Microcontroller & Embedded Systems-Module-3-2

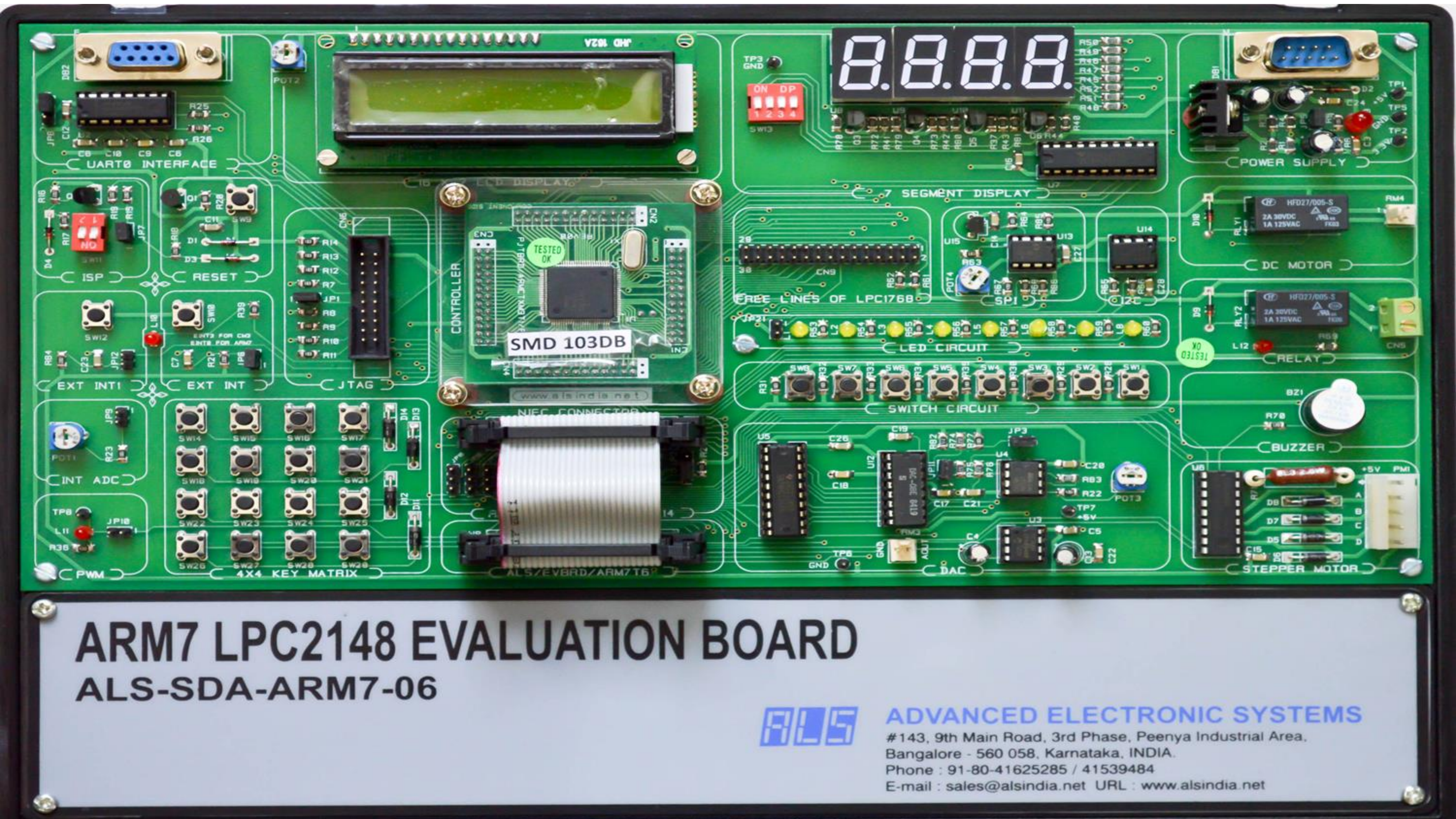
ESTD : 2001
An Institute with a Difference

Dr. Girijamma H A
Professor
Department of Computer Science and Engineering
RNS Institute of Technology
Bangalore -560098
Email: girijakasal@gmail.com
Contact: 9480031494

Module-3-2-Contents

2. The Typical Embedded System

Text book 2: Shibu K V, “Introduction to Embedded Systems”, Tata McGraw Hill Education, Private Limited, 2nd Edition.
Chapter 1(Sections 1.2 to 1.6) ,Chapter 2(Sections 2.1 to 2.6)


UART0 INTERFACE
16 X 2 LCD DISPLAY
7 SEGMENT DISPLAY
POWER SUPPLY
ISP
RESET
JTAG
CONTROLLER
SMD 103DB
TESTED OK
FREE LINES OF LPC1768
SPI
I2C
DC MOTOR
EXT INT1
EXT INT
LED CIRCUIT
RELAY
SWITCH CIRCUIT
INT ADC
BUZZER
PWM
4X4 KEY MATRIX
ALS/EVBRD/ARM7 T6
DAC
STEPPER MOTOR
ARM7 LPC2148 EVALUATION BOARD
ALS-SDA-ARM7-06
ALS
ADVANCED ELECTRONIC SYSTEMS
#143, 9th Main Road, 3rd Phase, Peenya Industrial Area,
Bangalore - 560 058, Karnataka, INDIA.
Phone : 91-80-41625285 / 41539484
E-mail : sales@alsindia.net URL : www.alsindia.net

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 (IDE)

» The IDE will contain a editor, compiler, linker, debugger, simulator, etc.

EMBEDDED FIRMWARE

» IDEs are different for different family of processors/ controllers.

» For example, Keil microvision3 IDE is used for all family member of 8051 microcontroller, since it contains the generic 8051 compiler C51.

2. Write the program in Assembly language using the instructions supported by your application's target processor/ controller.

» 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***'.

EMBEDDED FIRMWARE

- » The methods used for 'HEX File Creation' is different depending on the programming techniques used.
  - » 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'.
  - » Also third party tools are available, which may be of free of cost, for this conversion.

ESTD : 2001
An Institute with a Difference

EMBEDDED FIRMWARE

» For a beginner in the embedded software field, it is strongly recommended to use the high level language based development technique. The reasons for this being:

» Writing codes in a high level language is easy,

» Also the programs written in high level languages are not developer dependent.

» The embedded software development process in assembly language is tedious and time consuming.

OTHER SYSTEM COMPONENTS

- The ***other system components*** refer to the components/ circuits/ ICs which are necessary for the proper functioning of the embedded system.
- Some of these circuits may be essential for the proper functioning of the processor/ controller and firmware execution.
- Watchdog timer, Reset IC (or passive circuit), brown-out protection IC (or passive circuit) etc., are examples of circuits/ ICs which are essential for the proper functioning of the processors/ controllers.

OTHER SYSTEM COMPONENTS- **Reset Circuit**

ESTD : 2001
An Institute with a Difference

» 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 (Normally from vector address 0x0000 for conventional processors/ controllers.

» The reset signal can be either active high or active low.

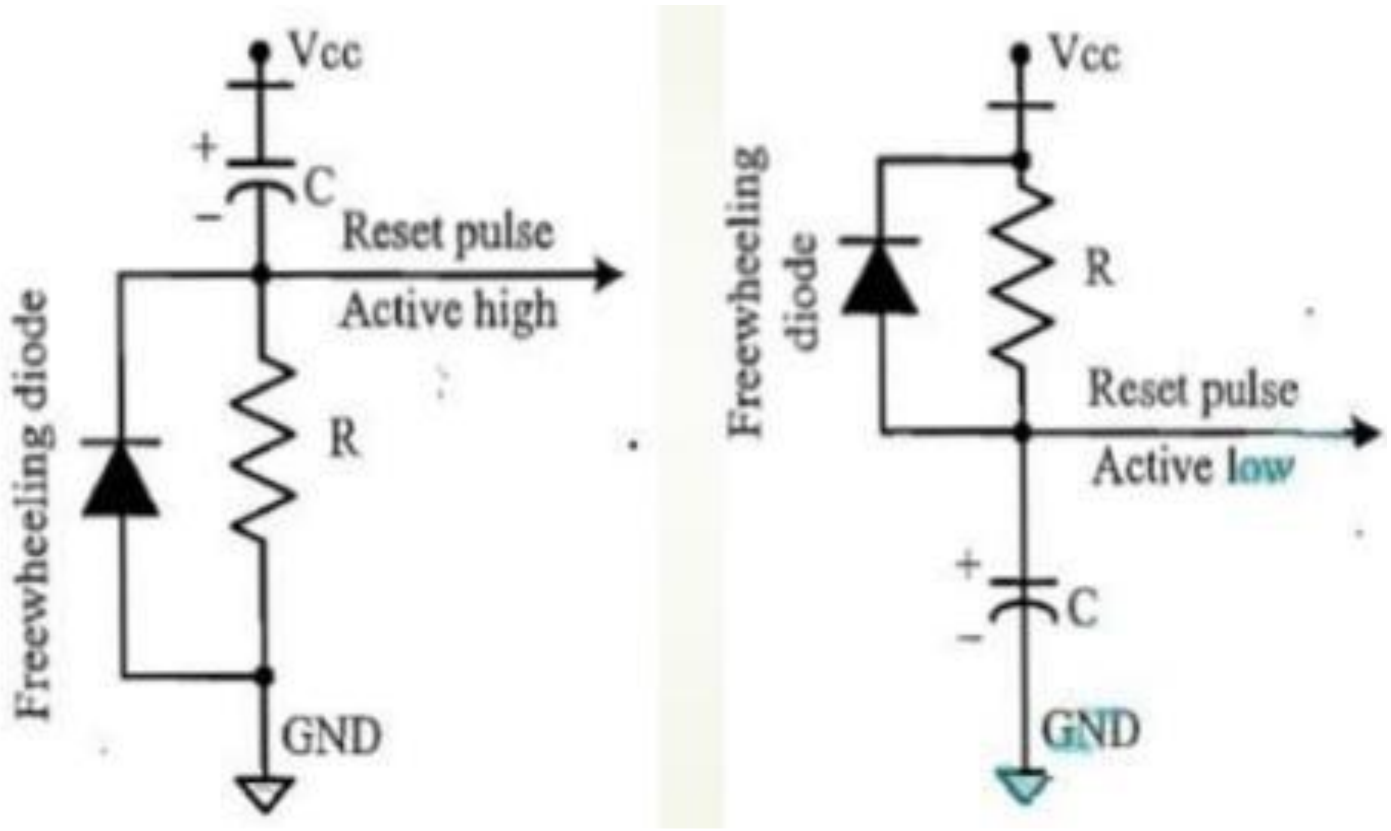
» Since the processor operation is synchronized to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilize before the internal reset state starts.

OTHER SYSTEM COMPONENTS- **Reset Circuit**

- The reset signal to the processor can be applied at power ON through an external passive reset circuit comprising a Capacitor and Resistor or through a standard Reset IC like MAX810 from Maxim Dallas. Select the reset IC based on the type of reset signal and logic level (CMOS/ TTL) supported by the processor/ controller in use.
- Some microprocessors /controllers contain built-in internal reset circuitry and they don't require external reset circuitry.

OTHER SYSTEM COMPONENTS- Reset Circuit

» The following 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*.



OTHER SYSTEM COMPONENTS- Brown-out Protection Circuit

» *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.

» It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold. The processor behavior may not be predictable if the supply voltage falls below the recommended operating voltage. It may lead to situations like data corruption.

» A brown-out protection circuit holds the processor/ controller in reset state, when operating voltage falls below the threshold, until it rises above the threshold voltage.

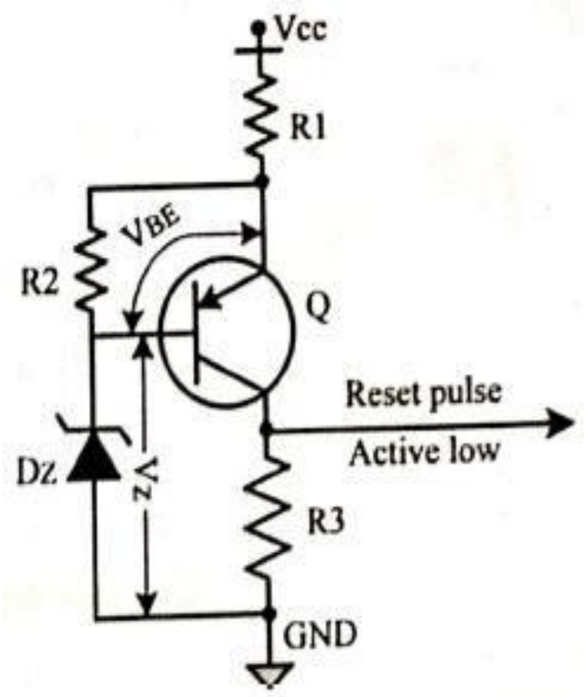
» Certain processors/ controllers support built in brown-out protection circuit which monitors the supply voltage internally.

OTHER SYSTEM COMPONENTS- Brown-out Protection Circuit

» If the processor/ controller don't integrate a built-in brown-out protection circuit, the same can be implemented using external passive circuits or supervisor ICs.

» The following Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/ controller with active low Reset logic.

» The Zener diode, Dz, and transistor, Q, forms the heart of this circuit. The transistor conducts always when the supply voltage V_{CC} is greater than that of the sum of V_{BE} and V_Z (Zener voltage). The transistor stops conducting when the supply voltage falls below the sum of V_{BE} and V_Z. Select the Zener diode with required voltage for setting the low threshold value for V_{CC}.



OTHER SYSTEM COMPONENTS- Oscillator Unit

» ***Oscillator Unit:*** A microprocessor/ microcontroller is a digital device made up of digital combinational and sequential circuits. The instruction execution of a microprocessor/ controller occurs in synchronization with a clock signal. The *oscillator unit* of the embedded system is responsible for generating the precise clock for the processor.

» Certain processors/ controllers integrate a built-in oscillator unit and simply require an external ceramic resonator/ quartz crystal for producing the necessary clock signals. Quartz crystals and ceramic resonators are equivalent in operation, however they possess physical difference.

» Certain devices may not contain built-in oscillator unit and require the clock pulses to be generated and supplied externally.

OTHER SYSTEM COMPONENTS- Oscillator Unit

» The speed of operation of a processor is primarily dependent on the clock frequency. However we cannot increase the clock frequency blindly for increasing the speed of execution. The logical circuits lying inside the processor always have an upper threshold value for the maximum clock at which the system can run, beyond which the system becomes unstable and non functional.

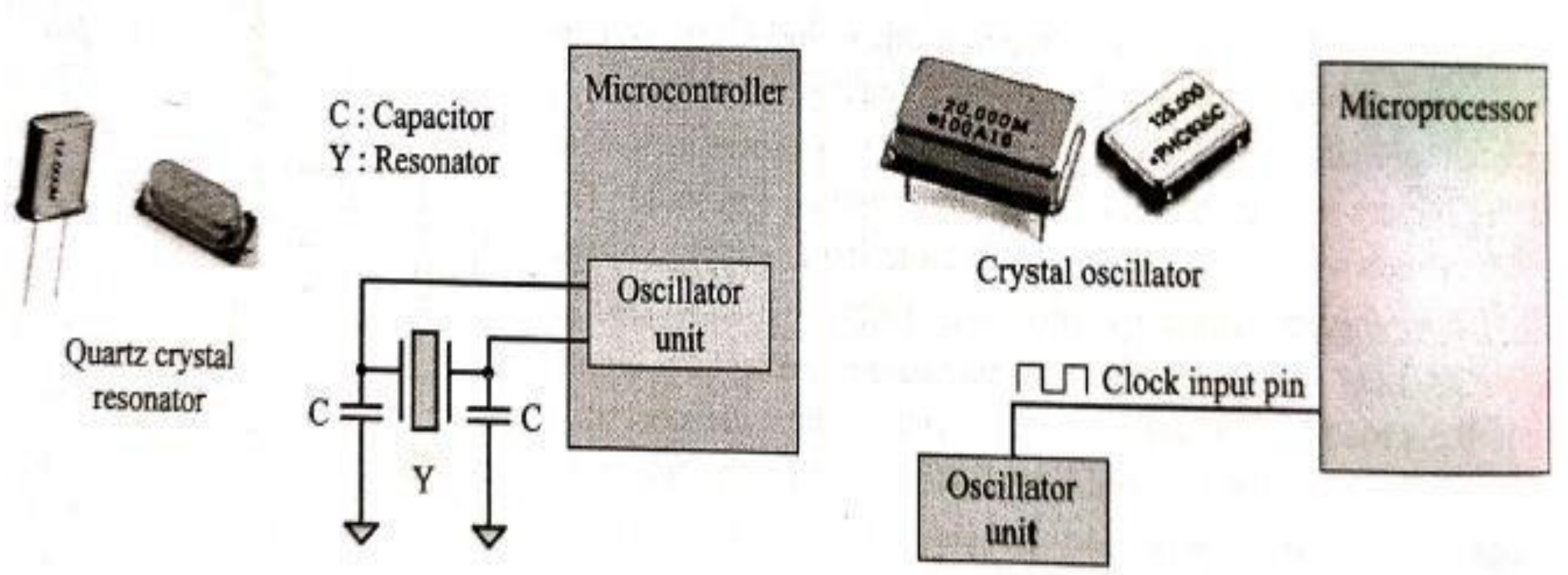
» The total system power consumption is directly proportional to the clock frequency.

The power consumption increases with increase in clock frequency.

» The accuracy of program execution depends on the accuracy of the clock signal.

OTHER SYSTEM COMPONENTS- Oscillator Unit

- The following Figure illustrates the usage of quartz crystal/ ceramic resonator and external oscillator chip for clock generation.



OTHER SYSTEM COMPONENTS- Real-Time Clock

» ***Real-Time Clock (RTC):*** *Real-Time Clock* 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.

» RTCs are available in the form of Integrated Circuits from different semiconductor manufacturers like Maxim/Dallas, ST Microelectronics etc.

» 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.

OTHER SYSTEM COMPONENTS- Real-Time Clock

- » For Operating System based embedded devices, a timing reference is essential for synchronizing the operations of the OS kernel.
- » The RTC can interrupt the OS .kernel by asserting the interrupt line of the processor/controller to which the RTC interrupt line is connected. The OS kernel identifies the interrupt in terms of the Interrupt Request (IRQ) number generated by an interrupt controller. One IRQ can be assigned to the RTC interrupt and the kernel can perform necessary operations like system date time updating, managing software timers etc when an RTC timer tick interrupt occurs.
- » The RTC can be configured to interrupt the processor at predefined intervals or to interrupt the processor when the RTC register reaches a specified value (used as alarm interrupt).

OTHER SYSTEM COMPONENTS- Watchdog Timer

- ***Watchdog Timer:*** In desktop Windows systems, if we feel our application is behaving in an abnormally or if the system hangs up, we have the ***'Ctrl + Alt + Del'*** to come out of the situation. What it happens to embedded system?
- We have a watchdog to monitor the firmware execution and reset the system processor/ microcontroller when the program execution hangs up. A *watchdog timer*, or simply a *watchdog*, is a hardware timer for monitoring the firmware execution.
- Depending on the internal implementation, the watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset the processor if the count reaches zero for a down counting watchdog, or the highest count value for an up counting watchdog.

OTHER SYSTEM COMPONENTS- Watchdog Timer

» If the watchdog counter is in the enabled state, the firmware can write a zero (for up counting watchdog implementation) to it before starting the execution of a piece of code and the watchdog will start counting.

» 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 the firmware execution completes before the expiration of the watchdog, you can reset the count by writing a 0 (for an up counting watchdog timer) to the watchdog timer register.

OTHER SYSTEM COMPONENTS- Watchdog Timer

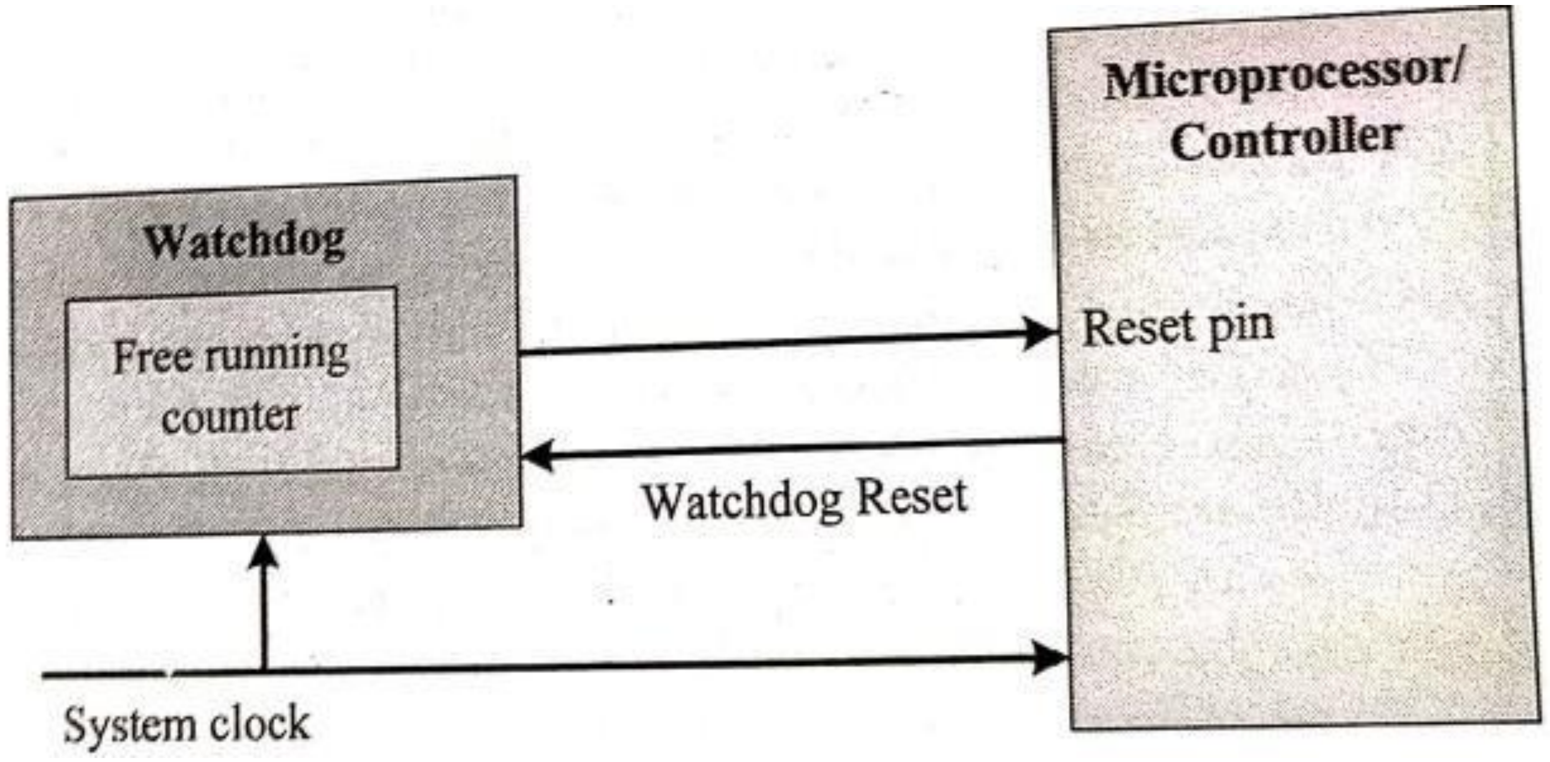
» If the processor/ controller doesn't contain a built in watchdog timer, the same can be implemented using an external watchdog timer IC circuit.

» The external watchdog timer uses hardware logic for enabling/ disabling, resetting the watchdog count, etc., instead of the firmware based 'writing' to the status and watchdog timer register.

» The microprocessor supervisor IC DS 1232 integrates a hardware watchdog timer in it.

OTHER SYSTEM COMPONENTS- Watchdog Timer

» The following Figure illustrates the implementation of the external watchdog timer based microprocessor based supervisor circuit for a small embedded system.



Thank You