Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Sistemas Elétricos de Automação e Energia
ENG10032 Microcontroladores

Roteiro de Laboratório 05
Interrupções

Prof. Walter Fetter Lages

17 de abril de 2019

1 Objetivo

O objetivo deste laboratório é entender o funcionamento das interrupções na Galileo Gen2 e o seu processamento no espaço do usuário.

2 Fundamentação Teórica

A maioria dos pinos de I/O disponíveis no *shield* da Galileo Gen2 pode gerar interrupções[1]. A tabela com o mapeamento dos pino de I/O indica quais pinos podem gerar interrupção.

As interrupções podem ser geradas nas bordas de subida, de descida ou ambas bordas do sinal. Note que os pinos `IO2` e `IO3` possuem dois pinos de GPIO conectados a eles. Isso é feito para possibilitar a geração de interrupções em ambas bordas do sinal.

Para que um pino de I/O seja usado como fonte de interrupção ele deve ser exportado e configurado como entrada através do diretório `/sys/class/gpio`.

2.1 Biblioteca para Acessar o `sysfs`

As portas GPIO são acessadas no Linux através da leitura e escrita de pseudo-arquivos em `/sys/class/gpio`. Para simplificar o processo de acessar estes pseudo-arquivos em programas em C e C++ será construída ao longo do curso uma biblioteca denominada `libgalileo2`. Inicialmente, esta biblioteca terá apenas a função `pputs()`, mostrada na Listagem 1, que será usada para escrever uma *string* em um arquivo especificado por seu *path*.

[1] Na primeira geração da Galileo, apenas os pinos `IO2` e `IO3` podiam gerar interrupções

Listagem 1: Arquivo `pputs.c`.

```
1  #include <fcntl.h>
2  #include <string.h>
3  #include <unistd.h>
4
5  #include <galileo2io.h>
6
7  int pputs(const char path[],const char s[])
8  {
9          int fd;;
10         int n;
11
12         if((fd=open(path,O_WRONLY)) == -1) return -1;
13         n=write(fd,s,strlen(s));
14         close(fd);
15         return n;
16 }
```

Note que o uso de funções como `system()`, `popen()` ou similares para acesso aos pseudo-arquivo em `/sys` no lugar das funções usuais de I/O de C/C++ não é recomendado, pois cada execução daquelas funções implica a execução de um `fork()` e a invocação de uma nova instância do *shell*, o que é extremamente ineficiente.

2.2 Habilitação da Interrupção

As interrupções são configuradas através do arquivo `/sys/class/gpio/gpioXX/edge`, onde `XX` é o número do gpio associado ao pino no *shield* da Galileo. As seguintes *strings* podem ser escritas no arquivo para configurar a interrupção:

`"rising"`: interrupção ativa na borda de subida do sinal

`"falling"`: interrupção ativa na borda de descida do sinal

`"both"`: interrupção ativa em ambas bordas do sinal

`"none"`: interrupção desabilitada

Uma vez habilitada, a interrupção pode ser recebida por um programa no espaço do usuário através de um *polling* no arquivo `/sys/class/gpio/gpioXX/value` com o uso da função `poll()` para esperar por um evento `POLLPRI`, como mostra a listagem 2.

Listagem 2: Programa para receber interrupções através da função `poll()`.

```
1  #include <fcntl.h>
2  #include <poll.h>
3  #include <stdio.h>
4  #include <unistd.h>
5
6  #include <galileo2io.h>
7
8  int main(int argc,char * argv[])
9  {
10         unsigned char c;
11         struct pollfd pfd;
12
13         if((pfd.fd=open("/sys/class/gpio/gpio6/value",O_RDONLY)) < 0)
14         {
15                 perror("Opening gpio6:");
16                 return -1;
17         }
18
19         /* Clear old values */
20         read(pfd.fd,&c,1);
21
22         pfd.events=POLLPRI;
23
24         puts("Waiting for interrupt...");
25
26         pputs("/sys/class/gpio/gpio6/edge","both");
27
28         poll(&pfd,1,-1);
29
30         lseek(pfd.fd,0,SEEK_SET);
31         read(pfd.fd,&c,1);
32
33         pputs("/sys/class/gpio/gpio6/edge","none");
34
35         close(pfd.fd);
36         return 0;
37 }
```

3 Experimentos

3.1 Criação da Biblioteca para Acessar o `sysfs`

1. No *host*, crie um diretório denominado `lib` e coloque nele o arquivo `pputs.c` mostrado na Listagem 1 e o `Makefile` mostrado na Listagem 3. Coloque no mesmo diretório os arquivos `pgets.c` e `i2_write_reg.c` (disponíveis no Moodle), que serão utilizados em laboratórios futuros, mas fazem parte da mesma biblioteca e são necessários para o `Makefile` funcionar corretamente.

Listagem 3: Arquivo `Makefile` para a biblioteca `libgalileo2`.

```
1  TARGET=libgalileo2.a
2  SRCS=pputs.c pgets.c i2c_write_reg.c i2c_read_reg.c
3
4  FLAGS=-O2 -Wall -MMD
5  INCLUDE=-I. -I../include
6  LIBDIR=
7  LIBS=
8
9  CC=$(CROSS_COMPILE)gcc
10 CCAR=$(CROSS_COMPILE)ar
11 CFLAGS=$(FLAGS) $(INCLUDE)
12 LDFLAGS=$(LIBDIR) $(LIBS)
13 CCARFLAGS=-crvs
14
15 all: $(TARGET)
16
17 $(TARGET): $(SRCS:.c=.o)
18         $(CCAR) $(CCARFLAGS) $@ $^
19
20 %.o: %.c
21         $(CC) $(CFLAGS) -c -o $@ $<
22
23 -include $(SRCS:.c=.d)
24
25 clean:
26         rm -f *~ *.bak *.o *.d
27
28 distclean: clean
29         rm -f $(TARGET)
```

2. Crie um diretório denominado `include` e coloque nele o arquivo `galile2io.h`, mostrado na Listagem 4 e os arquivos `i2cutil.h` e `jhdlcd.h` (disponíveis no Moodle), que serão utilizados em laboratórios futuros, mas fazem parte da mesma biblioteca são necessários para o `Makefile` funcionar corretamente.

Listagem 4: Arquivo de cabeçalho `galileo2io.h`.

```
1  #ifndef GALILEO2IO_H
2  #define GALILEO2IO_H
3
4  #ifdef __cplusplus
5  extern "C"
6  {
7  #endif
8
9  extern char * pgets(char *s,int size,const char path[]);
10 extern int pputs(const char path[],const char s[]);
11
12 #ifdef __cplusplus
13 };
14 #endif
15
16 #endif
```

3. Execute o `Makefile` para gerar a biblioteca.

3.2 Uso da Interrupção

4. Faça um `Makefile` para compilar o programa mostrado na listagem 2.

5. Identifique qual o pino de I/O gera as interrupções para o programa mostrado na listagem 2.

6. Faça um *script* de inicialização para configurar o pino identificado no item anterior como entrada. Configure permissões de leitura e escrita para o grupo `gpio` nos arquivos `value` e `edge` correspondentes ao pino em questão.

7. Coloque o *script* de inicialização no diretório `/etc/init.d` da Galileo e configure-o para ser executado na inicialização.

8. Reinicialize a Galileo.

9. Conecte o *push-button* do *Grove Starter Kit*, mostrado na Figura 1 e teste a geração de interrupção quando o interruptor é pressionado, usando o programa da listagem 2.

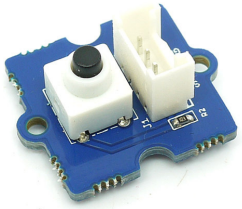

Figura 1: *Push-button*.

10. Modifique o programa para contar as interrupções geradas e verifique o seu funcionamento.

11. Modifique novamente o programa para utilizar apenas interrupções na borda de subida do sinal e verifique o seu funcionamento.

12. Modifique mais uma vez o programa para utilizar apenas interrupções na borda de descida do sinal e verifique o seu funcionamento.

13. Repita os experimentos usando o sensor de toque do *Grove Starter Kit*, mostrado na Figura 2.

Figura 2: Sensor de toque.