Sistemas Embebidos II

Semestre de Verão de 2010/2011

Primeira actividade prática

1. Protocolo SPI

Considere a implementação do protocolo SPI, pelo processo *bit-bang,* estudada em aula.
Utilize o programa seguinte para teste e estudo do protocolo.

```
#include <string.h>
#include "spi.h"
#include "utils.h"

Spi_device spis[] = {
      {            /* ENC28J60 */
            9,           /* Pino do GPIO para CS */
            10,          /* o sinal SCLK terá a frequência = PCLK / este valor */
            0,           /* modo SPI */
            8            /* numero de bits por palavra */
      }
};

int main() {
      U32 aux;
      lpc2106_init();
      spi_init(spis, sizeof_array(spis));
      while (1) {
            U8 output_buffer[] = {0x35, 0x7a};
            U8 input_buffer[sizeof(output_buffer)];
            spi_transaction_begin(&spis[0]);
            spi_transaction_transfer(&spis[0], 2, output_buffer, input_buffer);
            spi_transaction_end(&spis[0]);
      }
}
```

Com um osciloscópio, observe os sinais MOSI e SCLK e confira com o programa.

Para teste desta API vamos usar o controlador Ethernet ENC28J60. Este dispositivo possui uma interface SPI com palavras de 8 bits, modo 0 e transmite o bit de maior peso em primeiro lugar.

Estabeleça a montagem do módulo ENC28J60-H da Olimex, desenhando um esquema de ligações. Aconselha-se a ligação da entrada RST a um pino do GPIO. Assim, o módulo pode ser iniciado independentemente do resto do sistema.

Para teste das ligações ler um registo interno do ENC28J60 cujo conteúdo, após o *reset*, é um valor conhecido. Por exemplo, o registo ECON2 devolve o valor 0x80.
Para ler este registo deve activar o sinal RST (impulso a zero) e de seguida executar uma transacção de 2 bytes em que no primeiro é enviado o valor 0x1E e no segundo seria recebido o valor 0x80.

2. Consola

No desenvolvimento de software para sistemas embebidos a utilização de um canal série para entrada e saída de dados é uma ferramenta de apoio importante. Esta permite a apresentação de informação de forma legível ao programador facilitando assim a análise do comportamento dos programas.

Aconselha-se a implementação da seguinte interface, para servir de consola, sobre o canal série 0.

```
void console_init();
void console_write_char(char c);
size_t console_write_block(const char * data, size_t size);
void console_write_str(const char * str);
int console_printf(const char * fmt, ...);
int console_vprintf(const char * fmt, va_list vlp);
void console_dump_hex(U8 * buffer, size_t size);

char console_read_char();
size_t console_read_block(char * buffer, size_t size);
int console_scanf(const char *fmt, ...);
```

Baseie as funções anteriores sobre as seguinte interface de utilização do canal série.

```
void serial_init(U32 uart_base_address,
                 long baud_rate,
                 int data_bits,      /* 5, 6, 7, 8 */
                 int stop_bits,      /* 1, 2 */
                 char parity);       /* 'e', 'o', 'n' */

void serial_write_char(U32 uart_base_address, char c);

char serial_read_char(U32 uart_base_address);
```

3. Ethernet

Na aula teórica foi esboçada a implementação da seguinte interface de programação para acesso à rede Ethernet, sobre o controlador ENC28J60 da Microchip.

```
void ethernet_init(U8 * mac);
void ethernet_send(U8* packet, size_t packet_size);
size_t ethernet_recv(U8* buffer, size_t buffer_size);
```

Complete essa implementação e produza um módulo compilável.

3.1. Ensaio da recepção de pacotes

Utilize o programa abaixo para testar a recepção de pacotes Ethernet. Este programa, depois de iniciar o controlador Ethernet invoca sucessivamente a função **`ethernet_recv`**. Esta função verifica se existe algum pacote recebido por parte do controlador e devolve a sua dimensão. A dimensão zero indica que não foi recebido qualquer pacote.

Em redes locais como as do ISEL há sempre muitos pacotes em *broadcast*, se ligar a uma destas redes não demorará muito até o controlador receber um pacote.
Se ligar ao computador de desenvolvimento execute o comando **ping** para um endereço na mesma rede local. Irá receber logo um pacote **ARP request** que é transmitido em *broadcast*.

```
#include "console.h"
#include "ethernet.h"
#include "spi.h"
#include "utils.h"

Spi_device spis[] = {
      {             /* ENC28J60 */
            9,            /* Pino do GPIO para CS */
            10,           /* o sinal SCLK terá a frequência = PCLK / este valor */
            0,            /* modo SPI */
```

```
            8            /* numero de bits por palavra */
      }
};

static U8 ether_addr[] = {0x02, 0x65, 0x7A, 0x65, 0x71, 00};

static U8 packet[2000];

int main(void) {
      lpc2106_init();
      console_init();
      console_printf("LPC2106: ENC28J60-H Ethernet\n\r");
      spi_init(spis, lengthof(spis));
      ethernet_init(ether_addr);

      while (1) {
            size_t size;
            while (size = ethernet_recv(packet, sizeof(packet))) {
                  console_printf("Packet received\n\r");
                  console_dump_hex(packet, size);
            }
      }
}
```

3.2. Ensaio da transmissão de pacotes

Para testar o envio de pacotes, através da função **`ethernet_send`**, acrescente o seguinte troço de programa no ciclo **`while (1)`**. Este código envia um pacote Ethernet em *broadcast*, sempre que se prime uma tecla na consola.
Para verificar o correcto envio pode usar um programa espião TCP/IP como o WireShark.

```
            if (console_read_size() > 0) {
                  char c = console_read_char();
                  console_printf("tx\n\r");
                  U8 frame[78];
                  memcpy(&frame[0], "\xff\xff\xff\xff\xff\xff", 6);
                  memcpy(&frame[6], ether_addr, 6);
                  frame[12] = (sizeof(frame) - 14) >> 8;
                  frame[13] = (sizeof(frame) - 14);
                  memset(&frame[14], '2', sizeof(frame) - 14);
                  ethernet_send(frame, sizeof(frame));
            }
```

A função **`console_read_size`** não fazia parte da interface da consola realizada no ponto anterior. Esta função devolve o número de caracteres recebidos pelo UART e disponíveis para serem lidos pelas funções **`console_read_...`**. No caso de interacção por *polling* sobre o UART e não utilizando os FIFOs internos, esta função devolverá apenas os valores 0 ou 1, conforme o estado da *flag* DR.