Coding Conventions of TreeOS
2017-07

Preface:

The specification of programming can make the program more concise, neat, easy to read、understand and transplant that improve the portability, compatibility, easy maintenance, etc., improving the programming efficiency.

Of course, if there are too many rules to remember, it is a burden to the programmer.So make the specification as lean as possible we need to grab the key.

The code for programming runs throughout.TreeOS is open source software, and software that changes or extends it must also comply with the same programming specifications.

Below are the programming specifications for the TreeOS ComLib.

Ⅰ Basic principle

- all code must adopt the ANSI C standard
- strictly abide by the programming norms
- as far as possible, use simple grammar
- avoid complex statements
- do not use obscure expression
- do not use GOTO statements
- modify code, need to update related documents

Ⅱ Source code file

We have a certain function of source code files (such as a device driver) is called a module, it includes two files: executable file (. C) and header files (.h files).

Layout of the execution file:

- head of file
- include file (# include)
- global macros (Global macro definition)

Includes: conditional compilation selection, attribute (constant definition), IO pin definition, special statement.

- external declaration for global variable
- function declarations (the first part is the local function prototype,the last part is the global function prototype)

note: header files must make the pledge that we shall avoid repetition defines a numeric value in the include file .

Ⅲ File header

The file header is a annotate block placed on the head of the source code file, which includes copyright, company information, programmer, and the description information for text, software version and the history of revision, etc.

When every time the software is modified,a corresponding revision record should be made.

For example:

/***************** (C) COPYRIGHT 2012 BeiJing Guanglun Electronic Technology Co., LTD.******************/

The copyright of this software is owned by Beijing Guanglun Electronic Technology co., LTD.

All the programs offered by this software are for the purpose of learning and understanding the TreeOS operating system.

If you put the software on youself reference to your product, any adverse consequences caused by the software directly or indirectly after running, the software copyright owner will not do any compensation , also does not assume any legal liability.

The ultimate resolution of the software 、 copyright and parsing right is owned by the software author.

For details, please visit **http://www.treeos.com.**

If you have any questions or Suggestions, please email us at support@ treeos.com .

For the latest technical progress, please pay attention to WeChat public's "**Guanglun**

Electronic" or "**TreeOS**", a taobao store.

```
******************************************************************************
    Name for File：TreeOS_main.h
    TreeOS: this software is applicable to the real-time operating system of TreeOS.
    Configuration: default configuration for TreeOS Kepler11 development board.
    Version: belonging to the ComLib A2 component library, 2014
    Function:
    * treeos_main.c a configuration file
    Revision record:
    1. Completion date: 2014-10-1;By TreeOS Mrs Lin
******************************************************************************/
```

Iv The document contains

TreeOS requires that C files for each module must be equipped with a header file of the same name.

Module C file can list the header file only, you can also use a header file named **TreeOS_inc. h** including all files, although convenient (all module contain **TreeOS_inc.h** only), but can make the compilation time longer, and as TreeOS ComLib constantly expanding, the **TreeOS_inc.h** also need configuration based on the requirements.In addition, if the module contains header files required only, it can be intuitively seen that the module is coupled to other modules.

Such as:

```
#include<reg52.h> //M/
#include "TreeOS_main.h"
#include "TreeOS_mcu.h"
 #include "TreeOS_int.h"
 #include "TreeOS_hc574.h"
#include "TreeOS_hd61202.h"
#include "TreeOS_lcd_disp.h"

#include "TreeOS_I2C.h"
#include "TreeOS_24Cxx.h"
#include "TreeOS_pcf8563.h"

#include "TreeOS_RS232.h"
#include "TreeOS_beep.h"
#include "TreeOS_keyboard.h"
```

Ⅴ **Function declaration**

All the defined functions in the module are declared in the header file.

There are two parts, the first part is the local function prototype, the latter part is the global function prototype.

You need to call a function defined in another module, and you must include the header file of that module.

Ⅵ **comments**

The comment is written behind the statement and is gused / / enerally .

It's so long that write in multiple lines.The reason for the use / / is that writing is simple, and the second is that it is sometimes convenient to use/* */ to comment out a program (more intuitive than using #if and #endif).

Of course, you don't reject the use of annotations/*..*/, but the same annotation style is required for the same module (corresponding.c and.h files).

Ⅶ Indent format

Remember that the "{}" the function is near the left edge, and the statement is indented four Spaces relative to the left side of the column.

For a statement such as if, for, while, switch, its statement block “{” and “}” are not indented, and one row is placed separately, and the pair is in the same column.This makes it easy to distinguish between numerous {}.The statement inside {} is indented and the {} is exposed.

Such as:

```
if ((t - t0)  >  500)  //500ms  timekeeping
{
    t0 = t;
    for ( i  =  0;  i  <  5;  i++)
    {
        dsplay_chinese(hao+i*32,  0);
        } //for
 } //if
```

For case statements, indent 4 Spaces exposing the case .

Such as：

```
case UP:
    if (item)


    {
      ……
    }
    break;
case DOWN:
    if (item  <  5)
    {
      ……
    }
    break;
……
```

Ⅷ Naming rules

- **macro definition**

The macro definition is written in uppercase, dividing the words with an underline.Special circumstances allow the use of a lowercase letter and only the front, mainly to avoid conflict with other imported programs.

A macro defines a prefix (beginning) as part of the file name of the file in which it resides.

Such as:

```
#define UART_BUFFER_SIZE_SEND      20        //M/  send buffer bytes.
```

Note: a space between #define and macro name is ok.

- **the variable name and function name**

Try to as clearly as to make sense of it.There are two ways of naming it: one is Capitalizing the first letter of the word, so as to separate different words;The second is that all the words are in lower case and the middle is separated by an underline"_".Both methods are available, and the second method, though much more "_", is more conducive to reading.

The variable name and the function name are prefixed as part of the filename of the file.

Such as:

```
ui8 UartSendPrt = 0;
void lcd_init(void)。
```

Ⅸ Data type

To make it more efficient for all kinds of processor IDE and writing, TreeOS defines all data types with **typedef**, which are put in a dedicated header file, **treeos_typedef.h**.

Some platforms char are equivalent to signed char, others are equivalent to **unsigned char,** and for this purpose, you should explicitly use signed char or **unsigned char** in your code.

You can't use an int directly than using short and long.

Such as:

```
typedef  signed  long  i32;
typedef  signed  short  i16;
typedef signed char    i8;

typedef  code  signed  long  i32c;  //code  data,read  only
typedef  code  signed  short  i16c;  //code  data,read  only
typedef code signed char    i8c;      //code data,read only


         typedef  unsigned  long
ui32;  typedef  unsigned  short
ui16;  typedef  unsigned  char
ui8;

typedef  code  unsigned  long  ui32c;  //code  data,read  only
typedef  code  unsigned  short  ui16c;  //code  data,read  only
typedef code unsigned char    ui8c;      //code data,read only
```

Ⅹ Statements and expressions

- each line only put a statement;
- Such as:

```
                if（a < b）
return 1;
               else
                      return 0;
```

- add a blank lbetween blocks;
- each commas or semicolons should have one space;

  For example: ui8 send_data_24Cxx (ui16 SubAdr, ui8 ByteCnt ui8 * p)

  for (i = 0; i < 100; i++)

Note: the parentheses after the function name cannot have a blank ;Do not have blank between the () and the characters in parentheses.

- one yuan can not contain Spaces between operators and operands;
- For example：!a, ~b, i++, i--, *p, &x, (ui16)y etc.
- between binary and multiple operators or operands should have at least one space;

  For example：a + b; a = b; a | b; a > b; a >= b; etc.

some keywords to follow behind a space character

For example：for (i = 0; i < 100; i++)，if (a > b)，else {⋯ }，switch (a)，return (a) etc.

Ⅺ Structure and common body

The structure type is represented in capital letters (with 1 lower case), and the rules are defined by the macro.

Such as:

```
typedef struct        //define  hh:mm
{
    ui8 hour;         //0~23
    ui8 min;          //0~59
} tHHMM;
```

Ⅻ Function configuration

TreeOS is a configurable, tailored, componised operating system.
For specific applications, you need to configure the capabilities (in.h files), and even modify some of the software (in.c files).
All of these need to be modified with "//M/" (M means Modify) to remind users that they may need to be modified here.
Such as:

```
#define UART_BUFFER_SIZE_SEND    20    //M/  Send buffer bytes.
```

//M/Alert to modify the size of the serial port to send the buffer.
Too small to fit, too big to waste memory.