

# **W7200 User's Guide**

## **for STM32F10x Standard Library**

- USART
- GPIO
- Timer

**Version 1.0.0**



© 2012 WIZnet Co., Ltd. All Rights Reserved.

For more information, visit our website at <http://www.wiznet.co.kr>

---

# Table of Contents

1	Introduction.....	1
2	Environment Setting for Development .....	1
3	STM32F10x Standard Library Structures .....	2
4	Configuration .....	4
4.1	Check points .....	4
4.2	How to use it .....	4
5	Examples Demonstration .....	7
5.1	USART .....	7
5.2	GPIO .....	10
5.3	Timer.....	12

# 1 Introduction

W7200은 ARM의 32-bit Cortex-M3 MCU core인 STM32F103CB와 hardwired TCP/IP, MAC과 PHY (WIZnet W5200)가 하나로 합쳐진 WIZnet iMCU 제품이다. STMicroelectronics의 MCU가 내장되어 때문에 ST에서 제공하는 peripheral library를 사용하여 쉽게 주변장치를 활용할 수 있다.

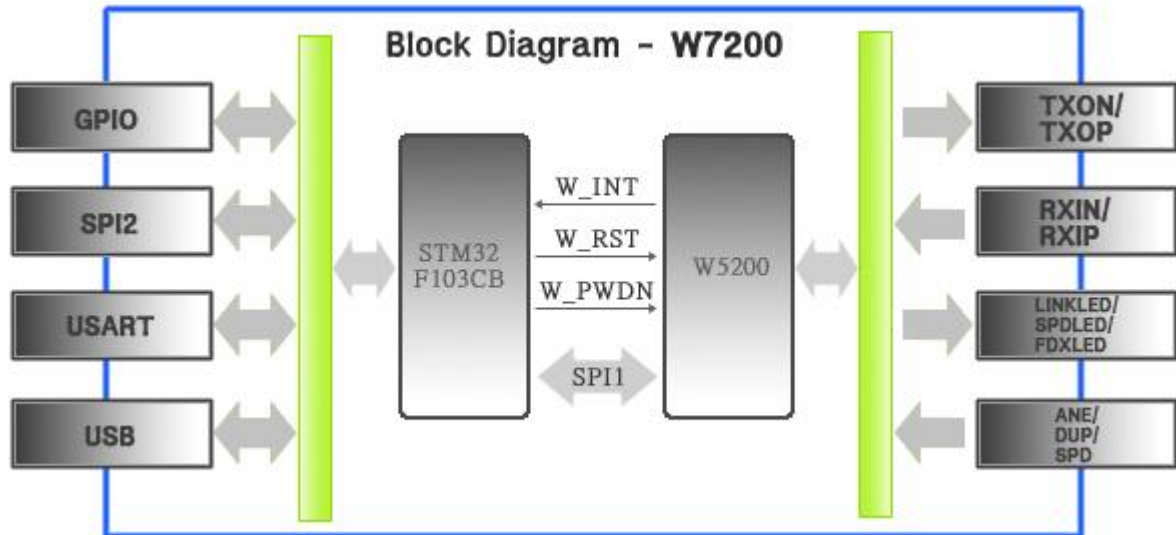


Figure 1. iMCU W7200 Block Diagram

본 문서에서는 STMicroelectronics에서 제공하는 Standard Library의 구성과 사용 방법, peripheral example 중 가장 기본적인 다음 항목들을 다룬다. 이 외에도 다양한 예제가 제공되고 있으므로 사용자는 필요에 따라 선택하여 주변장치의 동작 방법을 확인하고, MCU를 활용할 수 있다.

이 문서에서는 다음 주변장치의 기본적인 예제를 다룬다.

- USART : printf (+Echoback)
- GPIO : IO Toggle (+LED control)
- Timer : Time Base

# 2 Environment Setting for Development

W7200 Startup Guide 문서를 참고하여 IAR EWARM (Embedded Workbench for ARM) 개발환경을 구성하고, STMicroelectronics에서 제공하는 STM32 F10x Standard Peripheral Library를 준비한다. (STMicroelectronics website, [www.st.com/STM32](http://www.st.com/STM32)) 본 문서는 STM32 F10x Standard Peripheral Library version 3.5.0을 기준으로 작성되었으며, IAR EWARM version 6.5(30-day time limited evaluation license)를 이용하였다.

W7200 Startup Guide 문서는 WIZnet website 다음 Link에서 다운로드 할 수 있다.

⇒ [http://www.wiznet.co.kr/UpLoad\\_Files/ReferenceFiles/W7200\\_StartUpGuide\\_EWARM\\_v100K.pdf](http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W7200_StartUpGuide_EWARM_v100K.pdf)

### 3 STM32F10x Standard Library Structures

STM32 F10x Standard Peripheral Library는 크게 [Libraries], [Project], [Utilities] 폴더로 구성되어 있다. 최 상위 폴더에는 STM32 F10x Standard Peripheral Library에 대한 자세한 설명을 담은 HTML 도움말 파일(stm32f10x\_stdperiph\_lib\_um.chm)이 위치하므로 library 내용에 대한 자세한 설명은 이를 참조하기 바란다.

[Project] 폴더 내에는 두 개의 폴더가 있으며 각각의 내용은 다음과 같다.

[STM32F10x\_StdPeriph\_Examples] : STM peripheral drivers 이용 주변장치 제어 예제

[STM32F10x\_StdPeriph\_Template] : Compiler 별 project template (W7200: EWARM)

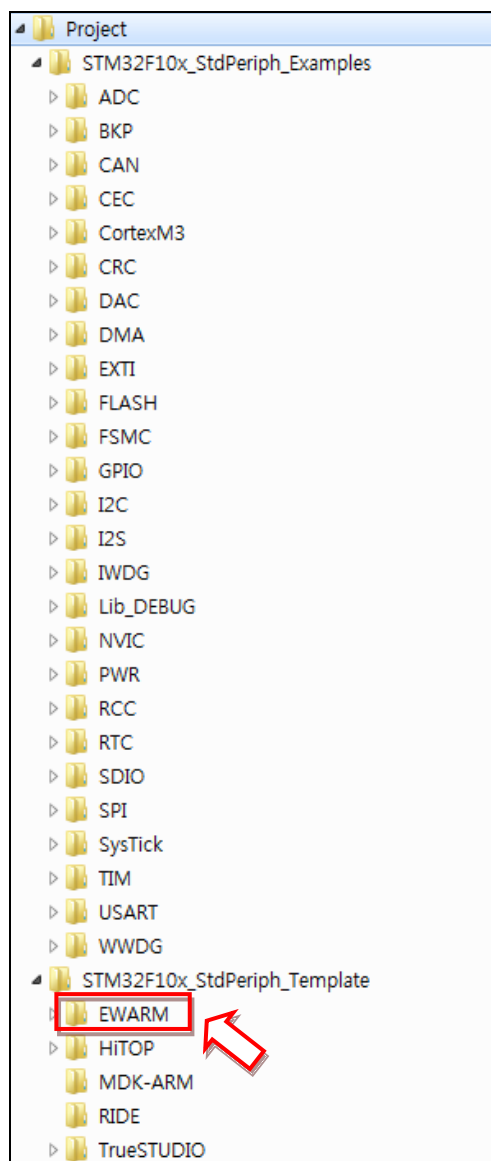


Figure 2. Folders List of Project

[Libraries] 폴더 내에는 두 개의 폴더가 있으며 각각의 내용은 다음과 같다.

[CMSIS] : ARM에서 제공하는 Cortex-M 시리즈 MCU library

※ CMSIS : *Cortex Microcontroller Software Interface Standard*

[STM32F10x\_StdPeriph\_Driver] : STMicroelectronics에서 제공하는 주변장치용 drivers

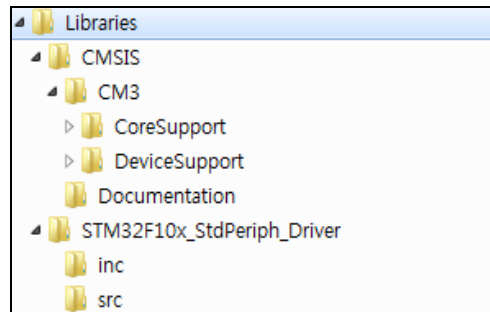


Figure 3. Folders List of Libraries

[Utilities] 폴더 내에는 한 개의 폴더가 있으며 내용은 다음과 같다.

[STM32\_EVAL] : STM MCU 라인업의 특징에 따른 MCU별(EVB) Set-up 설정 파일

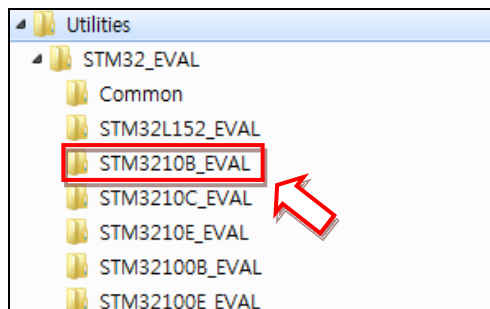


Figure 4. Folders list of STM32\_EVAL

## 4 Configuration

### 4.1 Check points

W7200은 칩 내부에 STM32 MCU와 W5200이 SPI 인터페이스로 연결되어 있다. 따라서 SPI 신호를 포함한 Table 1의 신호들은 이미 사용되고 있어 다른 용도로는 사용할 수 없다.

Table 1. W7200 Internal Connected (Unused) Pin List

No	W7200		Description
	STM32F103CB Pins, Pin name	W5200 Pins, Pin name	
1	2, PC13/TAMPER_RTC	49, W_INT	W5200 Interrupt pin
2	14, PA4/SPI_NSS	50, CSN	STM32 MCU(SPI1) - W5200 SPI Interface pin
3	15, PA5/SPI_SCK	51, SCLK	
4	16, PA6/SPI_MISO	53, MISO	
5	17, PA7/SPI_MOSI	52, MOSI	
6	45, PB8	53, W_RESET	W5200 Reset pin
7	46, PB9	54, PWDN	W5200 Power-down pin

### 4.2 How to use it

STMicroelectronics에서 제공하는 예제는 다음과 같은 과정을 거쳐 실행할 수 있다.

1. 다운로드 한 STM32 F10x Standard Peripheral Library의 압축 해제
2. 원하는 Examples 폴더의 예제 source code 파일을 template 폴더로 복사  

`$PROJ_DIR$ \ STM32F10x_StdPeriph_Template`
3. IAR EWARM을 이용하여 Template 폴더 내 EWARM의 project 파일을 실행
4. Rebuild를 선택하여 Binary image 파일을 생성한 뒤, W7200 보드에 프로그램
5. 예제 실행

W7200은 STM32F103CB MCU(128Kbytes Flash memory)를 포함하고 있어 **Medium-density devices**로 분류되므로, IAR EWARM의 Workspace에서 **STM3210B-EVAL** Set-up을 선택하여 진행하면 된다. Binary image 생성과 W7200 보드에 프로그램 하는 자세한 과정은 **W7200 Startup Guide** 문서의 ‘**2장 Configuration and Start the Project**’를 참조하기 바란다.

**Note**

**Setting for C Compiler Preprocessor**

컴파일이 올바르게 수행되려면 include 할 library의 위치가 정확히 표기되어야 한다. STM32 F10x Standard Peripheral Library Version 3.5.0에는 표기되어 있지만, 만약 이 부분이 없는 경우 Project > Options의 C/C++ Compiler 항목의 Preprocessor에서 다음과 같은 directory 정보와 symbol 정보를 추가한다.

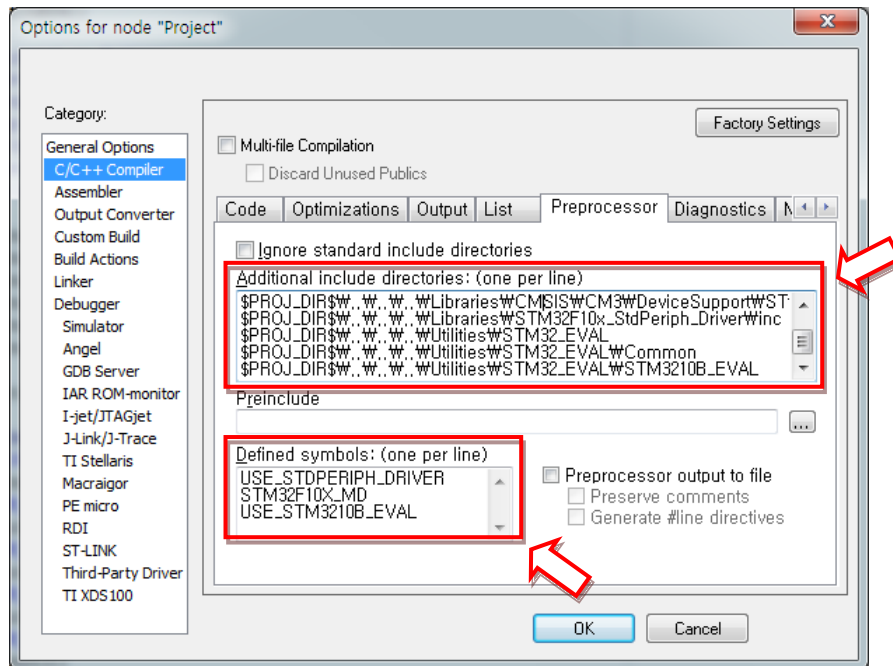


Figure 5. Setting for C/C++ Compiler Preprocessor

**Additional include directories: (one per line)**

```

$PROJ_DIR$\..\
$PROJ_DIR$\..\..\Libraries\CMSIS\CM3\CoreSupport
$PROJ_DIR$\..\..\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x
$PROJ_DIR$\..\..\Libraries\STM32F10x_StdPeriph_Driver\inc
$PROJ_DIR$\..\..\Utilities\STM32_EVAL
$PROJ_DIR$\..\..\Utilities\STM32_EVAL\Common
$PROJ_DIR$\..\..\Utilities\STM32_EVAL\STM3210B_EVAL
    
```

**Defined symbols: (one per line)**

```

USE_STDPERIPH_DRIVER
STM32F10X_MD
USE_STM3210B_EVAL
    
```

Note

**Make the Binary output file**

W7200 Startup guide를 따라 Flash Loader Demonstrator를 이용하여 W7200 보드에 프로그램 하려는 경우, Make의 결과물로 Binary image가 필요하다. STM32 F10x Standard Peripheral Library Version 3.5.0의 경우, Output binary를 생성하지 않는 것이 default이므로, Project > Options의 Output Converter 메뉴에서 **Generate additional output** 옵션을 활성화 한다.

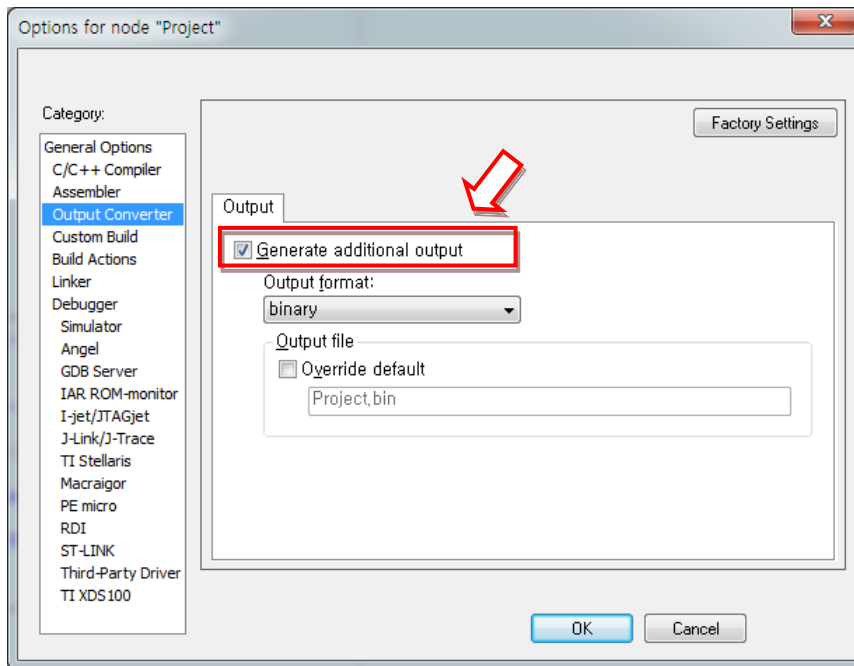


Figure 6. Enable the Generate Additional Output Option



## 5 Examples Demonstration

### 5.1 USART

본 문서에서는 STMicroelectronics에서 제공하는 USART 예제 중 USARTx를 이용하여 Serial terminal에 메시지를 출력하는 예제인 'printf'를 보인다.

4.2장의 'How to use it'을 따라 **USART > printf** 예제를 *STM32F10x\_StdPeriph\_Template* 폴더로 복사한다. 'printf' 폴더에는 다음과 같은 파일들이 있다.

Table 2. Source code list of the 'printf' example

File name	Description
main.c	main 함수, USART configuration과 printf 수행 포함
stm32f10x_conf.h	Library configuration 파일
stm32f10x_it.c	Interrupt service routine들이 포함된 파일
stm32f10x_it.h	Interrupt handler들의 header 파일
system_stm32f10x.c	System clock configuration 등 system initialization 함수 파일
readme.txt	'printf' 예제에 대한 설명 파일

#### Note

이 예제에는 USART interrupt에 대한 handling routine이 포함되어 있지 않으며, USART interrupt handling routine은 USART 예제 중 'Interrupt'에 구현되어 있으므로 참조 바란다.

Main 함수에는 다음과 같은 USART initialization code가 있다. 이 초기화 과정은 각종 설정 값을 USART\_InitStructure 구조체에 저장한 후 STM\_EVAL\_COMInit 함수를 이용하여 USART를 configuration 및 enable 한다.

#### Example code: USART Initialization

```
USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

STM_EVAL_COMInit(COM1, &USART_InitStructure);
```

또한 printf 함수로 문자열을 출력했을 때 USART로 출력될 수 있도록 구성된 fputc 함수가 정의되어 있다.

Example code: Retarget the C library printf function to the USART

```
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
PUTCHAR_PROTOTYPE
{
    /* Place your implementation of fputc here */
    /* e.g. write a character to the USART */
    USART_SendData(EVAL_COM1, (uint8_t) ch);

    /* Loop until the end of transmission */
    while (USART_GetFlagStatus(EVAL_COM1, USART_FLAG_TC) == RESET)
    {}

    return ch;
}
```

이제 초기화가 성공적으로 수행되면, printf() 함수를 사용하여 Serial terminal에 문자열을 출력할 수 있다. 이 때 Serial terminal은 위 Initialization 코드에 맞춰 다음 Table 3과 같이 설정 한다.

Table 3. Serial Terminal Setting

항목	설정 값
비트/초(Baud rate)	115200
Data 비트	8
패리티	없음
정지 비트	1
흐름 제어	없음

함수 printf 만을 포함하고 있던 기존 main 코드에 USART의 입출력 동작을 검증하기 위하여 Serial로 받은 입력을 그대로 다시 Serial을 통해 출력하는 Echoback기능을 추가한다. Echoback의 구현을 위하여 다음 코드를 하단의 while문 안에 삽입하면 된다. 아래 코드에 사용된 USART 관련 함수들은 stm32f10x\_usart.c에 포함되어 있다.

**Note**

Example code 내 음영 처리된 부분은 본 문서에서 추가한 내용이거나 iMCU7200EVB에 맞춰 수정된 부분이므로 활용 시 참고하기 바란다. 본 문서 내 모든 예제가 동일하다.

**Example code: Main Function**

```

Int main(void)
{
    unsigned char c;

    /* USART1 Initialization */

    ...
    STM_EVAL_COMInit(COM1, &USART_InitStructure);

    /* Output a message on Hyperterminal using printf function */
    printf("\n\rUSART Printf Example: retarget the C library printf function to the USART\n\r");

    // USART Echoback
    while(1)
    {
        while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == RESET) {}

        c = USART_ReceiveData(USART1); // Read a character from the USART
        USART_SendData(USART1, c); // Send a character to the USART
    }
}

```

다음은 iMCU7200EVB에서 예제에 Echoback을 더한 코드를 수행한 결과이다. printf 함수를 이용하여 예제에 포함된 메시지를 Serial terminal로 출력하였으며, ‘Hello! WIZnet!’을 직접 타이핑하여 Echoback을 확인하였다.

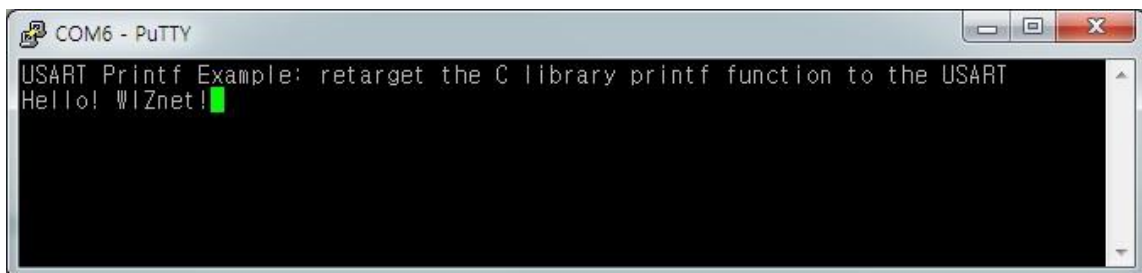


Figure 7. USART Demonstration on iMCU7200EVB

## 5.2 GPIO

본 문서에서 제공하는 GPIO 예제는 STMicroelectronics에서 제공하는 GPIO 예제 중 GPIO Pin을 이용하여 Set과 Reset 상태를 번갈아 가며 출력하는 예제인 'IO Toggle'이다.

### Note

※ STM32 library의 예제는 PD0와 PD2를 toggling하는 예제를 보이고 있으나, W7200에 사용된 STM32F103CB-48PIN은 PD0와 PD2를 지원하지 않는다. 본문에서는 PD0와 PD2를 iMCU7200EVB의 LED3, 4인 PB0와 PB1로 대체한다. 따라서 IO toggle이 수행될 때 EVB의 LED를 통해 GPIO의 toggle 동작을 직접 확인 가능하다.

Main 함수에는 다음과 같은 GPIO Initialization 코드가 포함되어 있다. USART와 유사하게 InitStructure 구조체에 원하는 핀의 선택과 속도, 동작 모드를 설정한 후 GPIO\_Init 함수를 이용하여 구조체의 값으로 configuration을 수행한다. 본문의 GPIO 초기화 예제 코드는 STM32 library의 IO Toggle 예제에서 PB0와 PB1을 사용하도록 수정한 것이다.

#### Example code: GPIO Initialization

```

/* GPIOB Periph clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

/* Configure PB0 and PB1 in output push-pull mode */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

GPIO_Init(GPIOB, &GPIO_InitStructure);
    
```

PB0와 PB1을 SET / RESET 시키기 위해서는 BSRR과 BRR 레지스터를 이용한다. GPIO\_BSRR 레지스터(Port Bit Set/Reset Register)는 32bit로써, 하위 16bit를 사용하여 해당 비트 번호와 대응(mapping)하는 핀을 SET 시킨다. GPIO\_BRR 레지스터(Port Bit Reset Register)는 BSRR과 유사하며 해당 비트 번호와 대응하는 핀을 RESET 시킨다. BSRR과 BRR이 포함된 General purpose I/O의 구조체는 stm32f10x.h에 선언되어 있다. 만약 Pin0를 SET 하려면 BSRR register를 0x00000001로, Pin15를 RESET 시키려면 BRR register를 0x00008000으로 설정하면 된다.

PB0와 PB1를 SET / RESET 시키는 예제 코드는 다음과 같다.

PB0를 제어하기 위해 0x00000001를, PB1을 제어하기 위해 0x00000002을 설정하면 SET과 RESET을 결정할 수 있다. BSRR에 설정하면 SET, BRR register에 설정하면 RESET 된다.

여기에 LED의 깜빡임을 확인할 수 있도록 Delay를 추가하였다. Delay 함수는 STM에서 제공하는 library에서 제공한다.

Example code: GPIO Toggle

```

while (1)
{
    /* Set PB0 and PB1 */
    GPIOB->BSRR = 0x00000003;           // 0x00000001 | 0x00000002
    Delay(10);                          // Delay for LED(I/O) toggling indication
    /* Reset PB0 and PB1 */
    GPIOB->BRR = 0x00000003;
    Delay(10);                          // Delay for LED(I/O) toggling indication
}

```

이제 iMCU7200EVb에 프로그램 하여 동작을 수행하면 LED의 깜빡임을 통해 IO toggle을 확인할 수 있다.

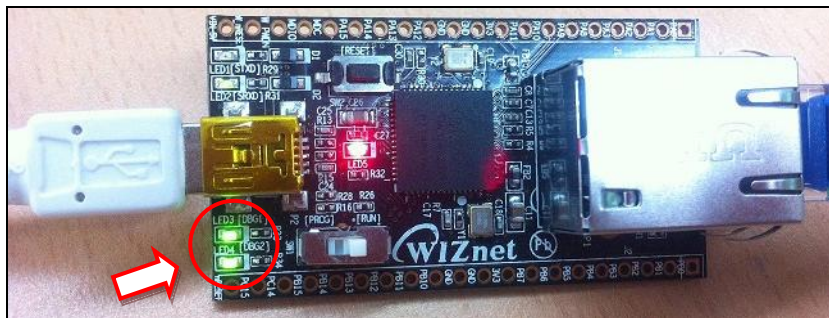


Figure 8. IO Toggle - LED on



Figure 9. IO Toggle - LED off

### 5.3 Timer

본 문서에서는 STMicroelectronics에서 제공하는 Timer 예제 중 TIM Time Base 예제를 보인다. 이 예제는 TIM2를 이용하여 Output Compare Timing mode로 동작하는 TIM peripheral의 configuration과 각각의 4개 채널이 각기 다른 Time base에 따라 Interrupt request를 생성함을 확인할 수 있다.

**Note**

※ STM32 library의 예제는 PC6, 7, 8, 9를 4개의 Timer Interrupt channel로 사용하고 있지만, W7200에 사용된 STM32F103CB-48PIN은 해당 핀을 지원하지 않는다. 따라서 본문에서는 GPIO PB11, 12, 13, 14를 output port로 이용한다.

4개의 채널은 각각 다음과 같이 설정되어 주기적으로 Interrupt request를 발생시킨다.

Table 4. Time base Configuration

Pin	TIM2 CC	CC Register value	Description
PB11	CC1	40961	CC1 update rate = TIM2 counter clock / CCR1_Val = 146.48Hz TIM2 Channel 1 generates an interrupt each 6.8ms
PB12	CC2	27309	CC2 update rate = TIM2 counter clock / CCR2_Val = 219.7Hz TIM2 Channel 2 generates an interrupt each 4.55ms
PB13	CC3	13654	CC3 update rate = TIM2 counter clock / CCR3_Val = 439.4Hz TIM2 Channel 3 generates an interrupt each 2.27ms
PB14	CC4	6826	CC4 update rate = TIM2 counter clock / CCR4_Val = 878.9Hz TIM2 Channel 4 generates an interrupt each 1.13ms

CC Register value의 값은 main.c 파일에 선언되어 있다.

Example code: CCR Value Declarations

```
__IO uint16_t CCR1_Val = 40961;
__IO uint16_t CCR2_Val = 27309;
__IO uint16_t CCR3_Val = 13654;
__IO uint16_t CCR4_Val = 6826;
```

예제와 다른 Pin을 사용하기 위해 변경해야 할 부분은 다음과 같다.

1. RCC\_Configuration : System clocks configuration
2. GPIO\_Configuration : General purpose I/O configuration
3. TIM2\_IRQHandler : TIM2 Interrupt service routine
- NVIC\_Configuration : Nested vectored interrupt controller configuration

위 순서에 따라 Pin 관련부를 수정한다. 수정된 부분은 다음 예제코드를 참조하라.  
먼저 RCC\_Configuration 함수의 GPIOC clock enable을 GPIOB clock enable로 수정한다.

#### Example code: RCC Configuration

```
void RCC_Configuration(void)
{
    /* PCLK1 = HCLK/4 */
    RCC_PCLK1Config(RCC_HCLK_Div4);

    /* TIM2 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

    /* GPIOB clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
}
```

GPIO\_Configuration 함수는 Timer interrupt가 발생하면 외부로 출력할 GPIO Pin을 초기화한다. 예제와 다른 Pin을 통해 TIM2 interrupt request를 확인하고 싶은 경우, 이 함수의 Pin 설정을 변경한다.

#### Example code: GPIO Configuration

```
void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* GPIOB Configuration: Pin4, 5, 6 and 7 as alternate function push-pull */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

TIM2\_IRQHandler 함수는 Timer interrupt가 발생한 경우 이를 처리하기 위한 interrupt handling routine이다. Interrupt request 발생 시 마다 이 부분이 호출되어 GPIO를 통해 외부에서 확인할 수 있도록 처리한다. 이 부분에서도 GPIO Pin의 변경사항만 적용하면 된다.

### Example code: TIM2\_IRQHandler

```
void TIM2_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM2, TIM_IT_CC1) != RESET)
    {
        TIM_ClearITPendingBit(TIM2, TIM_IT_CC1);
        /* Pin PB.11 toggling with frequency = 73.24 Hz */
        GPIO_WriteBit(GPIOB, GPIO_Pin_11, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_11)));
        capture = TIM_GetCapture1(TIM2);
        TIM_SetCompare1(TIM2, capture + CCR1_Val);
    }
    else if (TIM_GetITStatus(TIM2, TIM_IT_CC2) != RESET)
    {
        TIM_ClearITPendingBit(TIM2, TIM_IT_CC2);
        /* Pin PB.12 toggling with frequency = 109.8 Hz */
        GPIO_WriteBit(GPIOB, GPIO_Pin_12, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_12)));
        capture = TIM_GetCapture2(TIM2);
        TIM_SetCompare2(TIM2, capture + CCR2_Val);
    }
    else if (TIM_GetITStatus(TIM2, TIM_IT_CC3) != RESET)
    {
        TIM_ClearITPendingBit(TIM2, TIM_IT_CC3);
        /* Pin PB.13 toggling with frequency = 219.7 Hz */
        GPIO_WriteBit(GPIOB, GPIO_Pin_13, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_13)));
        capture = TIM_GetCapture3(TIM2);
        TIM_SetCompare3(TIM2, capture + CCR3_Val);
    }
    else
    {
        TIM_ClearITPendingBit(TIM2, TIM_IT_CC4);
        /* Pin PB.14 toggling with frequency = 439.4 Hz */
        GPIO_WriteBit(GPIOB, GPIO_Pin_14, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_14)));
        capture = TIM_GetCapture4(TIM2);
        TIM_SetCompare4(TIM2, capture + CCR4_Val);
    }
}
```



NVIC\_Configuration 함수는 Timer2 global interrupt를 초기화하는 코드이다. Timer2만 사용하는 현재 예제의 경우 수정할 부분은 없지만, 만약 Timer2 이외에 다른 타이머를 사용하고자 한다면 이 부분을 수정해야 한다.

Example code: NVIC Configuration

```
void NVIC_Configuration(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Enable the TIM2 global Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

    NVIC_Init(&NVIC_InitStructure);
}
```

수정된 부분을 적용하여 만든 Binary image를 iMCU7200EVB에 프로그램하고, PinPB11, 12, 13, 14에 Oscilloscope를 이용하여 Timer Interrupt 신호를 관측한 결과는 다음과 같다. 가장 상단부터 CC1, CC2, CC3, CC4의 순서이다.

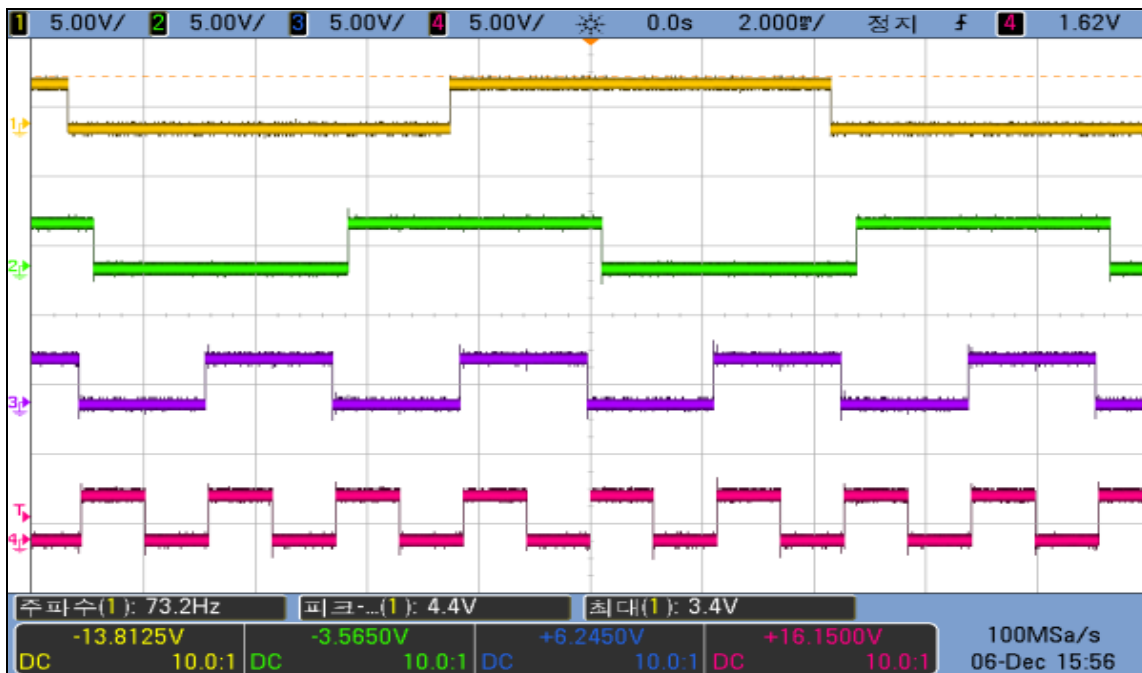


Figure 10. Oscilloscope Measurements : Timer interrupts

## Document History Information

Version	Date	Descriptions
Ver. 1.0.0	6Dec2012	Document Released

---

---

## Copyright Notice

Copyright 2012 WIZnet, Co., Ltd. All Rights Reserved.

Technical Support: [support@wiznet.co.kr](mailto:support@wiznet.co.kr)

Sales & Distribution: [sales@wiznet.co.kr](mailto:sales@wiznet.co.kr)

For more information, visit our website at <http://www.wiznet.co.kr>