DEVELOPMENT OF EMBEDDED SYSTEM APPLICATIONS IN LAB ENVIRONMENT: PROGRAMMING IN "C"

P.Thimmaiah
Asst.Professor, Dept. of Electronics, Sri Krishnadevaraya University, Anantapuramu, AP, India

Abstract: This paper describes the Development of embedded systems in 8051 Lab environment using 8051 microcontroller interrupts, programming in "C"and applications. These applications include an interfacing of seven segment display, Multi-segment display and LCD and these are controlled by external interrupts (INT0, INT1) signals through switches. In this experiment AT89C51 8-bit microcontroller is used to program the interrupts to seven segment display (up/down hexa decimal counter), Multi-segment display (2-digit decimal counter for up/down counter) and LCD. The Hardware description and software developments are presented in this paper. This paper is very useful for beginners like Engineering and PG students.

Keywords: Interrupts, seven segment display, multi-segment display, LCD and AT89C51 Microcontroller

1. Introduction

The Microcontroller can serve several devices. Interrupt is one of the most important and powerful concepts and features in microcontroller/processor applications [1]. Almost all the real world and real time systems built around microcontrollers and microprocessors make use of interrupts. The interrupts refer to a notification, communicated to the controller, by a hardware device or software, on receipt of which controller momentarily stops and responds to the interrupt. Whenever an interrupt occurs the controller completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a piece of code that tells the processor or controller what to do when the interrupt occurs. After the execution of ISR, controller returns back to the instruction it has jumped from (before the interrupt was received). The interrupts can be either hardware interrupts or software interrupts. If the interrupts are generated by the controller's in built devices, like timer interrupts, or by the interfaced devices, they are called the hardware interrupts. If the interrupts are generated by a piece of code, they are termed as software interrupts.

In polling, the microcontroller keeps checking the status of other devices, and while doing so it does no other operation and consumes all its processing time for monitoring [2]. This problem can be addressed by using interrupts. In interrupt method, the controller responds to only when an interruption occurs. Thus in interrupt method, controller is not required to regularly monitor the status (flags, signals etc.) of interfaced and inbuilt devices. When an interrupt is invoked, the microcontroller runs the interrupt service routine. For every interrupt, there is a fixed location set aside to hold the addresses of ISRs as shown below.

Interrupt	ROM Location
RESET	0000H
INT0 (EX0)	0003H
TIMER0 (TF0)	000BH
INT1 (EX1)	0013H
TIMER1 (TF1)	001BH
SERIAL COMMUNICATION INTERRUPTS (TI & RI)	0023H

The 8051 controller has six hardware interrupts of which five are available to the programmer. These are as follows:

A. RESET Interrupts

This is also known as Power on Reset (POR). When the RESET interrupt is received, the controller restarts executing code from 0000H location. This is an interrupt which is not available to or, better to say, need not be available to the programmer.

B. Timer Interrupts (TF0 & TF1)

Each Timer is associated with a Timer interrupt. A timer interrupt notifies the microcontroller that the corresponding Timer has finished counting.

C. External Interrupts (INT0&INT1)

There are two external interrupts EX0 and EX1 to serve external devices. Both these interrupts are active low. In AT89C51, P3.2 (INT0) and P3.3 (INT1) pins are available for external interrupts 0 and 1 respectively. An external interrupt notifies the microcontroller that an external device needs its service.

D. Serial Interrupts (TI & RI)

This interrupt is used for serial communication. When enabled, it notifies the controller whether a byte has been received or transmitted.

Interrupt Enable Register (EA):

D7	D6	D5	D4	D3	D2	D1	D0
EA	-	-	ES	ET1	EX1	ET0	EX0

Programming External Interrupts

The external interrupts are the interrupts received from the (external) devices interfaced with the microcontroller. They are received at INTx pins of the controller. These can be level triggered or edge triggered. In level triggered, interrupt is enabled for a low at INTx pin; while in case of edge triggering, interrupt is enabled for a high to low transition at INTx pin. The edge or level trigger is decided by the TCON register.

TCON (Timer/Counter) Register (Bit-addressable)

D7	D6	D5	D4	D3	D2	D1	D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Setting the IT0 and IT1 bits make the external interrupt 0 and 1 edge triggered respectively. By default these bits are cleared and so external interrupt is level triggered.

1.1 Seven segment display

Seven segment displays are used to indicate numerical information. Seven segments display can display digits from 0 to 9 and even we can display few characters like A, b, C, H, E, e, F, etc. These are very popular and have many more applications. Seven segment displays internally consist of 8 LEDs. In these LEDs, 7 LEDs are used to indicate the digits 0 to 9 and single LED is used for indicating decimal point. Generally seven segments are two types, one is common cathode and the other is common anode.
In common cathode, all the cathodes of LEDs are tied together and labeled as com. and the anode are left alone. In common anode, seven segment display all the anodes are tied together and cathodes are left freely. Figure-1 shows the internal connections of Common anode and cathode seven segment Display.

Common Cathode 7-segment Display

In this type of display, all the cathode connections of the LED segments are connected together to logic 0 or ground. The separate segments are lightened by applying the logic 1 or HIGH signal through a current limiting resistor to forward bias the individual anode terminals a to g.

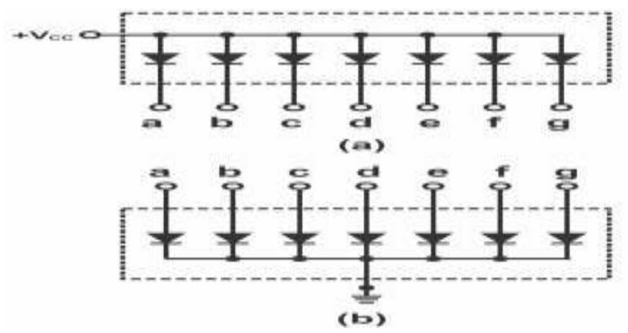


Figure 1: (a) Internal connection of Common anode type
(b) Internal connection of Common Cathode type

Common Anode 7-segment Display

In this type of display, all the anode connections of the LED segments are connected together to logic 1. The separate segments are lightened by applying of the logic 0 or LOW signal through a current limiting resistor to the cathode of the particular segment a to g. Below table shows the binary/hex values for displaying the digits on Common Anode seven segment display.

Table 1: Hex codes for digits (Common anode type)

Digit	dp	g	f	e	d	c	b	a	Hex code		Digit	dp	g	f	e	d	c	b	a	Hex code
0	1	1	0	0	0	0	0	0	C0		8	1	0	0	0	0	0	0	0	80
1	1	1	1	1	1	0	0	1	F9		9	1	0	0	1	1	0	0	0	98
2	1	0	1	0	0	1	0	0	A4		A	1	0	0	0	1	0	0	0	88
3	1	0	1	1	0	0	0	0	B0		b	1	0	0	0	0	0	1	1	83
4	1	0	0	1	1	0	0	1	99		C	1	1	0	0	0	1	1	0	C6
5	1	0	0	1	0	0	1	0	92		d	1	0	1	0	0	0	0	1	A1
6	1	0	0	0	0	0	1	0	82		E	1	0	0	0	0	1	1	0	86
7	1	1	1	1	1	0	0	0	F8		F	1	0	0	0	1	1	1	0	8E

2. Procedure

Click Keil µ Vision4 icon on the desktop. From the project menu, click on new project, create a new folder and give the project name. Identify the name on the controller which is given to you and Select the target device as Atmel 89c51/ Nuvoton / P89V51Rd2.Click “yes” at popup window named as standard 8051 start-up code. From file menu open new file. Type the program in the text editor and save as the program from the file menu with the extension “.asm” or “.c”. In the project window click the tree showing TARGET. From target right click on the folder named as source group1 and click `add files to source group. At the popup window select the saved program file with extension “.asm” or “.c” and add it. Click on the Translate (Ctrl+F7) button. Check if there are any errors. If so correct them and again click Translate. Repeat this process until there are no errors. Click on the build (F7). From the Debug menu click on start/stop (Ctrl+F5).In the project window the status of all the registers will be displayed. Click on the reset button labelled as left Arrow mark (Red coloured).Click on the one step button (F11) for executing the program line by line (Single Step) until the desired result displayed in the registers. Stop the debug process

2.1 Programming the microcontroller

After completion of the above procedure, right click on the target folder in the project window and select options for target (Alt+F7). At the popup window select the tab named as output and check the tick mark at create **HEX file** and click ok. Click on translate, build and rebuild buttons. Connect your target board along with controller to PC and follow the below procedure for your given controller.

For Atmel 89c51: Open WLPRO on the desktop and check if it is online mode or not at the bottom right. If so go to device menu and click on select device (Ctrl+D). Select your controller form the list and click ok. Click on read button from left side panel after reading it shows read completed at right side window. From the file menu click load file (Ctrl+L) and select your hex file which is created at your folder. Click on auto on the left side panel. It shows the messages at right side window as read completed, erase completed, program successful. Verify successful, protect completed. Take out your controller and put on the bread board and test the embedded system which is connected by you.

3. Embedded System Applications

3.1 Embedded system for seven segment display

Common anode type 7-segment display is connected to port-2 of Microcontroller. Code is developed for hexa decimal counter. When we press the INTO switch (S), it will count from 0 to F and F to 0 when we press the INT1 (S1) switch with predefined delay.

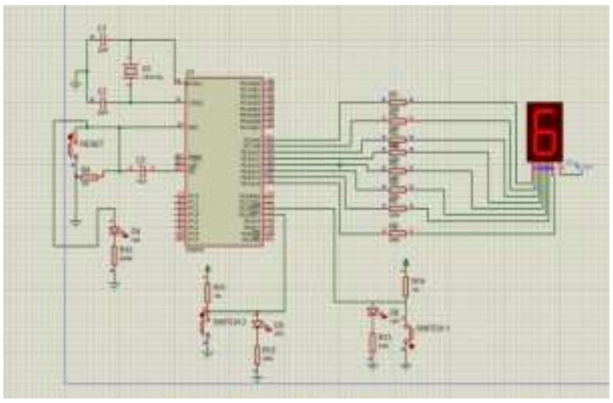
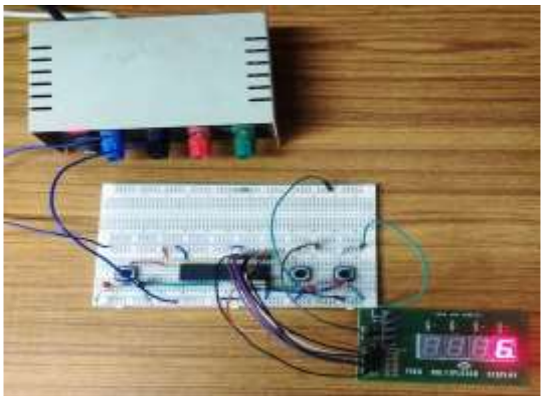



Figure 2: (a) Interfacing Circuit Diagram (b) Embedded system for Seven segment display

//This is the program for seven segment display to display 0-Fand F-0 digits//

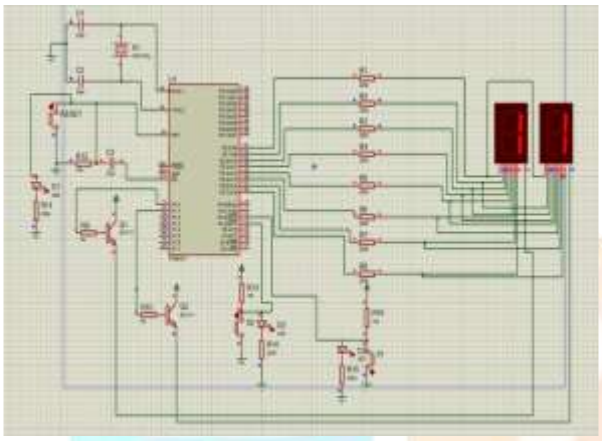
```
    #include<reg51.h>
     void delay();
     sbit S=P3^2;
     sbit S1=P3^3;
     void interrupt0();
     void interrupt1();
     void main()
     {
        int SW,SW1;
      IE=0x85;
        IT0=1;
        IT1=1;
        SW=S;
        SW1=S1;
        if(SW==0)
        {
           interrupt0();
       }
         if(SW1==0)
         {
          interrupt1();
         }
      }

     void interrupt0()
     {
       int v,t;
       int
ca[16]={0x0c0,0x0f9,0x0a4,0x0b0,
                  0x99,0x92,0x82,0xf8,0x80,
                    0x98, 0x88, 0x83,
0x0c6,
                    0x0a1, 0x86, 0x8e};
      for(v=0;v<15;v++)
      {
```

```
void interrupt1()
{
     int v,t;
     int
ca[16]={0x0c0,0x0f9,0x0a4,0x0b0,

0x99,0x92,0x82,0xf8,0x80,
                 0x98, 0x88, 0x83,
0x0c6,
                 0x0a1, 0x86, 0x8e};
     for(v=15;v>=0;v--)
     {
          for(t=0;t<20;t++)
          {
            P2=ca[v];
            delay();
          }
     }
 }

void delay()
 {
    int i;
    for(i=0;i<5;i++)
    {
      TMOD=0x01;
      TL0=0x055;
      TR0=1;
      while(TF0==0);
      TR0=0;
      TF0=0;
    }
  }
```

```
            for(t=0;t<20;t++)
            {
              P2=ca[v];
               delay();
             }
         }
      }
```

3.2 Embedded system for Multi segment display

Multi segment display is connected to port-2 of Microcontroller. Code is developed for 2-digit decimal counter. When we press the INTO switch (S), it will counts from 00 to 99 and 99 to 00 when we press the INT1 (S1) switch with sufficient delay. 2-digits of seven segments can be selected by connecting them to P1.0 and P1.2



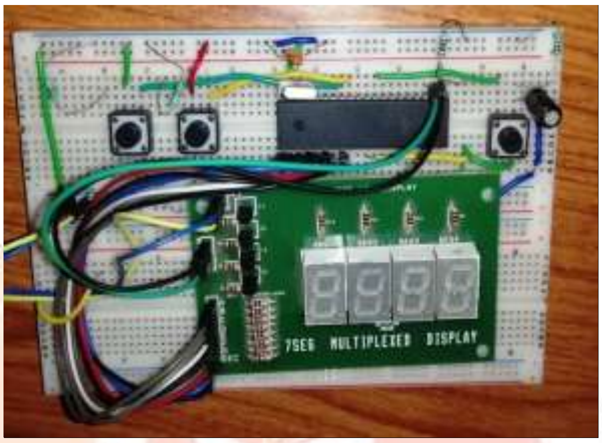


Figure 3: (a) Interfacing Circuit Diagram (b) Embedded system for of Multi segment display

//This is the program for multi segment to display 00-99 & 99-00 decimal numbers.//

```
#include<reg51.h>
void delay();
sbit s=P3^2;
sbit s1=P3^3;
void interrupt0();
void interrupt1();
void main()
{
  int SW1,SW;
  IE=0x85;
  IT0=1;
  IT1=1;
  SW=s;
  SW1=s1;
  if(SW==0)
 {
  interrupt0();
  }
  if(SW1==0)
 {
  interrupt1();
 }
}

void delay()
{
    int i;
   for(i=0;i<2000;i++)
    {
    }
}
```

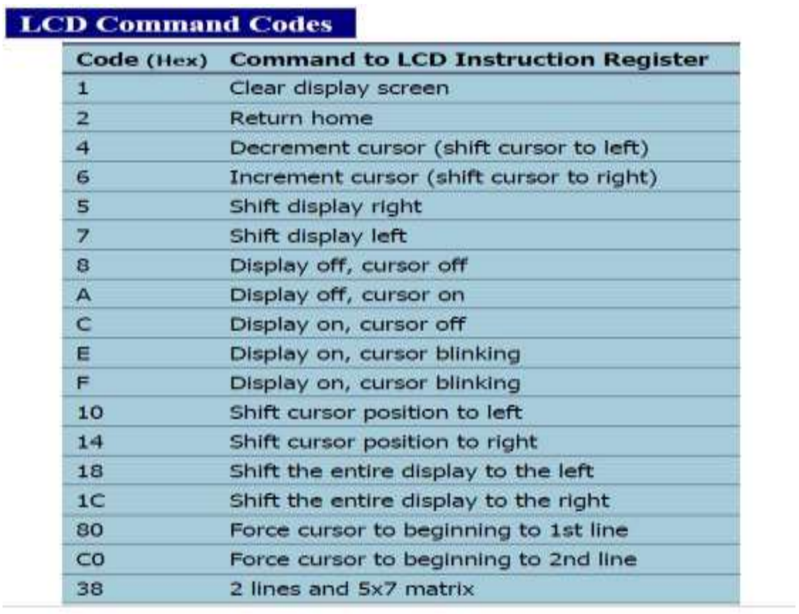
```
void interrupt0()
{
   int v1,v2,t
   int
ca[10]={0x0c0,0x0f9,0x0a4,

0x0b0,0x99,0x92,0x82,

0xf8,0x080,0x98};
   P1=0x00;
   for(v1=0;v1<10;v1++)
  {
     for (v2=0;v2<10;v2++)
     {
          for(t=0;t<20;t++)
          {
              P1=0x01;
              P2=ca[v1];
              delay();
              P1= 0x02;
              P2= ca[v2];
             delay();
          }
      }
   }
}
```

```
void interrupt1()
{
   int v1,v2,t
   int
ca[10]={0x0c0,0x0f9,0x0a4,

0x0b0,0x99,0x92,0x82,

0xf8,0x080,0x98};
   P1=0x00;
   for(v1=9;v1>=0;v1--)
   {
     for(v2=9;v2>=0;v2--)
       {
         for(t=0;t<20;t++)
         {
             P1=0x01;
             P2=ca[v1];
             delay();
             P1=0x02;
             P2=ca[v2];
            delay();
         }
      }
   }
}
```

3.3 Embedded system for Liquid crystal Display (LCD)

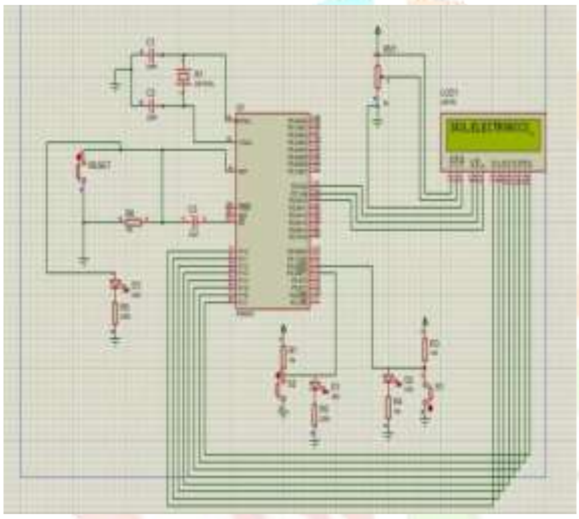
Liquid Crystal Display also called as LCD is very helpful in providing user interface. LCD module is a very common type of LCD module that is used in 8051 based embedded projects. These LCD's are very simple to interface with the controller as well as are cost effective. The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required. When RS is low (0), the data is to be treated as a command. When RS is high (1), the data being sent is considered as text data which should be displayed on the screen. When R/W is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively reading from the LCD. Most of the times there is no need to read from the LCD so this line can directly be connected to Gnd thus saving one controller line. The ENABLE pin is used to latch the data present on the data pins. A HIGH - LOW signal is required to latch the data. The LCD interprets and executes our command at the instant the EN line is brought low The various LCD commands are listed in table-2.

Table 2: LCD Commands List



LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix



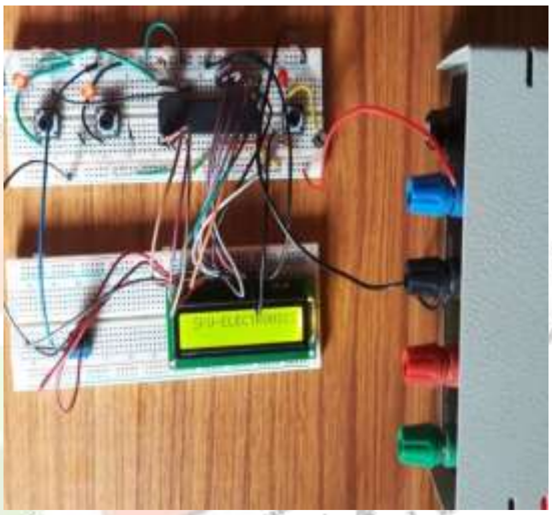


Figure 4: (a) Interfacing Circuit Diagram (b) Embedded system for LCD

// This is the program in "c" for LCD display//

```
#include<reg51.h>
void delay();
void lcdcmd(unsigned char value);
void lcddata(unsigned char value);
SFR P1 data=0xA0;
sbit rs=P2^0;
sbit rw=P2^1;
sbit en=P2^2;

void main()
{
  lcdcmd(0x38);
  delay();
  lcdcmd(0x0E);
  delay();
  lcdcmd(0x01);
  delay();
```

```
void lcdcmd(unsigned char value)
{
        P1 = value;
        rs = 0;
        rw = 0;
        en = 1;
        delay();
        en =0;
        return;
}

void lcddata(unsigned char value)
{
         P1 = value;
         rs = 1;
         rw = 0;
        lcdcmd(0x06);
        delay();
        lcdcmd(0xC0);
        delay();
        lcddata('S');
        delay();
        lcddata('K');
        delay();
        lcddata('U');
        delay();
        lcddata('_');
        delay();
        lcddata('E');
        delay();
        lcddata('L');
        delay();
        lcddata('E');
        delay();
        lcddata('C');
        delay();
        lcddata('T');
        delay();
        lcddata('R');
        delay();
        lcddata('O');
        delay();
        lcddata('N');
        delay();
        lcddata('I');
        delay();
        lcddata('C');
        delay();
        lcddata('S');
        delay();
    }
```

```
        en = 1;
        delay();
        en =0;
        return;
}

void delay()
{
        int i;
        for(i=0;i<20;i++)
        {
          TMOD = 0x01;
          TL0= 0xFD;
           TH0 = 0x4B;
           TR0 = 1;
          while(TF0==0);
          TR0=0;
           TF0=0;
          }
}
```

4. Conclusion

In this paper, Embedded systems for Seven segment display , Multi segment and LCD are developed around Atmel 89c51 microntroller using external interrupts and programmed using ‘C” language. All the systems were tested succefully .These are controlled by external interrupt (INT0, INT1) signals through switches have been presented.

References

[1] The 8051Microcontroller and Embedded systems- Muhammad Ali Mazidi, PHI Publishers.

[2] “Embedded Systems - Architecture, Programming and Design”, Raj Kamal, Publs. McGraw-Hill Education

[3] P.Thimmaiah, K.R.Rao, E.R.Gopal,“An inexpensive stepper motor interface for microcontrollers”, Lab Experiments, Vol. 3, pp. 185-191, Sep-2003.

[4] P.Thimmaiah, K.R.Rao, E.R.Gopal,“Interfacing Bipolar stepper motor with Microcontrollers”, J.Instrum.Soc.India.34 (3), pp. 163-169, 2004.

[5] P.Thimmaiah, K.R.Rao, E.R.Gopal,“Design and development of an embedded system for web camera control”, J.Instrum. Soc.India. 36 (2), pp. 125-132, 2006.

[6] Gade S S,Kanase A B,Shendge S B, Uplane M D.,“Serial Communication Protocol For Embedded Application”, International Journal of Information Technology and Knowledge Management, July-December 2010, Vol. 2, No.2, pp. 461-463.

[7] P.Thimmaiah “Concepts of 8051 Microcontroller External Interrupts: Programing and Applications”, Volume 1, Issue 1, September-October 2013, PP: 01-06, International Journal of Innovative Research Engineering & Multidisciplinary Physical Sciences (IJIRMPS).