1. OpenCM API Reference

This OpenCM904 API Reference documentation has been created by Martin Mason of Mt. San Antonio College of Physics and Engineering. Martin Mason. Thanks to Martin Mason for the development and contribution of the OpenCM series.

① OpenCM code sturcture

Usually, firmware codes begin with main.c, or main.cpp with void main().

```
#include <stdio.h>

void main(){

        board_Init();

        …

        while(1){

        …

        }

}
```

Default hardware initialization is in board_Init() function and code implemented in infinite while or for loops.

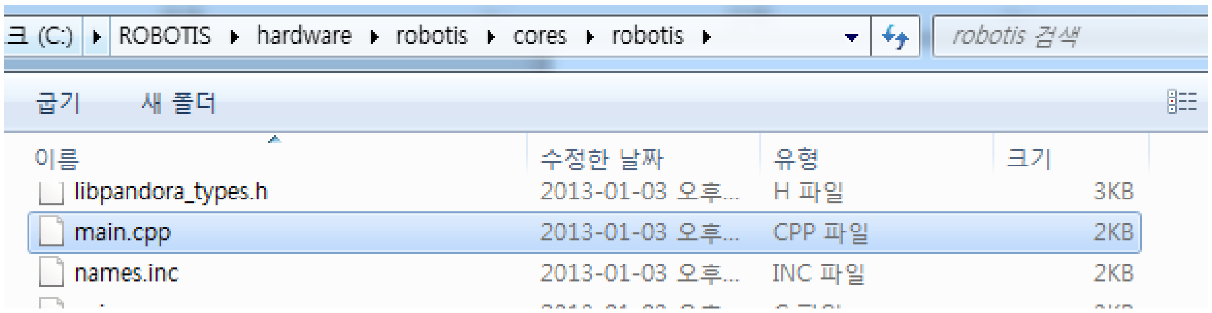
This way code structure of the OpenCM can be divided in hardware and parts.

Initialize hardware in setup(){}. Place algorithm under loop(){}.

```
void setup(){

        …//initialize hardware
}

void loop(){

        ...//user code

}
```

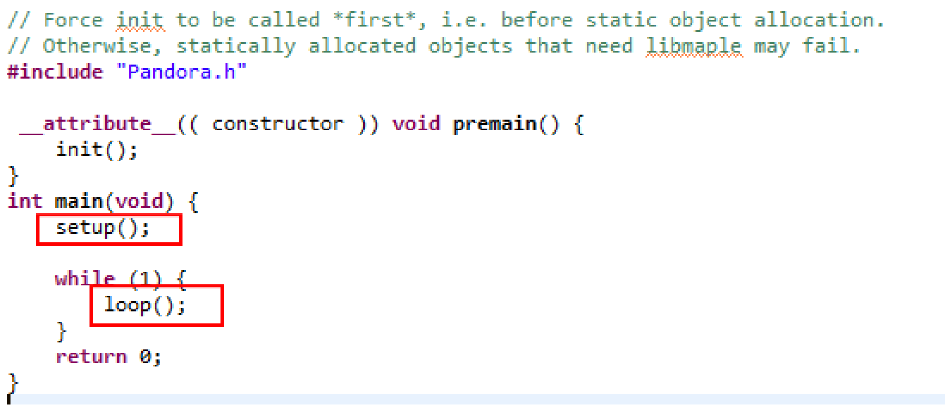
Both setup() and loop() reside in main.cpp; the user can create a downloadable binary file by implementing these.

ROBOTIS/hardware/robotis/cores/robotis/main.cpp



이름	수정한 날짜	유형	크기
libpandora_types.h	2013-01-03 오후...	H 파일	3KB
main.cpp	2013-01-03 오후...	CPP 파일	2KB
names.inc	2013-01-03 오후...	INC 파일	2KB

Open main.cpp...



```
// 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 Dynamixel API

Use Dynamixel class to control Dynamixel(s). To drive Dynamixel(s) Dxl.begin(baur_rate) is required.

Dynamixel class methods are based on Dynamixel SDK.

For more information on Dynamixel please consult the e-manuals.

.

http://support.robotis.com/

Methods:

Device Control Methods	void begin(int baud)	initialize Dynamixel at set baud rate.
	void end(void)	Pause Dynamixel.
High Level Communications	int readByte(int id, int address)	Read 1 byte from Dynamixel.
	void writeByte(int id, int address, int value)	Write 1 byte to Dynamixel.
	int readWord(int id, int address)	Read 1 word from Dynamixel
	void writeWord(int id, int address, int value)	Write 1word to Dynamixel.
	void ping(int id)	Verify Dynamixel connection status.
	void reset(int id)	Resets Dynamixel.
	int getResult(void)	Get response.
	void setPosition(int Servoid, int Position, int Speed)	Set position and velocity of Dynamixel ID.

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)	

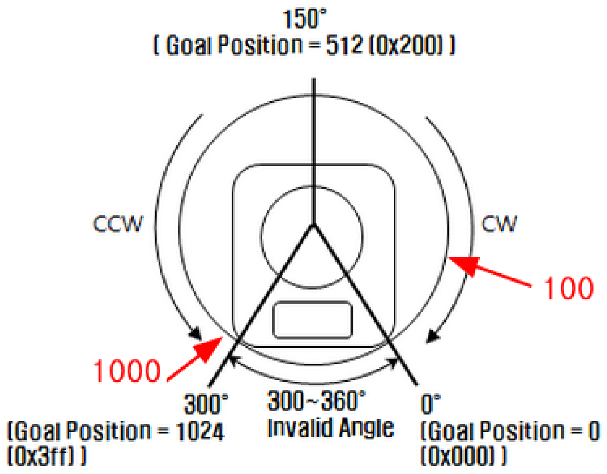
For more information on packet-related methods, utility method, low-level methods please consult the e-manuals.

http://support.robotis.com

Getting Started

The sketch code shown below sets a Dynamixel baud rate to 1Mbps with position switching between value 100 and 1000 with 1000ms pause in between.

```
void setup() {
   // sets Dynamixel baudrate to 1Mbps.
   // for values on baudrate visit support.robotis.com.
   Dxl.begin(1);
}

void loop() {
   delay(1000);            // wait for 1 second
   Dxl.writeWord(1, 30, 100); //set ID 1 to goal position of value 100
   delay(1000);            // wait for 1 second
   Dxl.writeWord(1, 30, 1000);//set ID 1 to goal position of value 1000
}
```



Code description

Every code requires setup() and loop().

Setup: setup() initializes the CM-900 upon power up or after pressing the reset button. Pin mode setup, device initialization also done in this function.

- Dxl.Begin()   Dxl assigned instance with begin() method initializes Dynamixel bus. Dxl class instance has more methods other than begin().

Loop: loop() runs setup() and repeatedly runs the OpenCM9.04.

- Dxl.writeword(1,30,100): the first value sets Dynamixel of ID 1. The second value sets goal position (30 of control table). For more information on control table please visit 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 (in milliseconds): set delay time.

Pre-defined constants

These predefined constants are convenient and make defining unnecessary. Please refer to the predefined constants listed below.

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

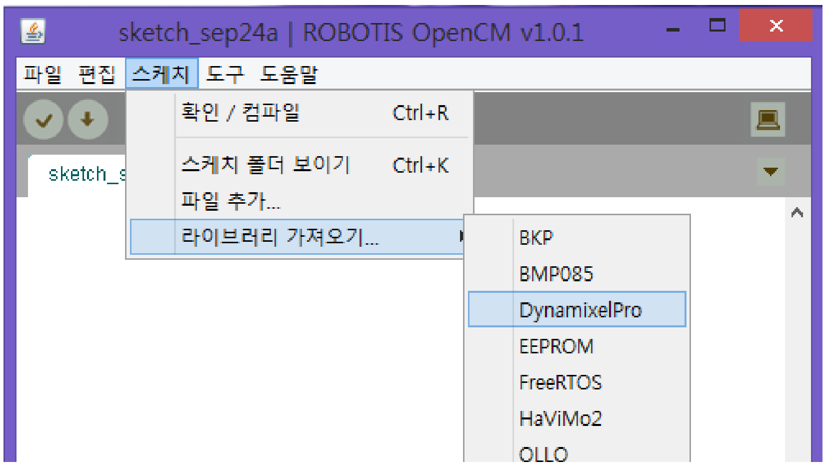
Name	DEC
BROADCAST_ID	254
DEFAULT_BAUDNUMBER	1
ID	2
LENGTH	3
INSTRUCTION	4
ERRBIT	4
PARAMETER	5
MAXNUM_RXPARAM	60
MAXNUM_TXPARAM	150

Error Messages

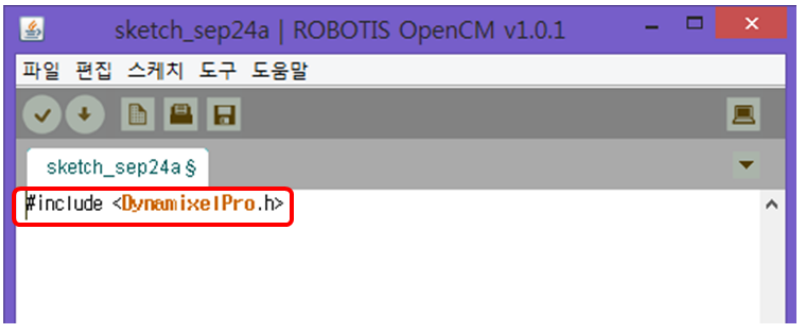
Name	DEC
ERRBIT_VOLTAGE	1
ERRBIT_ANGLE	2
ERRBIT_OVERHEAT	4
ERRBIT_RANGE	8
ERRBIT_CHECKSUM	16
ERRBIT_OVERLOAD	32
ERRBIT_INSTRUCTION	64

From the library folder including the header file dynamixel_address_tables.h provides an abundance of predefined constants. This also includes peripherals from ROBOTIS or third parties, such as HaViMo

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



Getting the library…-> select dxl and the #include will be listed (as shown below).


sketch_sep24a | ROBOTIS OpenCM v1.0.1
파일 편집 스케치 도구 도움말
sketch_sep24a §

```
#include <DynamixelPro.h>
```

Function Documentation: refer to ROBOTIS e-Manual v1.11.00.

Dynamixel class is assigned in Dxl instance.

Implement Dxl in this form: Dxl.instance(1st value, 2nd value,…).

Ex) Dxl.begin(1); // Initialize Dynamixel bus to 1Mbps.

Dxl.writeWord(2,30,512); // sets Dynamixel of ID 2 to goal position of value 512.

void begin(int baud);

Initialize Dynamixel bus.

Parameters

– baud

Baud is the value that determines communications speed. The relationship between baud rate and value is 200000 / (Value + 1). The following table lists the values and corresponding baud rates for each value. For example Dxl.begin(34) initializes Dynamixel bus with a baud rate of 57600bps.

.

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%

Default is 1Mbps.

Return Values

– A returned value of 1 means success; 0 for failure.

---

void end(void);

pauses Dynamixel.

---

int readByte(int id, int address);

Reads 1 byte of address data of Dynamixel. Use from results of communications getResult().

Parameters

- id

ID of Dynamixel

- address

Address from Dynamixel control table. For more information on Dynamixel control table visit support.robotis.com

.

Return Values

- read data value(s)

Example

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

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

    // using data

}
```

---

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

Writes 1 byte in address of Dynamixel control register.

Parameters

- id

ID of Dynamixel

- address

Address from Dynamixel control table. For more information on Dynamixel control table visit support.robotis.com

.

- value

Written data value(s).

Return Values

- none.

Example

```
Dxl.writeByte( 2, 19, 1 );

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

    // Succeed to write

}
```

---

int readWord(int id, int address);

A word is comprised of 2 bytes. A word is the control register address of the Dynamixel via its ID. Use this function to get a readout of the control register. For example, a read out of goal position (30 in address table) gives a word readout on for Goal Position (L) (30 in address table) and Goal Position (H) (31 in address table). Use getResult() method where communications is successful.

.

Parameters

– id

ID of Dynamixel

- address

Address value of Dynamixel from the control table

Return Values

- returns a word (2 bytes).

Example

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

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

        // process data here.

}
```

---

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

A word is comprised of 2 bytes. A word is the control register address of the Dynamixel via its ID. Use this function to write the control register. For example, a write goal position (30 in address table) then the writeword writes for Goal Position (L) (30 in address table) + Goal Position (H) (31 in address table). Use getResult() method where communications is successful.

Parameters

- id

ID of Dynamixel

- address

Address value of Dynamixel from the control table

- value

Value(s) to be written.

Return Values

– None

Example

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

---

void ping(int id);

Verifies Dynamixel bus for connected Dynamixel. Use Dxl.getResult() for verification.

Parameters

– id

ID of Dynamixel

Return Values

– none

Example

```
Dxl.ping( 2 );

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

        // verificatrion of ID 2 successful

}
```

---

void reset(int id);

Resets Dynamixel.

Parameters

- id

  ID of Dynamixel.

Return Values

- none

Example

```
Dxl.reset( 2 );

if( Dxl.getResult( ) == COMM_RXSUCCESS )

{

        // reset ID 2

}
```

---

int getResult(void);

Checks for packet communications result.

Parameters

- None

Return Values

- Returns value types shown below.

Value	Meaning
COMM_TXSUCCESS	Instruction packet successful
COMM_RXSUCCESS	Status packet reception successful
COMM_TXFAIL	Instruction packet transmission failed
COMM_RXFAIL	Status packet reception failed
COMM_TXERROR	Instruction Packet error
COMM_RXWAITING	Status Packet error
COMM_RXTIMEOUT	Dynamixel not responding.
COMM_RXCORRUPT	Status Packet corrupted

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)

Sets Dynamixel position and velocity. This function sets velocity and position registers simultaneously.

Parameters

- **Servoid** ID of Dynamixel.

- **Position** goal position of Dynamixel.

-**Speed** goal velocity of Dynamixel (0~1023 range).

Return Values

-none

Example

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

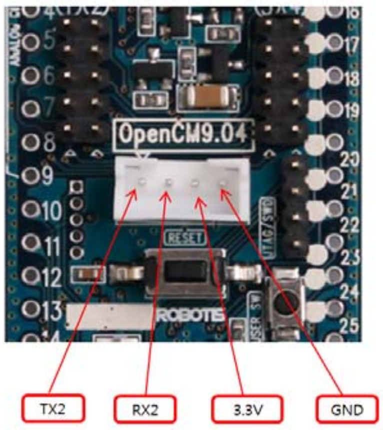
```
{
        // Verify position command has been received
}
```

③ RC100 API

RC100 is a device allows remote control of the OpenCM9.04 wirelessly. Simply connect to the OpenCM9.04's 4-pin connector and the CM-900. There's no need to implement instances; simply implement the functions listed below.

.

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



For detailed information about the 4-pin connector refer to the hardware portion of the OpenCM9.04

Device Control Methods	int begin(void)	ZigBee device initialized at 57600bps. devIndex default value is 0.
	void end(void)	Halts the device.
Data Methods	int writeData(int data)	Data transmission.
	int available(void)	Verify received data.
	int readData(void)	Returns value upon successful data reception.

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 General purpose input/output (GPIO)

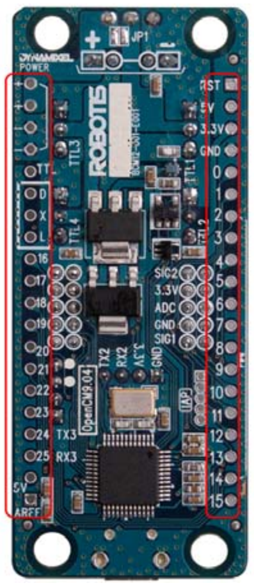
The functions listed below are based on Arduino and Leaflabs. Arduino's Reference are useful because they are relatively easy to understand C/C++ code instead of ARM's codes. Use the links below for more information on Arduino and LeafLabs.

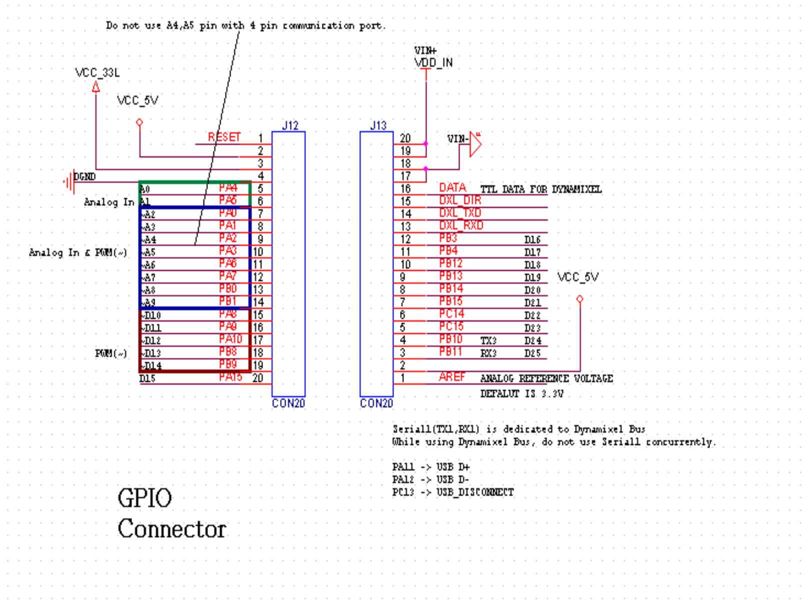
Arduino Reference : http://arduino.cc/en/Reference/HomePage
Leaflabs Maple Reference : http://leaflabs.com/docs/language.html

General Methods	pinMode(pin, WiringPinMode mode)	Sets pins for input and output.
Digital Methods	int digitalRead(pin)	Reads status of a specific High/Low pin. Said pin must be setup as input.
	digitalWrite(pin, value)	Writes High/Low to a specific pin. Said pin must be setup as output.
	togglePin(pin)	Set pin to toggle i.e. switch from Low to High and from High to Low.
Analog Methods	int analogRead(pin)	Read pin's analog value. Said pin must be setup as analog input.
	analogWrite(pin, duty cycle)	Writes analog data to pin. pwmWrite() duty cycle (0~65536 range) Duty cycle can be controlled.

For digital inputs use pins 0-25 of the OpenCM9.04.

Analog IN is engraved in the front portion of the OpenCM9.04 PCB. Implement analogWrite(), pwmWrite(), and TIMER on these specified pins.





---

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

Changes pin to mode.

Parameters:

- `pin` -

  Pin of the OpenCM9.04

- `mode` -

  Mode listed below for said pin.

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).

Description

pinMode() is a function in setup() where the pin can be set. The pin must be designated for writing.

Example

pinMode() sets LED to OUTPUT mode. Use the fuction digitalWrite() to write data (high/low). a blinking LED is the result.

```
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)

Reads pin High/Low status.

However, said pin must have INPUT_PULLUP or INPUT_PULLDOWN for input. Please refer to pinMode().

Parameters:
- `pin` -

  Assigns read pin.

Return: LOW or HIGH.

Description

If actual pin is not connected HIGH or LOW may be read randomly

Example

The following example the LED turns on and off repeatedly with the press of the button.

```
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)

Pins outpurs High/Low.

However, said pin must have OUTPUT_PULLUP or OUTPUT_PULLDOWN for output. Please refer to pinMode().

Parameters:

- `pin` -

  assigns write pin.

- `value` -

  HIGH(1) or LOW (0)

Description

The declared OUTPUT pin outputs 3.3V for HIGH and 0V for LOW.

Example

The following example is an implementation digitalWrite() function from LED Blink example.

```
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
}
```

The following allow replacement to toggleLED() inside loop(). This function is for the built-in LED.

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

Or replaced by togglePin().

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

---

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

Reads pin's analog value.

This featureis blocked until ADC is converted. Pin mode must be set to INPUT_ANALOG.

Parameters:

- `pin` -

  Analog pin read

Return: Voltage converted to 12-bit integer (0-4095).

Description

Reads analog value to assigned pin. The OpenCM9.04 has 16 12-bitchannels. This converts input of 0V to 3.3V to 0 to 4095. There are other factors that affect accuracy and must be taken into account.

To call this fuction   pinMode() function must be implementedand ANALOG_INPUT must be set. For more information please check pinMode().

Parameter Description

The number in this function is the analog pin number. These pin numbers are labeled in white on the OpenCM9.04's PCB silk screen along with ANALOG IN.

Note

If a pin is not connected then its readout is not possible.

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)

Given the declared duty cycle of the pin a PWM signal is possible. With OpenCM9.04 PWM signals can be controlled via duty cycle.

Duty cycle range between 0~65535.

Parameters:

- `pin` -

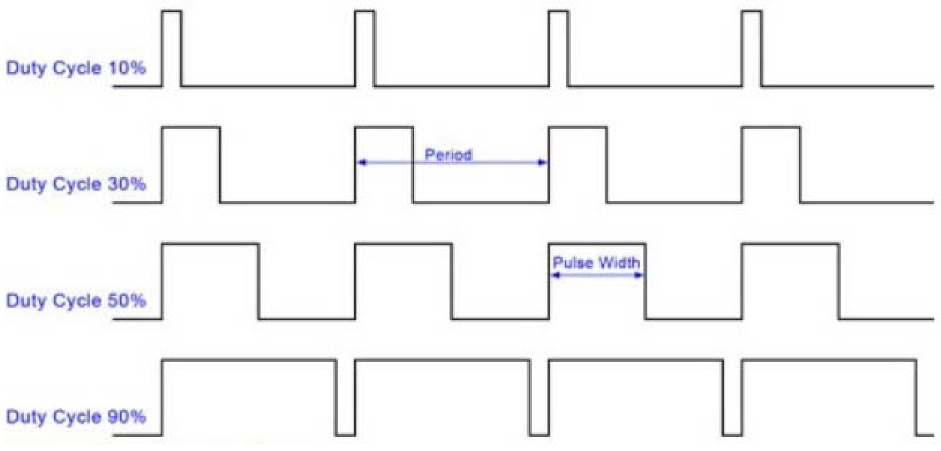
  PWM input pin

- `duty_cycle` -

  PWM signal duty cycle (0~65535)

Duty cycle can be controlled in the duty cycle input part of the function.

Values close to 0 the duty cycle is small (small HIGH area). Refer to the diagram below larger duty cycles (larger HIGH areas).



Example

The following is an example read from the potentiometer to control the brightness of the LED.

```
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)

Toggles the OpenCM9.04 STATUS LED.

When the LED in on status is called it turns off; when called in off it turns on.

Example

The following is the Blink example for STATUS LED.

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

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

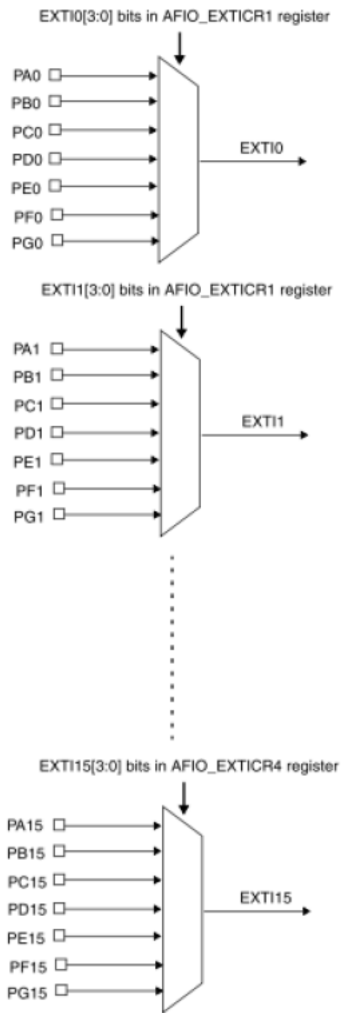
⑤ Interupt

Interrupt is a feature that allows specific actions to be performed based status return. An interrupt verification code is separately required because it utilizes hardware timer. Despite external devices having its own interrupt, it cannot exceed 16. For example pins 0 through 15 each with its own interrupt event there cannot be any more interrupts in code.

The following functions allow control of interrupts.

`attachInterrupt(`*pin*, voidFuncPtr *handler*, *mode*)	Adds interrupt handler for a specific pin.
`detachInterrupt(` *pin*)	Removes interrupt handler for a specific pin.
`noInterrupts()`	All interrupts disabled.
`Interrupts()`	All interrupts enabled.
`disableDebugPorts()`	JTAG/SWD disable option.
`enableDebugPorts()`	JTAG/SWD enable option.

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


EXTI0[3:0] bits in AFIO_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
EXTI0
EXTI1[3:0] bits in AFIO_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
EXTI1
EXTI15[3:0] bits in AFIO_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
EXTI15

Adapted from Maple Documentation

Parameters

- `pin` – pin #
- handler – interrupt event pointer.
- mode – interrupt form; falling edge or rising edge.

mode

interrupt event type.

Values:

- `RISING` -

  Trigger when LOW goes to HIGH.

- `FALLING` -

  Trigger when HIGH goes to LOW.

- `CHANGE` -

  When pins changes HIGH to LOW or LOW to HIGH; event trigger regardless.

Description

The delay() function cannot be used because interrupt is processed internally. Also values changed to millis () do not increase. Serial data reception process may be lost. To prevent data loss Volatile should be declared globally.

Example

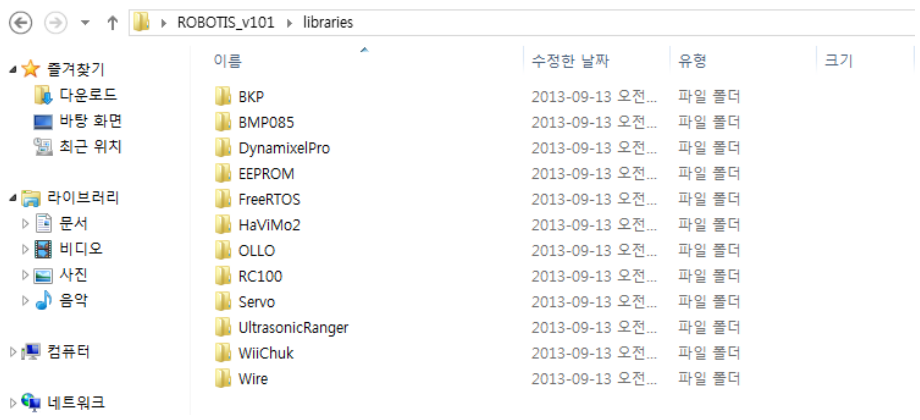
In this example pin 0 the LED turns on or off when there is a signal change.

```
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;
   }
}
```

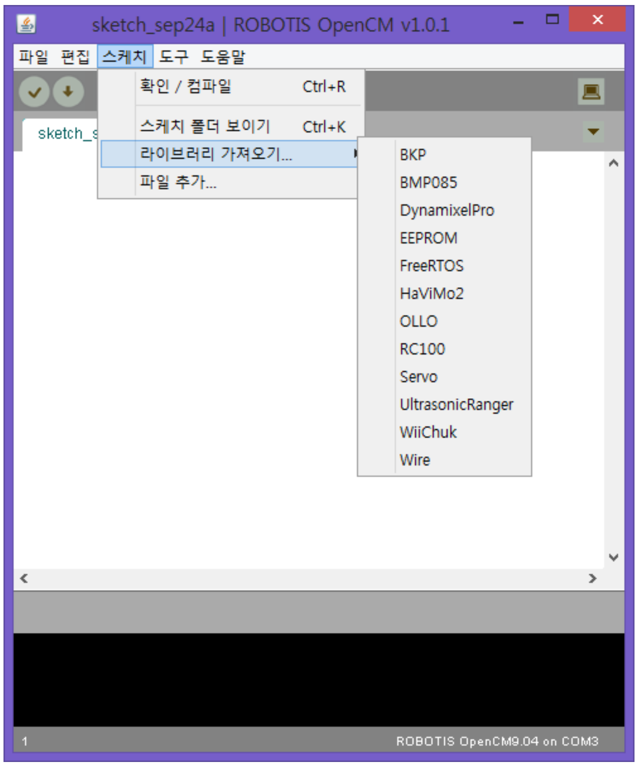
The function blink() is an interrupt handler. When pin 0 inout signal changes blink() gets called to high/low. In turn loop() follows the state (value) and sets the LED on/off.

⑥ User-created API library

When executing ROBOTIS OpenCM libraries from the folder \ROBOTIS\library get carried. This also includes user's libraries.



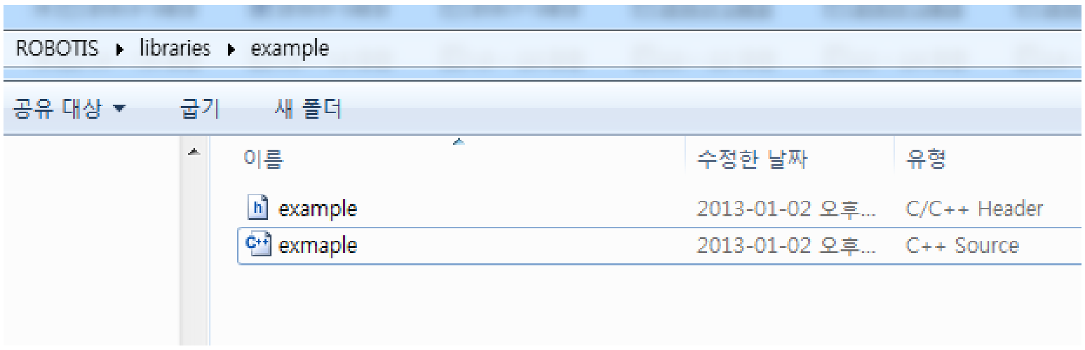
이름	수정한 날짜	유형	크기
BKP	2013-09-13 오전...	파일 폴더	
BMP085	2013-09-13 오전...	파일 폴더	
DynamixelPro	2013-09-13 오전...	파일 폴더	
EEPROM	2013-09-13 오전...	파일 폴더	
FreeRTOS	2013-09-13 오전...	파일 폴더	
HaViMo2	2013-09-13 오전...	파일 폴더	
OLLO	2013-09-13 오전...	파일 폴더	
RC100	2013-09-13 오전...	파일 폴더	
Servo	2013-09-13 오전...	파일 폴더	
UltrasonicRanger	2013-09-13 오전...	파일 폴더	
WiiChuk	2013-09-13 오전...	파일 폴더	
Wire	2013-09-13 오전...	파일 폴더	



A. C++ based library creation

Create file(s) in CPP format; API's in core library.

The following shows user-created example.h header file and example.cpp.



이름	수정한 날짜	유형
example	2013-01-02 오후...	C/C++ Header
exmaple	2013-01-02 오후...	C++ Source

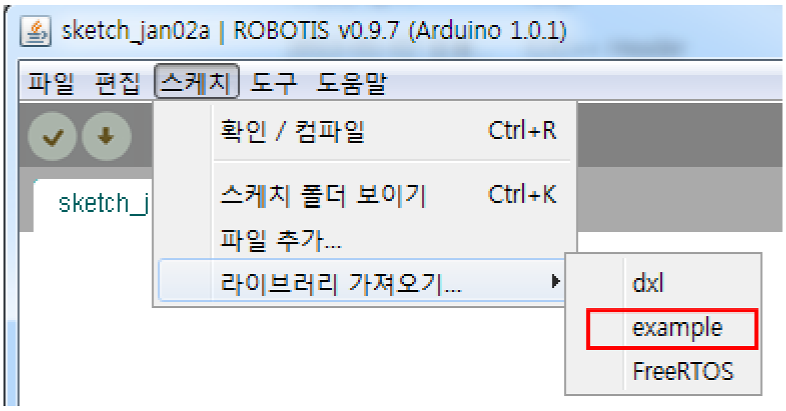
In example.h wirish.h has been declared. This makes APIs from digitalWrite() to Dynamixel API available from the OpenCM core library.

```
#include "wirish.h"

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

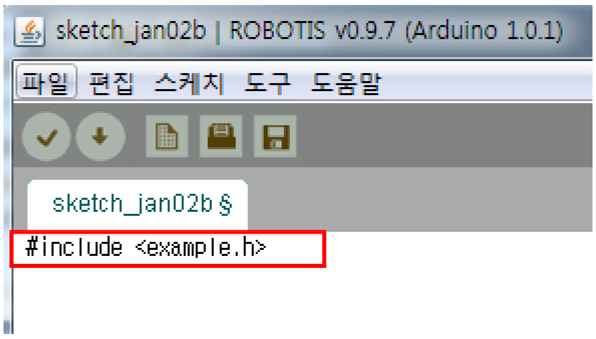
In example.cpp implement setupHelloWorld() and sendHelloWorld().

```
#include "example.h"

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

The project can be imported as shown below.


sketch_jan02a | ROBOTIS v0.9.7 (Arduino 1.0.1)
파일 편집 스케치 도구 도움말
확인 / 컴파일 Ctrl+R
스케치 폴더 보이기 Ctrl+K
파일 추가...
라이브러리 가져오기...
sketch_j
dxl
example
FreeRTOS

When adding #include <example.h> is automatically included.


sketch_jan02b | ROBOTIS v0.9.7 (Arduino 1.0.1)
파일 편집 스케치 도구 도움말
sketch_jan02b §

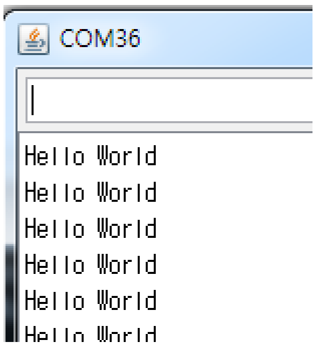
```
#include <example.h>
```

Now both setupHelloWorld() and sendHelloWorld() can be written. Note that in the #include preprocessor the < and > characters refer to ROBOTIS\libraries directory.

```
#include <example.h>

void setup(){

setupHelloWorld();

}

void loop(){

sendHelloWorld();

}
```

<sketch code>

Once compiled and downloaded to the OpenCM the results can be seen on serial monitor.

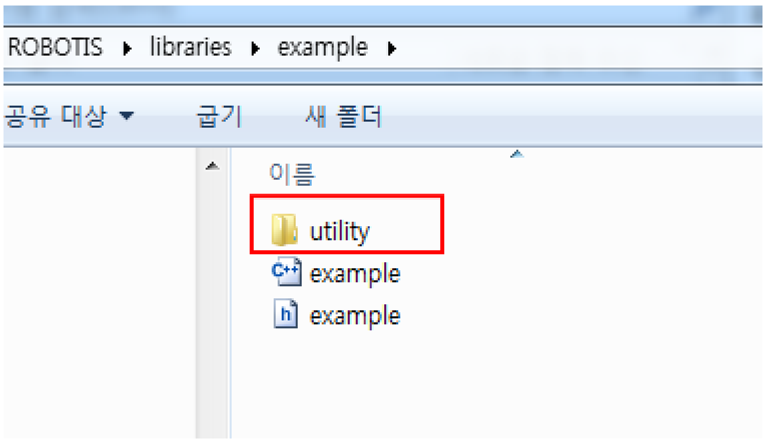


```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

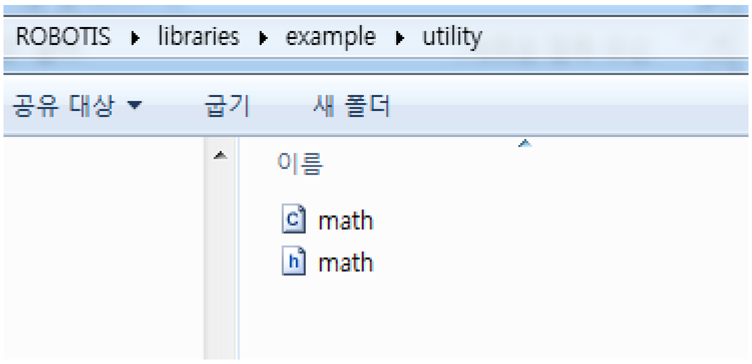
B. C-based library creation

Most firmware codes are written in C. C is also available for coding the OpenCM.

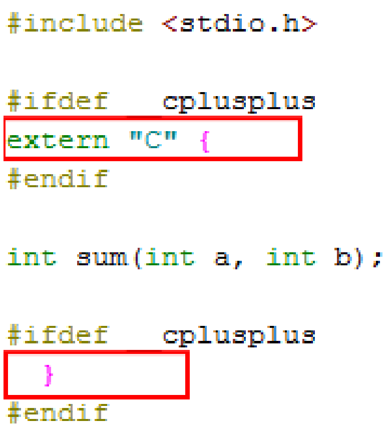
The following shows math.c, math.h files under utility folder under example library.



Both C-based files under utility directory.



The following show the contents of math.c and math.h files.

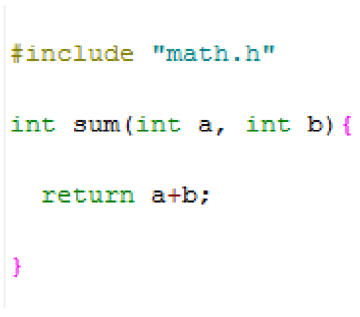


```
#include <stdio.h>

#ifdef   cplusplus
extern "C" {
#endif

int sum(int a, int b);

#ifdef   cplusplus
  }
#endif
```

contents of <math.h>

Let’s look at sum() in ROBOTIS OpenCMs tool chain. Both “<” and “>” characters imply looking for a directory called include.

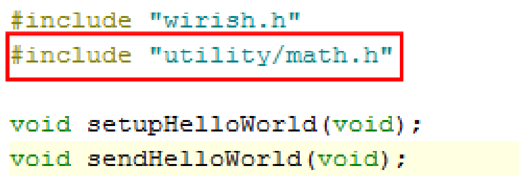
Extern “C” {} required C++-based compilers.


```
#include "math.h"

int sum(int a, int b){

  return a+b;

}
```

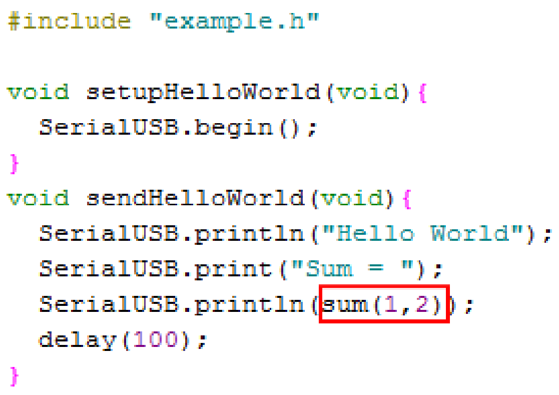
contents of <math.c>

The following show modified example.h and example.cpp.

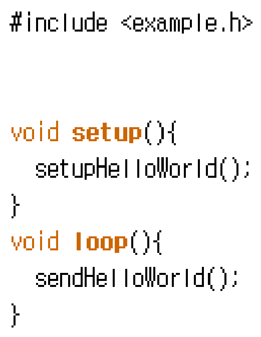
Declare #include “utility/math.h”

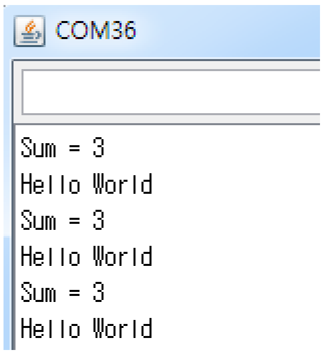

```
#include "wirish.h"
#include "utility/math.h"

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

Add sum() function to example.cpp.


```
#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);
}
```

sendHelloWorld() sketch code outputs sum(1,2) value.

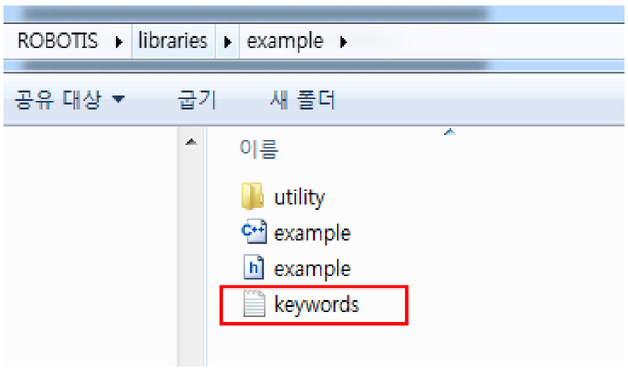

```
#include <example.h>


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



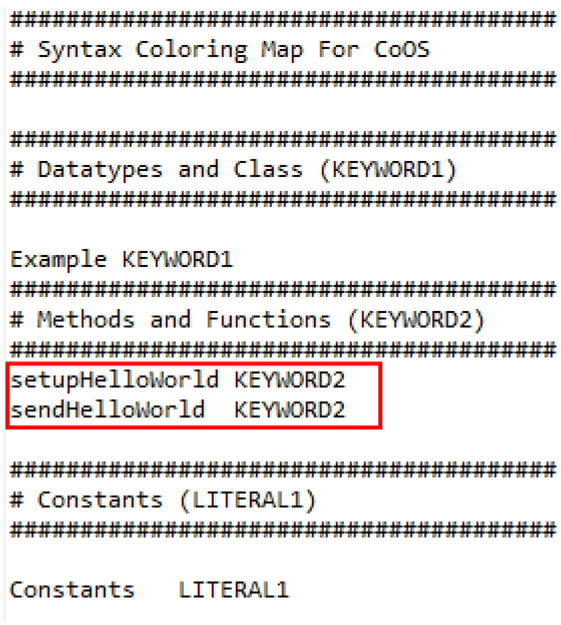
```
Sum = 3
Hello World
Sum = 3
Hello World
Sum = 3
Hello World
```

C. Syntax library highlights

- Add keywords.txt file for syntax.
- And place it in the same example directory.

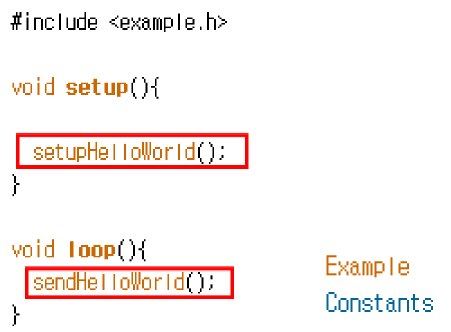


Add the contents of the file as shown below.



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

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

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

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

Constants    LITERAL1
```

In ROBOTIS CM-9 look for a color change when inputting setupHelloWorld() and sendHelloWorld().

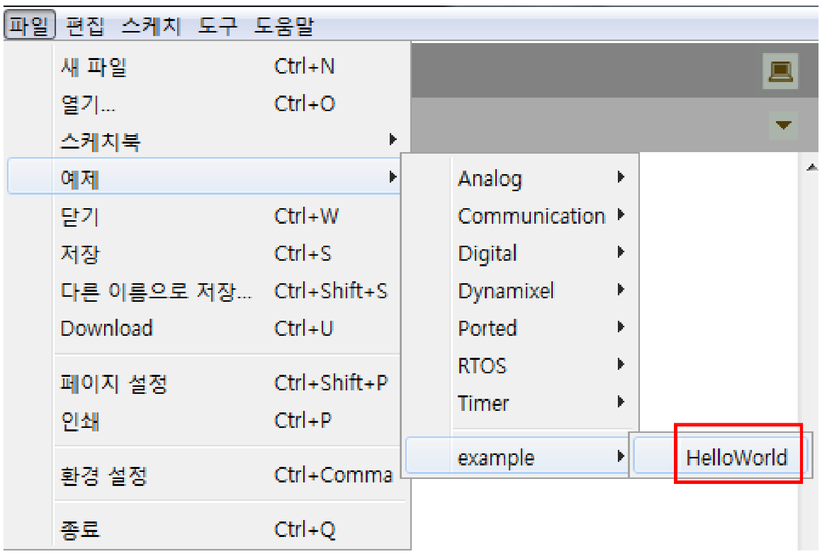


```
#include <example.h>

void setup(){

 setupHelloWorld();
}

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

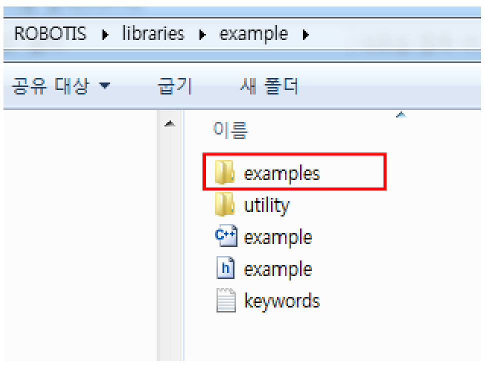
<문법 강조 기능 적용되었을 때>

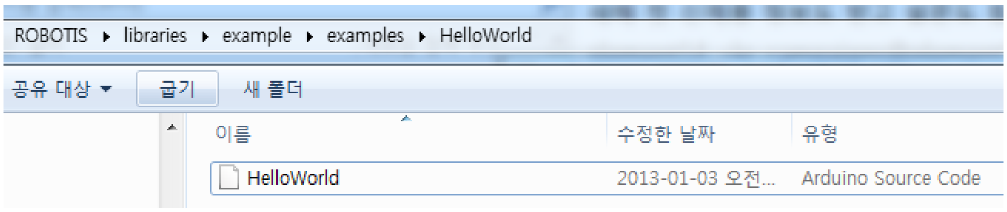
A. Registering a library

You can register your own library so it can be readily available in the OpenCM menu.



Simply place your sketch code inside the example directory. Restart ROBOTIS CM-9 and your custom-created library will shop up in the menu.





The name in the menu will reflect the name of the sketch code file.