Hi Friends,

Welcome to my AVR tutorial Series. In this article I will describe my Infrared Remote Control Library. Hope you will enjoy it.

I was very much amazed my remote controls since long. The simple circuits described in magazine were just operated like a switch and can only switch on and off an application. That means only single channel. That was of not much use. What I wanted was access to each key on remote control. I wished I could decode the signals generated by common remotes controls found in our homes. This way I could make multi channel remote controls for any project. So I went on and after some web research, coding and debugging I finally succeeded !

Actually I completed that more than one and half year ago (in may 2007). And now I thought I should also make it available to others who are in need of it so that they can use it in their projects. So I went on to make it more "clean" and also to make it more "easily portable" so that it can be used with different devices operating at different frequency. And now the result is here, a very easy to use library that can add remote control feature in any of your AVR project !

Using a IR remote has many advantages over using push buttons, like

1. Less I/O pins used. You only need to engage a single MCU pin and you get over 50 input buttons !!!
2. A typical DVD player remote costs only Rs40 to Rs 80 in India ($1 to $2 !!!). Compare with this the cost of using push buttons soldered on verobord.
3. At last its Wireless !!!

But the only problem is the "difficulty" involved but you don't need to care for that because you can use my library free of cost !

**Specifications**

This remote control decoding library decodes the remote signal encoded with NEC format which is also known as Japanese format also. This format is very popular in India I have tested it with my old "BPL" TV remote, Intex DVD player remote and other DVD player remotes that are available in electronic shops, all worked very fine. But there are other remote control formats too. Like RC5 which I think should be popular in US. My HP Pavilion Laptops remote control does not work with this library. It is because it uses RC6 encoding (I think so). So if this library does not work with your remote control it must be using other encodation technique.

**Hardware Setup**

**The Sensor**

This library requires minimum external hardware. To test the library I have also connected a 16x2 character LCD module which will be used for displaying the decoded value of the key press. The LCD is only required for testing and to note down the key code of the various keys on remote.

|  |
| --- |
| Connecting a TSOP1738 with AVR MCU |
| **Fig - Connecting the sensor.** |
|  |

|  |
| --- |
| TSOP1738 infrared sensor module interface with AVR |
| **Fig - A real TSOP1738 IR sensor module.** |
|  |

**The LCD Module**

We connect a LCD module to the AVR to run the test program. The display will be used to display the key code received from the remote. This demo will also help you know the key code of the various keys on your remote control. To learn more about LCD interface with AVRs [see this.](http://extremeelectronics.co.in/avr-tutorials/using-lcd-module-with-avrs/) The connection is different for ATmega8 and ATmega16/32 so I am giving both below.

|  |
| --- |
| http://extremeelectronics.co.in/avrtutorials/images/ir_lcd.GIF |
| **Fig - LCD module connection for ATmega8** |
|  |

|  |
| --- |
| http://extremeelectronics.co.in/avrtutorials/images/ir_lcd_M16.GIF |
| **Fig - LCD module connection for ATmega16 Or ATmega32** |
|  |

You can change the PINs that are used for interfacing the LCD with the MCU. [For more information on how to do that see this](http://extremeelectronics.co.in/avr-tutorials/using-lcd-module-with-avrs/). The N/C pins are not connected anywhere. The PIN names are printed on the back side of the LCD module. The project can be prototyped with any development board. Some easy to use development boards are.

1. [Home Made AVR Devboard](http://extremeelectronics.co.in/hardwares/home-made-avr-devboard/)
2. [xBoard™](http://xBoard.extremeelectronics.co.in)
3. [xBoard™ MINI](http://extremeelectronics.co.in/tools/xboard-mini/)

**Software Setup**

To make the library usable with different AVR MCU's efficiently I have two different version of the library one for ATmega8 and other for ATmega16 or 32. The library requires some accurate timing and thus cannot be used with any random clock frequency. But I have done my best to make it work with common clock frequency by using some c preprocessor tricks and conditional compilation. **The supported frequencies are 8MHz,12Mhz and 16MHz**. So before compiling don't forget to supply the F\_CPU in either the makefile or AVR Studio Project. In AVR Studio go to Project->Configuration option and set the frequency.

Welcome back. In [previous tutorial](http://extremeelectronics.co.in/code-libraries/using-ir-remote-with-avr-mcus/) I introduced my IR remote decoding library. In this tutorial I will continue our discussion and show you how to add IR remote controls support to your AVR projects.

**Step I**

[Download the library files](http://www.extremeelectronics.co.in/avrtutorials/download/IR.zip) and unzip them in a folder.

**Step II**

In AVR studio create a [new AVR-GCC project](http://extremeelectronics.co.in/avr-tutorials/part-iv-the-hello-world-project/). Then **copy** the following files to the project folder.

**IR remote related.**

**Source Files**

remote.c

**Header Files**

remote.h

rckeys.h

**LCD Related**

Source Files

lcd.c

Header Files

lcd.h

myutils.h

|  |
| --- |
| ***Note: Include file from the***  ***\lib\ATmega8 if you are using ATmega8***  ***\lib\ATmega16 if you are using ATmega16 or ATmega32*** |

After The files have been copied add them to your project by right clicking project view and selecting “Add Existing Source File(s)…” and then select the “lcd.c”. Similarly add other source files. In the same way add the header files by selecting "Add Existing Header File(s)..."

|  |
| --- |
| avr studio |
| **Adding files to projects.** |

If instead of AVR Studio you use Makefiles then add all the source files **names** in the source section of the makefile along with your main c file (which has same name as your project and has the main function).

**Step III.**

In your main C source file include the following files

#include "remote.h"

in addition to standard files

in your main() function initialize Remote Subsystem by calling

RemoteInit();

And your done with initialization.

**Step IV**

Once initialized the remote subsystem automatically receives command sent by remote and store them in a circular FIFO(first in first out queue). You can call the function **GetRemoteCmd()** to read from the buffer. The return type is a unsigned char (8bit) so the value can be from 0-255. This is the keycode of the key pressed **on the remote .** To know the key code of various keys simply run this example on MCU and press keys the key code will be displayed on the LCD. The GetRemoteCmd takes one argument which can be either 0 or 1. When you pass 1 the function checks the queue for command and if there none then it wait till a key is pressed on the remote control. **It returns only if a key is pressed and return its keycode.** In contrast to this if you pass 0 the function returns immediately if no key is pressed it returns a special constant RC\_NONE which is defined as 255 in the header file. Otherwise it returns the key code of pressed key.

To make every thing more clear I am giving the complete sample program which will just wait for keypress and then display its code.

**Sample Code**

*/\*#########################################################*

*Demo program for testing the IR Remote control libraries.*

*-----------------------------------------------------------*

*Copyright Avinash Gupta 2008 avinash@eXtremeElectronics.co.in*

*Web: www.eXtremeElectronics.co.in*

*Please give credit or reference to www.eXtremeElectronics.co.in*

*if you use this in your projects and find it useful.*

*##########################################################\*/*

#include <avr/io.h>

#include "remote.h"

#include "lcd.h"

**void** main()

{

uint8\_t cmd=0;

RemoteInit();

LCDInit(LS\_BLINK);

LCDClear();

LCDWriteString("IR RemoteDecoder");

LCDWriteStringXY(0,1,"Key Code:");

**while**(1)

{

cmd=GetRemoteCmd(1);

LCDWriteIntXY(10,1,cmd,3);

}

}