Universidade Federal de Pernambuco
Departamento de Eletrônica e Sistemas

ES455 - Desenvolvimento de Sistemas Embarcados de Tempo Real — Semestre de 2020.3
Prof. Hermano Cabral — 04/set/2020

3ª Atividade Assíncrona

1 Descrição da atividade

Esta atividade consiste em converter um programa que utiliza *polling* para detectar mudança de nível lógico em um pino do microcontrolador em um outro programa que use interrupções para o mesmo fim. Você deve descompactar o arquivo zip referente ao microcontrolador que esteja usando em uma pasta no seu computador para começar os trabalhos. Em seguida, compile e instale o programa usando as instruções abaixo referentes ao seu microcontrolador. O funcionamento do programa é simples. O nível lógico do pino INT0 controla se o led da placa estará aceso ou não. O programa verifica, através do polling da flag referente ao INT0 (ou EXTI0 no Stm32), se há mudança no nível lógico no pino. Caso haja, ele muda o estado do led de forma correspondente. O pino INT0 é configurado como entrada com um resistor de pull-up (ou pull-down, no caso da Stm32), de modo que, se não estiver conectado a nada (dizemos, nesse caso, que o pino está flutuando), seu nível lógico é 1. Conectando-o ao terra (ou ao Vcc, no caso da Stm32), veremos que o led se apaga (ou se acende, na Stm32).

Esta funcionalidade de interrupções externas nos permite medir, com a ajuda de um temporizador, a frequência e o período de uma onda quadrada periódica ou os tempos de ocorrências de mudanças de nível lógico de um onda quadrada aperiódica. Estudaremos com mais detalhes as interrupções externas em uma próxima aula. Esta atividade é para compararmos o tratamento de eventos via polling e via interrupção.

Como falado em sala de aula, os nomes das interrupções a serem usados no código devem seguir rigorosamente um padrão. No caso do ATMega328p, os nomes seguem a nomenclatura da Seção 12.4 à página 65 da Folha de Dados adicionando-se o sufixo `_vect` ao nome da interrupção, além de trocar o espaço por um caractere '_'. Para o microcontrolador STM32, os nomes são mais flexíveis, sendo definidos no arquivo de start-up (veja a Seção 3 abaixo). Como pode ser visto no arquivo startup_stm32f103xb.s, os nomes seguem a nomenclatura da Tabela 63 na Seção 10.1.2 do Manual de Referência. O nome da interrupção que desejamos é a EXTI0_IQRHandler.

Para gerarmos o programa binário a partir do código em C, devemos usar um compilador e um lincador. Normalmente, usamos o compilador e as bibliotecas do computador em que estamos executando os comandos. Por exemplo, se estamos em um ambiente Linux, estaremos usando o compilador e as bibliotecas da plataforma que estamos usando (por exemplo, a das CPUs Intel) para gerar código para esta mesma plataforma. O arquivo binário é gerado para ser executado na própria plataforma.

No caso do desenvolvimento para microcontrolador utilizando o Linux, a situação é um pouco diferente. Iremos usar ferramentas como o compilador que rodam na plataforma Linux para gerar o código binário para outra plataforma, como a AVR ou Stm32. O código binário é gerado para ser executado por outra CPU que não a do Linux. Este tipo de desenvolvimento é chamado de desenvolvimento cruzado, e é detalhado abaixo para as duas plataformas que usaremos. Basicamente, temos que escutar 4 passos:

1. Compilação de todos os códigos-fonte em C (ou outra linguagem apropriada) para arquivos binários contendo instruções para a CPU-destino. Esses arquivos binários são chamados de relocatáveis pois os endereços das variáveis globais e estáticas ainda não foram alocados definitivamente. Apenas na etapa seguinte é que estes endereços serão determinados.

2. Ligação (ou lincagem, do verbo em inglês *to link*) de todos os arquivos binários relocatáveis para gerar um arquivo binário único com todo o código e todos os dados arrumados de acordo com uma estrutura. Este binário tem outras informações, como as de debug, além do código e dados.

3. Geração de um arquivo binário contendo apenas as instruções e os dados por elas usados em um formato apropriado para uso no próximo passo.

4. Gravação do arquivo binário do passo anterior na memória flash do microcontrolador.

As seções abaixo detalham os procedimentos para as plataformas AVR e Stm32.

2 Instruções para geração do programa para AVR

Nesse caso, usamos o compilador cruzado avr-gcc para, no ambiente Linux, gerar um binário para o microcontrolador ATMega328p executando, em um terminal, a seguinte série de comandos:

```
avr-gcc -c -mmcu=atmega328p -Os int0.c
avr-gcc -mmcu=atmega328p -o int0.elf int0.o
avr-objcopy -j .text -j .data -O ihex int0.elf int0.hex
```

O primeiro comando é a compilação do código-fonte em C, gerando um arquivo binário relocatável, int0.o. O segundo comando realiza, através do compilador avr-gcc, a ligação de todos os arquivos binários, o que, neste caso, corresponde apenas a int0.o. O resultado é o arquivo int0.elf, que contém todo o programa binário, além de informações extras como as de debug.

O terceiro comando cria um arquivo binário, o int0.out, contendo apenas com as informações de dados e de código, excluindo as informações extras introduzidas pelo ligador (em inglês, *linker*). Isto é necessário para reduzir o tamanho do programa ao mínimo para que caiba dentro da memória flash do microcontrolador.

Para a transferência do arquivo int0.hex resultante para o microcontrolador, usamos o programa avrdude executando no terminal o comando

```
avrdude -P /dev/ttyUSB0 -c arduino -p atmega328p -b57600 -v -v -v -v -D
    -U flash:w:int0.hex:i
```

No comando acima, existem dois parâmetros que talvez precisem ser ajustados para que o comando funcione. O primeiro é o parâmetro /dev/ttyUSB0 que define a porta USB a ser usada. Para algumas placas, o parâmetro correto é o /dev/ttyACM0. A mensagem

```
avrdude: ser_open(): can't open device "/dev/ttyUSB0": No such file or director
```

é um forte indicativo que o parâmetro usado não é o correto. Em alguns raros casos, o 0 deve ser substituído por 1. Para saber se isso é necessário, execute no terminal o comando

```
ls -l /dev/ttyUSB* /dev/ttyACM*
```

e use os valores que forem apresentados, um de cada vez, até que tenha sucesso.

O segundo parâmetro que talvez precise de ajustes é o valor 57600, referente à velocidade de transferência da porta serial. Em alguns casos, é preciso mudar para 115200. A mensagem

```
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x00
```

é um indício de que a velocidade usada não é a correta.

3 Instruções para geração do programa para Stm32f103

Para a Stm32, usamos o compilador cruzado arm-none-eabi-gcc para, no ambiente Linux, gerar o arquivo binário contendo as instruções corretas para o microcontrolador. Devido ao fato deste compilador ser voltado para uma família mais genérica do que a do AVR acima, nós temos que especificar uma parte do código de inicialização, o que aqui é feito através do arquivo startup_stm32f103xb.s, e algumas informações para o lincador sobre o processador, o que é feito através do arquivo stm32f103xb.ld. Além disso, o arquivo stm32f103xb.h e alguns outros arquivos .h definem muitas constantes usadas no programa int0.c e cujos nomes seguem a nomeclatura do Manual de Referência do Stm32f103.

O primeiro arquivo pode ser compilado com o comando

```
arm-none-eabi-gcc -c -mcpu=cortex-m3 -mthumb startup_stm32f103xb.s
```

Isto gerará o arquivo startup_stm32f103xb.o, a ser usado depois na etapa de lincagem.

Em seguida, prosseguimos com a compilação do arquivo principal, int0.c através da execução do comando

```
arm-none-eabi-gcc -c -mcpu=cortex-m3 -mthumb -O3 int0.c
```

A lincagem é realizada pelo compilador usando o arquivo stm32f103xb.ld. O comando a ser executado é

```
arm-none-eabi-gcc -T stm32f103xb.ld -nostartfiles -o int0.elf int0.o
    startup_stm32f103xb.o
```

Por último, a geração do arquivo binário a ser gravado na memória flash é feita pelo comando

```
arm-none-eabi-objcopy -S -O ihex int0.elf int0.hex
```

Se você estiver usando um conversor USB-TTL, a gravação deve ser feita através do comando

```
stm32flash -w int0.hex /dev/ttyUSB0
```

Caso esteja usando o stlink, o comando para gravação é

```
st-flash --format ihex write int0.hex
```

É importante lembrar que, caso esteja usando o conversor USB-TTL, é necessário colocar o jump BOOT0 em 1 antes de gravar. Para rodar o programa gravado, este jump precisa estar em 0.