UNIT III

EMBEDDED FIRMWARE DEVELOPMENT ENVIRONMENT

3.3 CONCEPTS ANDEMBEDDED PROGRAMMINGINC,C++

Multitasking

Function *main* with a waiting loop

main ( ) passes the control to an RTOS
Each task controlled by RTOS and
Each task will also have the codes in an infinite loop
A waiting task is passed a signal by the RTOS to start.

Main () calling RTOS

```
 # define false
 0#definetrue 1
 /*****************************************************************/
void main (void){
 /* Call RTOS run here
*/Infinite loop in main
( ) while (1){ rtos.run();
 /* Infinite while loops follows in each task. So never there is return from the RTOS. */
 }
 }
 /*****************************************************************/
```

Task1

```
Void task 1(.. ){
 /*Declarations */
.
While (true){
 /*Codes that repeatedly execute*/
.
 /* Codes that execute on an event
*/If (flag1){. ;}; flag1 =0;
 /* Codes that execute for message to the kernel
 */message1 ();
 } }
 /*******************************************/
```

Task 2 ()

```
void task 2(... ){
 /*Declarations */
.
while (true){
 /*Codes that repeatedly execute*/
.
 /* Codes that execute on an event
*/if(flag2){.  ;}; flag2 =0;
 /* Codes that execute for message to the kernel
 */message2 ();
 } }
 /*******************************************/
```

TaskN_1 ()

```
voidtaskN_1 (….) {
 /*Declarations */
.
While (true){
 /*Codes that repeatedly execute*/
.
 /*Codes that execute on an event */
If (flagN_1){…. ;}; flagN_1 =0;
 /*Codes that execute for message to the kernel*/
 messageN_1 ();
 } }
 /*********************************************/
```

Task N

```
Void task N( . ) {
 /*Declarations */
.
While (true){
 /*Codes that repeatedly execute*/
.
 /* Codes that execute on an event
*/I f( flag N ){.          ;}; flag N=0;
 /* Codes that execute for message to the kernel
*/message N ();
 } }
 /*********************************************/
```

2. Polling for events and messages

_A Programming method is to facilitate execution of one of the multiple possible function calls and the function executes after polling

_Polling example is polling for a screen state (or Window menu)*j* and for a message from an

Mobile phone

_ Assume that screen state j is between 0 and K , among 0, 1, 2, ..or K – 1 possible states.(set of menus).

_ An interrupt is triggered from a touch screen GUI and an ISR posts an event- message *m*= 0, 1, 2, …, or N – 1 as per the selected the menu choice 0, 1,2, …, N – 1 when there are N menu- choices for a mobile phone user to select from a screen insta tej.

Polling for a menu selection from screen state

```
Void poll_menu K {/* Code for polling for choice from menu m for screen state K*/
 }
 }
 /*********************************/
```