UNIT III

EMBEDDED FIRMWARE DEVELOPMENT ENVIRONMENT

3.1 C Program Elements

Pre processor include Directive

_Header, configuration and other available source files are made the part of an embedded system program source file by this directive

Examples of Preprocessor include Directives

#include" *V x Work s. h*"/*Include V x Works functions*/
#include " *sem Lib. h*"/*Include Semaphore functions Library */#include "*task Lib .h*"/*Include e multi tasking functions Library*/
#include "*sys Lib .c*"/*Include system library for system functions */
include"*netDrvConfig*.txt"/* Include a text file that provides the 'Network Driver Configuration'. */
include "*prctl Handlers .c*" /* Include file for the codes for handling and actions as per the protocols used for driving streams to the network.*/

Pre processor Directive for the Definitions

• Global Variables─*# define volatile Boolean Int Enable*
• Constants ─*#definefalse0*

• Strings─*# define welcome msg* " Welcome To ABC Telecom"

Pre processor Macros

• Macro-A named collection of codes that is defined in a program as preprocessor directive.
• Differs from a function in the sense that once a macro is defined by a name, the compiler puts the corresponding codes at the macro at every place where that macro-name appears.re used for short codes only.

Difference between Macro and Function

• The codes for a function compiled once only
• On calling that function, the processor has to save the context ,and on return restore the context.
• Macros are used for short codes only.
• When a function call is used instead of macro, the overheads (context saving and return) will take time, T overheads that is the same order of magnitude as the time, T exec for execution of short codes with in a function.
• Use the function when the T over heads<<T exec and macro when T overheads ~= or > T exec.

Use of Modifiers

- auto
- unsigned
- static
- const
- register
- interrupt
- extern
- volatile
- volatile static

Use of infinite loops

_Infinite loops-Never desired in usual programming. Why ? The program will never end and never exit or proceed further to the codes after the loop.
_ Infinite loop is a feature in embedded system programming! Example:
A telephone is never switching off.
The system software in the telephone has to be always in a waiting loop that finds the ring on the line. An exit from the loop will make the system hardware redundant.

```
# define false
0#definetrue 1
Void main (void){
/* Call RTOS run here
*/rtos .run();
/* Infinite while loops follows in each task. So never there is return from the RTOS. */
}
Void task1(... ){
/* Declarations
*/.while (true){
/*Run Codes that repeatedly execute*/
/* Run Codes that execute on an
event*/if(flag1){.      ;}; flag1 =0;
/* Codes that execute for message to the kernel
*/message1 ();} }
```

Use of type def
_ Example ─ A compiler version may not process the declaration as anunsigned byte
_The 'unsigned character' can then be used as a data type.
_Declared as follows: type def un signed character port A data
_Used as follows: #definePbyteportAdata0xF1

Use of Pointers

Pointers are powerful tools when used correctly and according to certain basic principles.

#define COM ((structs io near*) 0x2F8);

This statement with a single master stroke assigns the addresses to all 8variables

By teat the sio Addresses

0x2F8: Byte at RBR/THR /DLATCH-
LByte0x2F9:By teat DLATCH-HByte
0x2FA: Byte at IER; 0x2FB: Byte at
LCR;0 x 2FC:Byte at MCR;
0x2FD: Byte at LSR; 0x2FE: Byte at
MSR0x2FF: Byte Dummy Character

Example

Free the memory spaces allotted to a data structure.
#define NULL (void)0x0000*
• Now statement *& COM ((structs io near*) = NULL ;*assigns the COM to Null and make free the memory between 0x2F8and 0x2FFforother uses.
Data structure
• Example ─ structures *io*
• Eight characters─ Seven for the bytes in BR/THR/ DLAT CHL Byte ,IER ,IIR ,LCR ,MCR ,LSR ,MSR registers of serial line device and one dummy variable **re consisting of 8 character variables structure for the COM port 2 in the UART serial line device at an IBMPC.**

Example of Data structure declaration

• Assume structured variable COM at the addresses beginning 0x2F8.
#define COM ((structsio near*) 0x2F8)
• COM isat 8addresses0x2F8-0x2FFandis a structure consisting of 8 character variables structure for the COM port 2 in the UART serial line device at an IBMPC.
#define COM1 ((structsio near*) 0x3F8);
It will give anotherstructuredvariableCOM1at addresses beginning0x3F8using the data structure declared earlier as *sio*

Use of functions

(i) Passing the Values (elements):
The values are copied into the arguments of the functions. When the function is executed in this way ,it does not change a variable's value at the function, which calls new function.
(ii) Passing the References
When an argument value to a function passes through a pointer ,the called function can change this value. On returning from this function , the new value may be available in the calling program or another function called by this function.

Useof Reentrant Function

• Re entrant function-A function usable by the several tasks and routines synchronously (at the same time).This is because all the values of its argument are retrievable from the stack.

Three conditions for a function called as *re entrant function*

1. All the arguments pass the values and none of the argument is a pointer (address) whenever calling function calls that function.
2. When an operation is not atomic, that function should not operate on any variable, which is declared outside the function or which an interrupt service routine uses or which is a global variable

but passed by reference and not passed by value as an argument into the function. [The value of such a variable or variables, which is not local, does not save on the stack when there is call to another program.]

3. That function does not call any other function that is not itself Re entrant.