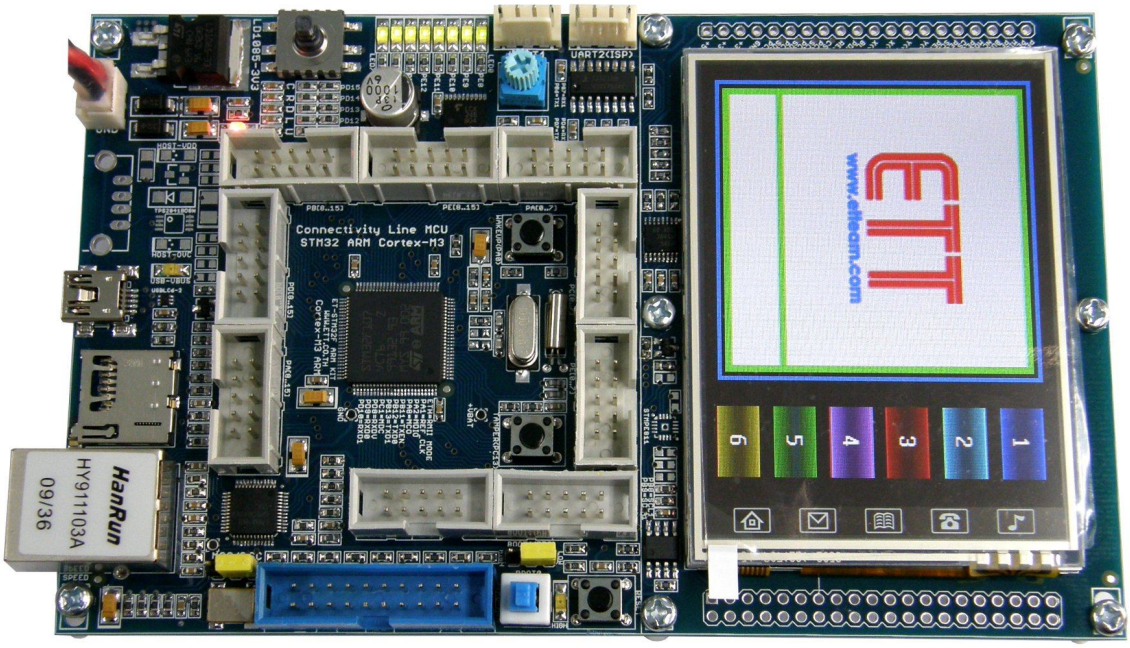
ET-STM32F ARM KIT (STM32F107VCT6)

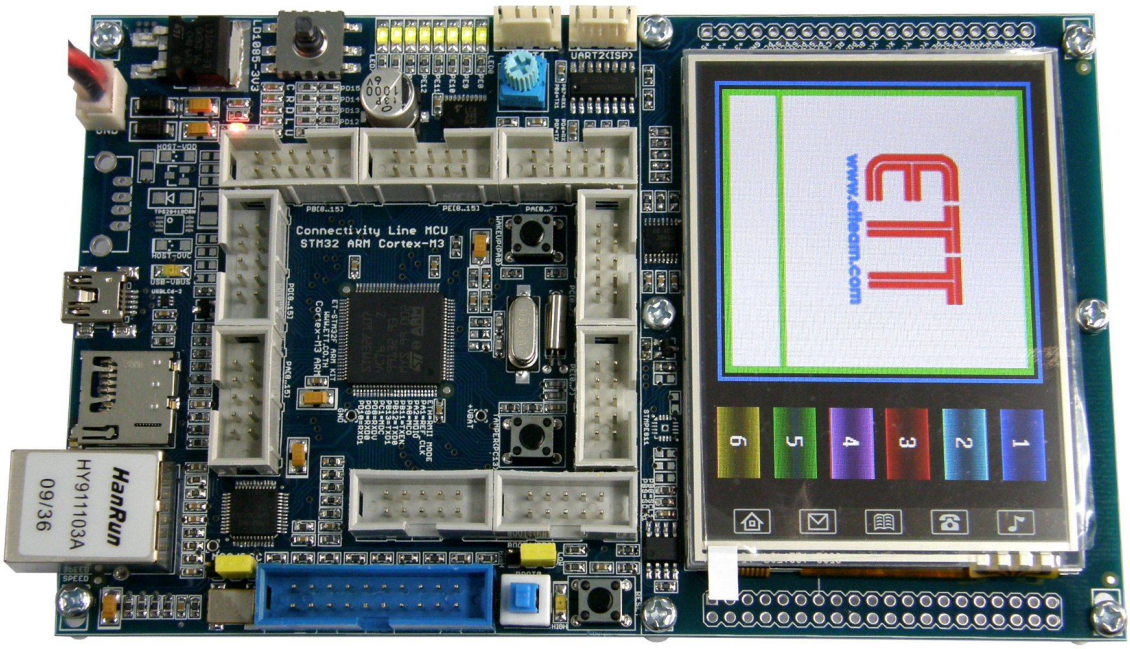


ET-STM32F ARM KIT เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล ARM Cortex M3 Core ซึ่งเลือกใช้ไมโครคอนโทรเลอร์แบบ 32-Bit ขนาด 100 Pin (100-LQFP) เบอร์ STM32F107VCT6 ของบริษัท ST (ST Microelectronics) ซึ่งทาง ST ได้ปรับปรุงพัฒนา MCU ตระกูลนี้ต่อเนื่องมาจากตระกูล STM32F103 ให้มีขีดความสามารถที่สูงกว่าเดิม โดยได้มีการเพิ่มระบบทรัพยากรสำหรับเชื่อมต่อกับ Ethernet LAN และ USB ซึ่งเพิ่มฟังก์ชั่นความสามารถให้สามารถทำงานเป็นแบบ Host/OTG ได้ด้วย โดยความสามารถอื่นๆยังมีอยู่เช่นเดิม ไม่ว่าจะเป็น SPI, I2C, CAN, ADC, DAC, Timer/Counter, PWM, Capture, UART,..ฯลฯ

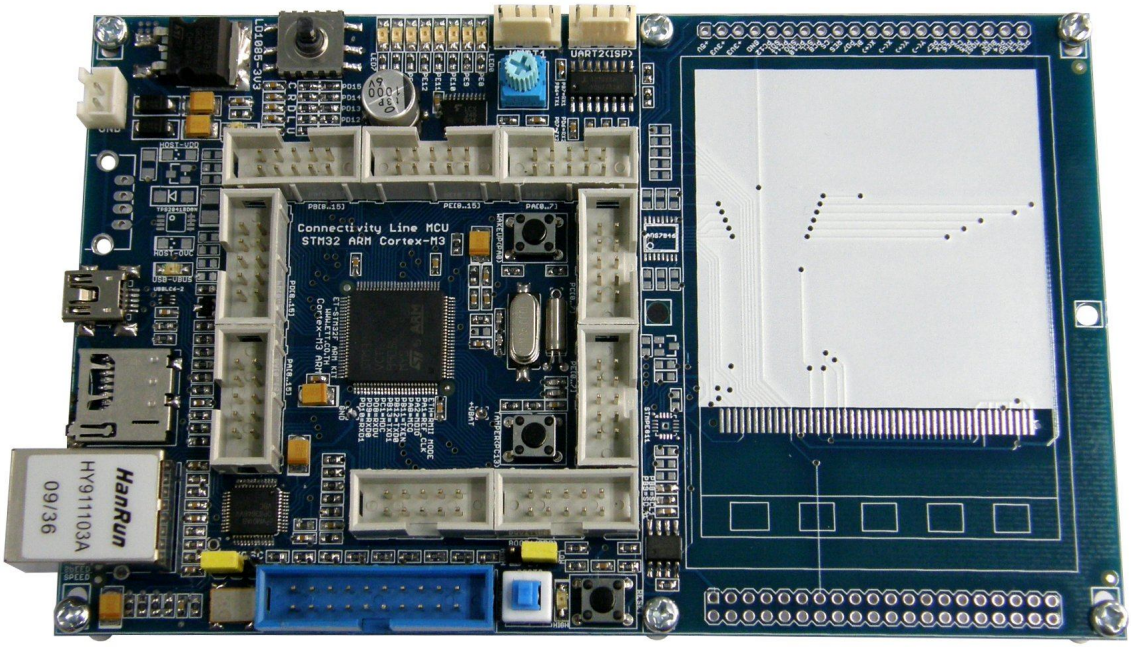
โดยระบบฮาร์ดแวร์ของบอร์ดที่ทางทีมงาน อีทีที ได้ออกแบบไว้นั้น จะรองรับทั้ง กลุ่มผู้ใช้ที่ต้องการศึกษา เรียนรู้ ทดลอง รวมไปถึงการนำไปดัดแปลง ประยุกต์ใช้งานจริงๆได้ด้วย โดยโครงสร้างของบอร์ดนั้นจะประกอบไปด้วยอุปกรณ์พื้นฐานที่จำเป็นต่อการ ศึกษาทดลอง ขั้นพื้นฐาน เช่น LED สำหรับแสดงค่า Output Logic, Push Button Switch และ Joy Switch สำหรับทดสอบ Logic Input, Volume ปรับค่าแรงดัน สำหรับทดสอบ A/D นอกจากนี้แล้วยังมีการจัดเตรียมอุปกรณ์ระดับสูงไว้รองรับการใช้งานด้วยไม่ว่าจะเป็น พอร์ต เชื่อมต่อ USB Device/Host/OTG,SD Card, พอร์ตเชื่อมต่อ Ethernet LAN, Graphic LCD, RS232 นอกเหนือจากนี้แล้วยังมี GPIO ต่างๆที่ว่างไว้ให้ผู้ใช้ออกแบบใช้งานร่วมกับอุปกรณ์อื่นๆได้เองตามความเหมาะสมอีกด้วย

คุณสมบัติของบอร์ด ET-STM32F ARM KIT (STM32F107VCT6)

1. ใช้ MCU ตระกูล ARM Cortex M3 เบอร์ STM32F107VCT6 ของ ST ซึ่งเป็น MCU ขนาด 32Bit
2. ภายใน MCU มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 256KB, Static RAM ขนาด 64KB
3. ใช้ Crystal 25.00 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 72 MHz เมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง
4. มีวงจร RTC(Real Time Clock) พร้อม XTAL ค่า 32.768KHz และ Battery Backup
5. รองรับการโปรแกรมแบบ In-System Programming(ISP) ผ่าน USART2 Boot-Loader(RS232)
6. มีวงจรเชื่อมต่อกับ JTAG ARM ขนาด 20 Pin มาตรฐาน เพื่อทำการ Debug แบบ Real Time ได้
7. Power Supply ใช้แรงดันไฟฟ้า +5VDC
8. มีวงจร USB 2.0 แบบ Full Speed รองรับการทำงานแบบ Device/Host/OTG ภายในตัว
9. มีวงจร Over Current Protection สำหรับ USB Host/OTG(Option)
10. มีวงจรเชื่อมต่อ Ethernet LAN 10/100Mb โดยใช้ขั้วต่อแบบ RJ45 มาตรฐาน จำนวน 1 ช่อง
11. มีวงจรเชื่อมต่อการ์ดหน่วยความจำแบบ SD Card(Micro SD) เชื่อมต่อแบบ SPI จำนวน 1 ช่อง
12. มีวงจรสื่อสาร RS232 โดยใช้ขั้วต่อแบบ 4-PIN มาตรฐาน ETT จำนวน 2 ช่อง
13. มีวงจรเชื่อมต่อ TFT LCD Color ขนาด 320x240 Pixel (3.2นิ้ว) พร้อม Touch Sensor
14. มีวงจร Push Button Switch จำนวน 2 ชุด พร้อมสวิตช์ RESET
15. มีวงจร Joy Switch แบบ 5 ทิศทาง สำหรับใช้งาน จำนวน 1 ชุด
16. มีวงจร LED แสดงสถานะเพื่อทดลอง Output จำนวน 8 ชุด พร้อมวงจร Buffer
17. มีวงจร สร้างแรงดัน 0-3V3 โดยใช้ตัวต้านทานปรับค่าได้สำหรับทดสอบ A/D จำนวน 1 ชุด
18. มี 80 Bit GPIO ใช้งานได้อิสระ 72 Bit GPIO โดยใช้ขั้วต่อแบบ 10Pin IDE จำนวน 9 ชุด สามารถเลือกใช้งานเป็น 72 Bit GPIO หรือ หรือใช้งานเป็นฟังก์ชั่นอื่นๆเช่น A/D,D/A,I2C,CAN,Ethernet
    a. 10 Bit สำหรับ Ethernet LAN (DP83848V RMII Interface Mode)
    b. 2 Bit สำหรับ USART1 และ 2 Bit สำหรับ USART2
    c. 1 Bit สำหรับ Volume ปรับแรงดันทดสอบ ADC14(PC4) และ 8 Bit สำหรับ LED
    d. 4 Bit สำหรับ SD Card Interface
    e. 6 Bit สำหรับ USB Device/Host/OTG Interface
    f. 10 Bit สำหรับ TFTLCD320x240 และ Touch Sensor(ADS7846)
    g. 5 Bit สำหรับ Joy Switch 5 Direction และ 2 Bit สำหรับ Push Button SW
    h. 5 Bit สำหรับ JTAG ARM Interface
    i. 2 Bit สำหรับ I2C Interface

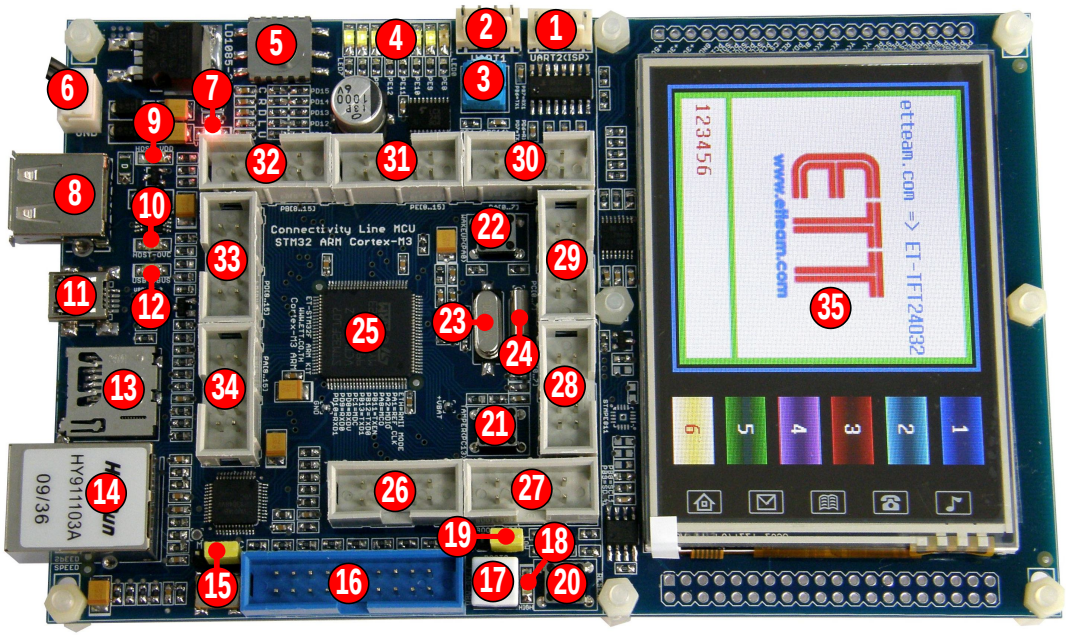


รูปแสดง โครงสร้างของบอร์ด **ET-STM32F ARM KIT & TFT LCD**



รูปแสดง โครงสร้างของบอร์ด **ET-STM32F ARM KIT**

โครงสร้างบอร์ด ET-STM32F ARM KIT



รูปแสดง ตำแหน่งของอุปกรณ์ต่างๆในบอร์ด ET-STM32F ARM KIT

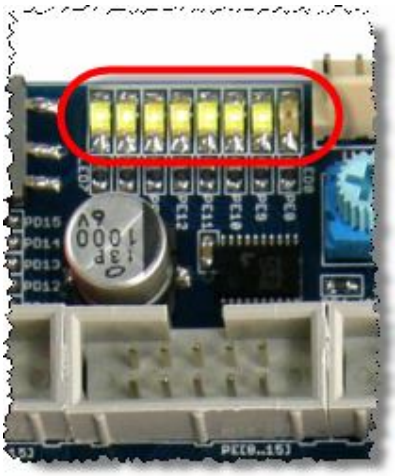
(***หมายเหตุ อุปกรณ์ USB Host/OTG เป็น Option ซึ่งจะไม่ได้ติดตั้งมาด้วยในบอร์ดมาตรฐาน***)

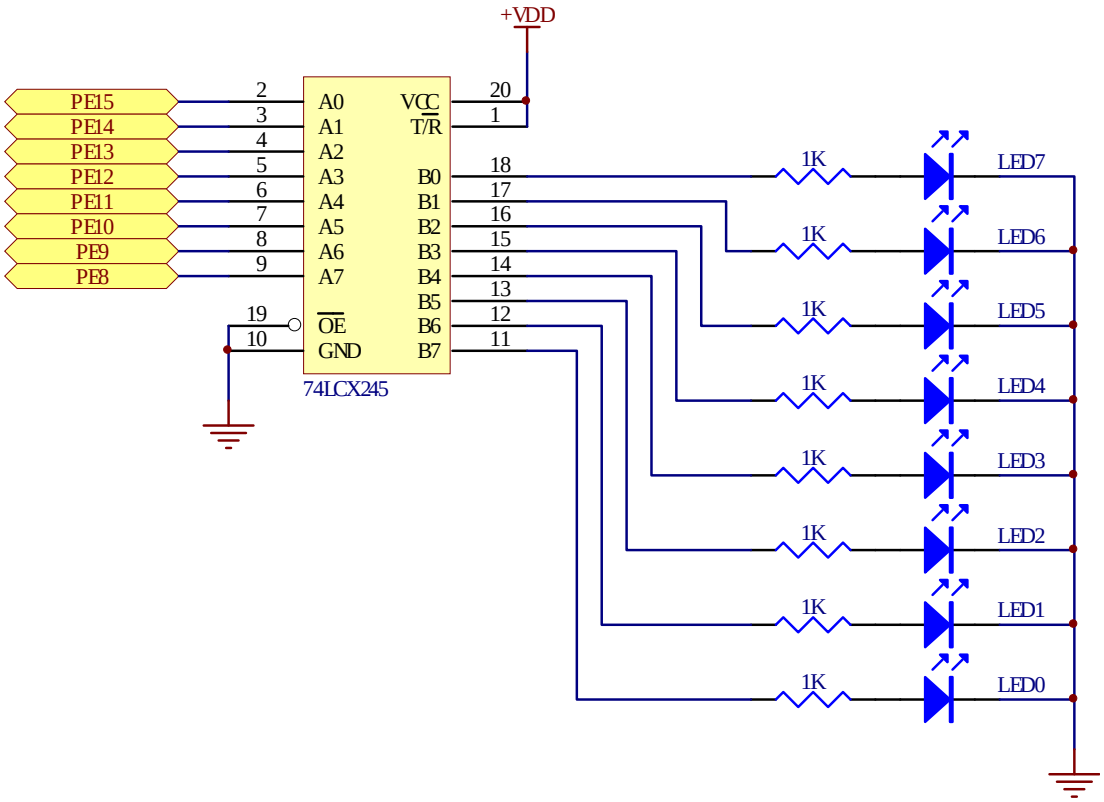
- **หมายเลข 1 คือ** ขั้วต่อ UART2(RS232) สำหรับใช้งาน และ Download Hex File ให้ CPU
- **หมายเลข 2 คือ** ขั้วต่อ UART1(RS232) สำหรับใช้งาน
- **หมายเลข 3 คือ** VR สำหรับปรับค่าแรงดัน 0-3V3 สำหรับทดสอบ A/D(PC4/ADC14)
- **หมายเลข 4 คือ** LED[0..7] ใช้ทดสอบ Logic Output ของ PE[8..15]
- **หมายเลข 5 คือ** Joy Switch แบบ 5 ทิศทาง
- **หมายเลข 6 คือ** ขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ดใช้ได้กับไฟ +5VDC
- **หมายเลข 7 คือ** LED แสดงสถานะของ Power +VDD(+3V3)
- **หมายเลข 8 คือ** ขั้วต่อ USB Host(Option)
- **หมายเลข 9 คือ** LED แสดงสถานะของ Host VDD(Option)
- **หมายเลข 10 คือ** LED แสดงสถานะของ Host Over Current(Option)
- **หมายเลข 11 คือ** ขั้วต่อ USB Device/OTG
- **หมายเลข 12 คือ** LED แสดงสถานะของ USB VBUS
- **หมายเลข 13 คือ** ช่องเสียบการ์ดหน่วยความจำสามารถใช้ได้กับ SD Card แบบ Micro-SD
- **หมายเลข 14 คือ** ขั้วต่อสัญญาณ Ethernet LAN แบบ RJ45

- **หมายเลข 15 คือ** Jumper(MCO/OSC) ใช้สำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกาที่จะป้อนให้กับ DP83848V
- **หมายเลข 16 คือ** ขั้วต่อ JTAG ARM สำหรับ Debug แบบ Real Time
- **หมายเลข 17 คือ** SW BOOT0 ใช้ร่วมกับ Jumper BOOT1 เพื่อเลือกโหมดการทำงานของ MCU ระหว่าง Boot Loader (BOOT0=1,BOOT1=0) กับ Run (BOOT0=0,BOOT1=0)
- **หมายเลข 18 คือ** LED แสดงสถานะทางโลจิกของ BOOT0 = 1(ON=Boot Loader, OFF=Run)
- **หมายเลข 19 คือ** Jumper BOOT1(PB2) ปรกติต้องกำหนดไว้เป็น Low เสมอ
- **หมายเลข 20 คือ** SW Reset
- **หมายเลข 21 คือ** SW Tamper(PC13)
- **หมายเลข 22 คือ** SW Wakeup(PA0)
- **หมายเลข 23 คือ** Crystal ค่า 25 MHz สำหรับใช้เป็นฐานเวลาระบบให้ MCU
- **หมายเลข 24 คือ** Crystal ค่า 32.768KHz สำหรับฐานเวลาให้ RTC ภายในตัว MCU
- **หมายเลข 25 คือ** MCU เบอร์ STM32F107VCT6 (100Pin LQFP)
- **หมายเลข 26 คือ** ขั้วต่อ GPIO PD[0..7]
- **หมายเลข 27 คือ** ขั้วต่อ GPIO PB[0..7]
- **หมายเลข 28 คือ** ขั้วต่อ GPIO PE[0..7]
- **หมายเลข 29 คือ** ขั้วต่อ GPIO PC[0..7]
- **หมายเลข 30 คือ** ขั้วต่อ GPIO PA[0..7]
- **หมายเลข 31 คือ** ขั้วต่อ GPIO PE[8..15]
- **หมายเลข 32 คือ** ขั้วต่อ GPIO PB[8..15]
- **หมายเลข 33 คือ** ขั้วต่อ GPIO PD[8..15]
- **หมายเลข 34 คือ** ขั้วต่อ GPIO PA[8..15]
- **หมายเลข 35 คือ** TFT LCD ขนาด 320x240 Dot พร้อม Touch Screen Sensor

การใช้งานวงจรขับ LED แสดงผล

LED แสดงผลของบอร์ด จะต่อวงจรแบบขับกระแส (Source Current) โดยใช้กับแหล่งจ่าย +3.3V ทำงานด้วยโลจิก “1” (+3V3) และหยุดทำงานด้วยโลจิก “0” (0V) โดยควบคุมการทำงานจาก GPIO มีทั้งหมด 8 ชุด คือ PE[8..15] โดยวงจรในส่วนนี้จะใช้สำหรับทดสอบการทำงานของ Output





โดยเมื่อต้องการใช้งานผู้ใช้ต้องกำหนดให้ PE[8..15] ทำหน้าที่เป็น GPIO Output Port เสียก่อน แล้วจึงควบคุม Logic ให้กับ PE[8..15] ตามต้องการ ดังตัวอย่าง

```
// 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;
```

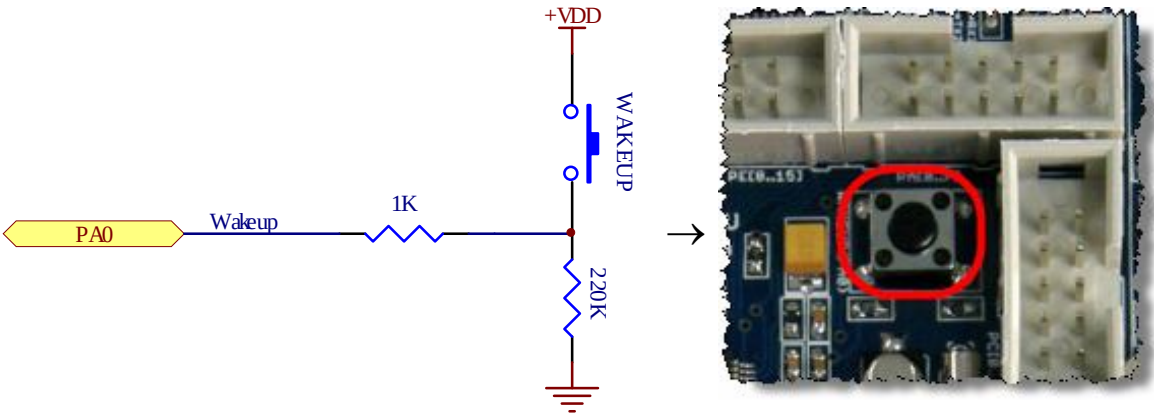
```
GPIO_TypeDef* GPIO_PORT[LEDn] = {LED0_GPIO_PORT,
                                 LED1_GPIO_PORT,
                                 LED2_GPIO_PORT,
                                 LED3_GPIO_PORT,
                                 LED4_GPIO_PORT,
                                 LED5_GPIO_PORT,
                                 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
```

ตัวอย่าง การกำหนดค่าการใช้งาน **PE[8..15]** เป็น **Output LED**

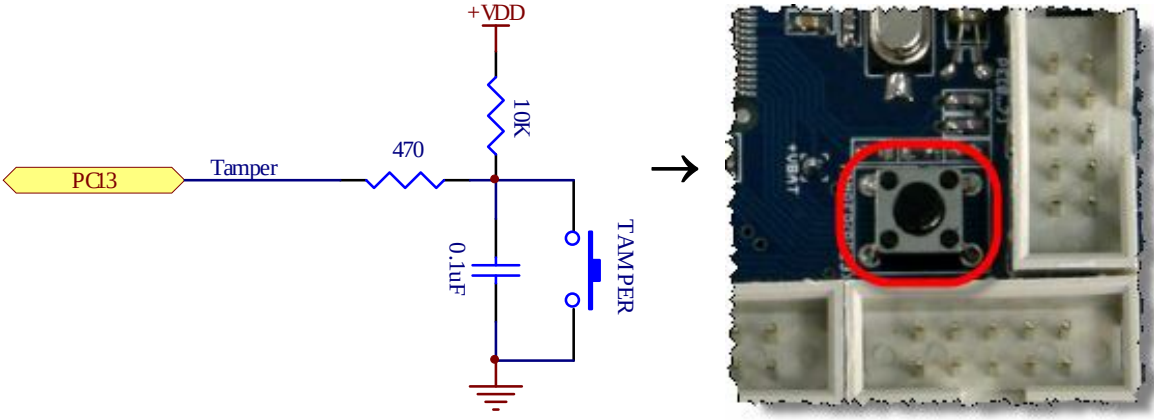
การใช้งานวงจร Push Button Switch

วงจร Push Button Switch จะใช้วงจร Switch แบบ กดติด-ปล่อยดับ (Push Button) โดยมีด้วยกัน 2 ชุด คือ สวิตช์ Wakeup และ สวิตช์ Tamper ซึ่งการทำงานของวงจร Push Button ทั้ง 2 ชุด จะมีความแตกต่างกัน กล่าวคือ จะให้ผลการทำงานเป็นโลจิกที่ตรงกันข้าม

Switch Wakeup(PA0) เมื่อกดจะให้สถานะที่ขา PA0 เป็นโลจิก HIGH เมื่อปล่อยจะให้สถานะที่ขา PA0 เป็นโลจิก LOW



Switch Tamper (PC13) เมื่อกดจะให้สถานะที่ขา PC13 เป็นโลจิก LOW เมื่อปล่อยจะให้สถานะที่ขา PC13 เป็นโลจิก HIGH



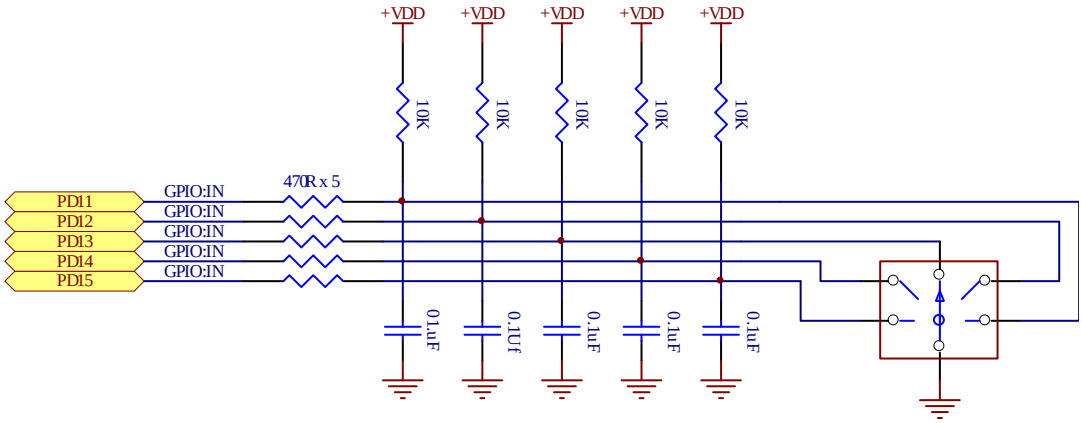
```
// Switch Wakeup(PA0)
#define WAKEUP_BUTTON_PORT          GPIOA
#define WAKEUP_BUTTON_CLK           RCC_APB2Periph_GPIOA
#define WAKEUP_BUTTON_PORT_SOURCE   GPIO_PortSourceGPIOA
#define WAKEUP_BUTTON_PIN           GPIO_Pin_0
#define WAKEUP_BUTTON_PIN_SOURCE    GPIO_PinSource0
#define WAKEUP_BUTTON_EXTI_LINE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

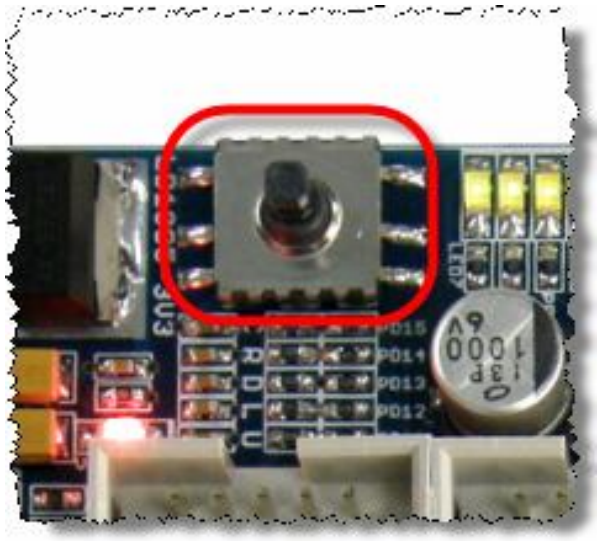
ตัวอย่าง การกำหนดค่าการใช้งาน **PA0** และ **PC13** เป็น **Input Switch**

การใช้งานวงจร Joy Switch

วงจร Joy Switch จะใช้ Joy Switch แบบ 5 ทิศทาง โดยมีโครงสร้างเป็นแบบ กดติด-ปล่อยดับ (Push Button) พร้อมวงจร Pull-Up ใช้กับแหล่งจ่าย +3.3V โดยในขณะที่สวิตช์ยังไม่ถูกกดจะให้ค่าสถานะเป็นโลจิก “1” แต่เมื่อสวิตช์ถูกกดอยู่จะให้สถานะเป็นโลจิก “0” ใช้สำหรับทดสอบการทำงานของ Input Logic และประยุกต์ใช้งานต่างๆ โดยใช้การเชื่อมต่อผ่าน GPIO Input ดังนี้

- Up Position จะใช้ PD11 ในหน้าที่ GPIO Input
- Left Position จะใช้ PD12 ในหน้าที่ GPIO Input
- Down Position จะใช้ PD13 ในหน้าที่ GPIO Input
- Right Position จะใช้ PD14 ในหน้าที่ GPIO Input
- Center Position จะใช้ PD15 ในหน้าที่ GPIO Input





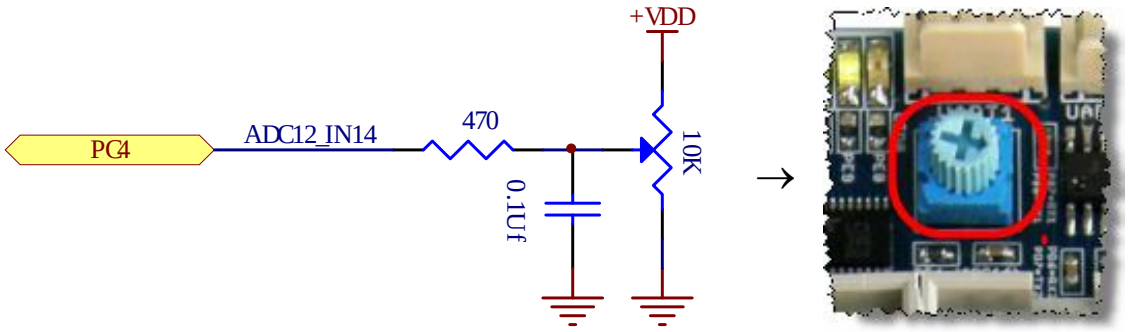
```
// ET-STM32F ARM KIT Hardware Board
// Switch = PD11(Joy-Up)
//        = PD12(Joy-Left)
//        = PD13(Joy-Down)
//        = PD14(Joy-Right)
//        = PD15(Joy-Select)

// 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
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน **Input Joy Switch**

การใช้งานวงจรปรับแรงดัน (0-3V3)

วงจรปรับแรงดันจะใช้ตัวต้านทานปรับค่าได้แบบเกือกม้า ชนิดมีแกนหมุนสำหรับปรับค่า โดยวงจรนี้ใช้กับแหล่งจ่าย +3.3V โดยจะให้ Output เป็นแรงดันซึ่งมีค่าระหว่าง 0V ถึง +3.3V ตามการปรับค่าของตัวต้านทาน จำนวน 1 ชุด โดย Output ที่ได้จะป้อนให้กับขาสัญญาณ PC4 สำหรับใช้สร้างแรงดัน Input เพื่อทดสอบการทำงานของวงจร 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
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน PC4 เป็น Analog Input ADC14

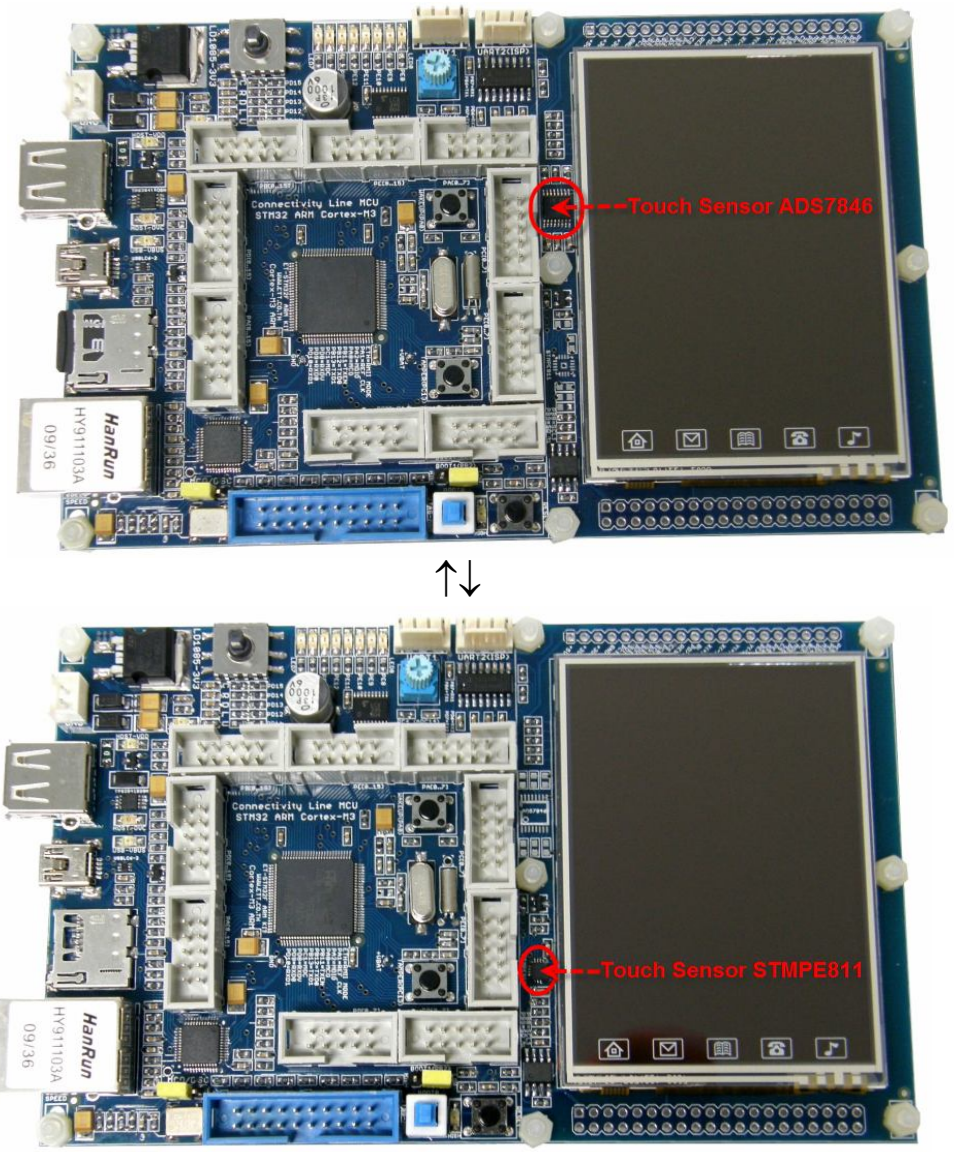
```
//Bargraph LED Display
const unsigned long led_graph[8]={0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
.
.
int ADCVal = 0;
.
.
.
ET_STM32_LED_Init(LED0);
ET_STM32_LED_Init(LED1);
ET_STM32_LED_Init(LED2);
ET_STM32_LED_Init(LED3);
ET_STM32_LED_Init(LED4);
ET_STM32_LED_Init(LED5);
ET_STM32_LED_Init(LED6);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

ตัวอย่าง การอ่านค่า **ADC Input** จาก **ADC14**

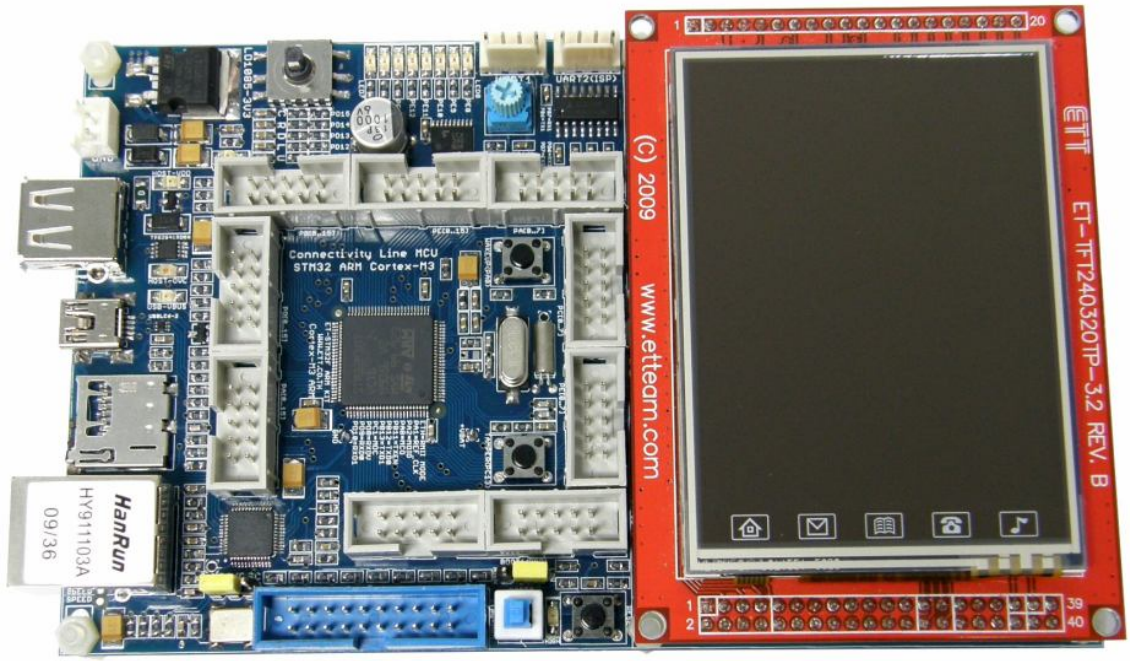
การใช้งานจอแสดงผล Graphic LCD แบบ TFT LCD

สำหรับการเชื่อมต่อกับ Graphic LCD นั้น วงจรของบอร์ด ET-STM32F ARM KIT ได้รับการออกแบบให้สามารถเชื่อมต่อกับGraphic LCD แบบ TFT LCD ขนาด 3.2นิ้ว ได้ 2 รูปแบบ คือ

- ใช้การติดตั้งโมดูล LCD เข้ากับบอร์ดโดยตรงแบบถาวร โดยใช้ TFT LCD รุ่น KWH032GM02-F05 โดยใช้การเชื่อมต่อสัญญาณกับ TFT LCD ในโหมด SPI โดย TFT LCD รุ่นนี้จะมี Sensor ของ Touch Screen รวมอยู่ด้วย ซึ่งอุปกรณ์ที่จะใช้สำหรับอ่านค่า Sensor ของ Touch Screen วงจรของบอร์ด ET-STM32F ARM KIT จะออกแบบให้สามารถสามารถเลือกใช้ชิพ เบอร์ STMPE811 ซึ่งใช้การเชื่อมต่อแบบ I2C หรือ อาจใช้ชิพ ADS7846 ซึ่งใช้การเชื่อมต่อแบบ SPI ก็ได้ (ขึ้นอยู่กับการติดตั้งชิพ ของบอร์ดในขั้นตอนการผลิต)



- ใช้การติดตั้งบอร์ดแสดงผลของ อีทีที รุ่น ET-TFT240320TP-3.2 REV.B ซึ่ง บอร์ดแสดงผลรุ่นนี้จะติดตั้ง TFT LCD รุ่น KWH032GM02-F05 พร้อมชิพ ADS7846 สำหรับอ่านค่า Touch Sensor ไว้เรียบร้อยแล้วภายในบอร์ด โดยใช้การติดตั้งผ่าน Connector และสามารถ ใส่ หรือ ถอด ออกจากบอร์ดได้โดยง่าย โดยต้องเลือกกำหนดรูปแบบการ Interface กับบอร์ดให้เป็นแบบ SPI ด้วย



การเชื่อมต่อ TFT LCD รุ่น KWH032GM02-F05

ส่วนของ TFT LCD รุ่น KWH032GM02-F05 จะใช้การเชื่อมต่อแบบ SPI Mode โดยจะใช้ SPI3 ของ MCU ในการติดต่อ โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- CS GLCD จะใช้ PC8 ในหน้าที่ GPIO Output
- SCL GLCD จะใช้ PC10 ในหน้าที่ SCK3 ของ SPI3
- SDO GLCD จะใช้ PC11 ในหน้าที่ MISO3 ของ SPI3
- SDI GLCD จะใช้ PC12 ในหน้าที่ MOSI3 ของ SPI3
- BL GLCD จะใช้ PD7 ในหน้าที่ GPIO Output

การเชื่อมต่อกับ Touch Screen Sensor โดยใช้ ADS7846

ส่วนของ Touch Screen ในกรณีใช้ชิพ ADS7846 จะใช้การเชื่อมต่อแบบ SPI โดยจะใช้ขา GPIO ในการสร้างสัญญาณเลียนแบบ SPI ขึ้นมาแทน โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- DCLK ADS7846 จะใช้ PE7 ในหน้าที่ GPIO Output (SPI:SCK)
- CS ADS7846 จะใช้ PE6 ในหน้าที่ GPIO Output (CS#)
- DOUT ADS7846 จะใช้ PE4 ในหน้าที่ GPIO Input (SPI:MISO)
- DIN ADS7846 จะใช้ PE5 ในหน้าที่ GPIO Output(SPI:MOSI)
- PENIRQ ADS7846 จะใช้ PE3 ในหน้าที่ GPIO Input

การเชื่อมต่อกับ Touch Screen Sensor โดยใช้ STMPE811

ส่วนของ Touch Screen ในกรณีใช้ชิพ STMPE811 จะใช้การเชื่อมต่อแบบ I2C ซึ่งมีตำแหน่งแอดเดรสของ Device ในการเชื่อมต่อของ I2C เท่ากับ 0x82 โดยจะใช้ I2C1 ของ MCU ในการติดต่อ โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- SDAT STMPE811 จะใช้ PB9 ในหน้าที่ SDA1 ของ I2C1
- SCLK STMPE811 จะใช้ PB8 ในหน้าที่ SCL1 ของ I2C1
- INT STMPE811 จะใช้ PE3 ในหน้าที่ 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);
```

ตัวอย่าง การกำหนดค่า **Pin** สำหรับใช้งาน **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);
}
```

แสดงตัวอย่าง ฟังก์ชั่นการ อ่าน เขียน ข้อมูลกับ ADS7846 โดยใช้ GPIO เลียบแบบ 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_7bit;
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);
.
.
.
```

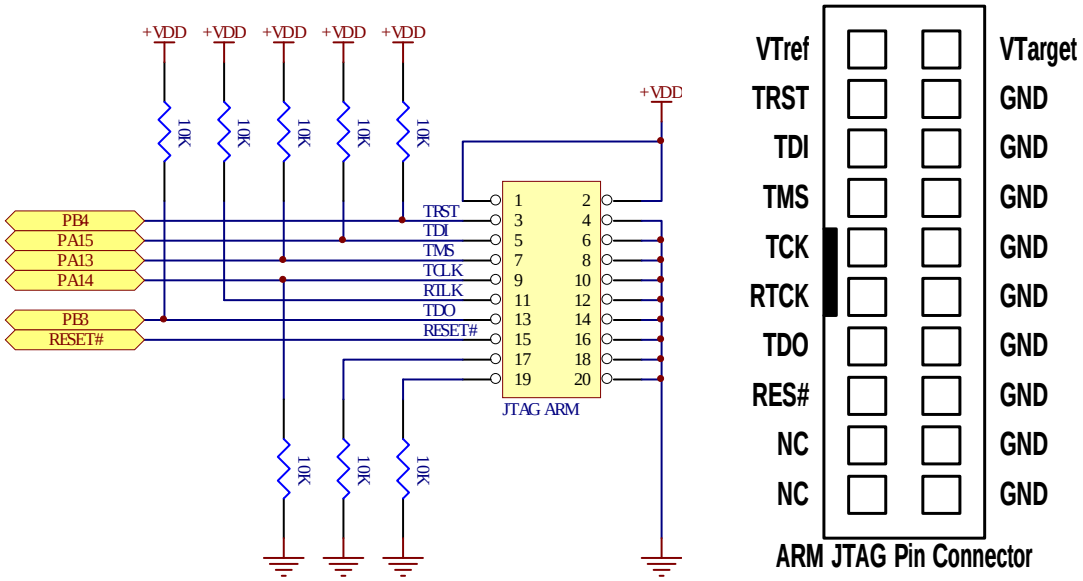
ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน Touch Sensor STMPE811

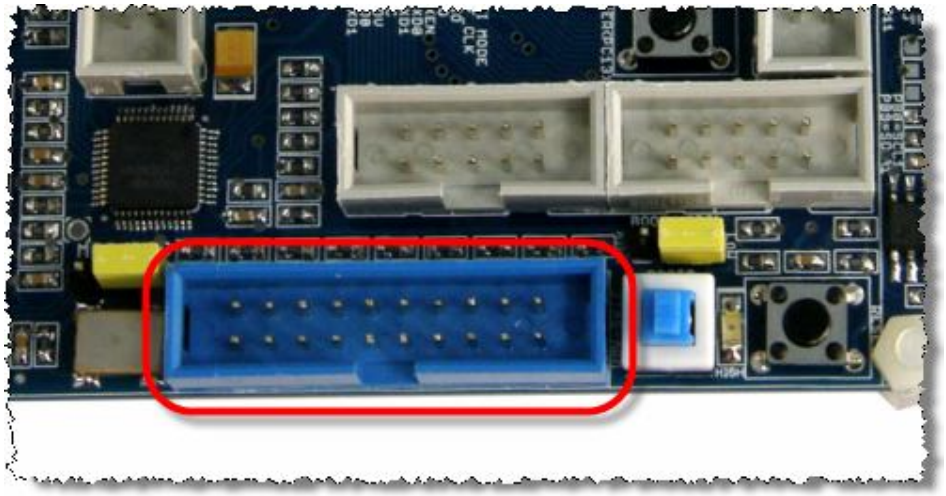
```
#define LCD_CS_PIN              GPIO_Pin_8              // PC8 = CS# GLCD
#define LCD_CS_GPIO_PORT        GPIOC
#define LCD_CS_GPIO_CLK         RCC_APB2Periph_GPIOC
#define LCD_BL_PIN              GPIO_Pin_7              // PD7 = BL GLCD
#define LCD_BL_GPIO_PORT        GPIOD
#define LCD_BL_GPIO_CLK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

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน GLCD

การใช้งาน JTAG ARM

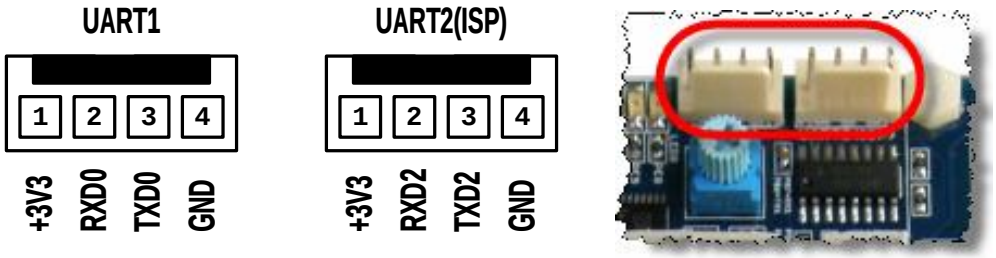
JTAG หรือ JTAG ARM จะเป็น Connector แบบ IDE 20 Pin สำหรับ Interface กับ JTAG Debugger โดยมีการจัดวงจรและสัญญาณตามมาตรฐานของ JTAG ดังนี้





พอร์ต RS232

เป็นสัญญาณ RS232 ซึ่งผ่านวงจรแปลงระดับสัญญาณ MAX3232 เรียบร้อยแล้ว โดยมีจำนวน 2 ช่อง ด้วยกันคือ UART1 และ UART2 โดยทั้ง 2 ช่องสามารถใช้เชื่อมต่อกับสัญญาณ RS232 เพื่อรับส่งข้อมูลได้ นอกจากนี้แล้ว UART2 ยังสามารถใช้งานเป็น ISP Download สำหรับทำการ Download Hex File ให้กับ MCU ได้ด้วย โดยในกรณีนี้ต้องใช้งานร่วมกับ Jumper BOOT1, Switch BOOT0 และ Switch Reset เพื่อ Reset ให้ CPU เริ่มต้นทำงานใน Boot-Loader Mode เพื่อทำการ Download Hex File ให้กับ CPU ได้ด้วย(ดูรายละเอียดเพิ่มเติมเรื่อง “การ Download Hex File ให้กับ MCU ของบอร์ด”)



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

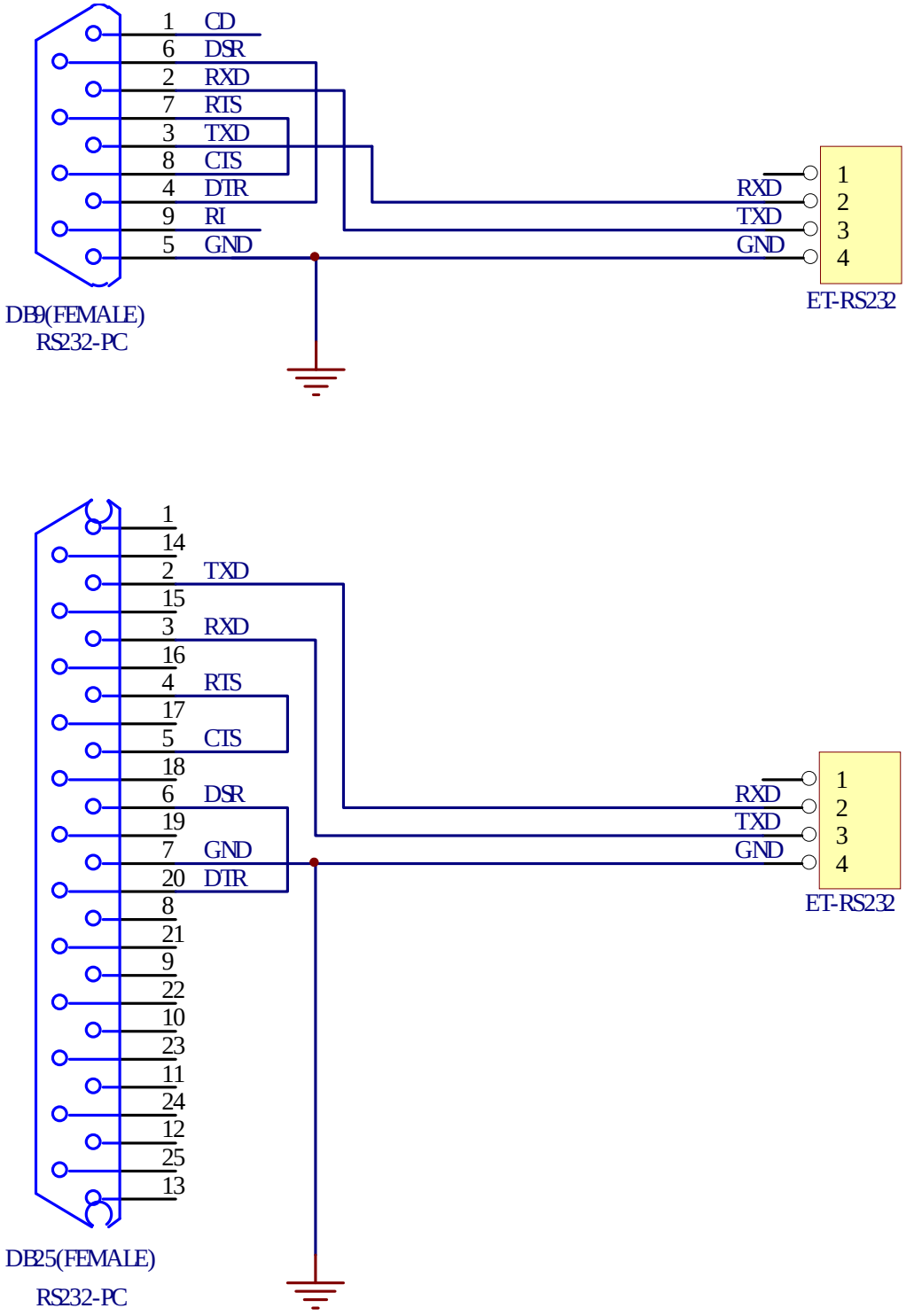
เนื่องจากระบบ Hardware USART ของ STM32F107VCT6 นั้นจะสามารถกำหนดขาสัญญาณในการเชื่อมต่อได้ 2 ชุด คือ Default และ Remap ซึ่งบอร์ด ET-STM32F ARM KIT นั้นเลือกใช้ขาสัญญาณชุด Remap เป็นจุดเชื่อมต่อกับ UART ดังนั้น ผู้ใช้ต้องกำหนดคำสั่งในการเลือกใช้ขาสัญญาณให้ถูกต้องด้วย สำหรับ Code ตัวอย่างการกำหนดค่า UART ในส่วนเริ่มต้นเป็นดังนี้

```
// 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);
```

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน UART1 และ UART2

สำหรับ Cable ที่จะใช้ในการเชื่อมต่อ RS232 ระหว่าง Comport ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้วต่อ UART1 และ UART2 ของบอร์ด ET-STM32F ARM KIT นั้น เป็นดังนี้



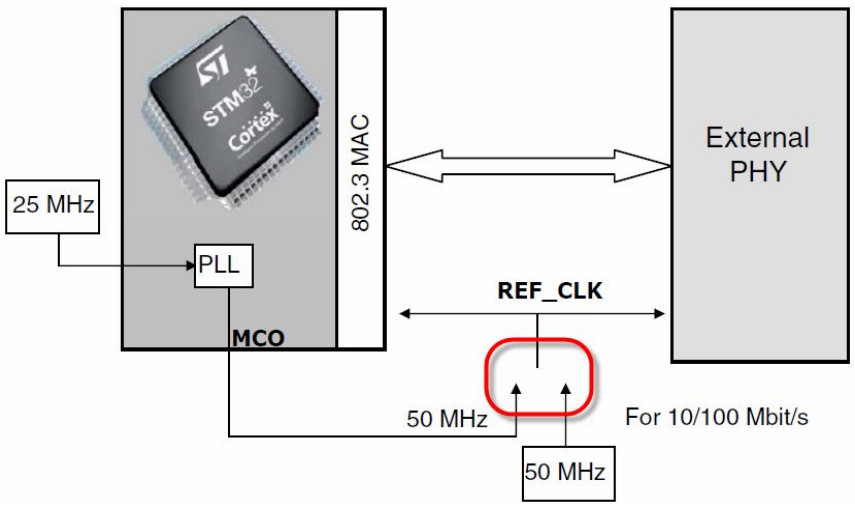
รูป แสดงวงจรสาย Cable สำหรับ RS232

Ethernet LAN

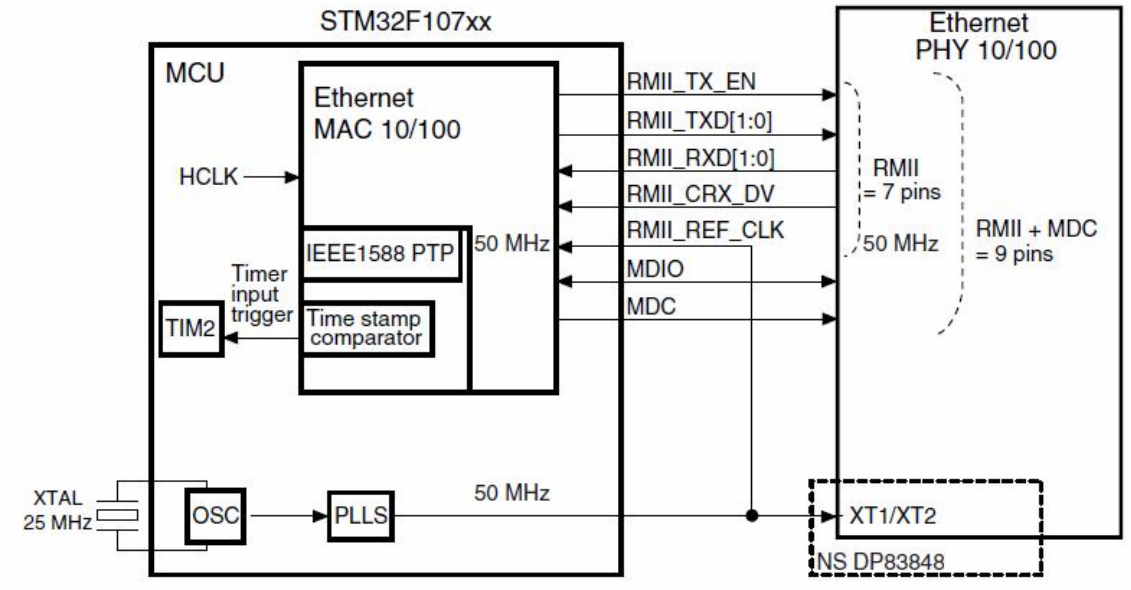
สำหรับการเชื่อมต่อกับเครือข่าย Network Ethernet LAN ระหว่างบอร์ด ET-STM32F ARM KIT นั้น จะใช้ขั้วต่อมาตรฐาน Ethernet แบบ RJ45 โดยวงจรส่วนนี้จะใช้ขาสัญญาณ ในการเชื่อมต่อโดยใช้ Chips Physical Ethernet เบอร์ DP83848V เป็น Driver ในการเชื่อมต่อโดยใช้วงจรการเชื่อมต่อระหว่าง STM32F107VCT6 กับ DP83848V ในโหมด RMII (Reduced Media Independent Interface) โดยใช้สัญญาณในการเชื่อมต่อจำนวน 9 เส้น เมื่อใช้สัญญาณนาฬิกาจากโมดูล Oscillator 50MHz หรือ ใช้สัญญาณในการเชื่อมต่อ 10 เส้น เมื่อใช้สัญญาณนาฬิกาจากวงจร MCO (ขา PA8) ดังนี้

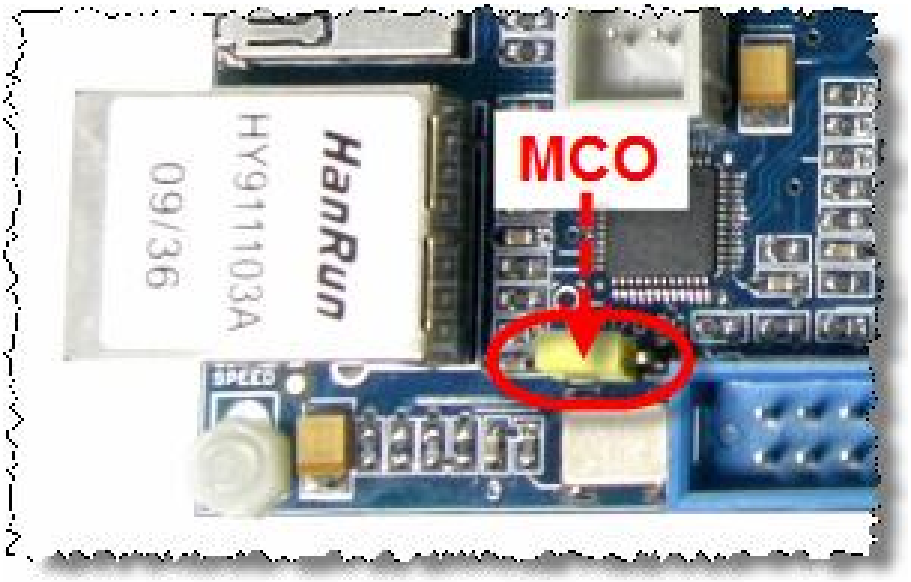
- PA1 เป็น REF_CLK(Default)
- PA2 เป็น MDIO(Default)
- PC1 เป็น MDC(Default)
- PB11 เป็น TX_EN(Default)
- PB12 เป็น TXD0(Default)
- PB13 เป็น TXD1(Default)
- PD8 เป็น CRS_DV(RMII Remap)
- PD9 เป็น RXD0(RMII Remap)
- PD10 เป็น RXD1(RMII Remap)

โดยระบบฮาร์ดแวร์ของบอร์ด ET-STM32F ARM KIT นั้น ออกแบบวงจรให้สามารถเลือกสัญญาณนาฬิกาค่าความถี่ขนาด 50MHz ให้กับวงจร Ethernet LAN Driver ได้ 2 แหล่ง คือ จากวงจร MCO ภายในตัว MCU(ขา PA8) และจากโมดูล Oscillator ความถี่ 50MHz จากภายนอก โดยใช้ Jumper MCO/OSC เป็นตัวเลือกแหล่งของสัญญาณนาฬิกา



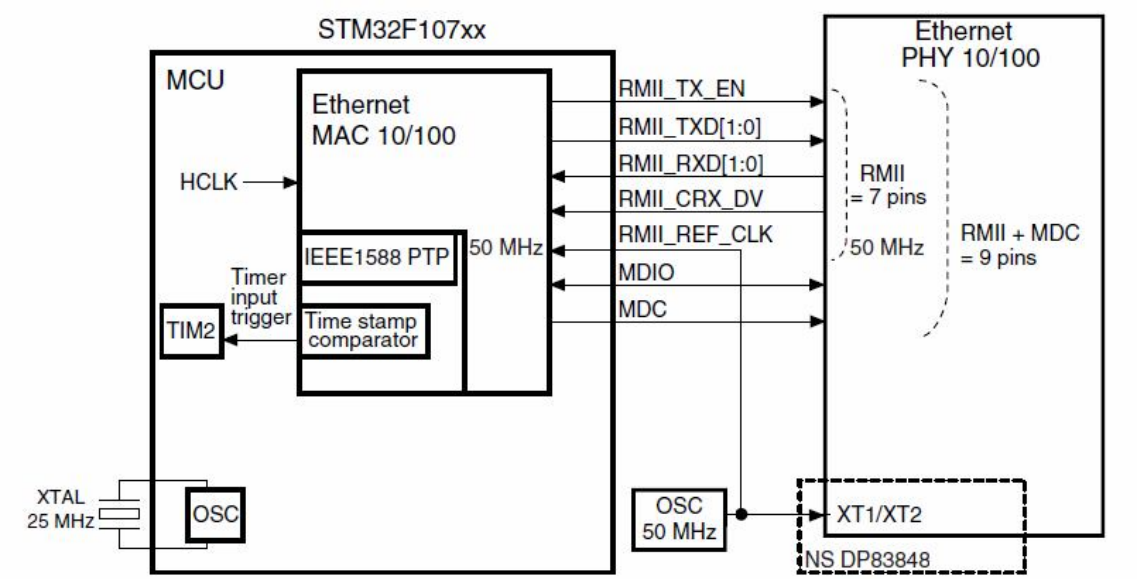
ในกรณีใช้สัญญาณนาฬิกา 50MHz จากวงจร MCO (ขา PA8) จะใช้ขาสัญญาณในการเชื่อมต่อจำนวน 10 ขา โดยใช้ PA8 เป็นขากำเนิดสัญญาณนาฬิกา 50MHz เพื่อป้อนให้กับ DP83848V โดยในส่วนของบอร์ด ET-STM32F ARM KIT นั้นให้เลือกกำหนด Jumper MCO/OSC ไว้ทางด้าน MCO และเพิ่มส่วนของโปรแกรมสำหรับ Initial การทำงานของวงจร MCO (Main Clock Oscillator) เพื่อ ทำหน้าที่สร้างสัญญาณนาฬิกาค่า 50MHz ส่งออกทางขา PA8 ด้วยดังรูป

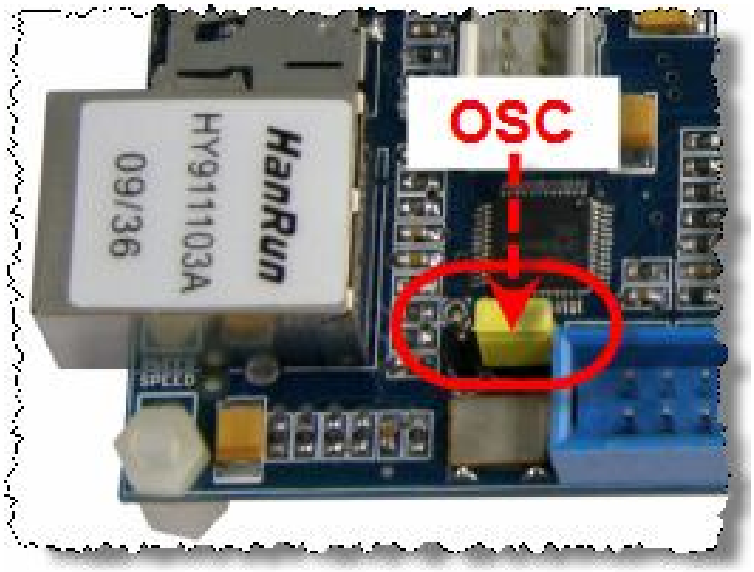




แสดงการเชื่อมต่อ Ethernet LAN ในโหมด RMII โดยใช้ MCO กำเนิดสัญญาณนาฬิกา ค่า 50MHz

ในกรณีใช้สัญญาณนาฬิกา 50MHz จากโมดูล Oscillator จะใช้ขาสัญญาณในการเชื่อมต่อจำนวน 9 ขา โดยต้องใช้ Oscillator ค่า 50MHz เป็นตัวกำเนิดสัญญาณนาฬิกาค่าความถี่ 50MHz เพื่อป้อนให้กับ DP83848V โดยในส่วนของบอร์ด ET-STM32F ARM KIT นั้นให้เลือกกำหนด Jumper MCO/OSC ไว้ทางด้าน OSC ด้วยดังรูป



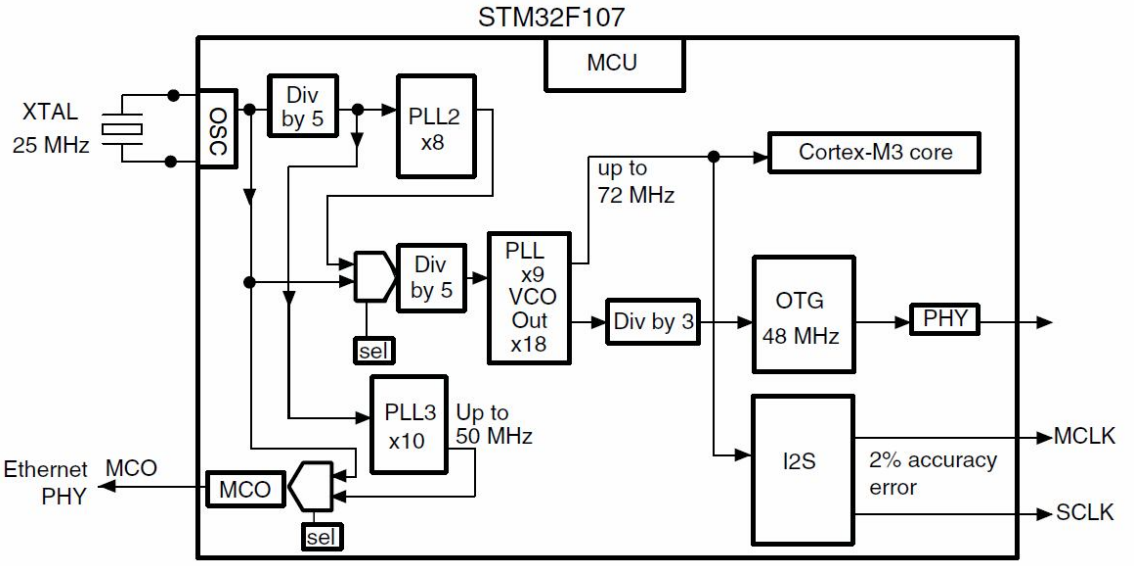


แสดงการเชื่อมต่อ Ethernet LAN ในโหมด RMII โดยใช้โมดูล Oscillator ค่า 50MHz

```
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);
```

ตัวอย่างการ **Initial Pin** สำหรับ **Interface** กับ **Ethernet Driver** เบอร์ **DP83848V**

โดยระบบฮาร์ดแวร์ของ STM32 มีวงจร MCO (Main Clock Output) โดยนำสัญญาณนาฬิกาหลักที่ป้อนให้กับ MCU (Crystal 25 MHz) ที่ผ่านวงจรหาร 5 มาแล้ว (5 MHz) เพื่อนำไปคูณความถี่ด้วยวงจร Phase-Lock-Loop (PLL3) เพื่อทำการคูณความถี่ให้สูงขึ้นอีก 10 เท่า สำหรับสร้างสัญญาณนาฬิกาความถี่ 50MHz เพื่อใช้ป้อนให้กับ Ethernet Driver(DP83848V) ได้ด้วย โดยใช้ขา PA8 ร่วมกับ Phase-Lock-Loop(PLL3) ดังตัวอย่าง



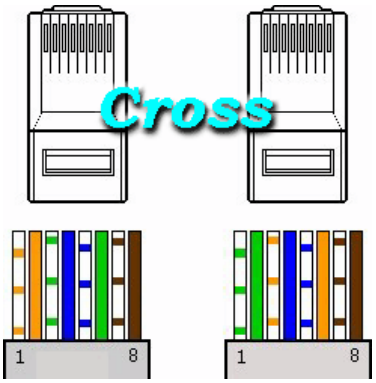
↓

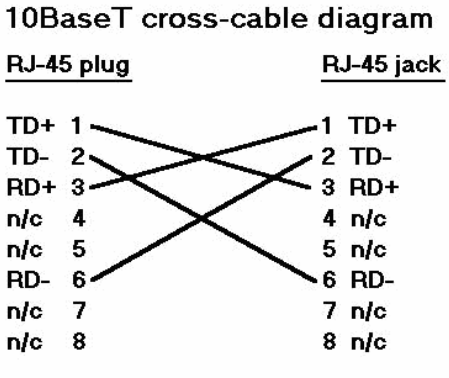
```
/* Start of Config MCO Clock = 50MHz on PA8 */
// Configure MCO (PA8) as alternate function push-pull
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;             //PA8 = MCO
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

// 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 */
```

ซึ่งเมื่อเลือกใช้ขาสัญญาณ PA8 ในหน้าที่ของ MCO Output เพื่อสร้างสัญญาณนาฬิกาค่าความถี่ 50MHz จะทำให้สามารถประหยัดต้นทุนทางฮาร์ดแวร์ได้มากขึ้น เพราะไม่จำเป็นต้องใช้โมดูล Oscillator ในการสร้างความถี่ 50MHz ได้

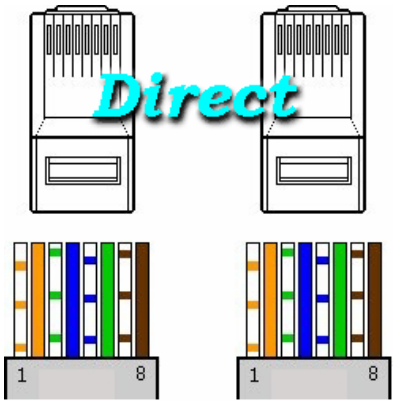
สำหรับวิธีการเชื่อมต่อสายสัญญาณ Ethernet LAN ของบอร์ดเข้ากับระบบเครือข่ายจะทำได้ 2 แบบด้วยกัน คือการต่อแบบ Direct Line และต่อผ่าน Hub

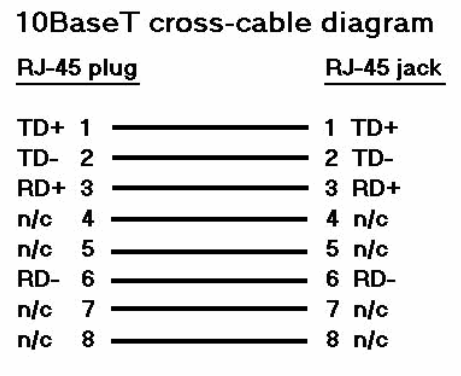
- กรณีที่ 1 คือ การเชื่อมต่อเข้ากับคอมพิวเตอร์โดยตรง สาย LAN จะต้องเข้าสายแบบ Cross





- กรณีที่ 2 คือ การเชื่อมต่อผ่าน Hub ของเครื่องคอมพิวเตอร์ Server จะต้องเข้าสายแบบ Direct

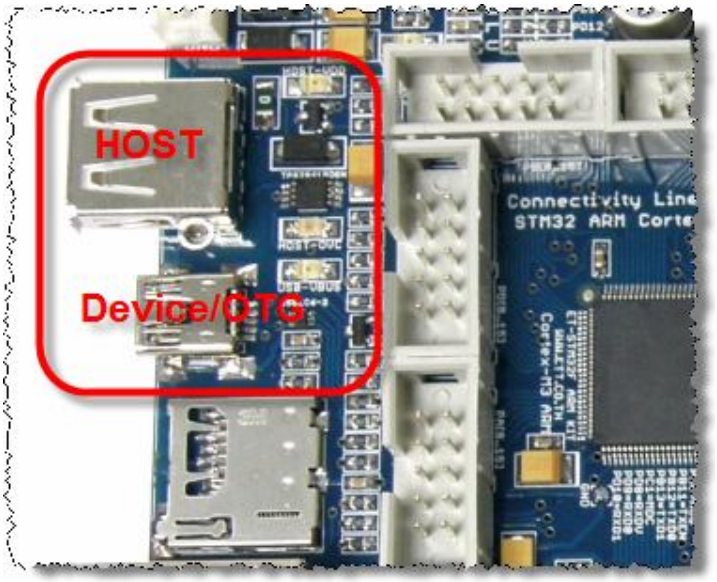




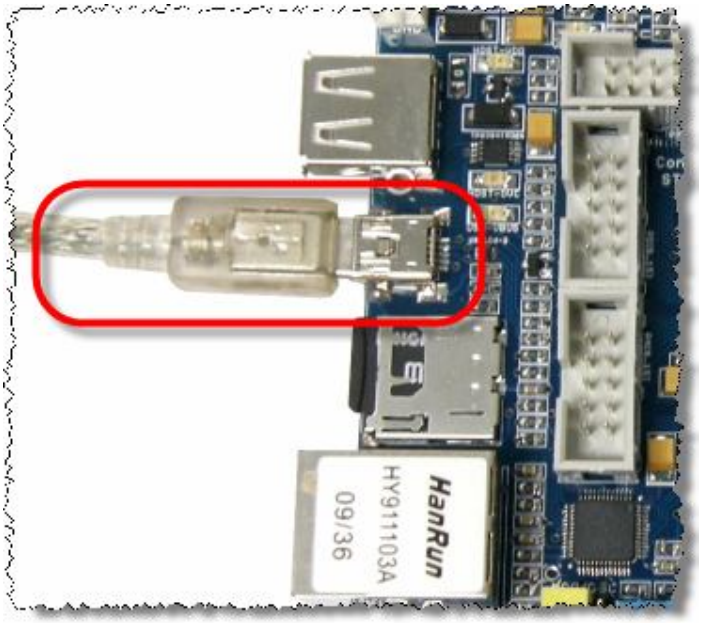
การใช้งาน USB

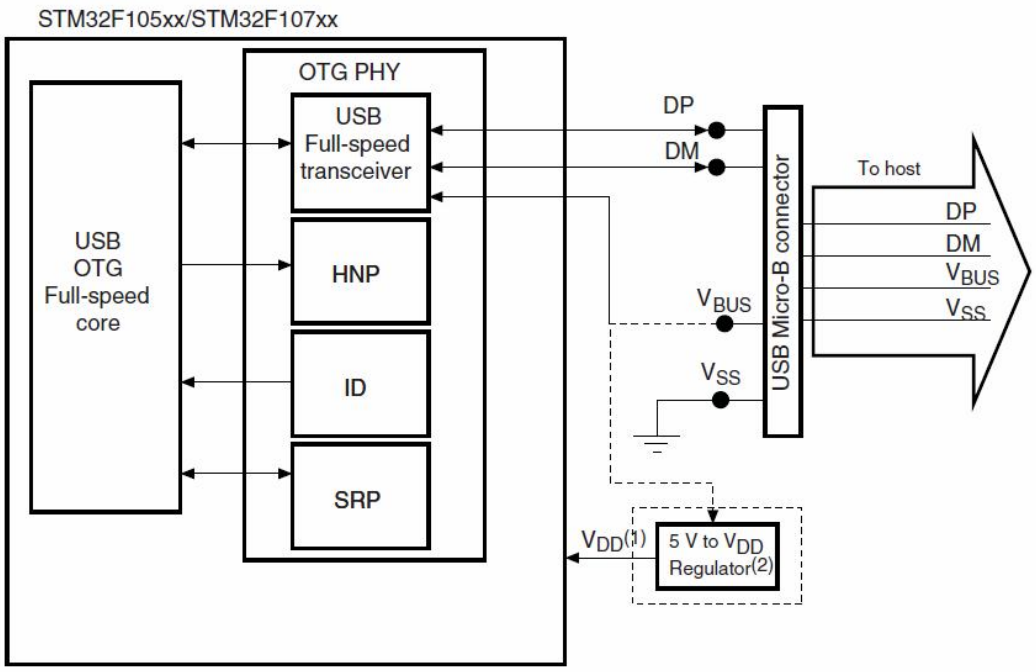
บอร์ด ET-STM32F ARM KIT ถูกออกแบบให้มีพอร์ตสำหรับเชื่อมต่อกับอุปกรณ์ USB ทั้งแบบ Device/OTG (On-The-Go) หรือ USB Host ก็ได้ ขึ้นอยู่กับการเขียนโปรแกรม กำหนดหน้าที่การทำงานของ USB ในตัว MCU ของ STM32F107VCT6 โดยจะใช้สัญญาณที่เกี่ยวข้องในการเชื่อมต่อกับ USB จำนวน 6 เส้น ดังนี้

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



USB Device



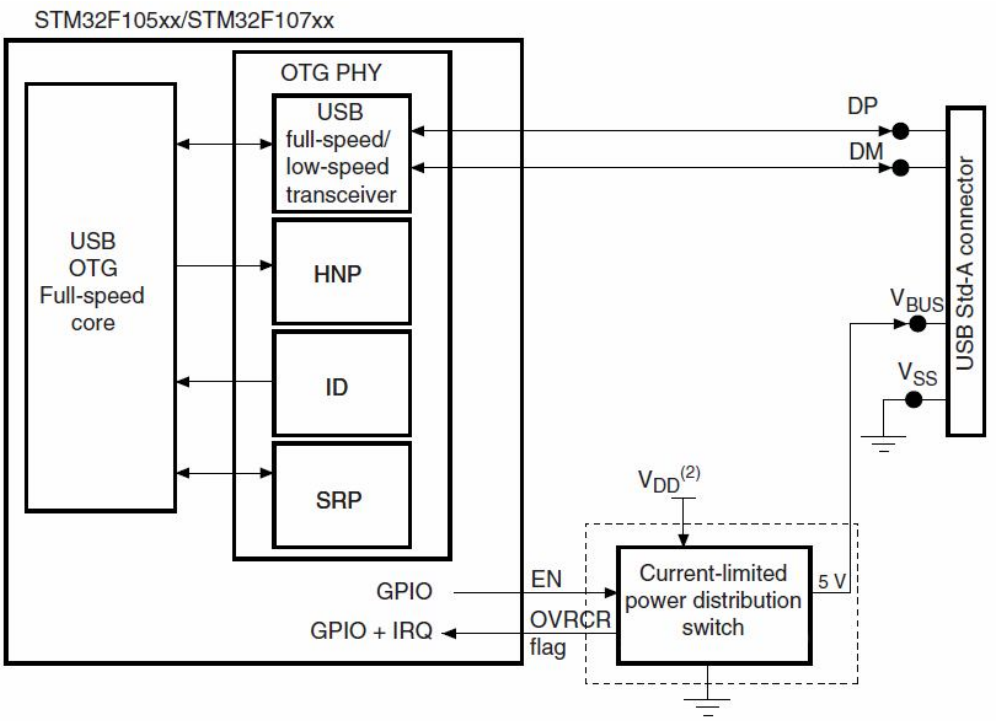


แผนผังแสดงการเชื่อมต่อ USB แบบ Device Mode

ในโหมดนี้ จะใช้สัญญาณในการเชื่อมต่อจำนวน 3 เส้น

- PA9 ทำหน้าที่เป็น USB_VBUS
- PA11 ทำหน้าที่เป็น USB_DM
- PA12 ทำหน้าที่เป็น USB_DP

USB Host

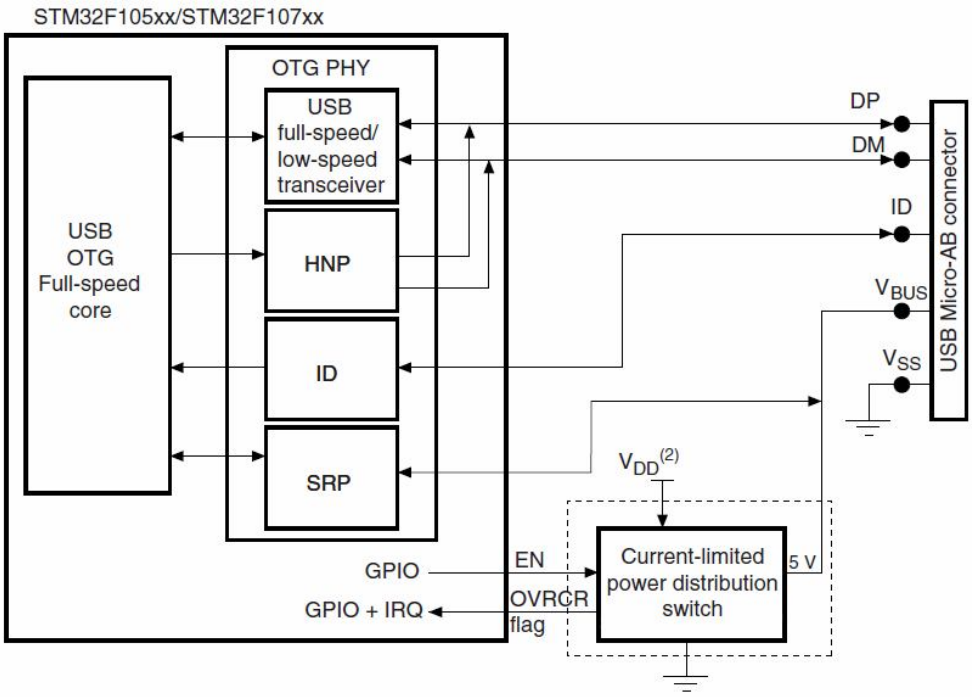


แผนผังแสดงการเชื่อมต่อ USB แบบ Host Mode

ในโหมดนี้ในโหมดนี้ จะใช้สัญญาณในการเชื่อมต่อจำนวน 4 เส้น

- PE1 ทำหน้าที่เป็น GPIO Input (Host OVRCR : Host Over Current)
- PC9 ทำหน้าที่เป็น GPIO Output (HOST_EN : Host Enable)
- PA11 ทำหน้าที่เป็น USB_DM
- PA12 ทำหน้าที่เป็น USB_DP

USB OTG



แผนผังแสดงการเชื่อมต่อ USB แบบ OTG Mode

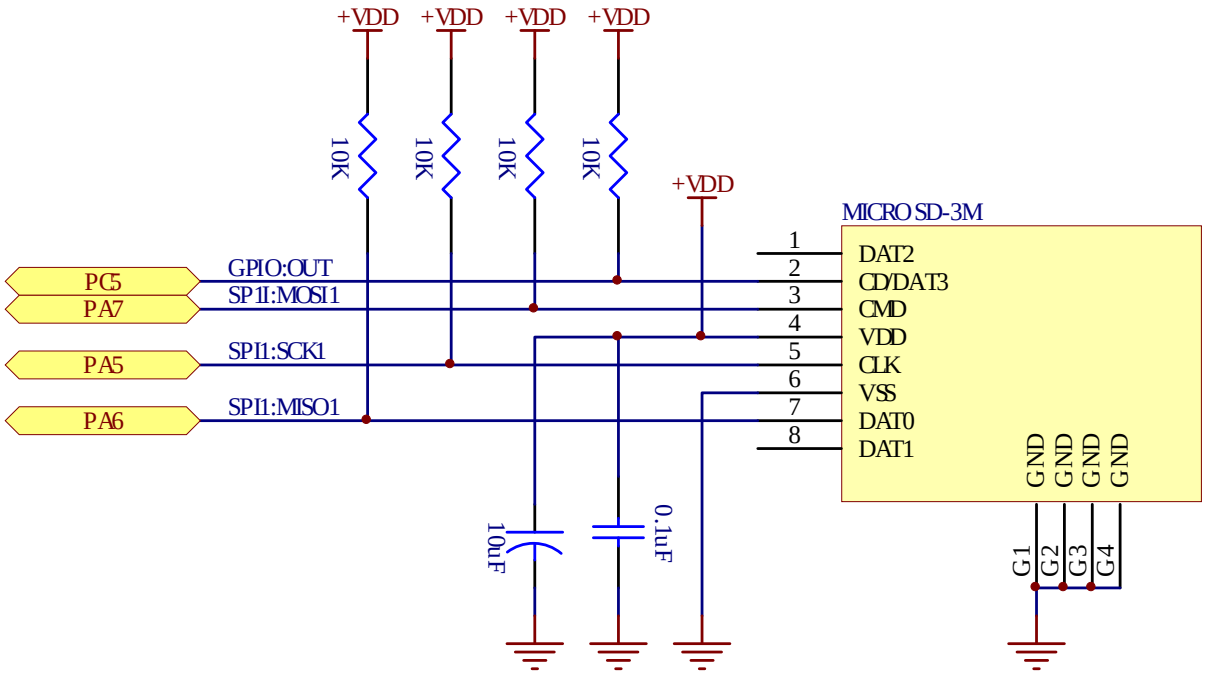
ในโหมดนี้ในโหมดนี้ จะใช้สัญญาณในการเชื่อมต่อจำนวน 6 เส้น

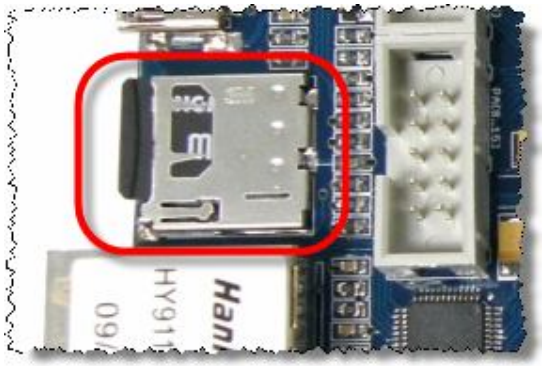
- PE1 ทำหน้าที่เป็น GPIO Input (Host OVRCR : Host Over Current)
- PC9 ทำหน้าที่เป็น GPIO Output (HOST_EN : Host Enable)
- PA9 ทำหน้าที่เป็น USB_VBUS
- PA10 ทำหน้าที่เป็น USB_ID
- PA11 ทำหน้าที่เป็น USB_DM
- PA12 ทำหน้าที่เป็น USB_DP

การ์ดหน่วยความจำ SD Card แบบ Micro-SD

โครงสร้างของบอร์ด ET-STM32F ARM KIT รองรับการเชื่อมต่อกับการ์ดหน่วยความจำ SD Card แบบ Micro-SD โดยใช้การเชื่อมต่อแบบ SPI โดยใช้ขาสัญญาณ PC5, PA[5..7] ในการเชื่อมต่อกับการ์ด ซึ่งในการติดต่อสั่งงาน การ์ดนั้น สามารถโปรแกรม Pin I/O ของ PC5 และ PA[5..7] ให้ทำงานในโหมด SPI โดยต้องกำหนดหน้าที่ของขาสัญญาณ ของ MCU เป็นดังนี้

- CLK ใช้ PA5 ในหน้าที่ SCK ของ SPI1
- DAT0 ใช้ PA6 ในหน้าที่ MISO ของ SPI1
- CMD ใช้ PA7 ในหน้าที่ MOSI ของ SPI1
- CD/DAT3 ใช้ PC5 ในหน้าที่ของ GPIO Output

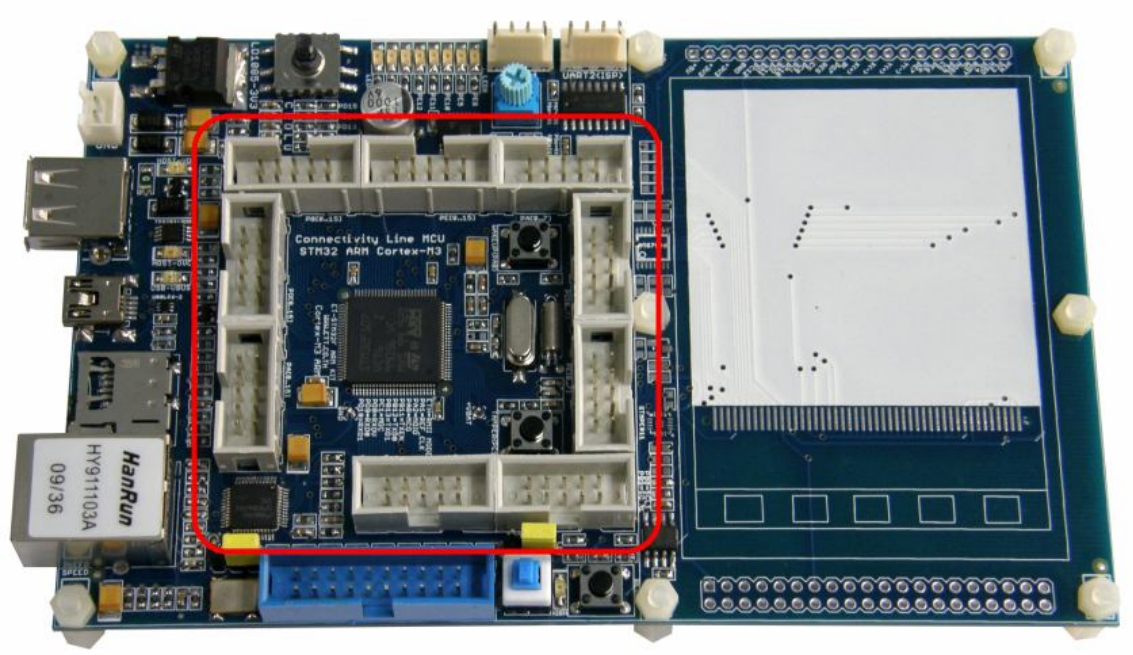


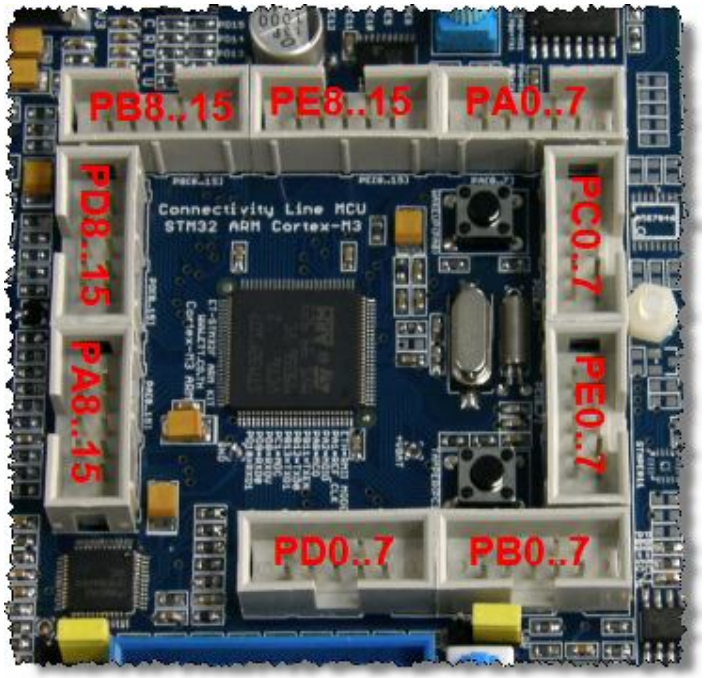


```
// ET-STM32F ARM KIT(STM32F107VCT6) Hardware Kit
// SD Card Interface(PC5 = CS#,PA5 = SCK,PA6 = MISO,PA7 MOSI)
#define SD_SPI                  SPI1
#define SD_SPI_PORT             GPIOA
#define SD_SPI_GPIO_PORT_CLOCK  RCC_APB2Periph_GPIOA
#define SD_SPI_PIN_SCK          GPIO_Pin_5
#define SD_SPI_PIN_MISO         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);
```

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน SD Card

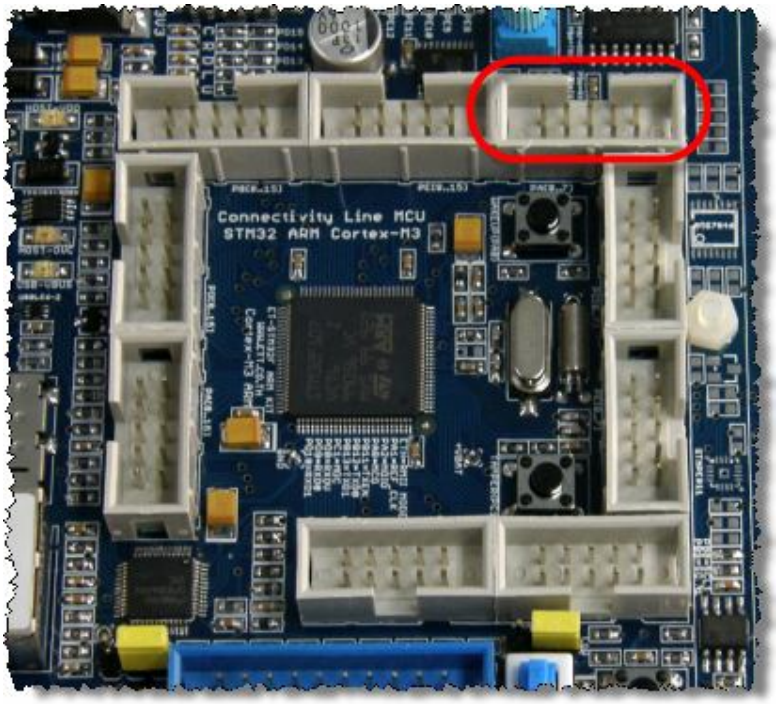
ขั้วต่อ Port I/O ต่าง ๆของบอร์ด

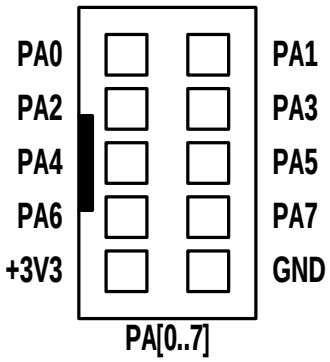




สำหรับขั้วต่อ Port I/O ของ CPU นั้น จะจัดเรียงออกมารอไว้ยังขั้วต่อแบบต่างๆ สำหรับให้ผู้ใช้เลือกต่อออกไปใช้งานตามต้องการ โดยใช้ขั้วต่อแบบ IDE 10 Pin จำนวน 9 ชุด ชุดละ 8 บิต โดยมีการจัดเรียงสัญญาณของขั้วต่อสำหรับเชื่อมต่อสัญญาณแต่ละชุดดังนี้

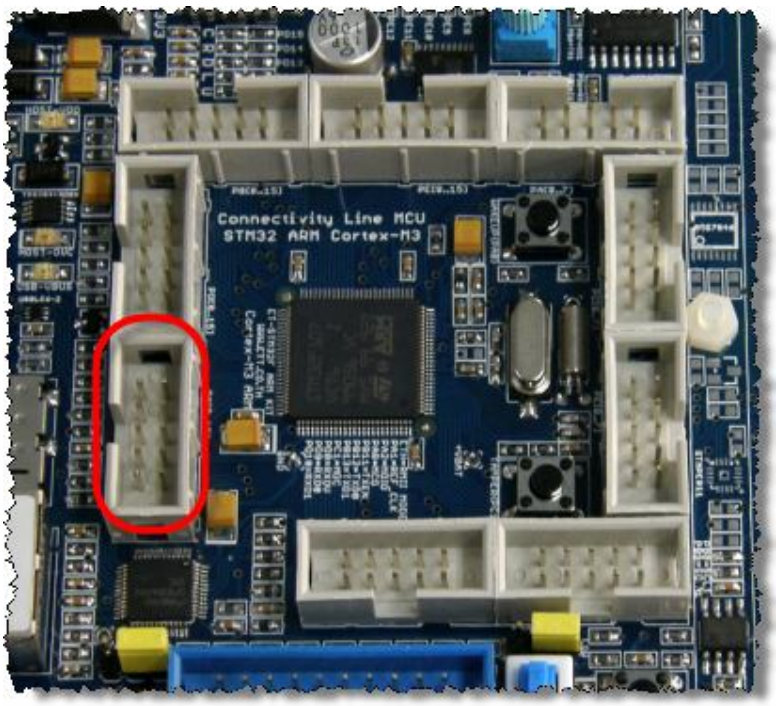
- ขั้วต่อ IDE 10 Pin ของ PA[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

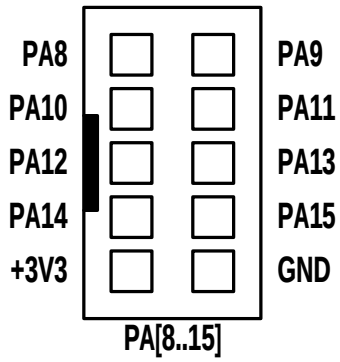




- PA0 ถูกจัดสรรให้ใช้งานกับ Switch Wakeup ด้วยแล้ว
- PA1 ถูกจัดสรรให้ใช้งานในหน้าที่ REF_CLK ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PA2 ถูกจัดสรรให้ใช้งานในหน้าที่ MDIO ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PA5 ถูกจัดสรรให้ใช้งานในหน้าที่ SCK ของ SPI1 ในการเชื่อมต่อกับ SD Card ด้วยแล้ว
- PA6 ถูกจัดสรรให้ใช้งานในหน้าที่ MISO ของ SPI1 ในการเชื่อมต่อกับ SD Card ด้วยแล้ว
- PA7 ถูกจัดสรรให้ใช้งานในหน้าที่ MOSI ของ SPI1 ในการเชื่อมต่อกับ SD Card ด้วยแล้ว

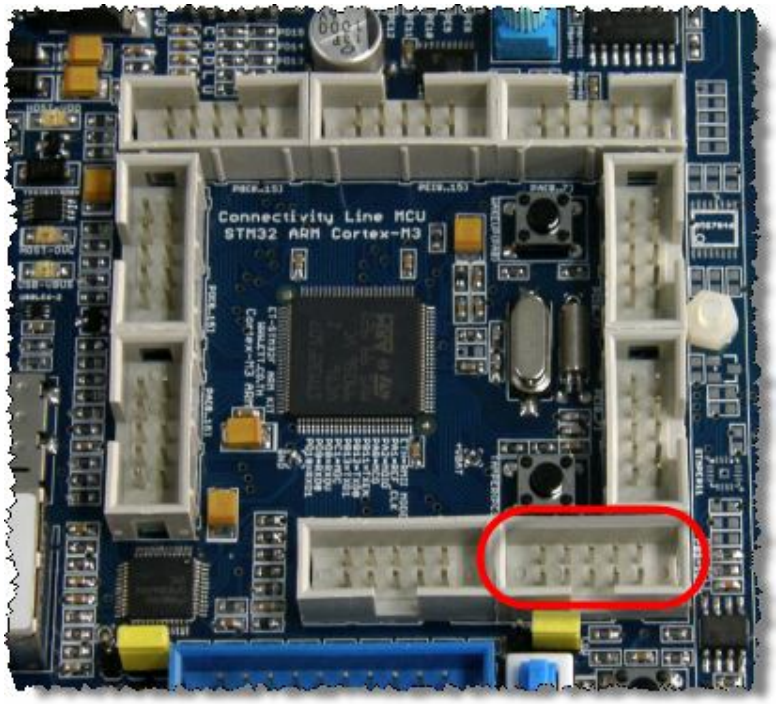
- ขั้วต่อ IDE 10 Pin ของ PA[8..15] มีการจัดเรียงสัญญาณไว้ดังนี้

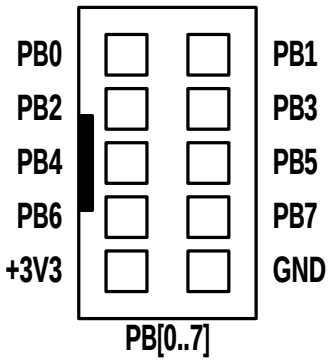




- PA8 ถูกจัดสรรให้ใช้งานในหน้าที่ MCO ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PA9 ถูกจัดสรรให้ใช้งานในหน้าที่ USB_FS_VBUS
- PA10 ถูกจัดสรรให้ใช้งานในหน้าที่ USB_FS_ID
- PA11 ถูกจัดสรรให้ใช้งานในหน้าที่ USB_FS_DM
- PA12 ถูกจัดสรรให้ใช้งานในหน้าที่ USB_FS_DP
- PA13 ถูกจัดสรรให้ใช้งานในหน้าที่ TMS ใจการเชื่อมต่อกับ JTAG ด้วยแล้ว
- PA14 ถูกจัดสรรให้ใช้งานในหน้าที่ TCLK ในการเชื่อมต่อกับ JTAG ด้วยแล้ว
- PA15 ถูกจัดสรรให้ใช้งานในหน้าที่ TDI ในการเชื่อมต่อกับ JTAG ด้วยแล้ว

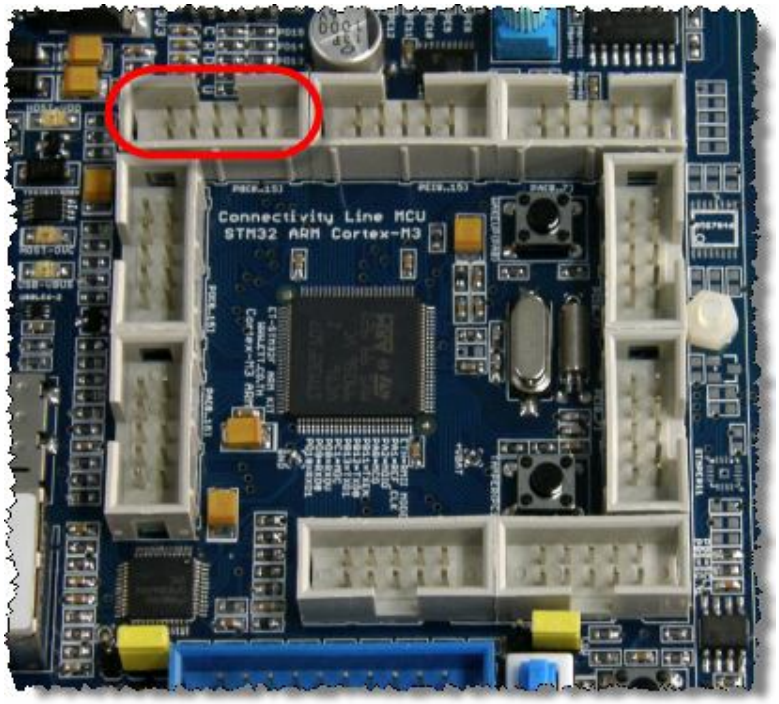
- ขั้วต่อ IDE 10 Pin ของ PB[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

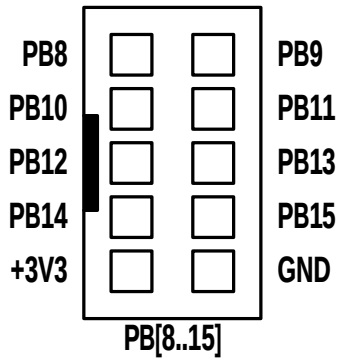




- PB2 ถูกจัดสรรให้ใช้งานในหน้าที่ BOOT1 ในการเชื่อมต่อกับ ISP Boot Loader แล้ว
- PB3 ถูกจัดสรรให้ใช้งานในหน้าที่ TDO ในการเชื่อมต่อกับ JTAG ด้วยแล้ว
- PB4 ถูกจัดสรรให้ใช้งานในหน้าที่ TRST ในการเชื่อมต่อกับ JTAG ด้วยแล้ว
- PB6 ถูกจัดสรรให้ใช้งานในหน้าที่ TXD1 ของ USART1 ด้วยแล้ว
- PB7 ถูกจัดสรรให้ใช้งานในหน้าที่ RXD1 ของ USART1 ด้วยแล้ว

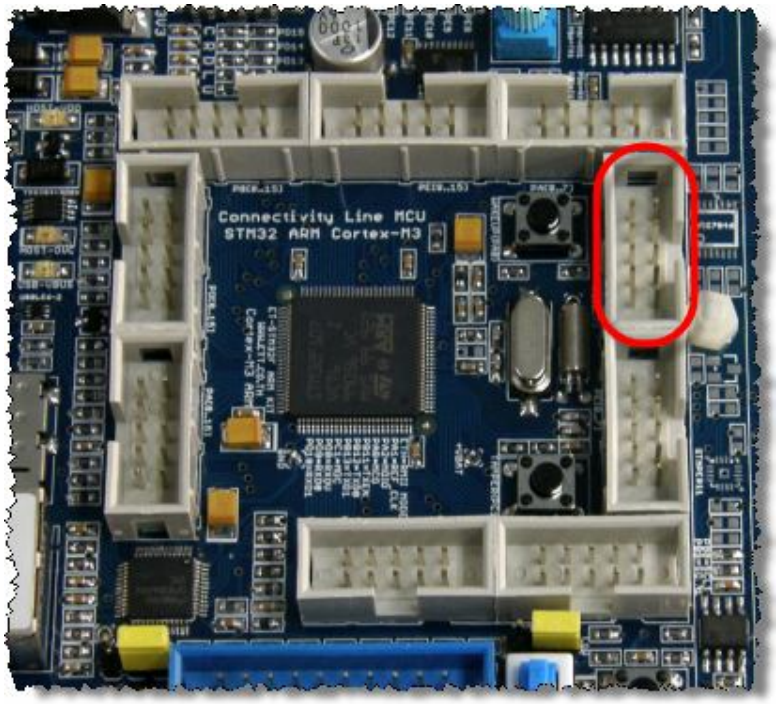
- ขั้วต่อ IDE 10 Pin ของ PB[8..15] มีการจัดเรียงสัญญาณไว้ดังนี้

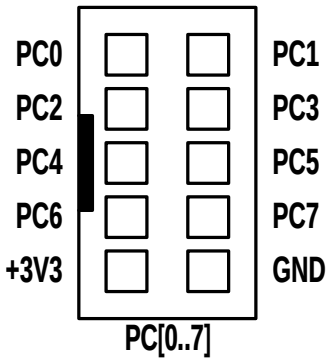




- PB8 ถูกจัดสรรให้ใช้งานในหน้าที่ SCL ของ I2C1 ด้วยแล้ว
- PB9 ถูกจัดสรรให้ใช้งานในหน้าที่ SDA ของ I2C1 ด้วยแล้ว
- PB11 ถูกจัดสรรให้ใช้งานในหน้าที่ TXEN ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PB12 ถูกจัดสรรให้ใช้งานในหน้าที่ TXD0 ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PB13 ถูกจัดสรรให้ใช้งานในหน้าที่ TXD1 ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว

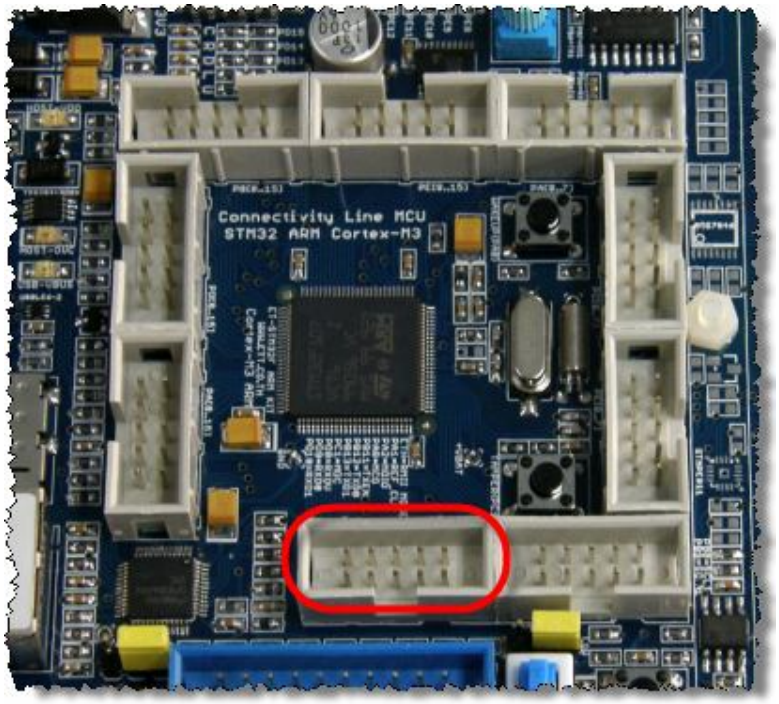
- ขั้วต่อ IDE 10 Pin ของ PC[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

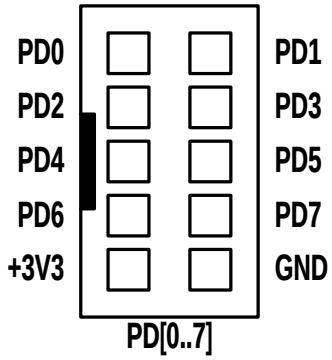




- PC1 ถูกจัดสรรให้ใช้งานในหน้าที่ MDC ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PC4 ถูกจัดสรรให้ใช้งานในหน้าที่ ADC ในการเชื่อมต่อกับ Volume ปรับค่าแล้ว
- PC5 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Output(CS#) ในการเชื่อมต่อกับ SD Card ด้วยแล้ว

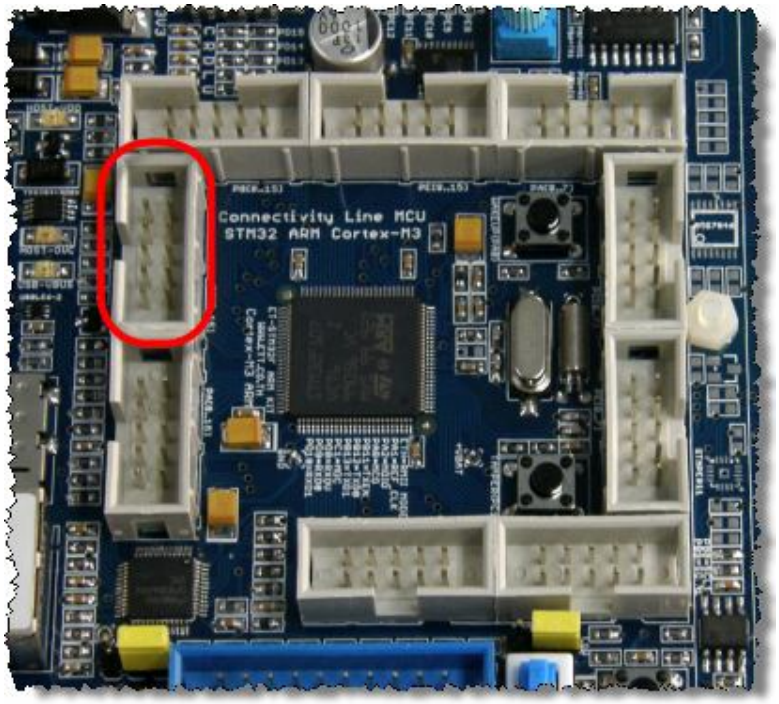
- ขั้วต่อ IDE 10 Pin ของ PD[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

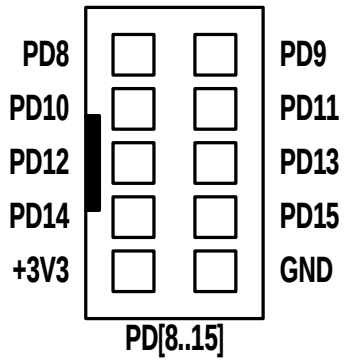




- PD5 ถูกจัดสรรให้ใช้งานในหน้าที่ TXD2 ของ USART2 ด้วยแล้ว
- PD6 ถูกจัดสรรให้ใช้งานในหน้าที่ RXD2 ของ USART2 ด้วยแล้ว
- PD7 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Out สำหรับควบคุม GLCD Backlight LED แล้ว

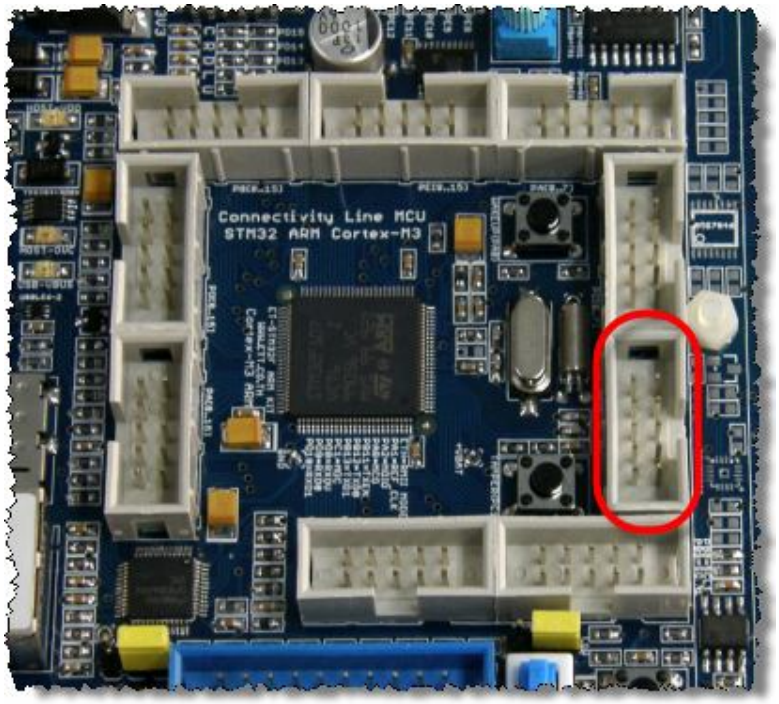
- ขั้วต่อ IDE 10 Pin ของ PD[8..15] มีการจัดเรียงสัญญาณไว้ดังนี้

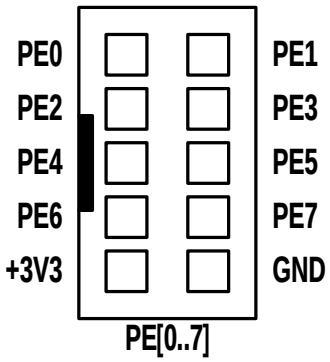




- PD8 ถูกจัดสรรให้ใช้งานในหน้าที่ CRS_DV ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PD9 ถูกจัดสรรให้ใช้งานในหน้าที่ RXD0 ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PD10 ถูกจัดสรรให้ใช้งานในหน้าที่ RXD1 ของ RMII ในการเชื่อมต่อกับ Ethernet ด้วยแล้ว
- PD11 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input ในการเชื่อมต่อกับ Joy Switch ด้วยแล้ว
- PD12 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input ในการเชื่อมต่อกับ Joy Switch ด้วยแล้ว
- PD13 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input ในการเชื่อมต่อกับ Joy Switch ด้วยแล้ว
- PD14 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input ในการเชื่อมต่อกับ Joy Switch ด้วยแล้ว
- PD15 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input ในการเชื่อมต่อกับ Joy Switch ด้วยแล้ว

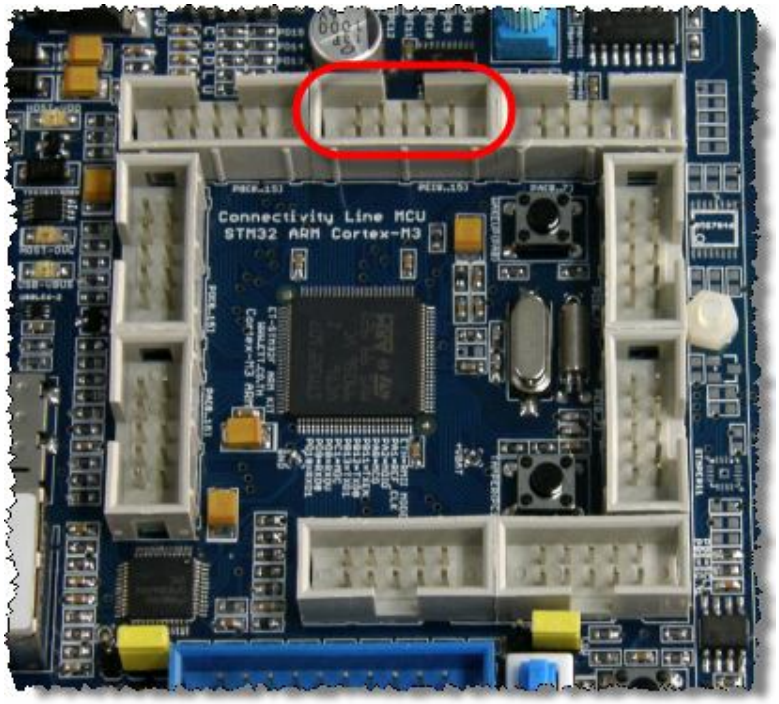
- ขั้วต่อ IDE 10 Pin ของ PE[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

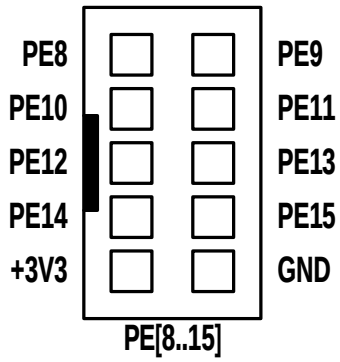




- PE1 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input(Host OVRCR : USB Host Over Current) ในการเชื่อมต่อกับ USB Host ด้วยแล้ว
- PE3 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input(PEN#) ในการเชื่อมต่อกับ Touch Sensor เบอร์ ADS7846 หรือ INT# ในกรณีเชื่อมต่อกับ Touch Sensor เบอร์ STMPE811 ด้วยแล้ว
- PE4 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Input(MISO) ในการเชื่อมต่อกับ ADS7846 ด้วยแล้ว
- PE5 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Output(MOSI) ในการเชื่อมต่อกับ ADS7846 ด้วยแล้ว
- PE6 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Output(CS#) ในการเชื่อมต่อกับ ADS7846 ด้วยแล้ว
- PE7 ถูกจัดสรรให้ใช้งานในหน้าที่ GPIO Output(SCK) ในการเชื่อมต่อกับ ADS7846 ด้วยแล้ว

- ขั้วต่อ IDE 10 Pin ของ PE[8..15] มีการจัดเรียงสัญญาณไว้ดังนี้



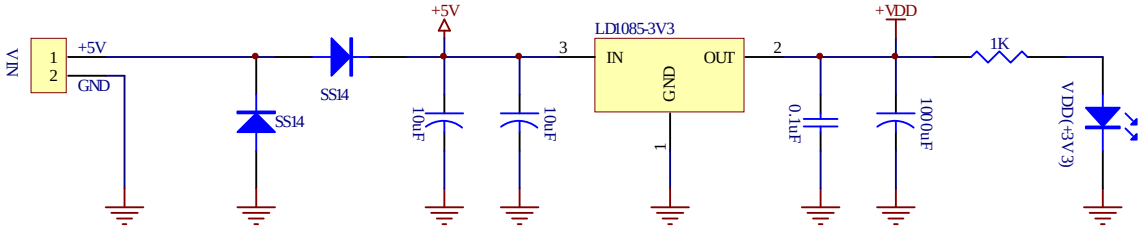


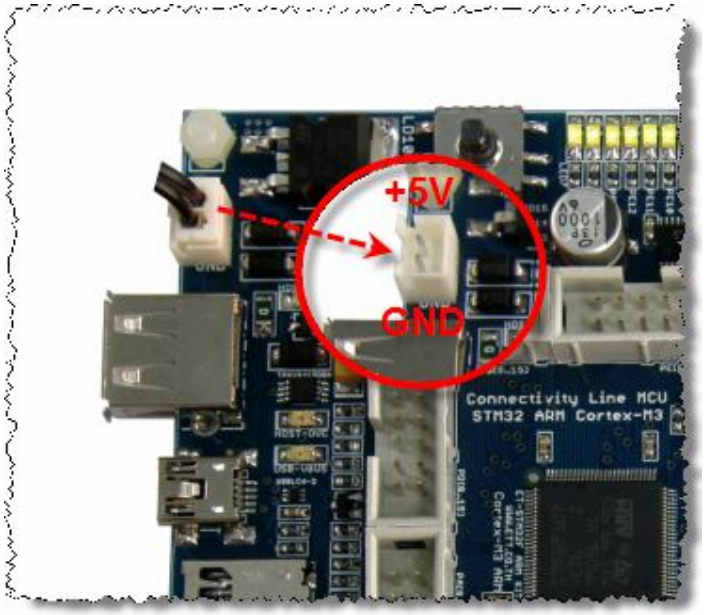
- PE8 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED0 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE9 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED1 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE10 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED2 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE11 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED3 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE12 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED4 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE13 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED5 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE14 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED6 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว
- PE15 ถูกจัดสรรให้ใช้งานในหน้าที่ขับ LED7 เพื่อใช้ทดสอบการทำงานของ Output ด้วยแล้ว

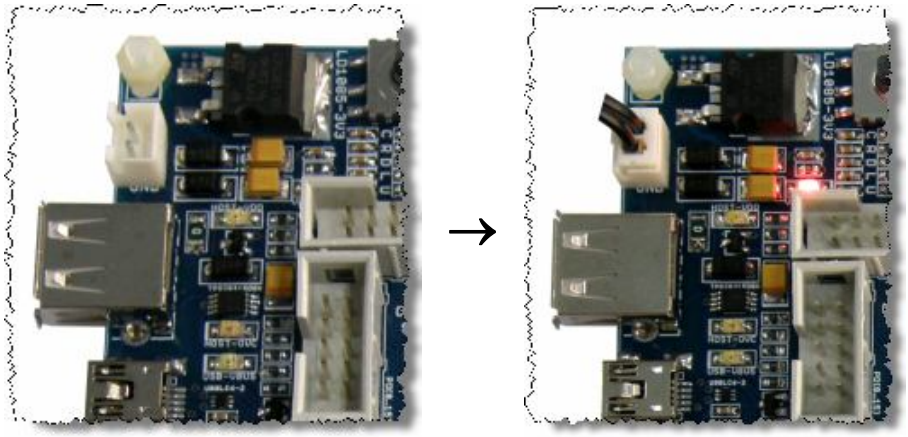
วงจรแหล่งจ่ายไฟ

วงจรแหล่งจ่ายไฟของบอร์ดใช้งานได้กับไฟ DC ขนาด +5V โดยใช้ขั้วต่อแบบ 2 Pin Block ป้องกันการเสียบสายกลับขั้ว พร้อมวงจร Regulate ขนาด +3V3/3A

โดยวงจรภาคแหล่งจ่ายไฟในส่วนที่เป็นวงจร Regulate ขนาด 3.3V นั้นจะจ่ายให้กับ CPU และวงจร I/O ของบอร์ดทั้งหมด ยกเว้น Backlight ของ LCD ซึ่งจะใช้แหล่งจ่ายไฟขนาด +5VDC โดยตรง







สรุป การจัดสรร GPIO ใช้งาน ของบอร์ด ET-STM32F ARM KIT

ตามปรกติแล้ว STM32F107VCT6 จะมีจำนวนขาสัญญาณ GPIO สำหรับใช้งานได้ทั้งหมดจำนวนทั้งสิ้น 5 พอร์ต คือ PA, PB, PC, PD และ PE โดยแต่ละพอร์ตจะมีจำนวน ขาสัญญาณพอร์ตละ 16 บิต หรือ 16 Pin ซึ่งรวมแล้วจะมีจำนวนทั้งสิ้น 80 Pin แต่ในกรณีของบอร์ด ET-STM32F ARM KIT นั้นได้มีการจัดสรร หน้าที่การทำงานของขาสัญญาณ GPIO ต่างๆไว้ให้แล้วจำนวน 60 Pin และ ปล่อยว่างไว้สำหรับให้ผู้ใช้ จัดสรรไปใช้งานได้เองโดยอิสระอีก จำนวน 20 Pin

แต่อย่างไรก็ตามในการนำบอร์ด ET-STM32F ARM KIT ไปประยุกต์ใช้งานนั้น ในบางครั้งผู้ใช้งานอาจไม่ต้องการใช้งานอุปกรณ์และวงจรตามที่ออกแบบไว้ในบอร์ด หรือมีความต้องการใช้งานเพียงบางส่วน ก็สามารถจะเลือกนำขาสัญญาณที่เชื่อมต่อไว้กับอุปกรณ์ส่วนที่ไม่ต้องการใช้งาน เพื่อนำไปดัดแปลงใช้งานอื่นๆได้เองตามความต้องการก็ได้ ซึ่งมีรายละเอียดการจัดสรร GPIO และหน้าที่การทำงานเป็นดังนี้

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

Pin	หน้าที่	อุปกรณ์
PB0	-	-
PB1	-	-
PB2	BOOT1	Jumper BOOT1
PB3	JTAG_TDO	JTAG
PB4	JTAG_TRST	JTAG
PB5	-	-
PB6	USART1_TX	UART1
PB7	USART1_RX	UART1

Pin	หน้าที่	อุปกรณ์
PB8	I2C1_SCL	24C01,STMPE811
PB9	I2C1_SDA	24C01,STMPE811
PB10	-	-
PB11	RMII_TXEN	Ethernet LAN
PB12	RMII_TXD0	Ethernet LAN
PB13	RMII_TXD1	Ethernet LAN
PB14	-	-
PB15	-	-

Pin	หน้าที่	อุปกรณ์
PC0	-	-
PC1	RMII_MDC	Ethernet LAN
PC2	-	-
PC3	-	-
PC4	ADC14	Volume VR1
PC5	GPIO Out	SD Card CD(CS#)
PC6	-	-
PC7	-	-

Pin	หน้าที่	อุปกรณ์
PC8	GPIO Out	GLCD CS#
PC9	HOST_EN	USB HOST/OTG
PC10	SPI3_SCK	GLCD WR#/SCL
PC11	SPI3_MISO	GLCD SDO
PC12	SPI3_MOSI	GLCD SDI
PC13	Tamper	Switch Tamper
PC14	OSC32_IN	RTC X-TAL
PC15	OSC32_OUT	RTC X-TAL

Pin	หน้าที่	อุปกรณ์
PD0	-	-
PD1	-	-
PD2	-	-
PD3	-	-
PD4	-	-
PD5	USART2_TX	UART2(ISP)
PD6	USART2_RX	UART2(ISP)
PD7	GPIO Out	GLCD BL LED

Pin	หน้าที่	อุปกรณ์
PD8	RMII_CRS_DV	Ethernet LAN
PD9	RMII_RXD0	Ethernet LAN
PD10	RMII_RXD1	Ethernet LAN
PD11	GPIO Input	Joy Switch Up
PD12	GPIO Input	Joy Switch Left
PD13	GPIO Input	Joy Switch Down
PD14	GPIO Input	Joy Switch Right
PD15	GPIO Input	Joy Switch Select

Pin	หน้าที่	อุปกรณ์
PE0	-	-
PE1	USB_OVRCR	USB HOST/OTG
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

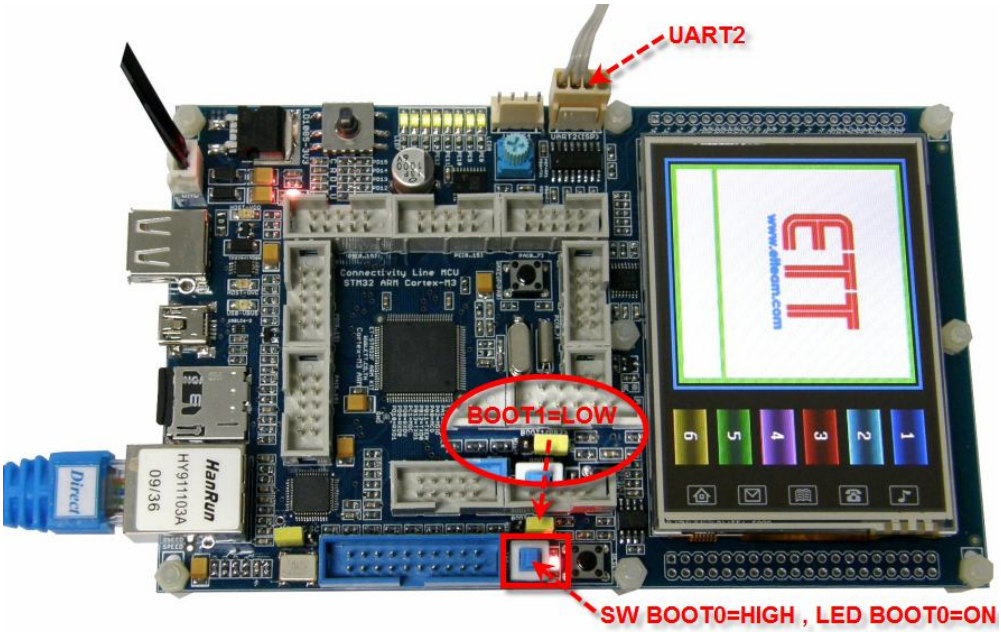
Pin	หน้าที่	อุปกรณ์
PE8	GPIO Out	LED0
PE9	GPIO Out	LED1
PE10	GPIO Out	LED2
PE11	GPIO Out	LED3
PE12	GPIO Out	LED4
PE13	GPIO Out	LED5
PE14	GPIO Out	LED6
PE15	GPIO Out	LED7

การ Download Hex file ให้กับ MCU ของบอร์ด

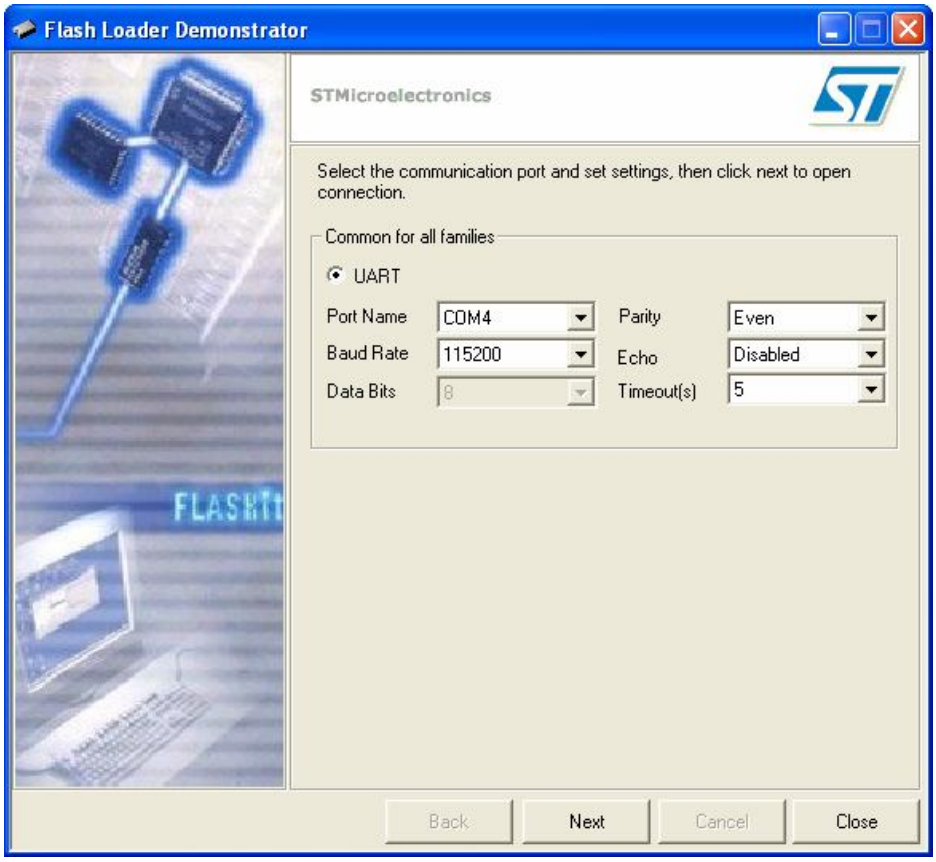
การ Download Hex File ให้กับหน่วยความจำ Flash ของ MCU ในบอร์ดนั้น จะใช้โปรแกรมชื่อ Flash Loader ของ “ST Microelectronics” ซึ่งจะติดต่อกับ MCU ผ่าน Serial Port ของคอมพิวเตอร์ PC โดยโปรแกรมดังกล่าวสามารถดาวน์โหลดฟรีได้ที่ http://ww.st.com

ขั้นตอนการ Download HEX File ให้กับ MCU

1. ต่อสายสัญญาณ RS232 ระหว่างพอร์ตสื่อสารอนุกรม RS232 ของ PC และบอร์ด UART2
2. จ่ายไฟเลี้ยงวงจรให้กับบอร์ด ซึ่งจะสังเกตเห็น LED PWR ติดสว่างให้เห็น
3. เลือกกำหนด Jumper BOOT1 ไว้ทางด้าน Low
4. กดเลือกสวิตช์ BOOT0 เป็น High โดยจะเห็น LED BOOT0 ติดสว่างให้เห็น

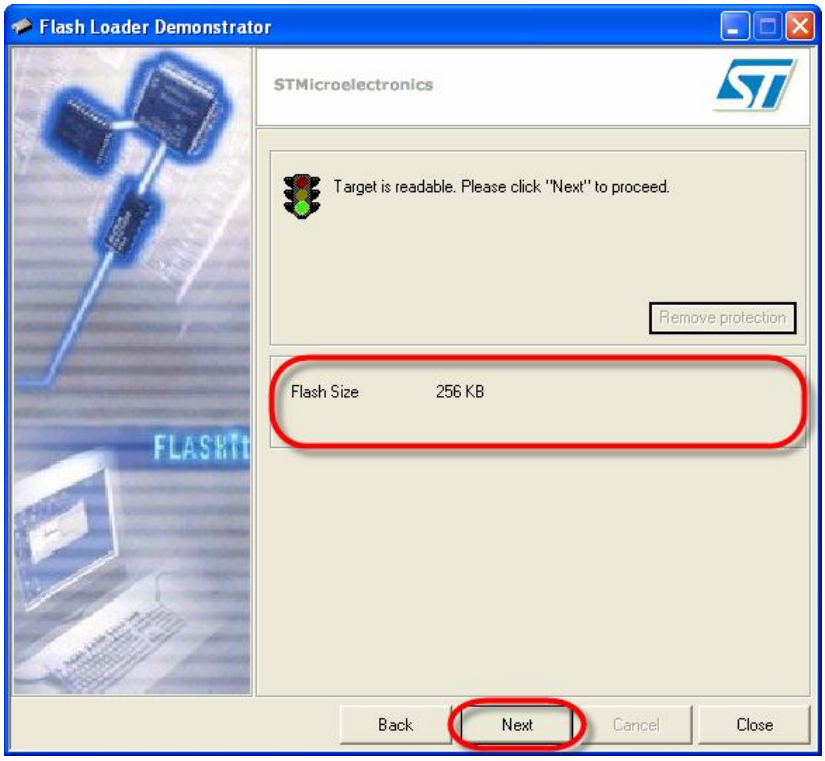


5. สั่ง Run โปรแกรม Flash Loader ซึ่งถ้าเป็น Version 2.10 จะได้ผลดังรูป

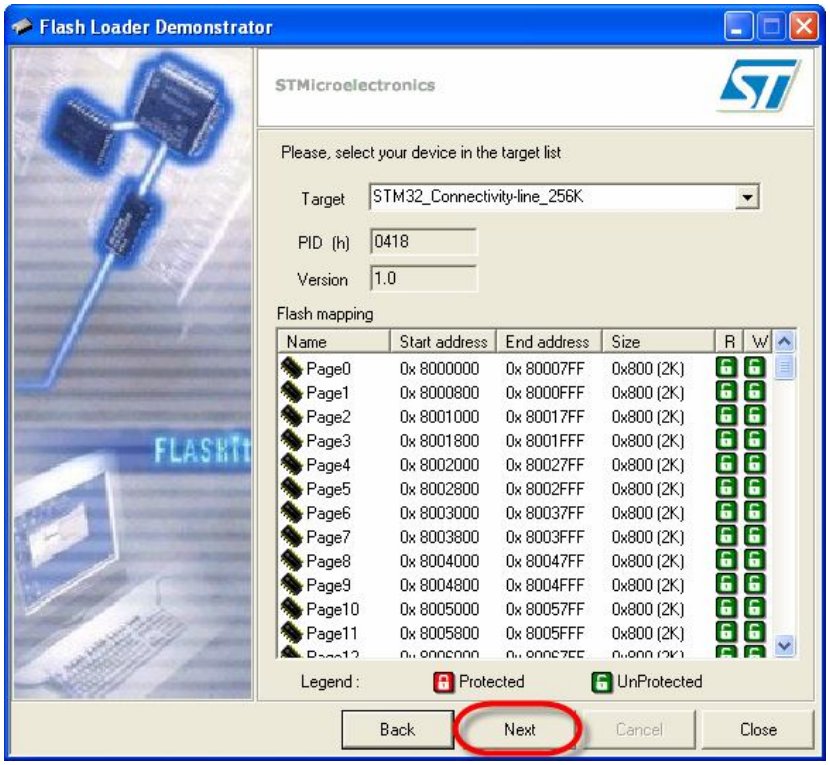


6. เริ่มต้นกำหนดค่าตัวเลือกต่างๆให้กับโปรแกรมตามต้องการ ซึ่งในกรณีนี้ใช้กับ STM32F107VCT6 ของบอร์ด ET-STM32F ARM KIT ของ อีทีที ให้เลือกกำหนดค่าต่างๆให้โปรแกรมดังนี้
   - เลือก COM Port ให้ตรงกับหมายเลข COM Port ที่ใช้งานจริง (ในตัวอย่างใช้ COM4)
   - ตั้งค่า Baud Rate เป็น 115200
   - เลือก Parity เป็น Even
   - เลือก Echo เป็น Disable
   - กำหนดค่า Timeout เป็น 5 วินาที
   - ให้กดสวิตช์ Reset ที่บอร์ด “ET-STM32F ARM KIT” เพื่อทำการ Reset ให้ MCU ทำงานใน Boot Loader โดยให้ตรวจสอบเงื่อนไขตามขั้นตอนดังต่อไปนี้
     - เลือก Jumper BOOT1 = Low
     - เลือก Switch BOOT0 ไว้ทางด้าน High(LED BOOT0 ติดสว่าง)
     - ต่อสายสัญญาณ RS232 เข้ากับ UART2(ISP) ของบอร์ดให้เรียบร้อย
     - กดสวิตช์ Reset

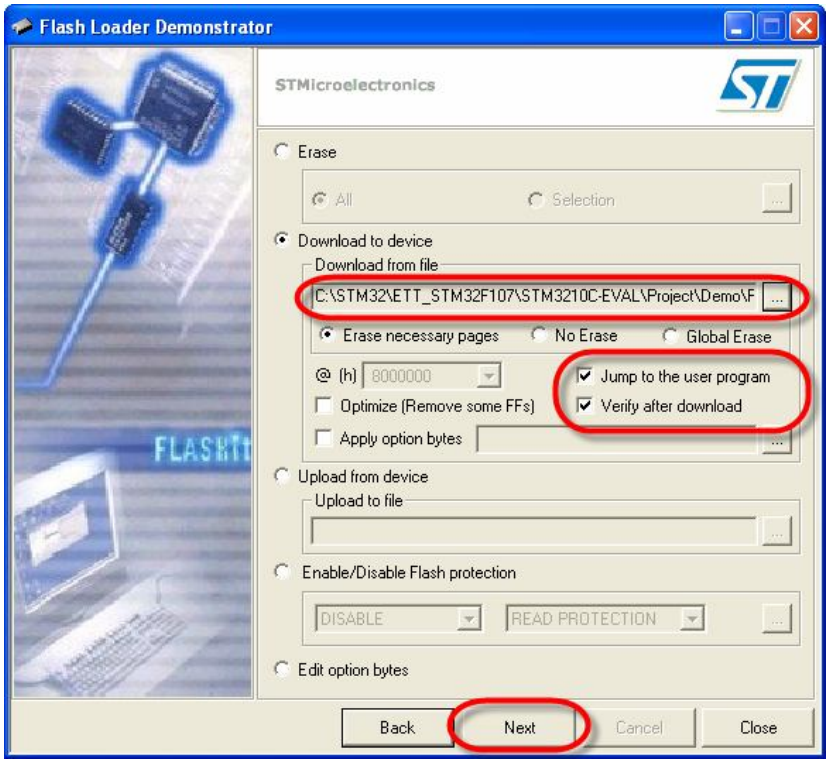
7. ถ้าทุกอย่างถูกต้องจะได้ผลดังรูป จากนั้นให้เลือก Next เพื่อไปยังขั้นตอนต่อไป ถ้าเกิดข้อผิดพลาดให้ลองตรวจสอบเงื่อนไขตามข้อ 6 และ กดสวิตช์ Reset ซ้ำใหม่



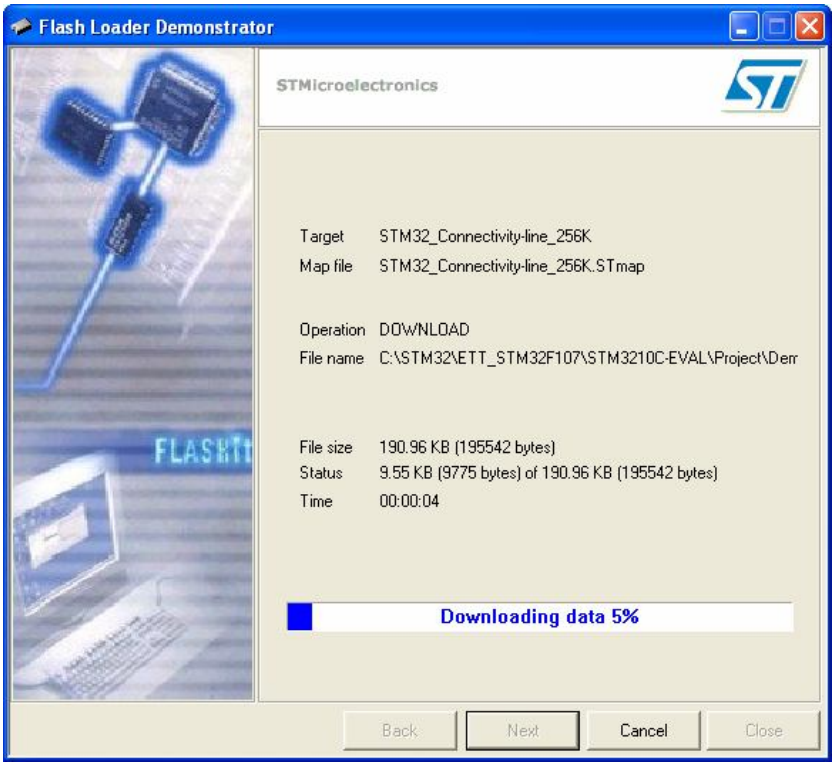
↓

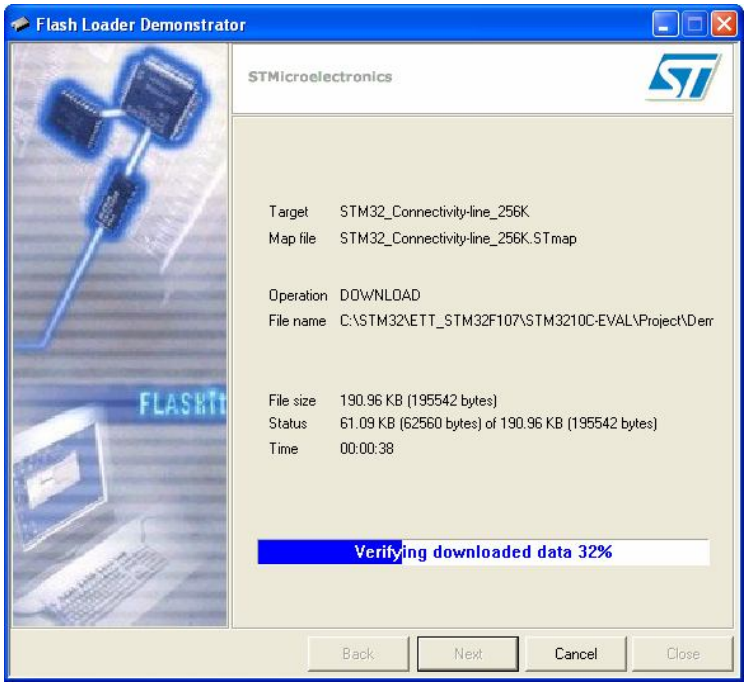


8. เลือกกำหนด Hex File ที่ต้องการ Download ให้กับบอร์ด แล้วเลือก Next ดังรูป

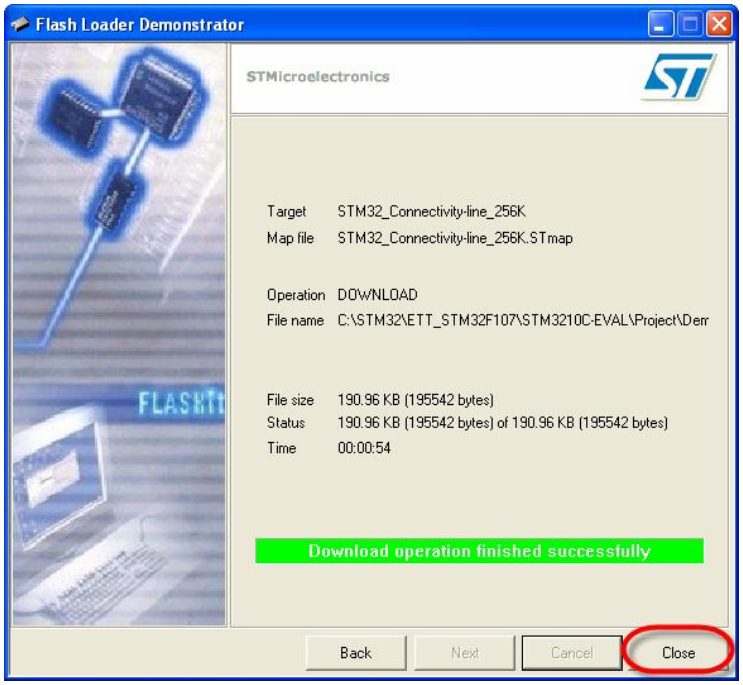


9. เมื่อโปรแกรม Flash Loader เริ่มต้นทำการ Download ข้อมูลให้กับ MCU ทันที โดยสังเกตการทำงานที่ Status bar โดยในขั้นตอนนี้ให้รอจนกว่าการทำงานของโปรแกรมจะเสร็จสมบูรณ์

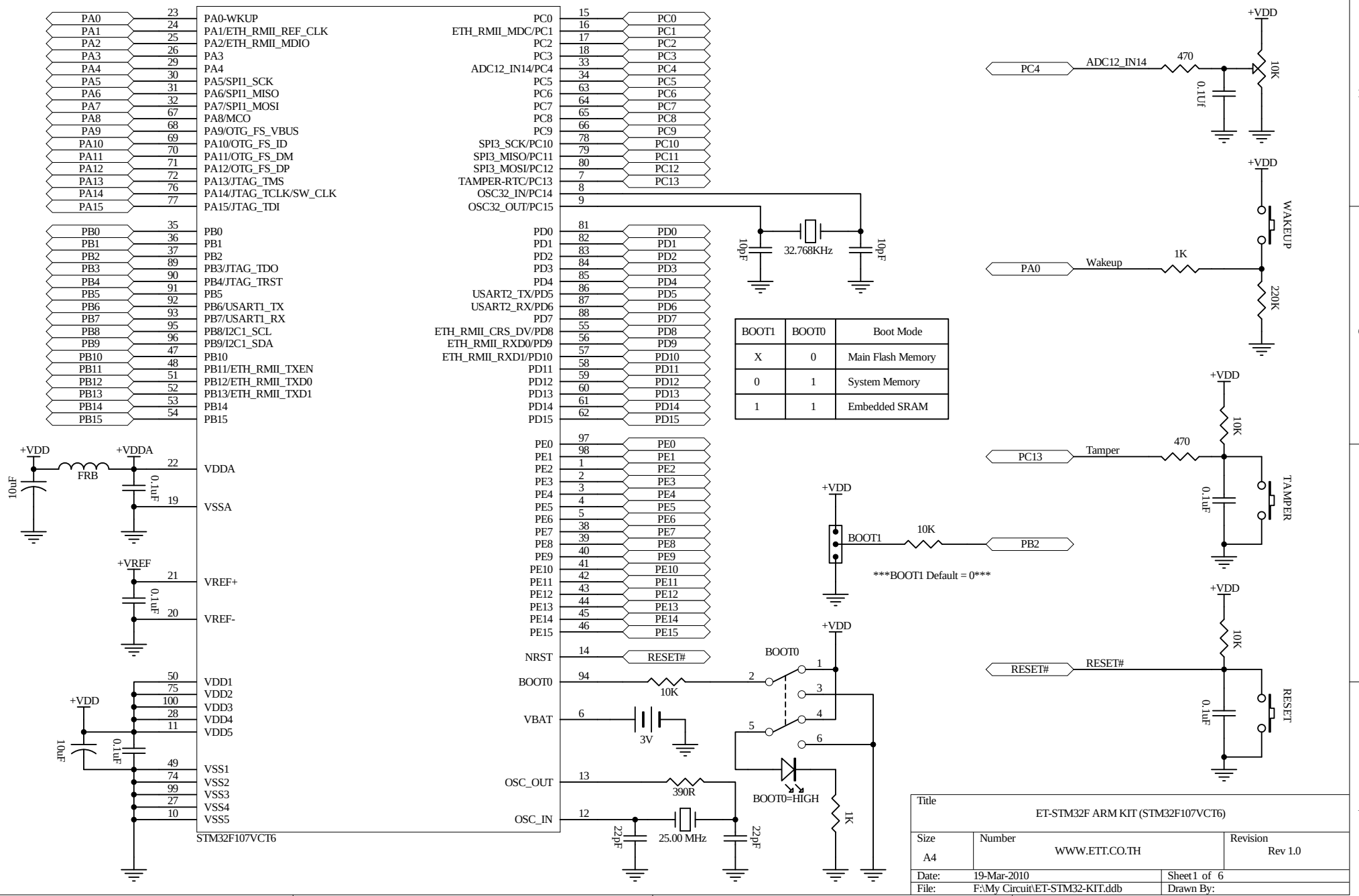


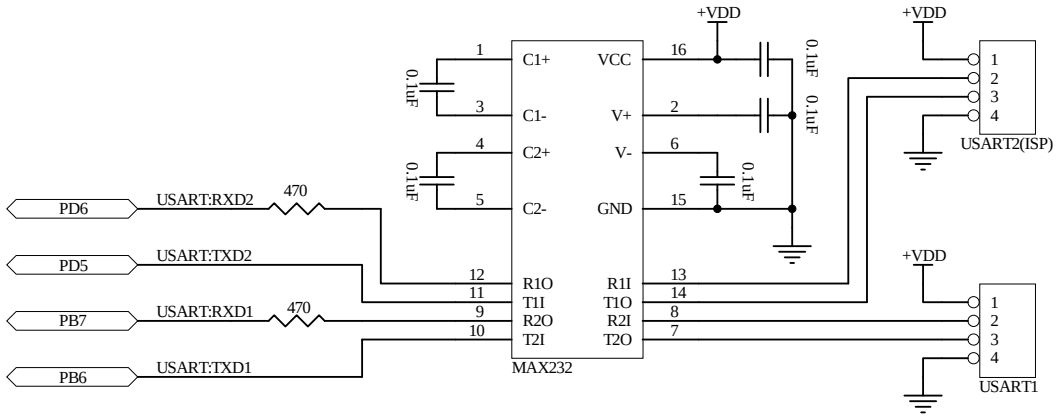


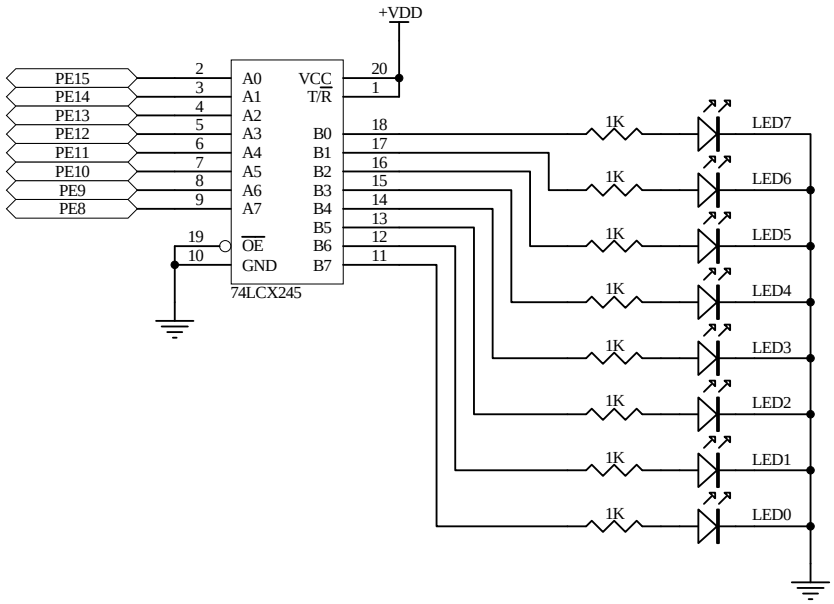
↓

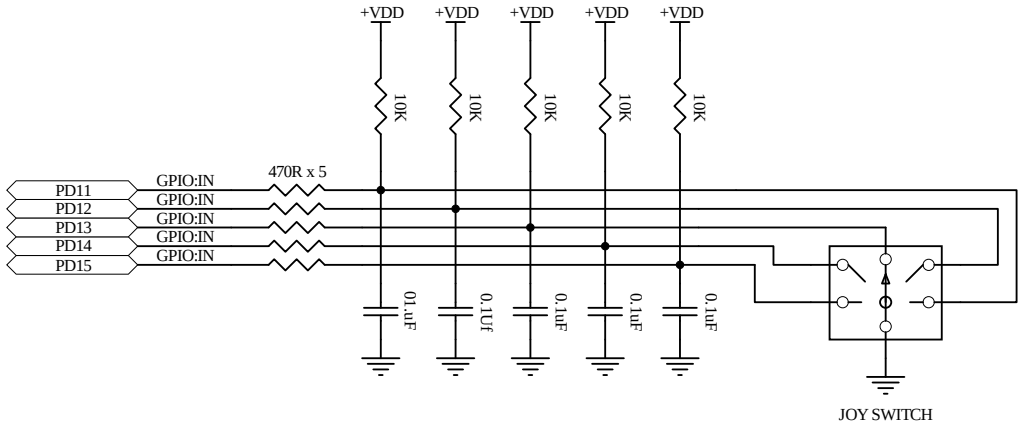


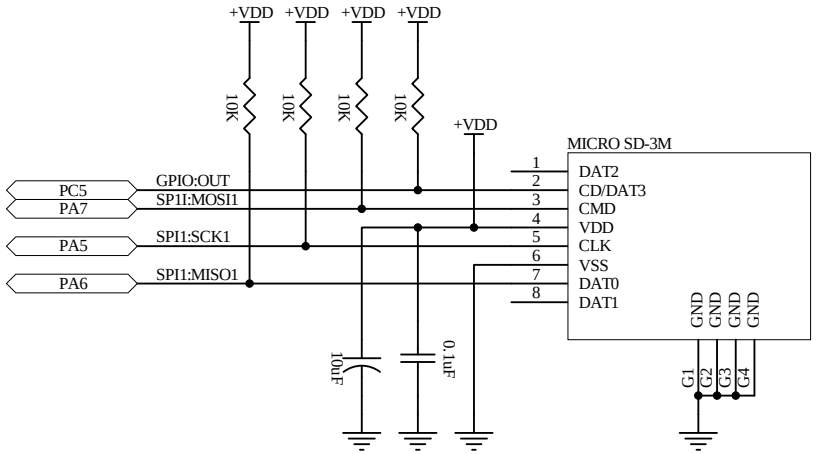
เมื่อทำงานของโปรแกรมเสร็จเรียบร้อยแล้ว ให้กดสวิตช์ BOOT0 กลับมาอยู่ในตำแหน่ง Low โดยจะสังเกตเห็น LED BOOT0 ดับ แล้วจึงกดสวิตช์ Reset ที่บอร์ด ซึ่ง MCU จะเริ่มต้นทำงานตามโปรแกรมที่สั่ง Download ให้ทันที


PA0 23 PA0-WKUP
PA1 24 PA1/ETH_RMII_REF_CLK
PA2 25 PA2/ETH_RMII_MDIO
PA3 26 PA3
PA4 29 PA4
PA5 30 PA5/SPI1_SCK
PA6 31 PA6/SPI1_MISO
PA7 32 PA7/SPI1_MOSI
PA8 67 PA8/MCO
PA9 68 PA9/OTG_FS_VBUS
PA10 69 PA10/OTG_FS_ID
PA11 70 PA11/OTG_FS_DM
PA12 71 PA12/OTG_FS_DP
PA13 72 PA13/JTAG_TMS
PA14 76 PA14/JTAG_TCLK/SW_CLK
PA15 77 PA15/JTAG_TDI
PB0 35 PB0
PB1 36 PB1
PB2 37 PB2
PB3 89 PB3/JTAG_TDO
PB4 90 PB4/JTAG_TRST
PB5 91 PB5
PB6 92 PB6/USART1_TX
PB7 93 PB7/USART1_RX
PB8 95 PB8/I2C1_SCL
PB9 96 PB9/I2C1_SDA
PB10 47 PB10
PB11 48 PB11/ETH_RMII_TXEN
PB12 51 PB12/ETH_RMII_TXD0
PB13 52 PB13/ETH_RMII_TXD1
PB14 53 PB14
PB15 54 PB15
PC0 15 PC0
ETH_RMII_MDC/PC1 16 PC1
PC2 17 PC2
PC3 18 PC3
ADC12_IN14/PC4 33 PC4
PC5 34 PC5
PC6 63 PC6
PC7 64 PC7
PC8 65 PC8
PC9 66 PC9
SPI3_SCK/PC10 78 PC10
SPI3_MISO/PC11 79 PC11
SPI3_MOSI/PC12 80 PC12
TAMPER-RTC/PC13 7 PC13
OSC32_IN/PC14 8
OSC32_OUT/PC15 9
PD0 81 PD0
PD1 82 PD1
PD2 83 PD2
PD3 84 PD3
PD4 85 PD4
USART2_TX/PD5 86 PD5
USART2_RX/PD6 87 PD6
PD7 88 PD7
ETH_RMII_CRS_DV/PD8 55 PD8
ETH_RMII_RXD0/PD9 56 PD9
ETH_RMII_RXD1/PD10 57 PD10
PD11 58 PD11
PD12 59 PD12
PD13 60 PD13
PD14 61 PD14
PD15 62 PD15
PE0 97 PE0
PE1 98 PE1
PE2 1 PE2
PE3 2 PE3
PE4 3 PE4
PE5 4 PE5
PE6 5 PE6
PE7 38 PE7
PE8 39 PE8
PE9 40 PE9
PE10 41 PE10
PE11 42 PE11
PE12 43 PE12
PE13 44 PE13
PE14 45 PE14
PE15 46 PE15
NRST 14 RESET#
BOOT0 94
VBAT 6
OSC_OUT 13
OSC_IN 12
VDD1 50
VDD2 75
VDD3 100
VDD4 28
VDD5 11
VSS1 49
VSS2 74
VSS3 99
VSS4 27
VSS5 10
VDDA 22
VSSA 19
VREF+ 21
VREF- 20
STM32F107VCT6
+VDD
+VDDA
FRB
10uF
0.1uF
+VREF
BOOT0
10K
3V
390R
BOOT0=HIGH
1K
22pF
25.00 MHz
32.768KHz
BOOT1
PB2
BOOT1 Default = 0
PC4
ADC12_IN14
470
0.1Uf
WAKEUP
PA0
Wakeup
220K
PC13
Tamper
TAMPER
RESET#
RESET
BOOT1 BOOT0 Boot Mode
X 0 Main Flash Memory
0 1 System Memory
1 1 Embedded SRAM
Title ET-STM32F ARM KIT (STM32F107VCT6)
Size A4
Number WWW.ETT.CO.TH
Revision Rev 1.0
Date: 19-Mar-2010
Sheet1 of 6
File: F:\My Circuit\ET-STM32-KIT.ddb
Drawn By:

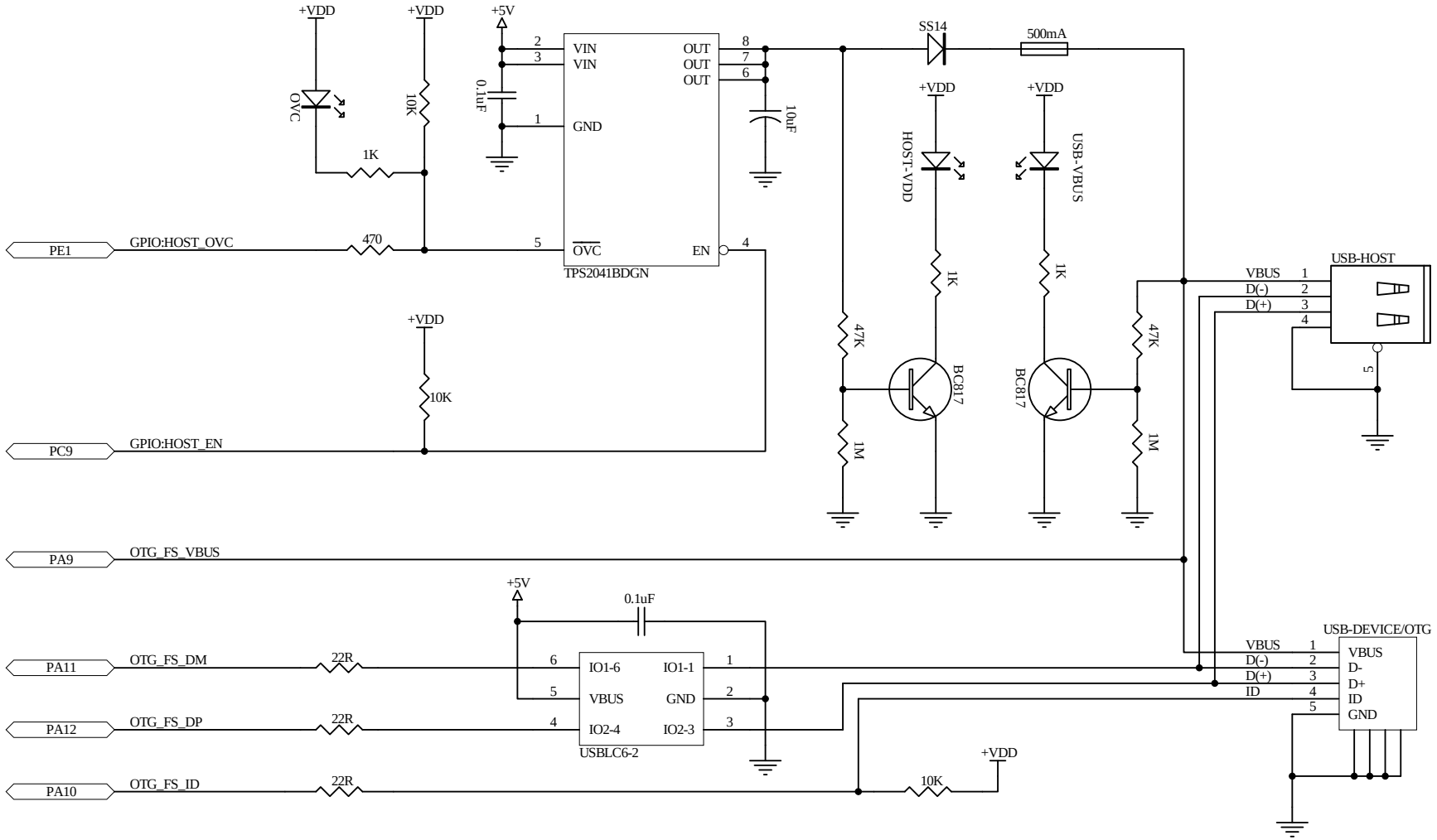




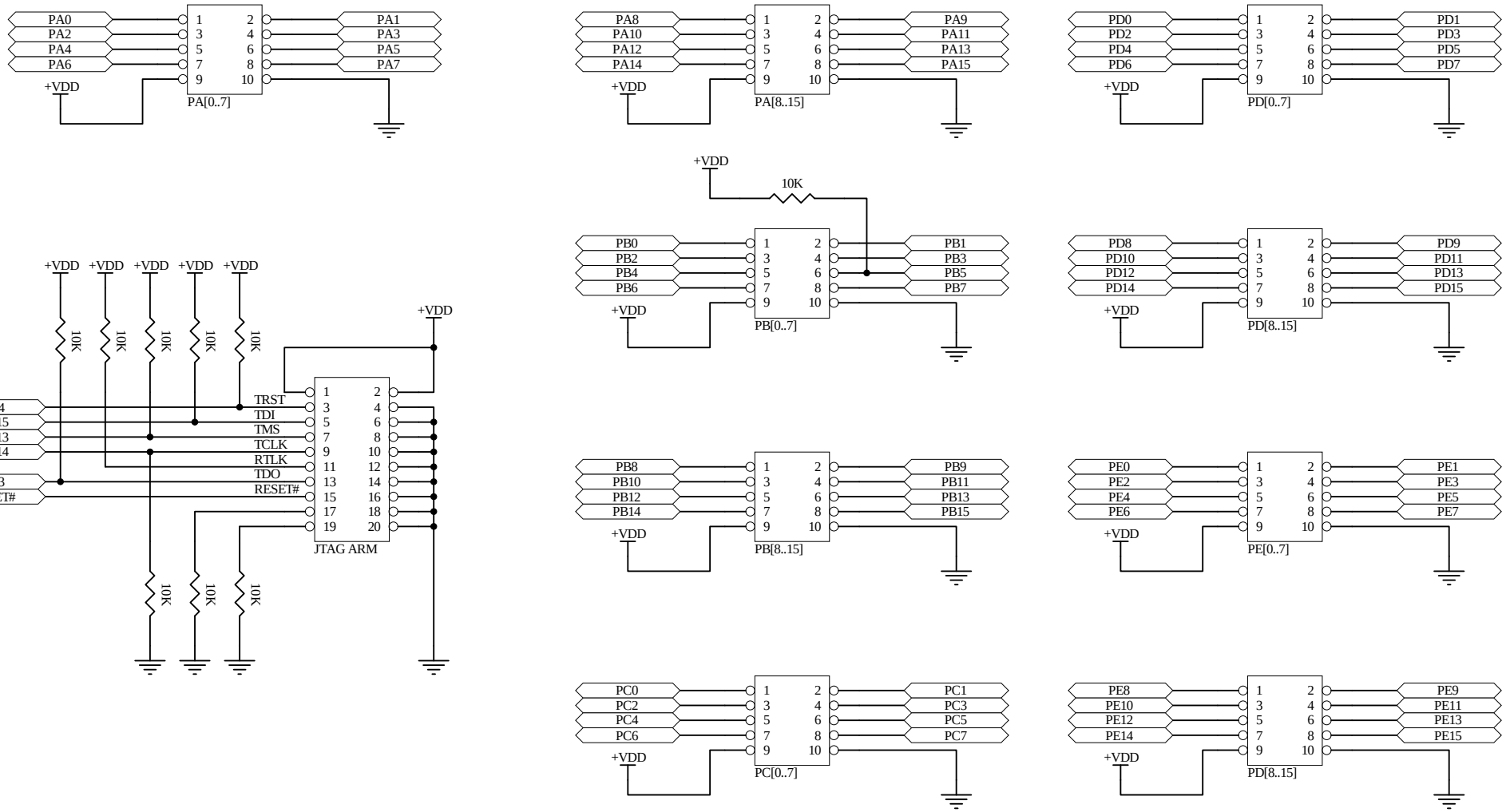




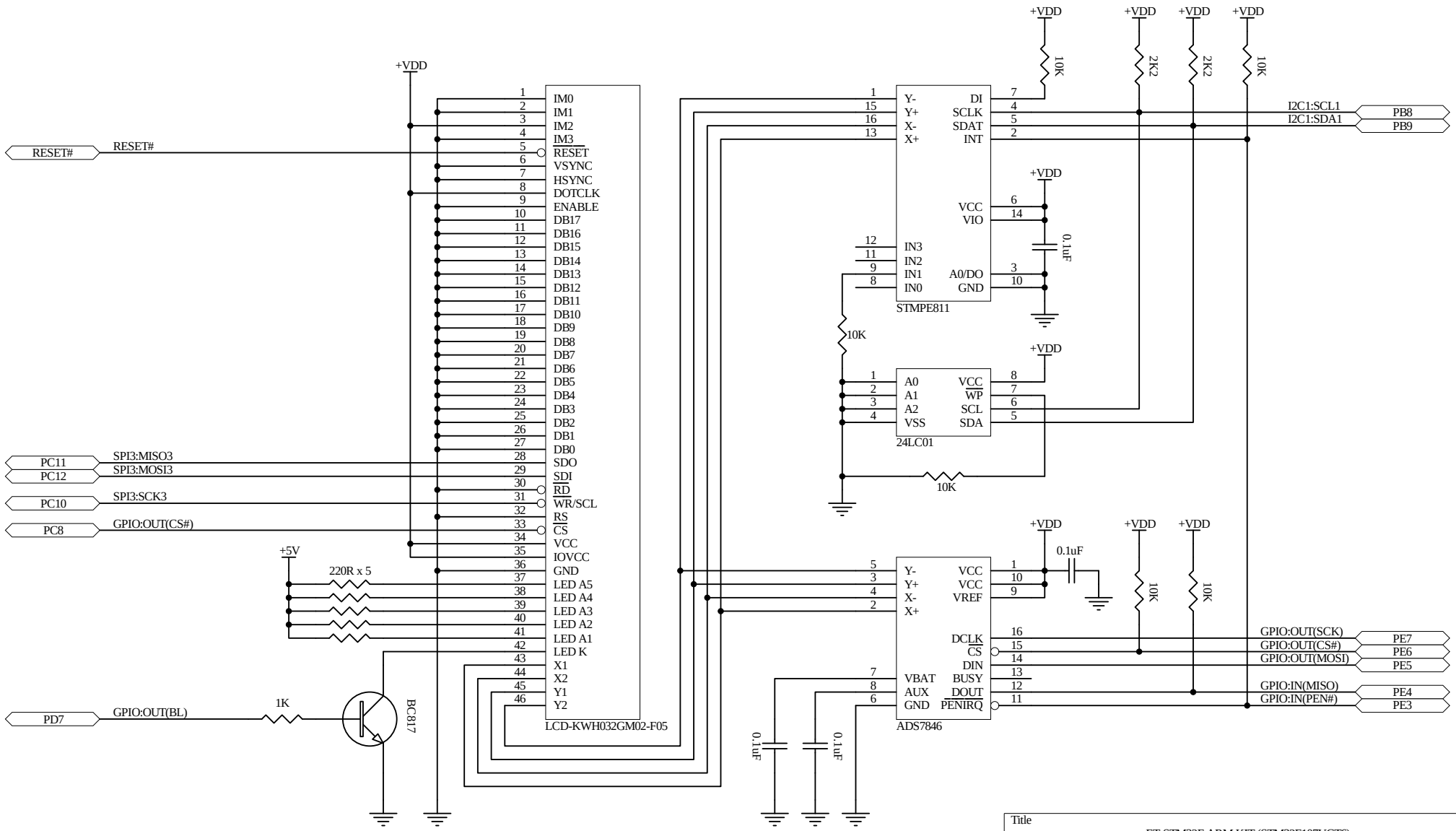
Title: ET-STM32F ARM KIT (STM32F107VCT6)		
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 17-Mar-2010		Sheet2 of 6
File: F:\My Circuit\ET-STM32-KIT.ddb		Drawn By:



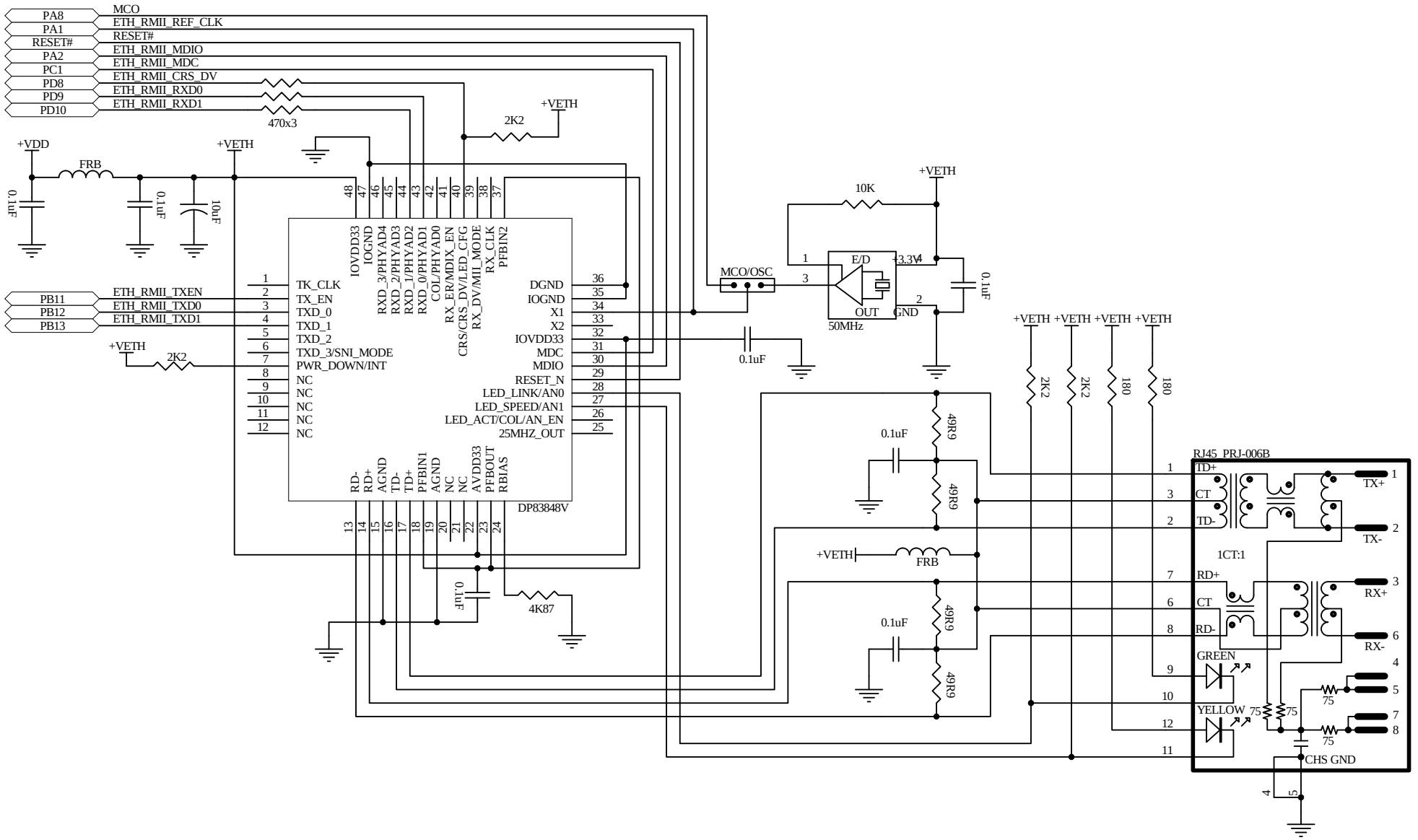
Title	ET-STM32F ARM KIT (STM32F107VCT6)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 10-May-2010		Sheet3 of 6
File: F:\My Circuit\ET-STM32-KIT.ddb		Drawn By:



Title	ET-STM32F ARM KIT (STM32F107VCT6)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 17-Mar-2010		Sheet4 of 6
File: F:\My Circuit\ET-STM32-KIT.ddb		Drawn By:



Title	ET-STM32F ARM KIT (STM32F107VCT6)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 19-Mar-2010		Sheet5 of 6
File: F:\My Circuit\ET-STM32-KIT.ddb		Drawn By:


PA8
MCO
PA1
ETH_RMII_REF_CLK
RESET#
RESET#
PA2
ETH_RMII_MDIO
PC1
ETH_RMII_MDC
PD8
ETH_RMII_CRS_DV
PD9
ETH_RMII_RXD0
PD10
ETH_RMII_RXD1
470x3
+VETH
2K2
+VDD
FRB
+VETH
0.1uF
0.1uF
10uF
PB11
ETH_RMII_TXEN
PB12
ETH_RMII_TXD0
PB13
ETH_RMII_TXD1
+VETH
2K2
TK_CLK
TX_EN
TXD_0
TXD_1
TXD_2
TXD_3/SNI_MODE
PWR_DOWN/INT
NC
NC
NC
NC
NC
IOVDD33
IOGND
RXD_3/PHYAD4
RXD_2/PHYAD3
RXD_1/PHYAD2
RXD_0/PHYAD1
COL/PHYAD0
RX_ER/MDIX_EN
CRS/CRS_DV/LED_CFG
RX_DV/MII_MODE
RX_CLK
PFBIN2
DGND
IOGND
X1
X2
IOVDD33
MDC
MDIO
RESET_N
LED_LINK/AN0
LED_SPEED/AN1
LED_ACT/COL/AN_EN
25MHZ_OUT
RD-
RD+
AGND
TD-
TD+
PFBIN1
AGND
NC
NC
AVDD33
PFBOUT
RBIAS
DP83848V
0.1uF
4K87
MCO/OSC
+VETH
10K
E/D
OUT
GND
+3.3V
50MHz
0.1uF
0.1uF
+VETH
FRB
0.1uF
49R9
49R9
0.1uF
49R9
49R9
+VETH +VETH +VETH +VETH
2K2
2K2
180
180
RJ45_PRJ-006B
TD+
CT
TD-
RD+
CT
RD-
GREEN
YELLOW
75
75
75
75
CHS GND
TX+
TX-
1CT:1
RX+
RX-
Title
ET-STM32F ARM KIT (STM32F107VCT6)
Size
A4
Number
WWW.ETT.CO.TH
Revision
Rev 1.0
Date:
17-Mar-2010
Sheet6 of 6
File:
F:\My Circuit\ET-STM32-KIT.ddb
Drawn By: