는 추후 이 부분만 변경하면 됨

04.BFRQSplitIntoFiles – 문제 정의 및 풀이

- ❖ 03.BusFrequency에서 Bus Frequency에 관련된 부분을 파일로 분리
 - 앞으로 기능별로 파일을 분리해서 작성할 것!

- busfreq.h

```
#ifndef __BUSDREQ_H__
#define __BUSDREQ_H__

#ifndef __cplusplus
extern "C" {
#endif

#include "project.h"

// MACRO -----
#if BFRQ_OPTION == BFRQ_2MHZ
#define BFRQ_HZ 2000000UL
#elif BFRQ_OPTION == BFRQ_20MHZ // f_VCO=80M, f_PLL=40M, f_BUS=20MHz
#define BFRQ_HZ 20000000UL
#define BFRQ_SYNDIV 9
#define BFRQ_REFDIV 0
#define BFRQ_POSTDIV 1
#define BFRQ_VCOFRQ 1
#define BFRQ_REFFRQ 1
#elif BFRQ_OPTION == BFRQ_40MHZ // f_VCO=80M, f_PLL=80M, f_BUS=40MHz
#define BFRQ_HZ 40000000UL
#define BFRQ_SYNDIV 9
#define BFRQ_REFDIV 0
#define BFRQ_POSTDIV 0
#define BFRQ_VCOFRQ 1
#define BFRQ_REFFRQ 1
#endif

// FUNCTION -----
void BFRQ_Init(void);

#ifndef __cplusplus
}
#endif
#endif
```

- busfreq.c

```
#include "derivative.h"
#include "busfreq.h"

// Bus Frequency를 설정하는 함수
void BFRQ_Init(void) {

    #if BFRQ_OPTION != BFRQ_2MHZ
    CLKSEL_PLLSEL = 0;
    PLLCTL = PLLCTL_CME_MASK|PLLCTL_SCME_MASK;
    SYNR_SYNDIV = BFRQ_SYNDIV;
    SYNR_VCOFRQ = BFRQ_VCOFRQ;
    REFDV_REFDIV = BFRQ_REFDIV;
    REFDV_REFFRQ = BFRQ_REFFRQ;
    POSTDIV = BFRQ_POSTDIV;
    PLLCTL = PLLCTL_CME_MASK|PLLCTL_PLLON_MASK
        |PLLCTL_SCME_MASK;
    while (!CRGFLG_LOCK) {
    }
    CRGFLG = CRGFLG_LOCKIF_MASK;
    CLKSEL_PLLSEL = 1; // PLL 이용
    #endif
}
```

- main.c

```
#include <hedef.h>
#include "derivative.h"
#include "project.h"
#include "busfreq.h"

/////////// 중간 생략 ...
void main(void) {
    BFRQ_Init();
```

04.BFRQSplitIntoFiles – 컴파일 및 디버깅

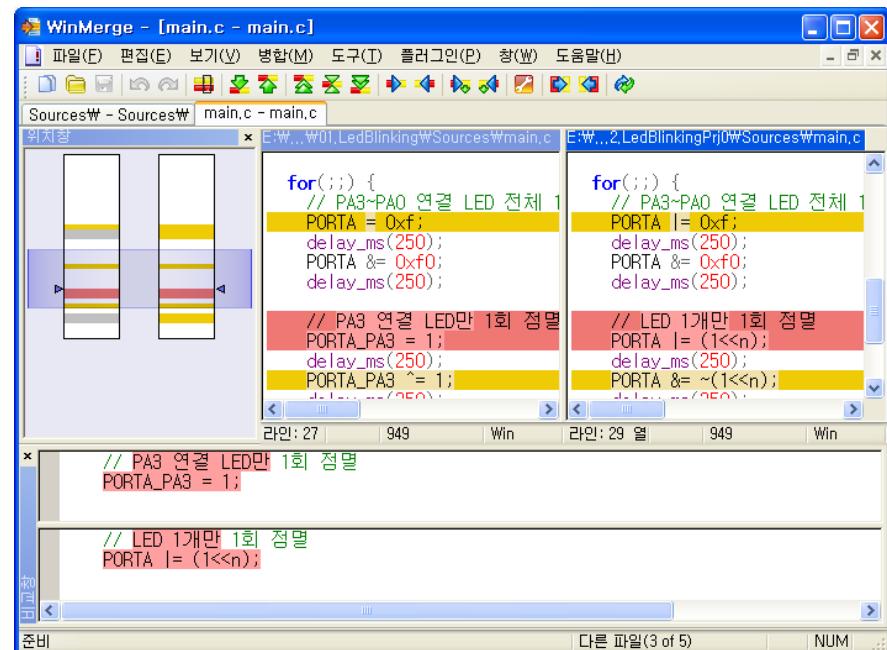
❖ 실습: 컴파일하고 디버깅 해보기

- 04.MySplitIntoFiles 프로젝트를 만들고
- 03.BusFrequency에서 Bus Frequency 관련 내용을 busfreq.h와 busfreq.c로 분리한 후
- 실행시켜 보고,
- 04.BFRQSplitIntoFiles 프로젝트의 소스 코드와 WinMerge로 비교해보기

❖ WinMerge

- 무료 소프트웨어로 Google에서 검색하여 설치
- 두 파일의 차이점을 보면서 편집도 가능한 유ти리티
- 소스 코드 관리에 매우 유용
- 사용법 매우 간단

❖ 느낀 점 및 참고사항 적기



정수형은 이렇게 사용

❖ C 언어 정수형의 단점

- 정수형의 길이와 부호가 컴파일러마다 다르다.
 - ◆ int 형: 16비트, 32비트 존재, 컴파일 옵션에 따라 unsigned int 또는 signed int 가 될 수 있다.
 - ◆ char 형이 16비트인 컴파일러도 있다.

❖ 그래서 C99 표준에서는 `stdint.h` 제공

- `uint8_t`, `int8_t` 등 부호와 길이가 명확한 정수형을 `typedef`로 제공

❖ 또는 회사마다 `typedef`을 이용해 부호와 길이가 명확한 정수형을 사용

❖ 우리가 사용하는 CodeWarrior에서는

- `stdint.h` 못찾음
- 대신 `stdtypes.h` 제공

❖ 앞으로 실습에 `stdtypes.h` 제공 정수형을 사용하자

- `Byte`, `sByte`, `Word`, `sWord`, `LWord`, `sLWord` 제공 → 각각의 부호 및 길이는?
- `uchar`, `schar`, `uint`, `sint`, `ulong`, `slong`도 제공
- `TRUE`, `FALSE` 제공
- `project.h`에 `#include <stdtypes.h>`를 추가함

8, 16, 32bit

05.DemoLed – 개요 및 문제 정의

❖ 학습 목표

- DEMO 보드의 LED를 제어할 수 있는 필수 함수를 만들 수 있다.

❖ 문제 정의 및 구상: DEMO 보드의 LED를 활용할 수 있는 함수 만들기

- 전역변수 (멤버 변수로...)
 - ◆ LED와 관련된 레지스터

- LED를 구동할 수 있는 초기화 함수
 - ◆ void DLED_Init(void);

- LED를 개별적으로 제어하는 함수 필요
 - ◆ void DLED_SetBits(Byte leds);
 - ◆ void DLED_ResetBits(Byte leds);
 - ◆ void DLED_ToggleBits(Byte leds);
 - ◆ void DLED_SetResetBits(Byte sleds, Byte rleds);
 - ◆ Byte DLED_GetBits(Byte leds);

- LED 전체를 제어하는 함수 필요
 - ◆ void DLED_Put(Byte val);
 - ◆ Byte DLED_Get(void);

- 다른 기능의 함수가 더 필요할까?

• demoled.h

```
#ifndef __DEMOLED_H__
#define __DEMOLED_H__

#if __cplusplus
extern "C" {
#endif

#include "project.h"

//-- MACRO -----
#define DLED_BV0 0x1
#define DLED_BV1 0x2
#define DLED_BV2 0x4
#define DLED_BV3 0x8
#define DLED_MASK 0xf

//-- FUNCTION -----
void DLED_Init(void);
void DLED_SetBits(Byte leds);
void DLED_ResetBits(Byte leds);
void DLED_ToggleBits(Byte leds);
void DLED_SetResetBits(Byte sleds, Byte rleds);
Byte DLED_GetBit(Byte led);
void DLED_Put(Byte val);
Byte DLED_Get(void);

#endif //__DEMOLED_H__
```

05.DemoLed – main.c

- ❖ 앞으로의 예제는 main.c가 다음과 같은 형태
- ❖ LED를 On/Off 해보는 예제

• main.c

```
#include "project.h"
#include "busfreq.h"
#include "demoled.h"

void InitPeripherals(void) {
    BFRQ_Init();
    DLED_Init();
    EnableInterrupts;
}

// ms millisecond 동안 시간을 지연시키는 함수
void delay_ms(unsigned int ms) {
    int i;

    while (ms--) {
        #if BFRQ_OPTION == BFRQ_2MHZ
            for (i=0; i<332; i++) {
        #elif BFRQ_OPTION == BFRQ_20MHZ
            for (i=0; i<332; i++) {
        #elif BFRQ_OPTION == BFRQ_40MHZ
            for (i=0; i<6665; i++) {
        #endif
        }
    }
}
```

```
void main(void) {
    int i;
    InitPeripherals();
    for (;;) {
        // 차례대로 하나씩 ON
        DLED_SetBits(DLED_LED0_BV);
        delay_ms(300);
        DLED_SetBits(DLED_LED1_BV);
        delay_ms(300);
        DLED_SetBits(DLED_LED2_BV);
        delay_ms(300);
        DLED_SetBits(DLED_LED3_BV);
        delay_ms(300);

        // 두 개씩 OFF
        DLED_ResetBits(DLED_LED3_BV|DLED_LED2_BV);
        delay_ms(300);
        DLED_ResetBits(DLED_LED1_BV|DLED_LED0_BV);
        delay_ms(300);

        // 전체 ON/OFF 2회
        for (i=0; i<2; i++) {
            DLED_Put(DLED_LED_MASK);
            delay_ms(500);
            DLED_Put(0x0);
            delay_ms(500);
        }
    }
}
```

05.DemoLed – 구현 실습

❖ 구현 실습

- 제공되는 05.0.DemoLed 프로젝트에서 demoled.c를 완성
- 완성 후 05.DemoLed와 비교해 볼 것!
- LED 함수 활용 예제를 main 함수에 만들어 볼 것!

• demoled.c

❖ 참고1: EnterCriticalSection, LeaveCriticalSection

- EnterCriticalSection();과 LeaveCriticalSection(); 사이에 코드를 작성해 넣을 것
- 내용은 추후 설명

❖ 참고2: Remove Object Code... 메뉴

- 변경된 파일만을 컴파일 후 링크
- 파일의 변경 유무는 object 파일과 source 파일의 날짜를 기준으로 함
- 컴퓨터의 시간 정보가 잘못 되어 생성되어 있는 object 파일의 날짜가 미래로 되어 있으면 아무리 소스 파일을 수정해도 반영되지 않음
- 이런 경우는 보통 작업 환경(PC)이 바뀌었을 때이므로, 작업 환경이 바뀌면 위 메뉴로 object 파일을 지워주고 시작해야 한다

❖ 느낀 점 및 참고사항 적기

05.DemoLed – demoled.c

```

// LED을 LOW로 설정하는 init() 함수
void LED_Init(void) {
    DDRD_DDRA0 = 1;
    DDRA_DDRA1 = 1;
    DDRA_DDRA2 = 1;
    DDRA_DDRA3 = 1;
    DDLED_DDRA0 = 0;
    DDLED_Put(0);
}

// LED를 ON으로 설정하는 SetBits() 함수
void DLDE_SetBits(BYTE led) {
    // DDRA와 DDRB를 1로 설정
    DDRA_DDRA1 = 1;
    DDRA_DDRA2 = 1;
    DDRA_DDRA3 = 1;
    DDRB_DDRA0 = 1;
    DDRB_DDRA1 = 1;
    DDRB_DDRA2 = 1;
    DDRB_DDRA3 = 1;
    // 해당 LED를 HIGH로 설정
    PORTA |= led;
    // LOW로 설정
    PORTA ^= led;
}

// LED를 OFF로 설정하는 ResetBits() 함수
void DLDE_ResetBits(BYTE led) {
    // DDRA와 DDRB를 0으로 설정
    DDRA_DDRA1 = 0;
    DDRA_DDRA2 = 0;
    DDRA_DDRA3 = 0;
    DDRB_DDRA0 = 0;
    DDRB_DDRA1 = 0;
    DDRB_DDRA2 = 0;
    DDRB_DDRA3 = 0;
    // 해당 LED를 LOW로 설정
    PORTA |= ~led;
}

// LED를 ON/OFF 상태를Toggle하는 Toggle() 함수
void DLDE_Toggle(BYTE led) {
    // DDRA와 DDRB를 1로 설정
    DDRA_DDRA1 = 1;
    DDRA_DDRA2 = 1;
    DDRA_DDRA3 = 1;
    DDRB_DDRA0 = 1;
    DDRB_DDRA1 = 1;
    DDRB_DDRA2 = 1;
    DDRB_DDRA3 = 1;
    // 해당 LED를 HIGH로 설정
    PORTA |= led;
    // LOW로 설정
    PORTA ^= led;
}

// LED를 ON/OFF 상태를Toggle하는 Toggle() 함수
void DLDE_Toggle(LED_t led) {
    // DDRA와 DDRB를 1로 설정
    DDRA_DDRA1 = 1;
    DDRA_DDRA2 = 1;
    DDRA_DDRA3 = 1;
    DDRB_DDRA0 = 1;
    DDRB_DDRA1 = 1;
    DDRB_DDRA2 = 1;
    DDRB_DDRA3 = 1;
    // 해당 LED를 HIGH로 설정
    PORTA |= led;
    // LOW로 설정
    PORTA ^= led;
}

// LED를 ON/OFF 상태를Toggle하는 Toggle() 함수
void DLDE_Toggle(LED_t led) {
    // DDRA와 DDRB를 1로 설정
    DDRA_DDRA1 = 1;
    DDRA_DDRA2 = 1;
    DDRA_DDRA3 = 1;
    DDRB_DDRA0 = 1;
    DDRB_DDRA1 = 1;
    DDRB_DDRA2 = 1;
    DDRB_DDRA3 = 1;
    // 해당 LED를 HIGH로 설정
    PORTA |= led;
    // LOW로 설정
    PORTA ^= led;
}
}

```

• demo | ed . c

06.DemoSwitch – 개요 및 회로도

❖ 학습 목표

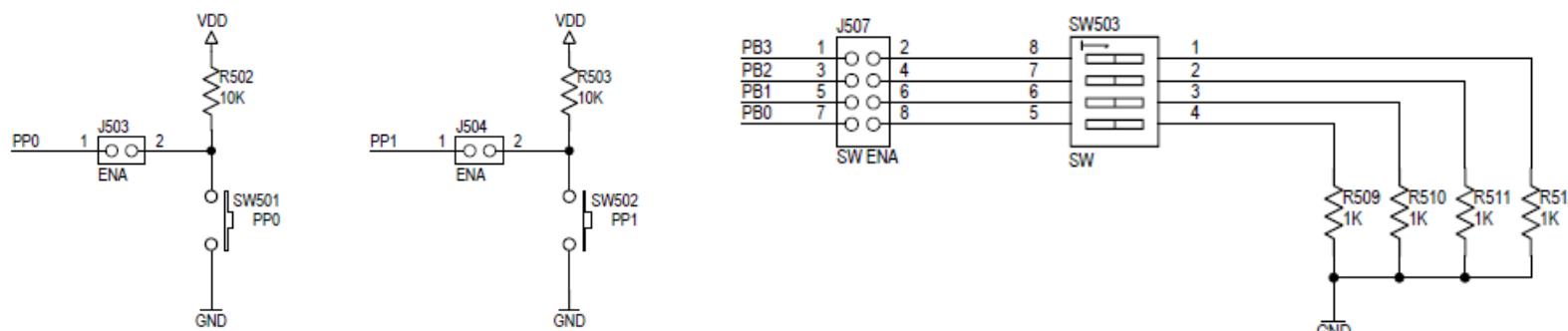
- DEMO 보드의 스위치 입력을 받을 수 있는 필수 함수를 만들 수 있다.

❖ 문제 정의

- DEMO 보드의 Tact Switch와 Dip Switch를 활용할 수 있는 함수 만들기

❖ 회로도

- Tact Switch에는 pull-up 저항이 존재
- DIP Switch에는 pull-up 저항이 없음 → 디지털 입력 포트의 pull-up을 연결해 주어야 함
- DIP Switch에 1k의 저항이 있는 이유는?



회로 구조: 터치 스위치 PP0를
눌렀을 때 DIP 스위치를 켜면
터치 스위치 PP1을 눌렀을 때
터치 스위치 PP0를 켜게 됩니다!

06.DemoSwitch – 구상 및 실습

❖ 구상

- 전역변수
 - ◆ Switch와 관련된 레지스터
- 스위치 ON일 때 레지스터 비트 값은 0
 - ◆ DSW_EN_REVERSE가 1이면
Switch ON 상태가 1로 리턴 되도록...
- Switch를 구동할 수 있는 초기화 함수
 - ◆ void DSW_Init(void);
- Tact Switch 상태를 읽는 함수 필요
 - ◆ Byte DSW_GetTactBits(Byte sw);
 - ◆ Byte DSW_GetTact(void);
- DIP Switch 상태를 읽는 함수 필요
 - ◆ Byte DSW_GetDipBits(Byte sw);
 - ◆ Byte DSW_GetDip(void);
- 다른 기능의 함수가 더 필요할까?

• demoswitch.h 중에서

```
//- MACRO -----
#define DSW_EN_REVERSE 1 // 0이면 그대로, 1이면 반전

#define DSW_TACT0_BV 0x1
#define DSW_TACT1_BV 0x2
#define DSW_TACT_MASK 0x3

#define DSW_DIP0_BV 0x1
#define DSW_DIP1_BV 0x2
#define DSW_DIP2_BV 0x4
#define DSW_DIP3_BV 0x8
#define DSW_DIP_MASK 0xf

//- FUNCTION -----
void DSW_Init(void);
Byte DSW_GetTactBits(Byte sw);
Byte DSW_GetTact(void);
Byte DSW_GetDipBits(Byte sw);
Byte DSW_GetDip(void);
```

06.DemoSwitch – main.c와 구현 실습

❖ 구현 실습

- 제공되는 06.0.DemoSwitch 프로젝트에서 demoswitch.c를 완성
- 완성 후 06.DemoSwitch와 비교해 볼 것!
- Switch 함수 활용 예제를 main 함수에 만들어 볼 것!

• main.c 중에서

```
void main(void) {
    InitPeripherals();
    for(;;) {
        // PP1 SW를 누르고 있으면 DIP SW 상태가 LED에 반영
        // PP1 SW를 안 누르고 있으면 PPO SW 상태가 PA0 LED에 반영
        if (DSW_GetTactBits(DSW_TACT1_BV)) {
            DLED_Put(DSW_GetDip());
        } else {
            if (DSW_GetTactBits(DSW_TACT0_BV))
                DLED_SetBits(DLED_LED0_BV);
            else
                DLED_ResetBits(DLED_LED0_BV);
        }
        delay_ms(10);
    }
}
```

❖ 느낀 점 및 참고사항 적기

06.DemoSwitch – demoswitch.c

```

// tact SWI[치]h[을] SWI[치]h[을] B1 B2 SWI[치]h[을]
// DSW[EN]_REVERSE GetTarget(void) {
    // B0 B1 B2 PPO SW, B1 B2 PPI SWI[치]h[을]
    // tact SWI[치]h[을] SWI[치]h[을] B1 B2 SWI[치]h[을]
    // DSW[EN]_REVERSE GetTarget(void) {
        //if DSW[EN]_REVERSE
        return (PTP & DSW[TACTMASK])~DSW[TACTMASK];
        //else
        return PTP & DSW[TACTMASK];
    }
    //if DSW[EN]_REVERSE
    return (PTP & DSW[TACTMASK])~DSW[TACTMASK];
    //else
    return PTP & DSW[TACTMASK];
}

// SWI[치]h[을] DIP SWI[치]h[을] SWI[치]h[을] DSW[GetTactBits] 을 전
// Byte DSW[EN]_REVERSE GetDipBits(Byte sws) {
    //if DSW[EN]_REVERSE
    return (PORTB & sws)~sws;
    //else
    return PORTB & sws;
}

// DIP SWI[치]h[을] SWI[치]h[을] B1 B2 SWI[치]h[을], DSW[GetTactBits] 을 전
// Byte DSW[EN]_REVERSE GetDip(void) {
    //if DSW[EN]_REVERSE
    return (PORTB & DSW[DIPMASK])~DSW[DIPMASK];
    //else
    return PORTB & DSW[DIPMASK];
}

// DIP SWI[치]h[을] SWI[치]h[을] B1 B2 SWI[치]h[을], DSW[GetTactBits] 을 전
// Byte DSW[EN]_REVERSE GetDip(void) {
    //if DSW[EN]_REVERSE
    return (PORTB & DSW[DIPMASK])~DSW[DIPMASK];
    //else
    return PORTB & DSW[DIPMASK];
}

```

• demoswiftch.ch

07.LedSwPrj0 – 실습 과제

❖ 문제

- PPO 스위치가 눌리면 아래 조건으로 동작, 그렇지 않으면 LED 전부 끄
- PA0 LED는 0.1초 간격으로 점멸 (0.1초 켜고, 0.1초 끄는 것을 반복)
- PA1 LED는 0.2초 간격으로 점멸
- PA2 LED는 0.3초 간격으로 점멸
- PA3 LED는 0.5초 간격으로 점멸

```
void main(void) {
    int n=0;
    initPeripherals();
    for(;;) {
        if ((DSW_GetTactBits(DSW_TACT0_BV)) {
            if ((n%10)==0)
                LED_ToggleBits(LED_LED0_BV);
            if ((n%20)==0)
                LED_ToggleBits(LED_LED1_BV);
            if ((n%30)==0)
                LED_ToggleBits(LED_LED2_BV);
            if ((n%50)==0)
                LED_ToggleBits(LED_LED3_BV);
        }
        if (++n == 300)
            delay_ms(10);
    }
}
```

08.LedSwPrj1 – 실습 과제

❖ 문제

- PPO 스위치를 누른 시점(Off→On)부터 PA0 LED를 2초 동안만 켜 후 끔
- PP1 스위치를 누른 시점(Off→On)부터 PA1 LED를 2초 동안만 켜 후 끔
- 위 두 동작이 각각 이루어져야 하며, 버튼을 누르고 있어도 LED는 2초만 켜져야 함
- LED가 켜 있는 2초 내에 다시 버튼을 누르면 LED는 누른 시점부터 다시 2초 동안만 켜져야 함

```
// 2초가 지나면?
if (++cnt0 >= 200) {
    cnt0 = 200;
    DLED_ResetBits(DLED_LED0_BV);
}

// 2초가 지나면?
if (++cnt1 >= 200) {
    cnt1 = 200;
    DLED_ResetBits(DLED_LED1_BV);
}

// 헬퍼 함수
void initPihera() {
    DLED_SetBits(DLED_LED0_BV);
    DLED_SetBits(DLED_LED1_BV);
}

// 헬퍼 함수
void main(void) {
    initPihera();
    int sts0_old=0, sts1_old=0, sts0_new, sts1_new;
    while(1) {
        if (sts0_new && (sts0_old==0)) {
            DLED_SetBits(DLED_LED0_BV);
            cnt0 = 0;
        }
        if (sts1_new && (sts1_old==0)) {
            DLED_SetBits(DLED_LED1_BV);
            cnt1 = 0;
        }
        if ((sts0_new && (sts1_new==0)) ||
            (sts1_new && (sts0_new==0))) {
            delay_ms(10);
        }
        sts0_old = sts0_new;
        sts1_old = sts1_new;
    }
}
```

```
void main(void) {
    initPihera();
    int cnt0=0, cnt1=0;
    int sts0_old=0, sts1_old=0, sts0_new, sts1_new;
    while(1) {
        if ((sts0_new && (sts1_new==0)) ||
            (sts1_new && (sts0_new==0))) {
            DLED_SetBits(DLED_LED0_BV);
            cnt0 = 0;
        }
        if ((sts1_new && (sts0_new==0)) ||
            (sts0_new && (sts1_new==0))) {
            DLED_SetBits(DLED_LED1_BV);
            cnt1 = 0;
        }
        if ((sts0_new && (sts1_new==0)) ||
            (sts1_new && (sts0_new==0))) {
            delay_ms(10);
        }
        sts0_old = sts0_new;
        sts1_old = sts1_new;
    }
}
```

09.LedSwPrj2 – 실습 과제

❖ 문제

- PA3 LED를 0.25초 간격으로 토글
- PA0~PA2 LED에 PPO 스위치가 눌린 회수의 8로 나눈 나머지를 표시

```
void main(void) {
    int bsts_old=0, bstsn_new, bcntt=0;
    init();
    for(;;) {
        bstsn_new = DSW_GetTactBits(DSW_TACT0_BV);
        // 힐을 누르면?
        if ((bstsn_new && (bstsn_old==0)) {
            // 힐을 누른 후에?
            if (++bcntt == 8)
                DLED_SetResetBits((Byte)bcntt, DLED_LED2_BV|DLED_LED1_BV|DLED_LED0_BV);
            bcntt = 0;
        }
        if (++bcntt == 25) {
            // 0.25초 카운트
            if (bstsn_new && (bstsn_old==0)) {
                DLED_ToggleBits(DLED_LED3_BV);
                bcntt = 0;
            }
        }
        bstsn_old = bstsn_new;
        delay_ms(10);
    }
}
```

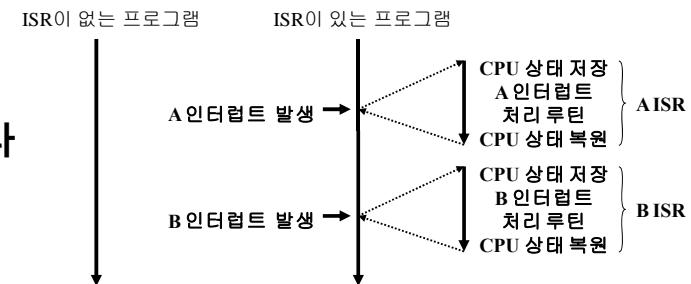
Interrupt에 대해서 (1)

❖ Interrupt란?

- 하나의 CPU로 Multi-taking을 구현할 수 있는 수단
- Interrupt를 잘 활용해야 여러 작업을 동시에 할 수 있다

❖ Interrupt 일반

- Interrupt Event 발생(Interrupt Request(IRQ)) → Interrupt Flag Set → Interrupt Vector Table이 가리키는 ISR(Interrupt Service Routine) 실행 → Interrupt Flag Clear
- Interrupt Event는 Interrupt Source에 의해 발생 (일반적으로 Hardware에 의해...)
 - ◆ 예) Serial Communication을 통해 1바이트를 전송 받음
- 그러면 해당 Interrupt Flag가 Set되고, 해당 Interrupt가 Enable 되어 있으면 Interrupt Vector Table이 가리키는 함수인 ISR이 즉각 실행됨
 - ◆ 물론, 해당 Interrupt도 Enable되어 있어야 하고, Global Interrupt도 Enable 되어 있어야 한다!
 - ◆ 하드웨어적으로 PC(Program Counter)를 Stack에 보관하고 ISR Call
 - ◆ 따라서 ISR에서는 당연히 모든 Register를 Call 전 상태로 되돌리고 Return해야 함
 - C/C++ 컴파일러가 알아서 함
- 어떤 MCU는 Global Interrupt에 대해 Enable/Disable이 있음 (대개 Flag Register에)
- Interrupt Enable/Disable의 의미
 - ◆ MCU에 따라, 말 그대로 Enable/Disable인 경우가 있고, Disable이 보류의 의미인 경우도 있음
 - ◆ 여기서 보류란 Pending된 Interrupt (= Flag가 Set된 Interrupt)
- Global Interrupt Disable은 보류의 의미, 즉 Pending된 상태로 놓아 두는 것
- MCU에 따라 ISR 실행이 끝나면 Flag가 자동으로 Clear 되는 경우도 있고, 사용자가 직접 Clear 해야 하는 경우도 있음



Interrupt에 대해서 (2)

❖ Interrupt 우선 순위(Priority) 일반

- ISR 실행 도중 우선 순위가 높은 Interrupt가 발생하면 해당 ISR이 실행됨
- ISR 실행 도중 우선 순위가 같거나 낮은 Interrupt가 발생하면 그 Interrupt는 Pending
- 우선 순위가 같은 Interrupt 여러 개가 Pending되어 있으면 Interrupt Vector Number 순으로 Interrupt의 ISR이 먼저 실행됨

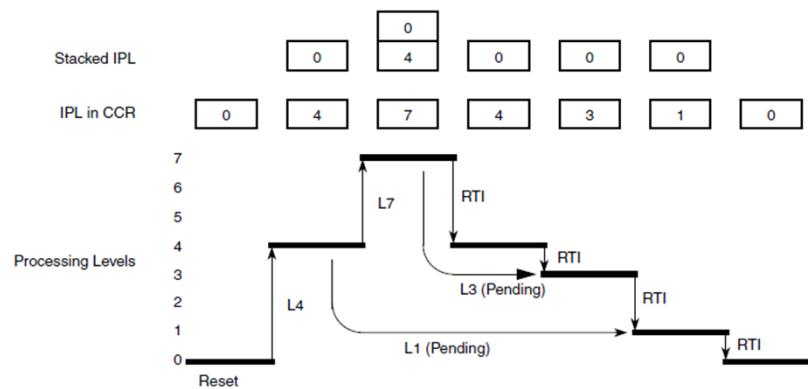


Figure 6-14. Interrupt Processing Example

❖ Interrupt를 사용함에 있어서의 문제점

- 하나의 자원(변수 또는 레지스터)을 ISR과 일반함수에서 Access함으로써 발생되는 문제
 - ◆ 즉 두 함수가 같은 자원을 공유할 때 발생하는 문제
- 참고: XGATE와 S12_CPU를 동시에 사용할 때는 이 둘 간의 문제는 H/W Semaphore로 해결
 - ◆ 이 교육에서는 논외
- S12_CPU만 사용할 때는 Critical Section을 다음 방법으로 해결
 - ◆ 앞 예제들에 포함되었던 project.h의 EnterCriticalSection(), LeaveCriticalSection() 이용

❖ ISR 실행은 매우 짧게

- Pending 된 Interrupt의 ISR이 수행완료 전에 같은 Interrupt가 또 발생하면?

무단 전재, 복제, 배포를 금합니다
의뢰 Interrupt Flag Set
의뢰 Interrupt Flag Clear
의뢰 Interrupt Flag Pending
의뢰 Interrupt Acknowledge

Interrupt에 대해서 (3)

❖ 16 bit MCU에서 ISR을 잘못 사용해서 발생하는 문제

- 16 bit를 초과하는 전역 변수는 ISR이나 일반 함수에서나 잘못된 값으로 인식될 수 있다.
- 16 bit 데이터는 ISR에서 변경한 값이 무시될 수 있다. (어떤 경우에?)

32비트 변수 data에
0xffffffff이 저장되어 있고, Little Endian 가정!

금은 ISR에서 다른 변수의 값을
0x0000과 0x2000이 순차적으로 저장된다.
ISR에서 다른 변수의 값을
0xffff0000과 0x1ffff0000이
금은 ISR에서 다른 변수의 값을
0xffff0000과 0x1ffff0000이

16비트 MCU를 사용할 때,
일반 함수에서 32비트 변수를 1 증가시키는 것은
메모리의 16비트 단위 데이터를 두 번에 걸쳐 CPU 레지스터로 읽어
1을 증가시킨 후 다시 두 번에 걸쳐 메모리에 저장하는 것!

일반함수에서 data에 1을 증가시키면, 0x20000000이 되고,
0x0000과 0x2000이 순차적으로 저장된다.
(Big Endian이면 0x2000, 0x0000 순, Little Endian이면 반대 순서)

0x0000 저장 직후, 메모리의 값은 0x1fff0000이며,
이 때 ISR이 data 값을 취하면 0x1ffff0000도 0x20000000도 아닌
섞인 값 0x1fff0000이 됨!

따라서 ISR과 일반 함수에서 같이 사용하는 전역 변수 중에서
위와 같은 현상이 발생할 수 있는 경우에는 일반 함수에서
CriticalSection 처리를 해야 한다. → 다음 페이지

Interrupt에 대해서 (4)

❖ EnterCriticalSection()와 LeaveCriticalSection()

- 이 방법은 Process Expert에서 사용하는 방법과 동일하며, 대부분의 MCU에서 이런 식으로 Critical Section을 사용!
 - ◆ Critical Section 사이에는 Interrupt가 발생해도 Pending된 상태가 됨!
 - ◆ CCR 레지스터 임시 저장에 전역변수를 사용
- project.c의 내용
 - ◆ volatile Byte CCR_reg;
- project.h의 내용
 - ◆ extern volatile Byte CCR_reg;
 - ◆ #define EnterCriticalSection() {__asm(pshc); __asm(sei); __asm(movb 1,SP+,CCR_reg); }
 - ◆ #define LeaveCriticalSection() {__asm(movb CCR_reg, 1,-SP); __asm(pulc); }

- Critical Section의 첫 부분에 EnterCriticalSection();
- Critical Section의 끝 부분에 LeaveCriticalSection();

```
void Func(void)
{
    EnterCriticalSection();

    ... // 이 사이의 Interrupt는 Pending 됨

    LeaveCriticalSection();
}
```

❖ 왜 EnableInterrupts, DisableInterrupts를 사용해서는 안 되는가?

- 많은 초심자들이 EnableInterrupts(__asm CLI), DisableInterrupts(__asm SEI) 이용해서 Critical Section 처리를 하는데 왜 문제가 될까?
 - ◆ 참고: 어떤 MCU는 SEI가 Enable Interrupts임

- 인터럽트 사용에 대해서는 앞으로 예제를 통해 계속 학습합니다.
- 예제를 통해 꼭 마스터 하기 바랍니다.

Interrupt에 대해서 (5)

❖ MC9S12XEP100의 Interrupt

- Reference Manual Chapter 6 Interrupt (S12XINTV2) 참조!
- Priority Level 1~7
- Interrupt Vectors의 위치는 Reference Manual 1.6 참조!

Table 1-13. Reset Sources and Vector Locations

Vector Address	Reset Source	CCR Mask	Local Enable
\$FFFE	Power-On Reset (POR)	None	None
\$FFFE	Low Voltage Reset (LVR)	None	None
\$FFFE	External pin RESET	None	None
\$FFFE	Illegal Address Reset	None	None
\$FFFC	Clock monitor reset	None	PLLCTL (CME, SCME)
\$FFFA	COP watchdog reset	None	COP rate select

Table 1-14. Interrupt Vector Locations (Sheet 1 of 4)

Vector Address ⁽¹⁾	XGATE Channel ID ⁽²⁾	Interrupt Source	CCR Mask	Local Enable	STOP Wake up	WAIT Wake up
Vector base + \$F8	—	Unimplemented instruction trap	None	None	—	—
Vector base+ \$F6	—	SWI	None	None	—	—
Vector base+ \$F4	—	XIRQ	X Bit	None	Yes	Yes
Vector base+ \$F2	—	IRQ	1 bit	IRQCR (IRQEN)	Yes	Yes
Vector base+ \$F0	\$78	Real time interrupt	1 bit	CRGINT (RTIE)	Refer to CRG interrupt section	
Vector base+ \$EE	\$77	Enhanced capture timer channel 0	1 bit	TIE (C0I)	No	Yes
Vector base + \$EC	\$76	Enhanced capture timer channel 1	1 bit	TIE (C1I)	No	Yes
Vector base+ \$EA	\$75	Enhanced capture timer channel 2	1 bit	TIE (C2I)	No	Yes
Vector base+ \$E8	\$74	Enhanced capture timer channel 3	1 bit	TIE (C3I)	No	Yes

무단 전재, 복제, 배포를 금합니다

Event Driven 방식에 의한 프로그래밍!

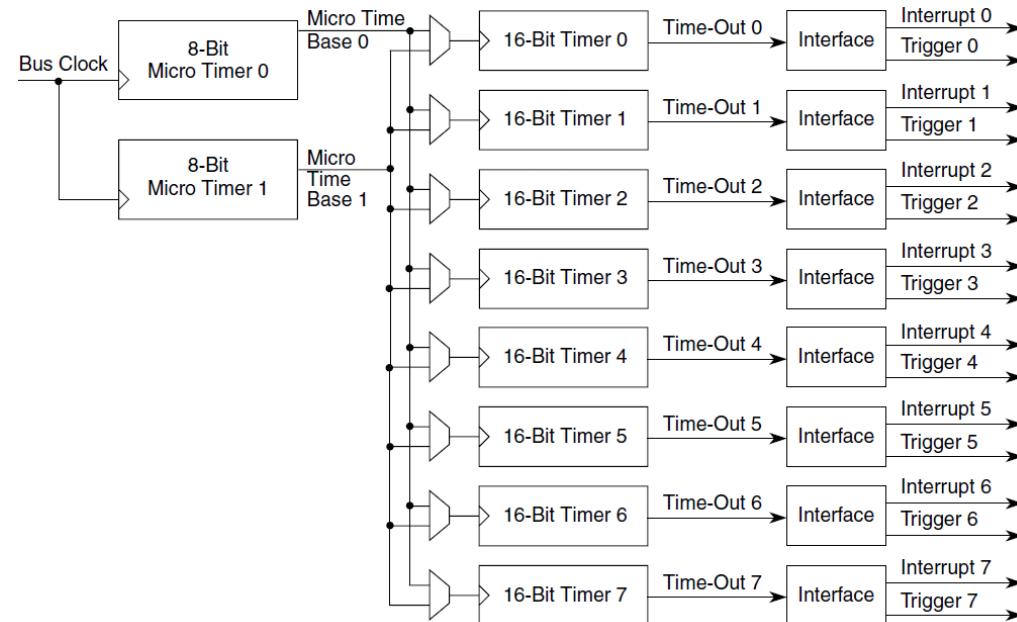
- ❖ 하나의 CPU로 멀티태스킹을 하려면 거의 반드시 Interrupt에 의한 Event Driven 방식의 프로그래밍을 해야 한다
 - Windows 프로그래밍에서 Event가 발생하면 Event에 의한 Message를 처리하는 것처럼...
 - 예) 버튼을 누르면 LED를 켠다.
 - ◆ 폴링방식: 버튼이 눌려질 때까지 기다려야 하며, 기다리는 동안 다른 작업을 하기가 힘들다.
 - ◆ 인터럽트 방식: 버튼이 눌려지면 Interrupt를 발생시켜(또는 일정시간 간격으로 Interrupt를 발생시켜 버튼 상태를 감지하여) Interrupt에 의해 LED를 켠다. 버튼이 눌려지는 상태를 기다릴 필요가 없다.
 - 과제: Event Driven Programming에 대해 학습
- ❖ CodeWarrior Processor Expert는 MCU의 각 기능을 메뉴에 의해 선택하면, Event Driven 방식의 코드를 생성해주는 기능이다.
 - 그러나 이 방식은 Event에 대응하는 함수의 실행 시간을 고려하지 않았기 때문에, Event에 대응하는 함수의 실행시간이 긴 경우 문제가 발생할 수 있음. → 이 문제는 인터럽트 우선순위와 데이터 구조를 통해 해결할 수 있긴 하지만... 어찌 되었든 MCU의 내용을 잘 알고 사용해야만 한다.
 - Processor Expert에 의해 생성한 코드를 자기 코드와 비교하는 것을 매우 권장!
- ❖ 이 교육에 활용하는 소스 코드에서는 Interrupt에 의해 Event 발생 여부만 알리는 구조를 선택
 - Event가 발생하면 바로 해당 함수가 background에서 실행되는 것이 아니라
 - main이 되는 반복 루프에서 Event 발생이 있으면 해당 처리를 foreground에서 하도록 함
 - 앞으로 진행되는 소스 코드를 보고 참고하세요!

PIT – Periodic Interrupt Timer (1)

- ❖ 일정 시간 간격으로 Interrupt를 발생시킬 수 있는 Timer
 - 활용도가 매우 높으므로 중요!

- ❖ MC9S12XEP100의 경우

- Chapter 17 참조
 - ◆ 훑어보기
- 2개의 8bit micro timer
 - ◆ 일종의 Prescaler
 - ◆ 두 개 중 선택 사용 가능
- 8개의 16bit timer
- Time-out period
 $= (\text{PITMTLD}+1) * (\text{PITLD}+1) / f_{\text{BUS}}$



PIT – Periodic Interrupt Timer (2)

❖ 관련 레지스터

- PITCFLMT (PIT Control and Force Load Micro Timer)
- PITFLT (PIT Force Load Timer)
- PITCE (PIT Channel Enable)
- PITMUX (PIT Multiplex)
- PITINTE (PIT Interrupt Enable)
- PITTF (PIT Time-Out Flag)
- PITMTLD0-1 (PIT Micro Timer Load)
- PITLD0-7 (PIT Load)
- PITCNT0-7 (PIT Counter)

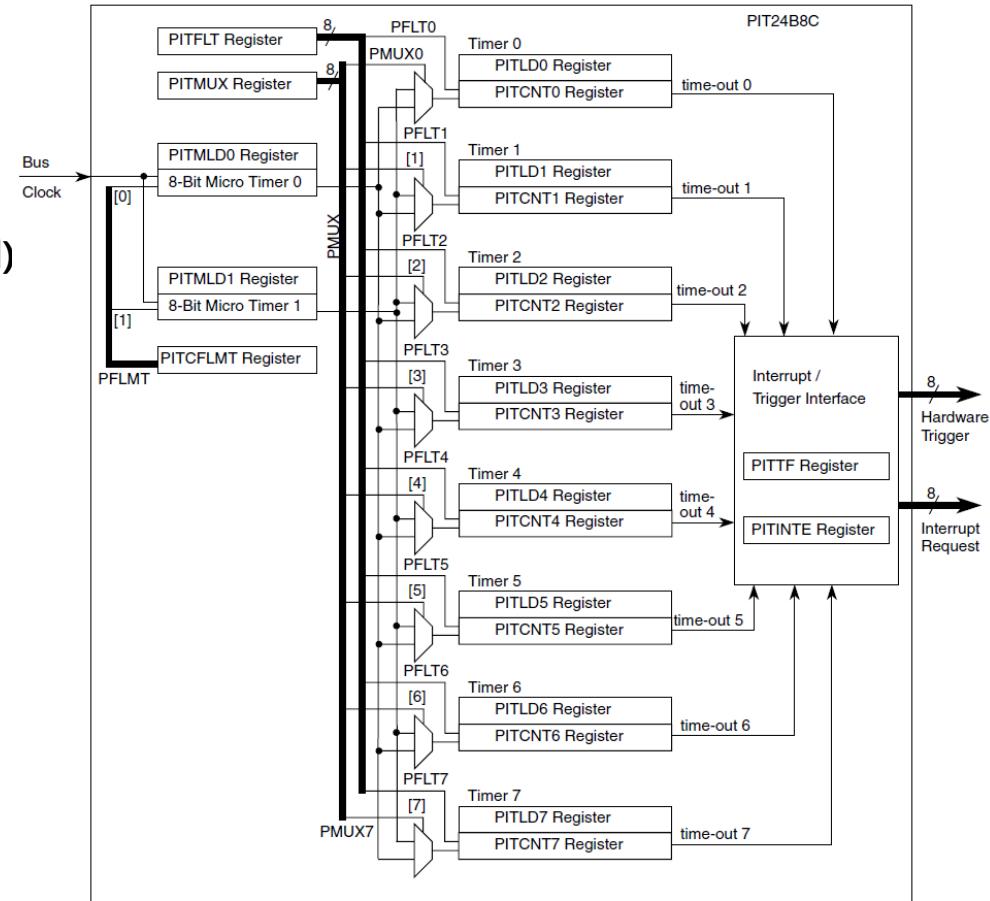


Figure 17-27. PIT24B8C Detailed Block Diagram

10.BaseTimer – 개요

❖ 학습 목표

- MCU의 Timer를 이용해서 시간 지연, 타이머, 런타임 기능을 구현할 수 있다.

❖ 문제

- millisecond 단위로
- 1. 시간 지연 기능 구현
- 2. 타이머 기능 구현(2개)
- 3. runtime 구현

❖ 어떻게

- PIT로 1 msec. 간격마다 인터럽트를 발생시켜 ISR 실행
- ISR에서 전역변수의 값을 1씩 증가시킴
- 이 변수를 runtime으로 활용하여 다음을 구현
 - ◆ 시간 지연
 - ◆ 타이머

10.BaseTimer – 구상

❖ 필요 변수

- PIT 관련 레지스터
- **btmr_msec** : millisecond 단위의 runtime 측정용 ← PIT로 증가시킴
- **btmr_tstart0**, **btmr_tstart1**: 타이머 시작시의 runtime 저장용
- **btmr_tset0**, **btmr_tset1**: 타이머 초기 설정값 저장용

❖ 함수들

- 초기화 함수
 - ◆ `void BTMR_Init(void);`
- Millisecond 단위로 시간 지연을 하는 함수
 - ◆ `void BTMR_DelayMs(Word msec);`
- 타이머 함수 – 타이머 시간을 정하고 이 타이머가 0이 되었는지 판단
 - ◆ `void BTMR_SetTimer0(LWord msec);`, `void BTMR_SetTimer1(LWord msec);`
 - ◆ `LWord BTMR_GetTimer0(void);`, `LWord BTMR_GetTimer1(void);`
- Runtime 함수 – msec. 단위의 총 경과 시간
 - ◆ `LWord BTMR_GetRuntime(void);`
 - ◆ `void BMTR_SetRuntime(LWord msec);`

10.BaseTimer – 구현 (1)

• basetimer.h

```
#ifndef _BASETIME_H_
#define _BASETIME_H_

#ifndef __cplusplus
extern "C" {
#endif

#include "project.h"

//----- MACRO -----
#define BTMR_PRIORITY PIT_PRIORITY

//----- FUNCTION -----
void BTMR_Init(void);
void BTMR_DelayMs(Word msec);
void BTMR_SetTimer0(LWord msec);
LWord BTMR_GetTimer0(void);
void BTMR_SetTimer1(LWord msec);
LWord BTMR_GetTimer1(void);
LWord BTMR_GetRuntime(void);
void BTMR_SetRuntime(LWord msec);

#ifndef __cplusplus
}
#endif
#endif
```

• basetimer.c의 전역변수 및 ISR

```
volatile LWord btmr_msec = 0; // runtime 저장 변수, 32bit이므로 2^32-1 msec까지 측정 가능
LWord btmr_tstart0, btmr_tstart1; // timer0, timer1의 시작시의 runtime 저장용
LWord btmr_tset0, btmr_tset1; // timer0, timer1의 설정 시간
```

```
// Vpit0 interrupt service routine
#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt VectorNumber_Vpit0 void BTMR_PIT0_ISR(void) {
    btmr_msec++;
    PITF_PTF0 = 1;
}

#pragma CODE_SEG DEFAULT
```

- ISR에서 사용하는 변수에는 volatile을 반드시 붙일 것 → 이유는 11.VolatileAndCLI에서 설명

- 맨 끝 PITF_PTF0 = 1

• MC9S12XEP100용 Codewarrior의 ISR 함수 구현법

```
#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt vector_number void ISR_name(void) {
}

#pragma CODE_SEG DEFAULT
```

여기서 **vector_number**는 MC9S12XEP100.h의 133번줄 이후 참조
ISR_name은 마음대로...

여기서 **#progma CODE_SEG** …는 이 함수의 메모리 위치와 관련
참고: 컴파일러 매뉴얼에서 .text 및 SEGMENT 참조

10.BaseTimer – 구현 실습

❖ 실습

- 제공되는 10.0.BaseTimer 프로젝트에서 basetimer.c를 완성
- 완성 후 10.BaseTimer 와 비교해 볼 것!

❖ 시간 관련 함수 활용 예제를 main 함수에 만들어 볼 것!

```
void main(void) {
    Word i;

    InitPeripherals();

    srand(0x1234);
    for(;;) {
        // LED0을 250msec. 간격으로 8번 토글
        for (i=1; i<=8; i++) {
            DLED_ToggleBits(DLED_LED0_BV);
            BTMR_DelayMs(250);
        }

        // LED1을 임의의 간격(50~149msec.)으로 5000 msec. 동안 토글
        BTMR_SetTimer0(5000);
        while (BTMR_GetTimer0()) {
            DLED_ToggleBits(DLED_LED1_BV);
            BTMR_DelayMs((rand()%100)+50); // 50~149
        }
        DLED_ResetBits(DLED_LED1_BV);
    }
}
```

- srand의 파라미터로 난수를 주는 방법에는 무엇이 있을까?

• Timer 값을 Seed 값으로...
• Timer 값을 초기화하는
• Timer 값을 초기화하는
• Timer 값을 초기화하는
• Timer 값을 초기화하는

❖ 느낀 점 및 참고사항 적기

10. BaseTimer – 구현 (2)

- basetimer.c의 초기화

```

        PITCHLMTPITE = 1; // Enable PIT
    PITCHLMTPITFZ = 1; // PIT counters are stalled in Freeze Mode
    PITCHLMTPINTE0 = 1; // Enable PIT Ch. 0 interrupt
    PITCHFPTF0 = 1; // Clear Flag
    PITCHPCED = 1; // Enable PIT Ch. 0
    PITCHPMUX0 = 0; // micro time base 0
    PITLD0 = (1000-1); // 1000 count
    #endif
    PITMLD1 = (40*5-1);
    PITMLD0 = (40-1);
    #elif BFRO_OPTION == BFRO_40MHz
    PITMLD1 = (20*5-1);
    PITMLD0 = (20-1);
    #elif BFRO_OPTION == BFRO_20MHz
    PITMLD1 = (2-1);
    PITMLD0 = (2-1);
    #if BFRO_OPTION == BFRO_2MHz
    PITMLD1 = (2*5-1);
    PITMLD0 = (2*5-1);
    #endif
    // 8bit Micro Timer 0을 1MHz(1usec)로 <- 65.536 msec 271
    // 8bit Micro Timer 1을 200kHz(5usec)로 <- 5*65.536 msec 271
    INT_CFDATA5 = BTMR_PRIORITY; // INT_CFDATA0~4를 Vpit0 (0x1f7A80x7)/2
    INT_CFDA0R = 0x70; // Vpit0 0x1f7A -> Vpit0x00f0 및
    PITCHLMTPITE = 0; // Disable PIT module and Clear PIT flag
    void BTMR_Init(void) {
        // 초기: 8bit Micro Timer0는 100Hz(1ms)로 설정됨
        // Periodic Interrupt Timer 0 초기화
    }
}

```

- 우선순위 설정

- Vector Table 주소를 이용해서 설정
 - Vpit0 는 0xFF7A;
 - INT_CFAADDR = Vpit & 0x00f0
 - INT_CFDATAAn = 우선순위 (0~7)
- 여기서 n은 (Vpit & 0xf)/2

- Processor Expert의 결과와 비교도 해 보았습니다!
- Reference Manual Chapter 17를 참고하여 작성

- 이름을 base timer라 한 이유는 8개의 PIT가 공유하는 2개의 micro timer를 여기서 설정하기 때문
- Micro Timer 0은 1usec 간격으로 설정
- Micro Timer 1은 5usec 간격으로 설정

10.BaseTimer – 구현 (3)

• basetimer.c의 timer0

• basetimer.c의 delay

```
• ISR에서 딜레이 타이머를 위한 Critical Section 설정

{
    btmr_start0 = BTMR_SetTimer0((Word)time0);
    btmr_start1 = BTMR_SetTimer1((Word)time1);

    while ((Word)(BTMR_GetRunTime() - to) < msec) {
        if (msec == 0)
            return;
        to = BTMR_GetRunTime();
    }
    msec = BTMR_GetDelayMs((Word)msec);
}

void BTMR_SetDelayMs(Word msec) {
    Word to;
    Word runtme;
    EnterCriticalSection();
    Loword runtme;
    LeaveCriticalSection();
    EnterCriticalSection();
    Loword runtme;
    Loword BTMR_GetRunTime(void) {
        // миллиsecond 단위로 runtme를 읽음
        // msec 단위로 msec단위로 msec를 읽음
        // msec 단위로 msec단위로 msec를 읽음
    }
    Loword BTMR_GetRunTime(void) {
        // миллиsecond 단위로 runtme를 읽음
        // msec 단위로 msec단위로 msec를 읽음
        // msec 단위로 msec단위로 msec를 읽음
    }
    Loword runtme;
    btmr_start0 = msec;
    btmr_start1 = BTMR_SetTimer0((Word)msec);
    btmr_start0 = BTMR_SetTimer1((Word)msec);
    while ((Word)(BTMR_GetRunTime() - to) < msec) {
        if (msec == 0)
            return;
        to = BTMR_GetRunTime();
    }
    msec = BTMR_GetDelayMs((Word)msec);
}
}

LeaveCriticalSection();
```

```
• basetimer.c의 delay

{
    Word etime;
    Word etime;
    if (etime < btmr_start0) {
        return (btmr_start0 - etime);
    }
    etime = BTMR_GetRunTime() - btmr_start0;
    if (etime < btmr_start1) {
        return (btmr_start1 - etime);
    }
    etime = BTMR_GetRunTime() - btmr_start1;
    return 0;
}
```

기계어 한 명령어, 1을 써야 0이 되는 I/O Reg.

- ❖ ISR은 기계어 명령어 사이에 끼어들어 실행될 수 있다
 - 즉 C 언어로 한 명령어지만 기계어로는 여러 명령어라면 ISR이 중간에 끼어들 수 있다
- ❖ 그렇다면, C 언어에서 기계어 한 명령어로 번역되는 것은?
 - X bit CPU의 경우, x bit 데이터를 x bit 데이터 버스로 CPU 레지스터와 메모리 사이로 이동시킬 때
 - ◆ 16bit CPU에서 32bit 변수를 A, B가 있을 때,
A=B; 는 기계어 두 명령어에 의해 이루어진다. 따라서 이 과정 중에 ISR이 실행될 수도...
 - I/O register의 한 비트만 0 또는 1로 변경하는 것이 기계어 한 명령어인 MCU도 존재
 - ◆ MC9S12XEP100도 이러한 종류의 MCU로
 - ◆ PORTA의 PA0만 1 또는 0으로 하는 경우는 1명령어이다.
 - ◆ PORTA |= 0x1; PORTA &= 0xfe; PORTA_PA0 = 1 or 0; 모두 기계어 한 명령어
 - ◆ 그러나 PORTA |= 0xf; 등은 기계어로 한 명령어가 아닐 수 있다.
- 번역된 어셈블리 코드를 보고 문제가 있는지 확인 필요!
 - 어떤 MCU는 I/O Register중 일부 영역만 한 명령어로 비트 대입 가능하다
 - 컴파일러 옵션에 따라 달라질 수도 있다.
- ❖ 1을 써야 0이 되는 I/O 레지스터가 있다
 - Interrupt Flag 가 1로 되어 있는 경우, 이 Flag를 다시 0으로 만들어 줄 때 1을 써야 한다.
 - 설사 한 명령어라도 한 Cycle이 아닌 Read-Modify-Write 동작 명령어는...
 - 왜 그럴까? → 과제

11.VolatileAndCLI – 오동작 예제 (1)

❖ 학습 목표

- volatile을 사용할 수 있다.
- Critical Section을 사용할 수 있다.

❖ 목적

- volatile의 사용 이유
- Critical Section 처리를 안 한 경우의 오동작을 확인
- EnterCriticalSection와 LeaveCriticalSection에 의한 Critical Section 처리

❖ 소스 수정

- 10.BaseTimer에서 basetimer.c를 basetimer_x.c로 변경한 후 다음의 내용을 추가

• basetimer_x.c의 ISR

```
// Vpit0 interrupt service routine
#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt VectorNumber_Vpit0 void BTMR_PIT0_ISR(void) {
    static int cnt = 0;

    btmr_msec++;
    if (++cnt==250) {
        cnt=0;
        PORTA ^= 0b11;
    }
    PITTF_PTF0 = 1;
}
#pragma CODE_SEG DEFAULT
```

• 250msec.마다 PA0, PA1 LED
를 토글하는 부분 추가

11. VolatileAndCLI – 오동작 예제 (2)

❖ main.c의 다음 부분은 무슨 문제가?

• main.c의 main

```
Word flag;  
  
void main(void) {  
    Word flag;  
  
    InitPeripherals();  
  
    // 이 부분에는 무슨 문제가 있을 수 있을까?  
    flag=1;  
    if (flag!=1) {  
        for (i=0; i<100; i++) {  
        }  
    }  
  
    for(;;) {  
        // 이 부분에는 무슨 문제가 있을까?  
        // basetimer_x.c의 ISR과 함께 검토!  
        // EnterCriticalSection();  
        DLED_PutAll(DLED_GetAll()^0b1100);  
        // DLED_ToggleBits(0b1100);  
  
        // LeaveCriticalSection();  
        for (i=0; i<10; i++) {  
        }  
    }  
}
```

• 어셈블리 번역 결과 (Word flag;)

```
28: // 이 부분에는 무슨 문제가 있을 수 있을까?  
29: flag=1;  
0004 c601 [1] LDAB #1  
0006 87 [1] CLRA  
0007 7c0000 [3] STD flag  
30: if (flag!=1) {  
31:     for (i=0; i<100; i++) {  
32:     }  
33: }  
34:
```

- 번역이 안 되었다!
- if 문이 당연히 거짓!

• 어셈블리 번역 결과 (volatile Word flag;)

```
28: // 이 부분에는 무슨 문제가 있을 수 있을까?  
29: flag=1;  
0004 c601 [1] LDAB #1  
0006 87 [1] CLRA  
0007 7c0000 [3] STD flag  
30: if (flag!=1) {  
000a fe0000 [3] LDX flag  
000d 04051c [3] DBEQ X,*+31 ;abs = 002c  
31:     for (i=0; i<100; i++) {  
0010 b705 [1] SEX A,X  
0012 08 [1] INX  
0013 8e0064 [2] CPX #100  
0016 25fa [3/1] BCS *-4 ;abs = 0012  
0018 2012 [3] BRA *+20 ;abs = 002c  
32: }  
33: }
```

- 번역이 되었다!
- flag가 ISR에 의해 변경되면 if 문이 참일 수도

volatile를 제거한 경우의
어셈블리 번역 결과입니다.
어셈블러 compiler가 이를
정确하게 번역하지 못했습니다.

11. VolatileAndCLI – 오동작 예제 (3)

❖ main.c의 다음 부분은 무슨 문제가?

❖ 해결 방법은?

1. Critical Section 이용
 - ◆ EnterCriticalSection()
 - ◆ LeaveCriticalSection()
2. Critical Section 처리된 DLED_Toggle 함수 이용

• main.c의 main

```
Word flag;  
  
void main(void) {  
    Word flag;  
  
    InitPeripherals();  
  
    // 이 부분에는 무슨 문제가 있을 수 있을까?  
    flag=1;  
    if (flag!=1) {  
        for (i=0; i<100; i++) {  
        }  
    }  
  
    for(;;) {  
        // 이 부분에는 무슨 문제가 있을까?  
        // basetimer_x.c의 ISR과 함께 검토!  
        // EnterCriticalSection();  
        DLED_Put(DLED_Get()^0b1100);  
        // DLED_ToggleBits(0b1100);  
        // LeaveCriticalSection();  
        for (i=0; i<10; i++) {  
        }  
    }  
}
```

- PA0, PA1 LED는 0.25초 간격으로 ISR에서 토글
- PA2, PA3 LED는 매우 빠른 속도로 토글되어야 정상
- 그러나 PA0, PA1 LED가 가끔씩/자주 0.25 간격으로 토글되지 않는 문제 발생!

11.VolatileAndCLI – volatile & Critical Section

- ❖ Interrupt를 사용할 경우 volatile을 쓰지 않으면 컴파일러가 최적화에 따라 무시하는 부분이 발생할 수 있다
 - volatile은 이러한 최적화를 못하게 하고 있는 그대로 번역함
 - 문제: 예제에서 volatile을 붙여야 하는 부분은?
 - 실습으로 확인해 보자! (C 소스와 어셈블리 소스 비교)
- ➔ ISR에서 사용하는 전역변수에는 volatile을 반드시 붙인다
(이런 이유로 전역변수나 다름 없는 I/O register에는 volatile이 붙음)
- ❖ S12_CPU에서는 Critical Section을 project.h(실습으로 만든 헤더파일)에 선언된 매크로 함수를 이용해서
 - EnterCriticalSection()로 시작해서 LeaveCriticalSection()로 끝나는 부분으로 처리
 - ◆ EnterCriticalSection(); // 시작 부분
 - ◆ Critical Codes;
 - ◆ LeaveCriticalSection(); // 끝 부분

UART 통신

❖ Universal Asynchronous Receiver and Transmitter

- UART 없는 MCU가 없을 정도로 가장 기본적인 통신 장치
- 여기에 전기적 레벨을 RS-232로 변환하여 PC에서도 사용
 - ◆ RS-232 레벨을 사용하면 전송 길이가 늘어남
 - ◆ 요즘 PC에는 없는 경우도 있지만, USB to RS-232 장치를 사용하면 됨
- 선 하나로 데이터를 주고 받음, 따라서 주는 쪽 선 하나 받는 쪽 선 하나 필요
 - ◆ A장치의 TXD로 B장치의 RXD에 데이터 보냄
 - ◆ A장치의 RXD로 B장치의 TXD 데이터 받음
- 전송 데이터의 동기 신호가 없으므로 서로 동기에 필요한 정보를 사전 설정
 - ◆ Baudrate, data 길이, Start 비트, Stop 비트, Parity 사용 여부 등
- 데이터 흐름 제어를 위해 별도의 신호를 사용하기도 함
- 과제: UART 통신에 대해 조사하기

❖ PC의 RS-232 통신 프로그램

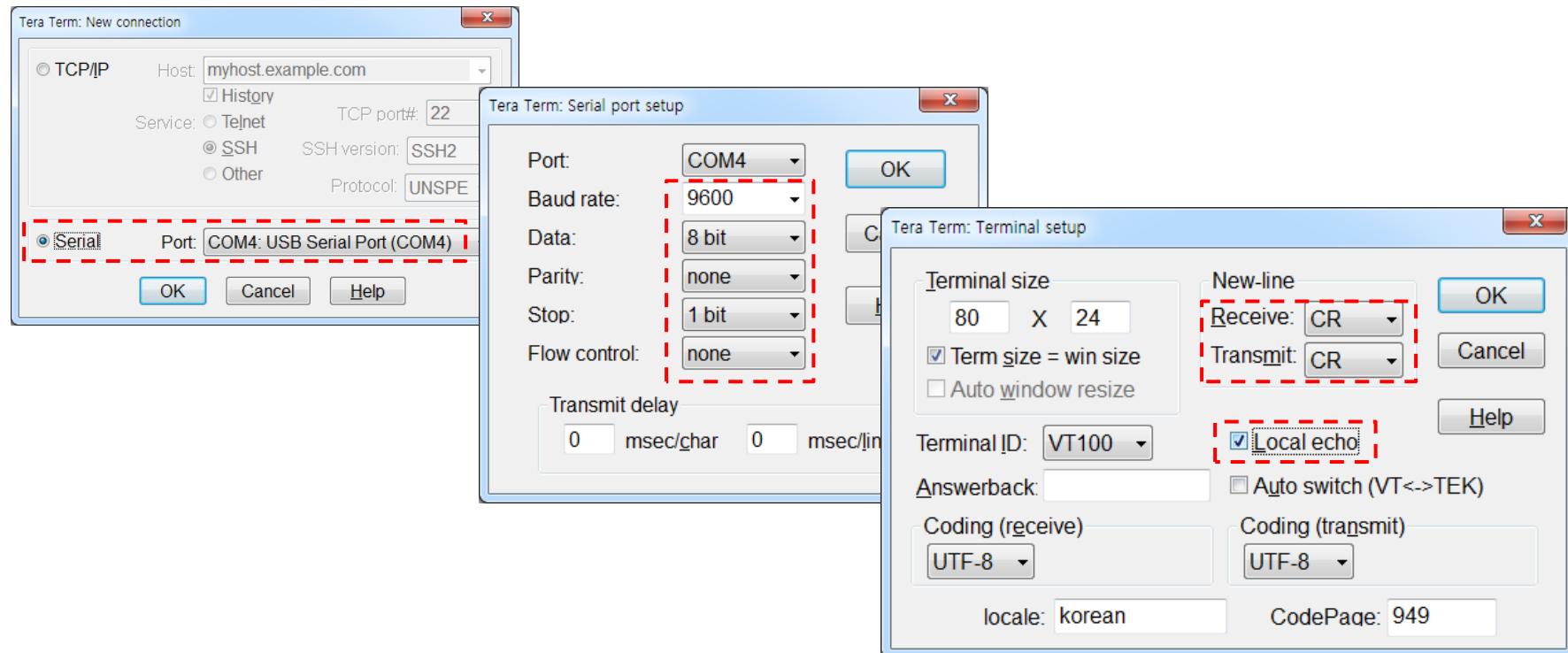
- Windows XP: 하이퍼터미널 기본 내장
- Windows 7: 기본 내장 프로그램 없으므로 Tera Term 사용 권장

❖ 실습

- Baudrate 등의 설정
- 줄바꿈 및 Echo 설정

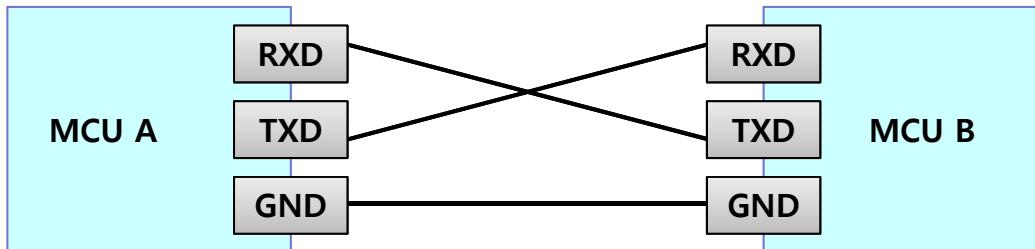
Tera Term

- ❖ 공개 소프트웨어이므로 Google에서 검색 및 다운로딩 가능
- ❖ 사용법
 - 시작 시 사용 Serial Port 선택
 - Setup 메뉴에서 Serial Port...로 설정
 - New-line이나 Local Echo가 필요한 경우, Setup 메뉴의 Terminal...로 설정

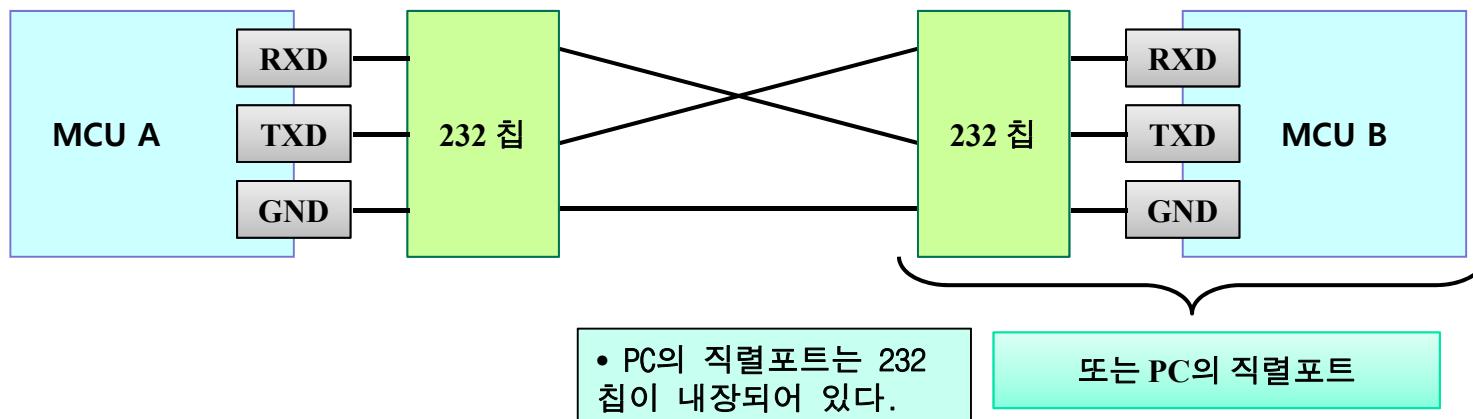


UART 통신 배선 방법

❖ 통신 거리가 짧을 경우의 배선 방법



❖ 통신 거리가 긴 경우의 배선 방법



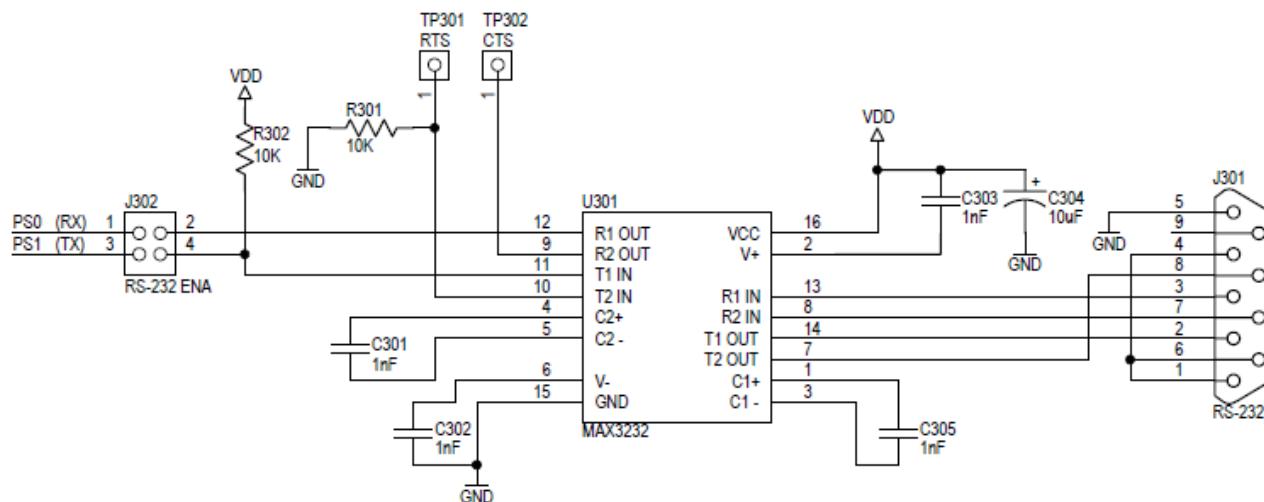
❖ UART↔ZigBee, UART↔Bluetooth, UART↔Ethernet 다 존재

- 필요에 따라 UART를 경우해서 ZigBee, Bluetooth, Ethernet 통신 사용 가능
- 이 교육에서 제공하는 UART↔Bluetooth를 이용해서 추후 과제에 편하게 사용하세요!
- 과제: 지급된 UART↔Bluetooth를 이용해서 PC와 연결하는 방법 그려보기

DEMO9S12XEP100의 RS-232 포트

❖ 특징

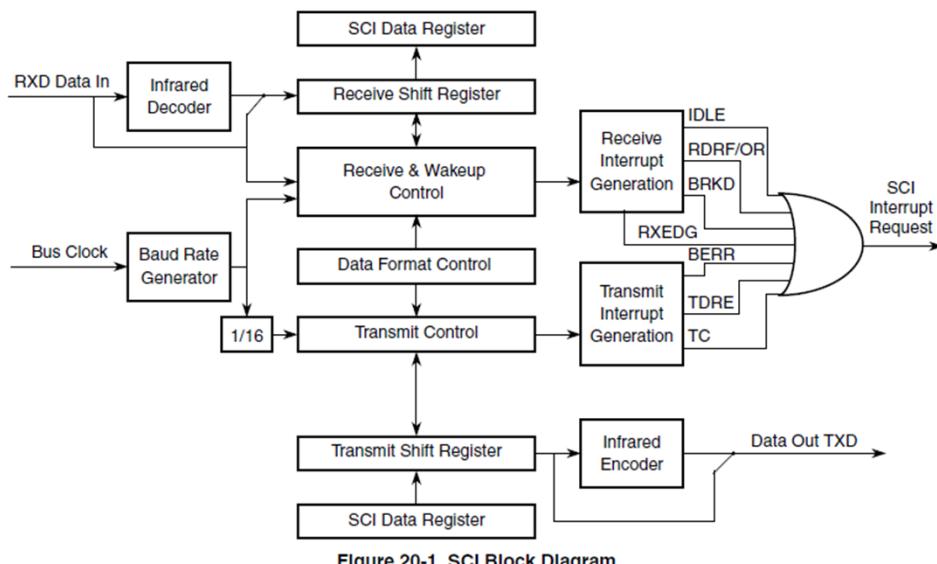
- MAX3232를 이용해서 MCU Digital Level을 RS-232 전압 Level로 변환해서 J301에 연결
 - J301은 Female DSUB 9P 커넥터
 - ◆ PC의 Serial Port는 Male 9P DSUB 커넥터
 - RX가 J301의 3번 핀에, TX가 J301의 2번 핀에 해당
 - ◆ PC의 Serial Port는 RX가 2번 핀, TX가 3번 핀, GND가 5번 핀
- 시리얼 연장 케이블로 PC와 연결해야 함!



MC9S12XEP100의 SCI

❖ SCI (Serial Communication Interface)

- 관련 자료: Reference Manual Chapter 20. Serial Communication Interface (S12SCIV5)
- SCI0에서 SCI7까지 총 8개나 존재
 - ◆ RXD_n, TXD_n ($n = 0..7$)
 - ◆ Reference Manual의 레지스터 이름은 SCIXYZ 형태지만, MC9S12XEP100.h에 SCInXYZ 형태
- TX, RX 인터럽트 모두 지원
 - ◆ 같은 interrupt vector 사용



Register Name	Bit 7	6	5	4	3	2	1	Bit 0	
0x0000 SCIBDH ¹	R	IREN	TNP1	TNP0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0001 SCIBDL ¹	R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0002 SCICR1 ¹	R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0000 SCIASR1 ²	R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIFF
0x0001 SCIACR1 ²	R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIIE
0x0002 SCIACR2 ²	R	0	0	0	0	0	BERRM1	BERRM0	BKDFE
0x0003 SCICR2	R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0004 SCISR1	R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0005 SCISR2	R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
0x0006 SCIDRH	R	R8	T8	0	0	0	0	0	0
0x0007 SCIDRL	R	R7	R6	R5	R4	R3	R2	R1	R0
	W	T7	T6	T5	T4	T3	T2	T1	T0

1.These registers are accessible if the AMAP bit in the SCISR2 register is set to zero.

2.These registers are accessible if the AMAP bit in the SCISR2 register is set to one.

= Unimplemented or Reserved

Figure 20-2. SCI Register Summary

20.SerialDebug – 목적, 문제 정의

❖ 학습 목표

- serialdebug.h 와 serialdebug.c 파일을 이용해 UART 통신으로 데이터를 주고 받을 수 있다.

❖ 목적

- MCU와 PC를 UART 통신으로 연결
- PC에서 MCU의 특정 함수를 실행시키거나
- MCU의 상태를 PC로 전송할 수 있도록 함
→ 즉, MCU에 키보드와 모니터를 부착하는 효과

❖ 문제

- 통신프로그램에서 'a'~'z'로 시작하는 문자열을 입력하고 엔터를 치면 'a'~'z'에 해당하는 MCU의 함수가 실행되도록 함. 이 때 첫 문자 이후 입력한 문자열이 해당 함수의 인자로 사용되도록 함
 - ◆ 예) "a 3 456 bcd"를 입력하면, 'a'에 해당하는 함수를 "3 456 bcd" 문자열을 인자로 해서 실행
 - ◆ Backspace 문자도 사용 가능하게 함
- 통신프로그램에서 'A'~'Z'를 입력하면 바로 'A'~'Z'에 해당하는 MCU의 함수가 실행되도록 함
- 참고: F/W에서 'a'~'z'는 **normal command**, 'A'~'Z'는 **hot command**로 정의함.

20.SerialDebug – 구상 (1)

❖ 필요 변수

- SCI 관련 레지스터
- volatile void (*sdbg_pNCmdCallback[SDBG_NCMD_NUM])(char *); // normal command용 함수 포인터 배열
- volatile void (*sdbg_pHCmdCallback[SDBG_HCMD_NUM])(void); // hot command용 함수 포인터 배열
- volatile sByte sdbg_RxFlag; // command가 전송되었는지의 여부 저장, 비트 별로 NCMD, HCMD
- volatile char sdbg_RxHCmd; // Hot command 저장 변수
- volatile char sdbg_RxNCmd[SDBG_NCMD_BUF_SIZE]; // Normal Command로 시작되는 문자열 저장 배열
- volatile sWord sdbg_RxNCmdPtr; // RxNCmd 배열의 위치
- volatile sWord sdbg_TxQueHeadPtr, sdbg_TxQueTailPtr; // 전송할 문자 큐 배열을 위한 포인터
- volatile char sdbg_TxQue[SDBG_TX_QUE_SIZE]; // 전송할 문자 큐 배열

❖ 과제

- Queue 구조에 대한 학습

20.SerialDebug – 구상 (2)

❖ 함수들

■ 인터럽트 함수

- ◆ interrupt SDBG_VectorNumber_Vsci void SDBG_SCI_ISR(void); // ISR

■ 초기화 및 커맨드 등록 함수

- ◆ void SDBG_Init(void); // 초기화 함수
- ◆ void SDBG_RegisterNCmd(char ncmd, void (*pCallback)(char *)); // normal command에 대응하는 함수 등록
- ◆ void SDGB_RegisterHCmd(char hcmd, void (*pCallback)(void)); // hot command에 대한 함수 등록

■ MCU → PC 전송

- ◆ void SDBG_PutCh(char ch); // 한 문자 전송
- ◆ void SDBG_PutStr(char* pstr); // 문자열 전송
- ◆ void SDBG_Printf(const char *format, ...); // printf처럼 사용할 수 있는 함수

■ PC → MCU 커맨드 전송 여부 및 전송된 커맨드 실행

- ◆ Bool SDBG_IsEvents(void); // hot command나 normal command가 전송되었는지 여부 판단
- ◆ void SDBG_ExecuteCallback(void); // IsEvents()함수가 참이면 전송된 command에 해당하는 callback 함수 실행

20.SerialDebug – 구현 (1)

• serialdebug.h

```
-- MACRO -----
// UART 통신 속도
#define SDBG_BAUDRATE 9600UL

// 사용하는 SCI Register
#define SDBG_SCIASR1    SC10ASR1
#define SDBG_SCIACR1    SC10ACR1
#define SDBG_SCIACR2    SC10ACR2
#define SDBG_SCIBD_IREN  SC10BD_IREN
#define SDBG_SCIBD_SBR   SC10BD_SBR
#define SDBG_SCICR1      SC10CR1
#define SDBG_SCICR2      SC10CR2
#define SDBG_SCICR2_TIE  SC10CR2_TIE
#define SDBG_SCISR1      SC10SR1
#define SDBG_SCISR1_TDRE SC10SR1_TDRE
#define SDBG_SCISR1_RDRF SC10SR1_RDRF
#define SDBG_SCISR2      SC10SR2
#define SDBG_SCISR2_AMAP SC10SR2_AMAP
#define SDBG_SCIDRL      SC10DRL

#define SDBG_VectorNumber_Vsci VectorNumber_Vsci0

#define SDBG_ASCII_CR 'Wr' // 13
#define SDBG_ASCII_LF 'Wn' // 10
#define SDBG_ASCII_BS 'Wb' // 8

// 한 문자 명령어
#define SDBG_HCMD_START 'A'
#define SDBG_HCMD_END   'Z'
#define SDBG_HCMD_NUM   (SDBG_HCMD_END-SDBG_HCMD_START+1)

// Wn으로 끝나는 한줄 명령어
#define SDBG_NCMD_START 'a'
#define SDBG_NCMD_END   'z'
#define SDBG_NCMD_NUM   (SDBG_NCMD_END-SDBG_NCMD_START+1)

#define SDBG_NCMD_BUF_SIZE 100
#define SDBG_TX_QUE_SIZE   100
```

• Baudrate 변경은 여기서

• serialdebug.c 전역 변수

```
volatile void (*sdbg_pNCmdCallback[SDBG_NCMD_NUM])(char *) ;
volatile void (*sdbg_pHCmdCallback[SDBG_HCMD_NUM])(void) ;

#define SDBG_RX_NCMD 0x1
#define SDBG_RX_HCMD 0x2

volatile sByte sdbg_RxFlag;
volatile char sdbg_RxHCmd;
volatile char sdbg_RxNCmd[SDBG_NCMD_BUF_SIZE];
volatile sWord sdbg_RxNCmdPtr;
volatile sWord sdbg_TxQueHeadPtr, sdbg_TxQueTailPtr;
volatile char sdbg_TxQue[SDBG_TX_QUE_SIZE];
```

20.SerialDebug – 구현 (2)

• serialdebug.c ISR

```
// Vsci Interrupt Service Routine
#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt SDBG_VectorNumber_Vsci void SDBG_SCI_ISR(void) {
    volatile Byte status, data;

    status = SDBG_SCISR1;

    // Receive
    if (status & SC10SR1_RDRF_MASK) {
        data = SDBG_SCIDRL;
        // HCmd: 한 문자 명령어
        if ((data>=SDBG_HCMD_START) && (data<=SDBG_HCMD_END)) {
            if (!(sdbg_RxFlag & SDBG_RX_HCMD)) {
                sdbg_RxHCmd = data;
                sdbg_RxFlag |= SDBG_RX_HCMD;
            }
        } else { // NCMD: 한 줄 명령어
            if (!(sdbg_RxFlag & SDBG_RX_NCMD)) {
                if (data==SDBG_ASCII_CR) {
                    sdbg_RxNCmd[sdbg_RxNCmdPtr] = 'W0';
                    sdbg_RxFlag |= SDBG_RX_NCMD;
                    sdbg_RxNCmdPtr++;
                } else if (data==SDBG_ASCII_BS) {
                    if (sdbg_RxNCmdPtr>0)
                        sdbg_RxNCmdPtr--;
                } else {
                    sdbg_RxNCmd[sdbg_RxNCmdPtr] = data;
                    if (sdbg_RxNCmdPtr < (SDBG_NCMD_BUF_SIZE-1)) {
                        sdbg_RxNCmdPtr++;
                    }
                }
            }
        }
        // Transmit
        if (status & SC10SR1_TDRE_MASK) {
            if (sdbg_TxQueHeadPtr!=sdbg_TxQueTailIPtr) {
                SDBG_SCIDRL = sdbg_TxQue[sdbg_TxQueTailIPtr];
                sdbg_TxQueTailIPtr = (sdbg_TxQueTailIPtr+1)%SDBG_TX_QUE_SIZE;
            } else {
                SDBG_SCICR2_TIE = 0;
            }
        }
    }
}#pragma CODE_SEG DEFAULT
```

• serialdebug.c 초기화

```
void SDBG_Init(void) {
    sWord i;

    SDBG_SC1SR2 = 0;
    SDBG_SC1SR2_AMAP = 1;
    SDBG_SC1ASR1 = 0x3;
    SDBG_SC1ACR1 = 0;
    SDBG_SC1ACR2 = 0;
    SDBG_SC1SR2_AMAP = 0;
    SDBG_SC1BD_TREN = 0;
    SDBG_SC1BD_SBR = (Word)((BFRQ_HZ/SDBG_BAUDRATE+8L)/16L);
    SDBG_SC1CR1 = (0*SCI0CR1_LOOPPS_MASK)
                  | (0*SCI0CR1_SCISWA1_MASK)
                  | (0*SCI0CR1_RSRC_MASK)
                  | (0*SCI0CR1_M_MASK)
                  | (0*SCI0CR1_WAKE_MASK)
                  | (0*SCI0CR1ILT_MASK)
                  | (0*SCI0CR1PE_MASK)
                  | (0*SCI0CR1PT_MASK);
    SDBG_SC1CR2 = (0*SCI0CR2_TIE_MASK)
                  | (0*SCI0CR2_TCIE_MASK)
                  | (1*SCI0CR2_RIE_MASK)
                  | (0*SCI0CR2_LIE_MASK)
                  | (1*SCI0CR2_TE_MASK)
                  | (1*SCI0CR2_RE_MASK)
                  | (0*SCI0CR2_RMU_MASK)
                  | (0*SCI0CR2_SBK_MASK);

    sdbg_RxFlag = 0;
    sdbg_RxNCmdPtr = 0;

    for (i=0; i<SDBG_NCMD_NUM; i++)
        sdbg_pNCmdCallback[i] = NULL;
    for (i=0; i<SDBG_HCMD_NUM; i++)
        sdbg_pHCmdCallback[i] = NULL;
}
```

• Reference Manual Chapter 20. 참조

- 우선순위 설정이 없다
→ 기본 1(제일 낮은 우선순위 사용)

20.SerialDebug – 구현 (3)

- serialdebug.c command callback 함수 등록

```
// NCmd callback 함수 등록
void sdbg_RegisterNCmd(char ncmd, void (*pCallback)(char *)) {
    if ((ncmd<SDBG_NCMD_START) || (ncmd>SDBG_NCMD_END)) {
        return;
    }
    sdbg_pNCmdCallback[ncmd-SDBG_NCMD_START] = pCallback;

// HCmd callback 함수 등록
void sdbg_RegisterHCmd(char hcmd, void (*pCallback)(void)) {
    if ((hcmd<SDBG_HCMD_START) || (hcmd>SDBG_HCMD_END)) {
        return;
    }
    sdbg_pHCmdCallback[hcmd-SDBG_HCMD_START] = pCallback;
```

- serialdebug.c 전송

```
// Tx Queue가 다 차면 TRUE 리턴, 그렇지 않으면 FALSE 리턴
int SDBG_IsTxQueFull(void) {
    sWord size;
    EnterCriticalSection();
    size = (sdbg_TxQueHeadPtr-sdbg_TxQueTailPtr+SDBG_TX_QUE_SIZE)%SDBG_TX_QUE_SIZE;
    LeaveCriticalSection();
    if (size<=(SDBG_TX_QUE_SIZE-2)) {
        return TRUE;
    }
    return FALSE;
}

// Tx Queue에 Ch 넣음. 단, Tx Queue가 차 있으면 빌 때까지 기다림
void SDBG_PutCh(char ch) {
    while (SDBG_IsTxQueFull()) {
    }
    if (ch == SDBG_ASCII_LF) {
        sdbg_TxQue[sdbg_TxQueHeadPtr] = SDBG_ASCII_CR;
        sdbg_TxQueHeadPtr = (sdbg_TxQueHeadPtr+1)%SDBG_TX_QUE_SIZE;
    }
    sdbg_TxQue[sdbg_TxQueHeadPtr] = ch;
    sdbg_TxQueHeadPtr = (sdbg_TxQueHeadPtr+1)%SDBG_TX_QUE_SIZE;
    SDBG_SC1CR2_TIE = 1;
}

// Tx Queue에 pstr이 가르키는 문자열 넣음
void SDBG_PutStr(char* pstr) {
    while (*pstr!='\0') {
        SDBG_PutCh(*pstr++);
    }
}

// Tx queue에 printf 명령어로 문자열 대입
void SDBG_Printf(const char *format, ...) {
    volatile va_list args;
    set_printf(SDBG_PutCh);
    va_start(args, format);
    (void)vprintf(format, args);
    va_end(args);
}
```

- 전송의 경우: Queue가 다 차 있으면 빌 때까지 기다렸다가 Queue에 전송할 문자를 채운다.

- 전송에 인터럽트를 사용하지 않는다면 한 문자 전송에 걸리는 시간은?

- 참고: 어떤 MCU는 전송 버퍼가 비워질 때 Interrupt를 한번 발생시키지만, 이 MCU는 비워져 있으면 계속 발생시킨다.

20.SerialDebug – 구현 (4)

• serialdebug.c event 유무 및 기타

```
// Event(한 문자 명령어 또는 한 줄 명령어)가 있으면  
// TRUE 리턴, 그렇지 않으면 FALSE 리턴  
int SDBG_IsEvents(void) {  
  
    if ((sdbg_RxFlag & SDBG_RX_NCMD)  
        || (sdbg_RxFlag & SDBG_RX_HCMD)) {  
        return TRUE;  
    }  
    return FALSE;  
}  
  
// 등록되어 있지 않은 command가 있을 때 출력  
void SDBG_PrintNoRegCmd(void) {  
    (void)SDBG_Printf("No Reg. Cmd\n");  
}
```

• serialdebug.c 전송 받은 command 처리

```
// SDBG_IsEvents()가 TRUE 일 때 Event에 해당하는 함수 실행  
void SDBG_ExecuteCallback(void) {  
    char i, cmd;  
  
    // 한 문자 명령어에 대한 callback 함수 호출  
    if (sdbg_RxFlag & SDBG_RX_HCMD) {  
        if ((sdbg_RxHCmd>=SDBG_HCMD_START) && (sdbg_RxHCmd<=SDBG_HCMD_END)) {  
            if (sdbg_pHCmdCallback[sdbg_RxHCmd-SDBG_HCMD_START] != NULL) {  
                sdbg_pHCmdCallback[sdbg_RxHCmd-SDBG_HCMD_START]();  
            } else {  
                SDBG_PrintfNoRegCmd();  
            }  
        } else {  
            SDBG_PrintfNoRegCmd();  
        }  
        sdbg_RxFlag &= ~SDBG_RX_HCMD;  
    }  
  
    // 한 줄 명령어에 대한 callback 함수 호출  
    if (sdbg_RxFlag & SDBG_RX_NCMD) {  
        for (i=0; i<sdbg_RxNCmdPtr; i++) {  
            if ((sdbg_RxNCmd[i] == SDBG_ASCII_LF) || sdbg_RxNCmd[i] == ' ') {  
                continue;  
            }  
            break;  
        }  
        if (i != sdbg_RxNCmdPtr) {  
            cmd = sdbg_RxNCmd[i];  
            if ((cmd>=SDBG_NCMD_START) && (cmd<=SDBG_NCMD_END)) {  
                if (sdbg_pNCmdCallback[cmd-SDBG_NCMD_START] != NULL) {  
                    sdbg_pNCmdCallback[cmd-SDBG_NCMD_START]((char *)&sdbg_RxNCmd[i+1]);  
                } else {  
                    SDBG_PrintfNoRegCmd();  
                }  
            } else {  
                SDBG_PrintfNoRegCmd();  
            }  
        } else {  
            SDBG_PrintfNoRegCmd();  
        }  
        sdbg_RxNCmdPtr = 0;  
        sdbg_RxFlag &= ~SDBG_RX_NCMD;  
    }  
}
```

20.SerialDebug – 사용 예제

• callback 함수

```
// 'A'를 누르면 실행되는 함수  
void Callback_A(void) {  
    DLED_SetBits(DLED_LED3_BV);  
    SDBG_Printf("LnLED_3 is ON!Ln");  
}
```

• ‘A’

```
// 'B'를 누르면 실행되는 함수  
void Callback_B(void) {  
    DLED_ResetBits(DLED_LED3_BV);  
    SDBG_Printf("LnLED_3 is OFF!Ln");  
}
```

• ‘B’

```
// "a 3 1"처럼 입력한 후 엔터를 누르면 실행되는 함수  
void Callback_a(char *str) {  
    sWord lednum, val;  
    if (sscanf(str, "%d %d", &lednum, &val)<2) {  
        SDBG_Printf("LnParameter Missing!Ln");  
        return;  
    }  
    lednum &= 0x3;  
  
    if (val)  
        DLED_SetBits((Byte)(1<<lednum));  
    else  
        DLED_ResetBits((Byte)(1<<lednum));  
  
    SDBG_Printf("LnLED_%u is %uLn", lednum, val);  
}
```

• “a 2 1”

```
// "b" 후 엔터를 누르면 실행되는 함수  
void Callback_b(char *str) {  
    (void)str;  
    SDBG_Printf("LnDIP Status = %x(hex)Ln", DSW_GetDip());  
}
```

• “b”

• 초기화, 등록, foreground

```
void main(void) {  
    unsigned int i=0;  
  
    InitPeripherals(); // 이 안에 SDBG_Init() 실행  
  
    SDBG_RegisterHCmd('A', Callback_A);  
    SDBG_RegisterHCmd('B', Callback_B);  
    SDBG_RegisterNCmd('a', Callback_a);  
    SDBG_RegisterNCmd('b', Callback_b);  
  
    for(;;){  
        if (SDBG_IsEvents()) {  
            SDBG_ExecuteCallback();  
        }  
    }  
}
```

• 메인 루프

20.SerialDebug – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- 반드시 Hot Command와 Normal Command를 만들어 사용해 보세요.

❖ 일러 두기

- UART 통신을 이용한 Debugging 기술은 매우 다양합니다.

- ◆ 단순 입출력에서
 - ◆ 하드웨어 디버거와 동일한 기능까지...

- 간단하지만 유용하게 제작

- ◆ 출력은 당연!
 - ◆ 한 문자 입력에 의한 실행도 가능하게...
 - 제어할 때 정말 편리하게 사용될 것입니다!

❖ 느낀 점 및 참고사항 적기

ISR 실행 시간은 어떻게 측정할 수 있을까?

❖ ISR 실행 시간을 측정해야 하는 이유

■ ISR 실행 시간이 길면

- ◆ Interrupt를 놓칠 수 있다 → 심각한 문제
- ◆ ISR이 아닌 부분이 느려진다
- ◆ Pending ISR의 Pending 시간이 길어져 문제가 될 수 있다.
 - 이 경우는 Priority로 해결할 수도...

❖ 어떻게 측정할 수 있을까?

■ Simulator 또는 Debugger를 통해 측정한다

- ◆ 어떤 MCU용 Simulator or Debugger에는 실행 시간을 M.C.(machine cycle)로 보여주는 기능도 있다.

■ Assembly Code로 계산한다

- ◆ Assembly Code의 실행시간을 Manual을 통해 계산 (복잡)

■ 오실로스코프로 측정한다

- ◆ ISR을 호출하기 전후에 디지털 출력 포트로 신호를 내보내서
- ◆ 또는 ISR 내부에서 시작 부분과 끝 부분에 디지털 출력 포트로 신호를 내보내서...
- ◆ 장비만 있으면 제일 간단할지도...

❖ 실습: basetimer.c 중에서 PIT0의 ISR 시간 측정하기

21.SdbgPrj – 과제

❖ 문제

- 시리얼 통신으로 “R”을 누르면 1초마다 런타임 출력 기능 On/Off 토글
 - ◆ 런타임 출력 On 상태면 1초에 한번씩 런타임을 출력
- 시리얼통신으로 “g x”를 입력하면 x에 해당하는 구구단을 출력하라
- 시리얼통신으로 “p x”를 입력하면 x까지의 정수 중 소수를 출력하라

```
Bool rFlag = FALSE;
void CALLBACK(void) {
    Word tNew, tOld=-1;
    Word i;
    Bool isPrime(Word n) {
        for (i=2; i*i<n; i++) {
            if ((n%i) == 0)
                return FALSE;
        }
        return TRUE;
    }
    if (CALLBACK_R)
        DBG_RegisterHCmd(R, CALLBACK_R);
    DBG_RegisterNCmd(p, CALLBACK_P);
    CALLBACK(i, val);
    if (scanf(str, "%d", &val)<1)
        DBG_Printf("Parameter Missing!%n");
    for (i=1; i<9; i++) {
        DBG_Printf("%d * %d = %d\n", val, i,
                  val*i);
    }
    DBG_Printf("WParameter Missing!%n");
}
void CALLBACK(char *str) {
    if (scanf(str, "%d", &val)<1)
        DBG_Printf("Parameter Missing!%n");
    sword i, val;
    if (DBG_LSEvents())
        DBG_ExecuteCaliback();
    for (;;) {
        if (DBG_LSEvents())
            DBG_ExecuteCaliback();
        if (rFlag)
            tNew = BTMR_GetRunTime()/1000UL;
        if (tNew != tOld)
            DBG_Printf("RunTime = %lu\n", tNew);
        tOld = tNew;
    }
}
void main() {
    Word i;
    for (i=1; i<1000000000; i++)
        if (isPrime(i))
            DBG_Printf("%d\n", i);
}

```

22.MyUart

❖ 학습 목표

- UART 통신 프로그램을 만들어 활용할 수 있다.

❖ 문제

- 21.SerialDebug와 동일한 형태로 MC9S12XEP의 SCI2를 이용하는 UART 통신 프로그램을 작성하라

❖ 방법

- serialdebug.h 와 serialdebug.c 를 각각 myuart.h 와 myuart.c로 복사한 후
- SDBG는 MUART로 변경, sdbg는 muart로 변경
- 이 외 두어 군데 더 수정

❖ 목적

- 추후 RXD2(PJ0핀), TXD2(PJ1핀)을 이용한 UART 통신에 사용하기 위해...
 - ◆ SCI0는 디버깅을 위한 통신에...
 - ◆ SCI2는 PC나 Smartphone과의 데이터 통신에...
- 나중에 각 팀의 용도에 맞게 프로그램을 변경해 사용하세요!

23.CharLcd – 개요

❖ 학습 목표

- Character LCD와 MCU의 인터페이스를 할 수 있다.
- Character LCD에 문자를 출력할 수 있다.
- Character LCD에 출력에 Periodic Interrupt Timer를 사용할 수 있다.
- 사용자 폰트를 만들어 출력할 수 있다.

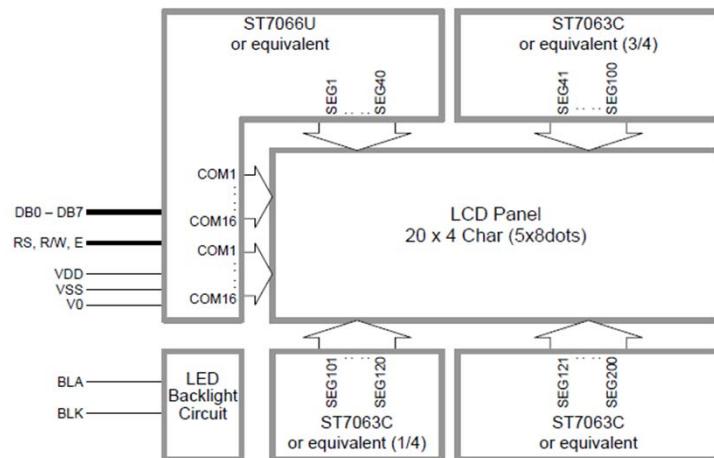
❖ 문제

- Character LCD를 MCU의 GPIO와 연결할 수 있는 회로를 설계한다
 - ◆ 8비트 인터페이스 대신 4비트 인터페이스를 이용한다
- Instruction Code와 Character Data를 출력할 수 있는 함수를 제공한다
- Character Data를 Printf 함수처럼 출력할 수 있는 함수를 제공한다
- Cursor의 위치를 이동시킬 수 있는 함수를 제공한다
- Instruction Code 중 Busy Flag를 읽을 수 있는 함수를 제공한다
- Periodic Interrupt Timer를 이용해서 불필요한 출력 시간 지연을 방지한다
 - ◆ 예를 들어 한 문자를 LCD에 보내는 데에는 10 usec 정도 걸리지만, LCD가 다음 문자를 받아 들이는 데에는 40 usec가 필요하다고 가정하자. 그러면 20문자를 출력하기 위해서는 $(10+40)*20 = 1000 \text{ usec} = 1 \text{ msec}$ 가 필요하다. 1 msec가 경우에 따라서 긴 시간이다. 한 함수로 이 작업을 수행하면, 이 시간 동안 다른 작업을 할 수 없다!
 - ◆ PIT와 Queue를 이용해서 ISR에서 한 문자씩 출력하게 하면 비록 총 처리 시간의 합은 ISR을 사용하지 않을 때보다 더 걸릴 수도 있다. 하지만, Queue에 올리는 함수의 실행시간은 짧으며, 이후 ISR이 일정 시간 간격으로 LCD에 한 문자씩 출력하므로 문자 출력 중간 중간에도 다른 작업을 할 수 있다.

23.CharLcd – 인터페이스

❖ TOPWAY Technology Co., Ltd의 LMB204BFC 제품의 User Manual 발췌

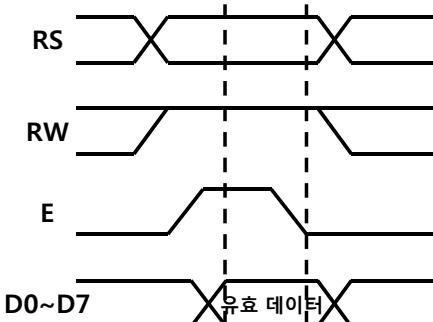
- 참고: Character LCD의 인터페이스 방법은 회사나 제품에 상관없이 거의 동일
- 메모리 R/W Bus를 이용할 수도 있지만, GPIO로도 간단히 구현 가능



1.4 Terminal Functions

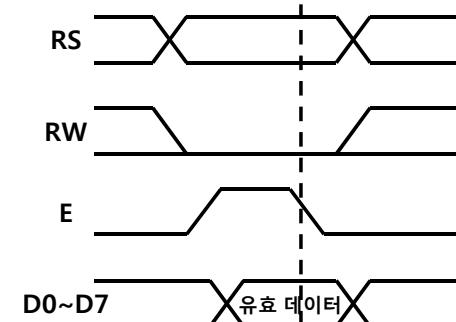
Pin No.	Pin Name	I/O	Descriptions
1	VSS	Power	Power supply, Ground (0V)
2	VDD	Power	Positive power supply
3	V0	Power	LCD contrast reference supply
4	RS	Input	Register Select RS=HIGH: transferring display data RS=LOW: transferring instruction data
5	R/W	Input	Read / Write Control: R/W=HIGH: Read mode selected R/W=LOW: Write mode selected
6	E	Input	Data Enable
7	DB0	I/O	Bi-directional tri-state Data bus In 8 bit mode, DB0 ~ DB7 are in use In 4 bit mode, DB4 ~ DB7 are in use, DB0~DB3 leave open
14	DB7		
15	BLA	Power	Backlight positive supply
16	BLK	Power	Backlight negative supply

➤ Read Operation



- RW가 H이고 E가 L→H→L 시점에서 LCD로부터 D0~D7을 읽을 수 있음
 - RS가 L이면 Instruction, H이면 Data에 해당

➤ Write Operation



- RW가 H이고 E가 L→H→L 시점의 D0~D7이 LCD로 저장됨
 - RS가 L이면 Instruction, H이면 Data에 해당

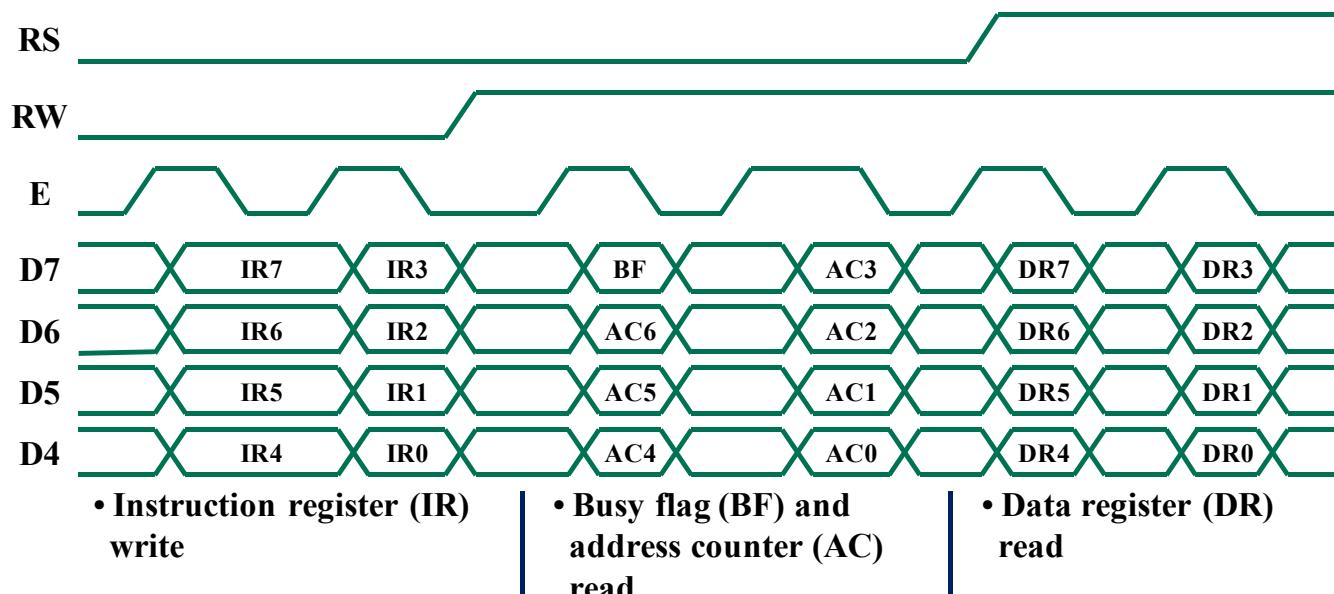
23.CharLcd – 4 bit 인터페이스

❖ 8 bit 인터페이스

- RS, RW, E, D0~D7 까지 총 11개의 GPIO 필요

❖ 4 bit 인터페이스

- RS, RW, E, D4~D7 까지 총 7개의 GPIO 필요
- 8 bit 데이터를 4 bit 씩 두 번 액세스



23.CharLcd – Command

❖ 참고하세요!

■ 어떤 LCD datasheet를 보면, 실행 시간이 39usec.이고, Clear Display만 1.53msec.

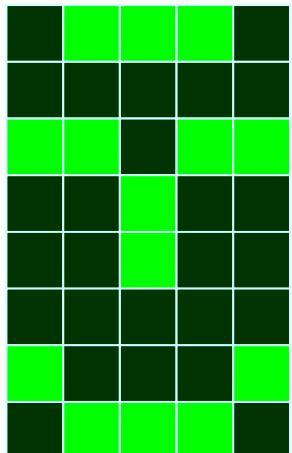
Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clock-Cycles						
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0						
Clear Display	0	0	0	0	0	0	0	0	0	1	Clear display & set address counter to zero	165						
Cursor Home	0	0	0	0	0	0	0	0	1	x	Set address counter to zero, return shifted display to original position. DD RAM contents remains unchanged.	3						
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direction (I/D) and specify automatic display shift (S).	3						
Display Control	0	0	0	0	0	0	1	D	C	B	Turn display (D), cursor on/off (C), and cursor blinking (B).	3						
Cursor / Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Shift display or move cursor (S/C) and specify direction (R/L).	3						
Function Set	0	0	0	0	1	DL	N	F	x	x	Set interface data width (DL), number of display lines (N) and character font (F).	3						
Set CGRAM Address	0	0	0	1	CGRAM Address					Set CGRAM address. CGRAM data is sent afterwards.		3						
Set DDRAM Address	0	0	1	DDRAM Address					Set DDRAM address. DDRAM data is sent afterwards.		3							
Busy Flag & Address	0	1	BF	Address Counter					Read busy flag (BF) and address counter		0							
Write Data	1	0	Data					Write data into DDRAM or CGRAM		3		3						
Read Data	1	1	Data					Read data from DDRAM or CGRAM		3		3						
<hr/>																		
x : Don't care	I/D	1 0	Increment Decrement				R/L	1 0	Shift to the right Shift to the left									
	S	1 0	Automatic display shift				DL	1 0	8 bit interface 4 bit interface									
	D	1 0	Display ON Display OFF				N	1 0	2 lines 1 line									
	C	1 0	Cursor ON Cursor OFF				F	1 0	5x10 dots 5x7 dots									
	B	1 0	Cursor blinking				DDRAM: Display Data RAM CGRAM: Character Generator RAM											
	S/C	1 0	Display shift Cursor move															

무단 전재, 복제, 배포를 금합니다



23.CharLcd – 사용자 문자 정의

- ❖ 최대 8자까지 사용자 문자를 정의하고 출력이 가능함



→ 01110 = 0x0e
→ 00000 = 0x00
→ 11011 = 0x1b
→ 00100 = 0x04
→ 00100 = 0x04
→ 00000 = 0x00
→ 10001 = 0x11
→ 01110 = 0x0e

- clcd.c에서 사용자 폰트 1에 등록한 예

```
Byte clcd_Font[CLCD_FONTPNUM*8] =  
{  
    0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f,  
    0x0e, 0x00, 0x1b, 0x04, 0x04, 0x00, 0x11, 0x0e,  
    0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00,  
    0x0a, 0x0a, 0x0a, 0x0a, 0x0a, 0x0a, 0x0a, 0x0a,  
    0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f,  
    0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00,  
    0x1f, 0x04, 0x1b, 0x00, 0x1f, 0x04, 0x10, 0x1f,  
    0x15, 0x0a, 0x15, 0x0a, 0x15, 0x0a, 0x15, 0x0a  
};
```

- LCD 두 번째 줄 제일 왼쪽에 사용자 폰트 1 표시

```
lcdGotoXY(0, 1); // 커서 이동  
lcdPutch(1); // 사용자 폰트 1 출력
```

23.CharLcd – 구상 (1)

❖ 전역 변수

- 인터페이스에 사용되는 GPIO 레지스터
- PIT 레지스터
- Byte clcd_Font[CLCD_FONTPNUM*8]
 - ◆ 사용자 폰트를 저장한 배열
- volatile sWord clcd_QueHeadPtr, clcd_QueTailPtr
 - ◆ Queue 포인터
- volatile Word clcd_Que[CLCD_QUE_SIZE];
 - ◆ 전송할 데이터를 보관하고 있는 Queue로 전송할 데이터는 하위 8비트
 - ◆ B8 자리(←CLCD_INST_MASK)가 1이면 Instruction Code, 0이면 Character Data

❖ 함수

- interrupt CLCD_VectorNumber_Vpit void CLCD_PIT_ISR(void)
 - ◆ PIT ISR
- void CLCD_Init(void);
 - ◆ 초기화 함수
- void CLCD_WriteCh(Byte data);
 - ◆ data를 Character Data로 하여 LCD에 전송
- void CLCD_WriteInst(Byte data);
 - ◆ data를 Instruction Code로 하여 LCD에 전송
- void CLCD_WriteCore(Byte data);
 - ◆ CLCD_WriteCh와 CLCD_WriteInst에서 호출하는 함수로 4-bit 인터페이스로 data를 LCD에 전송

23.CharLcd – 구상 (2)

❖ 함수 계속

- **Byte CLCD_ReadBusyFlag(void);**
 - ◆ LCD로부터 Busy Flag를 읽어 리턴

- **void CLCD_PutChDir(char ch);**
 - ◆ data를 Character Data로 하여 LCD에 전송하고 시간 지연 (약 100usec)
- **void CLCD_PutInstDir(Byte data);**
 - ◆ data를 Instruction Code로 하여 LCD에 전송하고 시간 지연 (약 100usec)

- **Bool CLCD_IsQueFull(void);**
 - ◆ Queue가 찼는지 검사
- **void CLCD_PutCh(char ch);**
 - ◆ data를 Character Data로 하여 Queue에 저장. Queue가 다 차 있으면 빌 때까지 기다림
- **void CLCD_PutInst(Byte data);**
 - ◆ data를 Instruction Code로 하여 Queue에 저장. Queue가 다 차 있으면 빌 때까지 기다림
- **void CLCD_GotoXy(Byte x, Byte y); // Cursor를 이동. zero based**
 - ◆ LCD의 커서를 x, y로 이동시킴. x는 맨 왼쪽 칸부터 0, y는 맨 윗줄부터 0
- **void CLCD_Printf(const char *format, ...); // Printf 함수**
 - ◆ printf 함수처럼 LCD에 문자열 출력. Queue에 저장
- **void CLCD_PrintfXy(Byte x, Byte y, const char *format, ...)**
 - ◆ x, y 위치부터 printf 함수처럼 문자열 출력

23.CharLcd – 구현 (1)

• charlcd.h

```
#ifndef _CLCD_H_
#define _CLCD_H_

#ifndef __cplusplus
extern "C" {
#endif

#include "project.h"

//----- MACRO -----
#define CLCD_COLUMN 16
#define CLCD_CLEAR 0x1
#define CLCD_CHOME 0x2
#define CLCD_CURSOR_ON 0xa
#define CLCD_CURSOR_OFF 0x8
#define CLCD_BUSY 0x80

#define CLCD_FONTPNUM 8

#define CLCD_SAMPLE_TIME 100 // usec
#define CLCD_QUE_SIZE 100
#define CLCD_INST_MASK 0x100
```

• PIT 주기

• Queue 사이즈

// CLCD에 사용되는 레지스터 및 인터럽트, 환경에 맞게 수정

```
#define CLCD_RS PTT_PTT1
#define CLCD_RW PTT_PTT2
#define CLCD_E PTT_PTT3
#define CLCD_D4 PORTA_PA4
#define CLCD_D5 PORTA_PA5
#define CLCD_D6 PORTA_PA6
#define CLCD_D7 PORTA_PA7
#define CLCD_D7IN PORTA_PA7
```

• 인터페이스에 사용되는 GPIO 입출력 레지스터. 필요에 따라 변경

```
#define CLCD_RS_DDR DDRT_DDRT1
#define CLCD_RW_DDR DDRT_DDRT2
#define CLCD_E_DDR DDRT_DDRT3
#define CLCD_D4_DDR DDRA_DDRA4
#define CLCD_D5_DDR DDRA_DDRA5
#define CLCD_D6_DDR DDRA_DDRA6
#define CLCD_D7_DDR DDRA_DDRA7
```

• 인터페이스에 사용되는 GPIO 방향 설정 레지스터. 원부분에 맞게 설정

```
#define CLCD_VectorNumber_Vpit VectorNumber_Vpit7
```

```
#define CLCD_PITLD PITLD7
#define CLCD_PITCNT PITCNT7
#define CLCD_PITMUX_PMUX PITMUX_PMUX7
#define CLCD_PITCE_PCE PITCE_PCE7
#define CLCD_PITTF_PTF_MASK PITTF_PTF7_MASK
#define CLCD_PITINTE_PINTE PITINTE_PINTE7
```

• PIT 관련 레지스터: 필요에 따라 변경

23.CharLcd – 구현 (2)

- charlcd.h 계속

```
//- FUNCTION -----
void CLCD_Init(void);

void CLCD_WriteCh(Byte data);
void CLCD_WriteInst(Byte data);
void CLCD_WriteCore(Byte data);
Byte CLCD_ReadBusyFlag(void);

void CLCD_PutChDir(char ch);
void CLCD_PutInstDir(Byte data);

Bool CLCD_IsQueFull(void);
void CLCD_PutCh(char ch);
void CLCD_PutInst(Byte data);
void CLCD_GotoXY(Byte x, Byte y);
void CLCD_Printf(const char *format, ...);
void CLCD_PrintfXY(Byte x, Byte y, const char *format, ...);

#ifndef __cplusplus
}
#endif

#endif
```

- charlcd.c 의 전역변수

```
Byte clcd_Font[CLCD_FONTPNUM*8] =
{
    0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, // User Font 0
    0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, // User Font 1
    0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, // User Font 2
    0x0a, 0x0a, 0x0a, 0x0a, 0x0a, 0x0a, 0x0a,
    0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f,
    0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00, 0x1f, 0x00,
    0x1f, 0x04, 0x1b, 0x00, 0x1f, 0x04, 0x10, 0x1f,
    0x15, 0x0a, 0x15, 0x0a, 0x15, 0x0a, 0x15, 0x0a // User Font 7
};

volatile sWord clcd_QueHeadPtr, clcd_QueTailPtr;
volatile Word clcd_Que[CLCD_QUE_SIZE];
```

23.CharLcd – 구현 (3)

• charlcd.c 의 PIT ISR

```
// Vpit Interrupt Service Routine
#pragma CODE_SEG __NEAR_SEG NON_BANKED

interrupt CLCD_VectorNumber_Vpit void CLCD_PIT_ISR(void) {
    if (clcd_QueHeadPtr!=clcd_QueTailIPtr) {
        if ((CLCD_ReadBusyFlag()&CLCD_BUSY)==0) {
            if (clcd_Que[clcd_QueTailIPtr]&CLCD_INST_MASK) {
                CLCD_WriteInst((Byte)(clcd_Que[clcd_QueTailIPtr] & 0xff));
            } else {
                CLCD_WriteCh((Byte)clcd_Que[clcd_QueTailIPtr]);
            }
            clcd_QueTailIPtr = (clcd_QueTailIPtr+1)%CLCD_QUE_SIZE;
        }
    } else {
        CLCD_PITINTE_PINTE = 0;
    }
    PITTF = CLCD_PITTF_PTF_MASK;
}
```

```
#pragma CODE_SEG DEFAULT
```

- Queue에 데이터가 없으면 Interrupt Disable
- Queue에 데이터가 있고, LCD가 Busy 상태가 아니면 데이터 전송!

• charlcd.c 의 시간 지연 함수

```
// 약 100us 단위로 시간을 지연시키는 함수
void CLCD_Delay(unsigned int us100) {
    volatile int i;

    while (us100--) {
#if BFRQ_OPTION == BFRQ_2MHZ
        for (i=0; i<33; i++) {
#elif BFRQ_OPTION == BFRQ_20MHZ
        for (i=0; i<332; i++) {
#elif BFRQ_OPTION == BFRQ_40MHZ
        for (i=0; i<665; i++) {
#endif
    }
}
    }
```

• charlcd.c 내부에서
만 사용됨

23.CharLcd – 구현 (4)

- charlcd.c 의 CLCD_Init

```
void CLCD_Init(void) {  
    Word fn, i, offs;  
  
    // 포트 초기화  
    CLCD_RS_DDR = 1;  
    CLCD_RW_DDR = 1;  
    CLCD_E_DDR = 1;  
    CLCD_D4_DDR = 1;  
    CLCD_D5_DDR = 1;  
    CLCD_D6_DDR = 1;  
    CLCD_D7_DDR = 1;  
  
    CLCD_E = 0;
```

```
// 타이머 초기화  
CLCD_PITLD = (CLCD_SAMPLE_TIME-1);  
CLCD_PITMUX_PMUX = 0;           // micro time base 0  
CLCD_PITCE_PCE = 1;             // Enable PIT Ch. n  
PITTF = CLCD_PITTF_PTF_MASK;    // Clear Flag  
CLCD_PITINTE_PINTE = 1;         // Enable PIT Ch. n Interrupt  
  
CLCD_Delay(500); // 50 msec 시간 지연. 왜? • 왜?  
  
// CLCD 초기화  
CLCD_PutInstDir(0x28);          // 4-bit interface, 2-line display  
CLCD_PutInstDir(0x28);          // 4-bit interface, 2-line display • 왜 두 번?  
CLCD_PutInstDir(0x06);          // Incremental cursor movement  
CLCD_PutInstDir(0x0c);          // Display ON, cursor OFF and blink OFF  
CLCD_PutInstDir(CLCD_CLEAR);    // Clear LCD screen  
CLCD_Delay(20);  
  
for (fn=0; fn<CLCD_FONTPNUM; fn++) {  
    for (i=0; i<8; i++) {  
        offs = fn*8+i;  
        CLCD_PutInstDir(0x40|offs);  
        CLCD_PutChDir(clcd_Font[offs]);  
    }  
}
```

• 사용자 문자 등록

23.CharLcd – 구현 (5)

- charlcd.c 의 핵심 출력부

```
void CLCD_WriteCh(Byte data) {
    CLCD_RS = 1;
    CLCD_RW = 0;
    CLCD_E = 0;
    CLCD_WriteCore(data);
}

void CLCD_WriteInst(Byte data) {
    CLCD_RS = 0;
    CLCD_RW = 0;
    CLCD_E = 0;
    CLCD_WriteCore(data);
}

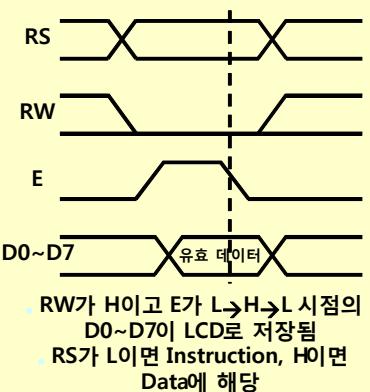
void CLCD_WriteCore(Byte data)
{
    CLCD_D4 = (data>>4) & 0x1;
    CLCD_D5 = (data>>5) & 0x1;
    CLCD_D6 = (data>>6) & 0x1;
    CLCD_D7 = (data>>7) & 0x1;

    CLCD_E = 1;
    // 600 nano seconds
    #if BFRQ_OPTION == BFRQ_2MHZ
        __asm(nop);
    #elif BFRQ_OPTION == BFRQ_20MHZ
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
        __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    #endif
}
```

```
void CLCD_PutChDir(char ch) {
    CLCD_WriteCh(ch);
    CLCD_Delay(1);
}

void CLCD_PutInstDir(Byte data) {
    CLCD_WriteInst(data);
    CLCD_Delay(1);
}
```

➤ Write Operation



- RW가 H이고 E가 L→H→L 시점의 D0~D7이 LCD로 저장됨
- RS가 L이면 Instruction, H이면 Data에 해당

```
#elif BFRQ_OPTION == BFRQ_40MHZ
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
#endif
    CLCD_E = 0;

    CLCD_D4 = data & 0x1;
    CLCD_D5 = (data>>1) & 0x1;
    CLCD_D6 = (data>>2) & 0x1;
    CLCD_D7 = (data>>3) & 0x1;

    CLCD_E = 1;
    // 600 nano seconds
    #if BFRQ_OPTION == BFRQ_2MHZ
        __asm(nop);
    #elif BFRQ_OPTION == BFRQ_20MHZ
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
        __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    #elif BFRQ_OPTION == BFRQ_40MHZ
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
        __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    #endif
    CLCD_E = 0;
```

- 데이터시트의 타이밍 정보를 고려

23.CharLcd – 구현 (6)

• charlcd.c 의 Busy Flag 읽기

```
Byte CLCD_ReadBusyFlag(void) {
    Byte data=0;

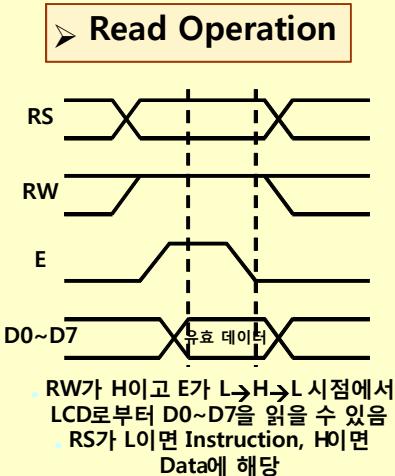
    CLCD_D7_DDR = 0;
    CLCD_D6_DDR = 0;
    CLCD_D5_DDR = 0;
    CLCD_D4_DDR = 0;

    CLCD_RS = 0;
    CLCD_RW = 1;
    CLCD_E = 0;

    CLCD_E = 1;
    // 320 nano seconds
#if BFRQ_OPTION == BFRQ_2MHZ
    __asm(nop);
#elif BFRQ_OPTION == BFRQ_20MHZ
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
#endif
    data = (CLCD_D7IN) ? 0x80 : 0;
    CLCD_E = 0;

    // 600 nano seconds
#if BFRQ_OPTION == BFRQ_2MHZ
    __asm(nop);
#elif BFRQ_OPTION == BFRQ_20MHZ
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop);

```



- 데이터시트의 타이밍 정보를 고려

```
#elif BFRQ_OPTION == BFRQ_40MHZ
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
#endif
    CLCD_E = 1;
    // 600 nano seconds
#if BFRQ_OPTION == BFRQ_2MHZ
    __asm(nop);
#elif BFRQ_OPTION == BFRQ_20MHZ
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
    __asm(nop); __asm(nop); __asm(nop); __asm(nop); __asm(nop);
#endif
    CLCD_E = 0;

    CLCD_D7_DDR = 1;
    CLCD_D6_DDR = 1;
    CLCD_D5_DDR = 1;
    CLCD_D4_DDR = 1;

    return data;
}
```

23.CharLcd – 구현 (7)

• charlcd.c 의 Queue 이용 출력

```
Bool CLCD_IsQueFull(void) {
    sWord size;
    EnterCriticalSection();
    size = (clcd_QueHeadPtr-clcd_QueTailPtr+CLCD_QUE_SIZE)%CLCD_QUE_SIZE;
    LeaveCriticalSection();
    if (size>=(CLCD_QUE_SIZE-2)) {
        return TRUE;
    }
    return FALSE;
}

void CLCD_PutCh(char ch) {
    while (CLCD_IsQueFull()) {
    }
    clcd_Que[clcd_QueHeadPtr] = ch;
    clcd_QueHeadPtr = (clcd_QueHeadPtr+1)%CLCD_QUE_SIZE;
    CLCD_PITINTE_PINTE = 1;
}

void CLCD_PutInst(Byte data) {
    while (CLCD_IsQueFull()) {
    }
    clcd_Que[clcd_QueHeadPtr] = (Word)data | CLCD_INST_MASK;
    clcd_QueHeadPtr = (clcd_QueHeadPtr+1)%CLCD_QUE_SIZE;
    CLCD_PITINTE_PINTE = 1;
}
```

- Queue가 차 있는 경우
는 시간 지연이 발생!

• charlcd.c 의 Cursor 이동 및 printf

```
void CLCD_GotoXy(Byte x, Byte y) {
    switch (y) {
    case 0:
        CLCD_PutInst(0x80+x); break;
    case 1:
        CLCD_PutInst(0xc0+x); break;
    case 2:
        CLCD_PutInst(0x94+x); break;
    case 3:
        CLCD_PutInst(0xd4+x); break;
    }
}

void CLCD_Printf(const char *format, ...) {
    volatile va_list args;
    set_printf(CLCD_PutCh);
    va_start(args, format);
    (void)vprintf(format, args);
    va_end(args);
}

void CLCD_PrintfXy(Byte x, Byte y, const char *format, ...) {
    volatile va_list args;
    CLCD_GotoXy(x, y);
    set_printf(CLCD_PutCh);
    va_start(args, format);
    (void)vprintf(format, args);
    va_end(args);
}
```

24.PushBtnEventPrj – 과제

❖ 문제

- 두 개의 Push Button 상태를 0.01초 주기의 Periodic Interrupt Timer1로 체크하여 두 버튼의 상태 변화와 시점을 Queue에 저장해두고, 그 결과를 3초마다 출력하여라
 - ◆ 즉 main loop에서 3초간 시간 지연을 하고, 지연 시간 동안의 버튼 상태 변화를 PIT1으로 queue에 저장해 두고, 3초 지연이 끝나면 queue의 정보를 출력

❖ 고찰

- 버튼의 감지 주기가 너무 짧으면? 너무 길면?
- 이런 방식으로 버튼 입력을 받을 때의 장단점은?

24.PushBtnEventPrj – 구현

- ISR

- main

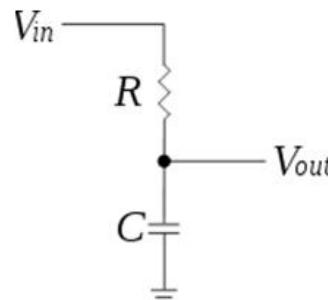
Analog to Digital Converter 일반

❖ MCU의 ADC

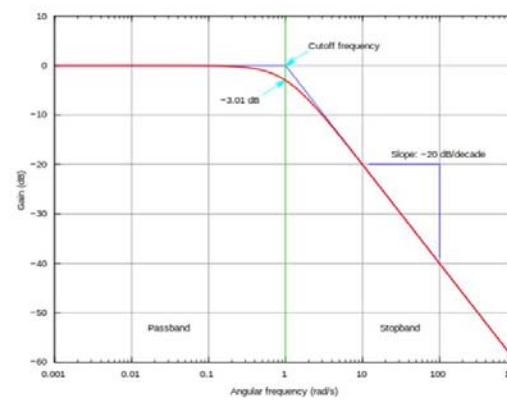
- 알아두어야 할 용어
 - ◆ Bit, Resolution, MUX(=Multiplexer), S/H(Sample and Holder), EOC (End of Conversion) 등
- MCU의 ADC는 대부분 SAR 방식
 - ◆ SAR은 DAC로 이분법으로 bit 수 만큼 반복 → 과제: SAR 방식 조사
 - ◆ Clock 필요
- 일반적인 A/D 변환 순서
 - ◆ MUX로 채널 선택
 - ◆ S/H 후 변환시작 (보통 변환을 시작하면 S/H가 진행된다)
 - ◆ 변환이 끝날 때까지 기다림
 - ◆ 변환이 끝나면 변환값 가져가기 (인터럽트도 지원)
 - ◆ 다음 변환을 위한 후처리
- 문제: f_ADC가 너무 높으면? 너무 낮으면?

❖ 참고: 노이즈를 줄이기 위한 1차 Low Pass Filter R-C 회로

- 출처: http://en.wikipedia.org/wiki/Low_pass_filter



$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$



MC9S12XEP100의 ATD (Analog to Digital)

❖ Features

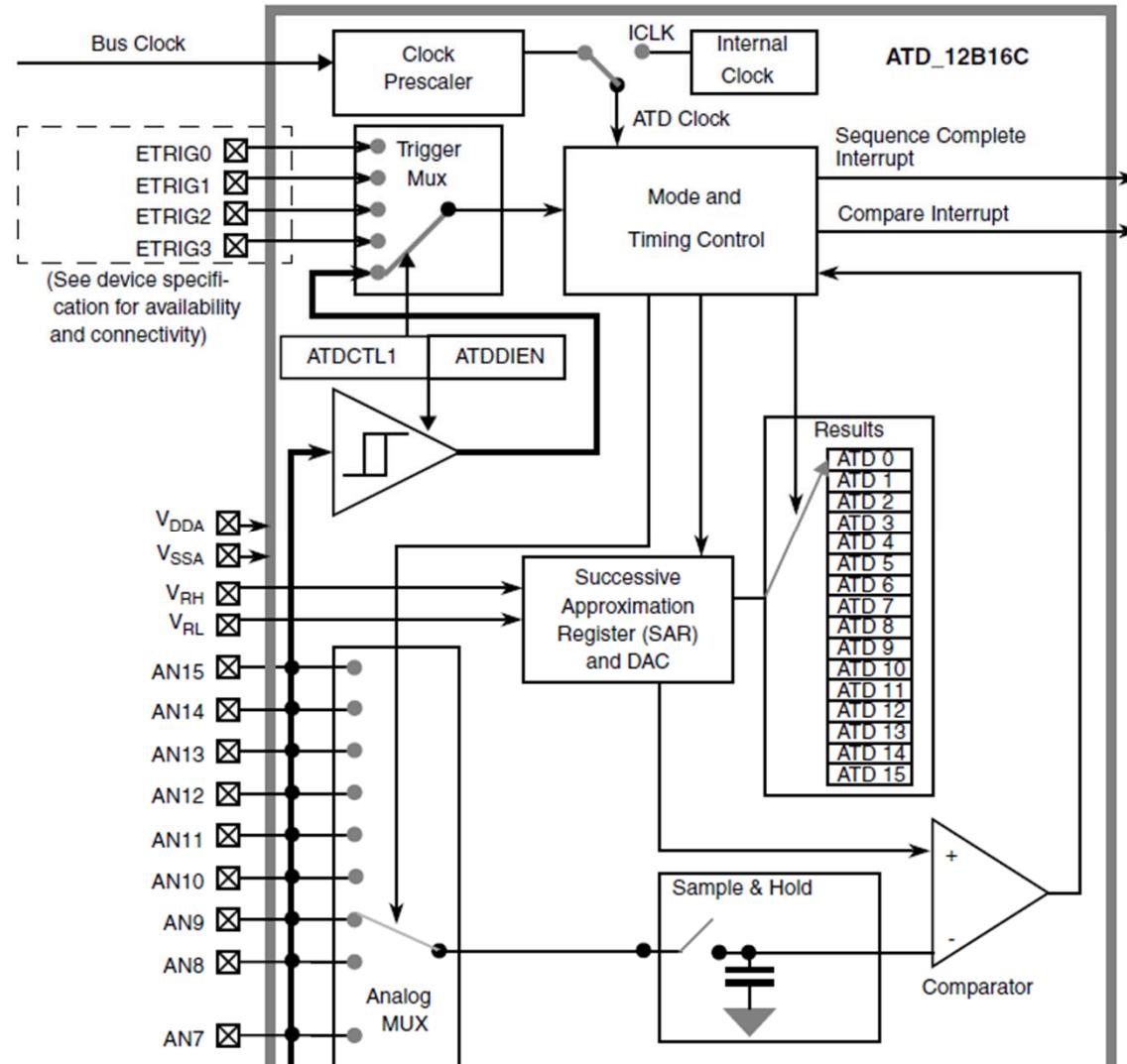
- 8-, 10-, or 12-bit resolution.
- Conversion in Stop Mode using internally generated clock
- Automatic return to low power after conversion sequence
- Automatic compare with interrupt for higher than or less/equal than programmable value
- Programmable sample time.
- Left/right justified result data.
- External trigger control.
- Sequence complete interrupt.
- Analog input multiplexer for 16 analog input channels.
- Special conversions for VRH, VRL, (VRL+VRH)/2.
- 1-to-16 conversion sequence lengths.
- Continuous conversion mode.
- Multiple channel scans.
- Configurable external trigger functionality on any AD channel or any of four additional trigger inputs. The four additional trigger inputs can be chip external or internal. Refer to device specification for availability and connectivity.
- Configurable location for channel wrap around (when converting multiple channels in a sequence).

MC9S12XEP100의 ATD (Analog to Digital)

❖ Block Diagram

❖ Registers

■ F/W에서는 ATD0로 시작!



30.AdcTest – 테스트 예제 (폴링 방식)

- ❖ Expansion Board 없이 DEMO9S12XEP100 만 이용!
- ❖ Ref. Manul Chapter 13 Analog-to-Digital Converter 참조!

• InitAdc와 GetAdc

```
void InitAdc(void) {  
    // Configures the ATD peripheral  
    // 8 bit data resolution  
    ATDOCTL0 = 0xf; // reset value  
    ATDOCTL1_ETRIGSEL = 0; // No external trigger souce  
    ATDOCTL1_ETRIGCH = 0; // external trigger channel (ignored)  
    ATDOCTL1_SRES = 2; // 0:8bit, 1:10bit, 2:12bit  
    ATDOCTL1_SMP_DIS = 1; // Discharge before Sampling  
    ATDOCTL2 = 0; // reset value  
    ATDOCTL3_DJM = 1; // Right justification  
    ATDOCTL3_FIFO = 0; // No FIFO mode  
    ATDOCTL3_FRZ = 2; // Finish current conversion, then freeze  
    ATDOCTL4_SMP = 7; // sample time select (24 clock)  
  
    // Prescaler: f_ATDCLK = f_BUS/(2*(PRS+1)) :(0.25 ~ 8.3 MHz)  
    #if BFRQ_OPTION == BFRQ_2MHZ  
        ATDOCTL4_PRS = 0; // 1 MHz  
    #elif BFRQ_OPTION == BFRQ_20MHZ  
        ATDOCTL4_PRS = 4; // 2 MHz  
    #elif BFRQ_OPTION == BFRQ_40MHZ  
        ATDOCTL4_PRS = 9; // 2 MHz  
    #endif  
}  
  
Word GetAdc(Byte ch) {  
    ATDOCTL5_Cx = ch; // Start conversion  
    while(!(ATDOSTATO & 0x80)) {  
    }  
    return ATDODR0;  
}
```

• 사용 예

```
void main(void) {  
    LWord rtime;  
    Word atd_d, atd_a;  
  
    InitPeripherals();  
    InitAdc();  
  
    for(;;) {  
        if (SDBG_IsEvents()) {  
            SDBG_ExecuteCallback();  
        }  
        if (MUART_IsEvents()) {  
            MUART_ExecuteCallback();  
        }  
  
        rtime = BTMR_GetRuntime();  
        atd_d = GetAdc(0);  
        atd_a = (Word)((100*(5-0)*(LWord)atd_d+(1<<11))>>12);  
        SDBG_Printf("%lu.%03lu sec.: Ch0 = %4u (%u.%02uV)",  
                    rtime/1000, rtime%1000,  
                    atd_d, atd_a/100, atd_a%100);  
        atd_d = GetAdc(1);  
        atd_a = (Word)((100*(5-0)*(LWord)atd_d+(1<<11))>>12);  
        SDBG_Printf("Ch1 = %4u (%u.%02uV)\n",  
                    atd_d, atd_a/100, atd_a%100);  
        BTMR_DelayMs(100);  
    }  
}
```

• 변환을 시작하는 시점은?

30.AdCTest – 정수에 의한 실수 연산 (고정소수점)

- ❖ 실수 연산 전용 프로세서를 탑재한 MCU는 별로 없다?!
 - 실수 연산 전용 프로세서가 없으면 C 언어의 실수 연산은 복잡한 정수 연산에 의해 행해진다
 - 하지만 간단한 정수 연산으로도 실수연산 효과를 낼 수 있다
- ❖ MCU의 기계어에는 정수 연산 중 +, -, >>, << 연산은 반드시 있고, *는 있는 경우도 있고, /는 없는 경우가 많다.
 - +, -, >>, <<은 모두 기계어 한 명령어로 됨
 - *는 한 명령어로 될 수 있음
 - → 이들 연산으로만 정수 연산을 하면 빠름! 빠름! 빠름!
- ❖ 예제로 배우는, 정수에 의한 실수 연산 - 정수형 변수 Data에 0.7을 곱하기!
 - (double)Data*0.7 // 엄청난 시간이 걸린다.
 - → Data * (1024*0.7)/1024
 - → Data * 717 / 1024
 - → (Data * 717) >> 10 // 오른쪽으로 한 비트 시프트 할 때마다 2로 나누는 효과
 - 그런데, 정수연산은 내림이므로 반올림을 고려하면
 - → Data * 0.7 + 0.5
 - → (Data * (1024*0.7) + 512)/1024
 - → (Data*717+512)>>10 // 이렇게 하는 걸 Q10 (2^{10})을 이용한다고 함. 유효숫자를 늘리고 싶으면 QX 중 X를 크게 하면 된다
- ❖ 질문1: 앞의 예제에서 100을 더 곱한 이유는?
- ❖ 질문2: 앞의 예제에서 (LWord) 형변환을 한 이유는?

30.AdCTest – 실습 및 고찰

❖ 실습

■ 기본 실습

- ◆ 제공되는 30.0.AdCTest 프로젝트에서 main.c의 InitAdc와 GetAdc 함수를 완성
- ◆ 완성 후 30.AdCTest와 비교해 볼 것!

■ 변환된 Digital 값을 Analog 전압으로 환산할 때, 정수 연산에 의한 방법과 실수 연산에 의한 방법의 속도 차이를 측정하고 그 결과를 적으시오.

- ◆ 수 만~ 수십 만 회 반복하는 데 걸린 시간으로 비교

■ 16채널의 총 변환 시간을 측정하고 그 결과를 적으시오.

❖ 느낀 점 및 참고사항 적기

31. MultiAdc – 목적, 문제 정의, 구상

❖ 목적 및 문제

- 목적: 여러 채널의 Analog 신호를 Digital로 변환
- 문제: 한번 명령으로 미리 선택한 채널에 대해서 각각 2^n 회 AD 변환하고 평균값 제공

❖ 필요 변수

- ADC 관련 레지스터
- volatile Byte madc_flag; // 상태 플래그
- volatile Byte madc_chnum; // 변환할 총 채널 수
- volatile Byte madc_ch_index; // 현 변환 채널 인덱스
- volatile Byte madc_sumnum; // 평균을 위한 반복측정 수
- volatile Byte madc_sum_index; // 현 반복 인덱스
- volatile Byte madc_ch[MADC_CH_MAX]; // 변환할 채널
- volatile Word madc_val[MADC_CH_MAX]; // 변환값 저장 배열
- Byte madc_0P5, madc_power; // 평균 계산을 위한 값

❖ 함수들

- interrupt VectorNumber_Vatd0 void MADC_VATD0_ISR(void) {
- void MADC_Init(void);
- void MADC_SetChAndAvg(Word channel, Byte power);
- void MADC_Start(void);
- Byte MADC_IsConverted(void);
- void MADC_Get(Word* padv);

- 초기화하고, 채널 및 평균에 사용할 측정 횟수 설정한 후
- 변환 시작하면
- 변환이 끝남을 확인하고
- 변환값을 배열로 받아감

31. MultiAdc – 구현 (1)

• multiadc.h 중에서

```
#define MADC_PRIORITY 3 // 1~7

// prescaler value for 1MHz f_ATDCLK
#if BFRQ_OPTION == BFRQ_2MHZ
#define MADC_PRS 0
#elif BFRQ_OPTION == BFRQ_20MHZ
#define MADC_PRS 9
#elif BFRQ_OPTION == BFRQ_40MHZ
#define MADC_PRS 19
#endif

#define MADC_VRH 5000 // mV
#define MADC_VRL 0 // mV
#define MADC_SPAN (MADC_VRH-MADC_VRL)
#define MADC_RES 12 // resolution (bit)

#define MADC_CH_MAX 16
#define MADC_CH00 0x0001
#define MADC_CH01 0x0002
#define MADC_CH02 0x0004
#define MADC_CH03 0x0008
#define MADC_CH04 0x0010
#define MADC_CH05 0x0020
#define MADC_CH06 0x0040
#define MADC_CH07 0x0080
#define MADC_CH08 0x0100
#define MADC_CH09 0x0200
#define MADC_CH10 0x0400
#define MADC_CH11 0x0800
#define MADC_CH12 0x1000
#define MADC_CH13 0x2000
#define MADC_CH14 0x4000
#define MADC_CH15 0x8000

#define MADC_POWER_MAX 3
#define MADC_AVG_1 0
#define MADC_AVG_2 1
#define MADC_AVG_4 2
#define MADC_AVG_8 3
```

• multiadc.c의 전역 변수 및 ISR

```
#define MADC_FLAG_COMPLETE 0x1 // 변환 종료
#define MADC_FLAG_ONCONV 0x2 // 변환 중

volatile Byte madc_flag; // 상태 플래그
volatile Byte madc_chnum; // 변환할 총 채널 수
volatile Byte madc_ch_index; // 현 변환 채널 인덱스
volatile Byte madc_sumnum; // 평균을 위한 반복측정 수
volatile Byte madc_sum_index; // 현 반복 인덱스
volatile Byte madc_ch[MADC_CH_MAX]; // 변환할 채널
volatile Word madc_val[MADC_CH_MAX]; // 변환값 저장 배열
Byte madc_OP5, madc_power; // 평균 계산을 위한 값

// Vatd0 Interrupt Service Routine
#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt VectorNumber_Vatd0 void MADC_VATD0_ISR(void) {
    // 미리 저장해 둔 채널 전체를 madc_sumnum 회수 반복 측정
    madc_val[madc_ch_index] += ATDODR0;
    if (++madc_ch_index == madc_chnum) {
        madc_ch_index = 0;
        if (++madc_sum_index == madc_sumnum) {
            madc_flag |= MADC_FLAG_COMPLETE;
            madc_flag &= ~MADC_FLAG_ONCONV;
            return;
        }
    }
    ATDOCTL5 = madc_ch[madc_ch_index];
}
#pragma CODE_SEG DEFAULT
```

31. MultiAdc – 구현 (2)

• multiadc.c의 초기화

```
// ADC를 Interrupt에 의해 여러 채널을 반복해서 측정하기 위한 초기화
void MADC_Init(void) {
    INT_CFADDR = 0xd0; // Vatd0: 0xffffd2 -> Vatd0&0xff
    INT_CFDATA1 = MADC_PRIORITY; // 0xffffd2중 제일_오른쪽_값/2

    ATDOCTL0 = 0x0fU; // reset value

    ATDOCTL1_ETRIGSEL = 0; // No external trigger source
    ATDOCTL1_ETRIGCH = 15; // external trigger channel (ignored)
    ATDOCTL1_SRES = 2; // 0:8bit, 1:10bit, 2:12bit
    ATDOCTL1_SMP_DIS = 1; // Discahrge before Sampling

    // AFFC=1: 결과 읽기로 해당 CCF[n] clear
    ATDOCTL2 = ATDOCTL2_AFFC_MASK;

    // Right justification, number of conversions per seq.=1
    ATDOCTL3 = ATDOCTL3_DJM_MASK|ATDOCTL3_S1C_MASK;

    ATDOCTL3_DJM = 1; // Right justification
    ATDOCTL3_FIFO = 0; // No FIFO mode
    ATDOCTL3_FRZ = 2; // Finish current conversion, then freeze

    ATDOCTL4_SMP = 7; // sample time select (24 clock)
    ATDOCTL4_PRS = MADC_PRS;

    madc_flag = 0;
    m adc_ch_index = 0;
    m adc_sum_index = 0;
    ATDOCTL2_ASCIE = 1; // Interrupt enable
}
```

• multiadc.c의 채널 및 측정횟수 등록

```
// ADC의 채널(channel)과 측정횟수(power) 설정
// channel의 B0부터 PAD0에 해당, 1이면 사용, 0이면 무시
// 측정회수는 2^power, power는 0, 1, 2, 3(MADC_POWER_MAX) 중 하나
void MADC_SetChAndAvg(Word channel, Byte power) {
    int i;
    Byte ch;

    // 변환할 채널을 설정
    if (channel == 0) {
        channel = 0x1;
    }
    for (ch=0, i=0; ch<MADC_CH_MAX; ch++) {
        if (channel & (1<<ch)) {
            m adc_ch[i++] = ch;
        }
    }
    m adc_chnum = (Byte)i;

    // 측정횟수 설정
    if (power>MADC_POWER_MAX) {
        power=MADC_POWER_MAX;
    }
    for (i=0, m adc_sumnum=1; i<power; i++) {
        m adc_sumnum *= 2;
    }
    m adc_power = power;
    m adc_0P5 = m adc_sumnum>>1;
}
```

31. MultiAdc – 구현 (3)

• multiadc.c의 변환 시작 및 완료 검사

```
// ADC 변환 시작
void MADC_Start(void) {
    int i;

    // 혹시라도 이전 변환이 끝나지 않았으면...
    while (madc_flag&MADC_FLAG_ONCONV) {}

    for (i=0; i<madc_chnum; i++) {
        madc_val[i] = 0;
    }

    madc_flag &= ~MADC_FLAG_COMPLETE;
    madc_ch_index = 0;
    madc_sum_index = 0;

    ATDOCTL5 = madc_ch[0];
    madc_flag |= MADC_FLAG_ONCONV;
}

// ADC 변환이 모두 완료했는지 여부, TRUE면 변환 끝
Byte MADC_IsConverted(void) {
    if (madc_flag & MADC_FLAG_COMPLETE) {
        return TRUE;
    }
    return FALSE;
}
```

• multiadc.c의 변환값 얻어오기

```
// 변환된 ADC 값을 padv 배열로 얻어옴
// MADC_SetChAndAvg로 설정한 채널 중 낮은 채널번호 순으로
// padv[0]부터 저장된다
void MADC_Get(Word* padv) {
    int i;

    for (i=0; i<madc_chnum; i++) {
        padv[i] = (madc_val[i] + madc_0P5)>>madc_power;
    }
}
```

31. MultiAdc – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- 예) “a 10 200”를 입력하면 10초 동안 200msec 간격으로 CH0, CH1, CH2의 변환값을 출력!

❖ 참고

- 이전 교육에는 MultiAdc를 소개하고 사용을 권장했으나
 - ◆ MultiAdc는 선택한 채널에 대해서
 - ◆ ADC 변환 종료 인터럽트를 이용하고,
 - ◆ 여기에 2^n 회 측정에 의한 평균도 적용
 - 결국, 변환 시작을 지시하고, 변환 종료를 기다렸다가, 변환값을 받아가는 구조!
 - 다시 말해, 내부적으로는 인터럽트를 사용하나 외부적으로는 실행 시간이 필요한 구조!
- 일정 시간 간격마다 A/D 변환하는 것도 좋을 듯! → 이 예제는 소개로 끝나고 다음 예제에서 소개하는 Adc.c 와 Adc.h를 사용!

❖ 느낀 점 및 참고사항 적기

Digital Low Pass Filter (1)

❖ Low Pass Filter란?

- 저주파 통과 필터!
- 일반적으로 전기적 노이즈는 고주파이므로 LPF로 노이즈 저감 가능!

❖ 1차 Digital Butterworth Low Pass Filter

- Sampling Time T_s

$$\frac{O(z)}{I(z)} = \frac{0.5(1-\alpha)(z+1)}{z-\alpha}$$

- Cut-off Frequency f_c

- k번째 입력, 출력 I_k, O_k

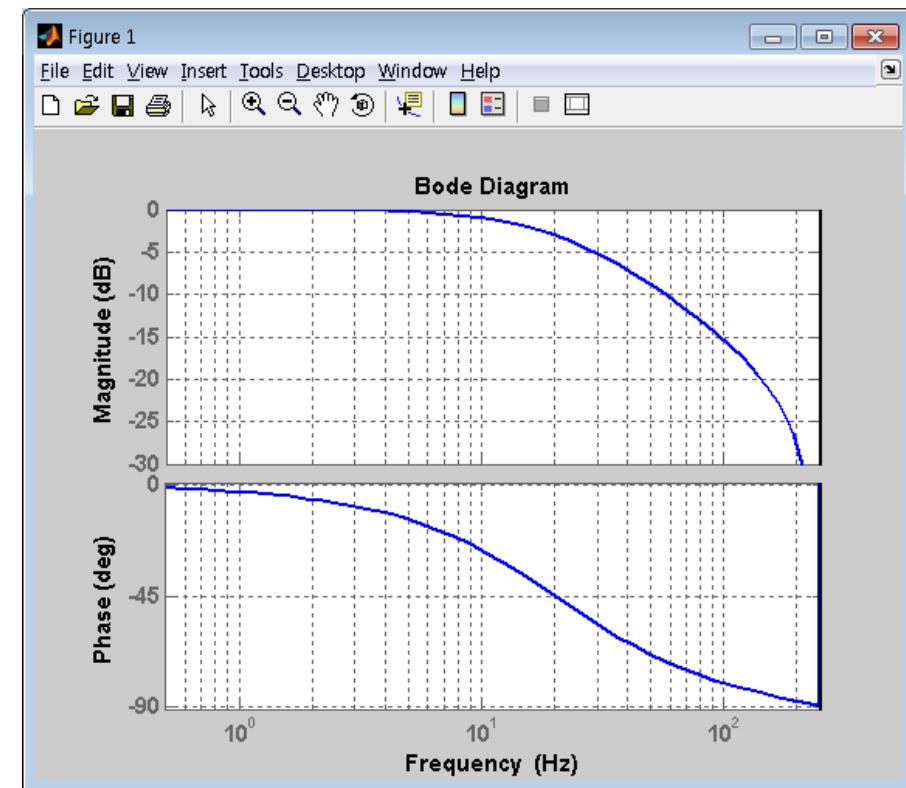
$$\begin{aligned} O_k &= \alpha O_{k-1} + 0.5(1-\alpha)(I_k + I_{k-1}) \\ &= \alpha O_{k-1} + \beta(I_k + I_{k-1}) \end{aligned}$$

$$where, \quad \alpha = \frac{1 - \tan(\pi T_s f_c)}{1 + \tan(\pi T_s f_c)}, \quad \beta = 0.5(1 - \alpha)$$

Digital Low Pass Filter (2)

- ❖ 신호처리(Signal Processing)과 피드백제어(Feedback Control)
 - 가장 큰 차이는?
- ❖ 2 msec. 샘플링 타임, 20Hz Cut-off Frequency의 전달함수
 - Sampling Time은 어떻게 정할까?
 - Nyquist Frequency는?
 - 주의할 점은?

```
Ts = 0.002;  
fc = 20;  
  
alpha = (1-tan(pi*Ts*fc))/(1+tan(pi*Ts*fc));  
beta = 0.5*(1-alpha);  
  
G_Lpf = tf(beta*[1 1], [1 -alpha], Ts);  
  
bode(G_Lpf)
```



32.FrunAdc – 개요

❖ 학습 목표

- 일정 시간 간격으로 A/D 변환을 할 수 있다.
 - ◆ Free Run ADC
- A/D 변환값에 1th Order Digital Low Pass Filter를 적용할 수 있다.

❖ 목적 및 문제

- 변환할 Mux. 채널 선택 기능
- Periodic Timer Interrupt1을 이용해 3 millisecond(=333.3Hz)마다 A/D 변환을 한다
- A/D 변환에 1th Digital Low Pass Filter를 적용한다
- LPF 적용 여부도 선택
- Update 여부
- Hook function 등록 기능

32.FrunAdc – 구상 (1)

❖ 변수들

- ADC 관련 레지스터
 - volatile Word fadc_en_channel=0xffff;
 - ◆ Analog-to-Digital 변환을 할 채널
 - volatile Word fadc_en_lpf=0xffff;
 - ◆ Low Pass Filter를 적용할 채널
 - volatile Word fadc_update_flag=FALSE;
 - ◆ Get 관련 함수 호출 후 Update가 되었는지 저장한 변수
 - volatile sWord fadc_in_k[FADC_CH_MAX] = {0,};
 - ◆ 현재 샘플에서의 ADC 변환값
 - volatile sWord fadc_in_km1[FADC_CH_MAX] = {0,};
 - ◆ 이전 샘플에서의 ADC 변환값
 - volatile sWord fadc_out_k[FADC_CH_MAX] = {0,};
 - ◆ 현재 샘플에서의 Low Pass Filter 적용 값으로 최종 값
 - volatile sWord fadc_out_km1[FADC_CH_MAX] = {0,};
 - ◆ 이전 샘플에서의 Low Pass Filter 적용 값
- void (*FADC_Func)(void) = NULL;
 - ◆ 이 변수가 NULL이 아니면 FADC_Func 함수를 호출
 - ◆ Free Run ADC에 맞춰 함수를 호출하는데 사용하기 위함

32.FrunAdc – 구상 (2)

❖ 함수들

- **void FADC_Init(Word channels, Word lpfs);**
 - ◆ 초기화 함수
 - ◆ channels은 ADC에 사용할 채널 정보. LSB부터 Channel 0에 해당
 - ◆ lpfs는 Low Pass Filter를 적용할 정보. LSB부터 Channel 0에 해당

- **void FADC_RegHookFunc(void (*fn)(void));**
 - ◆ AD 변환 후 호출할 함수를 등록하는 함수
 - ◆ void fn(void) 형태의 함수만 가능

- **Bool FADC_IsUpdated(void);**
 - ◆ FADC_GetAdv() FADC_GetAdvs() 후 AD 변환값이 업데이트 되었는지 유무를 리턴, TRUE/FALSE 리턴

- **Word FADC_GetAdv(Byte channel);**
 - ◆ channel에 해당하는 AD 변환값 리턴

- **void FADC_GetAdvs(Word* pCh);**
 - ◆ pCh 배열에 모든 AD 변환값을 담아오는 함수
 - ◆ FADC_CH_MAX, 즉 16개의 변환값을 리턴!

32.FrunAdc – 구현 (1)

• frunadc.h

```
#ifndef _FRUNADC_H_
#define _FRUNADC_H_

#ifndef __cplusplus
extern "C" {
#endif

#include "project.h"
#include "busfreq.h"

//----- MACRO -----
#define FADC_PRIORITY PIT1_PRIORITY

// Prescaler: f_ATDCLK = f_BUS/(2*(PRS+1)) :(0.25 ~ 8.3 MHz)
#if BFRQ_OPTION == BFRQ_2MHZ
    #define FADC_PRS 0 // 1MHz
#elif BFRQ_OPTION == BFRQ_20MHZ
    #define FADC_PRS 1 // 5MHz
#elif BFRQ_OPTION == BFRQ_40MHZ
    #define FADC_PRS 3 // 5MHz
#endif

#define FADC_VRH 5000 // mV
#define FADC_VRL 0 // mV
#define FADC_SPAN (FADC_VRH-FADC_VRL)
#define FADC_RES 12 // resolution (bit)

#define FADC_CH_MAX 16

#define FADC_CH00 0x0001
```

```
#define FADC_CH01 0x0002
#define FADC_CH02 0x0004
#define FADC_CH03 0x0008
#define FADC_CH04 0x0010
#define FADC_CH05 0x0020
#define FADC_CH06 0x0040
#define FADC_CH07 0x0080
#define FADC_CH08 0x0100
#define FADC_CH09 0x0200
#define FADC_CH10 0x0400
#define FADC_CH11 0x0800
#define FADC_CH12 0x1000
#define FADC_CH13 0x2000
#define FADC_CH14 0x4000
#define FADC_CH15 0x8000
#define FADC_CH_ALL 0xffff

#define FADC_SAMPLE_TIME 2000 // usec

//#define FADC_LPF_ALPHA0 16343 // fc = 0.2Hz
//#define FADC_LPF_BETA0 21
#define FADC_LPF_ALPHA0 16179 // fc = 1Hz
#define FADC_LPF_BETA0 102
#define FADC_LPF_ALPHA1 5899 // fc = 70Hz
#define FADC_LPF_BETA1 5243
#define FADC_LPF_ALPHA2 5899
#define FADC_LPF_BETA2 5243
#define FADC_LPF_ALPHA3 5899
#define FADC_LPF_BETA3 5243
#define FADC_LPF_ALPHA4 5899
#define FADC_LPF_BETA4 5243
```

- LPF alpha, beta 계수
- 목적에 맞게 변경

32.FrunAdc – 구현 (2)

• frunadc.h 계속

```
#define FADC_LPF_ALPHA5 5899
#define FADC_LPF_BETA5 5243
#define FADC_LPF_ALPHA6 5899
#define FADC_LPF_BETA6 5243
#define FADC_LPF_ALPHA7 5899
#define FADC_LPF_BETA7 5243
#define FADC_LPF_ALPHA8 5899
#define FADC_LPF_BETA8 5243
#define FADC_LPF_ALPHA9 5899
#define FADC_LPF_BETA9 5243
#define FADC_LPF_ALPHA10 5899
#define FADC_LPF_BETA10 5243
#define FADC_LPF_ALPHA11 5899
#define FADC_LPF_BETA11 5243
#define FADC_LPF_ALPHA12 5899
#define FADC_LPF_BETA12 5243
#define FADC_LPF_ALPHA13 5899
#define FADC_LPF_BETA13 5243
#define FADC_LPF_ALPHA14 5899
#define FADC_LPF_BETA14 5243
#define FADC_LPF_ALPHA15 5899
#define FADC_LPF_BETA15 5243

// FUNCTION -----
void FADC_Init(Word channels, Word lpfs);
void FADC_RegHookFunc(void (*fn)(void));
Bool FADC_IsUpdated(void);
Word FADC_GetAdv(Byte channel);
void FADC_GetAdvs(Word* pCh);
```

• frunadc.c 의 전역 변수

```
volatile Word fadc_en_channel=0xffff;
volatile Word fadc_en_lpf=0xffff;
volatile Word fadc_update_flag=FALSE;

volatile sWord fadc_in_k[FADC_CH_MAX] = {0,};
volatile sWord fadc_in_km1[FADC_CH_MAX] = {0,};
volatile sWord fadc_out_k[FADC_CH_MAX] = {0,};
volatile sWord fadc_out_km1[FADC_CH_MAX] = {0,};

void (*FADC_Func)(void) = NULL;

const sWord fadc_lpf_alpha[FADC_CH_MAX] =
{
    FADC_LPF_ALPHA0, FADC_LPF_ALPHA1, FADC_LPF_ALPHA2, FADC_LPF_ALPHA3,
    FADC_LPF_ALPHA4, FADC_LPF_ALPHA5, FADC_LPF_ALPHA6, FADC_LPF_ALPHA7,
    FADC_LPF_ALPHA8, FADC_LPF_ALPHA9, FADC_LPF_ALPHA10, FADC_LPF_ALPHA11,
    FADC_LPF_ALPHA12, FADC_LPF_ALPHA13, FADC_LPF_ALPHA14, FADC_LPF_ALPHA15
};

const sWord fadc_lpf_beta[FADC_CH_MAX] =
{
    FADC_LPF_BETA0, FADC_LPF_BETA1, FADC_LPF_BETA2, FADC_LPF_BETA3,
    FADC_LPF_BETA4, FADC_LPF_BETA5, FADC_LPF_BETA6, FADC_LPF_BETA7,
    FADC_LPF_BETA8, FADC_LPF_BETA9, FADC_LPF_BETA10, FADC_LPF_BETA11,
    FADC_LPF_BETA12, FADC_LPF_BETA13, FADC_LPF_BETA14, FADC_LPF_BETA15
};
```

32.FrunAdc – 구현 (3)

• frunadc.c 의 ISR

```
#pragma CODE_SEG __NEAR_SEG NON_BANKED
void FADC_ApplyLpf(Byte ch) {
    fadc_out_k[ch] = (sWord)((fadc_lpf_alpha[ch]*(sLWord)fadc_out_km1[ch]
        +fadc_lpf_beta[ch]*(sLWord)fadc_in_k[ch]+(sLWord)fadc_in_km1[ch])+8192)>>14;
    fadc_out_km1[ch] = fadc_out_k[ch];
    fadc_in_km1[ch] = fadc_in_k[ch];
}
interrupt VectorNumber_Vpit1 void FADC_PIT1_ISR(void) {
    Byte i;
    for (i=0; i<FADC_CH_MAX; i++) {
        if (fadc_en_channel&(1<<i)) {
            ATDOCTL5_Cx = i; // Start
            if (i>1) {
                if (fadc_en_lpf&(1<<(Byte)(i-1))) {
                    FADC_ApplyLpf(i-1);
                }
            }
            while(!ATD0STAT0_SCF) {
            }
            fadc_in_k[i] = ATD0DR0;
        }
    }
    if (fadc_en_lpf&0x1) {
        FADC_ApplyLpf(0);
    }
    fadc_update_flag = TRUE;
    if (FADC_Func!=NULL) FADC_Func();
    PITTF = PITTF_PTF1_MASK;
}
#pragma CODE_SEG DEFAULT
```

• Low Pass Filter

- i>1 인 경우에 왜 이렇게
이전 변환값에 Low Pass
Filter 적용을 했을까?

32.FrunAdc – 구현 (4)

• frunadc.c 의 초기화 함수

```
void FADC_Init(Word channels, Word lpfs) {  
  
    // 타이머 초기화  
    INT_CFADDR = 0x70; // Vpit1 0xff78 -> Vpit1&0x00f0 값  
    INT_CFDATA4 = FADC_PRIORITY; // INT_CFDATAAn에서 n은 Vpit1 (0xff78&0xf)/2  
  
    PITLD1 = (FADC_SAMPLE_TIME-1);  
    PITMUX_PMUX1 = 0;           // micro time base 0  
    PITCE_PCE1 = 1;             // Enable PIT Ch. n  
    PITTF = PITTF_PTF1_MASK;    // Clear Flag  
    PITINTE_PINTE1 = 1;         // Enable PIT Ch. n Interrupt  
  
    // ADC 초기화  
    ATDOCTL0 = 0x0fU;          // reset value  
    ATDOCTL1_ETRIGSEL = 0;     // No external trigger souce  
    ATDOCTL1_ETRIGCH = 0;      // external trigger channel (ignored)  
    ATDOCTL1_SRES = 2;         // 0:8bit, 1:10bit, 2:12bit  
    ATDOCTL1_SMP_DIS = 1;      // Discahrge before Sampling  
    ATDOCTL2 = ATDOCTL2_AFFC_MASK; // AFFC=1: 결과 읽기로 해당 CCF[n] clear  
    ATDOCTL3 = ATDOCTL3_DJM_MASK|ATDOCTL3_S1C_MASK; // Right justification  
                                //number of conversions per seq.=1  
    ATDOCTL3_FIF0 = 0;         // No FIFO mode  
    ATDOCTL3_FRZ = 2;          // Finish current conversion, then freeze  
    ATDOCTL4_SMP = 7;          // sample time select (24 clock)  
    ATDOCTL4_PRS = FADC_PRS;  // Prescale  
  
    fadc_en_channel = channels;  
    fadc_en_lpf = lpfs;  
}
```

• frunadc.c 의 기타 함수

```
void FADC_RegHookFunc(void (*fn)(void)) {  
    FADC_Func = fn;  
}  
  
Bool FADC_IsUpdated(void) {  
    return fadc_update_flag;  
}  
  
Word FADC_GetAdv(Byte channel) {  
    fadc_update_flag = FALSE;  
    return fadc_out_k[channel];  
}  
  
void FADC_GetAdvs(Word* pCh) {  
    Word i;  
  
    fadc_update_flag = FALSE;  
    for (i=0; i<FADC_CH_MAX; i++) {  
        pCh[i] = fadc_out_k[i];  
    }  
}
```

32.FrunAdc – 실습

- ❖ 실습: 컴파일하고 디버깅 해보기
 - 16채널의 AD 변환값 모두 출력해 보기
 - LPF의 영향을 살펴보기
- ❖ FrunAdc의 ISR 시간 측정하기
 - PIT를 이용해 ISR 시간을 측정하여 출력하기
 - f_ATDCLK 에 따른 ISR 시간 측정
 - 32.FrunAdc-IsrTime 과 비교
- ❖ 느낀 점 및 참고사항 적기

• frunadc.c 의 기타 함수

```
void PrintAdcVal(void) {  
    Word i, adv[FADC_CH_MAX];  
  
    FADC_GetAdvs(adv);  
  
    for (i=0; i<FADC_CH_MAX; i++) {  
        SDBG_Printf("%4d ", adv[i]);  
    }  
    SDBG_Printf("\n");  
}  
  
void main(void) {  
  
    InitPeripherals();  
  
    for(;;) {  
        if (SDBG_IsEvents()) {  
            SDBG_ExecuteCallback();  
        }  
        PrintAdcVal();  
        BTMR_DelayMs(125);  
    }  
}
```

GP2Y0A21YK 거리 측정 센서 (1)

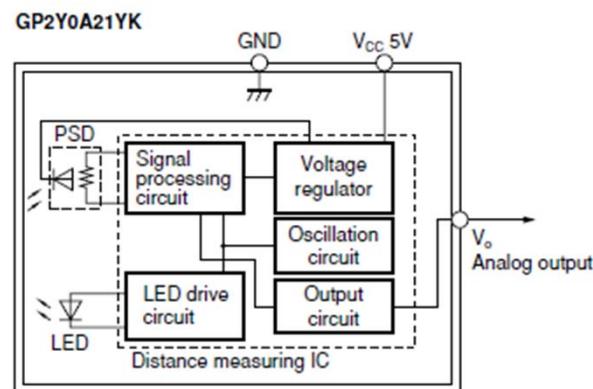
❖ 자료 출처: SHARP GP2Y0A21YK datasheet

❖ Features

- Less influence on the color of reflective objects, reflectivity
- Analog voltage output
- Detecting distance: 10 to 80 cm
- External control circuit is unnecessary
- Low cost

❖ Applications

- TVs
- Personal computers
- Cars
- Copiers

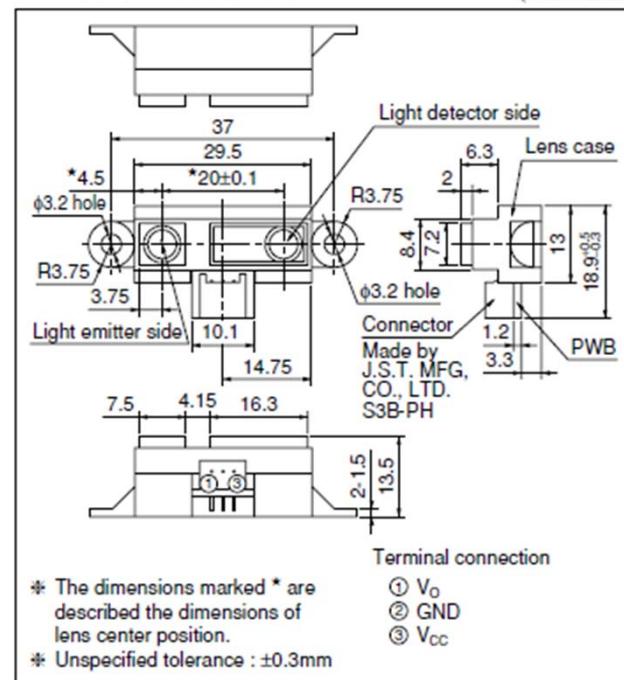


■ Absolute Maximum Ratings ($T_a=25^{\circ}\text{C}$, $V_{cc}=5\text{V}$)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{cc}	-0.3 to +7	V
Output terminal voltage	V_o	-0.3 to $V_{cc} + 0.3$	V
Operating temperature	T_{opr}	-10 to +60	$^{\circ}\text{C}$
Storage temperature	T_{stg}	-40 to +70	$^{\circ}\text{C}$

■ Outline Dimensions

(Unit : mm)



* The dimensions marked * are described the dimensions of lens center position.
* Unspecified tolerance : $\pm 0.3\text{mm}$

GP2Y0A21YK 거리 측정 센서 (2)

❖ 한번 해보기

- GP2Y0A21YK의 출력 전압을 mm 단위의 거리로 환산하는 함수 만들기

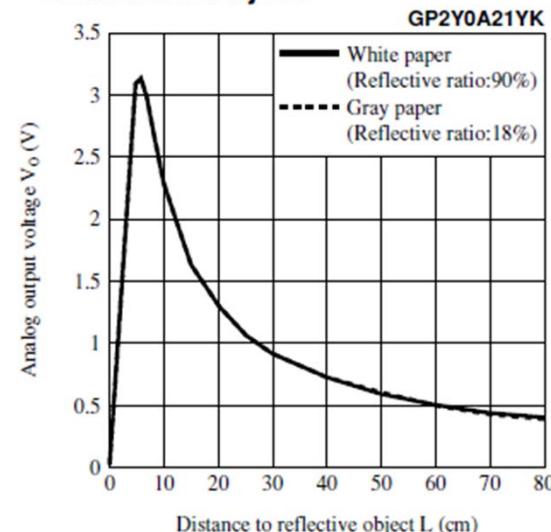
❖ GP2Y0A21YK에 대해

- Datasheet 확보하기
 - ◆ 전기적 특성 및 배선 자료 확보
 - ◆ 거리별 출력 전압 자료 확보

❖ 거리가 출력전압 특성

- 약 6cm보다 멀수록 출력전압이 낮아짐
- 약 6cm 이하에서는 가까울수록 출력전압 낮아짐
- 모형차에서 이 문제를 해결하는 방법은?

Fig.5 Analog Output Voltage vs. Distance to Reflective Object



❖ 약 6cm 이상에서는 멀수록 출력 전압이 낮아지나 선형이 아니다

- 해결 방법은?

- ◆ 이론 공부만 한 사람 → Curve Fitting에 적합한 식과 계수를 구하고, 이를 Firmware에 적용한다
- ◆ Firmware만 한 사람 → 그림으로부터 몇 개의 포인트에 대해 테이블을 만들고 Firmware에서는 이 테이블을 이용해 Linear Interpolation을 한다
- ◆ 둘 다 한 사람 → Curve Fitting을 하고 이 식으로부터 테이블을 만들고 Firmware에서는 Linear Interpolation을 한다
- ◆ 전문가는 → 정확한 거리 측정 목적이면 위와 같은 방법으로, 특정 거리 감지용이라면 그냥...

배터리 전압 및 GP2Y0A21YK 측정 실습

❖ Li-Po 배터리 사용시

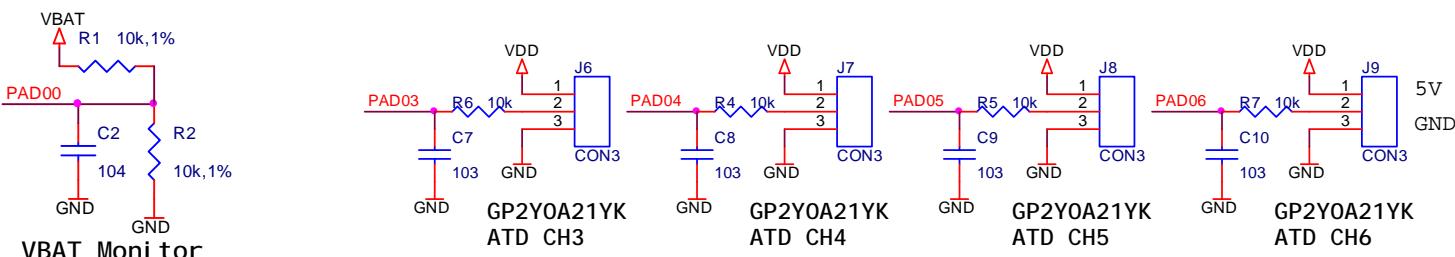
- Li-Po 배터리는 3.7V 셀 당 대략 2.4V 이하로는 사용하지 말 것을 권장
- 그러나 마진과 안전을 고려해서 3.0V 이하로는 절대 사용하지 말 것을 권장
- 따라서, 7.4V Li-Po 배터리는 약 6.0V 이하로는 사용하지 말 것을 권장
- 6.0V 이하로 계속 사용하면 어떻게 될까?

❖ 측정 실습

- Expansion Board for DEMO9S12XEP100의 VBAT 전압을 출력
- GP2Y0A21YK의 AD 변환값을 출력
- VBAT의 전압이 6V 이내면 모든 작동을 멈추고 LED 전부를 빠른 속도로 점멸

❖ 과제

- VBAT 측정을 위해 Digital Low Pass Filter의 cut-off frequency는 어느 정도로?
- 아래 R-C Low Pass Filter의 cut-off frequency는?



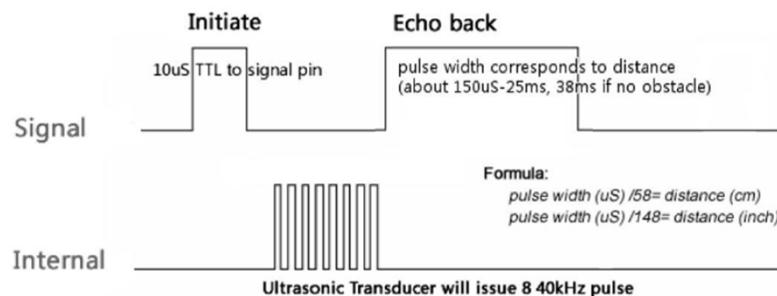
Ultrasonic Sensor Module

❖ 자료 출처

- Elec Freaks의 HC-SR04 Datasheet
- SRF-04 Datasheet

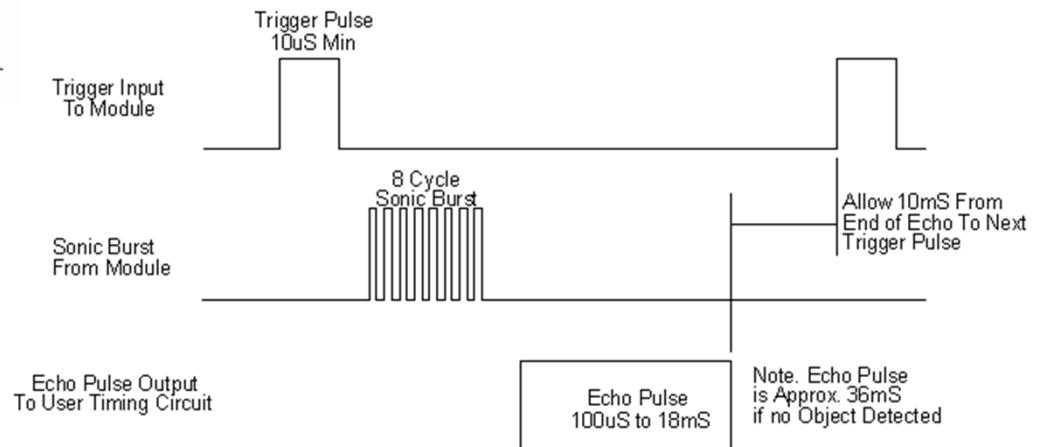
❖ HC-SR04 초음파 센서 모듈

- SRF-04과 동일한 사용법
- Range: 2cm ~ 4m
- Measuring Angle : 15 degree



• SRF04는 Echo Pulse의 길이가 최악의 경우 36msec 이지만, SRF04의 대치품인 HC-SR04는 190msec나 된다(Datasheet가 엉터리 였음. 테스트 후 알게 됨). 교육에는 HC-SR04를 사용하나, 이 부분이 문제가 될 수 있으므로 공통/자율 미션 과제수행에는 SRF04의 사용을 권장함. 예제도 SRF04의 36msec에 맞게 작성되어 있음!

SRF04 Timing Diagram



40.UsonicSenTest

❖ 문제

- Expansion Board의 J4에 연결되어 있는 Ultrasonic Sensor Module 사용
- 출력 펄스 시간을 Timer로 측정해서 거리로 환산한다

• 설정

```
void InitUSen(void) {  
    // 타이머 초기화  
    PITLD2 = 50000;  
    PITMUX_PMUX2 = 0;          // micro time base 0  
    PITCE_PCE2 = 1;            // Enable PIT Ch. n  
    PITTF = PITTF_PTF2_MASK;   // Clear Flag  
    // 포트 설정  
    DDRB_DDRB4 = 1;  
    PORTB_PB4 = 0;  
    DDRT_DDRT4 = 0;  
}  
  
void Delay10us(unsigned int us10) {  
    volatile int i;  
    while (us10--) {  
        #if BFRQ_OPTION == BFRQ_2MHZ  
            for (i=0; i<3; i++) {  
        #elif BFRQ_OPTION == BFRQ_20MHZ  
            for (i=0; i<33; i++) {  
        #elif BFRQ_OPTION == BFRQ_40MHZ  
            for (i=0; i<66; i++) {  
        #endif  
        }  
    }  
}
```

• 측정

```
PITFLT_PFLT2 = 1;  
  
PORTB_PB4 = 1;  
Delay10us(1);  
PORTB_PB4 = 0;  
  
for (cnt=0; cnt<5000; cnt++) {  
    if (PTIT_PTIT4==1) {  
        cnt_s = PITCNT2;  
        break;  
    }  
    Delay10us(1);  
}  
for (cnt=0; cnt<5000; cnt++) {  
    if (PTIT_PTIT4==0) {  
        cnt_e = PITCNT2;  
        break;  
    }  
    Delay10us(1);  
}  
cnt = cnt_s-cnt_e;  
  
SDBG_Printf("distance = %u mm\n", ((Word)(696*(LWord)cnt+2048))>>12);
```

• 왜 for 루프를?

Input Capture Timer

❖ MCU 외부 신호의 펄스 폭을 측정하는 기능

- 펄스 폭을 타이머로 측정하는 기능으로 보통 Input Capture Timer란 용어를 사용
- 펄스 Edge에서의 Timer 값을 Capture 하는 기능
- MC9S12XEP100에는 Reference Manual Chapter 14 Enhanced Capture Timer 참조
- 이 기능을 이용해서 초음파센서 모듈의 High Pulse 출력 시간을 정확히 측정할 수 있다!

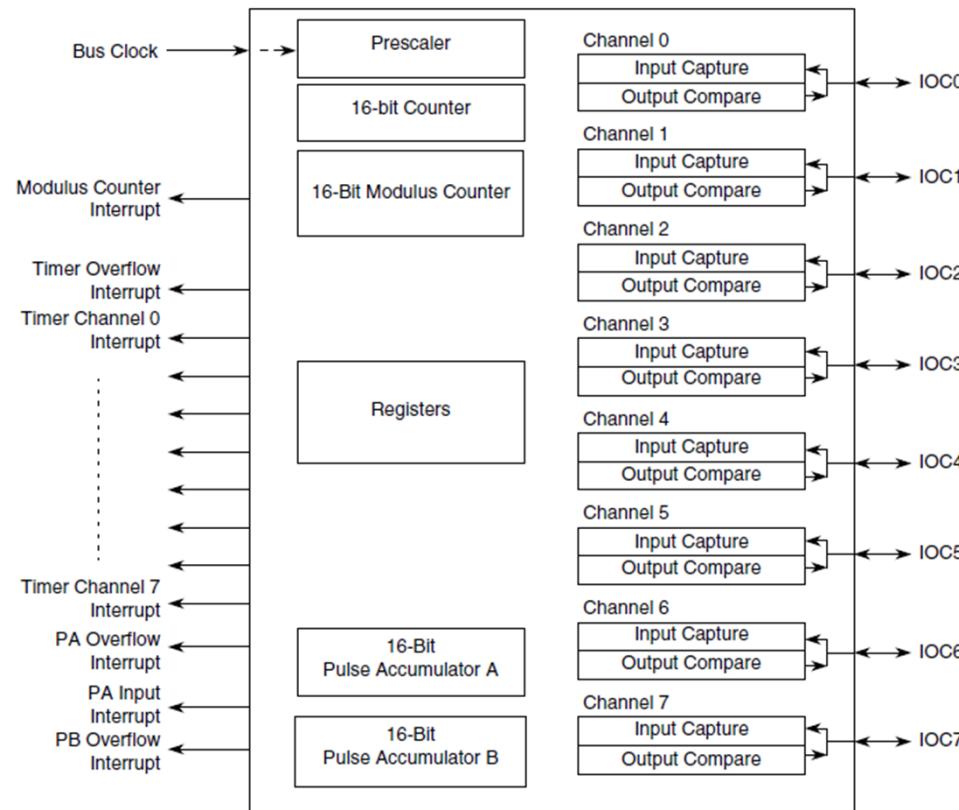


Figure 14-1. ECT Block Diagram

41.UsonicSen – 문제 정의 및 구상

❖ 문제

- HC-SR04 또는 SRF-04 초음파센서 모듈을 이용해서 거리를 측정 (총 4개까지)

❖ 변수

- Input Capture와 관련된 레지스터
- volatile Byte uss_flag; // 변환이 완료되었는지 여부
- volatile Byte uss_echo; // 현재 펄스 상승폭을 측정 중인지 여부
- volatile Word uss_tstart[4]; // rising edge에서의 시간 저장용
- volatile Word uss_tend[4]; // falling edge에서의 시간 저장용
- volatile LWord uss_t0; // start pulse 직전 runtime 시간 저장용

❖ 함수

- interrupt VectorNumber_Vectch4 void USS_ISR4(void); // ISR
-
- void USS_Init(void); // 초기화
- void USS_Start(Byte ch); // ch에 해당하는 초음파 센서 시작
- Bool USS_IsMeasured(Byte ch); // 측정 완료 여부 (TRUE/FALSE)
- Word USS_Get(Byte ch); // 결과 리턴 (단위는 TIMER count).
- Word USS_GetMm(Byte ch); // mm 단위로 결과 리턴

- 초기화하고
- 변환 시작한 후
- 변환이 끝남을 확인하고
- 변환값을 가져감

41.UsonicSen – 구현 (1)

• ultrasonic.h 중에서

```
//- MACRO -----
#define USS_PRIORITY 5

#if BFRQ_OPTION == BFRQ_2MHZ
#define USS_PR      1 // /2 -> max. 65.5 msec
#define USS_CNT2TIME 10 // 1 usec/cnt
#define USS_16US    16 // cnt
#define USS_CNT2MM_Q12 696
#elif BFRQ_OPTION == BFRQ_20MHZ
#define USS_PR      4 // /16 -> max. 54.4 msec
#define USS_CNT2TIME 8 // 0.8 usec/cnt
#define USS_16US    20 // cnt
#define USS_CNT2MM_Q12 557
#elif BFRQ_OPTION == BFRQ_40MHZ
#define USS_PR      5 // /32 -> max. 54.4 msec
#define USS_CNT2TIME 8 // 0.8 usec/cnt
#define USS_16US    20 // cnt
#define USS_CNT2MM_Q12 557
#endif

#define USS_TIMEOUT_MS 35 // timeout (msec)

#define USS_CH0 0
#define USS_CH1 1
#define USS_CH2 2
#define USS_CH3 3

#define USS_MAX USS_CH3
```

```
//- FUNCTION -----
void USS_Init(void);
void USS_Start(Byte ch);
Bool USS_IsMeasured(Byte ch);
Word USS_Get(Byte ch);
Word USS_GetMm(Byte ch);
```

41.UsonicSen – 구현 (2)

• ultrasonic.c 전역변수 및 ISR

```
#define USS_COMPLETE4 0x01
#define USS_COMPLETE5 0x02
#define USS_COMPLETE6 0x04
#define USS_COMPLETE7 0x08

#define USS_ECHO4 0x01
#define USS_ECHO5 0x02
#define USS_ECHO6 0x04
#define USS_ECHO7 0x08

volatile Byte uss_flag;
volatile Byte uss_echo;
volatile Word uss_tstart[4];
volatile Word uss_tend[4];
volatile LWord uss_t0;

#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt VectorNumber_Vectch4 void USS_ISR4(void) {
    if (uss_echo & USS_ECHO4) {
        uss_tend[0] = ECT_TC4;
        uss_flag |= USS_COMPLETE4;
    } else {
        uss_tstart[0] = ECT_TC4;
        uss_echo |= USS_ECHO4;
    }
    ECT_TFLG1_C4F = 1;
}
... 중간 생략

interrupt VectorNumber_Vectch5 void USS_ISR5(void) {
interrupt VectorNumber_Vectch7 void USS_ISR6(void) {
interrupt VectorNumber_Vectch7 void USS_ISR7(void) {
```

• ultrasonic.c 초기화

```
void USS_Init(void) {
    DDRB_DDRB4 = 1; DDRB_DDRB5 = 1; DDRB_DDRB6 = 1; DDRB_DDRB7 = 1;
    PORTB_PB4 = 0; PORTB_PB5 = 0; PORTB_PB6 = 0; PORTB_PB7 = 0;

    INT_CFADDR = 0xe0; // Vectch4~7: 0xffe6~0xffe0 -> Vectch&0xf0
    INT_CFDATA3 = USS_PRIORITY; // 0xffd6중 제일_오른쪽_값/2
    INT_CFDATA2 = USS_PRIORITY; // 0ffd4중 제일_오른쪽_값/2
    INT_CFDATA1 = USS_PRIORITY; // 0ffd2중 제일_오른쪽_값/2
    INT_CFDATA0 = USS_PRIORITY; // 0ffd0중 제일_오른쪽_값/2

    INT_CFADDR = 0xd0; // Vectovf: 0xffde -> Vectovf&0xf0
    INT_CFDATA7 = USS_PRIORITY; // 0xffde중 제일_오른쪽_값/2

    ECT_TSCR1 = 0; // ECT_TSCR1_TEN = 0;
    ECT_DLYCT = 0;
    ECT_DLYCT = 0;
    ECT_PACTL = 0;
    ECT_ICOVW = 0;

    ECT_TSCR2_TC4 = 0; // Disable Timer counter reset
    ECT_TSCR2_PR = USS_PR; // Prescaler

    ECT_TIOS = 0; // input capture

    ECT_TFLG1_C4F = 1; // reset interrupt request flag
    ECT_TFLG1_C5F = 1; // reset interrupt request flag
    ECT_TFLG1_C6F = 1; // reset interrupt request flag
    ECT_TFLG1_C7F = 1; // reset interrupt request flag

    ECT_TIE_C4I = 1; // enable interrupt
    ECT_TIE_C5I = 1; // enable interrupt
    ECT_TIE_C6I = 1; // enable interrupt
    ECT_TIE_C7I = 1; // enable interrupt

    ECT_TCTL3_EDG4x = 3; // capture on any edge
    ECT_TCTL3_EDG5x = 3; // capture on any edge
    ECT_TCTL3_EDG6x = 3; // capture on any edge
    ECT_TCTL3_EDG7x = 3; // capture on any edge

    ECT_TSCR1_TEN = 1;
}
```

41.UsonicSen – 구현 (3)

• ultrasonic.c 변환 시작

```
void USS_Start(Byte ch) {
    Word t0;
    if (ch==USS_CH0) {
        uss_tend[0] = 0xffff;
        uss_flag &= ~USS_COMPLETE4;
        uss_echo &= ~USS_ECHO4;
        PORTB_PB4 = 1;
    } else if (ch==USS_CH1) {
        uss_tend[1] = 0xffff;
        uss_flag &= ~USS_COMPLETE5;
        uss_echo &= ~USS_ECHO5;
        PORTB_PB5 = 1;
    } else if (ch==USS_CH2) {
        uss_tend[2] = 0xffff;
        uss_flag &= ~USS_COMPLETE6;
        uss_echo &= ~USS_ECHO6;
        PORTB_PB6 = 1;
    } else if (ch==USS_CH3) {
        uss_tend[3] = 0xffff;
        uss_flag &= ~USS_COMPLETE7;
        uss_echo &= ~USS_ECHO7;
        PORTB_PB7 = 1;
    }
    uss_t0 = BTMR_GetRuntime();
    t0 = ECT_TCNT;
    while ((ECT_TCNT-t0)<USS_16US) {
        PORTB_PB4 = 0;
        PORTB_PB5 = 0;
        PORTB_PB6 = 0;
        PORTB_PB7 = 0;
    }
}
```

• ultrasonic.c 변환 종료 여부

```
Bool USS_IsMeasured(Byte ch) {
    if ((BTMR_GetRuntime()-uss_t0)
        >=USS_TIMEOUT_MS) {
        uss_flag = USS_COMPLETE4
        |USS_COMPLETE5
        |USS_COMPLETE6|USS_COMPLETE7;
        return TRUE;
    }
    if (ch==USS_CH0) {
        if (uss_flag & USS_COMPLETE4) {
            return TRUE;
        }
    } else if (ch==USS_CH1) {
        if (uss_flag & USS_COMPLETE5) {
            return TRUE;
        }
    } else if (ch==USS_CH2) {
        if (uss_flag & USS_COMPLETE6) {
            return TRUE;
        }
    } else if (ch==USS_CH3) {
        if (uss_flag & USS_COMPLETE7) {
            return TRUE;
        }
    }
    return FALSE;
}
```

• ultrasonic.c 변환값 얻기

```
Word USS_Get(Byte ch) {
    if (ch>USS_MAX) {
        return 0xffff;
    }
    if (uss_tend[ch]==0xffff) {
        return 0xffff;
    }
    return (uss_tend[ch]-uss_tstart[ch]);
}
Word USS_GetMm(Byte ch) {
    LWord dist;
    if (ch>USS_MAX) {
        return 0xffff;
    }
    if (uss_tend[ch]==0xffff) {
        return 0xffff;
    }
    dist = (LWord)(uss_tend[ch]-uss_tstart[ch]);
    dist = (dist*USS_CNT2MM_Q12+2048)>>12;
    return (Word)dist;
}
```

41.UsonicSen – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- 'U'를 입력하면 30초간 초음파센서 한 개의 측정 결과가 count단위와 mm 단위로 출력됨

❖ 질문

- 이 예제는 한번에 하나의 초음파 센서만 측정하게 되어 있다. 만약, 동시에 여러 개의 초음파 센서를 사용한다면 무슨 문제가 있을까?

❖ 느낀 점 및 참고사항 적기

42.FrunUsonicSen

❖ 학습 목표

- 일정 시간 간격으로 초음파 센서 모듈로 거리를 측정할 수 있다.

❖ 목적 및 문제

- 변환 채널 선택 기능
 - ◆ 선택된 채널에 대해 동시에 초음파 센서 모듈에 Trigger Pulse 출력
- Periodic Timer Interrupt2을 이용해 50 millisecond마다 초음파 센서 모듈에 Trigger Pulse를 출력
- Enhanced Capture Timer를 이용해서 초음파 센서 모듈의 Echo 출력의 상승폭 시간 측정
- 측정값은 mm 단위로 리턴

Rotary Encoder

❖ Rotary Encoder란?

- 회전각 검출 센서

❖ Rotary Encoder E30 Series

- 자료 출처: Autonics E30 Series Datasheet

▣ 특징

- 외경 ϕ 30mm의 초소형 축형 로타리 엔코더
- 좁은 공간에 설치가 편리하다.
- 축의 관성 모멘트가 적다.
- 전원전압 : 5VDC, 12~24VDC \pm 5%
- 다양한 출력방식



사용하시기 전에 취급설명서에 있는 "안전을 위한 주의사항"을 반드시 읽고 사용하시기 바랍니다.



▣ 모델구성

E30S	4	—	3000	—	3	—	N	—	24	—	
시리즈명	축 외경	회전당 Pulse 수	출력상	제어 출력	전원전압	배선 사양					
외경 ϕ 30mm 축형	ϕ 4mm	분해능 참조	3 : A, B, Z 6 : A, \overline{A} , B, \overline{B} , Z, \overline{Z}	T : Totem Pole 출력 N : NPN 오픈콜렉터 출력 V : 전압 출력 L : Line Driver 출력(※)	5 : 5VDC \pm 5% 24: 12~24VDC \pm 5%	무표시: 일반형 (※) C: 배선인출 커넥터형					

* 표준품: E30S4-PULSE-3-N-24

* 표준품: A, B, Z

* Line Driver의 전원은
5VDC 전용입니다.

* 배선 길이: 250mm

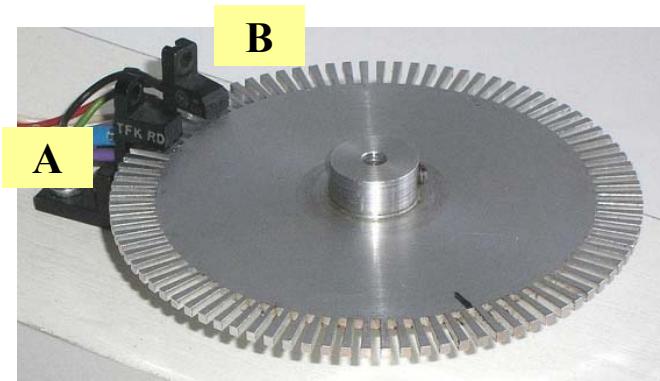
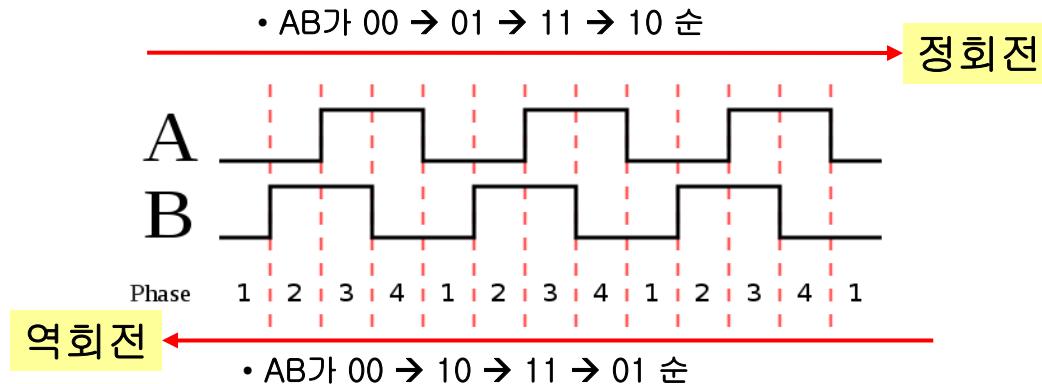
❖ 우리가 사용하는 Rotary Encoder는

- E30S4-PULSE-3-N-5

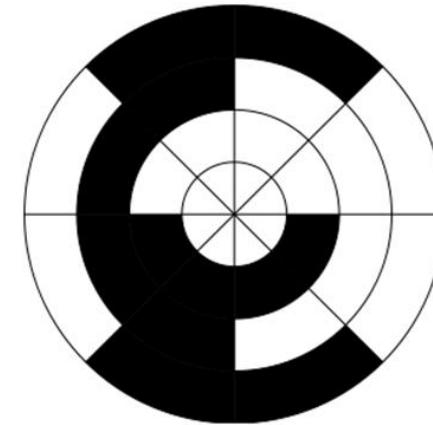
Rotary Encoder 원리

❖ 과제: Wikipedia(English)에서 Rotary Encoder 학습하기

- 제공하는 로타리 엔코더는 Incremental Type이나 Absolute Type도 학습해보세요!
- 체배에 대해서 반드시 학습할 것!
- 출처: http://en.wikipedia.org/wiki/Rotary_encoder



<http://commons.wikimedia.org/wiki/File:Encoder.jpg>



Gray Coding				
Sector	Contact 1	Contact 2	Contact 3	Angle
0	off	off	off	0° to 45°
1	off	off	ON	45° to 90°
2	off	ON	ON	90° to 135°
3	off	ON	off	135° to 180°
4	ON	ON	off	180° to 225°
5	ON	ON	ON	225° to 270°
6	ON	off	ON	270° to 315°
7	ON	off	off	315° to 360°

Rotary Encoder – Counting

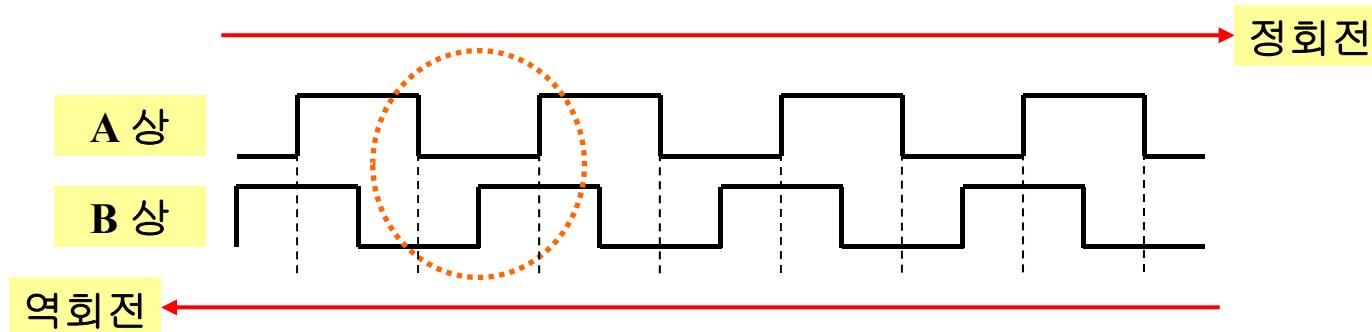
❖ Incremental Type의 경우!

❖ A의 Edge를 기준으로 한다면

- A Edge가 상승이고, B가 High면 정방향 반 펄스 이동
- A Edge가 하강이고, B가 Low면 정방향 반 펄스 이동
- A Edge가 상승이고, B가 Low면 역방향 반 펄스 이동
- A Edge가 하강이고, B가 High면 역방향 반 펄스 이동

※ 원래 엔코더 분해능의
두 배 가능 (2제배)

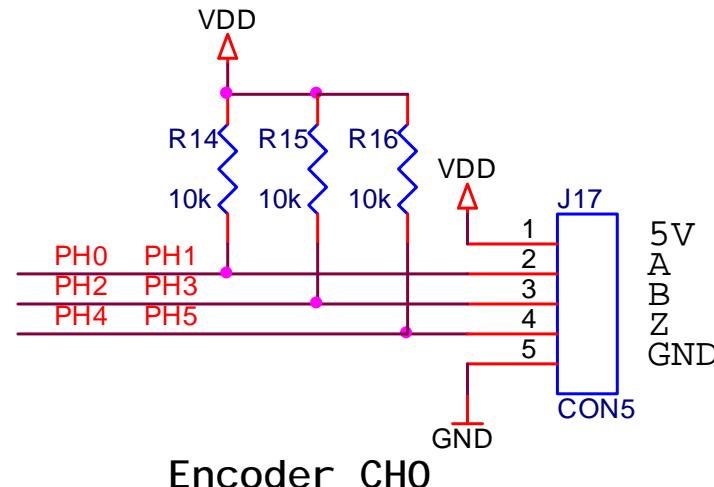
※ B상도 이용하면 원래
엔코더 분해능의 네 배 가능
(4제배)



50.RotaryEncoderTest – 개요

❖ 문제 및 실습

- Rotary Encoder의 A, B 상이 Expansion Board의 J17을 통해 DEMO9S12XEP100의 PH0과 PH2에 연결되어 있다.
- Serial로 ‘E’를 누르면 다음을 60초 동안 다음을 실행
 - ◆ 시작 시 회전각을 0으로
 - ◆ PH0과 PH2의 레벨 변화로 회전각을 측정하고, 그 결과를 200 msec 마다 출력
- 오실로스코프로 A, B 측정하기



50.RotaryEncoderTest – 구현

• ShowRotEncVal 함수

```
void ShowRotEncVal(void) {  
    sWord cnt = 0;  
    Byte a_new, b_new, a_old=0, b_old=0;  
    LWord t_new, t_old=-1;  
  
    DDRH_DDRH0 = 0;  
    DDRH_DDRH2 = 0;  
  
    BTMR_SetTimer0(60000UL);  
    while (BTMR_GetTimer0()>0) {  
        a_new = PTIH_PTIH0 ? 1 : 0;  
        b_new = PTIH_PTIH2 ? 1 : 0;  
  
        if ((a_old==0) && (a_new==1)) {  
            if (b_new==1) {  
                cnt++;  
            } else {  
                cnt--;  
            }  
        } else if ((a_old==1) && (a_new==0)) {  
            if (b_new==1) {  
                cnt--;  
            } else {  
                cnt++;  
            }  
        }  
    }  
}
```

```
/*  
 * if ((b_old==0) && (b_new==1)) {  
 *     if (a_new==1) {  
 *         cnt--;  
 *     } else {  
 *         cnt++;  
 *     }  
 * } else if ((b_old==1) && (b_new==0)) {  
 *     if (a_new==1) {  
 *         cnt++;  
 *     } else {  
 *         cnt--;  
 *     }  
 * }  
 */  
a_old = a_new;  
b_old = b_new;  
  
t_new = BTMR_GetRuntime()/200;  
if (t_new != t_old) {  
    t_old = t_new;  
    SDBG_Printf("%d\n", cnt);  
}  
}
```

- 주석문을 해제하면 4체배

51.RotaryEncoder – 문제 정의, 구상

❖ 문제

- Incremental Type의 Encoder의 회전각 검출

❖ 필요 변수

- External Interrupt 관련 레지스터
- volatile sLWord renc_val;

❖ 함수들

- void RENC_Init(void);
 - ◆ 초기화
- void RENC_Set(sLWord val);
 - ◆ val 값으로 Encoder Counting 값 설정
- sLWord RENC_Get(void);
 - ◆ Encoder Counting 값 리턴

• rotaryencoder.h 중에서

```
#define RENC_PRIORITY PORTH_PRIORITY  
  
#define RENC_2X      0  
#define RENC_4X      1  
#define RENC_MODE    RENC_4X // 체배 선택, 2X, 4X만 가능  
  
#define RENC_PPR_1X 1000 // Pulse Per Round  
  
#if RENC_MODE == RENC_2X  
#define RENC_PPR  (RENC_PPR_1X*2)  
#elif RENC_MODE == RENC_4X  
#define RENC_PPR  (RENC_PPR_1X*4)  
#endif
```

51.RotaryEncoder – 구현 (1)

- rotaryencoder.c의 전역 변수 및 ISR

```
volatile sLWord renc_val;

#pragma CODE_SEG __NEAR_SEG NON_BANKED
// PH0~PH3의 external interrupt ISR
// PH0: A phase rising edge
// PH1: A phase falling edge
// PH2: B phase rising edge
// PH3: B phase falling edge
interrupt VectorNumber_Vporth void RENC_ISR(void) {
    if (PIFH_PIFH0 == 1) {
        if (PTIH_PTIH2 == 1) {
            renc_val++;
        } else {
            renc_val--;
        }
        PIFH_PIFH0 = 1;
    } else if (PIFH_PIFH1 == 1) {
        if (PTIH_PTIH2 == 1) {
            renc_val--;
        } else {
            renc_val++;
        }
        PIFH_PIFH1 = 1;
    }
}
```

```
#if RENC_MODE == RENC_4X
else if (PIFH_PIFH2 == 1) {
    if (PTIH_PTIH0 == 1) {
        renc_val--;
    } else {
        renc_val++;
    }
    PIFH_PIFH2 = 1;
} else if (PIFH_PIFH3 == 1) {
    if (PTIH_PTIH0 == 1) {
        renc_val++;
    } else {
        renc_val--;
    }
    PIFH_PIFH3 = 1;
}
#endif
}
```

- MC9S12XEP100은 같은 포트에 있는 External Interrupt는 하나의 ISR을 공유한다.

51.RotaryEncoder – 구현 (2)

• rotaryencoder.c의 초기화

```
void RENC_Init(void) {
    INT_CFAADDR = 0xc0; // Vectovf: 0xffcc -> Vectovf&0xf0
    INT_CFDATA6 = RENC_PRIORITY; // 0xffcc중 제일_오른쪽_값/2

    DDRH_DDRH0 = 0; // input for A phase
    DDRH_DDRH1 = 0; // input for A phase
    DDRH_DDRH2 = 0; // input for B phase
    DDRH_DDRH3 = 0; // input for B phase

    PPSH_PPSH0 = 1; // rising edge for A phase
    PPSH_PPSH1 = 0; // falling edge for A phase
    PPSH_PPSH2 = 1; // rising edge for B phase
    PPSH_PPSH3 = 0; // falling edge for B phase

    renc_val = 0;

    PIEH_PIEH0 = 1;
    PIEH_PIEH1 = 1;
#if RENC_MODE == RENC_4X
    PIEH_PIEH2 = 1;
    PIEH_PIEH3 = 1;
#endif
}
```

• rotaryencoder.c의 초기값 및 회전각 얻기

```
// val 값으로 encoder 회전각(단위: pulse)를 설정
void RENC_Set(sLWord val) {
    ReadyCriticalSection();
    EnterCriticalSection();
    renc_val = val;
    LeaveCriticalSection();
}

// Pulse 단위의 Encoder 회전각을 리턴
sLWord RENC_Get(void) {
    sLWord val;
    ReadyCriticalSection();
    EnterCriticalSection();
    val = renc_val;
    LeaveCriticalSection();

    return val;
}
```

51.RotaryEncoder – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- 약 0.1 초 간격으로 계속해서 Encoder 값이 출력된다. 'S'를 누르면 0으로 초기화

❖ 고찰

- 예제는 2체배와 4체배를 지원한다 1체배는 안 되는 것일까?
- 측정에 오차가 발생한다면 어떤 경우에?

❖ 참고

- Encoder용 카운터 전용 칩도 있음(<http://www.lsicsi.com/counters.htm> 참조)
- Encoder용 카운터 내장 MCU도 있음
- 이 예제의 방법은 External Interrupt를 이용하나 간단한 회로를 추가하고, 외부 펄스 신호 카운팅 Timer/Counter 두 개(와 Periodic Interrupt Timer 1개)가 있으면 Firmware의 Load가 거의 없이 Encoder 회전각 측정 가능

❖ 느낀 점 및 참고사항 적기

52.RotaryEncoder – 알파 점수 부여

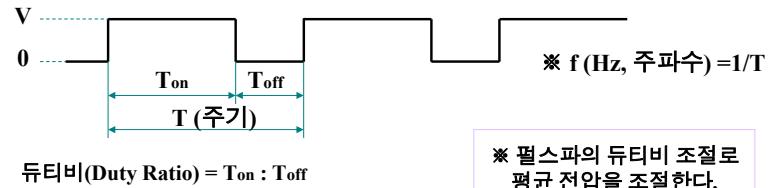
- ❖ 51.RotaryEncoder에 소개한 방법은 문제가 있다
 - Encoder의 회전 속도가 빠른 경우
 - ISR에 의한 처리가 빠른 속도에 의한 Pulse Detecting을 못 따라갈 수 있다.
 - 이러한 경우에는 Pulse Counting을 놓치게 되어 Encoder로부터 정확한 회전각을 검출할 수 없게 된다.

- ❖ 알파 점수 부여
 - DEMO9S12XEP100로 충분히 빠른 속도의 Encoder Pulse도 정확히 감지하여 회전각을 측정할 수 있는 F/W 작성하기
 - ◆ 사용 포트를 변경해도 됨!
 - ◆ 단, 회로의 추가는 저항과 커패시터만 가능!

PWM: Pulse Width Modulation

❖ PWM

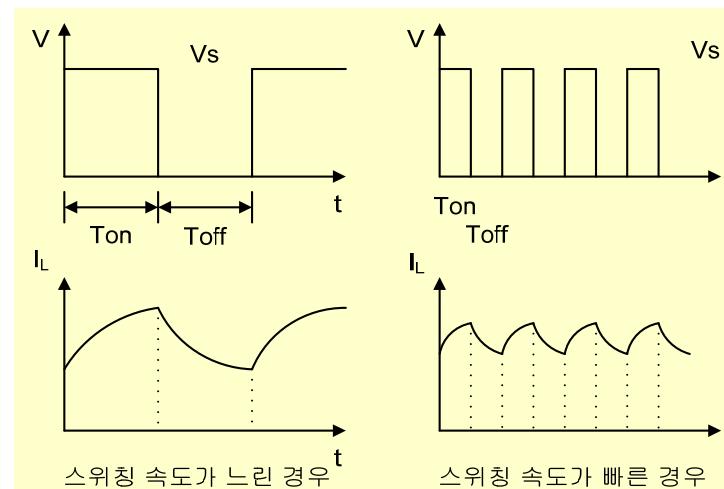
- 주기는 일정
- 펄스 Duty를 변경
- 평균 전압을 변경할 수 있다! 평균 전압 계산식은?



$$\text{평균 전압} = \frac{T_{on}}{T_{on} + T_{off}} V = \frac{T_{on}}{T} V$$

❖ 모터 또는 Inductance를 갖는 회로에 PWM 전압을 가하면?

- PWM 파형은 사각파라도 전류는?
- PWM 기본 주파수가 너무 낮으면? 너무 높으면?
- 참고: 실제 정밀 모터 제어에서는 Shunt Resistor로 Current를 Feedback해서 전류를 목표 전류로 제어
 - ◆ 왜 전류를 제어할까?



MC9S12XEP100의 PWM

❖ MC9S12XEP100의 PWM 참고 자료

■ Reference Manual Chapter 19 Pulse-Width Modulator

■ Features

- ◆ Eight independent PWM channels with programmable period and duty cycle
- ◆ Dedicated counter for each PWM channel
- ◆ Programmable PWM enable/disable for each channel
- ◆ Software selection of PWM duty pulse polarity for each channel
- ◆ Period and duty cycle are double buffered. Change takes effect when the end of the effective period is reached (PWM counter reaches zero) or when the channel is disabled.
- ◆ Programmable center or left aligned outputs on individual channels
- ◆ Eight 8-bit channel or four 16-bit channel PWM resolution
- ◆ Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
- ◆ Programmable clock select logic
- ◆ Emergency shutdown

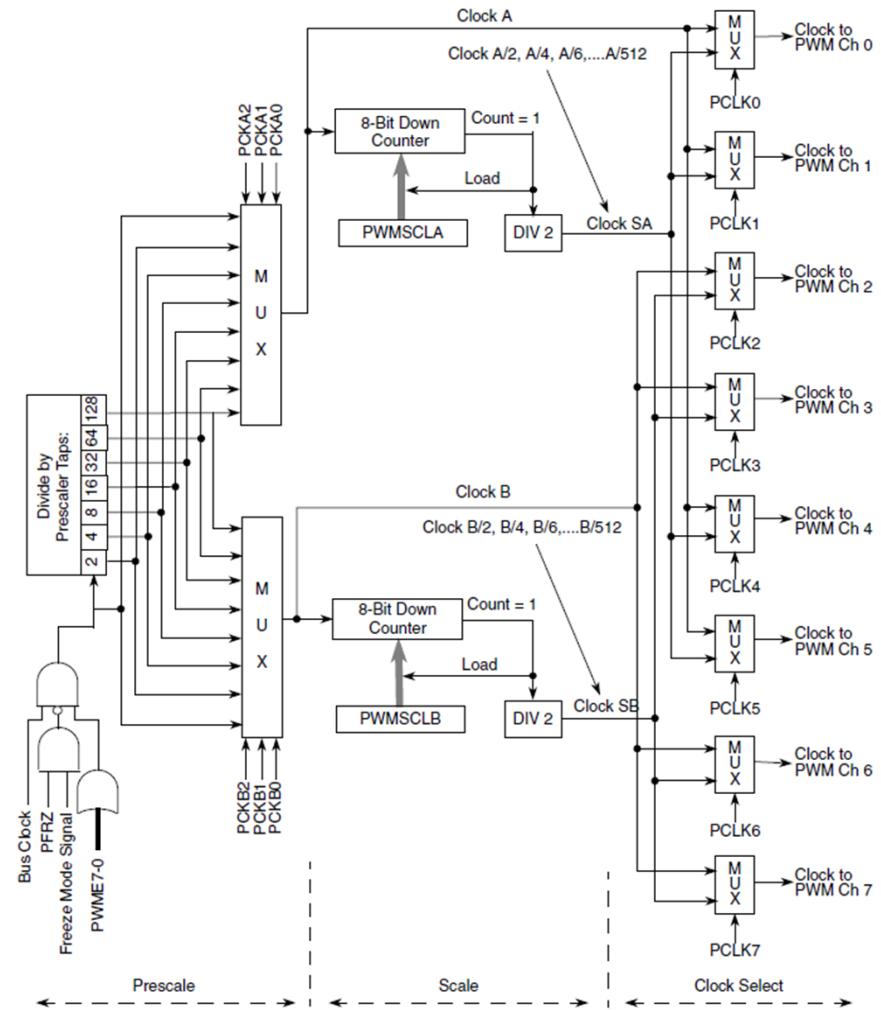


Figure 19-18. PWM Clock Select Block Diagram

60.Pwm – 개요

❖ 학습 목표

- PP3, PP5, PP7핀을 통해 PWM 파형을 출력할 수 있다.

❖ 문제 정의

■ PP5와 PP7의 PWM 출력

- ◆ 0~1023의 범위가 PWM 상승 펄스 0~100%가 되도록 한다.
- ◆ PWM 주기는 8kHz 이상이 되도록 한다.

■ PP3의 PWM 출력

- ◆ -600~600의 범위가 상승폭 0.9~2.1 millisecond가 되도록 한다.
- ◆ PWM 주기는 50Hz 정도가 되게 한다.
- ◆ 전원 투입 직후 상승폭이 1.5 millisecond가 되게 한다.

❖ 참고

- MC9S12XEP100의 PWM은 8bit 8개, 16bit 4개 가능
- 16bit로 사용할 경우에는 PP0과 PP1을 하나로 합쳐 PP1에 출력
 - ◆ 마찬가지로 PP2, PP3은 하나로 합쳐 PP3, PP4와 PP5는 PP5, PP6과 PP7은 PP7
- Clock은 A와 A를 분주한 SA, B와 B를 분주한 SB가 있으며,
 - ◆ PP0, PP1, PP4, PP5는 A와 SA만 가능
 - ◆ PP2, PP3, PP6, PP7은 B와 SB만 가능

60.Pwm – 구상

❖ 변수

■ PWM 관련 레지스터

❖ 함수

■ void PWM_Init(void);

◆ PWM 초기화 함수

■ void PWM_Set(Byte channel, sWord val);

◆ channel은 PWM_CH3, PWM_CH5, PWM_CH7
중 하나

◆ 각각 PP3, PP5, PP7에 해당

• pwm.h

```
-- MACRO --
#define PWM_CH3 3
#define PWM_CH5 5
#define PWM_CH7 7

#define PWM_PCKA 0
#define PWM_PCKB 0
```

```
#if BFRQ_OPTION == BFRQ_2MHZ
#define PWM_SCLA 1
#define PWM_SCLB 1 // 1MHz
#define PWM_DIGIT 7
#define PWM_SHIFT45 3
#define PWM_SHIFT67 3
#define PWM_PERIOD45 127 // f_PER: 7.8 kHz
#define PWM_PERIOD67 127 // f_PER: 7.8 kHz
#elif BFRQ_OPTION == BFRQ_20MHZ
#define PWM_SCLA 1 // 10MHz
#define PWM_SCLB 10 // 1MHz
#define PWM_DIGIT 10
#define PWM_SHIFT67 1
#define PWM_PERIOD45 1023 // 9.8 kHz
#define PWM_PERIOD67 2048 // 9.8 kHz
#elif BFRQ_OPTION == BFRQ_40MHZ
#define PWM_SCLA 2 // 10MHz
#define PWM_SCLB 20 // 1MHz
#define PWM_DIGIT 10
#define PWM_SHIFT67 2
#define PWM_PERIOD45 1023 // 9.8 kHz
#define PWM_PERIOD67 4096 // 9.8 kHz
#endif
#define PWM_PERIOD23 (20000-1) // f_PER: 50Hz

#define PWM_OP9MS_DTY 900 // 0.9 msec에 해당하는 DTY
#define PWM_1P5MS_DTY 1500 // 1.5 msec에 해당하는 DTY
#define PWM_2P1MS_DTY 2100 // 2.1 msec에 해당하는 DTY

#define PWM_MAXVAL23 (600) // PWM_2P1MS_DTY-PWM_1P5MS_DTY
#define PWM_MINVAL23 (-600) // PWM_OP9MS_DTY-PWM_1P5MS_DTY
#define PWM_MAXVAL45 (1023) // 2^10-1
#define PWM_MINVAL45 (0)
#define PWM_MAXVAL67 (1023) // 2^10-1
#define PWM_MINVAL67 (0)

-- FUNCTION --
void PWM_Init(void);
void PWM_Set(sByte channel, sWord val);
```

60.Pwm – 구현

• pwm.c

```
void PWM_Init(void) {
    // disable PWM
    PWME_PWME2 = 0;
    PWME_PWME3 = 0;
    PWME_PWME4 = 0;
    PWME_PWME5 = 0;
    PWME_PWME6 = 0;
    PWME_PWME7 = 0;

    PWMCTL_CON23 = 1; // Combine PWM2 & PWM3 (clock a or clock sa)
    PWMCTL_CON45 = 1; // Combine PWM4 & PWM5 (clock a or clock sa)
    PWMCTL_CON67 = 1; // Combine PWM6 & PWM7 (clock a or clock sa)
    PWMCTL_PFRZ = 0; // Allow PWM in Wait Mode
    PWMCTL_PSWAI = 0; // Go PWM Counter in Freeze Mode

    PWMCCE_CAE3 = 0; // Left Align
    PWMPCOL_PPOL3 = 1; // High output at the beginning
    PWMCCE_CAE5 = 0; // Left Align
    PWMPCOL_PPOL5 = 1; // High output at the beginning
    PWMCCE_CAE6 = 0; // Left Align
    PWMPCOL_PPOL7 = 1; // High output at the beginning
    PWMCNT23 = 0;
    PWMCNT45 = 0;
    PWMCNT67 = 0;

    PWMSDN = PWMSDN_PWMIF_MASK; // reset interrupt flag and...
    PWMDTY23 = PWM_1P5MS_DTY; // initial duty
    PWMPER23 = PWM_PERIOD23;
    PWMDTY45 = 0; // initial duty
    PWMPER45 = PWM_PERIOD45;
    PWMDTY67 = 0; // initial duty
    PWMPER67 = PWM_PERIOD67;

    // Init. Clock
    PWMPRCLK_PCKB = PWM_PCKB;
    PWMSCLB = PWM_SCLB;
    PWMCLK_PCLK3 = 1; // Clock SB
    PWMPRCLK_PCKA = PWM_PCKA;
    PWMSCLA = PWM_SCLA;
    PWMCLK_PCLK5 = 1; // Clock SA
    PWMCLK_PCLK7 = 0; // Clock B
```

```
PWME_PWME3 = 1;
PWME_PWME5 = 1;
PWME_PWME7 = 1;
}

void PWM_Set(sByte channel, sWord val) {
    switch (channel) {
        case PWM_CH3:
            if (val>PWM_MAXVAL23) {
                val = PWM_MAXVAL23;
            } else if (val<PWM_MINVAL23) {
                val = PWM_MINVAL23;
            }
            PWMDTY23 = PWM_1P5MS_DTY+val;
            break;

        case PWM_CH5:
            if (val>PWM_MAXVAL45) {
                val = PWM_MAXVAL45;
            } else if (val<PWM_MINVAL45) {
                val = PWM_MINVAL45;
            }
#ifndef BFRQ_OPTION == BFRQ_2MHZ
            val >>= PWM_SHIFT45;
#endif
            PWMDTY45 = val;
            break;

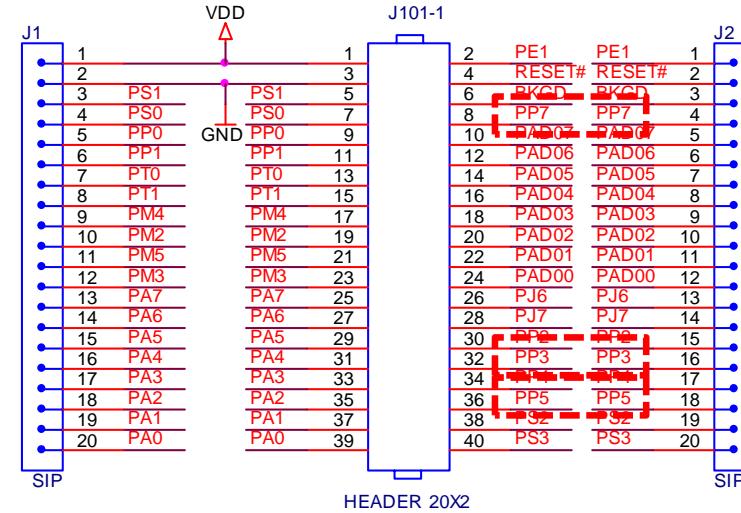
        case PWM_CH7:
            if (val>PWM_MAXVAL67) {
                val = PWM_MAXVAL67;
            } else if (val<PWM_MINVAL67) {
                val = PWM_MINVAL67;
            }
#ifndef BFRQ_OPTION == BFRQ_2MHZ
            val >>= PWM_SHIFT67;
#else
            val <<= PWM_SHIFT67;
#endif
            PWMDTY67 = val;
            break;
    }
}
```

60.Pwm – 실습

❖ 실습

- SDBG 명령어로 PWM 값을 변경하고,
오실로스코프로 PP3, PP5, PP7의 파형 보기

```
void SetPwm(char* str) {  
    sWord channel, val;  
  
    if (sscanf(str, "%d %d", &channel, &val)<2) {  
        SDBG_Printf("USAGE:p 3/5/7 pwm_val\n");  
        return;  
    }  
    SDBG_Printf("Ch.=%d, PWM=%d\n", channel, val);  
    PWM_Set((Byte)channel, val);  
}
```



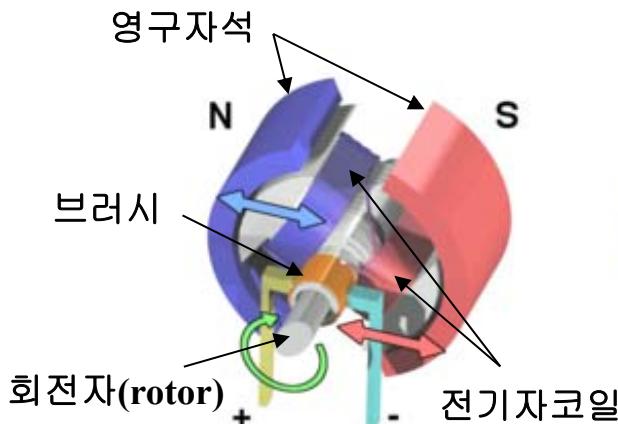
❖ 고찰

- 왜 PP3과 같은 PWM 파형을 만들었을까?
- PWMDTYn의 값을 변경하면 바로 적용될까? 그렇지 않다면 언제?

❖ 느낀 점 및 참고사항 적기

Brushed DC 모터

- ❖ 코일에 전압을 가하면 브러시에 의해 회전자가 회전



http://en.wikipedia.org/wiki/File:Electric_motor_cycle_1.png

http://en.wikipedia.org/wiki/File:Electric_motor_cycle_2.png

http://en.wikipedia.org/wiki/File:Electric_motor_cycle_3.png

http://en.wikipedia.org/wiki/File:Electric_motor.gif



Brushless DC 모터 (BLDC)

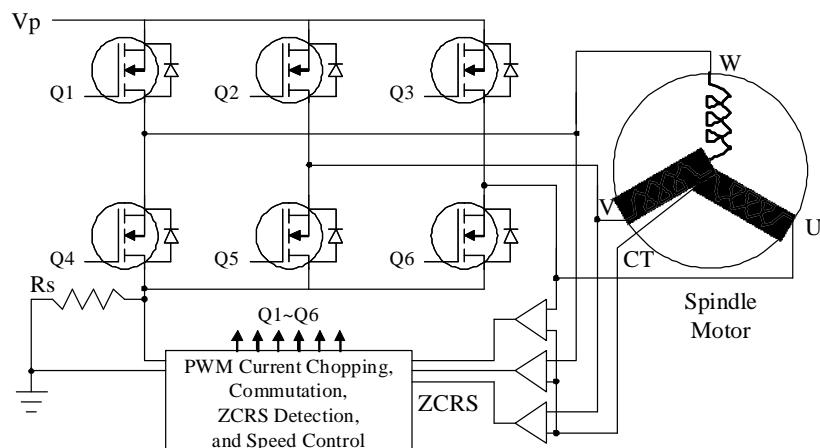
- ❖ 고정자: 코일, 회전자: 영구자석
- ❖ 코일 전류 방향 전환을 전자회로로
■ 정류(commutation)



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

기계 브러시 불필요
(전자 브러시로 대체)

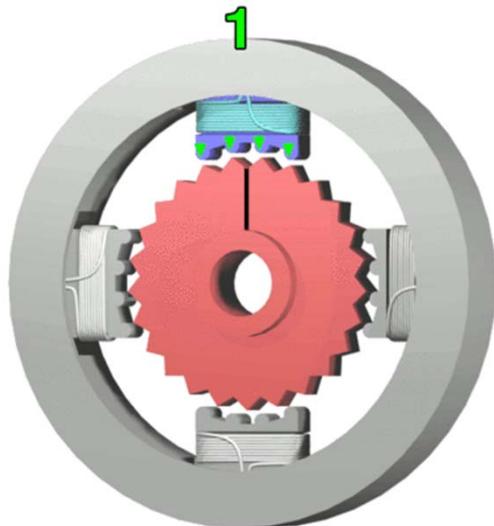
※ 동기(synchronous) 모터
→ 회전자 위치 검출 필요 (홀 센서)



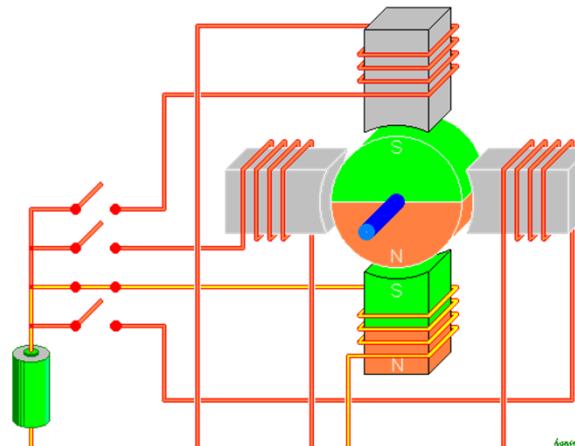
스텝 모터

❖ Brushless 모터의 일종

- 일정 각도(보통 1.8도) 간격으로 회전 가능



http://en.wikipedia.org/wiki/File:Stepper_Motor.gif



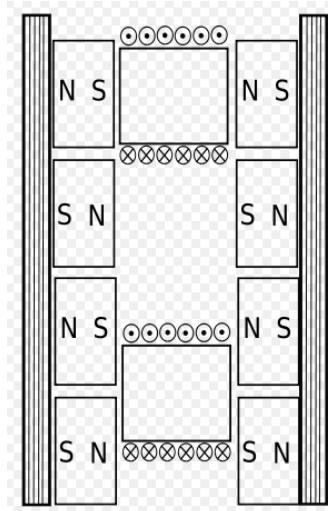
<http://upload.wikimedia.org/wikipedia/commons/7/74/Schrittmotor.PNG>



http://commons.wikimedia.org/wiki/File:Struttura_motore_passo-passo.jpg

Linear 모터, VCM 모터

- ❖ Linear 모터: 직선 운동 가능
- ❖ VCM: Voice Coil Motor



http://en.wikipedia.org/wiki/File:Linear_motor_U-tube.svg



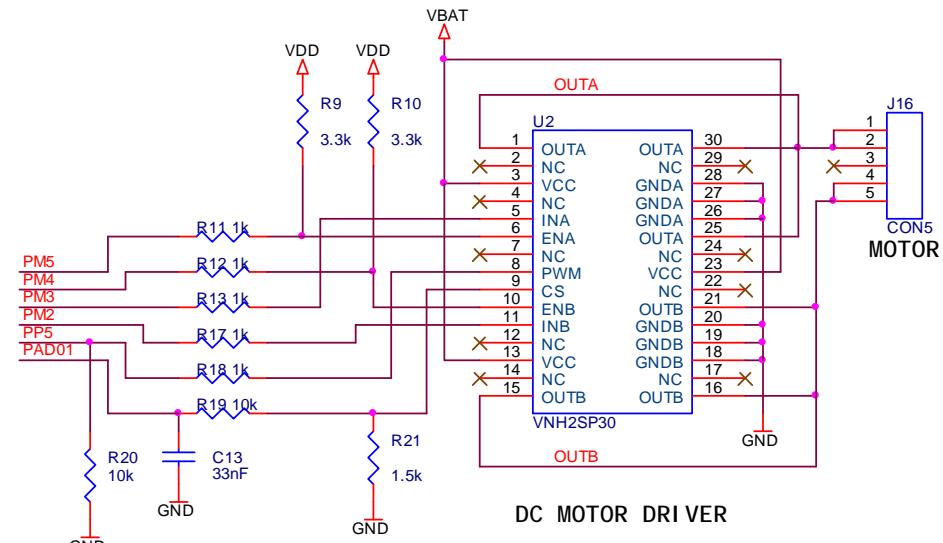
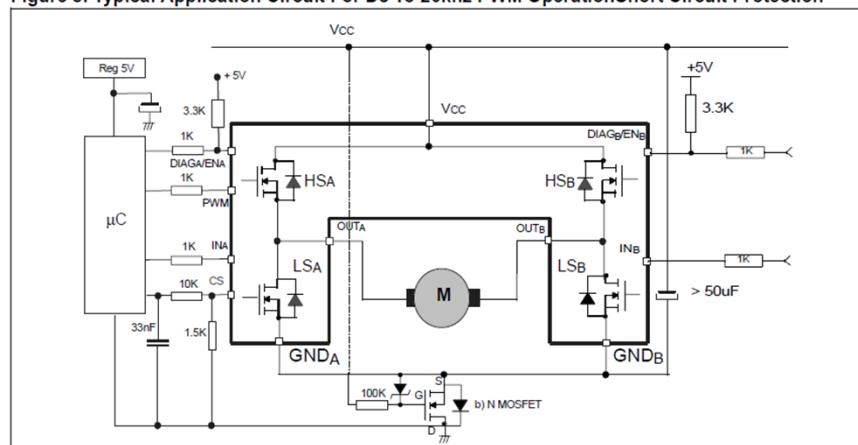
http://en.wikipedia.org/wiki/File:Hard_disk_dismantled.jpg

모터 드라이버 칩 VNH2SP30-E

❖ 자료 출처: VHN2SP30-E datasheet

- 19mΩ Rds(on)
- 30A

Figure 5. Typical Application Circuit For Dc To 20khz PWM OperationShort Circuit Protection

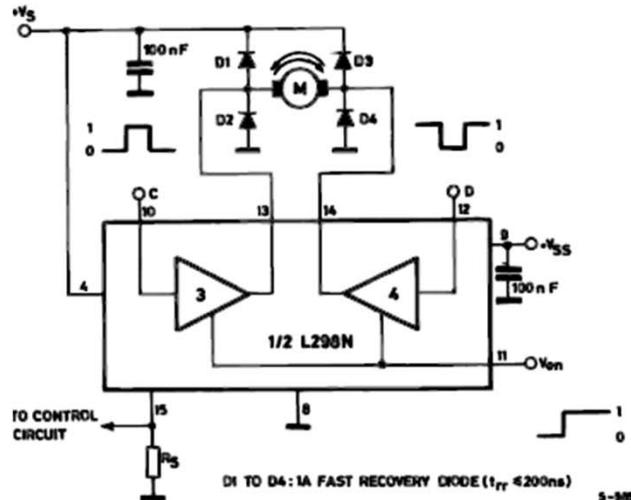


IN _A	IN _B	DIAG _A /ENA	DIAG _B /ENB	OUT _A	OUT _B	CS	Operating mode
1	1	1	1	H	H	High Imp.	Brake to V _{CC}
1	0	1	1	H	L	$I_{SENSE} = I_{OUT}/K$	Clockwise (CW)
0	1	1	1	L	H	$I_{SENSE} = I_{OUT}/K$	Counterclockwise (CCW)
0	0	1	1	L	L	High Imp.	Brake to GND

모터 드라이버 칩 L298

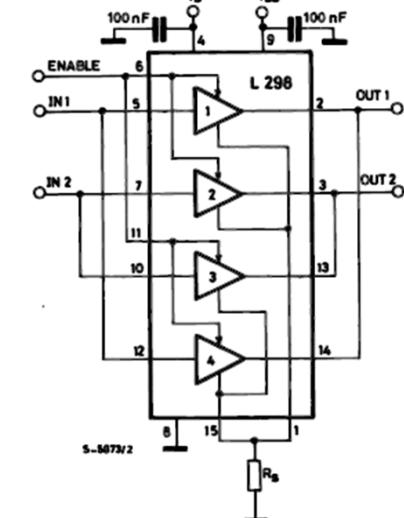
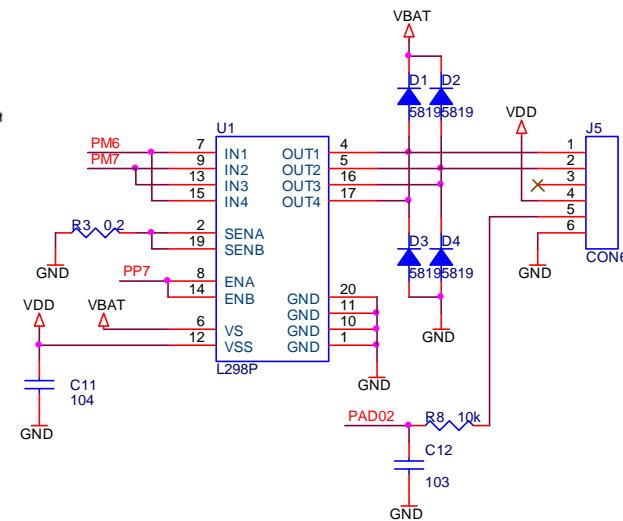
❖ 자료 출처: L298 datasheet

- 2A 2개 내장
- 2개 합치면 4A까지



Inputs		Function
$V_{en} = H$	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
$V_{en} = L$	C = X ; D = X	Free Running Motor Stop

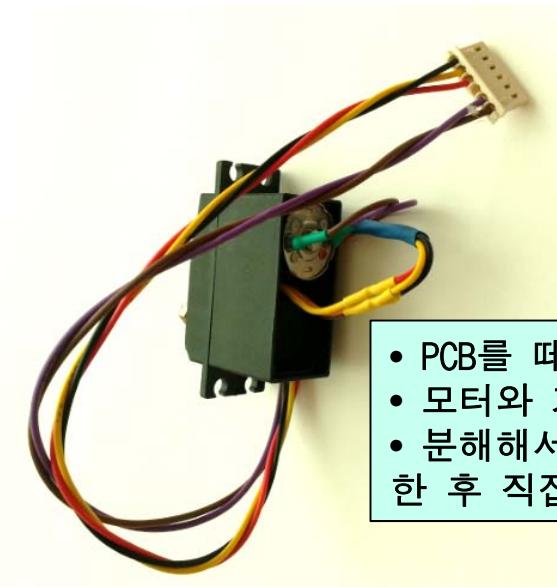
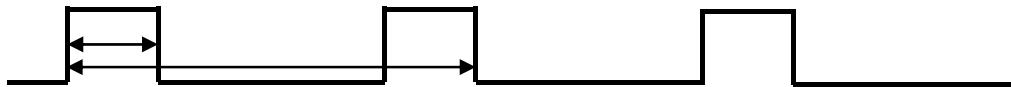
L = Low H = High X = Don't care



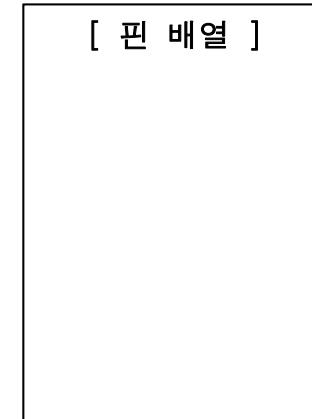
RC 서보 모터

❖ 제어 방법

- RC카 용으로 Pulse로 위치 신호 명령을 주는 서보 모터와 드라이버가 함께 내장된 모터
- 따라서, 일반 서보 모터와는 개념이 다르다는 사실에 유의!
- 주기는 보통 20msec. 정도
- 펄스 폭에 대한 일반적인 사양
 - ◆ Neutral 1.5msec.
 - ◆ 최소 0.9msec., 최대 2.1msec.



- PCB를 떼어낸 사진
- 모터와 가변저항 내장
- 분해해서 이렇게 연결
한 후 직접 구동 가능!



61. MotorControl – 개요

❖ 학습 목표

- pwm.h, pwm.c를 이용해서 모터를 구동할 수 있다.
 - ◆ VNH2SP30-E에 연결된 모터를 이용해서 모형차를 앞뒤로 이동시킬 수 있다.
 - ◆ RC Servo를 이용해서 모형차의 조향장치를 회전시킬 수 있다.

❖ 함수들

- void MCTL_Init(void);
 - ◆ 초기화
- void MCTL_RotateWheel(sWord angle);
 - ◆ 조향 장치 구동, -600~600 범위로 구동
- void MCTL_RotateEngine(sWord speed);
 - ◆ 모형차 전후진, -1023~1023 범위로 구동
- void MCTL_BrakeEngle(void);
 - ◆ 모형차 Brake

• motorcontrol.h

```
#include "project.h"
#include "pwm.h"

//-- MACRO -----
#define MCTL_WHEEL_MAX (PWM_MAXVAL23)
#define MCTL_WHEEL_MIN (PWM_MINVAL23)
#define MCTL_ENGINE_MAX (PWM_MAXVAL45)
#define MCTL_ENGINE_MIN (-PWM_MAXVAL45)

//-- FUNCTION -----
void MCTL_Init(void);
void MCTL_RotateWheel(sWord angle);
void MCTL_RotateEngine(sWord speed);
void MCTL_BrakeEngle(void);
```

61.MotorControl – 구현

• motorcontrol.c

```
void MCTL_Init(void) {
    DDRM_DDRM2=1; DDRM_DDRM3=1; DDRM_DDRM4=1; DDRM_DDRM5=1;
    PTM_PTm2=0; PTM_PTm3=0; PTM_PTm4=1; PTM_PTm5=1;
}

void MCTL_RotateWheel(sWord angle) {
    PWM_Set(PWM_CH3, angle);
}

void MCTL_RotateEngine(sWord speed) {

    PTM_PTm2 = 0;
    PTM_PTm3 = 0;

    if (speed>=0) {
        PTM_PTm2 = 0;
        PTM_PTm3 = 1;
        PWM_Set(PWM_CH5, speed);
    } else {
        PTM_PTm2 = 1;
        PTM_PTm3 = 0;
        PWM_Set(PWM_CH5, -speed);
    }
}

void MCTL_BrakeEngle(void) {
    PTM_PTm2 = 0;
    PTM_PTm3 = 0;
}
```

• main.c

```
void RotateWheel(char* str) {
    sWord angle;

    if (sscanf(str, "%d", &angle)<1) {
        SDBG_Printf("USAGE:w angle\n");
        return;
    }
    MCTL_RotateWheel(angle);
    SDBG_Printf("Wheel Angle %d\n", angle);
}

void RotateEngine(char* str) {
    sWord speed;

    if (sscanf(str, "%d", &speed)<1) {
        SDBG_Printf("USAGE:e speed\n");
        return;
    }
    MCTL_RotateEngine(speed);
    SDBG_Printf("Engine Speed %d\n", speed);
}

void main(void) {

    InitPeripherals();
    SDBG_RegisterNCmd('w', RotateWheel);
    SDBG_RegisterNCmd('e', RotateEngine);
    SDBG_RegisterHCmd('S', MCTL_BrakeEngle);

    ...
}
```

61.MotorControl – 컴파일 및 디버깅

❖ 실습: 컴파일하고 디버깅 해보기

- “w XXX”를 입력하면 XXX에 해당하는 PWM Duty로 J24에 연결된 RC 서보 모터 위치 제어
 - ◆ XXX는 -600 ~ 600 범위 (펄스폭 0.9msec ~ 2.1 msec 에 해당)
 - ◆ XXX가 0이면 1.5msec 정중앙
- “e YYY”를 입력하면 YYY에 해당하는 PWM 전압으로 J16에 연결된 모터 제어
 - ◆ YYY는 -1023~1023
 - ◆ 음수면 역회전, 양수면 정회전
- ‘S’를 입력하면 정지

❖ 고찰

- RC 서보 모터의 반응 속도가 느리면 어떻게 하면 좋을까?

❖ 느낀 점 및 참고사항 적기

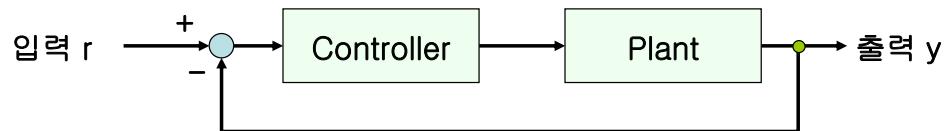
서보(Servo)에 대해서...

❖ 서보

- 서보(Servo)의 어원: 노예
- 서보(Servo): 1. = Servomechanism, 2. = Servomotor
- 중요 사실: 서보에는 피드백 제어 개념이 포함됨
 - ◆ 피드백 제어(=페루프제어): 출력이 입력과 같아지도록 출력을 피드백해서 제어에 반영 (상대적 용어: 오픈 루프 제어, 시퀀스 제어)

❖ 서보 모터

- 서보 모터가 따로 있을까?
 - ◆ 어떤 모터도 서보에 사용되면 모두 서보 모터
 - ◆ 서보를 하려면 모터의 출력(위치, 속도)을 측정해야 함
 - ◆ 원하는 출력(위치, 속도)이 잘 나오게 하려면 (= 피드백 제어가 잘 되게 하려면) 모터 특성(토크 리플, 마찰, 선형성 등)이 좋아야 함
- ➔ 모터의 출력(위치, 속도)를 측정하기 위한 센서(엔코더 등)가 부착되어 있으면 서보 모터 특성이 좋으면 금상첨화
 - ◆ 회사에서 좋은 서보 적용 제품은?
 - 타사와 같은 부품을 사용하더라도 더 높은 서보 성능이...
 - ◆ 이후 쓴 부품으로 좋은 부품과 동일한 성능이 나오게...

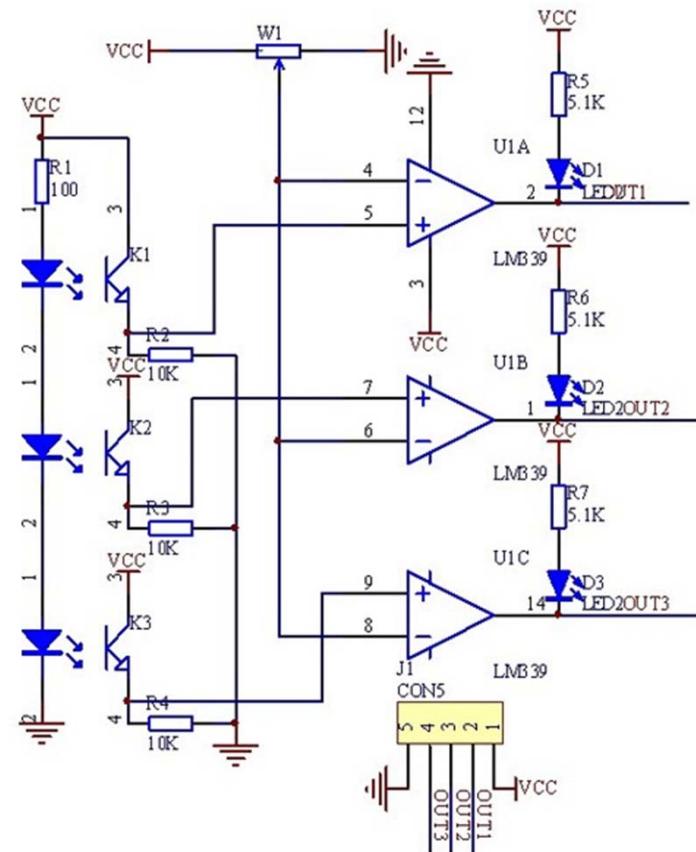
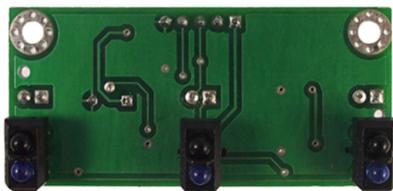
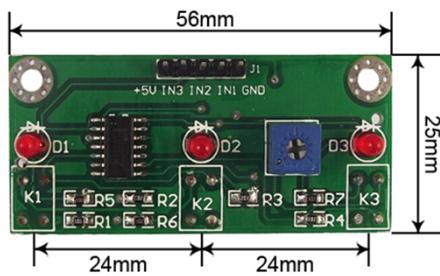


❖ 레귤레이팅과 트래킹

- 한 위치를 유지하는 것이 레귤레이팅
- 위치를 이동하는 것이 트래킹 – Trajectory Control

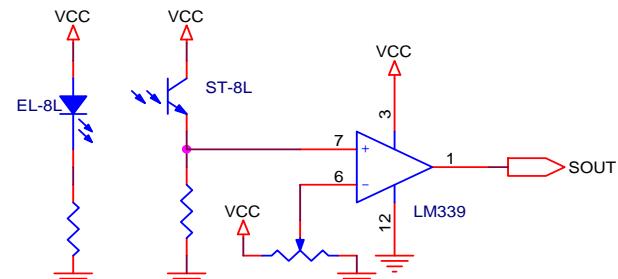
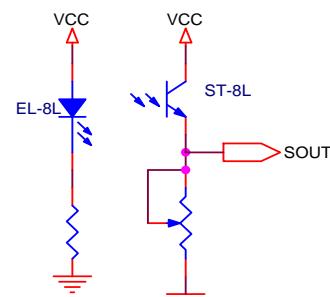
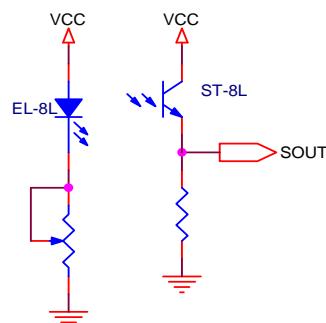
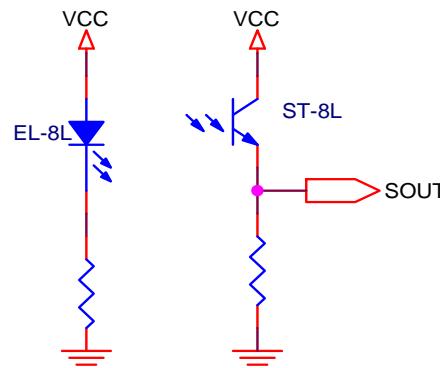
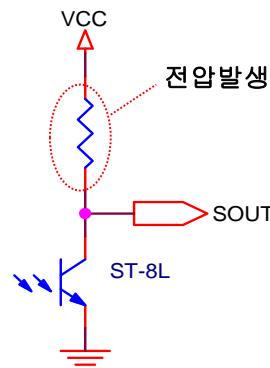
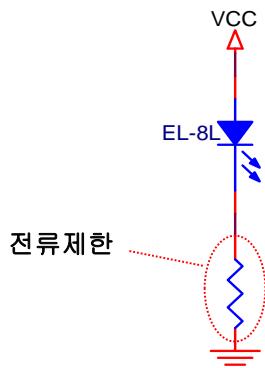
라인 감지 센서 상용 제품

❖ 자료 출처: 한진데이터 쇼핑몰의 적외선방식 트래킹 센서 (P0178)



라인 감지 센서 제작 (1)

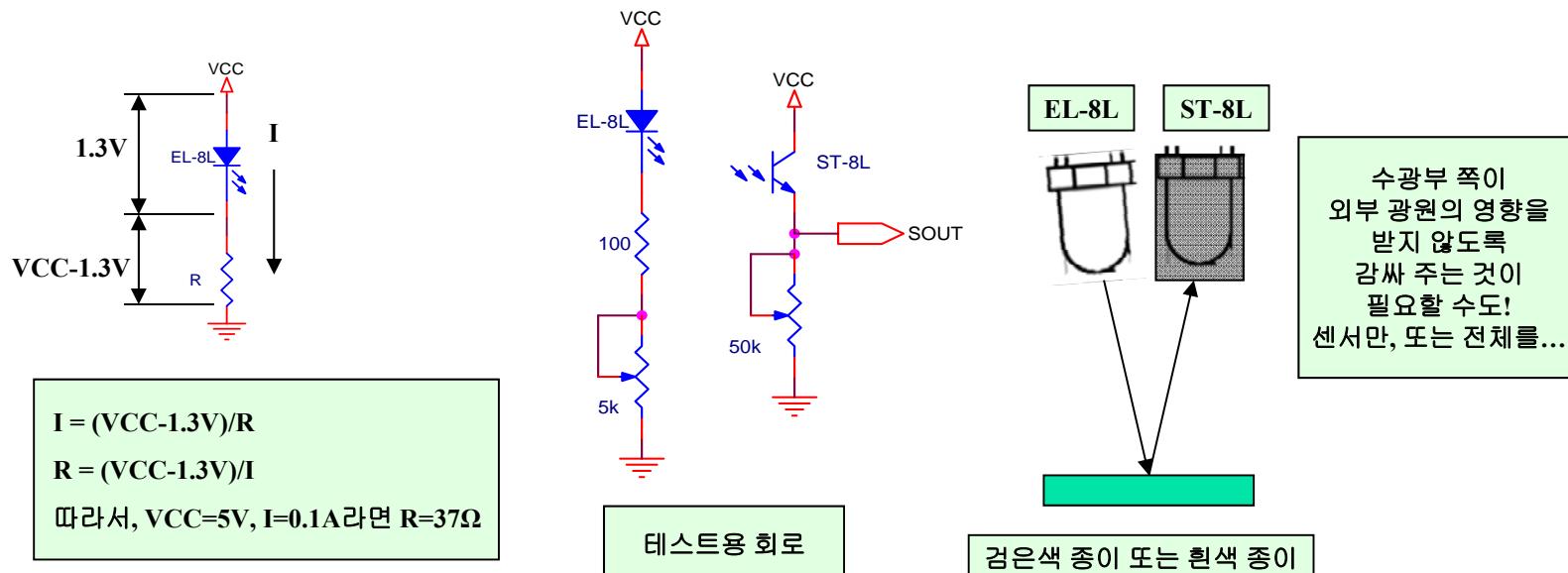
❖ 이런 방법들이 존재



라인 감지 센서 제작 (2)

❖ 테스트를 위한 회로

- 그런데, 앞에서처럼 회로를 만들어 테스트를 하면 문제가 발생할 수 있겠다. 왜냐고? 발광부의 경우 최대 전류가 100mA인데, 가변저항을 돌리다가 가변저항의 값이 0이 되면 발광부에 매우 많은 전류가 흘러 탈 수 있기 때문이지. 회로를 수정해야겠다. 먼저 100mA가 흐르기 위한 저항 값을 계산해보니 아래와 같군. 안전율을 고려해서 적어도 100Ω 이상의 저항을 달아야겠다. 테스트용이니까 적당한 크기의 가변저항을 선택해야겠군. 발광부의 경우 5kΩ을 달면 최소 전류 0.7mA 정도가 되겠고, 수광부의 경우는 50kΩ을 달면 0.1mA도 검출할 수 있겠군. $V=IR$ 의 관계로 계산해보면 나오지.
- 테스트로 적당한 저항값을 선정!



라인 감지 센서 제작 (3)

❖ 정리

1. 사용할 발광부, 수광부 소자를 결정한다.
2. 아날로그 전압을 측정할 지, 디지털 전압을 측정할 지 결정한다.
 1. 디지털 전압을 사용한다면, LM339와 같은 Comparator를 사용해서 MCU의 GPIO에 연결한다. 출력이 Open Collector임을 주의!
 2. 아날로그 전압을 바로 사용한다면, MCU의 ADC Multiplexer에 연결한다.
3. 어떤 방식으로 회로를 구성할 지 결정한다.
4. 발광부, 수광부 한 쌍만을 이용해서 저항 결정을 위한 테스트를 충분히 한다.
5. 테스트를 통해 결정된 설계값을 반영하여 라인 감지 센서부를 제작한다.
6. 수강부가 외부 광원의 영향을 덜 받도록 조치를 취한다.

❖ 다른 좋은 방법이 있다면?

- 색상 감지 센서를 사용하는 것은?
- 비싸더라도 상용 센서를 사용하는 것은?
- 기타?

❖ 공유할 수 있는 라인 감지 센서 회로부 PCB를 만든다면?

- 어떻게 설계를 해서?
- 누가?

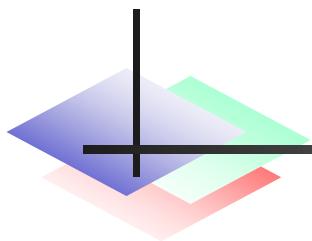
마치면서...

❖ MCU 응용 교육을 마치면서...

- 많은 예제를 만들다 보니 시간이 정말 많이 걸렸습니다.
- 보통의 MCU 교육은 Reference Manual을 기반으로 해서 MCU의 여러 기능을 소개하고 해당되는 레지스터를 소개하는 식입니다. 여기에 간단한 몇 가지 장치를 추가해서 응용하는 예제 몇 개를 소개하는 정도지요.
- 이런 식의 교육도 괜찮지만, MCU에 여러 장치를 물려 동시에 사용할 때에는 응용이 쉽지 않은 단점이 있습니다.
- 그래서 저는 실제 시스템에서 유용하게 사용할 수 있는 F/W 소스 코드를 어떻게 만들어 나가야 되는지에 중심을 두었습니다. 그래서 이 교육을 통해 제공되는 소스 코드는 여러분들의 작품 제작에 매우 유용하게 사용될 것입니다.
- 작품 제작에는 여기 소개된 소스 코드만 사용되는 것이 아닙니다. 신뢰성 있는 소스 코드를 만들어 미션을 100번 시도면 100번 다 성공하는 그러한 알고리즘과 이를 잘 뒷받침할 수 있는 소스 코드를 만들기 바랍니다.

❖ MCU 응용은 누가?

부록



DEMO9S12XEP100의 확장

부록: 안전에 대해

❖ 우려되는 안전 사고

- 리튬 폴리머 배터리의 배선, 방전, 충전 중 사고
 - ◆ Short 확인! 확인! 확인!
 - ◆ 과전류에 의한 열 발생 여부 확인 및 대처
 - ◆ 아무도 없을 때 충전하지 말기!
 - ◆ 이동 중 배터리 관리 철저!
- 인두에 의한 화상 사고
- 가공 기계(드릴 등)에 의한 상해
- 기타 안전 사고...

❖ 커넥터를 반드시 사용할 것!

- 커넥터를 제작 사용하여 전기적 Short가 발생하지 않도록 주의!
- 커넥터 제작 후 제대로 제작했는지 필히 검사!

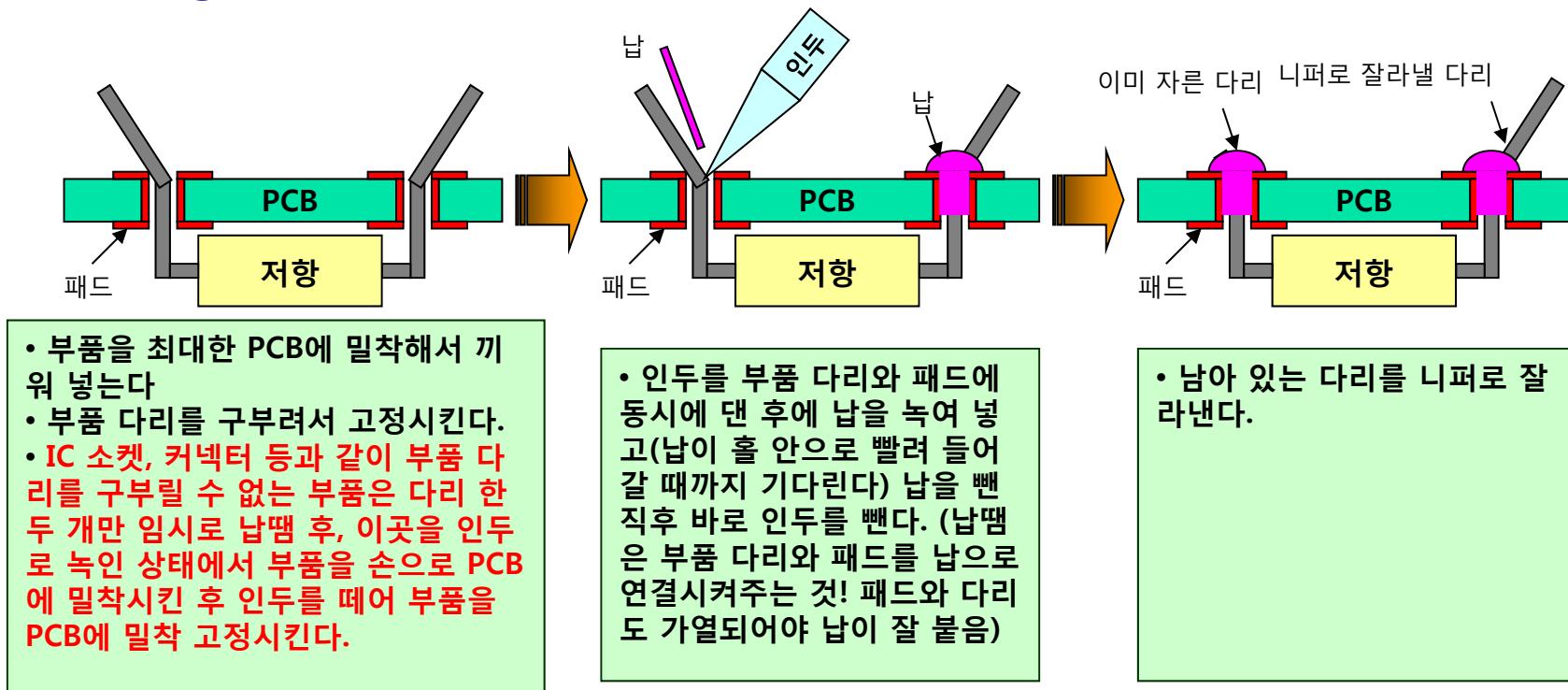
❖ 안전은 아무리 강조해도 지나치지 않습니다. 안전에 최대한 신경을...

부록: 납땜을 잘 하려면... (1)

❖ 단면기판에 납땜할 때의 방법을 양면기판에 적용하지 말 것!

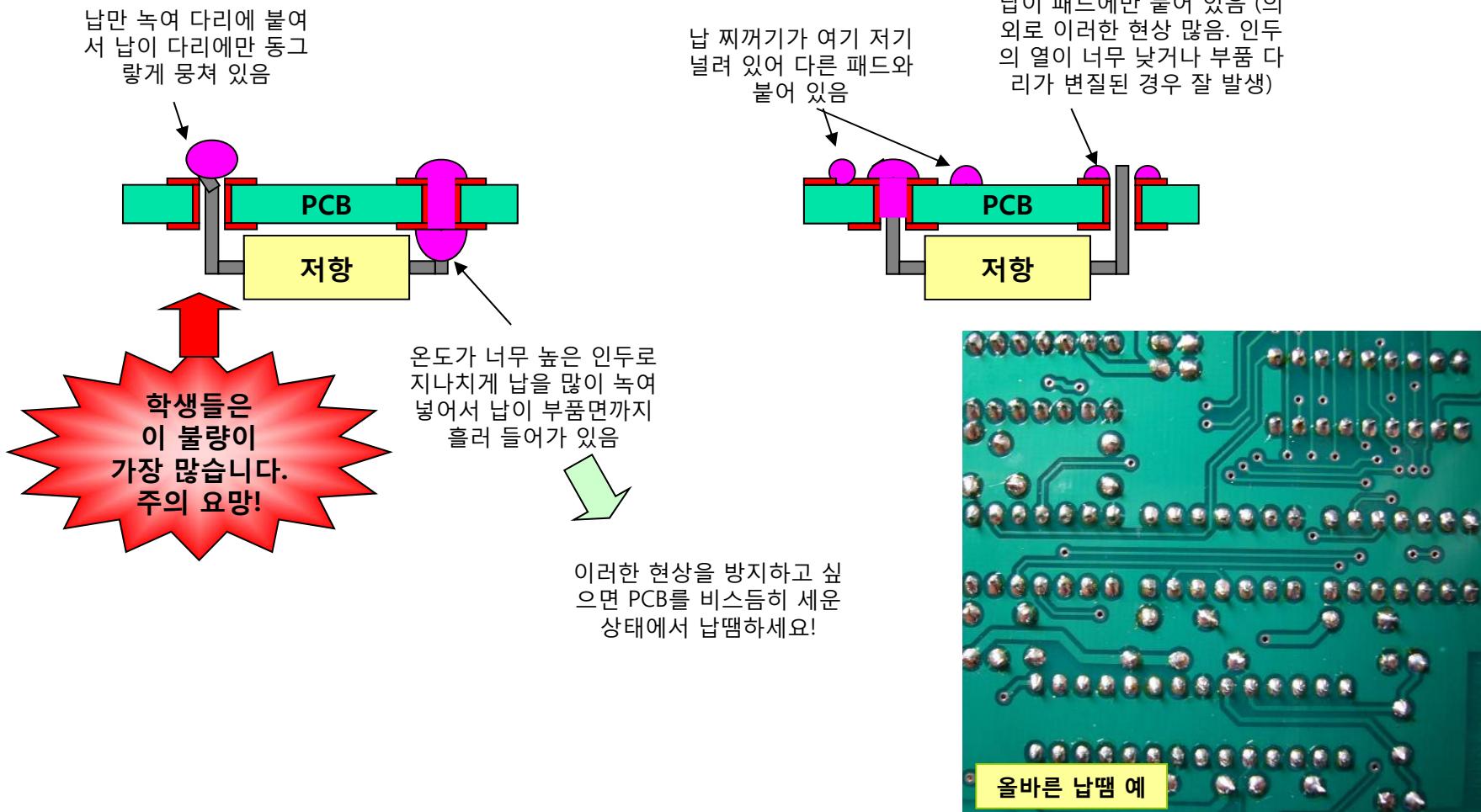
- 부품 다리를 완전히 접지 말 것
- 부품 다리를 자른 후에 납땜하지 말 것
- 고온의 인두기에 납을 많이 사용하지 말 것:
 - ◆ 인두기가 너무 고온이면 납이 PCB 홀로 너무 많이 흘러 들어가 납땜을 하는 면 반대쪽으로 다량의 납이 흘러나옴

❖ Through Hole을 이용하는 부품의 납땜 방법



부록: 납땜을 잘 하려면... (2)

❖ 참고



부록: 납땜을 잘 하려면... (3)

❖ SMD

- SMD란?
 - ◆ Surface Mounted Device : 표면 실장 장치/부품
- SMD 납땜 방법
 - ◆ <http://www.youtube.com>에서
 - ◆ SMD soldering이란 검색어로 검색 후 동영상 시청
- SMD 납땜 준비물
 - ◆ 온도 조절 가능 인두
 - ◆ 플럭스 펜 또는 솔더 페이스트 등

❖ 전선의 연결

- 와이어를 연결한 후 반드시 납땜으로 고정
- 열수축 튜브를 이용해서 절연



납땜

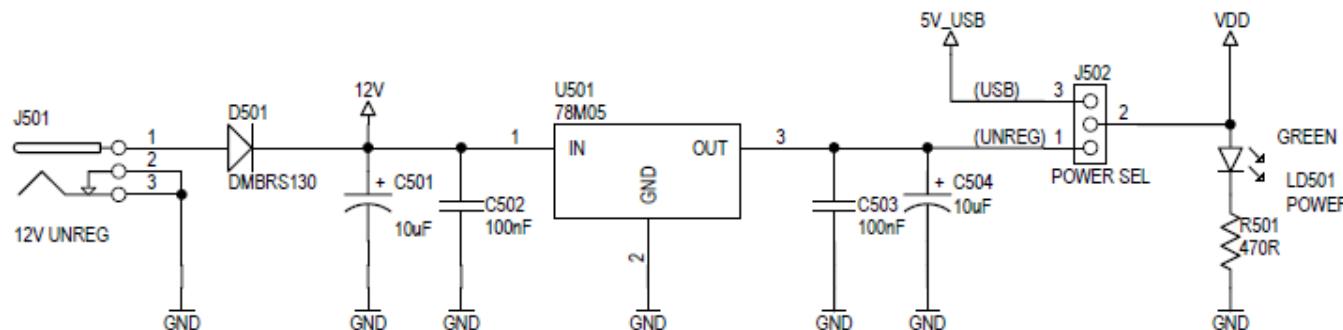


내부에 선을 납땜 한 후
에 열수축 튜브로 감쌈

부록: DEMO9S12XEP100의 Regulator 교체 이유

❖ DEMO9S12XEP100 보드의 J501 DC 어댑터 잭에 의한 전원 공급

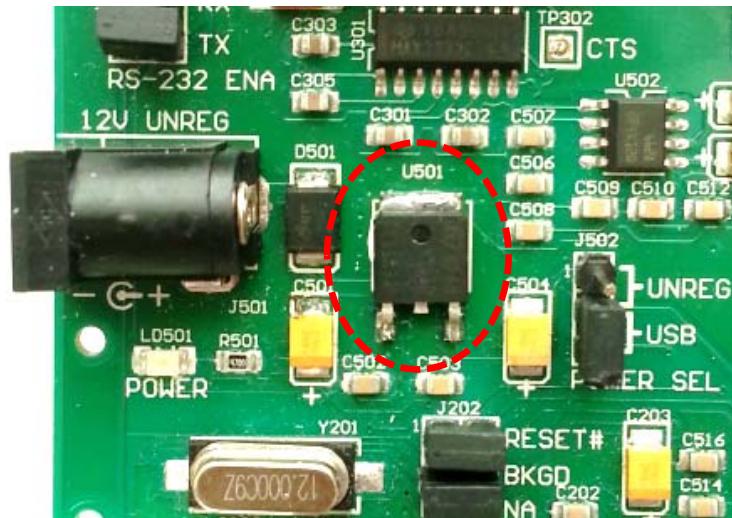
- SoftTec 제공 회로도를 보면 J501에 공급되는 전원이 Schottky 다이오드를 거쳐 78M05에 연결
 - ➔ VDD(5V)보다 대략 2V 이상 높은 전압이 필요!
 - ➔ 배터리 전압이 약 7V 이하면 VDD 전압이 5V 보다 낮아지고 이에 따라 문제가 발생할 수 있음
 - ◆ Max. Current가 0.5A 밖에 되지 않음
- 참고: Schottky diode의 양단을 전선으로 연결해서 Schottky diode에 의한 0.3V 정도의 전압 강하를 방지할 수 있음
 - ◆ 이 다이오드는 왜 있을까?
 - ◆ 이 방법 사용 시 전원 극성에 반드시 유의할 것!
- 해결방법1: 78M05를 Package도 같고 Pin 배치도 같은 LDO Voltage Regulator로 변경
 - ◆ LDO로 교체하면 약 1V 정도의 마진을 더 확보하게 됨
 - ◆ 찾아보니 몇 개 나옴
- 해결방법2: MCU VDD를 위해 별도의 배터리를 사용
- 해결방법3: 방법1과 방법2 동시 사용!



부록: DEMO9S12XEP100의 Regulator 교체 방법

- ❖ DEMO9S12XEP100 보드의 J501 DC 어댑터 잭에 의한 전원 공급
 - U501의 기존 78M05 칩을 들어 내고
 - L4941BDT-TR을 납땜
- ❖ L78M05CDT
 - Max. 0.5A, 5V Regulator
 - Dropout volatage : 350mA일 때, 2V
- ❖ L4941BDT-TR
 - Very low drop Max. 1A, 5V regulator
 - Dropout volatage : 1A일 때, Typ. 450mV, Max. 700mV

- ❖ 변경 후의 장점 2가지는?



부록: DEMO9S12XEP100의 전원 입력

❖ J501을 통한 전원 입력

- U501 Voltage Regulator를 L4941BDT-TR로 변경했기 때문에
- D501의 전압 강하 0.3V와 U501의 전압 강하 0.7V(1A 일 때 최대값)을 합해
- 최소 DC 6V 이상을 입력해야 함

❖ J501에 DC 6V 이상 입력

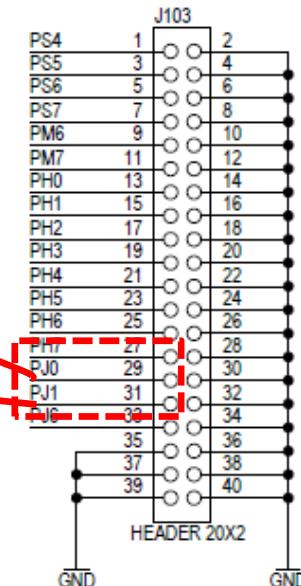
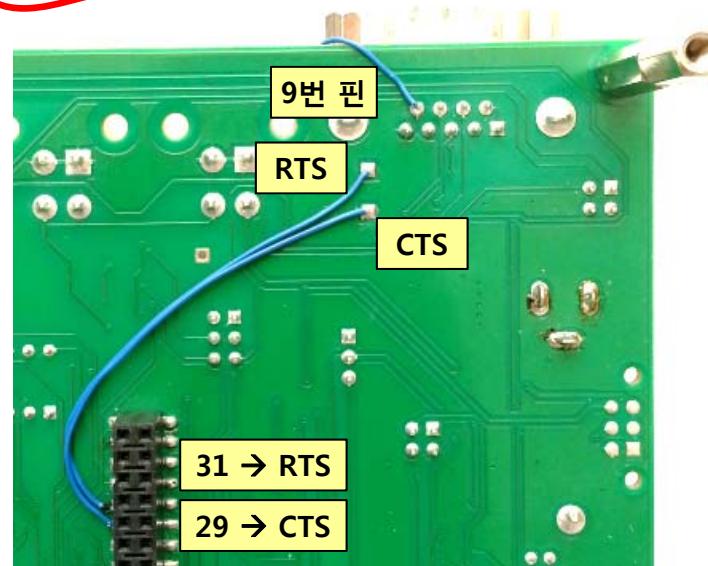
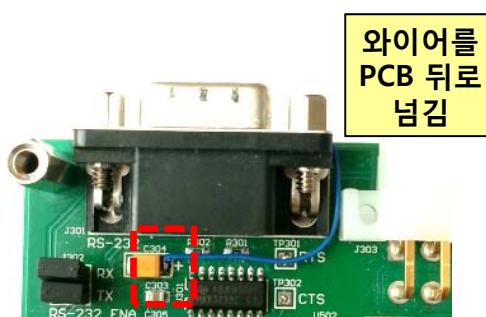
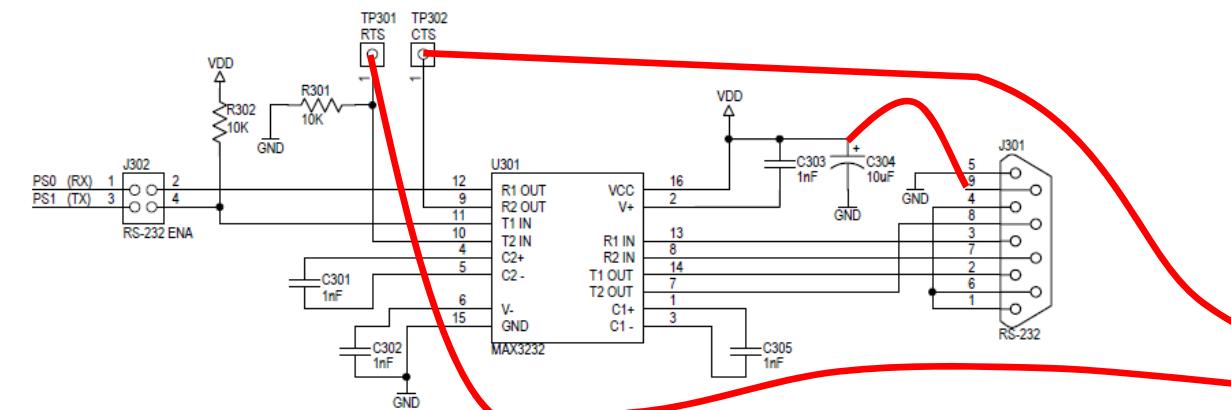
- AA 건전지 5~6개가 적당
- AA 충전지는 6개가 적당
- 참고: 전압이 높을수록 U501의 발열량도 많아짐
- J502의 점퍼를 UNREG에 연결



부록: DEMO9S12XEP100의 SCI2 확장

❖ SCI2 (RXD2와 TXD2)의 확장

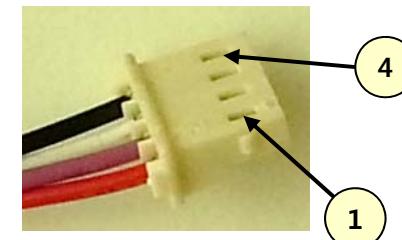
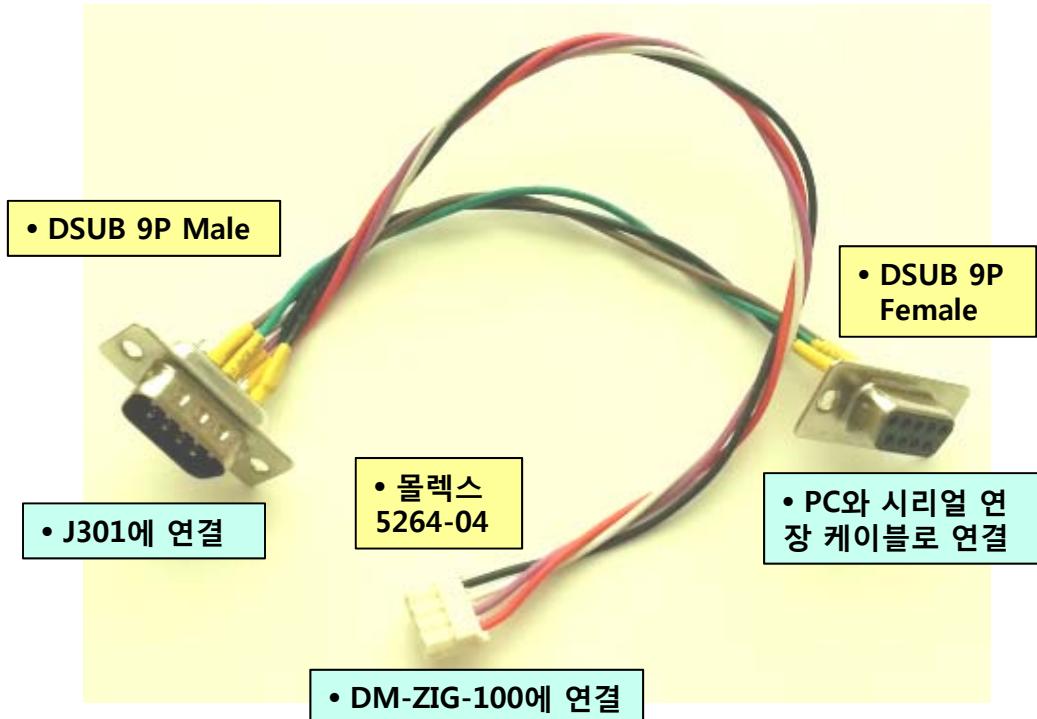
- RTS와 CTS를 사용하지 않으므로 CTS에 RXD2(PJ0)를 RTS에 TXD2(PJ1)를 연결
- C304의 VDD를 J301의 9번 핀(RS-232에서는 RI 입력)에 연결하여 확장



부록: DEMO9S12XEP100의 J301 확장 케이블

❖ 목적

- J301을 통해 SCI0과 SCI2 통신이 가능하도록!
 - ◆ SCI0은 시리얼 연장 케이블에 연결
 - ◆ SCI2는 DM-ZIG-100에 연결 (Bluetooth 통신 또는 ZigBee 통신을 위해)



DSUB 9P Male	DSUB 9P Female	5264-04
2번	2번	
3번	3번	
5번	5번	4번
7번		3번
8번		2번
9번		1번

부록: Expansion Board – 개요 (1)

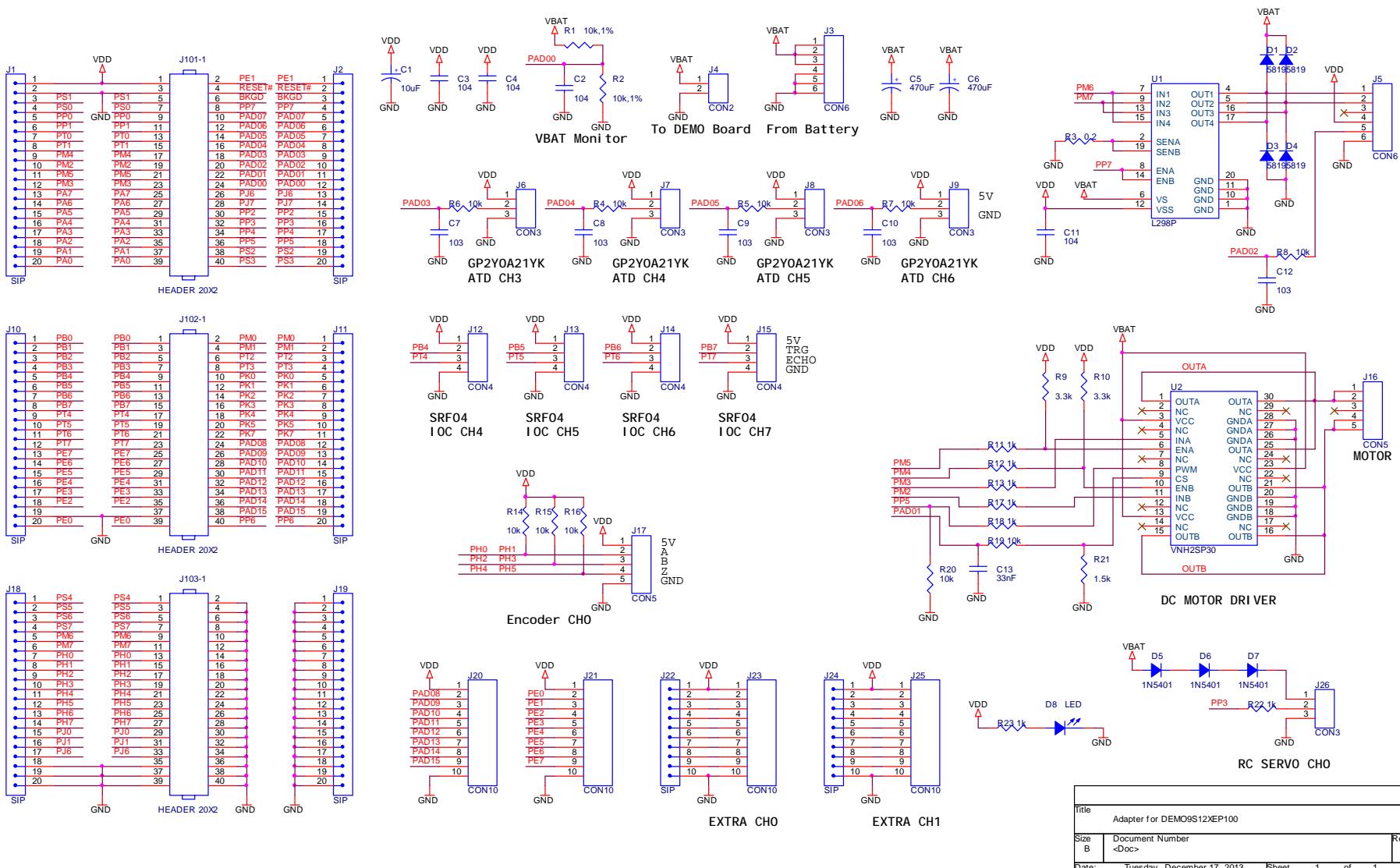
- ❖ 먼저: 이 PCB의 사용은? 교육에서는 필수, 과제에서는 선택!
- ❖ 기능: DEMO9S12XEP100의 윗면에 부착되어 확장 커넥터 등을 제공
 - 자세한 기능은 schematic을 통해 확인
- ❖ 설계 단상
 - 다양한 사용자 의도를 충족시키기 위해 전선 배선을 위한 훌 마련!
 - 배터리 및 모터의 샘플 부재 & 관련 자료 부재로 커넥터 임의 선택
 - ◆ MOLEX 5267, 5264 계열
 - ◆ 참고: MOLEX 커넥터의 핀 번호가 실제 PCB에는 반대로...
 - 제가 어느 회사의 제품을 참고하여 MOLEX 커넥터의 PCB footprint를 만들었는데 그 제품이 반대였습니다. 저는 그 PCB만 믿고 여기 저기에 MOLEX 커넥터의 핀 번호를 부여했는데... '꺼진 불도 다시 보자!'처럼 아무리 사소한 것이라도 반드시 확인하고 넘어갑시다!
 - 배터리 및 DC 모터 연결 커넥터의 경우 전류를 매우 많이 사용할 것으로 생각되어 접촉저항을 줄이기 위해 커넥터의 2~3핀을 할당 (Schematic 참조)
 - A/D 채널에는 R-C LPF 적용
 - RC Servo Motor는 1N4007 3개를 이용해서 전압이 약 2~3V 강하된 전원을 연결 (RC 서보 모터의 전원 스펙을 맞추기 위해서...)
 - 전원 방향이 틀린 경우에 대한 보호 회로(= MOSFET 추가) 사용 안 함
 - ◆ 전원 방향을 반대로 해서 연결하면 Expansion Board가 망가질 수도!
 - ◆ 일부러 이렇게 했습니다. 정신 없이 하다가 한번 태워 보시라고...

부록: Expansion Board – 개요 (2)

❖ 제작 순서 및 사용법

1. DEMO9S12XEP100의 Voltage Regulator 변경 작업 수행
 2. DEMO9S12XEP100의 J505 점퍼 OFF (PAD00과 가변저항 연결 OFF)
 3. DEMO9S12XEP100의 J506 점퍼 OFF (PAD01과 CDS 연결 OFF)
 4. 필요에 따라 J502의 점퍼를 UNREG 또는 USB로
 - ◆ Expansion Board를 장착하면 점퍼 옮기는 것이 힘듦 → 핀셋 등을 이용
 5. SCI2 (PJ0, PJ1)을 MAX3232에 연결, DSUB 9P 커넥터에 5V 연결 작업 수행
-
- 이후 Expansion Board를 DEMO9S12XEP100의 **윗면**에 부착
 - ◆ 3개의 20×2 커넥터(전체 높이 24mm)와 PCB 서포터(15mm)로 고정
 - Schematic을 참고하여 전선으로 필요한 부분을 배선해서 사용

부록: Expansion Board – Schematic

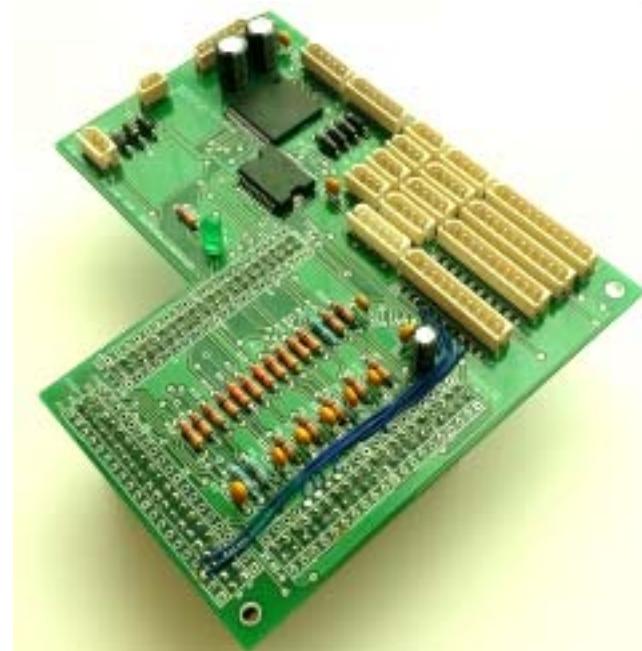
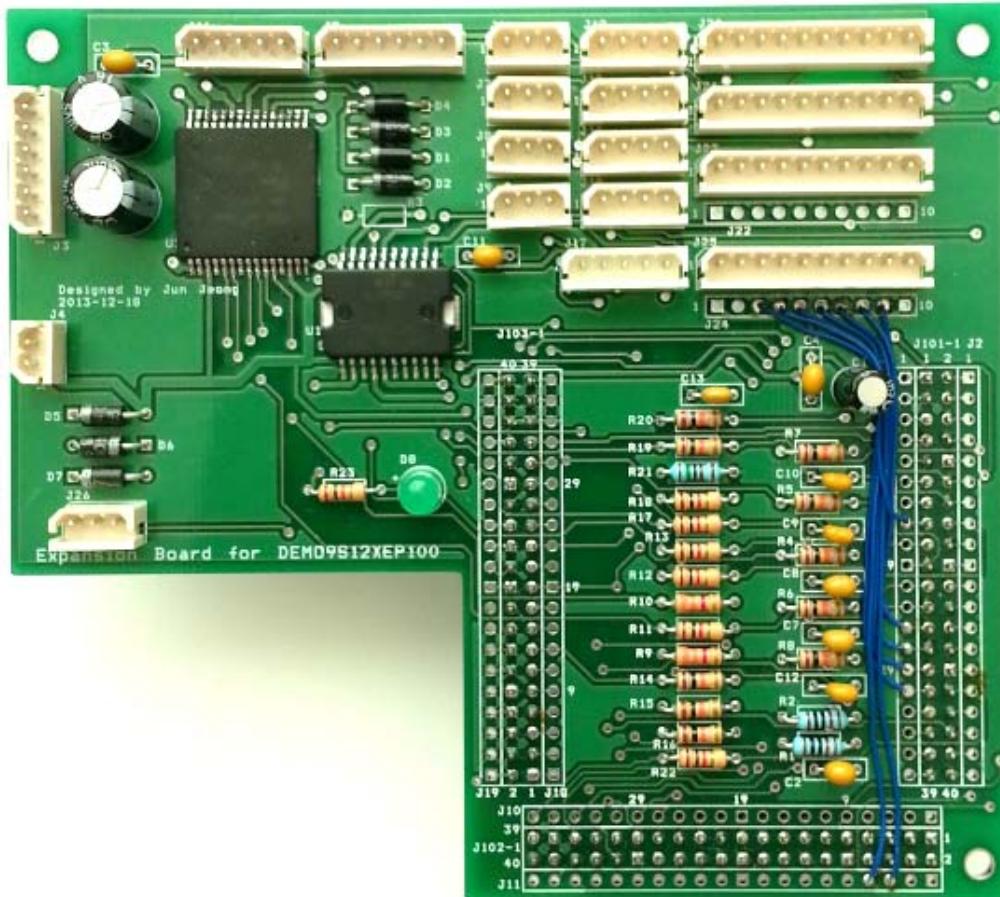


Title		
Adapter for DEMO9S12XEP100		
Size B	Document Number <Doc>	Rev 1.0
Date: Tuesday, December 17, 2013	Sheet 1	of 1

부록: Expansion Board – 제작

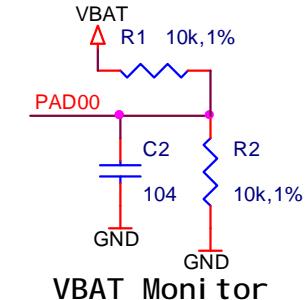
❖ 주의 사항

- J101-1, J102-1, J103-1에 헤더핀의 납땜은 Expansion 보드를 DEMO9S12XEP100 보드에 결합한 후에 진행 → 그래야 헤더핀의 위치가 정확히 맞아 두 보드의 결합/해체가 원활함

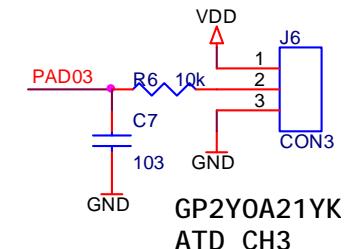


부록: Expansion Board – 주요 회로 설명 (1)

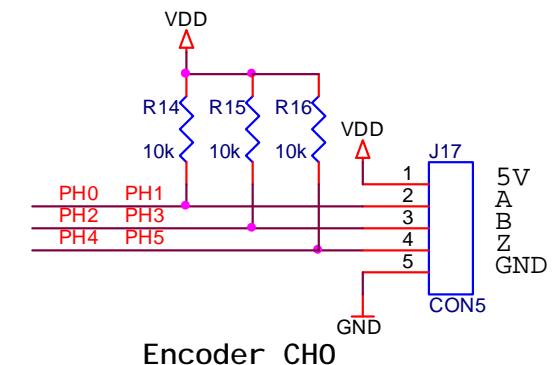
- ❖ 배터리 전압을 측정하기 위해 사용 ($V_{PAD02} = V_{BAT}/2$)
 - Li-Po 배터리는 3.7V 셀 당 대략 2.4V 이하로는 사용하지 말 것을 권장
 - 그러나 마진과 안전을 고려해서 3.0V 이하로는 절대 사용하지 말 것을 권장
 - 따라서, 7.4V Li-Po 배터리는 약 6.0V 이하로는 사용하지 말 것을 권장
 - 그런데, 뒤에 설명할 Voltage Regulator를 고려해서 **약 6.4V 이하로는 사용하지 말 것!**



- ❖ 1차 Analog R-C Low Pass Filter로 고주파 노이즈 제거
 - Roll-off 는 -20dB/decade , Cut-off Frequency는 $1/(2\pi RC)$ Hz
 - 주의 사항: Phase Delay는 Cut-off Freq.에서 -45도, 1/10의 Cut-off Freq.에서 변화하기 시작함. → Phase Delay가 있으므로 실시간 제어에서는 반드시 Phase Delay도 고려해야 함!



- ❖ Encoder 출력 신호가 Open Collector Type 이므로
 - 저항을 사용해서 부하를 걸어줘야 디지털 출력이 됨!



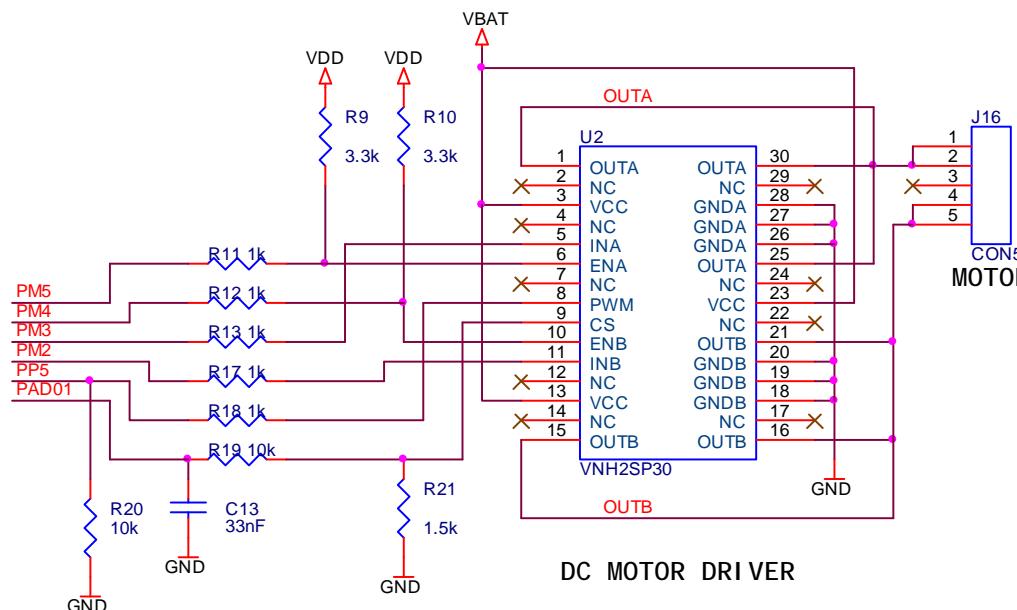
부록: Expansion Board – 주요 회로 설명 (2)

❖ 모터 드라이버에 달린 저항들은?

- 모터 드라이버의 잘못된 사용으로 파괴되는 경우 MCU의 디지털 출력 핀이 망가지지 않도록 보호하기 위해서... 권장 회로에 있어서 사용했습니다!

❖ R21은? 모터 드라이버에서 모터의 전류를 측정하기 위한 회로

- R-C 필터 설계는 PWM 주파수와 F/W를 고려해서 설정
- Datasheet 참조



부록: Expansion Board – DEMO9S12XEP100 결합

❖ 결합/해체 시 유의사항

- 무리한 힘을 가하지 말 것!
- 결합할 때나 빼 때나 조금씩 조금씩!



부록: DM-ZIG-100 – 제작 및 회로도

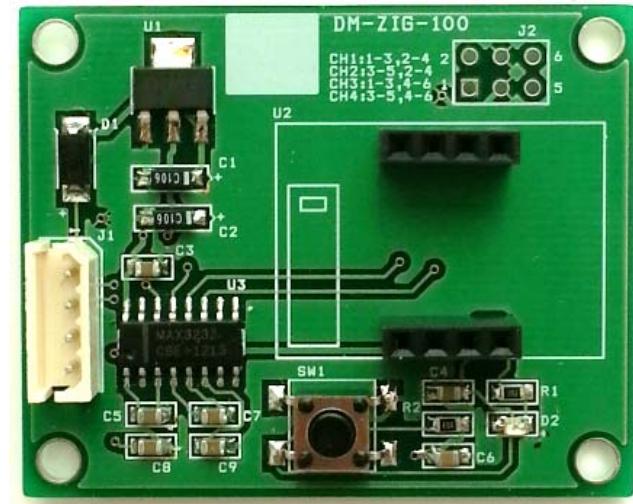
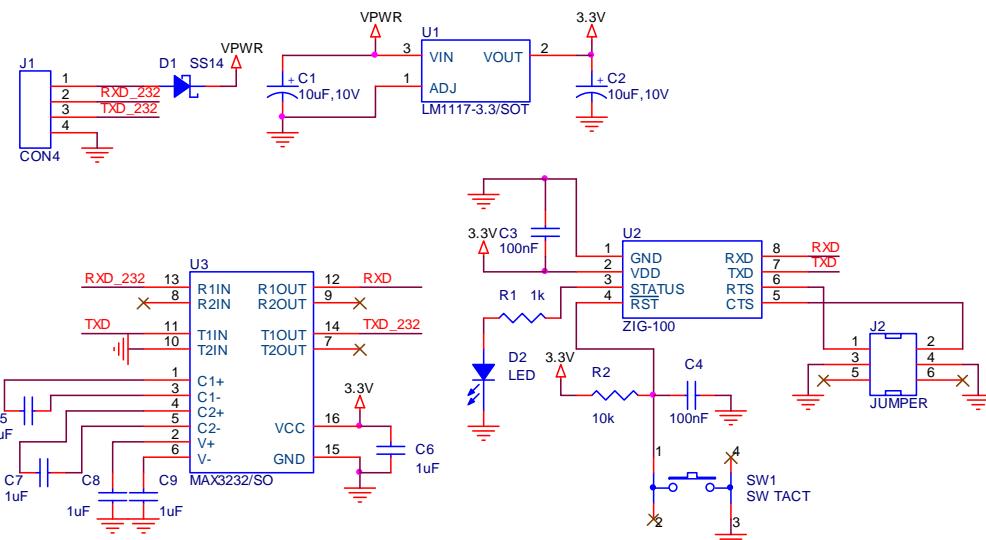
❖ 보드 사이즈

■ 50mm × 40mm

❖ 제작 시 주의점

- 방향이 있는 부품의 납땜 주의
 - ◆ 다이오드, 커패시터
- FB-155BC 장착 시 방향 주의

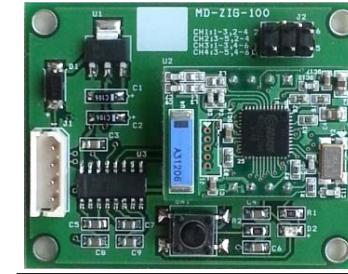
❖ 회로도 및 완성 사진



부록: DM-ZIG-100 – 목적 및 연결

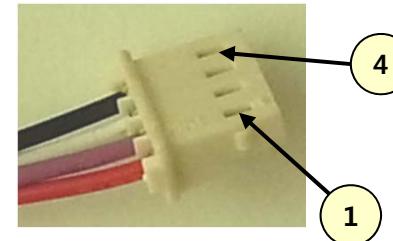
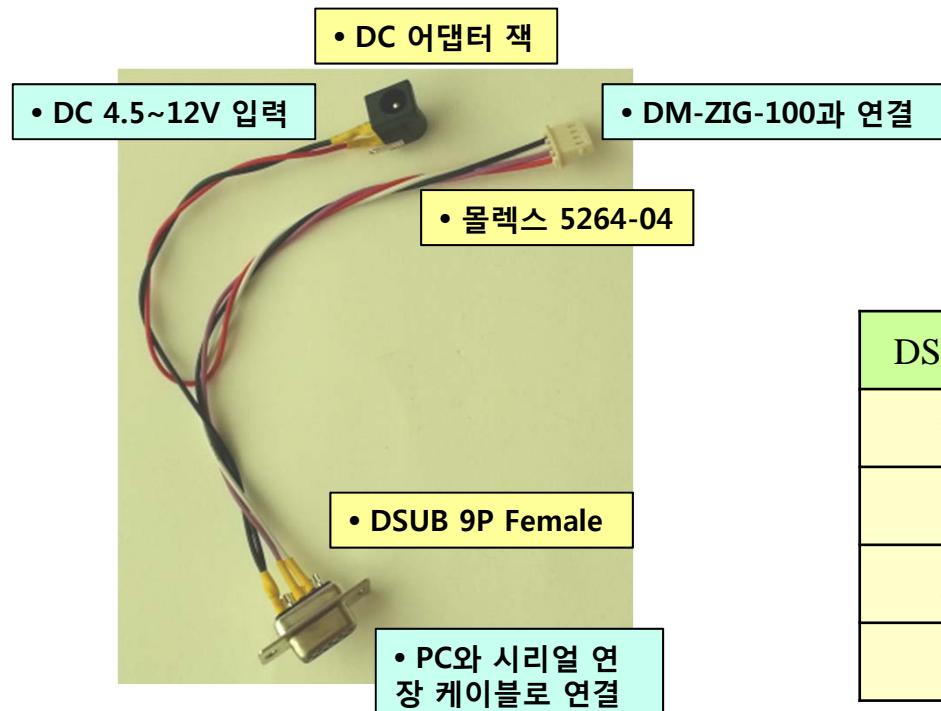
❖ DM-ZIG-100에 바로 연결할 수 있는 통신 모듈

- (주)로보티즈 사의 ZigBee 통신 모듈 ZIG-100
- (주)펌테크 사의 FB155BC
- 등등



• ZIG-100 연결 사진

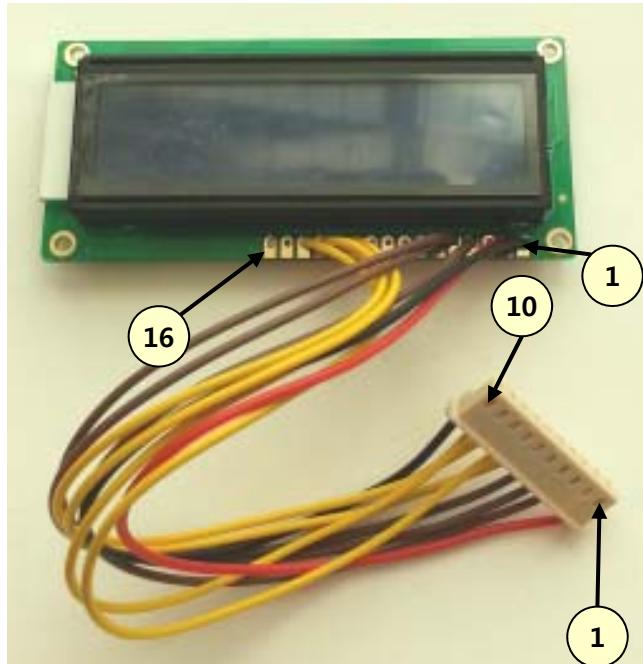
❖ PC와의 연결



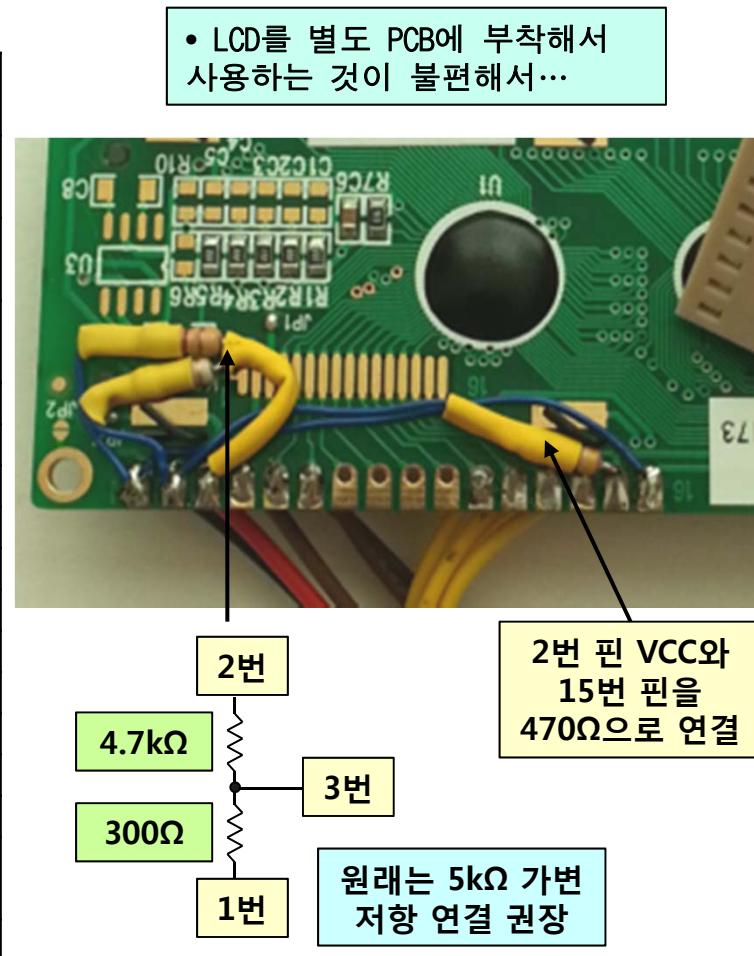
DSUB 9P	5264-04	DC 파워잭
2번	3번	
3번	2번	
5번	4번	-극
	4번	+극

부록: Character LCD 연결 준비

- ❖ Back Light의 전류 소모량을 줄이기 위해 470Ω 직렬 저항 삽입
- ❖ 실험결과 Contrast 조절 전압은 0.3V 정도면 적당
 - $4.7k\Omega$ 과 300Ω 저항 사용



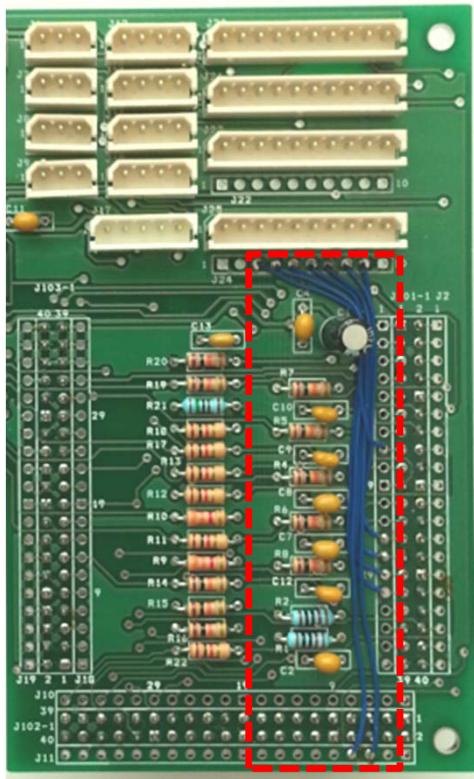
LCD	5264-10
1번	300Ω 통해 3번
2번	4.7kΩ 통해 3번
3번	위 두 저항 연결
4번	
5번	3번
6번	4번
7번	5번
8번	
9번	
10번	
11번	
12번	6번
13번	7번
14번	8번
15번	470Ω 통해 2번
16번	1번



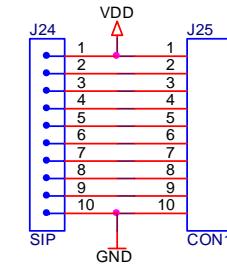
부록: DEMO9S12XEP100와 Character LCD 연결

❖ J25 사용의 경우

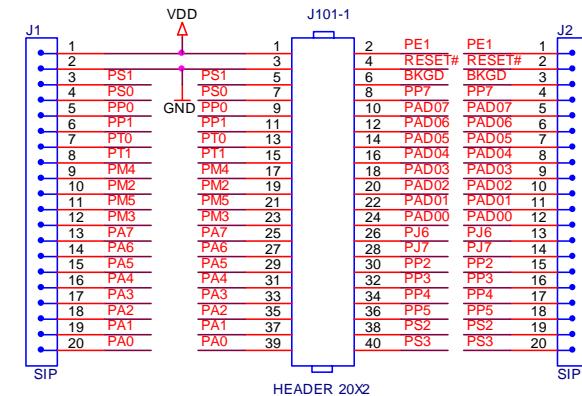
- PA7~PA4를 LCD의 D7~D4에 사용
- PT1~PT3을 LCD의 RS, RW, E에 사용



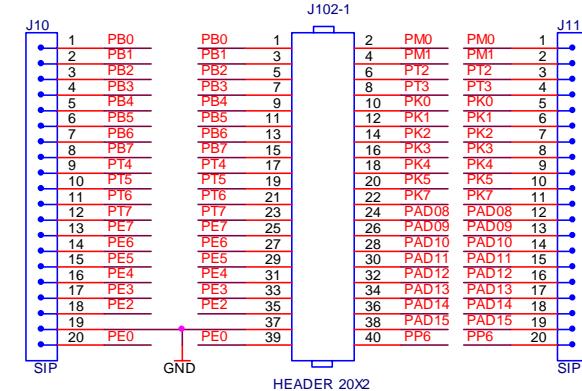
MC9S12XEP100	J25
	2번
J101-1의 15번	3번
J102-1의 6번	4번
J102-1의 8번	5번
J101-1의 31번	6번
J101-1의 29번	7번
J101-1의 27번	8번
J101-1의 25번	9번



EXTRA CH1



HEADER 20X2



HEADER 20X2

부록: 기타 커넥터 연결

- ❖ **Rotary Encoder E30S4**
- ❖ **적외선 거리 측정 센서 GP2Y0A21YK**
- ❖ **초음파 센서 모듈**
- ❖ **각종 모터류**
 - 회로도 참조
- ❖ **배터리 및 스위치**
 - 충전지는 충전까지 고려하여 커넥터 설치!

Rotary Encoder E30S4-200-3-N-5	몰렉스 5264-05
+V (Brown)	1
OUT A (Black)	2
OUT B (White)	3
OUT Z (Orange)	4
0V (Blue)	5

GP2Y0A21YK	몰렉스 5264-03
3 (Vcc)	1
1 (Vo)	2
2 (GND)	3

초음파센서 모듈	몰렉스 5264-04
VCC	1
Trigger Pulse	2
Echo	3
GND	4

부록: FB155BC 두 개 서로 연결

❖ 관련 자료

- (주)펌테크의 문서 참조!
- http://firmtech7.cafe24.com/bizdemo4649/product/bluetooth/sub01_product01.php

❖ 설정 변경

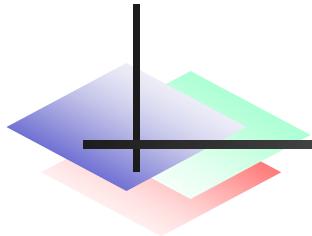
- PC와 연결하여 Tera Term 등으로 변경
 - ◆ 9600bps 사용
- Tact 스위치를 누른 상태에서 전원 투입
 - ◆ 2초 이상 누르고 있으면 초기화 되므로 주의!
- 설정 방법
 - ◆ BLUETOOTH PARAMETERS에서 한 쪽은 MASTER로 다른 한 쪽은 SLAVE로
 - ◆ SECURITY PARAMETERS에서 PIN CODE를 동일하게
 - ◆ SYSTEM PARAMETERS에서 CONNECTION MODE는 MODE1, MODE2, MODE3 중 목적에 맞게 선택
 - ◆ SYSTEM PARAMETERS에서 STATUS MESSAGE는 DISABLED로



❖ 이렇게 하면!

- PC와 DEMO9S12XEP100을 무선으로 연결할 수 있습니다.
- 또한, 스마트폰의 Bluetooth와도 연결이 가능합니다.
 - ◆ 인터넷에서 자료를 찾아보면 스마트폰 용 Application도 쉽게 개발할 수 있을 것입니다.

차량 조향 모델과 제어

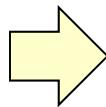


2014년 1월 22일

동양미래대학교
로봇자동화공학부
박 정 완

NGV 연구장학생 외에는 무단 전재, 복제, 배포를 금합니다.

기구학의 필요성



❖ 자동 주행 중 조향의 문제점

- 차선을 따라 안전 운행을 하기 위해서는 적정한 회전 반경을 유지가 필수
- 차량의 회전 반경을 알기 위해선 조향각(핸들의 조정)과 회전 반경 사이의 관계가 필요

❖ 조향각과 회전 반경의 관계를 어떻게 알 수 있는가?

- 차량의 운동(움직임) 해석이 선행되어야 한다.
- 물체의 운동을 해석하는 학문인 기구학 해석이 필요하다.

❖ 기구학이란?

- 운동하는 물체의 위치, 속도, 가속도를 해석하기 위한 학문
- 기구: 하나의 독립된 물체가 아닌 복수의 물체가 서로 연결되어 있는 것
- BUT, 기구학 해석은 물체들의 연결 방식에 따라 달라진다. 즉, 기구의 구조가 달라지면 해석 결과는 동일하지 않다.

전륜 조향 차량 기구학

❖ Ackermann Steering Model

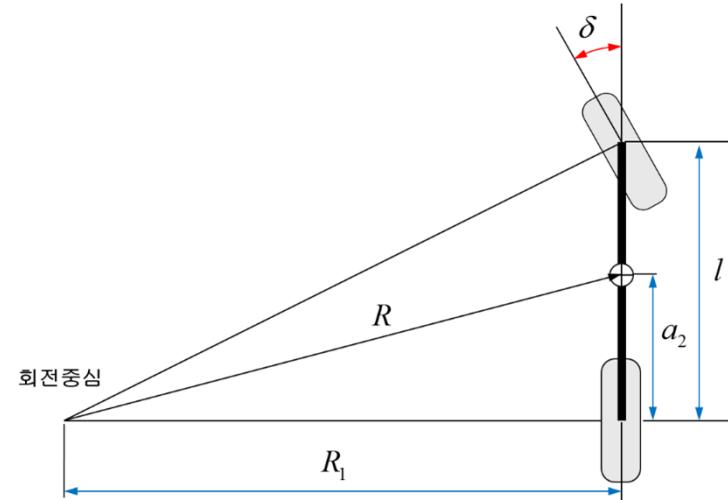
- 전륜 조향 차량의 조향 모델로 사용
- 타이어와 지면 사이의 마찰력이 충분히 커서 미끄럼이 없는 경우(저속 선회)
- 앞 바퀴 조향의 4륜 차량의 경우 선회 시 내륜과 외륜의 회전 중심이 일치하여야 하기 때문에 조향각은 서로 다르다.
- 두 앞바퀴의 조향각으로부터 차량의 등가 조향각을 구하면 다음과 같이 구할 수 있다.

$$\cot \delta = \frac{\cot \delta_i + \cot \delta_o}{2}$$

- 이때, 차량 중심의 회전 반경은 다음과 같다.

$$R = \sqrt{a_2^2 + l^2 \cot^2 \delta}$$

- 등가 조향각을 이용하여 4륜 차량을 등가 자전거 모델로 치환하여 이용한다.



Proof. $\tan \delta_i = \frac{l}{R_l - \frac{w}{2}}$

$$\tan \delta_o = \frac{l}{R_l + \frac{w}{2}}$$

$$\therefore \cot \delta = \frac{R_l}{l} = \frac{\cot \delta_i + \cot \delta_o}{2}$$



$$R_l = \frac{w}{2} + \frac{l}{\tan \delta_i} = -\frac{w}{2} + \frac{l}{\tan \delta_o}$$

$$2R_l = \frac{l}{\tan \delta_i} + \frac{l}{\tan \delta_o} = l(\cot \delta_i + \cot \delta_o)$$

전륜 조향 차량 기구학

❖ Ackermann Steering Model

- 진행 속도를 v 라 하면, 자전거 모델의 횡방향 속도는 다음과 같이 얻을 수 있다.

$$v_{lat} = v \tan \delta$$

- 자전거 모델의 후륜이 순간 회전 중심이므로 횡 방향 속도에서 회전 속도를 구할 수 있다.

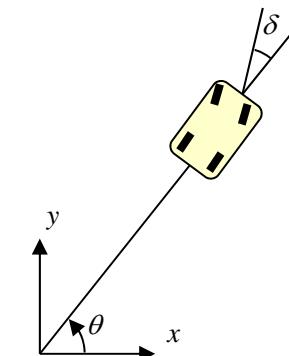
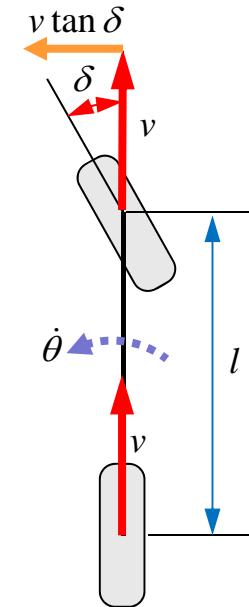
$$\dot{\theta} = \frac{v \tan \delta}{l}$$

- 위에서 구한 관계식을 이용하여 공간 상에서 차량의 운동을 표현하면 다음과 같은 관계를 얻을 수 있다.

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = v \frac{\tan \delta}{l}$$



✓ 순간 회전 중심: 임의의 순간 물체의 운동을 단순 회전 운동으로 표현할 수 있는 점

전륜 조향 차량 기구학

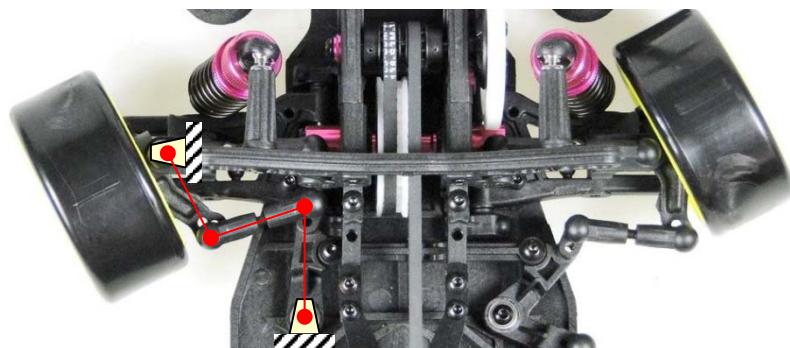
❖ 조향 장치

- 바퀴의 방향을 바꿔 주는 기구
- 조향 장치를 기구학 해석하여 조향각을 구할 수 있음
- 일반적인 차량의 경우 다음과 같은 rack & pinion gear box를 이용한 조향 장치가 주로 사용됨
- 휠과 rack 기어의 변위 관계와 rack기어와 양쪽 바퀴의 조향 각도와의 관계를 구하면 최종적으로 차량의 조향각을 구할 수 있음



❖ 조향 장치의 기구학 해석

- 무선 조종 자동차의 경우



Rack gearbox steering mechanism



전륜 차량 조향 기구학

❖ 조향 장치 기구학

■ Vector loop equation

$$\vec{l}_1 + \vec{l}_2 + \vec{l}_3 = \vec{x}_o + \vec{y}_o$$

$$l_1 \cos \theta_1 + l_2 \cos \theta + l_3 \cos \theta_3 = x_o$$

$$l_1 \sin \theta_1 + l_2 \sin \theta + l_3 \sin \theta_3 = y_o$$

$\theta_1 = \theta_{1_{initial}} + \square \theta_1$ 일 때, $\theta_3 = \theta_{3_{initial}} + \square \theta_3$ 는 얼마인가?
Input Output $\delta_{i(oro)}$

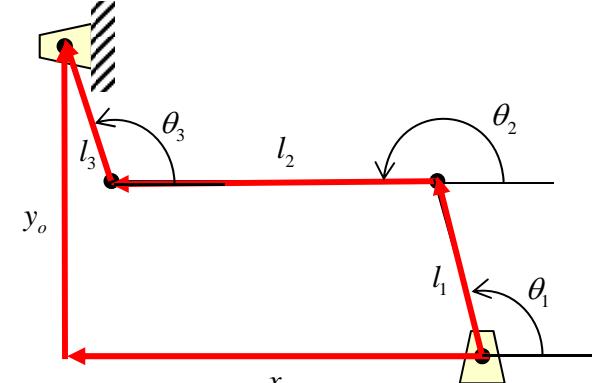
$$A = x_o - l \cos \theta_1$$

$$B = y_o - l_1 \sin \theta_1$$

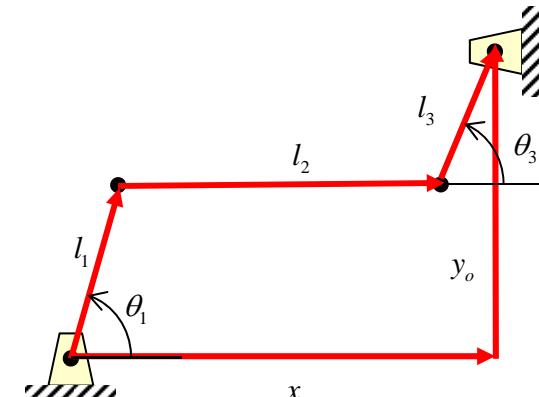
$$C = A^2 + B^2 + l_3^2 - l_2^2$$

Left side $\theta_3 = 2 \tan^{-1} \left\{ \frac{2Bl_3 - \sqrt{(2Bl_3)^2 - (C - 2Al_3)(C + 2Al_3)}}{C + 2Al_3} \right\}$

Right side $\theta_3 = 2 \tan^{-1} \left\{ \frac{2Bl_3 + \sqrt{(2Bl_3)^2 - (C - 2Al_3)(C + 2Al_3)}}{C + 2Al_3} \right\}$



Left side



Right side

전륜 차량 조향 기구학

❖ Example

- 바퀴의 폭이 $150mm$, 전후륜 사이의 거리는 $205mm$, 조향 장치의 링크 길이가 각각 $l_1 = 25mm$, $l_2 = 60mm, l_3 = 15mm$ 인 무선 조종 자동차가 있다. 이때, 서보 모터를 10° 회전 시켰을 때, 자동차의 조향각과 회전 반경은 얼마인가?

초기 각도

Left $\theta_1 = 100^\circ, \theta_2 = 180^\circ, \theta_3 = 105^\circ$

Right $\theta_1 = 80^\circ, \theta_2 = 0^\circ, \theta_3 = 75^\circ$

Left

$$A = -59.6730$$

$$B = 15.6168$$

$$C = 429.7487$$

$$\theta_3 = 93.7631^\circ$$

$$\delta_{left} = \theta_3 - \theta_{3_{initial}} = 93.7631^\circ - 105^\circ = -11.2369^\circ$$

Right

$$A = 68.2235$$

$$B = 14.1091$$

$$C = 1478.5107$$

$$\theta_3 = 56.6593^\circ$$

$$\delta_{right} = \theta_3 - \theta_{3_{initial}} = 56.6593^\circ - 75^\circ = -18.3406^\circ$$

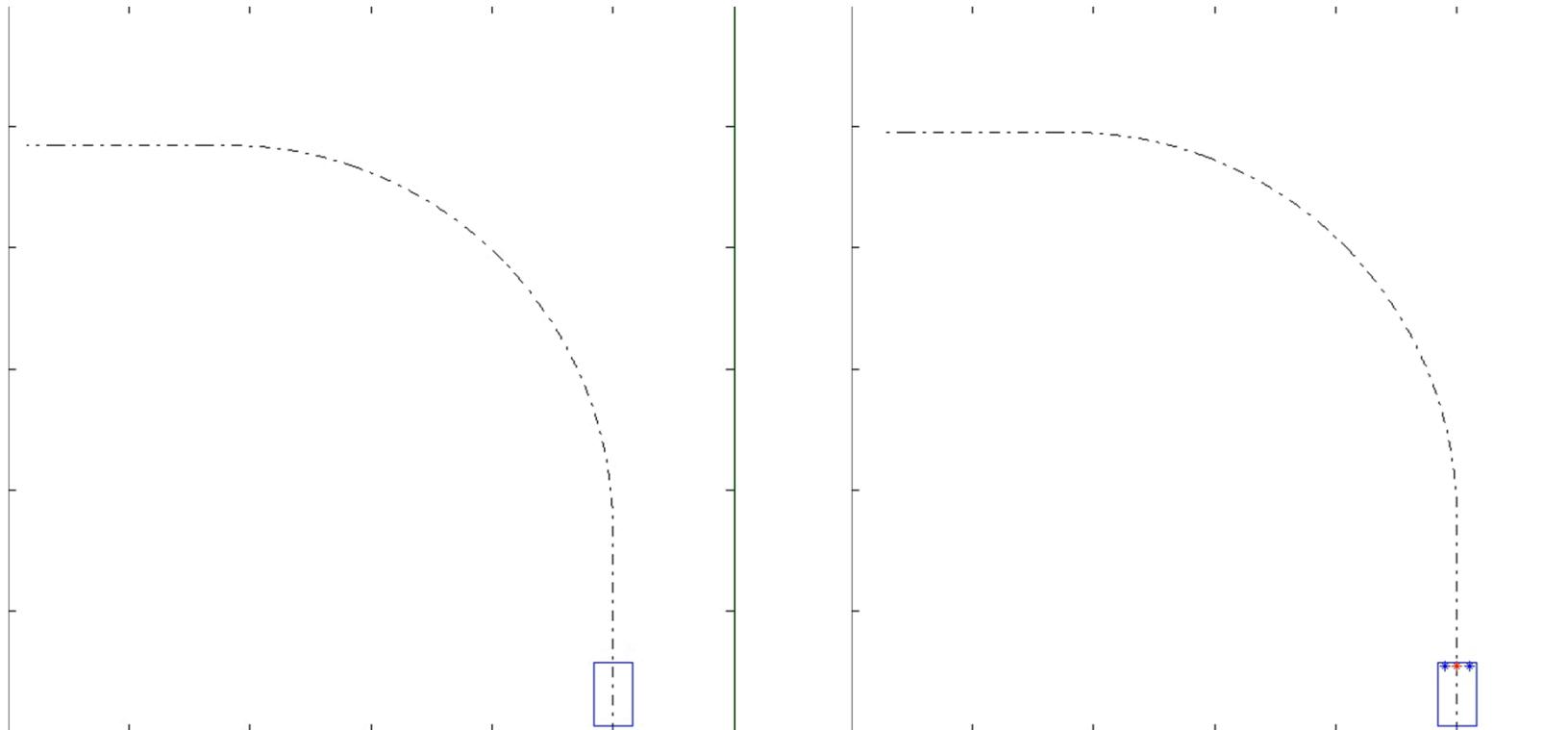


$$\delta = \tan^{-1} \left\{ \frac{2}{\cot \delta_{left} + \cot \delta_{right}} \right\} = -13.9526^\circ$$

$$R = \sqrt{\left(\frac{l}{2} \right)^2 + l^2 \cot^2 \delta} = 831.4536mm$$

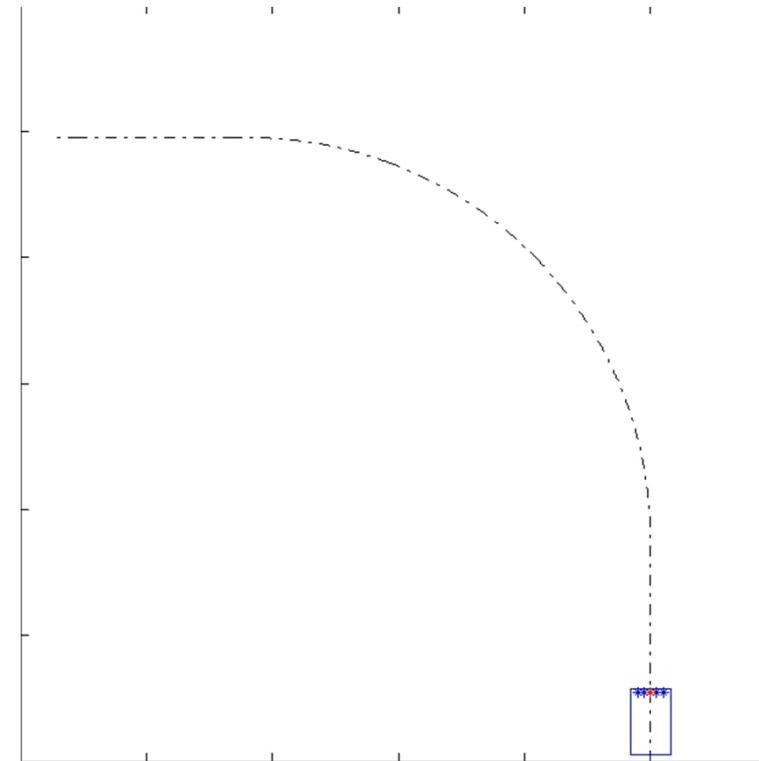
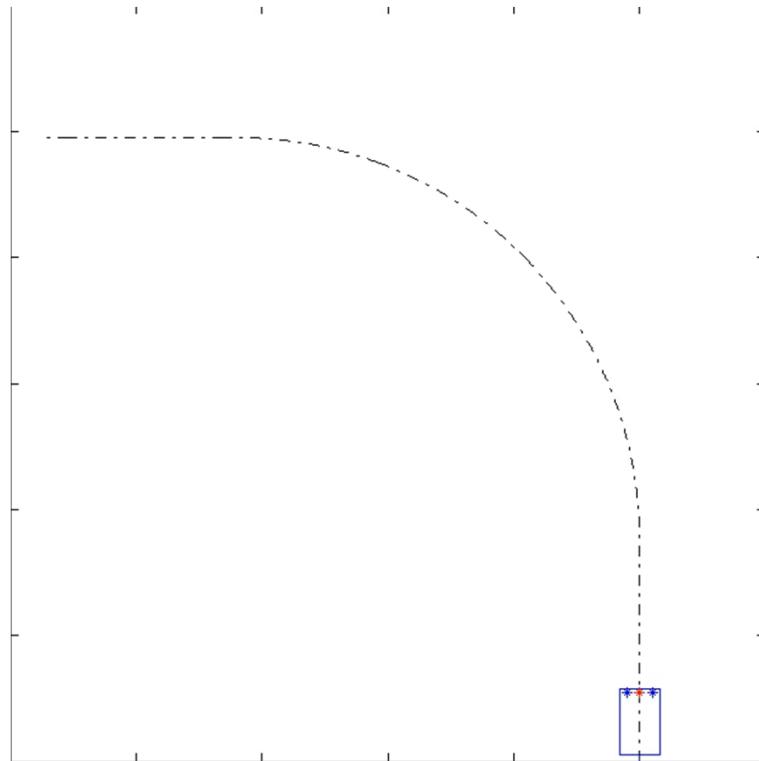
시뮬레이션

❖ 센서 종류에 따른 결과



전륜 차량 조향 기구학

❖ 센서 개수에 따른 결과



전륜 차량 조향 기구학

❖ 센서 위치에 따른 결과

