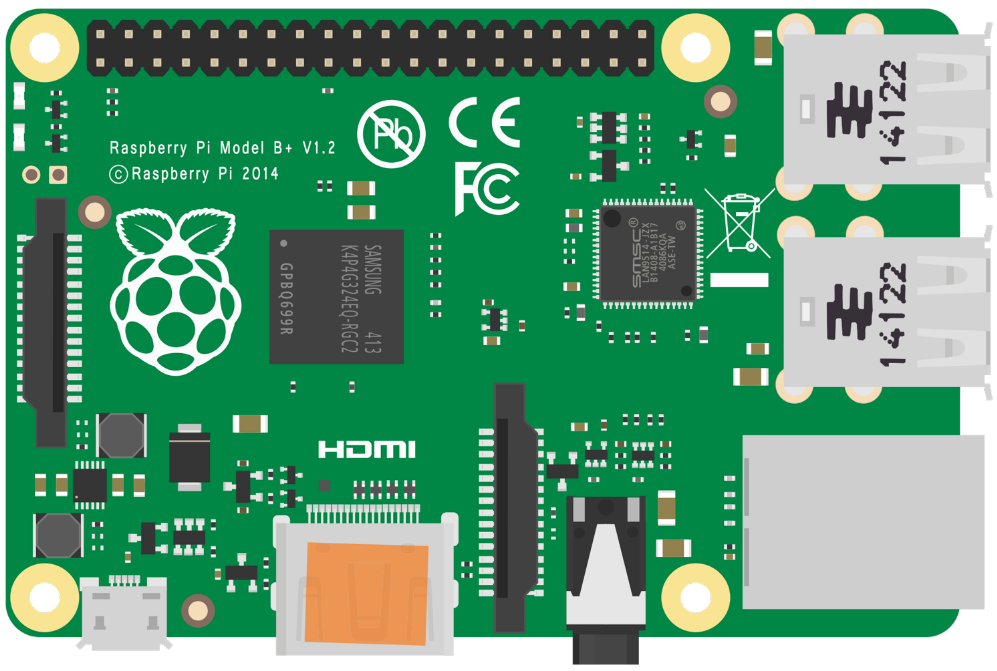


AVR, Raspberry Pi, VW Beta: VAG CDC Faker

Posted on September 8, 2013

I was tired connecting my cellphone with an old cassette adapter to my VW Golf with an Beta 5 in it. I needed something new, something cool. And here it is, a fully functional interface between the Beta 5 and any device you want to play the music with.

This project raised out of an ebay search: I just wanted to know if there is a solution to get an AUX input on the car radio. Yes there is, even complete mp3 players are available. But that's not what I wanted – at first!

Then I started digging the web for more info about the protocol the radio talks to cd changers. The idea was to build a tiny thingy that fakes a cd changer and simply enables the AUX input (as it is available in several online stores). But during the development my ambition became greater and I wanted to read the keys pressed on the radio to remote control my RPi.

1. Understanding the Protocol

First of all this is the pinout of the radio: rkieslinger.de/steckerbelegungen/vag-stecker.htm (http://rkieslinger.de/steckerbelegungen/vag-stecker.htm) The interesting cell is no. 3, the blue one. DATA IN simply is MOSI of an 8bit SPI interface with special timings between the bytes CLOCK is SCK of the SPI DATA OUT is the key code of the pressed key on the radio itself

The radio needs a sequence of bytes to enable the AUX input and display CD# and TR#. It looks likes this:

```
frame cd#   tr#   time  time  mode  frame frame
0x34, 0xBE, 0xFF, 0xFF, 0xFF, 0xFF, 0xCF, 0x3C
```

cd# and tr# are sent inverted. So this sequence will display: CD1 TR00 mode sets the playmode (PLAY|SHFFL|SCAN) As the beta doesn't support time display, I will ignore these bytes.

If cd mode is switched on/off or keys are pressed, 4 byte messages will be sent from the radio. They are coded like RC5 messages. Each message has a startsequence. A low byte has a short low phase, a high byte a longer low phase. Everything you need to know can be found in the pictures here: martinsuniverse.de/projekte/cdc_protokoll/cdc_protokoll.html (http://martinsuniverse.de/projekte/cdc_protokoll/cdc_protokoll.html)

2. Remote commands from the radio

As you know 4 byte messages are sent. The following pattern describes the control codes:

```
head head cmd  !cmd
0x53 0x2C 0xAA 0x55
```

Here is the complete list of all bytes from all keys:

```
switch on in cd mode/radio to cd (play)
0x53 0x2C 0xE4 0x1B
0x53 0x2C 0x14 0xEB

switch off in cd mode/cd to radio (pause)
0x53 0x2C 0x10 0xEF
0x53 0x2C 0x14 0xEB

next
0x53 0x2C 0xF8 0x7

prev
0x53 0x2C 0x78 0x87

seek next
0x53 0x2C 0xD8 0x27 hold down
0x53 0x2C 0xE4 0x1B release
0x53 0x2C 0x14 0xEB

seek prev
0x53 0x2C 0x58 0xA7 hold down
0x53 0x2C 0xE4 0x1B release
0x53 0x2C 0x14 0xEB

cd 1
0x53 0x2C 0x0C 0xF3
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

cd 2
0x53 0x2C 0x8C 0x73
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

cd 3
0x53 0x2C 0x4C 0xB3
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

cd 4
0x53 0x2C 0xCC 0x33
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

cd 5
0x53 0x2C 0x2C 0xD3
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

cd 6
0x53 0x2C 0xAC 0x53
0x53 0x2C 0x14 0xEB
0x53 0x2C 0x38 0xC7
send new cd no. to confirm change, else:
0x53 0x2C 0xE4 0x1B beep, no cd (same as play)
0x53 0x2C 0x14 0xEB

scan (in 'play', 'shffl' or 'scan' mode)
0x53 0x2C 0xA0 0x5F

shuffle in 'play' mode
0x53 0x2C 0x60 0x9F

shuffle in 'shffl' mode
0x53 0x2C 0x08 0xF7
0x53 0x2C 0x14 0xEB
```

The micro controller will only send the 3rd byte after error check over uart. See below.

3. The Hardware I picked

- Atmel ATmega32A
- 1117 Regulator (maybe not the best, it's heating up at >50mA due to the high drop voltage)
- FTDI FT230X as uart usb bridge
- ISO connector to fit into the radio

4. Programming the AVR

First thing about the uart communication. I use the library of Peter Fleury: homepage.hispeed.ch/peterfleury (http://homepage.hispeed.ch/peterfleury/)

To be independent from clockrates I utilized delay loops instead of timers to create correct timings. Hopefully my comments in the code explains everything very well.

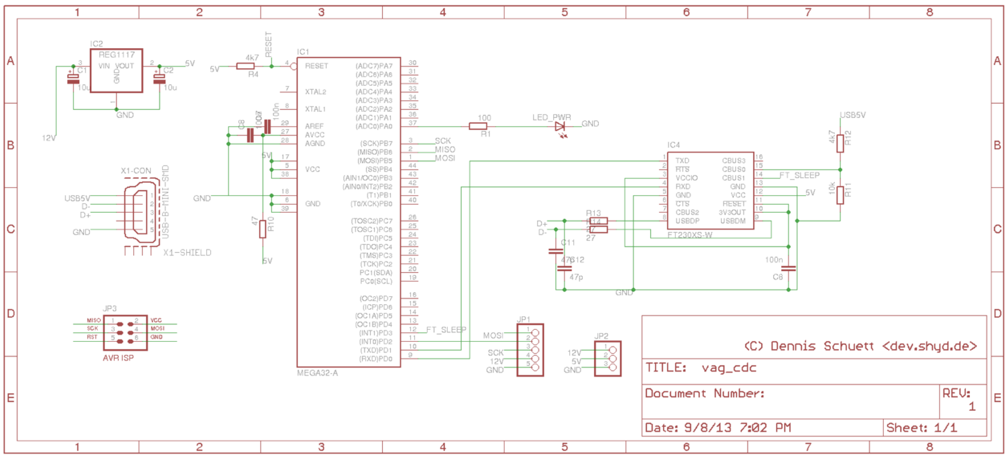
```
/*
 * VAG_CDC.c
 *
 * Created: 23.06.2013 20:00:51
 *  Author: Dennis Schuett, dev.shyd.de
 */

#define F_CPU 8000000UL

#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "uart.h"

#define UART_BAUD_RATE 9600
#define LED_PWR PA0
#define RADIO_OUT PD2
#define FT_CBUS1 PD3
#define RADIO_OUT_IS_HIGH (PIND & (1<<RADIO_OUT))

#define CDC_PREFIX1 0x53
#define CDC_PREFIX2 0x2C

#define CDC_END_CMD 0x14
#define CDC_PLAY 0xE4
#define CDC_STOP 0x10
#define CDC_NEXT 0xF8
#define CDC_PREV 0x78
#define CDC_SEEK_FWD 0xD8
#define CDC_SEEK_RWD 0x58
#define CDC_CD1 0x0C
#define CDC_CD2 0x8C
#define CDC_CD3 0x4C
#define CDC_CD4 0xCC
#define CDC_CD5 0x2C
#define CDC_CD6 0xAC
#define CDC_SCAN 0xA0
#define CDC_SFL 0x60
#define CDC_PLAY_NORMAL 0x08

#define MODE_PLAY 0xFF
#define MODE_SHFFL 0x55
#define MODE_SCAN 0x00

uint16_t captimehi = 0;
uint16_t captimelo = 0;
uint8_t capturingstart = 0;
uint8_t capturingbytes = 0;
uint32_t cmd = 0;
uint8_t cmdbit = 0;
uint8_t newcmd = 0;
uint8_t shutdownpending = 0;

uint8_t getCommand(uint32_t cmd);
void shutdown();
void softreset();

volatile uint8_t cd;
volatile uint8_t tr;
volatile uint8_t mode;

ISR(INT0_vect) //remote signals
{
        if(RADIO_OUT_IS_HIGH)
        {
                if (capturingstart || capturingbytes)
                {
                        captimelo = TCNT1;
                }
                else
                        capturingstart = 1;
                TCNT1 = 0;

                //eval times
                if (captimehi > 8300 && captimelo > 3500)
                {
                        capturingstart = 0;
                        capturingbytes = 1;
                        //uart_puts("startseq found\r\n");
                }
                else if(capturingbytes && captimelo > 1500)
                {
                        //uart_puts("bit 1\r\n");
                        cmd = (cmd<<1) | 0x00000001;
                        cmdbit++;
                }
                else if (capturingbytes && captimelo > 500)
                {
                        //uart_puts("bit 0\r\n");
                        cmd = (cmd<<1);
                        cmdbit++;
                }
                else
                {
                        //uart_puts("nothing found\r\n");
                }
                if(cmdbit == 32)
                {
                        //uart_puts("new cmd\r\n");
                        newcmd = 1;
                        cmdbit = 0;
                        capturingbytes = 0;
                }
        }
        else
        {
                captimehi = TCNT1;
                TCNT1 = 0;
        }
}

ISR(INT1_vect) //ft230x cbus1
{
        if (PIND & (1<<FT_CBUS1)) //reset radio display to 'PLAY' CD01 TR00
        {
                //PORTA &= ~(1<<LED_PWR);
                cd = 0xBE;
                tr = 0xFF;
                mode = 0xFF;
        }
        else //usb connect
        {
                PORTA |= (1<<LED_PWR);
                _delay_ms(50);
                PORTA &= ~(1<<LED_PWR);
                cd = 0xB0;
                tr = 0x00;
        }
}

uint8_t spi_xmit(uint8_t val) {
        SPDR = val;
        while(!(SPSR & (1<<SPIF)));
        return SPDR;
}

void send_package(uint8_t c0, uint8_t c1, uint8_t c2, uint8_t c3, uint8_t c4, uint8_t c5, ui
nt8_t c6, uint8_t c7)
{
        spi_xmit(c0);
        _delay_us(874);
        spi_xmit(c1);
        _delay_us(874);
        spi_xmit(c2);
        _delay_us(874);
        spi_xmit(c3);
        _delay_us(874);
        spi_xmit(c4);
        _delay_us(874);
        spi_xmit(c5);
        _delay_us(874);
        spi_xmit(c6);
        _delay_us(874);
        spi_xmit(c7);
}
```

```
int main(void)
{
        cd = 0xBE;
        tr = 0xFF;
        mode = 0xFF;
        //LEDs
        DDRA |= (1<<LED_PWR);

        //pullup
        PORTD |= (1<<FT_CBUS1);

        uart_init(UART_BAUD_SELECT(UART_BAUD_RATE, F_CPU));

        //init SPI
        DDRB |= (1<<PB5) | (1<<PB7)| (1<<DDB4);

        // SPI Type: Master
        // SPI Clock Rate: 62,500 kHz
        // SPI Clock Phase: Cycle Start
        // SPI Clock Polarity: Low
        // SPI Data Order: MSB First
        SPCR=0x57;//at 8MHz
        //SPCR=0x56;//at 4MHz
        SPSR=0x00;

        //beta commands -> cdc
        TCCR1B |= (1<<CS11);    // no prescaler 8 -> 1 timer clock tick is 1us long
        GICR |= (1<<INT0) | (1<<INT1);
        MCUCR |= (1<<ISC00) | (1<<ISC10); //any change on INT0 and INT1
        sei();

        //init led on
        PORTA |= (1<<LED_PWR);
        _delay_ms(500);
        //uart_puts("VAG_CDC ready...\r\n");
        uart_putc(0xAA);
        uart_putc(0x55);
        PORTA &= ~(1<<LED_PWR);

        send_package(0x74,0xBE,0xFE,0xFF,0xFF,0xFF,0x8F,0x7C); //idle
        _delay_ms(10);
        send_package(0x34,0xFF,0xFE,0xFE,0xFE,0xFF,0xFA,0x3C); //load disc
        _delay_ms(100);
        send_package(0x74,0xBE,0xFE,0xFF,0xFF,0xFF,0x8F,0x7C); //idle
        _delay_ms(10);

    while(1)
    {
                int r = uart_getc();
                //r has new data
                if(r <= 0xFF)
                {
                        //send inverted CD No.
                        if((r & 0xC0) == 0xC0)
                        {
                                if (r == 0xCA)
                                        mode = MODE_SCAN;
                                else if (r == 0xCB)
                                        mode = MODE_SHFFL;
                                else if (r == 0xCC)
                                        mode = MODE_PLAY;
                                else
                                        cd = 0xFF^(r & 0x0F);
                        }
                        //send inverted TR No.
                        else
                                tr = 0xFF^r;
                }
                //                disc  trk  min  sec
                //send_package(0x34,cd,tr,0xFF,0xFF,0xFF,0xCF,0x3C);
                send_package(0x34,cd,tr,0xFF,0xFF,mode,0xCF,0x3C);
        _delay_ms(41);
                if (newcmd)
                {
                        newcmd = 0;
                        uint8_t c = getCommand(cmd);
                        if (c)
                        {
                                uart_putc(c);
                                /*if (c == CDC_STOP)
                                {
                                        shutdownpending = 1;
                                }
                                else if (shutdownpending && c == CDC_END_CMD)
                                {
                                        shutdownpending = 0;

                                        PORTA &= ~(1<<LED_PWR);
                                        _delay_ms(100);
                                        shutdown();
                                        PORTA |= (1<<LED_PWR);
                                }*/
                        }
                }
    }
}

uint8_t getCommand(uint32_t cmd)
{
        if (((cmd>>24) & 0xFF) == CDC_PREFIX1 && ((cmd>>16) & 0xFF) == CDC_PREFIX2)
                if (((cmd>>8) & 0xFF) == (0xFF^((cmd) & 0xFF)))
                        return (cmd>>8) & 0xFF;
        return 0;
}
```

Additional info can be found here: <www.mikrocontroller.net/topic/28549> (in German)

5. The Circuit

Nothing special here, just an ISP header for programming the AVR, USB connector, voltage regulator, LED,… For feature circuits in the glovebox I provided 12V, 5V and GND through further pins. I was thinking of powering the RPi directly from my faker. But I don't know if the radio is able to provide around 400mAmps. Even the regulator would heat up like hell at these currents.



The FT230X is programmed as selfpowered and CBUS1 as output for the #POWER_ON singnal.

6. Bringing all together: MPD and RPi

Now we need a wrapper for `mpc` to let mpd act as our cd changer. I've written a python script to evaluate the radio commands and send cd and track numbers back periodically. The script starts at startup by the `rc.local` and checks for the connected faker. The last listened cd no. will be stored in a file to send it to the faker if something is restarted or reconnected. I let the script stop mpd to save power and make sure the current settings are synced with the filesystem, because a powerfail will cause a corrupted playlist. The music is stored in `/var/vag_cdc/cdX` where X is the cd no. 1 to 6. You can put up to 99 tracks in each folder due to radio display limitations.

```
#!/usr/bin/python

import os
import serial
import string
import time

# bytes sent when radio keys are pressed
CDC_END_CMD = chr(0x14)
CDC_PLAY = chr(0xE4)
CDC_STOP = chr(0x10)
CDC_NEXT = chr(0xF8)
CDC_PREV = chr(0x78)
CDC_SEEK_FWD = chr(0xD8)
CDC_SEEK_RWD = chr(0x58)
```

```
CDC_CD1 = chr(0x0C)
CDC_CD2 = chr(0x8C)
CDC_CD3 = chr(0x4C)
CDC_CD4 = chr(0xCC)
CDC_CD5 = chr(0x2C)
CDC_CD6 = chr(0xAC)
CDC_CDSET = chr(0x38)
CDC_SCAN = chr(0xA0)
CDC_SFL = chr(0x60)
CDC_PLAY_NORMAL = chr(0x08)

# control bytes to set the display
CMD_SCAN = chr(0xCA)
CMD_SHFFL = chr(0xCB)
CMD_PLAY = chr(0xCC)

CD_MASK = 0xC0

cd_filename = "/var/vag_cdc/last_cd"
global ser
ser = None
timeout_seek = 0.5
timeout_normal = 2

# send current cd no. to radio and save it for next reboot
def set_cd_no(val):
        ser.write(val)
        fo = open(cd_filename, "wb")
        fo.write(val)
        fo.close()

# load new cd-dir
def play_cd(cd_no):
        if ser.read() == CDC_END_CMD and ser.read() == CDC_CDSET:
                r = os.popen("mpc ls cd" + str(cd_no)).read()
                if r != "":
                        set_cd_no(chr(CD_MASK + cd_no))
                        os.popen("mpc clear")
                        os.popen("mpc ls cd" + str(cd_no) + " | mpc add")
                        os.popen("mpc play")

# init serial and last cd no. for correct radio display
def connect():
        while(1):
                try:
                        global ser
                        ser = serial.Serial('/dev/ttyUSB0', 9600)
                        ser.timeout = timeout_normal #we need a timeout to update the track
no.
                        ser.writeTimeout = timeout_normal
                        try:
                                fo = open(cd_filename, "rb")
                                cd = fo.read(1)
                                fo.close()
                                ser.write(cd)
                        except:
                                print "no last cd no. [not sending cd#]"
                        return
                except:
                        time.sleep(2) #wait before retrying

try:
        os.popen("/etc/init.d/mpd restart") #restart mpd due to alsa-equalizer
        connect()

        # read cmds from the radio and act like a cd changer
        while(1):
                try:
                        c = ser.read()

                        if c == CDC_PLAY:
                                if ser.read() == CDC_END_CMD:
                                        os.popen("/etc/init.d/mpd start")
                                        os.popen("mpc play")
                                        os.popen("mpc repeat on")

                        elif c == CDC_STOP:
                                if ser.read() == CDC_END_CMD:
                                        #os.popen("mpc pause")
                                        # stop service to sync current status
                                        os.popen("/etc/init.d/mpd stop")

                        elif c == CDC_NEXT:
                                os.popen("mpc next")

                        elif c == CDC_PREV:
                                os.popen("mpc prev")

                        elif c == CDC_SEEK_FWD:
                                ser.timeout = timeout_seek
                                while(1):
                                        os.popen("mpc seek +00:00:10")
                                        if ser.read() == CDC_PLAY and ser.read() == CDC_END_
CMD:
                                                break
                                ser.timeout = timeout_normal

                        elif c == CDC_SEEK_RWD:
                                ser.timeout = timeout_seek
                                while(1):
                                        os.popen("mpc seek -00:00:10")
                                        if ser.read() == CDC_PLAY and ser.read() == CDC_END_
CMD:
                                                break
                                ser.timeout = timeout_normal

                        elif c == CDC_CD1:
                                play_cd(1)

                        elif c == CDC_CD2:
                                play_cd(2)

                        elif c == CDC_CD3:
                                play_cd(3)

                        elif c == CDC_CD4:
                                play_cd(4)

                        elif c == CDC_CD5:
                                play_cd(5)

                        elif c == CDC_CD6:
                                play_cd(6)

                        elif c == CDC_SCAN:
                                os.popen("mpc update")

                        elif c == CDC_SFL:
                                os.popen("mpc random on")

                        elif c == CDC_PLAY_NORMAL:
                                if ser.read() == CDC_END_CMD:
                                        os.popen("mpc random off")

                        #get current track no. (radio display will be delayed up to {timeout
_normal})
                        mpc = os.popen("mpc |grep ] #").read()
                        try:
                                mpc = mpc.split("/", 1)
                                mpc = mpc[0].split("#", 1)
                                tr = chr(string.atoi(mpc[1], 16))
                                ser.write(tr)
                        except:
                                print "not playing"

                        #get current playmode
                        mpc = os.popen("mpc | grep random:").read()
                        try:
                                mpc = mpc.split("random: ", 1)
                                if mpc[1].startswith("on"):
                                        ser.write(CMD_SHFFL)
                                elif mpc[1].startswith("off"):
                                        ser.write(CMD_PLAY)
                        except:
                                print "not playing"
                except (serial.SerialException, serial.SerialTimeoutException):
                        print "serial port unavailable, reconnecting..."
                        ser.close()
                        connect()
                except:
                        raise
except (KeyboardInterrupt):
        if ser is not None:
                ser.close()
                print "port closed!"
        print "KeyboardInterrupt detected, exiting..."
```

7. Some further ideas

The project is mostly complete - for now. But there are some things I have in mind that could be quite nice to have. It would be awesome if it had bluetooth to remote control my android phone. Or mute the radio at incoming calls and act as a hands free set. But what I’ve done lately is preparing the RPi with Wifi, so it connects automatically to my Wifi, when parking in the garage to sync the music folders with rsync.

Tags: Hardware (/tags#Hardware) C/C++ (/tags#C/C++) Raspberry Pi (/tags#Raspberry Pi) Microcontroller (/tags#Microcontroller) Python (/tags#Python)

← PREVIOUS POST (/2013/06/KERNEL-3-9-5-READY-TO-USE/)

NEXT POST → (/2017/05/SMTP-MAILING-MAILGUN-POSTFIX/)

39 Comments developer.shyd.de. Disqus' Privacy Policy 1 Login

Recommend Tweet Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Bartek • 5 years ago
Hi
Great project, much respect :)
Im wondering, will it work with Alfa Romeo/Fiat radio units?
Pinouts are basically the same. I need to build sth like this for Alfa Romeo since there is no possibility to play any external source
Reply • Share ›

Maciek • 6 years ago
Hi! Last time I interested in your project and I'm trying to build CDC Faker but on Arduino UNO with MP3 Player Shield. I tried to connect Arduino with radio but Arduino didn't see any signals on RX pin (UART). The same situation were when I sent prepared message to the radio - nothing. Have You any ides how can I solve this problem?
Thanks for all tips.
Reply • Share ›

Guest → Maciek • 6 years ago
There is no UART in VW Beta, it use SPI.
Reply • Share ›

shyd Mod → Guest • 6 years ago
Exactly beta <-> arduino needs to be SPI as described, arduino to whatever can be UART as in my project.
Reply • Share ›

Lacho • 6 years ago
Great post, thanks for the info! Do you think the same commands can work for a RCD300 radio (vw passat)?
Reply • Share ›

shyd Mod → Lacho • 6 years ago
Hi Lacho, my guess: yes, it should. The RCD300 pinout looks like this: View – disq.us
These are the same pins I use.
So give it a try and let me know, if you are making any progress.
Reply • Share ›

Lacho → shyd • 6 years ago
I'll try it, fingers crossed :) btw I noticed in the schematic the SPI pins are connected directly - that doesn't need any level shifting, right? They run on 5v?
Reply • Share ›

shyd Mod → Lacho • 6 years ago
Yes, they run at 5V. Good luck then!
Reply • Share ›

Lacho → shyd • 6 years ago
So I have some progress... the good news is that the SPI commands seem to work - I managed to get the cd changer screen to show up.

The strange thing is that no audio gets out of it - I tried both on the AUX pins and the CD-R/L pins. I thought that when it recognizes a cd changer it will unmute them, but perhaps I'm missing something...
Another strange thing is that the SPI communication seems quite unstable - it "magically" started working and now magically stopped working :) And I used to get some weird readings on the screen like CD FE, TRACK 0F... And when it was working, it was like once out of 5 times :)

working, it was like once out of 5 times :)

What I do is to continuously send this sequence with 41ms delay:
0x34, 0xBE, 0xFF, 0xFF, 0xFF, 0xFF, 0xCF, 0x3C

I've kept the delays as in your code.

I suspect it may be something in the hardware, my mcu uses 3.3v on the SPI lines, I hooked it up directly to the radio, also tried level shifting to 5v... no difference. It runs on 62500 HZ, mode 0. I've connected the CDC GND, CLOCK and DATA IN pins.
If you notice anything wrong with the above, please let me know.

I'll leave it for today and will continue digging tomorrow.
Thanks again for all the info - it's very useful!

1 ^ | v • Reply • Share ›

Dainius Cerniauskas ➦ Lacho • 9 months ago
Did you ever figure this out? I'm trying to do the same on my RCD 300 but struggling with getting the CD changer screen to show up.

^ | v • Reply • Share ›

shyd Mod ➦ Lacho • 6 years ago
That sounds great! Try to change the cd and track number. My radio can display hex too. Keep in mind that audio GND is not the circuit GND. Do you have a scope to analyse the edges after level shifting? Feel free to write me mails to avoid long reply chains in here.

^ | v • Reply • Share ›

Radek Pavlištík ➦ shyd • 6 months ago
Hello, Its right to send repeatly 0x34,0xBE,0xFf,0xFF,0xFF,0xFf,0xCF,0x3C for enable AUX? In my case when i press on radio CDC that show NO CD. Or i must answer on some command from radio? Thank you

^ | v • Reply • Share ›

shyd Mod ➦ Radek Pavlištík • 6 months ago
Hi, yes it is correct. That's why this line of code is there. If you don't repeatedly send any message "NO CD" will show up.
You don't have to directly answer commands sent from the head unit. But if you change the cd or track no. you'll probably have a better UX.
Hopefully this is what you need, otherwise let me know :-)

^ | v • Reply • Share ›

Andrew • 6 years ago
Great work, Can you give a detailed wiring diagram how Raspberry is connected to the radio?

^ | v • Reply • Share ›

shyd Mod ➦ Andrew • 6 years ago
Thanks!
The RPi is not connected directly to the radio. It's connected through USB to my circuit which is connected to the CD-Changer socket on the radio.

^ | v • Reply • Share ›

Andrew ➦ shyd • 6 years ago
Thanks for the explaination. Now I understand how it is working :)

^ | v • Reply • Share ›

tomas • 6 years ago
what about this? :) http://www.cooking-hacks.co...

^ | v • Reply • Share ›

Tomas • 6 years ago
arduino code:

http://kovo-blog.blogspot.s...

^ | v • Reply • Share ›

shyd Mod ➦ Tomas • 6 years ago
Great! Thanks for sharing!

^ | v • Reply • Share ›

Tomas ➦ shyd • 6 years ago
you welcome, also have a plane for attiny85 (greate site about spi: http://gammon.com.au/spi), maybe put it all (including your code) in one file with lot of #IFDEF ... and for everyone hase alter cars withot CAN-BUS and with display on cluster: http://kovo-blog.blogspot.s...

^ | v • Reply • Share ›

Tomas Tomas • 6 years ago
s/hase alter/has older/g
Reply • Share ›

nail Tomas • 6 years ago
ogrh, I just try spi on attiny85 and is total crap :) after all I switch to "shiftOut()" function from arduino, this works for me, only problem si now, that tiny has just 8bit timers :(
Reply • Share ›

shyd Mod nail • 6 years ago
You can try to manage capturing with 8-bit timers by adapting the captimes and setting the prescalers correctly.
Reply • Share ›

Tomas shyd • 6 years ago
I know, i just playing with it, i don't know why ist it not working corectly, anyway, i probably abandon this attiny "project" cose I just found that it's posible to buy atmega88 in square smd package on ebay for about 1$ so it's better in way of HW SPI then ATTINY and smaller package then atmega8 But I will try attiny for today for last time :) anyway, attiny85 source (work in progress) http://ubuntuone.com/2QEgIx...
Reply • Share ›

tomas Tomas • 5 years ago
I do my homework and make some new code for arduino uno also I sit and understande code ported from vwcdpic to avr and manage to configure it for run on atmega328 board: https://github.com/tomaskov...
Reply • Share ›

shyd Mod tomas • 5 years ago
Thanks for your link!
Reply • Share ›

tomas shyd • 5 years ago
did you have anytime problem with radio saying "no CDx" after change of disk?
Reply • Share ›

shyd Mod tomas • 5 years ago
Yes, you need to reply with a new CD no quickly. Otherwise the radio beeps.
Reply • Share ›

tomas shyd • 5 years ago
hmm ok, I will try to put it in switch function, strange is that CD1, CD2 and CD5 are alway ok, CD3 was never ok, CD4 and CD6 sometimes do that .... thx
Reply • Share ›

tomas tomas • 5 years ago
hello again, if anyone want your version run on arduino here is code to look (https://github.com/tomaskov..., i do some tweeks for make it better work with my audi concert1 head unit, also to make buttons next/prev CD works, this need to tune mpd control script https://github.com/tomaskov... and video: ▶ Audi concert mpd control — disq.us
Reply • Share ›

Vilius • 6 years ago
Is it possible to get stereo sound from this?
Reply • Share ›

shyd Mod Vilius • 6 years ago
Of course it is!
Reply • Share ›

Vilius shyd • 6 years ago
maybe you can give some advice, how to do that? p.s. great work!!!
Reply • Share ›

shyd Mod Vilius • 6 years ago
Well, how can I help you in particular? Maybe you write me an email and I can answer your questions.
The RPi supports stereo out of the box.
Reply • Share ›

Vilius shyd • 6 years ago
can i get your email than? or atleast, can you write me on viliux112@gmail.com

P.s. i already modified the schematics for stereo, so i guess code left

^ | ⌄ • Reply • Share ›

su541 • 7 years ago

Do you think, that i can use an "arduino uno" as hardware? It uses an amtel atmega328.

^ | ⌄ • Reply • Share ›

shyd Mod ➦ su541 • 7 years ago

Yes you can. Maybe you need to modify some settings and registers.

^ | ⌄ • Reply • Share ›

summi • 7 years ago

Here somebody has done the bluetooth part:
http://www.instructables.co...
Both part together would be similar like this:
http://www.dension.com/prod...

^ | ⌄ • Reply • Share ›

shyd Mod ➦ summi • 7 years ago

Nice link about the bluetooth part! But I thought more about a version without the RPi.
The ready to buy product is what I found, too. But did you take a look at the price? I don't count the RPi as it is a multipurpose devce. Everything together in my project is less than EUR 20.

^ | ⌄ • Reply • Share ›