CHAPTER 4
REAL-TIME HARDWARE CO-PROCESSOR IMPLEMENTATION

4.1 INTRODUCTION

A number of hardware co-processor designs have been implemented by researchers during recent years to overcome the overheads involved in the software-only implementation of RTOS. The aimed configurations are heavily influenced by operational goals of the hardware setup. The co-processor is intended to be an overall centralized task controller for the system. It performs the function in accordance with a specified scheduling policy. The co-processor decides which tasks should be executed on the target processor. Within the co-processor, a number of scheduling policies may be available. Scheduling and task switching decisions are made within the unit using a policy selected by the application programmer. It is assumed that task communication and synchronization functions are application, not support level functions. Hence their selection and use is determined by the application designer not by the co-processor. Most of the co-processors have processors some use 16 bit processors for fast computation of results. A scheduler is used to schedule the multiple tasks in a real-time system.

It is proposed to setup two different hardware environments to execute developed algorithm in the designed simulator. The proposed
simulator has been implemented with two types of hardware setup: 1) Low end microcontroller i.e., 8051 based system and 2) High end system based on ARM7.

4.2 COMPONENTS USED FOR DEVELOPING HARDWARE BOARD

The major components used in the hardware boards are

- Power supply
- Serial port
- LCD
- Microcontrollers (8051 and ARM LPC 2148)
- 5 Devices (Fan, Buzzer, LED, Switch, Bulb)

4.2.1 Power Supply

Power supply transfers electric power from a source to a load using electronic circuits. Some of the requirements of power supply unit are small size, lightweight, low cost, and high power conversion efficiency. It is also possible to generate multiple voltages using linear power supplies. In multi output power supply, a single voltage must be converted into the required system voltages (for example, +5V, +12V and -12V) with very high power conversion efficiency. The multi output power supply is used in the hardware board to supply power. Figure 4.1 shows the power supply
4.2.2 Universal asynchronous receiver transmitter

The Serial Port is harder to interface than the Parallel Port. In most cases, devices connected to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. The UART function is shown in Figure 4.2

```c
char rx_getc()
{
    while(RI==0);//first to receive and check the condition
    RI=0;
    return(SBUF);
}

void tx_putc(char tx)
{
    SBUF=tx;
    while(TI==0);//first to transmit and check the condition
    TI=0;
}
```

Figure 4.2 Sample code for UART
4.2.3 Liquid Crystal Display (LCD)

The LCD indicates to the user which task is currently executed by the processor in the hardware board. Figure 4.3 shows the sample program of LCD working in ARM processor hardware board.

```c
void task_check(char wh)
{
    if(wh==0)
    {
        singlecommand(0xc0);lcddisplay("task - 1 ");
        singlecommand(0xc0);lcddisplay("task - 1 ");
        task1(sec);
    }
    else if(wh==6)
    {
        singlecommand(0xc0);lcddisplay("task - 4 ");
        singlecommand(0xc0);lcddisplay("task - 4 ");
        task4(sec);
    }
    else if(wh==8)
    {
        singlecommand(0xc0);lcddisplay("task - 5 ");
        singlecommand(0xc0);lcddisplay("task - 5 ");
        task5(sec);
    }
}
```

Figure 4.3 Sample code for LCD Display

4.3 DEVELOPING HARDWARE BOARD FOR 8051 MICROCONTROLLER

Microprocessors and micro controllers are widely used in embedded system products. They are freescale 6811, Intel 8051, Zilog Z8 and PIC 16X. Each of these microcontrollers have unique instruction set and
register set; therefore they are not compatible with each other. Program written for one will not run on others. Three criteria for choosing microcontroller are as follows. Meeting the computing need of task effectively and cost efficiently, availability of software development tools such as compilers assemblers and debuggers, wide availability and reliable sources of microcontroller.

Based on these requirements Intel 8051 microcontroller is used in this research work. The view of 8051 based board is shown in Figure 4.4.

![Figure 4.4 Hardware for 8051 board](image)
4.3.1 Assigning 5 Tasks

The LED is initialized to 1, a pointer is declared with array as shown in Figure 4.5. Task one is put into serial cable using TX command and transmitted. If the LED is not initialized, the command will transfer to the new line.

```c
void task1(int v1)
{
    char *ptr;
    tx_puts("task1 ");
    ptr=&array[0];
    ptr++;
    ptr++;
    led1=1;
    on_led_time(*ptr,v1);
    led1=0;
    array[0]=99;
    array[1]=99;
    new_line();
}
```

**Figure 4.5 LED Task 1 Code**

<table>
<thead>
<tr>
<th>Task 1</th>
<th>LED</th>
<th>The value of the task 1 input is given in the front end by the user.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 2</td>
<td>Buzzer</td>
<td>Based on the input value of task 2 from the front end and scheduling algorithm the buffer will sound.</td>
</tr>
<tr>
<td>Task 3</td>
<td>Bulb</td>
<td>The bulb in the hardware board is assigned to task 3.</td>
</tr>
<tr>
<td>Task 4</td>
<td>Motor</td>
<td>The parameters for task 4 has been assigned by user in the front end pane.</td>
</tr>
<tr>
<td>Task 5</td>
<td>Switch</td>
<td>Contains the switch. The switch transmits the current based on the value given to task 5 by the user on the front end to the scheduler.</td>
</tr>
</tbody>
</table>
Figure 4.6 shows the circuit diagram of 8051 based hardware setup.

Figure 4.6 Circuit Diagram of Hardware Board using 8051 Microcontroller

4.3.2 Compiler

RIDE-51 is an 8051 IDE with ANSI-C Compiler, simulator, and debugger by Raisonance. Figures 4.7 and 4.8 show the Ride compiler output view.

Figure 4.7 RIDE compiler
4.3.3 Creation of Hex File

WinISP is a utility that combines the power of in-system flash programming with the user friendliness of a graphical interface.

The step by step procedure to set up 8051 board is given below:

- The board is prepared and the 8051 microcontroller is firmly inserted.
- The serial cable is connected to the hardware board and the serial cable slot of PC Power up the ISP board.
- Boot up the system, start MS Windows and the WinISP software.
- In WinISP, select the chip type 89c51 and the serial port number.
- Force the microcontroller into ISP mode. To do this, press and release the reset switch.

- The system is now ready. The buttons on the left of the WinISP Window are used to load Intel hex files, program/erase/verify the part, set/reset the security bits, etc.

- The microcontroller can be reset at any time to run the program. Please note the following remarks:

The steps to execute file on the target board are shown in Figure 4.9.

![Figure 4.9 Steps to execute file on the Target Board](image)

**Figure 4.9 Steps to execute file on the Target Board**

### 4.3.4 Interfacing

There are two interfacing aspects to be considered; first, the connection between the co-processor and the target processor, second, the connection (if any) from the co-processor to the outside world.
Language, compiler and run-time issues: It is a basic requirement that the co-processor must not restrict the language or compiler used for the application software, with one proviso. The definitions of the overall task design, task structure, attributes and communication components are clearly an application-level function. These must be specified in the target software by the application designer.

**Back end hardware design:** Hardware Design for the simulator inputs are received from the front end through serial port connection. The target processor receives the values from the front end of the system. This target processor controls all the electrical devices on receiving the input, performs the commands and also displays the result in the front end of the system. Sample code in Figure 4.10 shows the five tasks which were executed from the hardware.

```c
void task1(int v1)
{
    char *ptr;
    tx_puts("task1 ");
    ptr=&array[0];
    ptr++;
    led1=1;
    on_led_time(*ptr,v1);
    led1=0;
    array[0]=99;
    array[1]=99;
    new_line();
}
```

*Figure 4.10 (Continued)*
void task2(int v1)
{
    char *ptr;
    tx_puts("task2 ");
    ptr=&array[2];
    ptr++; 
    led2=1;
    on_led_time(*ptr,v1);
    led2=0;
    array[2]=99; 
    array[3]=99;
    new_line(); //control goes to nextline
}
.
.
.

void task5(int v1)
{
    char *ptr;
    tx_puts("task5 ");
    ptr=&array[8];
    ptr++; 
    led5=1;
    on_led_time(*ptr,v1);
    led5=0;
    array[8]=99; 
    array[9]=99;
    new_line();
}

Figure 4.10 Sample Code for Hardware
**Hardware software integration** Win x Talk 1.5 has been used to set the baud rate of serial port and also used to see the execution of tasks by the processor. Figure 4.11 shows Win x Talk 1.5 showing the execution of tasks by the processor. The user can view the communication between software and hardware using Win x Talk 5 interface.

![Figure 4.11 Execution of tasks in Win x Talk 5 software](image)

**4.3.5 Scheduler Implementation with 8051**

Figure 4.12 shows the overall setup of the hardware connection to the pc via serial port. The inputs are given in the front end by the user. In figure 4.12 the front end shows that the task 2 is executing.
4.4 DEVELOPING HARDWARE BOARD USING ARM MICROCONTROLLER (LPC 2148)

The LPC2142/2148 microcontrollers are based on a 32/16-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with 64 kB and 512 kB of embedded high-speed flash memory. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty. Due to their tiny size and low power consumption, LPC2142/2148 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for realtime applications. ARM contains 32 bits with features like UART vector interrupt controller, etc, ARM is selected to implement in the research work. The block diagram for hardware implementation with ARM is shown in Figure 4.13.
4.4.1 Compiler

IAR Embedded Workbench integrated development environment (IDE) has been used. This IDE is used for the development cycle. The compiler and the linker are used to create the application for the ARM core. Figure 4.14 shows the IDE.
The hardware setup using ARM has been developed. Figure 4.15 shows the hardware setup with the simulator.

**Figure 4.15 Hardware LPC 2148 board**

### 4.4.2 Assigning of Tasks

<table>
<thead>
<tr>
<th>Task 1</th>
<th>FAN</th>
<th>The parameters for task 4 has been assigned by user in the front end pane.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 2</td>
<td>Buzzer</td>
<td>Based on the input value of task 2 from the front end and scheduling algorithm the buffer will sound.</td>
</tr>
<tr>
<td>Task 3</td>
<td>Switch</td>
<td>Contains the switch. The switch transmits the current based on the value given to task 5 by the user on the front end to the scheduler</td>
</tr>
<tr>
<td>Task 4</td>
<td>Bulb</td>
<td>The bulb in the hardware board is assigned to task 4.</td>
</tr>
<tr>
<td>Task 5</td>
<td>LED</td>
<td>The value of task 5 input is given in the front end by the user.</td>
</tr>
</tbody>
</table>
```c
void task5(int v1)
{
    char *ptr;
    bulb=0;
    led=0;
    singlecommand(0xc0); lcddisplay("task - 5 ");
    numberofdatastx(" Task5 ");
    ptr=&array[8];
    ptr++;
    ptr++;
    led=1;
    on_led_time(*ptr,v1);
    led=0;
    array[8]=99;
    array[9]=99;
    newline();
}
```

**Figure 4.16 LED Task 5 Source code ARM**

In Figure 4.16 program the, a pointer is declared with array. LCD will display task 5 in its screen. If LED is 1 gets allocated. If the LED is not initialized the command will transfer to the new line and Figure 4.17 shows the sample code.
void check_fcfs()
{
char i, j, kk, wh, t, lop=0, yes=0;
char st_array[15];
char array1[8];
led=0;
strcpy(st_array, array);

numberofdatastx("schedule - FCFS");  // scheduling
newline();
sec=0;
do{
    wh=0;
    yes=0;
    for(i=0; i<10; i++)
        array1[i]=99;
    j=0;
    for (i=0; i<strlen(array);)
    {
        if(sec>array[i])
        {
            yes=1;
            array1[j]=i;
            j++;
        }
        i=i+2;
    }
}while(yes=1);

terminate;

void check_sjf()
{

Figure 4.17 (Continued)
char i, j, kk, wh, t, lop=0, yes=0;
char st_array[15];
char array1[8];
strcpy(st_array, array);
numberofdatastx("schedule - SFJ");
newline();
task_check(wh);
lop++;
}while(lop<((strlen(array))/2));
strcpy(array, st_array);

Figure 4.17 Sample code for hardware using ARM

4.4.3 Implementation with Arm LPC 2148

Figure 4.18 shows the overall setup. The hardware is connected to the pc via serial port. The inputs are given in the front end by the user

Figure 4.18 Overall setup using ARM
4.5 ADVANTAGES OF PROPOSED SIMULATOR

- Platform independent access and use of the simulator for learning.

- User-friendly interface that requires minimal training / re-training.

- Users will be able to access only the latest implementation of the simulator with no ambiguity of versions of the application.

- Ease of maintenance from a Programming / maintenance group perspective and run Simulation to view the chronogram (timing diagram) and understand the way the tasks are scheduled in real-time using the selected scheduling policy.

4.6 CONCLUSION

The hardware setup for both 8051 and ARM have been implemented and integrated with software in this chapter. Future works involve development of virtual hardware using virtual reality concepts. There is also a possibility of integrating the sensors into the hardware and getting input from hardware while the graph is displayed on the simulator.