Design and Integration of Embedded Systems (BMEVIMIMA11)

Homework Assignment

(Option 1: Prototype implementation and testing of the protocol)

Name: Ayaz Samadli

Neptun Code: BCSPT6

I declare that this homework has been done by myself, Ayaz Samadli

Contents

Introduction

I chose the first option since I love low level programming. I have been created two different functions namely field_obj and control_obj to simulate the protocol between field object and control object.

Firstly, I declared 5 constant integer variables, they will be used for the state of the objects (state of our functions). The field_obj and control_obj functions will return one of these cases, the Reset is sent when the clkReset is higher or equal than tRTMax by field object or control object. It means, if object doesn't get any signal during tRTMax it reset the connection. OBJ1 is sent to Control element by the Field object once in a tSync period for trying to make a connection, if Control Object accept the request it send OBJ2 message and handshake happens. After the connection is established field object starts to send OBJUP message to the control object and control object sends OBJ2 message to the field object instead. This is the data transferring phase. In addition, if the objects don't send any signal, they return -1.

```
const int RESET = 0;
const int OBJ1 = 1;
const int OBJ2 = 2;
const int OBJUP = 3;
const int OBJDOWN = 4;
```

Field Object function:

It takes 3 arguments and its return type is integer.

```
int field_obj(int input_for_field_object, int time_for_sync, int time_for_reset)
```

Input_for_field_object is the output of the control object (state of control object), time_for_sync and time_for_reset is the duration of synchronization and resetting, respectively.

clkReset and clkSync are the timers of field object in real industry, to simulate these timers I just take the current time by the help of time function from the "time.h" library.

```
clkReset = (ll) time(NULL);
clkSync =  (ll) time(NULL);
```

Then if else condition blocks define what to do.

```
if(clkReset >= tRTMax){
    Field_State = RESET;
    reset_time(&clkReset, &tRTMax, &reset_duration);
    reset_time(&clkSync, &tSync, &Sync_duration);
}
```

If the clkReset >= tRTMax then the State of the Field Object turns to RESET and the timers are resetted. Reset_time is the user defined function. It takes three arguments as following:

```
void reset_time(ll *a, ll *b, ll *c){
    ll now = (ll) time(NULL);
    *a = (ll) now;
    *b = *a + *c;
}
```

The parameters are taken as a pointer, because I want to change them globally not a local change inside of the function. When we get the arguments as a pointer, the changes happen in the address of the memory where the variables are stored, that is why their values change in everywhere. The data type of the variables is long long int, I defined long long int as ll for easy writing. The first parameter takes the current time and the second parameter takes the current time + duration of the response (reset or synchronization).

The second condition checks if the output of the control object is Reset.

```
else if(input_for_field_object == RESET){
    reset_time(&clkReset, &tRTMax, &reset_duration);
    reset_time(&clkSync, &tSync, &Sync_duration);
    return -1;
}
```

If the reset state is got, the timers are resetted and the field object should not send any signal, that is why the function returns -1. You can ask why I do not change the state of the Field Object. Because if the state of the field object was changed we could not know what was the previous state of the field object. Therefore, we should create another variable and keep the previous state. However, it makes the code more complicated that is why I don't change the state of the field but return -1.

Then, we check if the OBJ2 or OBJDOWN messages are got or not.

```
else if(input_for_field_object == OBJ2 || input_for_field_object == OBJDOWN){
    reset_time(&clkReset, &tRTMax, &reset_duration);
    if(clkSync >= tSync){
        Field_State = OBJUP;
        reset_time(&clkSync, &tSync, &Sync_duration);
    }
    else{
        return -1;
    }
}
```

If we have already got one of these messages it the clkReset should be resetted but we do not hurry to change the clkSync. Firstly, we check if the clkSyc is greater or equal than tSync, if yes we reset the clkSync and Field state changes to OBJUP, else field object should not send any signal that is why the function returns -1.

If the above conditions are false it is checked if the clkSync is greater or equal than tSync. If yes, the state of the control object is checked, if it is one of the OBJ2 or OBJDOWN the state of the field changes to the OBJUP, otherwise to the OBJ1. After all, the clkSync is resetted.

```
else if(clkSync >= tSync){
    if(Control_State == OBJ2 || Control_State == OBJDOWN){
        Field_State = OBJUP;
    }
    else{
        Field_State = OBJ1;
    }
    reset_time(&clkSync, &tSync, &Sync_duration);
}
```

If all the above conditions are not true the function returns -1. If the state of the field object is changed the function will return this new state.

```
else{
    return -1;
}

return Field_State;
```

Control Object function

It takes two argument, the output of the field object and the duration of the reset.

```
int control_obj(int input_for_control_object, ll time_for_reset){
```

In this function we do not use clkSync, because it is meaningless to make the controller wait, the controller should be fast as much as possible. However, let's say the controller has some delay, in this case we should create message queue. For example, if the clkSync for the field object is less than clkSync of control element. In this case, after getting some responses from the field object, the control object should send the answer to the first signal. In addition, we are not asked to use clkSync for control object as seen from the below diagram:

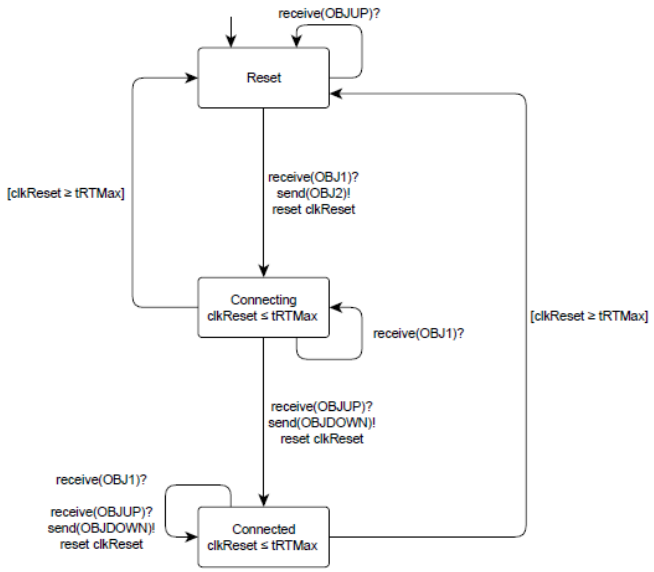

Figure 2: Operation of the control object

If the clkReset is greater or equal than tRTMax the clkReset is resetted and the Control State changes to RESET.

```
if(clkReset_Controller >= tRTMax_Controller){
    reset_time(&clkReset_Controller, &tRTMax_Controller, &reset_duration_Controller);
    Control_State = RESET;
}
```

If the output of the field object is OBJ1 clkReset is resetted and the state of the control object changes to OBJ2.

```
 else if(input_for_control_object == OBJ1){
     Control_State = OBJ2;
     reset_time(&clkReset_Controller, &tRTMax_Controller, &reset_duration_Controller);
 }
```

If the output of the field object is OBJUP the state of the control object changes to OBJDOWN and the clkReset is resetted.

```
else if(input_for_control_object == OBJUP){
    Control_State = OBJDOWN;
    reset_time(&clkReset_Controller, &tRTMax_Controller, &reset_duration_Controller);
}
```

If the output of the field object is RESET the clkReset time is resetted and the function returns -1 (no signal)

```
else if(input_for_control_object == RESET){
    reset_time(&clkReset_Controller, &tRTMax_Controller, &reset_duration_Controller);
    return -1;
}
```

If all the above conditions are not truth the function returns -1. If the state of the field object is changed the function will return this new state.

```
    else{
        return -1;
    }

    return Control_State;
```

The Scheduler (main function)

Firstly, we initialize and reset all the timers

```
int main(){

    int field_message, control_message = -1, i;
    ll r_duration = 12, r_c_duration = 12, sync_duration = 2, begin_time;

    clkReset = (ll) time(NULL);
    clkSync = (ll) time(NULL);
    begin_time = (ll) time(NULL);

    reset_time(&clkReset, &tRTMax, &r_duration);
    reset_time(&clkSync, &tSync, &sync_duration);
    reset_time(&clkReset_Controller, &tRTMax_Controller, &r_c_duration);
```

Within the 45 iterations, firstly we take the output of the field object, then print some messages accordingly.

```
for(i = 0 ; i < 45; i++){

    field_message = field_obj(control_message, sync_duration , r_duration);

    ll sec = (ll) time(0) - begin_time;

    if(field_message == OBJ1){
        printf("%ld th second: \t" , sec);
        printf("Connecting... 'OBJ1' is sent by the Field Object to the Control Object \n");
    }
    else if(field_message == OBJUP){
        printf("%ld th second: \t" , sec);
        printf("Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object \n");
    }
    else if(field_message == RESET){
        printf("%ld th second: \t" , sec);
        printf("Restarting... Connection is resetted by the Field Object \n");
    }
```

The same for the control object and wait one second after each iteration:

```
    control_message = control_obj(field_message, r_c_duration);

    if(control_message == OBJ2){
        printf("%ld th second: \t" , sec);
        printf("Connected... 'OBJ2' message is sent by the Control Object to the Field Object \n");
    }
    else if(control_message == OBJDOWN){
        printf("%ld th second: \t" , sec);
        printf("Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object \n");
    }
    else if(control_message == RESET){
        printf("%ld th second: \t" , sec);
        printf("Restarting... Connection is resetted by the Field Object \n");
    }
    sleep(1);
}
```

Test Drive

The field object function stays the same but I replace the control object and schedular with test drive.

```
if(field_message == OBJ1){
    printf("%ld th second: \t" , sec);
    printf("Connecting... 'OBJ1' is sent by the Field Object to the Control Object \n");
    control_message = OBJ2;
}
else if(field_message == OBJUP){
    printf("%ld th second: \t" , sec);
    printf("Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object \n");
    control_message = OBJDOWN;
}
else if(field_message == RESET){
    printf("%ld th second: \t" , sec);
    printf("Restarting... Connection is resetted by the Field Object \n");
    control_message = -1;
}
else{
    no_signal = true;
}
```

So, I change the state of the control object within the main function and check the field function's response.

```
    if(i >=20 && i<=23){
        control_message = -1;
    }
    else if(no_signal){
    // do nothing
    }
    else if(control_message == OBJ2){
        printf("%ld th second: \t" , sec);
        printf("Connected... 'OBJ2' message is sent by the Control Object to the Field Object \n");
    }
    else if(control_message == OBJDOWN){
        printf("%ld th second: \t" , sec);
        printf("Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object \n");
    }
    else if(control_message == RESET){
        printf("%ld th second: \t" , sec);
        printf("Restarting... Connection is resetted by the Field Object \n");
    }
    sleep(1);
}
```

I add some condition for the restarting the connection. So, if the iterator is between 20 and 23, control object does not send any signal to the field object. The clkSync is 2 second so, after two iteration field object restarts the connection.

The results

When the process starts the objects are in the reset state and the clkSync is 2 seconds. That is why in the 2nd second field object starts to send OBJ1 message to the control object, this is the connection phase and control object accepts it so the phase changes to the connected. Then field and control objects send the data to each other. I do not break the connection as it is required in the task; The log of schedular:

```
C:\Users\Ayaz Samadli\Desktop\Scheduler\scheduler.exe
2 th second:    Connecting... 'OBJ1' is sent by the Field Object to the Control Object
2 th second:    Connected... 'OBJ2' message is sent by the Control Object to the Field Object
4 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
4 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
6 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
6 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
8 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
8 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
10 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
10 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
12 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
12 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
14 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
14 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
16 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
16 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
18 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
18 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
20 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
20 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
22 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
22 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
24 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
24 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
26 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
26 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
28 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
28 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
30 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
30 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
```

The branch coverage summary for scheduler:

```
C:\Users\Ayaz Samadli\Desktop\Scheduler>gcov -b -c scheduler
File 'scheduler.c'
Lines executed:80.00% of 80
Branches executed:100.00% of 38
Taken at least once:81.58% of 38
Calls executed:70.27% of 37
Creating 'scheduler.c.gcov'
```

In the test drive I do not send signal to the field in between 20-23 seconds just for artificially resetting the connection. The log of the test drive:

```
C:\Users\Ayaz Samadli\Desktop\emedded_main\main.exe
2 th second:    Connecting... 'OBJ1' is sent by the Field Object to the Control Object
2 th second:    Connected... 'OBJ2' message is sent by the Control Object to the Field Object
4 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
4 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
6 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
6 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
8 th second:    Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
8 th second:    Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
10 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
10 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
12 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
12 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
14 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
14 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
16 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
16 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
18 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
18 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
20 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
22 th second:   Connecting... 'OBJ1' is sent by the Field Object to the Control Object
24 th second:   Restarting... Connection is resetted by the Field Object
26 th second:   Connecting... 'OBJ1' is sent by the Field Object to the Control Object
26 th second:   Connected... 'OBJ2' message is sent by the Control Object to the Field Object
28 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
28 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
30 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
30 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
32 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
32 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
34 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
34 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
36 th second:   Data Transfering... 'OBJUP' message is sent by the Field Object to the Control Object
36 th second:   Data Transfering... 'OBJDOWN' message is sent by the Control Object to the Field Object
```

The branch coverage summary for test drive

```
C:\Users\Ayaz Samadli\Desktop\emedded_main>gcov -b -c main
File 'main.c'
Lines executed:94.74% of 38
Branches executed:100.00% of 20
Taken at least once:95.00% of 20
Calls executed:90.00% of 20
Creating 'main.c.gcov'

File 'FieldObject.h'
Lines executed:89.47% of 38
Branches executed:100.00% of 18
Taken at least once:83.33% of 18
Calls executed:83.33% of 12
Creating 'FieldObject.h.gcov'
```

To sum up, I have implemented all the desired functionalities and they work perfect.