Decoder selbst bauen – Teil 4

DECODER-SOFTWARE

Die Programme, die im zweiten Teil der Artikelserie für den Arduino entstanden sind, können wir für den Zubehördecoder leicht abgewandelt weiterverwenden. Außerdem müssen wir Funktionen ergänzen, die es ermöglichen, eine Selectrix Adresse über den SX-Bus zu programmieren – der ATtiny2313 hat nämlich keine USB-Schnittstelle.

Fotos Michael Blank

Teil 1	•	Digitalprotokolle • Aufbau Gleissignal • Selectrix • Vor- und Nachteile der Protokolle (1/2013)
Teil 2	•	Datenstrom Decodierung • Das C-Programm (2/2013)
Teil 3	•	Selectrix Weichen- und Signaldecoder • Universeller ATtiny2313 (3/2013)
Teil 4	**•**	**Software für den ATtiny2313 • ISP-Programmierung • Einstellung und Betriebspraxis**

Die Software besteht jetzt aus drei wesentlichen Teilen:

1) die Decodierung des Selectrix-Signals (inklusive „Rückkanal“ zur Zentrale)
2) die Hauptschleife („Main Loop“) zur Ansteuerung der Ausgänge
3) das Programm zum Einstellen der SX-Adresse

In den C-Dateien (= Source Files) sxtiny.h/sxtiny.c ist Teil 1 enthalten, die Dateien sxread.h/sxread.c implementieren die Hauptschleife und die Adresseinstellung.

Zur Decodierung des zyklisch von der Zentrale übertragenen Selectrix-Signals gibt es gegenüber dem in Teil 2 beschriebenen Programm nicht viel hinzuzufügen – allerdings besteht jetzt in der Interrupt-Service-Routine die Möglichkeit, bei bestimmten Adressen auch Daten auf den SX-Bus zurückzuschreiben. Sobald die Programmiertaste gedrückt wird, gibt der Decoder die eingespeicherten Werte, zum Beispiel seine eigene Adresse, auf Kanal 0 aus. Siehe hierzu im Programmlisting die Funktion switch_data(), die um entsprechende digitalWrite() Befehle erweitert wurde.

Um diese Werte an die Zentrale zu übermitteln, wird über Port-D/Pin 1 und einen 150-Ohm-Widerstand die Selectrix-„Rückkanal“-Leitung (Pin 5 des SX-Busses, siehe Teil 2) auf 0 V oder 5 V gelegt, um ein 0- oder 1-bit im entsprechenden Datenframe an die Zentrale zu schicken. Diese interpretiert die Werte und kann so die momentanen Einstellungen des Decoders lesen. Gibt die Zentrale dann auf Kanal 0 einen neuen Wert aus (z.B. „80“), dann versteht der Decoder (im Programmiermodus) dies als Aufforderung, nach Beenden des Programmiermodus diese neue Adresse zu verwenden. Dieses Verfahren ist die Standard-Decoder-Programmierung im Selectrix System.

EINSTELLUNG DES DECODERS

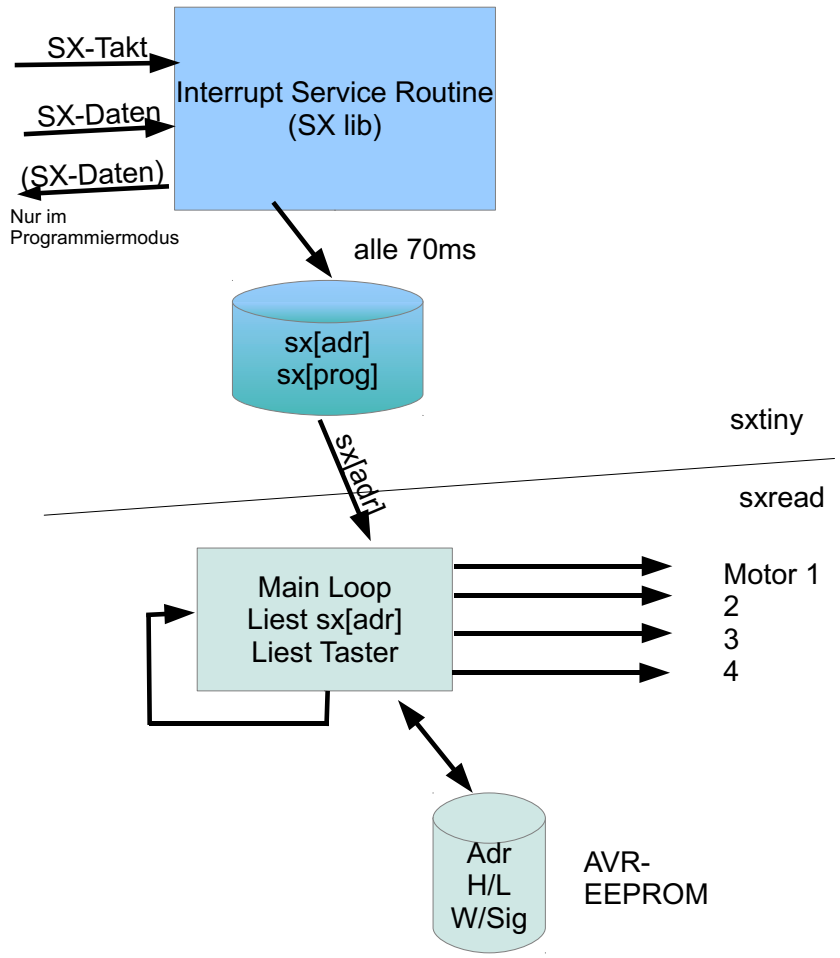
Im wdec1-Decoder werden weitere Werte benutzt, um den Decoder entweder für vier Weichenmotoren oder für acht Signallampen verwenden zu können. Für die Weichenmotore sind jeweils zwei Endstufen gegenläufig gepolt, daher werden hier nur vier Selectrix-Bits benötigt. Auf einem Selectrix-Kanal (8 bit) kann ein wdec1 auf die unteren vier Bit lauschen, ein weiterer unabhängig davon die oberen vier Bit auswerten. Zur Ansteuerung von Signallampen kann man die acht Bit unabhängig voneinander verwenden. In diesem Modus ist daher ist die Programmierung der Ausgänge anders: ein Ausgang gleich ein SX-Bit.

Insgesamt hat der Decoder die folgenden Einstellmöglichkeiten:
SX-Kanal 0: Decoder-Adresse
SX-Kanal 1: Sub-Adresse (untere 4 Bit/ obere 4 Bit)
SX-Kanal 2: Weichenmotor- oder Signaldecoder.
SX-Kanal 3: Seriennummer (high byte)
SX-Kanal 4: Seriennummer (low byte)

Die Seriennummer dient dem einfachen Unterscheiden von Decodern gleichen Typs.

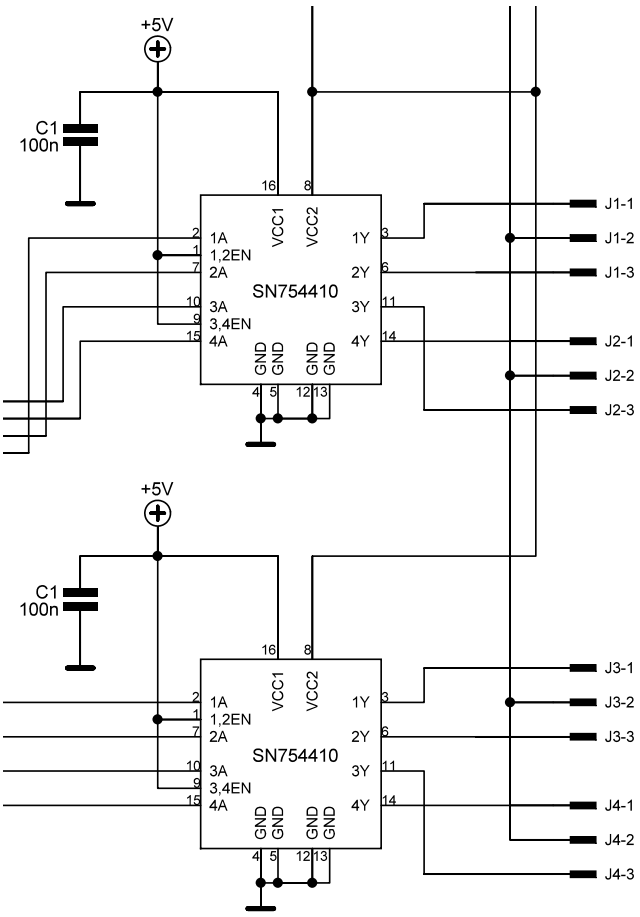
Während sich die SX-Decodierungs-Funktionen nicht sehr vom Arduino-Code unterscheiden, gibt es zusätzliche Routinen zum Lesen und Schreiben des EEPROM im AVR-Prozessor. EPROM-Speicherzellen verlieren ihre Informationen im Gegensatz zum RAM (und damit zum Programmspeicher) nicht

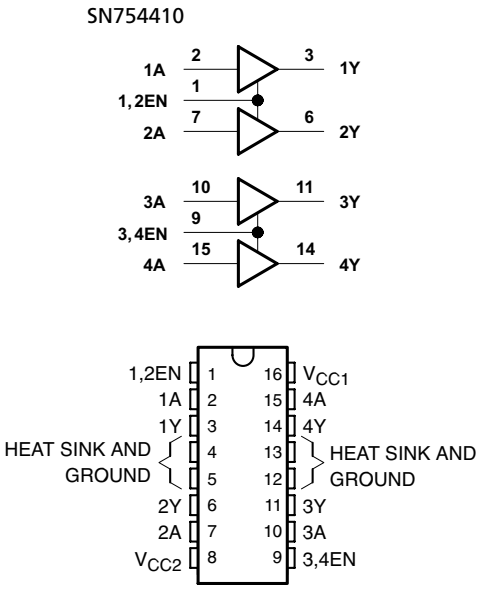
FIG. 1 – WEICHEN/SIGNALDECODER SOFTWARE – ÜBERSICHT



Aufbau des SX Datenstroms. Dargestellt ist nur ein Teil der 16 ausgesandten Pakete. Es bedeuten: B0 .. B7 = Bits 0–7; „1“ = log. 1; S = Sync (0 0 0 1); A = Spannungs- und Trennbits und Basisadresse; D0 .. D6 = Unterpakete.

LEISTUNGSTREIBER UND H-BRÜCKEN





FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)
† In the thermal shutdown mode, the output is in a high-impedance state regardless of the input levels.

Der verwendete Baustein SN754410 ist eigentlich für die Ansteuerung von Brushless-DC-Motoren entwickelt worden. Da man die Stromrichtung durch eine Last mit vier Schaltern in einem H-förmigen Aufbau umsteuern kann, spricht man bei elektronischen Motorendstufen von H-Brücken. Eine halbe H-Brücke entspricht einem der senkrechten Striche des H. Die Schalter sind Transistoren in FET-Technik mit (sehr) geringem on-Widerstand. Da es bei der Auslegung der Transistoren einen Unterschied macht, ob sie zwischen positiver Versorgung und Last oder zwischen Last und Masse schalten sollen, unterscheidet man bei Einzelbausteinen zwischen Hi- und Lo-Treibern. In den meisten Treiber-ICs sind je ein Hi- und Lo-Treiber mit einer passenden Ansteuerschaltung, die verhindert, dass beide Transistoren gleichzeitig leiten, zu einer Halb-H-Treiberstufe zusammengefasst. Das SN754410 enthält gleich vier Halb-H-Stufen, kann also einen Brushless-Motor mit zwei Spulen oder aber zwei Doppelspulen-Magnetantriebe ansteuern.

beim Abschalten der Stromversorgung. Um die Decoder-Einstellungen dauerhaft zu speichern, werden die Daten im EEPROM hinterlegt.

MAIN-LOOP

Der Kern fast aller Microcontroller-Programme ist eine Endlos-Schleife, das Programm soll so lange laufen, wie die Stromversorgung eingeschaltet ist. In dieser Schleife (meist „main()“ genannt) wird der Taster abgefragt („Soll der Programmiermodus eingeschaltet werden?“), werden die aktuellen Werte des SX-Kanals des Decoders aus einer globalen Variablen gelesen („Wie sind die Werte der acht Bit meiner Decoderadresse?“) und werden dann die Treiber für die Weichenmotoren angesteuert mit jeweils VCC (positive Versorgungsspannung) an einem Anschluss des Motors und GND (Masse) am anderen.

Es ist keine Abschaltung der Motorausgänge vorgesehen, da die von uns verwendeten Tortoise®-Antriebe dies nicht brauchen und die meisten anderen Weichenmotoren (z.B. von Conrad) einen eingebauten Endabschalter haben.

Im Gegensatz zum Arduino, der über eine USB-Schnittstelle und einen Bootlader verfügt (d.h. bereits beim ersten Einschalten läuft auf dem Microprozessor ein Programm, das auf Befehle auf dem USB-Bus lauscht und in der Lage ist, weitere Software in den Controller zu laden), kommt der ATtiny2313 zunächst einmal ohne jede Program-

mierung. Um ihm diese zuzuweisen ist eine spezielle Hardware notwendig. Die Programmierung ist zwar direkt in der fertigen Schaltung möglich („in-circuit-programming"), aber man braucht einen entsprechenden Programmer, z.B. den AVRISP.

Außerdem ist ein Programm zum Übersetzen des C-Codes notwendig wie z.B. Atmel-Studio oder, unter Linux die Kombination aus Eclipse, der avr-gcc-toolchain sowie avrdude. Wenn Sie den unveränderten Binär-Code verwenden wollen, so können Sie den Binärfile (sx-wdec1.hex, siehe Webseite) auch ohne Compilieren direkt in den Flash-Speicher des ATtiny2313 schreiben (mit avrdude-Programm und der AVRISP-Hardware). Neben dem Binärcode braucht der ATtiny noch ein paar Einstellungen, mit denen ihm mitgeteilt wird, dass er z.B. den internen Oszillator mit 8 MHz Taktfrequenz verwenden soll. Diese Programmierung wird bei AVR Prozessoren in die sogenannte „Fuses" geschrieben.

Die AVR-Entwicklungsumgebungen sind auf den unten genannten Weblinks bereits ausführlich beschrieben worden. Es ist daher nicht sinnvoll, sie hier in der DiMo zu wiederholen.

Damit sind wir am Ende unserer Reihe über das Selbstbauprojekt „Selectrix-Weichen- und -Signaldecoder" angekommen. Um den Nachbau zu erleichtern, können Sie eine Leer-Platine (solange Vorrat reicht) und auch die Spezialbauteile (programmierter ATtiny2313, Treiber SN754410) über den IBM-Klub-Böblingen, Sparte Modellbahn, beziehen. Bitte kontaktieren Sie mich bei Interesse über die unten stehende Email-Adresse oder den Verlag

Michael Blank
michael@oscale.net.

LINKS

Links zur Selectrix-Bibliothek: http://www.oscale.net/sx-wdec

AVR Programmierung: http://www.mikrocontroller.net/articles/AVR

Atmel Studio: http://www.atmel.com/microsite/atmel_studio6