Multifunktionstastenlenkrad ohne Wickelfederumbau

Teil 1: Allgemeine Lösung.

von Martin Enke

Tempomat ist links viel praktischer

Das Problem

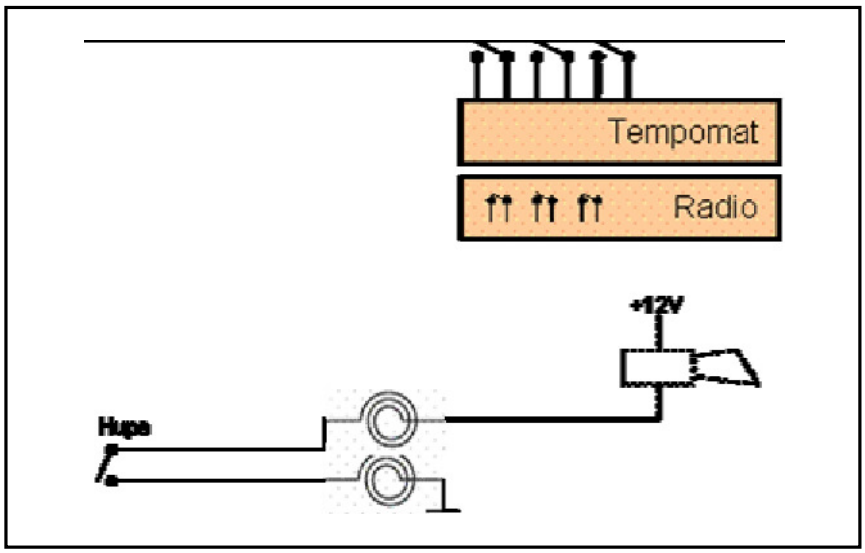
Hat ein Lenkrad Multifunktionstasten, müssen die Signale der Tasten irgendwie über die Lenksäule an den entsprechenden Kommandoempfänger geleitet werden. Normalerweise gelangen diese Signale über eine sogenannte Wickelfeder (fälschlicherweise oft als "Schleifringe" bezeichnet) vom beweglichen Lenkrad in den fest stehenden Teil der Karosserie. Möchte man solche Tasten nachrüsten, braucht man also eine entsprechende Wickelfeder. Wenn das Fahrzeug aber nicht in einer Variante mit Multifunktionstasten erhältlich war, dann gibt es auch keine Wickelfeder dafür. Bei meinem BMW Z3 ist das so. Es gibt Anleitungen, wie man andere Wickelfedern anpassen kann, aber schon die Beschaffung dieser ist mit Kosten von ca. EUR 60 verbunden. Außerdem funktioniert damit der Tempomat des Z3 noch lange nicht, denn das Tempomatsteuergerät versteht die Multifunktionstasten nicht.

Eine weitere Motivation für dieses Projekt war, dass ich die linken Tasten für den Tempomat nutzen wollte, weil ich dies persönlich praktischer finde. Ich wollte also auch die Multifunktionstasten frei belegen können.

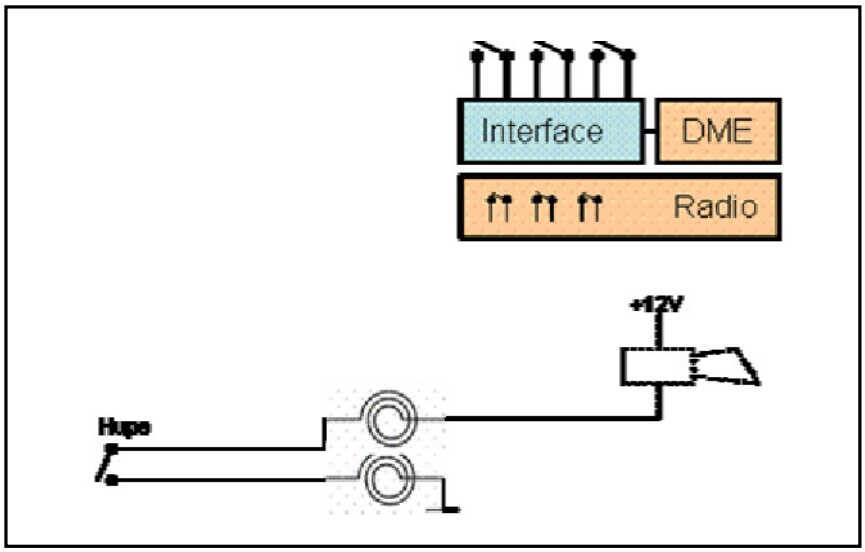
von Martin Enke (xdata) für z3-roadster-forum.de

Ausgangssituation

Um das Problem und die Lösung verständlich zu machen, stelle ich hier schematisch die Ausgangssituation dar:

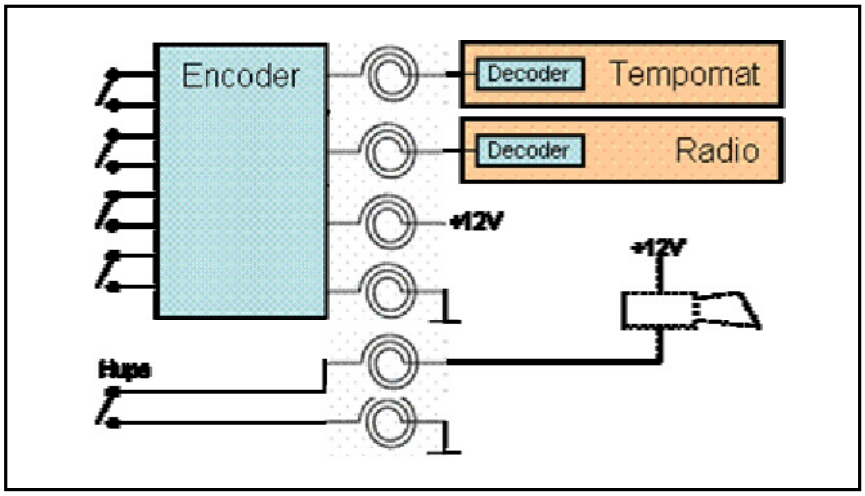


Ausgangssituation beim BMW Z3 VFL



Ausgangssituation beim BMW Z3 FL. DME = Digitale Motorelektronik. Der FL hat sogenanntes E-Gas. Dabei wird die Drosselklappe nicht vom Gaszug betätigt sondern von einem Elektromotor. Und der wird von einem Prozessor gesteuert der eben auch wie ein Tempomat arbeiten kann.

Wie in der Abbildung ersichtlich, haben Tempomat und Radio ihre eigenen Schalter oder Tasten. Die Hupe wird im Lenkrad auf Masse geschaltet. Warum ich das so dargestellt habe, erklärt sich in der nächsten Abbildung.

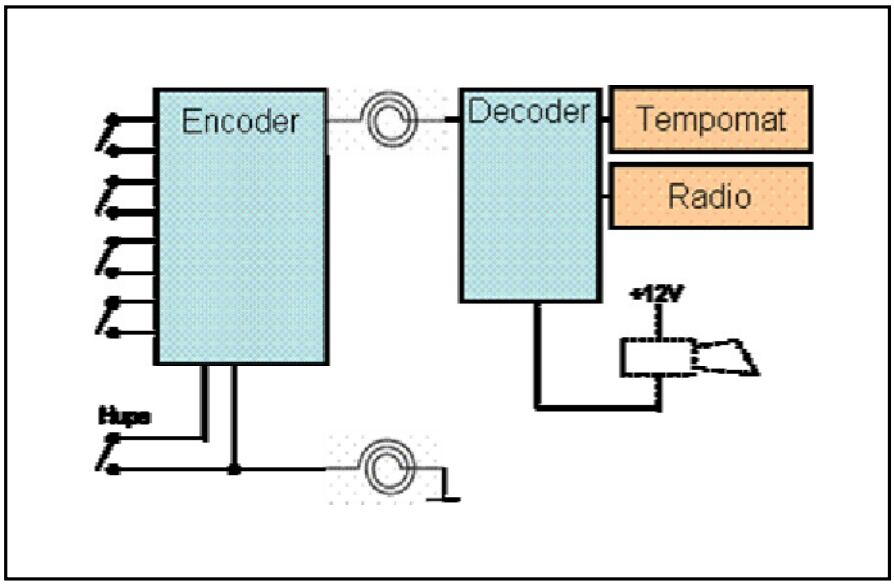


Blockschaltbild des Aufbaus beim BMW E39 mit Multifunktionstastenlenkrad

Der E39 hat eine sechsadrige Wickelfeder. Die sechs Adern allein reichen natürlich nicht, um die Signale von zehn Funktionstasten, der Hupe und der Beleuchtung zu übertragen. Die Tasten werden also kodiert, um Adern zu sparen. Das so kodierte Signal muss natürlich auf der anderen Seite wieder dekodiert werden, weshalb sowohl das Radio als auch das Tempomatsteuergerät einen Decoder hat. Das ist der Grund, warum die Anpassung einer E39 Wickelfeder für unsere Zwecke nur die halbe Miete ist: Man muss auch ein Steuergerät mit eingebautem Decoder haben – oder eine bereits vorbereitete Motorelektronik, wie etwa beim Z3 FL. Das E39 Steuergerät ist nicht pinkompatibel zum Z3 VFL Kabelbaum. Natürlich braucht man auch ein Radio mit Decoder. Hat man ein Original-BMW-Radio, kann man da mal Glück haben, ansonsten muss man einen externen Decoder kaufen und anpassen. Ein externer Decoder für das Tempomatsteuergerät existiert meines Wissens nicht. Das Protokoll zwischen den Multifunktionstasten und dem Radio nennt BMW "I-Bus". Welches Protokoll für den Tempomat verwendet wird, weiß ich nicht. Ich vermute aber, dass es dem I-Bus-Protokoll sehr ähnlich ist, weil es im selben Mikroprozessor im Encoder erzeugt wird.

von Martin Enke (xdata) für z3-roadster-forum.de

Die Lösung



Blockschaltbild der Lösung des Wickelfederproblems mit dem 1-Draht-Bus

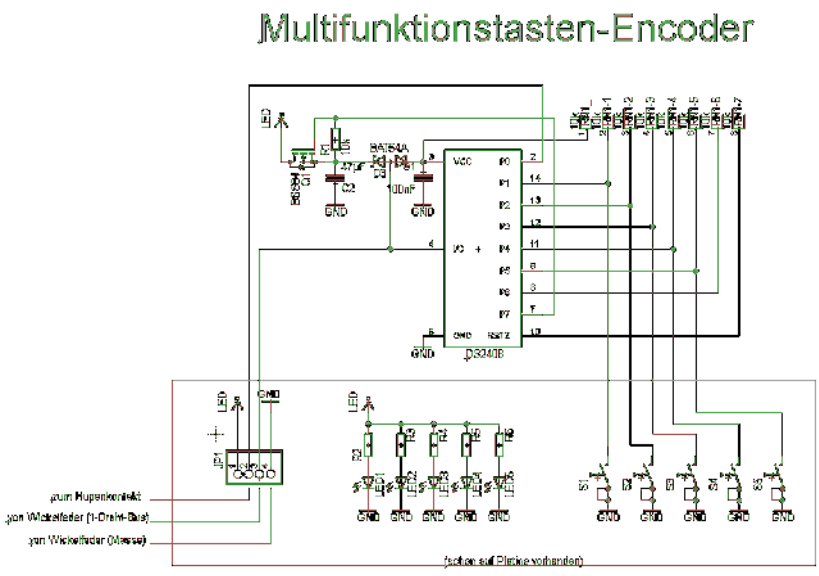
Wie man sieht, kommt diese Lösung mit zwei Adern in der Wickelfeder aus. Dafür müssen aber sowohl Encoder als auch Decoder durch eine Variante ersetzt werden, die mit weniger Leitungen auskommt. Ich benutze das sogenannte "Ein-Draht"-System (1-Wire-Bus). Es heißt so, weil es – welch Überraschung – mit einem Draht auskommt. (Was zwar nicht ganz stimmt, weil es natürlich auch noch eine Masseverbindung braucht, aber wir wollen mal nicht so streng sein.) Fakt ist, dass dieses System dasselbe leistet wie das von BMW, nur eben mit viel weniger Bauteilen und Adern. Als Adern benutze ich das, was schon da ist: Die Hupleitung wird zur Datenleitung, Masse bleibt Masse.

Warum nutzt BMW nicht dieses System? Weil es noch nicht erfunden war, als der E39 entwickelt wurde.

Der Bauplan

Für den Nachbau sind keine großen Elektronikkenntnisse erforderlich. Etwas handwerkliches Geschick und ein Lötkolben reichen aus.

Zuerst müssen also die Encoder in die Tasten eingebaut werden. Hier der Schaltplan:

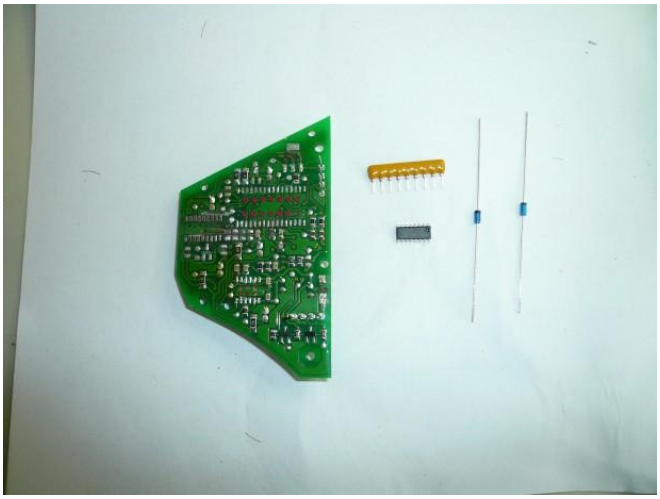


Schaltplan des Encoders

von Martin Enke (xdata) für z3-roadster-forum.de

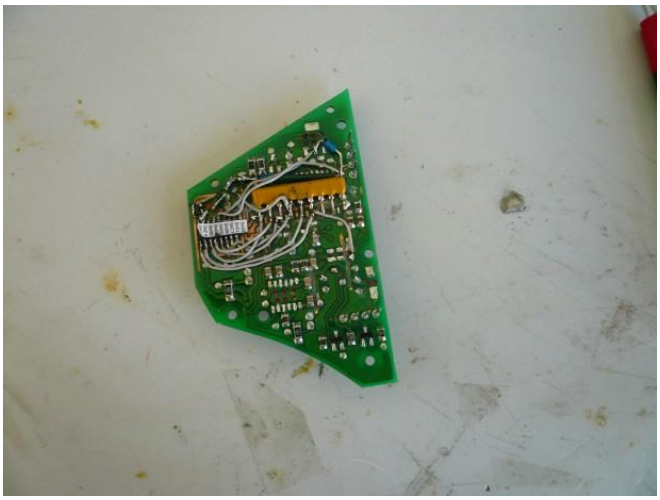
Es sind nur ein paar Teile: Neben dem IC braucht man ein Widerstandsnetzwerk, eine Doppeldiode, zwei Kondensatoren und ein MOSFETransistor. Die Kondensatoren und der Transistor sind unkritisch und können auch durch ähnliche Teile ersetzt werden, genauso die Dioden; es sollten aber Schottky-Teile sein. Das ist alles für unter EUR 10 zu bekommen und ist so klein, dass es bequem auf die Fläche der Tastenplatine passt. Die Beleuchtungs-LEDs behalten ihre Vorwiderstände. Die anderen Widerstände werden ausgelötet. Die Schaltung wird in beide Tastenseiten eingebaut und die Pole 3 und 4 von JP1 dann parallel geschaltet und über die Wickelfeder zum Decoder geführt.

So sieht die Platine des rechten Tastenfeldes aus.



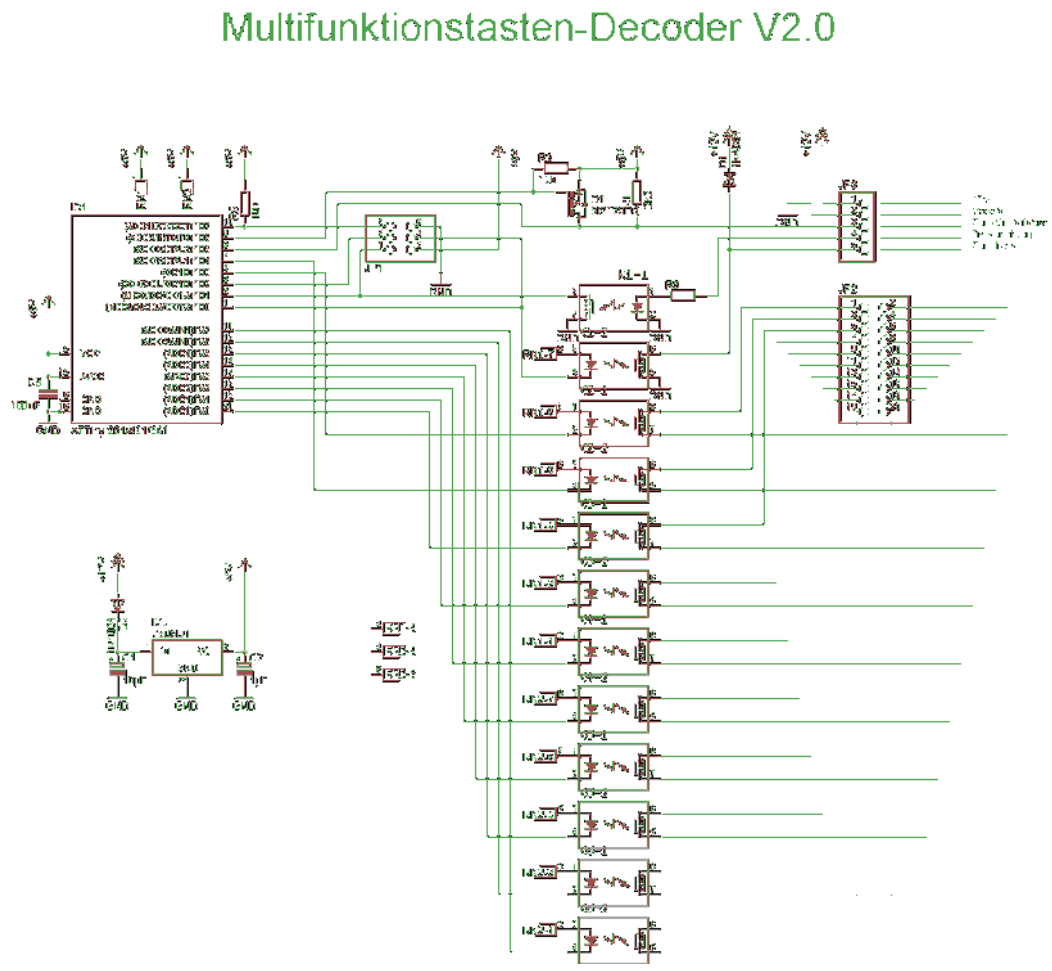
Zuerst werden die nicht mehr benötigten Teile ausgelötet. Rechts sind die neuen Teile. Kondensatoren und LED-Vorwiderstände werden übernommen.

von Martin Enke (xdata) für z3-roadster-forum.de



Die neuen Teile in eingelötetem Zustand.

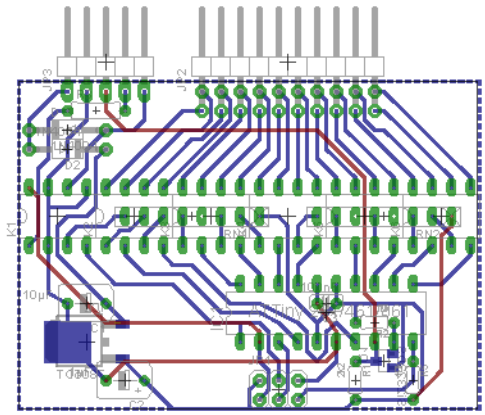
Bevor die Tasten eingebaut werden muss noch die ID des Encoder-Chips ausgelesen werden weil der Decoder diese ID braucht sobald mehr als ein Encoder angeschlossen wird. Und wir haben ja zwei Tastenfelder...



Schaltplan des Decoders

Der Decoder hat drei Anschlüsse: JP1 ist der Programmieranschluss für den Prozessor, JP2 hat die den Tasten entsprechenden Kontakte, und JP3 ist der Versorgungsanschluss. Die Schaltung lässt sich leicht auf einer Lochrasterplatine (5x6 cm) aufbauen. Auch hier sind nur wenige Teile notwendig: Als Prozessor habe ich einen von Atmel gewählt, weil der billig ist und keinen Quarz braucht und auch in einem DIP-Gehäuse zu bekommen ist, sich

also ideal für Bastler eignet. Die Photo-Mosfets sind die Preistreiber hier: Sie kosten ca. EUR 2,50 pro Stück und man braucht sechs davon. Also kommt man auf rund EUR 20 für alles.



Platinenlayout

So kann die Schaltung aufgebaut werden. Das ist etwa die Originalgröße.

Der Prozessor braucht ein Programm:

```
#include <avr/io.h>
#ifndef F_CPU
#define F_CPU 1000000UL
#endif
#include <util/atomic.h>
#include <util/delay.h>

#define ON 0
#define OFF 1
#define HIGH 1
#define LOW 0
#define Left 0
#define Right 1

#define OW_PORT PORTB
#define OW_PORT_IN      PINB
#define OW_DDR          DDRB
#define OW_PIN          PB5

/*******************************/
/*** Masken für die Ports des ***/
/*** DS2408                   ***/
/*******************************/

#define MASK_LEFT_HORNS         0x01
#define MASK_LEFT_1                     0x02
#define MASK_LEFT_2                     0x04
#define MASK_LEFT_3                     0x08
#define MASK_LEFT_4                     0x10
#define MASK_LEFT_5                     0x20
#define MASK_LEFT_LIGHT         0x80

#define MASK_RIGHT_HORNS0x01
#define MASK_RIGHT_1            0x02
#define MASK_RIGHT_2            0x04
#define MASK_RIGHT_3            0x08
#define MASK_RIGHT_4            0x10
#define MASK_RIGHT_5            0x20
#define MASK_RIGHT_LIGHT0x80

/*****************************/
/*** Ein- und Ausgänge *******/
/*****************************/

#define POWER PB6
#define LIGHT PB1
#define HORNS PB0
```

von Martin Enke (xdata) für z3-roadster-forum.de

```
#define LEFT_1 PB3
#define LEFT_2 PB4
#define LEFT_3 PA0
#define LEFT_4 PA1
#define LEFT_5 PA2

#define RIGHT_1 PA3
#define RIGHT_2 PA4
#define RIGHT_3 PA5
#define RIGHT_4 PA6
#define RIGHT_5 PA7



/***************************/
/*** Funktions-Prototypen ***/
/***************************/
unsigned char ow_reset(void);
void ow_wr_bit(unsigned char bit);
unsigned char ow_rd_bit(void);
void ow_wr_byte(unsigned char byte);
unsigned char ow_rd_byte(void);
int main (void);
void PowerUp();
void ReadButtons(unsigned char index);
void SetOutputs();
void SetPortABit(unsigned char bit, unsigned char level);
void SetPortBBit(unsigned char bit, unsigned char level);

/************************/
/*** Globale Variablen ***/
/************************/

unsigned char m_WriteByte;
unsigned char m_ReadByte[2];
unsigned char rom_c[8];
unsigned char roms[2][8] = {
        {0,0,0,0,0,0,0,0},      // hier muss die ID des DS2408 der rechten Seite eingetragen werden
        {0x29,0x21,0xc7,0x0e,0x00,0x00,0x00,0xc4}       // hier muss die ID deS DS2408 der linken Seite eingetragen
werden
};

unsigned char m_Is_Left_Avail, m_Is_Right_Avail;

/*****************************************************************************************************/
/*** Main
/*****************************************************************************************************/

int main(void){

        unsigned char i;

        // Ports initialisierne
        PORTB |= _BV(LIGHT);            // Dashboard light
        DDRB &= ~_BV(LIGHT);
        DDRB |= _BV(POWER);                     // Schliesst den Pull-Up kurz, um die Kondensatoren der parasitären
Stromversorgung zu laden
        DDRB |= _BV(HORNS);                     // Ausgang für die Hupe

        DDRB |= _BV(LEFT_1);            // Ausgänge der Tasten
        DDRB |= _BV(LEFT_2);
        DDRA |= _BV(LEFT_3);
        DDRA |= _BV(LEFT_4);
        DDRA |= _BV(LEFT_5);

        DDRA |= _BV(RIGHT_1);
        DDRA |= _BV(RIGHT_2);
        DDRA |= _BV(RIGHT_3);
        DDRA |= _BV(RIGHT_4);
        DDRA |= _BV(RIGHT_5);

        PowerUp();

        if(ow_reset())while(1);         //falls keine Tasten angeschlossen sind, ist hier Schluss

        // wenn nur ein Tastenfeld angeschlossen ist, wird hier die ID ausgelesen. Nur zum debuggen sinnvoll, kann man
dann oben eintragen
        ow_wr_byte(0x33);
```

von Martin Enke (xdata) für z3-roadster-forum.de

```
        for (i = 0; i < 8; i++) rom_c[i] = ow_rd_byte();

        m_Is_Left_Avail = roms[Left][0] == 0x29 ? 1 : 0;
        m_Is_Right_Avail = roms[Right][0] == 0x29 ? 1 : 0;

    while(1){
                m_WriteByte = ((PINB & _BV(LIGHT)) == 0) ? 0x7F : 0xFF; //Schreibe auf die DS2408, je nachdem; ob die
Armaturenbeleuchtung eingeschaltet ist
                m_ReadByte[Left] = 0xFF;
                m_ReadByte[Right] = 0xFF;
                if(m_Is_Left_Avail) ReadButtons(Left);
                if(m_Is_Right_Avail) ReadButtons(Right);
                SetOutputs();
                PowerUp();
    }
}

/*************************************************************/
/*** Kurzschluss des Pull-Ups, um die Kondensatoren zu laden ***/
/*************************************************************/
void PowerUp(){
        PORTB &= ~_BV(POWER);
        _delay_ms(50);
        PORTB |= _BV(POWER);
}

void ReadButtons(unsigned char index){
        unsigned char i;
        ow_reset();
        ow_wr_byte(0x55);
        for (i = 0; i < 8; i++) ow_wr_byte(roms[index][i]);
        ow_wr_byte(0x5a);
        ow_wr_byte(m_WriteByte);
        ow_wr_byte(~m_WriteByte);
        m_ReadByte[index] = ow_rd_byte();       //danach sollte m_ReadByte immer 0xaa sein.
        m_ReadByte[index] = ow_rd_byte();       //jetzt steht der Zustand des DS2408 in m_ReadByte
}

void SetOutputs(){
        SetPortBBit(HORNS, m_ReadByte[0] & MASK_LEFT_HORNS);
        SetPortBBit(LEFT_1, m_ReadByte[0] & MASK_LEFT_1);
        SetPortBBit(LEFT_2, m_ReadByte[0] & MASK_LEFT_2);
        SetPortABit(LEFT_3, m_ReadByte[0] & MASK_LEFT_3);
        SetPortABit(LEFT_4, m_ReadByte[0] & MASK_LEFT_4);
        SetPortABit(LEFT_5, m_ReadByte[0] & MASK_LEFT_5);

        SetPortBBit(HORNS, m_ReadByte[1] & MASK_RIGHT_HORNS);
        SetPortABit(RIGHT_1, m_ReadByte[1] & MASK_RIGHT_1);
        SetPortABit(RIGHT_2, m_ReadByte[1] & MASK_RIGHT_2);
        SetPortABit(RIGHT_3, m_ReadByte[1] & MASK_RIGHT_3);
        SetPortABit(RIGHT_4, m_ReadByte[1] & MASK_RIGHT_4);
        SetPortABit(RIGHT_5, m_ReadByte[1] & MASK_RIGHT_5);
}


void SetPortABit(unsigned char bit, unsigned char level){
        if (level == LOW){
                PORTA &= ~_BV(bit);
        }else{
                PORTA |= _BV(bit);
        }
}

void SetPortBBit(unsigned char bit, unsigned char level){
        if (level == LOW){
                PORTB &= ~_BV(bit);
        }else{
                PORTB |= _BV(bit);
        }
}


/*****************************************************************************/
/***  Basis 1-Wire-Funktionen  *********************************************/
/*****************************************************************************/

unsigned char ow_reset(void)
{
```

von Martin Enke (xdata) für z3-roadster-forum.de

```
        unsigned char zw;
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
                OW_DDR |= _BV(OW_PIN);                                  // Configure the pin as an output.
                OW_PORT &= ~_BV(OW_PIN);                        // Pull the bus low.
                _delay_us(500); // 500 uS                                       // Wait the required time.
                OW_DDR &= ~_BV(OW_PIN);                                 // Switch to an input, enable the pin change
interrupt, and wait.
                _delay_us(70);
                zw = OW_PORT_IN & _BV(OW_PIN);
                _delay_us(420);
        }
        return zw;              // Rueckgabe: 0 = Slave vorhanden, 1 = kein Slave vorhanden
}

void ow_wr_bit(unsigned char bit){
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
                OW_DDR |= _BV(OW_PIN);
                OW_PORT &= ~_BV(OW_PIN);
                if (bit == 0x00) {
                        _delay_us(90);
                        OW_PORT |= _BV(OW_PIN);
                        _delay_us(30);
                }else {
                        _delay_us(10);
                        OW_PORT |= _BV(OW_PIN);
                        _delay_us(50);
                }
        }
}

unsigned char ow_rd_bit(void){
        unsigned char reply;
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
                OW_DDR |= _BV(OW_PIN);
                OW_PORT &= ~_BV(OW_PIN);
                _delay_us(2);
                OW_DDR &= ~_BV(OW_PIN);
                _delay_us(11);
                reply = (OW_PORT_IN & _BV(OW_PIN)) == 0x00 ? 0x00 : 0x01;
                _delay_us(47);
        }
        return reply;
}

void ow_wr_byte(unsigned char byte){
        unsigned char position;

        for (position = 0x00; position < 0x08; position++) {
                ow_wr_bit(byte & 0x01);
                byte = (byte >> 1);
        }
}

unsigned char ow_rd_byte(void){
        unsigned char position;
        unsigned char byte = 0x00;
        for (position = 0x00; position < 0x08; position++) byte += (ow_rd_bit() << position);
        return byte;
}

/*****************************************************************************************/
```

Wer nicht programmieren möchte oder kann, darf mir gerne den Prozessor zuschicken und kann den Programmieranschluss auch noch wegrationalisieren.

Dieser Teil der Anleitung ist der "generelle", will sagen: Am Ausgang des Decoders stehen die Signale der Multifunktionstasten. Man muss ja nicht zwingend einen Tempomaten oder ein Radio steuern, vielleicht hat man ja andere Pläne. Oder ein anderes Auto als den BMW Z3 (obwohl ich nicht verstehen kann wieso man ein anders Auto fahren sollte).

Im zweiten Teil, dem "speziellen", werde ich die Anbindung von Tempomat und Radio an den Decoder besprechen am Beispiel des BMW Z3 VFL.

von Martin Enke (xdata) für z3-roadster-forum.de

Bei Fragen oder Vorschlägen: Bitte PN an meinen Nickname:

xdata

Bei

www.z3-roadster-forum.de