Enviando Dados Pela Porta Serial na placa LaunchPad MSP430 Utilizando o ambiente ENERGIA

Autor: Renne Takao Meguro Portal

Ribeirão Preto 14/07/2012

Olá Pessoal, dando continuidade ao artigo do Blog do nosso amigo Terry Laundos (Terryvel), resolvi explorar algumas funcionalidades do ambiente ENERGIA (http://energia.github.com/Energia/) que é sem duvida nenhuma uma mão na roda pra desenvolvimento de protótipos rápidos com a Launchpad, na verdade foi uma customização do ambiente do arduino voltado para nossa placa.

O compilador que este ambiente utiliza é o MSPGCC , isto significa que seu uso é livre depende só da imaginação e capacidade de memoria do microcontrolador MSP430 disponível para comportar seu firmware.

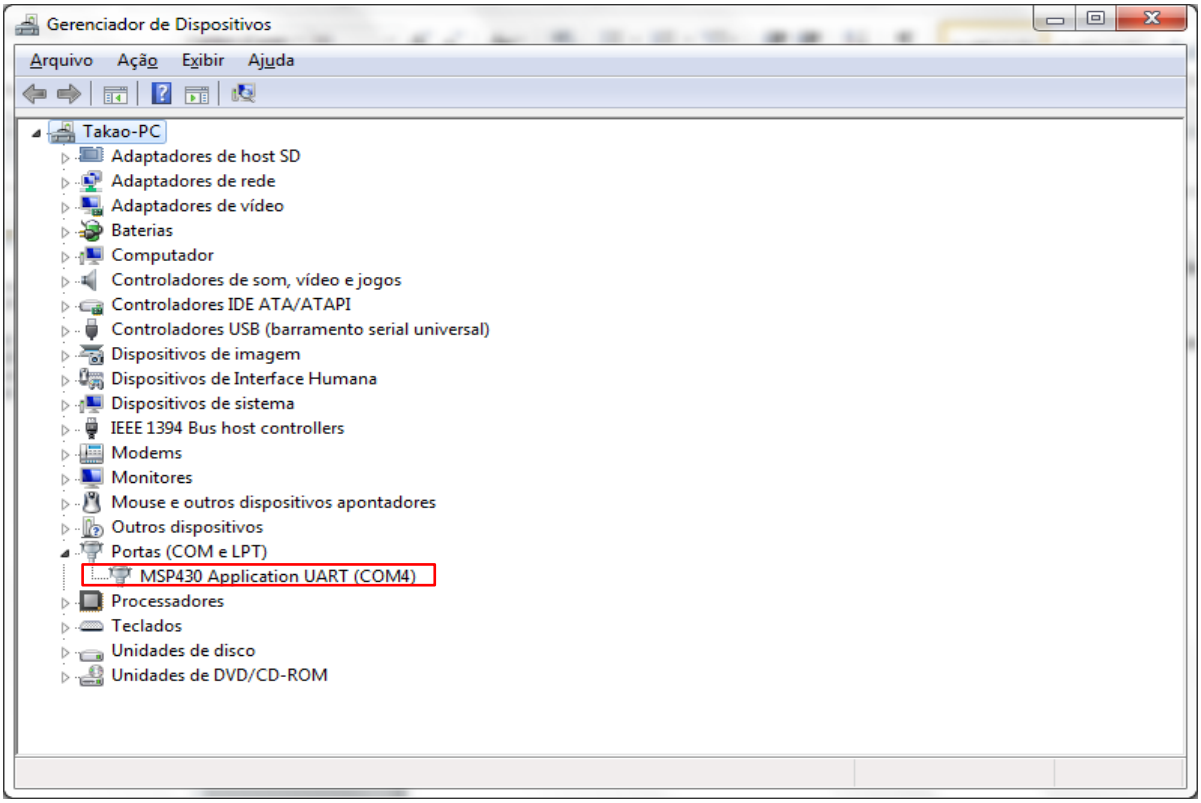
Então vamos para nossa aplicação exemplo bem simples.

Antes de qualquer coisa você vai precisar ter o driver da Lauchpad instalado, eu instalei o CCS 5 disponivel no site da Texas, pois ele já instala dos drivers da placa.

Após instalado voce deve verificar se a sua launchpad está conectada e operante no seu sistema. No meu caso eu utilizo o Windows 7.

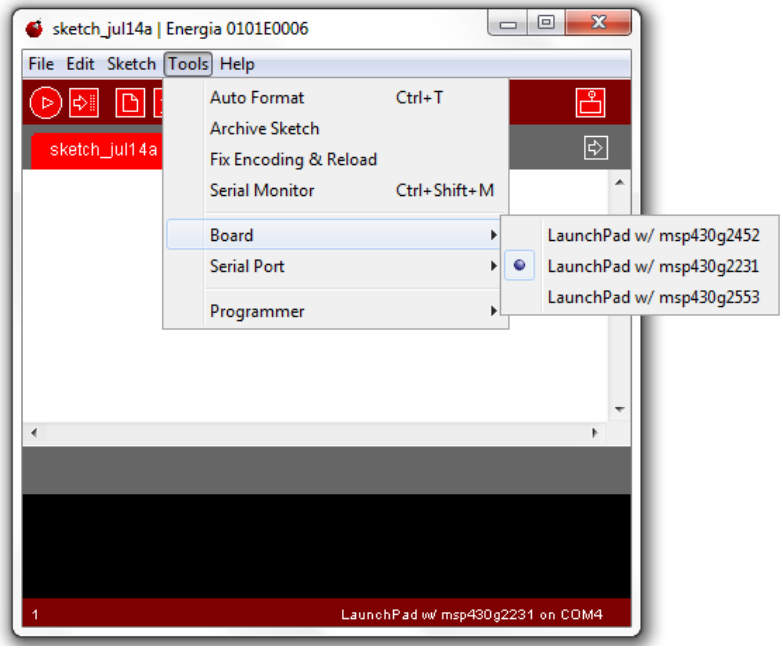
Para verificar se sua launchpad está ok verifique se ela está configurada em uma porta COM conforme ilustrada na figura abaixo:

Painel de Controle -> Sistema -> Gerenciador de Dispositivos



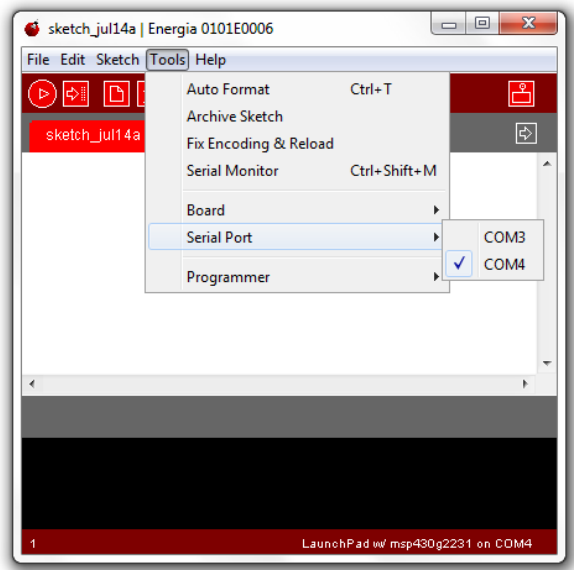
Estando Ok, agora vamos para o próximo passo. Agora é configurar o ENERGIA para trabalhar com sua Placa.

Primeiramente devemos setar o nosso microcontrolador presente na Launchpad. No meu caso utilizo o **MSP430g2231**



Após setado o microcontrolador vamos setar a porta COM que será utilizada para comunicação com a launchpad.

O ambiente ENERGIA irá listar para você as portas COM disponíveis no seu PC, basta clicar sobre a porta que está conectada sua launchpad.



Pronto, agora temos nosso ambiente configurado e pronto para trabalhar.

Os exemplos presentes precisam de configurações extras para funcionar, mas isso vamos abordar em outros artigos, pois nosso foco aqui é fazer uma comunicação serial simples.

Primeiramente precisamos ALTERAR a função **void TimerSerial::RxIsr(void)** presente na Lib padrão do ENERGIA, pois o arquivo de cabeçalho para o msp430g2231.h mudou entre msp430-gcc versão 4.5.3 e 4.6.3 afetando código original. Para ser honesto, a menos que você executar o seu chip msp430g2231 em 16MHz, você não vai ser capaz de enviar e receber ao mesmo tempo. Simplesmente não há ciclos suficientes quando você adicionar todos a sobrecarga abstração código (FORUM ENERGIA).

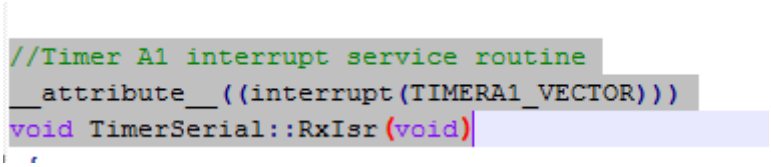
Antes de alteramos receberemos esse erro do compilador ao rodar nosso programa exemplo:

```
C:\Users\usuario\Desktop\energia-0101E0006\hardware\msp430\libraries\TimerSerial\TimerSerial.cpp: In static member function 'static void TimerSerial::RxIsr()':

C:\Users\usuario\Desktop\energia-0101E0006\hardware\msp430\libraries\TimerSerial\TimerSerial.cpp:205:16: error: 'TA0IV_TACCR1' was not declared in this scope
```

Primeiramente devemos alterar a função presente o arquivo **TimerSerial.cpp**

C:**CAMINHO_DO_SEU_AMBIENTE**\energia-0101E0006\hardware\msp430\libraries\TimerSerial\TimerSerial.cpp

Ao abrir o arquivo com seu editor padrão ou compilador, procure pelo trecho de código da função conforme ilustrado na figura abaixo:



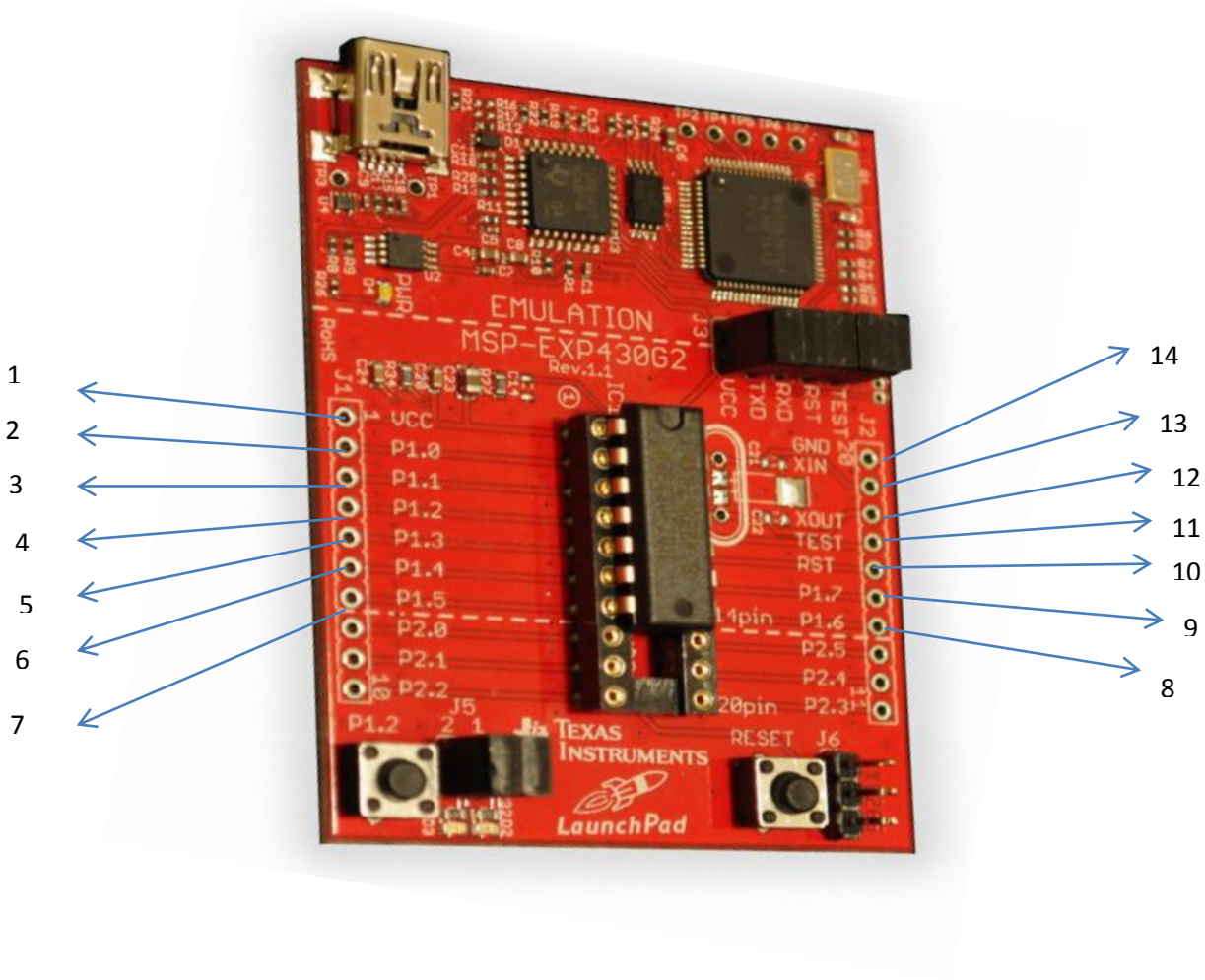
Você deverá substituir o código da função por este aqui:

```
//Timer A1 interrupt service routine
__attribute__((interrupt(TIMERA1_VECTOR)))
void TimerSerial::RxIsr(void)
 {
            static unsigned char rxBitCnt = 8;
            static unsigned char rxData = 0;

// reading TAIV auto-resets the interrupt flag
   volatile uint16_t reset_TAIV = TAIV; (void) reset_TAIV;

   TACCR1 += TICKS_PER_BIT;          // Setup next time to sample

   if (TACCTL1 & CAP)
   {   // Is this the start bit?
      TACCTL1 &= ~CAP;             // Switch capture to compare mode
      TACCR1 += TICKS_PER_BIT_DIV2;   // Sample from the middle of D0
   }
   else
    {
      rxData >>= 1;
      if (TACCTL1 & SCCI) {         // Get bit waiting in receive latch
```

```
        rxData |= 0x80;
      }
      rxBitCnt--;
      if (rxBitCnt == 0) {          // All bits RXed?
        store_rxchar(rxData);      // Store in ring_buffer
        rxBitCnt = 8;              // Re-load bit counter
        TACCTL1 |= CAP;            // Switch compare to capture mode
        TACCR1 += TICKS_PER_BIT;   // account for the stop bit
      }
    }

}
```

Após feita a alteração, salve o arquivo **TimerSerial.cpp**

Antes de qualquer coisa, só para entender como funciona o ambiente, os pinos são setados conforme a figura abaixo:



Agora com nosso ambiente devidamente configurado, vamos ao código de teste.

Abra o ambiente ENERGIA e salve o arquivo como: TesteSerial.ino

Você pode baixar o código aqui.

Vamos colar o seguinte código:

```
/*******************************
TESTE SERIAL LAUNCHPAD MSP430 - ENERGIA
 BY: RENNE TAKAO MEGURO PORTAL
14/07/2012
*******************************/
#include <TimerSerial.h>    //header

int cont;                   //declara contador
TimerSerial mySerial;       //cria objeto

void setup()
{
 mySerial.begin();  // inicializa Serial (9600)
 pinMode(2,OUTPUT);  // seta PINO 2 (LED1) para saída
  cont=0;  //inicializa contador
}
void loop() {

 digitalWrite(2, LOW);     // Apaga LED1
 delay(5000);              // Delay
 digitalWrite(2, HIGH);    // Liga LED1

 mySerial.print("\nEnviando Caracter\n");
 cont++;                   //incrementa contador
 mySerial.print(cont,DEC);  //envia caractere no formato Decimal
 mySerial.print("\n");
}
```

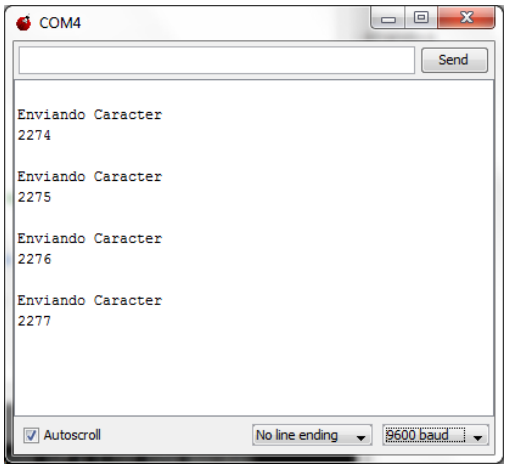
Agora é só compilar o código e verificar se não dá nenhum erro.

Compilar

Se não ocorrer nenhum erro é só enviar o programa para placa conforme ilustrado na figura abaixo:

Agora vamos verificar se tudo está conforme a gente escreveu no programa.

O LED1 deverá piscar a cada contagem e a mensagem deverá aparecer na tela mostrando o valor do contador. Para isso o ENERGIA possui o **Serial Monitor** (Tools -> Serial Monitor)



A janela abrirá e você deverá visualizar o envio dos dados pela launchpad.

Pessoal, espero ter ajudado a todos que buscaram por esse recurso e que possa ter despertado aos entusiastas as inúmeras possibilidades de coisas que este ambiente pode proporcionar de forma rápida e aplicada nos projetos de robótica, automação e outros segmentos da área eletrônica.

*Este pequeno artigo é livre para cópia desde que seja citada a fonte.

Um abraço e até os próximos artigos.

Renne Takao Meguro Portal

takao.portal@alsukkar.com.br

Dir. Tec. Al Sukkar Biotecnologia Industrial

+55 16 3024-8512