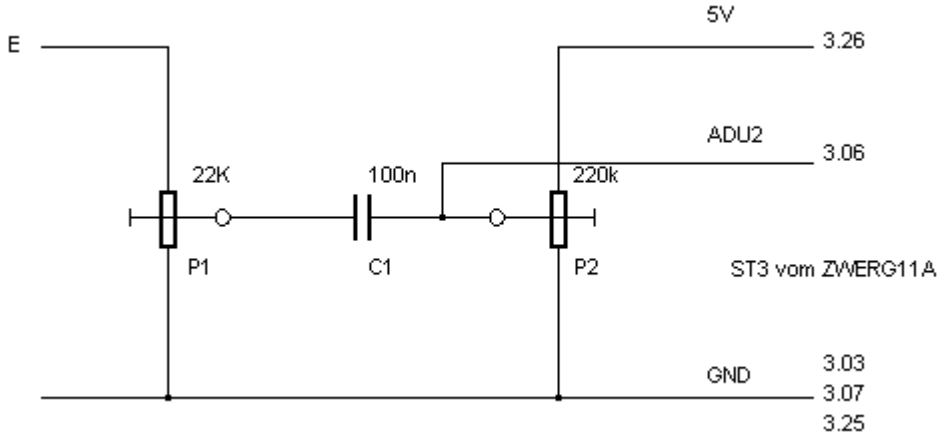
3.5. Speicheroszillograph für graphisches LCD TLX-711A

Zur Realisierung eines Speicheroszillographen mit der LCD TLX-711A wird ein A/D-Umsetzer des 68HC11 benötigt. Die 256 Werte des A/D-Umsetzers müssen durch 4 geteilt werden, da die LCD nur 64 Pixel in Y-Richtung besitzt. In X-Richtung hat die LCD 240 Pixel. Um auch ein Koordinatenkreuz zeichnen zu können, bleiben in X-Richtung 232 Pixel zur Darstellung der Meßwerte übrig. Da aber der 1. Meßwert immer auf der Y-Achse liegt, können 233 Werte angezeigt werden. Der A/D-Umsetzer benötigt zur Umsetzung 128 Takte. Die Zeit für einen Meßwert beträgt damit (128*0,5µs)/4=16µs, wenn der A/D-Umsetzer mit dem 2 MHz Systemtakt gesteuert wird. Da der A/D-Umsetzer im Start/Stop arbeitet kommen zu den 128 Takten noch die 7 Takte des Befehls BRCLR und die 4 Takte des Befehls STAB, die am Anfang des Unterprogramm **convert** stehen. Damit ergibt sich pro Pixel eine durchschnittliche Zeit von (139*0,5µs)/4=17,375µs. Die Frequenz, bei der eine Periode gerade noch angezeigt werden kann, berechnet sich dann zu 1/(233*17,375µs)=247Hz. Bei kleineren Frequenzen ist dann keine vollständige Darstellung der Periode möglich. Die größte darstellbare Frequenz muß noch mindestens 20 Pixel pro Periode haben, um den Verlauf noch erkennen zu können. Sie ergibt sich damit zu 1/(20*17,375µs)=2877,6Hz. Aus diesen kurzen Berechnungen ist ersichtlich, daß ein solcher Oszillograph nur für Frequenzen kleiner 2,8KHz sinnvoll eingesetzt werden kann. Da die Meßwerte digital gespeichert werden, ist es aber auch bei kleinsten Frequenzen möglich, den Verlauf flimmerfrei anzuzeigen.
Bei der Verwendung eines ZWERG11A der Firma MCT Paul & Scherer aus Berlin ist der Eingangsspannungsbereich der A/D-Umsetzer fest auf 0 ... 5V eigestellt, da V_{RL} mit Masse und V_{RH} mit der Betriebsspannung des 68HC11 verbunden sind. Für die Anzeige von Signalen mit positiven und negativen Halbwellen (wie bei Sinussignalen) ist am Eingang des A/D-Umsetzers noch folgende Beschaltung notwendig.



Mit dem Potentiometer P1 wird die Eingangsspannung auf einen Wertebereich von 0 ... 5V heruntergeteilt. Damit sind am Eingang E auch Spannungen größer 5 V von Spitze zu Spitze möglich. Durch den Kondensator C1 wird der Eingang des A/D-Umsetzers vom eigentlichen Meßobjekt für Gleichspannungen getrennt. Mit Hilfe des Potentiometers P2 kann dem Eingangssignal eine Gleichspannung überlagert werden, die für Signale mit betragsmäßig gleich großem negativen und positiven Spannungsmaximum auf 2,5V einzustellen ist. Das folgende Programm geht von einer solchen Eingangsbeschaltung aus und hat deshalb den Triggerpunkt für den Nulldurchgang eines Eingangssignales auf 2,5V am Eingang des A/D-Umsetzers gleich 128 (= 80 hexadezimal) festgelegt. Damit lassen sich dann Sinussignale auf dem LCD anzeigen. Als Takt für die A/D-Umsetzer wird der Systemtakt eingesetzt. An der X-Achse sind Marken für 1ms und 2ms und an der Y-Achse sind Marken für 1V und 2V angebracht. Diese Spannung am Schleifer von P1 ist aber nur dann gleich der Spannung am Eingang E, wenn sich der Schleifer von P1 am oberen Anschlag befindet. Das Programm löscht am Anfang den ASCII- und den Grafik-Speicher der LCD. Danach wird das Koordinatenkreuz gezeichnet und beschriftet. Anschließend wartet das Programm auf einen Nulldurchgang des Eingangssignals von einer negativem Spannung zu einer positiven Spannung. Mit dem 1. Meßwert >=0V nach negativen Werten beginnt die Abspeicherung der 233 Meßwerte im RAM des 68HC11. Die Programmschleife zur Abspeicherung muß dabei immer weniger als 128 Takte lang sein. Die größte länge von 124 Takten wird erreicht, wenn der Übergang aus der Schleife **poswave** zu **w1** erfolgt. Eine Taktung des A/D-Umsetzers mit

dem RC-Osszilator von 2,8828MHz ist deshalb auch nicht möglich, da 124*0,5µs größer ist als 128*0,347µs. Sind die 233 Meßwerte erfaßt, dann werden sie auf der LCD angezeigt. Danach geht das Programm in eine Warteschleife. Ist nach der Wartezeit noch ein Signal am Eingang des A/D-Umsetzers vorhanden, so werden die alten Meßwerte auf der LCD wieder gelöscht, und es wird nach dem Neuzeichnen des Koordinatenkreuzes mit einer neuen Messung begonnen. Diese Vorgänge werden vom Programm ständig wiederholt. Das Programm ist 509 Byte lang, es sind also nur noch 3 Byte frei. Das Quellprogramm ist so gestaltet, daß beim Aufruf der Datei MAKEOSZI.BAT als Parameter eine Ziffer von 1 bis 4 angegeben werden kann. Durch die Ziffer wird bei der Übersetzung einer der 4 A/D-Umsetzer von ADU1 bis ADU4 ausgewählt. In dem folgenden Quellprogramm sind bereits alle Ersetzungen für den ADU2 durch den Preprozessor ausgeführt wurden. Die LCD wird durch das Programm in einen Mode versetzt, in dem der Inhalt des ASCII-Bildwiederholspeichers (40*8 Zeichen ab Adresse $0000) und der Inhalt des Pixelspeichers (2560*6 Pixel ab Adresse $1000) mit einer EXOR-Funktion verknüpft werden und somit beide gleichzeitig als Überlagerung angezeigt werden. Dadurch ist es möglich, die X- und Y-Achse im ASCII-Mode zu beschriften.

```
* MICROCONTROLLER *

EEPROM      equ     $b600           eeprom area

REG         equ     $1000           register base

portc       equ     REG+$03         port c data reg.
portb       equ     REG+$04         port b data reg.
portcl      equ     REG+$05         port c latched data reg.
ddrc        equ     REG+$07         port c data direction reg.

adctl       equ     REG+$30         a/d control/status reg.
adr1        equ     REG+$31         a/d result reg. 1
adr2        equ     REG+$32         a/d result reg. 2
adr3        equ     REG+$33         a/d result reg. 3
adr4        equ     REG+$34         a/d result reg. 4

option      equ     REG+$39         system configuration options


oportc      equ     $03             port c data reg.
oportb      equ     $04             port b data reg.
oportcl     equ     $05             port c latched data reg.
oddrc       equ     $07             port c data direction reg.

oadctl      equ     $30             a/d control/status reg.
oadr1       equ     $31             a/d result reg. 1
oadr2       equ     $32             a/d result reg. 2
oadr3       equ     $33             a/d result reg. 3
oadr4       equ     $34             a/d result reg. 4

ooption     equ     $39             system configuration options
```

```
                bss

                ds.b    232+1           * Puffer für Meßwerte. 1.Pixel
*                                       * ist auf der Y-Achse

setreset        ds.b    1               * 1/0 - Bit setzen/rücksetzen
mem             ds.b    1               * Arbeitszelle für Pixel
ad1             ds.b    1               * Ergebnis 1. ADU-Zyklus
ad2             ds.b    1               * Ergebnis 2. ADU-Zyklus
ad3             ds.b    1               * Ergebnis 3. ADU-Zyklus
ad4             ds.b    1               * Ergebnis 4. ADU-Zyklus


maxcol          equ     40              * Anzahl Spalten
maxpos          equ     320             * Anz.von Cursorpos.
ramende         equ     setreset-1      * höchste nutzbare RAM-Adresse
widthcol        equ     6               * Spaltenbreite
xmax            equ     239
xmax1           equ     xmax-1
x00             equ     xmax-ramende
ymax            equ     63
y00             equ     31


trigger         equ     128             * Triggerschwelle 0 < t < 255

adust           equ     $00             * obere 4 ADU-Steuerbits für ADCTL


adu             equ     2-1

*       * Der ADU mit der Adresse (oadr2) wird verwendet *


                text
                org     EEPROM
start           lds     #$00ff          * Stack = $00ff

*       * ADU initialisieren *
                ldaa    #$80
                staa    option          * ADUs einschalten

*       * LCD initialisieren *
                jsr     lcd2ini         * Init.-Routine
                ldx     #REG
                jsr     cls             * TEXT-Bildschirm löschen
```

```
*       * Grafik-Bildschirm löschen *
                clr     setreset        * Pixel löschen
                ldab    #ymax+1         * Y - Position für Löschen
clr0            ldaa    #xmax+1         * X - Position für Löschen
clr1            psha
                pshb
                deca                    * X von 239 ... 0
                decb                    * Y von  63 ... 0
                jsr     pixel
                pulb
                pula
                deca
                bne     clr1
                decb
                bne     clr0

*       * X-Y Achsen zeichnen *
start1          ldaa    #1
                staa    <setreset       * Pixel setzen = Linien zeichnen

xachse          ldab    #x00            * B = Anfang X-Achse
xachse1         tba                     * A = B
                psha                    * X-Pos. merken
                ldab    #y00            * B = Y-Pos.
                jsr     pixel           * Pixel setzen
                pulb                    * B = X-Pos.
                incb                    * X-Pos. erhöhen
                cmpb    #xmax+1         * Test, ob Ende erreicht
                bne     xachse1         * wenn nein --> xachse1

yachse          ldab    #ymax+1         * B = Anfang Y-Achse
yachse1         pshb                    * Y-Pos. merken
                ldaa    #x00            * A = X-Pos.
                decb                    * Y von YMAX ... 0
                jsr     pixel           * Pixel setzen
                pulb                    * B = Y-Pos.
                decb                    * Y-Pos. erhöhen
                bne     yachse1         * wenn nein --> yachse1

*       * Y-Achse beschriften *
                ldaa    #xmax1-ramende  * X-Pos.
                ldab    #y00-13         * Y-Pos.        1,0V
                jsr     pixel           * Pixel setzen
                ldaa    #xmax1-ramende  * X-Pos.
                ldab    #y00-26         * Y-Pos.        2,0V
                jsr     pixel           * Pixel setzen

                ldd     #80             * 3.Zeile
                jsr     lcd2pos         * Cursor setzen
                ldaa    #"1"
                jsr     lcd2put         * Zahl schreiben
```

```
*       * X-Achse beschriften *
                ldaa    #x00+58         * X-Pos.        1,0ms
                ldab    #y00+1          * Y-Pos.
                jsr     pixel           * Pixel setzen
                ldaa    #x00+115        * X-Pos.        2,0ms
                ldab    #y00+1          * Y-Pos.
                jsr     pixel           * Pixel setzen

                ldd     #160+11         * 4.Zeile Pos. 11
                jsr     lcd2pos         * Cursor setzen
                ldaa    #"1"
                jsr     lcd2put         * Zahl schreiben


*       * Start 1. ADU-Zyklus *
                ldab    #adu+adust      * B = adu, nur oadr2 im Start Stop
                stab    oadctl,x        * ADCTL = B --> Start ADU

*       * allg. Initialisierung *
                ldy     #ramende        * IY = ramende, Zeiger

*       * auf negative Halbwelle warten *
negwave         bsr     convert         * auf Ende-Umsetzung warten
                cmpa    <ad4            * c <- T-M (Carryflag)
                blo     negwave         * bra, wenn c=1 (T <= M)

*       * auf positive Halbwelle warten *
poswave         bsr     convert         * auf CCF warten        46+6=   52\
                cmpa    <ad4            * c <- A-T (Carryflag)          3  |
                bhs     poswave         * bra, wenn c=0 (T > M)         3  |

*       * Meßwerte aufnehmen --> positive Halbwelle beginnt *
                cmpa    <ad1            *                               3  |
                blo     w1              * 1. Wert schon positiv         3  |
                cmpa    <ad2            *                               3  |_79
                blo     w2              * 2. Wert schon positiv         3  | Takte
                cmpa    <ad3            *                               3  |
                blo     w3              * 3. Wert schon positiv         3  |
                bra     w4              * nur 4. Wert positiv           3  /

value           bsr     convert         * auf CCF warten        46+6=   52\
w1              ldaa    <ad1            * A = Umsetzwert                3  |
                staa    0,y             * <IY> = A                      5  |
                dey                     * IY = IY-1, wenn z=1, fertig   4  |
                beq     evalue          * wenn IY = 0 -> evalue         3  |
w2              ldaa    <ad2            * A = Umsetzwert                3  |_112
                staa    0,y             * <IY> = A                      5  | Takte
                dey                     * IY = IY-1, wenn z=1, fertig   4  |
                beq     evalue          * wenn IY = 0 -> evalue         3  |
w3              ldaa    <ad3            * A = Umsetzwert                3  |
                staa    0,y             * <IY> = A                      5  |
                dey                     * IY = IY-1, wenn z=1, fertig   4  |
                beq     evalue          * wenn IY = 0 -> evalue         3  |
w4              ldaa    <ad4            * A = Umsetzwert                3  |
                staa    0,y             * <IY> = A                      5  |
                dey                     * IY = IY-1, wenn z=1, fertig   4  |
                bne     value           * wenn IY != 0 -> value         3  /
```

```
*       * Meßwerte anzeigen *
evalue          ldaa    #1              * A = 1, Bit setzen
                jsr     werte_zl

*       * etwa 3sec. anzeigen *
                ldy     #0
                ldaa    #15
wait            dey
                bne     wait
                deca
                bne     wait

                ldab    #adu+adust      * B = adu, nur oadr2 im Start Stop
*       * auf positive Halbwelle warten *
poswave2        bsr     convert         * auf Ende-Umsetzung warten
                adda    #10             * Trigger + 10
                cmpa    <ad4            * c <- T-M (Carryflag)
                bhs     poswave2        * nicht löschen, wenn kein Signal

*       * Meßwerte löschen *
                ldaa    #0              * A = 0, Bit rücksetzen
                jsr     werte_zl

                jmp     start1          * neu messen


*       * Unterprogramme *

*       * auf Ende-Umsetzung warten *
convert         brclr   $80,oadctl,x,convert  * wenn 0 -> convert          7  \
                stab    oadctl,x        * ADCTL = B --> Start ADU       4  |
                ldaa    oadr1,x         *                               4  |
                staa    <ad1            *                               3  |
                ldaa    oadr2,x         *                               4  |
                staa    <ad2            *                               3  |_46
                ldaa    oadr3,x         *                               4  |Takte
                staa    <ad3            *                               3  |
                ldaa    oadr4,x         *                               4  |
                staa    <ad4            *                               3  |
                ldaa    #trigger-1      * A = Schwellspannung           2  |
                rts                     * Rücksprung zum Aufruf         5  /
```

```
************** LCD2 module ***************************************************************
*                                                                                        *
* Date          Version        What                                        Who       *
* 25.10.91      1.00           create                                      he        *
* ---------------------------------------------------------------------------------------*
*                                - Notes -                                               *
*                                                                                        *
* Wie LCD1.A, aber für grafische LCDisplays.                                             *
*                                                                                        *
* ACHTUNG:                                                                               *
*  - lcd2dat/cmd haben zusätzlich den gewünschten Status als Parameter.                  *
* ---------------------------------------------------------------------------------------*
*                                - Usage -                                               *
*                                                                                        *
* include in application                                                                 *
* ---------------------------------------------------------------------------------------*
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                      *
*****************************************************************************************

****** init LCD (no register save)
*
*
lcd2ini         ldx     #lcditb                 * init table address        *
lcdi1           ldab    0,x                     * get # of parameters       *
lcdi2           inx                             * next...                   *
                ldaa    0,x                     * get parameter/cmd         *
                decb                            * parameter?                *
                bmi     lcdi3                   * no, write cmd             *
                pshb
                ldab    #3                      * requested status          *
                bsr     lcd2dat                 * write parameter           *
                pulb
                bra     lcdi2                   * more parameters...        *
lcdi3           ldab    #3                      * requested status          *
                bsr     lcd2cmd                 * write cmd                 *
                inx                             * next...                   *
                tst     0,x                     * end?                      *
                bpl     lcdi1                   * no, more cmds...          *
                rts

lcditb          dc.b    0,$81                   * mode = ROMzg TX xor Gr
                dc.b    2,$00,$10,$24           * Adr.-pointer =$1000
                dc.b    2,$00,$00,$21           * CU  -pointer =0
                dc.b    2,$00,$10,$42           * Graphic Home Addr. = $1000
                dc.b    2,$28,$00,$43           *   Column = 40 zu 6 Bit
                dc.b    2,$00,$00,$40           * Text Home Addr.= 0
                dc.b    2,$28,$00,$41           *   Column = 40
                dc.b    0,$9c                   * Text,Graphic = ON
                dc.b    -1
```

```
****** set write position
*       ACCD = address (see LCD data sheet)
*
lcd2pos         pshb
                psha
                tba
                ldab    #3              * requested status      *
                bsr     lcd2dat         * address LSB           *
                pula
                bsr     lcd2dat         * address MSB           *
                psha
                ldaa    #$24
                bsr     lcd2cmd         * address pointer set   *
                pula
                pulb
                rts

****** write cmd to LCD
*       ACCA = cmd
*       ACCB = requested status (see LCD data sheet)
*
lcd2cmd         pshb
                psha
                ldaa    #$21            * prepare cmd write     *
                bra     lcdd1

****** write data to LCD
*       ACCA = data
*       ACCB = requested status (see LCD data sheet)
*
lcd2dat         pshb
                psha
                ldaa    #$20            * prepare data write    *
lcdd1           clr     ddrc            * portc all inputs      *
                psha
lcdd2           ldaa    #$71
                staa    portb           * status                *
                ldaa    #$11
                staa    portb           * read enable           *
                tba
                anda    portc
                cba                     * requested status?     *
                bne     lcdd2           * no, try again...      *
                pulb
                stab    portb           * cmd/data write enable *
                ldaa    #$ff
                staa    ddrc            * portc all outputs     *
                pula
                staa    portc           * cmd/data out          *
                orab    #$70
                stab    portb           * latch cmd/data*
                pulb
                rts
```

```
****** put char in ACCA to lcd (address incremented)
*
lcd2put         psha
                pshb
                ldab    #3              * requested status      *
                suba    #" "            * blank is code 0       *
                bsr     lcd2dat
                ldaa    #$c0
                bsr     lcd2cmd         * data write addr incr  *
                pulb
                pula
                rts


**** CLS for LCD
*
cls             ldd     #0
                jsr     lcd2pos
                ldd     #maxpos
cls1            psha
                ldaa    #" "
                jsr     lcd2put
                pula
                subd    #1
                bne     cls1
                jsr     lcd2pos
                rts


*       * Meßwerte anzeigen/löschen *
werte_zl        staa    <setreset       * Zustand in setreset speichern
                ldy     #ramende        * IY = ramende
werte2          ldaa    0,y
                staa    <mem            * aktuellen Meßwert puffern
                ldaa    #ymax           *  A = 63
                lsr     mem             * Division mit 4 durch Rechtsversch.
                lsr     mem
                suba    <mem            * A = 63 - <IY>/4 --> Y-Position
                psha                    * Y-Pos. merken
                pshy
                pulb
                pulb                    * B = Low(IY) = X
                ldaa    #xmax
                sba                     * A = A - B = xmax-Low(IY) --> X-Pos.
                pulb                    * B = Y-Pos.
                pshy                    * Zeiger auf Meßwerte merken
                bsr     pixel           * Pixel setzen/rücksetzen
                puly
                dey                     * Zeiger auf nächsten Meßwert
                bne     werte2
                rts
```

```
*       * Pixel setzen/rücksetzen *             0----------------> 239=xmax
*       * A = X-Pos.   von 0 ... 239            |
*       * B = Y-Pos.   von 0 ... 63             |
*       * (setreset) = 1/0 setzen/rücksetzen    v 63=ymax

pixel           psha                            * X-Pos. merken

                ldaa    #maxcol                 * A = maxcol für Pixel
                mul                             * D = A*B = maxcol*ZeileY = Zeiger
                addd    #$1000                  * Anf.-Adr. Pixelram
                xgdy                            * IY = Zeiger auf Y-Pos.

                clra
                pulb                            * D = X-Pos.
                pshx
                ldx     #widthcol               * IX = Spaltenbreite
                idiv                            * IX = D/IX ;  D = Rest
                xgdx                            * IXlow --> B
                aby
                xgdx                            * B = Rest
                ldaa    #widthcol-1
                sba                             * Bits 0 ... 5

                psha                            * Bitposition merken

*       * Addreßpointer zu LCD senden *
                xgdy                            * D = Position
                jsr     lcd2pos

*       * Bit-Set Commando aufbauen und senden *
                pula                            * A = Bitpos.(untere 3 Bit )
                oraa    #$f8                    * Bit setzen annehmen ( 4.Bit )
                ldab    <setreset               * B = setreset
                andb    #1                      * test, ob Setzen gewollt
                bne     setbit                  * wenn ja, weiter bei setbit
                anda    #$f7                    * wenn nein, 4.Bit in A rücksetzen
setbit          ldab    #3                      * B = 3, Status LCD
                jmp     lcd2cmd                 * Bit setzen Commando
```