Desenvolv. de Sistemas Embarcados em Tempo Real

Prof. Hermano Cabral

Departmento de Eletrônica e Sistemas — UFPE

23 de julho de 2024

Plano de Aula

Tema central

- Chibios — Hardware Abstraction Layer

Plano de Aula

Tema central

- Chibios — Hardware Abstraction Layer

Objetivos

- Conhecer as funcionalidades de geração de sinais PWM do ChibiOS

HAL - módulo de sinais PWM

Características gerais

- O módulo de PWM implementa um driver PWM que representa um temporizador composto de:
  - Um seletor de clock (clock prescaler)
  - Um contador crescente
  - Um vetor de canais PWM, cada qual com uma saída associada

HAL - módulo de sinais PWM

Características gerais

- Quando dos eventos de comparação (reset do contador ou canal PWM), um callback opcional pode ser chamado.

HAL - módulo de sinais PWM

Características gerais

- Quando dos eventos de comparação (reset do contador ou canal PWM), um callback opcional pode ser chamado.
- O número de canais por driver PWM depende do hardware e é especificado pela constante PWM_CHANNELS
  - No caso do ATMega328p, o nº de canais é 2, e existem apenas 2 drivers, PWMD1 e PWMD2
  - No caso do Stm32, o nº de canais é 4, e existem 4 drivers

HAL - módulo de sinais PWM

Arquivos de implementação

- Este módulo é implementado através dos seguintes arquivos dentro do diretório os/hal:
  - include/hal_pwm.h e src/hal_pwm.c, arquivos para a interface de alto nível do HAL
  - ports/AVR/hal_pwm_lld.h e ports/AVR/hal_pwm_lld.c, arquivos para a implementação de hardware de baixo nível

Arquivos de configuração

- Para se usar o driver PWM é preciso configurar os arquivos mcuconf.h e halconf.h.

Arquivos de configuração

- Para se usar o driver PWM é preciso configurar os arquivos mcuconf.h e halconf.h.
- No arquivo mcuconf.h, devemos mudar as linhas referentes ao temporizador que queremos usar:
  - #define AVR_PWM_USE_TIM1 TRUE, etc

HAL - módulo de sinais PWM

Arquivos de configuração

- Para se usar o driver PWM é preciso configurar os arquivos mcuconf.h e halconf.h.
- No arquivo mcuconf.h, devemos mudar as linhas referentes ao temporizador que queremos usar:
  - #define AVR_PWM_USE_TIM1 TRUE, etc
- No arquivo halconf.h devemos selecionar a funcionalidade de PWM:
  - #define HAL_USE_PWM TRUE

HAL - módulo de sinais PWM

Configuração do código

- Para configurarmos a geração dos sinais temos que configurar duas coisas:
  - o driver, que controla o temporizador
  - o canal, que controla a geração do sinal

HAL - módulo de sinais PWM

```
typedef uint16_t pwmcnt_t;
typedef void (*pwmcallback_t)(PWMDriver *pwmp);

typedef struct {
    uint32_t                    frequency;
    pwmcnt_t                    period;
    pwmcallback_t               callback;
    PWMChannelConfig            channels[PWM_CHANNELS];
} PWMConfig;
```

Configuração do código

- A configuração do driver PWM é feita através da estrutura PWMConfig.

Configuração do código

- A descrição dos membros da estrutura PWMConfig é:
  - `frequency` – frequência do temporizador
  - `period` – duração, em pulsos de clock, do período do sinal
  - `PWM callback` – função a ser chamada quando da volta do contador a 0
  - `channels` – vetor de tamanho PWM_CHANNELS de configurações de canal

Configuração do código

- A descrição dos membros da estrutura PWMConfig é:
  - `frequency` – frequência do temporizador
  - `period` – duração, em pulsos de clock, do período do sinal
  - `PWM callback` – função a ser chamada quando da volta do contador a 0
  - `channels` – vetor de tamanho PWM_CHANNELS de configurações de canal
- É importante notar que, devido à limitação da quantidade de valores para o divisor de frequência dos temporizadores no ATMega328p, a frequência acima no caso deste MCU deve ser exatamente F_CPU dividida por um dos valores do divisor

HAL - módulo de sinais PWM

```
typedef uint8_t pwmmode_t;
typedef void (*pwmcallback_t)(PWMDriver *pwmp);

typedef struct {
    pwmmode_t                   mode;
    pwmcallback_t               callback;
} PWMChannelConfig;
```

- Cada canal de um driver PWM pode ser configurado através da estrutura `PWMChannelConfig`:
  - `mode` – modo do canal: `PWM_OUTPUT_DISABLED`, `PWM_OUTPUT_ACTIVE_HIGH` ou `PWM_OUTPUT_ACTIVE_LOW`
  - `callback` – função a ser chamada quando o contador é igual ao registrador do canal

HAL - módulo de sinais PWM

```
#define PWM_OUTPUT_MASK                 0x0FU
#define PWM_OUTPUT_DISABLED             0x00U
#define PWM_OUTPUT_ACTIVE_HIGH          0x01U
#define PWM_OUTPUT_ACTIVE_LOW           0x02U
```

Configuração do código

- Os possíveis valores do modo de operação de cada canal estão mostrados acima.

HAL - módulo de sinais PWM

```
#define PWM_OUTPUT_MASK                 0x0FU
#define PWM_OUTPUT_DISABLED             0x00U
#define PWM_OUTPUT_ACTIVE_HIGH          0x01U
#define PWM_OUTPUT_ACTIVE_LOW           0x02U
```

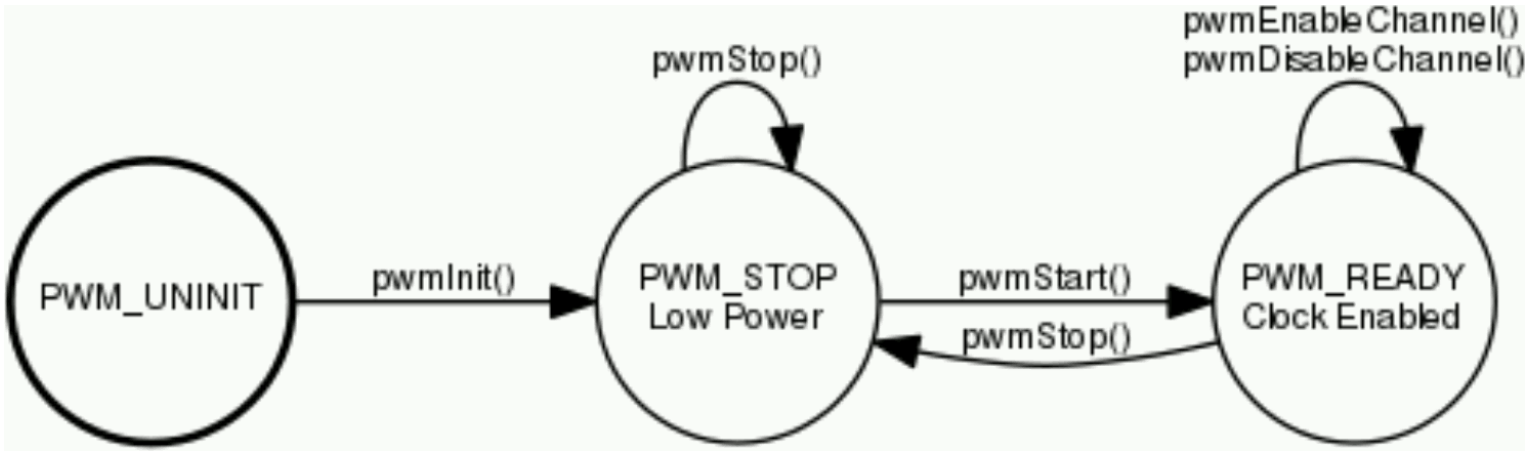
Configuração do código

- Os possíveis valores do modo de operação de cada canal estão mostrados acima.
- O callback é chamado quando o sinal passa da parte ativa para a inativa

Configuração do código

- Estas duas estruturas são definidas no arquivo os/hal/ports/AVR/hal_pwm_lld.h.

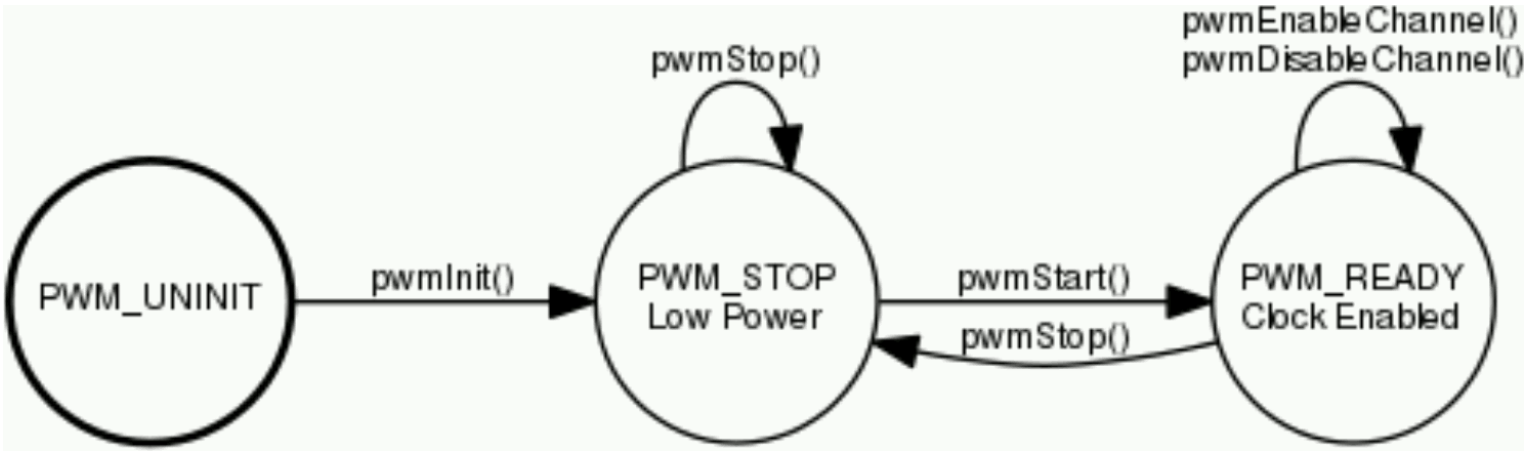
HAL - módulo de sinais PWM



Máquina de estados

- Um driver do módulo PWM tem 2 estados principais: PWM_STOP e PWM_READY.

HAL - módulo de sinais PWM



Máquina de estados

- Um driver do módulo PWM tem 2 estados principais: PWM_STOP e PWM_READY.
- Apenas em PWM_READY o driver pode produzir os sinais PWM.

HAL - módulo de sinais PWM

```
PWMConfig driver_config = {.frequency = PWM_FREQUENCIA,
                           .period = PWM_PERIODO,
                           .callback = pwm_periodo_cb,
                           .channels =
                           {{PWM_OUTPUT_ACTIVE_HIGH, 0},
                            {PWM_OUTPUT_DISABLED, 0}}
};

pwmStart(&PWMD1, &driver_config);
pwmEnableChannel(&PWMD1, 0, 0);
```

Operação

- Para utilizarmos o módulo PWM devemos fazer o seguinte procedimento:
  - Criar uma variável para configuração do driver e inicializá-la
  - Inicializar o driver com a configuração escolhida
  - Habilitar os canais desejados

Operação

- As principais funções são:
  - pwmStart() – inicia o clock do timer e inicializa a configuração
  - pwmStop() – para o clock
  - pwmEnableChannel() – habilita um canal, passado como parâmetro, de acordo com a configuração dada em pwmStart() e com o ciclo de trabalho passado como parâmetro (em pulsos de clock)
  - pwmDisableChannel() – desabilita um canal
  - pwmEnablePeriodicNotification() e pwmDisablePeriodicNotification() – habilita ou desabilita o callback do reset do contador
  - pwmEnableChannelNotification() e pwmDisableChannelNotification() – habilita ou desabilita o callback da comparação

HAL - módulo de sinais PWM

```
void pwmStart(PWMDriver *pwmp, const PWMConfig *config);
void pwmStop(PWMDriver *pwmp);
void pwmChangePeriod(PWMDriver *pwmp, pwmcnt_t period);
void pwmEnableChannel(PWMDriver *pwmp,
                      pwmchannel_t channel,
                      pwmcnt_t width);
void pwmDisableChannel(PWMDriver *pwmp, pwmchannel_t channel);
void pwmEnablePeriodicNotification(PWMDriver *pwmp);
void pwmDisablePeriodicNotification(PWMDriver *pwmp);
void pwmEnableChannelNotification(PWMDriver *pwmp, pwmchannel_t
channel);
void pwmDisableChannelNotification(PWMDriver *pwmp, pwmchannel_t
channel);
```

Operação

- As assinaturas das funções estão mostradas acima.

HAL - módulo de sinais PWM

Principais funções

- Os drivers são declarados como PWMD1, PWMD2, PWMD3, etc.
- É importante lembrar que os callbacks são executados no contexto de interrupções e por isso devem-se tomar alguns cuidados:
  - Ser o mais breve possível
  - Usar apenas funções terminadas em I ou X
  - Caso precisemos fazer tarefas longas, devemos usar threads
  - Para sistemas de tempo real, é recomendável evitar callbacks, exceto se forem realmente breves