Desenvolv. de Sistemas Embarcados em Tempo Real

Prof. Hermano Cabral

Departamento de Eletrônica e Sistemas — UFPE

8 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 I2C do HAL

HAL - módulo de comunicação I2C

Introdução

- Este módulo implementa um driver genérico para comunicação I2C, um protocolo de comunicação serial síncrono half-duplex.

HAL - módulo de comunicação I2C

Introdução

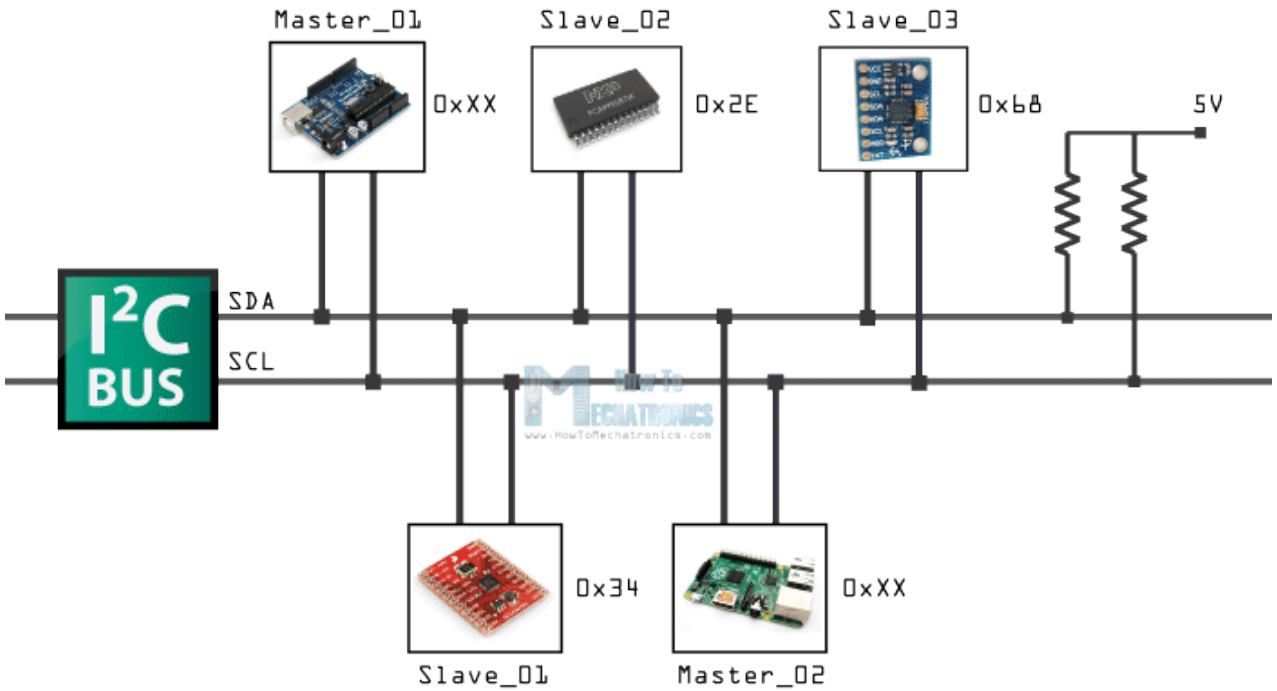
- Este módulo implementa um driver genérico para comunicação I2C, um protocolo de comunicação serial síncrono half-duplex.
- I2C usa 2 fios para comunicação, um para dados e controle e outro para o clock.

HAL - módulo de comunicação I2C

Introdução

- Este módulo implementa um driver genérico para comunicação I2C, um protocolo de comunicação serial síncrono half-duplex.
- I2C usa 2 fios para comunicação, um para dados e controle e outro para o clock.
- Na literatura, o pino de dados é denotado por SDA e o de relógio, de SCL.

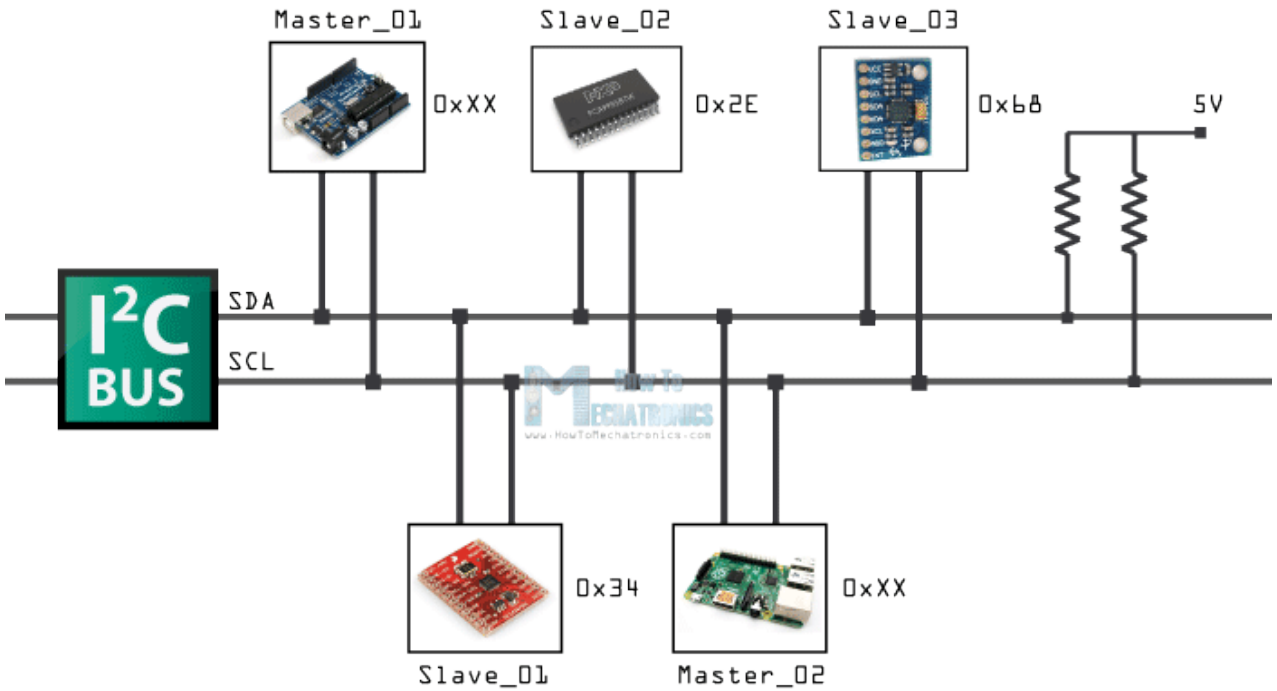
HAL - módulo de comunicação I2C



Descrição

- Em I2C, os 2 fios são na realidade um barramento onde os pinos estão na configuração condutor aberto.

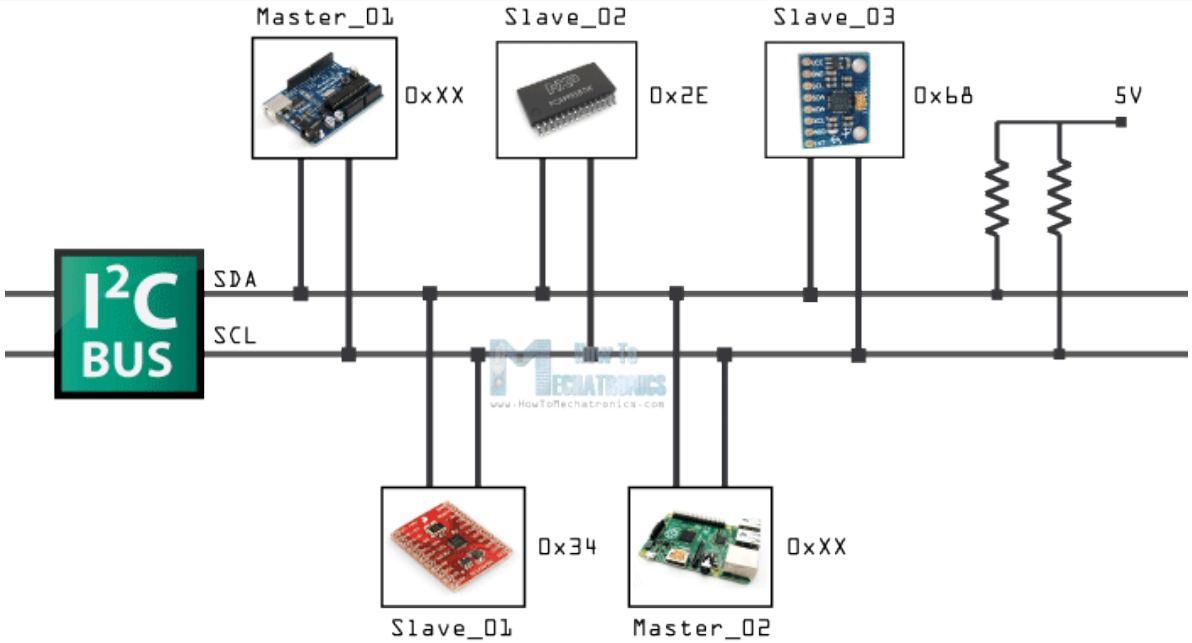
HAL - módulo de comunicação I2C



Descrição

- Em I2C, os 2 fios são na realidade um barramento onde os pinos estão na configuração condutor aberto.
- Neste barramento podemos colocar uma grande quantidade de dispositivos que podem se comunicar entre si.

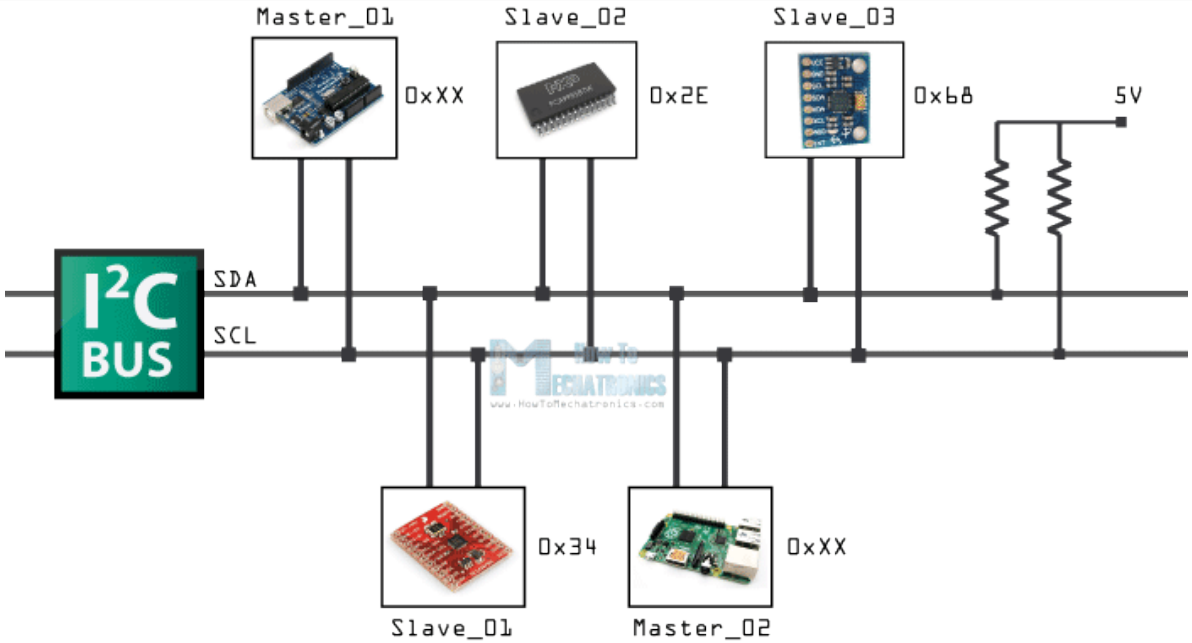
HAL - módulo de comunicação I2C



Descrição

- Em uma comunicação I2C, existe o conceito de mestre e escravo.

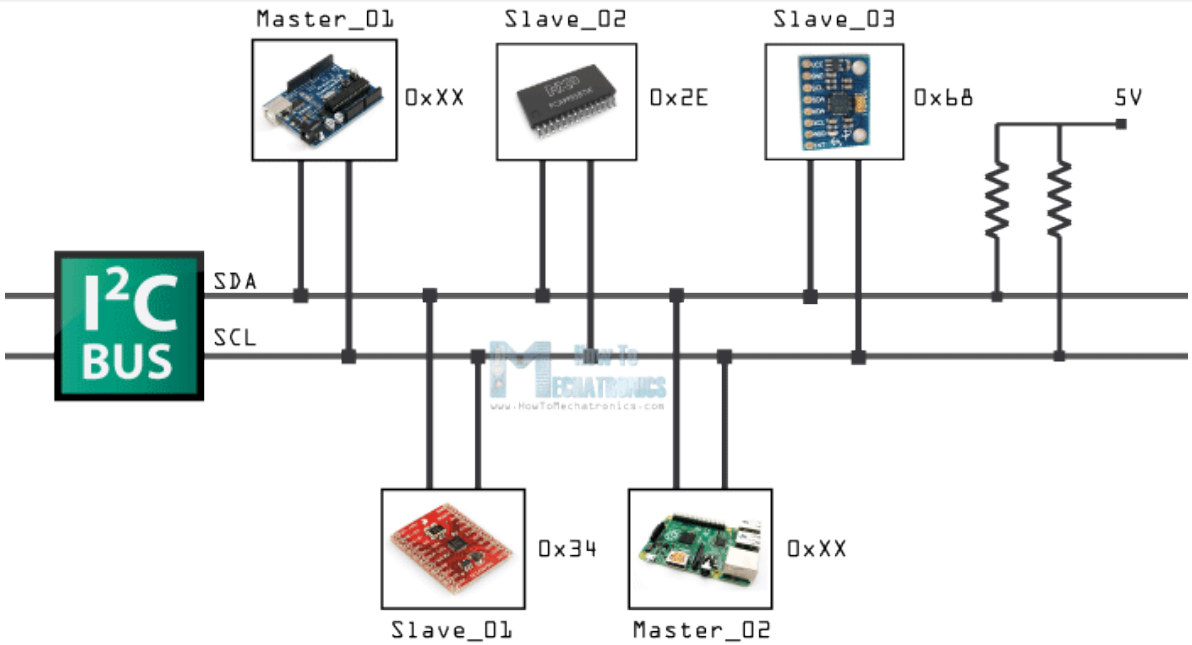
HAL - módulo de comunicação I2C



Descrição

- Em uma comunicação I2C, existe o conceito de mestre e escravo.
- O dispositivo-mestre controla o clock, e portanto a velocidade de transmissão.

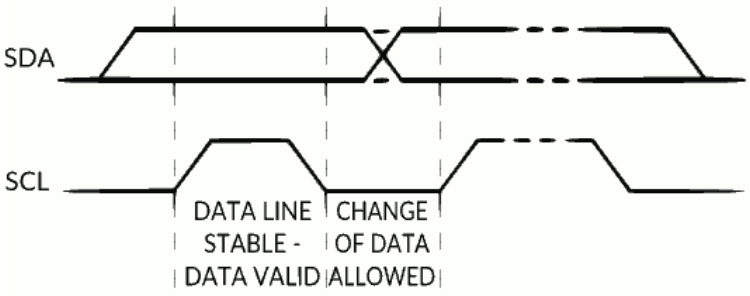
HAL - módulo de comunicação I2C

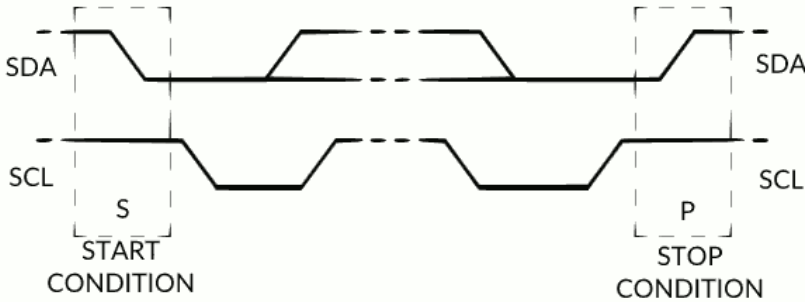


Descrição

- Em uma comunicação I2C, existe o conceito de mestre e escravo.
- O dispositivo-mestre controla o clock, e portanto a velocidade de transmissão.
- O dispositivo-escravo possui um endereço que é usado para distingui-lo dos outros dispositivos-escravo.

HAL - módulo de comunicação I2C

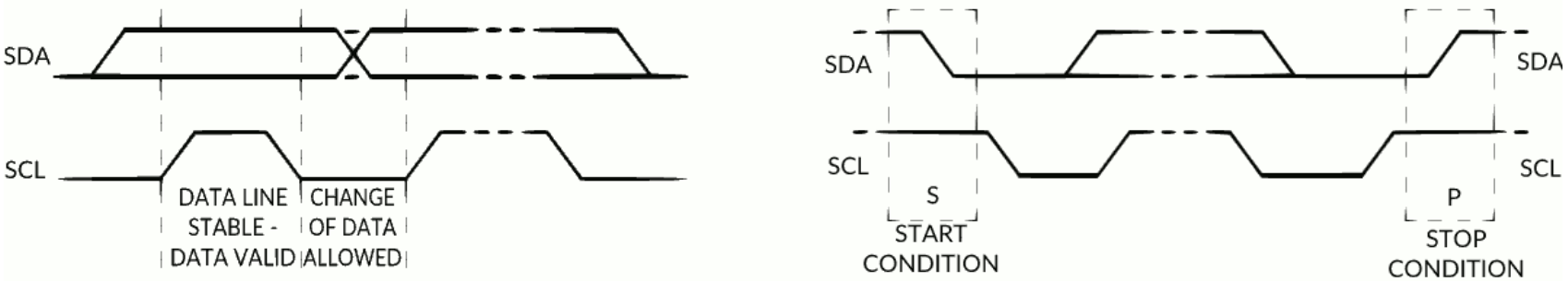




Descrição

- Em um microcontrolador, existe um módulo interno que cuida do protocolo I2C.

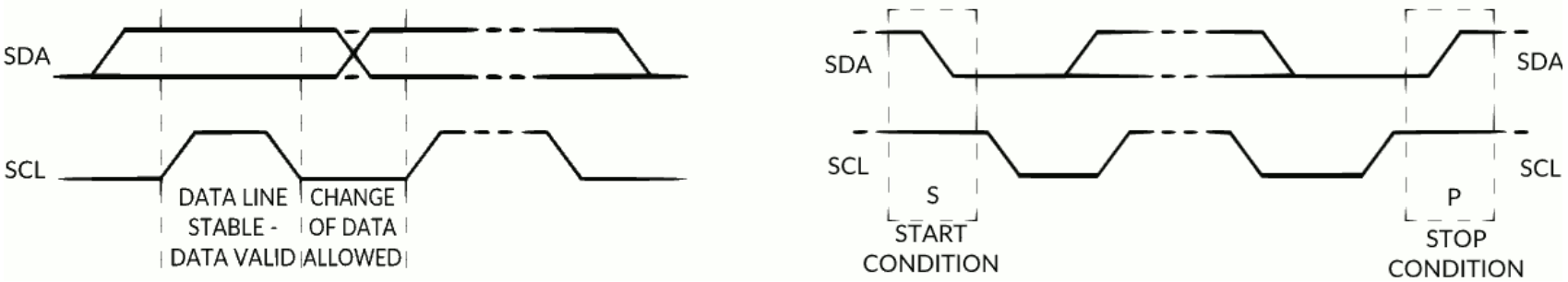
HAL - módulo de comunicação I2C



Descrição

- Em um microcontrolador, existe um módulo interno que cuida do protocolo I2C.
- Em I2C, toda mudança na linha de dados deve ser feita quando o clock estiver no nível baixo.

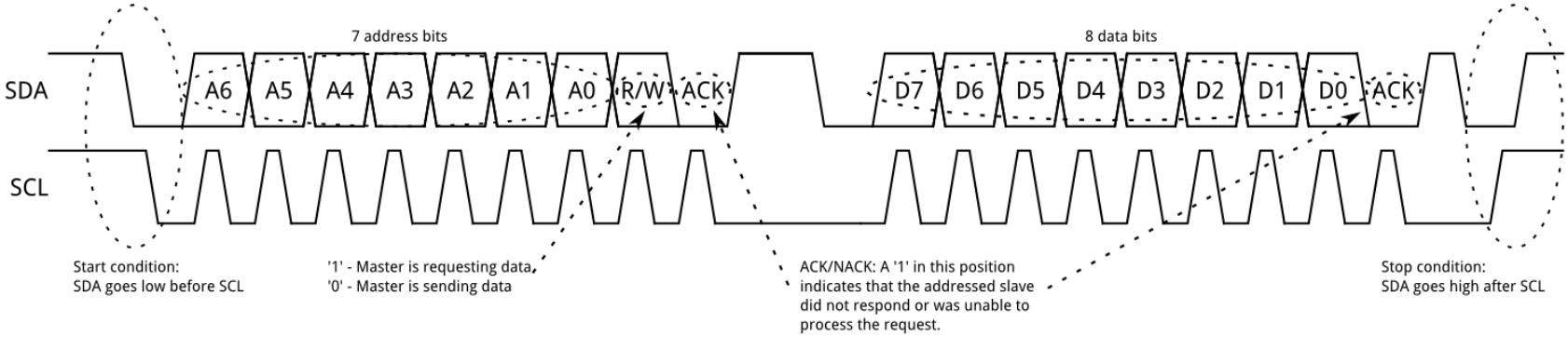
HAL - módulo de comunicação I2C



Descrição

- Em um microcontrolador, existe um módulo interno que cuida do protocolo I2C.
- Em I2C, toda mudança na linha de dados deve ser feita quando o clock estiver no nível baixo.
- A exceção é o início e o fim de uma comunicação, onde o barramento é requisitado ou liberado.

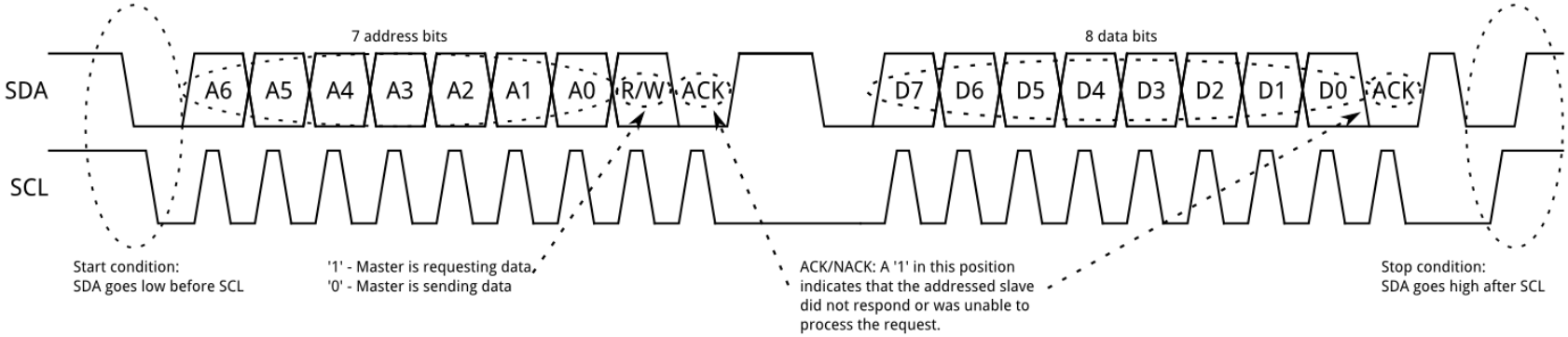
HAL - módulo de comunicação I2C



Descrição

- Todo byte transmitido deve ser dado acknowledgment pelo receptor.

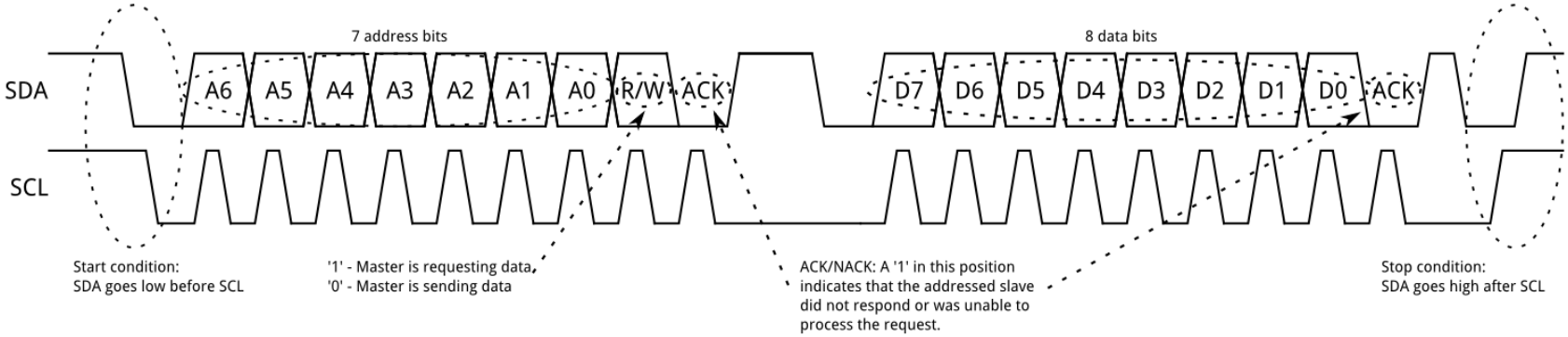
HAL - módulo de comunicação I2C



Descrição

- Todo byte transmitido deve ser dado acknowledgment pelo receptor.
- Toda comunicação I2C inicia-se com o 1º byte indicando o endereço do escravo.

HAL - módulo de comunicação I2C



Descrição

- Todo byte transmitido deve ser dado acknowledgment pelo receptor.
- Toda comunicação I2C inicia-se com o 1º byte indicando o endereço do escravo.
- O 1º bit deste 1º byte também indica o sentido da transmissão.

HAL - módulo de comunicação I2C

Arquivos de implementação

- Este módulo é implementado através dos seguintes arquivos dentro do diretório os/hal:
  - include/hal_i2c.h e src/hal_i2c.c, para a interface de alto nível do HAL
  - ports/AVR/hal_i2c_lld.h e ports/AVR/hal_i2c_lld.c, para a implementação de baixo nível em hardware

Arquivos de configuração

- Para se usar o driver I2C, é preciso configurar os arquivos mcuconf.h e halconf.h.

Arquivos de configuração

- Para se usar o driver I2C, 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_I2C_USE_I2C1 TRUE`
  - etc

Arquivos de configuração

- Para se usar o driver I2C, 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_I2C_USE_I2C1 TRUE`
  - etc
- No arquivo halconf.h devemos selecionar a funcionalidade da serial:
  - `#define HAL_USE_I2C TRUE`

HAL - módulo de comunicação I2C

```
typedef struct {
    uint32_t        clock_speed;
} I2CConfig;
```

Configuração

- A configuração é feita através da estrutura I2CConfig e depende do microcontrolador que é usado.

HAL - módulo de comunicação I2C

```
typedef struct {
    uint32_t        clock_speed;
} I2CConfig;
```

Configuração

- A configuração é feita através da estrutura I2CConfig e depende do microcontrolador que é usado.
- Para o ATMega328p, a estrutura consiste de um único campo, clock_speed, que define a velocidade do sinal de relógio.

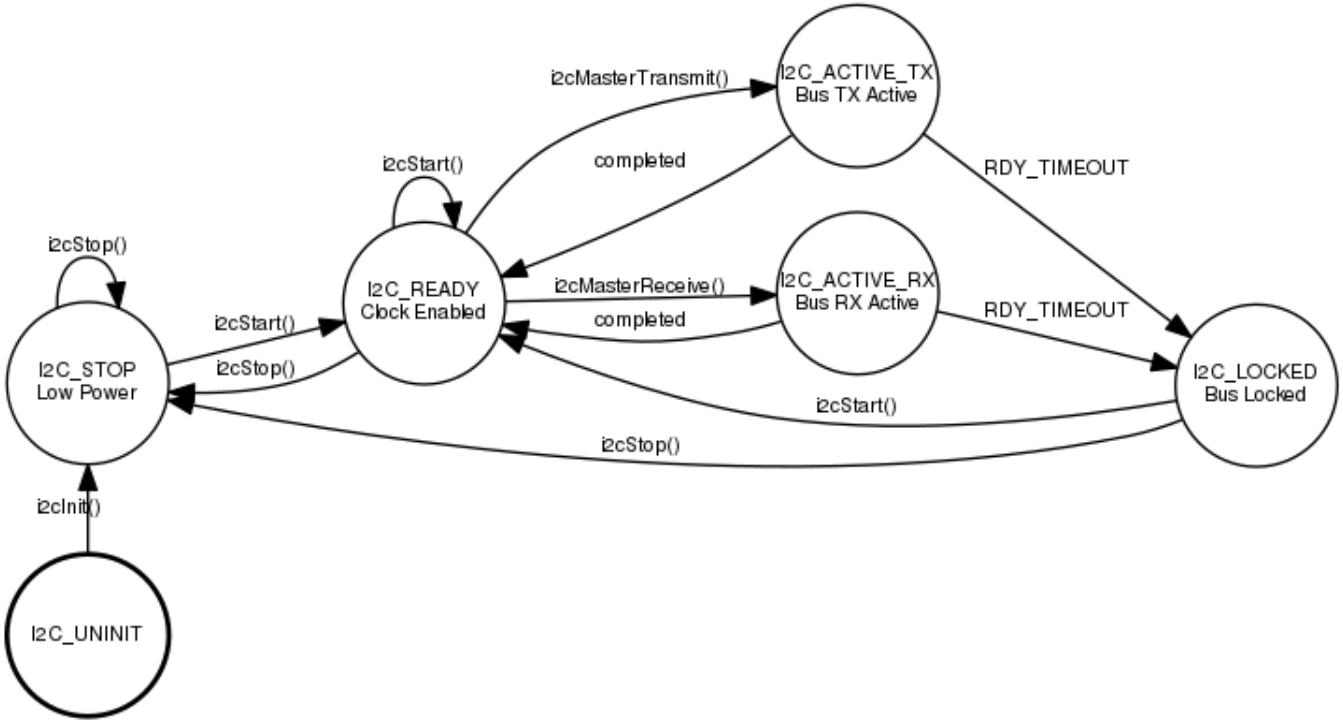
HAL - módulo de comunicação I2C

```
typedef struct {
    uint32_t        clock_speed;
} I2CConfig;
```

Configuração

- A configuração é feita através da estrutura I2CConfig e depende do microcontrolador que é usado.
- Para o ATMega328p, a estrutura consiste de um único campo, clock_speed, que define a velocidade do sinal de relógio.
- O padrão I2C define como velocidade máxima 400 kHz.

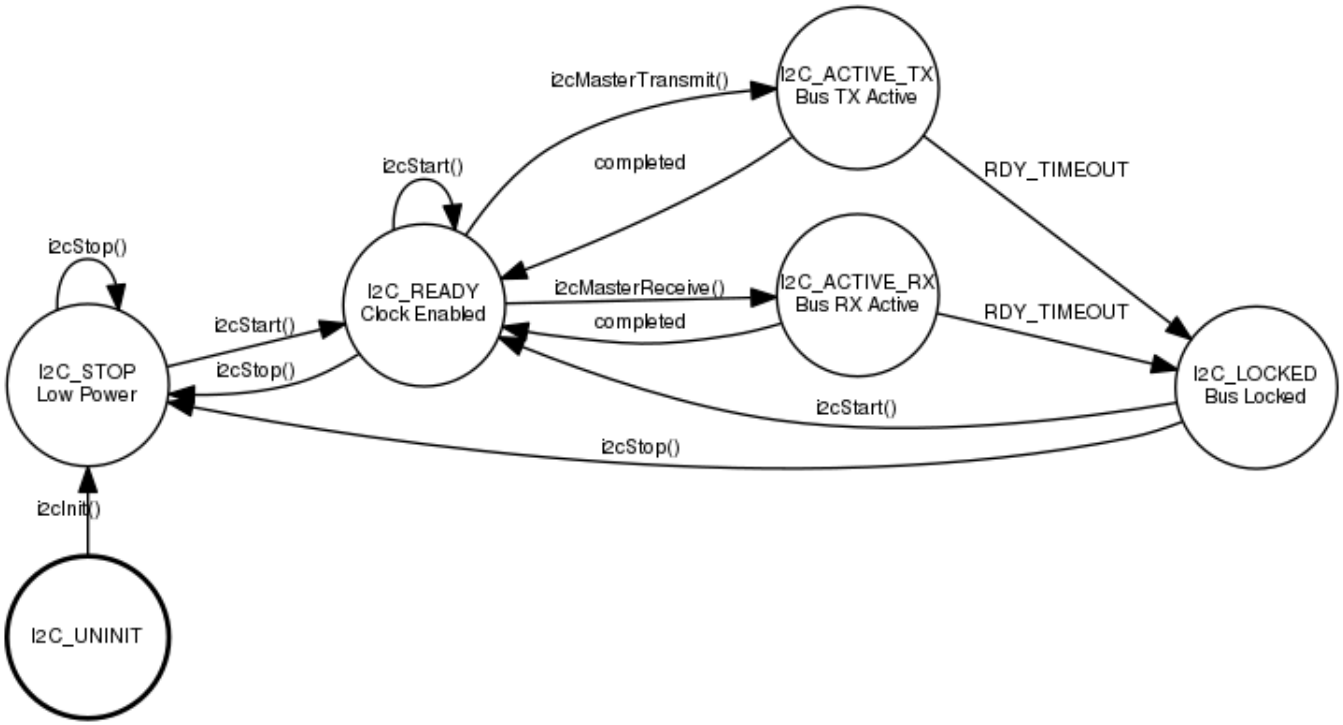
HAL - módulo de comunicação I2C



Máquina de estados

- Os principais estados deste módulo são:
  - I2C_STOP – o hardware encontra-se desativado
  - I2C_READY – pronto para iniciar a comunicação
  - I2C_ACTIVE_TX / I2C_ACTIVE_RX – comunicação iniciada

HAL - módulo de comunicação I2C



Máquina de estados

- Os principais estados deste módulo são:
  - I2C_LOCKED – erro na comunicação

Operação

- O módulo I2C tem as seguintes funções básicas:
  - `i2cStart()`, `i2cStop()`: para iniciar e parar o driver
  - `i2cMasterTransmitTimeout()`: para transmitir uma sequência de bytes do mestre para o escravo e, opcionalmente, receber outra sequência.
  - `i2cMasterReceiveTimeout()`: para receber uma sequência de bytes do escravo para o mestre.

HAL - módulo de comunicação I2C

Operação

- O módulo I2C tem as seguintes funções básicas:
  - `i2cStart()`, `i2cStop()`: para iniciar e parar o driver
  - `i2cMasterTransmitTimeout()`: para transmitir uma sequência de bytes do mestre para o escravo e, opcionalmente, receber outra sequência.
  - `i2cMasterReceiveTimeout()`: para receber uma sequência de bytes do escravo para o mestre.
- O módulo não usa buffers, e por isso dá a opção de timeouts

HAL - módulo de comunicação I2C

```
void i2cStart(I2CDriver *i2cp, const I2CConfig *config);
void i2cStop(I2CDriver *i2cp);
msg_t i2cMasterTransmitTimeout(I2CDriver *i2cp,
                               i2caddr_t addr,
                               const uint8_t *txbuf, size_t txbytes,
                               uint8_t *rxbuf, size_t rxbytes,
                               sysinterval_t timeout);
msg_t i2cMasterReceiveTimeout(I2CDriver *i2cp,
                              i2caddr_t addr,
                              uint8_t *rxbuf, size_t rxbytes,
                              sysinterval_t timeout);
```

Operação

- As assinaturas das funções são as acima.

Operação

- Existem duas constantes comumente usadas para o valor do timeout:
  - `#define TIME_IMMEDIATE ((sysinterval_t) 0)`
  - `#define TIME_INFINITE ((sysinterval_t) -1)`
- <3->`TIME_INFINITE` é usado para especificar que a função só deve ser retornar quando a operação completar.

Operação

- Existem duas constantes comumente usadas para o valor do timeout:
  - `#define TIME_IMMEDIATE ((sysinterval_t) 0)`
  - `#define TIME_INFINITE ((sysinterval_t) -1)`
- `TIME_IMMEDIATE` é usado quando não queremos que a função seja retida se a operação não puder ser completada de imediato.
- <3->`TIME_INFINITE` é usado para especificar que a função só deve ser retornar quando a operação completar.

Operação

- Para usarmos o módulo I2C, seguimos o seguinte procedimento:
  - Criamos uma variável para configuração do driver e inicializamo-la.
  - Inicializamos o driver com a configuração escolhida.
  - Escrevemos ou lemos do módulo usando uma das funções apropriadas.

HAL - módulo de comunicação I2C

```
#define PEER_ADDRESS 0X0A

int main(void) {
    uint8_t msg[] = "Hello, World!";
    I2CConfig config = {.clock_speed = 200000};

    halInit();
    chSysInit();

    i2cStart(&I2CD1, &config);
    i2cMasterTransmitTimeout(&I2CD1, PEER_ADDRESS, msg,
    sizeof(msg), 0, 0, TIME_INFINITE);

    while (1) {}
}
```

Máquina de estados

- Um exemplo de uso do módulo I2C para o ATMega328p está mostrado acima.