임베디드시스템설계
EMBEDDED SYSTEM DESIGN

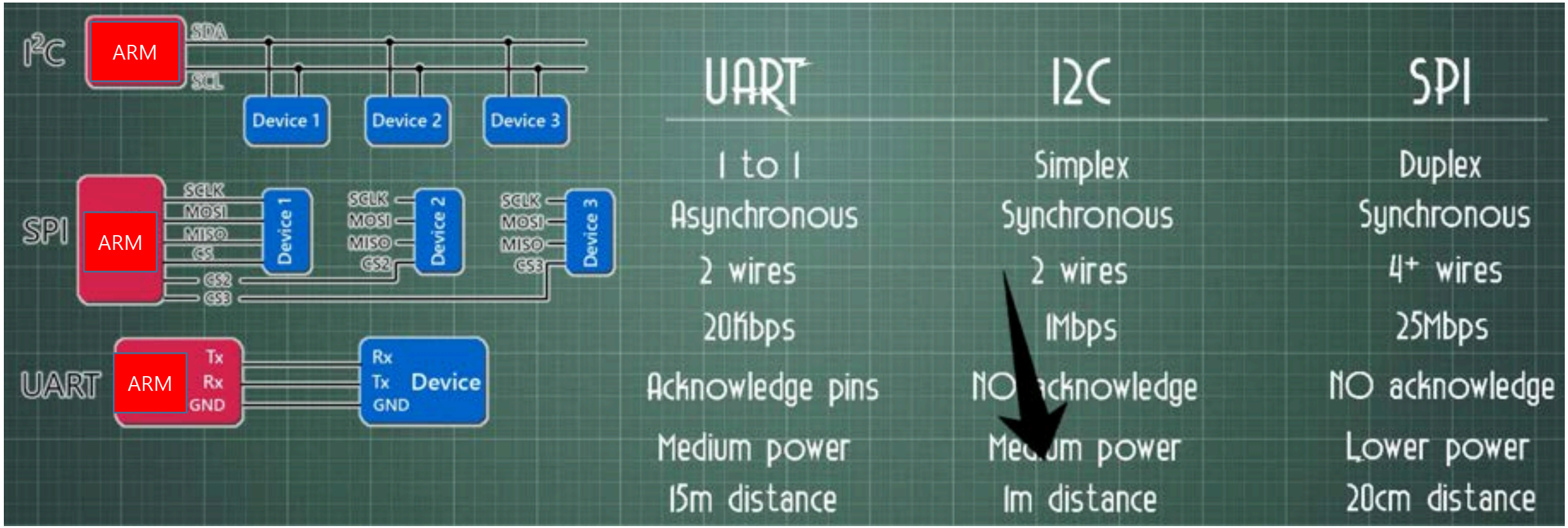
CHAPTER 12

I^2C 통신

KYUNG HEE UNIVERSITY

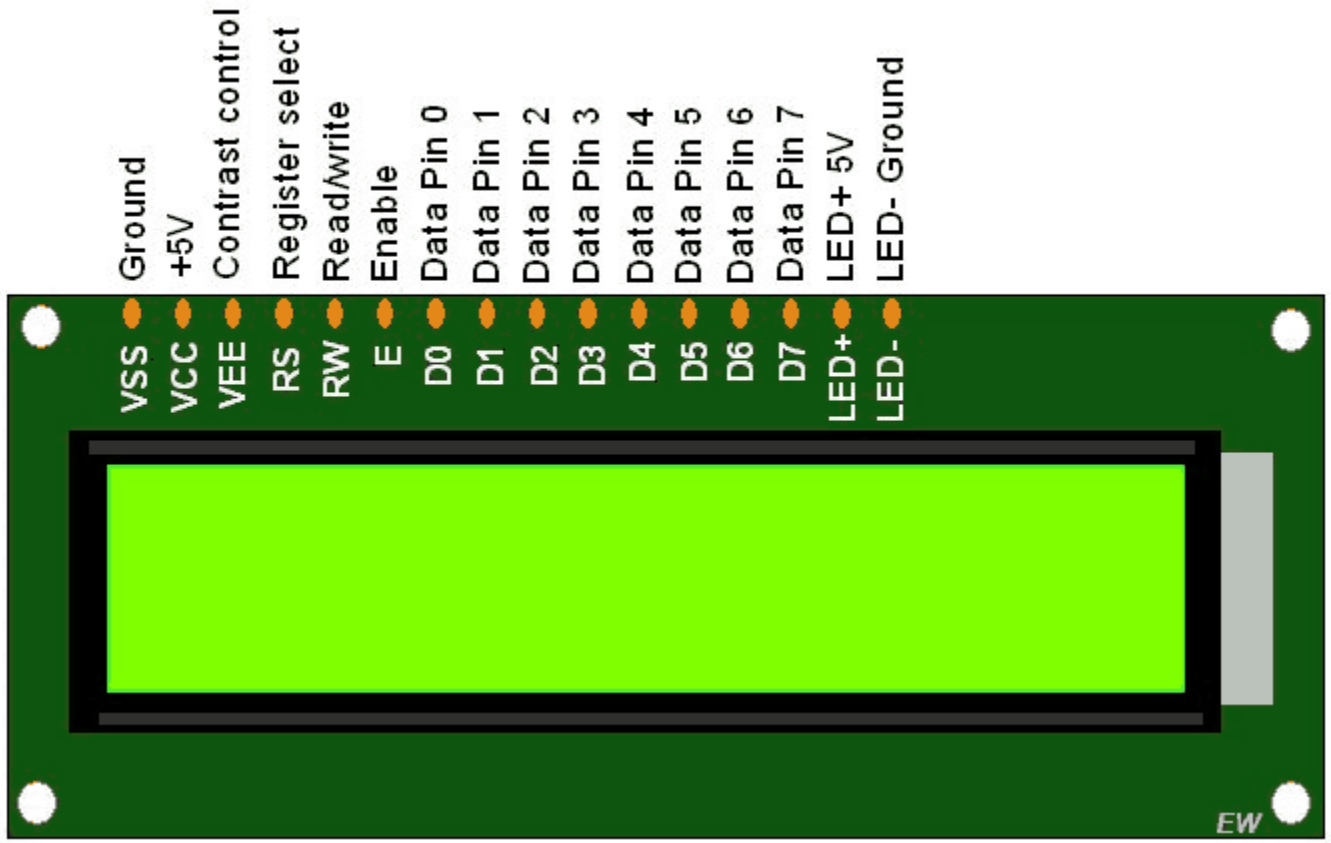
Human-Robot Interaction Laboratory

12.1 UART, I²C, SPI 비교



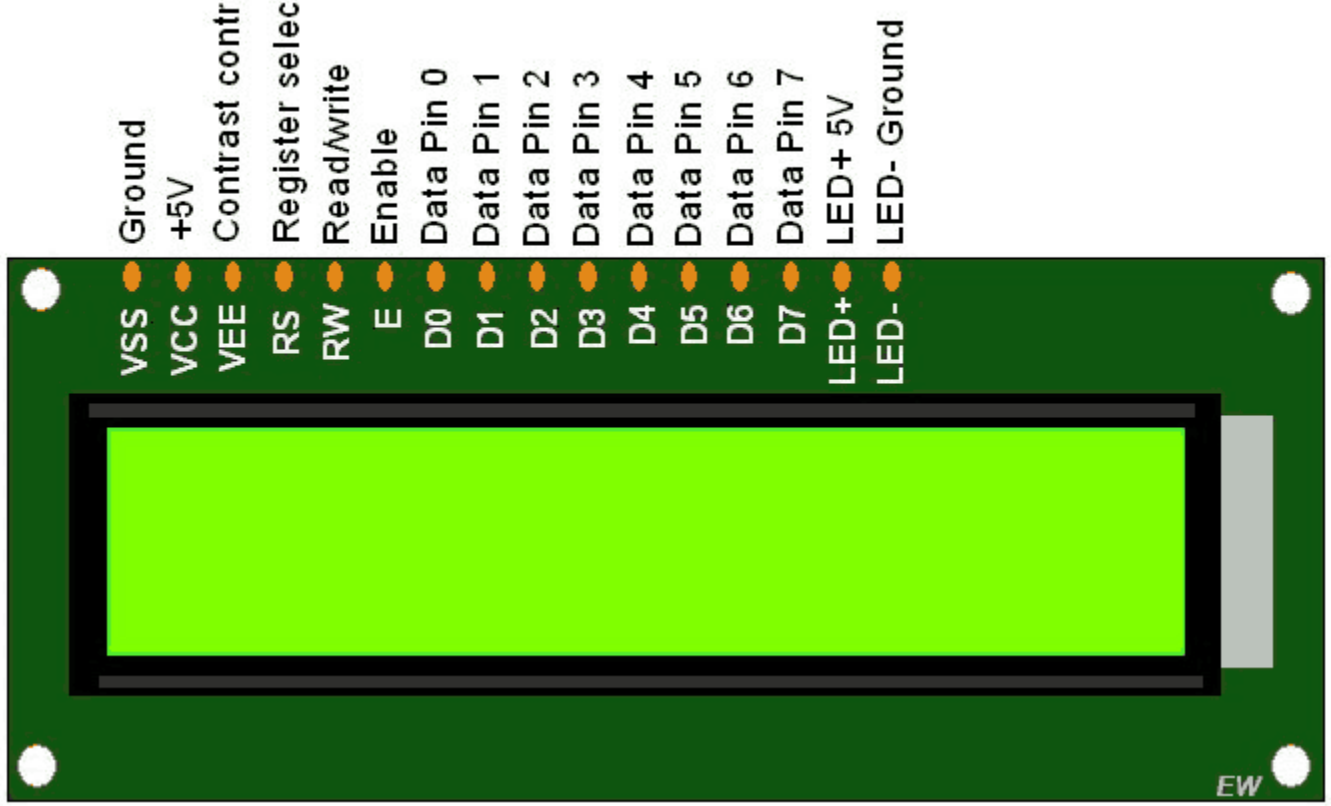
12.2 LCD 16X2를 GPIO, UART와 I²C로 제어 비교

- 아래의 LCD는 GPIO, UART와 I2C로 제어 가능함
  - GPIO로 제어할 때,
    - GPIO가 많이 필요함 (그림처럼 최대 12개)
    - Parallel interface임
  - UART로 제어 시
    - UART 제어기 필요함 (따로 구매해야 함)
    - Serial interface 이므로 pin 연결이 간단함
  - I²C로 제어시
    - I²C 제어기 필요함 (따로 구매해야 함)
    - Serial interface 이므로 pin 연결이 간단함

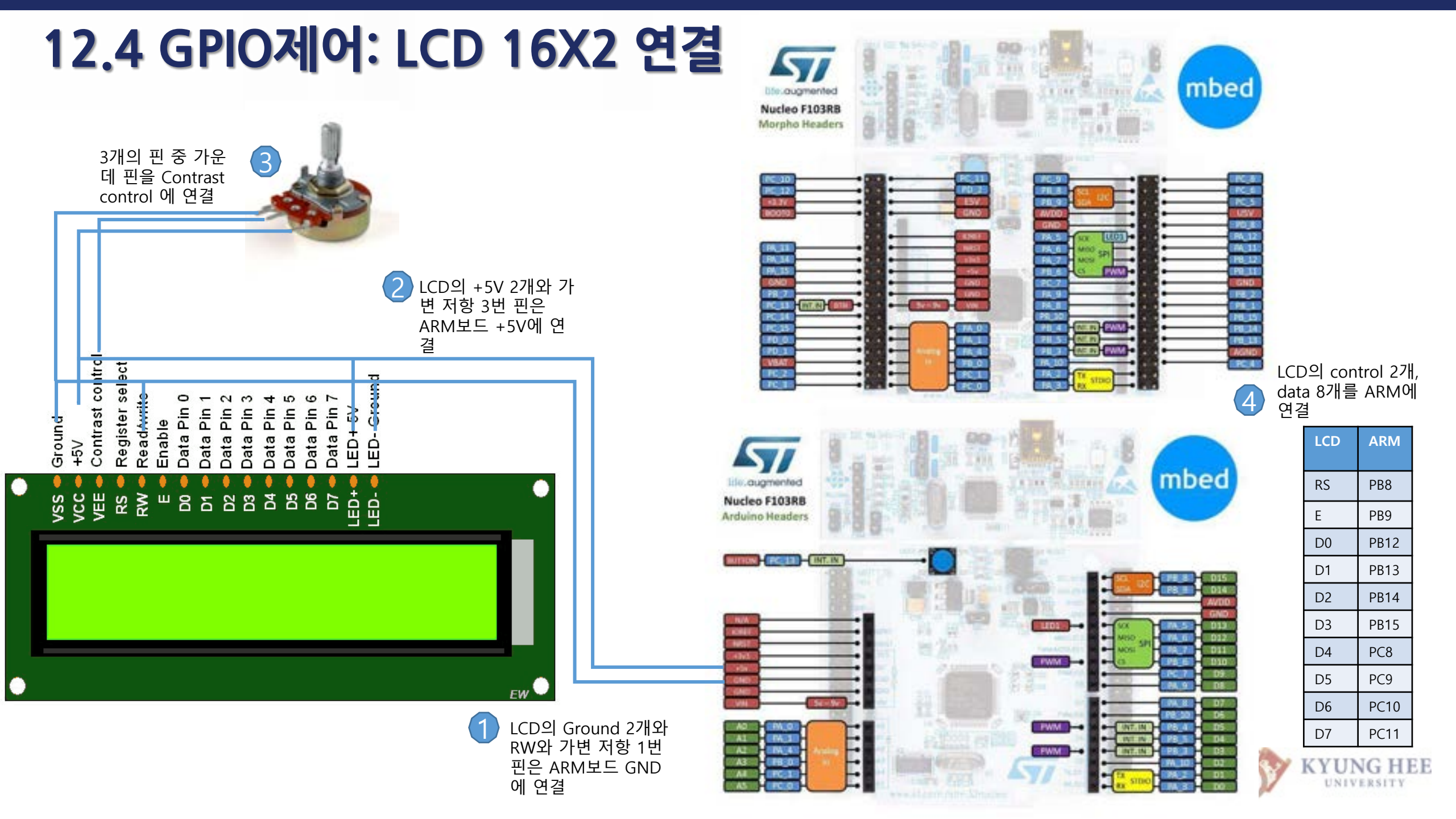


12.3 LCD 16X2 구조

- LCDs (Liquid Crystal Displays) are used for displaying status or parameters in embedded systems.
- LCD 16x2 is 16 pin device which has 8 data pins (D0-D7) and 3 control pins (RS, RW, EN). The remaining 5 pins are for supply and backlight for the LCD.
- The control pins help us configure the LCD in command mode or data mode. They also help configure read mode or write mode and also when to read or write.
- LCD 16x2 can be used in 4-bit mode or 8-bit mode depending on the requirement of the application. In order to use it we need to send certain commands to the LCD in command mode and once the LCD is configured according to our need, we can send the required data in data mode to display on it.



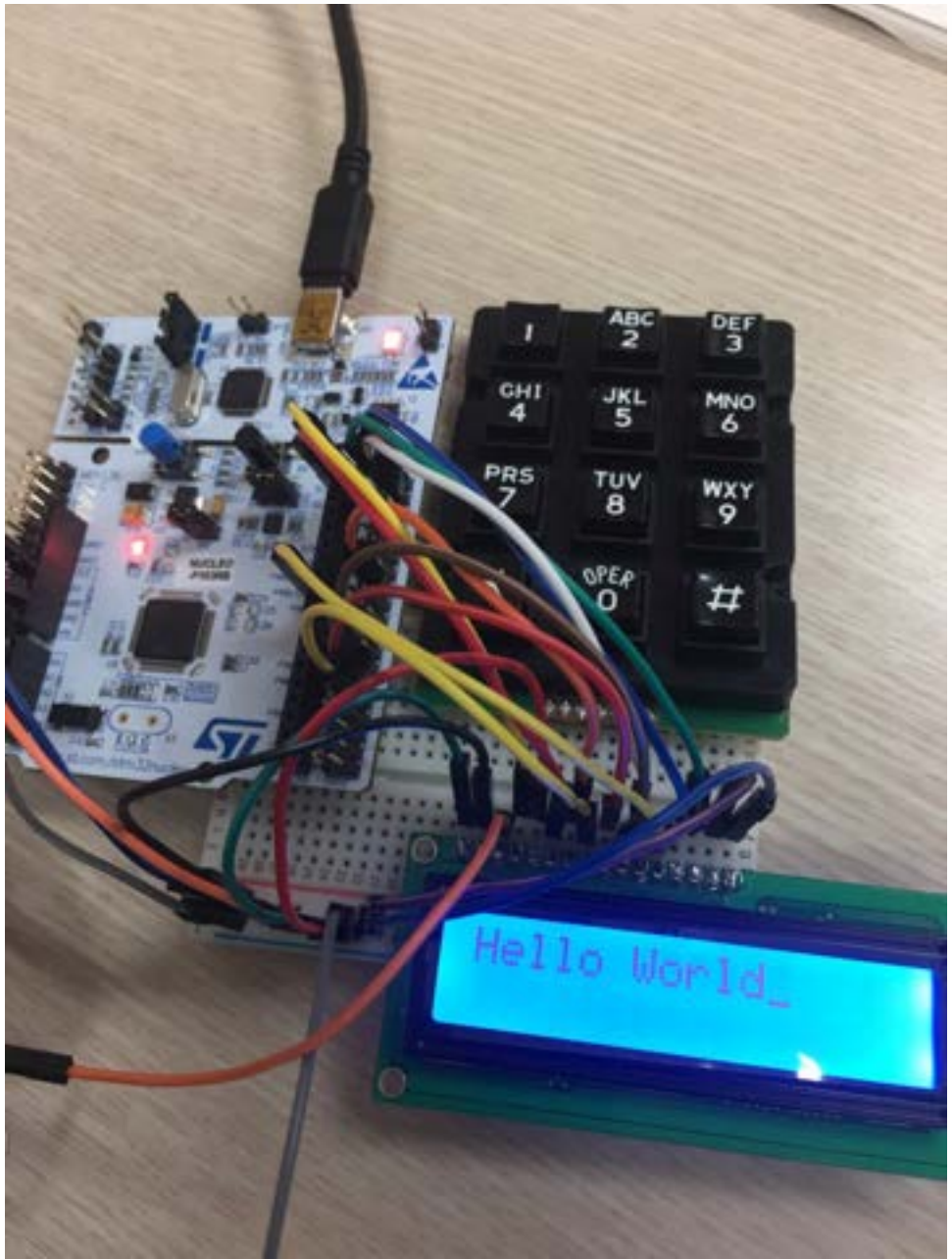
12.4 GPIO제어: LCD 16X2 연결



LCD	ARM
RS	PB8
E	PB9
D0	PB12
D1	PB13
D2	PB14
D3	PB15
D4	PC8
D5	PC9
D6	PC10
D7	PC11

KYUNG HEE UNIVERSITY

12.4 GPIO제어: LCD 16X2 연결 - 완성 사진



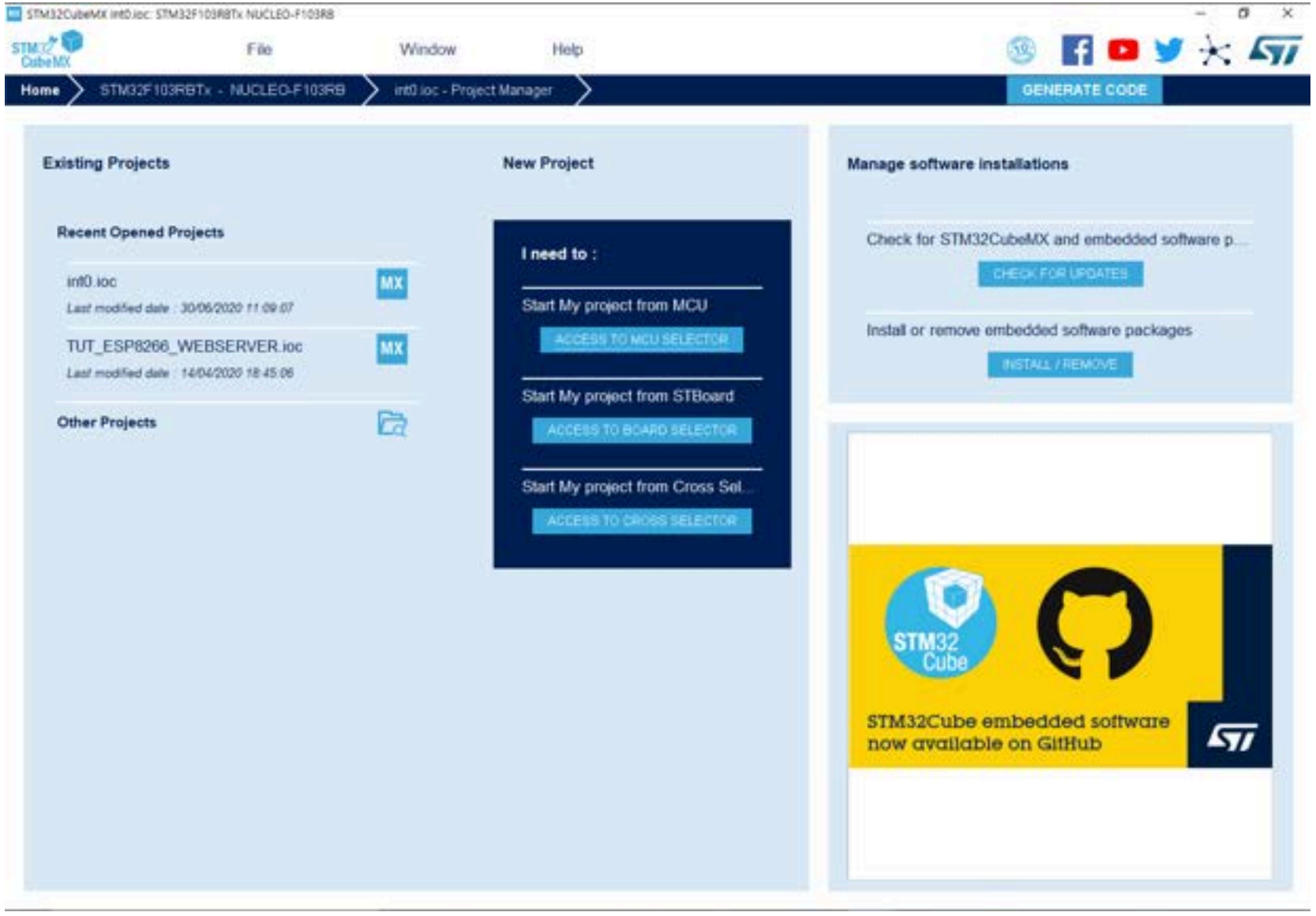
KYUNG HEE UNIVERSITY

12.4 GPIO제어: LCD 16X2 연결 - CubeMX

CubeMX 사용

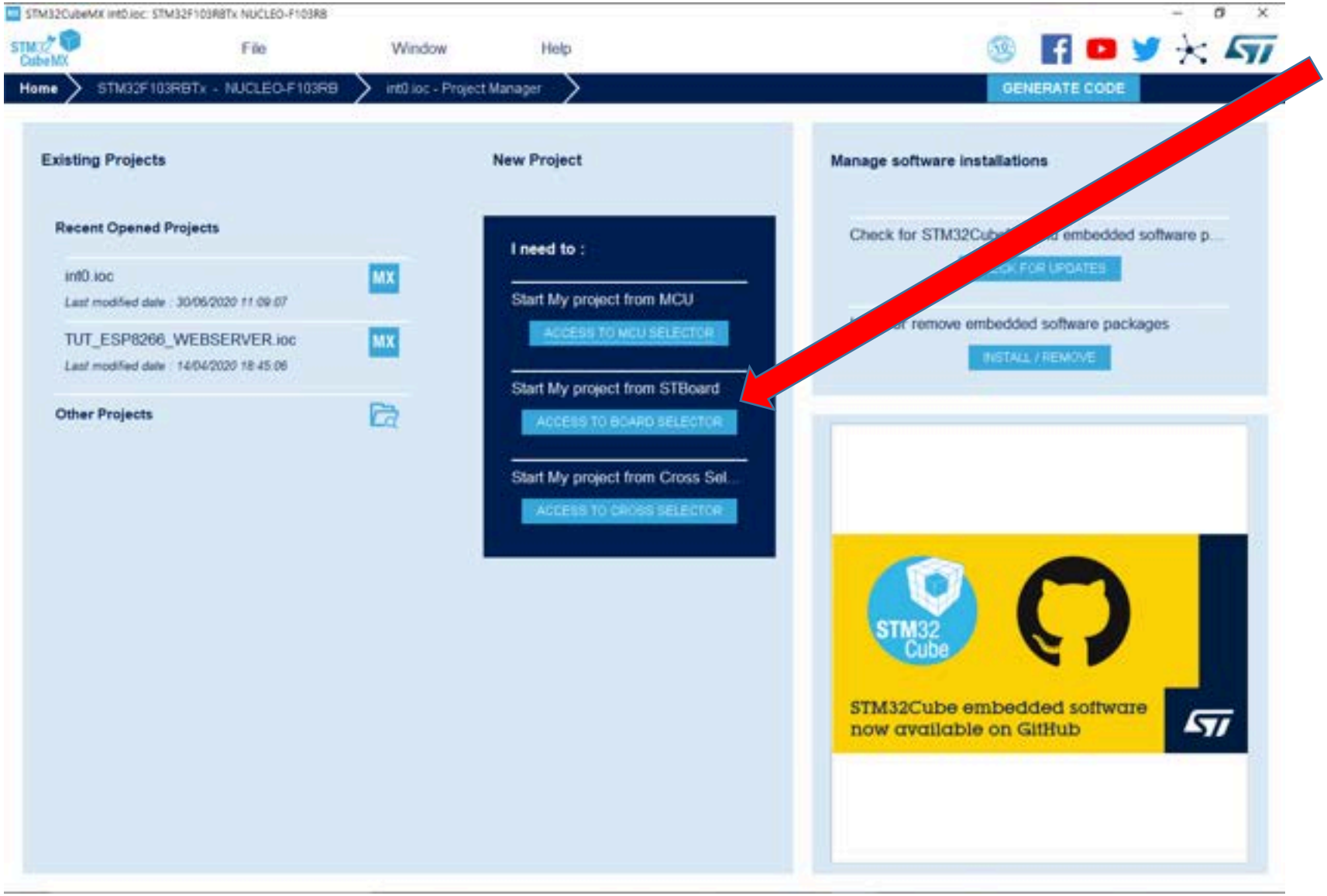
CubeMX 실행

초기화면



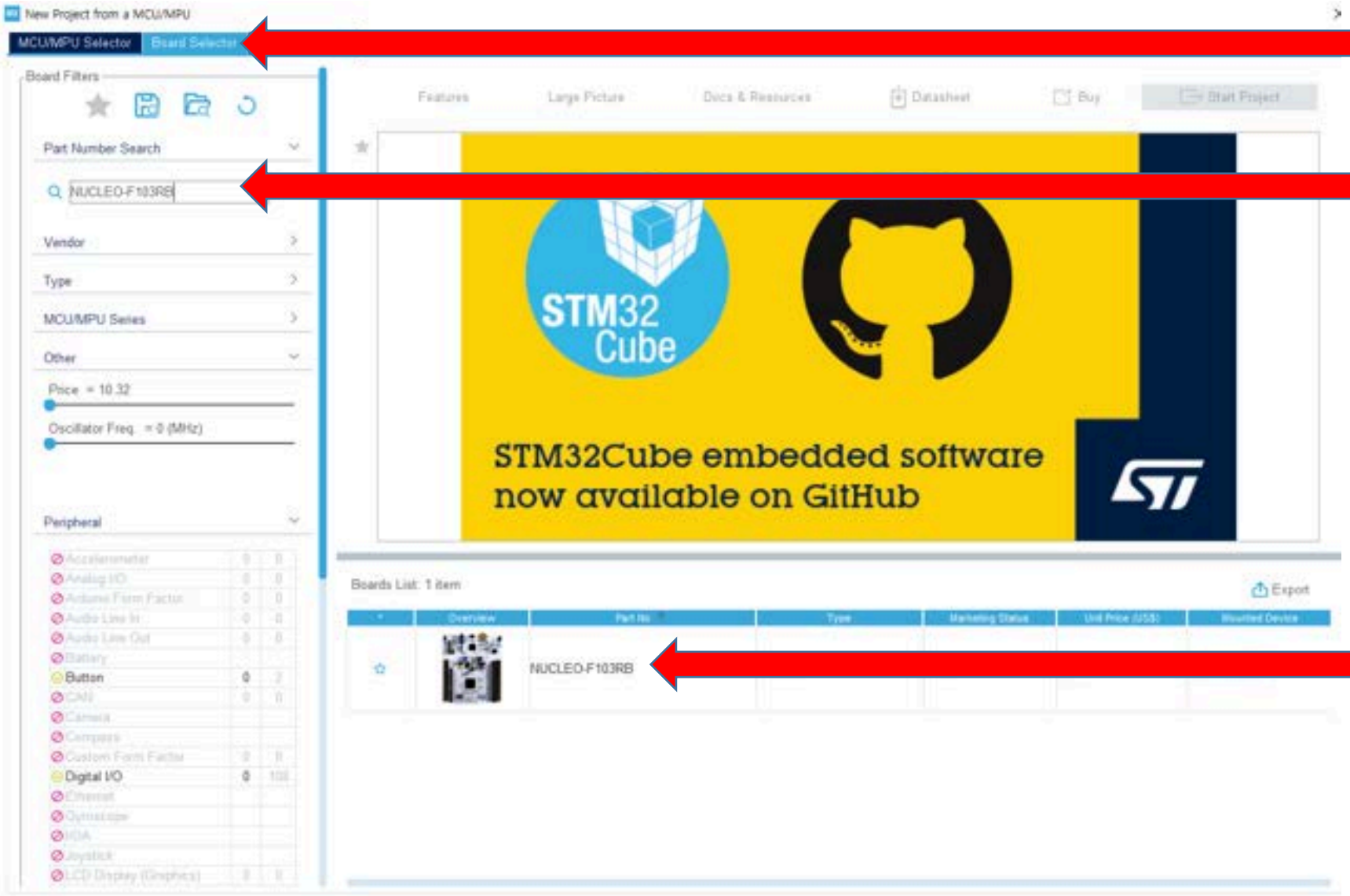
12.4 GPIO제어: LCD 16X2 연결 - CubeMX

CubeMX로 예제 11.1 구현하기



보드 선택

12.4 GPIO제어: LCD 16X2 연결 - CubeMX



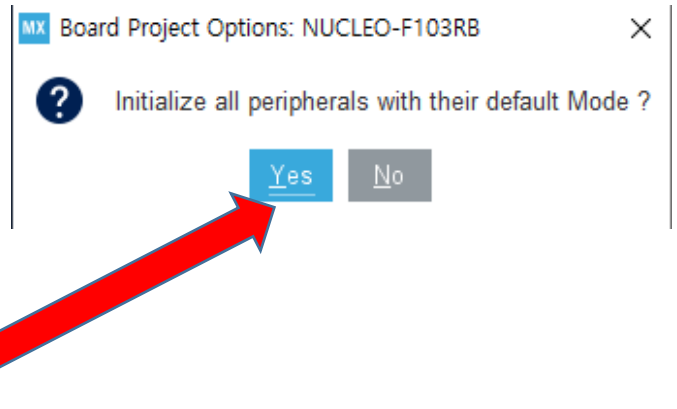
보드 선택

NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

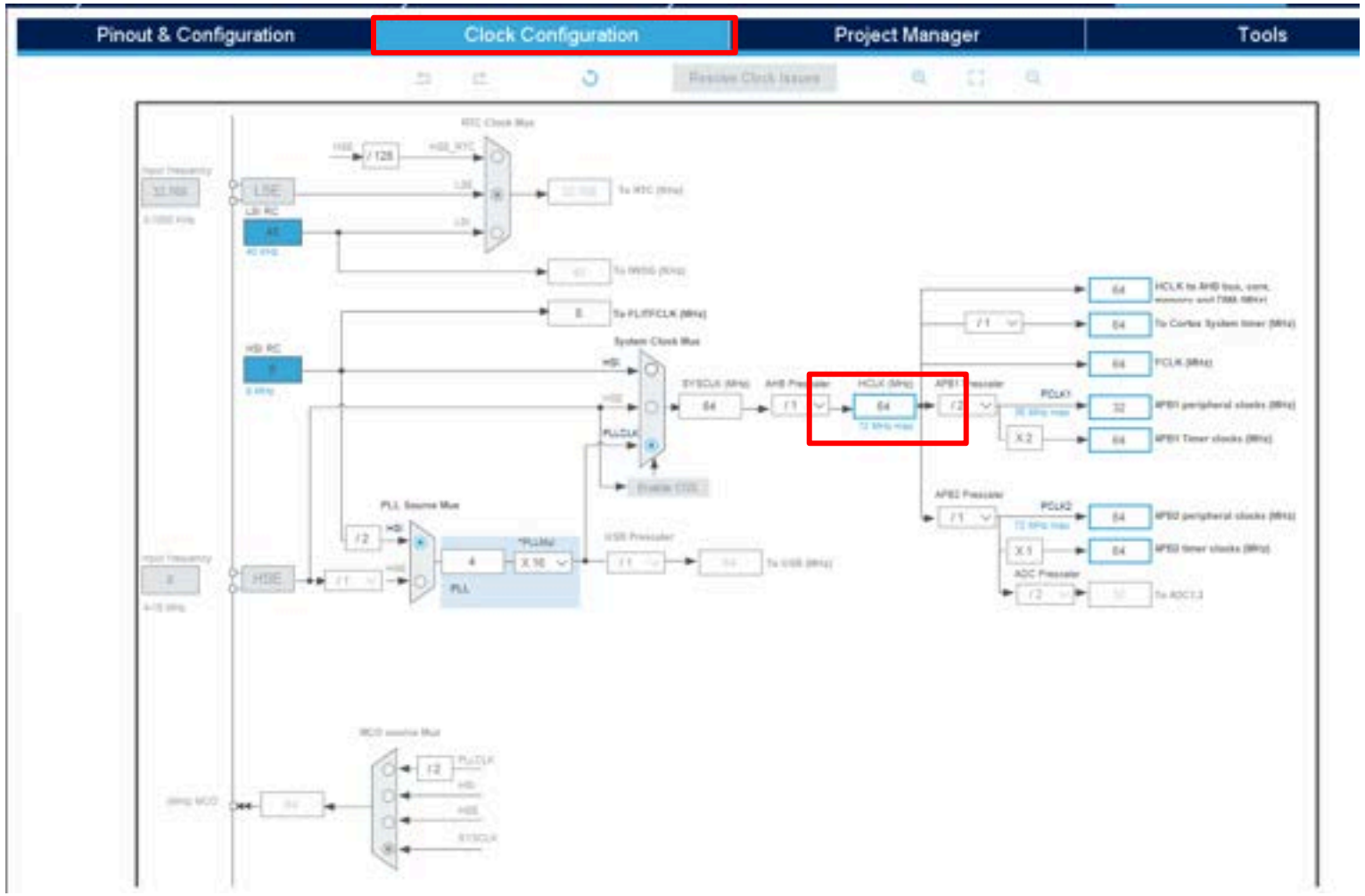
KYUNG HEE UNIVERSITY

12.4 GPIO제어: LCD 16X2 연결 - CubeMX



Yes 선택하면 오른쪽과 같이 칩이 보임

12.4 GPIO제어: LCD 16X2 연결 - CubeMX



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

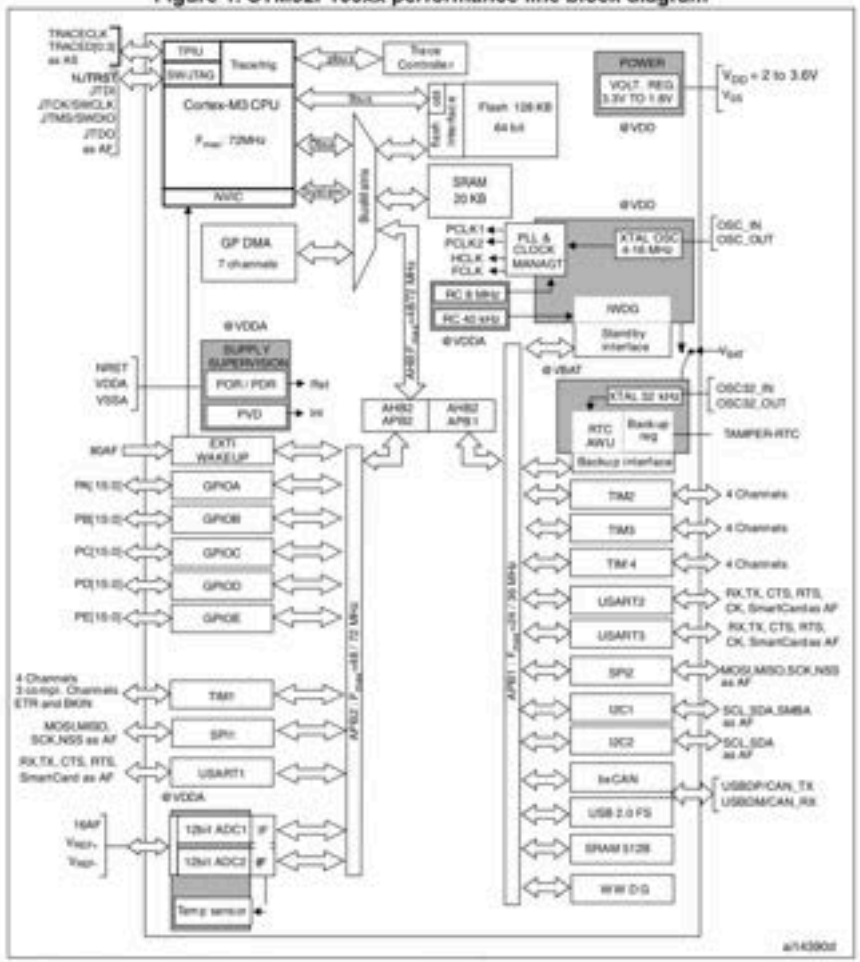
12.4 GPIO제어: LCD 16X2 연결 - CubeMX

```
  *       APB2 Prescaler          = 1
  *       PLLMUL                  = 16
  *       Flash Latency(WS)       = 2
  */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
        RCC_ClkInitTypeDef clkinitstruct = {0};
        RCC_OscInitTypeDef oscinitstruct = {0};

        /* Configure PLL ------------------------------------------------------*/
        /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
        /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
        /* Enable HSI and activate PLL with HSi_DIV2 as source */
        oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
        oscinitstruct.HSEState        = RCC_HSE_OFF;
        oscinitstruct.LSEState        = RCC_LSE_OFF;
        oscinitstruct.HSIState        = RCC_HSI_ON;
        oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
        oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
        oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
        oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
        oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

        if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }

        /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
            clocks dividers */
        clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
        clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
        clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
        clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
        clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

        if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }
}

// ------------------------------------------------------------------------- //
//  -- <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```

Done

34 of 42

Figure 1. STM32F103xx performance line block diagram



1. T_A = –40 °C to +105 °C (junction temperature up to 125 °C).
2. AF = alternate function on I/O port pin.

12.4 GPIO제어: LCD 16X2 연결 - CubeMX

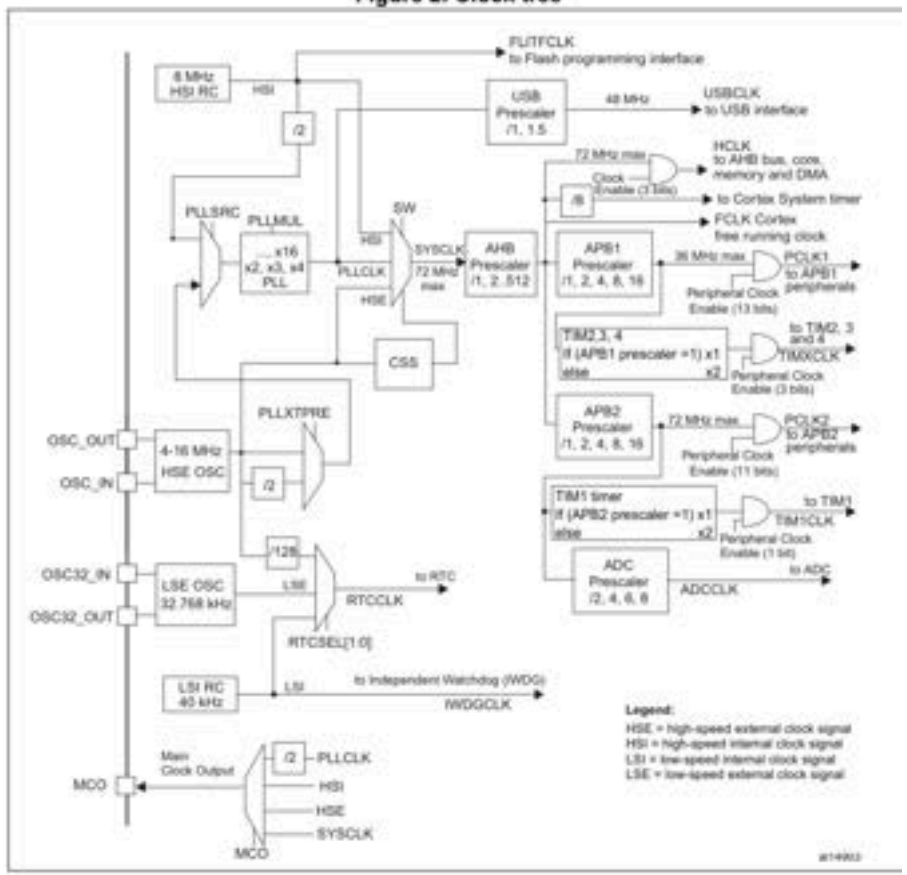
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
	RCC_ClkInitTypeDef clkinitstruct = {0};
	RCC_OscInitTypeDef oscinitstruct = {0};

	/* Configure PLL ------------------------------------------------------*/
	/* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
	/* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
	/* Enable HSI and activate PLL with HSi_DIV2 as source */
	oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
	oscinitstruct.HSEState        = RCC_HSE_OFF;
	oscinitstruct.LSEState        = RCC_LSE_OFF;
	oscinitstruct.HSIState        = RCC_HSI_ON;
	oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
	oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
	oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
	oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
	oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

	if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
		/* Initialization Error */
		while(1);
	}

	/* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
	    clocks dividers */
	clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
	clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
	clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
	clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
	clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

	if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
		/* Initialization Error */
		while(1);
	}
}
```

Done 34 of 42

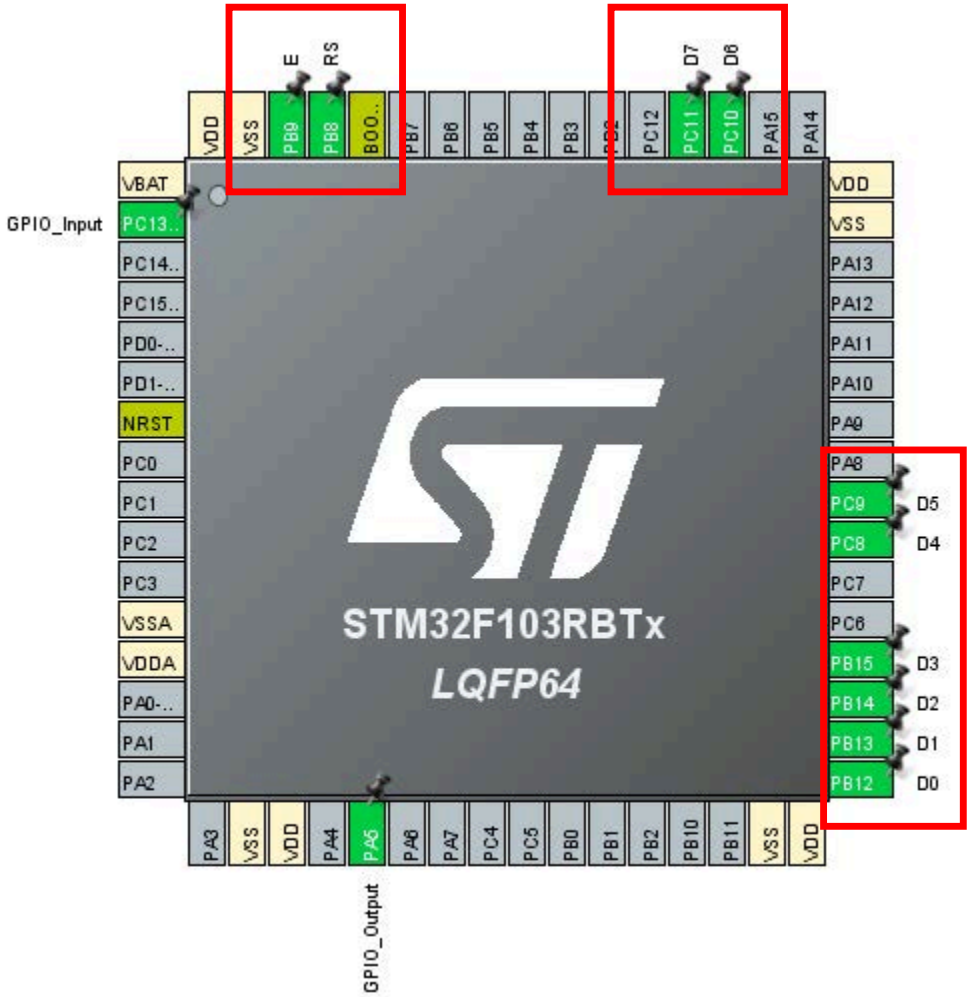
Figure 2. Clock tree



1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

12.4 GPIO제어: LCD 16X2 연결 - CubeMX

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

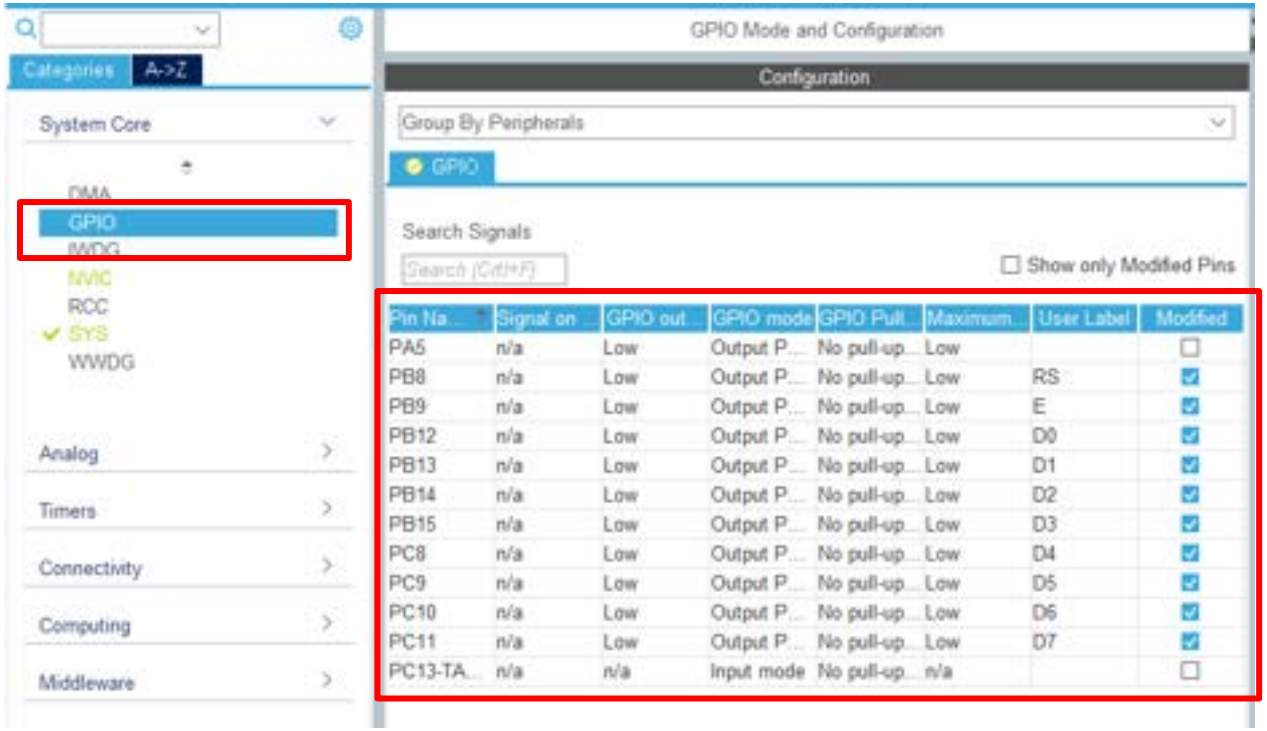


Pinout & Configuration 탭에서

GPIO 출력핀 설정

LCD	ARM
RS	PB8
E	PB9
D0	PB12
D1	PB13
D2	PB14
D3	PB15
D4	PC8
D5	PC9
D6	PC10
D7	PC11

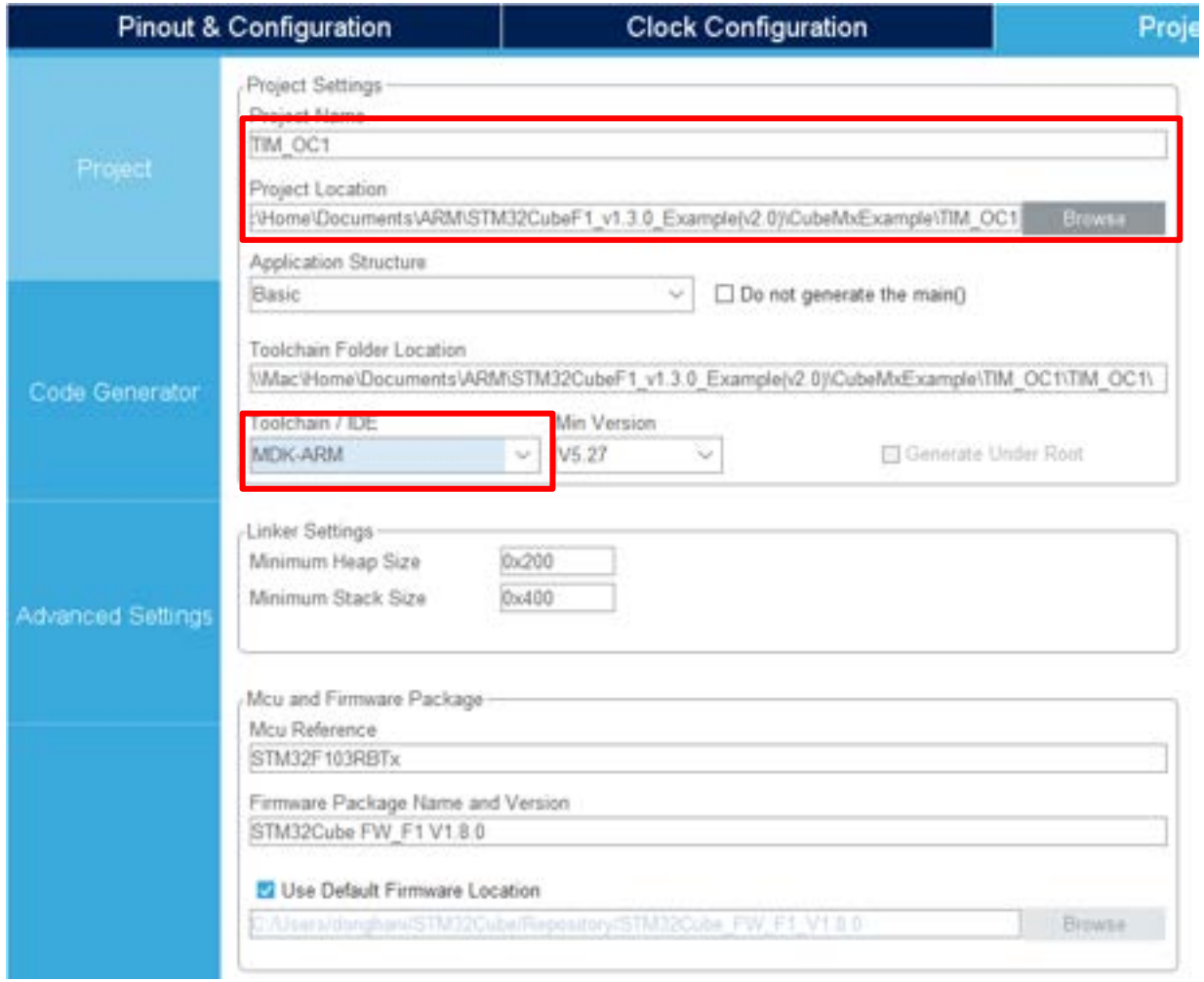
12.4 GPIO제어: LCD 16X2 연결 - CubeMX



GPIO 확인

12.4 GPIO제어: LCD 16X2 연결 - CubeMX

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기



Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

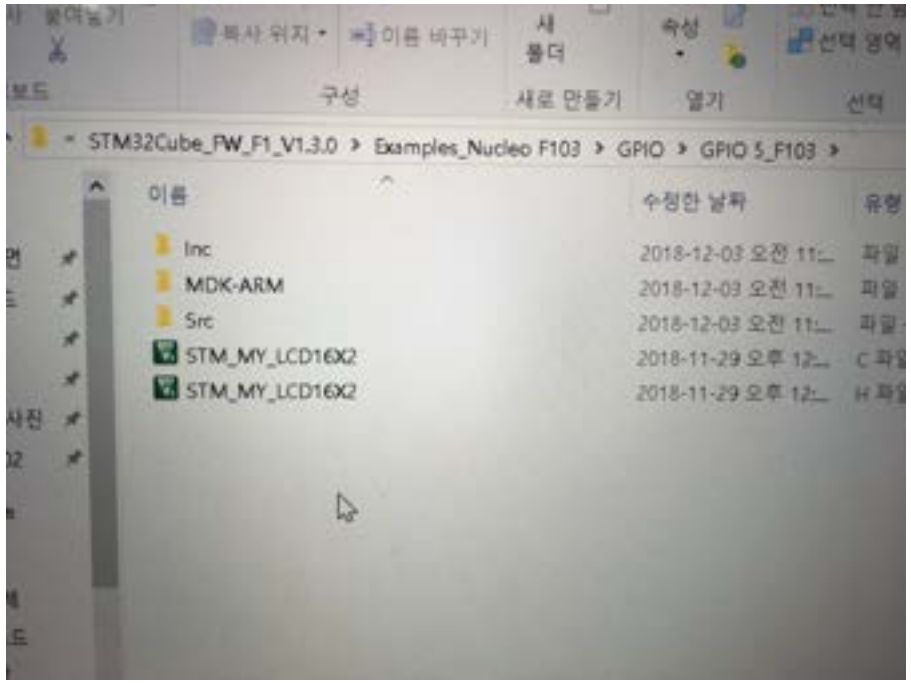
Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

12.4 GPIO제어: LCD 16X2 연결 - 프로그래밍

1. STM_MY_LCD16X2.c와 STM_MY_LCD16X2.h를 프로젝트 폴더에 추가

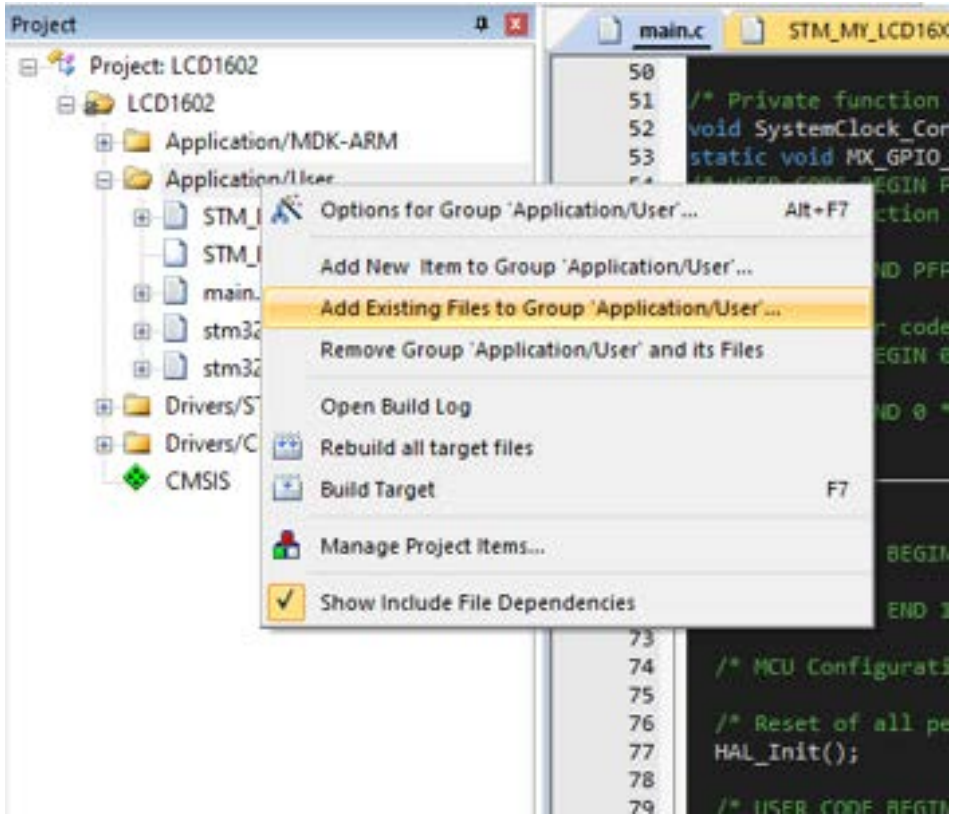
- STM_MY_LCD16X2.c와 STM_MY_LCD16X2.h를 KLAS에서/인터넷에서 다운로드 후 프로젝트 project 폴더에 옮기기

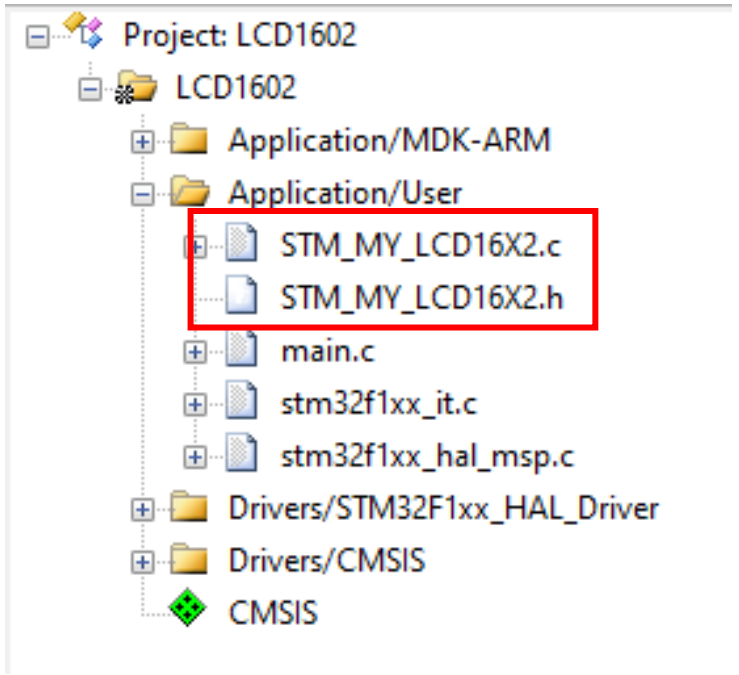


KYUNG HEE UNIVERSITY

12.4 GPIO제어: LCD 16X2 연결 - 프로그래밍

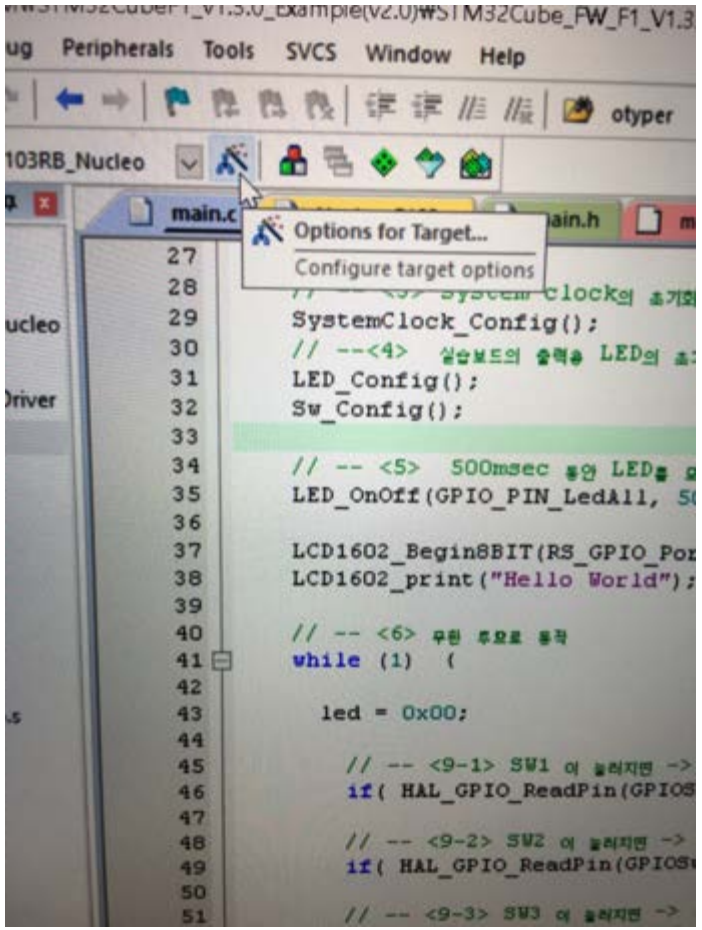
2. Keil uVision 프로그램에서 왼쪽의 프로젝트 창에서 Application/User 폴더를 오른쪽 클릭하고, Add Existing Files to Group을 클릭한다. 그리고, 앞슬라이드의 STM_MY_LCD16X2.c와 STM_MY_LCD16X2.h를 각각 add 한다.

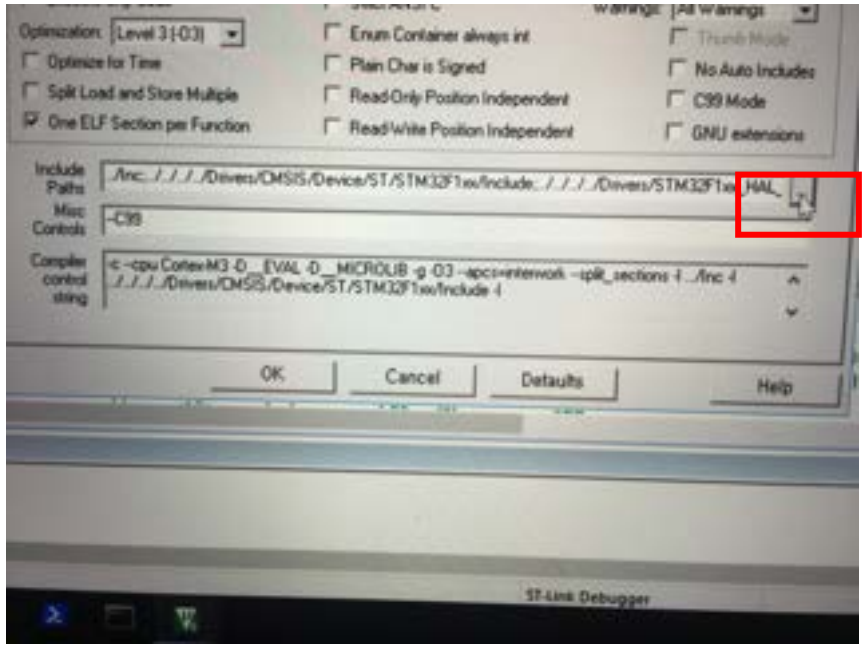


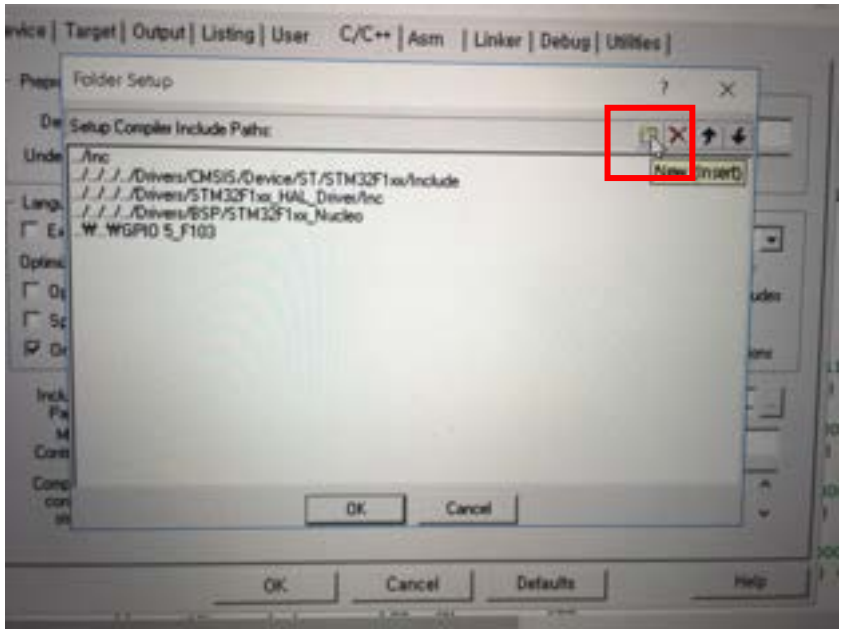


12.4 GPIO제어: LCD 16X2 연결 - 프로그래밍

3. Keil uVision 프로그램에서 상단의 Options for Target 아이콘을 클릭하고, 열린 창에서 C/C++ 탭을 클릭한다. 오른쪽 하단에 있는 Include Paths의 오른쪽 끝의 버튼을 클릭한다. Folder Setup 창이 열리면 오른쪽 상단의 New 버튼을 누르고, STM_MY_LCD16X2.c와 STM_MY_LCD16X2.h를 옮겨논 폴더를 path에 추가한다.







KYUNG HEE UNIVERSITY

12.4 GPIO제어: LCD 16X2 연결 - CubeMX

```
21  /* Includes ------------------------------
22  #include "main.h"
23
24  /* Private includes ----------------------
25  /* USER CODE BEGIN Includes */
26  #include "STM_MY_LCD16X2.h"
27  #include <stdbool.h>
28  /* USER CODE END Includes */
```

→ main() 함수 앞 쪽에 #include "STM_MY_LCD16X2.h"
#include <stdbool.h> 추가

12.4 GPIO제어: LCD 16X2 연결 - 프로그래밍

4. main.h 파일 상단에 다음과 같이 define을 추가한다.

main.c | STM_MY_LCD16X2.c | STM_MY_LCD16X2.h | main.h

```
53  /* Exported functions prototypes ---------------------------
54  void Error_Handler(void);
55
56  /* USER CODE BEGIN EFP */
57
58  /* USER CODE END EFP */
59
60  /* Private defines -----------------------------------------
61  #define D0_Pin GPIO_PIN_12
62  #define D0_GPIO_Port GPIOB
63  #define D1_Pin GPIO_PIN_13
64  #define D1_GPIO_Port GPIOB
65  #define D2_Pin GPIO_PIN_14
66  #define D2_GPIO_Port GPIOB
67  #define D3_Pin GPIO_PIN_15
68  #define D3_GPIO_Port GPIOB
69  #define D4_Pin GPIO_PIN_8
70  #define D4_GPIO_Port GPIOC
71  #define D5_Pin GPIO_PIN_9
72  #define D5_GPIO_Port GPIOC
73  #define D6_Pin GPIO_PIN_10
74  #define D6_GPIO_Port GPIOC
75  #define D7_Pin GPIO_PIN_11
76  #define D7_GPIO_Port GPIOC
77  #define RS_Pin GPIO_PIN_8
78  #define RS_GPIO_Port GPIOB
79  #define E_Pin GPIO_PIN_9
80  #define E_GPIO_Port GPIOB
81  /* USER CODE BEGIN Private defines */
```

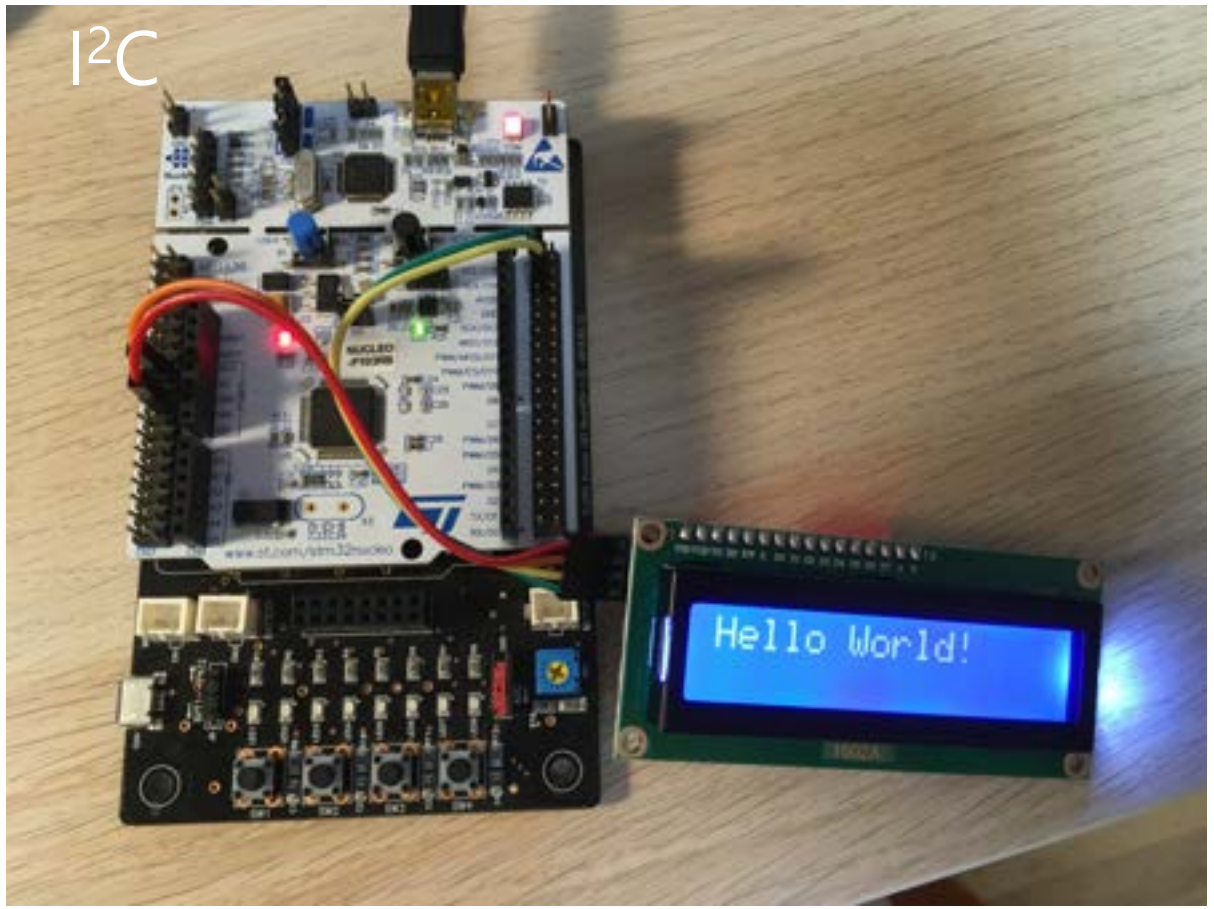
12.4 GPIO제어: LCD 16X2 연결 - 프로그래밍

```
89
90    /* Initialize all configured peripherals */
91    MX_GPIO_Init();
92    /* USER CODE BEGIN 2 */
93    LCD1602_Begin8BIT(RS_GPIO_Port, RS_Pin, E_Pin, D0_GPIO_Port, D0_Pin, D1_Pin, D2_Pin, D3_Pin, D4_GPIO_Port, D4_Pin, D5_Pin, D6_Pin, D7_Pin);
94    LCD1602_print("Hello World");
95
96    /* USER CODE END 2 */
97
98    /* Infinite loop */
99    /* USER CODE BEGIN WHILE */
100   while (1)
101   {
102
103     /* USER CODE END WHILE */
104
105     /* USER CODE BEGIN 3 */
106
107   }
108   /* USER CODE END 3 */
109 }
```

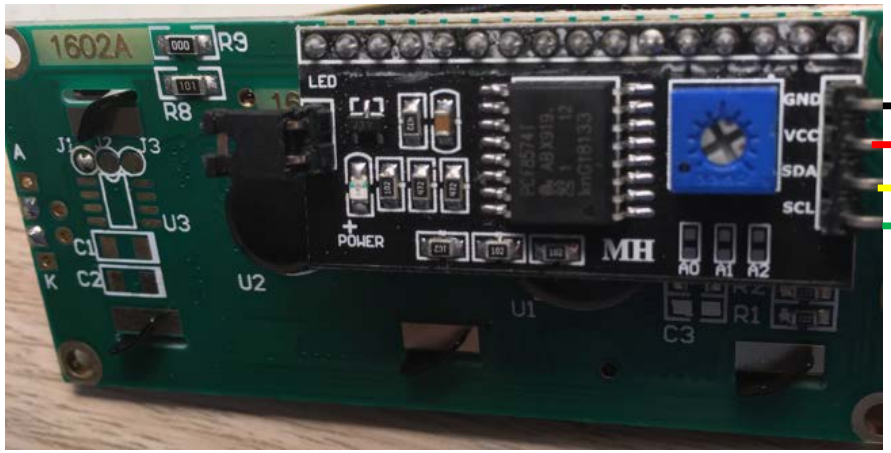
12.4 GPIO제어: LCD 16X2 연결 - 참고 자료

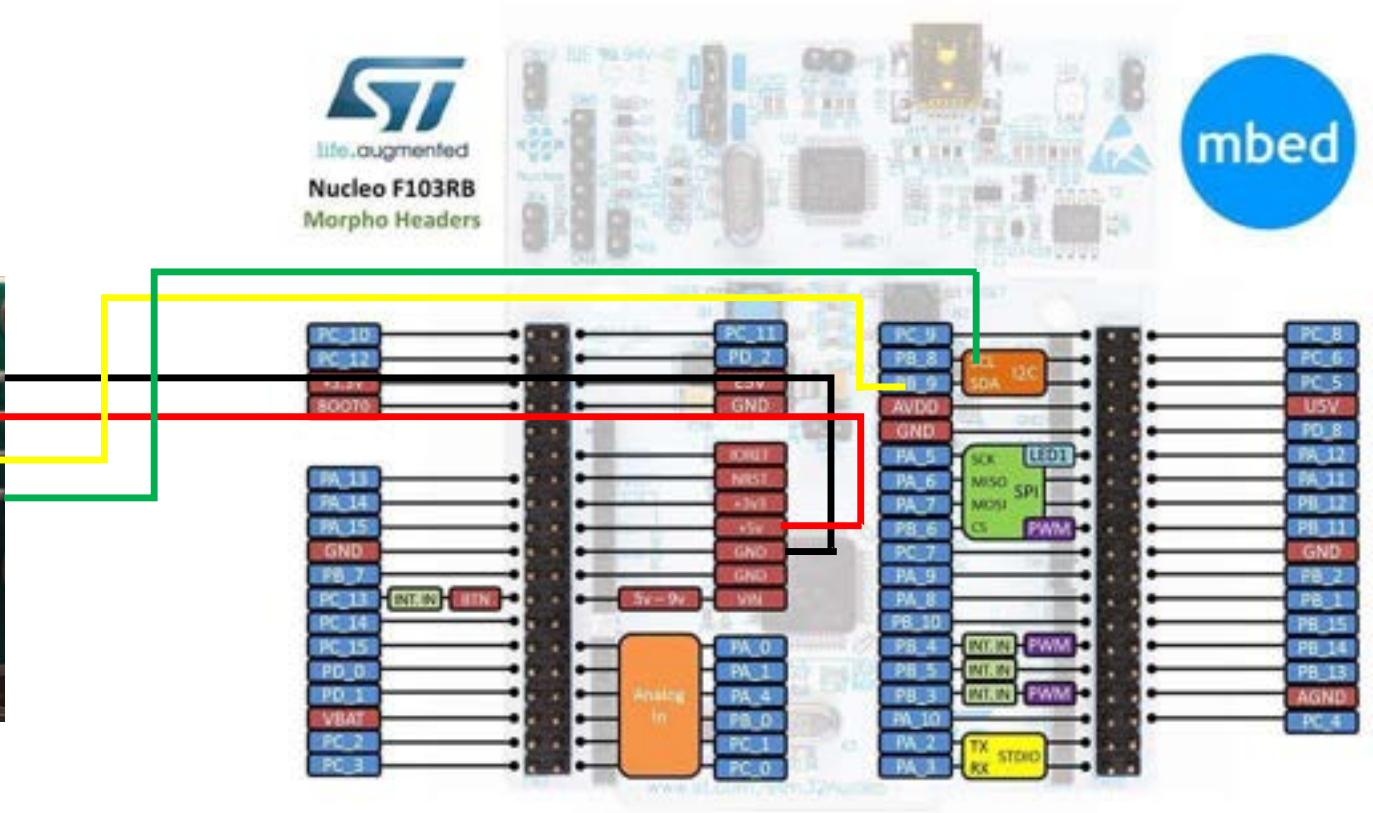
- STM32F4 Discovery 보드에 CubeMX, Keil uVision으로 만드는 과정 영상:
  https://www.youtube.com/watch?v=zfn5YqFlqbc

12.5 I²C제어: LCD 16X2 연결 - 완성사진



12.5 I²C제어 : LCD 16X2 연결



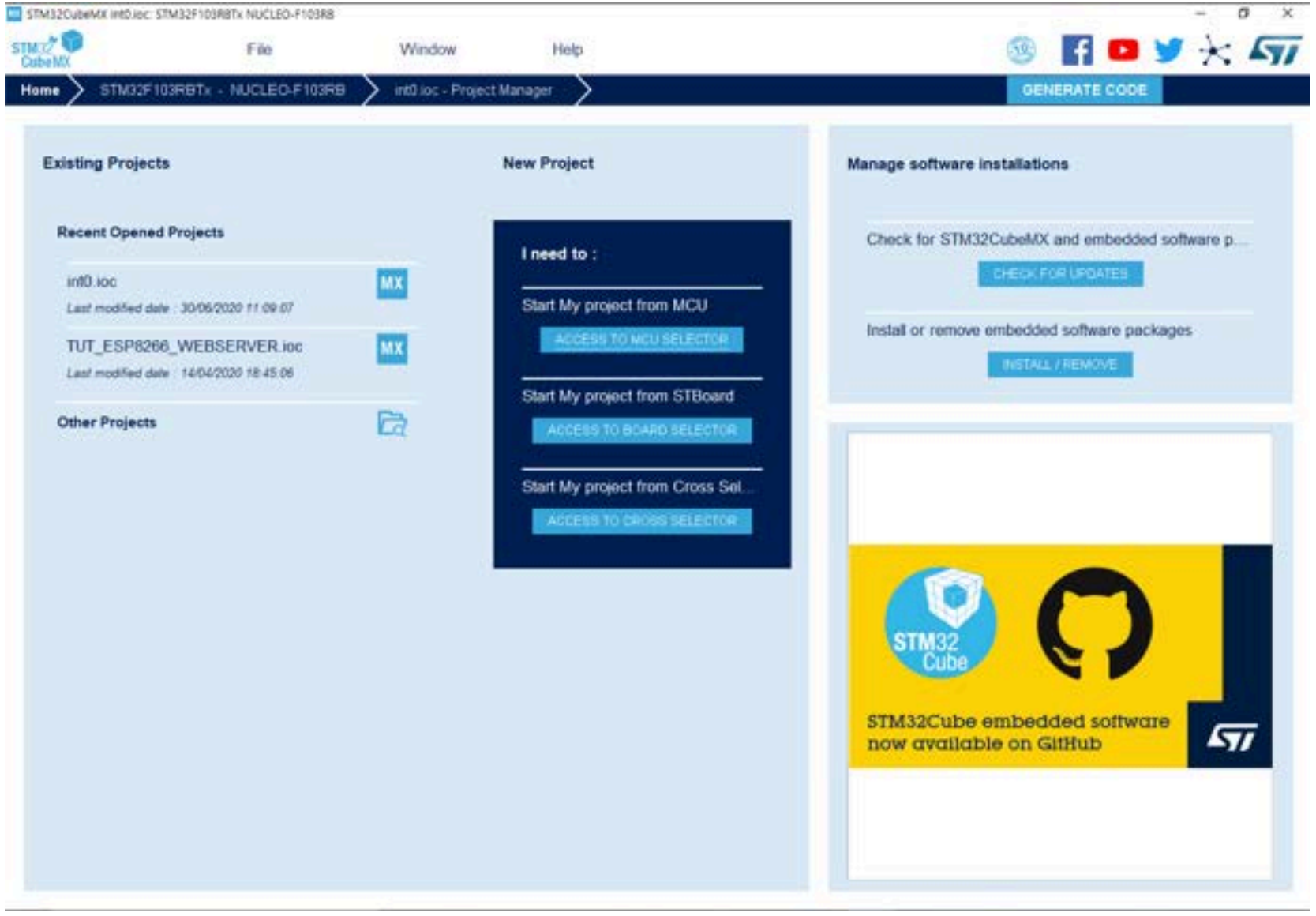


12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

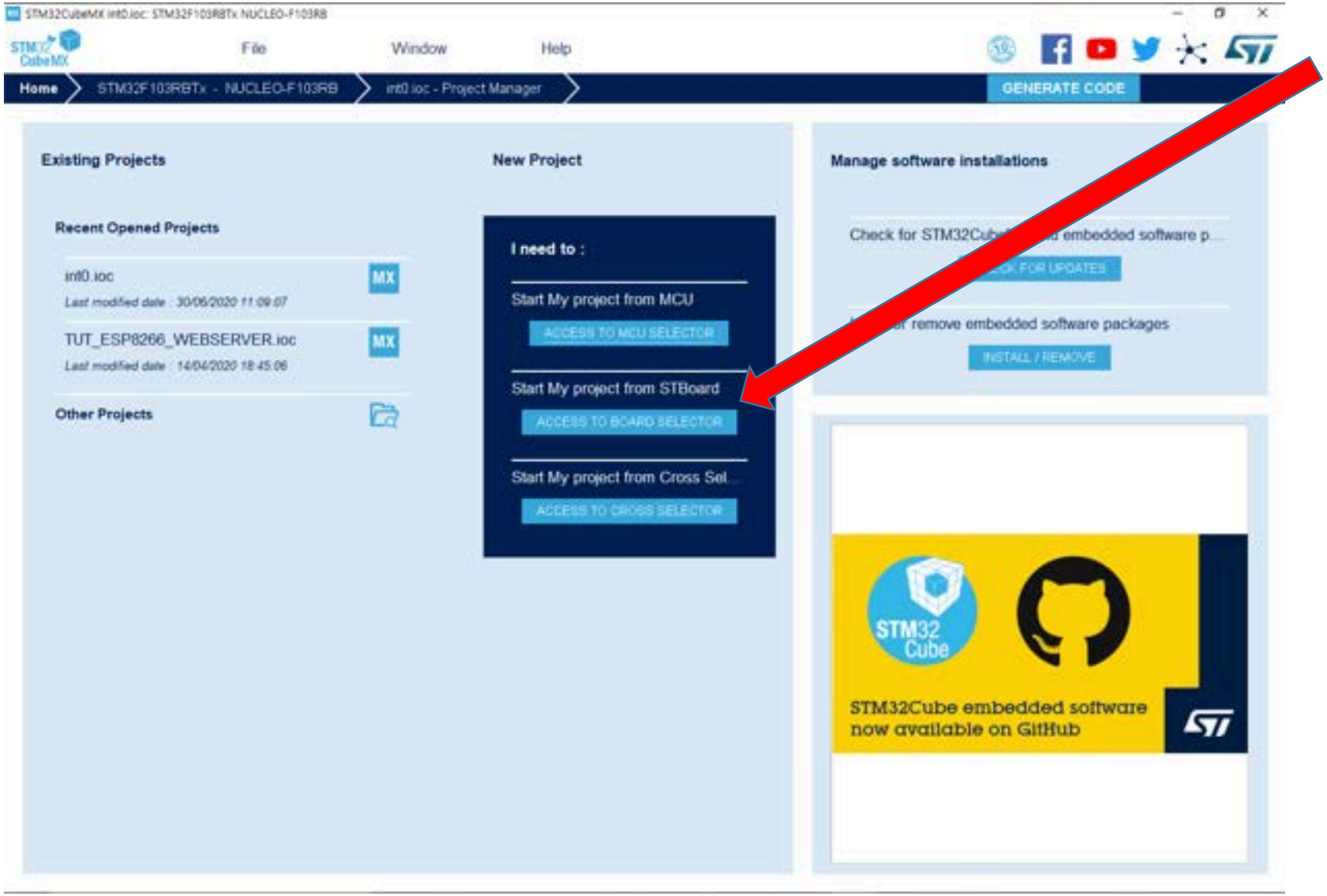
CubeMX 사용

CubeMX 실행

초기화면

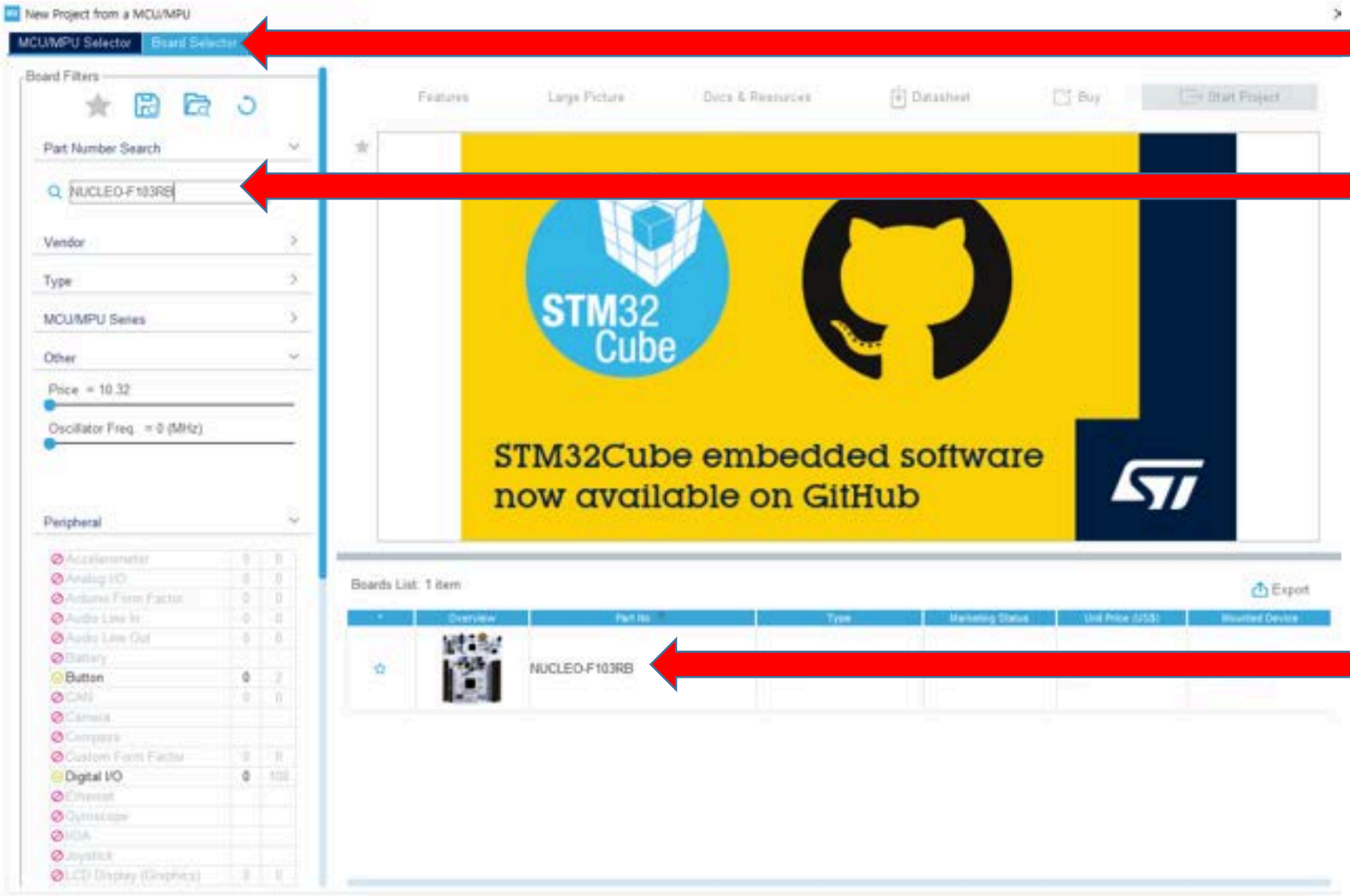


12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



보드 선택

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



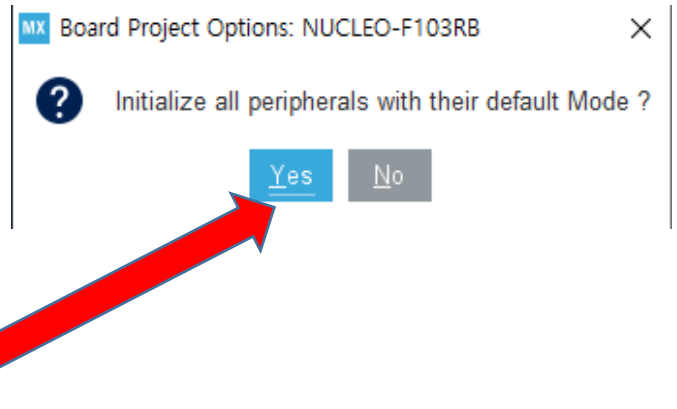
보드 선택

NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

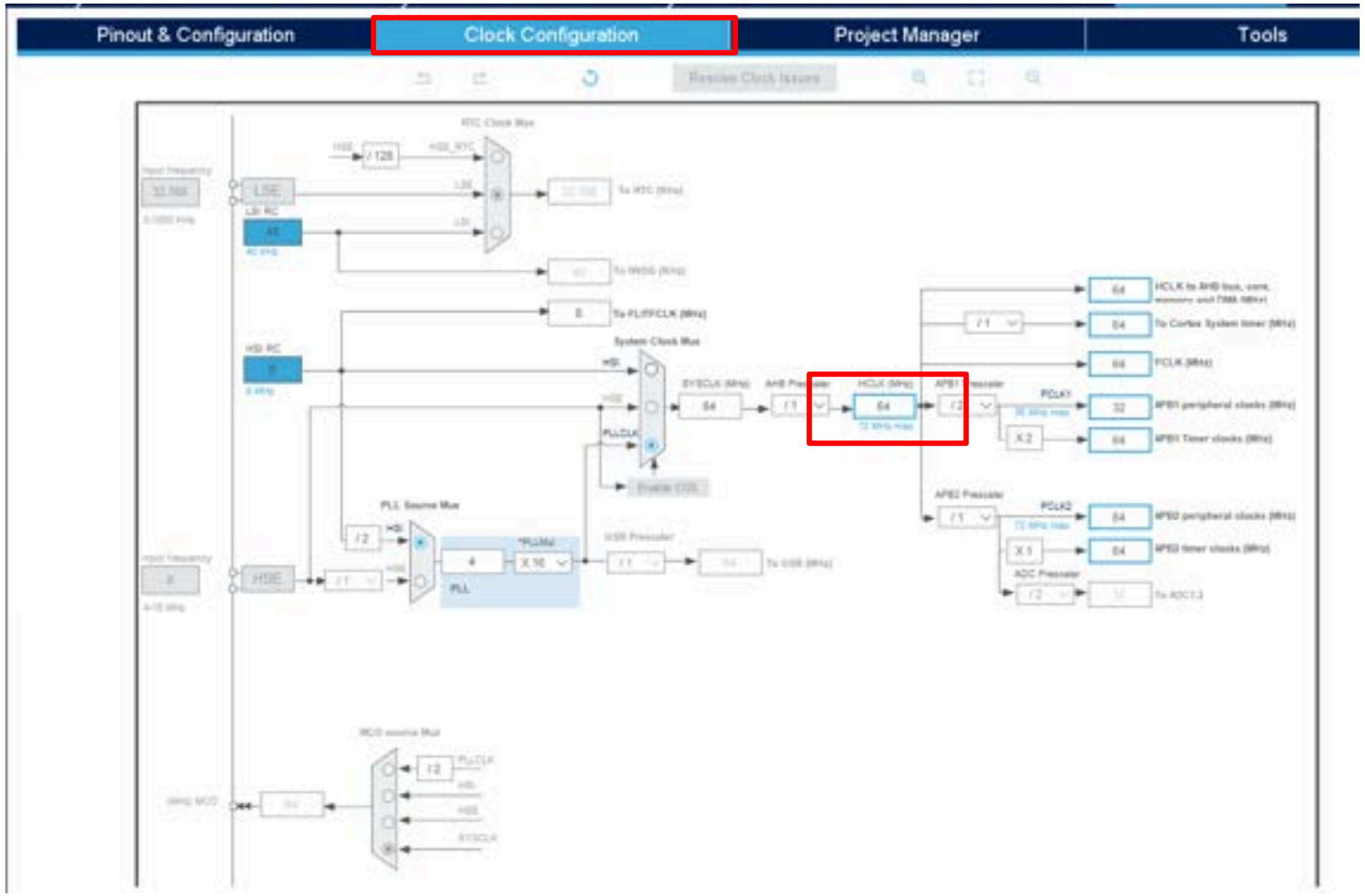
KYUNG HEE UNIVERSITY

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



Yes 선택하면 오른쪽과 같이 칩이 보임

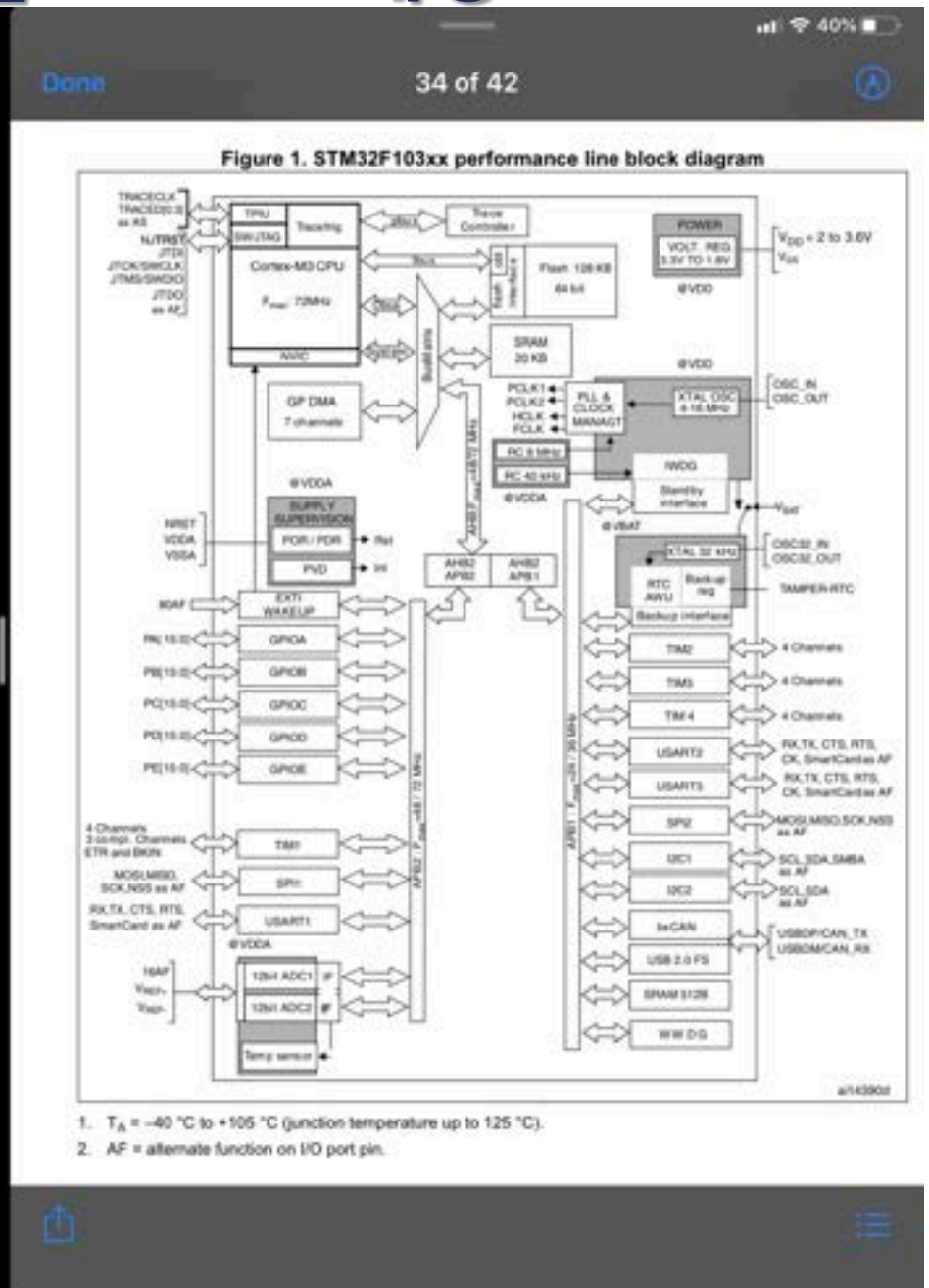
12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

```
  *       APB2 Prescaler          = 1
  *       PLLMUL                  = 16
  *       Flash Latency(WS)       = 2
  */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
        RCC_ClkInitTypeDef clkinitstruct = {0};
        RCC_OscInitTypeDef oscinitstruct = {0};

        /* Configure PLL ------------------------------------------------------*/
        /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
        /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
        /* Enable HSI and activate PLL with HSi_DIV2 as source */
        oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
        oscinitstruct.HSEState        = RCC_HSE_OFF;
        oscinitstruct.LSEState        = RCC_LSE_OFF;
        oscinitstruct.HSIState        = RCC_HSI_ON;
        oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
        oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
        oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
        oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
        oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

        if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }

        /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
            clocks dividers */
        clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
        clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
        clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
        clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
        clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

        if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }
}

// ------------------------------------------------------------------------- //
//  --  <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```



KYUNG HEE UNIVERSITY

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

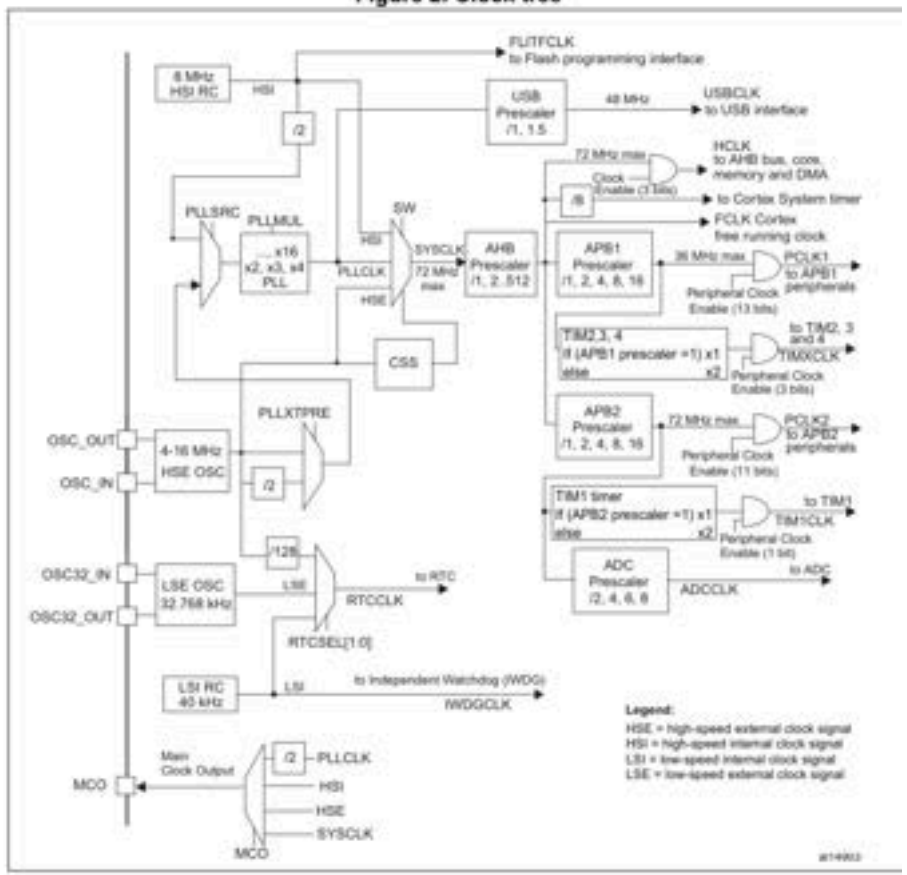
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
        RCC_ClkInitTypeDef clkinitstruct = {0};
        RCC_OscInitTypeDef oscinitstruct = {0};

        /* Configure PLL ------------------------------------------------------*/
        /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
        /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
        /* Enable HSI and activate PLL with HSi_DIV2 as source */
        oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
        oscinitstruct.HSEState        = RCC_HSE_OFF;
        oscinitstruct.LSEState        = RCC_LSE_OFF;
        oscinitstruct.HSIState        = RCC_HSI_ON;
        oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
        oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
        oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
        oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
        oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

        if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }

        /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
            clocks dividers */
        clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
        clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
        clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
        clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
        clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

        if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                /* Initialization Error */
                while(1);
        }
}
```

Done 34 of 42

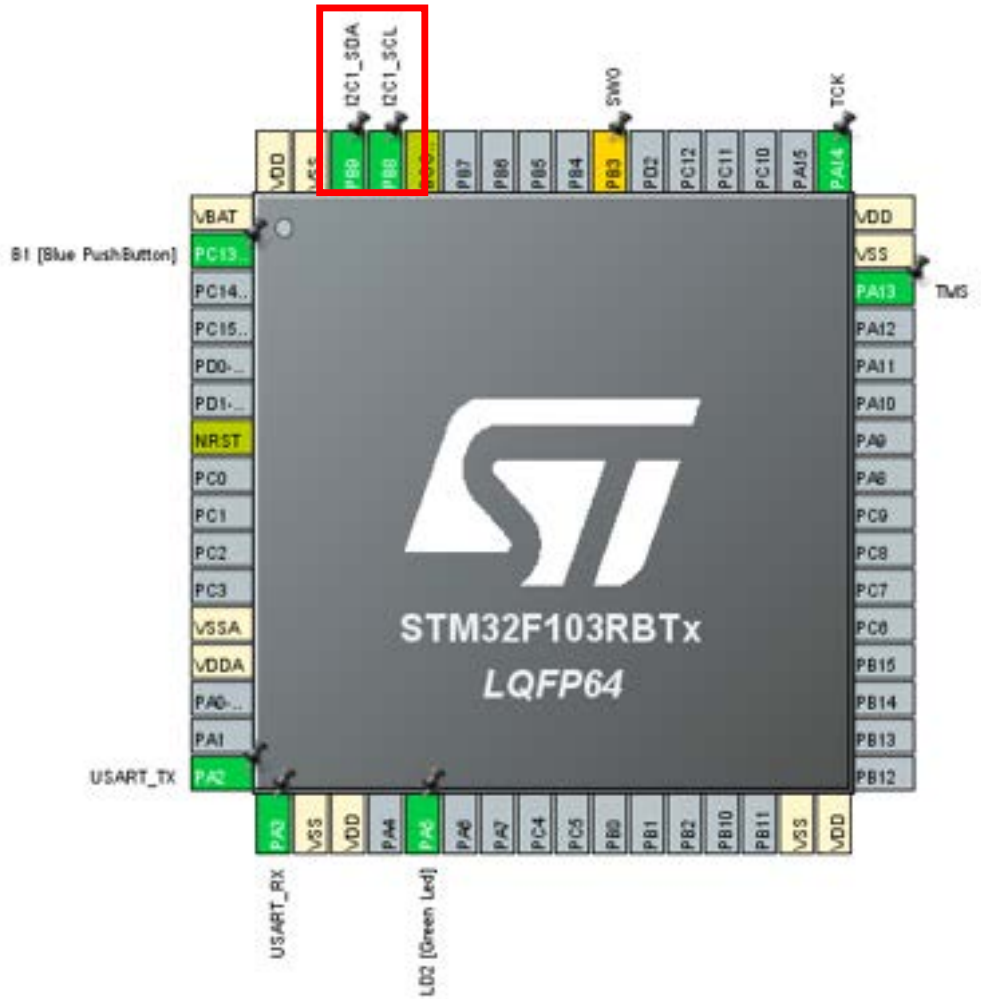
Figure 2. Clock tree



1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

KYUNG HEE UNIVERSITY

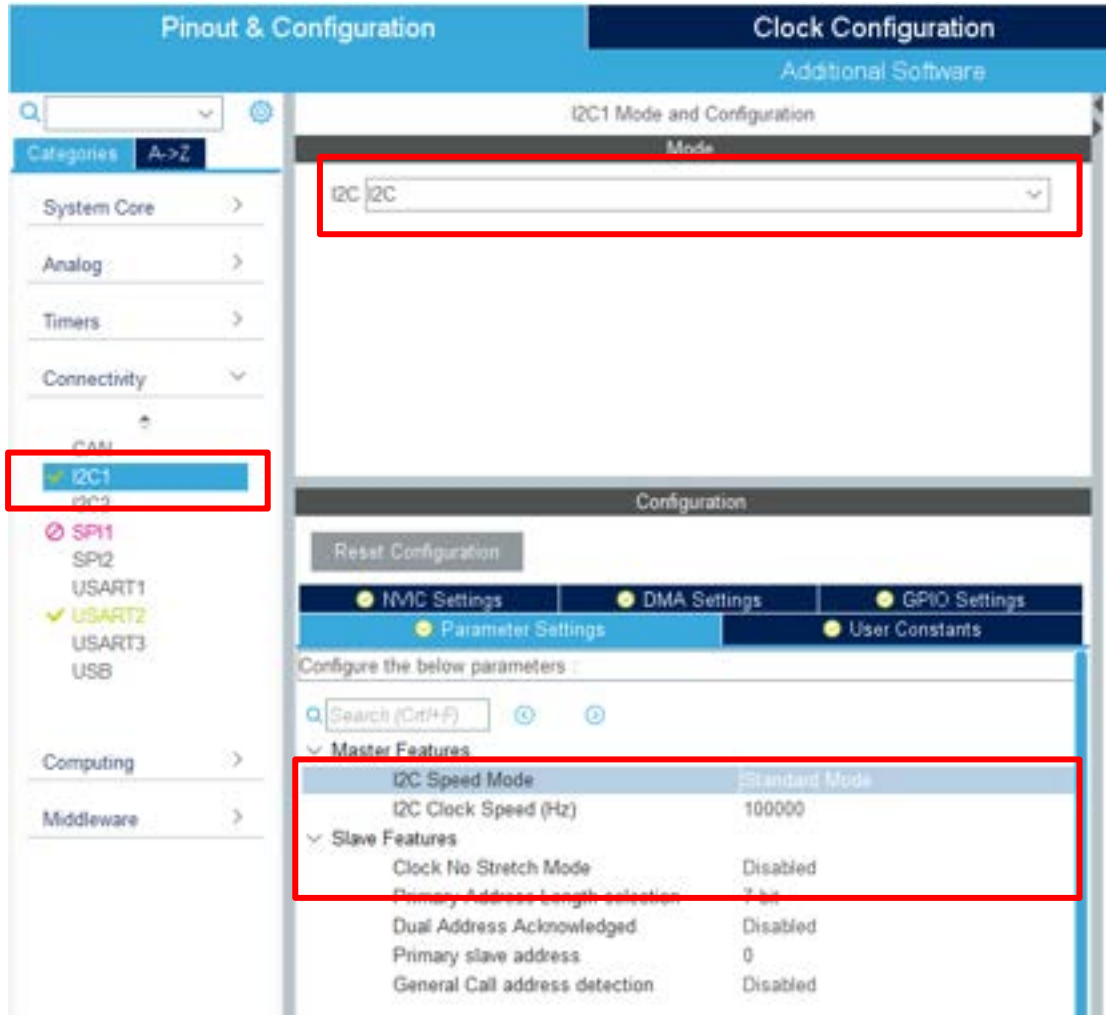
12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



Pinout & Configuration 탭에서

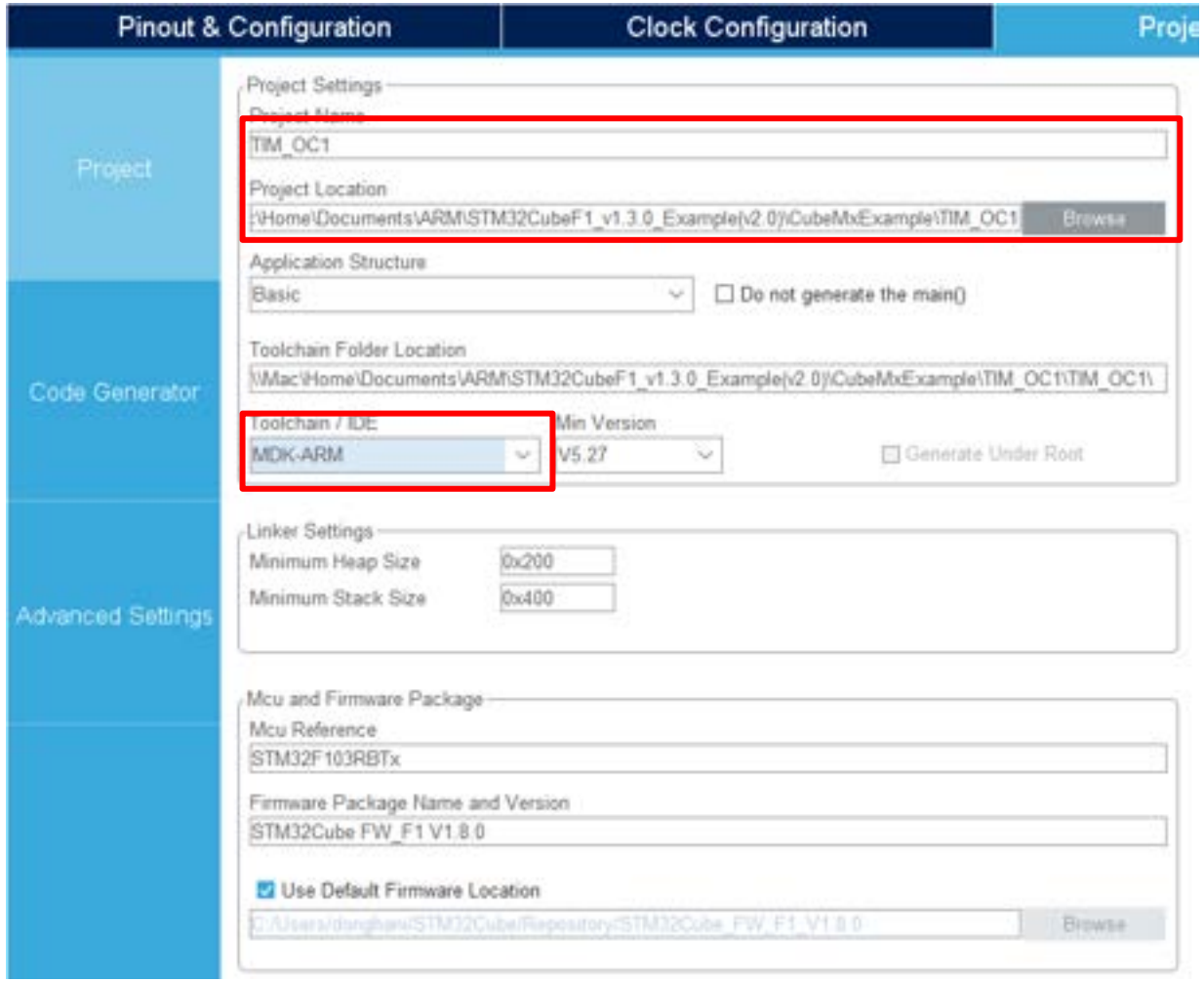
I2C1_SDA, I2C1_SCL을 PB9, PB8에 설정

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



Connectivity에서 I2C1을 선택하고, 나머지는 그대로 유지

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍



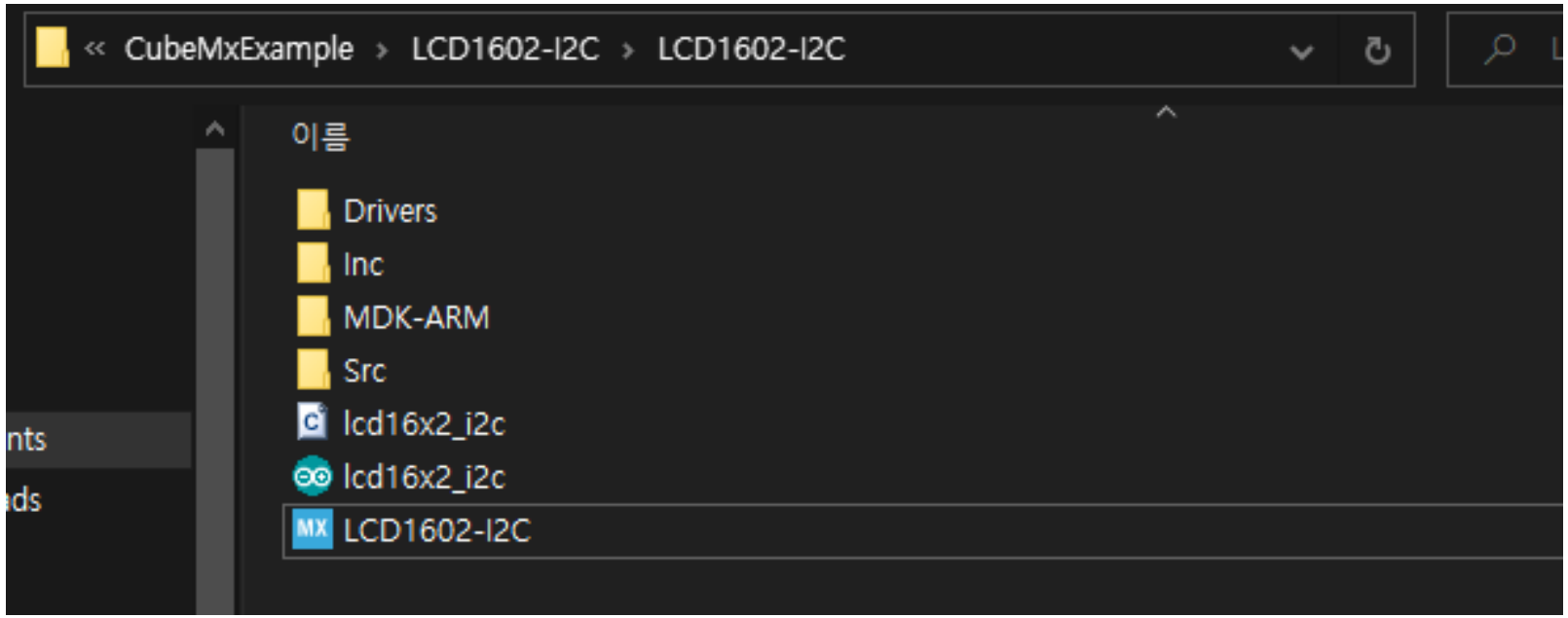
Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

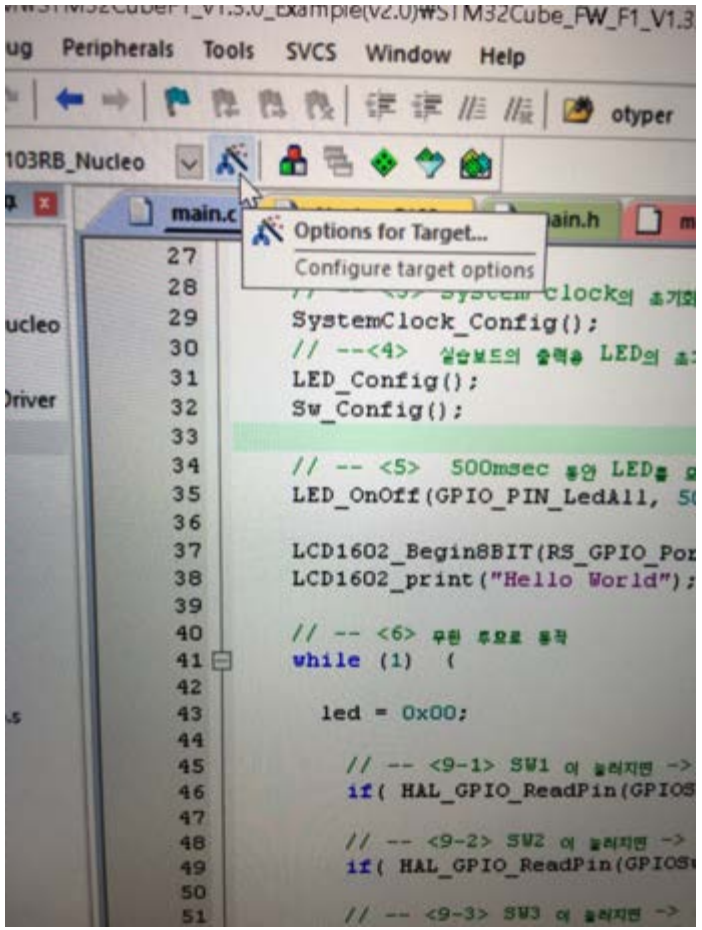
12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

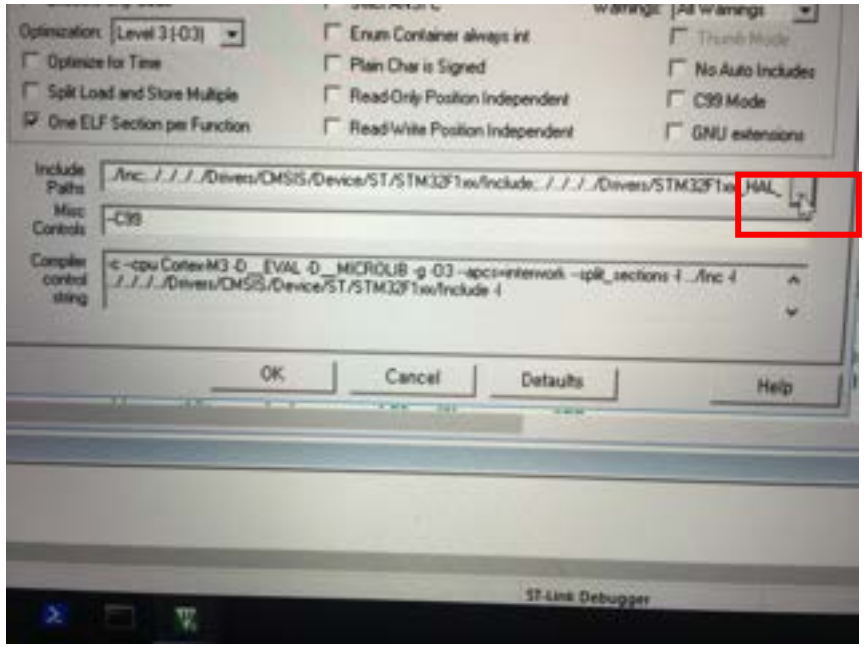
- lcd16x2_i2c.c와 lcd16x2_i2c.h를 KLAS에서/인터넷에서 다운로드 후 프로젝트 project 폴더에 옮기기

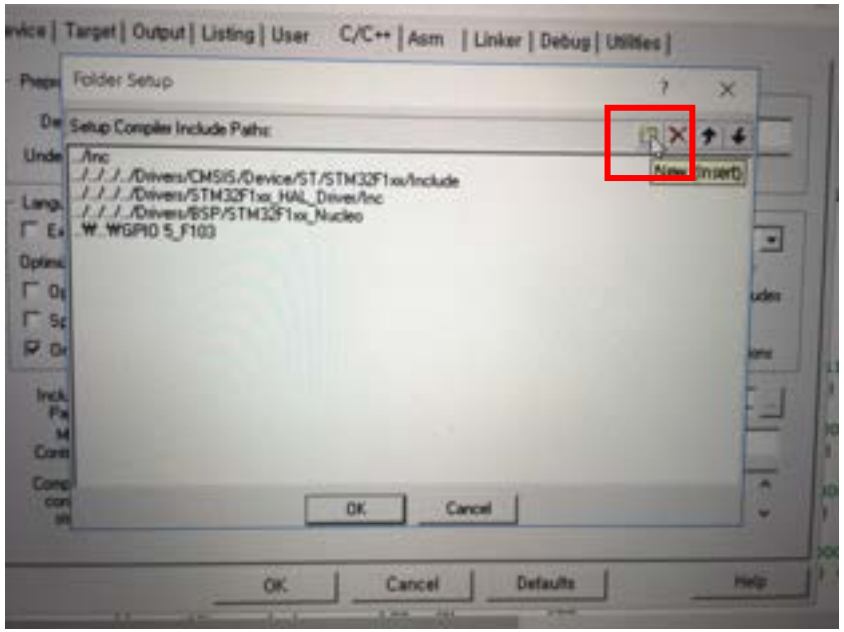


12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

- Keil uVision 프로그램에서 상단의 Options for Target 아이콘을 클릭하고, 열린 창에서 C/C++ 탭을 클릭한다. 오른쪽 하단에 있는 Include Paths의 오른쪽 끝의 버튼을 클릭한다. Folder Setup 창이 열리면 오른쪽 상단의 New 버튼을 누르고, lcd16x2_i2c.c와 lcd16x2_i2c.h를 옮겨논 폴더를 path에 추가한다.







12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

```
21  /* Includes ------------------------
22  #include "main.h"
23
24  /* Private includes ----------------
25  /* USER CODE BEGIN Includes */
26  #include "lcd16x2_i2c.h"
27  /* USER CODE END Includes */
```

→ main() 함수 앞 쪽에 #include “lcd16x2_i2c.h"추가

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

```
92    /* Initialize all configured peripherals */
93    MX_GPIO_Init();
94    MX_I2C1_Init();
95    MX_USART2_UART_Init();
96    /* USER CODE BEGIN 2 */
97    if(lcd16x2_i2c_init(&hi2c1))
98    {
99      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
100   }
101
102   lcd16x2_i2c_printf("Hello World!");
103
104   /* USER CODE END 2 */
105
106   /* Infinite loop */
107   /* USER CODE BEGIN WHILE */
108   while (1)
109   {
110     /* USER CODE END WHILE */
111
112     /* USER CODE BEGIN 3 */
113   }
```

12.5 I²C제어 : LCD 16X2 연결 - 프로그래밍

- STM32 HAL with CubeMX: Tutorial 42 - I2C LCD16x2 driver영상:
https://www.youtube.com/watch?v=e-KgHsQPkwg