Desenvolv. de Sistemas Embarcados em Tempo Real

Prof. Hermano Cabral

Departamento de Eletrônica e Sistemas — UFPE

6 de agosto de 2024

Plano de Aula

Tema central

- Chibios — Hardware Abstraction Layer

Plano de Aula

Tema central

- Chibios — Hardware Abstraction Layer

Objetivos

- Conhecer as características do módulo de comunicação serial do HAL

HAL - módulo de comunicação serial

Introdução

- Este módulo implementa um driver para comunicação serial full duplex.

HAL - módulo de comunicação serial

Introdução

- Este módulo implementa um driver para comunicação serial full duplex.
- O módulo usa buffers para comunicação entre a aplicação e o hardware.

HAL - módulo de comunicação serial

Introdução

- Este módulo implementa um driver para comunicação serial full duplex.
- O módulo usa buffers para comunicação entre a aplicação e o hardware.
- Além disso, o módulo oferece uma interface de uso mais rica, podendo ser usado, por exemplo, como destino para uma função semelhante à printf().

HAL - módulo de comunicação serial

Arquivos de implementação

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

Arquivos de configuração

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

Arquivos de configuração

- Para se usar o driver serial 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_SERIAL_USE_USART0 TRUE`
  - etc

Arquivos de configuração

- Para se usar o driver serial 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_SERIAL_USE_USART0 TRUE`
  - etc
- No arquivo halconf.h devemos selecionar a funcionalidade da serial:
  - `#define HAL_USE_SERIAL TRUE`

HAL - módulo de comunicação serial

Arquivos de configuração

- No arquivo halconf.h podemos definir também a velocidade de transmissão:
  - `#define SERIAL_DEFAULT_BITRATE 38400`

HAL - módulo de comunicação serial

```
typedef struct {
    uint16_t   sc_brr;
    uint8_t    sc_bits_per_char;
} SerialConfig;
```

Uso - configuração

- A configuração é feita através da estrutura SerialConfig e depende do microcontrolador que é usado.

HAL - módulo de comunicação serial

```
typedef struct {
    uint16_t   sc_brr;
    uint8_t    sc_bits_per_char;
} SerialConfig;
```

Uso - configuração

- A configuração é feita através da estrutura SerialConfig e depende do microcontrolador que é usado.
- Para o ATMega328p, a estrutura tem os seguintes campos:
  - `sc_brr`: valor para definir a velocidade de transmissão.
  - `sc_bits_per_char`: valor definindo o número de bits em um byte transmitido.

Uso - configuração

- Em geral, como podemos definir a velocidade de transmissão padrão no arquivo `halconf.h`, simplesmente usamos a configuração padrão.

Uso - configuração

- Em geral, como podemos definir a velocidade de transmissão padrão no arquivo `halconf.h`, simplesmente usamos a configuração padrão.
- Caso queiramos mudar a velocidade, usamos a macro `UBRR2x(b)` para escolher o valor de `sc_brr`.

Uso - configuração

- Em geral, como podemos definir a velocidade de transmissão padrão no arquivo `halconf.h`, simplesmente usamos a configuração padrão.
- Caso queiramos mudar a velocidade, usamos a macro `UBRR2x(b)` para escolher o valor de `sc_brr`.
- Para a quantidade de bits por caractere, fazemos uso das constantes `USART_CHAR_SIZE_5`, `USART_CHAR_SIZE_6`, `USART_CHAR_SIZE_7`, `USART_CHAR_SIZE_8` e `USART_CHAR_SIZE_9`.

Uso - configuração

- Em geral, qualquer que seja a plataforma, como podemos definir a velocidade de transmissão padrão no arquivo halconf.h, usamos a configuração-padrão.
  - A configuração-padrão usa os parâmetros 8N1.

Uso - configuração

- Em geral, qualquer que seja a plataforma, como podemos definir a velocidade de transmissão padrão no arquivo halconf.h, usamos a configuração-padrão.
  - A configuração-padrão usa os parâmetros 8N1.
  - Podemos usar a configuração-padrão passando um ponteiro nulo como o ponteiro para a estrutura de configuração.

HAL - módulo de comunicação serial

Introdução

- Para usarmos o módulo serial, seguimos o seguinte procedimento:
  - Criamos uma variável para configuração do driver e inicializamo-la (se não for usar a configuração-padrão).
  - Inicializamos o driver com a configuração escolhida.
  - Escolhemos o modo dos pinos TX e RX (no caso do Stm32).
  - Escrevemos ou lemos do módulo usando uma das funções apropriadas.

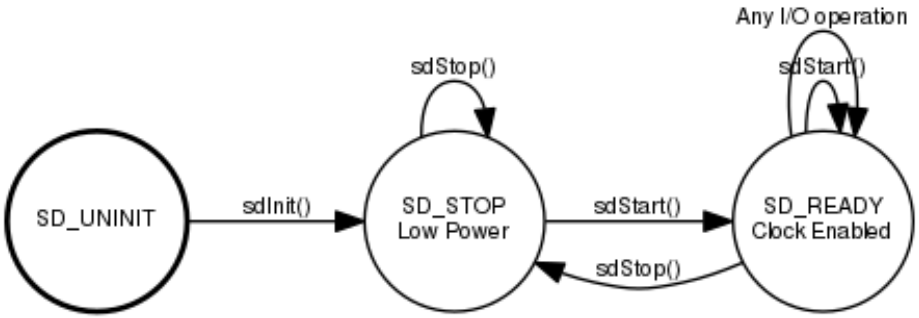
HAL - módulo de comunicação serial

```
int main(void) {
    uint8_t msg[] = "Hello, World!";
    SerialConfig config = {.sc_brr = UBRR2x(500000),
                           .sc_bits_per_char = USART_CHAR_SIZE_8
    };

    halInit();
    chSysInit();

    sdStart(&SD1, &config);
    sdWrite(&SD1, msg, sizeof(msg));

    while (1) {}
}
```

Exemplo

- Um exemplo de uso do módulo serial está mostrado acima.
- Podemos usar um ponteiro nulo para SerialConfig caso queiramos a configuração padrão definida como 8N1 e com os valores do arquivo halconf.h.

HAL - módulo de comunicação serial



Máquina de estados

- O módulo segue a máquina de estados acima.

HAL - módulo de comunicação serial

Observações

- O módulo serial utiliza buffers para a recepção e transmissão de bytes pela serial.

Observações

- O módulo serial utiliza buffers para a recepção e transmissão de bytes pela serial.
- Assim, não escrevemos ou lemos diretamente do hardware do periférico, mas sim de posições na memória.

HAL - módulo de comunicação serial

Observações

- O módulo serial utiliza buffers para a recepção e transmissão de bytes pela serial.
- Assim, não escrevemos ou lemos diretamente do hardware do periférico, mas sim de posições na memória.
- Isso torna o uso do módulo mais semelhante à programação sequencial sem sofrer atrasos significativos.

HAL - módulo de comunicação serial

Observações

- O módulo serial utiliza buffers para a recepção e transmissão de bytes pela serial.
- Assim, não escrevemos ou lemos diretamente do hardware do periférico, mas sim de posições na memória.
- Isso torna o uso do módulo mais semelhante à programação sequencial sem sofrer atrasos significativos.
- O problema é que os buffers são finitos, e podem ser sobrescritos.

Uso - configuração

- Para o início das operações e configuração da interface, usamos a função `sdStart()`.

Uso - configuração

- Para o início das operações e configuração da interface, usamos a função `sdStart()`.
- Para encerrar a operação, usamos a função `sdWrite()`.

Principais funções

- Depois de configurado e inicializado, fazemos uso do módulo através das seguintes funções:
  - Para escrita: `sdPut()` e `sdWrite()`.

Principais funções

- Depois de configurado e inicializado, fazemos uso do módulo através das seguintes funções:
  - Para escrita: `sdPut()` e `sdWrite()`.
  - Para leitura: `sdGet()` e `sdRead()`.

Principais funções

- Depois de configurado e inicializado, fazemos uso do módulo através das seguintes funções:
  - Para escrita: `sdPut()` e `sdWrite()`.
  - Para leitura: `sdGet()` e `sdRead()`.
- Estas funções bloqueiam se não for possível completar a operação naquele momento.

HAL - módulo de comunicação serial

```
void sdStart(SerialDriver *sdp, const SerialConfig *config);
msg_t sdPut(SerialDriver *sdp, uint8_t b);
msg_t sdGet(SerialDriver *sdp);
size_t sdWrite(SerialDriver *sdp, uint8_t *bp, size_t n);
size_t sdRead(SerialDriver *sdp, uint8_t *bp, size_t n);
void sdStop(SerialDriver *sdp);
```

Principais funções

- As assinaturas das funções são as acima.

Principais funções

- As versões não-bloqueantes da escrita e da leitura adicionam a palavra `Timeout` no nome da função:
  - Para escrita: `sdPutTimeout()` e `sdWriteTimeout()`.

Principais funções

- As versões não-bloqueantes da escrita e da leitura adicionam a palavra `Timeout` no nome da função:
  - Para escrita: `sdPutTimeout()` e `sdWriteTimeout()`.
  - Para leitura: `sdGetTimeout()` e `sdReadTimeout()`.

Principais funções

- Além disso, essas funções não podem ser usadas no contexto de interrupções.

Principais funções

- Além disso, essas funções não podem ser usadas no contexto de interrupções.
- Neste caso, existem as funções apropriadas que tem a letra 'I' no final:
  - Para escrita: `sdPutI()` e `sdWriteI()`.

Principais funções

- Além disso, essas funções não podem ser usadas no contexto de interrupções.
- Neste caso, existem as funções apropriadas que tem a letra 'I' no final:
  - Para escrita: `sdPutI()` e `sdWriteI()`.
  - Para leitura: `sdGetI()` e `sdReadI()`.

HAL - módulo de comunicação serial

Operação

- Para usarmos o módulo serial, seguimos o seguinte procedimento:
  - Criamos uma variável para configuração do driver e inicializamo-la (se não for usar a configuração-padrão).
  - Inicializamos o driver com a configuração escolhida.
  - Escolhemos o modo dos pinos TX e RX (no caso do Stm32).
  - Escrevemos ou lemos do módulo usando uma das funções apropriadas.

HAL - módulo de comunicação serial

```
int main(void) {
    uint8_t msg[] = "Hello, World!";
    SerialConfig config = {.sc_brr = UBRR2x(500000),
                           .sc_bits_per_char = USART_CHAR_SIZE_8
    };

    halInit();
    chSysInit();

    sdStart(&SD1, &config);
    sdWrite(&SD1, msg, sizeof(msg));

    while (1) {}
}
```

Exemplo

- Um exemplo de uso do módulo serial está mostrado acima.
  - Podemos usar um ponteiro nulo para SerialConfig caso queiramos a configuração padrão definida como 8N1 e com os valores do arquivo halconf.h.

HAL - módulo de comunicação serial

Introdução

- Além das funções anteriores, o módulo serial também implementa uma interface do tipo channel, que basicamente é uma stream de caracteres.

HAL - módulo de comunicação serial

Introdução

- Além das funções anteriores, o módulo serial também implementa uma interface do tipo channel, que basicamente é uma stream de caracteres.
- Isto permite o módulo ser usado em todo lugar onde se espera uma stream

HAL - módulo de comunicação serial

Introdução

- Além das funções anteriores, o módulo serial também implementa uma interface do tipo channel, que basicamente é uma stream de caracteres.
- Isto permite o módulo ser usado em todo lugar onde se espera uma stream
- Talvez o uso mais comum seja o da função chprintf()

HAL - módulo de comunicação serial

Introdução

- Além das funções anteriores, o módulo serial também implementa uma interface do tipo channel, que basicamente é uma stream de caracteres.
- Isto permite o módulo ser usado em todo lugar onde se espera uma stream
- Talvez o uso mais comum seja o da função chprintf()
- Esta função fornece as principais funcionalidades da função printf() enviando a string resultante pelo canal passado como primeiro parâmetro

HAL - módulo de comunicação serial

Introdução

- Além das funções anteriores, o módulo serial também implementa uma interface do tipo channel, que basicamente é uma stream de caracteres.
- Isto permite o módulo ser usado em todo lugar onde se espera uma stream
- Talvez o uso mais comum seja o da função chprintf()
- Esta função fornece as principais funcionalidades da função printf() enviando a string resultante pelo canal passado como primeiro parâmetro
- Para usá-la, precisamos fazer algumas adições no projeto:
  - Incluir o arquivo chprintf.h em nosso código
  - Incluir "$(CHIBIOS)/os/hal/lib/streams/chprintf.c" na lista de arquivos a compilar na variável CSRC no arquivo Makefile
  - Incluir "$(CHIBIOS)/os/hal/lib/streams" na variável INCDIR no arquivo Makefile

HAL - módulo de comunicação serial

```
#include "hal_serial.h"
#include "chprintf.h"

int main(void) {
    const char msg[] = "Hello, %s!";
    SerialConfig config = {.sc_brr = UBRR2x(500000),
                           .sc_bits_per_char = USART_CHAR_SIZE_8
    };

    halInit();
    chSysInit();

    sdStart(&SD1, &config);
    chprintf((BaseSequentialStream *) &SD1, msg, "World");

    while (1) {}
}
```

Introdução

- Um exemplo de uso do módulo serial com a função chprintf() está mostrado acima.