INSTITUTO FEDERAL
Santa Catarina

Avaliação 03

Implementação de classes em *C++* para uso de periféricos do microcontrolador AVR Atmel ATmega2560

Curso: Engenharia de Telecomunicações
Disciplina: STE29008 – Sistemas Embarcados
Professor: Roberto de Matos

Alunos
Maria Fernanda Tutui
Marina Souza
Paulo Sell

Junho de 2019

1 Introdução

Este relatório apresenta abordagens para o uso de periféricos (interfaces seriais, *timer* e interrupções externas) do AVR **Atmel ATmega2560** na placa *Arduino Mega*. Segundo (LIMA; VILLAçA, 2012), as principais características de um **AVR** são:

- Executam a maioria das instruções em um ou dois ciclos de *clock*;
- Alta integração e grande número de periféricos com efetiva compatibilidade com toda a família AVR
- Possuem vários modos para redução do consumo de energia
- Preço acessível

As implementações foram desenvolvidas através da biblioteca **AVR-LibC** e a linguagem de programação **C++**. A IDE (*Integrated Development Environment*) **Eclipse neon**, modificada com um *plugin* para a injeção dos códigos no AVR, também foi utilizada para auxílio nos desenvolvimentos. As sessões seguintes irão apresentar as implementações feitas, bem como uma breve análise das implementações.

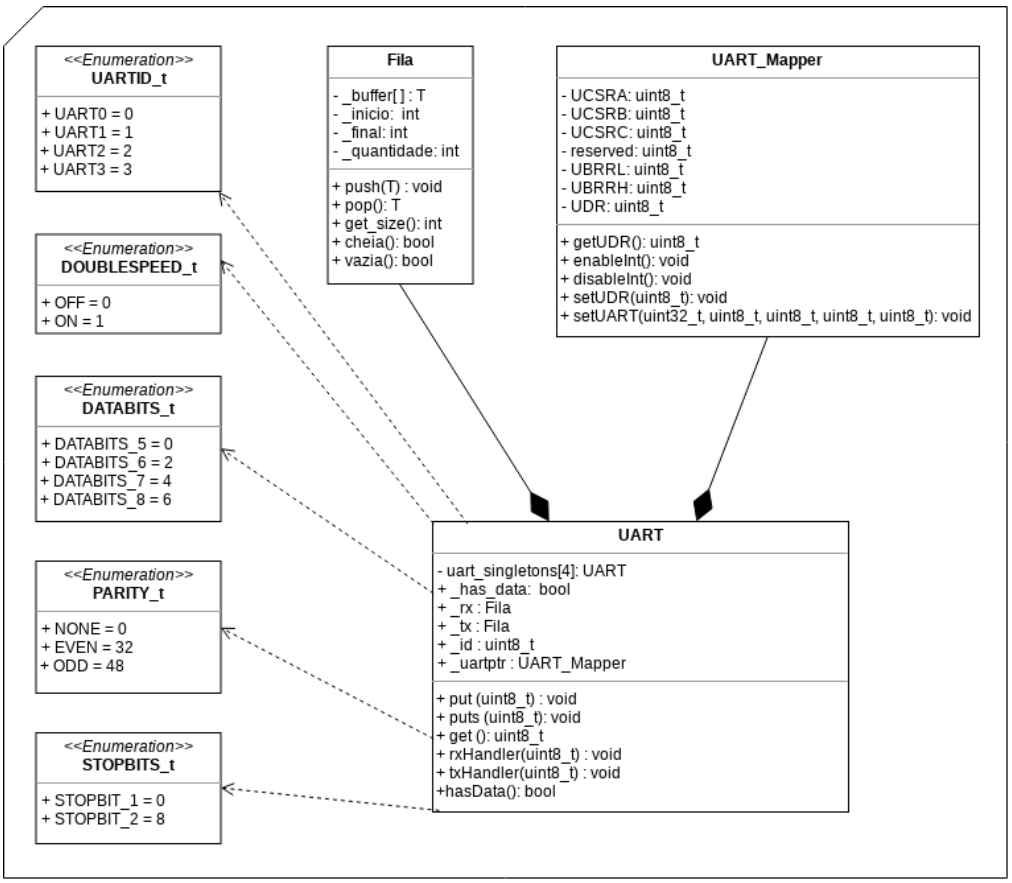
2 Implementações

2.1 UART não-bloqueante

Afim de permitir a utilização de todas as quatro interfaces seriais do AVR Atmel ATmega2560, três classes foram criadas: classe UART, classe UART_Mapper e classe Fila.

A classe UART_Mapper é responsável por instanciar os registradores de controle de cada interface serial, a partir de um mapeamento das posições de memória de cada registrador no AVR. Através da classe UART, o usuário é capaz de declarar a interface que deseja utilizar, informando sua identificação. Esta classe possui uma fila (classe Fila) de dados recebidos e uma fila de dados para transmitir. Desta forma a classe é não-bloqueante, utilizando as *flags* de interrupção de chegada de dados e fim de transmissão da interface serial. A figura 1 ilustra o diagrama de classes desta implementação.

Figura 1: Diagrama



Esta implementação é bastante poderosa. Ela permite que o usuário possa utilizar todas interfaces seriais do microcontrolador ao mesmo tempo e de forma não bloqueante, possibilitando o fluxo de um programa.

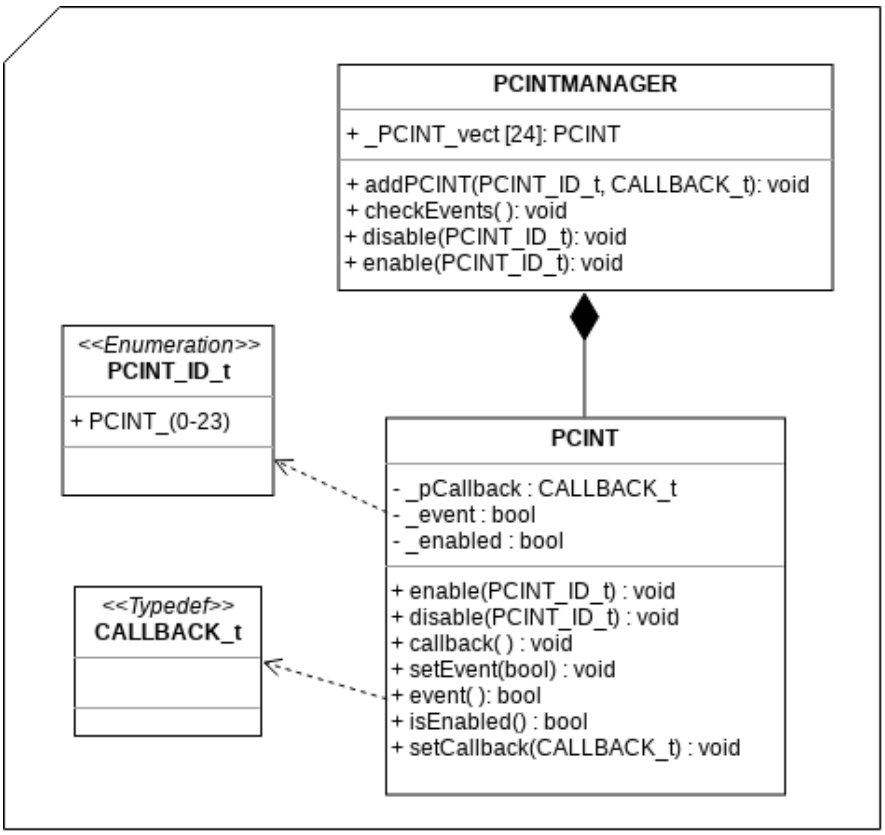
2.2 Interrupção por mudança de pino (*PCINT*)

O AVR *Atmel ATmega2560* possui 24 interrupções do tipo *PCINT*. Estas interrupções não são tão prioritárias quanto as interrupções externas e estão divididas em três vetores de interrupção. Ou seja, quando o microcontrolador percebe que houve uma interrupção em um dos três vetores, cabe ao desenvolvedor fazer o controle e identificar qual pino gerou a interrupção.

Nesta implementação criou-se uma classe chamada ***PCINT*** que tem acesso direto aos registradores de cada vetor de interrupção e uma classe chamada ***PCINTMANAGER***, que permite ao usuário indicar qual interrupção será utilizada e verifica qual pino gerou cada interrupção.

O diagrama de classes desta implementação pode ser visto na figura 2

Figura 2: Diagrama



Quando ocorre uma interrupção nesta implementação, ela não é diretamente atendida. No tratamento da interrupção a classe de gerência apenas estabelece que um evento aconteceu em um determinado pino do microcontrolador. Apenas quando o método de verificação de eventos é invocado pelo usuário é que a função de tratamento da interrupção acionado é chamada e realiza seu trabalho.

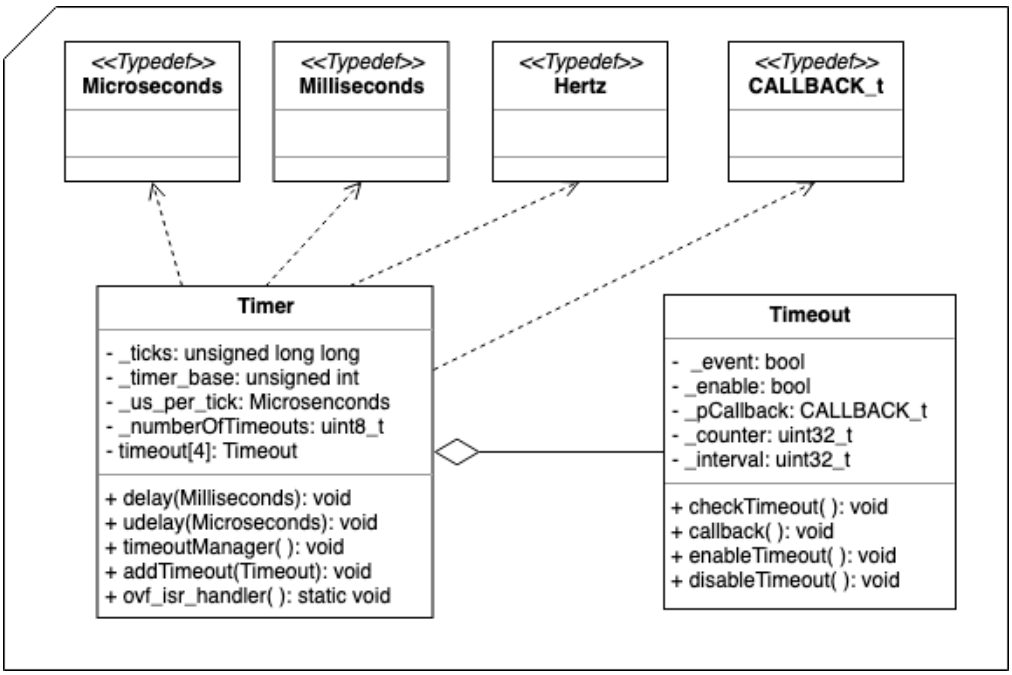
2.3 *Timer* e *timeouts*

Para o periférico *timer*, implementou-se uma classe chamada **Timeout** que permite ao usuário informar o intervalo de tempo e a função de tratamento que o programa deve invocar quando o intervalo de tempo escolhido for estourado.

Esta implementação possibilita o usuário criar os *timeouts*, registra-los no *timer* e eventualmente desabilitar o *timeouts* criados. Desta forma, o usuário tem maior controle e sobre cada *timeout* feito.

O diagrama de classes desta implementação pode ser visto na figura abaixo.

Figura 3: Diagrama



Paralelamente, uma outra implementação foi feita. Desta vez, o usuário não tem controle dos objetos *Timeout*, podendo apenas registrar os *timeouts* na classe *Timer*. Sendo assim, um *timeout* não pode ser desabilitado conforme o usuário desejar. A tabela 1 faz uma rápida comparação entre as duas implementações no uso da memória de programa e memória de dados.

Tabela 1: Comparação das duas versões implementadas

	Implementação com controle dos *timeouts*	**Implementação sem controle dos *timeouts***
Memória de programa	143 *bytes*	3684 *bytes*
Memória de dados	151 *bytes*	3458 *bytes*

3 Conclusão

Este relatório apresentou a implementação de classes para uso de alguns periféricos do AVR Atmel ATmega2560, programando-as com a linguagem de programação *C++* através da biblioteca AVR-LibC. As classes permitem ao usuário trabalhar com todas as interfaces seriais do microcontrolador, todas as *PCINT* e instanciar até quatro *timeouts*, dando liberdade para o programador. As atividades foram desenvolvidas com a IDE Eclipse neon, para facilitar a injeção dos códigos no microcontrolador e com o auxílio do *datasheet* do microcontrolador para verificar a funcionalidade de cada registrador utilizado.

Referências

LIMA, C.; VILLAçA, M. *AVR e Arduino: técnicas de projeto*. 2. ed. [S.l.: s.n.], 2012.