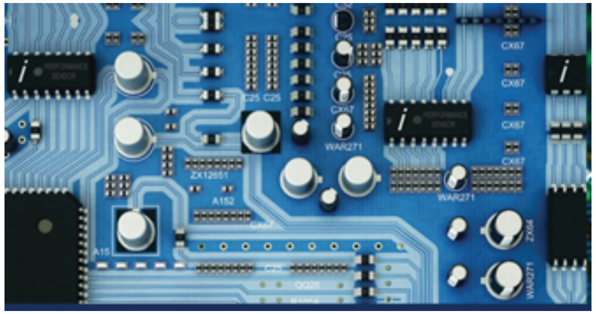
UESTC1008: Microelectronic Systems

Academic year 2019/2020 – Semester 2 – Presentation 3

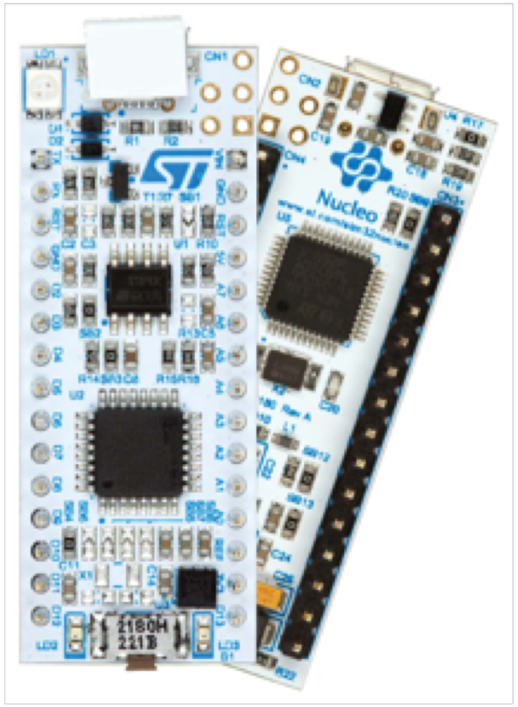
Qammer H. Abbasi, Lei Zhang, Sajjad Hussain, Muhammad Ali Imran and Guodong Zhao
{qammer.abbasi,Lei.zhang,sajjad.Hussain,Muhammad.Imran,guodong.zhao}@glasgow.ac.uk

“A good student never steals or cheats”

Agenda



- Review of previous lecture
- Embedded C
- Programming recap
- Embedded C for mbed
- Summary



Embedded C

- Embedded C is the most popular embedded software language in the world
- Embedded C, even if it’s similar to C, and embedded languages in general requires a different kind of thought process to use
- Embedded systems, like cameras or TV boxes, are simple computers that are designed to perform a single specific task
- They are also designed to be efficient and cheap when performing their task

Embedded C

- Embedded systems are supposed to
  - use a low power to operate, and
  - be as cheap as possible
- As an embedded system programmer, you will have simple hardware to work with
- You will have very little RAM, ROM and very little processing power and stack space
- Your goal is to write programs that are able to leverage this limited processing power for maximum effect
- As an ordinary C programmer, you don’t have as many constraints

Embedded C

- Embedded C lies somewhere between being a high level language and a low level language
- Embedded C, unlike low level assembly languages, is portable
- It can run on a wide variety of processors, regardless of their architecture
- Unlike high level languages, Embedded C requires less resources to run and isn’t as complex
- Some experts estimate that C is 20% more efficient than a modern language like C++
- Another advantage of Embedded C is that it is comparatively easy to debug

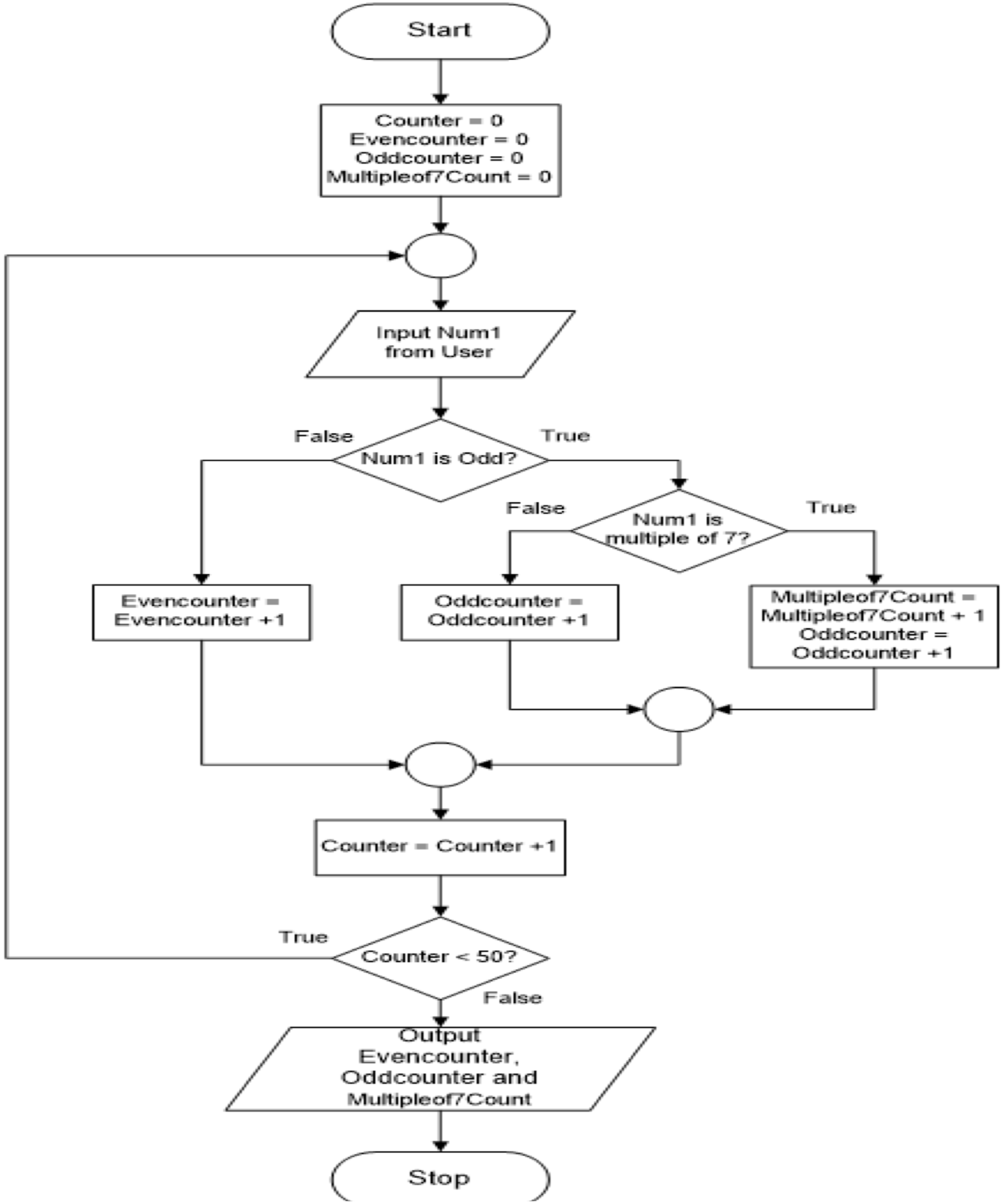
Embedded C

- Another major difference between Embedded C and Regular C is the absence of a convential operating system in Embedded Systems
- When you write a regular C program, you access it from within your operating system software, run it and then, when you’re done, you exit back into your operating system
- With an Embedded C program, you have no operating system to fall back on!
- Your program will, for all intents and purposes, act like the operating system for the embedded device

Revision on Regular C

Survey on sli.do

Identify the C operations in the flow chart



Identify the C operations in the C code

```c
#include <stdio.h>

void main()
{
  int n,  num1;
  int Counter , Evencounter , Oddcounter , Multipleof7Count ;

  Counter = 0;
  Evencounter = 0;
  Oddcounter = 0;
  Multipleof7Count = 0;

  do
  {
     printf("Enter the number = ");
     scanf("%d", &num1);

     if(num1%2 == 1)
     {
        if(num1%7 == 0)
        {
           Multipleof7Count++;
           Oddcounter++;
        }
        else
        {
           Oddcounter++;
        }
     }
     else
     {
        Evencounter++;
     }
     Counter++;
   }while (Counter < 50);

   printf("Evencounter = %d, Oddcounter = %d, MultipleOf7Counter = %d \n",
Evencounter,Oddcounter,Multipleof7Count);

}
```

Embedded C for mbed

Compiler Directive — #define directive

- #define directive
  - Define a *symbolic name* or *symbolic constant* to be a particular string of characters.

For example: #define PI   3.14

In LPC17xx.h file:

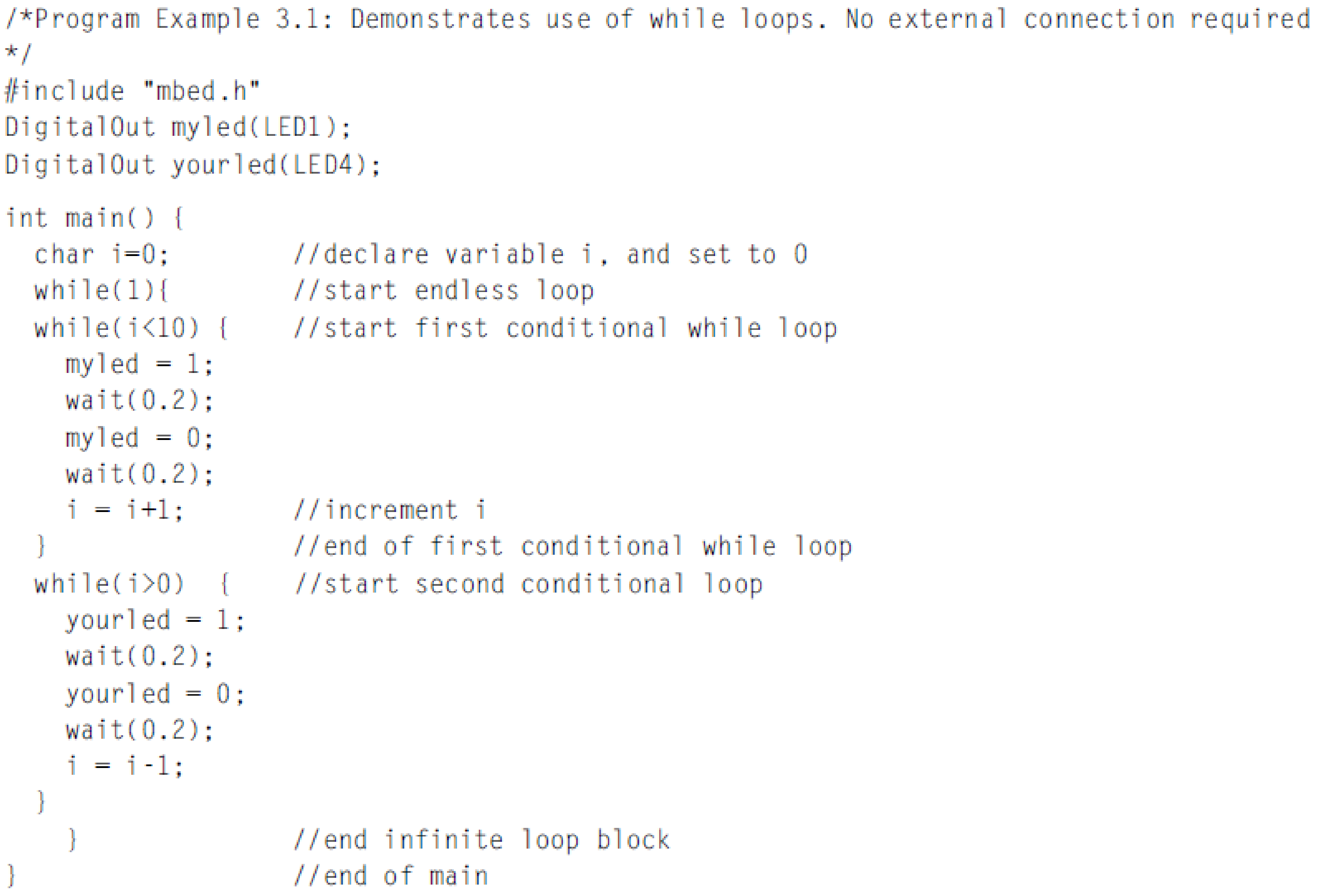
```
typedef struct
{
….
} GPIO_TypeDef;

#define GPIO0_BASE  constant_value
```

```
#define GPIO0 ((GPIO_TypeDef *) GPIO0_BASE)
```

Compiler directives are messages to the compiler Compiler directives all start with a hash, #

An example of mbed Program



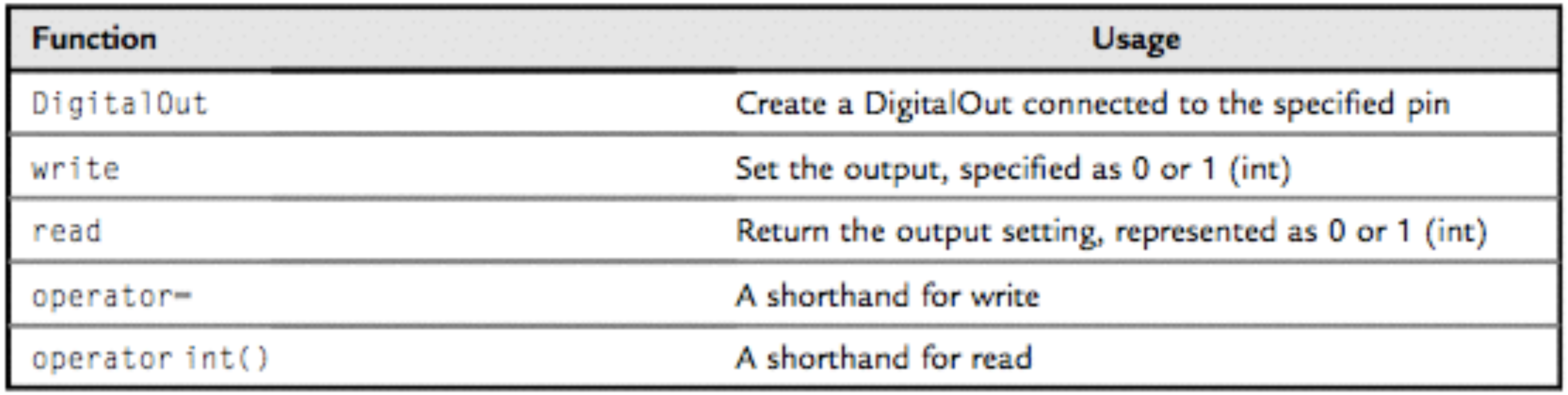
```
/*Program Example 3.1: Demonstrates use of while loops. No external connection required
*/
#include "mbed.h"
DigitalOut myled(LED1);
DigitalOut yourled(LED4);

int main() {
  char i=0;        //declare variable i, and set to 0
  while(1){        //start endless loop
  while(i<10) {    //start first conditional while loop
    myled = 1;
    wait(0.2);
    myled = 0;
    wait(0.2);
    i = i+1;       //increment i
  }                //end of first conditional while loop
  while(i>0)  {    //start second conditional loop
    yourled = 1;
    wait(0.2);
    yourled = 0;
    wait(0.2);
    i = i-1;
  }
    }              //end infinite loop block
}                  //end of main
```

Understanding the mbed API

```
//program variable myled is created, and linked with mbed LED1
DigitalOut myled(LED1);
myled = 1; // this is not a normal = operator
```

*) If you check mbed.h file, you will see this: #include “DigitalOut.h”
*) In DigitalOut.h file, you will know DigitalOut is defined as a class.



Function	Usage
DigitalOut	Create a DigitalOut connected to the specified pin
write	Set the output, specified as 0 or 1 (int)
read	Return the output setting, represented as 0 or 1 (int)
operator=	A shorthand for write
operator int()	A shorthand for read

Member function of Class **DigitalOut**

how to use member functions: e.g., myled.read()

myled = 1 **is the same as** myled.write(1)

mbed Peripheral components are defined as classes.

Running Programs on mbed Board

- Write C code in Keil uVision or online compiler
  - generate .cpp file
- Compile using uVision or online compiler
  - generate .bin file (machine code)
- Download the machine code
  - Copy .bin to mbed board
- Test on the mbed board

API

- An application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software
- In general terms, it's a set of clearly defined methods of communication between various software components
- A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer

(Source: https://en.wikipedia.org/wiki/Application_programming_interface)

mbed API

- API documentation is a quick and concise reference containing what you need to know to use a library or work with a program
- It details functions, classes, return types, and more
- In mbed, API documentation for programs and libraries is fully supported both within the Compiler and in the code listings on the public site
- mbed API Link: https://developer.mbed.org/handbook/API-Documentation#extra-features

SDK

- A Software Development Kit (SDK) is a package of pre-written code that developers can re-use in order to minimize the amount of unique code that they need to develop themselves
- SDKs can help to prevent unnecessary duplication of effort in a development community

mbed SDK

- The mbed Software Development Kit (SDK) is a C/C++ microcontroller software platform relied upon by tens of thousands of developers to build projects fast
- The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects.
- mbed SDK Link: https://developer.mbed.org/handbook/mbed-SDK

Summary

- Programming recap and Embedded C
- Embedded C for mbed
- What will we study in next lecture.