บทความ ARM7
ตอนที่ 3 เตือนสติด้วย ARM7 ผ่าน RTC

ศุภชัย บุศราทิจ (raek@etteam.com)
คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏเพชรบุรี
วันศุกร์ที่ 26 มกราคม พ.ศ. 2550

บทนำ

หลังจากผ่านช่วงหักเหของชีวิตครั้งแล้วครั้งเล่าสิ่งที่ทำให้ผมดำรงอยู่และกล้าที่จะเผชิญก็คือการใช้สติพิจารณาตามหลักพุทธศาสนา และผมขอขอบพระคุณ "ดังตฤณ" เป็นอย่างสูงที่เป็นผู้หนึ่งที่ส่งเสริมให้พบธรรมดังกล่าว โดยเฉพาะความกรุณาในเรื่องของการเผยแพร่หนังสือในเว็บ dungtrin.com ซึ่งเป็นประโยชน์อย่างสูง และสิ่งหนึ่งที่ผมนำมาใช้คือระบบเตือนสติของตน ซึ่งเป็นสิ่งที่ผมกระทำตามคำแนะนำในหนังสือที่ได้บอกว่าให้หานาฬิกาปลุกที่สามารถเตือนได้ทุก 2 นาที เมื่อเตือนในแต่ละครั้งก็ให้ถามตัวเองว่าหายใจด้วยลมหายใจแบบใดและรู้สึกอย่างไร

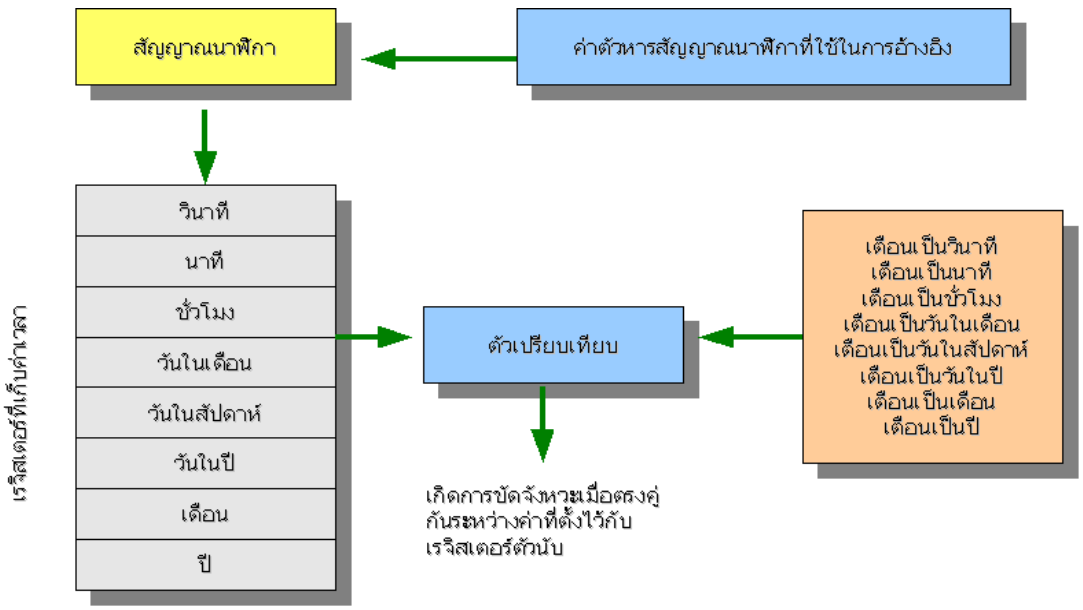
ด้วยเหตุนี้ผมเลยมีความตั้งใจว่าจะลองใช้นาฬิกาเวลาจริง (RTC : real time clock) ตั้งเวลาไว้ที่ 2 นาที แล้วรายให้หลอดแอลอีดีกะพริบ 2 ครั้งในทุก 2 นาที ทั้งที่แต่เดิมนั้นตั้งใจเอาไว้ว่าจะทำตัวอย่างนับเวลาถอยหลัง 10 วินาที คือ ลดจาก 9 เป็น 8 เป็น 7 ไปเรื่อย ๆ จนถึง 0 ซึ่งขอเลื่อนไปเป็นครั้งหน้า แต่ถ้าใครจะลองทำก่อนก็ไม่น่าจะมีอุปสรรคใด เพราะถ้าสามารถควบคุมการทำงานของระบบนาฬิกาเวลาจริงได้ เราจะสามารถเปรียบเทียบเวลาที่เปลี่ยนแปลงไปในแต่ละวินาทีได้ หลังจากนั้นก็แสดงผลออกไปที่อุปกรณ์เซเว่นเซ็กเมนต์ (7-segment) แต่ถ้าไม่รีบร้อนอะไร ครั้งหน้าผมจะเขียนบทความนี้ให้ได้อ่านกันเป็นกรณีศึกษา

อุปกรณ์ประกอบการทดลอง

1. บอร์ด ET-Base ARM7 LPC2103
2. บอร์ด ET-TEST 10P/OUT
3. บอร์ด ET-USB/RS232 สำหรับแปลงพอร์ต USB ให้เป็น RS232
4. โปรแกรม Keil uVision3 (mdk304) และ Keil-GCC/ARM 3.3.1

หลักการทำงานของนาฬิกาเวลาจริง

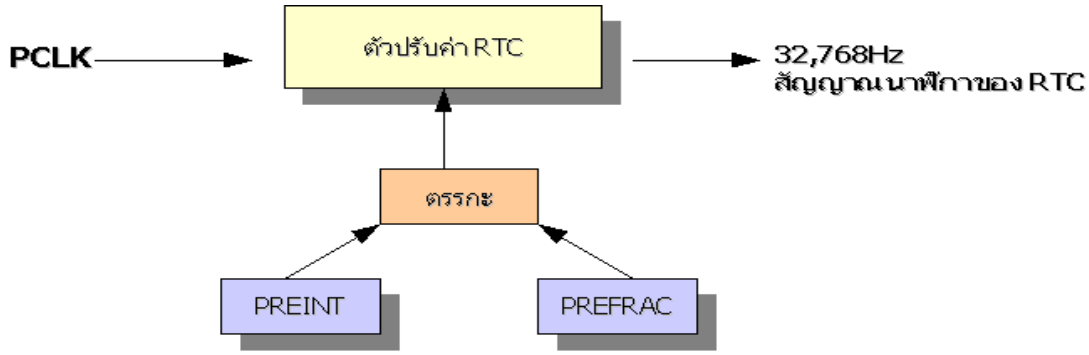
รูปที่ 1 ผังอธิบายภายในของโมดูลนาฬิกาเวลาจริง



จากรูปที่ 1 การทำงานของสัญญาณนาฬิกานั้นจะเกิดจากค่าตัวหารสัญญาณนาฬิกาที่ใช้ในการอ้างอิงเป็นตัวกระตุ้นการนับสัญญาณนาฬิกา หลังจากนั้นจะมีการบันทึกการนับค่าลงในเรจิสเตอร์ที่ทำหน้าที่เก็บค่าของวินาที นาที ชั่วโมง วันในเดือน วันในสัปดาห์ วันในปี เดือน และปี ซึ่งเราสามารถนำค่าเหล่านี้มาทำการเปรียบเทียบกับค่าที่เรากำหนด โดยตัวเปรียบเทียบนี้จะสร้างสัญญาณการขัดจังหวะเมื่อค่าในเรจิสเตอร์ตรงตามที่เรากำหนดเอาไว้

ข้อกำหนดของนาฬิกาเวลาจริงในอาร์ม 7 คือ ถ้าไม่ใส่ถ่านสำหรับเก็บค่าเวลาจะไม่สามารถจำเวลาที่ตั้งเอาไว้ได้ และค่าปีสูงสุดที่สามารถเก็บได้นั้นเป็นปี ค.ศ. 2099

รูปที่ 2 ผังเรจิสเตอร์ในการสร้างความถี่สัญญาณนาฬิกาสำหรับนาฬิกาเวลาจริง



จากรูปที่ 2 การทำงานของนาฬิกาเวลาจริงจะนำค่าความถี่จาก PCLK มาทำการหารค่าที่กำหนดเพื่อให้ได้ความถี่ของสัญญาณนาฬิกาขาออกเป็น 32,768Hz ซึ่งการทำงานของการหารความถี่นั้นจะเกิดจากการทำตรรกะระหว่างเรจิสเตอร์ PREINT และ PREFRAC ดังสมการต่อไปนี้

PREINT = (int)(PCLK / 32768) – 1
PREFRAC = PCLK – ((PREINT+1) * 32768)

จากสมการทั้งสองทำให้เราทราบว่า PREINT นั้นเก็บจำนวนเต็มของการหาร PCLK ด้วย 32768 และ PREFRAC เก็บเศษจากการหาร PCLK ด้วย 32768 ด้วยเหตุนี้เมื่อนำมาใช้กับบอร์ดของอีทีทีซึ่งใช้สัญญาณนาฬิกาเป็น 19.6608 MHZ ใช้วงจร PLL คูณ 3 ทำให้เกิดสัญญาณที่มีความถี่ดังนี้

CCLK = 19.6608 * 3
= 58.9824 Mhz
PCLK = CCLK / 2
= 58.9824 / 2
= 29.4912 Mhz

เมื่อนำมาหาค่า PREINT และ PREFRAC จะได้ค่าดังต่อไปนี้

PREINT = (int)(29491200 / 32768) -1
= 899
PREFRAC = 299491200 – ((899 + 1) * 32768)
= 0

ในตารางที่ 1 เป็นรายชื่อเรจิสเตอร์ทั้ง 26 ตัวที่ทำหน้าที่เกี่ยวกับนาฬิกาเวลาจริง ซึ่งทุกตัวยกเว้น PREINT และ PREFRAC จะไม่ถูกล้างค่าหลังจากมีการรีเซ็ตระบบ

เมื่อได้ค่าเริ่มต้นของการทำงานครบถ้วน ขั้นตอนต่อไปเป็นสั่งเริ่มการทำงานของนาฬิกาเวลาจริง ดังนั้น โค้ดของโปรแกรมที่เป็นส่วนของการเริ่มต้นการทำงานจึงเขียนได้ดังต่อไปนี้

```
/* กำหนดค่าตัวปรับค่านาฬิกาเวลาจริง */
PREINT = 899
PREFRAC = 0
/* เริ่มต้นทำงานนาฬิกาเวลาจริง */
CCR = 1;
```


เขียนโดย ศุภชัย บุศราทิจ (raek@etteam.com) คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏเพชรบุรี

ตารางที่ 1 รายการเรจิสเตอร์ที่เกี่ยวข้องกับนาฬิกาเวลาจริง

ชื่อ	ขนาด (บิต)	ความหมาย	การติดต่อ	ค่าหลังการรีเซ็ต	ตำแหน่ง
ILR	**2**	Interrupt Location Register	อ่าน/เขียน	**-**	0xE0024000
CTC	**15**	Clock Tick Counter	อ่าน	**-**	0xE0024004
CCR	**4**	Clock Control Register	อ่าน/เขียน	**-**	0xE0024008
CIIR	**8**	Counter Increment Interrupt Register	อ่าน/เขียน	**-**	0xE002400C
AMR	**8**	Alarm Mask Register	อ่าน/เขียน	**-**	0xE0024010
CTIME0	**(32)**	Consolidated Time Register 0	อ่าน	**-**	0xE0024014
CTIME1	**(32)**	Consolidated Time Register 1	อ่าน	**-**	0xE0024018
CTIME2	**(32)**	Consolidated Time Register 2	อ่าน	**-**	0xE002401C
SEC	**6**	Seconds Register	อ่าน/เขียน	**-**	0xE0024020
MIN	**6**	Minutes Register	อ่าน/เขียน	**-**	0xE0024024
HOUR	**5**	Hours Register	อ่าน/เขียน	**-**	0xE0024028
DOM	**5**	Day of Month Register	อ่าน/เขียน	**-**	0xE002402C
DOW	**3**	Day of Week Register	อ่าน/เขียน	**-**	0xE0024030
DOY	**9**	Day of Year Register	อ่าน/เขียน	**-**	0xE0024034
MONTH	**4**	Months Register	อ่าน/เขียน	**-**	0xE0024038
YEAR	**12**	Years Register	อ่าน/เขียน	**-**	0xE002403C
ALSEC	**6**	Alarm Value for Seconds	อ่าน/เขียน	**-**	0xE0024060
ALMIN	**6**	Alarm Value for Minutes	อ่าน/เขียน	**-**	0xE0024064
ALHOUR	**5**	Alarm Value for Hours	อ่าน/เขียน	**-**	0xE0024068
ALDOM	**5**	Alarm Value for Day of Month	อ่าน/เขียน	**-**	0xE002406C
ALDOW	**3**	Alarm Value for Day of Week	อ่าน/เขียน	**-**	0xE0024070
ALDOY	**9**	Alarm Value for Day of Year	อ่าน/เขียน	**-**	0xE0024074
ALMON	**4**	Alarm Value for Month	อ่าน/เขียน	**-**	0xE0024078
ALYEAR	**12**	Alarm Value for Year	อ่าน/เขียน	**-**	0xE002407C
PREINT	**13**	Prescale value, integer portion	อ่าน/เขียน	**0**	0xE0024080
PREFRAC	**15**	Prescale value, fractional portion	อ่าน/เขียน	**0**	0xE0024084

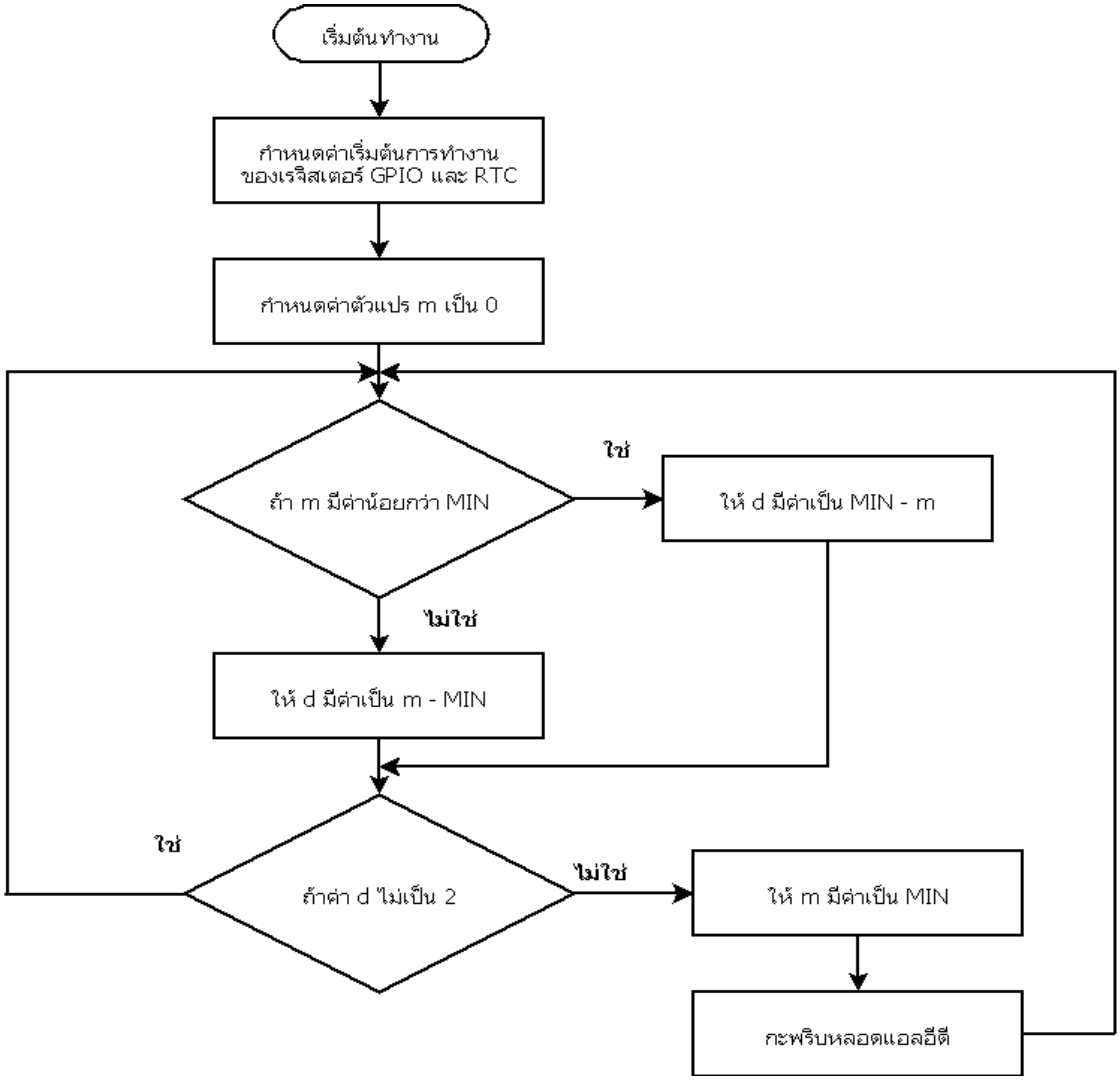
ออกแบบโปรแกรม

หลักการทำงานของโปรแกรมในตัวอย่างครั้งนี้ คือ กำหนดให้นาฬิกาเวลาจริงทำงาน หลังจากนั้นตรวจสอบว่าเวลาผ่านไป 2 นาทีแล้วหรือไม่ ถ้าใช่ก็จะกะพริบหลอดแอลอีดี 1 ครั้ง และทำอย่างนี้วนไปเรื่อย ๆ ดังการทำงานในรูปที่ 3 และแปลงเป็นโค้ดโปรแกรมภาษาซีได้ดังโค้ดที่ 1

ผลการทดลอง

ในการต่อวงจรนั้นจะเชื่อมต่อสายสัญญาณบอร์ด ET-Base ARM7 LPC2103 จากขา P0.8 ถึง P0.15 ไปที่บอร์ด ET-TEST 10P/OUT ซึ่งเป็นหลอดแอลอีดีจำนวน 8 หลอด พบว่าโปรแกรมสามารถทำงานได้ถูกต้องตามความต้องการ

รูปที่ 3 ผังการทำงานของโปรแกรม



สรุป

จากบทความนี้ผู้เขียนหวังว่าคงเป็นประโยชน์ต่อการทำความเข้าใจกับการทำงานของนาฬิกาเวลาจริงในลักษณะของการกำหนดค่า อ่านค่า และสั่งเริ่มต้นทำงาน ในบทความครั้งต่อไปจะเป็นการนำไปใช้กับการขัดจังหวะและเปลี่ยนอุปกรณ์แสดงผลจากหลอดแอลอีดีมาเป็นเซเว่นเซ็กเมนต์

ในโค้ดตัวอย่างที่ 2 จะเป็นการใช้นาฬิกาเวลาจริงที่ทางอีทีทีเขียนเอาไว้เป็นแนวทางสำหรับแจกจ่ายมาพร้อมบอร์ดทดลองอาร์ม 7 ผมก็เลยนำมาแปะไว้ในบทความนี้ด้วยเพื่อจะได้ใช้ประกอบทำความเข้าใจเป็นอีกแนวหนึ่ง

สุดท้ายนี้ต้องขอขอบคุณทางทีมงานอีทีที คุณกอบกิจ เติมผาติ และครอบครัวที่ยังเป็นกำลังใจสำคัญในการเขียนบทความของผม เพราะบุคคลเหล่านี้ไม่เห็นความสำคัญของงานที่ผมทำ ก็คงไม่มีบทความเหล่านี้ให้อ่าน และผมก็คงไม่ได้เขียนอะไรที่ผมได้ศึกษาค้นคว้าแล้วนำมาเผยแพร่ให้อ่านกัน

หมายเหตุ ขอขอบคุณ OpenOffice.org / Nvu / Dia / TheGIMP / FileZilla / Keil ARMGCC ที่ออกแบบเครื่องมือดี ๆ ที่ช่วยลดค่าใช้จ่ายในการซื้อซอฟต์แวร์ลิขสิทธิ์ของผมได้

โค้ดที่ 1 โปรแกรมกะพริบหลอดแอลอีดีทุก 2 นาที

```
#include <lpc21xx.h>

int main(void)
{
   int d,m,i,j;
   m = 0;

   PINSEL0 = 0x00000005;
   IODIR0 = 0x0000FF00; /* P0.8 ถึง P0.15 เป็นขาน่าออก */
   IOCLR0 = 0x0000FF00; /* สั่งให้หลอดแอลอีดีดับ */
   PREINT = 899;
   PREFRAC = 0;
   CCR = 1; /* นาฬิกาเวลาจริงเริ่มทำงาน */

   do {
      do {
         if (m < MIN) {
            d = MIN – m;
         } else {
            d = m – MIN;
         }
      } while (d != 2);
      m = MIN;
      IOSET0 = 0x0000FF00; /* สั่งให้หลอดแอลอีดีสว่าง */
      for (i=0; i<1000; i++) {
         for (j=0; j<1000; j++);
      }
      IOCLR0 = 0x0000FF00;
   } while (1); /* วนรอบเรื่อย ๆ ไม่มีวันจบการทำงาน */
   return 1;
}
```

โค้ดที่ 2 โปรแกรมตัวอย่างการใช้นาฬิกาเวลาจริงของ LPC2103

```
/*****************************************************/
/* Examples Program For "ET-ARM7 BASE LPC2103" Board */
/* Target MCU  : Philips ARM7-LPC2103                */
/*           : X-TAL : 19.6608 MHz                   */
/*           : Run Speed 58.9824MHz (With PLL)       */
/* Keil Editor : uVision3 V3.03a                     */
/* Compiler    : Keil CARM V2.50a                    */
/* Create By   : Eakachai Makarn (WWW.ETT.CO.TH)     */
/* Last Update : 17/April/2006                       */
/* Function    : Example Read RTC + Display on UART0 */
/*****************************************************/
// Read/Write Internal RTC of LPC2103
// Display Result on UART0(9600,N,8,1)

#include <LPC2103.H>          // LPC2103 MPU Register
#include <stdio.h>            // For Used Function printf

#define MASKSEC 0x3F          // Second 00..59   00000000:00000000:00xxxxxx
```

```
#define MASKMIN 0x3F00        // Minute 00..59   00000000:00xxxxxx:00000000
#define MASKHR  0x1F0000      // Hour 00..23     000xxxxx:00000000:00000000

/* pototype  section */
void init_serial0 (void);           // Initil UART-0
int putchar (int ch);               // Put Char to UART-0
int getchar (void);                 // Get Char From Uart-0

int main(void)
{
  unsigned char Hour,Minute,Second,Last_Second;       // RTC Buffer Data
  init_serial0();                                     // Initilial UART0 = 9600,N,8,1
  printf("\n\n\nET-ARM7 STAMP LPC2103...TEST RTC\n"); // Call prinff Function

  // Initial Internal RTC Function
  CCR &= 0x00;          // Reset All Bit
  CCR |= 0x10;          // CLKSRC = 1 = Used EXT 32.768 KHz
  CCR |= 0x02;          // Reset Clock     (0000 0010)
  CCR &= 0xFD;          // Release Reset  (1111 1101)
  CCR |= 0x01;          // Start RTC Clock
  CCR  = 0x11;          // Start RTC Clock Used EXT 32.768 KHz

  // Setup Start Time For RTC = 00:00:00
  HOUR = 0x00;
  MIN  = 0x00;
  SEC  = 0x00;
  Last_Second = 0x00;

  // Start Test Read RTC and Display on UART0 //
  while(1)
  {
    do                  // Repeat Get Second until Second Change
        {
          Hour   = (CTIME0 & MASKHR)>>16;  // Read Hour
          Minute = (CTIME0 & MASKMIN)>>8;  // Read Minute
          Second = CTIME0 & MASKSEC;       // Read Second
    }
    while(Last_Second == Second);          // Repeat if Second Not Change

    Last_Second = Second;                  // Update Current Second

    //************************************//
    // Display Clock = Hour:Minute:Second //
    //************************************//
    printf("\rReal Time Clock = ");         // Print Message String
        printf(" %2d : %2d : %2d",Hour,Minute,Second);
  }
}
```

```
/*****************************/
```

```
/* Initial UART0 = 9600,N,8,1 */
/* VPB(pclk) = 29.4912 MHz    */
/*****************************/
void init_serial0 (void)
{
  PINSEL0 &= 0xFFFFFFF0;                // Reset P0.0,P0.1 Pin Config
  PINSEL0 |= 0x00000001;                // Select P0.0 = TxD(UART0)
  PINSEL0 |= 0x00000004;                // Select P0.1 = RxD(UART0)

  U0LCR &= 0xFC;                        // Reset Word Select(1:0)
  U0LCR |= 0x03;                        // Data Bit = 8 Bit
  U0LCR &= 0xFB;                        // Stop Bit = 1 Bit
  U0LCR &= 0xF7;                        // Parity = Disable
  U0LCR &= 0xBF;                        // Disable Break Control
  U0LCR |= 0x80;                        // Enable Programming of Divisor Latches

  // U0DLM:U0DLL = 29.4912MHz / [16 x Baud]
  //             = 29.4912MHz / [16 x 9600]
  //             = 192 = 0x00C0
  U0DLM = 0x00;                         // Program Divisor Latch(192) for 9600 Baud
  U0DLL = 0xC0;
  U0LCR &= 0x7F;                        // Disable Programming of Divisor Latches
  U0FCR |= 0x01;                        // FIF0 Enable
  U0FCR |= 0x02;                        // RX FIFO Reset
  U0FCR |= 0x04;                        // TX FIFO Reset
  U0FCR &= 0x3F;
}

/*****************************/
/* Write Character To UART0 */
/*****************************/
int putchar (int ch)
{
  if (ch == '\n')
  {
    while (!(U0LSR & 0x20));      // Wait TXD Buffer Empty
    U0THR = 0x0D;             // Write CR
  }
  while (!(U0LSR & 0x20));        // Wait TXD Buffer Empty
  return (U0THR = ch);            // Write Character
}

/*****************************/
/* Read Character From UART0 */
/*****************************/
int getchar (void)
{
  while (!(U0LSR & 0x01));  // Wait RXD Receive Data Ready
  return (U0RBR);         // Get Receice Data & Return
}
```