Laboratorio 4:

E/S programada

control de una UART y comunicación con un terminal serie

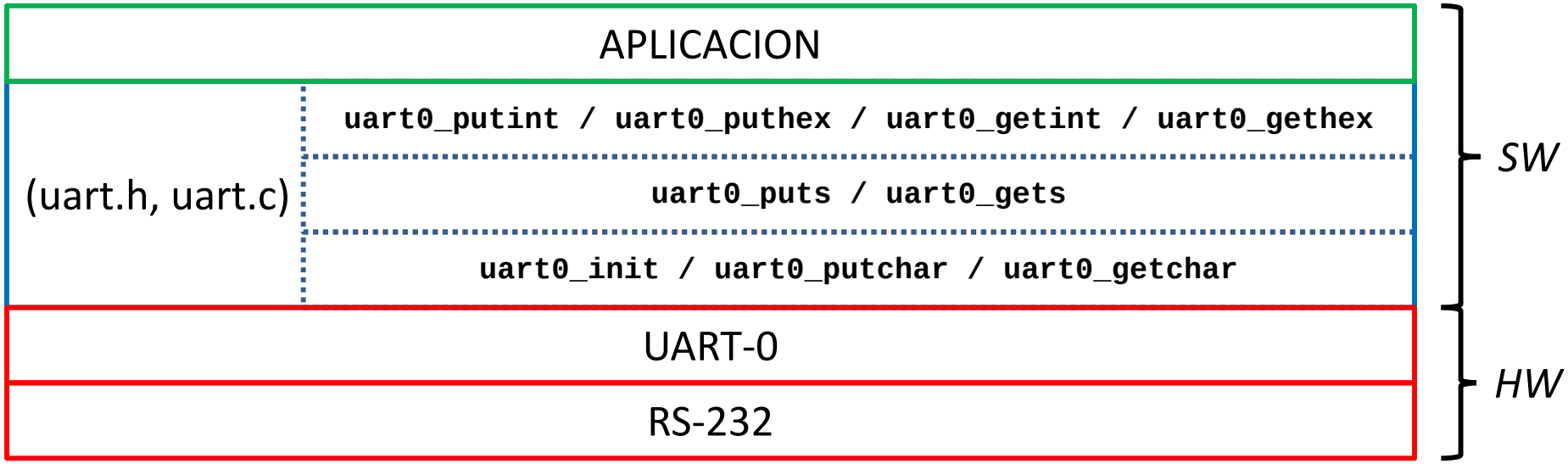
Programación de sistemas y dispositivos

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

J.M. Mendías
2016

Presentación

- Desarrollar una capa de firmware para la comunicación con un host remoto a través de un canal RS-232.
  - Las características (formato, velocidad, etc.) serán fijas.
  - No se realizará tratamiento de errores de transmisión.
  - El envío/recepción de datos será por pooling (E/S programada).
  - Implementaremos 9 funciones en 3 niveles.
    - Inicialización y envío/recepción de caracteres.
    - Envío/recepción de cadenas de caracteres.
    - Envío/recepción de números decimales/hexadecimales como cadenas de caracteres.



Presentación

- El host remoto de prueba será un PC sobre el que corre Termite un emulador de terminal serie.

- Las características de bajo nivel del terminal son:
  - Formato de datos: normal (no infrarrojos), sin paridad, 1 bit de stop, 8 bits de datos
  - Velocidad: 115200 baudios
  - Sin control flujo (manual).

- Las características de alto nivel del terminal son:
  - Los caracteres se transmiten en ASCII
  - Hace eco local de los caracteres que envía.
  - Las líneas finalizan con LF ('\n')
    - Es decir, cuando se pulsa return en el host, envía un LF ('\n').
  - Cuando recibe LF ('\n') avanza línea y retorna el carro.

Termite debe estar configurado según **`checklist-configuración.pdf`**

Conexión con un terminal

configuración UART0 (i)

- Formato de datos:
  - ULCON0[6] = 0 — normal (no infrarrojos)
  - ULCON0[5:3] = 0 — sin paridad
  - ULCON0[2] = 0 — 1 bit de stop
  - ULCON0[1:0] = 3 — 8 bits de datos
- Velocidad de transmisión: 115200 baudios
  - UBRDIV0 = 34 — 64MHz / (115200 × 16) - 1
- Control automático de flujo: desactivado
  - UMCON0[4] = 0

- Para fijar en C el valor de cada uno de los campos de un registro
  - Si todos los campos son constantes, puede asignarse directamente en hexadecimal:

```
ULCON0 = 0x3; // 00000011b
```

  - Si algún campo es variable o si queremos mejorar la legibilidad:

```
ULCON0 = (0<<6) | (0<<3) | (0<<2) | (3);
```

Conexión con un terminal

configuración UART0 (ii)

- Modo de transmisión: programado (por pooling)
  - UCON0[1:0] = 1 — Rx: polling/interrupt mode
  - UCON0[3:2] = 1 — Tx: polling/interrupt mode
  - UCON0[4] = 0 — no break
  - UCON0[5] = 0 — no loopback
- Interrupciones por eventos: desactivadas
  - UCON0[6] = 0 — por error
  - UCON0[7] = 0 — por Rx timeout
- Tx/Rx FIFO: activadas
  - UFCON0[0] = 1
  - UFCON0[1] = 0 — Rx FIFO normal (no reset)
  - UFCON0[2] = 0 — Rx FIFO normal (no reset)

- Resumen:
  - UFCON0 = 0x1 (XXXX.X001)
  - UMCON0 = 0x0 (0000.000X)
  - ULCON0 = 0x3 (X000.0011)
  - UBRDIV0 = 0x22 (34)
  - UCON0 = 0x5 (XX.0000.0101)

J.M. Mendías
2016

Conexión con un terminal

acceso a datos transmitidos por UART0

- Para recibir un carácter por polling:
  - Esperar mientras Rx FIFO esté vacía (esperar mientras UFSTAT0[3:0] == 0)
    - De no usar FIFO, habría que esperar mientras el Rx Buffer esté vacío (UTRSTAT0[0] == 0)
  - Leer URXH0

- Para enviar un carácter por polling :
  - Esperar mientras Tx FIFO esté llena (esperar mientras UFSTAT0[9] == 1)
    - Alternativamente podría esperar mientras UFSTAT0[7:4] == 15
    - De no usar FIFO, habría que esperar mientras el Tx Buffer esté lleno (UTRSTAT0[1] == 0)
  - Escribir en UTXH0

- Adicionalmente se puede ignorar el contenido de:
  - UMSTAT0, porque no se hace control de flujo
  - UERSTAT0, porque no se hace gestión de errores de transmisión
  - UTRSTAT, porque dado las Tx/Rx FIFO están activadas se usa UFSTAT0 en su lugar

Driver de la UART-0

uart.h

```
#ifndef __UART_H__
#define __UART_H__

#include <common_types.h>

void uart0_init( void );
```
Inicializa la UART0. Sin argumentos por tener características fijas.

```
void uart0_putchar( char ch );
```
Envía un carácter

```
char uart0_getchar( void );
```
Espera la recepción de un carácter y lo devuelve

```
void uart0_puts( char *s );
```
Envía una cadena de caracteres

```
void uart0_gets( char *s );
```
Espera la recepción de una cadena y la almacena.

```
void uart0_putint( int32 i );
```
*Envía un **entero con signo** como una cadena de **dígitos decimales***

```
void uart0_puthex( uint32 i );
```
*Envía un **entero sin signo** como una cadena de **dígitos hexadecimales***

```
int32 uart0_getint( void );
```
*Espera la recepción de una cadena de **dígitos decimales** que interpreta como un **entero con signo** que devuelve*

```
uint32 uart0_gethex( void );
```
*Espera la recepción de una cadena de **dígitos hexadecimales** que interpreta como un **entero sin signo** que devuelve*

```
#endif
```

Driver de la UART-0

uart.c

- Para enviar una cadena de caracteres:
  - Los caracteres se envían de uno en uno hasta alcanzar el centinela de fin de cadena ('\0') que no se envía.

- Para enviar en decimal un entero con signo :
  - Se construye dígito a dígito una cadena de dígitos decimales (caracteres '0'...'9' y '-') que representan al número y después se envía la cadena.
  - Los dígitos que forman la cadena son los restos de sucesivas divisiones enteras por 10
  - Para obtener el carácter correspondiente al dígito basta con sumarle el carácter '0'

- Para enviar en hexadecimal un entero sin signo :
  - Se construye dígito a dígito una cadena de dígitos hexadecimales (caracteres '0'...'9' y 'A'...'F') que representan al número y después se envía la cadena.
  - Los dígitos que forman la cadena son los restos de sucesivas divisiones enteras por 16
  - Para obtener el carácter correspondiente al dígito basta con sumarle el carácter '0' si es 0...9. Si el dígito es A...F (10...15) hay que restarle 10 y sumarle el carácter 'A'.

Driver de la UART-0

uart.c

- Para recibir una cadena de caracteres:
  - Los caracteres se reciben y almacenan de uno en uno hasta detectar el fin de línea LF ('\n') que no se almacena. En su lugar se almacena el centinela fin de cadena ('\0')

- Para recibir en decimal un entero con signo:
  - Se recibe una cadena y después se recorre carácter a carácter acumulando el valor de cada carácter al valor acumulado hasta el momento multiplicando por 10
  - Para obtener el valor de un carácter basta con restarle el carácter '0'
  - El primer carácter puede ser '-' en cuyo caso hay que negar el valor calculado

- Para recibir en hexadecimal un entero sin signo :
  - Se recibe una cadena y después se recorre carácter a carácter acumulando el valor de cada carácter al valor acumulado hasta el momento multiplicando por 16
  - Para obtener el valor de un carácter basta con restarle el carácter '0' si es '0'...'9'. Si el carácter es 'A'...'F' hay que restarle el carácter 'A' y sumarle 10.

Driver de la UART-0

uart.c

```
void uart0_init( void )
{
  UFCON0 = ...;
  UMCON0 = ...;
  ULCON0 = ...;
  UBRDIV0 = ...;
  UCON0 = ...;
}

void uart0_putchar( char ch )
{
  while( ... );
  UTXH0 = ...;
}

char uart0_getchar( void )
{
  while( ... );
  return ...;
}
```

```
void uart0_puthex( uint32 i )
{
  char buf[8 + 1];
  char *p = buf + 8;
  uint8 c;

  *p = '\0';
  do {
    c = i & 0xf;
    if( c < 10 )
      *--p = '0' + c;
    else
      *--p = 'a' + c - 10;
    i = i >> 4;
  } while( i );
  uart0_puts( p );
}
```

Los caracteres se generan comenzando por el menos significativo

Almacena fin de cadena

Resto de la división por 16

Almacenaje del carácter

División por 16

Envía la cadena

Inicialización del sistema

systemLab4.c

- En este lab ampliaremos el número de dispositivos que inicializamos:

```
#include <s3c44b0x.h>
#include "systemLab4.h"

static void port_init( void );

void sys_init( void )
{
  WTCON = 0;  ····· Watchdog deshabilitado
  INTMSK = ~0;  ····· Enmascara todas las interrupciones

  LOCKTIME = ...;  ····· Estabilización del PLL: 512 us
  PLLCON = ...;  ····· Frecuencia del MCLK: 64 MHz
  CLKSLOW = ...;  ····· Frecuencia del  MCLK_SLOW: 500 KHz
  CLCKCON = ...;  ····· Modo de funcionamiento normal y Reloj distribuido a todos lo controladores

  SBUSCON = ...;  ····· Prioridades de bus del sistema fijas: LCD > ZDMA > BDMA > IRQ (por defecto)

  SYSCFG = ...;  ····· Cache deshabilitada

  port_init();  ····· Utilizar la función ya desarrollada en el laboratorio 3
}
```

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/