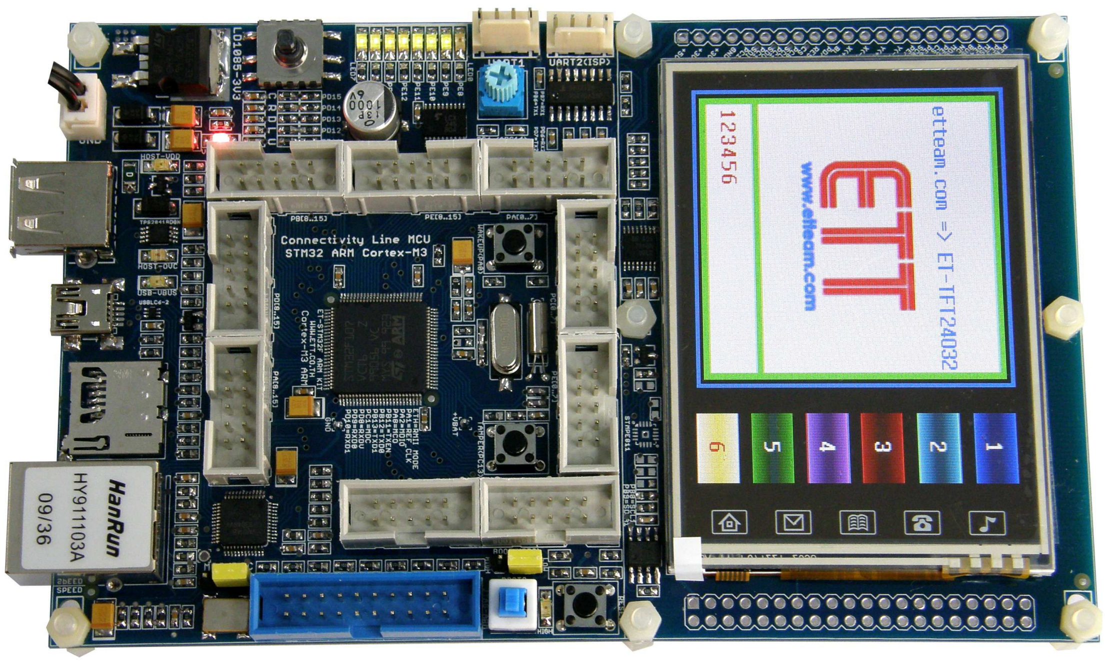
ET-STM32F ARM KIT(STM32F107VCT6)



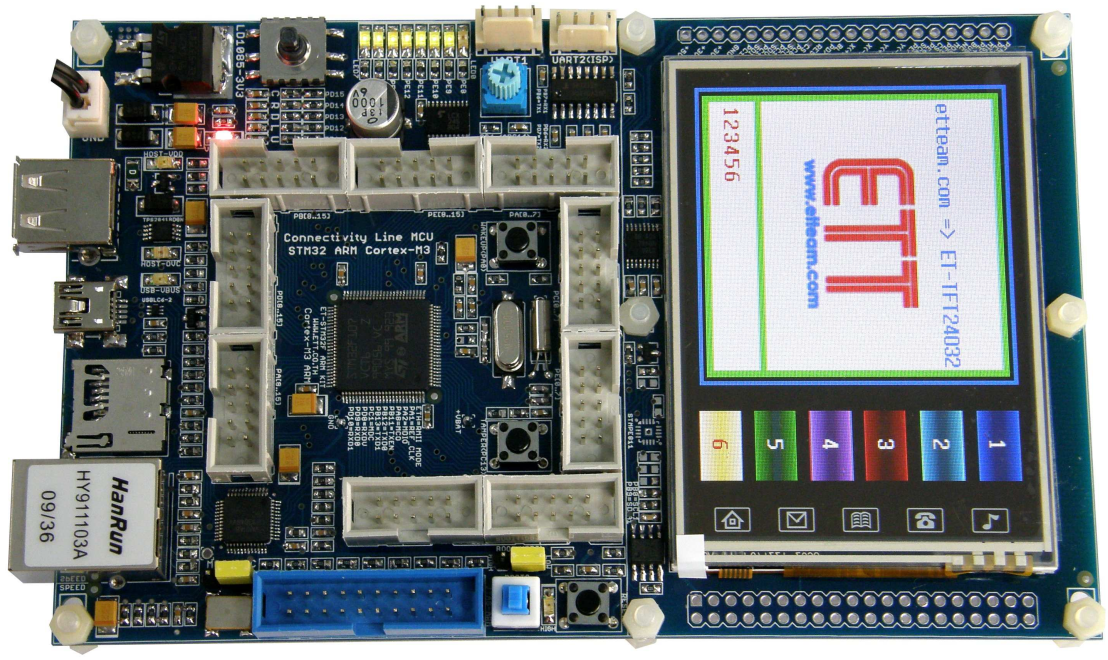
"ET-STM32F ARM KIT" is Board Microcontroller in the family of ARM Cortex M3 Core that uses 32-Bit 100Pin (100-LQFP) Microcontroller No.STM32F107VCT6 from ST Company (ST Microelectronics). ST Company has continuously improved and developed this MCU family from STM32F103; so, its ability and efficiency are better. It adds more resource systems to connect with Ethernet LAN and USB; so, it is able to operate as function Host/OTG. For other resources or their abilities remain the same such as SPI, I2C, CAN, ADC, DAC, Timer/Counter, PWM, Capture, UART, …, and etc.

ETT Team has designed the Hardware System of Board that supports group of customers who require learning, studying, demonstrating and group of customers who require modifying or applying this board to the real projects. The board structure consists of the basic devices that are necessary to initially learn and study such as LED to display operating status of Output Logic, Push Button Switch and Joy Switch to test Logic Input, Volume to adjust voltage to test A/D. Moreover, it provides high-level devices for user to support various operations such as Port USB Device/Host/OTG, SD Card, Port Ethernet LAN, Graphic LCD, and RS232. Furthermore, there are available GPIOs; so, user can design these GPIOs to use with other devices suitably by self.

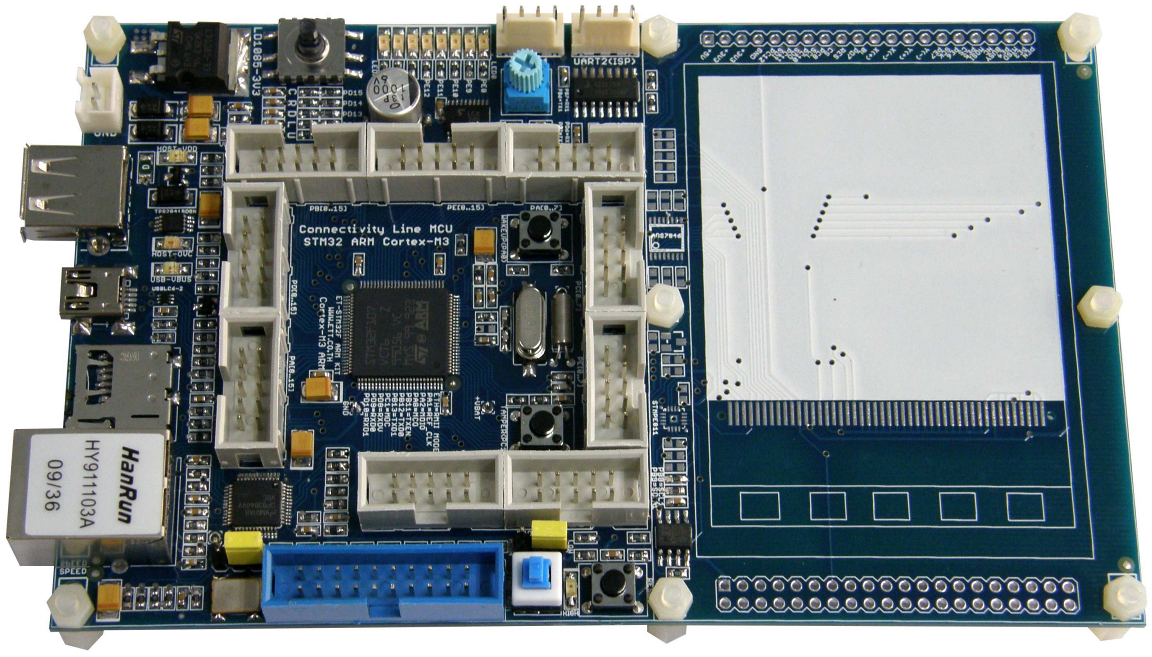
Specifications of Board "ET-STM32F ARM KIT (STM32F107VCT6)"

1. Use 32-Bit MCU in the family of ARM Cortex M3 No.STM32F107VCT6 from ST
2. Has 256KB Flash Memory and 64KB RAM internal MCU
3. Use Crystal 25.00MHz; MCU can process data by the maximum high speed of 72MHz when using with Phase-Locked Loop(PLL) inside MCU
4. Has RTC(Real Time Clock) with XTAL 32.768KHz and Battery Backup
5. Support In-System Programming(ISP) through USART2 Boot-Loader(RS232)
6. Has circuit to connect with standard 20Pin JTAG ARM for Real Time Debug
7. Use +5VDC Power Supply
8. Has Circuit USB 2.0 Full Speed to support Device/Host/OTG operation inside itself
9. Has Circuit Over Current Protection for USB Host/OTG
10. Has Circuit to connect Ethernet LAN 10/100Mb by using standard Connector RJ45 1-Channel
11. Has Circuit to connect SD Card Memory (Micro SD) for SPI Interface 1-Channel
12. Has Circuit RS232 Communication by using Connector 4-PIN standard ETT 2-Channel
13. Has Circuit to connect TFT LCD Color 320x240Pixel (3.2") with Touch Sensor
14. Has 2 sets of Circuit Push Button Switch with Switch RESET
15. Has Circuit Joy Switch 5-Direction
16. Has 8 sets of Circuit LED to display the operating status of testing Output with Circuit Buffer
17. Has Circuit to generate 0-3V3 Voltage by using 1 set of Adjustable Resistor to test A/D

18. Has 80Bit GPIO; in this case, there are 72Bit GPIO that can be used independently by using Connector 10Pin IDE. Theses 72Bit GPIO is allocated to 9 sets and user can set their functions to be 72Bit GPIO or other functions such as A/D, D/A, I2C, CAN, Ethernet and etc.
    a. 10Bit for Ethernet LAN(DP83848V RMII Interface Mode)
    b. 2Bit for USART1 and 2Bit for USART2
    c. 1Bit for Volume to adjust Voltage to test ADC14(PC4) and 8Bit for LED
    d. 4 Bit for SD Card Interface
    e. 6Bit for USB Device/Host/OTG Interface
    f. 10Bit for TFTLCD320x240 and Touch Sensor(ADS7846)
    g. 5Bit for Joy Switch 5 Direction and 2Bit for Push Button SW
    h. 5Bit for JTAG ARM Interface
    i. 2Bit for I2C Interface

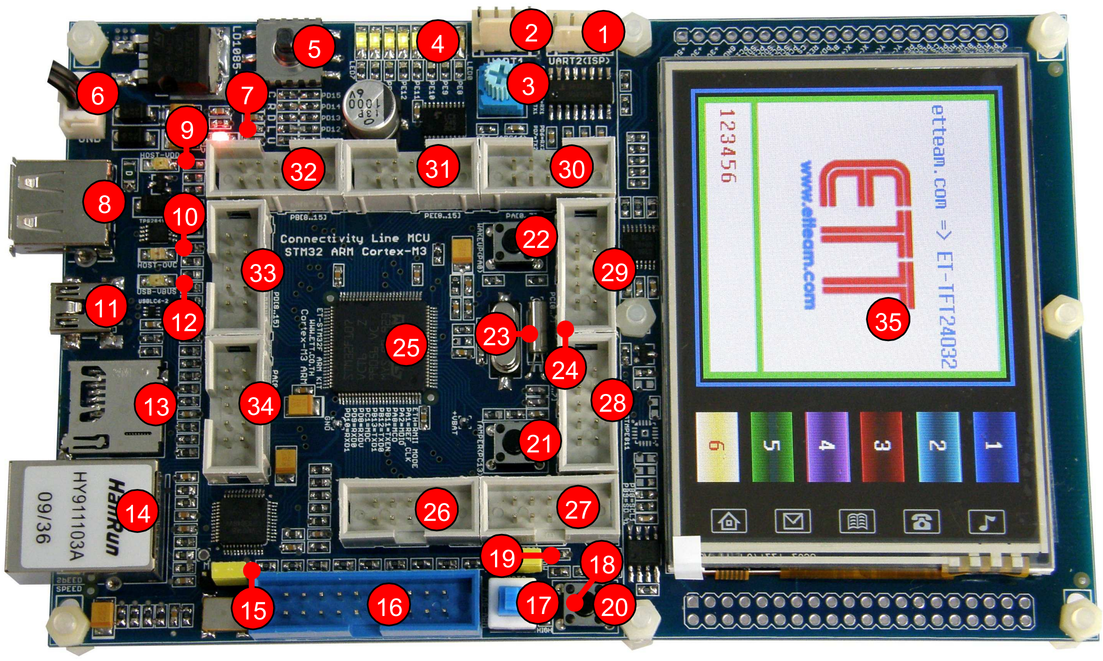


Picture shows the structure of Board ET-STM32F ARM KIT & TFT LCD.



Picture shows the structure of Board ET-STM32F ARM KIT.

Structure of Board ET-STM32F ARM KIT



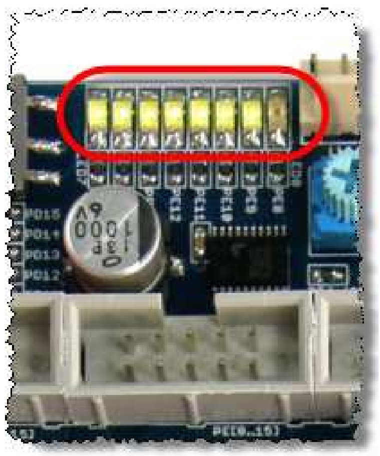
Picture shows position of devices on Board ET-STM32F ARM KIT.

- **No.1:** It is Connector UART2(RS232) to use and Download Hex File into CPU.
- **No.2:** It is Connector UART1(RS232) to use.
- **No.3:** It is VR to adjust 0+3V3 Voltage for testing A/D(PC4/ADC14).
- **No.4:** It is LED[0..7] to test Logic Output of PE[8..15].
- **No.5:** It is Joy Switch 5-Direction.
- **No.6:** It is Connector +5VDC Power Supply for board.
- **No.7:** It is LED to display the operating status of Power +VDD(+3V3).
- **No.8:** It is Connector USB Host.
- **No.9:** It is LED to display the operating status of Host VDD.
- **No.10:** It is LED to display the operating status of Host Over Current.

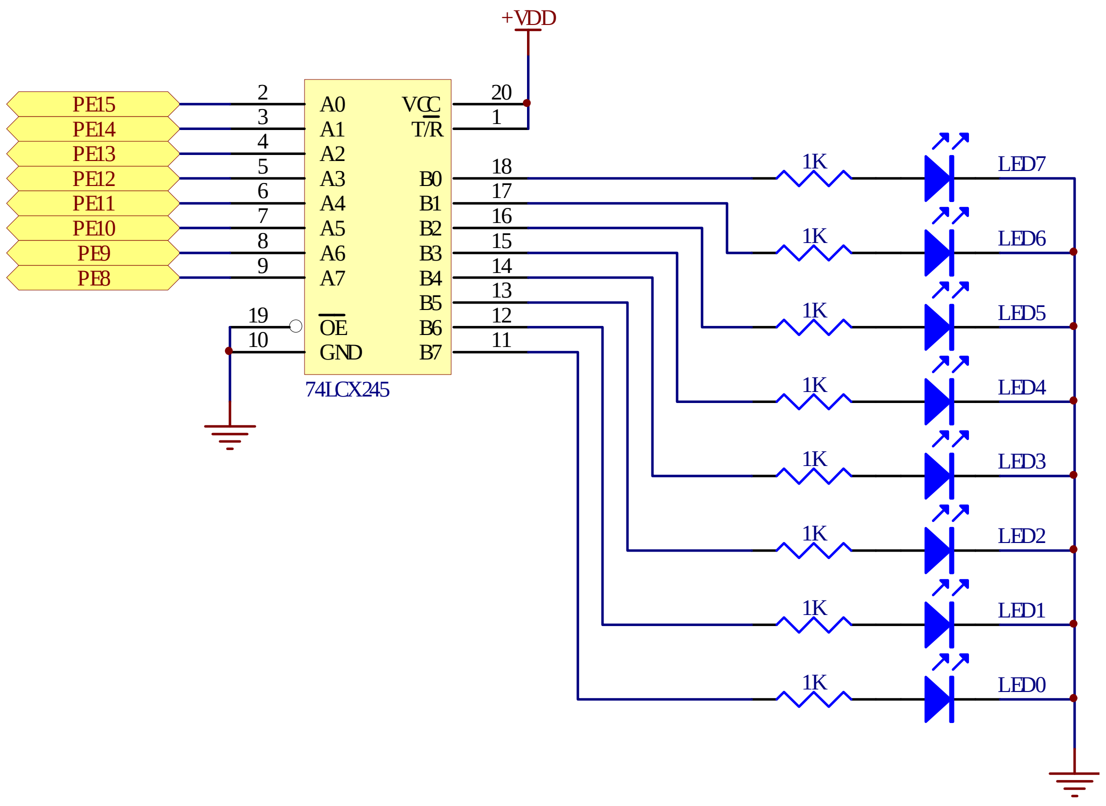
- **No.11:** It is Connector USB Device/OTG.
- **No.12:** It is LED to display the operating status of USB VBUS.
- **No.13:** It is socket to insert memory card; in this case, it is compatible with Micro SD Card.
- **No.14:** It is Connector RJ45 Ethernet LAN.
- **No.15:** It is Jumper(MCO/OSC) to choose the Signal Clock Generator to supply it to DP83848V.
- **No.16:** It is Connector JTAG ARM for Real Time Debug.
- **No.17:** It is SW BOOT0 that is used with Jumper BOOT1 to choose operation mode of MCU between Boot Loader(BOOT0=1,BOOT1=0) and Run(BOOT0=0,BOOT1=0).
- **No.18:** It is LED to display Logic Status of BOOT0 = 1(ON=Boot Loader, OFF=Run).
- **No.19:** It is Jumper BOOT1(PB2); normally, it is always set to be LOW.
- **No.20:** It is SW RESET.
- **No.21:** It is SW Tamper(PC13).
- **No.22:** It is SW Wakeup(PA0).
- **No.23:** It is Crystal 25MHz to be the Time Base System for MCU.
- **No.24:** It is Crystal 32.768KHz to be the Time Base for RTC internal MCU.
- **No.25:** It is MCU No.STM32F107VCT6(100Pin LQFP).
- **No.26:** It is Connector GPIO PD[0..7].
- **No.27:** It is Connector GPIO PB[0..7].
- **No.28:** It is Connector GPIO PE[0..7].
- **No.29:** It is Connector GPIO PC[0..7].
- **No.30:** It is Connector GPIO PA[0..7].
- **No.31:** It is Connector GPIO PE[8..15].
- **No.32:** It is Connector GPIO PB[8..15].
- **No.33:** It is Connector GPIO PD[8..15].
- **No.34:** It is Connector GPIO PA[8..15].
- **No.35:** It is 320x240 Dot TFT LCD with Touch Screen Sensor.

Circuit LED Display

LED Display of board is Source Current Interface by using with +3.3V Power Supply. It runs by Logic "1" (+3V3) and then stops running by Logic "0" (0V). In this case, the operation is controlled 8 sets of GPIO; PE[8..15]. This circuit is used to test the operation of Output.



↓



If user requires using this function, it needs to set PE[8..15] to be GPIO Output Port first, and then control Logic for PE[8..15] as required as in the example below;

```
// ET-STM32F ARM KIT Hardware Board : LED[0..7] = PE[8..15]
#define LEDn                            8

#define LED0_GPIO_PORT                  GPIOE
#define LED0_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED0_GPIO_PIN                   GPIO_Pin_8

#define LED1_GPIO_PORT                  GPIOE
#define LED1_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED1_GPIO_PIN                   GPIO_Pin_9

#define LED2_GPIO_PORT                  GPIOE
#define LED2_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED2_GPIO_PIN                   GPIO_Pin_10

#define LED3_GPIO_PORT                  GPIOE
#define LED3_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED3_GPIO_PIN                   GPIO_Pin_11

#define LED4_GPIO_PORT                  GPIOE
#define LED4_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED4_GPIO_PIN                   GPIO_Pin_12

#define LED5_GPIO_PORT                  GPIOE
#define LED5_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED5_GPIO_PIN                   GPIO_Pin_13

#define LED6_GPIO_PORT                  GPIOE
#define LED6_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED6_GPIO_PIN                   GPIO_Pin_14

#define LED7_GPIO_PORT                  GPIOE
#define LED7_GPIO_CLK                   RCC_APB2Periph_GPIOE
#define LED7_GPIO_PIN                   GPIO_Pin_15

typedef enum
{
  LED0 = 0,
  LED1 = 1,
  LED2 = 2,
  LED3 = 3,
  LED4 = 4,
  LED5 = 5,
  LED6 = 6,
  LED7 = 7
} Led_TypeDef;
GPIO_TypeDef* GPIO_PORT[LEDn] = {LED0_GPIO_PORT,
                                 LED1_GPIO_PORT,
                                 LED2_GPIO_PORT,
                                 LED3_GPIO_PORT,
                                 LED4_GPIO_PORT,
                                 LED5_GPIO_PORT,
                                 LED6_GPIO_PORT,
                                 LED7_GPIO_PORT};

const uint16_t GPIO_PIN[LEDn] = {LED0_GPIO_PIN,
                                 LED1_GPIO_PIN,
                                 LED2_GPIO_PIN,
                                 LED3_GPIO_PIN,
                                 LED4_GPIO_PIN,
                                 LED5_GPIO_PIN,
                                 LED6_GPIO_PIN,
                                 LED7_GPIO_PIN};

const uint32_t GPIO_CLK[LEDn] = {LED0_GPIO_CLK,
                                 LED1_GPIO_CLK,
                                 LED2_GPIO_CLK,
                                 LED3_GPIO_CLK,
                                 LED4_GPIO_CLK,
                                 LED5_GPIO_CLK,
                                 LED6_GPIO_CLK,
                                 LED7_GPIO_CLK};


GPIO_InitTypeDef  GPIO_InitStructure;
.
.
.
/* Enable the GPIO_LED Clock */
RCC_APB2PeriphClockCmd(GPIO_CLK[LED0], ENABLE);

/* Configure the GPIO_LED pin */
GPIO_InitStructure.GPIO_Pin = GPIO_PIN[LED0];
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIO_PORT[LED0], &GPIO_InitStructure);
.
.
```

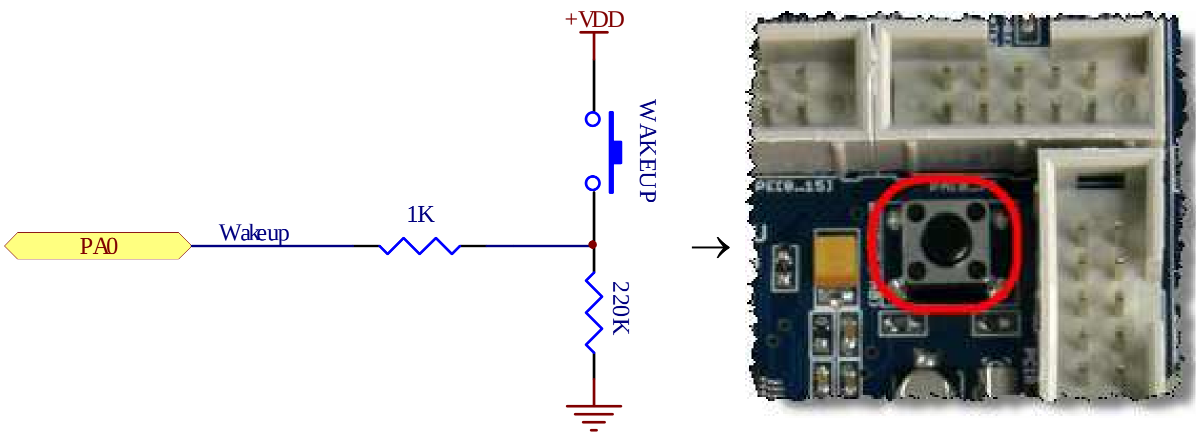
```
GPIO_PORT[LED0]->BSRR = GPIO_PIN[LED0]; //ON LED0
GPIO_PORT[LED0]->BRR = GPIO_PIN[LED0];  //OFF LED0
GPIO_PORT[LED0]->ODR ^= GPIO_PIN[LED0]; //Toggle LED0
```

Example of setting PE[8..15] to be Output LED

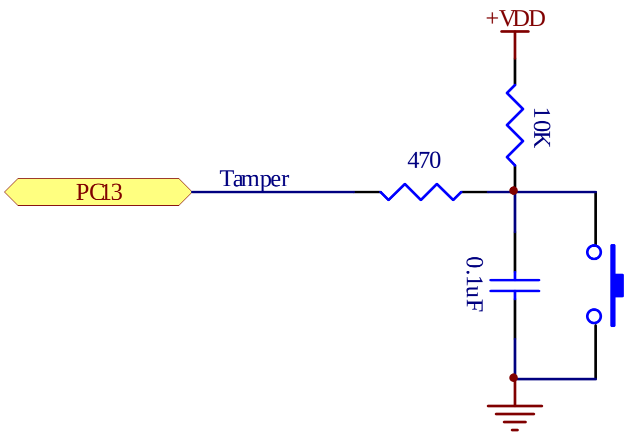
Push Button Switch

This circuit uses Push Button Switch; in this case, there are 2 sets of Switch; Switch Wakeup and Switch Tamper. The operating results of both Circuit Push Buttons are different because they give the opposite Logic.

If Switch Wakeup(PA0) is pressed, it makes the Logic status at Pin PA0 be HIGH; on the other hand, if it is released, it makes the Logic status at Pin PA0 be LOW instead.



If Switch Tamper(PC13) is pressed, it makes the Logic status at Pin PC13 be LOW; on the other hand, if it is released, it makes the Logic status at Pin PC13 be HIGH instead.



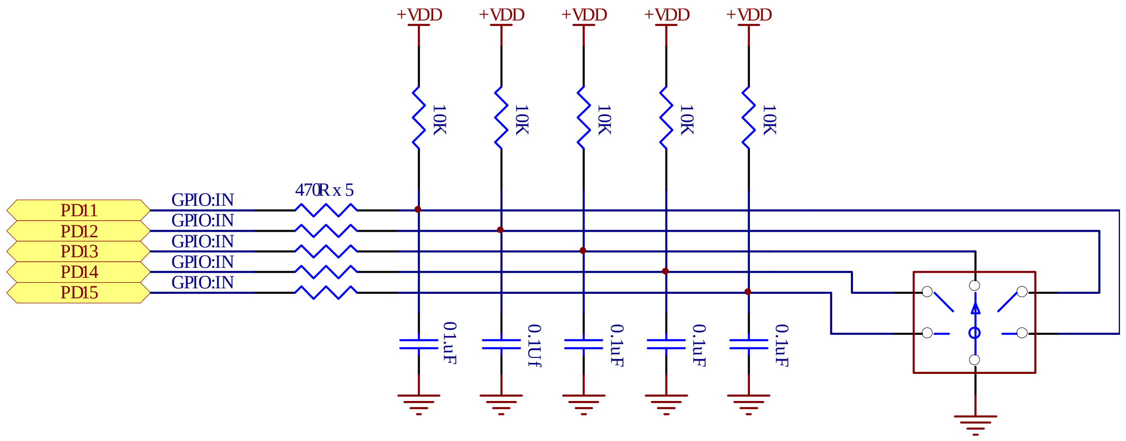
```
// Switch Wakeup(PA0)
#define WAKEUP_BUTTON_PORT          GPIOA
#define WAKEUP_BUTTON_CLK           RCC_APB2Periph_GPIOA
#define WAKEUP_BUTTON_PORT_SOURCE   GPIO_PortSourceGPIOA
#define WAKEUP_BUTTON_PIN           GPIO_Pin_0
#define WAKEUP_BUTTON_PIN_SOURCE    GPIO_PinSource0
#define WAKEUP_BUTTON_EXTI_LINE     EXTI_Line0
#define WAKEUP_BUTTON_IRQn          EXTI0_IRQn

// Switch Tamper(PC13)
#define TAMPER_BUTTON_PORT          GPIOC
#define TAMPER_BUTTON_CLK           RCC_APB2Periph_GPIOC
#define TAMPER_BUTTON_PORT_SOURCE   GPIO_PortSourceGPIOC
#define TAMPER_BUTTON_PIN           GPIO_Pin_13
#define TAMPER_BUTTON_PIN_SOURCE    GPIO_PinSource13
#define TAMPER_BUTTON_EXTI_LINE     EXTI_Line13
#define TAMPER_BUTTON_IRQn          EXTI15_10_IRQn

#define BUTTON_MODE                 Mode_GPIO
#define BUTTONn                     7

typedef enum
{
  Button_WAKEUP = 0,
  Button_TAMPER = 1,
  Button_UP = 2,
  Button_LEFT = 3,
  Button_DOWN = 4,
  Button_RIGHT = 5,
  Button_SELECT = 6
} Button_TypeDef;
.
.
.
ET_STM32_PB_Init(Button_WAKEUP, BUTTON_MODE);
ET_STM32_PB_Init(Button_TAMPER, BUTTON_MODE);
.
.
.
//Wakeup(Toggle Logic:Press=1,Release=0)
if (ET_STM32_PB_GetState(Button_WAKEUP) == 1)
{ .. }    //Press
else
{ .. }    //Release

//Tamper(Press=0,Release=1)
if (ET_STM32_PB_GetState(Button_TAMPER) == 0)
{ .. }    //Press
else
{ .. }    //Release
```

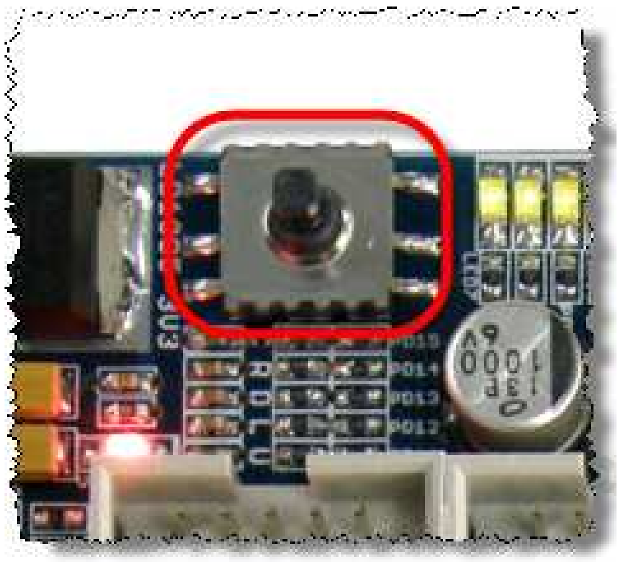
Example of setting PA0 and PC13 to be Input Switch

Circuit Joy Switch

This circuit uses Joy Switch 5-Direction and its structure is Push Button with Circuit Pull-Up that is compatible with +3.3V Power Supply. If this switch is not pressed, it makes Logic Status be "1"; on the other hand, if it is pressed, it makes Logic Status be "0" instead. This is used to test the operation of Input Logic and applied to various modifications by connecting through GPIO Input as follows;

- Up Position uses PD11 to be Function GPIO Input.
- Left Position uses PD12 to be Function GPIO Input.
- Down Position uses PD13 to be Function GPIO Input.
- Right Position uses PD14 to be Function GPIO Input.
- Center Position uses PD15 to be Function GPIO Input.





```
// ET-STM32F ARM KIT Hardware Board
// Switch = PD11(Joy-Up)
//        = PD12(Joy-Left)
//        = PD13(Joy-Down)
//        = PD14(Joy-Right)
//        = PD15(Joy-Select)

// Joy Up(PD11)
#define UP_BUTTON_PORT                 GPIOD
#define UP_BUTTON_CLK                  RCC_APB2Periph_GPIOD
#define UP_BUTTON_PORT_SOURCE          GPIO_PortSourceGPIOD
#define UP_BUTTON_PIN                  GPIO_Pin_11

#define UP_BUTTON_PIN_SOURCE           GPIO_PinSource11
#define UP_BUTTON_EXTI_LINE            EXTI_Line11
#define UP_BUTTON_IRQn                 EXTI15_10_IRQn

// Joy Left(PD12)
```

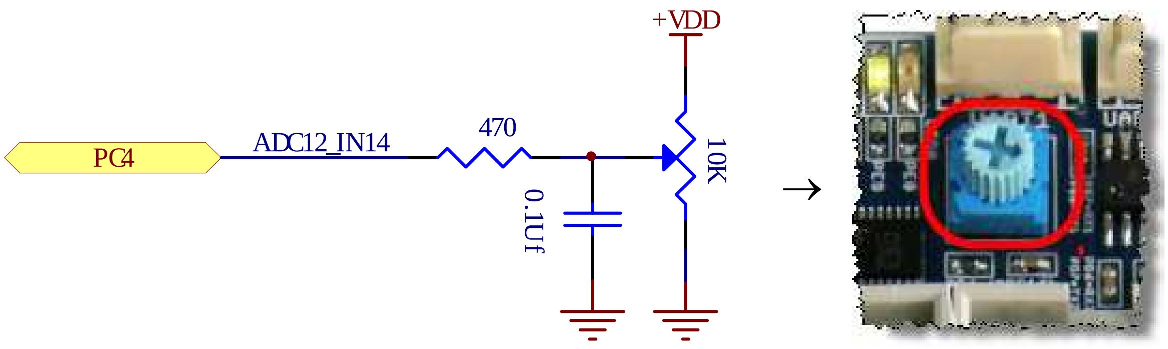
```
#define LEFT_BUTTON_PORT              GPIOD
#define LEFT_BUTTON_CLK               RCC_APB2Periph_GPIOD
#define LEFT_BUTTON_PORT_SOURCE       GPIO_PortSourceGPIOD
#define LEFT_BUTTON_PIN               GPIO_Pin_12
#define LEFT_BUTTON_PIN_SOURCE        GPIO_PinSource12
#define LEFT_BUTTON_EXTI_LINE         EXTI_Line12
#define LEFT_BUTTON_IRQn              EXTI15_10_IRQn

// Joy Down(PD13)
#define DOWN_BUTTON_PORT              GPIOD
#define DOWN_BUTTON_CLK               RCC_APB2Periph_GPIOD
#define DOWN_BUTTON_PORT_SOURCE       GPIO_PortSourceGPIOD
#define DOWN_BUTTON_PIN               GPIO_Pin_13
#define DOWN_BUTTON_PIN_SOURCE        GPIO_PinSource13
#define DOWN_BUTTON_EXTI_LINE         EXTI_Line13
#define DOWN_BUTTON_IRQn              EXTI15_10_IRQn

// Joy Right(PD14)
#define RIGHT_BUTTON_PORT             GPIOD
#define RIGHT_BUTTON_CLK              RCC_APB2Periph_GPIOD
#define RIGHT_BUTTON_PORT_SOURCE      GPIO_PortSourceGPIOD
#define RIGHT_BUTTON_PIN              GPIO_Pin_14

#define RIGHT_BUTTON_PIN_SOURCE       GPIO_PinSource14
#define RIGHT_BUTTON_EXTI_LINE        EXTI_Line14
#define RIGHT_BUTTON_IRQn             EXTI15_10_IRQn

// Joy Select(PD15)
#define SELECT_BUTTON_PORT            GPIOD
#define SELECT_BUTTON_CLK             RCC_APB2Periph_GPIOD
#define SELECT_BUTTON_PORT_SOURCE     GPIO_PortSourceGPIOD
#define SELECT_BUTTON_PIN             GPIO_Pin_15

#define SELECT_BUTTON_PIN_SOURCE      GPIO_PinSource15
#define SELECT_BUTTON_EXTI_LINE       EXTI_Line15
#define SELECT_BUTTON_IRQn            EXTI15_10_IRQn
#define BUTTON_MODE                   Mode_GPIO
#define BUTTONn                       7

typedef enum
{
  Button_WAKEUP = 0,
  Button_TAMPER = 1,
  Button_UP = 2,
  Button_LEFT = 3,
  Button_DOWN = 4,
  Button_RIGHT = 5,
  Button_SELECT = 6
} Button_TypeDef;
.
```

```
.
.
ET_STM32_PB_Init(Button_UP, BUTTON_MODE);
ET_STM32_PB_Init(Button_LEFT, BUTTON_MODE);
ET_STM32_PB_Init(Button_DOWN, BUTTON_MODE);
ET_STM32_PB_Init(Button_RIGHT, BUTTON_MODE);
ET_STM32_PB_Init(Button_SELECT, BUTTON_MODE);
.
.
.
//Up(Press=0,Release=1)
if (ET_STM32_PB_GetState(Button_UP) == 0)
{
  ..  //Press
}
else
{
  ..  //Release
}

//Down(Press=0,Release=1)
if (ET_STM32_PB_GetState(Button_DOWN) == 0)
{
  ..  //Press
}
else
{
  ..  //Release
}
.
.
.
if (ET_STM32_PB_GetState(Button_SELECT) == 0)
{
  ..  //Press
}
else
{
  ..  //Release
}
```

Example of setting values for Input Joy Switch

Circuit Adjustable VR (0-3V3)

This circuit uses Adjustable VR Resistor that is compatible with +3.3V Power Supply. It gives the Output Voltage in the range of 0V to +3.3V according to adjusting Resistor. This Output is supplied to Pin PC4 to generate Input Voltage to test the operation of Circuit A/D(PC4).



```
void ET_STM32_ADC_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  ADC_InitTypeDef ADC_InitStructure;

  /* Enable ADC1 clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  /* Configure PC.04 (ADC Channel14) as analog input */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  /* ADC1 Configuration */
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
  ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);

  /* ADC1 regular channel14 configuration */
  ADC_RegularChannelConfig(ADC1,ADC_Channel_14,1,
                           ADC_SampleTime_13Cycles5);

  ADC_Cmd(ADC1, ENABLE); // Enable ADC1
  ADC_SoftwareStartConvCmd(ADC1, ENABLE); //Start ADC1 Conversion
}
```

Example of setting PC4 to be Analog Input ADC14

```
//Bargraph LED Display
const unsigned long
led_graph[8]={0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
.
.
int ADCVal = 0;
.
.
.
ET_STM32_LED_Init(LED0);
ET_STM32_LED_Init(LED1);
ET_STM32_LED_Init(LED2);
ET_STM32_LED_Init(LED3);
ET_STM32_LED_Init(LED4);
ET_STM32_LED_Init(LED5);
ET_STM32_LED_Init(LED6);
ET_STM32_LED_Init(LED7);

//Initial ADC(ADC14:PC4)
ET_STM32_ADC_Configuration();
.
.
.
while (1)
{
  ADCVal = ADC_GetConversionValue(ADC1);     //Read ADC
  ET_STM32_LED_Write(led_graph[ADCVal/512]); //Display ADC to Bargraph
LED
}
.
.
.
void ET_STM32_LED_Init(Led_TypeDef Led)
{
  GPIO_InitTypeDef  GPIO_InitStructure;

  /* Enable the GPIO_LED Clock */
  RCC_APB2PeriphClockCmd(GPIO_CLK[Led], ENABLE);

  /* Configure the GPIO_LED pin */
  GPIO_InitStructure.GPIO_Pin = GPIO_PIN[Led];
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

  GPIO_Init(GPIO_PORT[Led], &GPIO_InitStructure);
}
.
.
.
void ET_STM32_LED_Write(uint8_t value)
{
 GPIO_WriteBit(GPIO_PORT[LED0],GPIO_PIN[LED0],(value&0x01) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED1],GPIO_PIN[LED1],(value&0x02) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED2],GPIO_PIN[LED2],(value&0x04) ? Bit_SET :
Bit_RESET);
```

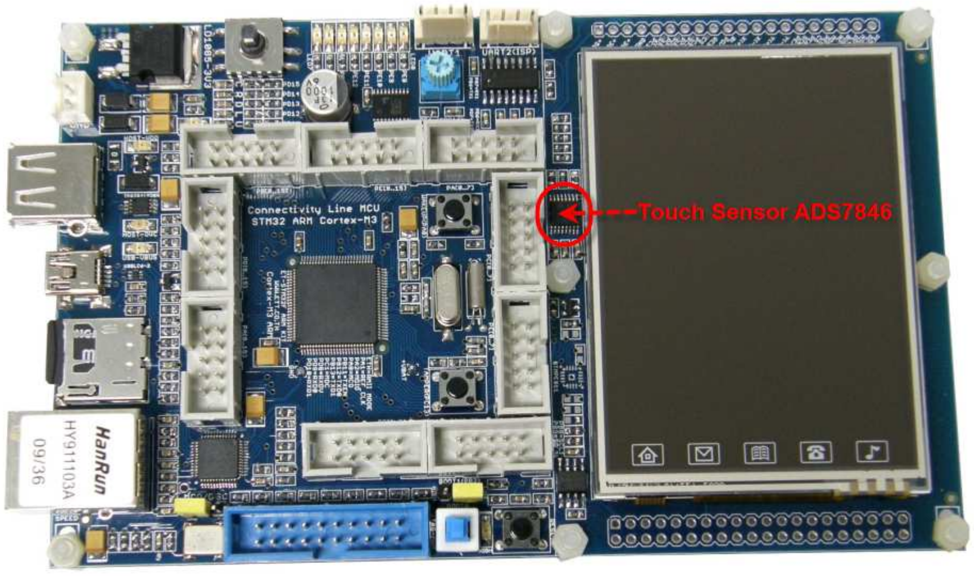
```
 GPIO_WriteBit(GPIO_PORT[LED3],GPIO_PIN[LED3],(value&0x08) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED4],GPIO_PIN[LED4],(value&0x10) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED5],GPIO_PIN[LED5],(value&0x20) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED6],GPIO_PIN[LED6],(value&0x40) ? Bit_SET :
Bit_RESET);
 GPIO_WriteBit(GPIO_PORT[LED7],GPIO_PIN[LED7],(value&0x80) ? Bit_SET :
Bit_RESET);
}
```

Example of reading ADC Input from ADC14

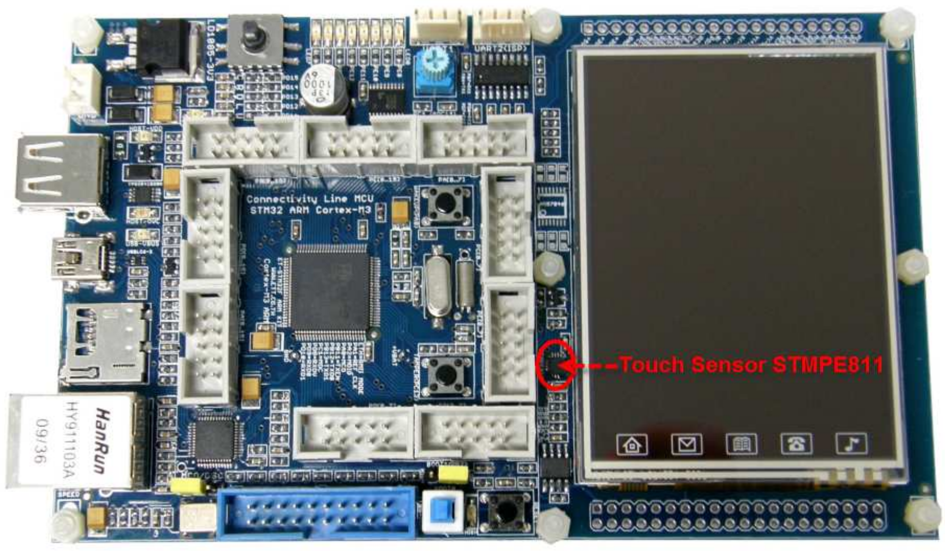
TFT Graphic LCD Display

There are 2 types to connect Board ET-STM32F ARM KIT with 3.2" TFT Graphic LCD as described below;

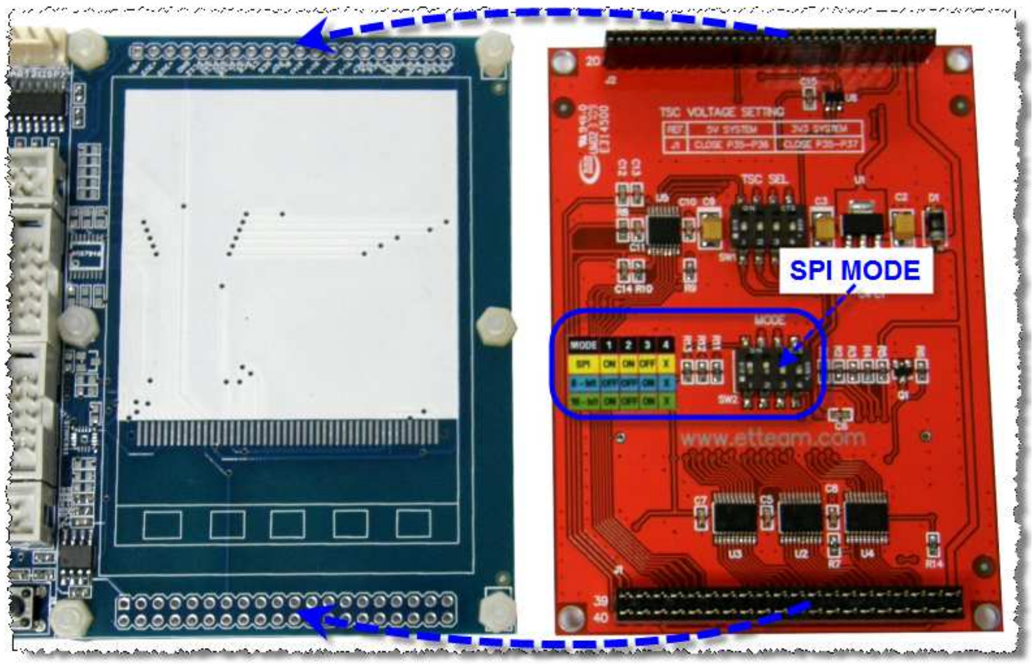
- Firstly, it directly installs Module LCD in board permanently by using TFT LCD version "KWH032GM02-F05" and it connects signal and TFT LCD in Mode SPI. This TFT LCD Version includes Sensor of Touch Screen. There are 2 devices that user can choose to read Sensor value of Touch Screen for Board ET-STM32F ARM KIT; Chip No.STMPE811 for I2C Interface or Chip ADS7846 for SPI Interface (in this case, it depends on the process of making and installing Chip in board).



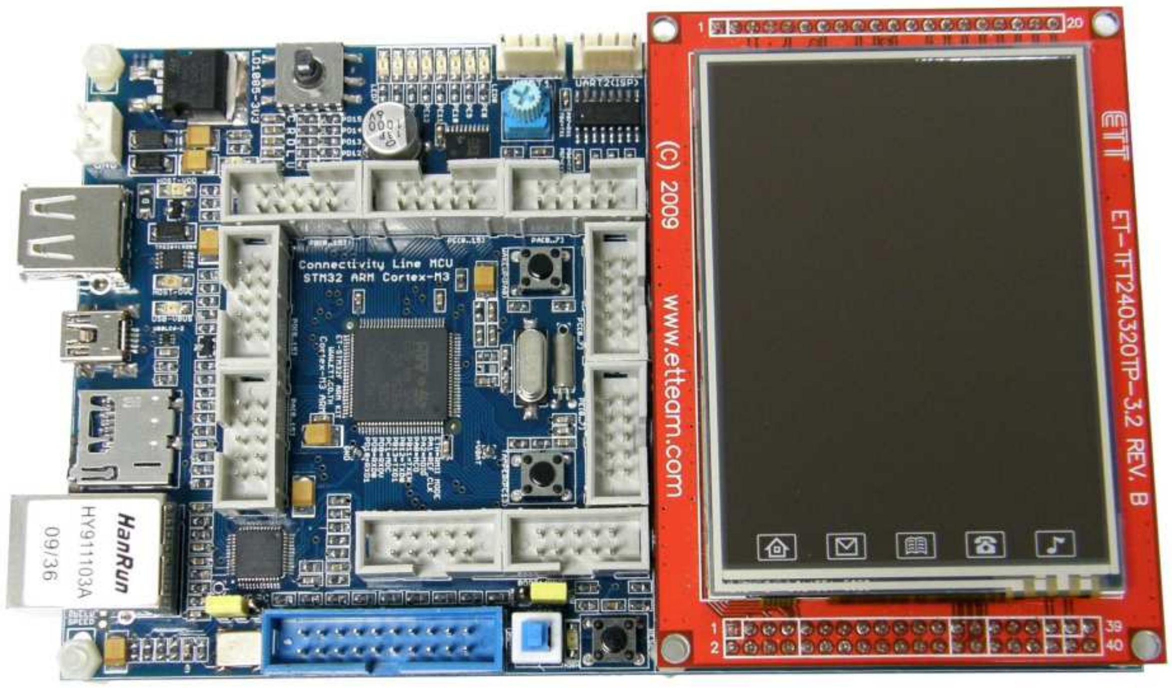
↑↓



- It installs ETT Board Display version “ET-TFT240320TP-3.2 REV.B” that has already been installed TFT LCD version “KWH032GM02-F05” with Chip ADS7846 to read Touch Sensor values internal board. It is installed through Connector; so, it is able to add or remove it from board easily. Moreover, it also needs to set the SPI Interface for this board.



↓



How to connect TFT LCD version "KWH032GM02-F05"

In the part of TFT LCD version "KWH032GM02-F05" is SPI Interface Mode by using SPI3 of MCU to connect. It uses signals as follows;

- CS GLCD uses PC8 to be Function GPIO Output.
- SCL GLCD uses PC10 to be Function SCK3 of SPI3.
- SDO GLCD uses PC11 to be Function MISO3 of SPI3.
- SDI GLCD uses PC12 to be Function MOSI3 of SPI3.
- BL GLCD uses PD7 to be Function GPIO Output.

How to connect with Touch Screen Sensor by using ADS7846

If in the part of Touch Screen uses Chip ADS7846, it is SPI Interface; in this case, it uses Pin GPIO to generate signal that imitates SPI instead. It uses signals as follows;

- DCLK ADS7846 uses PE7 to be Function GPIO Output (SPI:SCK).
- CS ADS7846 uses PE6 to be Function GPIO Output (CS#).

- DOUT ADS7846 uses PE4 to be Function GPIO Input (SPI:MISO).
- DIN ADS7846 uses PE5 to be Function GPIO Output(SPI:MOSI).
- PENIRQ ADS7846 uses PE3 to be Function GPIO Input.

How to connect with Touch Screen Sensor by using STMPE811

If in the part of Touch Screen uses Chip STMPE11, it is I2C Interface that has the Address Position of Device for the I2C Interface as 0x82. It uses I2C1 of MCU to connect and it uses signals as follows;

- SDAT STMPE811 uses PB9 to be Function SDA1 of I2C1.
- SCLK STMPE811 uses PB8 to be Function SCL1 of I2C1.
- INT STMPE811 uses PE3 to be Function GPIO Input.

```
#define TCS_GPIO_CLK   RCC_APB2Periph_GPIOE
#define TCS_GPIO_PORT GPIOE
#define TCS_PEN_PIN   GPIO_Pin_3      // PE3 = PEN# Touch Sensor
#define TCS_MISO_PIN  GPIO_Pin_4      // PE4 = MISO Touch Sensor
#define TCS_MOSI_PIN  GPIO_Pin_5      // PE5 = MOSI Touch Sensor
#define TCS_CS_PIN    GPIO_Pin_6      // PE6 = CS# Touch Sensor
#define TCS_SCK_PIN   GPIO_Pin_7      // PE7 = SCK Touch Sensor
.
.
GPIO_InitTypeDef GPIO_InitStructure;
.
.
/* Enable GPIO clock */
RCC_APB2PeriphClockCmd(TCS_GPIO_CLK, ENABLE);

/* Configure CS in Output Push-Pull mode */
GPIO_InitStructure.GPIO_Pin = TCS_CS_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(TCS_GPIO_PORT, &GPIO_InitStructure);

/* Configure MOSI in Output Push-Pull mode */
GPIO_InitStructure.GPIO_Pin = TCS_MOSI_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(TCS_GPIO_PORT, &GPIO_InitStructure);
```

```
/* Configure SCK in Output Push-Pull mode */
GPIO_InitStructure.GPIO_Pin = TCS_SCK_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(TCS_GPIO_PORT, &GPIO_InitStructure);

/* Configure PEN as input floating */
GPIO_InitStructure.GPIO_Pin = TCS_PEN_PIN;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(TCS_GPIO_PORT, &GPIO_InitStructure);

/* Configure MISO as input floating */
GPIO_InitStructure.GPIO_Pin = TCS_MISO_PIN;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(TCS_GPIO_PORT, &GPIO_InitStructure);
```

Example of setting Pin for using Touch Screen ADS7846

```
unsigned char TCS_SPI_Read_Write(unsigned char DataByte)
{
  unsigned char Bit,result = 0 ;

  // 8 Bit Write
  for (Bit = 0; Bit < 8; Bit++)
  {
    // Clock  High(Prepared Write Data)
    GPIO_WriteBit(TCS_GPIO_PORT, TCS_SCK_PIN, Bit_SET);

    // Write Data to MOSI Pin : MSB First
    if((DataByte & 0x80)== 0x80)
    {
      // Set Bit Data(MOSI) = 1
      GPIO_WriteBit(TCS_GPIO_PORT, TCS_MOSI_PIN, Bit_SET);
    }
    else
    {
      // Reset Bit Data(MOSI) = 0
      GPIO_WriteBit(TCS_GPIO_PORT, TCS_MOSI_PIN, Bit_RESET);
    }

    // Clock Low(Strobe Data & Read)
    GPIO_WriteBit(TCS_GPIO_PORT, TCS_SCK_PIN, Bit_RESET);

    // Shift Next Bit Data
    DataByte <<= 1;

    // Read Data From MISO Pin
    result <<= 1;

    if (GPIO_ReadInputDataBit(TCS_GPIO_PORT,TCS_MISO_PIN) == Bit_SET)
    {
      result |= 0x01;
    }
  }
  return (result);
}
```

It shows an example function of reading/writing data into ADS7846 by using GPIO that imitates SPI

```
// ET-STM32F ARM KIT(STM32F107VCT6) Hardware Kit
// I2C1(Remap)  = PB8:SCL1
//              = PB9:SDA1
#define I2C_TCS                       I2C1
#define I2C_TCS_CLK                   RCC_APB1Periph_I2C1
#define I2C_TCS_GPIO                  GPIOB
#define I2C_TCS_GPIO_CLK              RCC_APB2Periph_GPIOB
#define I2C_TCS_SCL                   GPIO_Pin_8
#define I2C_TCS_SDA                   GPIO_Pin_9
#define I2C_TCS_Speed                 400000
#define I2C_TCS_SLAVE_ADDRESS7        0x82
.
.
.
GPIO_InitTypeDef  GPIO_InitStructure;
I2C_InitTypeDef  I2C_InitStructure;

/* I2C Periph clock enable */
RCC_APB1PeriphClockCmd(I2C_TCS_CLK, ENABLE);

/* GPIO Periph clock enable */
RCC_APB2PeriphClockCmd(I2C_TCS_GPIO_CLK, ENABLE);

/* Enable the I2C1 Pins Software Remapping */
GPIO_PinRemapConfig(GPIO_Remap_I2C1, ENABLE);

/* Configure I2C_TCS pins: SCL and SDA */
GPIO_InitStructure.GPIO_Pin =  I2C_TCS_SCL | I2C_TCS_SDA;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
GPIO_Init(I2C_TCS_GPIO, &GPIO_InitStructure);


/* I2C configuration */
I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
I2C_InitStructure.I2C_OwnAddress1 = I2C_TCS_SLAVE_ADDRESS7;
I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
I2C_InitStructure.I2C_AcknowledgedAddress=I2C_AcknowledgedAddress_7
bit;
```

```
I2C_InitStructure.I2C_ClockSpeed = I2C_TCS_Speed;

/* I2C Peripheral Enable */
I2C_Cmd(I2C_TCS, ENABLE);
/* Apply I2C configuration after enabling it */
I2C_Init(I2C_TCS, &I2C_InitStructure);
.
.
.
```

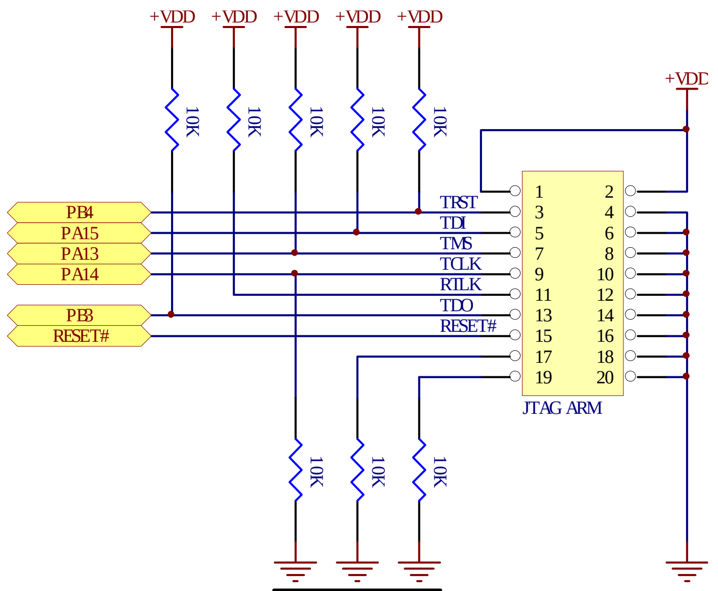
Example of setting Pin for using Touch Sensor STMPE811

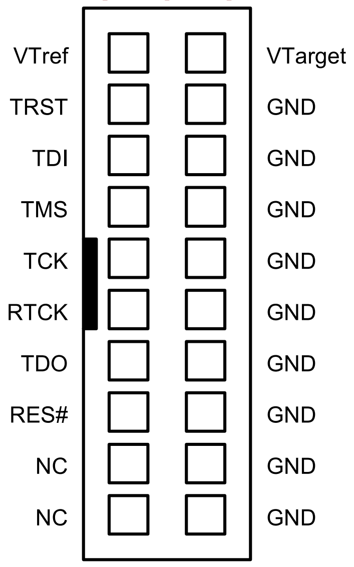
```
#define LCD_CS_PIN              GPIO_Pin_8              // PC8 = CS# GLCD
#define LCD_CS_GPIO_PORT        GPIOC
#define LCD_CS_GPIO_CLK         RCC_APB2Periph_GPIOC
#define LCD_BL_PIN              GPIO_Pin_7              // PD7 = BL GLCD
#define LCD_BL_GPIO_PORT        GPIOD
#define LCD_BL_GPIO_CLK         RCC_APB2Periph_GPIOD

#define LCD_SPI_SCK_PIN         GPIO_Pin_10              //SPI3
#define LCD_SPI_MISO_PIN        GPIO_Pin_11
#define LCD_SPI_MOSI_PIN        GPIO_Pin_12
#define LCD_SPI_GPIO_PORT       GPIOC
#define LCD_SPI_GPIO_CLK        RCC_APB2Periph_GPIOC
#define LCD_SPI                 SPI3
#define LCD_SPI_CLK             RCC_APB1Periph_SPI3

GPIO_InitTypeDef GPIO_InitStructure;
SPI_InitTypeDef  SPI_InitStructure;

/* Enable GPIO clock */
RCC_APB2PeriphClockCmd(LCD_CS_GPIO_CLK | LCD_BL_GPIO_CLK, ENABLE);

/* Configure NCS in Output Push-Pull mode */
GPIO_InitStructure.GPIO_Pin = LCD_CS_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(LCD_CS_GPIO_PORT, &GPIO_InitStructure);

/* Configure BL in Output Push-Pull mode */
GPIO_InitStructure.GPIO_Pin = LCD_BL_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(LCD_BL_GPIO_PORT, &GPIO_InitStructure);

RCC_APB2PeriphClockCmd(LCD_SPI_GPIO_CLK|RCC_APB2Periph_AFIO, ENABLE);
GPIO_PinRemapConfig(GPIO_Remap_SPI3, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI3, ENABLE);

/* Configure SPI pins: SCK, MISO and MOSI */
GPIO_InitStructure.GPIO_Pin = LCD_SPI_SCK_PIN |
                              LCD_SPI_MISO_PIN |
                              LCD_SPI_MOSI_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(LCD_SPI_GPIO_PORT, &GPIO_InitStructure);

/* SPI Config */
SPI_I2S_DeInit(LCD_SPI);
SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_High;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
SPI_Init(LCD_SPI, &SPI_InitStructure);
SPI_Cmd(LCD_SPI, ENABLE);
```

Example of setting Pin for using GLCD

JTAG ARM

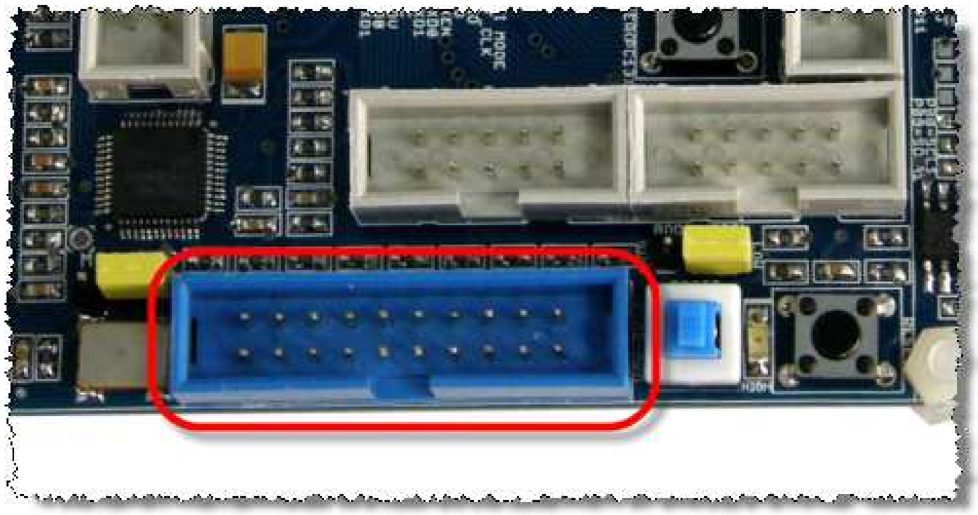
JTAG or JTAG ARM is Connector IDE 20Pin to interface with JTAG Debugger. It arranges circuit signals according to standard of JTAG as follows;





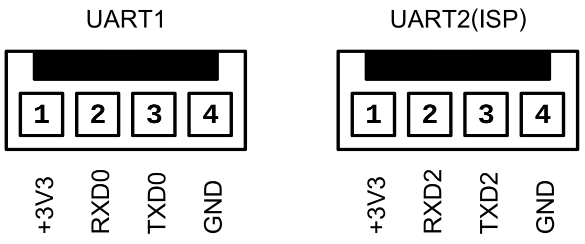
ARM JTAG Pin Connector

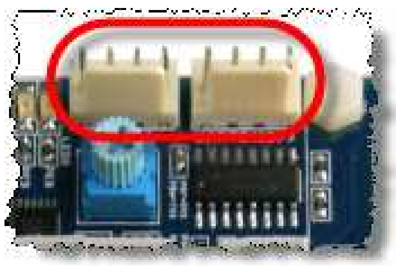
↓



Port RS232

This port is Signal RS232 that has already been converted by MAX232. In this case, there are 2 channels; UART1 and UART2. Both channels can be connected with Signal RS232 to transmit/receive data; moreover, UART2 can be used with ISP Download to download HEX File into MCU. In this case, it needs to use with Jumper BOOT1, Switch BOOT0 and Switch Reset to reset CPU to start running in Boot-Loader Mode and it can also Download Hex File into CPU (read more details from "How to Download HEX File into CPU of Board").





- PB6 is TXD1(USART1_TX:Remap), PB7 is RXD1(USART1_RX:Remap).
- PD5 is TXD2(USART2_TX:Remap), PD6 is RXD2(USART2_RX:Remap).

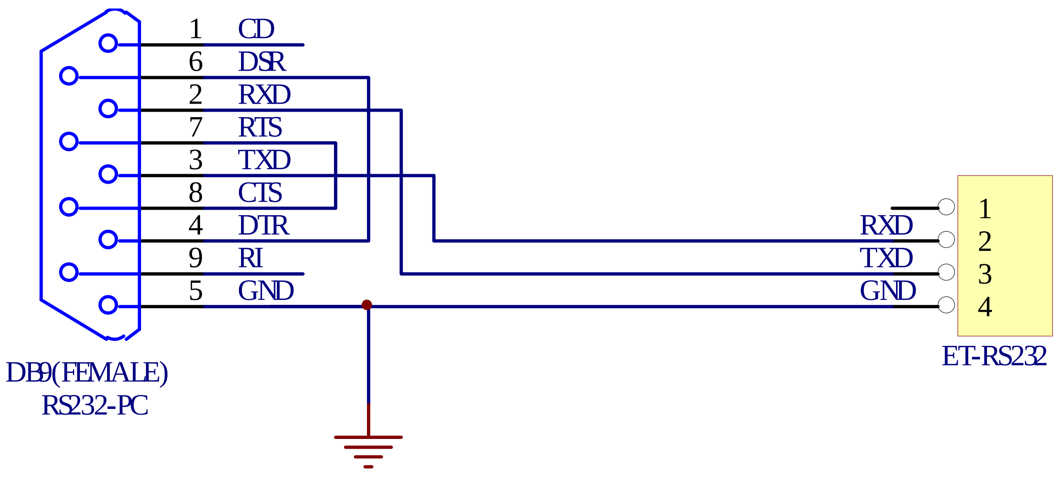
There are 2 sets of Pin that can be set by Hardware USART System of STM32F107VCT6 to interface; Default and Remap. This Board ET-STM32F ARM KIT uses the set of Pin Remap to be the connecting point with UART; so, user has to set commands to choose pin correctly. The example Code of setting UART in the initial part is shown below;

```
// ET-STM32F ARM KIT(STM32F107VCT6) Hardware Kit
// UART1(Remap) = PB7:RX1,PB6:TX1
#define EVAL_COM1           USART1                    //COM1 = USART1
#define EVAL_COM1_GPIO      GPIOB                     //USART1  Port  =
PB
#define EVAL_COM1_CLK       RCC_APB2Periph_USART1     //Enable   USART1
Clock
#define EVAL_COM1_GPIO_CLK  RCC_APB2Periph_GPIOB      //Enable PB Clock
#define EVAL_COM1_RxPin     GPIO_Pin_7                //RX1=PB7
#define EVAL_COM1_TxPin     GPIO_Pin_6                //TX1=PB6

// UART2(Remap) = PD6:RX2,PD5:TX2
#define EVAL_COM2           USART2                    //COM2 = USART2
#define EVAL_COM2_GPIO       GPIOD                       //USART2 Port =
PD
#define EVAL_COM2_CLK       RCC_APB1Periph_USART2     //Enable    UART2
Clock
#define EVAL_COM2_GPIO_CLK  RCC_APB2Periph_GPIOD      //Enable PD Clock
#define EVAL_COM2_RxPin     GPIO_Pin_6                //RX2=PD6
#define EVAL_COM2_TxPin     GPIO_Pin_5                //TX2=PD5
GPIO_InitTypeDef GPIO_InitStructure;

/* Enable GPIO clock */
RCC_APB2PeriphClockCmd(EVAL_COM1_GPIO_CLK     |     RCC_APB2Periph_AFIO,
ENABLE);

/* Enable the USART1 Pins Software Remapping */
GPIO_PinRemapConfig(GPIO_Remap_USART1, ENABLE);
RCC_APB2PeriphClockCmd(EVAL_COM1_CLK, ENABLE);

/* Configure USART Tx as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = EVAL_COM1_TxPin;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(EVAL_COM1_GPIO, &GPIO_InitStructure);

/* Configure USART Rx as input floating */
GPIO_InitStructure.GPIO_Pin = EVAL_COM1_RxPin;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(EVAL_COM1_GPIO, &GPIO_InitStructure);

/* USART configuration */
USART_Init(EVAL_COM1, USART_InitStruct);

/* Enable USART */
USART_Cmd(EVAL_COM1, ENABLE);
```

```
/* Enable GPIO clock */
RCC_APB2PeriphClockCmd(EVAL_COM2_GPIO_CLK     |     RCC_APB2Periph_AFIO,
ENABLE);

/* Enable the USART2 Pins Software Remapping */
GPIO_PinRemapConfig(GPIO_Remap_USART2, ENABLE);
RCC_APB1PeriphClockCmd(EVAL_COM2_CLK, ENABLE);

/* Configure USART Tx as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = EVAL_COM2_TxPin;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(EVAL_COM2_GPIO, &GPIO_InitStructure);

/* Configure USART Rx as input floating */
GPIO_InitStructure.GPIO_Pin = EVAL_COM2_RxPin;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(EVAL_COM2_GPIO, &GPIO_InitStructure);

/* USART configuration */
USART_Init(EVAL_COM2, USART_InitStruct);

/* Enable USART */
USART_Cmd(EVAL_COM2, ENABLE);
```

Example of setting Pin for UART1 and UART2

The Cable that connects RS232 between Comport of computer PC and Connector UART1 and UART2 of Board ET-STM32F ARM KIT is shown below;



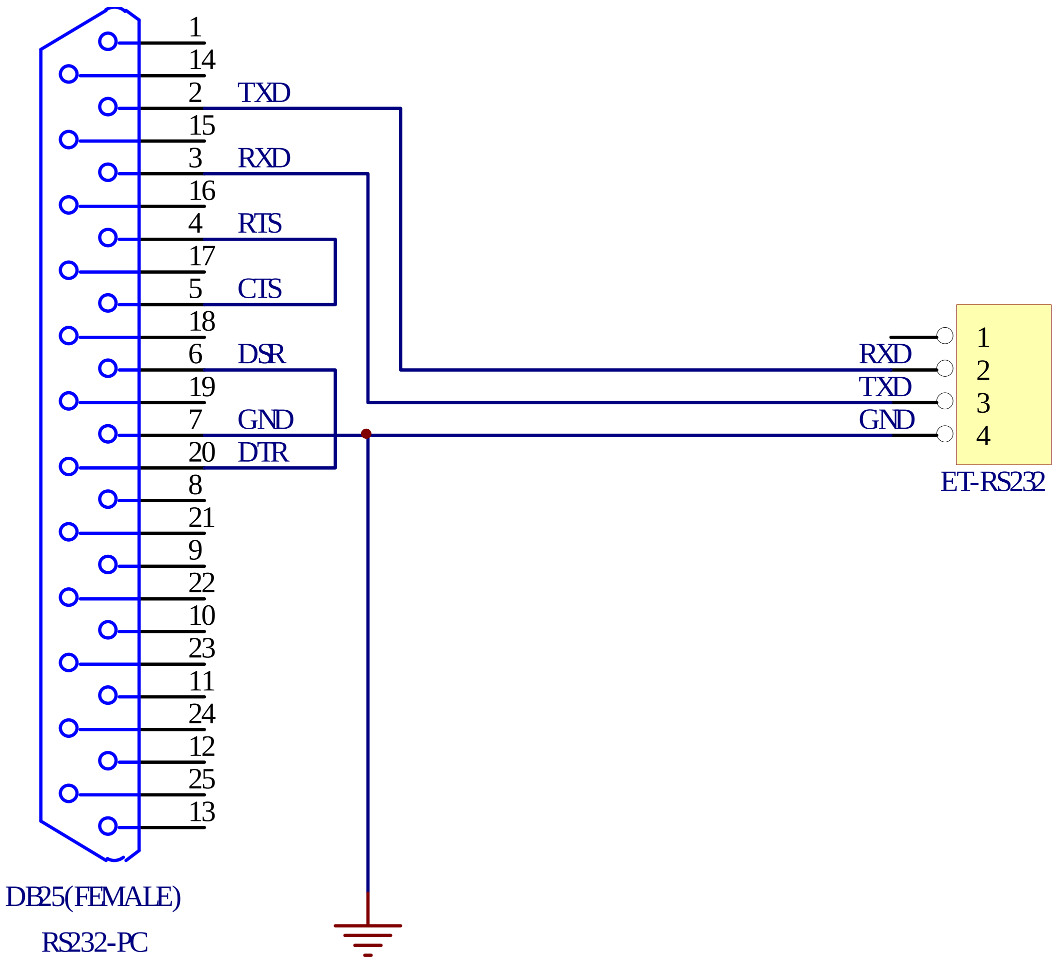


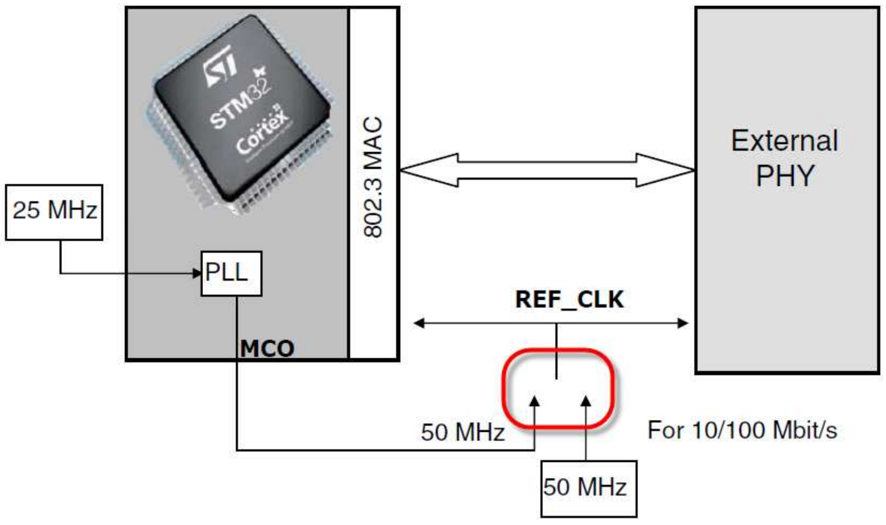
Figure displays Circuit of Cable for RS232.

Ethernet LAN

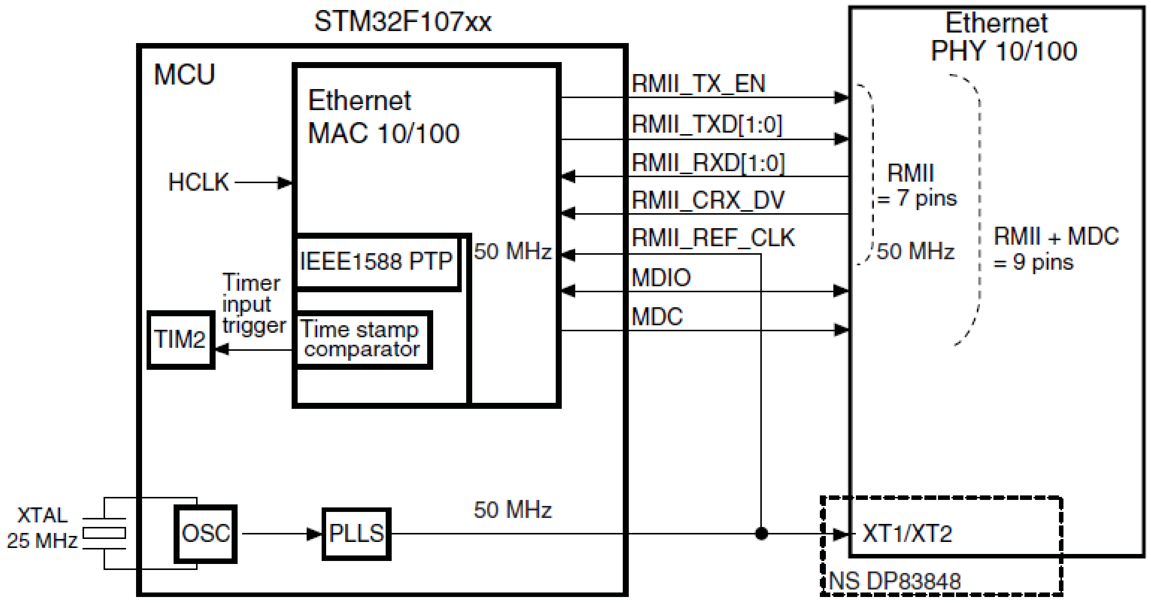
The connection between Network Ethernet LAN and Board ET-STM32F ARM KIT is to use the standard Connector RJ45 Ethernet. This circuit in this part uses signal pin to connect that is Chip Physical Ethernet No.DP83848V to be Driver and uses circuit to connect signals between STM32F107VCT6 and DP83848V in Mode RMI(Reduced Media Independent Interface). In this case, there are 9 cables when it uses Signal Clock from Module Oscillator 50MHz; on the other hand, there are 10 cables when it uses Signal Clock from Circuit MCO(PIN PA8) as follows;

- PA1 is REF_CLK(Default).
- PA2 is MDIO(Default).
- PC1 is MDC(Default).
- PB11 is TX_EN(Default).
- PB12 is TXD0(Default).
- PB13 is TXD1(Default).
- PD8 is CRS_DV(RMII Remap).
- PD9 is RXD0(RMII Remap).
- PD10 is RXD1(RMII Remap).

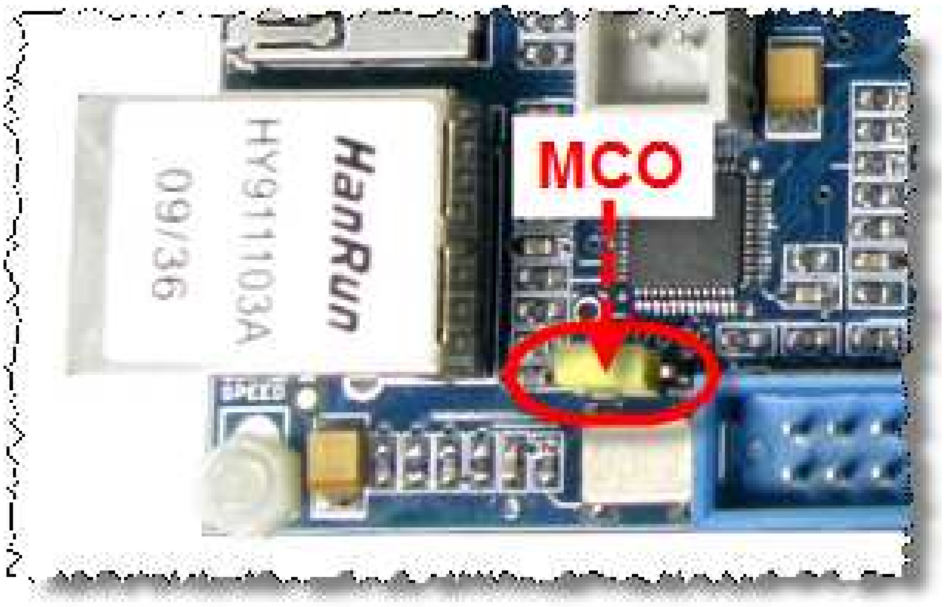
There are 2 sources of Signal Clock 50MHz that can be chosen by Hardware System of Board ET-STM32F ARM KIT to supply to Circuit Ethernet LAN Driver. Firstly, it is from Circuit MCO internal MCU (Pin PA8); and secondly, it is from external Module Oscillator 50MHz. In this case, it uses Jumper MCO/OSC to choose the source of Signal Clock.



In case of using Signal Clock 50MHz from Circuit MCO(Pin PA8); there are 10 pins for connection and it uses Pin PA8 to generate Signal Clock 50MHz and supply to DP83848V. In the part of Board ET-STM32F ARM KIT, it needs to set Jumper MCO/OSC to the MCO position. Moreover, it adds the part of program to initial the operation of Circuit MCO(Main Clock Oscillator) to generate Signal Clock 50MHz and then send out through Pin PA8 as shown below;

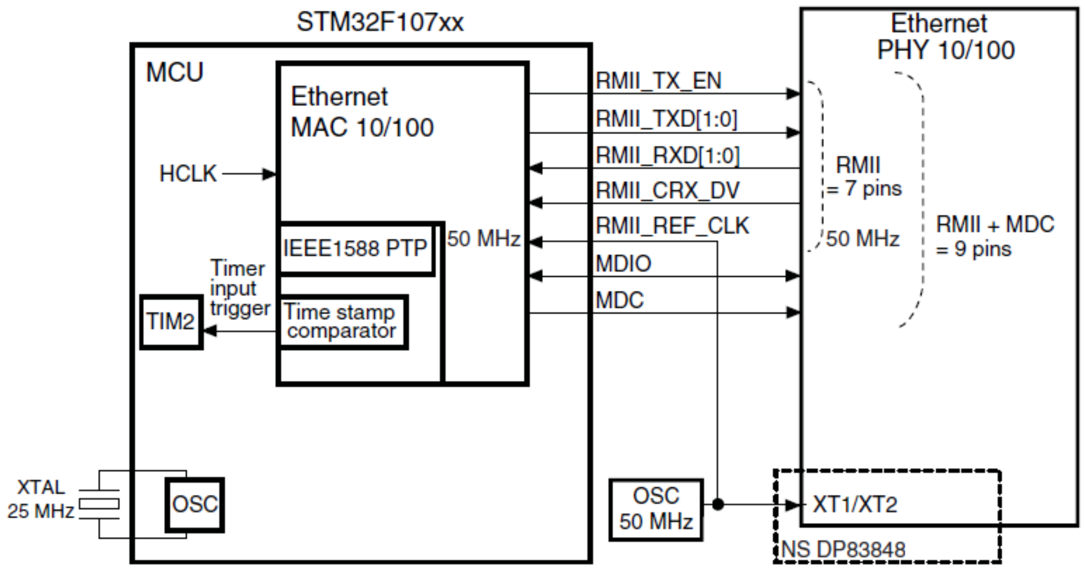


↓

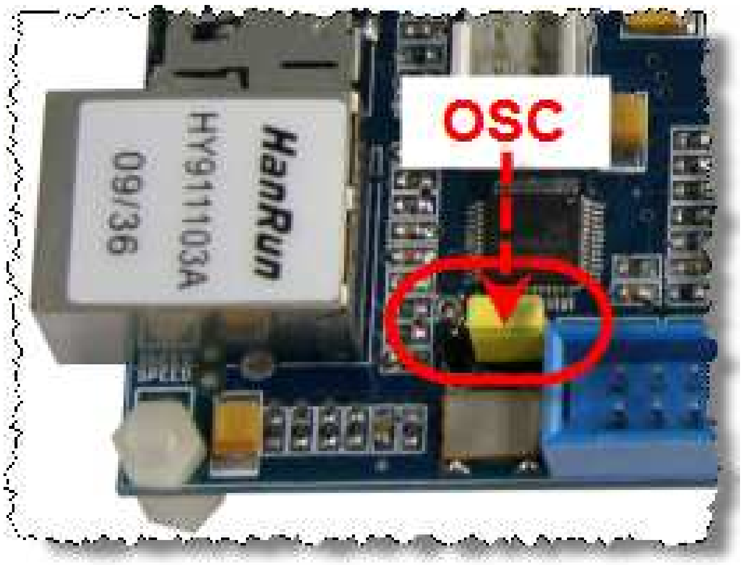


It displays how to connect Ethernet LAN in Mode RMII by using MCO to generate Signal Clock 50MHz.

In case of using Signal Clock 50MHz from Module Oscillator, there are 9 pins for connection. It uses Oscillator 50MHz to generate Signal Clock 50MHz and then supply to DP83848V. In the part of Board ET-STM32F ARM KIT, it needs to set Jumper MCO/OSC to the OSC position as shown below;



↓



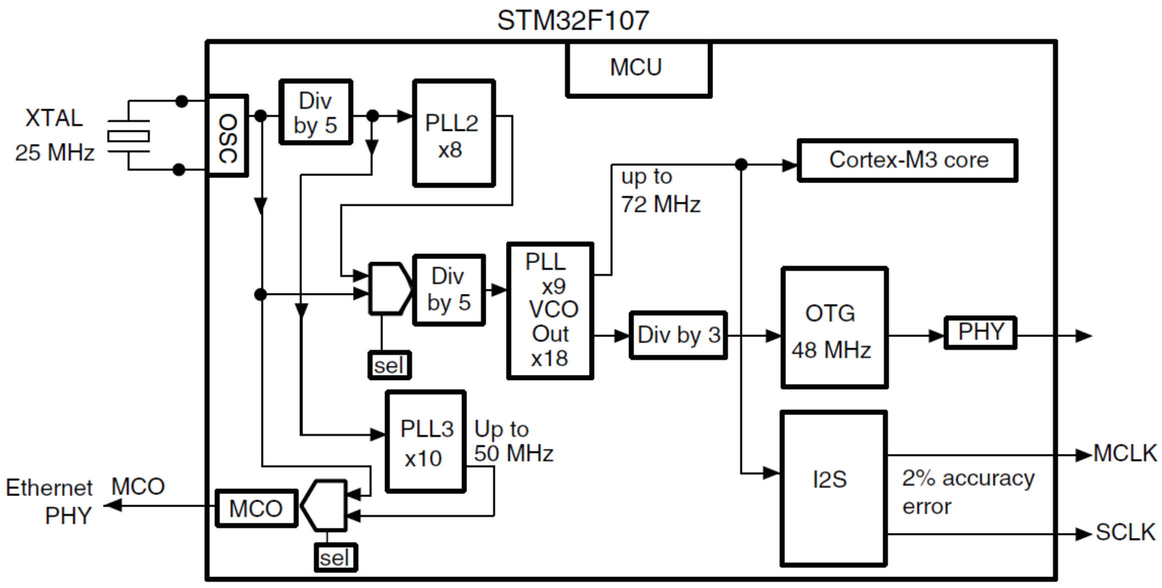
It displays how to connect Ethernet LAN in Mode RMII by using Module Oscillator 50MHz.

```
GPIO_InitTypeDef GPIO_InitStructure;
.
.
/* Configure PA2 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;       //PA2 = ETH_RMII_MDIO
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Configure PC1 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;       //PC1 = ETH_RMII_MDC
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* Configure PB11, PB12 and PB13 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 |     //PB11 = ETH_RMII_TXEN
                              GPIO_Pin_12 |     //PB12 = ETH_RMII_TXD0
                              GPIO_Pin_13;      //PB13 = ETH_RMII_TXD1
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOB, &GPIO_InitStructure);

/* Configure PA1 as input */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;       //PA1                  =
ETH_RMII_REF_CLK
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);

// ETHERNET pins remapp in ET-STM32F ARM KIT board: RX_DV and RxD[1:0]
// PD8=CRS_DV(RMII Remap),PD9=RXD0(RMII Remap),PD10=RXD1(RMII Remap)
GPIO_PinRemapConfig(GPIO_Remap_ETH, ENABLE);

/* Configure PD8, PD9, PD10 as input */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 |      //PD8                  =
ETH_RMII_CRS_DV
                              GPIO_Pin_9 |      //PD9  = ETH_RMII_RXD0
                              GPIO_Pin_10;      //PD10 = ETH_RMII_RXD1
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

It is example of Initial Pin to interface with Ethernet Drive No.DP83848V.

There is Circuit MCO(Main Clock Output) internal the Hardware system of STM32; it supplies the main Signal Clock to MCU(Crystal 25MHz) that has already been divided by 5(5MHz) and it multiplies frequency by Circuit Phase-Lock-Loop(PLL3) to increase the frequency by 10 times. It uses this frequency to generate the

Signal Clock 50MHz to supply to Ethernet Driver(DP83848V); in this case, it uses Pin PA8 with Phase-Lock-Loop(PLL3) as below;



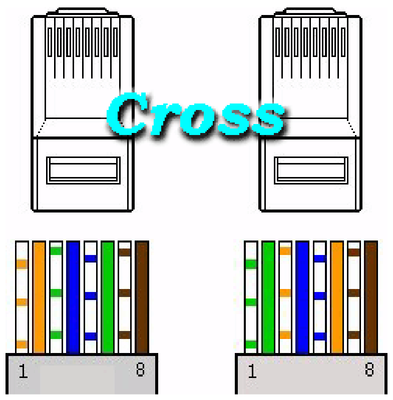
↓

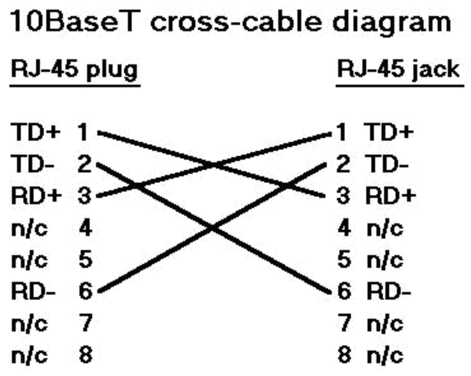
```
/* Start of Config MCO Clock = 50MHz on PA8 */
// Configure MCO (PA8) as alternate function push-pull
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;             //PA8 = MCO
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

// set PLL3 clock output to 50MHz (25MHz / 5 * 10 = 50MHz)
RCC_PLL3Config(RCC_PLL3Mul_10);

// Enable PLL3
RCC_PLL3Cmd(ENABLE);

// Wait till PLL3 is ready
while (RCC_GetFlagStatus(RCC_FLAG_PLL3RDY) == RESET){}

// Get clock PLL3 clock on PA8 pin
RCC_MCOConfig(RCC_MCO_PLL3CLK);
/*End of Initial MCO Clock = 50MHz on PA8 */
```

When Pin PA8 is chosen to be function MCO Output to generate Signal Clock 50MHz, it saves the Hardware costs because it is unnecessary to use Module Oscillator to generate Frequency 50MHz any more.

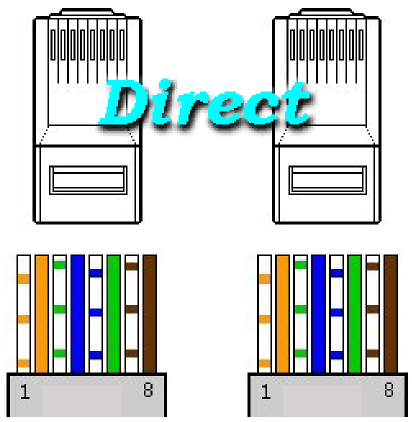
There are 2 methods to interface Cable Ethernet LAN of Board and Network System; Direct Line Interface and Hub Interface.

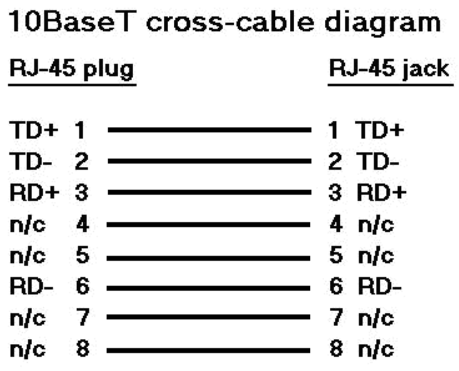
- **1st case**: It directly connects cable with computer PC; in this case, it must interface LAN Cable as Cross type.





- **2nd case**: It interfaces signal through Hub of computer Server and it must interface cable as Direct type.

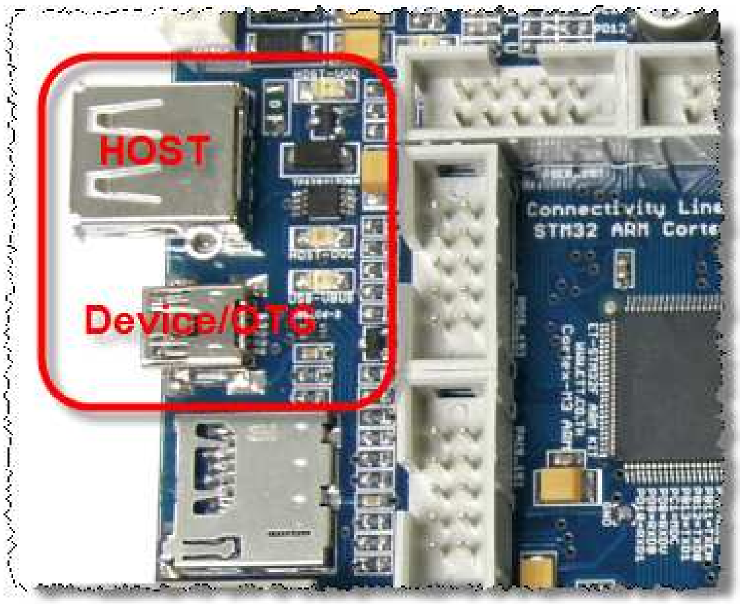




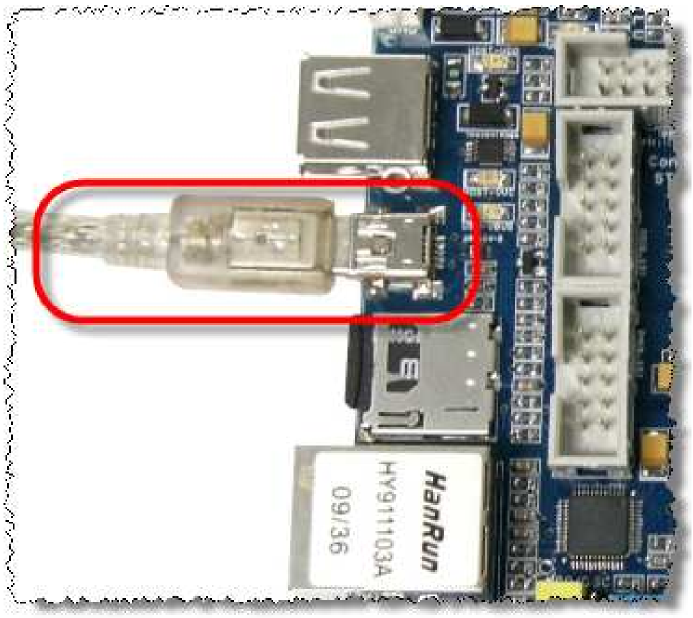
USB

Board ET-STM32F ARM KIT has Port USB for either Device/OTG(On-The-Go) or USB Host; in this case, it depends on writing program to set function of USB internal MCU of STM32F107VCT6. There are 6 cables that are related to USB Interface as follows;

- PE1 is Function GPIO Input(Host OVC:Host Over Current).
- PC9 is Function GPIO Output (HOST_EN:Host Enable).
- PA9 is Function OTG_FS_VBUS
- PA10 is Function OTG_FS_ID.
- PA11 is Function OTG_FS_DM.
- PA12 is Function OTG_FS_DP.



USB Device



↓

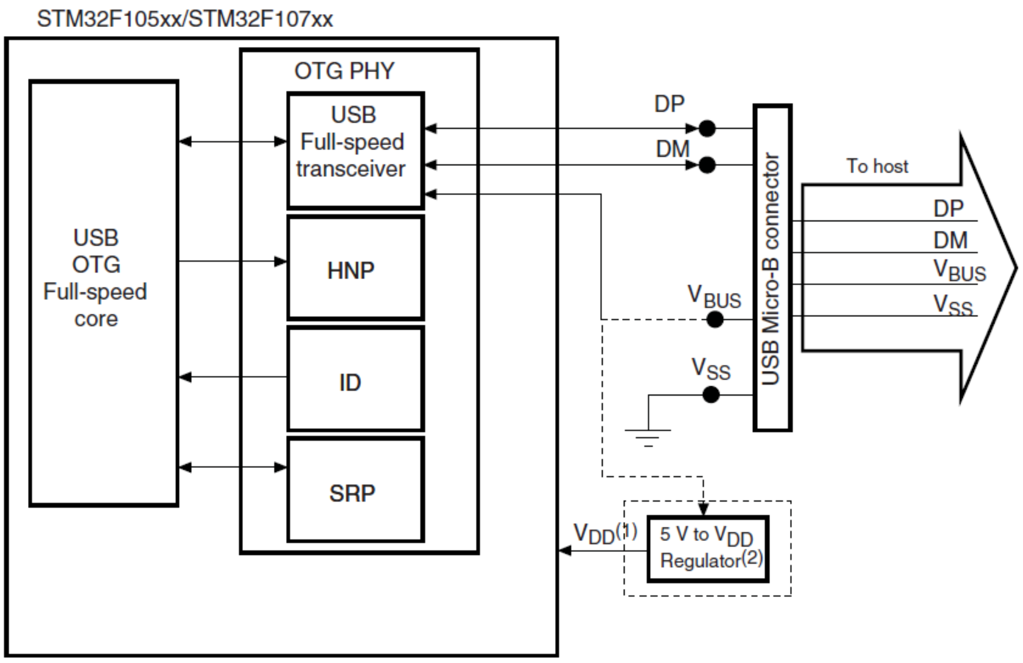
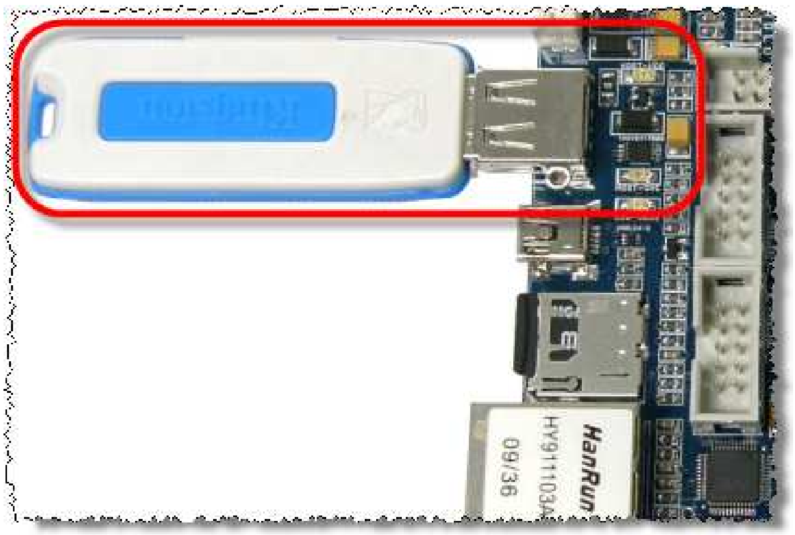


Diagram shows the USB connection as Device Mode.

For this operation mode, there are 3 cables to connect as follows;

- o PA9 is Function USB_VBUS.
- o PA11 is Function USB_DM.
- o PA12 is Function USB_DP.

USB Host



↓

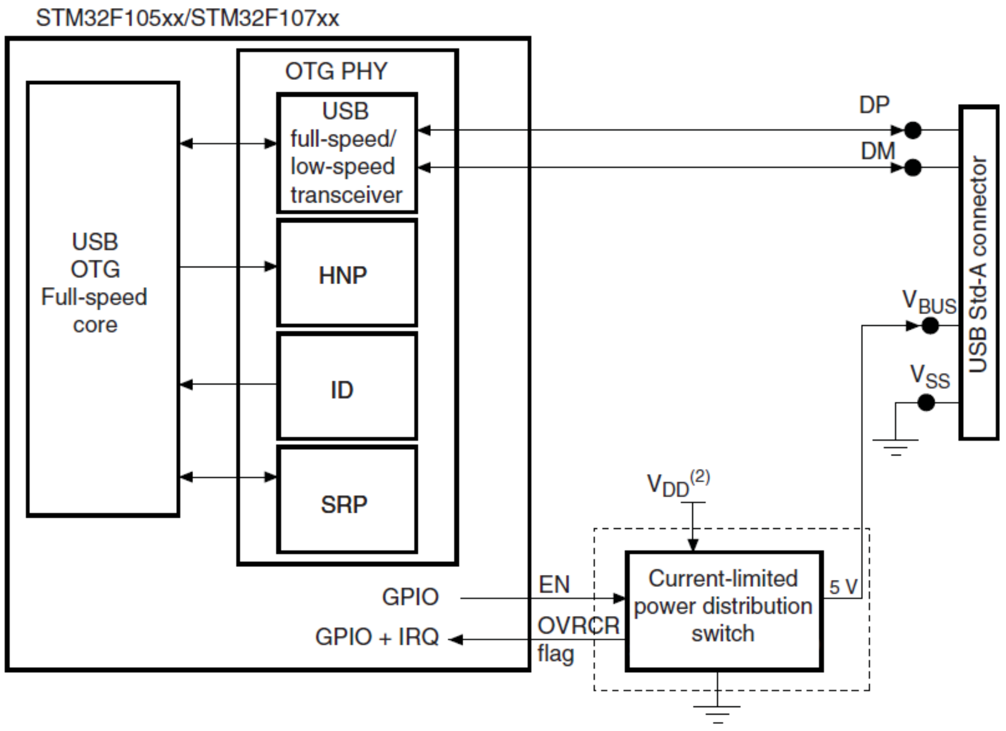


Diagram shows the USB Connection as Host Mode.

For this operation mode, there are 4 cables to connect as follows;

- o PE1 is Function GPIO Input(Host OVRCR:Host Over Current).

- PC9 is Function GPIO Output(HOST_EN:Host Enable).
- PA11 is Function USB_DM.
- PA12 is Function USB_DP.

USB OTG

↓

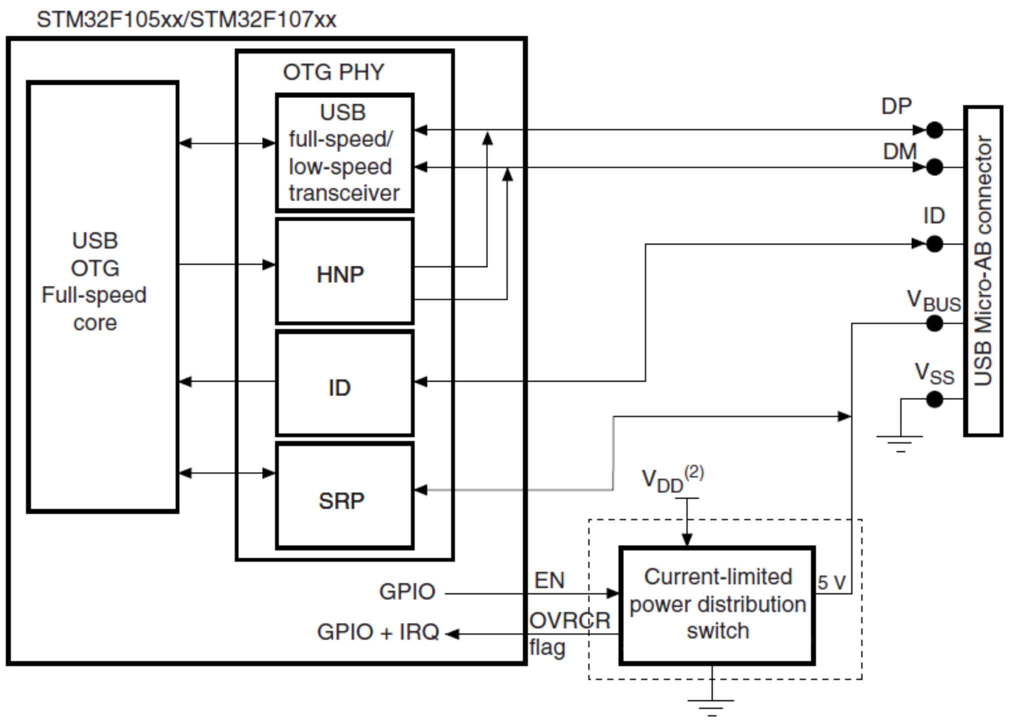


Diagram shows the USB connection as OTG Mode.

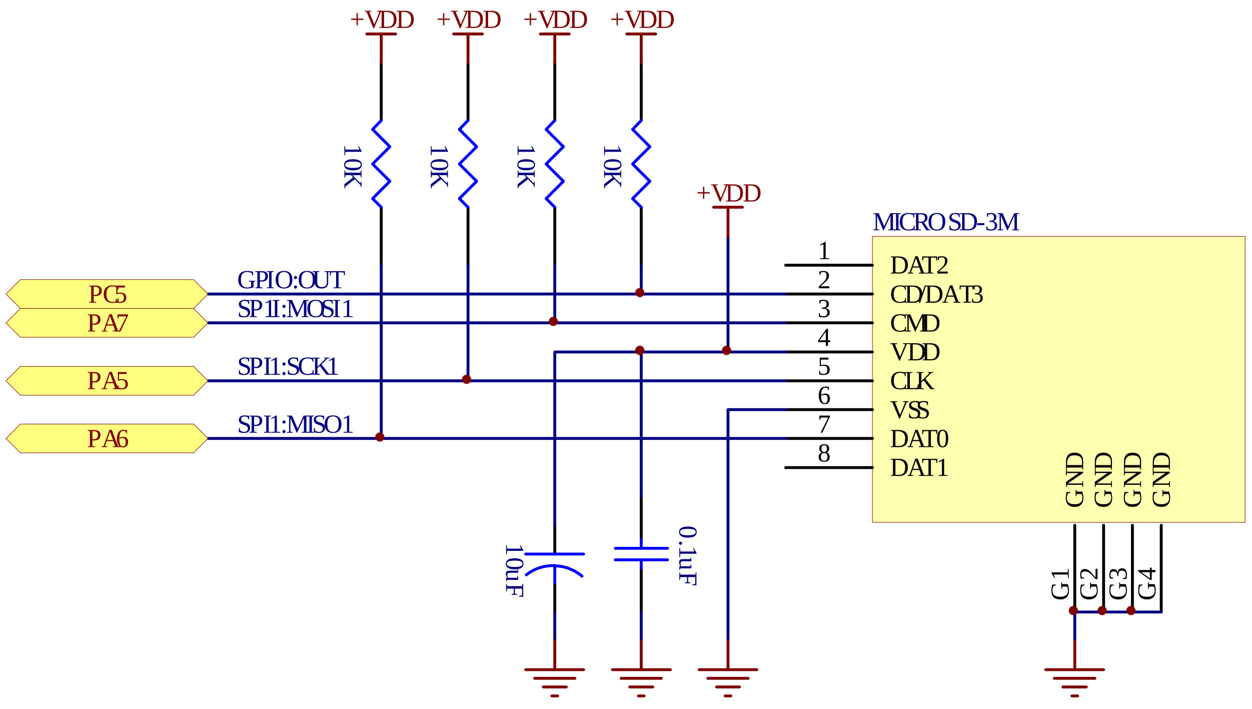
For this operation mode, there are 6 cables to connect as follows;

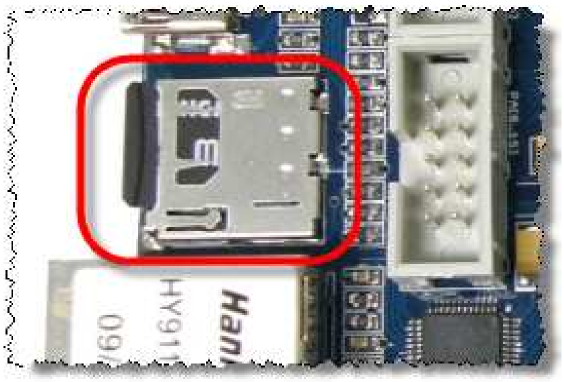
- o PE1 is Function GPIO Input(Host OVRCR:Host Over Current).
- o PC9 is Function GPIO Output(HOST_EN:Host Enable).
- o PA9 is Function USB_VBUS.
- o PA10 is Function USB_ID.
- o PA11 is Function USB_DM.
- o PA12 is Function USB_DP.

Micro-SD Card

The Structure of Board ET-STM32F ARM KIT supports the connection with Micro-SD Card. It is SPI Interface by using Pin PC5,PA[5..7] to connect with the card. If user wants to command the card, can program Pin I/O of PC5 and PA[5..7] to operate in Mode SPI. Moreover, it needs to set Pin funtion of MCU as follows;

- CLK uses PA5 to be Function SCK of SPI1.
- DAT0 uses PA6 to be Function MISO of SPI1.
- CMD uses PA7 to be Function MOSI of SPI1.
- CD/DAT3 uses PC5 to be Function GPIO Output.



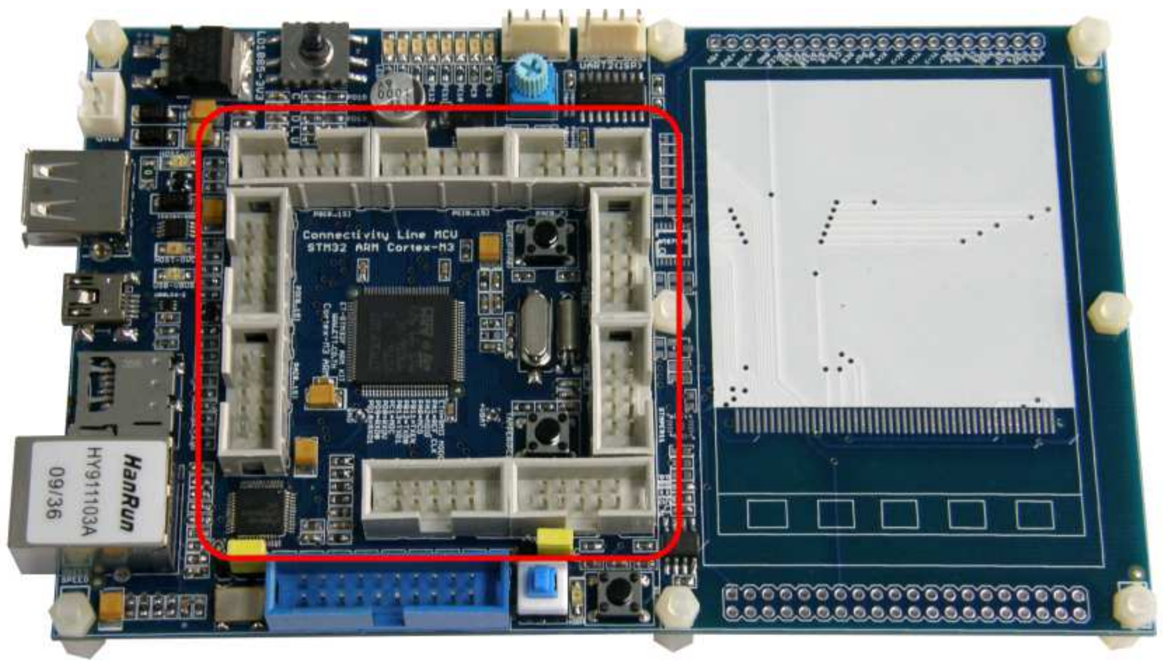


```
// ET-STM32F ARM KIT(STM32F107VCT6) Hardware Kit
// SD Card Interface(PC5 = CS#,PA5 = SCK,PA6 = MISO,PA7 MOSI)
#define SD_SPI                      SPI1
#define SD_SPI_PORT                 GPIOA
#define SD_SPI_GPIO_PORT_CLOCK      RCC_APB2Periph_GPIOA
#define SD_SPI_PIN_SCK              GPIO_Pin_5
#define SD_SPI_PIN_MISO             GPIO_Pin_6
#define SD_SPI_PIN_MOSI             GPIO_Pin_7
#define SD_CS_PORT                  GPIOC
#define SD_CS_GPIO_PORT_CLOCK       RCC_APB2Periph_GPIOC
#define SD_CS_PIN                   GPIO_Pin_5
#define SD_CS_LOW()
GPIO_ResetBits(SD_CS_PORT,SD_CS_PIN)
#define SD_CS_HIGH()                 GPIO_SetBits(SD_CS_PORT,
SD_CS_PIN)

GPIO_InitTypeDef  GPIO_InitStructure;
SPI_InitTypeDef   SPI_InitStructure;

/* SD_SPI_PORT and SD_CS_PORT Periph clock enable */
RCC_APB2PeriphClockCmd(SD_SPI_GPIO_PORT_CLOCK |
                       SD_CS_GPIO_PORT_CLOCK | \
                       RCC_APB2Periph_AFIO, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);

/* Configure SD_SPI pins: SCK, MISO and MOSI */
GPIO_InitStructure.GPIO_Pin = SD_SPI_PIN_SCK |
                              SD_SPI_PIN_MISO |
                              SD_SPI_PIN_MOSI;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(SD_SPI_PORT, &GPIO_InitStructure);

/* Configure CS pin */
GPIO_InitStructure.GPIO_Pin = SD_CS_PIN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(SD_CS_PORT, &GPIO_InitStructure);
```

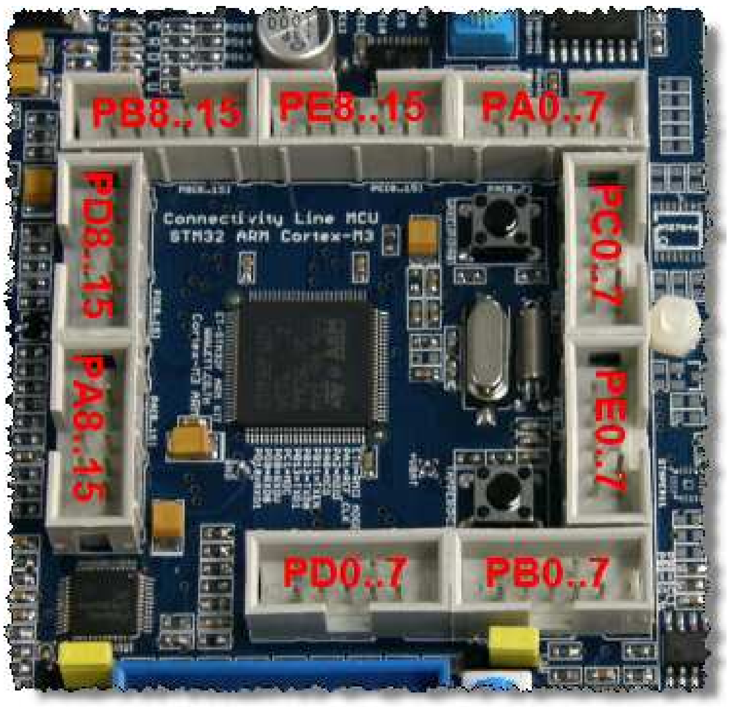
```
/* SD_SPI Config */
SPI_InitStructure.SPI_Direction =
SPI_Direction_2Lines_FullDuplex;
SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;

/* Clock speed = fPCLK1 / 256 = 280 kHz at 72 MHz PCLK1 clk. */
SPI_InitStructure.SPI_BaudRatePrescaler =
SPI_BaudRatePrescaler_256;
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
SPI_InitStructure.SPI_CRCPolynomial = 7;
SPI_Init(SD_SPI, &SPI_InitStructure);

/* SD_SPI enable */
SPI_Cmd(SD_SPI, ENABLE);
```

Example of setting Pin for SD Card

Port I/O Connectors of Board

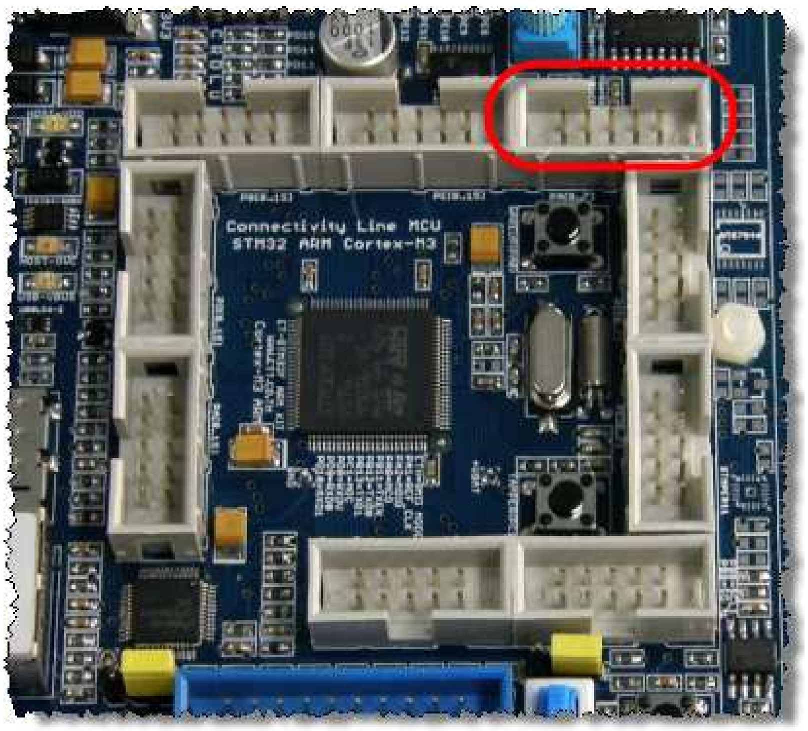


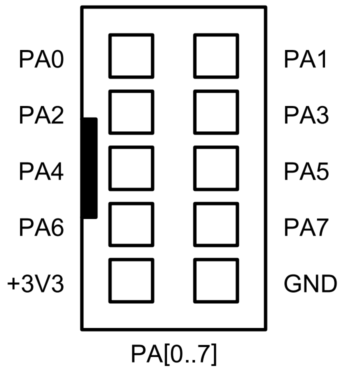
↓



Connector Port I/O of CPU is arranged outside, so user can choose and connect them as preferred. There are 9 sets of Connector IDE 10Pin and each set has 8 Bit. It arranges signal of connector for each set as follows;

- Connector IDE 10Pin of PA[0..7] is arranged as follows;

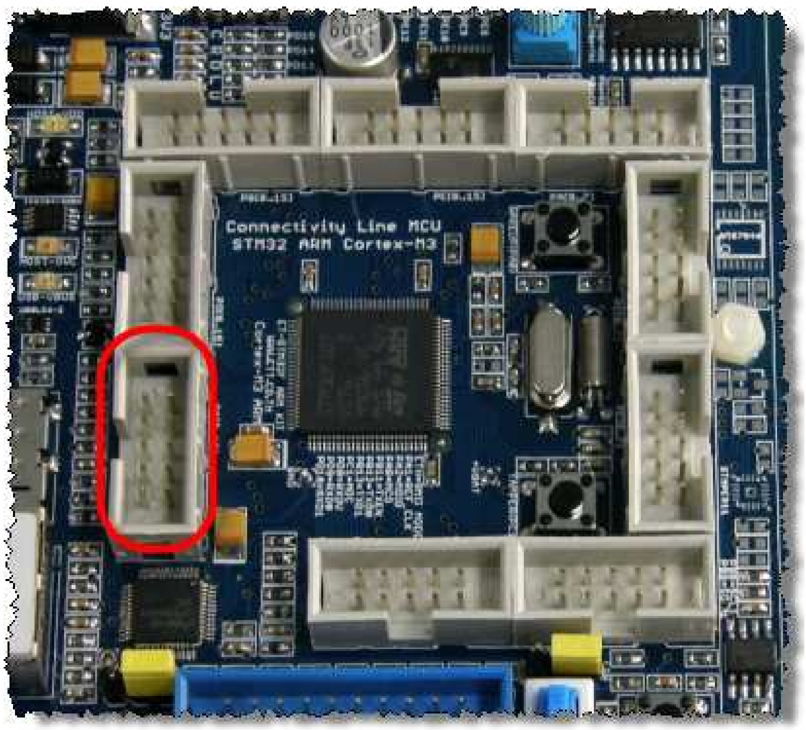




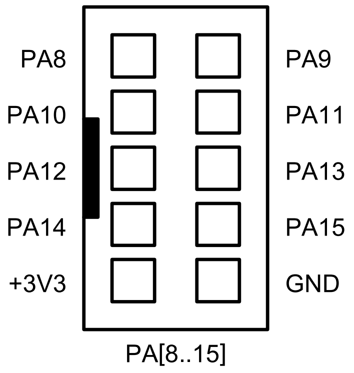
  - PA0 has been chosen to use with Switch Wakeup.
  - PA1 has been chosen to be Function REF_CLK of RMII to interface with Ethernet.
  - PA2 has been chosen to be Function MDIO of RMII to interface with Ethernet.
  - PA5 has been chosen to be Function SCK of SPI1 to interface with SD Card.

- PA6 has been chosen to be Function MISO of SPI1 to interface with SD Card.
- PA7 has been chosen to be Function MOSI of SPI1 to interface with SD Card.

- Connector IDE 10Pin of PA[8..15] is arranged as follows;



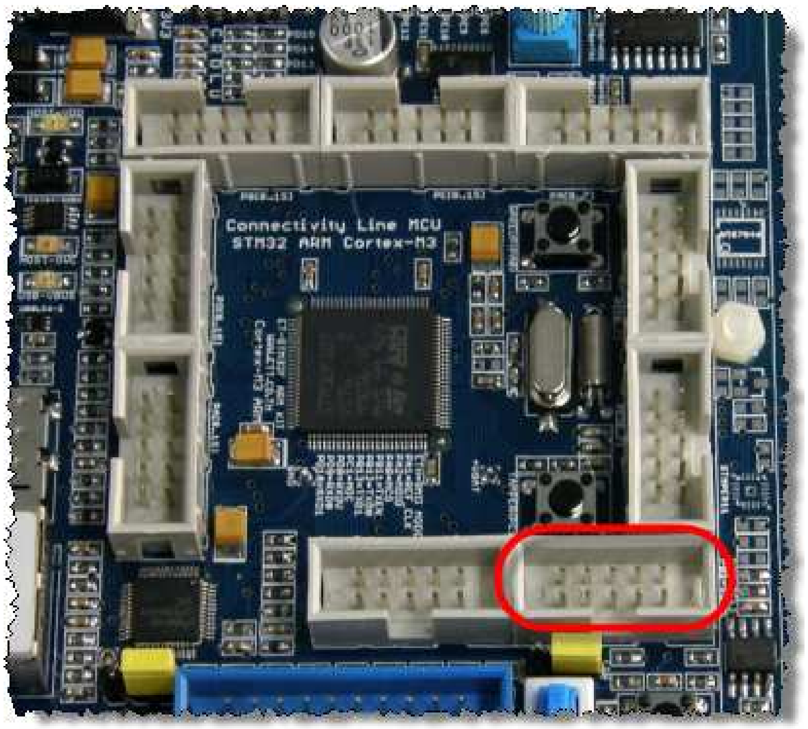
↓



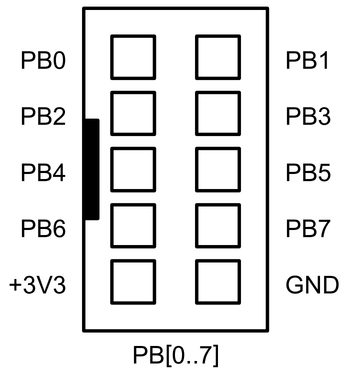
- PA8 has been chosen to be Function MCO to interface with Ethernet.

- PA9 has been chosen to be Function USB_FS_VBUS.
- PA10 has been chosen to be Function USB_FS_ID.
- PA11 has been chosen to be Function USB_FS_DM.
- PA12 has been chosen to be Function USB_FS_DP.
- PA13 has been chosen to be Function TMS to interface with JTAG.
- PA14 has been chosen to be Function TCLK to interface with JTAG.
- PA15 has been chosen to be Function TDI to interface with JTAG.

- Connector IDE 10Pin of PB[0..7] is arranged as follows;

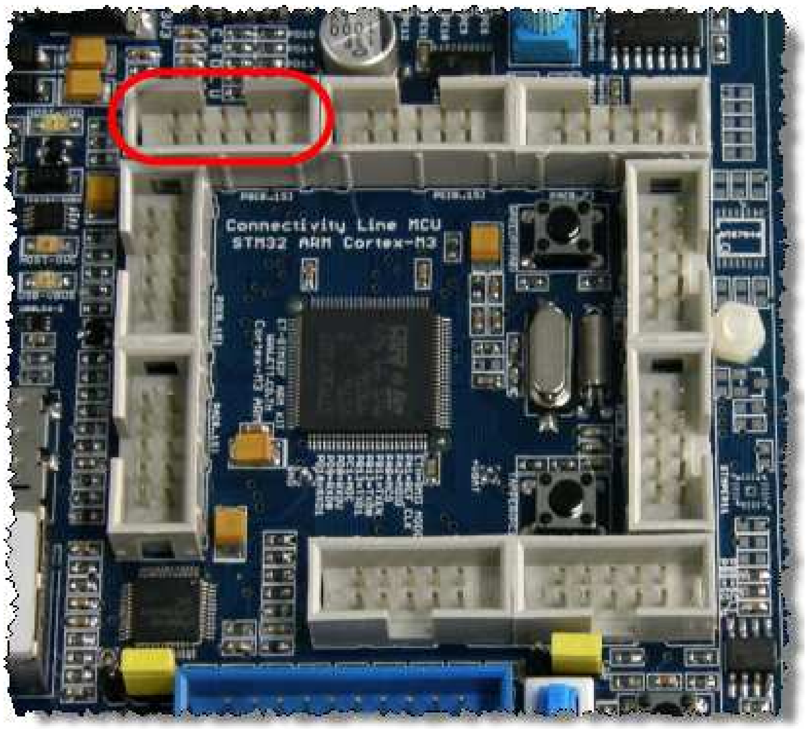


↓

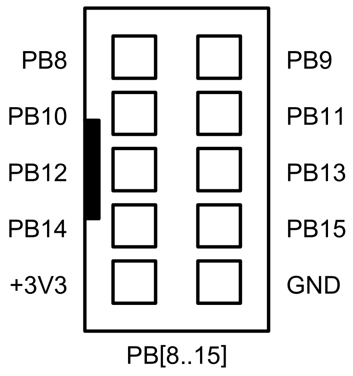


- PB2 has been chosen to be Function BOOT1 to interface with ISP Boot Loader.
- PB3 has been chosen to be Function TDO to interface with JTAG.
- PB4 has been chosen to be Function TRST to interface with JTAG.
- PB6 has been chosen to be Function TXD1 of USART1.
- PB7 has been chosen to be Function RXD1 of USART1.

- Connector IDE 10Pin of PB[8..15] is arranged as follows;

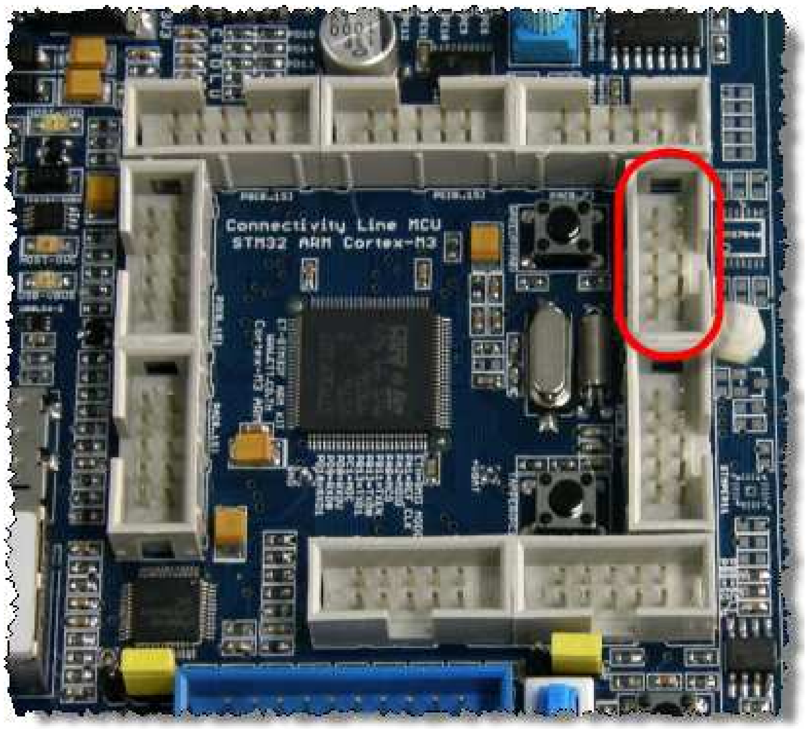


↓

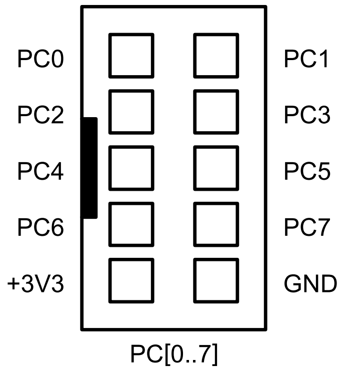


- PB8 has been chosen to be Function SCL of I2C1.
- PB9 has been chosen to be Function SDA of I2C1.
- PB11 has been chosen to be Function TXEN of RMII to interface with Ethernet.
- PB12 has been chosen to be Function TXD0 of RMII to interface with Ethernet.
- PB13 has been chosen to be Function TXD1 of RMII to interface with Ethernet.

- Connector IDE 10Pin of PC[0..7] is arranged as follows;

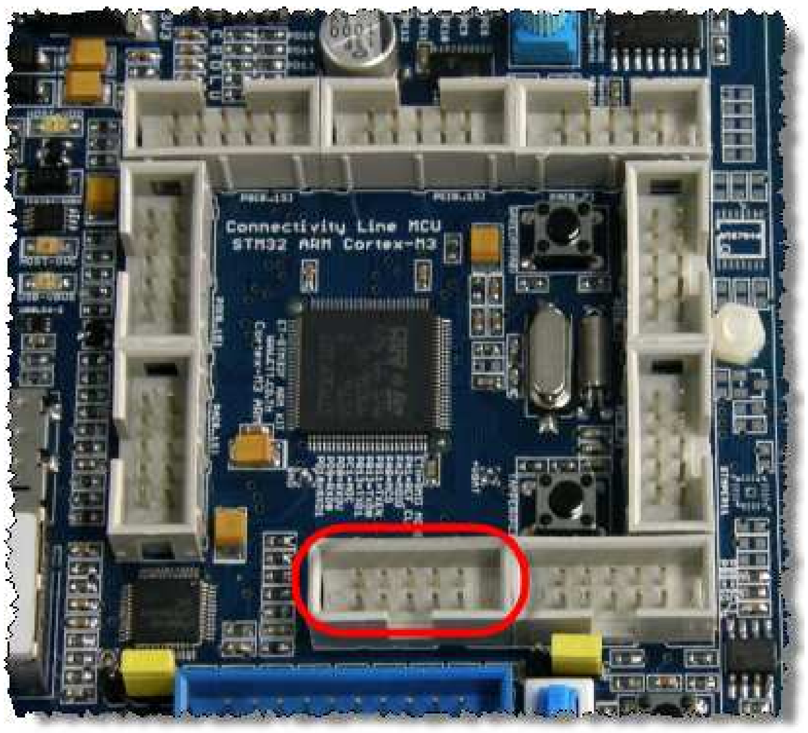


↓

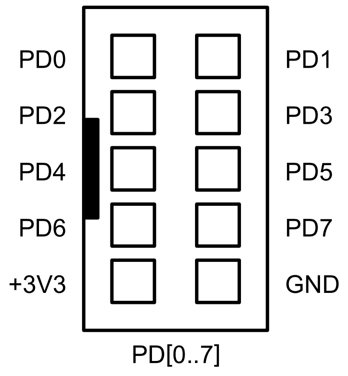


  - PC1 has been chosen to be Function MDC of RMII to interface with Ethernet.
  - PC4 has been chosen to be Function ADC to interface with Volume.
  - PC5 has been chosen to be Function GPIO Output(CS#) to interface with SD Card.

- Connector IDE 10Pin of PD[0..7] is arranged as follows;

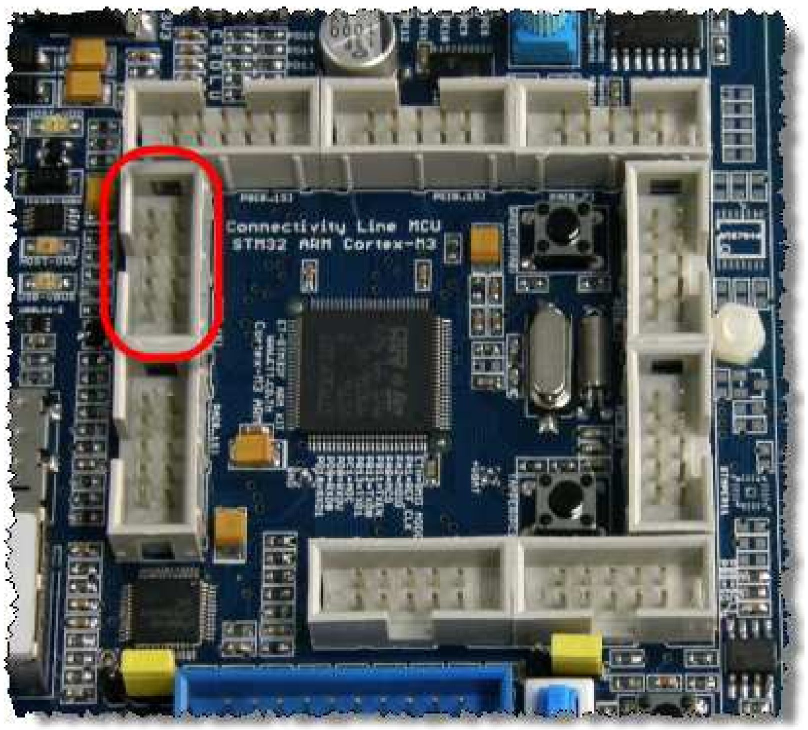


↓

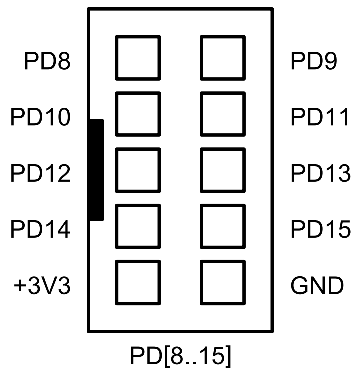


  - PD5 has been chosen to be Function TXD2 of USART2.
  - PD6 has been chosen to be Function RXD2 of USART2.
  - PD7 has been chosen to be Function GPIO Out to control GLCD Backlight LED.

- Connector IDE 10Pin of PD[8..15] is arranged as follows;



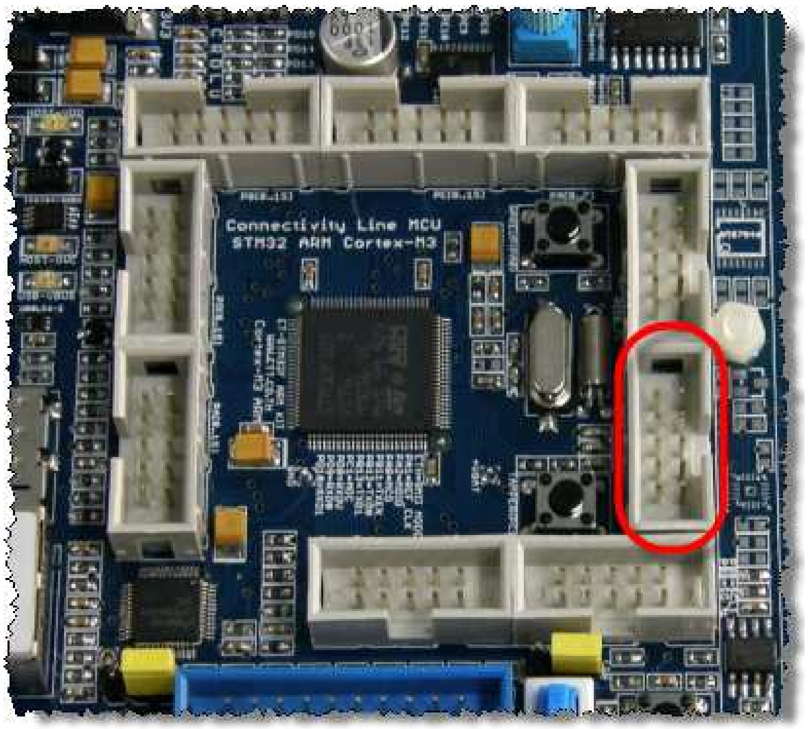
↓



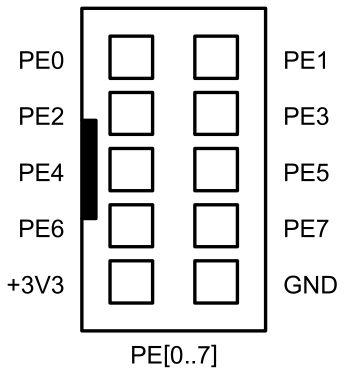
- PD8 has been chosen to be Function CRS_DV of RMII to interface with Ethernet.
- PD9 has been chosen to be Function RXD0 of RMII to interface with Ethernet.
- PD10 has been chosen to be Function RXD1 of RMII to interface with Ethernet.

- PD11 has been chosen to be Function GPIO Input to interface with Joy Switch.
- PD12 has been chosen to be Function GPIO Input to interface with Joy Switch.
- PD13 has been chosen to be Function GPIO Input to interface with Joy Switch.
- PD14 has been chosen to be Function GPIO Input to interface with Joy Switch.
- PD15 has been chosen to be Function GPIO Input to interface with Joy Switch.

- Connector IDE 10Pin of PE[0..7] is arranged as follows;

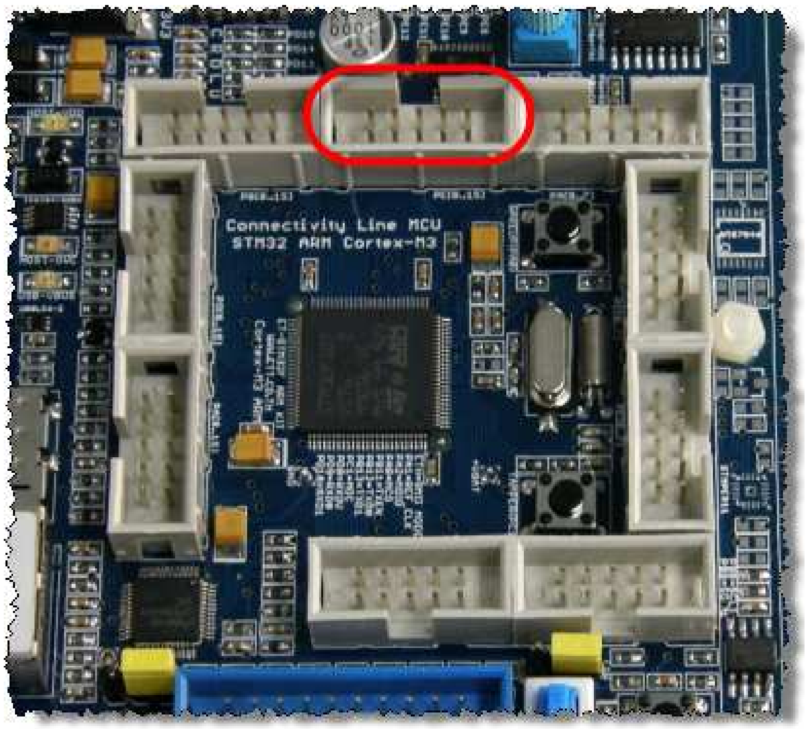


↓

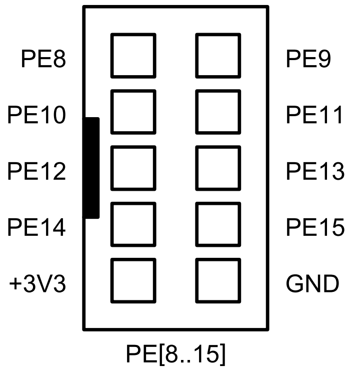


- PE1 has been chosen to be Function GPIO Input(Host OVRCR:USB Host Over Current) to interface with USB Host.
- PE3 has been chosen to be Function GPIO Input(PEN#) to interface with Touch Sensor No.ADS7846 or INT# to interface with Touch Sensor No.STMPE811.
- PE4 has been chosen to be Function GPIO Input(MISO) to interface with ADS7846.
- PE5 has been chosen to be Function GPIO Output(MOSI) to interface with ADS7846.
- PE6 has been chosen to be Function GPIO Output(CS#) to interface with ADS7846.
- PE7 has been chosen to be Function GPIO Output(SCK) to interface with ADS7846.

- Connector IDE 10Pin of PE[8..15] is arranged as follows;



↓



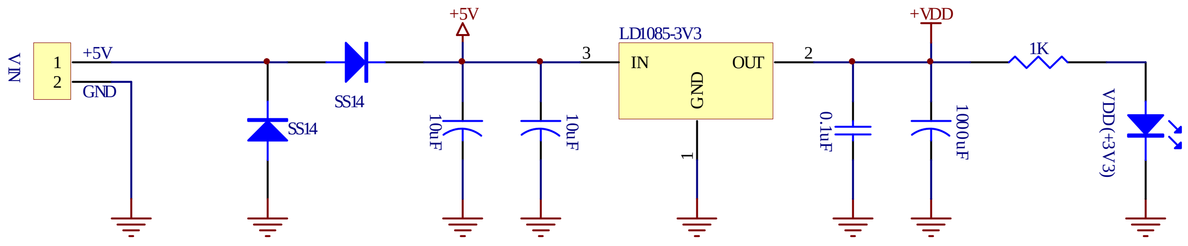
- PE8 has been chosen to drive LED0 to test the operation of Output.
- PE9 has been chosen to drive LED1 to test the operation of Output.
- PE10 has been chosen to drive LED2 to test the operation of Output.

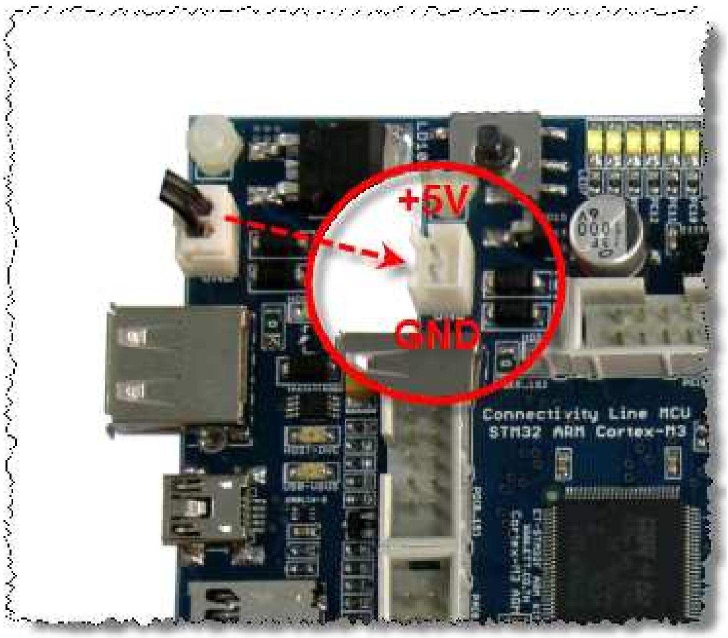
- PE11 has been chosen to drive LED3 to test the operation of Output.
- PE12 has been chosen to drive LED4 to test the operation of Output.
- PE13 has been chosen to drive LED5 to test the operation of Output.
- PE14 has been chosen to drive LED6 to test the operation of Output
- PE15 has been chosen to drive LED7 to test the operation of Output.

Circuit Power Supply

The Circuit Power Supply of board uses +5VDC and it uses Connector 2Pin Block to prevent user from interfacing connector wrongly. Moreover, it includes Circuit Regulate +3V3/3A.

The Circuit Power Supply in the part of Circuit Regulate 3.3V supplies power to CPU and all Circuit I/O of board, except Backlight of LCD because it directly uses +5VDC Power Supply instead.




+5V
GND
Connectivity Line MCU
STM32 ARM Cortex-M3

→

Summarize GPIO allocation for Board ET-STM32F ARM KIT

Normally, STM32F107VCT6 has 5 Ports of Pin GPIO in total; PA, PB, PC, PD, and PE. Each port has 16Bit or 16Pin; so, there are 80Pin in total. For Board ET-STM32F ARM KIT, it has allocated 60Pin for GPIO Pins and available 20Pin for user to be free to use them as preferred.

Sometimes, user wants to apply or modify Board ET-STM32F ARM KIT without using devices or circuits that are designed and provided on board, or user requires using some devices only; so, user can choose pins that are interfaced with the unwanted devices and then modify them as preferred by self. Details of GPIO allocation and function are described below;

Pin	Function	Devices
PA0	Wakeup	Switch Wakeup
PA1	RMII_REF_CLK	Ethernet LAN
PA2	RMII_MDIO	Ethernet LAN
PA3	-	-
PA4	-	-
PA5	SPI1_SCK	SD Card CLK
PA6	SPI1_MISO	SD Card DAT0
PA7	SPI1_MOSI	SD Card CMD

Pin	Function	Devices
PA8	MCO	Ethernet LAN
PA9	FS_VBUS	USB OTG/Device
PA10	FS_ID	USB OTG
PA11	FS_DM	USB Data
PA12	FS_DP	HOST/OTG/Device
PA13	JTAG_TMS	JTAG
PA14	JTAG_TCLK	JTAG
PA15	JTAG_TDI	JTAG

Pin	Function	Devices
PB0	-	-
PB1	-	-
PB2	BOOT1	Jumper BOOT1
PB3	JTAG_TDO	JTAG
PB4	JTAG_TRST	JTAG

Pin	Function	Devices
PB8	I2C1_SCL	24C01,STMPE811
PB9	I2C1_SDA	24C01,STMPE811
PB10	-	-
PB11	RMII_TXEN	Ethernet LAN
PB12	RMII_TXD0	Ethernet LAN

PB5	-	-	PB13	RMII_TXD1	Ethernet LAN
PB6	USART1_TX	UART1	PB14	-	-
PB7	USART1_RX	UART1	PB15	-	-

Pin	Function	Devices	Pin	Function	Devices
PC0	-	-	PC8	GPIO Out	GLCD CS#
PC1	RMII_MDC	Ethernet LAN	PC9	HOST_EN	USB HOST/OTG
PC2	-	-	PC10	SPI3_SCK	GLCD WR#/SCL
PC3	-	-	PC11	SPI3_MISO	GLCD SDO
PC4	ADC14	Volume VR1	PC12	SPI3_MOSI	GLCD SDI
PC5	GPIO Out	SD Card CD(CS#)	PC13	Tamper	Switch Tamper
PC6	-	-	PC14	OSC32_IN	RTC X-TAL
PC7	-	-	PC15	OSC32_OUT	RTC X-TAL

Pin	Function	Devices	Pin	Function	Devices
PD0	-	-	PD8	RMII_CRS_DV	Ethernet LAN
PD1	-	-	PD9	RMII_RXD0	Ethernet LAN
PD2	-	-	PD10	RMII_RXD1	Ethernet LAN
PD3	-	-	PD11	GPIO Input	Joy Switch Up
PD4	-	-	PD12	GPIO Input	Joy Switch Left
PD5	USART2_TX	UART2(ISP)	PD13	GPIO Input	Joy Switch Down
PD6	USART2_RX	UART2(ISP)	PD14	GPIO Input	Joy Switch Right
PD7	GPIO Out	GLCD BL LED	PD15	GPIO Input	Joy Switch Select

Pin	Function	Devices	Pin	Function	Devices
PE0	-	-	PE8	GPIO Out	LED0
PE1	USB_OVRCR	USB HOST/OTG	PE9	GPIO Out	LED1

PE2	-	-
PE3	GPIO Input	ADS7846 PEN#
PE4	GPIO Input	ADS7846 DOUT
PE5	GPIO Out	ADS7846 DIN
PE6	GPIO Out	ADS7846 CS#
PE7	GPIO Out	ADS7846 DCLK

PE10	GPIO Out	LED2
PE11	GPIO Out	LED3
PE12	GPIO Out	LED4
PE13	GPIO Out	LED5
PE14	GPIO Out	LED6
PE15	GPIO Out	LED7

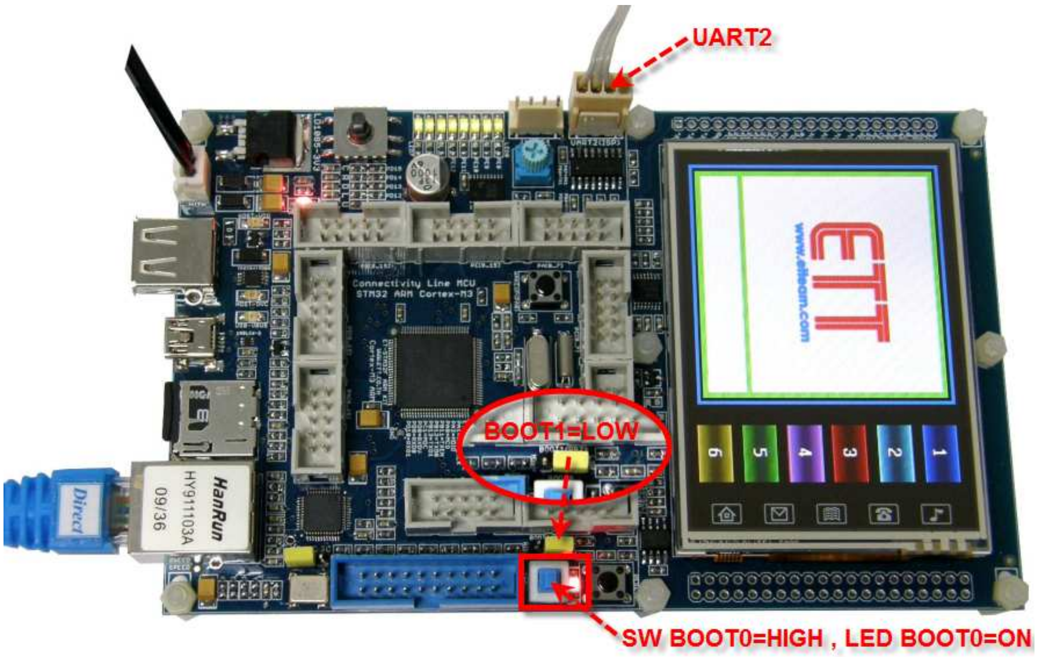
How to Download HEX File into MCU of Board

The method to download HEX File into Flash Memory of MCU on board is to use Program Flash Loader of "ST Microelectronics". It is connected with MCU through Serial Port of computer PC; moreover, user can download this program free from website: http://ww.st.com

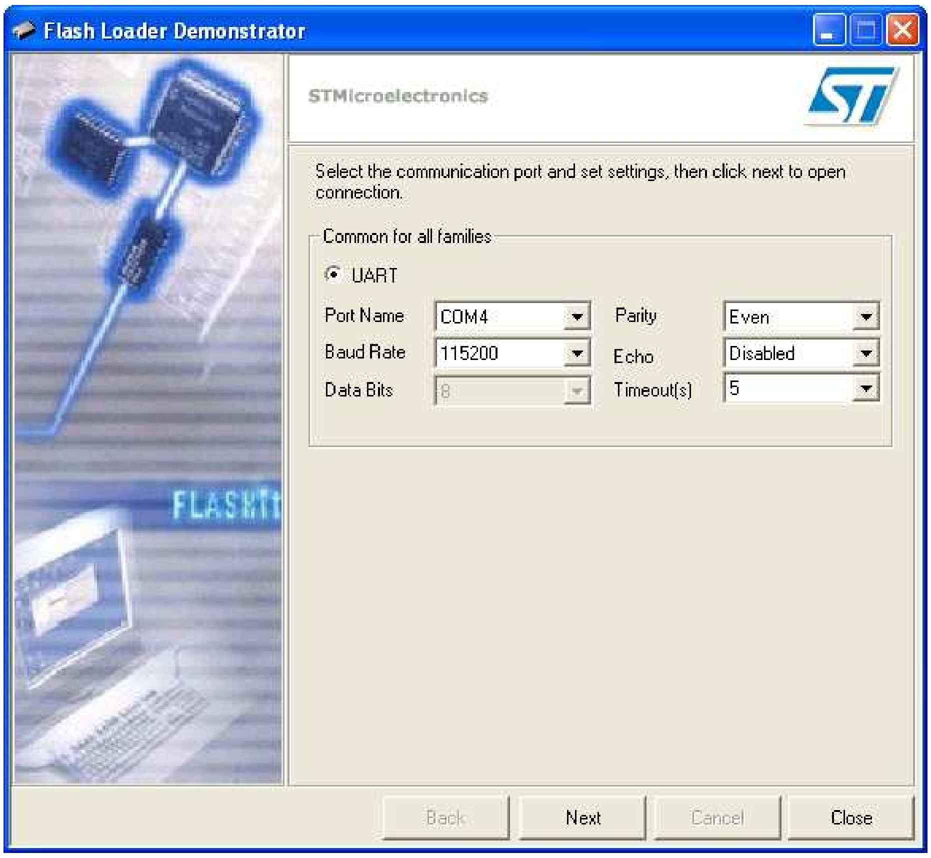
Procedures of downloading HEX File into MCU

1. Interface RS232 Cable between RS232 Serial Port Communication of PC and Board UART2.
2. Supply power into board; in this case, it makes LED PWR be status ON.
3. Set Jumper BOOT1 to the LOW position.
4. Press Switch BOOT0 to be High; it makes LED BOOT0 be status ON.

↓



5. Run Program Flash Loader; if it is Version 2.10, its result will be displayed as below;



Select the communication port and set settings, then click next to open connection.

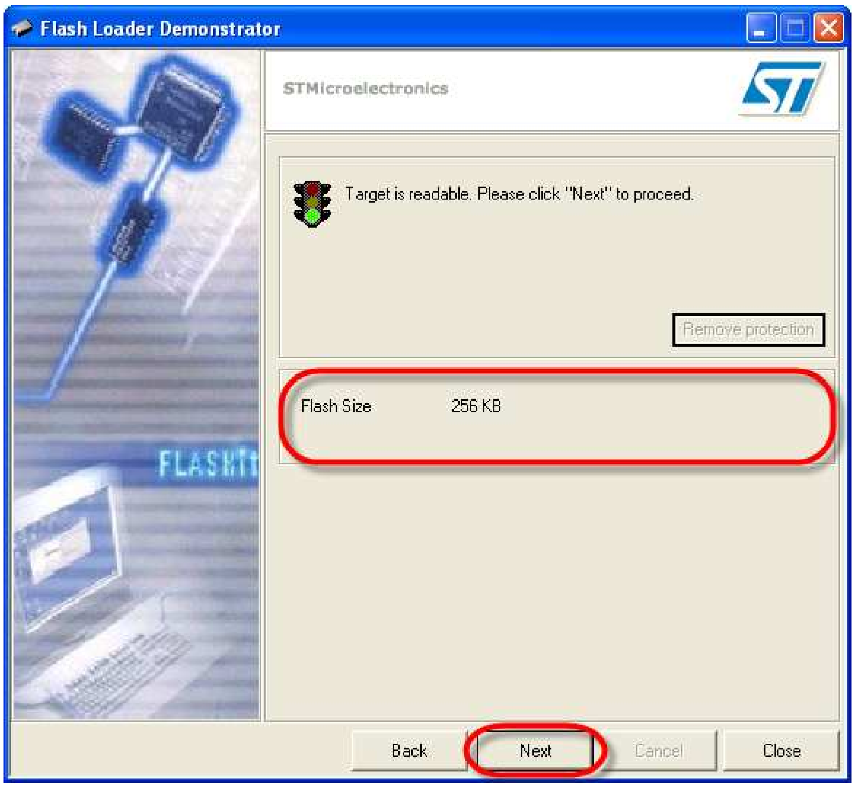
Common for all families

UART

Port Name	COM4	Parity	Even
Baud Rate	115200	Echo	Disabled
Data Bits	8	Timeout(s)	5

6. Firstly, set values for program as required; if it is STM32F107VCT6 of Board ET-STM32F ARM KIT from ETT, it needs to set values for program as follows;
    - Choose COM Port number according to the real COM Port number (this example, it is COM4)
    - Set Baud Rate as 115200
    - Set Parity as Even
    - Set Echo as Disable
    - Set Timeout as 5 seconds
    - Press Switch RESET on Board “ET-STM32F ARM KIT” to reset MCU to operate in Boot Loader, it needs to check conditions as below;
        - Set Jumper BOOT1 = Low
        - Set Switch BOOT0 to the HIGH position (LED BOOT0 is ON)
        - Interface RS232 Cable with UART2(ISP) of board completely
        - Press Switch RESET

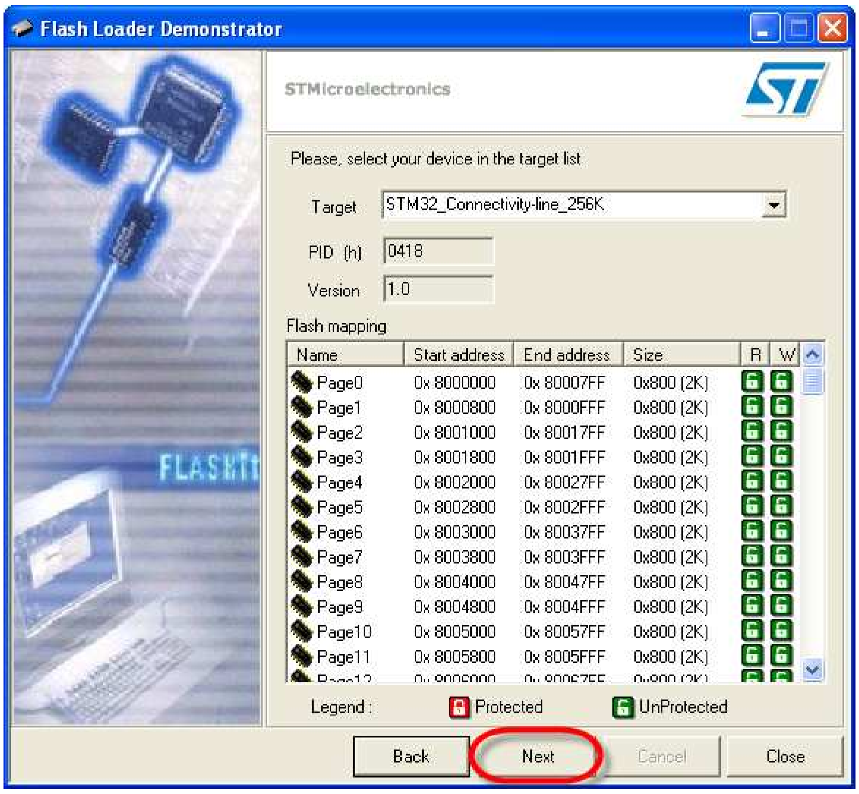
7. If everything is correct, its operating result will be displayed as picture below and then click Next to go to the next step. However, if there is any error, please check the condition in the step 6 and then press Switch RESET again.



Target is readable. Please click "Next" to proceed.

Flash Size 256 KB

↓



Please, select your device in the target list

Target STM32_Connectivity-line_256K

PID (h) 0418

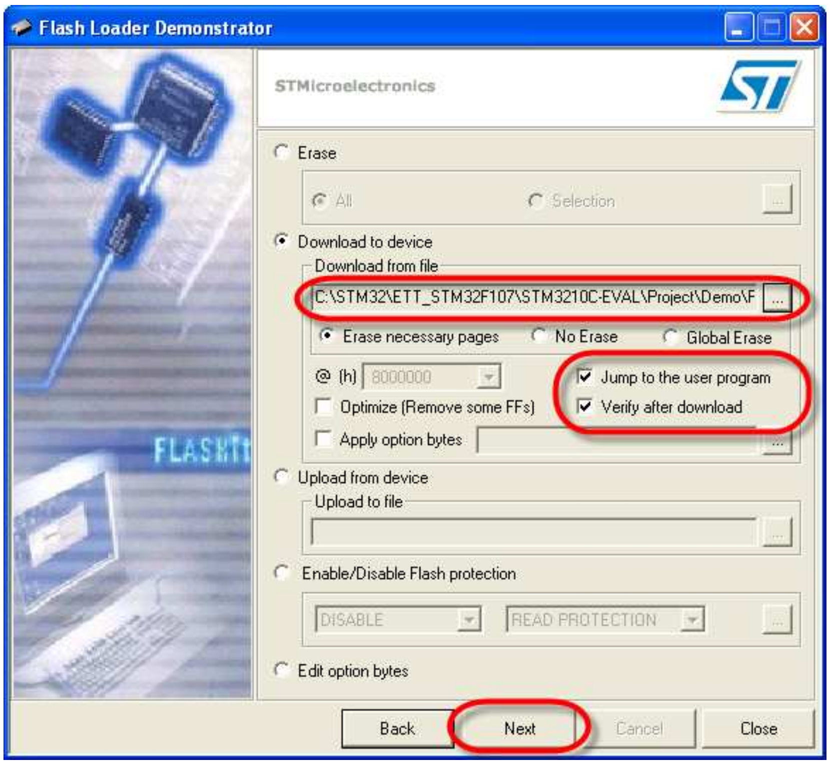
Version 1.0

Flash mapping

Name	Start address	End address	Size	R	W
Page0	0x 8000000	0x 80007FF	0x800 (2K)	🟩	🟩
Page1	0x 8000800	0x 8000FFF	0x800 (2K)	🟩	🟩
Page2	0x 8001000	0x 80017FF	0x800 (2K)	🟩	🟩
Page3	0x 8001800	0x 8001FFF	0x800 (2K)	🟩	🟩
Page4	0x 8002000	0x 80027FF	0x800 (2K)	🟩	🟩
Page5	0x 8002800	0x 8002FFF	0x800 (2K)	🟩	🟩
Page6	0x 8003000	0x 80037FF	0x800 (2K)	🟩	🟩
Page7	0x 8003800	0x 8003FFF	0x800 (2K)	🟩	🟩
Page8	0x 8004000	0x 80047FF	0x800 (2K)	🟩	🟩
Page9	0x 8004800	0x 8004FFF	0x800 (2K)	🟩	🟩
Page10	0x 8005000	0x 80057FF	0x800 (2K)	🟩	🟩
Page11	0x 8005800	0x 8005FFF	0x800 (2K)	🟩	🟩

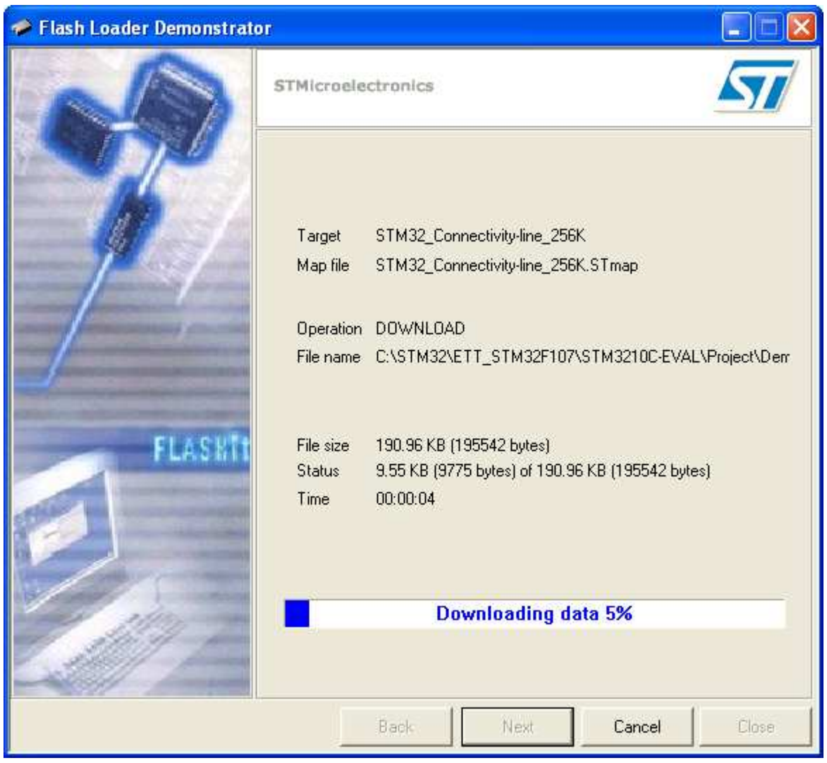
Legend : 🟥 Protected 🟩 UnProtected

8. Choose the HEX File that user requires to download into board and then click Next as shown below;



Erase
All
Selection
Download to device
Download from file
C:\STM32\ETT_STM32F107\STM3210C-EVAL\Project\Demo\F
Erase necessary pages
No Erase
Global Erase
@ (h) 8000000
[✓] Jump to the user program
Optimize (Remove some FFs)
[✓] Verify after download
Apply option bytes
Upload from device
Upload to file
Enable/Disable Flash protection
DISABLE
READ PROTECTION
Edit option bytes

9. Program Flash Loader starts downloading data into MCU instantly and user can see its operation from the Status Bar. In this case, user needs to wait for a while until the program runs successfully.

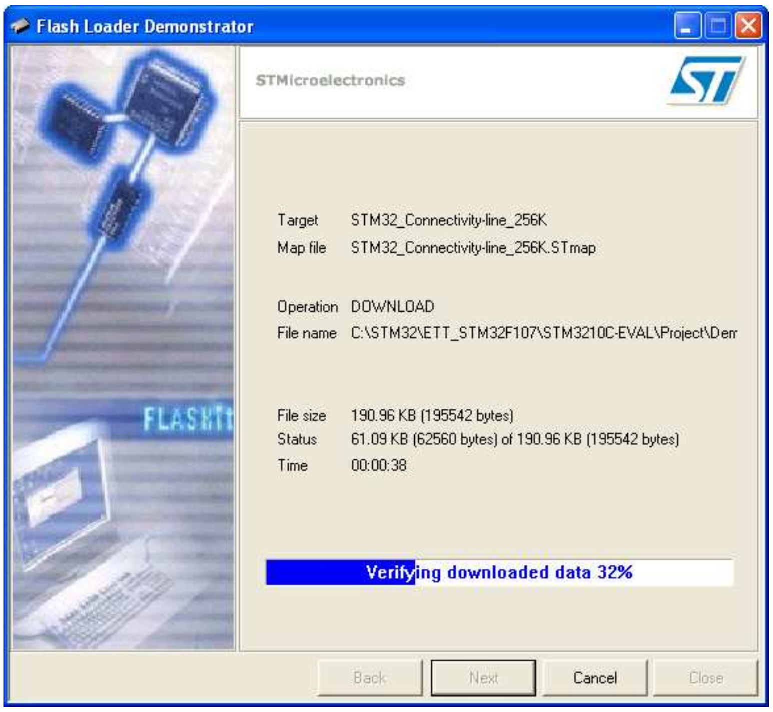


Target STM32_Connectivity-line_256K
Map file STM32_Connectivity-line_256K.STmap

Operation DOWNLOAD
File name C:\STM32\ETT_STM32F107\STM3210C-EVAL\Project\Dem

File size 190.96 KB (195542 bytes)
Status 9.55 KB (9775 bytes) of 190.96 KB (195542 bytes)
Time 00:00:04

Downloading data 5%



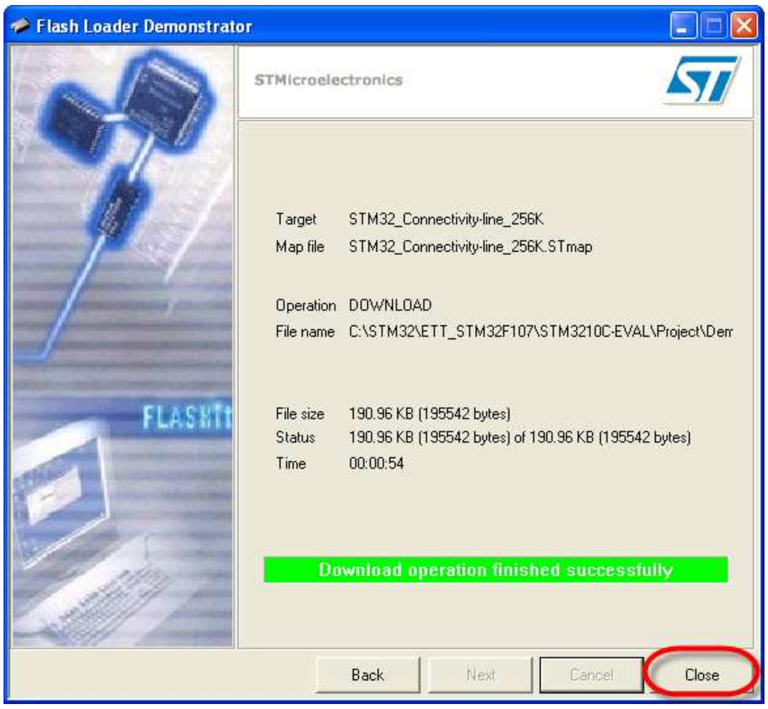
Target STM32_Connectivity-line_256K
Map file STM32_Connectivity-line_256K.STmap

Operation DOWNLOAD
File name C:\STM32\ETT_STM32F107\STM3210C-EVAL\Project\Dem

File size 190.96 KB (195542 bytes)
Status 61.09 KB (62560 bytes) of 190.96 KB (195542 bytes)
Time 00:00:38

Verifying downloaded data 32%

↓



Target STM32_Connectivity-line_256K
Map file STM32_Connectivity-line_256K.STmap

Operation DOWNLOAD
File name C:\STM32\ETT_STM32F107\STM3210C-EVAL\Project\Dem

File size 190.96 KB (195542 bytes)
Status 190.96 KB (195542 bytes) of 190.96 KB (195542 bytes)
Time 00:00:54

Download operation finished successfully

When the program has already run successfully, set Switch BOOT0 to the LOW position; in this case, it makes LED BOOT0 be status OFF. Next, press Switch RESET on board and it makes MCU start running according to the downloaded program instantly.