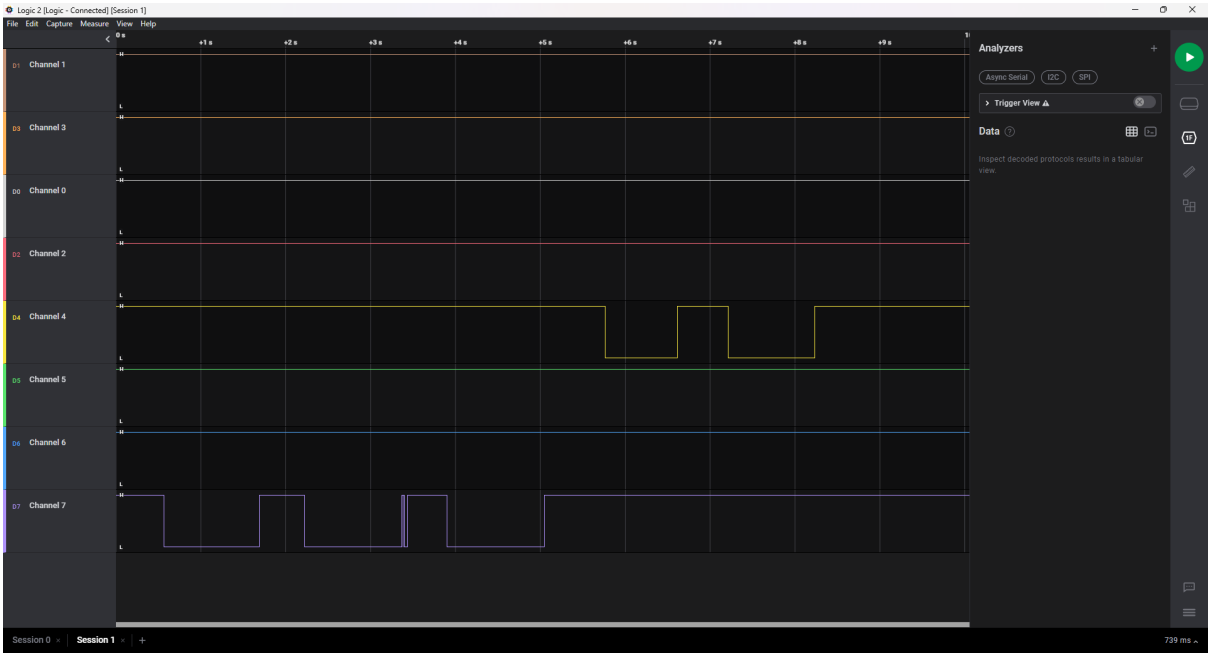
FEEC - UNICAMP;
EA871;
EXPERIMENTO DA TAREFA 2;
VINÍCIUS ESPERANÇA MANTOVANI, RA: 247395;

Ítem 1)

a) Inicialmente, durante os três clicks em NMI, observamos a cor verde, acompanhada pela queda do sinal de canal 7 no aplicativo do analisador, conforme esperado, pois as botoeiras são ativo-baixo. Em seguida, nos dois clicks em IRQA5, percebemos a luz vermelha, acompanhada pela queda para 0 do sinal de canal 4 no aplicativo do analisador lógico, conforme as seguinte figura:



b)

Identificação	Valores, config_basica	Valores, config_especi	Valores, inicializacao	Valores, laço	função do registrador
GPIOE_PDDR	0x00000000	0x00000000	0x00600000	0x00600000	determina o sentido de saída do sinal nos pinos PTE (usado nesse caso para determinar o do pino PTE21)
GPIOA_PDDR	0x00000000	0x00000000	0x00000000	0x00000000	determina o sentido de entrada dos sinais dos pinos PTA (usado nesse caso para determinar o do pino PTA4)
PORTA_PCR5	0x00000000	0x00000105	0x000b0105	0x000b0105	determina o uso de configuração por GPIO para o pin correspondente , por meio da alteração dos bits de MUX.

NVIC_ISER	0x00000000	0x00000000	0x40000000	0x40000000	Controle de interrupções, habilitação de interrupções (neste caso, habilita a interrupção de valor IRQ30)
NVIC_ICPR	0x00000000	0x00000000	0x00000000	0x00000000	Controle de interrupções, limpeza de pendências (neste caso, limpa pendências de IRQ30)
NVIC_IPR7	0x00000000	0x00000000	0x00c00000	0x00c00000	Registrador de prioridades de interrupções (neste caso, seta a prioridade de IRQ30 em 3)
SIM_SCGC5	0x00000180	0x00002380	0x00002380	0x00002380	Controla os sinais de relógio do sistema (neste caso ativa os sinais de relógio de PORTE e PORTA)
PORTE_PCR20	0x00000000	0x00000005	0x00000005	0x00000005	determina o uso de configuração por GPIO para o pin correspondente , por meio da alteração dos bits de MUX.
PORTA_PCR4	0x00000000	0x00000107	0x00000107	0x00000107	determina o uso de configuração por GPIO para o pin correspondente , por meio da alteração dos bits de MUX.
GPIOE_PSOR	non-readable	non-readable	non-readable	non-readable	Inicializa as botoeiras em estado alto (inativas)

c) Inicialmente, para o caso de NMI, o registrador GPIOA_PDIR tem seu valor verificado, para que, caso o valor do quinto bit da direita para a esquerda seja 1, então temos que a botoeira está desativada e, caso seja 0, a botoeira está ativada. Assim, é ativado o sinal de PTE21 e, consequentemente, a luz verde é acesa. Já para o caso de IRQA5, ocorre uma interrupção, tratada de maneira que, caso o sétimo bit da direita para a esquerda de PORTA_PCR5 seja 1, então ocorreu borda de subida e a chave esta aberta (desativada a botoeira), mas, caso contrário, a botoeira está ativada e a chave está fechada, portanto, GPIOE_PCOR tem seu valor alterado no equivalente a PTE22, acendendo a luz vermelha. **Ou seja, os eventos usados como base da ativação das botoeiras são, a alteração no quinto bit da direita para a esquerda de GPIOA_PDIR para o caso de NMI e, para o caso de IRQA5, a alteração no sétimo bit da direita para a esquerda em PORTA_PCR5.**
d) Deve-se colocar um breakpoint na rotina config_especifica para verificar qual das botoeiras tem a rotina de interrupção ativada, pois é nesta rotina que se ativam as interrupções desejadas.

Foram necessárias para a configuração de interrupção da botoeira IRQA5, a configurações de NVIC_ISER, que diz respeito à habilitação de interrupção, a configuração de NVIC_ICPR, responsável pela limpeza de pendências de interrupções devida e, finalmente, a configuração de NVIC_IPR7 responsável pela determinação da prioridade da interrupção dada. Ou seja, foi necessário configurar o NVIC, controlador vetorial de interrupções.

A Botoeira NMI funciona por meio do mecanismo de polling, implementado na main, por meio de um loop que trabalha com as alterações no valor de GPIOA_PDIR, determinando o acendimento da luz a partir do valor do bit deste registrador que representa o input equivalente à botoeira em questão.

e) Como existe debounce por hardware no caso estudado, espera-se que a rotina de tratamento de interrupção em questão seja chamada apenas duas vezes desde o pressionamento até a soltura da botoeira, pois o debounce impede que as oscilações do sinal causem mudanças abruptas no valor do sinal, impedindo, portanto que vá de 0 para 1 durante o pressionamento, o que faz com que a rotina seja chamada apenas no começo desse pressionamento e na soltura da botoeira.

   Por meio da análise feita olhando o LED e daquela pelos sinais captados pelo analisador lógico, percebemos que o debouncer do sistema funciona bem.

   Por fim, fazendo um teste de unidade, seguindo o procedimento dado no roteiro, notamos que a rotina de interrupção é chamada, realmente, duas vezes apenas, uma no começo do pressionamento da botoeira e outra na soltura.

f) No código, notamos que os registradores GPIOE_PSOR, GPIOE_PDDR, GPIOx_PCOR, e GPIOA_PDDR exercem efeito sobre os pinos correspondentes no caso de alteração do bit respectivo para ‘0’. Já os demais citados na questão (GPIOx_PTOR, GPIOx_PDIR e GPIOx_PDOR) podem sofrer escrita de bit ‘0’ sem alterar os pinos correspondentes.
g) A escrita de ‘0’ tem efeito sobre o estado da IRQ correspondente no caso dos registradores NVIC_ISER, NVIC_ICPR, NVIC_IPR7. Os demais não foram configurados em config_especifica e não afetam o estado da IRQ correspondente no caso de escrita de ‘0’.
h) Para a configuração do módulo NVIC, foram usados os registradores: NVIC_ISER, NVIC_ICPR e NVIC_IPR7.

   NVIC participa da execução, uma vez que é ele que controla as interrupções, ou seja, está constantemente determinando quais interrupções devem ser tratadas e em que ordem. De forma que, a qualquer momento, se pressionada a botoeira, neste caso, por exemplo, a rotina de interrupção é chamada e a interrupção é devidamente tratada.
i) No caso de excluirmos as duas linhas determinadas no roteiro, aconteceria que a rotina de tratamento de interrupções não seria chamada em bordas de descida (pressionamento da botoeira), logo, a cor do LED não seria alterada quando a botoeira fosse apertada. Já no caso de excluirmos a habilitação da IRQ 30 em NVIC, o que ocorreria seria que a interrupção não poderia ser chamada, logo, o mesmo efeito seria observado, uma vez que a rotina de interrupção não seria chamada em hipótese alguma.
j) O nome da rotina de interrupção não é arbitrário, pois a chamada da rotina de interrupção depende dele, ou seja, ele é previamente determinado e deve assumir o

nome necessário para que a interrupção seja tratada por essa função caso o evento ocorra.

A linha if ((*(uint32_t volatile *) 0x40049014u) & 0x1000000) serve para verificar o valor do bit (interruption flag) do registrador de estado necessário, para que, caso tenha o valor 1, de fato, a interrupção seja tratada conforme o necessário. Já a linha (*(uint32_t volatile *) 0x40049014u) |= (1<<24) serve para limpar o valor da flag, ou seja, redefiní-la para que a interrupção não seja tratada novamente de maneira errônea.

Ítem 2)

Inicialmente, ao trocar os pte para os ptb. Assim, ao testar a mudança, percebi diretamente o funcionamento correto. Em seguida, ao ativar e configurar as interrupções para o acionamento dos três LEDs, não obtive sucesso por alguns testes devido a erros nos endereços de pta4 e pta12 na rotina de tratamento de interrupções e, devido a erro nos valores colocados nos registrador GPIOA_PDIR, no entanto, ao arrumar esses erros, obtive sucesso nesta etapa, também. Então, ao trocar o funcionamento para o modo toggle, não ocorreu nenhum erro, com exceção do led b, no entanto, este erro ocorreu meramente por conta de ter escrito errado o endereço do registrador. Ademais, no geral, os testes de unidade e de funcionamento foram bem sucedidos e, o programa, por fim, é executado e funciona normalmente de acordo com o que foi instruído no roteiro.