S J P N Trust's

Hirasugar Institute of Technology, Nidasoshi.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi

ECE Dept.
ARM
VI Sem
2017-18

Department of Electronics & Communication Engg.

Course : ARM Microcontroller & ES-15EC62. . **Sem.: 6th (2017-18)**

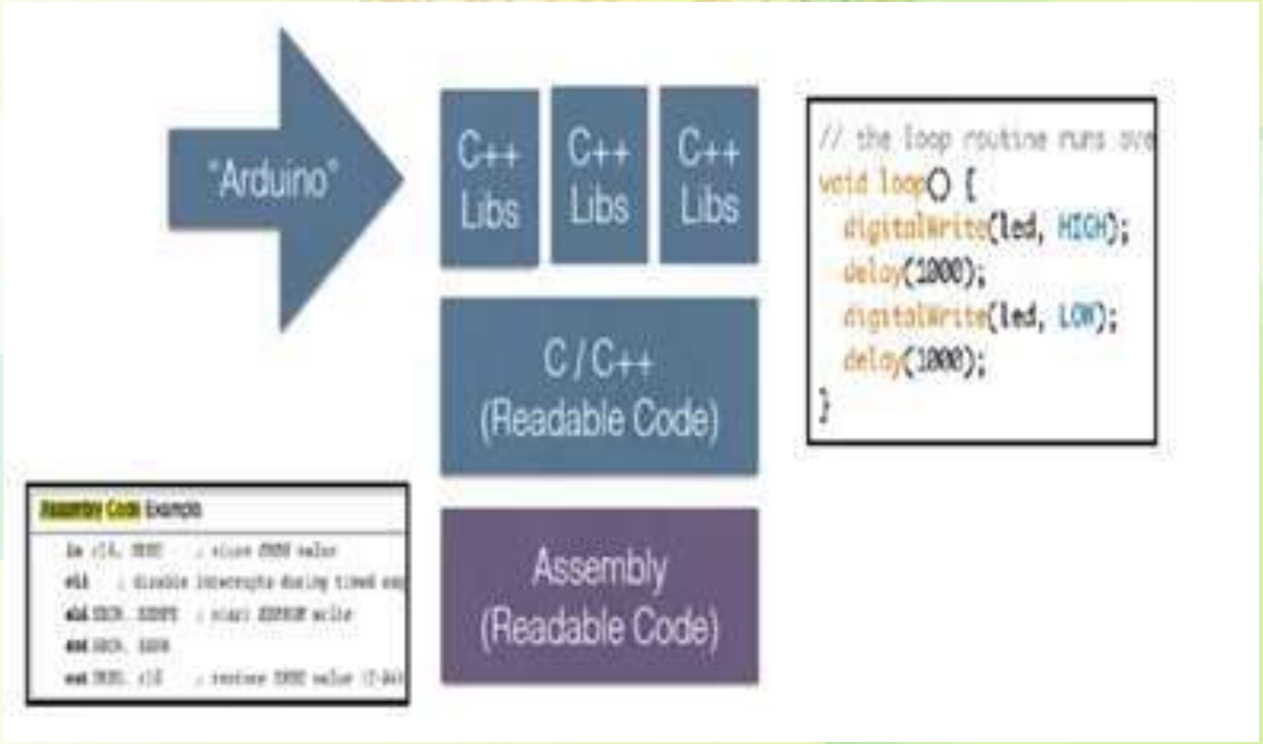
Course Coordinator:

Prof. Sachin S Patil

Embedded Firmware

 Embedded firmware refers to the **control algorithm (Program instructions)** and /or the **configuration settings that an embedded system developer dumps** into the code (program) memory of the embedded system. It is an un-avoidable part of an embedded system. There are various methods available for developing the embedded firmware. They are listed below:

1. Write the program in **high level languages like Embedded C/C++ using an** Integrated Development Environment.
2. Write the program in **Assembly language** using the instructions supported by your application's target processor/controller.



```
// the loop routine runs ove
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

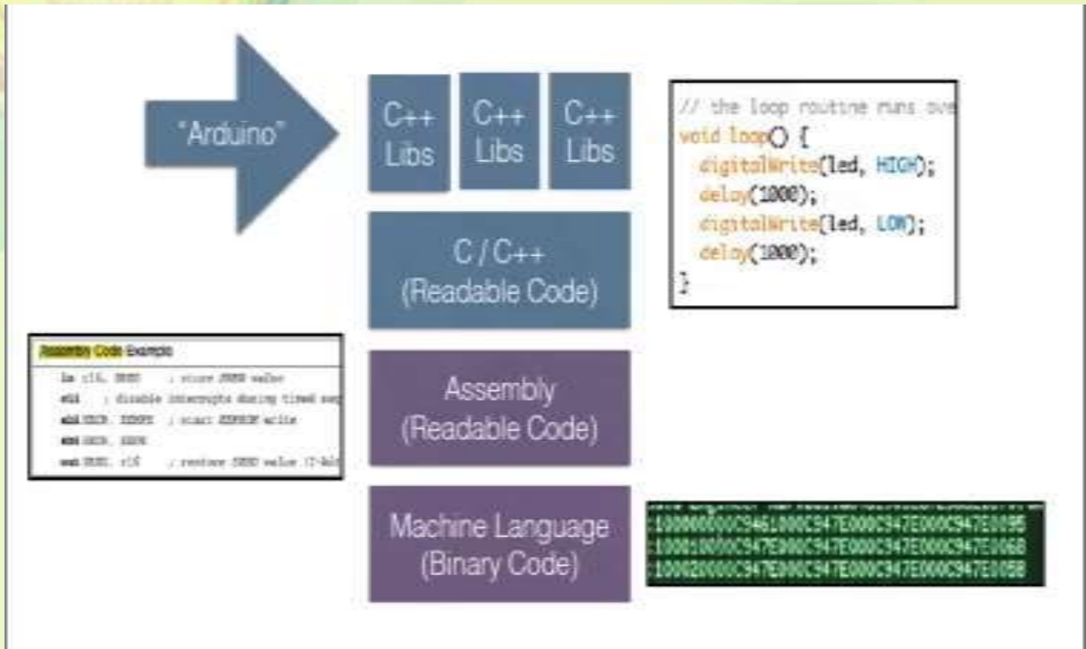
```
in r16, SREG        ; store SREG value
cli                 ; disable interrupts during timed seq
sbi EECR, EEMPE     ; start EEPROM write
sbi EECR, EEPE
out SREG, r16       ; restore SREG value (I-bi
```

 The instruction set for each family of processor/controller is different and the program written in either of the methods given above should be converted into a processor understandable machine code **before loading it into the program memory.**

 The process of converting the program written in either a high level language or processor/controller specific Assembly code to machine readable binary code is called '***HEX File Creation'.***

 If the program is written in Embedded C/C++ using an IDE, **the cross compiler included** in the IDE converts it into corresponding processor/controller understandable '*HEX File'.*

 If you are following the Assembly language based programming technique, you can use the utilities supplied by the processor/controller vendors to convert the source code into '*HEX File'.*



```
// the loop routine runs ove
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

```
in r16, SREG     ; store SREG value
cli     ; disable interrupts during timed seq
sbi EECR, EEMPE  ; start EEPROM write
sbi EECR, EEPE
out SREG, r16    ; restore SREG value (I-bit
```

```
:100000000C9461000C947E000C947E000C947E0095
:100010000C947E000C947E000C947E000C947E0068
:100020000C947E000C947E000C947E000C947E0058
```

Other System Components

 The other system components refer to the components/circuits/Ics which are necessary for the proper functioning of the embedded system.
 Reset Circuit, Brown-out Protection Circuit, Oscillator Unit, Real- Time Clock (RTC), Watchdog Timer are examples of circuits/Ics which are essential for the proper functioning of the processor/controllers.

Reset Circuit

 The reset circuit is essential to ensure that the device is not operating at a voltage level where **the device is not guaranteed to operate, during system power ON. The reset signal brings** the **internal registers and the different hardware systems of the processor/controller to a** known state and starts the firmware execution from the reset vector.
 The reset signal can be either active high (The processor undergoes reset when the reset pin of the processor is at logic high) or active low (The processor undergoes reset when the reset pin of the processor is at logic low). Since the processor operation is synchronized to a clock signal, **the reset pulse should be wide enough to give time for the clock oscillator to** stabilize before the internal reset state starts.
 Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry. Figure illustrates a resistor capacitor based passive reset circuit for active high and low configurations. The reset pulse width can be adjusted by changing the resistance value R and capacitance value C.

RC-Based Reset Circuit

Brown-out Protection Circuit

 The brown-out protection circuit **prevents the processor/controller** from **unexpected program execution behavior when the supply voltage to the processor/controller falls below a specified voltage.**

Oscillator

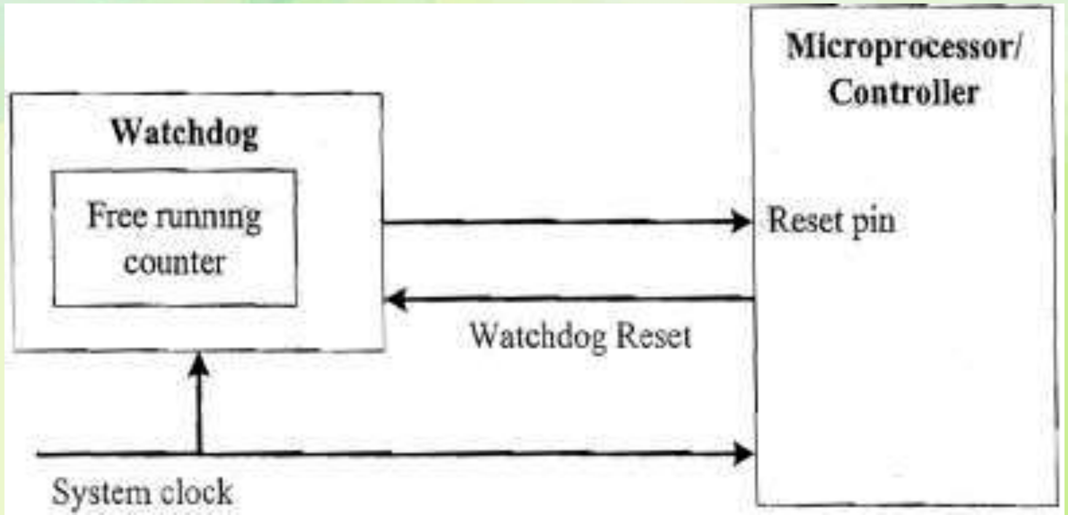
 The Oscillator unit **generates clock signals for synchronizing the operations of the processor.**

Real-Time Clock (RTC)

- **Real-Time Clock (RTC) is a system component responsible for keeping** track of time. RTC holds **information like current time (in hours,** minutes and seconds) in 12 hour/24 hour format, date, month, year, day of the week, etc. and supplies timing reference to the system. RTC is intended to function **even in the absence of power. The RTC chip contains a** microchip for holding the time and date related information and **backup battery cell for functioning in the absence of power, in a single IC** package. The RTC chip is interfaced to the processor or controller of the embedded system. For Operating System based embedded devices, a timing reference is essential **for synchronizing the operations of the OS** kernel.

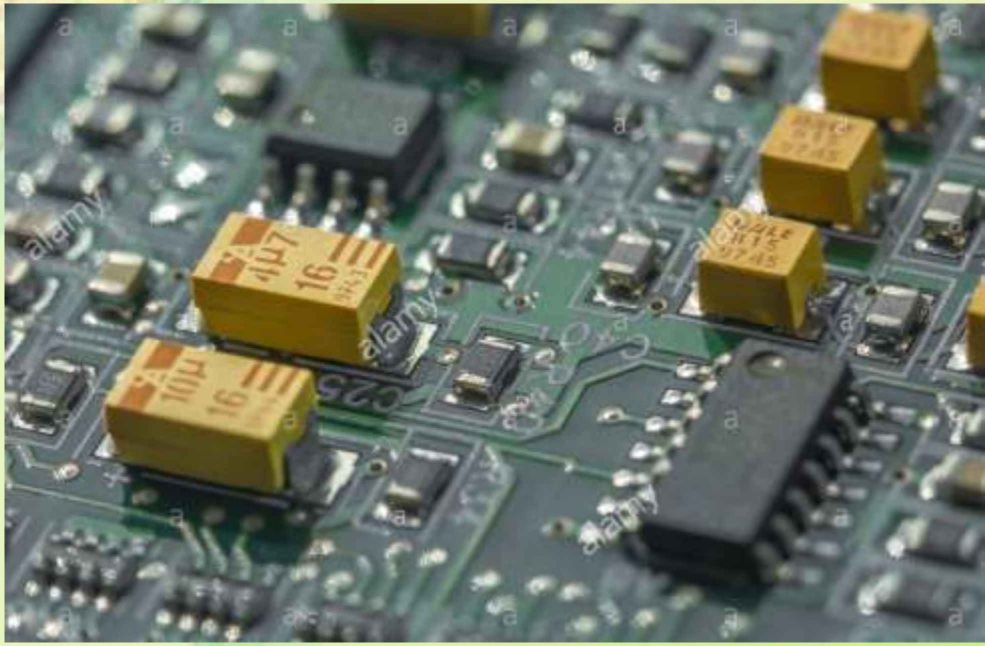
Watchdog Timer

- A watchdog is **to monitor the firmware execution and reset the system** processor/microcontroller when the program execution hangs up or generates an Interrupt in case **the execution time for a task is exceeding the maximum allowed limit.**
- **If the firmware execution doesn’t complete due to malfunctioning, within the time** required by the watchdog to reach the maximum count, the counter will generate a reset pulse and this **will reset the processor (if it is connected to the reset line of the processor).**
- Most of the processors implement watchdog **as a built-in component and provides status** register to control the watchdog timer (like enabling and disabling watchdog functioning) and watchdog timer register for writing the count value. **If the processor/controller doesn’t contain a built in watchdog timer, the same can be** implemented **using an external watchdog timer IC circuit.**



PCB and Passive Components

 Printed Circuit Board (PCB) is the backbone of every embedded system. **After finalizing the components and the inter-connection** among them, a schematic design is created and according to the **schematic PCB is fabricated. PCS acts as a platform for mounting all the necessary components as per the design requirement. Also it acts as a platform for testing your embedded firmware. You can also find some** passive **electronic components like resistor, capacitor, diodes, etc. on your** board. They are the co-workers of various chips contained in your embedded hardware. They are very essential for the proper functioning of your embedded system.



Queries ….?