1. OpenCM API Reference 매뉴얼

OpenCM9.04의 API Reference는 Mt. San Antonio College의 Physics and Engineering Martin Mason 교수님의 CM-900 API Reference 문서를 기반으로 제작되었습니다. Martin Mason 교수님의 OpenCM 시리즈 발전에 대한 기여(Contribution)에 대해 매우 감사합니다

① OpenCM 코드 구조

전통적인 펌웨어 코드는 main.c 또는 main.cpp안에 void main()에서 프로그램이 시작됩니다.

```
#include <stdio.h>

void main(){

        board_Init();

        …

        while(1){

        …

        }

}
```

기본적으로 하드웨어 관련 초기화를 board_Init()같은 함수를 만들어서 일괄적으로 수행하고 구현하려고 하는 코드를 while(1){}이나 for(;;){}같은 무한 루프 안에서 구현합니다.

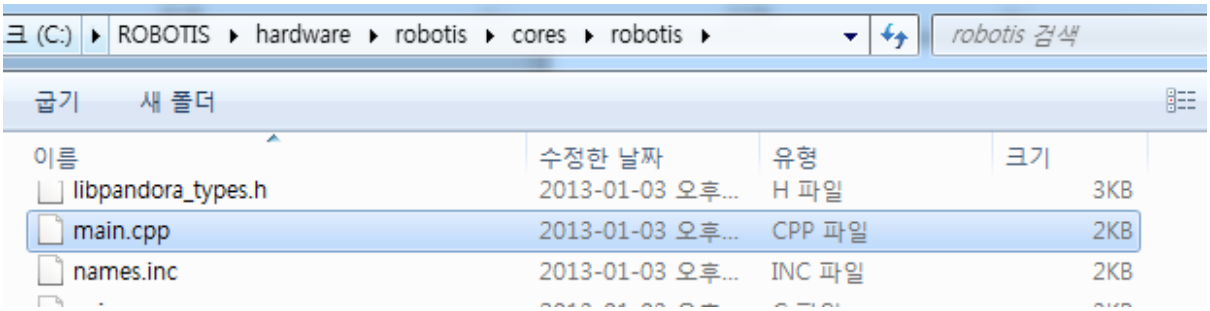
결국 하드웨어 초기화 하는 부분과 루프로 나눌 수 있는데 OpenCM의 소스 구조는 이러한 방식을 따릅니다.

여기서 setup(){}안에서 하드웨어 초기화를 수행하고 시작할 때 단 한번 수행을 합니다. Loop(){}는 계속해서 반복해서 사용자 코드를 실행합니다.

```
void setup(){
        ...//하드웨어 초기화
}
void loop(){
        ...//사용자 코드
}
```

결국 이러한 setup()과 loop()는 아래의 경로의 main.cpp파일안에 함수의 원형이 들어 있고 사용자는 setup()과 loop()함수를 구현함으로써 하나의 다운로드 가능한 Binary 파일을 만듭니다.

ROBOTIS₩hardware₩robotis₩cores₩robotis₩main.cpp



위의 main.cpp를 열어보면...

```
// Force init to be called *first*, i.e. before static object allocation.
// Otherwise, statically allocated objects that need libmaple may fail.
#include "Pandora.h"

__attribute__(( constructor )) void premain() {
    init();
}
int main(void) {
    setup();

    while (1) {
        loop();
    }
    return 0;
}
```

② 다이나믹셀 API

다이나믹셀은 Dynamixel class를 통해 모든 컨트롤을 할 수 있습니다. 다이나믹셀을 구동하기 위해서는 반드시 Dxl.begin()를 선언해야 합니다.

Dynamixel class의 모든 method는 기존 ROBOTIS Dynamixel SDK 기반으로 만들어 졌으며 모든 호환성을 고려해서 함수 이름에 따른 기능이 동일 합니다.

다이나믹셀에 대해 좀 더 많은 명령어는 아래의 ROBOTIS의 E-manual를 참고 바랍니다.

http://support.robotis.com/

Methods:

Device Control Methods	void begin(int baud)	지정된 baud 속도로 다이나믹셀을 초기화합니다.
	void end(void)	다이나믹셀의 사용을 중지합니다.
High Level Communications	int readByte(int id, int address)	해당 ID 의 다이나믹셀로 한 바이트 읽습니다.
	void writeByte(int id, int address, int value)	해당 ID 의 다이나믹셀로 한 바이트 씁니다.
	int readWord(int id, int address)	해당 ID 의 Address 에 연속된 2 바이트를 읽어 옵니다.
	void writeWord(int id, int address, int value)	해당 ID 의 Address 에 연속된 2 바이트를 씁니다.
	void ping(int id)	해당 ID 의 다이나믹셀의 연결상태를 확인합니다.
	void reset(int id)	해당 ID 의 다이나믹셀을 리셋 합니다.
	int getResult(void)	마지막 명령어에 대한 응답을 받습니다.
	void setPosition(int Servoid, int Position, int Speed)	해당 ID 의 위치와 속도를 설정합니다.

Packet Methods	void setTxPacketId(int id);	
	void setTxPacketInstruction(int instruction)	
	void setTxPacketParameter(int index, int value)	
	void setTxPacketLength(int length)	
	int getRxPacketParameter(int index)	
	int getRxPacketLength(void)	
	int getRxPacketError(int errbit)	
Utility methods	int makeWord(int lowbyte, int highbyte)	
	int getLowByte(int word)	
	int getHighByte(int word)	
Low Level Communications	void txPacket(void)	
	void rxPacket(void);	
	void txrxPacket(void)	

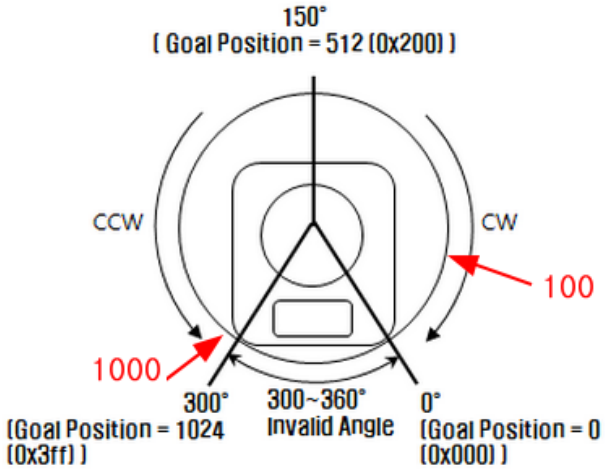
Packet관련 메소드, Utility 메소드, 저수준 메소드관련 설명은 로보티즈 E-manual을 이용 바랍니다.

http://support.robotis.com

다이나믹셀 시작하기(Getting Started):

아래의 스케치 코드는 다이나믹셀을 1Mbps속도로 초기화 하고 단순히 100위치와 1000위치 사이를 1초 간격으로 반복적으로 왔다갔다하는 예제입니다.

```
void setup() {
   // 다이나믹셀을 1Mbps의 속도로 초기화 합니다.
   // 다이나믹셀 baud rate 테이블은 support.robotis.com에서 확인할 수 있습니다.
   Dxl.begin(1);
}

void loop() {
   delay(1000);            // 1초 대기
   Dxl.writeWord(1, 30, 100); //1번 ID의 다이나믹셀을 100위치로 이동
   delay(1000);            // 1초 대기
   Dxl.writeWord(1, 30, 1000);//1번 ID의 다이나믹셀을 1000위치로 이동
}
```



코드 설명(Explaining the code):

모든 프로그램은 반드시 setup()과 loop()가 있어야 합니다.

Setup: setup()은 OpenCM9.04의 전원이 인가 할 때 혹은 리셋 버튼을 눌렀을 때 단 한번 수행되는 함수 입니다. 핀 모드 설정, 장치 초기화는 여기에서 수행하는 것이 좋습니다.

- Dxl.Begin() Dxl 은 미리 선언된 클래스 인스턴스이며, begin()메소드는 다이나믹셀 통신 버스(Serial0)를 초기화 합니다. Dxl 클래스 인스턴스는 begin()메소드 외에서 많은 메소드를 포함하고 있습니다.

Loop: loop()함수는 OpenCM9.04의 setup()을 수행하고 계속 반복적으로 수행되는 함수입니다.

- Dxl.writeword(1,30,100) ID 1 번 다이나믹셀에 100 위치로 이동을 수행합니다. 두 번째 인자 30 은 다이나믹셀 Goal Position 주소를 의미하고 다른 컨트롤 주소는 다이나믹셀 모델에 따른 아래의 컨트롤 테이블을 참고 바랍니다.(support.robotis.com)

30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-

delay (time in milliseconds) : 지정된 시간만큼 Delay를 수행합니다.

미리 선언된 상수들(Pre-defined constants):

일부 상수들은 미리 선언되어 있으므로 따로 정의할 필요 없이 바로 사용 가능합니다.
아래의 다이나믹셀 상수들은 미리 선언된 상수이므로 참고 바랍니다.

Get Result Flags

Name	DEC
COMM_TXSUCCESS	0
COMM_RXSUCCESS	1
COMM_TXFAIL	2
COMM_RXFAIL	3
COMM_TXERROR	4
COMM_RXWAITING	5
COMM_RXTIMEOUT	6
COMM_RXCORRUPT	7

Instruction Commands

Name	Hex
INST_PING	0x01
INST_READ	0x02
INST_WRITE	0x03
INST_REG_WRITE	0x04
INST_ACTION	0x05
INST_RESET	0x06
INST_DIGITAL_RESET	0x07
INST_SYSTEM_READ	0x0C
INST_SYSTEM_WRITE	0x0D
INST_SYNC_WRITE	0x83
INST_SYNC_REG_WRITE	0x84

Packet Instructions

Name	DEC
BROADCAST_ID	254
DEFAULT_BAUDNUMBER	1
ID	2
LENGTH	3

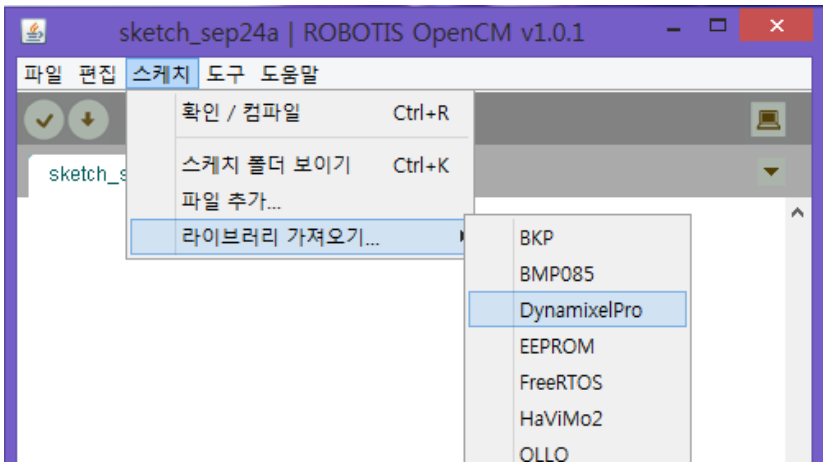
INSTRUCTION	4
ERRBIT	4
PARAMETER	5
MAXNUM_RXPARAM	60
MAXNUM_TXPARAM	150

Error Messages

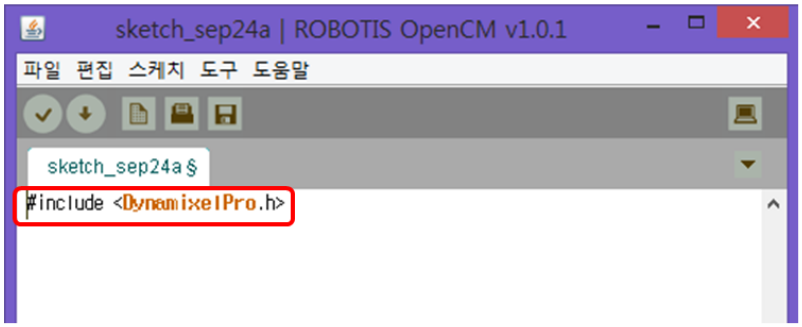
Name	DEC
ERRBIT_VOLTAGE	1
ERRBIT_ANGLE	2
ERRBIT_OVERHEAT	4
ERRBIT_RANGE	8
ERRBIT_CHECKSUM	16
ERRBIT_OVERLOAD	32
ERRBIT_INSTRUCTION	64

라이브러리 폴더의 dynamixel_address_tables.h를 Include하면 더욱 풍부한 상수들을 바로 사용 가능합니다. 로보티즈에서 제공하는 센서 장치들 외에도 다른 Vendor 장치(HaViMo) 상수 테이블도 포함되어 있으니 참고하세요

`\ROBOTIS\libraries\DynamixelPro\dynamixel_address_tables.h`



라이브러리 가져오기...-> DynamixelPro를 선택하면 아래와 같이 자동으로 #include 됩니다.



Function Documentation: ROBOTIS e-Manual v1.11.00 에서 참고하였습니다.

지금부터 기술하는 함수들은 모두 Dynamixel 클래스의 Member Method 입니다. Dynamixel 클래스는 미리 Dxl 이라는 인스턴스로 선언되어 있으며 Dxl.메소드이름(인자 1, 인자 2,…) 방식으로 사용하면 됩니다.

Ex) Dxl.begin(1); // 다이나믹셀 버스를 1Mbps 로 초기화 합니다.

Dxl.writeWord(2,30,512); // 2 번 ID 를 가진 다이나믹셀에 30 번 명령어 주소에 512 값을 씁니다.

void begin(int baud);

다이나믹셀 버스를 초기화 합니다.

Parameters

- baud

Baud 인자를 통해서 원하는 속도로 다이나믹셀 버스를 초기화 합니다. 계산식은 200000 / (Value + 1)와 같고 int baud는 Value와 같은 값입니다. 아래의 테이블은 다이나믹셀 Baud rate에 따른 Value 값입니다. 예를 들어 Dxl.begin(1)로 초기화를 수행하면 다이나믹셀 버스 속도는 1000000 bps가 되고 Dxl.begin(34);이면 버스 속도는 57600 bps가 됩니다.

Value	Actual BPS	Standard BPS	Uncertainty
0	2000000	2000000	0
1	1000000	1000000	0%
3	500000	500000	0%
4	400000	400000	0%
7	250000	250000	0%
9	200000	200000	0%
16	117647	115200	-2.124%
34	57142	57600	0.794%
103	19230	19200	-.16%
207	9615	9600	-.16%

기본적으로 다이나믹셀은 1Mbps 속도를 많이 씁니다.

Return Values

- 반환되는 Value 가 1 이면 성공을 의미하고 0 이면 실패를 의미합니다.

---

void end(void);

다이나믹셀 사용을 중지합니다.

---

int readByte(int id, int address);

해당 ID의 0다이나믹셀 address에 있는 데이터를 1바이트 읽습니다. 통신 과정에서의 결과는 getResult()함수로 확인할 수 있습니다.

Parameters

- id

제어 하려는 다이나믹셀 ID

- address

다이나믹셀의 컨트롤 레지스터 주소이며 원하는 내용을 support.robotis.com에서 다이나믹셀의 모델별로 컨트롤 테이블을 찾아 볼 수 있습니다.

Return Values

- 읽어 온 데이터 값

Example

```
data = Dxl.readByte( 2, 36 );
if( Dxl.getResult( ) == COMM_RXSUCCESS )
{
    // using data
}
```

---

void writeByte(int id, int address, int value);

다이나믹셀의 특정 컨트롤 레지스터 address에 1 Byte를 씁니다..

Parameters

- id

제어하려는 다이나믹셀의 ID

- address

컨트롤 레지스터 Address 입니다. 가지고 계신 다이나믹셀의 컨트롤 테이블을 확인하신 후 사용 바랍니다.

- value

다이나믹셀에 쓸려는 값입니다.

Return Values

- 없습니다.

Example

```
Dxl.writeByte( 2, 19, 1 );
if( Dxl.getResult( ) == COMM_RXSUCCESS )
{
    // Succeed to write
}
```

---

int readWord(int id, int address);

한 개의 word 단위는 2 바이트에 해당하고, 한 개의 word 단위로 해당 ID 의 다이나믹셀 컨트롤 레지스터 주소에서 읽기 동작을 수행 합니다. 이 함수로 읽기 동작을 수행하면 컨트롤 레지스터 주소와 연속하는 레지스터 주소에 있는 값을 읽어 들입니다. 예를 들면 30 번지 Goal Position address 에 있는 값 1 워드를 읽는다면 30 번지 Goal Position(L) 1 바이트 + 31 번지 Goal Position(H) 1 바이트 더해서 2 바이트(1 워드)를 읽게 됩니다. 마찬가지로 getResult()메소드로 통신 성공 여부를 알 수 있습니다.

Parameters

- id

제어하려는 다이나믹셀의 ID

- address

컨트롤 레지스터 Address 입니다. 가지고 계신 다이나믹셀의 컨트롤 테이블을 확인하신 후 사용 바랍니다

Return Values

- 읽어 들인 데이터(1 워드)를 반환합니다.

Example

```
data = Dxl.readWord( 2, 36 );

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

      // 이 부분에서 data 를 처리하면 좋습니다.

}
```

---

void writeWord(int id, int address, int value);

한 개의 word 단위는 2 바이트에 해당하고, 한 개의 word 단위로 해당 ID 의 다이나믹셀 컨트롤 레지스터 주소에서 쓰기 동작을 수행 합니다. 이 함수로 쓰기 동작을 수행하면 컨트롤 레지스터 주소와 연속하는 레지스터 주소에 있는 값을 씁니다. 예를 들면 30 번지 Goal Position address 에 있는 값 1 워드를 쓰고자 한다면 30 번지 Goal Position(L) 1 바이트 + 31 번지 Goal Position(H) 1 바이트 더해서 2 바이트(1 워드)를 쓰게 됩니다. 마찬가지로 getResult()메소드로 통신 성공 여부를 알 수 있습니다.

Parameters

- id

제어하려는 다이나믹셀의 ID

- address

다이나믹셀의 컨트롤하려는 명령어 레지스터 주소(Address)

- value

다이나믹셀에 쓰고자 하는 값입니다.

Return Values

- None

Example

```
Dxl.writeWord( 2, 30, 512 );
if( Dxl.getResult( ) == COMM_RXSUCCESS )
{
    // Write success
}
```

---

void ping(int id);

해당 다이나믹셀이 버스내에서 연결이 되어 있는지 확인합니다. 타임아웃을 가지고 있으며 Dxl.getResult()로 확인 가능합니다.

Parameters

- id

연결 확인하려는 다이나믹셀 ID

Return Values

- 반환값 없음

Example

```
Dxl.ping( 2 );
if( Dxl.getResult( ) == COMM_RXSUCCESS )
{
        // 다이나믹셀 ID 2 번에 대한 처리 수행
}
```

---

void reset(int id);

다이나믹셀을 리셋합니다.

Parameters

- id

리셋을 수행할 Dynamixel ID 입니다.

Return Values

- 없음

Example

```
Dxl.reset( 2 );
```

```
if( Dxl.getResult( ) == COMM_RXSUCCESS )
{
        // reset ID 2
}
```

---

int getResult(void);

마지막 명령어의 응답을 확인합니다.

Parameters

- None

Return Values

- 아래 테이블은 getResult()로 확인할 수 있는 응답의 종류를 나열하였습니다.

Value	Meaning
COMM_TXSUCCESS	Instruction packet 성공
COMM_RXSUCCESS	Status packet 수신 성공
COMM_TXFAIL	오류로 인해 Instruction packet 전송 실패
COMM_RXFAIL	오류로 인해 Instruction packet 수신 실패
COMM_TXERROR	Instruction Packet에 문제가 발생함
COMM_RXWAITING	Status Packet이 아직 도착하지 않음
COMM_RXTIMEOUT	해당 다이나믹셀이 응답하지 않음.
COMM_RXCORRUPT	Status Packet에 문제가 있음

Example

```
result = Dxl.getResult( );
if( result == COMM_TXSUCCESS )
```

```
{

}

else if( result == COMM_RXSUCCESS )

{

}

else if( result == COMM_TXFAIL )

{

}

else if( result == COMM_RXFAIL)

{

}

else if( result == COMM_TXERROR )

{

}

else if( result == COMM_RXWAITING )

{

}
```

void setPosition(int Servoid, int Position, int Speed);//Made by Martin S. Mason(Professor @Mt. San Antonio College)

주어진 속도(Speed)로 원하는 위치(Position)으로 이동합니다. 실제로 위치와 속도에 대한 레지스터 어드레스를 한번에 설정하는 편리한 래핑 함수입니다.

Parameters

- **Servoid** 제어하려는 다이나믹셀 ID입니다.

- **Position** 이동하려는 목표 위치값입니다. 다이나믹셀 모델에 따라 다를 수 있으니 E-manual을 참고 바랍니다.

-Speed 0~1023 사이의 숫자로 지정한 목표 속도입니다. 너무 낮은 속도 값은 다이나믹셀 모델에 따라 이동하지 않는 문제 발생할 수 있습니다. 자세한 내용은 E-manual을 참고 바랍니다.

Return Values

-없음

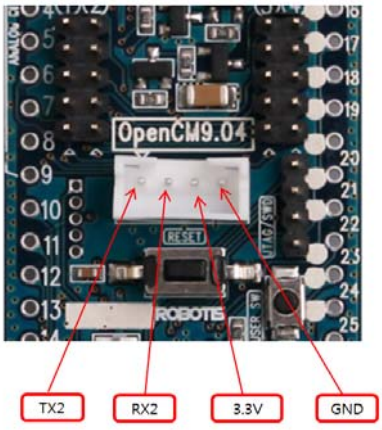
Example

```
result = Dxl.setPosition(3,500,600 );

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

        // Verify position command has been received

}
```

③ RC100 API

RC100은 OpenCM9.04의 장치와 원격 통신에 사용 될 수 있는 저전력 무선 통신입니다. 실제로, 아래의 4Pin 시리얼 통신 포트를 가진 통신 장치들은 OpenCM9.04의 4핀 포트에 바로 연결할 수 있습니다. 아래의 열거된 함수는 클래스 인스턴스 없이 그냥 함수 레벨로 사용합니다.

[BT-110A] or [BT-110A Set] [BT-210], [ZIG-110A Set] or [LN-101]



4핀 통신포트에 대한 자세한 핀 정보는 OpenCM9.04 하드웨어 사양을 참고하세요

Device Control Methods	int zgbInitialize(int devIndex)	지그비 장치를 57600bps 로 초기화합니다. devIndex 는 특별한 의미는 없으므로 0 으로 초기화 하면 됩니다.
	void zgbTerminate(void)	장치 사용을 중지합니다.
Data Methods	int zgbTxData(int data)	장치로 데이터를 전송합니다.
	int zgbRxCheck(void)	도착한 데이터가 있는지 확인합니다.
	int zgbRxData(void)	받은 데이터가 있으면 반환합니다.

Example:

In loop() checks for RC100 data reception.

```
#include <RC100.h>   // include RC100 library


RC100 Controller; //Instanciate RC100 class


void setup() {
  pinMode(BOARD_LED_PIN, OUTPUT);
  Controller.begin(); // init RC100 device on serial2 as 57600 bps


}
int RcvData =0;


void loop() {


  if(Controller.available()){ // if data is available from RC100
      RcvData = Controller.readData(); // read data
      SerialUSB.print("RcvData = ");
      SerialUSB.println(RcvData); // print the data


      if(RcvData & RC100_BTN_1) // if pressed button1 on RC100
        digitalWrite(BOARD_LED_PIN,LOW); // turn on LED


      delay(100);
```

```
    }
  digitalWrite(BOARD_LED_PIN,HIGH); // otherwise, turn off LED

}
```

④ 일반 IO 입출력(GPIO)

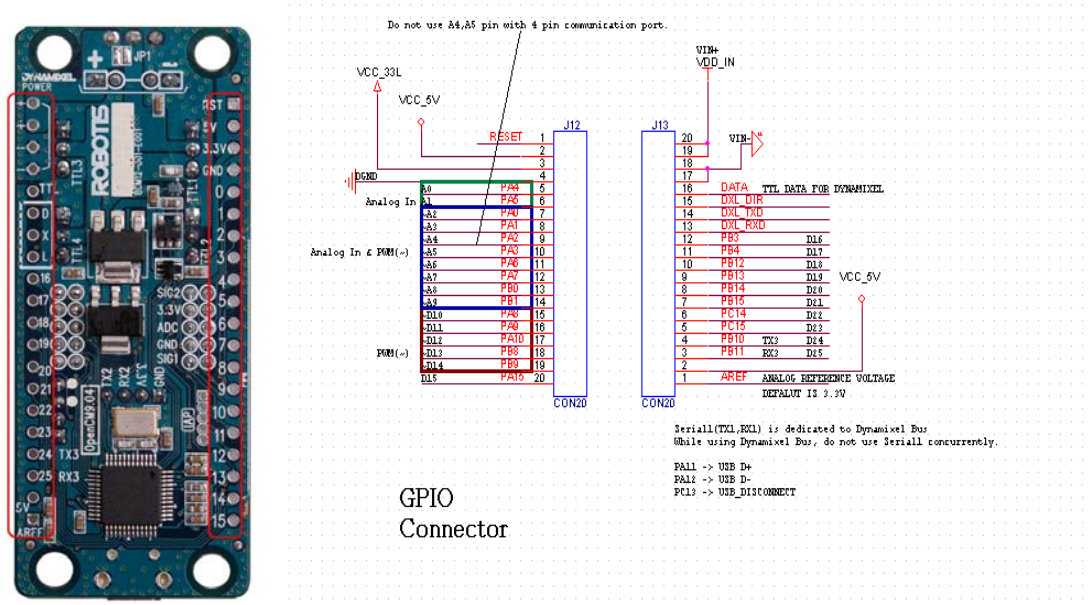
아래 함수들은 대부분 Arduino 와 Leaflabs의 Maple Language Reference로부터 도입되었습니다. 아두이노의 Reference는 쉽고 간단해서 복잡한 C/C++레벨의 ARM 코드보다 유용합니다. 아래의 사이트에서 아두이노 및 Leaflabs의 Reference API에 대한 문서의 내용을 확인하실 수 있습니다.

Arduino Reference : http://arduino.cc/en/Reference/HomePage
Leaflabs Maple Reference : http://leaflabs.com/docs/language.html

General Methods	pinMode(pin, WiringPinMode mode)	pin 을 입력(INPUT) 또는 출력(OUTPUT)으로 설정합니다.
Digital Methods	int digitalRead(pin)	특정 pin 의 High/Low 상태를 읽습니다. 단, 해당 pin 이 입력(INPUT)으로 설정되어야 합니다.
	digitalWrite(pin, value)	특정 pin 에 High 나 Low 를 씁니다. 단, 해당 pin 이 출력(OUTPUT)으로 설정되어야 합니다.
	togglePin(pin)	핀의 출력을 반대로 Toggle 합니다 예를 들면 Low 상태에서 High 로 High 에서 Low 로 토글합니다.
Analog Methods	int analogRead(pin)	아날로그 pin 의 값을 읽습니다. 단, 해당 pin 이 INPUT_ANALOG 으로 설정되어야 함.
	analogWrite(pin, duty cycle)	해당 pin 에 아날로그 출력을 합니다. pwmWrite()와 동일한 함수이고 duty cycle 에 0~65536 사이의 값으로 Duty cycle 을 조정할 수 있습니다.

Digital Methods로 표시된 함수들은 OpenCM9.04의 모든 핀(0~25번)에서 사용가능 합니다.

Analog Methods의 경우 OpenCM9.04의 PCB에 앞면에 새겨진 Analog IN, analogWrite(), pwmWrite()은 회로도에 표시된 핀만 가능합니다.



---

void pinMode(pin, mode) (adopted from Leaf Labs Maple Documentation)

pin을 mode를 변경합니다.

- `pin` -

  변경해야 될 pin, OpenCM9.04의 PCB 실크 번호를 확인하세요

Parameters:

- `mode` -

  아래의 나열된 Mode로 해당 pin을 변경할 수 있습니다.

Values:

- `OUTPUT` -

  Basic digital output: when the pin is HIGH, the voltage is held at +3.3v (Vcc) and when it is LOW, it is pulled down to ground.

- `OUTPUT_OPEN_DRAIN` -

  In open drain mode, the pin indicates “low” by accepting current flow to ground and “high” by providing increased impedance.

  An example use would be to connect a pin to a bus line (which is pulled up to a

positive voltage by a separate supply through a large resistor). When the pin is high, not much current flows through to ground and the line stays at positive voltage; when the pin is low, the bus “drains” to ground with a small amount of current constantly flowing through the large resistor from the external supply. In this mode, no current is ever actually sourced from the pin.

- `INPUT` -

  Basic digital input.

  The pin voltage is sampled; when it is closer to 3.3v (Vcc) the pin status is high, and when it is closer to 0v (ground) it is low. If no external circuit is pulling the pin voltage to high or low, it will tend to randomly oscillate and be very sensitive to noise (e.g., a breath of air across the pin might cause the state to flip).

- `INPUT_ANALOG` -

  This is a special mode for when the pin will be used for analog (not digital) reads.

  Enables ADC conversion to be performed on the voltage at the pin.

- `INPUT_PULLUP` -

  The state of the pin in this mode is reported the same way as with INPUT, but the pin voltage is gently “pulled up” towards +3.3v.

  This means the state will be high unless an external device is specifically pulling the pin down to ground, in which case the “gentle” pull up will not affect the state of the input.

- `INPUT_PULLDOWN` -

  The state of the pin in this mode is reported the same way as with INPUT, but the pin voltage is gently “pulled down” towards 0v.

  This means the state will be low unless an external device is specifically pulling the pin up to 3.3v, in which case the “gentle” pull down will not affect the state of the input.

- `INPUT_FLOATING` -

  Synonym for INPUT.

- `PWM` -

  This is a special mode for when the pin will be used for PWM output (a special case of digital output).

- PWM_OPEN_DRAIN -

  Like PWM, except that instead of alternating cycles of LOW and HIGH, the voltage on the pin consists of alternating cycles of LOW and floating (disconnected).

Discussion

pinMode()는 보통 setup() 함수 내에서 해당 핀의 설정을 변경할 때 수행되는 함수 입니다. 해당 핀을 쓰기 위해서는 반드시 선언해야 주어야 합니다.

Example

아래의 예제는 pinMode()를 통해 OpenCM9.04 에 기본으로 내장된 LED 를 OUTPUT 으로 설정하고 digitalWrite()함수로 반복적으로 껏다 켯다를 반복하는 Blink 예제입니다.

```
void setup() {
   pinMode(BOARD_LED_PIN, OUTPUT);     // sets the LED pin as output
}

void loop() {
   digitalWrite(BOARD_LED_PIN, HIGH);   // sets the LED on
   delay(1000);                        // waits for a second
   digitalWrite(BOARD_LED_PIN, LOW);    // sets the LED off
   delay(1000);                        // waits for a second
}
```

---

uint32 digitalRead(uint8 pin) (adopted from Leaf Labs Maple Documentation)

해당 핀으로부터 High/Low 상태를 읽습니다.

단, 이전에 반드시 해당 핀이 INPUT 이나 INPUT_PULLUP 또는 INPUT_PULLDOWN 으로 선언되어야 합니다. pinMode()함수를 참조하세요.

Parameters:

- pin -

  읽을 핀을 지정합니다.

Return: LOW 또는 HIGH.

Discussion

만약에 제어하려는 핀에 회로적으로 아무것도 연결이 되어 있지 않다면 HIGH 나 LOW가 랜덤하게 읽혀질 수도 있으니 참고하세요

Example

다음 예제는 버튼이 눌러지면 LED 가 껏다 켜졌다를 반복합니다.

```
void setup() {
  pinMode(BOARD_LED_PIN, OUTPUT);
  pinMode(BOARD_BUTTON_PIN, INPUT);
}

void loop() {
  int val = digitalRead(BOARD_BUTTON_PIN);   // reads the input pin
  togglePin(BOARD_LED_PIN);
}
```

---

void digitalWrite(uint8 pin, uint8 value) (adopted from Leaf Labs Maple Documentation)

해당 핀에 High/Low 를 출력합니다.

단, 이전에 반드시 해당 핀이 OUTPUT 이나 OUTPUT_PULLUP 또는 OUTPUT_PULLDOWN 으로 선언되어야 합니다. pinMode()함수를 참조하세요.

Parameters:

- `pin` -

  제어할 핀을 지정합니다.

- `value` -

  HIGH(1) 또는 LOW (0)

Discussion

핀 모드가 OUTPUT 으로 설정된 경우 HIGH 는 3.3V 가 되고 LOW 는 0V 가 됩니다.

Example

다음 예제는 이러한 digitalWrite()함수로 만들어진 LED Blink 예제 입니다. 자세한 설명은 위를 참고하세요.

```
void setup() {
  pinMode(BOARD_LED_PIN, OUTPUT);     // sets the digital pin as output
}

void loop() {
  digitalWrite(BOARD_LED_PIN, HIGH);   // sets the LED on
  delay(1000);                      // waits for a second
```

```
  digitalWrite(BOARD_LED_PIN, LOW);    // sets the LED off
  delay(1000);                      // waits for a second
}
```

아래와 같이 loop() 부분을 toggleLED()로 대체할 수 있습니다. 이 함수는 기본적으로 built-in LED 만 제어하도록 미리 준비된 함수 입니다.

```
void loop(){
        toggleLED();
        delay(1000);
}
```

또는 togglePin()이라는 함수로도 대체 가능합니다.

```
void loop(){
        togglePin(BOARD_LED_PIN);
        delay(1000);
}
```

---

uint16 **analogRead**(uint8 pin) (adopted from Leaf Labs Maple Documentation)

해당 핀으로부터 아날로그 값을 읽습니다.

이 기능은 ADC 변환이 있기까지 Block 상태로 있습니다. 그리고 반드시 핀 모드가 INPUT_ANALOG 상태로 되어 있어야 합니다.

Parameters:
- pin - 아날로그 값을 읽을 핀

Return: 변환된 전압값이 반환되고 그 범위는 12bit ADC 변환이 가지는 0~4095 에 해당합니다.

Discussion

지정된 아날로그 핀의 값을 읽습니다. OpenCM9.04 는 16 채널, 디지털 변환기 12 비트 아날로그가 포함되어 있습니다. 이 컨버터는 0 과 4095 사이의 정수 값으로 0 과 3.3 볼트 사이의 입력 전압을 매핑 할 수 있습니다. 그러나, 전체 정확도와 정밀도에 영향을 줄 수 있는 요인이 많으므로 이 부분을 고려하셔야 합니다.

이 함수를 호출하기 전에는 반드시 pinMode()함수를 이용해서 ANALOG_INPUT 으로 설정이 되어 있어야 합니다. 자세한 설명은 pinMode()함수를 참고하세요.

Parameter Discussion

함수의 인자로 넘겨주는 pin 은 아날로그 핀의 숫자를 의미합니다. OpenCM9.04 PCB 의 실크스크린(흰색 글자)에 그 숫자가 표시되어 있으며 ANALOG IN 이라고 표시된 영역 밑의 핀들이 이러한 기능을 가집니다.

Note

만약에 아날로그 핀에 회로적으로 아무것도 연결이 되어 있지 않으면 어떠한 값이 읽혀질지 알 수 없습니다.

Example

```
int analogPin = 3;     // Potentiometer wiper (middle terminal) connected
                    // to analog pin 3. outside leads to ground and +3.3V.
                    // You may have to change this value if your board
                    // cannot perform ADC conversion on pin 3.

int val = 0;          // variable to store the value read

void setup() {
  pinMode(analogPin, INPUT_ANALOG); // set up pin for analog input
}

void loop() {
  val = analogRead(analogPin);    // read the input pin
  SerialUSB.println(val);         // print the value, for debugging with
                              // a serial monitor
}
```

---

analogWrite(uint8 pin, uint16 duty_cycle) (adopted from Leaf Labs Maple Documentation)

주어진 핀에 지정된 duty_cycle 로 PWM 신호를 만듭니다. OpenCM9.04 는 아날로그 출력을 PWM 신호의 duty cycle 을 조정해서 만듭니다.

Duty cycle 은 반드시 최대값(0~65535)을 넘지 않도록 주의해야 합니다.

Parameters:

- `pin` -

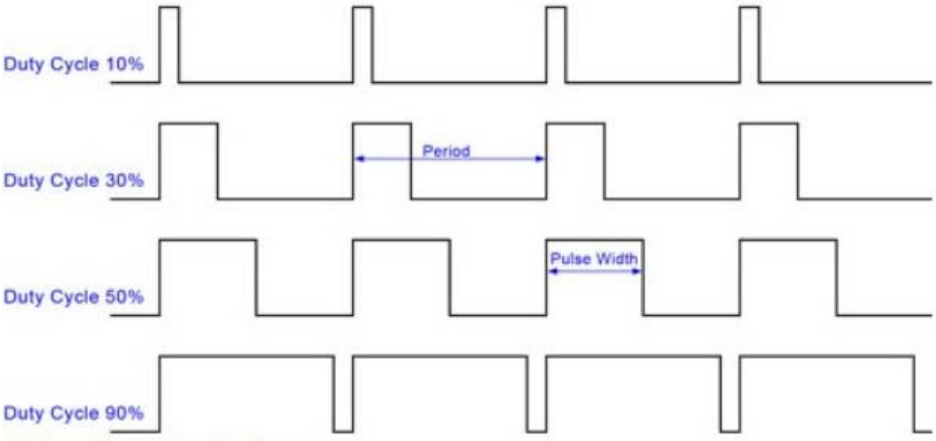
  PWM 을 입력할 핀

- `duty_cycle` -

  PWM 신호 duty cycle 비율, 0~65535

Duty cycle 인자에 0~65535 사이의 값을 넣어서 아래와 같은 비율로 Duty cycle을 컨트롤 할 수 있습니다.

0에 가까운 작은 값을 넣을수록 Duty Cycle이 작아지고(HIGH영역이 작아짐) 커질수록 아래의 그림처럼 Duty Cycle이 커집니다.(HIGH영역이 커짐)



Example

아래의 예제는 가변저항으로부터 읽은 값을 통해 LED 의 밝기를 조절하는 예제입니다.

```
int analogPin = 3;    // potentiometer connected to analog pin 3

void setup() {
  pinMode(BOARD_LED_PIN, OUTPUT);   // sets the LED pin as output

  pinMode(analogPin, INPUT_ANALOG); // sets the potentiometer pin as
                                    // analog input
}
void loop() {
  int val = analogRead(analogPin);       // read the input pin

  pwmWrite(BOARD_LED_PIN, val * 16);  // analogRead values go from 0
                                      // to 4095, pwmWrite values
                                      // from 0 to 65535, so scale roughly
}
```

---

void toggleLED**()** (adopted from Leaf Labs Maple Documentation)

OpenCM9.04 에 내장된 STATUS LED 를 토글하는 함수 입니다.

LED 가 켜져 있는 상태에서 호출하면 꺼지고, 꺼진 상태에서 호출하면 켜집니다.

Example

아래의 예제는 OpenCM9.04 의 STATUS LED 를 Blink 하는 예제 입니다.

```
void setup() {
    pinMode(BOARD_LED_PIN, OUTPUT);
}

void loop() {
    toggleLED();
    delay(100);
}
```

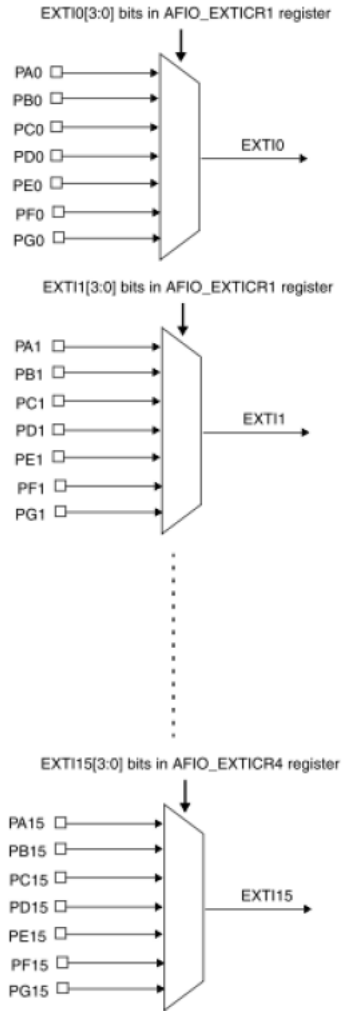
⑤ 인터럽트

인터럽트는 해당핀의 상태가 변화되는 것을 감지해서 특정 Action을 수행하는 기능입니다. 이러한기능은 하드웨어 타이머를 사용하기 때문에 특별히 계속해서 인터럽트가 일어나는지 확인하는 코드가 따로 필요한 것은 아닙니다. 모든 핀이 외부 인터럽트를 사용할 수 있지만 16개의 서로 다른 인터럽트를 넘을 수 없습니다. 예를 들면 0~15번 핀에 서로 다른 인터럽트 이벤트를 만드셨다면 더 이상의 인터럽트 이벤트 코드를 만들 수는 없습니다.

다음 함수들로 인터럽트를 제어할 수 있습니다.

`attachInterrupt`(*pin*, voidFuncPtr *handler*, *mode*)	특정 핀에 인터럽트 핸들러를 붙입니다.
`detachInterrupt`(*pin*)	특정 핀에 인터럽트를 핸들러를 제거합니다.
`noInterrupts()`	모든 인터럽트를 Disable합니다.
`Interrupts()`	모든 인터럽트를 허용합니다.
`disableDebugPorts()`	JTAG/SWD 옵션을 Disable합니다.
`enableDebugPorts()`	JTAG/SWD 옵션을 허용합니다.

`attachInterrupt`(uint8 *pin*, voidFuncPtr *handler*, ExtIntTriggerMode *mode*)



Adapted from Maple Documentation

Parameters

- `pin` – 핀 번호
- handler – 인터럽트 이벤트가 감지되었을 때 수행되는 함수 포인터입니다. 인자는 Void 이고 반환 인자도 void 입니다.
- mode – 인터럽트가 감지되는 형태입니다. Falling edge 또는 Rising edge 가 있을 수 있습니다.

mode

감지되는 인터럽트 이벤트 신호 유형입니다.

Values:

- `RISING` -

  LOW 에서 HIGH 로 변화될 때 트리거 됩니다.

- `FALLING` -

  HIGH 에서 LOW 로 변화될 때 트리거 됩니다.

- `CHANGE` -

  HIGH 에서 LOW 또는 LOW 에서 HIGH 같이 핀에 신호의 변화가 있을 때 무조건 인터럽트를 트리거 합니다.

Discussion

함수 내부의 인터럽트 처리 과정에서 실행되기 때문에, 지연 함수인 delay()는 작동하지 않습니다, 또한 millis ()에 의해 반환 된 값이 증가하지 않습니다. 시리얼 데이터를 수신하는 과정에서 데이터가 손실 될 수도 있습니다. 이러한 데이터 유실을 방지하려면 전역 변수에 반드시 Volatile 을 선언하셔야 합니다.

Example

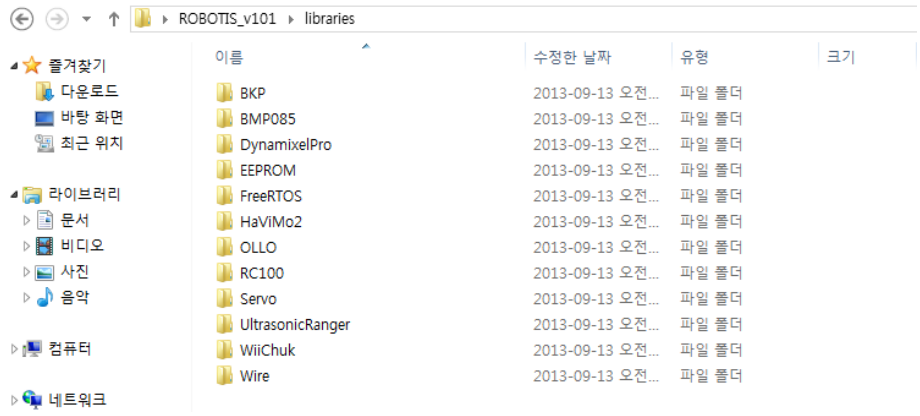
다음 예제는 0 번 핀에 신호의 변화가 있을 때마다 LED 가 켜지거나 꺼지는 예제 입니다. Blink() 함수처럼 함수 포인터를 이용해 attachInterrupt()를 사용하는 간단하면서도 좋은 예제입니다.

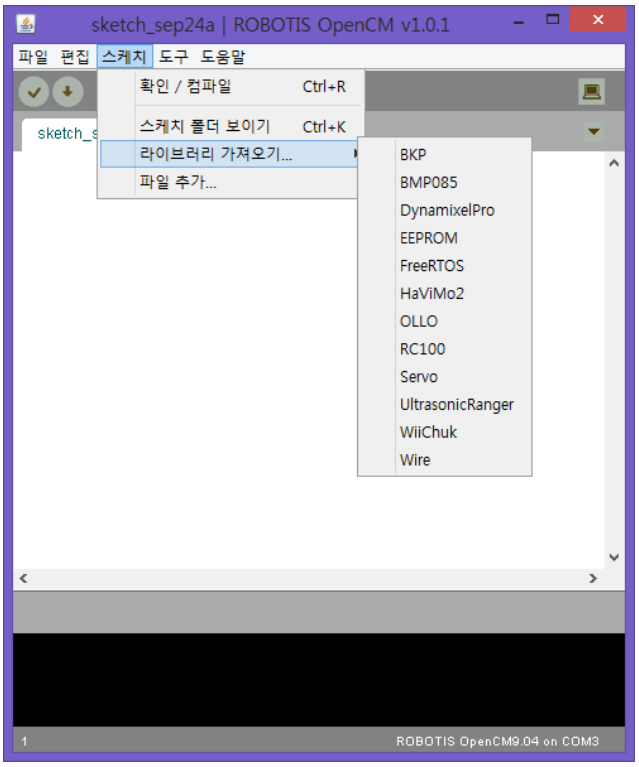
```
volatile int state = LOW; // must declare volatile, since it's
                      // modified within the blink() handler

void setup() {
   pinMode(BOARD_LED_PIN, OUTPUT);
   pinMode(0, INPUT);
   attachInterrupt(0, blink, CHANGE);
}

void loop() {
   digitalWrite(BOARD_LED_PIN, state);
}

void blink() {
   if (state == HIGH) {
      state = LOW;
   } else { // state must be LOW
      state = HIGH;
   }
}
```

예제에서, blink() 함수는 인터럽트 핸들러라고 할 수 있습니다. 0 번 핀의 입력신호가 변화할 때마다 blink()함수가 호출되고 state 변수 값도 따라서 HIGH 또는 LOW 로 변화합니다. 그러면 loop()함수에서 state 값에 따라 LED 가 꺼졌다가 켜졌다가를 반복하는 좋은 외부 인터럽트 예제 입니다.

⑥ 사용자 라이브러리/API 제작

ROBOTIS OpenCM이 실행되면 아래의 ₩ROBOTIS₩library폴더의 라이브러리들을 일괄적으로 가져옵니다. 또한 자기만의 라이브러리를 제작 배포할 수 있습니다. 이 페이지는 이러한 사용자 라이브러리를 제작 배포하는 방법을 기술합니다.



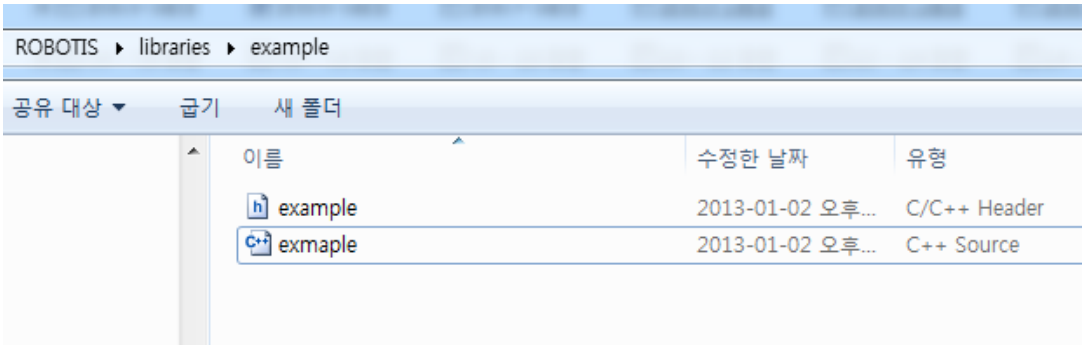


A. 기본 CPP기반 라이브러리 제작

라이브러리는 기본적으로 아래와 같이 CPP파일로 작성하고 API는 Core library를 그대로 쓰면 됩니다.

여기서는 example이라는 간단한 라이브러리로 실습해 보겠습니다.

아래와 같이 example.h라는 헤더파일과 example.cpp파일을 만들어주세요



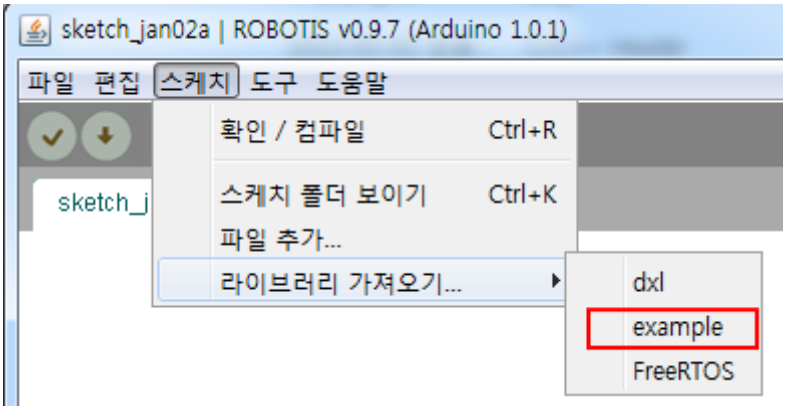
example.h에는 아래와 같이 wirish.h라는 헤더파일을 선언하면 digitalWrite()부터 다이나믹셀 API까지 OpenCM의 Core Library를 전부 사용하실 수 있습니다.

```
#include "wirish.h"

void setupHelloWorld(void);
void sendHelloWorld(void);
```

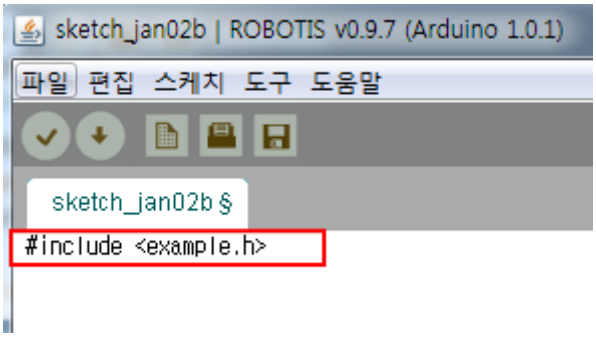
이제 example.cpp에서 setupHelloWorld()와 sendHelloWorld()함수를 구현하면 됩니다.

```
#include "example.h"

void setupHelloWorld(void){
  SerialUSB.begin();
}
void sendHelloWorld(void){
  SerialUSB.println("Hello World");
  delay(100);
}
```

그리고 이렇게 만든 example이라는 라이브러리를 아래와 같이 자동으로 현재 스케치 프로젝트에 Import시킵니다.



이렇게 자동으로 #include <example.h>가 포함 됩니다.

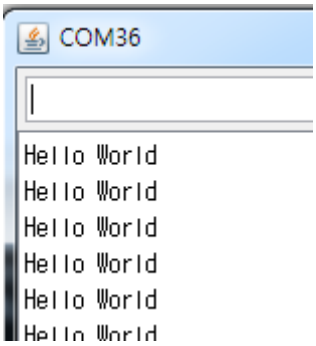


이제 위에서 구현했던 setupHelloWorld()와 sendHelloWorld()를 아래와 같이 자유롭게 쓰실 수 있습니다. 참고로 <>표시로 include하게 되면 ROBOTIS₩libraries디렉토리를 검색하게 됩니다. 만약 거기에 include할 파일이 없으면 에러가 발생하니 참고 바랍니다.

```
#include <example.h>

void setup(){

setupHelloWorld();

}

void loop(){

sendHelloWorld();

}
```

<스케치 코드>

이 코드를 컴파일 후 OpenCM에 다운로드 해보면 USB Serial을 통해 아래와 같은 시리얼모니터 출력을 볼 수 있습니다.

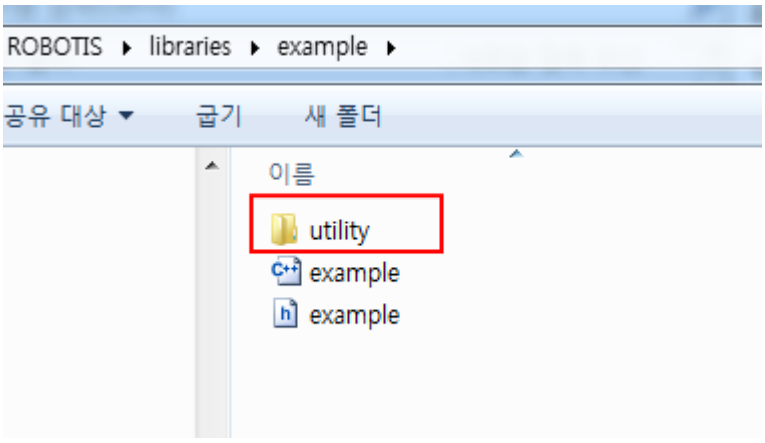


B. C기반 라이브러리 제작

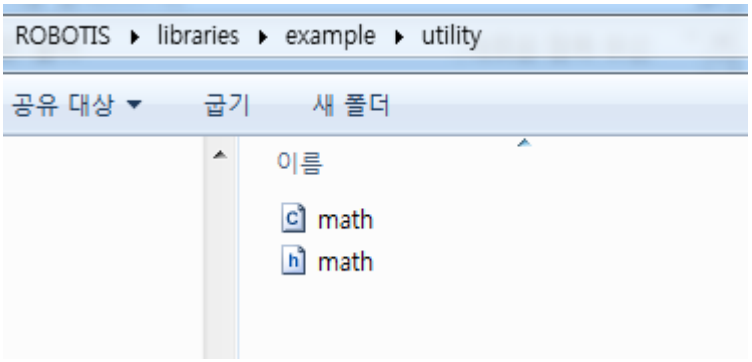
실제로 많은 펌웨어 라이브러리들은 C기반으로 제작된 것이 많이 있는데 이번 파트는 이렇게 C기반으로 제작된 라이브러리를 OpenCM 기반 라이브러리로 제작하는 방법에 대해 기술합니다.

여기서 C기반 라이브러리를 math.c와 math.h로 가정하고 만드는 실습을 진행합니다.

아래와 같이 내용이 없는 math.c, math.h 파일을 만들고 이전 과정에서 만들었던 example 라이브러리 폴더에서 utility라는 하위 디렉토리를 하나 만들어서 복사를 수행합니다.



아래의 utility 디렉토리 안에 복사를 수행합니다.



여기서 ROBOTIS OpenCM 프로그램은 방금 전에 만든 utility폴더 안에 있는 math.h같은 헤더파일은 자동으로 include하지 않고 example폴더 안의 헤더파일만 include되므로 참고 바랍니다.

이제 math.c와 math.h파일의 내용을 아래와 같이 채워 보겠습니다.

```
#include <stdio.h>

#ifdef   cplusplus
extern "C" {
#endif

int sum(int a, int b);

#ifdef   cplusplus
  }
#endif
```

<math.h>파일의 내용 입니다.

ROBOTIS OpenCM의 기본 툴체인인 Code Sourgery g++ Lite에 있는 stdio.h 표준 입출력 헤더파일로 sum()이라는 덧셈을 수행하는 함수를 만들어 보겠습니다. 여기서 <> 표시로 include하면 툴체인의 include폴더 안에 내용을 찾습니다. 고급 개발을 위해서는 툴체인의 include 디렉토리 안에 어떤 표준 함수가 있는지 살펴보는 것도 중요합니다.

여기서 extern “C” {}라는 전처리 과정이 중요한데 C++ 기반으로 컴파일을 할 때 C함수들은 extern “C”라는 prefix가 필요합니다.

```
#include "math.h"

int sum(int a, int b){

  return a+b;

}
```

<math.c>파일의 내용입니다.

이렇게 만든 C기반 sum()함수를 어떻게 쓰는지 살펴 보겠습니다.

전 단계에서 만든 example라는 라이브러리의 example.h파일과 example.cpp파일의 수정이 필요합니다.

아래와 같이 #include “utility/math.h”를 선언합니다.

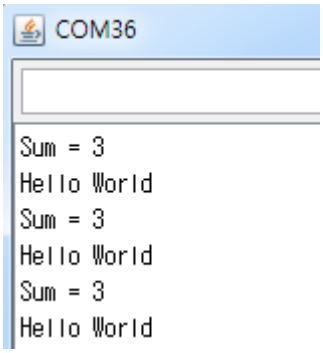
```
#include "wirish.h"
#include "utility/math.h"

void setupHelloWorld(void);
void sendHelloWorld(void);
```

이제 example.cpp에서 sum()함수를 쓸 수 있습니다.

```
#include "example.h"

void setupHelloWorld(void){
  SerialUSB.begin();
}
void sendHelloWorld(void){
  SerialUSB.println("Hello World");
  SerialUSB.print("Sum = ");
  SerialUSB.println(sum(1,2));
  delay(100);
}
```

이제 sendHelloWorld()가 들어간 아래의 스케치 코드의 출력은 sum(1,2) 값이 나올 것입니다.

```
#include <example.h>


void setup(){
  setupHelloWorld();
}
void loop(){
  sendHelloWorld();
}
```

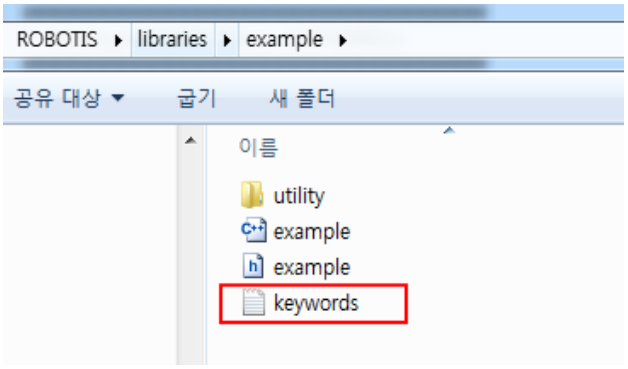


C. 라이브러리 문법 강조

이렇게 만들어진 API들도 스케치 코드에서 문법 강조 기능이 필요할 것입니다.

문법 강조 기능을 쓰기 위해서는 keywords.txt파일을 추가로 만들어 주어야합니다.

아래와 같이 example 라이브러리 디렉토리 안에 만들어 보겠습니다.



그 내용은 아래와 같이 입력하시면 됩니다.

```
########################################
# Syntax Coloring Map For CoOS
########################################

########################################
# Datatypes and Class (KEYWORD1)
########################################

Example KEYWORD1
########################################
# Methods and Functions (KEYWORD2)
########################################
setupHelloWorld KEYWORD2
sendHelloWorld  KEYWORD2

########################################
# Constants (LITERAL1)
########################################

Constants   LITERAL1
```

그리고 ROBOTIS OpenCM 프로그램을 재시작하면 이전 단계에서 만들었던 setupHelloWorld()와 sendHelloWorld() API를 타이핑했을 때 Color가 변하는 것을 확인 하실 수 있습니다.

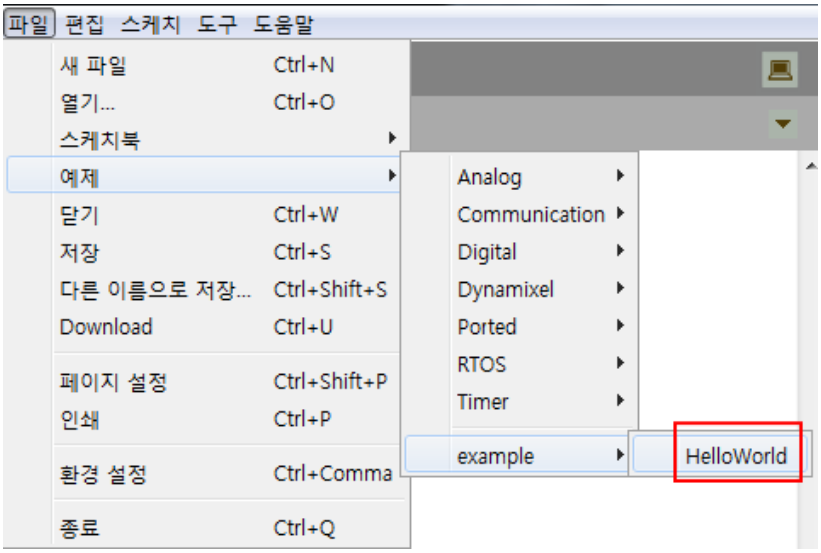
```
#include <example.h>

void setup(){

 setupHelloWorld();
}

void loop(){
  sendHelloWorld();
}
```

Example
Constants

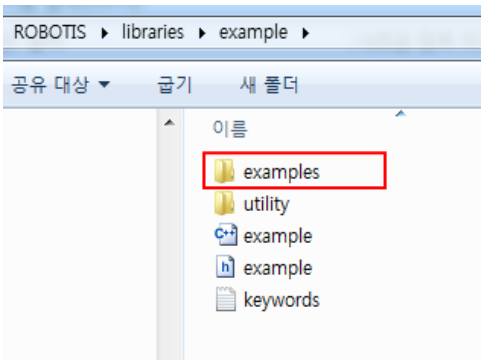
<문법 강조 기능 적용되었을 때>

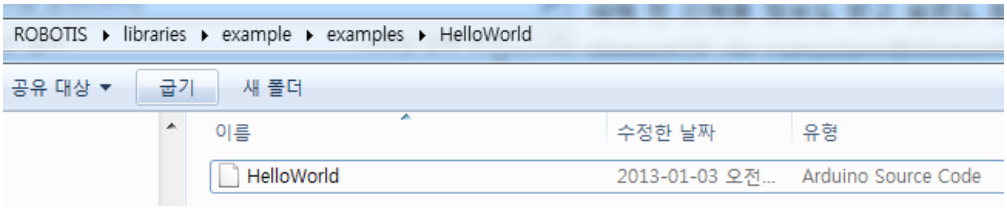
D. 라이브러리 예제 등록

지금까지 만든 라이브러리로 예제를 만들었다면 아래와 같이 파일 메뉴 아래의 예제 메뉴에 등록 시킬 수 있습니다.



예제 등록은 아래와 같이 이전 단계에서 만든 example 라이브러리 폴더에서 **examples**라는 디렉토리를 생성한 뒤에 스케치 코드를 넣으면 됩니다. 그리고 ROBOTIS OpenCM를 재시작하면 위의 화면과 같이 확인 할 수 있습니다.





반드시 위와 같이 스케치 코드 파일 이름(HelloWorld.ino)과 폴더 이름 HelloWorld을 일치시켜 줘야 재시작 했을 때 IDE에 등록이 됩니다.