Embedded System Design

Chapter 3: C Programming for ARM Microcontroller

1. C Program Basics
2. ARM Cortex-M C Compiler
3. ARM software library
1. Basic Programming for Embedded C

- Simple structure for embedded C program

```c
#include <...> ; // Library declaration

int x, y, z; // Global variables

void function1 () { } // Function declaration
void function2() { }

void main() // main program
{
    int i, j, k; // Local variable
    ...
    // Initialization

    while (1) // main process, loop forever
    {
    }
}
```
Example program

```c
#include "inc/lm4f120h5qr.h"
//************************** Blinky LED **********************************
int main(void) {
    volatile unsigned long ulLoop;
    SYSCTL_RCGC2_R = SYSC TL_RCGC2_GPIOF; // Enable the GPIO port
    ulLoop = SYSCTL_RCGC2_R; // Do a dummy read to insert a few cycles
    GPIO_PORTF_DIR_R = 0x08; // Set the direction as output
    GPIO_PORTF_DEN_R = 0x08; // Enable the GPIO pin for digital function.
    while(1) // Loop forever
    { GPIO_PORTF_DATA_R |= 0x08; // Turn on the LED.
      for(ulLoop = 0; ulLoop < 200000; ulLoop++) { }  
      GPIO_PORTF_DATA_R &= ~(0x08); // Turn off the LED
      for(ulLoop = 0; ulLoop < 200000; ulLoop++) { }  
    }
}
```
## Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (bits)</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>8</td>
<td>0 ÷ 255</td>
</tr>
<tr>
<td>unsigned short int</td>
<td>8</td>
<td>0 ÷ 255</td>
</tr>
<tr>
<td>unsigned int</td>
<td>16</td>
<td>0 ÷ 65535</td>
</tr>
<tr>
<td>unsigned long int</td>
<td>32</td>
<td>0 ÷ 4294967295</td>
</tr>
<tr>
<td>signed char</td>
<td>8</td>
<td>-128 ÷ 127</td>
</tr>
<tr>
<td>signed short int</td>
<td>8</td>
<td>-128 ÷ 127</td>
</tr>
<tr>
<td>signed int</td>
<td>16</td>
<td>-32768 ÷ 32767</td>
</tr>
<tr>
<td>signed long int</td>
<td>32</td>
<td>-2147483648 ÷ 2147483647</td>
</tr>
<tr>
<td>float</td>
<td>32</td>
<td>±1.17549435082E-38 ÷ ±6.80564774407E38</td>
</tr>
<tr>
<td>double</td>
<td>32</td>
<td>±1.17549435082E-38 ÷ ±6.80564774407E38</td>
</tr>
<tr>
<td>long double</td>
<td>32</td>
<td>±1.17549435082E-38 ÷ ±6.80564774407E38</td>
</tr>
<tr>
<td>No.</td>
<td>Keyword</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>asm</td>
<td>Insert assembly code</td>
</tr>
<tr>
<td>2</td>
<td>auto</td>
<td>Specifies a variable as automatic (created on the stack)</td>
</tr>
<tr>
<td>3</td>
<td>break</td>
<td>Causes the program control structure to finish</td>
</tr>
<tr>
<td>4</td>
<td>case</td>
<td>One possibility within a switch statement</td>
</tr>
<tr>
<td>5</td>
<td>char</td>
<td>8 bit integer</td>
</tr>
<tr>
<td>6</td>
<td>const</td>
<td>Defines parameter as constant in ROM</td>
</tr>
<tr>
<td>7</td>
<td>continue</td>
<td>Causes the program to go to beginning of loop</td>
</tr>
<tr>
<td>8</td>
<td>default</td>
<td>Used in switch statement for all other cases</td>
</tr>
<tr>
<td>9</td>
<td>do</td>
<td>Used for creating program loops</td>
</tr>
<tr>
<td>10</td>
<td>double</td>
<td>Specifies variable as double precision floating point</td>
</tr>
</tbody>
</table>
### Keywords for Embedded C (2)

<table>
<thead>
<tr>
<th>No.</th>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>else</td>
<td>Alternative part of a conditional</td>
</tr>
<tr>
<td>12</td>
<td>extern</td>
<td>Defined in another module</td>
</tr>
<tr>
<td>13</td>
<td>float</td>
<td>Specifies variable as single precision floating point</td>
</tr>
<tr>
<td>14</td>
<td>for</td>
<td>Used for creating program loops</td>
</tr>
<tr>
<td>15</td>
<td>goto</td>
<td>Causes program to jump to specified location</td>
</tr>
<tr>
<td>16</td>
<td>if</td>
<td>Conditional control structure</td>
</tr>
<tr>
<td>17</td>
<td>int</td>
<td>16 bit integer (same as short on the 6811 and 6812)</td>
</tr>
<tr>
<td>18</td>
<td>long</td>
<td>32 bit integer</td>
</tr>
<tr>
<td>19</td>
<td>register</td>
<td>Specifies how to implement a local</td>
</tr>
<tr>
<td>20</td>
<td>return</td>
<td>Leave function</td>
</tr>
</tbody>
</table>
## Keywords for Embedded C (3)

<table>
<thead>
<tr>
<th>No.</th>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>short</td>
<td>16 bit integer</td>
</tr>
<tr>
<td>22</td>
<td>signed</td>
<td>Specifies variable as signed (default)</td>
</tr>
<tr>
<td>23</td>
<td>sizeof</td>
<td>Built-in function returns the size of an object</td>
</tr>
<tr>
<td>24</td>
<td>static</td>
<td>Stored permanently in memory, accessed locally</td>
</tr>
<tr>
<td>25</td>
<td>struct</td>
<td>Used for creating data structures</td>
</tr>
<tr>
<td>26</td>
<td>switch</td>
<td>Complex conditional control structure</td>
</tr>
<tr>
<td>27</td>
<td>typedef</td>
<td>Used to create new data types</td>
</tr>
<tr>
<td>28</td>
<td>unsigned</td>
<td>Always greater than or equal to zero</td>
</tr>
<tr>
<td>29</td>
<td>void</td>
<td>Used in parameter list to mean no parameter</td>
</tr>
<tr>
<td>30</td>
<td>volatile</td>
<td>Can change implicitly</td>
</tr>
</tbody>
</table>
Scope

• The **scope** of a variable is the portion of the program from which it can be referenced.

• If we declare a local variable with the **same name** as a global object or another local in a superior block, the new variable temporarily supersedes the higher level declarations.

```c
unsigned char x; /* a regular global variable*/
void sub(void){
    x=1;
    { unsigned char x; /* a local variable*/
        x=2;
        { unsigned char x; /* a local variable*/
            x=3;
            PORTA=x;}
        PORTA=x;}
    PORTA=x;}
```
Static Variables

• **Static variables** are defined in RAM permanently.
  
  – **Static global**: can only be accessed within the file where it is defined.
  
  – **Static local**: can only be accessed within the function where it is defined

```c
static short TheGlobal; /* a static global variable*/
void main(void){
    TheGlobal=1000;
}

void main(void){
    static short TheLocal; /* a static local variable*/
    TheLocal=1000;
}
```
Volatile variables

• **Volatile** is a variable that can change value outside the scope of the function

• Applications:
  – memory-mapped peripheral
  – Global variables which can be changed by interrupts
  – Global variables which are access by many tasks

```c
void main(void)
{
    volatile unsigned char *p = (char *) 0x8000;
    while (*p == 0);
}
```
Externals

- Objects that are defined outside of the present source module have the external storage class.
- The compiler knows an external variable by the keyword `extern` that must precede its declaration.
- Only global declarations can be designated extern

```c
extern short ExtGlobal;  /* an external global variable*/
void main(void){
    ExtGlobal=1000;
}
```
Delay in C programming

• Delay techniques:
  – Loop
    • Simple, not precise
  – Timer / Interrupt
    • Complex, precise

```c
void loop_delay()
{
    unsigned int i;
    for(i=0;i<1000;i++);
}
```
De-bouncing

• De-bouncing techniques
  – Hardware
    • Using a capacitor
  – Software
    • Check twice the status of the button
Timeout

• Timeout: solve the problem when it has to be waiting an event for long time.

• Solution
  – Counter loop
  – Timer

```c
long timeout_loop = TIMEOUT_INIT;
...
while(++timeout_loop !=0);
```
2. ARM Cortex-M C Compiler

• Tool chains:
  – Keil™ RealView® Microcontroller Development Kit
  – MentorGraphics Sourcery CodeBench for ARM EABI
  – IAR Embedded Workbench®
  – Texas Instruments Code Composer Studio™

• References
  – Texas Instrument, “ARM Optimizing C/C++ Compiler”
2. ARM Cortex-M C Compiler

- Software development flow for ARM Cortex-M
  - The **compiler** accepts C/C++ source code and produces ARM assembly language source code
  - The **assembler** translates assembly language source files into machine language
2. ARM Cortex-M C Compiler

- The **linker** combines relocatable object files into a single absolute executable object file.
- The **archiver** allows you to collect a group of files into a single archive file, called a library.
2. ARM Cortex-M C Compiler

- armcl [options] [filenames] [--run_linker [link_options] object files]]
  - **Armcl**: Command that runs the compiler and the assembler.
  - **Options**: Options that affect the way the compiler processes input files.
  - **Filenames**: One or more C/C++ source files, assembly language source files, or object files.
  - **--run_linker**: Option that invokes the linker. The **--run_linker** option's short form is -z.
  - **link_options**: Options that control the linking process.
  - **object files**: Name of the additional object files for the linking process.

- Example:
  - armcl symtab.c file.c seek.asm --run_linker --library=lnk.cmd --output_file=myprogram.out
2. ARM Cortex-M C Compiler

• Examples:

  armcl *c ; compiles and links
  armcl --compile_only *.c ; only compiles
  armcl *.c --run_linker Ink.cmd ; compiles and links using a command file
  armcl --compile_only *.c --run_linker lnk.cmd ; only compiles (--compile_only overrides --run_linker)
2. ARM Cortex-M C Compiler

- Invoking the Linker Separately

```
armcl --run_linker {--rom_model | --ram_model} filenames
[options] [--output_file= name.out] --library= library [lnk.cmd]
```

Example:
- `armcl --run_linker --rom_model prog1 prog2 prog3 --output_file=prog.out --library=rtsv4_A_be_eabi.lib`
3. ARM Software Library

- TI’s ARM Cortex-M microcontroller
  - StellarisWare
  - TivaWare

- ST’s ARM Cortex-M microcontroller
  - ST8 firmware
  - STM32 firmware

- Documents
StellarisWare

- an **extensive suite of software** designed to simplify and speed development of Stellaris-based microcontroller applications
- operates with **all LM3S and LM4F series Stellaris MCUs**
- StellarisWare software includes:
  - Stellaris Peripheral Driver Library
  - Stellaris Graphics Library
  - Stellaris USB Library
  - Stellaris Code Examples
TivaWare

• **On 15 Apr. 2012**, TI recommends that new design should use TivaWare and Tiva family MCUs

• TivaWare for C Series library includes:
  – TivaWare Peripheral Driver Library
  – TivaWare Graphics Library
  – TivaWare USB Library
  – TivaWare IQMath Library
• **GPIO driver:**
  – Driverlib/gpio.c
  – Driverlib/gpio.h

• **Most useful functions:**
  – long `GPIOPinRead`(unsigned long ulPort, unsigned char ucPins)
  – void `GPIOPinWrite`(unsigned long ulPort, unsigned char ucPins, unsigned char ucVal)

• **Examples:**
  – `X = GPIOPINRead(GPIO_PORTF_BASE, GPIO_PIN_0);`
  – `GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3 | GPIO_PIN_2, 7);`
GPIO

• void **GPIODirModeSet**(unsigned long ulPort, unsigned char ucPins, unsigned long ulPinIO)
  – Description: Sets the direction and mode of the specified pin(s).
  – **ulPort** is the base address of the GPIO port
  – **ucPins** is the bit-packed representation of the pin(s).
  – **ulPinIO** is the pin direction and/or mode.
    • **GPIO_DIR_MODE_IN**: software controlled input
    • **GPIO_DIR_MODE_OUT**: software controlled output
    • **GPIO_DIR_MODE_HW**: under hardware control
System Clock

• void **SysCtlClockSet** (unsigned long ulConfig)
  – This function configures the clocking of the device

• ulConfig:
  – **Clock divider**: SYSCTL_SYSDIV_1, SYSCTL_SYSDIV_2, ...
    SYSCTL_SYSDIV_64
  – **Use of PLL**: SYSCTL_USE_PLL, SYSCTL_USE_OSC
  – **External crystal frequency**: SYSCTL_XTAL_1MHZ,
    SYSCTL_XTAL_4MHZ, SYSCTL_XTAL_8MHZ,
  – **Oscillator source**: SYSCTL_OSC_MAIN, SYSCTL_OSC_INT

• Examples:
  – SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);
System Clock

• unsigned long `SysCtlClockGet(void)`
  – return the processor clock rate

• void `SysCtlDelay(unsigned long ulCount)`
  – Provides a small delay.
  – `ulCount` is the number of delay loop
  – The loop takes 3 cycles/loop

• Example:
  – `SysCtlDelay(SysCtlClockGet() / 10 / 3);`
Class Assignment

The following assignments are applied for MCU LM4F120H5QR

1. Write a program to generate a clock signal 0.5Hz at PD0
2. Write a function to read the status of a button with de-bounced capability.
3. Write a program to control 8 single LEDs at port PB. Each LED is ON alternately from LSB LED to MSB LED.
4. Write a program to control 7-segment LED with the control signal A,B,C,D,E,F,G at port PB0 to PB6. The 7-segment LED shows the counted number form 0 to 9 for every 0.5s.
5. Write a function to read a 4x4 matrix keyboard with 16 buttons using key-scanning method.
STM32F4 Library - GPIO

• **GPIO configuration**

```c
//Declare a variable GPIO_InitStructure

GPIO_InitTypeDef GPIO_InitStructure;

Example 1:

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOD, &GPIO_InitStructure);

Example 2:

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```
STM32F4 Library - GPIO

- `uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`
  - `GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_12);`

- `uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx)`
  - `uint16_t D = GPIO_ReadInputData(GPIOD);`

- `uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`
  - `GPIO_ReadOutputDataBit(GPIOD, GPIO_Pin_12);`

- `uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx)`
  - `uint16  X = GPIO_ReadOutputData(GPIOD);`
STM32F4 Library - GPIO

- void **GPIO_SetBits**(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
  - GPIO_SetBits(GPIOD, GPIO_Pin_12);
  - GPIO_SetBits(GPIOB, GPIO_Pin_13);

- void **GPIO_ResetBits**(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
  - GPIO_ResetBits(GPIOD, GPIO_Pin_12 | GPIO_Pin_13);

- void **GPIO_Write**(GPIO_TypeDef* GPIOx, uint16_t PortVal)
  - GPIO_Write(GPIOB, 0x000F);

- void **GPIO_ToggleBits**(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
  - GPIO_ToggleBits(GPIOC, GPIO_Pin_1);
STM32F4 Library - Systick

- `uint32_t SysTickConfig (uint32_t ticks)`
  - `SysTick_Config(SystemCoreClock / 1000);`
  - `Delay(100); //delay 100 ms`

```c
void Delay(__IO uint32_t nTime)
{
    TimingDelay = nTime;
    while(TimingDelay != 0);
}

void TimingDelay_Decrement(void)
{
    if (TimingDelay != 0x00)
    {
        TimingDelay--;
    }
}

void SysTick_Handler(void)
{
    TimingDelay_Decrement();
}
```
Assignments

The following assignments are applied for MCU STM32F407VGT6

1. Write a program to generate a clock signal 5Hz at PB0, and a clock signal 10Hz at PB1.

2. Write a program to count a 8bit number and display on 8 single LEDs at port PB[7:0].

3. Write a program described as followings:
   1. There are 3 buttons: START, STOP, MODE
   2. 8 single LEDs are controlled by Port B [7:0]
   3. START: display LEDs according to the MODE
   4. STOP: turn off all the LEDs
   5. MODE: change the Mode for LED display
      1. Mode 1: each LED turns ON from LED0 to LED7
      2. Mode 2: each LED turns ON from LED7 to LED0