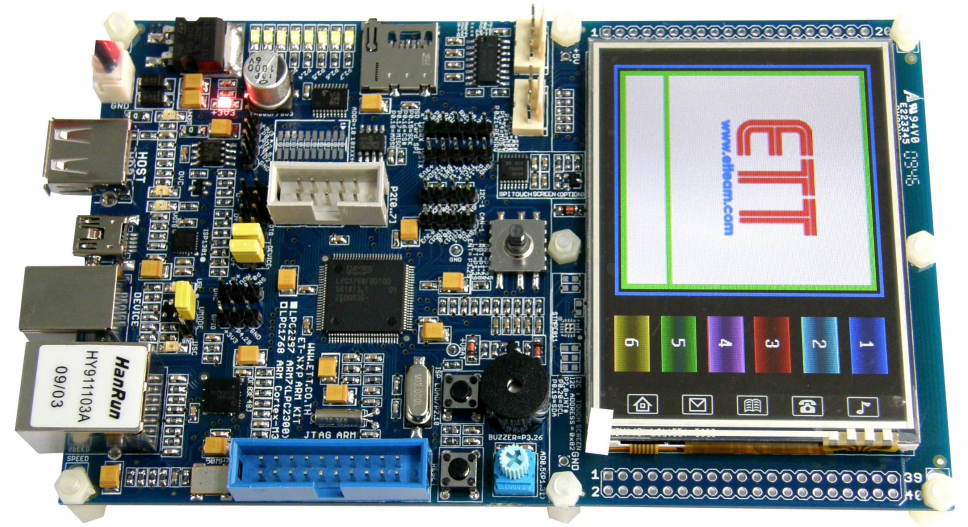
ET-NXP ARM KIT (LPC1768)

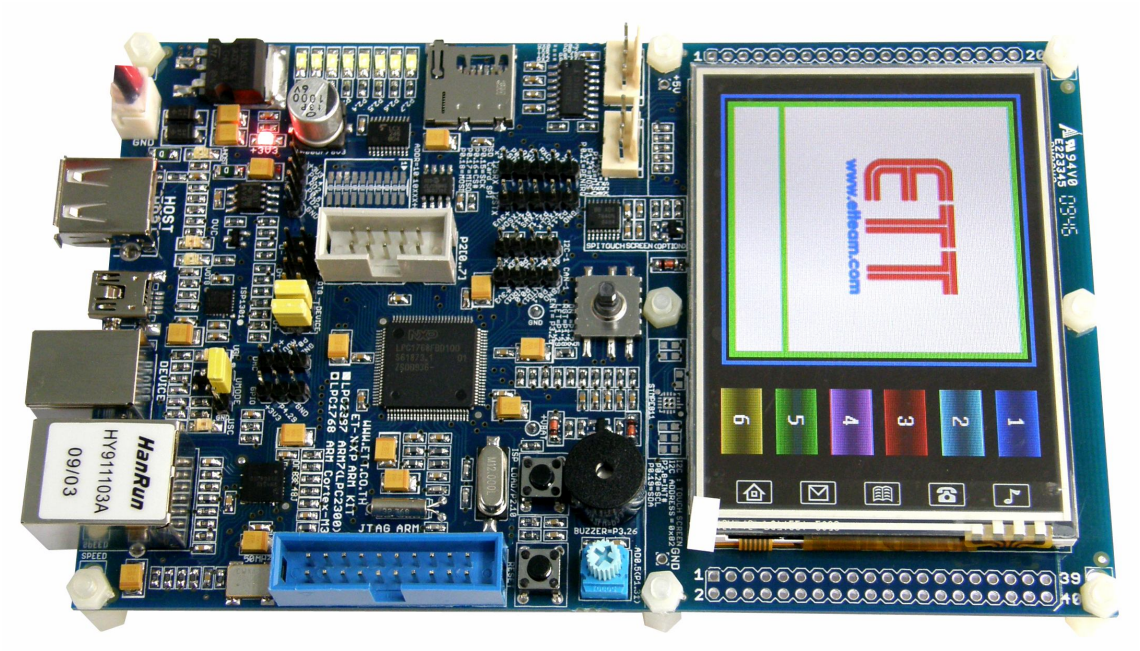


ET-NXP ARM KIT (LPC1768) เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล ARM Cortex M3 Core ซึ่งเลือกใช้ไมโครคอนโทรเลอร์ 32-Bit ขนาด 100 Pin(LQFP) เบอร์ LPC1768 ของ NXP โดยระบบทรัพยากรต่างๆที่อยู่ในตัวของ LPC1768 ถือว่ามีความสมบูรณ์แบบมากพอสมควร เหมาะต่อการนำไปศึกษาเรียนรู้เป็นอย่างยิ่ง เพราะถ้าสามารถศึกษาการใช้งานทรัพยากรต่างๆภายใน MCU ตัวนี้ได้อย่างเข้าใจแล้ว จะสามารถนำไปดัดแปลงแก้ไข และพัฒนาต่อยอด สร้างเป็น Application ในรูปแบบต่างๆได้มากมาย เนื่องจากระบบฮาร์ดแวร์ของ LPC1768 ได้รวบรวมเอาอุปกรณ์ที่จำเป็นต่างๆต่อการใช้งาน บรรจุไว้ภายในโครงสร้างของ MCU เพียงตัวเดียว ไม่ว่าจะเป็น ระบบ USB, Ethernet, การ์ดหน่วยความจำแบบ SD Card, ADC, DAC, Timer/Counter, PWM, Capture, I2C, SPI, UART,..ฯลฯ

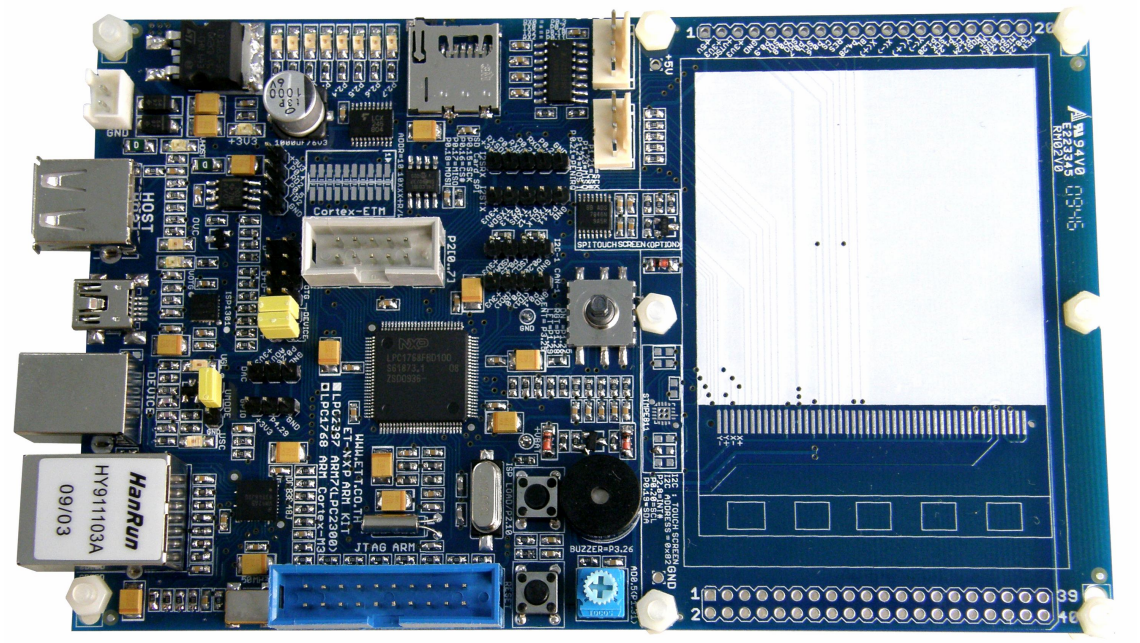
ดังนั้นทางทีมงาน อีทีที จึงได้ศึกษาค้นคว้าถึงรายละเอียดต่างๆของ LPC1768 และนำมาออกแบบสร้างเป็นบอร์ด ไมโครคอนโทรลเลอร์ รุ่น “ET-NXP ARM KIT(LPC1768)” ขึ้นมา เพื่อหวังให้ผู้ใช้ที่สนใจหาซื้อไปศึกษา ทดลอง เรียนรู้ และพัฒนาต่อยอดใช้งานได้ตามความต้องการ ภายใต้งบประมาณที่สมเหตุสมผล โดยจุดประสงค์หลักของการออกแบบบอร์ด ET-NXP ARM KIT(LPC1768) นั้น จะรองรับทั้งกลุ่มผู้ใช้ที่ต้องการ ศึกษา 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, Mini-Speaker หรือ Buzzer สำหรับสร้างเสียง Beep ต่างๆ นอกจากนี้แล้วยังมีการจัดเตรียมอุปกรณ์ระดับสูงไว้รองรับการใช้งานด้วยไม่ว่าจะเป็น พอร์ต เชื่อมต่อ USB Device/Host/OTG,SD Card, พอร์ตเชื่อมต่อ Ethernet LAN, Graphic LCD, RS232 นอกเหนือจากนี้แล้วยังมี GPIO ต่างๆที่ว่างไว้ให้ผู้ใช้ออกแบบใช้งานร่วมกับอุปกรณ์อื่นๆได้เองตามความเหมาะสมอีกด้วย

คุณสมบัติของบอร์ด ET-NXP ARM KIT (LPC1768)

1. ใช้ MCU ตระกูล ARM Cortex M3 เบอร์ LPC1768 ของ NXP ซึ่งเป็น MCU ขนาด 32Bit
2. ภายใน MCU มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 512KB, Static RAM ขนาด 64KB
3. ใช้ Crystal 12.00 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 100 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) และ In-Application Programming (IAP) ผ่านทาง On-Chip Boot-Loader Software ทางพอร์ต UART0 (RS232)
6. มีวงจรเชื่อมต่อกับ JTAG ARM ขนาด 20 Pin มาตรฐาน เพื่อทำการ Debug แบบ Real Time ได้
7. Power Supply ใช้แรงดันไฟฟ้า +5VDC สามารถใช้แหล่งจ่ายได้จาก 2 แหล่ง คือจากภายนอกโดยใช้ขั้วต่อแบบ 2 Pin Connector และ จากขั้ว USB Device พร้อมวงจร Regulate +3V3/3A
8. มีวงจร USB Device 2.0 แบบ Full Speed ภายในตัว (USB Function มี 32 End Point)
9. มีวงจร USB Host พร้อมวงจร Over Current Protection
10. มีวงจร USB OTG โดยใช้ ISP1301 เป็น OTG(On-The-Go) Transceiver
11. มีวงจรเชื่อมต่อ Ethernet LAN 10/100Mb โดยใช้ขั้วต่อแบบ RJ45 มาตรฐาน จำนวน 1 ช่อง
12. มีวงจรเชื่อมต่อการ์ดหน่วยความจำแบบ SD Card(Micro SD) เชื่อมต่อแบบ SPI จำนวน 1 ช่อง
13. มีวงจรสื่อสาร RS232 โดยใช้ขั้วต่อแบบ 4-PIN มาตรฐาน ETT จำนวน 2 ช่อง
14. มีวงจรเชื่อมต่อ TFT LCD Color ขนาด 320x240 Pixel (3.2นิ้ว) พร้อม Touch Screen
15. มีวงจร Push Button Switch จำนวน 1 ชุด พร้อมสวิตช์ RESET
16. มีวงจร Joy Switch แบบ 5 ทิศทาง สำหรับใช้งาน จำนวน 1 ชุด
17. มีวงจร LED แสดงสถานะเพื่อทดลอง Output จำนวน 8 ชุด พร้อมวงจร Buffer
18. มีวงจร สร้างแรงดัน 0-3V3 โดยใช้ตัวต้านทานปรับค่าได้สำหรับทดสอบ A/D จำนวน 1 ชุด
19. มีวงจรกำเนิดและขับเสียง Beep โดยใช้ Mini Speaker หรือ Buzzer จำนวน 1 ชุด
20. มี 22 Bit GPIO อิสระ สำหรับประยุกต์ต่างๆ เช่น D/A,I2C,I2S,CAN และ Input / Output
    a. Header 10Pin IDE (P2[0..7]) สำหรับ GPIO หรือ Full-Duplex Serial UART
    b. 3 Pin Header(P0[26]) สำหรับ GPIO หรือ D/A
    c. 3 Pin Header(P4[29]) สำหรับ GPIO
    d. 4 Pin Header(P0[19..20]) สำหรับ GPIO หรือ I2C Bus
    e. 4 Pin Header(P0[0..1]) และ P0[4..5] สำหรับ GPIO หรือ CAN1 และ CAN2 Bus
    f. 5 Pin Header(P0[23..25] และ P2[11..13] สำหรับ GPIO หรือ I2S-RX และ I2S-TX

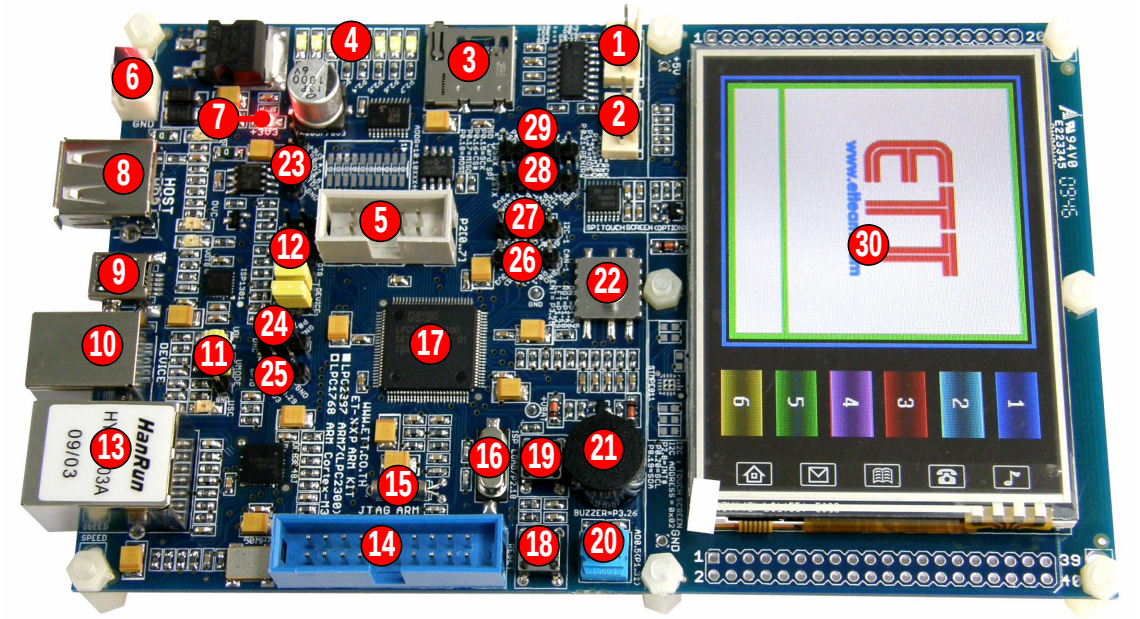


รูปแสดง โครงสร้างของบอร์ด **ET-NXP ARM KIT LPC1768 & TFT LCD**



รูปแสดง โครงสร้างของบอร์ด **ET-NXP ARM KIT (LPC1768)**

โครงสร้างบอร์ด ET-NXP ARM KIT (LPC1768)



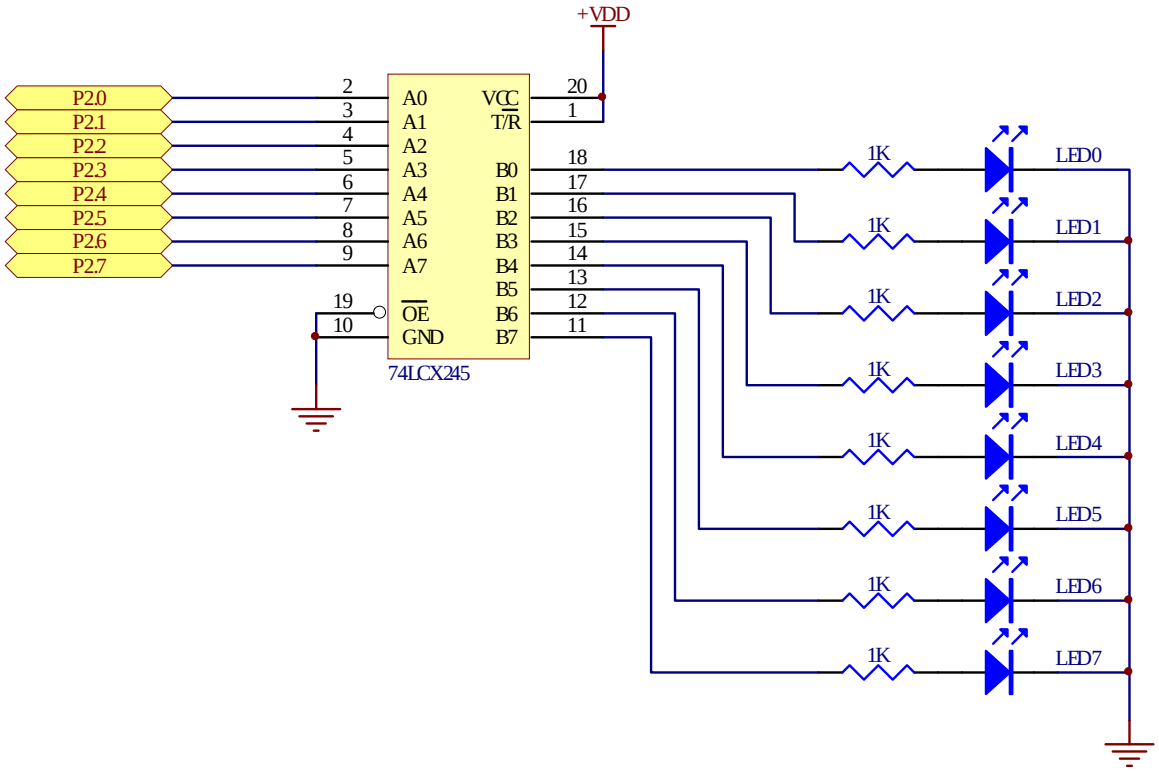
รูปแสดง ตำแหน่งของอุปกรณ์ต่างๆในบอร์ด ET-NXP ARM KIT (LPC1768)

- **หมายเลข 1 คือ** ขั้วต่อ UART0(RS232) สำหรับใช้งาน และ Download Hex File ให้ CPU
- **หมายเลข 2 คือ** ขั้วต่อ UART2(RS232) สำหรับใช้งาน
- **หมายเลข 3 คือ** ช่องเสียบการ์ดหน่วยความจำสามารถใช้ได้กับ SD Card แบบ Micro-SD
- **หมายเลข 4 คือ** LED[0..7] ใช้ทดสอบ Logic Output ของ P2[0..7]
- **หมายเลข 5 คือ** ขั้วต่อ GPIO(P2[0..7]
- **หมายเลข 6 คือ** ขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ดใช้ได้กับไฟ +5VDC
- **หมายเลข 7 คือ** LED แสดงสถานะของ Power +VDD(+3V3)
- **หมายเลข 8 คือ** ขั้วต่อ USB Host
- **หมายเลข 9 คือ** ขั้วต่อ USB OTG
- **หมายเลข 10 คือ** ขั้วต่อ USB Device
- **หมายเลข 11 คือ** Jumper(UMODE) สำหรับเลือกโหมดการ Connect ของ USB Device
- **หมายเลข 12 คือ** Jumper(USB) สำหรับเลือกโหมด USB ระหว่าง Device/OTG/Host
- **หมายเลข 13 คือ** ขั้วต่อสัญญาณ Ethernet LAN แบบ RJ45
- **หมายเลข 14 คือ** ขั้วต่อ JTAG ARM สำหรับ Debug แบบ Real Time
- **หมายเลข 15 คือ** Crystal ค่า 32.768KHz สำหรับฐานเวลาให้ RTC ภายในตัว MCU

- **หมายเลข 16 คือ** Crystal ค่า 12 MHz สำหรับใช้เป็นฐานเวลาระบบให้ MCU
- **หมายเลข 17 คือ** MCU เบอร์ LPC1768 (100Pin LQFP)
- **หมายเลข 18 คือ** SW RESET
- **หมายเลข 19 คือ** SW ISP LOAD หรือ P2.10/EINT0
- **หมายเลข 20 คือ** VR สำหรับปรับค่าแรงดัน 0-3V3 สำหรับทดสอบ A/D(P1[31]/AD0[5])
- **หมายเลข 21 คือ** Buzzer สำหรับใช้กำเนิดเสียง
- **หมายเลข 22 คือ** Joy Switch แบบ 5 ทิศทาง
- **หมายเลข 23 คือ** ขั้วต่อ CAN2 หรือ GPIO P0[4..5]
- **หมายเลข 24 คือ** ขั้วต่อ D/A หรือ GPIO P0.26
- **หมายเลข 25 คือ** ขั้วต่อ GPIO P4.29
- **หมายเลข 26 คือ** ขั้วต่อ CAN1 หรือ GPIO P0[0..1]
- **หมายเลข 27 คือ** ขั้วต่อ I2C1-Bus หรือ GPIO P0[19..20]
- **หมายเลข 28 คือ** ขั้วต่อ I2STX หรือ GPIO P2[11..13]
- **หมายเลข 29 คือ** ขั้วต่อ I2SRX หรือ GPIO P0[23..25]
- **หมายเลข 30 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 P2[0..7] โดยวงจรในส่วนนี้จะใช้สำหรับทดสอบการทำงานของ Output



โดยเมื่อต้องการใช้งานผู้ใช้ต้องกำหนดให้P2[0..7] ทำหน้าที่เป็น GPIO Output Port เสียก่อนแล้วจึงควบคุม Logic ให้กับ P2[0..7] ตามต้องการ ดังตัวอย่าง

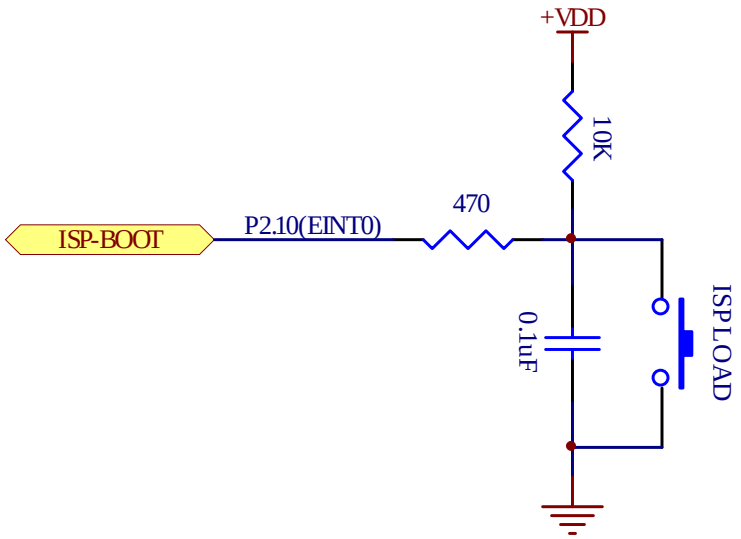
```
// Config Pin GPIO = P2[0..7] Drive LED
LPC_PINCON->PINSEL4 &= ~(0xFFFF);  // Reset P2[0..7] = GPIO
LPC_GPIO2->FIODIR   |=   0xFF;     // P2[0..7] = Outputs
LPC_GPIO2->FIOCLR    =   0xFF;     // Turn-OFF all LED
.
.
LPC_GPIO2->FIOSET    = (1<<0);     // ON  LED[0]
LPC_GPIO2->FIOCLR    = (1<<0);     // OFF LED[0]
LPC_GPIO2->FIOPIN   ^= (1<<0);     // Toggle LED[0]
```

ตัวอย่าง การกำหนดค่าการใช้งาน **P2[0..7]** เป็น **Output LED**

การใช้งานวงจร Push Button Switch

วงจร Push Button Switch จะใช้วงจร Switch แบบ กดติด-ปล่อยดับ (Push Button) พร้อมวงจร Pull-Up ใช้กับแหล่งจ่าย +3.3V โดยในขณะที่สวิตช์ยังไม่ถูกกดจะให้ค่าสถานะเป็นโลจิก “1” แต่เมื่อสวิตช์ถูกกดอยู่จะให้สถานะเป็นโลจิก “0”ใช้สำหรับทดสอบการทำงานของ Input Logic โดยวงจรส่วนนี้จะใช้ P2.10 ในการเชื่อมต่อ สามารถ ทำหน้าที่ ได้ 3 แบบด้วยกัน คือ

- สำหรับ ISP Download ผ่าน RS232(UART0) โดยใช้ร่วมกับสวิตช์ RESET
- ทดสอบ Input ของ P2[10]
- ทดสอบการ Interrupt ของ EINT0



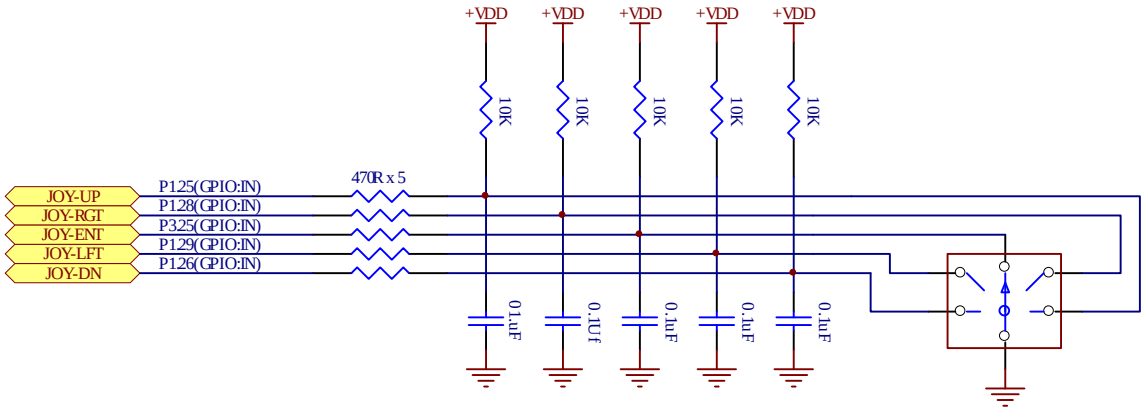

```
LPC_PINCON->PINSEL4 &= ~(0x03<<20);     // Reset P2.10 = GPIO
LPC_GPIO2->FIODIR   &= ~(1UL<<10);      // P2.10 = Input
.
.
if ((LPC_GPIO2->FIOPIN >> 10) & 0x01)   //SW = Release
{
  ...
}
else                                    // SW = Press
{
  ...
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน P2.10 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 P1.25 ในหน้าที่ GPIO Input
- Down Position จะใช้ P1.26 ในหน้าที่ GPIO Input
- Right Position จะใช้ P1.28 ในหน้าที่ GPIO Input
- Left Position จะใช้ P1.29 ในหน้าที่ GPIO Input
- Center Position จะใช้ P3.25 ในหน้าที่ GPIO Input

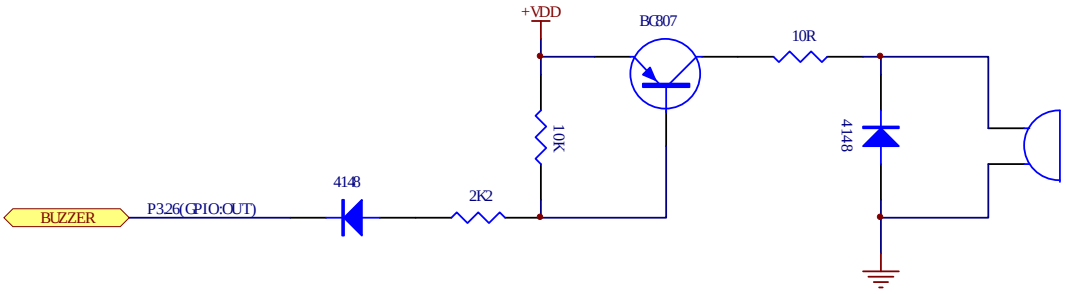


```
//Joy Switch
//P1.25,P1.26,P1.28,P1.29,P3.25 = Joy Switch
LPC_PINCON->PINSEL3 &= ~(0x03<<18);              //P1.25 = GPIO
LPC_PINCON->PINSEL3 &= ~(0x03<<20);              //P1.26 = GPIO
LPC_PINCON->PINSEL3 &= ~(0x03<<24);              //P1.28 = GPIO
LPC_PINCON->PINSEL3 &= ~(0x03<<26);              //P1.29 = GPIO
LPC_GPIO1->FIODIR   &= ~((1UL<<25)|(1UL<<26)|
                         (1UL<<28)|(1UL<<29));   //P1.25,26,28,29=In

LPC_PINCON->PINSEL7 &= ~(0x03<<18);              //P3.25 = GPIO
LPC_GPIO3->FIODIR   &= ~(1UL<<25);               //P3[25]= Input
.
.
//Joy Up = P1.25
if ((LPC_GPIO1->FIOPIN >> 25) & 0x01)            // SW = Release
{
  ...
}
else                                             // SW = Press
{
  ...
}

//Joy Down = P1.26
if ((LPC_GPIO1->FIOPIN >> 26) & 0x01)            // SW = Release
{
  ...
}
else                                             // SW = Press
{
  ...
}

//Joy Right = P1.28
if ((LPC_GPIO1->FIOPIN >> 28) & 0x01)            // SW = Release
.
.
.
.
.
//Joy Center = P3.25
if ((LPC_GPIO3->FIOPIN >> 25) & 0x01)            // SW = Release
{
  ...
}
else                                             // SW = Press
{
  ...
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน **Input Joy Switch**

การใช้งาน วงจรกำเนิดเสียง

วงจรกำเนิดเสียง จะใช้ลำโพงขนาดเล็ก (Mini Speaker) พร้อมด้วยวงจรทรานซิสเตอร์แบบ NPN สำหรับขับกระแสให้กับลำโพง ใช้กับแหล่งจ่ายขนาด +3.3V ทำงานด้วยโลจิก “1” และหยุดทำงานด้วยโลจิก “0” โดยในการทำงานนั้นต้องส่งสัญญาณโลจิกที่เป็นความถี่ต่างๆให้กับลำโพงเพื่อสร้างเป็นความถี่เสียงย่านต่างๆ ตามต้องการ โดยใช้การควบคุมจาก P3[26]



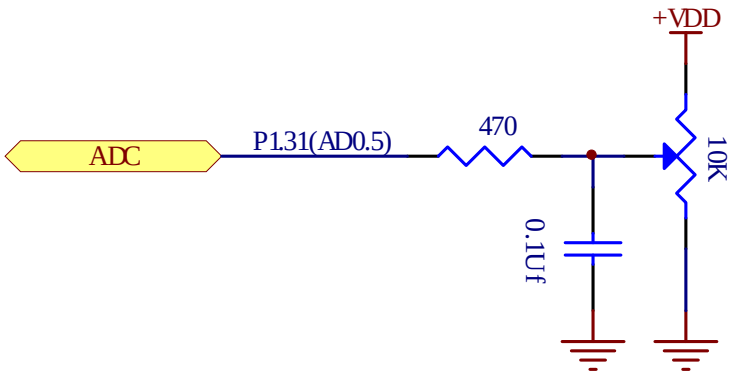
โดยเมื่อต้องการใช้งานผู้ใช้ต้องกำหนดให้P3[28] ทำหน้าที่เป็น GPIO Output Port เสียก่อนแล้วจึงควบคุม Logic ให้กับ P3[28] ON/OFF เป็นความถี่ ตามต้องการดังตัวอย่าง

```
//Config Pin GPIO = P3[26] Drive Mini Speaker Generate Beep
LPC_PINCON->PINSEL7 &= ~(3 << 20);          // Reset P3.26 = GPIO
LPC_GPIO3->FIODIR   |=  (1UL<<26);          // P3[26] = Output

// Loop Generate Beep on Speaker(P3.26)
while(1)                                    // Loop Continue
{
  for (i = 0; i < 500; i++)                 // Start Beep Pulse
  {
    LPC_GPIO3->FIOPIN ^= (1 << 26);         // Toggle P3[26]
    delay(5000);
  }

  delay(10000000);                          // Stop Beep Pulse
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน P3.26 เป็น Output ขับ Buzzer

การใช้งานวงจรปรับแรงดัน (0-3V3)

วงจรปรับแรงดันจะใช้ตัวต้านทานปรับค่าได้แบบเกือกม้า ชนิดมีแกนหมุนสำหรับปรับค่า โดยวงจรนี้ใช้กับแหล่งจ่าย +3.3V โดยจะให้ Output เป็นแรงดันซึ่งมีค่าระหว่าง 0V ถึง +3.3V ตามการปรับค่าของตัวต้านทาน จำนวน 1 ชุด โดย Output ที่ได้จะป้อนให้กับขาสัญญาณ P1[31] สำหรับใช้สร้างแรงดัน Input เพื่อทดสอบการทำงานของวงจร A/D (P1[31])



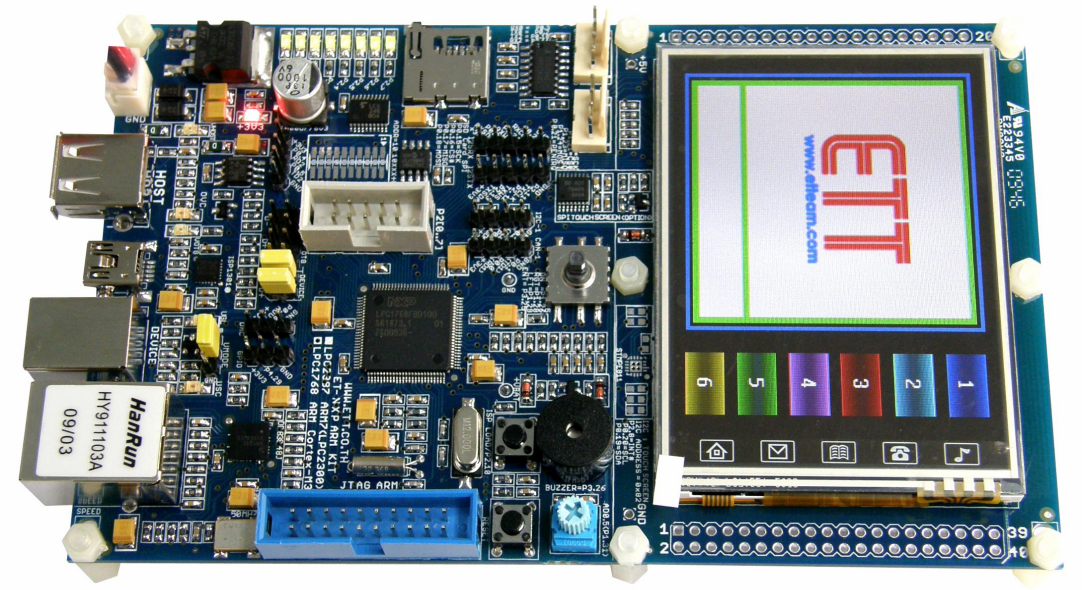

```
LPC_PINCON->PINSEL3   &= ~(3UL<<30);        // Reset P1.31 = GPIO
LPC_PINCON->PINSEL3   |=  (3UL<<30);        // Config P1.31 = AD0.5
LPC_SC->PCONP         |=  (1UL<<12);        // Enable power to ADC
LPC_ADC->ADCR         =   (1UL<< 5) |       // select AD0.5 pin
                          (1UL<< 8) |       // ADC clock is 18MHz/2
                          (1UL<<21);        // enable ADC

unsigned int val;
.
.
while(1)
{
  LPC_ADC->ADCR |=  (1<<24);                // start conversion
  while (!(LPC_ADC->ADGDR & (1UL<<31)));    // Wait Conversion end
  val = ((LPC_ADC->ADGDR >> 4) & 0xFFF);    // read converted value
  LPC_ADC->ADCR &= ~(7<<24);                // stop conversion
  .
  .
  .
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน P1.31 เป็น Analog Input AD0.5

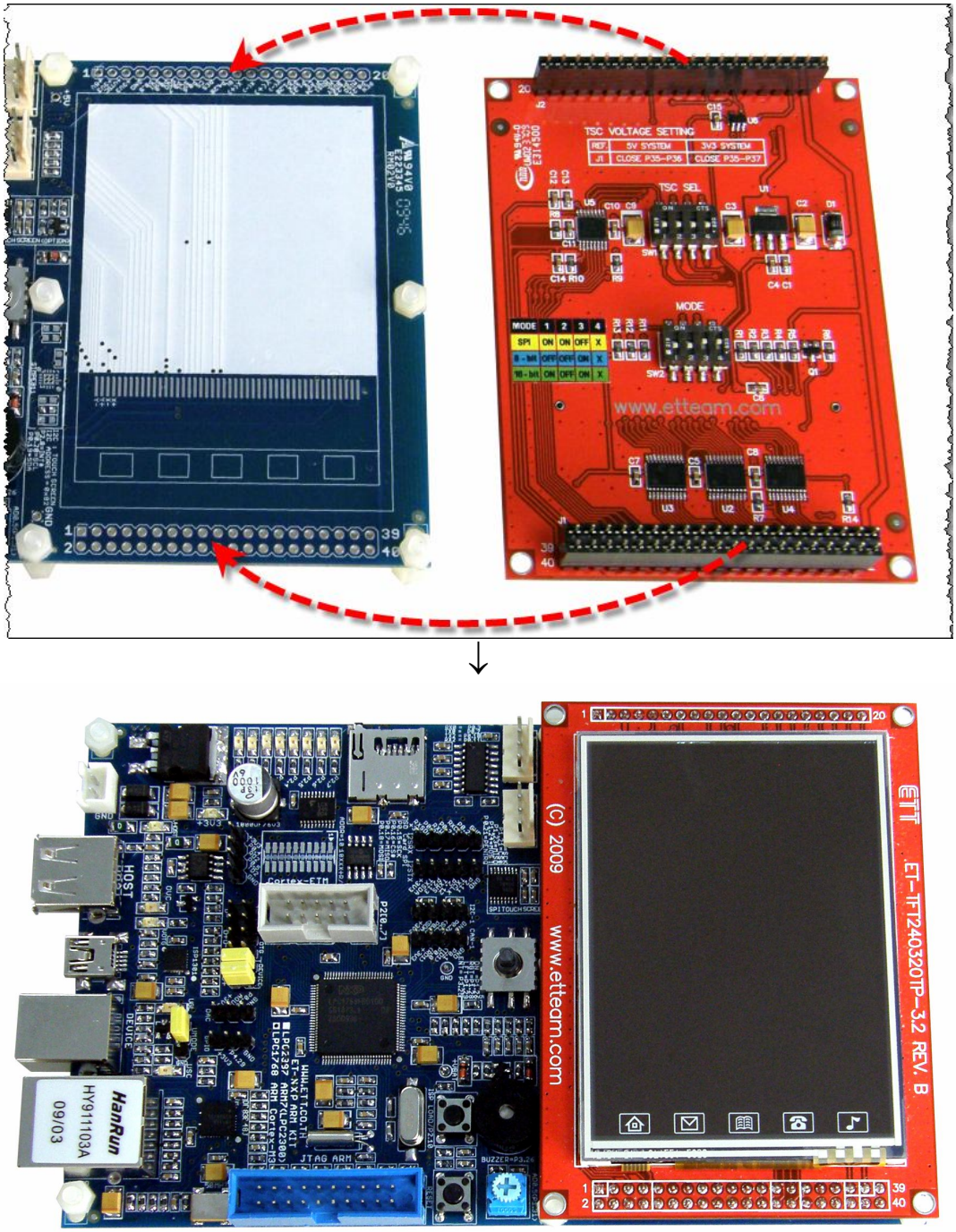
การใช้งานจอแสดงผล Graphic LCD แบบ TFT LCD

สำหรับการเชื่อมต่อกับ Graphic LCD นั้น วงจรของบอร์ด ET-NXP ARK KIT(LPC1768) ได้รับการออกแบบให้สามารถเชื่อมต่อกับ Graphic 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-NXP ARM KIT (LPC1768) 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 SSP1 ของ MCU ในการติดต่อ โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- CS GLCD จะใช้ P0.6 ในหน้าที่ GPIO Output
- SCL GLCD จะใช้ P0.7 ในหน้าที่ SCK1 ของ SSP1
- SDO GLCD จะใช้ P0.8 ในหน้าที่ MISO1 ของ SSP1
- SDI GLCD จะใช้ P0.9 ในหน้าที่ MOSI1 ของ SSP1
- BL GLCD จะใช้ P4.28 ในหน้าที่ GPIO Output

การเชื่อมต่อกับ Touch Screen Sensor โดยใช้ ADS7846

ส่วนของ Touch Screen ในกรณีใช้ชิพ ADS7846 จะใช้การเชื่อมต่อแบบ SPI โดยจะใช้ SSP0 ของ MCUในการติดต่อ โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- DCLK ADS7846 จะใช้ P1.20 ในหน้าที่ SCK0 ของ SSP0
- CS ADS7846 จะใช้ P1.21 ในหน้าที่ GPIO Output
- DOUT ADS7846 จะใช้ P1.23 ในหน้าที่ MISO0 ของ SSP0
- DIN ADS7846 จะใช้ P1.24 ในหน้าที่ MOSI0 ของ SSP0
- PENIRQ ADS7846 จะใช้ P0.21 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

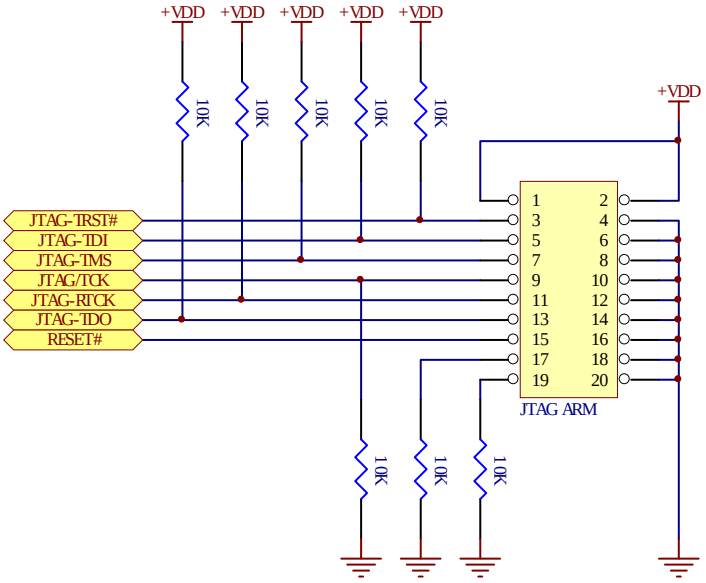
- SDAT STMPE811 จะใช้ P0.19 ในหน้าที่ SDA1 ของ I2C1
- SCLK STMPE811 จะใช้ P0.20 ในหน้าที่ SCL1 ของ I2C1
- INT STMPE811 จะใช้ P2.8 ในหน้าที่ GPIO Input

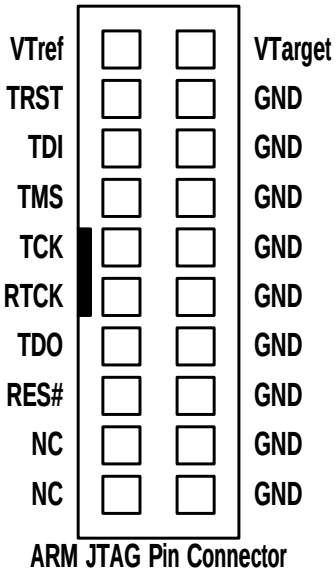
```
/* Config P1[20..24] to SSP0 For Read Touch LCD(ADS7846) */
LPC_PINCON->PINSEL3 &= ~(3UL<<10);     // Reset P1.21 Mode = GPIO
LPC_GPIO1->FIODIR   |=  (1UL<<21);     // P1.21 = ADS7846 CS(Output)
LPC_GPIO1->FIOPIN   |=  (1UL<<21);     // P1.21 = High

LPC_PINCON->PINSEL1 &= ~(3UL<<10);     // Reset P0.21 Mode = GPIO
LPC_GPIO0->FIODIR   &= ~(1UL<<21);     // P0.21 = PENIRQ(Input)

//Config SSP0 Pin Connect
LPC_PINCON->PINSEL3 |=  (3UL<<8);      // Select P1.20 = SCK0(SSP0)
LPC_PINCON->PINSEL3 |=  (3UL<<14);     // Select P1.23 = MISO0(SSP0)
LPC_PINCON->PINSEL3 |=  (3UL<<16);     // Select P1.24 = MOSI0(SSP0)

LPC_SC->PCONP       |=  (1<<21);       // Enable power to SSPI0 block
LPC_SC->PCLKSEL1    &= ~(3<<10);       // PCLKSP0 = CCLK/4 (18MHz)
LPC_SC->PCLKSEL1    |=  (1<<10);       // PCLKSP0 = CCLK   (72MHz)
LPC_SSP0->CPSR       =   72;           // 72MHz / 72 = 1MHz(max 2MHz)

LPC_SSP0->CR0        =  (   0 << 7) | // CPHA = 0
                        (   0 << 6) | // CPOL = 0
                        (   0 << 4) | // Frame format = SPI
                        ((8-1)<< 0);  // Data size = 8 bits
LPC_SSP0->CR1        =  (   1 << 1);  // Enable SSP0

/* Config P0.6,P0.7,P0.8,P0.9 to SSP1 For Control GLCD */
LPC_PINCON->PINSEL9 &= ~(3 << 24);     // Reset P4.28 Mode = GPIO
LPC_GPIO4->FIODIR   |=  (1UL<<28);     // Pin P4.28 = Output(BL)
LPC_GPIO4->FIOPIN   &= ~(1UL<<28);     // Turn-OFF GLCD Backlight

LPC_PINCON->PINSEL0 &= ~(3UL<<12);     // Reset P0.6 Mode = GPIO
LPC_GPIO0->FIODIR   |=  (1 <<  6);     // P0.6 = GPIO output(CS GLCD)
LPC_GPIO0->FIOSET    =  (1 <<  6);     // Set P0.6 = High

LPC_PINCON->PINSEL0 &= ~(3UL<<14);     // Reset P0.7 Mode = GPIO
LPC_PINCON->PINSEL0 |=  (2UL<<14);     // Select P0.7 = SCK1(SSP1)
LPC_PINCON->PINSEL0 &= ~(3UL<<16);     // Reset P0.8 Mode = GPIO
LPC_PINCON->PINSEL0 |=  (2UL<<16);     // Select P0.8 = MISO1(SSP1)
LPC_PINCON->PINSEL0 &= ~(3UL<<18);     // Reset P0.9 Mode = GPIO
LPC_PINCON->PINSEL0 |=  (2UL<<18);     // Select P0.9 = MOSI1(SSP1)

LPC_SC->PCONP       |=  (1 << 10);     // Enable power to SSP1 block
LPC_SC->PCLKSEL0    |=  (2 << 20);     // SSP1 clock = CCLK/2 (36MHz)
LPC_SSP1->CPSR       =  2;             // Clock Rate = 18MHz

LPC_SSP1->CR0        =  (1<<7) |       // CPHA = 1
                        (1<<6) |       // CPOL = 1
                        (0<<4) |       // Frame format = SPI
                        ((8-1)<< 0);   // Data size = 8 bits
LPC_SSP1->CR1        =  (1<<1);        // Enable SSP1
```

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน GLCD และ Touch Screen

การใช้งาน JTAG ARM

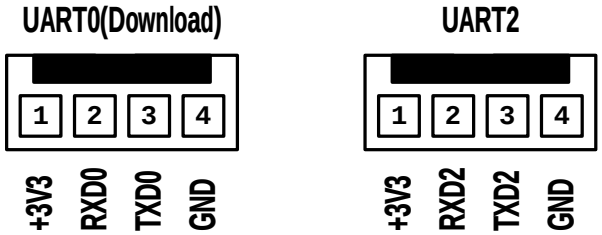
JTAG หรือ JTAG ARM จะเป็น Connector แบบ IDE 20 Pin สำหรับ Interface กับ JTAG Debugger โดยมีการจัดวงจรและสัญญาณตามมาตรฐานของ JTAG ดังนี้





พอร์ต RS232

เป็นสัญญาณ RS232 ซึ่งผ่านวงจรแปลงระดับสัญญาณ MAX3232 เรียบร้อยแล้ว โดยมีจำนวน 2 ช่อง ด้วยกันคือ UART0 และ UART2 โดยทั้ง 2 ช่องสามารถใช้เชื่อมต่อกับสัญญาณ RS232 เพื่อรับส่งข้อมูลได้ นอกจากนี้แล้ว UART0 ยังสามารถใช้งานเป็น ISP Download สำหรับทำการ Download Hex File ให้กับ MCU ได้ด้วย โดยในกรณีนี้ต้องใช้งานร่วมกับ SW ISP LOAD และ SW RESET เพื่อ Reset ให้ CPU เริ่มต้นทำงานใน Boot-Loader Mode เพื่อทำการ Download Hex File ให้กับ CPU ได้ด้วย(ดูรายละเอียดเพิ่มเติมเรื่อง "การ Download Hex File ให้กับ MCU ของบอร์ด")



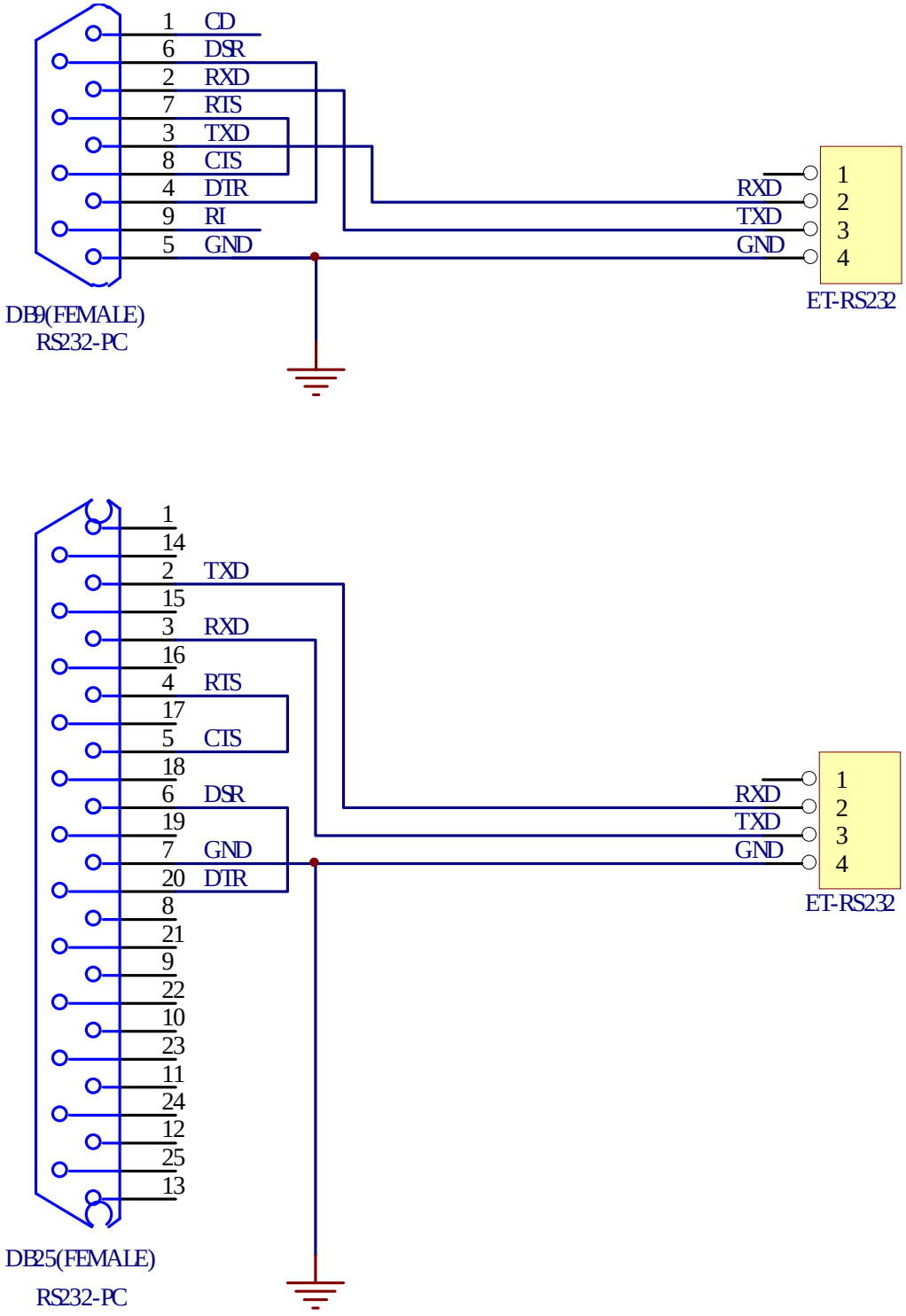
- UART-0 ใช้ขาสัญญาณจาก P0.2(TXD0) และ P0.3(RXD0)
- UART-2 ใช้ขาสัญญาณจาก P0.10(TXD2) และ P0.11(RXD2)

เนื่องจากระบบ Hardware UART ของ LPC1768/LPC2387 นั้นจะสามารถกำหนดขาสัญญาณในการเชื่อมต่อได้หลายจุด ตัวอย่างเช่น UART2 สามารถเลือกใช้ขาสัญญาณ P0[10] กับ P0[11] หรือ P2[8] กับ P2[9] ก็ได้ ซึ่งบอร์ด ET-NXP ARM KIT นั้นเลือกใช้ขาสัญญาณชุด P0[10] กับ P0[11] เป็นจุดเชื่อมต่อกับ UART2 ดังนั้น ผู้ใช้ต้องกำหนดคำสั่งในการเลือกใช้ขาสัญญาณให้ถูกต้องด้วย และข้อควรระวังอีกประการหนึ่งในการใช้งาน UART ก็คือ ค่า Default ของ UART2 จะถูกปิดการทำงานไว้ ดังนั้นผู้ใช้ต้องสั่งเปิดการทำงานของวงจร UART2 ก่อนที่จะสั่ง Initial ค่าต่างๆให้กับ UART ด้วย ไม่เช่นนั้นจะไม่สามารถสั่งงาน UART ได้ สำหรับ Code ตัวอย่างการกำหนดค่า UART ในส่วนเริ่มต้นเป็นดังนี้

```
// Config UART0 Connect to P0[2]:P0[3]
LPC_PINCON->PINSEL0 &= ~(0x03<<4);              // Reset P0.2 = GPIO
LPC_PINCON->PINSEL0 |=  (0x01<<4);              // Config P0.2 = TxD0
LPC_PINCON->PINSEL0 &= ~(0x03<<6);              // Reset P0.3 = GPIO
LPC_PINCON->PINSEL0 |=  (0x01<<6);              // Config P0.3 = RxD0

// Config UART2 Connect to P0[10]:P0[11]
LPC_PINCON->PINSEL0 &= ~(0x03<<20);             // Reset P0.10 = GPIO
LPC_PINCON->PINSEL0 |=  (0x01<<20);             // Config P0.10 = TxD2
LPC_PINCON->PINSEL0 &= ~(0x03<<22);             // Reset P0.11 = GPIO
LPC_PINCON->PINSEL0 |=  (0x01<<22);             // Config P0.11 = RxD2
PCONP               |= 0x01000000;              // UART2 Power-ON
```

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน UART0 และ UART1

สำหรับ Cable ที่จะใช้ในการเชื่อมต่อ RS232 ระหว่าง Comport ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้วต่อ UART0 และ UART2 ของบอร์ด ET-NXP ARM KIT นั้น เป็นดังนี้



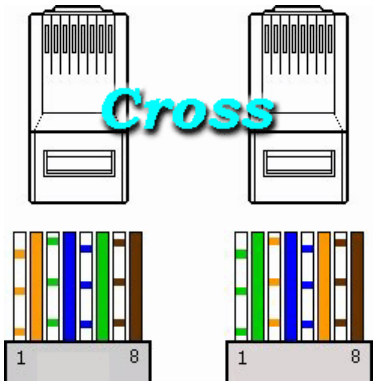
รูป แสดงวงจรสาย Cable สำหรับ RS232

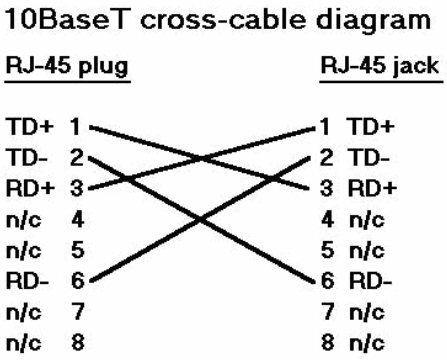
Ethernet LAN

สำหรับการเชื่อมต่อกับเครือข่าย Network ระหว่างบอร์ด ET-NXP ARM KIT (LPC1768) นั้น จะใช้ขั้วต่อมาตรฐาน Ethernet แบบ RJ45 โดยวงจรส่วนนี้จะใช้ขาสัญญาณ P1[0,1,4,8,9,10,14..17] ในการเชื่อมต่อโดยใช้ Chips Physical Ethernet เบอร์ DP83848 เป็น Driver ในการเชื่อมต่อ

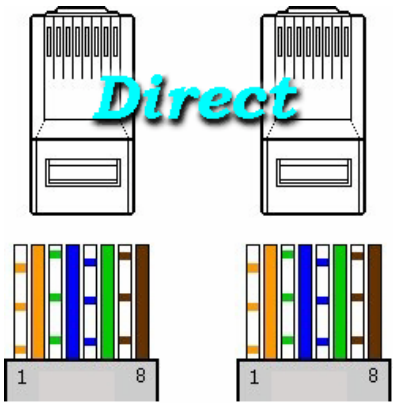
สำหรับวิธีการเชื่อมต่อสายสัญญาณ Ethernet LAN ของบอร์ดเข้ากับระบบเครือข่ายจะทำได้ 2 แบบด้วยกัน คือการต่อแบบ Direct Line และต่อผ่าน Hub

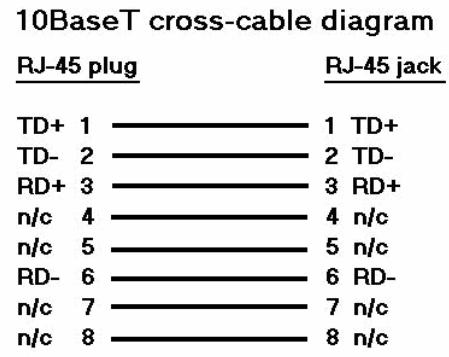
- กรณีที่ 1 คือ การเชื่อมต่อเข้ากับคอมพิวเตอร์โดยตรง สาย LAN จะต้องเข้าสายแบบ Cross





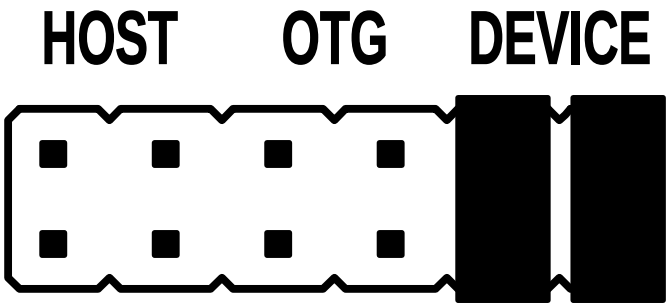
- กรณีที่ 2 คือ การเชื่อมต่อผ่าน Hub ของเครื่องคอมพิวเตอร์ Server จะต้องเข้าสายแบบ Direct



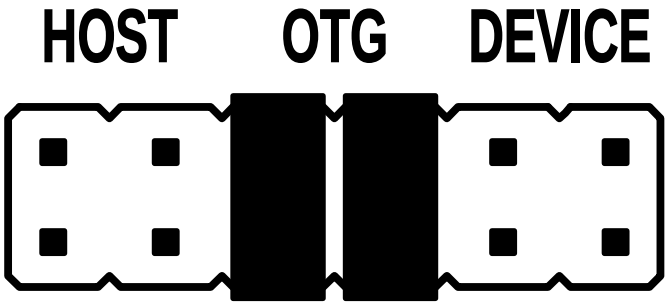


การใช้งาน USB

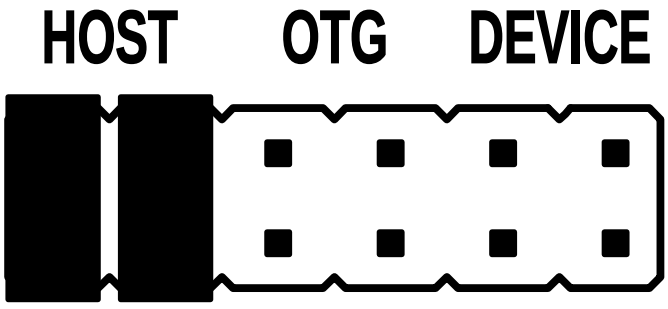
บอร์ด ET-NXP ARM KIT (LPC1768) ถูกออกแบบให้มีพอร์ตสำหรับเชื่อมต่อกับอุปกรณ์ USB ทั้งแบบ Device หรือ USB Host หรือ USB OTG (On-The-Go) ก็ได้ ขึ้นอยู่กับการเขียนโปรแกรม กำหนดหน้าที่การทำงานของ USB ในตัว MCU ของ LPC1768 โดยในส่วนของ Hardware นั้น ผู้ใช้จะต้องเลือกกำหนด Jumper เพื่อเลือกเชื่อมต่อสัญญาณ USB ของ MCU LPC1768 เข้ากับวงจรของ USB Port ให้ตรงกับความต้องการใช้งานด้วย โดยสามารถเลือกกำหนดการเชื่อมต่อ USB ได้ 3 แบบ ดังนี้คือ



การเลือกใช้งาน USB เป็น USB DEVICE



การเลือกใช้งาน USB เป็น USB OTG



การเลือกใช้งาน USB เป็น USB HOST

USB Device Mode

ในโหมดนี้จะใช้สัญญาณจำนวน 5 เส้นในการเชื่อมต่อ โดยต้องโปรแกรมหน้าที่ของขาสัญญาณสำหรับใช้ในการเชื่อมต่อกับ USB Bus เป็นดังนี้

- USB D(+) จะใช้ P0.29 ในหน้าที่ USB_D+ ของ USB Device Mode
- USB D(-) จะใช้ P0.30 ในหน้าที่ USB_D- ของ USB Device Mode
- USB CONNECT จะใช้ P2.9 ในหน้าที่ USB_CONNECT(USC LED) ของ USB Device Mode
- USB UP LED P1.18 ในหน้าที่ USB_UP_LED(UGL LED) ของ USB Device Mode
- USB VBUS จะใช้ P1.30 ในหน้าที่ USB VBUS ของ USB Device Mode

ในกรณีของ USB Device นั้น จะมี Jumper UMODE สำหรับเลือกกำหนดรูปแบบการเชื่อมต่อกับ USB Host ด้วยว่าจะใช้การเชื่อมต่อแบบ Direct Connect หรือ จะใช้การเชื่อมต่อ Soft Connect โดยใช้ P2.9 (USB_CONNECT) เป็นขาควบคุมการเชื่อมต่อ โดยถ้าต้องการใช้การเชื่อมต่อแบบ Soft Connect ให้เลือก Jumper UMODE ไว้ทางด้าน P2.9 แล้วเขียนโปรแกรมกำหนดให้ P2.9 ทำหน้าที่ควบคุมการเชื่อมต่อของ USB Device กับ Bus แต่ถ้าไม่ต้องการใช้ Soft Connect ให้เลือก Jumper UMODE ไว้ทางด้าน GND เพื่อเปิดการ เชื่อมต่อ USB Device กับ Bus ตลอดเวลา โดยจะมี LED USC สำหรับแสดงสถานะของสัญญาณ ให้ทราบด้วย โดย LED USC จะติดสว่างเมื่อ สัญญาณควบคุมการเชื่อมต่อ USB Device อยู่ในสถานะ Active

โดยในโหมด USB Device นี้จะมี LED ที่ใช้แสดงสถานะของการทำงานของ USB จำนวน 2 ดวง คือ

- USC ใช้แสดงสถานะของสัญญาณควบคุมการ Connect Bus โดยจะติดสว่างให้เห็นเมื่อวงจรถูกสั่งให้ Connect Bus
- UGL ใช้แสดงสถานะ เมื่อมีการ Connect ของ USB Device กับ Host Bus ได้สำเร็จเรียบร้อยแล้ว

USB OTG Mode (USB On-The-Go)

ในโหมดนี้จะใช้สัญญาณจำนวน 5 เส้นในการเชื่อมต่อ โดยจะใช้ชิพเบอร์ ISP1301 ทำหน้าที่เป็น USB OTG Transceiver โดยต้องโปรแกรมหน้าที่ของขาสัญญาณสำหรับใช้ในการเชื่อมต่อกับ USB Bus เป็นดังนี้

- USB D(+) จะใช้ P0.29 ในหน้าที่ USB_D+ ของ USB OTG Mode
- USB D(-) จะใช้ P0.30 ในหน้าที่ USB_D- ของ USB OTG Mode
- USB SDA จะใช้ P0.27 ในหน้าที่ USB_SDA ของ USB OTG Mode
- USB SCL จะใช้ P0.28 ในหน้าที่ USB_SCL ของ USB OTG Mode
- USB INT จะใช้ P0.22 ในหน้าที่ GPIO Input

โดยในโหมด USB OTG นี้จะมี LED ที่ใช้แสดงสถานะของการทำงานของ USB จำนวน 2 ดวง คือ

- VOTG ใช้แสดงสถานะของ VBUS ของ USB Vbus
- UGL ใช้แสดงสถานะ เมื่อมีการ Connect ของ USB Bus กับ อุปกรณ์ภายนอกที่นำมาเชื่อมต่อได้สำเร็จเรียบร้อยแล้ว

USB Host Mode

ในโหมดนี้จะใช้สัญญาณจำนวน 5 เส้นในการเชื่อมต่อ โดยจะใช้ชิพเบอร์TPS2055 หรือ TPS2031 ทำหน้าที่เป็นตัวควบคุมแหล่งจ่าย VBUS ของ Host พร้อมทั้งทำหน้าที่เป็น Host Over Current Protection ด้วย โดยต้องโปรแกรมหน้าที่ของขาสัญญาณสำหรับใช้ในการเชื่อมต่อกับUSB Bus เป็นดังนี้

- USB D(+) จะใช้ P0.29 ในหน้าที่ USB_D+ ของ USB Host Mode
- USB D(-) จะใช้ P0.30 ในหน้าที่ USB_D- ของ USB Host Mode
- USB OVRCR จะใช้ P1.27 ในหน้าที่ USB_OVRCR ของ USB Host Mode
- USB PPWR จะใช้ P1.19 ในหน้าที่ USB_PPWR ของ USB Host Mode
- USB UP LED P1.18 ในหน้าที่ USB_UP_LED(UGL LED) ของ USB Host Mode

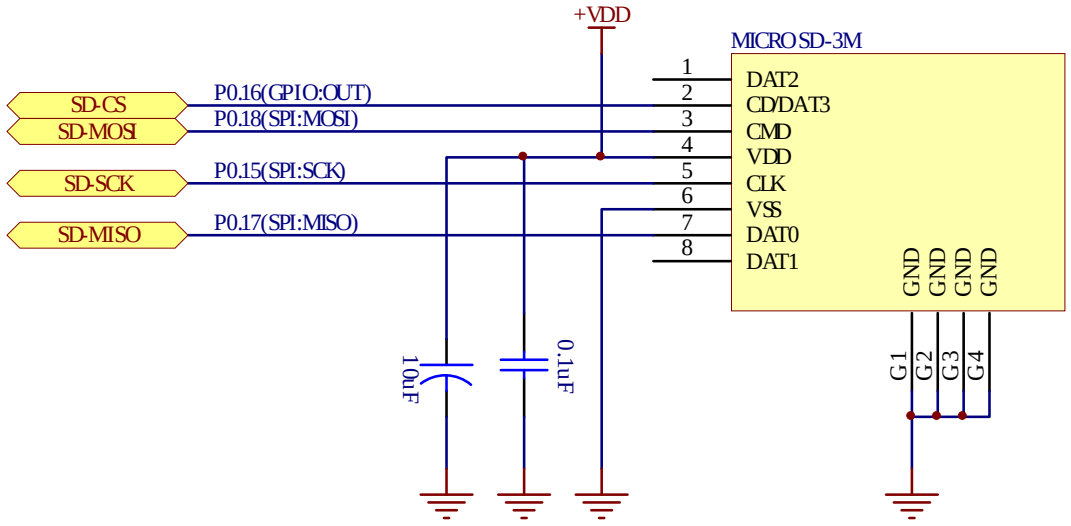
โดยในโหมด USB Host นี้จะมี LED ที่ใช้แสดงสถานะของการทำงานของ USB จำนวน 3 ดวง คือ

- VHOST ใช้แสดงสถานะของ VBUS ของ USH Host Bus โดยจะติดสว่างให้เห็นเมื่อวงจรถูกสั่งให้จ่ายไฟ VBUS ออกไปยัง Host Port
- OVC ใช้แสดงสถานะ เมื่อมีการดึงกระแสจาก USB Host Port สูงเกินกว่าที่กำหนดไว้ โดยเมื่อ LED ดวงนี้ติดสว่าง จะมี Output Logic “0” ส่งไปยัง USB OVRCE(P1.27) ของ MCU เพื่อแจ้งให้ MCU รับทราบด้วย
- UGL ใช้แสดงสถานะ เมื่อมีการ Connect ของ USB Host Bus กับ อุปกรณ์ภายนอกที่นำมาเชื่อมต่อได้สำเร็จเรียบร้อยแล้ว

การ์ดหน่วยความจำ SD Card แบบ Micro-SD

บอร์ด ET-NXP ARM KIT (LPC1768) รองรับการเชื่อมต่อกับการ์ดหน่วยความจำ SD Card แบบ Micro-SD โดยใช้การเชื่อมต่อแบบ SPI โดยใช้ขาสัญญาณ P0[15..18] ในการเชื่อมต่อกับการ์ด ซึ่งในการติดต่อสั่งงาน การ์ดนั้น สามารถโปรแกรม Pin I/O ของ P0[15..18] ให้ทำงานในโหมด SPI โดยต้องกำหนดหน้าที่ของขาสัญญาณ P0[15..18] ของ MCU เป็นดังนี้

- CLK ใช้ P0.15 ในหน้าที่ SCK ของ SPI
- CD/DAT3 ใช้ P0.16 ในหน้าที่ของ GPIO Output
- DAT0 ใช้ P0.17 ในหน้าที่ MISO ของ SPI
- CMD ใช้ P0.18 ในหน้าที่ MOSI ของ SPI



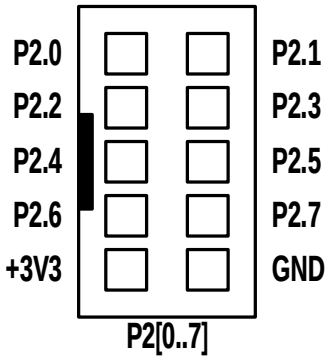
```
//Config P0.15,P0.16,P0.17,P0.18 = SPI Interface
LPC_SC->PCONP       |=  (1 << 8);           //Enable power to SPI
LPC_SC->PCLKSEL0    &= ~(3<<16);            //PCLK_SPI=CCLK/4(18MHz)
LPC_SC->PCLKSEL0    |=  (1<<16);            //PCLK_SPI=CCLK(72MHz)
LPC_SPI->SPCCR = 180;                       //72MHz/180=400kBit

// SSEL is GPIO, output set to high.
LPC_GPIO0->FIODIR  |=  (1<<16);             //P0.16 is output
LPC_GPIO0->FIOPIN  |=  (1<<16);             //set P0.16 = high
LPC_PINCON->PINSEL1 &= ~(3<<0);             //P0.16 = GPIO

// SCK, MISO, MOSI are SSP pins.
LPC_PINCON->PINSEL0 &= ~(3UL<<30);          //P0.15 cleared
LPC_PINCON->PINSEL0 |=  (3UL<<30);          //P0.15 SCK
LPC_PINCON->PINSEL1 &= ~((3<<2)|(3<<4));    //P0.17, P0.18 cleared
LPC_PINCON->PINSEL1 |=  ((3<<2)|(3<<4));    //P0.17 MISO,P0.18 MOSI

//Config SPI = Master,8Bit,CPOL=0,CPHA=0
LPC_SPI->SPCR &= ~(1<<3);                   //CPHA = 0 Rising Clock
LPC_SPI->SPCR &= ~(1<<4);                   //CPOL = 0
LPC_SPI->SPCR |=  (1<<5);                   //MSTR = 1 = Master SPI
LPC_SPI->SPCR &= ~(1<<6);                   //LSBF = 0 = MSB First
LPC_SPI->SPCR &= ~(1<<7);                   //SPIE = 0 = Disable INT

LPC_SPI->SPCR &= ~(15<<8);                  //BIT = 0000(Bits Format)
LPC_SPI->SPCR |=  (1<<11);                  //BIT = 1000(8 Bit Data)
LPC_SPI->SPCR |=  (1<<2);                   //Enable SPI
```

ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน SD Card

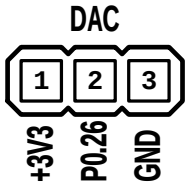
ขั้วต่อ Port I/O ต่าง ๆของบอร์ด

สำหรับขั้วต่อ Port I/O ของ CPU นั้น จะจัดเรียงออกมารอไว้ยังขั้วต่อแบบต่างๆ สำหรับให้ผู้ใช้เลือกต่อออกไปใช้งานตามต้องการ โดยมีด้วยกัน 8 ชุดดังนี้

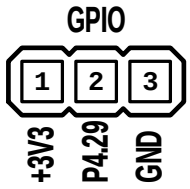
- ขั้วต่อ IDE 10 Pin จำนวน 2 ชุด ชุดละ 8 บิต คือ P2[0..7] โดยมีการจัดเรียงสัญญาณไว้ดังนี้



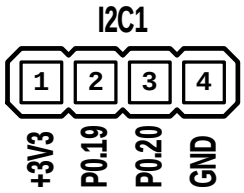
- ขั้วต่อ DAC เป็น Header ขนาด 1x3 ใช้เป็นจุดเชื่อมต่อ P0[26] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น D/A(Aout) ได้ตามต้องการ
  - P0.26 = AOUT หรือ D/A



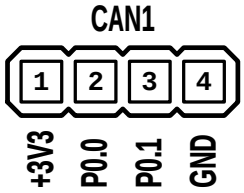
- ขั้วต่อ GPIO เป็น Header ขนาด 1x3 ใช้เป็นจุดเชื่อมต่อ P4[29] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่อื่นๆตามค่า Config ของ P4[29] ตามต้องการ
  - P4.29 = GPIO



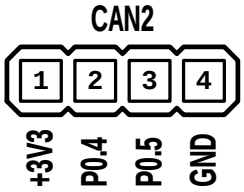
- ขั้วต่อ I2C-1 เป็น Header ขนาด 1x4 ใช้เป็นจุดเชื่อมต่อ P0[19..20] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น I2C Bus ได้ตามต้องการ
    - P0.19 = SDA1
    - P0.20 = SCL1



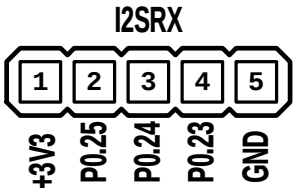
- ขั้วต่อ CAN-1 เป็น Header ขนาด 1x4 ใช้เป็นจุดเชื่อมต่อ P0[0..1] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น Can Bus (CAN-1) ได้ตามต้องการ
    - P0.0 = RD1
    - P0.1 = TD1



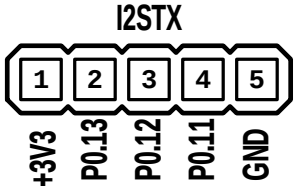
- ขั้วต่อ CAN-2 เป็น Header ขนาด 1x4 ใช้เป็นจุดเชื่อมต่อ P0[4..5] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น Can Bus (CAN-2) ได้ตามต้องการ
    - P0.4 = RD2
    - P0.5 = TD2



- ขั้วต่อ I2SRX เป็น Header ขนาด 1x5 ใช้เป็นจุดเชื่อมต่อ P0[23..25] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น I2SRX ได้ตามต้องการ
  - P0.23 = RXCLK
  - P0.24 = RXWS
  - P0.25 = RXSDA



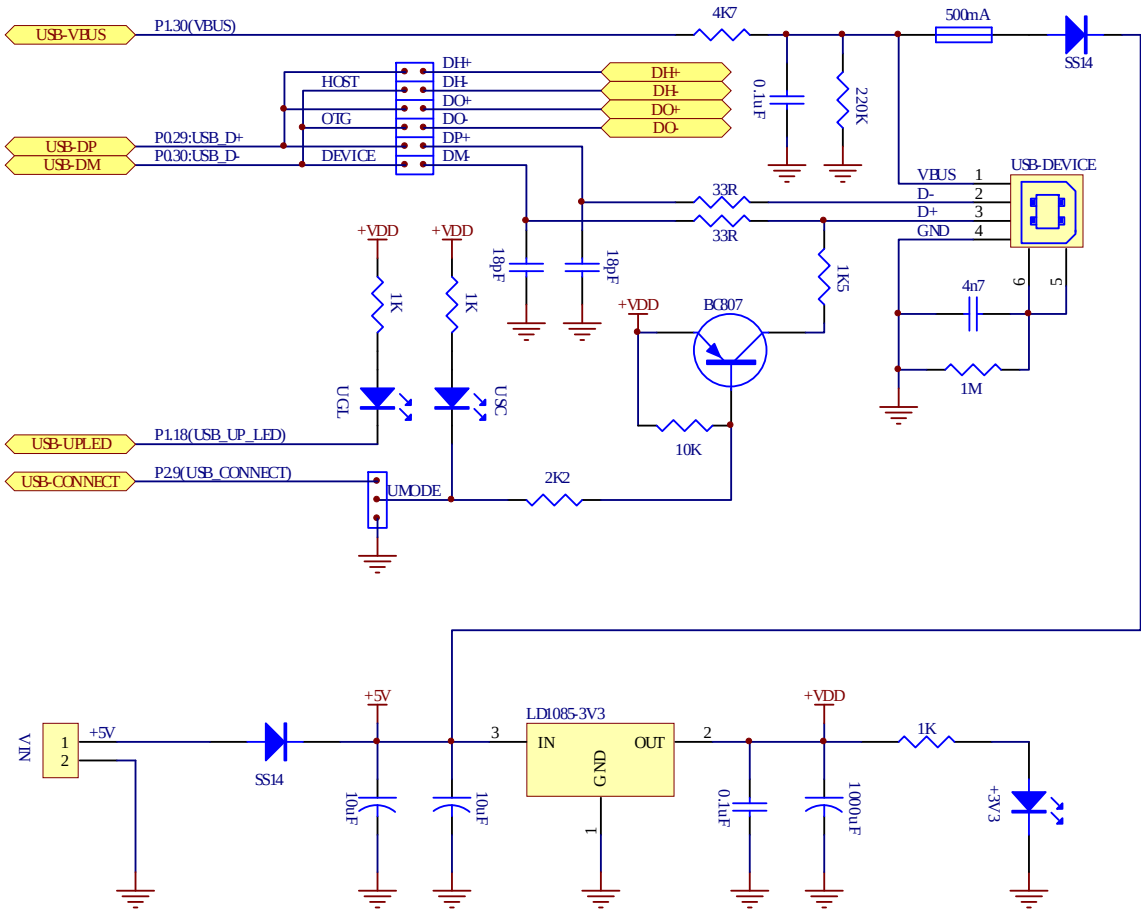
- ขั้วต่อ I2STX เป็น Header ขนาด 1x5 ใช้เป็นจุดเชื่อมต่อ P2[11..13] ซึ่งสามารถใช้ทำหน้าที่เป็น GPIO ทั่วไป หรือใช้ทำหน้าที่เป็น I2STX ได้ตามต้องการ
  - P2.11 = TXCLK
  - P2.12 = TXWS
  - P2.13 = TXSDA



วงจรแหล่งจ่ายไฟ

วงจรแหล่งจ่ายไฟสามารถใช้งานได้กับไฟ DC ขนาด +5V ได้ ซึ่งสามารถต่อไฟเลี้ยงให้บอร์ดได้ทั้งจุดต่อที่เป็น Connector หรือ จะใช้แหล่งจ่ายไฟจาก USB Device ก็ได้ตามต้องการ โดยไฟที่ต่อให้นี้จะถูกส่งต่อไปเข้าวงจร Regulate ขนาด +3V3/3A

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

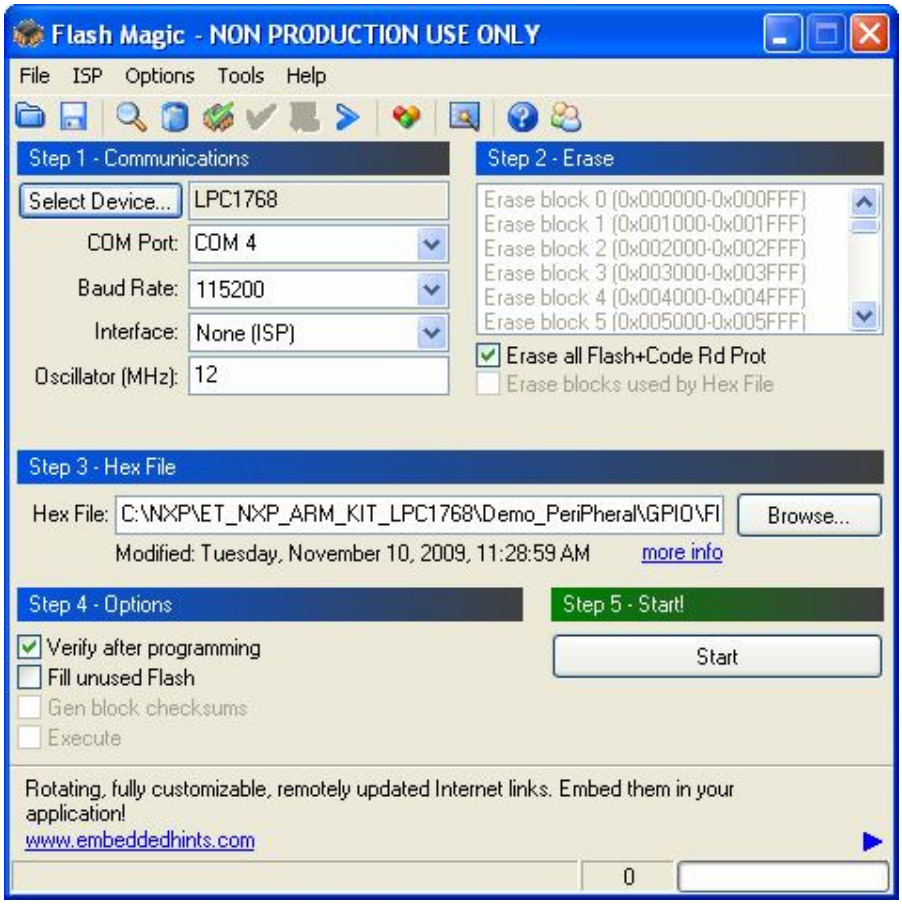


การ Download Hex file ให้กับ MCU ของบอร์ด

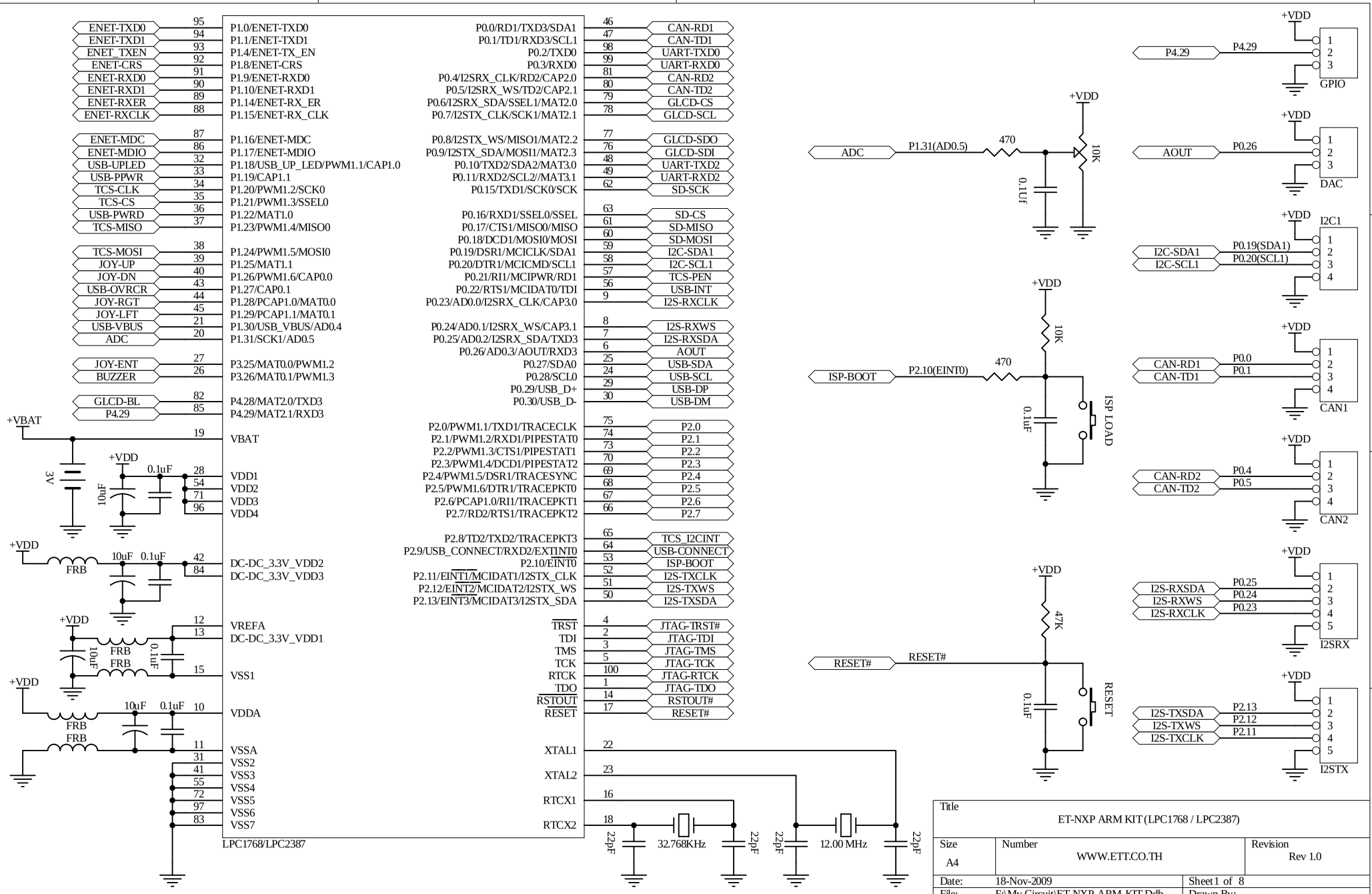
การ Download Hex File ให้กับหน่วยความจำ Flash ของ MCU ในบอร์ดนั้น จะใช้โปรแกรมชื่อ Flash Magic ของ “Embedded System Academy,Inc” ซึ่งจะติดต่อกับ MCU ผ่าน Serial Port ของคอมพิวเตอร์ PC โดยโปรแกรมดังกล่าวสามารถดาวน์โหลดฟรีได้ที่ www.esacademy.com

ขั้นตอนการ Download HEX File ให้กับ MCU

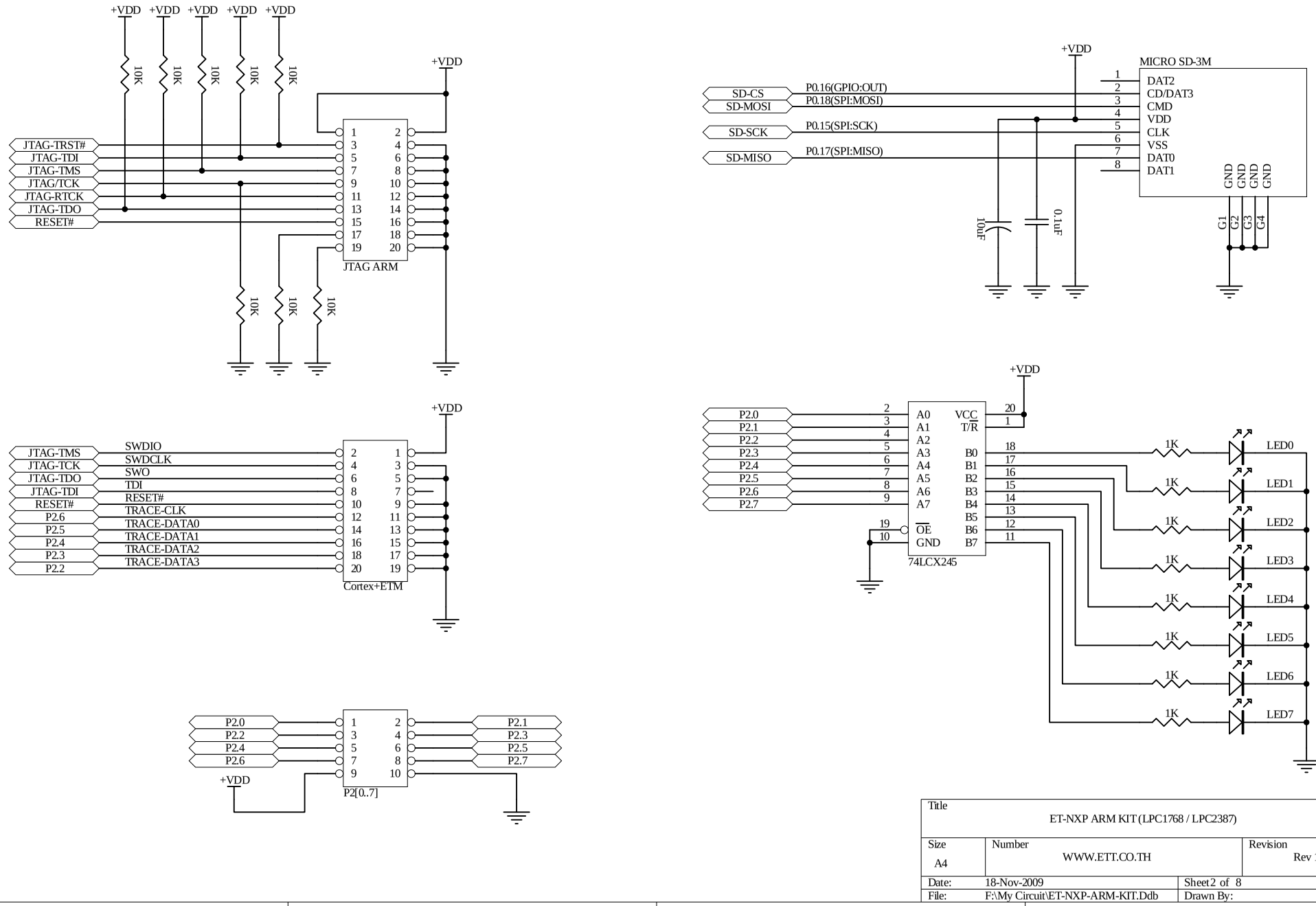
1. ต่อสายสัญญาณ RS232 ระหว่างพอร์ตสื่อสารอนุกรม RS232 ของ PC และบอร์ด UART0
2. จ่ายไฟเลี้ยงวงจรให้กับบอร์ด ซึ่งจะสังเกตเห็น LED PWR ติดสว่างให้เห็น
3. สั่ง Run โปรแกรม Flash Magic ซึ่งถ้าเป็น Version 5.39.1797 จะได้ผลดังรูป



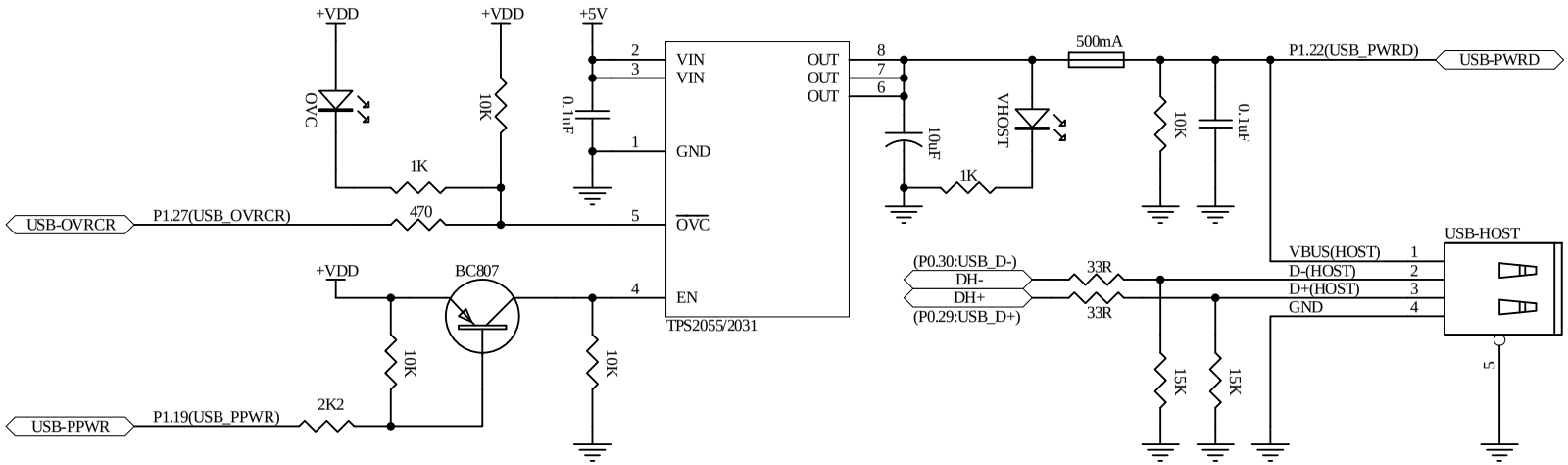
4. เริ่มต้นกำหนดค่าตัวเลือกต่างๆให้กับโปรแกรมตามต้องการ ซึ่งในกรณีนี้ใช้กับ LPC1768 ของบอร์ด ET-NXP ARM KIT(LPC1768) ของ อีทีที ให้เลือกกำหนดค่าต่างๆให้โปรแกรมดังนี้
    - เลือก COM Port ให้ตรงกับหมายเลข COM Port ที่ใช้งานจริง (ในตัวอย่างใช้ COM4)
    - ตั้งค่า Baud Rate อยู่ที่ระหว่าง 2400 - 115200 ซึ่งถ้าเลือกใช้ค่า Baud rate สูงๆ แล้วเกิด Error ให้ลดค่า Baud rate ให้ต่ำลง จากตัวอย่างใช้ค่า 115200
    - กำหนด Device เป็น LPC1768
    - กำหนด Interface เป็น None ISP
    - กำหนดค่าคริสตอล ออสซิลเลเตอร์ ให้ตรงกับที่ใช้ในจริงภายในบอร์ด โดยกำหนดให้มีหน่วยเป็น MHz ในที่นี้ใช้ค่า 12.000MHz ซึ่งต้องกำหนดเป็น 12
    - ให้กดสวิตช์ ISP LOAD และ RESET ที่บอร์ด “ET-NXP ARM KIT” เพื่อทำการ Reset ให้ MCU ทำงานใน Boot Loader ตามขั้นตอนดังต่อไปนี้
        - กดสวิตช์ ISP LOAD ค้างไว้
        - กดสวิตช์ RESET โดยที่สวิตช์ ISP LOAD ยังกดค้างอยู่
        - ปล่อยสวิตช์ RESET โดยที่สวิตช์ ISP LOAD ยังกดค้างอยู่
        - ปล่อยสวิตช์ ISP LOAD เป็นลำดับสุดท้าย
5. เลือกรูปแบบการลบข้อมูลเป็น **“Erase all Flash + Code Rd Prot”**
6. เลือกกำหนด Option เป็น **“Verify after programming”**
7. ให้คลิกเมาส์ที่ **“Browse”** เพื่อทำการเลือกกำหนด HEX File ที่จะทำการสั่ง Download
8. ให้ทำการคลิกเมาส์ที่ **“Start”** ซึ่งโปรแกรม Flash Magic จะเริ่มต้นทำการ Download ข้อมูลให้กับ MCU ทันที โดยสังเกตการทำงานที่ Status bar โดยในขั้นตอนนี้ให้รอจนกว่าการทำงานของโปรแกรมจะเสร็จสมบูรณ์
9. เมื่อทำงานของโปรแกรมเสร็จเรียบร้อยแล้ว ให้กดสวิตช์ Reset ที่บอร์ด ซึ่ง MCU จะเริ่มต้นทำงานตามโปรแกรมที่สั่ง Download ให้ทันที



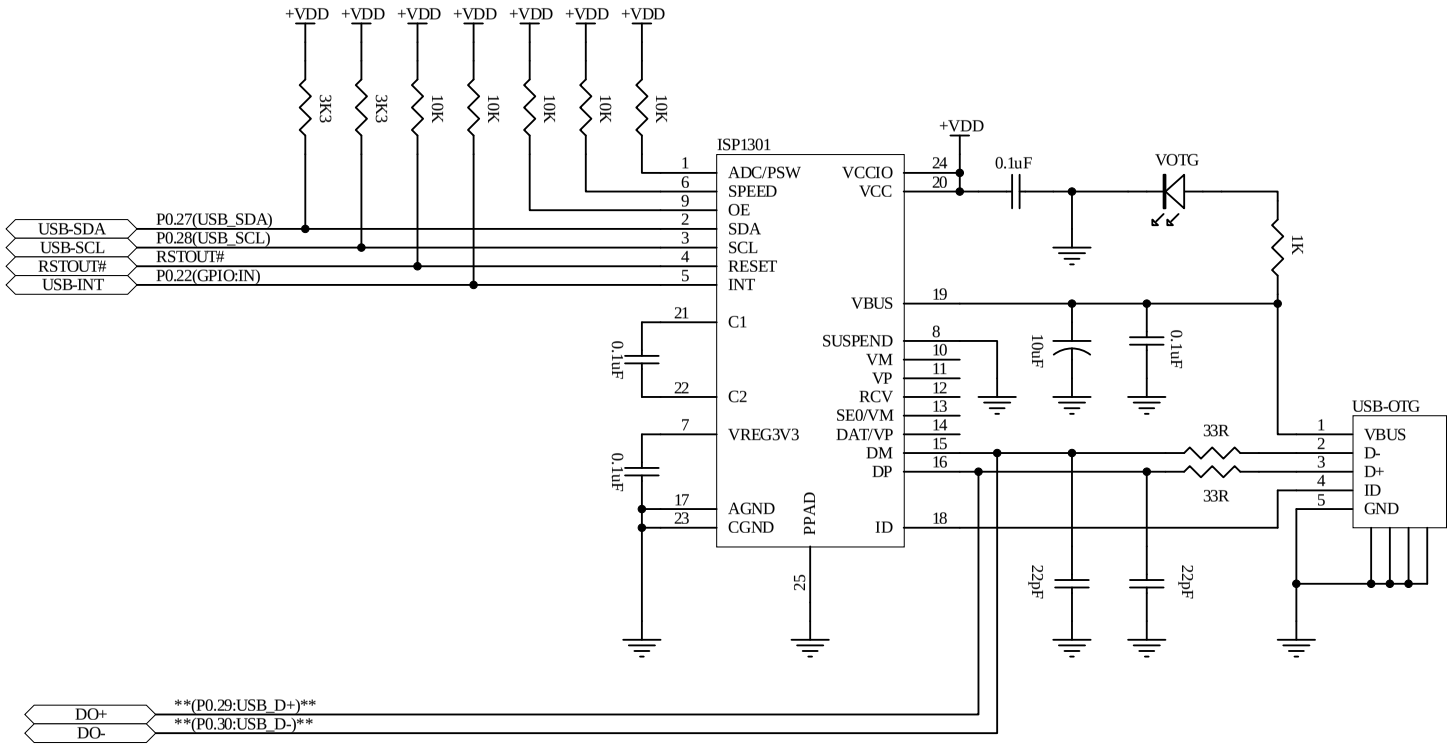
Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet1 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:



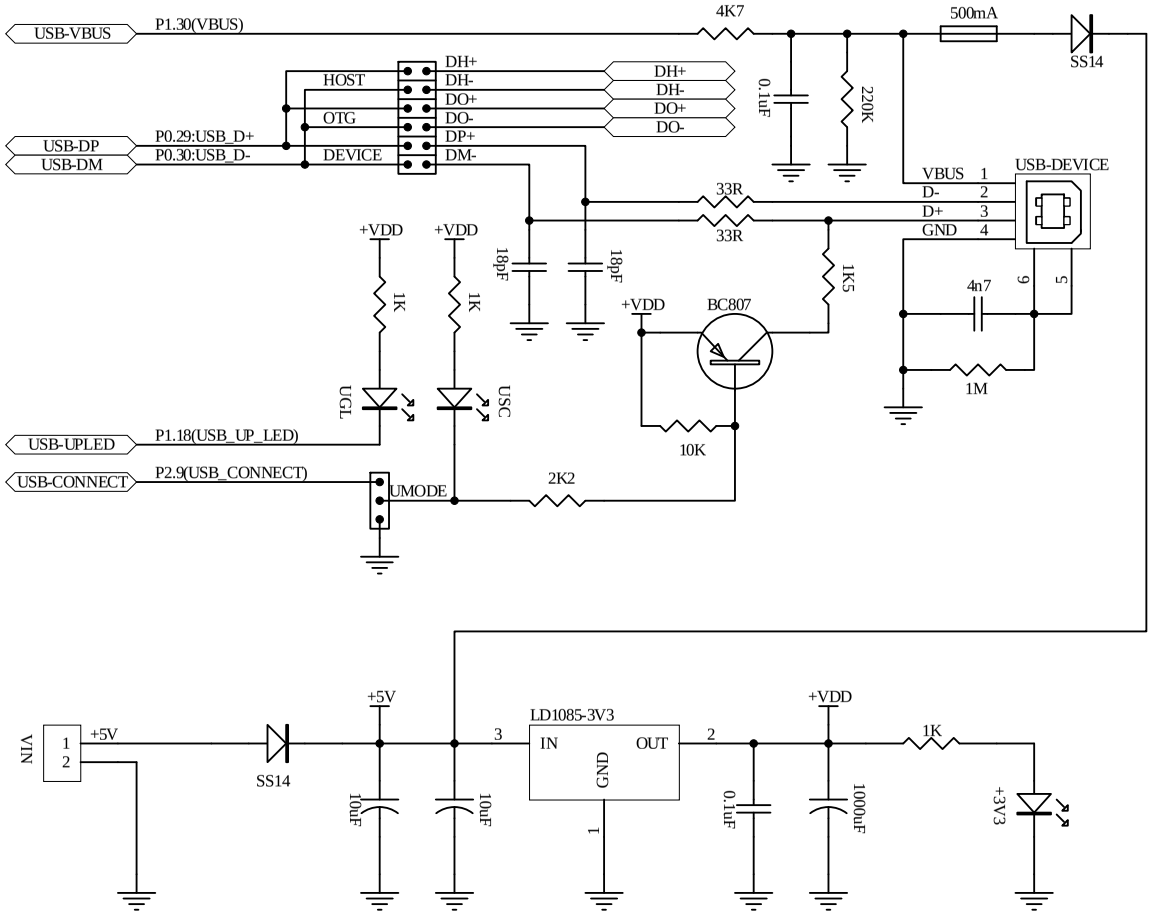
Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet 2 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:



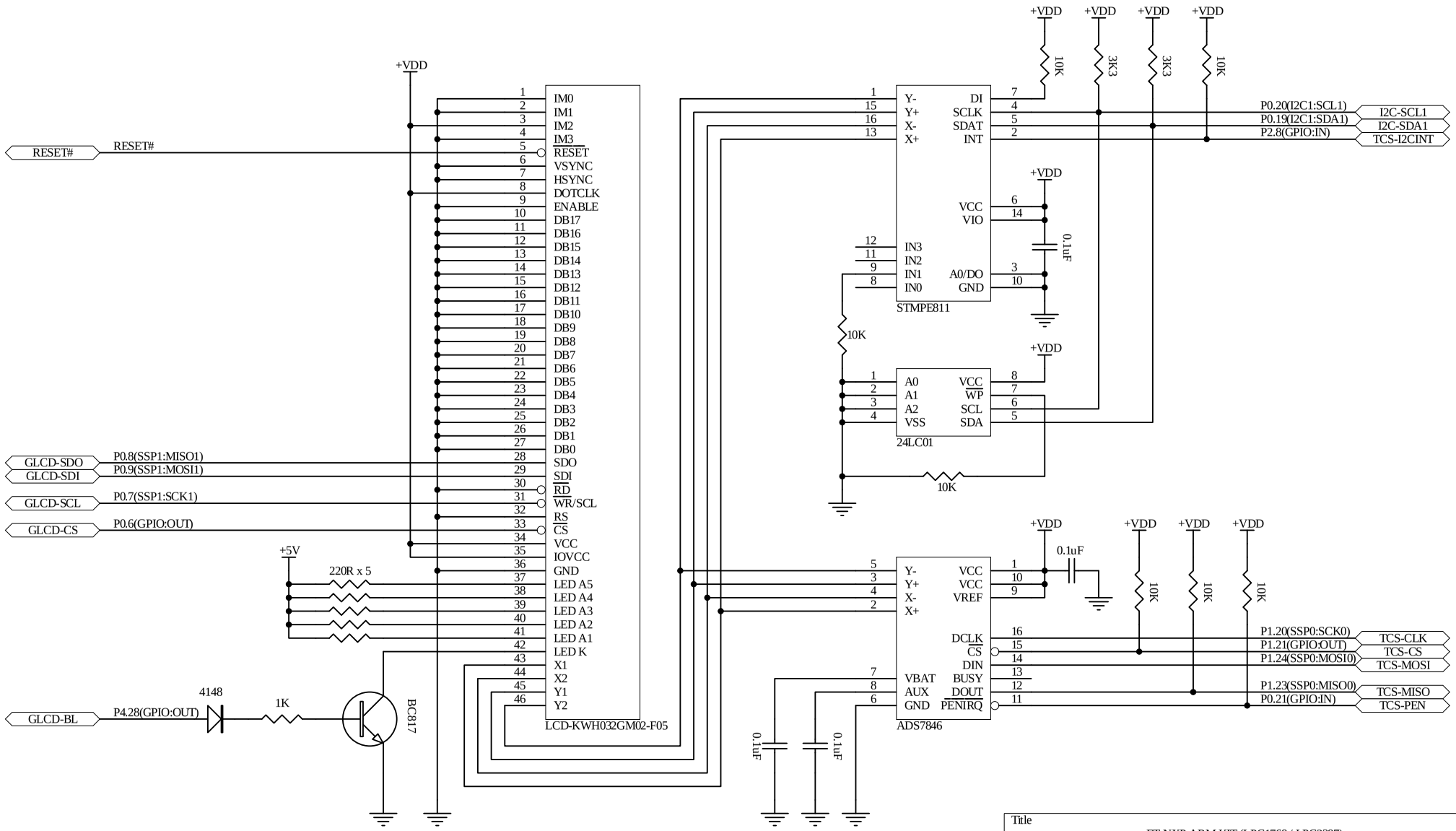
Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet3 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:



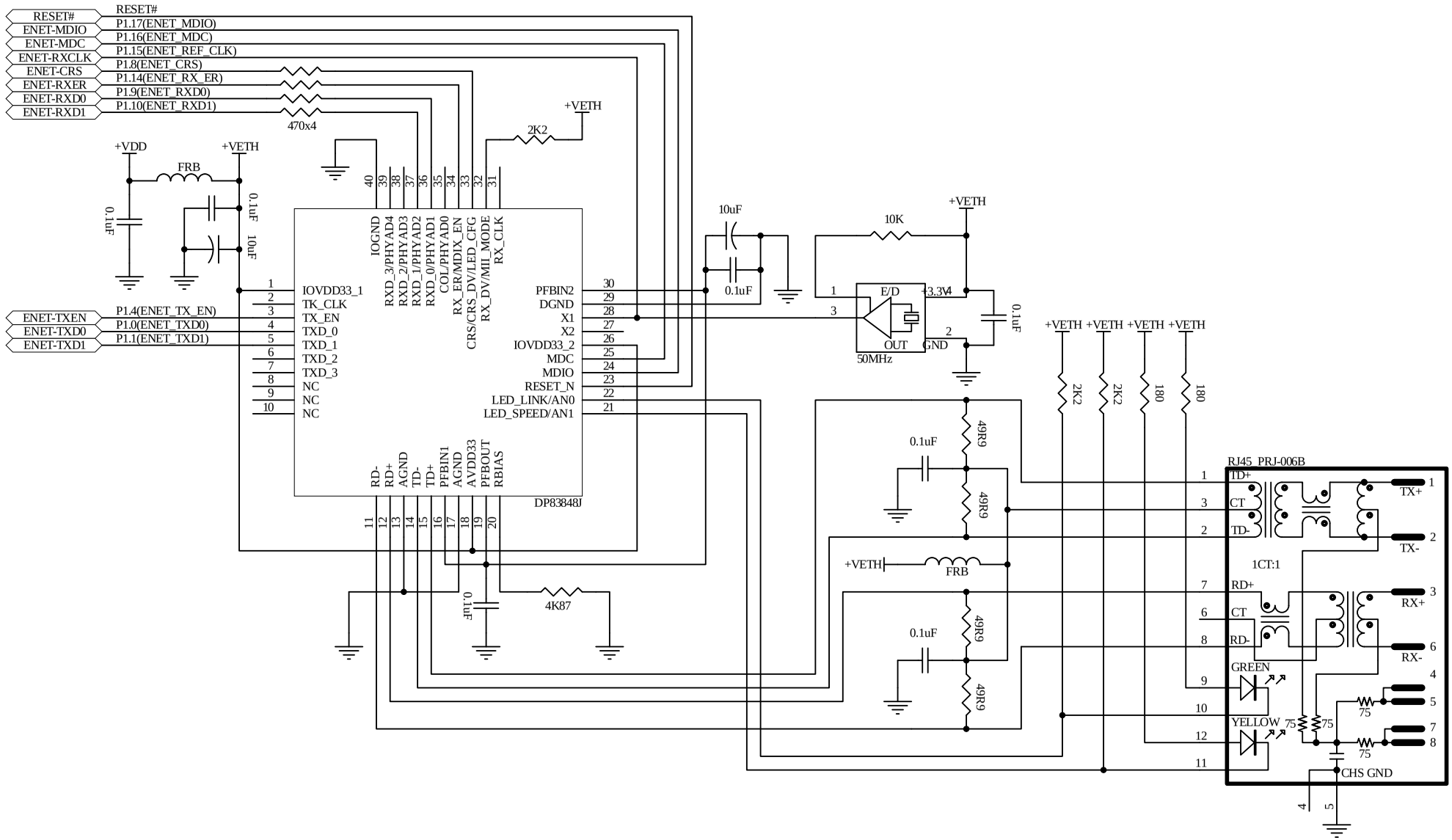
Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet 4 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:

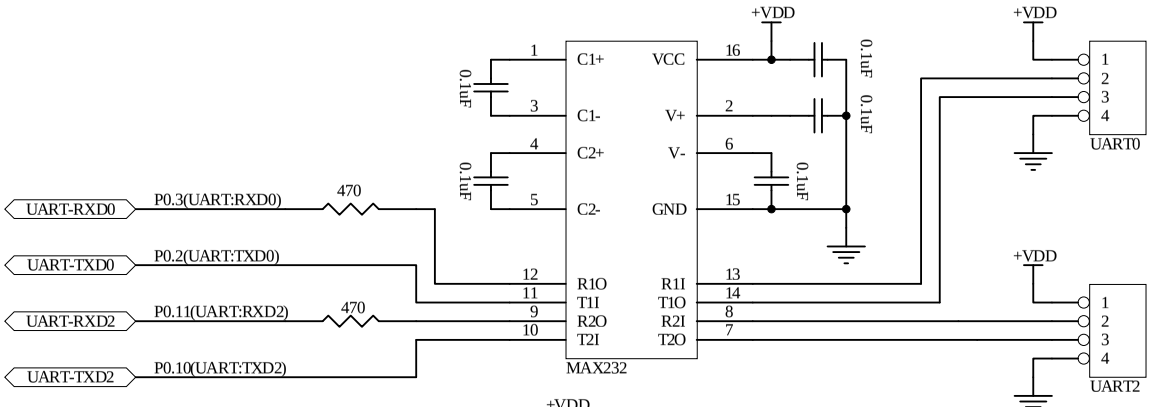


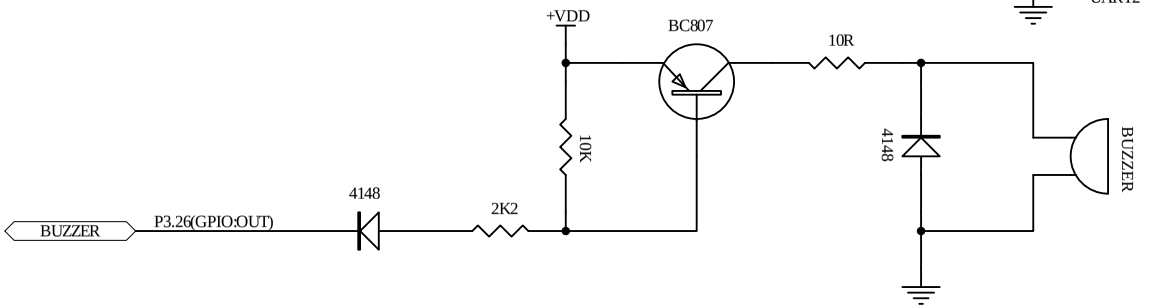
Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet 5 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:

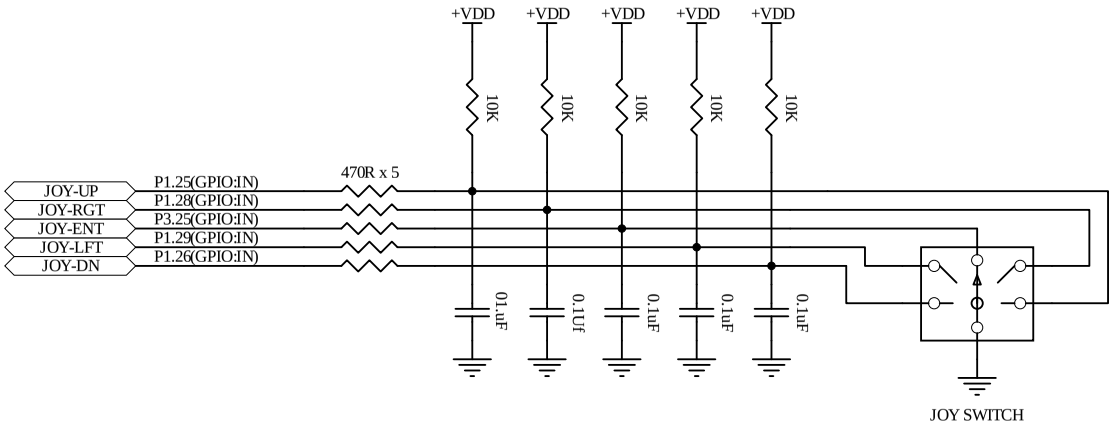


Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 18-Nov-2009		Sheet 6 of 8
File: F:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By:


RESET#
P1.17(ENET_MDIO)
P1.16(ENET_MDC)
P1.15(ENET_REF_CLK)
P1.8(ENET_CRS)
P1.14(ENET_RX_ER)
P1.9(ENET_RXD0)
P1.10(ENET_RXD1)
470x4
+VETH
2K2
+VDD
FRB
0.1uF
10uF
IOVDD33_1
TK_CLK
TX_EN
TXD_0
TXD_1
TXD_2
TXD_3
NC
NC
NC
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
X1
X2
IOVDD33_2
MDC
MDIO
RESET_N
LED_LINK/AN0
LED_SPEED/AN1
RD-
RD+
AGND
TD-
TD+
PFBIN1
AGND
AVDD33
PFBOUT
RBIAS
DP83848J
ENET-TXEN
ENET-TXD0
ENET-TXD1
P1.4(ENET_TX_EN)
P1.0(ENET_TXD0)
P1.1(ENET_TXD1)
4K87
10K
E/D
+3.3V
OUT
GND
50MHz
49R9
180
RJ45_PRJ-006B
TD+
CT
TD-
RD+
CT
RD-
GREEN
YELLOW
75
TX+
TX-
RX+
RX-
1CT:1
CHS GND
Title
ET-NXP ARM KIT (LPC1768 / LPC2387)
Size
A4
Number
WWW.ETT.CO.TH
Revision
Rev 1.0
Date:
18-Nov-2009
Sheet 7 of 8
File:
F:\My Circuit\ET-NXP-ARM-KIT.Ddb
Drawn By:







Title	ET-NXP ARM KIT (LPC1768 / LPC2387)	
Size: A4	Number: WWW.ETT.CO.TH	Revision: Rev 1.0
Date: 15-Dec-2009		Sheet 8 of 8
File: G:\My Circuit\ET-NXP-ARM-KIT.Ddb		Drawn By: