USB IR Remote Receiver (AVR Multiprotocol Receiver) description

General description of the device

The USB IR Remote Receiver is based on Frank M. IRMP - Infrarot-Multiprotokoll-Decoder and V-USB.
The main motivation for this project was the outdated Igor Plug remote receiver. As Frank M. is descripting RC5 is obsolete I decided to combine Frank M. multiprotocol receiver with V-USB to be able to receive also other protocols through the USB port. Also newer motherboards do not always include a serial COM port.

The device is working as low speed USB 1.1 device. As it get recognized as HID device on Windows no special driver is needed (tested on Windows 7 x86 and WinXP SP3)

The USB IR Remote Receiver has also a PowerOn function to start/shutdown the host PC through the remote control. The first received IR code will be stored in EEPROM. If a received IR code
equals the stored code the device will switch on a output pin for ~250ms. (see AVR source description)

Supported software:
DVBViewer: copy 'USB_IR_Remote_Receiver.dll' to \DVBViewer\Plugins and enable the input plugin in the DVBViewer options
Girder: copy 'USB_IR_Remote_Receiver.dll' to \Girder\Plugins and enable the plugin in the Girder options. Girder versions since 3.2.9 are supported
Native: also the 'USB_IR_Remote_Receiver.dll' can be used for selfmade programs with the functions InitNative and InitPAnsiChar (see below)

Right now these IR protocols are supported by the included IRMP version from '16.04.2010':

Protokoll	Hersteller
SIRCS	Sony
NEC	NEC, Yamaha, Canon, Tevion, Harman/Kardon, Hitachi, JVC, Pioneer, Toshiba, Xoro, Orion, NoName und viele weitere japanische Hersteller.
SAMSUNG	Samsung
SAMSUNG32	Samsung
MATSUSHITA	Matsushita
KASEIKYO	Panasonic, Technics, Denon und andere japanische Hersteller, welche Mitglied der "Japan's Association for Electric Home Application" sind.
RECS80	Philips, Nokia, Thomson, Nordmende, Telefunken, Saba
RECS80EXT	Philips, Technisat, Thomson, Nordmende, Telefunken, Saba
RC5	Philips und andere europäische Hersteller
DENON	Denon
RC6	Philips und andere europäische Hersteller
APPLE	Apple
NUBERT	Nubert, z.B. Subwoofer System
B&O	Bang & Olufsen

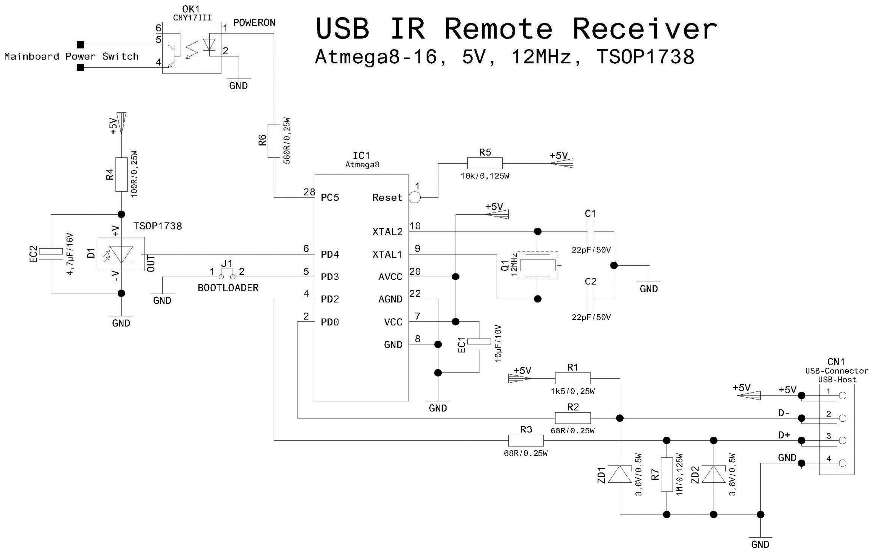
Change log

23.04.2010:
DLL: v1.0.0.4, Added Key Suppression, Bugfix of DVBViewer on showing settings/option dialog
AVR: optimized detection for PowerON IR code, Bugfix for more than 255 IR code repeats

16.04.2010:
DLL: v1.0.0.3, Added InitNative, InitPAnsiChar and Girder functions.
AVR: Updated Irmp to 16.04.2010

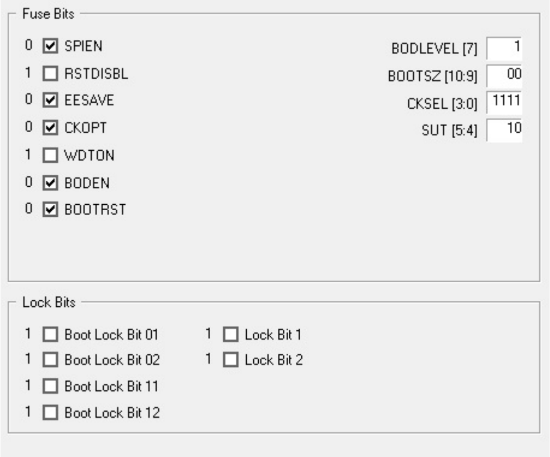
12.04.2010:
AVR: Updated Irmp to 12.04.2010, new protocol B&Q

Hardware schematic



The diodes ZD1 and ZD2 are needed to comply to the USB standard.
The IR receiver can be also one of this types:

Vishay TSOP 1738, Vishay TSOP 1838, Vishay TSOP 11xx series, Siemens SFH 5110, Radio Shack 276-0137, Mitsumi IR Preamp KEY-COOSV (0924G), Toshiba TK19 444 TFMS 5360, Temic TFMS 5380 by Telefunken Semiconductors, Sharp IS1U60, Everlight IRM-8601S, Sony SBX 1620-12, Sharp GP1U271R
Optional: OK1 if the PowerOn function is used to start/shutdown the host PC
Optional: J1 if a bootloader will be used: BootloadHID
If the bootloader get be used be sure to write the right fuse bits. Otherwise the bootloader will be deleted after the first flashing.



Fuse Bits

0 ☑ SPIEN
1 ☐ RSTDISBL
0 ☑ EESAVE
0 ☑ CKOPT
1 ☐ WDTON
0 ☑ BODEN
0 ☑ BOOTRST

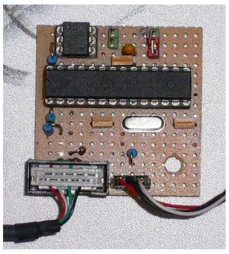
BODLEVEL [7] 1
BOOTSZ [10:9] 00
CKSEL [3:0] 1111
SUT [5:4] 10

Lock Bits

1 ☐ Boot Lock Bit 01
1 ☐ Boot Lock Bit 02
1 ☐ Boot Lock Bit 11
1 ☐ Boot Lock Bit 12
1 ☐ Lock Bit 1
1 ☐ Lock Bit 2

Also take look to the file *Makefile* of the project *bootloadHID*.

Picture of an assembled USB IR Remote Receiver device



AVR Source description

To compile the source Atmel AVR Studio and WinAVR will be needed.

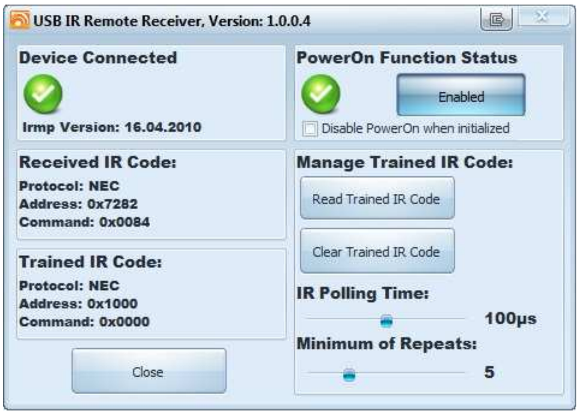
irmpconfig.h:
(pin can be changed if the IR receiver diode is not connected to PD4)

```
#define IRMP_PORT  PORTB
#define IRMP_DDR   DDRD
#define IRMP_PIN   PIND
#define IRMP_BIT   4       // use PD4 as IR input
```

configUSBIRRemoteReceiver.h:

```
//enter here the Irmp build date:
const char IrmpVersion[] = "16.04.2010";
#define USE_PowerOnFunction                                                        /* unkomen if you don't want to use the PowerOn function */

#ifdef USE_PowerOnFunction
#define SWITCH_PORT             PORTC                                              /* PORTx - register for Switch output */
#define SWITCH_BIT              PC5                                                /* bit where OK1 will be connected */
#define SWITCH_DDR              DDRC                                               /* Switch data direction register */
#endif
```

Screen shot settings/option dialog



USB IR Remote Receiver, Version: 1.0.0.4

Device Connected
Irmp Version: 16.04.2010

PowerOn Function Status
Enabled
☐ Disable PowerOn when initialized

Received IR Code:
Protocol: NEC
Address: 0x7282
Command: 0x0084

Manage Trained IR Code:
Read Trained IR Code
Clear Trained IR Code

Trained IR Code:
Protocol: NEC
Address: 0x1000
Command: 0x0000

IR Polling Time: 100µs
Minimum of Repeats: 5

Close

Device Connected: If the device got connected to the host the icon will change to green and the included Irmp build date will be shown.
PowerOn Function Status: If the PowerOn function is enabled the icon will be green. Use the button to enable/disable the PowerOn function. If the option "Disable PowerOn when initialized"
is checked the PowerOn function will be automatically disabled when the plugin get's initialized. So the IR code what is used for PowerOn can be used on the host

for a different function.
Received IR Code: Will show the last received IR code
Trained IR Code: Will show the IR code what got stored in EEPROM for PowerOn
Manage Trained IR Code:
- **Read Trained IR Code:** Reads the IR code stored in EEPROM
- **Clear Trained IR Code:** Will clear the actual stored IR code in EEPROM. The next received IR code will be stored as new PowerOn IR code.
- **IR Polling Time:** For different IR receiver diodes it is possible to adjust the polling time. A usefull time is between 66 - 100µs
- **Minimum of Repeats:** The device will debounce the remote control. With the minimum of repeats the number of needed repeats can be adjusted until the same IR code getting sent again to the host.
  A repeat of 0 means no change to the remote control.

USB_IR_Remote_Receiver.ini

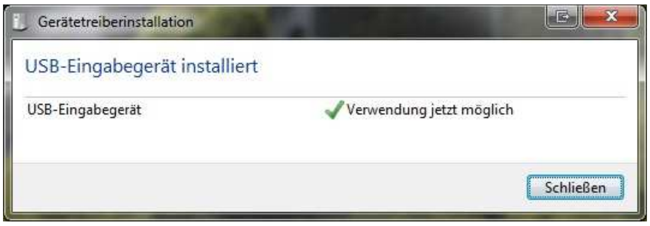
[IRMP Protocols]
List of supported protocols of the device (depend on IRMP version).
If a new protocol is added to the device the DLL doesn't need to be updated. Just add the new protocol in this section:
name_of_protocol=IRMP_ID

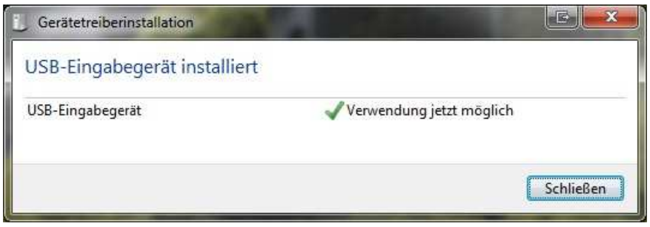
[Settings]
Default the device is opened delayed with 2000ms. This value can be changed when adding:
StartUpDelay=desired_value_in_milliseconds
{Default 2000}

Default there is no key suppression. The received IR codes are getting forwarded as they are getting decoded and received ba the DLL.
If the host software isn't fast enougth to handle fast repeating IR codes a timer for key suppresion can be used.
This option disallow a received IR to get sent to the host software if a previous received IR code got sent and the key suppression is still active.
Key Suppression=desired_value_in_milliseconds
{Default 0}

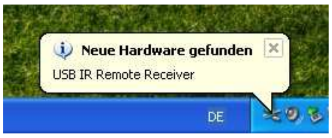
Add this value to enable a log memo in the settings dialog:
Debug=1
{Default 0}

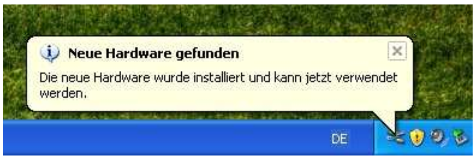
USB IR Remote Receiver device installation on Windows 7 x86, just plug in the device. The driver will be installed automatically


Gerätetreiberinstallation
USB-Eingabegerät installiert
USB-Eingabegerät ✓ Verwendung jetzt möglich


Gerätetreiberinstallation
USB-Eingabegerät installiert
USB-Eingabegerät ✓ Verwendung jetzt möglich

USB IR Remote Receiver device installation on Windows XP SP3, just plug in the device. The driver will be installed automatically


Neue Hardware gefunden
USB IR Remote Receiver


Neue Hardware gefunden
Die neue Hardware wurde installiert und kann jetzt verwendet werden.

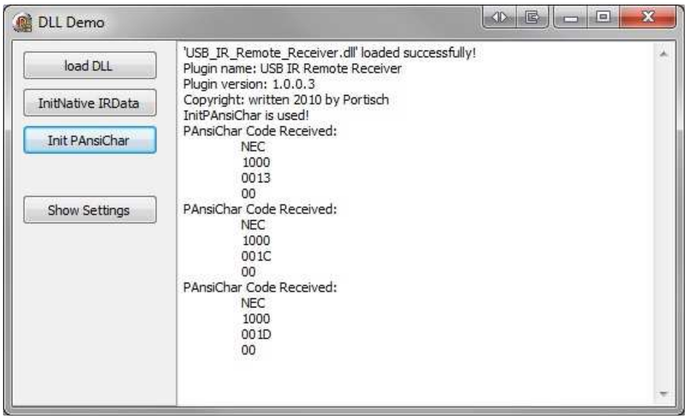
Function prototypes in "USB IR Remote Receiver.dll" library:

The DLL will handle the communication between the HID AVR device and the host.
The USB IR Remote Receiver DLL can be used directly with DVBViewer in the '\Plugins' directory as input plugin.
The USB IR Remote Receiver DLL can be also used directly with Girder in the '\Plugins' directory. Tested with Girder 3.2.9 (last Freeware version), 4.0.5.2 and 5.0.10.

Also it can be used for other programs to handle the device with the functions InitNative or InitPAnsiChar.

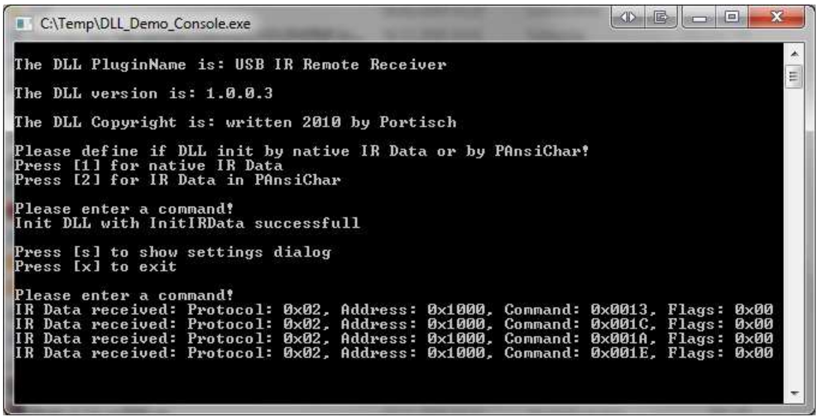
Screen shots from the demo sources

Example output with the DLL_Demo.exe (source Delphi 2010):

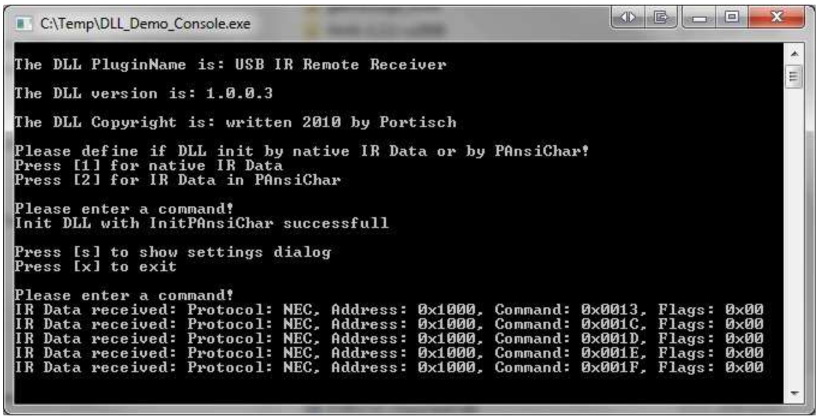


```
'USB_IR_Remote_Receiver.dll' loaded successfully!
Plugin name: USB IR Remote Receiver
Plugin version: 1.0.0.3
Copyright: written 2010 by Portisch
InitPAnsiChar is used!
PAnsiChar Code Received:
        NEC
        1000
        0013
        00
PAnsiChar Code Received:
        NEC
        1000
        001C
        00
PAnsiChar Code Received:
        NEC
        1000
        001D
        00
```

Example output with the DLL_Demo_Console.exe (source Visual Studio 2008):



```
The DLL PluginName is: USB IR Remote Receiver

The DLL version is: 1.0.0.3

The DLL Copyright is: written 2010 by Portisch

Please define if DLL init by native IR Data or by PAnsiChar!
Press [1] for native IR Data
Press [2] for IR Data in PAnsiChar

Please enter a command!
Init DLL with InitIRData successfull

Press [s] to show settings dialog
Press [x] to exit

Please enter a command!
IR Data received: Protocol: 0x02, Address: 0x1000, Command: 0x0013, Flags: 0x00
IR Data received: Protocol: 0x02, Address: 0x1000, Command: 0x001C, Flags: 0x00
IR Data received: Protocol: 0x02, Address: 0x1000, Command: 0x001A, Flags: 0x00
IR Data received: Protocol: 0x02, Address: 0x1000, Command: 0x001E, Flags: 0x00
```



```
The DLL PluginName is: USB IR Remote Receiver

The DLL version is: 1.0.0.3

The DLL Copyright is: written 2010 by Portisch

Please define if DLL init by native IR Data or by PAnsiChar!
Press [1] for native IR Data
Press [2] for IR Data in PAnsiChar

Please enter a command!
Init DLL with InitPAnsiChar successfull

Press [s] to show settings dialog
Press [x] to exit

Please enter a command!
IR Data received: Protocol: NEC, Address: 0x1000, Command: 0x0013, Flags: 0x00
IR Data received: Protocol: NEC, Address: 0x1000, Command: 0x001C, Flags: 0x00
IR Data received: Protocol: NEC, Address: 0x1000, Command: 0x001D, Flags: 0x00
IR Data received: Protocol: NEC, Address: 0x1000, Command: 0x001E, Flags: 0x00
IR Data received: Protocol: NEC, Address: 0x1000, Command: 0x001F, Flags: 0x00
```

Delphi:

```
function Version():PAnsiChar; stdcall external USBIRRemoteReceiver name 'Version';
function Copyright():PAnsiChar; stdcall external USBIRRemoteReceiver name 'Copyright';
function PluginName():PAnsiChar; stdcall external USBIRRemoteReceiver name 'PluginName';
procedure ShowSettings(Handle : THandle); stdcall external USBIRRemoteReceiver name 'ShowSettings';

function InitNative(Callback : TCallbackNativeIRData):boolean; stdcall external USBIRRemoteReceiver name 'InitNative';
function InitPAnsiChar(Callback : TCallbackPAnsiChar):boolean; stdcall external USBIRRemoteReceiver name 'InitPAnsiChar';

type
   TCallbackNativeIRData = procedure(IRCode : TIRData); stdcall;

type
   TCallbackPAnsiChar = procedure(Protocol, Address, Command, Flags : PAnsiChar); stdcall;

type
   TIRData = packed record
      Protocol : byte;
      Address : word;
      Command : word;
      Flags : byte;
   end;
```

C++ Builder / Microsoft Visual C++:

```
#define USBIRRemoteReceiver    "USB_IR_Remote_Receiver.dll";
```

```
const char * __stdcall Version();
const char * __stdcall Copyright();
const char * __stdcall PluginName();
void __stdcall ShowSettings( HWND Handle );

BOOL __stdcall InitNative( CallbackNativeIRData * Callback );
BOOL __stdcall InitPAnsiChar( CallbackPAnsiChar * Callback );

typedef void ( __stdcall * CallbackNativeIRData) ( IRMP_DATA IRCode );
typedef void ( __stdcall * CallbackPAnsiChar) ( char * Protocol , char * Address , char * Command , char * Flags  );

typedef struct
{
   byte    Protocol;
   word    Address;
   word    Command;
   byte    Flags;
} IRMP_DATA;
```

```
function Version():PAnsiChar; stdcall external USBIRRemoteReceiver name 'Version';
```

Function will return the plugin version as PAnsiChar.

Parameters

none

Return values

Function will return USB IR Remote Receiver DLL version as PAnsiChar.

```
function Copyright():PAnsiChar; stdcall external USBIRRemoteReceiver name 'Copyright';
```

Function to get copyright of USB IR Remote Receiver DLL.

Parameters

none

Return values

Function will return copyright as PAnsiChar.

```
function PluginName():PAnsiChar; stdcall external USBIRRemoteReceiver name 'PluginName';
```

Function to get USB IR Remote Receiver DLL plugin name.

Parameters

none

Return values

Function will return USB IR Remote Receiver DLL plugin name as PAnsiChar.

```
procedure ShowSettings(Handle : THandle); stdcall external USBIRRemoteReceiver name 'ShowSettings';
```

Procedure to show options/settings dialog of USB IR Remote Receiver DLL.

Parameters

Handle

[in] Window handle of the calling program. The dialog will be shown modal.

Notes:

The options/settings dialog will only be shown if the DLL got initialized first by *InitNative* / *InitPAnsiChar*.

```
function InitNative(Callback : TCallbackNativeIRData):boolean; stdcall external USBIRRemoteReceiver name 'InitNative';
```

Function to init USB IR Remote Receiver for native callback.

Parameters

Callback

[in] Procedure pointer of a procedure of type struct TCallbackNativeIRData.

Return values

If the function succeeds, the return value is *True*.
If the function fails, the return value is *False*.

Notes:

The device get only opend if the *Callback* procedure pointer isn't *NIL / NULL*.
For closing the device, unloading/disable the plugin call this function again with *Callback = NIL / NULL*.

The defined Callback procedure will receive the IR codes like the type struct TIRData.

```
function InitPAnsiChar(Callback : TCallbackPAnsiChar):boolean; stdcall external USBIRRemoteReceiver name 'InitPAnsiChar';
```

Function to init USB IR Remote Receiver for native callback.

Parameters

Callback

[in] Procedure pointer of a procedure of type struct TCallbackPAnsiChar.

Return values

If the function succeeds, the return value is *True*.
If the function fails, the return value is *False*.

Notes:

The device get only opend if the *Callback* procedure pointer isn't *NIL / NULL*.
For closing the device, unloading/disable the plugin call this function again with *Callback = NIL / NULL*.

The defined Callback procedure will receive the IR codes as *Protocol, Adress, Command* and *Flags* : **PAnsiChar**.

For more information see also the demo application.