Softprocessor SP3: Subprograms and Header

July 8, 2011

Abstract

For the embedded computer system from the first softprocessor lecture a little library of usefull functions will be developed and tested.

1 Datatypes, functions and header

1.1 Datatypes

Variables in a C-program have all one type, describing the value range and the representation. Both information have to be declared with the name of a variable before utilization. The configured computer system from the first lecture can only handle integers with or without sign[1]. The types are:

	8 bit	16 bit	32 bit
unsigned	**unsigned char**	**unsigned short**	**unsigned int**
signed	**char**	**short**	**int**

In the following example the variables CtA and CtB are signed 32 bit variables. Variable »z« is one signed 8 bit variable:

```
int CtA, CtB;
unsigned char z;
```

1.2 Functions

A bigger program is splitted in parts for clearness. These parts are in different files. The first partition is a main program, which can be started with »Run« or after »Reset«, and some subprograms, to solve subtasks, called by the main program if needed. A subprogram consists of an interface with calling name and parameter values, and the executable statement sequence In C subprograms are declared as functions with as many as needed calling values and one return value:

[1]For utilization of floating point numbers a floating point unit has to be built in the processor.

```
type function functionname(parameters)
{
 {statement;}
 [return returnvalue;]
}
```

Parameters are input variables, they have local names and types. With the values transferred at the function call the result is calculated by the function. The type of the result must match the type of the function and will be submitted with the return statement. The following function has two signed 32 bit numbers as input parameters, one signed 32 bit signed number as return value. It returns the comparison result of the two input parameters:

```
int function Test(int a, b) return a==b;
```

The main program is a function too. The return value is usually of the type »int« and the return value is the execution status (correct ending or number of error type):

```
int main(parameters){
 {statement;}
 [return executionstatus;]
}
```

In an embedded system the main program is an infinite loop, which never ends itself. So there is no return value also. Functions with no return value will be defined as »void[2]«.

1.3 Header

Bigger programs will be separated in translation units. Connected functions will be combined in source code files. Definitions of function calls from source code files and also the definitions of project comprehensive used constants and macros belongs to the corresponding header file. The header file can be referenced by integration of the source code. Header files form an interface between units. In program libraries header files are the visible part, while the rest is pretranslated, so the source code is not visible.

In excercise 3.1 a little library »utils.c« with usefull subprograms and macros for character processing shall be written. The header file »utils.h« therefore is:

```
#ifndef UTILS_H
#define UTILS_H
#include <xparameters.h>  // hardware addresses
#include <xuartlite_l.h>  // driver for the UART
```

[2] keyword for »no type«

```
#include <xgpio_l.h>      // driver for the parallel interface
// macro to receive a character via UART
#define getchar() \
              XUartLite_RecvByte(XPAR_XPS_UARTLITE_0_BASEADDR)
// macro to send a character via UART
#define putchar(c) \
          XUartLite_SendByte(XPAR_XPS_UARTLITE_0_BASEADDR, c)
// definitions of export functions
int isupper(char c);
int islower(char c);
int isalpha(char c);
int isdigit(char c);
int iscntrl(char c);
int isspace(char c);
int isalnum(char c);
int ispunct(char c);
int isprint(char c);
int toupper(char c);
int tolower(char c);
#endif
```

Statements starting with »#« are for the preprocessor and will be processed at program translation. In the example the preprocessor at first checks if the symbol »UTILS_H« is defined. If it is the case the rest of the file content will be ignored. If not, the preprocessor defines the symbol and processes all lines up to »#endif«. The include statements inserts the content of the three header files »xparameters.h« etc. in the program, which will be translated. Even the header, which will be included, are framed with »#ifndef ... #endif« in the same manner. So no header can be included several times. Both with »#define« defined macros are for easy receiving and sending of a character via UART and were introduced and used in the lecture before. The definitions of the export functions corresponds to the function definitions without the statement sequences. The related source code file must have the exact same calling definition with the exact same type and identifier completed with the statement sequence.

Aufgabe 3.1: Character processing library

Develope the source code file »utils.c« for the header »utils.h«. The functions shall operate as follows:

1. isupper: return of »1«, if »c« is an upper alphabetic character, else »0«.
2. islower: return of »1«, if »c« is a lower alphabetic character, else »0«.
3. isalpha: return of »1«, if »c« is an alphabetic character, else »0«.
4. isdigit: return of »1«, if »c« is a digit, else »0«.

5. iscntrl: return of »1«, if »c« is a control flow character, that means a character with a value less then 0x20 or greater 0x7E, else »0«.
6. isspace: return of »1«, if »c« is a separating character, else »0«. Seperators are space, , the tabulator »\t«, line break »\n«, carriage return »\r«, »\f« and »\v«.
7. isalphanum: return of »1« for true, if »c« is a digit or an alphabetic letter, else »0«.
8. ispunct: return of »1«, if »c« is a printable character, but not a digit and not an alphabetical character, else »0«.
9. isprint: return of »1«, if »c« is a printable character, that means no control flow character, else »0«.
10. toupper: If the character is a lower character, the corresponding upper character shall be given back, else the character itself.
11. tolower: If the character is an upper character, the corresponding lower character shall be given back, else the character itself.

For testing copy the single library functions into the file »main.c« in a new project and make the following adjustments:

- Insert the include statement for »utils.h«.
- Delete the macro definitions in »main.c«, they are now in »utils.h«, to avoid multiple definitions.
- In the infinite loop between receive and transmit the value represented by the switches shall be read and one the corresponding subprogram shall be called. For the first nine functions return via UART character »0« if the result value is »0« and character »1« if the result value is »1«. By the last two functions after every function call the result value shall be send back.

Test example:

position of switches:	1 (isupper)	...	10 (toupper)
Input:	`aBc2011␣Hallo`	...	`aBc2011␣Hallo`
Output:	`0100000010000`	...	`ABC2011␣HALLO`

Aufgabe 3.2: Input and ouput of number values

1. Insert in »utils.h« and »utils.c« the function

```
unsigned short term_read_unsigned();
```

which reads digits from the UART followed by a separating character. After receiving, the number value shall be calculated with this recursion formula:

$$\begin{aligned} W_0 &= 0 \\ W_{n+1} &= 10 * W_n + z \end{aligned}$$

(z – value of received digit; W_n – number value after receiving n digits). Separating characters before the first digit shall be ignored. If the digit sequence is interrupted by an unproper character or the value of the number is greater than biggest representable number value, the biggest representable number value 0xFFFF shall be returned.

2. Insert in »utils.h« and »utils.c« another function

   ```
   void term_write_unsigned(unsigned z);
   ```

   for translation of the number value »z« in an alphabetic character sequence. The sequence shall be send characterwise. The algorithm contains two loops. In the first loop in ´the variable »x« the biggest decimal power less than the number value »z« is searched. In the second loop the following will be repeated while $x > 0$

   - the digit value »c« as integer quotient from number value and place value (decimal power) shall be calculated
     $$c = z/x$$
   - from the number value the product of number value and place value will be subtracted
     $$z = z - c \cdot x$$
   - the place value will be divided trough ten
     $$x = x/10$$
   - the digit value will be changed in the character value of the digit and sent via UART.

   After the digit sequence a space character shall be sent.

3. For testing change the statement sequence in the infinite loop of the main program so that it is waiting inside for two digit sequences ending with a separating character. The sum of the two digit sequences followed by a space character shall be send back via UART.

Test example:

Input:	137␣344␣1647␣56
Output:	481␣1703␣