3.7 MIDI-Input zur Anzeige oder Ansteuerung der 128 MIDI-Töne

Das digitale Musikinterface MIDI beruht auf der seriellen Übertragung von Kommandos und Daten mit einer Breite von 8 Bit. Die seriellen Daten haben den gleichen Aufbau wie bei den seriellen Interfacen von Computern mit einem RS232-Interface. Der Unterschied besteht nur in der Festlegung der logischen Pegel. Während sie beim RS232 auf +12V und -12V festgelegt sind, betragen sie beim MIDI 0 mA und 5 mA. Es handelt sich also um ein Stomschleifeninterface, bei dem auf der Empfängerseite durch einen Optokoppler auch eine galvanische Trennung der über das MIDI verbundenen Geräte erreicht wird. Für MIDI ist eine serielle Datenrate von 31250 Baud festgelegt. Weiterhin ist das Interface auf ein Stopbit und keine Parität festgelegt. Damit müssen zur Übertragung eines Bytes immer 10 Bit (1 Startbit + 8 Datenbit + 1 Stopbit) gesendet werden. Die effektive Datenrate ergibt sich also zu 3125 Byte/Sekunde. Die Länge einer 1/16 Note beträgt meist mehr als etwa 90 ms. Im MIDI sind 6 Takte pro 1/16 Note festgelegt. Die Periodendauer des MIDI-Taktes muß also etwa 15 ms betragen. In dieser Zeit können dann 46 Byte übertragen werden. Da ein Kommando zum Einschalten oder Ausschalten eines Tones aus 3 Byte besteht, können in einer Taktperiode 15 Kommandos übertragen werden. Um diese Anzahl noch zu vergrößern, muß in modernen MIDI-Systemen das Kommandobyte in mehreren aufeinanderfolgenden gleichartigen Kommandos nur bei der 1. Note gesendet werden. Damit auch für das Ausschalten eines Tones das gleiche Kommando wie beim Einschalten verwendet werden kann, benutzen moderne MIDI-Geräte das Einschaltkommando mit der Anschlagstärke = 0 zum Ausschalten eines Tones. Damit müssen für n Tonkommandos nur noch 2*n+1 Byte übertragen werden. Bei 46 Byte pro MIDI-Takt sind damit (46-1)/2=22 Kommandos übertragbar. Der MIDI-Takt wird im 68HC11 aus dem internen Takt von 2 MHz mit Hilfe eines Timerkanals gewonnen. Der Teilerwert für den Timer ergibt sich also zu 15ms / 0.5µs = 30000. Um diesen Wert wird dann auch immer das Vergleichsregister des Timers in der Interuptbehandlung erhöht, so das genau nach 30000 Taktimpulsen des 2 MHz Systemtaktes ein neuer Interupt ausgelöst wird. In dieser Interuptbehandlung wird dann auch immer der Takt für die Ausgangstreiberkette aus 74HCT374 erzeugt und somit werden immer nur zu diesem Zeitpunkt die Ausgänge umgeschalten. Das Interuptbehandlungsprogramm beginnt mit der Marke **frq_ir** am Ende des Listings. Mit den Befehlen

```
bclr    $20,oportb,x    und
bset    $20,oportb,x
```

wird am Bit 5 von Port B der Taktimpuls für die Ausgangstreiber 74HCT374 erzeugt. Damit keine Daten am MIDI-Input verloren gehen werden die empfangen Byte in einer 2. Interuptroutine für das serielle Interface in einen Ringpuffer eingetragen. Die Interuptroutine beginnt mit der Marke **sciint** im Assemblerlisting und steht unmittelbar vor der Timerroutine. Beim Empfang eines Bytes wird dieses in den Puffer im RAM von Marke **pu_anf** bis **pu_end** eingetragen und der Schreibpointer auf der Adresse **poi_in** incrementiert. Dabei wird das obere Pufferende überwacht und beim Erreichen wieder auf den Pufferanfang zurückgesetzt, so das sich ein Ringpuffer ergibt. Das Eigentliche Steuerprogramm list dann die Byte aus diesem Puffer mit dem Unterprogramm **read** und benutzt dazu den Pointer auf der Adresse **poi_out**. Dieses Unterprogramm liest immer dann ein Byte aus dem Puffer, wenn der Bytezähler auf der Adresse **anz_zei** nicht null ist. Der Inhalt dieser Speicherstelle wird immer beim Eintragen eines Bytes durch die Interuptroutine **sciint** incrementiert und beim auslesen des Puffers mit dem Unterprogramm **read** decrementiert. Die Interuproutine überwacht dann noch einige Fehlerzustände des seriellen Interface und den Pufferüberlauf und zeigt diese mit den Bits PA7 für Interfacefehler und PA5 für Pufferüberlauf an. PA5 dürfte nie aktiv werden, da der Processor auf keinerlei externe Signale bei der Ausgabe auf die Latchschaltkreise 74HCT374 warten muß und für den Empfang eines Byte vom MIDI schon 320µs gebraucht werden, während das Programm in seiner längsten Schleife zur Auswertung von 3 Byte für eine Note mit 3 SCI-Interupts und einem Timerinterupt nur 869 Takte = 434,5µs benötigt. PA7 kann bei defekten, zulangen oder ungeschirmten Kabeln oder bei defekten in den Empfängern oder Sendern gesetzt werden. Eine LED an diesem Ausgang könnte das optisch signalisieren. Am Ausgang PA6 steht der halbe interne MIDI-Takt zur Verfügung. Im MIDI-Steuerprogramm von der Marke **midi_zy** bis **read** werden dann die empangenen MIDI-Sequenzen ausgewertet und die entsprechenden Bits in den 74HCT374 durch das Steuerprogramm ab der Marke **out** gesetzt oder gelöscht. Der Zustand aller 128 möglichen Ausgänge wird außerdem auf den Speicherstellen **okt01** bis **okt16** im RAM gespeichert, dadurch muß keine Hardware zum Zurücklesen der Ausgangsbits realisiert werden. Beim

Setzen wird dann immer nur das Bit zum Inhalt des zugehörigen Bytes mit einer OR-Verknüpfung gesetzt und auf den entsprechenden 74HCT374 ausgegeben. Das Löschen erfolgt mit einem AND-Befehl und einer inversen Bitmaske auf die gleiche Weise. Die benötigte Hardware ist im folgenden Bild zu sehen.

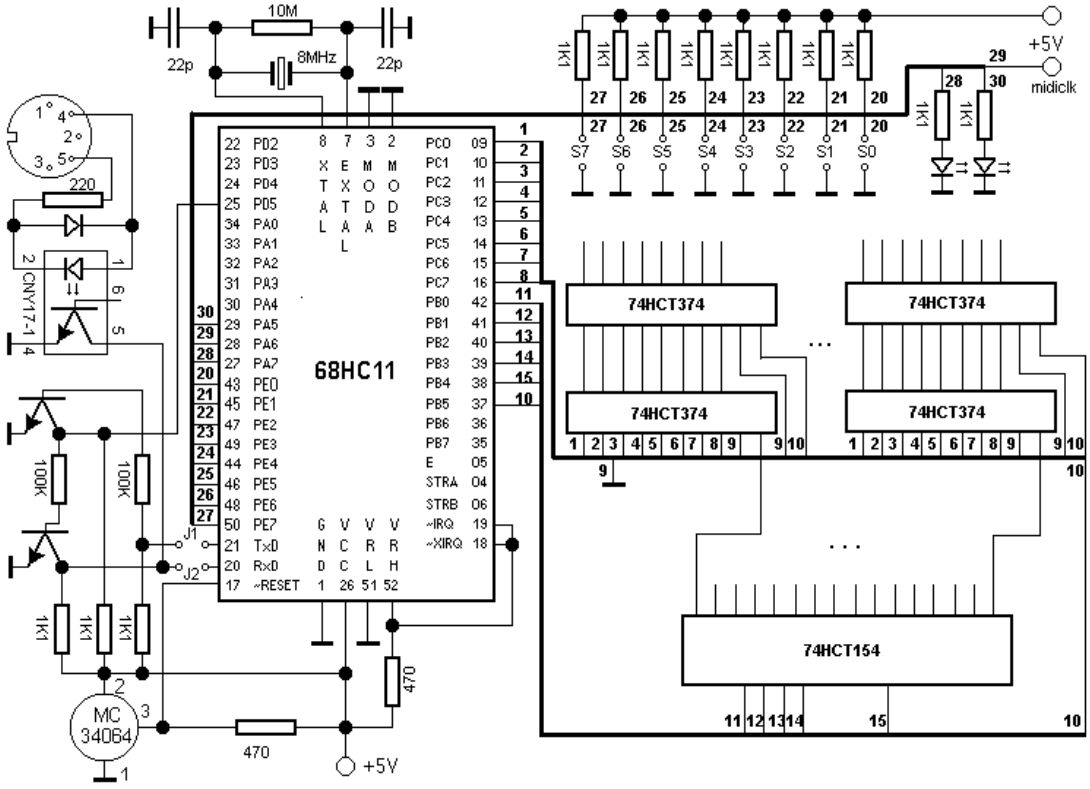

Bild1: Schaltbild des MIDI-Input mit einem 68HC11A8

Der Materialaufwand für die obige Schaltung wird etwa bei 70,- DM liegen. Als Netzteil kann ein 5V Steckernetzteil verwendet werden. Die Stromaufnahme der obigen Schaltung ist kleiner als 40 mA. Beim Programmieren des EEPROM im 68HC11 sollte aber sicherheitshalber ein Netzteil verwendet werden, das bis etwa 1A belastbar ist. Bei der Wahl des Netzteiles sollte aber auch die Versorgung der anzusteuernden Elektronik berücksichtigt werden und gegebenenfals daraus mit einem 7805 die Spannung für die obige Schaltung gewonnen werden. Für die Programmierung des EEPROM im 68HC11 müssen die Leitung TxD und RxD über einen V24 Pegelwandler (MC145407) mit RxD und TxD von einem PC verbunden werden. Dazu sind in der Schaltung die Brücken J1 und J2 vorgesehen. Nach öffnen beider Brücken kann dann ein Pegelwandler an TxD und RxD des 68HC11 angeschlossen werden. Das Listing des Steuerprogramm ist auf den folgenden Seiten abgedruckt. Die verarbeiteten MIDI-Sequenzen sind am Anfang als Kommentar aufgeführt. Eine Liste aller MIDI-Sequenzen ist unter

http://www.midi.org/about-midi/table1.htm

zu finden.

```
file: midi.a

************** Beispiel für MIDI-IN ***************************************************
*                                                                                     *
* Date          Version       What                                              Who *
* 17.02.2000    2.00          create                                            Scö *
* ---------------------------------------------------------------------------------- *
*                             - Notes -                                               *
* Es wird ein MIDI-Input realisiert, der für die maximal 128 Töne im MIDI             *
* je einen Ausgang schalten kann. Da die gängigen Keybourds nur 61, 73                *
* oder 88 Tasten haben, kann das System auf 64, 80 oder 88 Ausgänge ab-               *
* gerüstet werden. Mit 5 DIL-Schaltern kann dann der Ausgang 0 in 1er                 *
* Schritten nach oben verschoben werden, so das er der Nummer 36 bei 61               *
* Tasten, 24 bei 72 Tasten und 21 bei 88 Tasten entspricht. Da mit 5 DIL-             *
* Schaltern nur Werte von n=0 ... 31 eingestellt werden können, wird im               *
* Programm immer eine 10+n von der Notennummer abgezogen. 3 weitere DIL-              *
* Schalter werden zur Einstellung des MIDI-Kanals verwendet. Damit ist                *
* der Modul für die MIDI-Kanäle 1 bis 8 verwendbar.                                   *
* Folgende MIDI-Sequenzen werden ausgewertet:                                         *
*               Kommando      Tonnummer             Anschlagstärke                    *
* - RESET       11111111   (gibt internen Clock wieder frei)                          *
* - MIDI Clock  11111000   (deaktiviert internen Clock)                               *
* - AKTIV       11111110   (wird zum MIDI-OUT durchgereicht)                          *
* - CLEARall    1011kkkk      0111100x              00000000                          *
*   oder        1011kkkk      01111011              00000000                          *
* - PEDAL   EIN 1011kkkk      01000000              0xxxxxxx       >0                 *
* - PEDAL   AUS 1011kkkk      01000000              00000000                          *
* - S_Pedal EIN 1011kkkk      01000011              0xxxxxxx       >0                 *
* - S_Pedal AUS 1011kkkk      01000011              00000000                          *
* - Ton AUS     1000kkkk      0nnnnnnn              0xxxxxxx                          *
*   oder        1001kkkk      0nnnnnnn              00000000                          *
* - Ton EIN     1001kkkk      0nnnnnnn              0vvvvvvv       >0                 *
*                                                                                     *
* Folgen mehrere gleichartige Kommandos hinter einander, dann muß nur bei             *
* dem 1. Ton das Kommando gesendet werden. Ihm folgen dann immer nur noch             *
* die Tonnummern mit den Anschlagstärken entsprechend der Anzahl gleich-              *
* zeitig ein oder auszuschaltender Töne.                                              *
*                                                                                     *
*    kkkk                     =      Kanalnummer von 0 ... 15 (Die Kodierung 0        *
*                                    entspricht dem MIDI-Kanal 1                      *
*                                                                                     *
*    nnnnnnn                  =      Nummer des Tones von 0 ... 127                   *
*                                                                                     *
*    vvvvvvv                  =      Anschlagstärke von 1 ... 127                     *
*                             =      0 entspricht Note AUS !!!                        *
*                                                                                     *
*    x                        =      Bit kann beliebigen Wert annehmen                *
*                                                                                     *
* ---------------------------------------------------------------------------------- *
*                             - Usage -                                               *
* ---------------------------------------------------------------------------------- *
* Copyright (C) Dr. Siegmar Schöne                                                    *
*************************************************************************************



*****************             **********************************************
*****************             *                                            *
*****************             *           Kontanten für MIDI-IN            *
*****************             *                                            *
*****************             **********************************************
file: e:\prak\zwerg11p\lib.a\hc11.h

       0000b600 EEPROM        equ    $b600          * eeprom area

       00001000 REG           equ    $1000          * register base

       00001000 porta         equ    REG+$00        * port a data reg.
       00001002 pioc          equ    REG+$02        * parallel i/o control reg.
       00001003 portc         equ    REG+$03        * port c data reg.
       00001004 portb         equ    REG+$04        * port b data reg.
       00001005 portcl        equ    REG+$05        * port c latched data reg.
       00001007 ddrc          equ    REG+$07        * port c data direction reg.
       00001008 portd         equ    REG+$08        * port d data reg.
       00001009 ddrd          equ    REG+$09        * port d data direction reg.
       0000100a porte         equ    REG+$0a        * port e data reg.
```

```
0000100b cforc        equ     REG+$0b         * timer compare force reg.
0000100c oc1m         equ     REG+$0c         * output compare 1 mask reg.
0000100d oc1d         equ     REG+$0d         * output compare 1 data reg.
0000100e tcnt         equ     REG+$0e         * timer count reg. 16 bit
0000100e tcnth        equ     REG+$0e         * timer count reg. high
0000100f tcntl        equ     REG+$0f         * timer count reg. low
00001010 tic1h        equ     REG+$10         * timer input capture reg. high
00001011 tic1l        equ     REG+$11         * tic1 low
00001012 tic2h        equ     REG+$12         * tic2 high
00001013 tic2l        equ     REG+$13         * tic2 low
00001014 tic3h        equ     REG+$14         * tic3 high
00001015 tic3l        equ     REG+$15         * tic3 low
00001016 toc1h        equ     REG+$16         * timer output compare reg. high
00001017 toc1l        equ     REG+$17         * toc1 low
00001018 toc2h        equ     REG+$18         * toc2 high
00001019 toc2l        equ     REG+$19         * toc2 low
0000101a toc3h        equ     REG+$1a         * toc3 high
0000101b toc3l        equ     REG+$1b         * toc3 low
0000101c toc4h        equ     REG+$1c         * toc4 high
0000101d toc4l        equ     REG+$1d         * toc4 low
0000101e toc5h        equ     REG+$1e         * toc5 high
0000101f toc5l        equ     REG+$1f         * toc5 low
00001020 tctl1        equ     REG+$20         * timer control
00001021 tctl2        equ     REG+$21         * timer control reg. 2
00001022 tmsk1        equ     REG+$22         * main timer interrupt mask reg.1
00001023 tflg1        equ     REG+$23         * main timer interrupt flag reg.1
00001024 tmsk2        equ     REG+$24         * misk. timer interrupt mask reg.2
00001025 tflg2        equ     REG+$25         * misc. timer interrupt flag
00001026 pactl        equ     REG+$26         * pulse accumulator control reg.
00001027 pacnt        equ     REG+$27         * pulse accumulator count reg.
00001028 spcr         equ     REG+$28         * spi control reg.
00001029 spsr         equ     REG+$29         * spi status reg.
0000102a spdr         equ     REG+$2a         * spi data reg.
0000102b baud         equ     REG+$2b         * sci baud rate control reg.
0000102c sccr1        equ     REG+$2c         * sci control reg. 1
0000102d sccr2        equ     REG+$2d         * sci control reg. 2
0000102e scsr         equ     REG+$2e         * sci status reg.
0000102f scdr         equ     REG+$2f         * sci data reg.
00001030 adctl        equ     REG+$30         * a/d control/status reg.
00001031 adr1         equ     REG+$31         * a/d result reg. 1
00001032 adr2         equ     REG+$32         * a/d result reg. 2
00001033 adr3         equ     REG+$33         * a/d result reg. 3
00001034 adr4         equ     REG+$34         * a/d result reg. 4
00001035 bprot        equ     REG+$35         * block protect reg.
00001039 option       equ     REG+$39         * system configuration options
0000103a coprst       equ     REG+$3a         * arm/reset cop timer circuitry
0000103b pprog        equ     REG+$3b         * eeprom programming reg.
0000103c hprio        equ     REG+$3c         * highest priority interrupt and misc.
0000103d init         equ     REG+$3d         * ram and i/o mapping reg.
0000103e test1        equ     REG+$3e         * factory test reg.
0000103f config       equ     REG+$3f         * configuration control reg.

00000000 oporta       equ     $00             * port a data reg.
00000002 opioc        equ     $02             * parallel i/o control reg.
00000003 oportc       equ     $03             * port c data reg.
00000004 oportb       equ     $04             * port b data reg.
00000005 oportcl      equ     $05             * port c latched data reg.
00000007 oddrc        equ     $07             * port c data direction reg.
00000008 oportd       equ     $08             * port d data reg.
00000009 oddrd        equ     $09             * port d data direction reg.
0000000a oporte       equ     $0a             * port e data reg.
0000000b ocforc       equ     $0b             * timer compare force reg.
0000000c ooc1m        equ     $0c             * output compare 1 mask reg.
0000000d ooc1d        equ     $0d             * output compare 1 data reg.
0000000e otcnth       equ     $0e             * timer count reg. high
0000000f otcntl       equ     $0f             * timer count reg. ow
00000010 otic1h       equ     $10             * timer input capture reg. high
00000011 otic1l       equ     $11             * tic1 low
00000012 otic2h       equ     $12             * tic2 high
00000013 otic2l       equ     $13             * tic2 low
00000014 otic3h       equ     $14             * tic3 high
00000015 otic3l       equ     $15             * tic3 low
00000016 otoc1h       equ     $16             * timer output compare reg. high
00000017 otoc1l       equ     $17             * toc1 low
00000018 otoc2h       equ     $18             * toc2 high
00000019 otoc2l       equ     $19             * toc2 low
```

```
       0000001a otoc3h        equ     $1a             * toc3 high
       0000001b otoc3l        equ     $1b             * toc3 low
       0000001c otoc4h        equ     $1c             * toc4 high
       0000001d otoc4l        equ     $1d             * toc4 low
       0000001e otoc5h        equ     $1e             * toc5 high
       0000001f otoc5l        equ     $1f             * toc5 low
       00000020 otctl1        equ     $20             * timer control reg. 1
       00000021 otctl2        equ     $21             * timer control reg. 2
       00000022 otmsk1        equ     $22             * main timer interrupt mask reg.1
       00000023 otflg1        equ     $23             * main timer interrupt flag reg.1
       00000024 otmsk2        equ     $24             * misk. timer interrupt mask reg.2
       00000025 otflg2        equ     $25             * misc. timer interrupt flag
       00000026 opactl        equ     $26             * pulse accumulator control reg.
       00000027 opacnt        equ     $27             * pulse accumulator count reg.
       00000028 ospcr         equ     $28             * spi control reg.
       00000029 ospsr         equ     $29             * spi status reg.
       0000002a ospdr         equ     $2a             * spi data reg.
       0000002b obaud         equ     $2b             * sci baud rate control reg.
       0000002c osccr1        equ     $2c             * sci control reg. 1
       0000002d osccr2        equ     $2d             * sci control reg. 2
       0000002e oscsr         equ     $2e             * sci status reg.
       0000002f oscdr         equ     $2f             * sci data reg.
       00000030 oadctl        equ     $30             * a/d control/status reg.
       00000031 oadr1         equ     $31             * a/d result reg. 1
       00000032 oadr2         equ     $32             * a/d result reg. 2
       00000033 oadr3         equ     $33             * a/d result reg. 3
       00000034 oadr4         equ     $34             * a/d result reg. 4
       00000035 obprot        equ     $35             * block protect reg.
       00000039 ooption       equ     $39             * system configuration options
       0000003a ocoprst       equ     $3a             * arm/reset cop timer circuitry
       0000003b opprog        equ     $3b             * eeprom programming reg.
       0000003c ohprio        equ     $3c             * highest priority interrupt and misc.
       0000003d oinit         equ     $3d             * ram and i/o mapping reg.
       0000003e otest1        equ     $3e             * factory test reg.
       0000003f oconfig       equ     $3f             * configuration control reg.
file: e:\prak\zwerg11p\lib.a\bit.h

* Austausch der binären Operanden durch hexadezimale Operanden

                              list
file: midi.a

       000000c4 SCIVEC        equ     $00c4
       000000dc adr_iroc2     equ     $00c4+24        * vect=tbl+offset=c4+3*8=$dc

       0000008c PULAE         equ     140
       00007530 clkfrq        equ     30000
       0000000a offton        equ     10


*****************           *******************************************
*****************           *                                         *
*****************           *      RAM-Zellen für MIDI-IN             *
*****************           *                                         *
*****************           *******************************************

                              bss
      00000000                org     $0000
```

```
*****************           * Puffer für die 128 Töne

0000   00000001 okt01        ds.b   1
0001   00000001 okt02        ds.b   1
0002   00000001 okt03        ds.b   1
0003   00000001 okt04        ds.b   1

0004   00000001 okt05        ds.b   1
0005   00000001 okt06        ds.b   1
0006   00000001 okt07        ds.b   1
0007   00000001 okt08        ds.b   1

0008   00000001 okt09        ds.b   1
0009   00000001 okt10        ds.b   1
000a   00000001 okt11        ds.b   1
000b   00000001 okt12        ds.b   1

000c   00000001 okt13        ds.b   1
000d   00000001 okt14        ds.b   1
000e   00000001 okt15        ds.b   1
000f   00000001 okt16        ds.b   1

*****************           * Arbeitszellen für den Empfangspuffer

0010   00000001 poi_in       ds.w   1             * Empfangs-Pointer
0012   00000001 poi_out      ds.w   1             * Sende-Pointer
0014   00000001 anz_zei      ds.b   1             * Anzahl Zeichen im Puffer

*****************           * Arbeitszellen für die Kommandoauswertung

0015   00000001 kom_b        ds.b   1             * letztes Kommando
0016   00000001 ton_nr       ds.b   1             * letzte Tonnummer - Offset
0017   00000001 tmp          ds.b   1

*****************           * Empfangspuffer für MIDI-Sequenzen

0018   0000008c pu_anf       ds.b   PULAE         * Empfangspuffer
       000000a4 pu_end       equ    *

*****************           * Stack für das Programm

00a4   0000001b stk           ds.b   27            * Stack des PG
       000000bf stack         equ    *             * muß kleiner $00c4 sein!!!


*****************           ***********************************************
*****************           *                                             *
*****************           *     EEPROM Programmspeicher für MIDI-IN     *
*****************           *                                             *
*****************           ***********************************************

                              text
       0000b600               org    EEPROM

b600  8e00c3    start        lds    #$00c3        * stack! ($00c3)

       0000b603 INT           equ    *

b603  8620                   ldaa    #$20
file: midi.a

b605  b71009                 staa    ddrd          * ZWERG11A! *

b608  731007                 com    ddrc          * Port C alles Ausgang
b60b  bdb764                 jsr    clrall        * alle Töne aus
b60e  8680                   ldaa   #$80
b610  b71026                 staa   pactl         * PA7 ist Output
```

```
*****************           * Interrupt für OC2 und SCI initialisieren

b613  867e                    ldaa    #$7e
b615  97c4                    staa    <SCIVEC         * jmp code *
b617  18ceb79b                ldy     #sciint
b61b  18dfc5                  sty     <SCIVEC+1       * SCI int handler
*****************           LDAA    #$7E            * OC2 Interruptadresse eintragen
b61e  97dc                    STAA    <adr_iroc2      * OPC vom JMP laden!
b620  ceb7d1                  LDX     #frq_ir
b623  dfdd                    STX     <adr_iroc2+1    * ADR vom JMP laden!

*****************                   * SCI - Initialisieren

*****************           ldaa    #$30            * f/(13*1) = 9600 Baud
b625  8620                    ldaa    #$20            * f/( 4*1) =31250 Baud
b627  b7102b                  staa    baud
b62a  862c                    ldaa    #$2c            * Freigabe RxINT, Tx und Rx
b62c  b7102d                  staa    sccr2

b62f  ce1000    res           ldx     #REG            * Anfang Registerblock
b632  1c2240                  bset    $40,otmsk1,x    * OC2 Interrupt freigeben
b635  1c2040                  bset    $40,otctl1,x    * TOGGLE für OC2=PA6

*****************                   * Arbeitszellen initialisieren

b638  18ce0018                ldy     #pu_anf
b63c  18df10                  sty     <poi_in
b63f  18df12                  sty     <poi_out
b642  7f0014                  clr     anz_zei

b645  0e                      cli                                     * ints on


*****************           *********************************************
*****************           *                                           *
*****************           *     MIDI-Sequenzen auswerten              *
*****************           *                                           *
*****************           *********************************************

*****************           * MIDI-Wartezyklus    ************************

      0000b646 midi_zy        equ     *
b646  bdb73d                  jsr     read            * Kommando lesen

b649  8580                    bita    #$80
b64b  2618                    bne     newcom          * Es ist ein neues Kommando

*****************           * War das letzte Kommando TonON oder TonOFF?

b64d  d615                    ldab    <kom_b
b64f  c4e0                    andb    #$e0
b651  c180                    cmpb    #$80
b653  26f1                    bne     midi_zy         * es war kein ON oder OFF Kommando
b655  36                      psha
b656  9615                    ldaa    <kom_b
b658  840f                    anda    #$f             * nur Kanalnummer
b65a  f6100a                  ldab    porte           * DIL-Schalter einlesen
b65d  c407                    andb    #7              * nur Kanalauswahlschalter
b65f  11                      cba
b660  32                      pula
b661  26e3                    bne     midi_zy         * NEIN, anderer Kanal!!!
b663  203d                    bra     nextn           * kein Kommando -> nächste Note

*****************           * Ist es RESET?

b665  81ff      newcom        cmpa    #$FF
b667  2606                    bne     no_res

*****************           * RESET für MIDI-Empfänger

b669  0f                      sei
b66a  bdb764                  jsr     clrall
b66d  20c0                    bra     res
```

```
*****************           * Ist es Aktivtest?

b66f  81fe      no_res        cmpa   #$fe
b671  2605                    bne    no_aktt        * kein Aktivtest

b673  bdb75b                  jsr    sciput
b676  20ce                    bra    midi_zy

       0000b678 no_aktt       equ    *

*****************           * Ist es ein MIDI-CLK?

b678  81f8                    cmpa   #$f8
b67a  260b                    bne    nomclk
*
b67c  1d2240                  bclr   $40,otmsk1,x   * OC2 Interrupt sperren
b67f  1d0420                  bclr   $20,oportb,x   * CLK für 2. Reihe 74HCT374 ein
b682  1c0420                  bset   $20,oportb,x   * CLK für 2. Reihe 74HCT374 aus
b685  20bf                    bra    midi_zy

       0000b687 nomclk        equ    *

*****************           * Ist es der eingestellte Kanal?

b687  9715                    staa   <kom_b
b689  840f                    anda   #$f            * nur Kanalnummer
b68b  f6100a                  ldab   porte          * DIL-Schalter einlesen
b68e  c407                    andb   #7             * nur Kanalauswahlschalter
b690  11                      cba
b691  26b3                    bne    midi_zy        * NEIN, anderer Kanal!!!

*****************           * Es ist ein Kommando für diesen Kanal!!!

b693  9615                    ldaa   <kom_b
b695  84f0                    anda   #$f0
b697  81b0                    cmpa   #$b0
b699  2776                    beq    chann_mod      * Alle Töne aus? oder Pedal

b69b  121560a7                brset  $60,kom_b,midi_zy * Bits müssen 0 sein bei ON oder off

*****************           * Es muß Note ON oder OFF sein

b69f  bdb73d                  jsr    read           * Tonnummer lesen

*****************           * Tonnummer - Offset an den DIL-Schaltern

b6a2  800a      nextn         suba   #offton
b6a4  250b                    bcs    ton_l
b6a6  f6100a                  ldab   porte          * DIL-Schalter einlesen
b6a9  c4f8                    andb   #$f8           * nur Notenoffset
b6ab  54                      lsrb
b6ac  54                      lsrb
b6ad  54                      lsrb
b6ae  10                      sba
b6af  2403                    bcc    pedal          * gültiger Ton

b6b1  7eb725    ton_l         jmp    ton_low        * Ton ist zu klein

       0000b6b4 pedal                equ            *
*****************           Pedale werden als Ton 126 und 127 behandelt!
b6b4  9716                    staa   <ton_nr        * Tonnummer - Offset merken

*****************           * Bitmaske Bilden aus den unteren 3 Bit der Tonnummer

b6b6  8407                    anda   #7             * nur untere 3 Bit
b6b8  4c                      inca
b6b9  c6ff                    ldab   #$ff           * Maske in B und CY vorbereiten
b6bb  0c                      clc                   * CY=0 damit eine 0 reingeschoben wird
b6bc  59        b_msk         rolb
b6bd  4a                      deca
b6be  26fc                    bne    b_msk          * BIT-Maske bilden
```

```
*****************           * Ausgewältes Bit ist in B = 0, alle anderen sind 1

b6c0  37                    pshb                * BIT-Maske retten
b6c1  9616                  ldaa   <ton_nr

*****************           * Auswählen des 74HCT374 entsprechend der oberen 4 Bit

*****************           anda   #$78
b6c3  44                    lsra
b6c4  44                    lsra
b6c5  44                    lsra                * a = Bytenummer

*****************           * Alte Adresse am Port B ausblenden und neue einfügen

b6c6  f61004                ldab   portb
b6c9  c4f0                  andb   #$f0         * alte Adresse löschen
b6cb  d717                  stab   <tmp
b6cd  16                    tab                 * b := a = Bytenummer
b6ce  9a17                  oraa   <tmp         * neue Adresse dazu ODERn
b6d0  b71004                staa   portb        * Bytenummer ausgeben

b6d3  4f                    clra
b6d4  188f                  xgdy                * Bytenummer = Y = Datenadresse im RAM
b6d6  33                    pulb                * BIT-Maske zurückladen

*****************           * TON ein oder aus?

b6d7  9615                  ldaa   <kom_b
b6d9  84f0                  anda   #$f0         * Kanalnummer löschen
b6db  8180                  cmpa   #$80         * NoteOFF?
b6dd  270b                  beq    noteoff
b6df  8190                  cmpa   #$90         * NoteON?
b6e1  2712                  beq    noteon
b6e3  81b0                  cmpa   #$b0
b6e5  270e                  beq    noteon       * Pedal ON/OFF
b6e7  7eb646                jmp    midi_zy

*****************           * Es ist Kommando NoteOFF

       0000b6ea noteoff     equ    *

b6ea  183c                  pshy
b6ec  8d4f                  bsr    read         * 3. Byte bei TON-aus lesen
b6ee  1838                  puly

       0000b6f0 noteoff2    equ    *
b6f0  18e400                andb   okt01,y      * Note ausschalten vorbereiten
b6f3  200d                  bra    out

*****************           * Es ist Kommando NoteON

       0000b6f5 noteon      equ    *

b6f5  183c                  pshy
b6f7  8d44                  bsr    read         * 3. Byte bei TON-ein lesen
b6f9  1838                  puly

b6fb  4d                    tsta
b6fc  27f2                  beq    noteoff2     * 3. Byte ist 0!! --> NoteOFF

b6fe  53                    comb                * Maske negieren
b6ff  18ea00                orab   okt01,y      * Note einschalten vorbereiten

*****************           * NOTE im 74HCT374 ein oder aus

b702  f71003    out         stab   portc        * Daten für 74HCT374 ausgeben
b705  18e700                stab   okt01,y      * Daten im RAM merken

*****************           * Daten in den 74HCT374 stroben

b708  1d0410                bclr   $10,oportb,x * ~OE am 74HCT154 ein
b70b  1c0410                bset   $10,oportb,x * ~OE am 74HCT154 aus
b70e  7eb646                jmp    midi_zy
```

```
*****************           * Alle Töne ausschalten?

b711  8d2a      chann_mod     bsr    read
b713  8140                    cmpa   #64            * 064 rechtes Klavierpedal
b715  271d                    beq    klpre
b717  8143                    cmpa   #67            * 067 Soft-Pedal
b719  271e                    beq    softpe
b71b  817b                    cmpa   #123           * 123 All Notes OFF
b71d  270b                    beq    alloff
b71f  84fe                    anda   #$fe           * 120 All Sound Off und
b721  8178                    cmpa   #120           * 121 RESET all Controllers
b723  2705                    beq    alloff

       0000b725 ton_low       equ    *
b725  8d16                    bsr    read           * 3. Byte lesen
b727  7eb646    zyk           jmp    midi_zy

*****************           * JA alle aus

b72a  8d11      alloff        bsr    read           * 3. Byte lesen
b72c  4d                      tsta
b72d  26f8                    bne    zyk
b72f  bdb764                  jsr    clrall
b732  20f3                    bra    zyk

b734  8b3f      klpre         adda   #63            * 64+63=127 --> Rechtes Pedal
b736  7eb6b4    pedalw        jmp    pedal          * Weiter wie Note ON

b739  8b3b      softpe        adda   #59            * 67+59=126 --> Soft Pedal
b73b  20f9                    bra    pedalw

*****************           * MIDI-Wartezyklus Ende ********************************

*****************           * Unterprogramme

*****************           ********************************************
*****************           *                                          *
*****************           *      Zeichen aus Empangspuffer lesen     *
*****************           *                                          *
*****************           ********************************************

       0000b73d read          equ    *
b73d  9614                    ldaa   <anz_zei
b73f  4d                      tsta
b740  27fb                    beq    read

b742  7a0014                  dec    anz_zei
b745  18de12                  ldy    <poi_out
b748  1808                    iny
b74a  188c00a4                cpy    #pu_end
b74e  2604                    bne    no_endr
b750  18ce0018                ldy    #pu_anf
b754  18df12    no_endr       sty    <poi_out
b757  18a600                  ldaa   0,y            * Daten aus Puffer lesen
b75a  39                      rts

*****************           ********************************************
*****************           *                                          *
*****************           *              Ausgabe Seriell             *
*****************           *                                          *
*****************           ********************************************

b75b  7d102e    sciput        tst    scsr
b75e  2afb                    bpl    sciput
b760  b7102f                  staa   scdr
b763  39                      rts
```

```
*****************           *******************************************
*****************           *                                         *
*****************           *           Löschen aller Töne            *
*****************           *                                         *
*****************           *******************************************

b764  36        clrall        psha
b765  37                      pshb
b766  183c                    pshy
b768  7f1003                  clr     portc          * Alle Bit auf 0
b76b  8630                    ldaa    #$30
b76d  b71004                  staa    portb          * Port B setzen

*****************           * 47HCT374 löschen

b770  c610                    ldab    #16
*****************             ldx     #REG
b772  2003                    bra     loe374
b774  7c1004    loe374z       inc     portb
b777  1d0410    loe374        bclr    $10,oportb,x
b77a  1c0410                  bset    $10,oportb,x
b77d  1d0420                  bclr    $20,oportb,x
b780  1c0420                  bset    $20,oportb,x
b783  5a                      decb
b784  26ee                    bne     loe374z

*****************           * Arbeitszellen löschen

b786  18ce0010                ldy     #16
b78a  4f                      clra
b78b  1809      loeram        dey
b78d  18a700                  staa    okt01,y
b790  1808                    iny
b792  1809                    dey
b794  26f5                    bne     loeram
b796  1838                    puly
b798  33                      pulb
b799  32                      pula
b79a  39                      rts

*****************           *******************************************
*****************           *                                         *
*****************           *      Interuptbehandlung für Rx vom SCI  *
*****************           *                                         *
*****************           *******************************************

       0000b79b sciint        equ     *
b79b  f6102e                  ldab    scsr
b79e  c42e                    andb    #$2e           * RDRF, OR, NF, FE
b7a0  272e                    beq     no_zei         * kein Zeichen da, kein Fehler
b7a2  b6102f                  ldaa    scdr           * Daten lesen
b7a5  c506                    bitb    #6             * NF und FE
b7a7  2627                    bne     no_zei         * BREAK, oder Leitung gestört
b7a9  c508                    bitb    #8             * OR
b7ab  2703                    beq     no_rder
b7ad  1c0080                  bset    $80,oporta,x   * PA7 set wenn RD-Fehler

       0000b7b0 no_rder       equ     *

b7b0  18de10                  ldy     <poi_in
b7b3  1808                    iny
b7b5  188c00a4                cpy     #pu_end
b7b9  2604                    bne     no_end
b7bb  18ce0018                ldy     #pu_anf
b7bf  18df10    no_end        sty     <poi_in
b7c2  18a700                  staa    0,y
b7c5  189c12                  cpy     <poi_out       * Pufferüberlauf ??
b7c8  2603                    bne     no_err
b7ca  1c0020                  bset    $20,oporta,x   * Pufferüberlauf setzen (PA5)
b7cd  7c0014    no_err        inc     anz_zei        * Anzahl Zeichen im Puffer +1

b7d0  3b        no_zei        rti
```

```
*****************           *********************************************
*****************           *                                           *
*****************           *      IR - Interne MIDI-CLK-Frequenz       *
*****************           *                                           *
*****************           *********************************************


       0000b7d1 frq_ir        equ    *
*****************             LDX    #REG           * Anfang des Steuerregisterblocks
b7d1  1d0420                  bclr   $20,oportb,x   * CLK für 2. Reihe 74HCT374 ein
b7d4  1c0420                  bset   $20,oportb,x   * CLK für 2. Reihe 74HCT374 aus
b7d7  cc7530                  LDD    #clkfrq        * neuen Interrupt Zeitpunkt
b7da  e318                    ADDD   otoc2h,X       * bestimmen
b7dc  ed18                    STD    otoc2h,X       * ...
b7de  1c2340                  BSET   $40,otflg1,X   * Interruptflag wieder loeschen
b7e1  3b                      RTI                   * IR - Ende

*****************           * Ende MIDI
```