MIDIfyME

by raffael segmüller

Version	Datum	Änderung	Visum
1.0	12.10.2007	Draft	raffi
1.1	29.04.2008	Corrections, some more text	raffi
1.2	20.05.2008	Alles auf Deutsch. Einfügen erster Bilder vom Board, Hardwarebeschreibungen vom Board	raffi
1.3	02.08.2008	Code Revision und neue Schematix Code getestet mit Kanalwahl und invertbit	raffi

1. Vorwort

Mit dem MIDIfyME Board soll ermöglicht werden Geräte welche keine MIDI Schnittstelle besitzen mit ebensolcher zu steuern. Natürlich ist der Wahl der Anschlusspunkte freie Wahl gegeben sodass nicht bloss Noten sondern auch Bends gesteuert werden wollen.

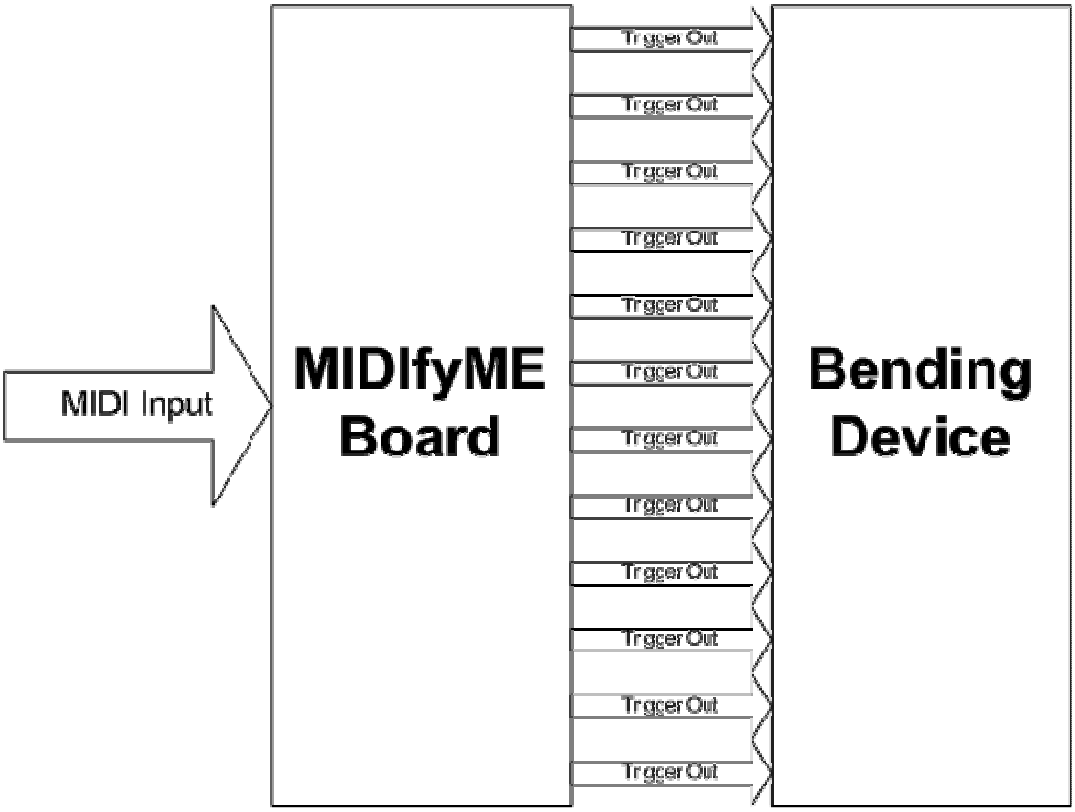
Das MIDI Interface ist mit einem ATmega168 Prozessor aufgebaut, welcher auch in der Plattform Arduino verwendet wird. Der Bootloader von Arduino ist ebenfalls programmiert, so kann der uP herausgenommen und im Arduino Board neu programmiert werden.

Die Triggerausgänge werden über Optokoppler vom Prozessor angesteuert. So sind die Bending Kontakte elektrisch vom Prozessor getrennt, was Fehlfunktionen oder gar die Zerstörung des Gerätes vermeiden soll.

Die Software soll ausschliesslich "nonCommercial" eingesetzt werden.

Tnx for fair use!!

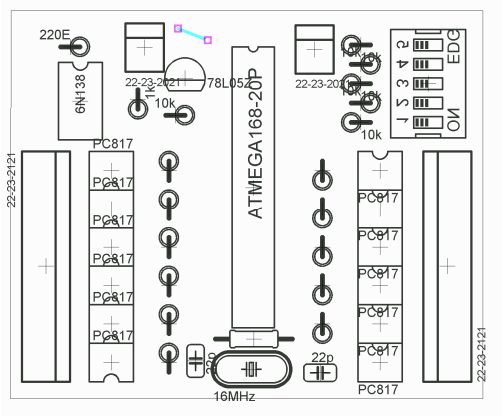
1.1. Prinzipschema



2. Hardware Teil

Alles was zum Aufbau des Boards benötigt wird.

2.1. Bestückungsplan

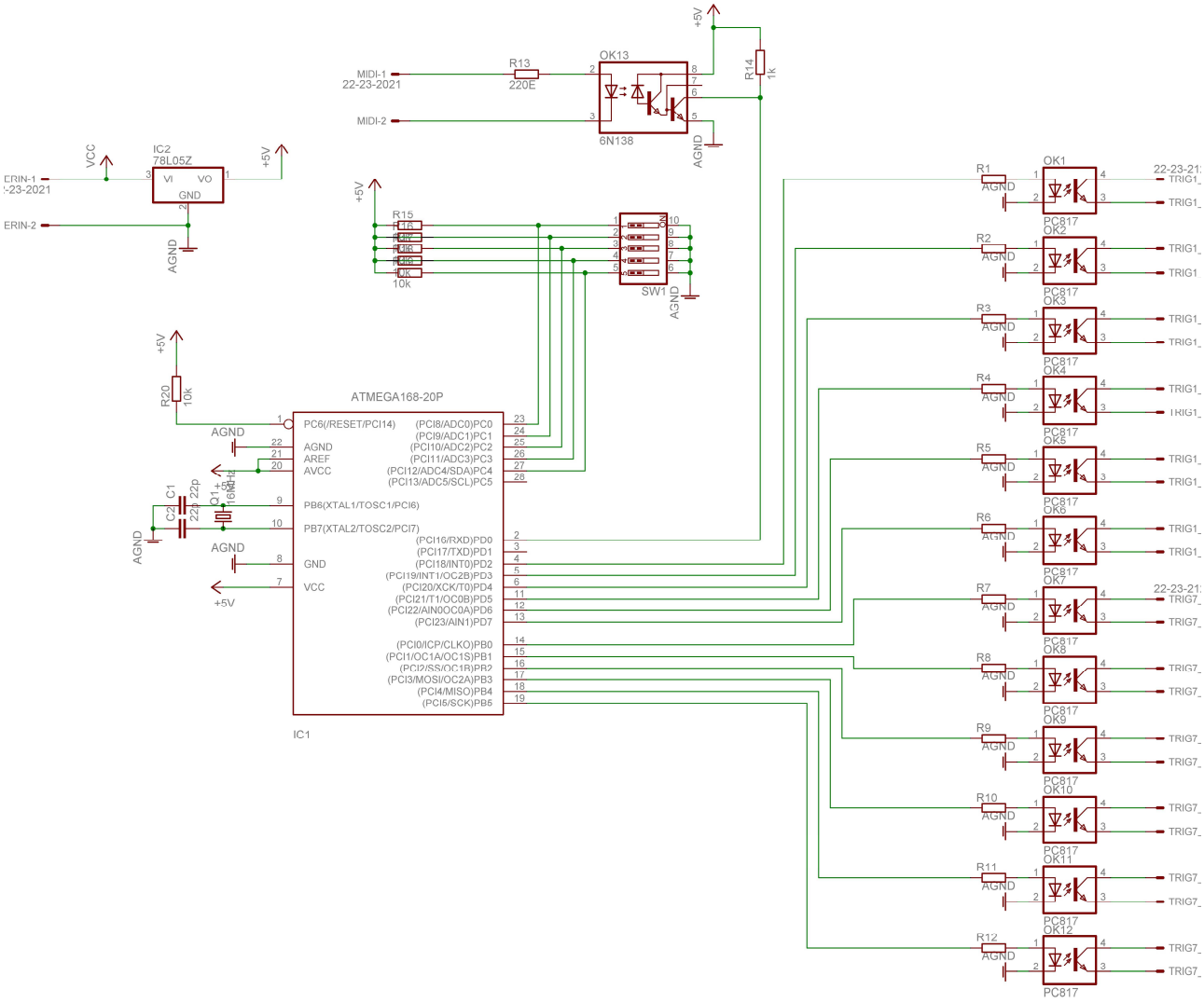


Bestückungsreihenfolge (bottom - up)

- Drahtbrücke
- IC Sockel vom Prozessor
- Optokoppler
- Quarz und Kondensatoren
- Widerstände
- DIP Switch
- Spannungsregler LM7805
- Steckerleisten

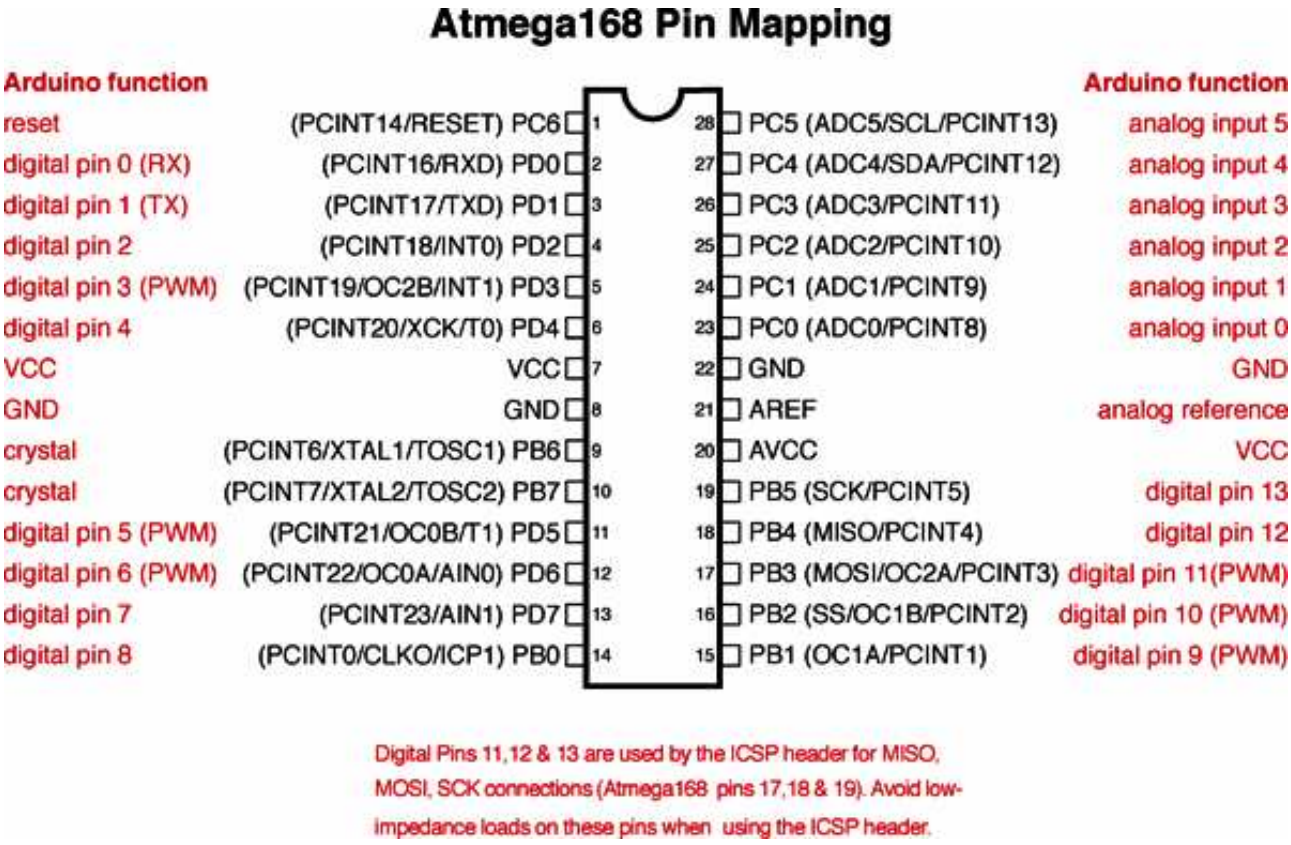
2.2. Schema

zur Übersicht noch das Schema:



2.3. Details zum verwendeten Prozessor

Hier noch die Pinbelegungen des Processors (Quelle: Arduino.cc)



3. Funktionsbeschreibungen

3.1. MIDI Eingang

Die MIDI Schnittstelle ist als Current-Loop aufgebaut. Und wird standardmässig mit einem Optokoppler aufgebaut. Der 6N138.

Es werden Noten von C1 bis H1 den Triggerausgängen 1 bis 12 zugeordnet

Der MIDI Kanal wird mit dem DIP Switch eingestellt.

3.2. Trigger Ausgänge

Auf dem Board ist für jeden Triggerausgang ein Optokoppler mit Transistorausgang vorhanden. So werden die gewünschten Anschlüsse gut entkoppelt.

Man kann natürlich auch direkt am Ausgang des Prozessors seine Drähte anschliessen. Was aber zu unerwünschten wie aber auch gewünschten Effekten führen kann (Probieren mit Poti also Vorwiderstand).

3.3. DIP Switch

3.3.1. DIP Switch Nummer 1-4 (MIDI Kanal)

Mit den ersten vier Schaltern des DIP Switch kann der MIDI Empfangskanal eingestellt werden. Die Eingabe erfolgt Binär, anbei eine Tabelle als Einstellhilfe:

DIP Nummer->	**1**	**2**	**3**	**4**
Kanal 1	0	0	0	0
Kanal 2	0	0	0	1
Kanal 3	0	0	1	0
Kanal 4	0	0	1	1
Kanal 5	0	1	0	0
Kanal 6	0	1	0	1
Kanal 7	0	1	1	0
Kanal 8	0	1	1	1
Kanal 9	1	0	0	0
Kanal 10	1	0	0	1
Kanal 11	1	0	1	0
Kanal 12	1	0	1	1
Kanal 13	1	1	0	0
Kanal 14	1	1	0	1
Kanal 15	1	1	1	0
Kanal 16	1	1	1	1

Tabelle: MIDI Kanalwahl Einstellungen am DIP Switch

3.3.2. DIP Switch Nummer 5 (Invert)

Mit dem fünften Schalter am DIP Switch können alle Ausgänge invertiert werden. D.h. die Ausgänge der Optokoppler sind dann permanent durchgeschaltet. Je nach angeschlossenem Gerät kann dies nützlich sein.

Die Positionen des DIP Switch werden beim Start des Prozessors eingelesen und können nicht während dem Betrieb geändert werden.

Wenn der MIDI Kanal geändert werden soll muss das Gerät kurz aus- und eingeschaltet werden um den Prozessor neuzustarten und die Werte neu einzulesen.

4. Software

Für die Interessierten ein paar Worte zu MIDI und zur Software auf dem Prozessor.

4.1. MIDI Details

4.1.1. Note On Message im Detail

Besteht aus 3 Bytes: Statusbyte, Notebyte und Velocitybyte

Statusbyte: Erstes Nibble -> message type (Note On - Off, CC, ...)
Zweites Nibble -> MIDI Kanal

Notebyte: MSB immer 0
Rest vom Byte -> Notennummer 0- 127 (C0 – C9 ?)

Velocitybyte: MSB immer 0
Rest vom Byte -> Velocity Wert von 0 – 127

Beispiel:

	1001 xxxx	0xxx xxxx	0xxx xxxx
	Statusbyte	**Notebyte**	**Velocitybyte**
Hex	9 0	2 4	6 4
Dezimal	144	36	100
Binär	1001 0000	0010 0100	0110 0100

4.1.2. Note Off Message im Detail

Besteht aus 2 Bytes. Statusbyte und Notebyte.

Das Velocitybyte wird nicht benötigt da die Note ja "abgeschaltet" wird.

Tatsache ist aber das die meisten Keyboards und Sequenzer keine richtigen NoteOff's schicken sondern ein NoteOn mit 0 Velocity.

Der Code kann beides interpretieren.

4.2. Code auf dem Atmega168

```
//variables setup

byte incomingByte;
byte note;
byte velocity;

int midichannel;

int val1 = 0;           // the values of the DIP Switch will go into theese ones
int val2 = 0;
int val3 = 0;
int val4 = 0;
int invertbit_val = 0;

int dip1 = 0;           // labels for the analog inputs of the DIP switch
int dip2 = 1;
int dip3 = 2;
int dip4 = 3;
int invertbit = 4;

int statusLed = 13;         // select the pin for the LED

int action=2;               //0 =note off ; 1=note on ; 2= nada


//setup: declaring iputs and outputs and begin serial
void setup() {
  pinMode(statusLed,OUTPUT);   // declare the LED's pin as output
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
```

```
  // initialize all triggers, set them one time to a state
  digitalWrite(1,LOW);
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);

  // read analogues - to set midichannel and invertbit
  val1 = analogRead(dip1);    // read the values from DIP 1 to 5
  val2 = analogRead(dip2);
  val3 = analogRead(dip3);
  val4 = analogRead(dip4);
  invertbit_val = analogRead(invertbit);

  // set the midichannel
  if      ((val1 > 500) && (val2 > 500) && (val3 > 500) && (val4 > 500))  // 16 -
actually it 15 ... bad computers...
    { midichannel = 15; }
  else if ((val1 > 500) && (val2 > 500) && (val3 > 500) && (val4 < 500))  // 15
    { midichannel = 14; }
  else if ((val1 > 500) && (val2 > 500) && (val3 < 500) && (val4 > 500))  // 14
    { midichannel = 13; }
  else if ((val1 > 500) && (val2 > 500) && (val3 < 500) && (val4 < 500))  // ..
    { midichannel = 12; }
  else if ((val1 > 500) && (val2 < 500) && (val3 > 500) && (val4 > 500))
    { midichannel = 11; }
  else if ((val1 > 500) && (val2 < 500) && (val3 > 500) && (val4 < 500))
    { midichannel = 10; }
  else if ((val1 > 500) && (val2 < 500) && (val3 < 500) && (val4 > 500))
    { midichannel =  9; }
  else if ((val1 > 500) && (val2 < 500) && (val3 < 500) && (val4 < 500))
    { midichannel =  8; }
  else if ((val1 < 500) && (val2 > 500) && (val3 > 500) && (val4 > 500))
    { midichannel =  7; }
  else if ((val1 < 500) && (val2 > 500) && (val3 > 500) && (val4 < 500))
```

```
    { midichannel =  6; }
  else if ((val1 < 500) && (val2 > 500) && (val3 < 500) && (val4 > 500))
    { midichannel =  5; }
  else if ((val1 < 500) && (val2 > 500) && (val3 < 500) && (val4 < 500))
    { midichannel =  4; }
  else if ((val1 < 500) && (val2 < 500) && (val3 > 500) && (val4 > 500))
    { midichannel =  3; }
  else if ((val1 < 500) && (val2 < 500) && (val3 > 500) && (val4 > 500))  // ..
    { midichannel =  2; }
  else if ((val1 < 500) && (val2 < 500) && (val3 < 500) && (val4 > 500))  //  2
    { midichannel =  1; }
  else if ((val1 < 500) && (val2 < 500) && (val3 < 500) && (val4 < 500))  //  1
    { midichannel =  0; }

  // start serial with midi-baudrate 31250
  // or enter 38400 for debugging over serial interface
  Serial.begin(31250);
  digitalWrite(statusLed,HIGH);

}


//loop: wait for serial data, and interpret the message
void loop ()
  {
  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();          // read the incoming byte:

    // wait for as status-byte,
    // note on or off
    if (incomingByte == (144 + midichannel))   // note on message starting starting  144
      {
      action=1;
      }
     else if (incomingByte== (128 + midichannel))   // note off message starting  128
      {
      action=0;
      }
else if ( (action==0)&&(note==0) )                // if we received a "note off", we
                                                  //wait for which note (databyte)
      {
```

```
      note=incomingByte;
      playNote(note, 0);
      note=0;
      velocity=0;
      action=2;
      }
     else if ( (action==1)&&(note==0) )     // if we received a "note on",
                                            // we wait for the note (databyte)
      {
      note=incomingByte;
      }
     else if ( (action==1)&&(note!=0) )     // ...and then the velocity
      {
      velocity=incomingByte;
      playNote(note, velocity);
      note=0;
      velocity=0;
      action=0;
      }
     else
      {
      //nada
      }
  }
}


void playNote(byte note, byte velocity)
{
  int value=LOW;
  if ( ((velocity > 10) && (invertbit_val < 500)) || ((velocity < 10) && (invertbit_val
> 500)) )       // note on,  not inverted
    {
    value = HIGH;
    }
  if (note>=36 && note<44)         //since we don't want to "play" all notes
                                   // we wait for a note between 36 & 44 (C0-C1)
    {
    byte myPin = note - 34;        // to get a pinnumber between 2 and X
    digitalWrite(myPin, value);
    }
}
```