

Versuchslösungen für Praktikum Rechnertechnik

Steve Moser, Marcus Stegner

6. Januar 2003

Inhaltsverzeichnis

1	Einleitung und Versuch1	3
1.1	Allgemein	4
1.2	Aufgabe 1	4
1.2.1	Teilaufgabe 1	4
1.2.2	Teilaufgabe 2	8
1.2.3	Teilaufgabe 3	9
2	Versuch2	11
2.1	Aufgabe 2	12
3	Versuch3	18
3.1	Aufgabe 3	19
3.1.1	Teilaufgabe 1	19
3.1.2	Teilaufgabe 2	20
3.1.3	Teilaufgabe 3	21
4	Versuch4	24
4.1	Aufgabe 4	25
4.1.1	Teilaufgabe 1	25
4.1.2	Teilaufgabe 2	26
4.1.3	Teilaufgabe 3	28
5	Versuch5	31

5.1	Aufgabe 5	32
5.1.1	Teilaufgabe 2	32
5.1.2	Teilaufgabe 3	38
5.1.3	Teilaufgabe 4	38

Kapitel 1

Einleitung und Versuch1

1.1 Allgemein

Im Rahmen der Vorlesung Informatik der Technik 5.Semester Rechnertechnik, bei Herrn Prof.Högl, an der Fachhochschule Augsburg ist ein Vorlesungsbegleitendes Praktikum zu absolvieren. Dieses findet im zwei Wochenturnus ab Vorlesungsbeginn statt.

Die Entwicklungsumgebung ist eine Windows 2000 Plattform mit dem BD32 (einem Debugger) als Schnittstelle zum NF300. Hauptaufgabe ist die Programmierung des NF300 mit Motorola MC68332 CPU und Peripherie. Die Aufgaben sind bei Herrn Prof.Dr Bayer nach erfolgreicher Demonstration mit einem abschliessenden "Listing" abzugeben. Die Erfüllung des Praktikums ist Prüfungsvoraussetzung.

Die kompletten Lösungsansätze können auch unter www.anno74.de,link studium,link rt, eingesehen werden.

1.2 Aufgabe 1

Bei dem Ersten Program geht es darum die Entwicklungsumgebung kennen zu lernen:

1.2.1 Teilaufgabe 1

```
1
2 *****
3 *
4 * Project:      Blinking LED
5 * Filename:     blink.asm
6
7 *
8 *****
9
10
11 * Description:*****
12 *
13 * Blinking LED's on Port E
14 *
15 *****
16
17
18 * Addressmap with EQUates:*****
```

```

19  *
20  RAM      EQU      $0          ;RAM section
21  STACK   EQU      RAM+$800    ;Stack section
22  PGM     EQU      RAM+$C00    ;Program start
23  ROM     EQU      $800000     ;ROM section
24  *
25  ****
26
27
28  * IN/OUT-Parameters:****
29  *
30  * IN:   none
31  * OUT:  none
32  *
33  ****
34
35
36  * Definitions:****
37  *
38  *** 68332 global registerdefinitions ***
39  *
40  * SIM
41  SIMCR   EQU      $FFFA00     ;SIM Configuration Register
42  SYPCR   EQU      $FFFA20     ;System Protection Register
43  CSBARBT EQU      $FFFA48     ;Chip Select Base Address Boot Register
44  CSBAR0  EQU      $FFFA4C     ;Chip Select Base Address Register 0
45
46  *PORT E
47  PEPAR   EQU      $FFFA16     ;Pin Assignment Register
48  DDRE    EQU      $FFFA14     ;Data Direction Register
49  PORTE   EQU      $FFFA12     ;Port E is decoded on 2
50
51  *PORT F
52  PFPAR   EQU      $FFFA1E     ;Pin Assignment Register
53  DDRF    EQU      $FFFA1C     ;Data Direction Register
54  PORTF   EQU      $FFFA1A     ;Port F is decoded on 2
55
56
57  *
58  * Globals
59  SL      EQU      1024        ;Stack length

```

```

60 *
61 *****
62
63
64 *** Init Presets *****
65 *
66 *** Place FEPR0M at $800000, because unused
67 *
68         ORG         CSBARBT ;set on CSBARBT
69         DC.W        $8003   ;at 8M, size 64KB
70 *
71 *** Place ext. RAM at $0, 256KB, 0WS ***
72 *
73         ORG         CSBAR0           ;Set on SIM CSs
74         DC.W        $0005           ;CSBAR0, ext. RAM_RD
75         DC.W        $6830           ;CSOR0
76         DC.W        $0005           ;CSBAR1, ext. RAM_WR_LO
77         DC.W        $3030           ;CSOR1
78         DC.W        $0005           ;CSBAR2, ext. RAM_WR_HI
79         DC.W        $5030           ;CSOR2
80 *
81 *** Initialize SIM and system protection
82 * Switch off watchdogs, also while FREEZE is active
83 *
84         ORG         SIMCR
85         DC.W        $60CF           ;FREEZE settings
86         ORG         SYPCR
87         DC.W        $0000           ;no system protection
88 *
89 *** Define Stack ***
90 *
91         ORG         STACK           ;Stackarea starts at $800
92 ST      DS.B        SL              ;Length is 1KByte
93 *
94 *****
95
96
97 * Program: *****
98 *
99 *** Used processor-registers:
100 * D0: LED-on-register

```

```

101 * D1: LED-off-register
102 * D2: wait time
103 *
104 *** Start in SRAM ***
105         ORG         PGM                 ;Programcode at $C00
106 *
107 *** Initialize registers ***
108 *
109 * Initialize stack pointer
110 BLINK   MOVE.L    #ST+SL,A7             ;load Stack Pointer
111 * Initial Program Counter is initialized by the corresponding DO-F1
112 *
113 * Initialize Port E
114         MOVE.W    #$0000,PEPAR         ;I/O instead of systembus
115         MOVE.W    #$000F,DDRE         ;4 bit as output
116 *
117 * Initialize Port F
118         MOVE.W    #$0000,PFPAR         ;I/O instead of systembus
119         MOVE.W    #$0000,DDRF         ;all 8 bit as input
120 *
121 * Initialize CPU32 registers
122         MOVE.W    #$0000,D0             ;initialize with LED = on
123         MOVE.W    #$FFFF,D1             ;initialize with LED = off
124 *
125 *** Blink: main **
126 *
127 START   MOVE.W    D0,PORTE             ;LED on   >Testpunkt (SP,PC)
128         BSR      WAIT                  ;enjoy it
129         MOVE.W    D1,PORTE             ;LED off
130         BSR      WAIT                  ;enjoy it
131         BRA      START                 ;do forever
132 *
133 *** Subroutine ***
134 *
135 WAIT
136         MOVE.W    #$05,D3             ;Zaehler aussen -->Testpunkt (SP,PC, D3)
137 LOOP2   MOVE.W    #$8000,D2           ;Zaehler innen
138 LOOP1   DBRA     D2,LOOP1
139         DBRA     D3,LOOP2
140         RTS                      ;return      -->Testpunkt (D2,D3)
141

```

```

142  *
143  *** For the Assembler...  ***
144          END BLINK
145  *
146  ****
147

```

1.2.2 Teilaufgabe 2

Dieses Programm realisiert ein Lauflicht auf dem NF300. Aus Platzspargründen wird auf die Initialisierung bei den folgenden Programmen verzichtet. Ausser es ist erforderlich. Dies wird dann explizit angegeben.

```

1
2  *** Blink: main **
3  *
4  START
5  RECHTS
6          MOVE.W  PORTE,D4
7          AND.W   #$00F0,D4
8          BTST   #4,D4
9          BEQ    LINKS
10
11         MOVE.W  D0,PORTE          ;LED on          -->Testpunkt (SP,PC)
12         BSR    WAIT              ;enjoy it
13         MOVE.W  D1,PORTE          ;LED off
14         ROR    #1,D0
15         BSR    WAIT              ;enjoy it
16         BRA    RECHTS            ;do forever
17
18  LINKS
19         MOVE.W  PORTE,D4
20         AND.W   #$00F0,D4
21         BTST   #7,D4
22         BEQ    RECHTS
23
24         MOVE.W  D0,PORT   ;LED on          -->Testpunkt (SP,PC)
25         BSR    WAIT      ;enjoy it
26         MOVE.W  D1,PORTE ;LED off

```

```

27         ROL      #1,D0      ;rotate one bit to the left
28         BSR      WAIT      ;enjoy it
29         BRA      LINKS     ;do forever
30
31      *
32      *** Subroutine ***
33      *
34      WAIT
35             MOVE.W  #$05,D3   ;Zaehler aussen -->Testpunkt (SP,PC, D3)
36      LOOP2   MOVE.W  #$8000,D2 ;Zaehler innen
37      LOOP1   DBRA    D2,LOOP1
38             DBRA    D3,LOOP2
39             RTS                    ;return          -->Testpunkt (D2,D3)
40
41      *
42      *** For the Assembler... ***
43             END BLINK
44      *
45      ****

```

Das Hauptproblem bestand darin heraus zu finden, wie der Port E angesprochen wird und wie man einen Bitmanipulation vornimmt.

1.2.3 Teilaufgabe 3

Das folgende Programm sucht nach einem bestimmten Muster im Speicher. Wenn dieses gefunden wird, wird eine Zählervariable incrementiert und in einem unabhängigen Register abgelegt. So kann man im debugger mit dem Befehl window on das entsprechende Register überprüfen.

```

1
2      *** mem: main **
3      *
4      START
5             MOVE.l  #1000,A0  ;beginne ab der adresse #1000
6             MOVE.w  (A0),D1   ;schiebe inhalt der adresse an a0 in d1
7             move.w  #MS,D2    ;schiebe konstante MS nach D2
8             move.l  #$0,D7

```

```

9  VGL
10
11      CMP      D1,D2      ;vgl D1 mit konstanten
12      BEQ      ADDIEREN  ;wenn gefunden gehe zu ADDIEREN
13  NEXT      MOVE.w  (A0)+,D1 ;zähle speicherbereich 2 weiter
14      CMP.w   #$00FF,A0
15      BEQ      STOPPEN
16      BRA      VGL      ;gehe zu VGL
17
18  ADDIEREN
19      ADD.w   #1,D7
20      BRA      NEXT
21  STOPPEN
22
23      BRA      STOPPEN
24  *
25  *** For the Assembler... ***
26      END MEM
27  *
28  ****
29
30  MS      dc      1234
31

```

Kapitel 2

Versuch2

2.1 Aufgabe 2

Die Aufgabe aus mehreren Teilen. Zum einem soll das LCD Display des NF300 angesprochen werden, zum anderen soll ein String ausgegeben werden, der auf Knopfdruck blinkt. Weiterhin soll ein Integerwert ausgegeben werden, bei dem auf die Überprüfung des Vrozeichens nicht verzichtet wird. Wir haben als "Fleissaufgabe" die Tasten so belegt dass auf Knopfdruck 1 der String erscheint, auf Knopfdruck 2 der Integer erscheint und zusätzlich die Tasten 3 und 4, mit denen man entweder den String oder Integer Blinken lassen kann.

Probleme dabei waren:

Initialisierung des Displays

Die Blinkfunktion

Die Unterscheidung ob der Integer eine positive Zahl ist oder negativ.

Die permanente Abfrage der Tasten.

```
1  main
2      bsr      main_own
3  forever bra  forever
4
5
6  * Initialize the LCD display
7  lcd_init
8      move.b  #$38,LCD_IR      ; DL=1 (8-Bit), N=1, F=0
9      bsr      lcd_busy
10     move.b  #$0f,LCD_IR      ; 1 D C B (Display, Cursor, Blink)
11     bsr      lcd_busy
12     move.b  #$01,LCD_IR      ; Clear Display, Cursor Home
13     bsr      lcd_busy
14     move.b  #$06,LCD_IR      ; 1 I/D S
15     bsr      lcd_busy
16     rts
17
18 * Write a data character to the LCD display
19 lcd_char
20     move.b  d0,LCD_DR
21     bsr      lcd_busy
22     rts
23
24 * Wait until LCD display is ready for another read/write cycle
25 lcd_busy
```

```

26         btst     #7,LCD_IR           ; Bit 7 ist 1 falls LCD Busy
27         bne.b   lcd_busy           ; Z-Bit ist invertiertes Bit-7
28         rts
29
30     * Move cursor one position to the right
31     cu_right
32         move.b   #$14,LCD_IR
33         bsr     lcd_busy
34         rts
35
36
37     *Output of a given String-----
38     LCD_STRING
39         lea     STRING1,A0
40
41     go     move.b   (A0),d0   ;load content of A0 to D0 (Stringbegining)
42         cmp.b   #0,d0       ;compare if str end
43         beq     back        ;if yes return
44         bsr     lcd_char    ;else put to display
45         add     #1,A0        ;next char
46         bra     go          ;goto begining of subroutine
47     back  rts
48
49     *-----
50
51     *Output of an integer-----
52     LCD_INT
53         lea     INT,A0
54         move.l   (A0),D0
55         move.l   #0,d1       ;init d1 as counter
56
57     testbit8
58         btst    #15,D0      ;test bit 16, cause of word.lenght (int)
59         bne     ifminus    ;if zero bit is invertet,(minus(F)) goto
60         bra     divoutput  ;else
61     ifminus
62
63         move.l   d0,d4       ;save d0 (integer) to D4
64         move.b   #'-',D0     ;put a char to D0
65         bsr     lcd_char    ;put to display
66         move.l   d4,d0       ;move back integer to d0

```

```

67         neg.l   D0           ,negate complete register
68
69
70  divoutput
71         divu.w  #10,d0       ;divide d0 trough 10, modulo in d0 high
72         move.l  d0,d2       ;save in another register to modify
73         lsr.l   #8,d2       ;shifte 8 bit (16 not allowed)
74         lsr.l   #8,d2
75         add     #'0',d2     ;add 30 to make a char
76         move.b  d2,-(a7)    ;push to stack
77         add     #1,d1       ;add counter d1 one
78         and.l   #$0000FFFF,d0 ;delete d0 high (make high16bit zero)
79         cmp.l   #0,d0       ;if mod == 0, ready to go to display
80         bne    divoutput
81
82
83  output_number
84         move.b  (a7)+,d0    ;pop stack to displayregister
85         bsr    lcd_char    ;display char
86         sub     #1,d1       ;decrement counter
87         cmp.l   #0,d1       ;if zero go back to call routine
88         bne    output_number ;else branch once
89         rts
90
91
92
93  *-----
94  INT_BLINK
95         move.l  #50,d5     ;d5 as counterregister
96         bsr    LCD_INT     ;put integer to screen
97  loop1  bsr    SHUTDSP     ;display off
98         bsr    wait        ;wait bevor display on
99         bsr    SHOWDSP     ;Display on
100        bsr    wait
101        dbra   d5,loop1    ;loop
102
103        rts
104  *-----
105  STRING_BLINK
106        move.l  #10,d5
107        bsr    LCD_STRING

```

```

108
109 loop
110     bsr     SHUTDSP
111     bsr     wait
112     bsr     SHOWDSP
113     bsr     wait
114     dbra   d5,loop
115     rts
116
117
118
119 *-----
120 CLRDSP
121     move.b  #$01,LCD_IR      ; Clear Display, Cursor Home
122     bsr     lcd_busy
123     rts
124 *-----
125 *-----
126 SHUTDSP
127
128     move.b  #$08,LCD_IR
129     bsr     lcd_busy
130     rts
131 *-----
132 *-----
133 SHOWDSP
134     move.b  #$0C,LCD_IR
135     bsr     lcd_busy
136     rts
137 *-----
138 *proof pressed button and loop-----
139 PROOF_INPUT
140
141 anfang
142     MOVE.W  PORTE,D4
143     AND.W   #$00F0,D4
144 tast1
145     BTST   #4,D4
146     bne   taste2
147     bsr   CLRDSP
148     bsr   LCD_STRING

```

```

149         bra      anfang
150
151 taste2
152         BTST     #5,D4
153         bne     taste3
154         bsr     CLRDSP
155         bsr     LCD_INT
156         bra     anfang
157
158 taste3
159         BTST     #6,D4
160         bne     taste4
161         bsr     CLRDSP
162         bsr     INT_BLINK
163         bra     anfang
164
165 taste4
166         BTST     #7,D4
167         bne     anfang
168         bsr     CLRDSP
169         bsr     STRING_BLINK
170         bra     PROOF_INPUT
171         rts
172
173
174
175 *Own main
176 main_own
177         bsr     lcd_init           ;init display (clear, set cursor,..)
178         bsr     PROOF_INPUT
179         rts
180
181 * Wait loop
182
183 wait
184         move.W   #$9000,D3           ;Zaehler aussen -->Testpunkt
185         move.W   #$FFFF,D2         ;Zaehler innen
186 LOOP1   dbra    D2,LOOP1
187 LOOP2   dbra    D3,LOOP2
188         rts                       ;return           -->Testpunkt (D2,D3)
189

```

```
190
191
192 ever      dc.b      'H','a','l','l','o','l', 0
193 STRING1  dc.b      'G','u','t','e','n',' ','T','a','g', 0
194          even
195 INT      dc.l      -1234
196
197
198
199          end
```

Kapitel 3

Versuch3

3.1 Aufgabe 3

3.1.1 Teilaufgabe 1

Hier wurde gefordert das SRAM auf dem NF300 zu initialisieren, und dort das Suchprogramm nach einem Muster suchen zu lassen.

Probleme:

Das SRAM lässt sich nur einmal nach der Initialisierung beschreiben. Lösung: Vor der Initialisierung Strom kappen :-

Wir hatten zusätzlich das Problem dass unsere Suchvariable bis Abgabetermin immer auf einer geraden Speicheradresse lag und somit einwandfrei funktionierte. Bei Abgabe lag diese Variable auf einer ungeraden Adresse und somit stürzte das Programm laufend ab. Ein "even" hat dieses Problem gelöst.

```
1 *****
2 *init the sram
3
4 RAMBAR EQU $FFFB04 ;address of RAM Base Address Reg
5
6
7 *place rambar to adress 600000
8     org RAMBAR
9     dc.w $6000
10
11
12
13 *****
14
15
16
17 * Program: *****
18 *
19 *** Used processor-registers:
20 * D0: konstante
21 *
22 *** Start in SRAM ***
23     ORG PGM ;Programcode at $C00
24 *
25 *** Initialize registers ***
26 *
```

```

27 * Initialize stack pointer
28 BLINK  MOVE.L  #ST+SL,A7          ;load Stack Pointer
29 * Initial Program Counter is initialized by the corresponding DO-F
30 *
31 *** mem: main **
32 *
33 START
34 strt1  MOVE.l  #$600000,A0        ;beginne ab der adresse rambar
35        MOVE.w  #1234,(A0)
36        MOVE.w  (A0),D1           ;schiebe inhalt der adresse an a0 i
37        MOVE.l  #$0,D6            ;zählregister auf 0 setzen
38        MOVE.w  (A0),D2           ;inhalt A0 in d2
39
40 VGL
41
42        CMP     D1,D2              ;vgl D1 mit konstanten
43        BEQ     ADDIEREN           ;wenn gefunden gehe zu ADDIEREN
44 NEXT   add.w   #2,A0              ;zähle speicherbereich 2 weiter
45        MOVE.w  (A0),D1
46        CMP.l  #$600800,A0
47        BEQ     STOPPEN
48        BRA     VGL                ;gehe zu VGL
49
50 ADDIEREN
51        ADD.w   #1,D6
52        BRA     NEXT
53 STOPPEN
54
55        BRA     STOPPEN
56 *
57 *** For the Assembler... ***
58        END MEM
59 *
60 *****
61

```

3.1.2 Teilaufgabe 2

Das Bit 0 von Port E soll schnellstmöglichs seinen Zusatznd wechseln.

Das Bit 14 des Registers SYNCR regelt den Takt, voreingestellt ist der Teiler 2. Dieser

wird nun von uns ausgeschaltet und manuell manipuliert über den Debugger.

```
1  *** Blink: main **
2  *
3  START
4      move.w  d0,PORTE
5
6  toggle  eori.w  %#00000001,PORTE
7      BRA      toggle          ;do forever
8
9
10 *
11 *** For the Assembler... ***
12     END BLINK
13 *
14 *****
15
16 *to modify cpu cycle do following steps
17 *
18 *1.in debugger start prog
19 *2.mm $FFFA04 <enter>
20 *3.$7F08      doppelt
21 *4. . <to leave>
22 *$3F08 normaler takt
```

3.1.3 Teilaufgabe 3

Es soll ein periodischer Timer programmiert werden der in bestimmten Abständen einen Interrupt auslöst.

```
1
2  *PIT
3  PICR      EQU      $FFFA22          ;PIT Control Register
4  PITR      EQU      $FFFA24          ;PIT Timer Register
5
6
7  Belegen der Vektortabelle
8      MOVE.L  #ISR,$100
9
```

```

10 * Initialisiere PIT mit 20ms
11     move.w    #$0640,PICR      ;IPL = 6 / number = $40
12     move.w    #$00AF,PITR     ;prescaler=512,time=125ms
13
14 * Initialisiere Interruptmaske
15     andi.w    #%1111100011111111,SR    ;irq maske setzen
16     ori.w     #%0000010100000000,SR    ;
17
18
19 ***  main  **
20 *
21     MOVE.W    #$FFFE,PORTE
22 START
23     eori.w    #%00000001,PORTE
24 *     BSR     WAIT          ;enjoy it
25     BRA      START        ;do forever
26 *
27 *** Subroutine ***
28 *
29 WAIT
30     MOVE.W    #$05,D3        ;Zaehler aussen
31 LOOP2  MOVE.W    #$8000,D2   ;Zaehler innen
32 LOOP1  DBRA    D2,LOOP1
33     DBRA    D3,LOOP2
34     RTS          ;return
35
36
37 *****
38 *ISR
39
40 ISR    movem.l D0-D7,-(A7)    ;Retten der Datenregister
41     eori.w    #%000010,PORTE ;toggle output E(2)
42     addq.b   #1,D5          ;testzaehler
43     movem.l  (A7)+,D0-D7    ;Zurückschreiben der Register
44     rte
45
46 *****
47
48
49 *
50 *** For the Assembler... ***

```

```
51          END BLINK
52      *
53      *****
54
```

Kapitel 4

Versuch4

4.1 Aufgabe 4

4.1.1 Teilaufgabe 1

Es soll mit Hilfe des windowsprogramms "Hyperterminal" die Serielle Schnittstelle des NF300 angesprochen werden. Die erste Aufgabe ist es einen String über die Serielle Schnittstelle an den Hyperterminal zu senden und ihn dort erscheinen zu lassen.

```
1  ****main****
2
3
4  main
5
6
7  SCCR0    equ    $FFFC08 ;Adressbereiche der Register benennen SCBR=
8  SCCR1    equ    $FFFC0A ;wird benutzt um sci-bus zu konfigurieren
9  SCSR     equ    $FFFC0C ;sci-bus status register
10 SCDR     equ    $FFFC0E ;sci-bus data register
11
12         move.w  #$37,SCCR0 ;baudrate in sccr0 einstellen baudrate
13                               ;auf 9600 einstellen sieh TAB in Doku MO
14         move.w  #12,SCCR1  ;= 1100 --> enable receiver/transmitter
15                               ; (Parity Enable=0), (Mode Select=0) 8 b
16
17 load    movea.l #text,a0    ;Anfangsadresse nach a0
18
19 proof   btst.b  #0,SCSR     ;solange transmission data register =1 =
20         beq     proof
21         move.b  (a0),SCDR+1 ;information auf ausgang
22         cmp.b  #0,(a0)+    ;teste, ob string-ende schon erreicht
23         beq     forever
24         bra    proof
25
26
27 forever bra    forever
28
29     end main
30
31 text   dc.b    'Tach auch',0 ;wichtig dass textdeklarationen am
32
```

4.1.2 Teilaufgabe 2

Wie oben aber mit der Erweiterung dass sowohl Text, der auf der Tastatur eingegeben wird, auf dem LCD Display des NF300 erscheint, wie auch als Echo funktion dann auf dem Hyperterminal.

```

1  main
2      bsr      main_own
3  forever bra   forever
4
5
6  * Initialize the LCD display
7  lcd_init
8      move.b   #$38,LCD_IR      ; DL=1 (8-Bit), N=1, F=0
9      bsr      lcd_busy
10     move.b   #$0f,LCD_IR      ; 1 D C B (Display, Cursor, Blink)
11     bsr      lcd_busy
12     move.b   #$01,LCD_IR      ; Clear Display, Cursor Home
13     bsr      lcd_busy
14     move.b   #$06,LCD_IR      ; 1 I/D S
15     bsr      lcd_busy
16     rts
17
18 * Write a data character to the LCD display
19 lcd_char
20     move.b   d0,LCD_DR
21     bsr      lcd_busy
22     rts
23
24 * Wait until LCD display is ready for another read/write cycle
25 lcd_busy
26     btst     #7,LCD_IR        ; Bit 7 ist 1 falls LCD Busy
27     bne.b   lcd_busy         ; Z-Bit ist invertiertes Bit-7
28     rts
29
30
31 *Own main

```

```

32  main_own
33
34  QSMCR    equ    $FFFC00    ;QSM configuration register (um in den sw
35                                     switchen)
36  SCSR     equ    $FFFC0C    ;sci-bus status register (data there ?)
37  SCDR     equ    $FFFC0E    ;sci-bus data register (write and catch o
38  SCCR0    equ    $FFFC08    ;baud-rate sci-control-register 0
39  SCCR1    equ    $FFFC0A    ;sci-control-register 1 for configuring
40                                     parity/stop/loop/wake usw.
41
42          move.w  #0,d0
43          bsr     lcd_init
44
45          move.w  #$37,SCCR0    ;baudrate in sccr0 einstellen
46          move.w  #12,SCCR1    ;no parity, 8 bit im sccr1
47          move.w  #$0000,QSMCR  ;busoperation ( supervisor-mode)
48                                     ,eigentlich bit 0 niederer byte sup
49
50
51  loop     btst.b  #6,SCSR+1    ;prüfe auf 0 ab, wenn ja (keine neuen da
52                                     ;TDRE transmission data received empty k
53          beq     loop         ;wenn ja (neue daten vorhanden),gehe wei
54          move.b  SCDR+1,d0    ;schreibe in register das erhaltene zeic
55
56          bsr     lcd_char    ;gib das zeichen am lcd aus
57
58
59  test     btst.b  #0,SCSR     ;transmission data register flag 0==date
60                                     werden noch gesendet
61          beq     test         ;warte
62          move.b  d0,SCDR+1    ;switch auf senden damit echo auf hypert
63          bra     loop         ;zurück und auf neue daten warten
64
65
66
67  end
68

```

4.1.3 Teilaufgabe 3

Vom PC werden mittels HyperTerminal das Laufflicht aus dem 1. Versuch gesteuert. Mit den Zeichen "+" (schneller), "-" (langsamer), "l" (links) und "r"(rechts) wird die Richtung und die Geschwindigkeit des Laufflichts geändert. Dazu liest eine Interruptroutine die Kommandos ein und gibt einen entsprechenden Hinweis auf dem LCD-Display aus, während das Hauptprogramm zur Ansteuerung der LEDs permanent weiterläuft. Der Interruptvektor und der Interruptlevel wird mit dem Register QILR/QIVR eingestellt.

```
1  *** Seriell Port Init ***
2
3  MOVE.W  #$0640,QILR      ;Interruptlevel=6 wird gesetzt ->
4                          loest interrupt
5                          mit nr. aus:(Y0110),
6                          Interruptvektor=$40 (vectornumber 01000000)
7  move.w  #$84,QSMCR      ;set supervisor mode & interrupt id;
8                          supervisormode wird gesetzt
9  MOVE.W  #$37,SCCR0      ;Baudrate 9600
10 MOVE.W  #$2C,SCCR1      ;Transmitter, Receiver und Receiver
11                          Interrupt Enable
12                          aktiviert;ab hier wird obiger
13                          interruptlevel
14                          fuer recieve ausgelöst
15
16
17  * Interrupt - Einstellungen
18
19  MOVE.L  #Itp,VT+($40*4) ;schreibe adresse der interrupt
20                          routine auf
21                          adresse 100 der vector tabelle
22                          ;ab adresse 100 steht die adresse
23                          der interruptroutine
24  ANDI.W  #%111110001111111,SR ;$F8FF Interruptmaske auf
25                          Prioritaet 5 setzen
26                          (alle darüber werden durchgelassen
27                          ,in unserem falle 6)
28  ORI.W   #%0000010100000000,SR ;auf 5 setzen (oben wird gelöscht)
29
30  ***  main  **
31  *
32
```

```

33         move.w  #$FE,d0
34         move.w  #$FF,d1
35
36         move.w  #0,d4
37
38  START
39         MOVE.W  D0,PORTE        ;LED on
40         BSR     WAIT           ;enjoy it
41         MOVE.W  D1,PORTE        ;LED off
42
43         CMP.W   #1,D5           ;D5 gibt die richtung an 1 fuer
44                                 rechts
45                                 0 fuer links
46         BEQ    roright        ;goto rechts
47
48         CMP.W   #0,D5
49         BEQ    roleft
50
51  roleft  ROL.B   #1,D0           ;left
52         BRA    START
53
54  roright ROR.B   #1,D0           ;right
55         BRA    START
56
57  WAIT
58         move.w  d2,-(a7)
59         MOVE.w  D4,D3
60  _w01    MOVE.w  #$F000,D2       ;preset time
61  _w02    DBRA   D2,_w02         ;exceeded?
62         DBRA   D3,_w01
63         move.w  (a7)+,d2
64         rts
65
66  *****
67  *Itp
68  Itp     btst.b  #6,SCSR        ;das "read bit" wird zurückgesetzt,
69                                 ;damit der interrupt wieder ausgelöst
70                                 werden kann
71         move.l  D0,-(A7)       ;Retten der Datenregister
72
73         move.b  SCDR+1,d0      ;hole zeichen das den interrupt

```

```

74                                     ausgeloesst hat,
75                                     um den vergleich zu machen
76         cmp.b    #'+' ,d0
77         beq     schneller
78         cmp.b    #'-' ,d0
79         beq     langsamer
80         cmp.b    #'l' ,d0
81         beq     links
82         cmp.b    #'r' ,d0
83         bra     rechts
84         bra     Itp_ende
85
86 links
87         move.w   #0,d5           ;d5 ist vergleichsregister,
88                                     für links oder rechts (s.o.)
89         bra     Itp_ende
90
91 rechts
92         move.w   #1,d5
93         bra     Itp_ende
94
95 schneller
96         cmp     #3,d4           ;dient zum schleifen verkuerzen
97                                     falls schon 3 dann geht nicht schneller
98                                     das lauflicht wäre nicht mehr sichtbar
99                                     (getestet)
100        bcs     Itp_ende
101        sub.w   #3,d4           ;ansonsten verkürze die aeussere
102                                     schleife um 3
103                                     ; dann wird die wait schleife verkuerzt
104        bra     Itp_ende
105
106 langsamer
107        add.w   #3,d4           ;die wait schleife wird nun verlängert,die
108                                     aeussere schleife um 3 vergroessert
109        bra     Itp_ende
110
111 Itp_ende
112        move.l   (A7)+,D0       ;Zurückschreiben der Register
113        rte
114

```

Kapitel 5

Versuch5

5.1 Aufgabe 5

5.1.1 Teilaufgabe 2

Schreiben Sie Funktionen zum Setzen und Lesen der aktuellen Zeit (ohne Interrupts!). Die aktuelle Zeit soll auf dem LCD-Display dargestellt werden. Die aktuelle Zeit beim Setzen des RTC kann in Ihrem Programmtext enthalten sein.

```
1  **
2  * Main program
3  main
4
5  * Initialize stack pointer
6      move.l  #sseg+STACKSZ,a7 ;load stack pointer
7
8  * Initial Program Counter is initialized by the corresponding
9  * DO-File
10
11 * Initialize Port E
12     move.w  #$0000,PEPAR      ;I/O instead of systembus
13     move.w  #$000F,DDRE      ;4 bit as output
14
15 * Initialize Port F
16     move.w  #$0000,PFPAR      ;I/O instead of systembus
17     move.w  #$0000,DDRF      ;all 8 bit as input
18
19 * CS5 (LCD) and CS3 (RTC) enable
20     or.w    #$2200,CSPAR0
21
22 * Init LCD Chipselect
23     move.w  #$2000,CSBAR5     ;Basisadresse LCD, 2K Block
24     move.w  #$7D30,CSOR5     ;CS Options
25
26 * QSM Init
27     move.w  #$0000,QSMCR
28
29     bsr    main_demo_rtc
30     bsr    forever
31
32 *** End of main program
```

```

33
34 **
35 * Init QSPI. CLK Speed is given by system_clk / (2 * SPBR)
36 * LOOPQ : Bit 10 in SPCR3
37 qspi_init
38     move.w  %#0000001100000011,SPCR1  ;disable QSPI pins
39     move.b  #$00,PORTQS      ;PCS3=0 and all else 0
40     move.b  #$7e,DDRQS      ;1=out, 0=in
41     move.b  #$7f,PQSPAR     ;SPI Outputs
42
43     move.w  %#1000000111111111,SPCR0  ;Master, 16-bits,
44                                           CPHA=1, SPBR
45     move.w  %#0000000000000000,SPCR2  ;No Interrupts, ENDQP=0
46     move.w  %#0000000000000000,SPCR3
47     rts
48
49 **
50 * Write a byte to an 8-bit address
51 * d0 -- data
52 * d1 -- address
53 qspi_wr_byte
54     move.b  d1,SPTR
55     move.b  d0,SPTR+1
56     move.b  #$49,SPCR      ;Control RAM: BITSE, PCS3
57     bsr     _qspi_enable   ;start transmission
58     bsr     _qspi_wuf     ;wait until tx finished
59     rts
60
61 **
62 * Dummy write to the SPI bus. Use this function
63   as a delay between
64 * multiple write accesses to the RTC device.
65   It sends 8 zero bits on the
66 * SPI bus without activating any chip select signals.
67 qspi_wr_delay
68     move.b  #0,SPTR
69     move.b  #0,SPTR+1
70     move.b  #$00,SPCR      ;Control RAM: 8 bit, No CS
71     bsr     _qspi_enable   ;start transmission
72     bsr     _qspi_wuf     ;wait until tx finished
73     rts

```

```

74
75 **
76 * Read a byte from a given address and return it in d0.
77 * d1 -- address
78 qspi_rd_byte
79     move.b  d1,SPTR
80     move.b  #0,SPTR+1
81     move.b  #$c9,SPCR          ;Control RAM: BITSE, PCS3
82     bsr     _qspi_enable      ;start transmission
83     bsr     _qspi_wuf         ;wait until tx finished
84     move.w  SPRR,d2
85     rts
86
87 **
88 * Start QSPI transmission
89 _qspi_enable
90     ori.w   #$8000,SPCR1      ;Enable QSPI
91     rts
92
93 **
94 * Wait until finished (Wait until the SPIF bit becomes one)
95 _qspi_wuf
96     move.b  SPSR,d0
97     andi.b  #$80,d0
98     cmp.b   #$80,d0
99     bne     _qspi_wuf
100    bclr   #7,SPSR            ;Clear SPIF bit
101    rts
102
103 **
104 * Enter infinite loop
105 forever
106 _fe01  bra     _fe01
107        rts
108
109 **** main_RTC ****
110
111 main_demo_rtc
112     bsr     qspi_init
113
114 * Initialize the RTC by writing to it's status register

```

```

115  _md01
116      move.b  #$8f,d1          ;write address status register
117      move.b  #$00,d0          ;data
118      bsr     qspi_wr_byte
119      bsr     qspi_wr_delay    ;wait
120
121  * write time and date to RTC
122  _write
123      move.b  #$80,d1          ;write address seconds
124      move.b  #$00,d0          ;data (00 sec)
125      bsr     qspi_wr_byte
126      bsr     qspi_wr_delay    ;wait
127
128      move.b  #$81,d1          ;write address minutes
129      move.b  #$00,d0          ;data (00 min)
130      bsr     qspi_wr_byte
131      bsr     qspi_wr_delay    ;wait
132
133      move.b  #$82,d1          ;write address hours
134      move.b  #$15,d0          ;data (15 h)
135      bsr     qspi_wr_byte
136      bsr     qspi_wr_delay    ;wait
137
138      move.b  #$83,d1          ;write address day
139      move.b  #$04,d0          ;data (Th)
140      bsr     qspi_wr_byte
141      bsr     qspi_wr_delay    ;wait
142
143      move.b  #$84,d1          ;write address date
144      move.b  #$07,d0          ;data (7)
145      bsr     qspi_wr_byte
146      bsr     qspi_wr_delay    ;wait
147
148      move.b  #$85,d1          ;write address month
149      move.b  #$11,d0          ;data (11)
150      bsr     qspi_wr_byte
151      bsr     qspi_wr_delay    ;wait
152
153      move.b  #$86,d1          ;write address year
154      move.b  #$02,d0          ;data (02)
155      bsr     qspi_wr_byte

```

```

156         bsr      qspi_wr_delay    ;wait
157
158 * Read from the RTC registers in an endless loop
159 _w02
160         bsr      lcd_init
161
162         move.b   #$06,d1    ;read address year
163         bsr      qspi_rd_byte
164         bsr      _calc     ;create ASCII-code and write to LCD display
165
166         move.b   #$05,d1    ;read address month
167         bsr      qspi_rd_byte
168         bsr      _calc     ;create ASCII-code and write to LCD display
169
170         move.b   #$04,d1    ;read address date
171         bsr      qspi_rd_byte
172         bsr      _calc     ;create ASCII-code and write to LCD display
173
174         move.b   #'T',d5
175         bsr      lcd_char          ;write 'T' to LCD display
176
177         move.b   #$02,d1    ;read address hour
178         bsr      qspi_rd_byte
179         bsr      _calc     ;create ASCII-code and write to LCD display
180
181         move.b   #':',d5
182         bsr      lcd_char          ;write ':' to LCD display
183
184         move.b   #$01,d1    ;read address minutes
185         bsr      qspi_rd_byte
186         bsr      _calc     ;create ASCII-code and write to LCD display
187
188         move.b   #$00,d1    ;read address seconds
189         bsr      qspi_rd_byte
190         bsr      _calc     ;create ASCII-code and write to LCD display
191
192         bsr      wait      ;waiting
193         bsr      wait      ;waiting
194         bsr      wait      ;waiting
195
196         bra      _w02

```

```

197
198         rts
199
200 * calculate and create ASCII-Code and write to LCD display
201 _calc
202         move.b  d2,d5    ;store D2 in D5
203         and.b   #$F0,d5
204         lsr.b   #$4,d5  ;rotate upper nibble to lower nibble
205         add.b   #$30,d5 ;create ASCII-code
206         bsr    lcd_char ;write data to LCD display
207
208         move.b  d2,d5    ;store D2 in D5
209         and.b   #$0F,d5
210         add.b   #$30,d5 ;create ASCII-code
211         bsr    lcd_char ;write data to LCD display
212
213         rts
214
215 * Initialize the LCD display
216 lcd_init
217         move.b  #$38,LCD_IR ; DL=1 (8-Bit), N=1, F=0
218         bsr    lcd_busy
219         move.b  #$0f,LCD_IR ; 1 D C B (Display, Cursor, Blink)
220         bsr    lcd_busy
221         move.b  #$01,LCD_IR ; Clear Display, Cursor Home
222         bsr    lcd_busy
223         move.b  #$06,LCD_IR ; 1 I/D S
224         bsr    lcd_busy
225         rts
226
227 * Write a data character to the LCD display
228 lcd_char
229         move.b  D5,LCD_DR
230         bsr    lcd_busy
231         rts
232
233 * Wait until LCD display is ready for another read/write cycle
234 lcd_busy
235         btst   #7,LCD_IR ; Bit 7 ist 1 falls LCD Busy
236         bne.b  lcd_busy ; Z-Bit ist invertiertes Bit-7
237         rts

```

```

238
239 * Move cursor one position to the right
240 cu_right
241     move.b  #$14,LCD_IR
242     bsr    lcd_busy
243     rts
244
245
246 * Delay loop *
247 wait
248     move.w  d7,-(a7)
249     move.w  #$FFFF,d7          ; number of wait loops
250 _w01    dbra  d7,_w01
251     move.w  (a7)+,d7
252     rts
253
254     end

```

5.1.2 Teilaufgabe 3

Wurde uns gütigerweise erlassen.

5.1.3 Teilaufgabe 4

Der RTC beinhaltet 96 byte RAM, in dem man beliebige Daten unterbringen kann. Dieser Speicherbereich wird, wie die gesamte Uhr, über eine Batterie versorgt, wenn die Stromversorgung des Rechners abgesteckt wird. Schreiben Sie zwei Funktionen, die einen beliebigen String im RTC-RAM ablegen und anschliessend wieder in das normale RAM auslesen können. Ist Ihr String tatsächlich nach einem kurzen Stromausfall wieder vorhanden?

```

1
2  **
3  * Main program
4  main
5
6  * Initialize stack pointer
7      move.l  #sseg+STACKSZ,a7 ;load stack pointer

```

```

8
9 * Initial Program Counter is initialized by the corresponding
10 * DO-File
11
12 * Initialize Port E
13     move.w  #$0000,PEPAR    ;I/O instead of systembus
14     move.w  #$000F,DDRE    ;4 bit as output
15
16 * Initialize Port F
17     move.w  #$0000,PFPAR    ;I/O instead of systembus
18     move.w  #$0000,DDRF    ;all 8 bit as input
19
20 * CS5 (LCD) and CS3 (RTC) enable
21     or.w    #$2200,CSPAR0
22
23 * Init LCD Chipselect
24     move.w  #$2000,CSBAR5   ;Basisadresse LCD, 2K Block
25     move.w  #$7D30,CSOR5   ;CS Options
26
27 * QSM Init
28     move.w  #$0000,QSMCR
29
30     bsr     main_demo_rtc
31     bsr     forever
32
33 *** End of main program
34
35 **
36 * Init QSPI. CLK Speed is given by system_clk / (2 * SPBR)
37 * LOOPQ : Bit 10 in SPCR3
38 qspi_init
39     move.w  %#0000001100000011,SPCR1 ;disable QSPI pins
40     move.b  #$00,PORTQS    ;PCS3=0 and all else 0
41     move.b  #$7e,DDRQS    ;1=out, 0=in
42     move.b  #$7f,PQSPAR   ;SPI Outputs
43
44     move.w  %#1000000111111111,SPCR0 ;Master, 16-bits,
45                                     CPHA=1, SPBR
46     move.w  %#0000000000000000,SPCR2 ;No Interrupts, ENDQP=0
47     move.w  %#0000000000000000,SPCR3
48     rts

```

```

49
50 **
51 * Write a byte to an 8-bit address
52 * d0 -- data
53 * d1 -- address
54 qspi_wr_byte
55     move.b  d1,SPTR
56     move.b  d0,SPTR+1
57     move.b  #$49,SPCR      ;Control RAM: BITSE, PCS3
58     bsr     _qspi_enable   ;start transmission
59     bsr     _qspi_wuf     ;wait until tx finished
60     rts
61
62 **
63 * Dummy write to the SPI bus. Use this function
64   as a delay between
65   * multiple write accesses to the RTC device.
66   It sends 8 zero bits on the
67   * SPI bus without activating any chip select signals.
68 qspi_wr_delay
69     move.b  #0,SPTR
70     move.b  #0,SPTR+1
71     move.b  #$00,SPCR     ;Control RAM: 8 bit, No CS
72     bsr     _qspi_enable   ;start transmission
73     bsr     _qspi_wuf     ;wait until tx finished
74     rts
75
76 **
77 * Read a byte from a given address and return it in d0.
78 * d1 -- address
79 qspi_rd_byte
80     move.b  d1,SPTR
81     move.b  #0,SPTR+1
82     move.b  #$c9,SPCR     ;Control RAM: BITSE, PCS3
83     bsr     _qspi_enable   ;start transmission
84     bsr     _qspi_wuf     ;wait until tx finished
85     move.w  SPRR,d2
86     rts
87
88 **
89 * Start QSPI transmission

```

```

90  _qspi_enable
91      ori.w    #$8000,SPCR1    ;Enable QSPI
92      rts
93
94  **
95  * Wait until finished (Wait until the SPIF bit becomes one)
96  _qspi_wuf
97      move.b   SPSR,d0
98      andi.b   #$80,d0
99      cmp.b    #$80,d0
100     bne      _qspi_wuf
101     bclr    #7,SPSR    ;Clear SPIF bit
102     rts
103
104  **
105  * Enter infinite loop
106  forever
107  _fe01  bra      _fe01
108      rts
109
110  **** main_RTC ****
111
112  main_demo_rtc
113      bsr      qspi_init
114
115  * Initialize the RTC by writing to it's status register
116  _md01
117      move.b   #$8f,d1          ;write address status register
118      move.b   #$00,d0          ;data
119      bsr      qspi_wr_byte
120      bsr      qspi_wr_delay    ;wait
121
122  * write time and date to RTC
123  _write
124      move.b   #$80,d1          ;write address seconds
125      move.b   #$00,d0          ;data (00 sec)
126      bsr      qspi_wr_byte
127      bsr      qspi_wr_delay    ;wait
128
129      move.b   #$81,d1          ;write address minutes
130      move.b   #$00,d0          ;data (00 min)

```

```

131         bsr      qspi_wr_byte
132         bsr      qspi_wr_delay    ;wait
133
134         move.b   #$82,d1          ;write address hours
135         move.b   #$15,d0          ;data (15 h)
136         bsr      qspi_wr_byte
137         bsr      qspi_wr_delay    ;wait
138
139         move.b   #$83,d1          ;write address day
140         move.b   #$04,d0          ;data (Th)
141         bsr      qspi_wr_byte
142         bsr      qspi_wr_delay    ;wait
143
144         move.b   #$84,d1          ;write address date
145         move.b   #$07,d0          ;data (7)
146         bsr      qspi_wr_byte
147         bsr      qspi_wr_delay    ;wait
148
149         move.b   #$85,d1          ;write address month
150         move.b   #$11,d0          ;data (11)
151         bsr      qspi_wr_byte
152         bsr      qspi_wr_delay    ;wait
153
154         move.b   #$86,d1          ;write address year
155         move.b   #$02,d0          ;data (02)
156         bsr      qspi_wr_byte
157         bsr      qspi_wr_delay    ;wait
158
159         * Read from the RTC registers in an endless loop
160         _w02
161         bsr      lcd_init
162
163         move.b   #$06,d1          ;read address year
164         bsr      qspi_rd_byte
165         bsr      _calc              ;create ASCII-code and write to LCD
166
167         move.b   #'-',d5
168         bsr      lcd_char          ;write '-' to LCD display
169
170         move.b   #$05,d1          ;read address month
171         bsr      qspi_rd_byte

```

```

172     bsr     _calc           ;create ASCII-code and write to LCD
173
174     move.b #'-',d5
175     bsr     lcd_char       ;write '-' to LCD display
176
177     move.b  #$04,d1       ;read address date
178     bsr     qspi_rd_byte
179     bsr     _calc        ;create ASCII-code and write to LCD display
180
181     move.b  #'T',d5
182     bsr     lcd_char       ;write 'T' to LCD display
183
184     move.b  #$02,d1       ;read address hour
185     bsr     qspi_rd_byte
186     bsr     _calc        ;create ASCII-code and write to LCD
187
188     move.b  #':',d5
189     bsr     lcd_char       ;write ':' to LCD display
190
191     move.b  #$01,d1       ;read address minutes
192     bsr     qspi_rd_byte
193     bsr     _calc        ;create ASCII-code and write to LCD
194
195     move.b  #' ',d5
196     bsr     lcd_char       ;write ' ' to LCD display
197
198     bsr     wait           ;waiting
199     bsr     _wram         ;write data string to RTC RAM
200
201     bsr     wait           ;waiting
202     bsr     wait           ;waiting
203     bsr     lcd_init
204
205     bsr     _rram         ;read data string from RTC RAM to I
206
207 *     bra     _w02
208
209     rts
210
211 * calculate and create ASCII-Code and write to LCD display
212 _calc

```

```

213         move.w  d2,d5    ;store D2 in D5
214         and.b   #$F0,d5
215         ror.b   #$4,d5   ;rotate upper nibble to lower nibble
216         add.b   #$30,d5  ;create ASCII-code
217         bsr     lcd_char      ;write data to LCD display
218
219         move.w  d2,d5    ;store D2 in D5
220         and.b   #$0F,d5
221         add.b   #$30,d5  ;create ASCII-code
222         bsr     lcd_char      ;write data to LCD display
223
224         rts
225
226 * Initialize the LCD display
227 lcd_init
228         move.b  #$38,LCD_IR  ; DL=1 (8-Bit), N=1, F=0
229         bsr     lcd_busy
230         move.b  #$0f,LCD_IR  ; 1 D C B (Display, Cursor, Blink)
231         bsr     lcd_busy
232         move.b  #$01,LCD_IR  ; Clear Display, Cursor Home
233         bsr     lcd_busy
234         move.b  #$06,LCD_IR  ; 1 I/D S
235         bsr     lcd_busy
236         rts
237
238 * Write a data character to the LCD display
239 lcd_char
240         move.b  D5,LCD_DR
241         bsr     lcd_busy
242         rts
243
244 * Wait until LCD display is ready for another read/write cycle
245 lcd_busy
246         btst    #7,LCD_IR    ; Bit 7 ist 1 falls LCD Busy
247         bne.b  lcd_busy     ; Z-Bit ist invertiertes Bit-7
248         rts
249
250 * Move cursor one position to the right
251 cu_right
252         move.b  #$14,LCD_IR
253         bsr     lcd_busy

```

```

254         rts
255
256
257 * Delay loop *
258 wait
259         move.w  d7,-(a7)
260         move.w  #$FF00,d7           ; number of wait loops
261 _w01     dbra   d7,_w01
262         move.w  (a7)+,d7
263         rts
264
265
266 *write to 96 Byte RTC RAM
267 _wram
268         MOVE.L   #String,A0
269         move.b  #$A0,D1 ;start write adress 96 Byte RAM
270         BSR     _wram2
271         RTS
272
273 _wram2
274         MOVE.B  (A0)+,D0
275         bsr     qspi_wr_byte
276         bsr     qspi_wr_delay ;wait
277         addq.b  #1,D1
278         CMP.W   #0,D0
279         BNE     _wram2
280         RTS
281
282
283 *read from 96 Byte RTC RAM to LCD
284
285 _rram
286         move.b  #$20,d1           ;start read address 96 Byte RAM
287         BSR     _rram2
288         RTS
289
290 _rram2
291         bsr     qspi_rd_byte
292         move.b  d2,d5
293         bsr     lcd_char
294         addq.b  #1,D1

```

```
295         CMP.W    #0,D0
296         BNE      _rram2
297         RTS
298
299
300 String  dc.b    'Hallo',0
301
302         end
303
```