Aula prática N.º 11

Objetivos

- Compreender os mecanismos básicos que envolvem a comunicação série assíncrona.
- Implementar funções básicas de comunicação série através de uma UART, usando interrupções.

Introdução

O modo como a UART gera as interrupções é configurável através dos bits **`UTXISEL<1:0>`** e **`URXISEL<1:0>`** do registo **`UxSTA`**. Configurando esses bits com a combinação binária "**`00`**", são geradas interrupções nas seguintes situações:

- Receção: uma interrupção é gerada quando o FIFO de receção tem, pelo menos, um novo carater para ser lido; a rotina de serviço à interrupção pode efetuar a respetiva leitura do registo **`UxRXREG`**.
- Transmissão: uma interrupção é gerada quando o FIFO de transmissão tem, pelo menos, uma posição livre; a rotina de serviço à interrupção pode colocar no registo **`UxTXREG`** um novo carater para ser transmitido;

Para além destas duas configurações específicas, é ainda necessário configurar, como para todas as outras fontes de interrupção, os bits de *enable* das interrupções (residentes nos *Interrupt Enable Control Registers*) e os bits que definem a prioridade (residentes nos *Interrupt Priority Registers*).

Para a ativação da interrupção de receção é necessário configurar o bit **`UxRXIE`** (*receive interrupt enable*) e para a interrupção de transmissão o bit **`UxTXIE`** (*transmit interrupt enable*). Para a definição da prioridade devem ser configurados os 3 bits **`UxIP`** (a configuração de prioridade é comum a todas as fontes de interrupção de uma mesma UART).

Cada UART pode ainda gerar uma interrupção quando é detetada uma situação de erro na receção de um carater. Os erros detetados são de três tipos: erro de paridade, erro de *framing* e erro de *overrun*. Se se pretender fazer a deteção destes erros por interrupção, então é também necessário ativar essa fonte de interrupção, isto é, ativar o bit **`UxEIE`**.

Finalmente, é importante referir que a cada UART está atribuído um único vetor para as 3 possíveis fontes de interrupção. Essa situação obriga a que a identificação da fonte de interrupção tenha que ser feita por *software* na rotina de serviço à interrupção (rotina associada ao vetor atribuído à UART que está a ser usada). A identificação é feita através do teste dos *interrupt flag bits* de cada uma das três fontes possíveis de interrupção, isto é, **`UxRXIF`**, **`UxTXIF`** e **`UxEIF`**.

Trabalho a realizar

Parte I

1. Escreva o programa principal para teste da receção de um caracter por interrupção. O programa principal deve ter a seguinte estrutura:

```
int main(void)
{
   // Configure UART2: 115200, N, 8, 1
   // Configure UART2 interrupts, with RX interrupts enabled
      and TX interrupts disabled:
         enable U2RXIE, disable U2TXIE (register IEC1)
         set UART2 priority level (register IPC8)
         clear Interrupt Flag bit U2RXIF (register IFS1)
         define RX interrupt mode (URXISEL bits)
   // Enable global Interrupts

   while(1);
   return 0;
}
```

2. Escreva a rotina de serviço à interrupção, para receber um caracter e retransmitir esse mesmo caracter. A transmissão deve ser feita por *polling*.

   A estrutura da rotina de serviço à interrupção deve ser a seguinte:

```
void _int_(VECTOR_UART2) isr_uart2(void)
{
   if (UART2 Rx interrupt flag is set)
   {
      // Read character from FIFO (U2RXREG)
      // Send the character using putc()
      // Clear UART2 Rx interrupt flag
   }
}
```

3. Altere a rotina de serviço à interrupção e faça as inicializações necessárias, de modo a que o LED ligado ao porto RC14 da placa DETPIC32-IO acenda quando for recebido o carater '**`T`**', e que apague quando for recebido o caracter '**`t`**'.

Parte II

Pretende-se agora implementar a transmissão de um *string*, usando interrupções. Para isso deverá ser implementada uma função **`putstrInt`**() que armazena os caracteres da *string* a enviar num *buffer* temporário, que serão depois lidos sequencialmente pela rotina de serviço à interrupção e colocados no *buffer* de transmissão da **`UART`**. Ou seja, a função **`putstrInt()`** não atua, direta ou indiretamente, na **`UART`**, sendo essa interação feita exclusivamente pela rotina de serviço à interrupção.

O *buffer* temporário é constituído por uma zona de armazenamento para os caracteres a transmitir e por duas variáveis, uma para armazenar o número de caracteres que ainda falta transmitir e outra que indica a posição do *buffer* onde se encontra o próximo caracter a transmitir.

O *buffer* temporário pode ser declarado através de uma estrutura do tipo:

```
typedef struct
{
   char mem[100]; // Storage area
   int nchar;     // Number of characters to be transmitted
   int posrd;     // Position of the next character to be transmitted
} t_buf;
```

e uma instância global da estrutura pode ser declarada (fora da função **main()**) através de:

```
volatile t_buf txbuf;
```

O acesso a cada um dos membros da estrutura é feito através de **txbuf.mem**, **txbuf.nchar** e **txbuf.posrd**.

1. Escreva a função **putstrInt()** que deve armazenar todos os caracteres da *string* a transmitir na zona de armazenamento do *buffer* temporário, e atualizar o valor das variáveis auxiliares do mesmo (**nchar** e **posrd**).

```
void putstrInt(char *s)
{
   while(txbuf.nchar > 0); // Wait while the buffer is not empty
   // Copy all characters of the string "s" to the buffer
   while(...) {
      ...
   }
   // Initialize "posrd" variable with 0
   // Enable UART2 Tx interrupts
}
```

2. Escreva a rotina de serviço à interrupção, que leia um caracter do *buffer* temporário e o coloque no *buffer* de transmissão da UART. A estrutura da rotina de serviço à interrupção deve ser a seguinte:

```
void _int_(VECTOR_UART2) isr_uart2(void)
{
   if (UART2 Tx interrupt flag is set)
   {
      if(txbuf.nchar > 0) {   // At least one character to be transmitted
         // U2TXREG = ...     // Read 1 character from the buffer and
                                 send it
         // Update buffer "posrd" and "nchar" variables
      } else {
         // Disable UART2 Tx interrupts
      }
      // Clear UART2 Tx interrupt flag
   }
}
```

3. Escreva o programa principal para teste da transmissão de uma *string* por interrupção. O programa principal deve ter a seguinte estrutura:

```
int main(void)
{
   // Configure UART2: 115200, N, 8, 1
   // Configure UART2 interrupts, with RX and TX interrupts DISABLED
         disable U2RXIE, disable U2TXIE (register IEC1)
         set UART2 priority level (register IPC8)
         define TX interrupt mode (UTXISEL bits)
   // Enable global Interrupts
   // Initialize buffer variable "nchar" with 0
   while(1)
   {
      putstrInt("Test string which can be as long as you like as long as
                 it is no longer than 100 characters\n");
   }
   return 0;
}
```

Parte III

1. Retome o programa que escreveu na parte 1 da aula prática número 9 e inclua a configuração da UART2 e de receção por interrupção, que implementou anteriormente. Faça ainda as seguintes alterações:

   a) Declare duas variáveis globais, "**voltMin**" e "**voltMax**", que devem armazenar o valor mínimo e o valor máximo da tensão que o sistema mediu desde o seu arranque. Inicialize adequadamente essas variáveis e atualize-as na rotina de serviço à interrupção do módulo A/D, isto é, na rotina de serviço onde é calculado um novo valor de tensão.

```
void _int_(VECTOR_ADC) isr_adc(void)
{
   (...)
   // Update variables "voltMin" and "voltMax"
   // Reset AD1IF flag
}
```

   b) Implemente a rotina de serviço à interrupção da UART2 de modo a que quando for recebido da porta série o caracter "**M**" transmita o valor da variável "**voltMax**", e quando for recebido da porta série o caracter "**m**" transmita o valor da variável "**voltMin**", ambos em decimal, na forma "**VMxx=i.dV**" (ex: **VMax=2.3V**). A transmissão deve ser feita por *polling*.

   Sugestão: converta o valor das variáveis "**voltMax**" e "**voltMin**" para BCD e, de seguida, codifique em ASCII os dois dígitos de cada variável; após a transmissão da string "**VMxx=**" envie sequencialmente os dois códigos ASCII, começando pelo código correspondente ao dígito mais significativo.

```
void _int_(VECTOR_UART2) isr_uart2(void)
{
   char c = ...// Read character from FIFO
   if(c == 'M')
      // Send "voltMax" to the serial port UART2
   else if(c == 'm')
      // Send "voltMin" to the serial port UART2
   // Clear UART2 rx interrupt flag
}
```

Elementos de apoio

- Slides das aulas teóricas.
- PIC32 Family Reference Manual, Section 08 – Interrupts.
- PIC32 Family Reference Manual, Section 21 – UART.
- PIC32MX5XX/6XX/7XX, Family Data Sheet, Pág. 74 a 76.

PDF criado em 03/06/2022