Embedded Firmware :-

Unit - III

1) Introduction :-

→ Embedded firmware refers to control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (program) memory of embedded system.

Methods :-

(i) write program in high level languages like embedded C/C++ using an (integrated development environment)

→ The IDE will contain (editor, compiler, linker, debugger, simulator etc)

→ IDE's are different for different family of processors/controllers.

Eg:- Keil microvision3

→ IDE is used for all family members 8051 MC since it contain generic 8051 compiler C51.

(ii) write a program in Assembly programming language using instructions supported by your applications target processor/controller.

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

→ The process of converting program written in either a high level language (or) processor/controller specific.

→ Assembly code to ~~binary~~ ~~code~~ machine readable binary code is called "Hex file converter" (or) "Hex file creation".

→ If program is written in embedded C/C++ using an IDE, the cross compiler included in IDE converters, which corresponds processor/controller understandable 'Hex file'.

Scanned by CamScanner

→ 2nd method:-

following Assembly level based Programming technique.

which can use utilities supplied by processor/controller vendors to convert to source code to Hex file.

→ The beginner in Embedded software field strongly recommends high level language.

The reason for writing Embedded to use high level language is easy, the code written in high level language is highly portable which mean can use same code to run on different MP/MC. with slight modification.

→ The Embedded software development process in Assembly language is tedious and time consuming.

↳ As developer should know instruction sets of MP/MC

— x —

2) Other System Components:-

refers components/circuits IC's. necessary for proper functioning of E.S.

→ Some of circuits are essential for proper functioning of MP/MC and firmware execution.

Eg:- watchdog timer, Reset IC (or) Passive circuit; brown-out protection IC (or) Passive circuit)

→ Some of controllers are [(SOC's) system on chip] integrate these components with a single IC.

Scanned by CamScanner

NPTEL :- E.S

Embedded software :-

Software interacting with physical world
→ It takes time
→ It consumes power
→ It does not terminate

RCLEPTICL
Reliability
Concurrency
Liveliness

Requirements :-

1. Reliability :-
   Human invention is not possible for error handling
2. Cost effectiveness
3. Low power consumption
4. Efficient use of memory
5. Performance requirements
   → timeliness

Timeliness :-

→ Computation takes time.
→ Eg a physical process which with it interacts evolves over time.
→ Satisfying timing constraints
→ Performance gain due to use of elaborate cache schemes, speculative instruction execution, and branch prediction are avoided in MC and DSPs Eg consume system efficiency & reliability.

Scanned by CamScanner

→ Concurrency :-
E.S Engage physical world where multiple things happen at once.
→ E.S need to react to stimulus from a no of and variety of sensors and retain control over actuator
→ Live-ness :-
Program must not terminate (or) block waiting for events that will never occur

Features :-

Interfaces :-
① Components constrained a/c to interface based upon static behavior.
Eg:- speech codel ; cellphone

② Heterogeneity :-
Different computational styles and implementation technologies.
→ interact with events occurring.
→ Irregular in time (alarms, user commands, sensor trigger etc.)
→ Regularly in time (sampled sensor data and actuator control signals)

→ Reactivity
→ Reacts continuously to their environment at rate of environment
→ safety critical.

Scanned by CamScanner

Reset Circuit:-

→ is essential to ensure that device is not operating at voltage level where device is not guaranteed to operate during power ON.

→ The reset signal brings internal registers & the different hardware system of MP/MC to know the state and start firmware execution from reset vector.

→ normally vector address for MP/MC is 0x0000

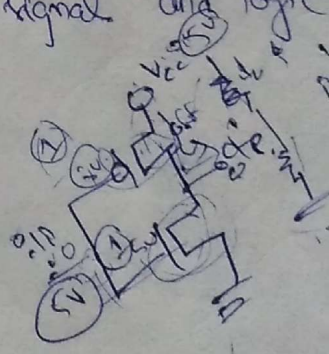
→ The reset vector can be relocated to an address for MP/MC supporting bootloader.

→ The reset signal can be either high/low logic.

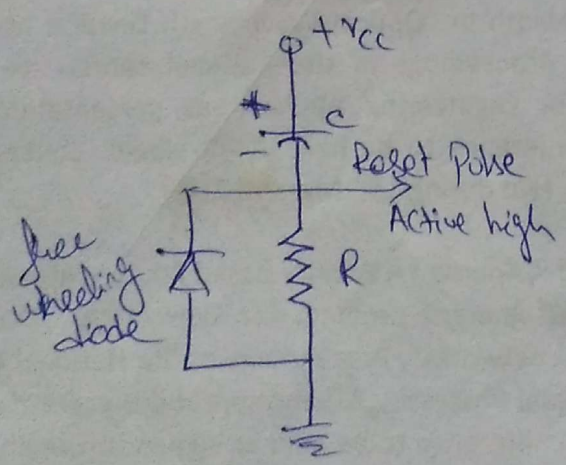
→ Since the processor operation is synchronised to a clock signal, the reset pulse should be wide enough to give time for clock oscillator to stabilize before internal reset ~~pulse~~ starts.

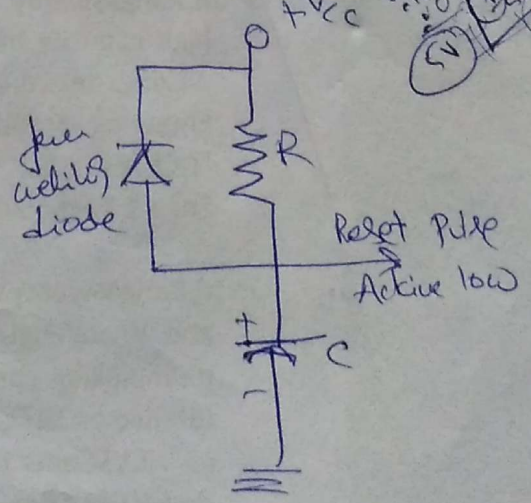
→ The reset signal to MP can be applied at power ON through an external passive reset circuit comprising a capacitor and resistor. (or) through a standard reset IC like. MAX 810 from maximum Dallas.

→ The Reset IC based on type of reset signal and logic level (CMOS/TTL) supported by MP/MC.



Internal ckt:-





Scanned by CamScanner

Reset circuit for (MC 8051)

Reset is an Active high i/p when Reset is set to high, 8051 goes back to Power on State.

the 8051 is reset by holding the RST high for at least two machine cycles, and returning it to low.

2 - methods → ① Power on Reset
→ ② manual Reset

① Power on Reset :- ⓐ initially charging of Capacitor make RST high
ⓑ when Capacitor charges fully it blocks DC.

② manual Reset :- closing the switch momentarily will make RST high. After reset the Program counter is loaded with 0000H but the Content on chip RAM is not affected.

Reg :-	Content
P.C	0000h
Accumulator	00h
B-reg	00h
PSW	00h.
SP	07h
DPTR	0000h
All Ports	FFh.

(ii) Brown out Protection circuit :-

Prevents circuit the µP/µC from un-Expected Program Execution behavior when the supply voltage to µP/µC fall below a specified voltage.

→ It is essentially battery Powered device since there are greater chances for the battery voltage drop below the required threshold.

→ The Processor behavior may not be Predictable if the supply voltage fall below the recommend operating voltage.

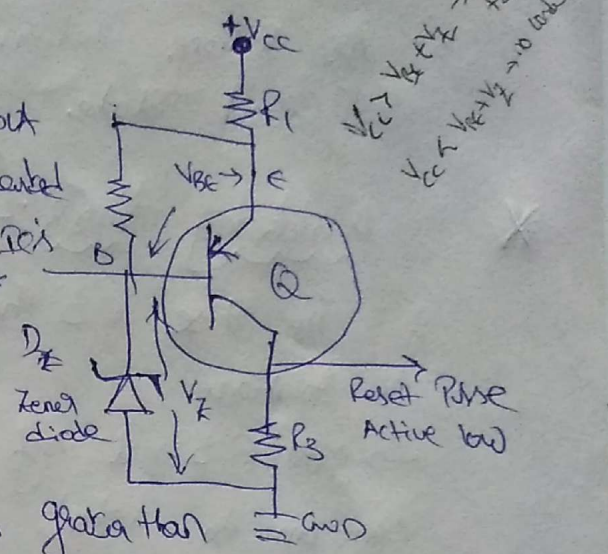
Scanned by CamScanner

which leads Process/controller situation like data corruption. (iii - 5)

→ A brown out protection circuit holds the P/C in reset state when operating voltage falls below threshold state.

In such cases brown out protection circuit monitors the supply voltage internally. If P/C doesnt support built in brown out protection circuit which monitor supply voltage internally.

If P/C doesnt integrated a built in brown out protection circuit which the same can be implemented using external passive circuits (or) supervisors IC's



The circuit having zener diode D_z and transistor Q form the ckt. the transistor conducts always when supply voltage V_{cc} is greater than that of sum of V_{BE} and V_z (zener voltage)

→ The transistor stop conducting when supply voltage falls below sum of V_{BE} and V_z "

→ An brown out can cause one of 3 things for dc supply system. these things in turn can damage connected embedded system.

(a) An unregulated direct current supply will produce a lower o/p voltage for electronic circuits. the o/p ripple voltage will decrease in line with usually reduced load current.

(b) A linear direct current regulated supply will maintain the o/p voltage unless the brown out is severe and i/p voltage drops below the dropout voltage for the regulator, at which point the o/p voltage will fall and high levels of ripple from the rectified/reservoir capacitor will appear on o/p.

(c) A switched-mode power supply which has regulated o/p will be affected. As i/p voltage falls, the current draw will increase to maintain the same o/p voltage and current, until such a point that power supply malfunctions.

Scanned by CamScanner

→ Brown outs can cause unexpected behavior in systems with digital control circuits. Reduced voltages can bring control signals below the threshold at which logic circuits can reliably detect which state is being represented.

→ As voltage return to normal levels the logic can latch at an in correct state; even can't happen state become possible.

* Watchdog timers:-

For those E.S that can't be constantly watched by human, watchdog is an solution.

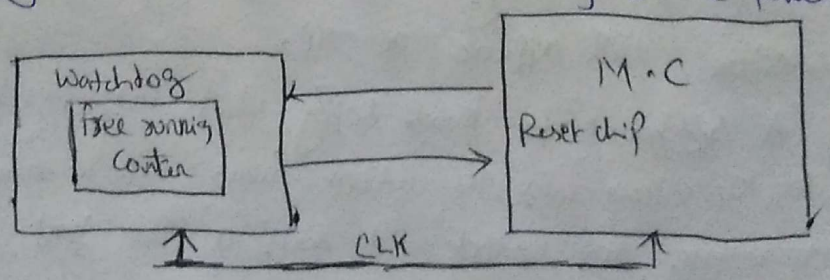
→ most E.S need self reliant. It's not usually possible to wait for some one to reboot them if software hangs.

→ Some embedded designs, such as space probes, are simply not accessible to human operators. If their software ever hangs, such systems are permanently disabled.

A watchdog timer is a piece of hardware that can be used to automatically detect software anomalies and reset the processor if any occur. Generally watchdog timer is based on a counter that count down from some initial value to zero.

→ The E.S stores the counters initial value and periodically restarts it.

→ If counter ever reaches zero before software restarts it, the software is presumed to be malfunctioning and processors reset signal is asserted. The processor (and E.Software its running) will be restarted as if a human operator had cycled the power.



Scanned by CamScanner

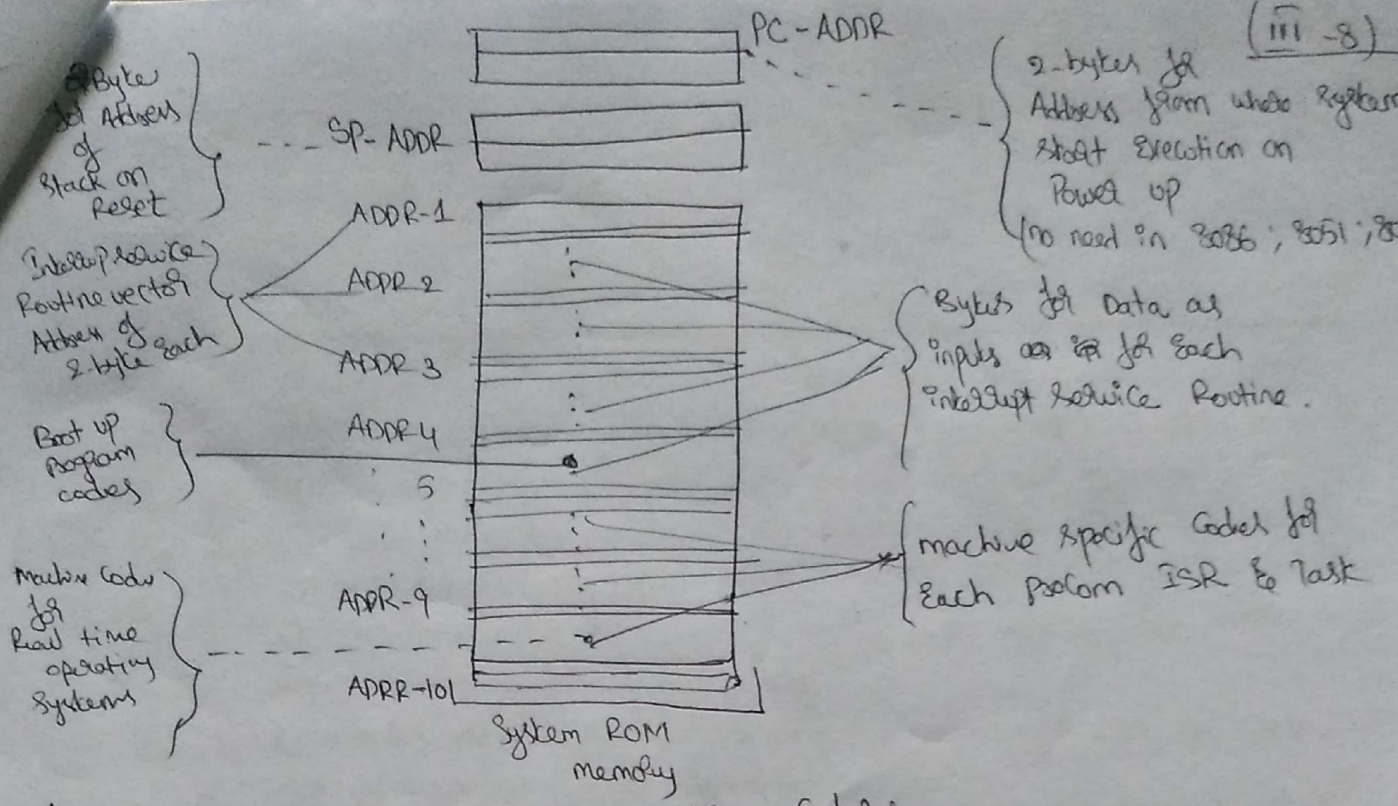
Embedded Software in a System:-

→ " is like a brain of E.S

(i) Machine implementable software for system
(ii) Coding of software in machine codes
(iii) Software in Processor Specific Assembly language
(iv) Software in High level language
(v) Program models for software designing
(vi) Software for Concurrent processing & scheduling multiple tasks
(vii) " " Device Drivers & Device managment in an O.S.
(viii) " Tools for Designing an E.S.

(i) Machine implementable software for system:-

→ E.S executes software that is specific to given application of system.
→ The instruction codes and data in final phase are placed in ROM (or) flash memory for all tasks that are executed when the system runs.
→ The software is also called ROM image
→ E.S is also a unique placement and arrangments of bytes for instructions and data.
→ Each code (or) datanum is available only in the bits and byte format.
→ The system requires bytes at each ROM address, according to tasks being executed.
→ The machine implementable software file is therefore like a table having in each rows the address and bytes.
→ The table has reached as ROM image for targeted hardware.
→ The image consists of boot up program, stack address, pointers, Program counter address pointers, application programs, ISR's, RTOS; input data and vector address.
→ Final stage software is also called the ROM image.
→ By changing this image, the same hardware platform will work differently and can be used for entirely different applications for new upgrades.



(ii) Coding of software in machine Codes:-

→ The Programmer defines the addresses and corresponding bytes (or) bits at Each address.

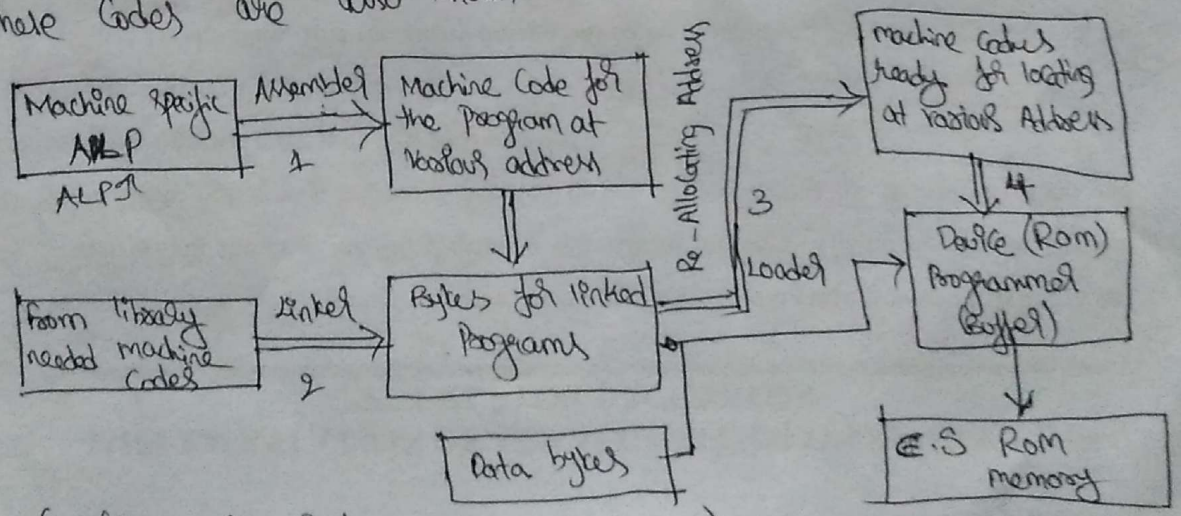
→ In Configuring some specific physical device (or) subsystem, machine Code based Coding is used.

(iii) Software in Processor specific Assembly Language:-

→ A Program (or) a small specific part Can be Coded in AL (Assembly language) using an assembler after understanding the Processor and its instruction set.

→ Assembly language code is extremly usefull for Configuring physical devices like ports, a line display, interface, ADC and DAC and reading into (or) transmitting from a buffer.

→ these Codes are also known low-level Codes.



(Process Converting A.L into machine language.)

Steps:- (Fig 9)

Step 1:- The Assembler translates the assembly software into machine code using a step called assembling.

Step 2:- Linking:- Linker links these codes with the other codes required. Linking is necessary because of the number of codes to be linked for final binary file.

Step 3:- The Loader program performs the task of reallocating the codes after finding the physical memory address available at a given instant. The loader is part of operating system and places code in to memory after reading .exe file.

→ The available memory address may not start from 0x0000 and binary codes have to be loaded at different address during the run.

Step 4:- The system design process is locating those codes as ROM image. the codes are permanently placed at address actually available in ROM.

→ Step 5:- A laboratory system called device programmer takes as input the ROM image file and finally burns image into PROM (or) flash.

The process of placing the codes in PROM (or) flash is also called burning.

(iv) Software in High level language:-

→ Coding in Assembly language is very time consuming in most cases. Software developed in High level language. C (or) C++ (or) Java in most cases.

fig:- Shows process of converting C-program into ROM image file.

Preprocessor Commands
main functions
Interrupt service Routines
Task 1 → N
Kernel & Scheduler
Standard library function including new protocol func for sending & receiving stack.

→ A compiler generates object code. It assembles codes according to processor instruction set and other specifications.

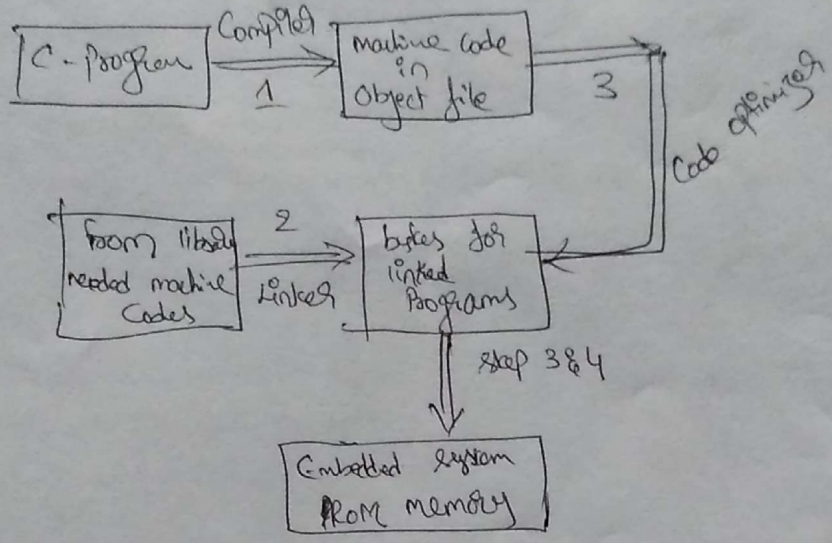
→ The C-compiler for embedded system must as a final step of compilation use a code optimizer that optimize code before linking.

Scanned by CamScanner

→ After Compilation linker links the object codes with other. (iii-10)

→ C; C++ Java; Visual C++ are language for software development

→ C-Program has various layers: Process Command; main function; task and library functions; Interrup service routine & kernel.



Converting C-Program to ROM image.

(V) Program Models for software designing:- SOCFMU

(a) Sequential Program model
(b) Object oriented " "
(c) Control and data flow graph (or) Synchronous Data flow graph (or) Multi thread Graph
(d) Finite State machine for data ~~graph~~ path.
(e) Multithread model for concurrent processing of Process (or) threads (or) tasks
(f) Universal modeling Language (UML). for obj oriented Programming.

(VI) Software for Concurrent Processing and scheduling of multiple tasks and ISRs using an RTOS:-

(*) E.S Program is most often designed using multiple Processes (or) multiple tasks; (or) Multithreads

(*) the multiple tasks are Processed most often by OS not sequentially but concurrently.

(*) Concurrent Processing tasks can be interrupted for ~~for~~ running the ISR and higher Priority task present the running of lower priority.

Scanned by CamScanner

→ An OS Provides for Processes, memory devices, IOs and file system management

→ A file system specifies a way file Created, name Used, Copied, saved, (or) deleted, opened and closed.

→ file system is software for files on disk, flash memory, memory card, memory stick.

→ RIOS Provide the OS functions for Coding the system, Provides Inter Process Communication functions and Controls the passing of messages and signals

(vii) Software for Device Drivers and Device managment in O.S :-

→ E.S Designed to Perform multiple functions & Control multiple Physical and virtual devices.

→ E.S Physical devices → like timers, displays, Keyboards, flash memory, Parallel Ports and network cards.

→ E.S virtual devices are as follows :-

(a) A file :- (of records opened, read, written, and closed and saved as a streams of bytes (or) words)

(b) A Pipe :- (sending, Receiving a stream of bytes from source to destination)

(c) A socket :- (" " " " b/w client and server software and b/w source and destination Computing system).

(d) A RAM disk :- (for using RAM in a way similar to files on disk)

→ A file is a data structure, ((or) virtual device) which sends records (characters (or) words) to a data sink (ex- Program function) which stores data from data source.

(viii) Software tools for desining an Embedded System :-

(a) Editor :- writing C Codes (or) Assembly mnemonics; (Creates source file).

(b) Interrupter :- (line by line) translation to machine executable codes

(c) Compiler :- uses Complete set of codes includes code functions & expression from library routines. (Creates object file).

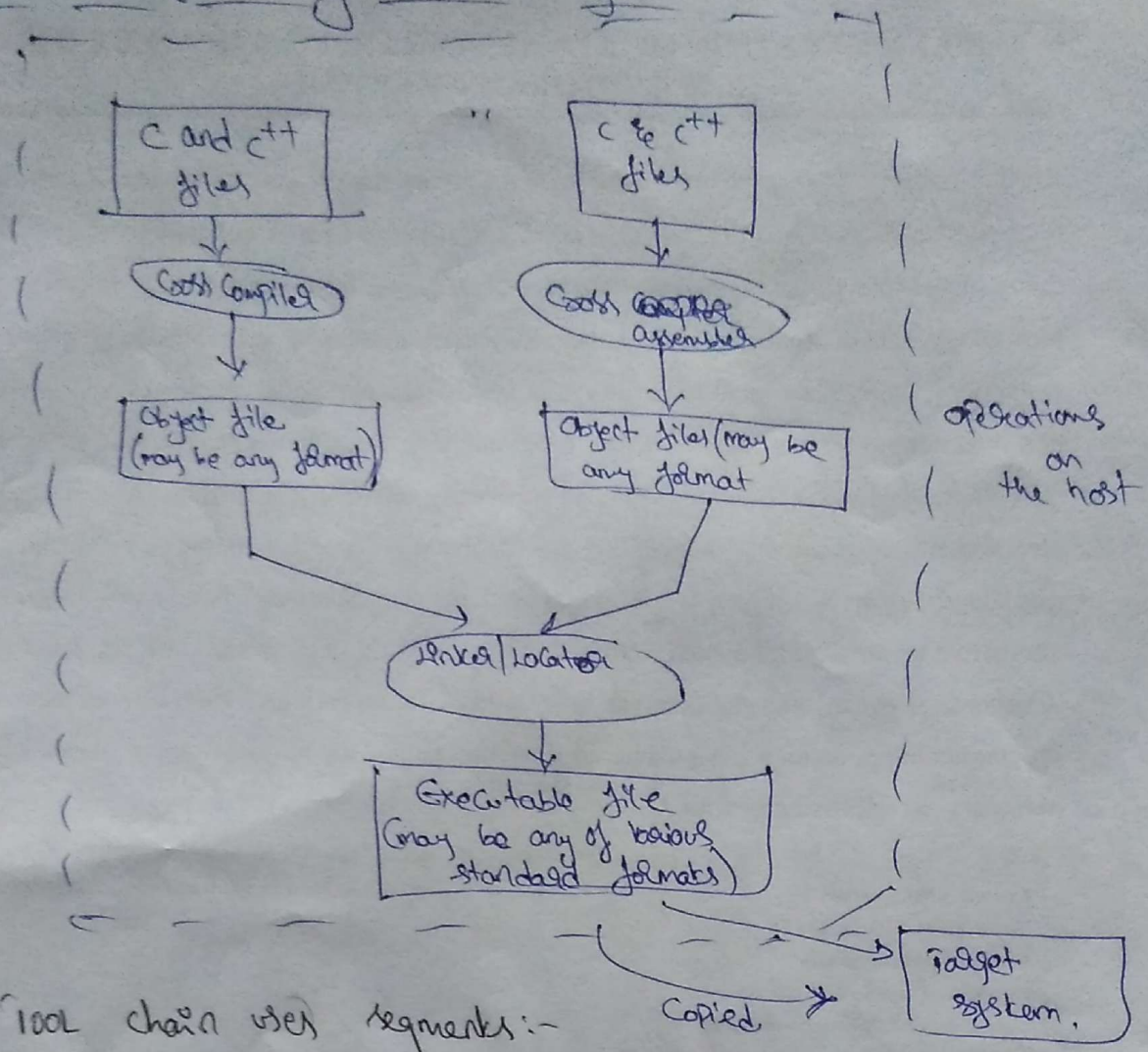
(d) Assembler :- translates assembly mnemonics into binary opcodes into executable files

(e) Cross Compiler :- Converts object codes to executable codes for Processors

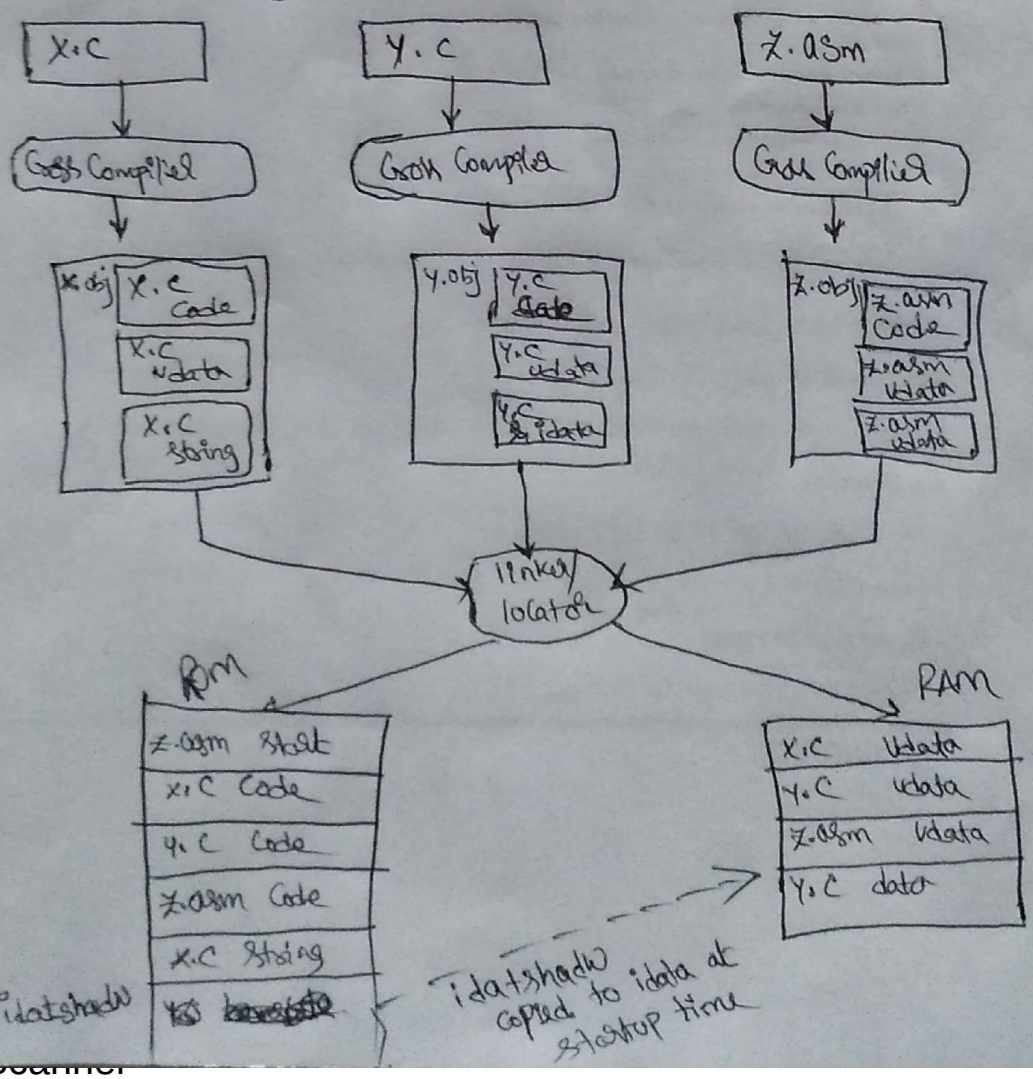
(f) Simulator :- Simulates all function of an Embedded system ckt including additional mem

Scanned by CamScanner

Tool chain for building Embedded Software:-



Tool chain uses segments:-



Scanned by CamScanner

*) Embedded Firmware design approaches :-

Approaches

Conventional Procedural based design (or) also known as (Super Loop Based approach)

→ Adopted for application that are not time critical and no response time is important.

→ Executable task by task.

Process :-

① Configure the common parameters & perform initialisation for various hardware component memory, register etc.,

② Start task - 1 and execute it

③ Execute the second task

④ Execute the next task

⑤ ⋮

⑥ ⋮

⑦ Execute last defined task

⑧ Jump back to first task and follow the same

infinite loop based approach.

Eg:-

```
void main ( )
{
Configurations ( );
Intialization ( );
while (1) {
{
Task 1 ;
Task 2 ;
:
Task N;
}
}
```

'C' Code

super loop based approach.

Embedded operating system Based design.

Eg:- of Embedded Products using microsoft windows XP OS are Personal Digital assistant (PDAs); Hand held devices | Portable devices and point of sale (POS) terminals.

→ RTOS based on approach demanding Real time response

Eg:- Video game toy containing

→ Key Pad and display unit

→ Designed for low cost Embedded Product

drawback :- lack of real timeliness, at which each task has to be executed.

Scanned by CamScanner

Embedded C - Programs:-

→ A Program is a set of instructions that are grouped together to accomplish tasks

→ The set of instructions are written using Assembly, C, C++, Java.

Eg:- of C- Program:-

```
# include <avr/io.h>  →  [io.h located under folder avr]
int main (void)
{
int a ;  ← declare variable name ;
a = 10;
return 0;  ← turns value '0' to O.S
}
```

Data types:-

int → 10 pens

float → 93.4 Percent

char → 4

Statements:-

① Expression statements :-

A=10 ; (or) A=B ;

Printf ("Hello");

② Repetation statements:-

while loops ; for loops

do while loops.

③ Conditional execution and Selection statements

① if statement

② if else statement

③ Switch statements.

④ Special Control statements

① return statement

② break statement

③ Continue statement

Scanned by CamScanner

Processors :-

→ `# define Color blue` // Replace tokens in current file with specified replacement token(s).

→ Embed file within the current file :-

`# include <arr/io.h>`

→ Conditional compilation sections of current file → we can include (source code) to main program

```
# if def DEBUG
    // Print debug values here with memory addresses
# else   // do something else here
# End if
```

→ Example for # define and # include

```
# define TRUE 1
# define FALSE 0
# define Buffer Size 512
# define MAX_ROW 86
~~# define MAX_ROW~~
# define MAX_COLUMN 80
int hour,
int min,
int sec.
```

→ Using # include in main program file (main.c)

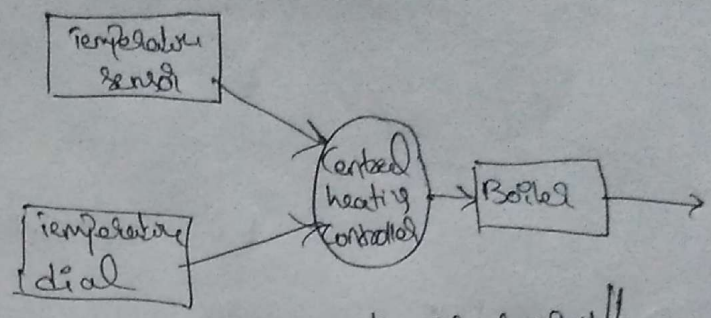
```
# include "mydef.h"
int main (void)
{
return 0;
}
```

Example for superloop software :-

```
void main (void)
{
x_init ();      // Before for task x //
while (1)       // For ever (super loop) //
{
x( );           // Before the task //
}
}
```

Scanned by CamScanner

Program for Central heating Controller:-




```
#include "cen_Heat.h"
void main (void)
{                          // initialize the system//
C_Heat_Init ();,
while (1)          // for ever (super Loop)
{
                           // find out what temperature the user required//
C-Heat-Get-Required_Temperature ();  // via the user interfaces//.
C-Heat_GET_Actual-Temperature ();    // find out what actual room temperature is via
                                     temperature sensor//
C-Heat-Control-Boiler ();  // Adjust the gas burner; as required
}}.
```

A simple desktop C-Program:-

```
#include <Stdio.h>
int main (void)
{ printf ("Hello world\n");
  return 0;
}
```

Embedded C Program:-

```
void main (void)
{
LED Flash_Int ();
while (1)          // change the LED state(off -> on)
{
LED_FLASH_CHANGE_STATE ();
Delay_Loop_wait (1000); //delay for approx 1000ms.
}}
```

Eg:- A simple super loop application copies value from P_1 to P_2.
read from P_1 and copies to P_2.

```
#include <Reg.52.H>
void main (void)
{ unsigned char Port1_value;
  P1 = 0XFF;      //must setup P1 for reading//
  while (1)
  { Port_1_value = P1;   // Read the value of P1 //
    P2 = Port1_value;    // copy to value of P2 //
  }}
```

Scanned by CamScanner

Hello World Embedded Program using 8051 :-

```
# include <reg 52.h>
~~void LED_Flash_Init (void);~~
Sbit LED_Pin = P1^5;        // LED is Connected to Pin
bit LED_State_G;            // stores LED state

void LED_Flash_Init (void);
void LED_FLASH_Change_State (void);
void DELAY_LOOP_WAIT (Const unsigned int);   // function Prototypes.
                                   /* - - - - - - - - - - - - - - */
void main (void)
{
  LED_FLASH_Init ( );
while (1)                          // change the LED state (OFF to ON (or) vise versa)
{ LED_FLASH_change_State ();
  DELAY_LOOP_WAIT (1000); // Delay for *approx* 1000 ms.
}}
                                   /* - - - - - - - - - - - - - -
```

```
else
{
LED_State_Gr = 1;
LED_Pin = 1;
}}.
```

/* - - -