klee@informatik.unibw-muenchen.de 9.3.99

MPROZ – A 16 bit minimal processor

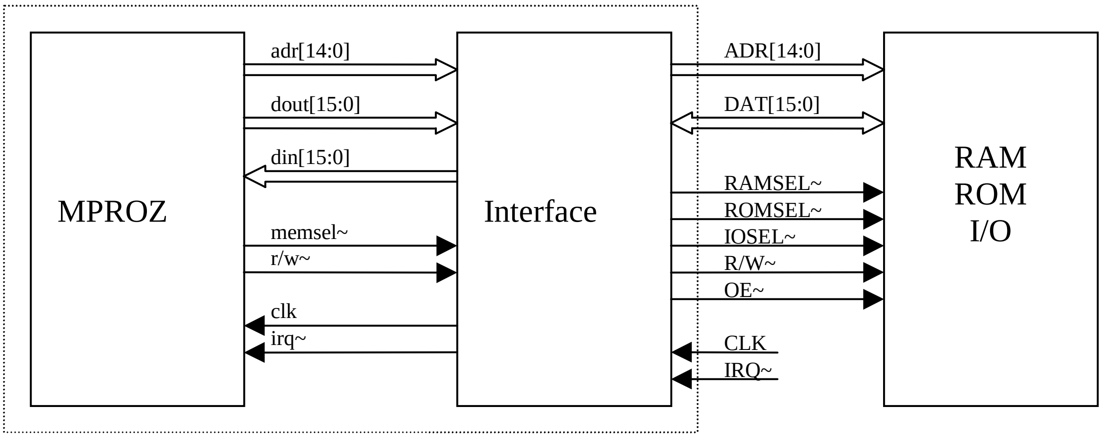
MPROZ is a simplified version of the 16 bit processor XPROZ.

The schematics (WORKVIEW format) and the assembler (QBasic) can be found in mproz.zip

1. Processor architectur

1.1 Interface

ADR[14:0]	: 15 bit address bus
DAT[15:0]	: 16 bit data bus
RAMSEL~	: RAM access
ROMSEL~	: ROM access
IOSEL~	: I/O access
R/W~	: read / write
OE~	: output enable
CLK	: clock
IRQ~	: interrupt request



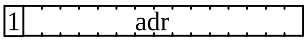
1.2 Register

To minimize the hardware, there is only a 15 bit program counter (PC) and a 1 bit flag (F). No other user accessible registers exist. After a reset PC and F are initialized to 0.

1.3 Instructions

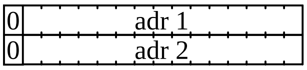
MPROZ supports three instructions:

br adr



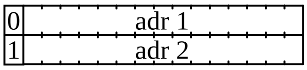
Load adr into PC if F=0
Clear F

add adr1,adr2



Add the contents of memory location adr1 and the contents of memory location adr2 and store the result in memory location adr2. Store the carry of the addition in F.

nor adr1,adr2



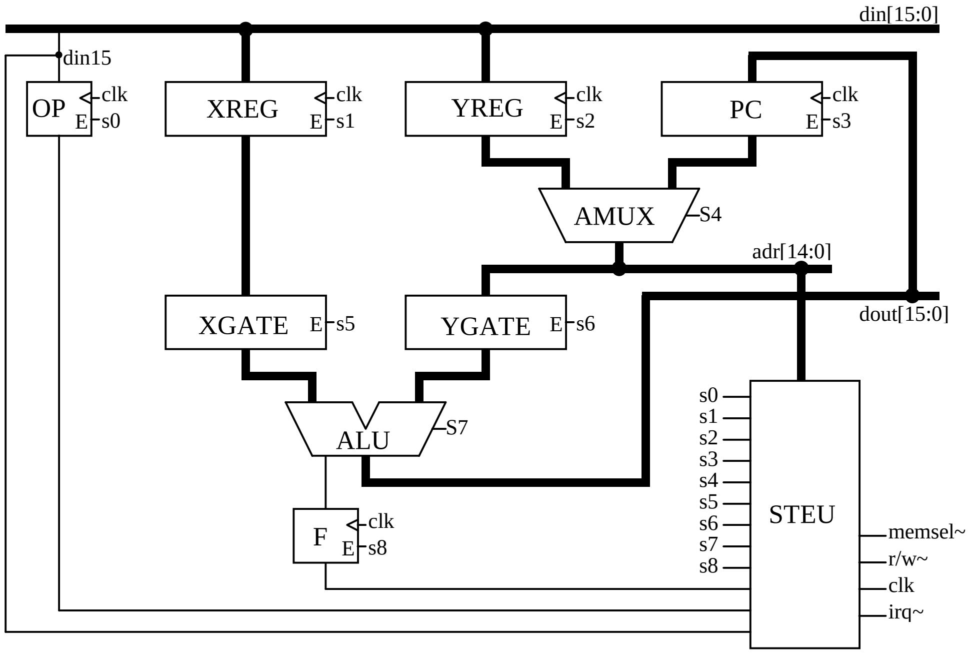
Calculate the NOR function of the contents of memory location adr1 and the contents of memory location adr2 and store the result in memory location adr2. F=1 if result =0 , else F=0.

1.4 Interrupt

Because there is no instruction to set or reset the interrupt enable flag (ie), this is done by accessing memory location $4000. A read from $4000 enables interrupts (ie=1) and a write to $4000 disables interrupts (ie=0). An external interrupt request (IRQ~) must be asserted until the interrupt is serviced by MPROZ. The interrupt is serviced by MPROZ, when the interrupt enable flag is set and a branch instruction with F=0 is executed. Thereby it is not necessary to save the program counter (PC) and the flag (F), but it is sufficient to save the interrupted branch instruction. This instruction is saved at the memory location which address is stored in memory location 1 (this normally should be address $4000, because then the ie flag is automatically cleared, when the instruction is saved). The execution of the interrupt program starts at address 2. The return from interrupt is done by a branch to $4000, which also automatically sets the ie flag.

2. Implementation

2.1 MPROZ



OP:	1 bit register				
F:	1 bit register				
PC:	15 bit register				
XREG:	16 bit register				
YREG:	16 bit register				
AMUX:	IF (s4=0)	THEN	out=in1	ELSE	out=in2
XGATE:	IF (s5=0)	THEN	out=$0000	ELSE	out=in
YGATE:	IF (s6=0)	THEN	out=$0001	ELSE	out=in
ALU:	IF (s7=0)	THEN	out=in1 ADD in2 ,		F=carry
		ELSE	out=in1 NOR in2 ,		F=zero
STEU:	generates control signals s0-s8, memsel~ und r/w~				

state	action
q0	PC→adr, PC+1→PC, din→XREG,YREG
q1	YREG→adr, din→XREG
q2	PC→adr, din→ YREG, din15→OP
q3	YREG→adr, din→YREG
q4	XREG [ADD|NOR] YREG → DOUT, [Carry|Zero] → F, DIN→XREG
q5	PC→adr, PC+1→PC, din→ YREG
q6	YREG→adr, XREG+0→dout
q7	[XREG | 1]+0→PC falls F=0, 0→F

$int = irq\sim + \overline{ie} + F$

Zustand	s8	s7	s6	s5	s4	s3	s2	s1	s0	memsel	r/w~
q0	0	0	1	0	1	1	1	1	-	0	1
q1	-	-	-	-	0	0	-	1	-	0	1
q2	-	-	-	-	1	0	1	0	1	0	1
$q2^*$	-	0	1	0	1	1	1	0	1	0	1
q3	-	-	-	-	0	0	1	0	0	0	1
q4	1	OP	1	1	0	0	-	1	-	1	0
q5	0	0	1	0	1	1	1	0	-	0	1
q6	0	0	0	1	0	0	-	-	-	0	0
q7	1	0	0	int	-	$\overline{F}$	-	0	-	1	1

$q2^*$: for 3 address instructions instead of 2 address instructions

$s0 = q2$

$s1 = q0 + q1 + q4$

$s2 = 1$

$s3 = q0 + q5 + q7 \cdot \overline{F}$

$s4 = q0 + q2 + q5$

$s5 = q4 + q6 + q7 \cdot int$

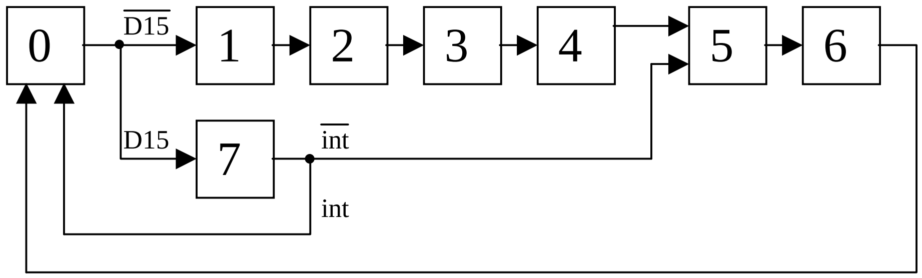
$s6 = \overline{q6 + q7}$

$s7 = q4 \cdot OP$

$s8 = q4 + q7$

$memsel\sim = q4 + q7$

$r/w\sim = \overline{q4 + q6}$



$d0 = q6 + q7 \cdot int$

$d1 = q0 \cdot \overline{D15}$

$d2 = q1$

$d3 = q2$

$d4 = q3$

$d5 = q4 + q7 \cdot \overline{int}$

$d6 = q5$

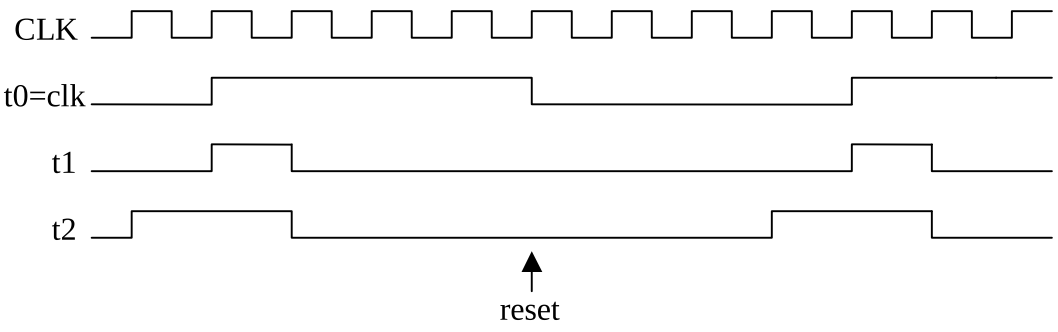
$d7 = q0 \cdot D15$

2.2 Interface

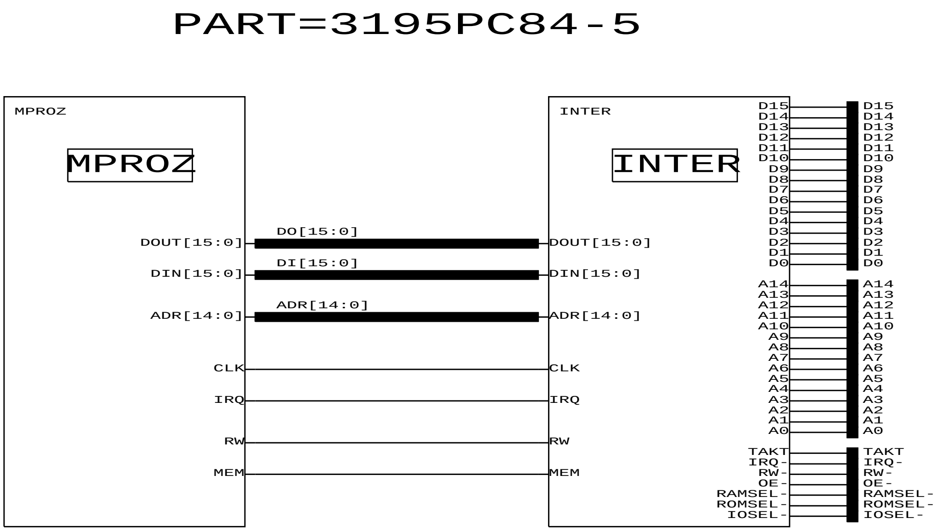
The interface generates the output signals (RAMSEL~, ROMSEL~, IOSEL~, R/W~ and OE~).

address		
$0000 - $3fff	ROMSEL~	64 kByte
$4000 - $7dff	RAMSEL~	63 kByte
$7e00 - $7fff	IOSEL~	1 kByte

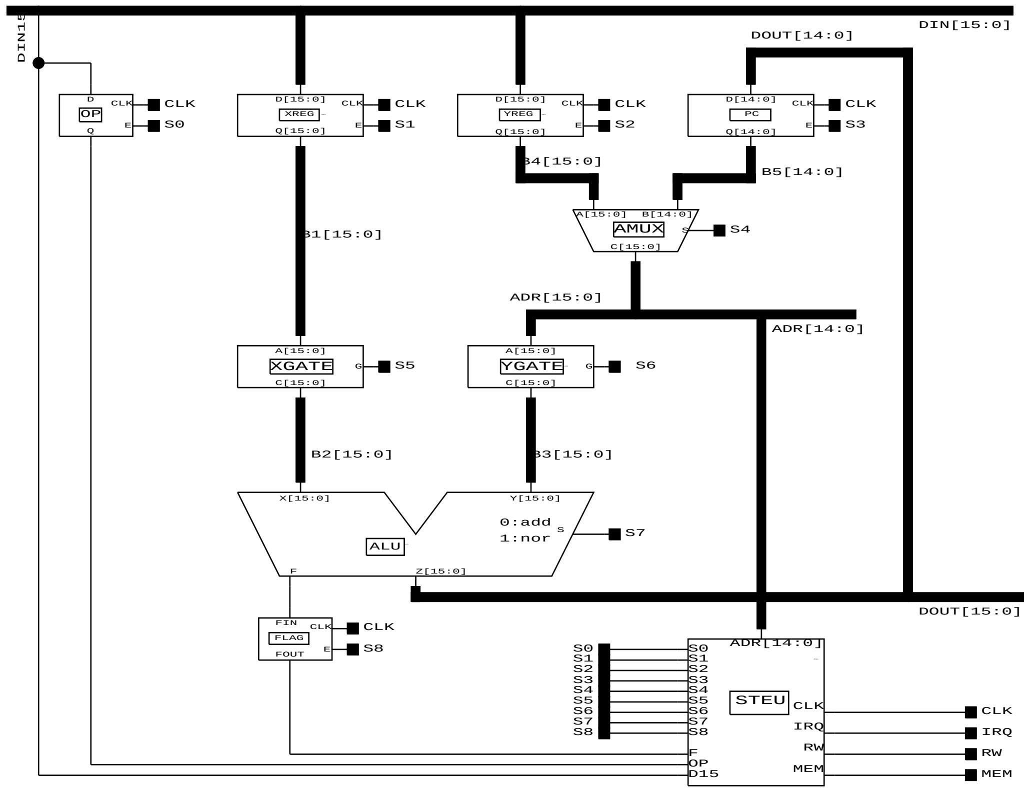
For exact timing, the external clock signal is divided by 8.



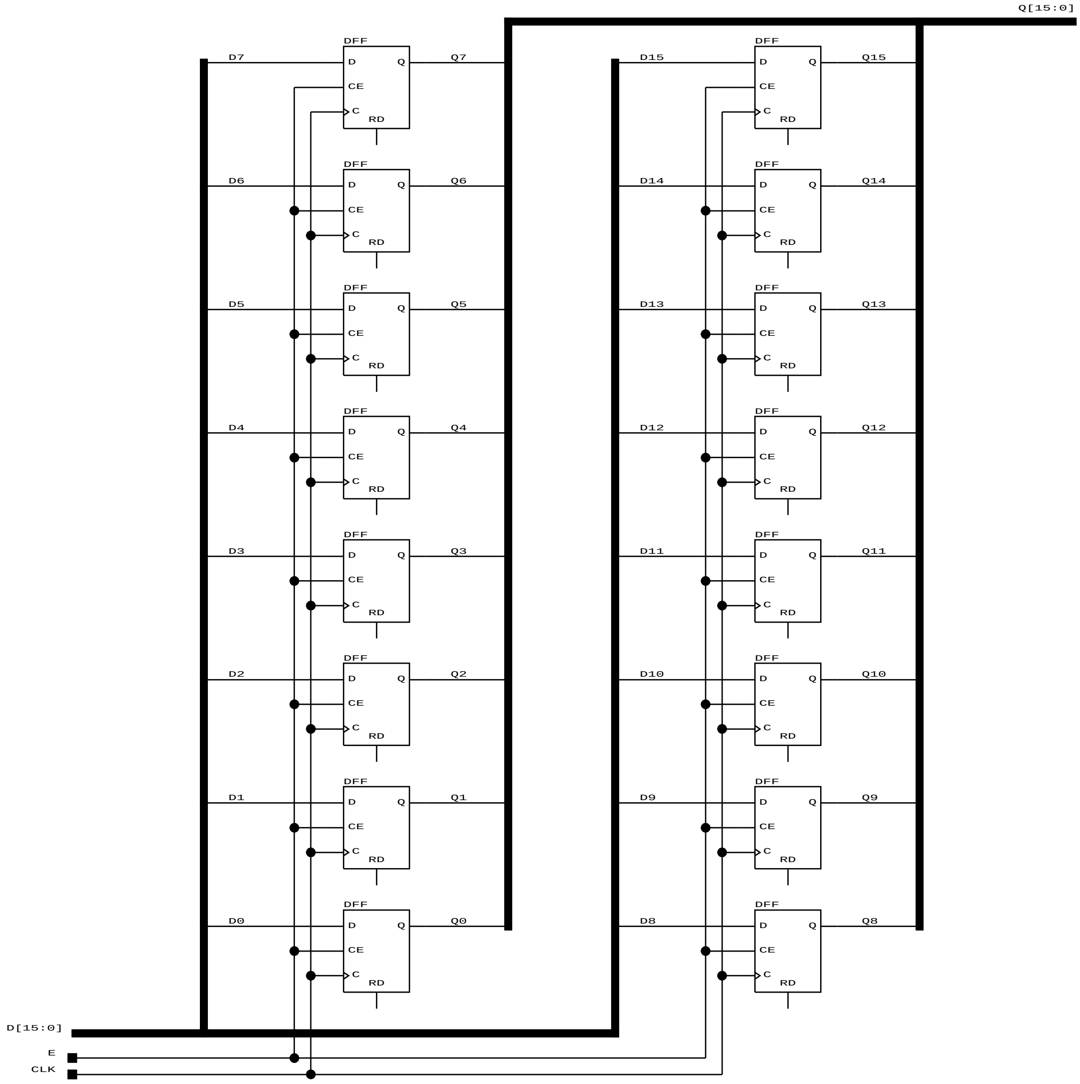
2.3 Schematics



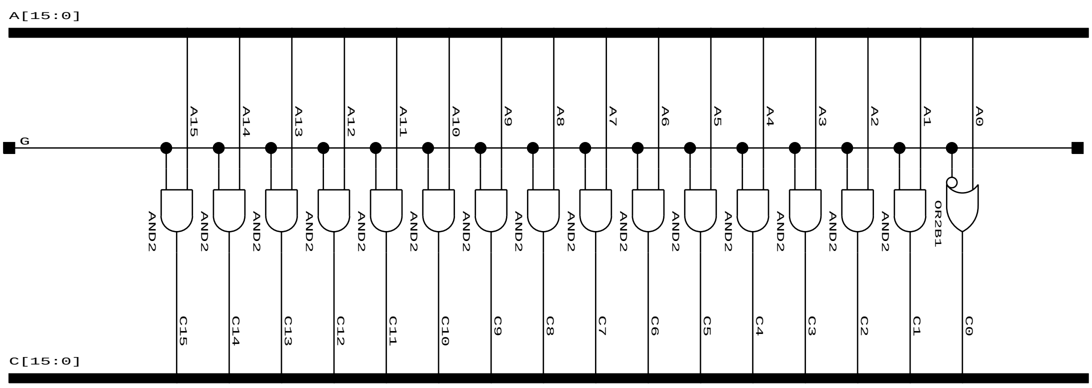
2.3.1 MPROZ



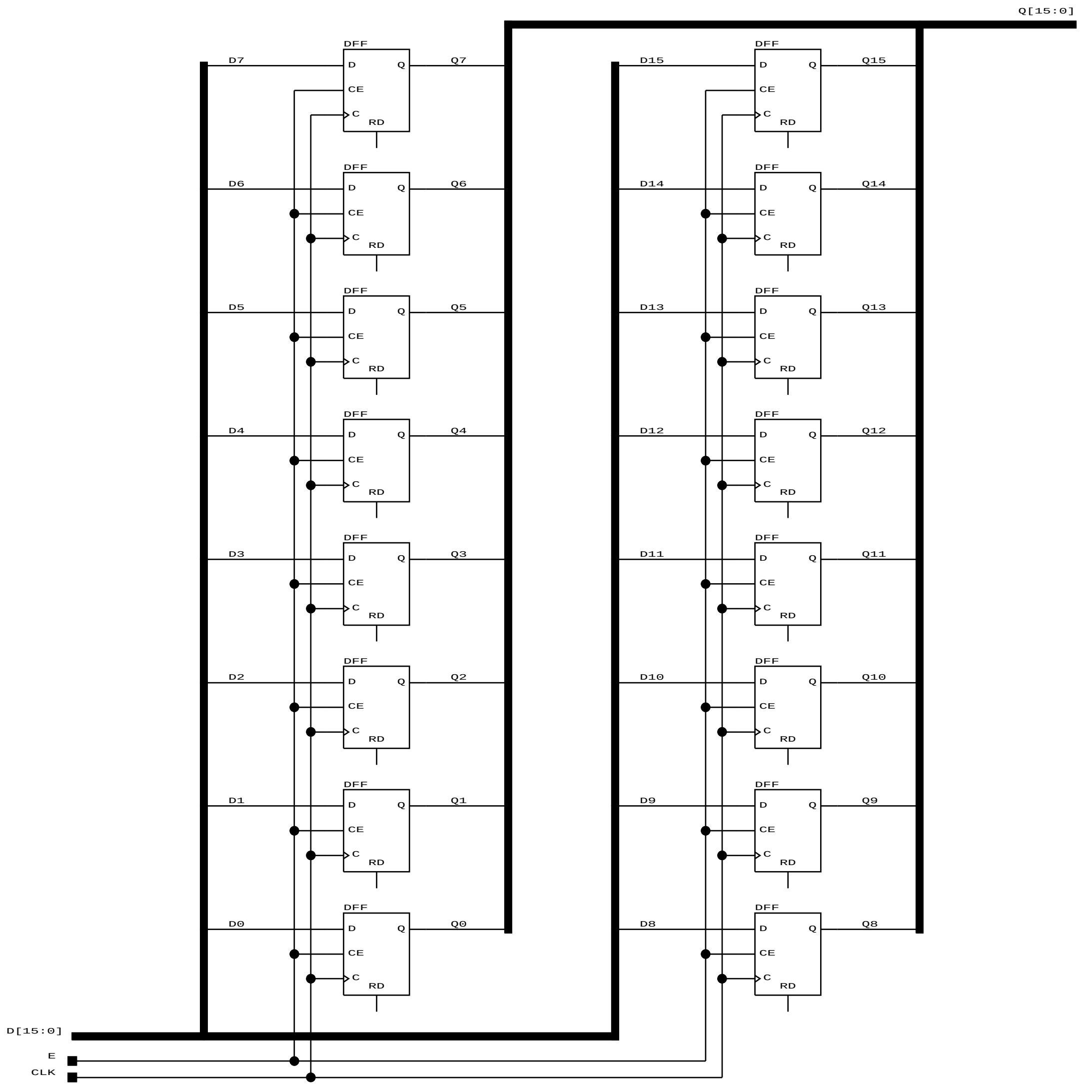
2.3.1.1 XREG



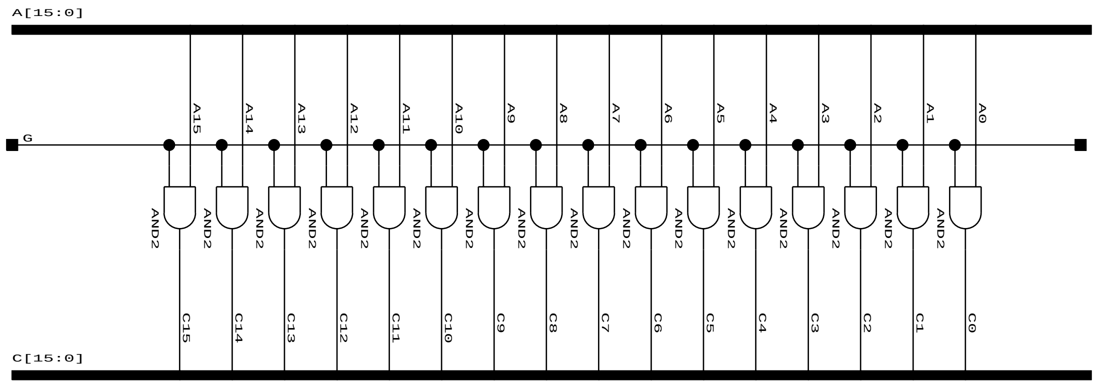
2.3.1.2 XGATE



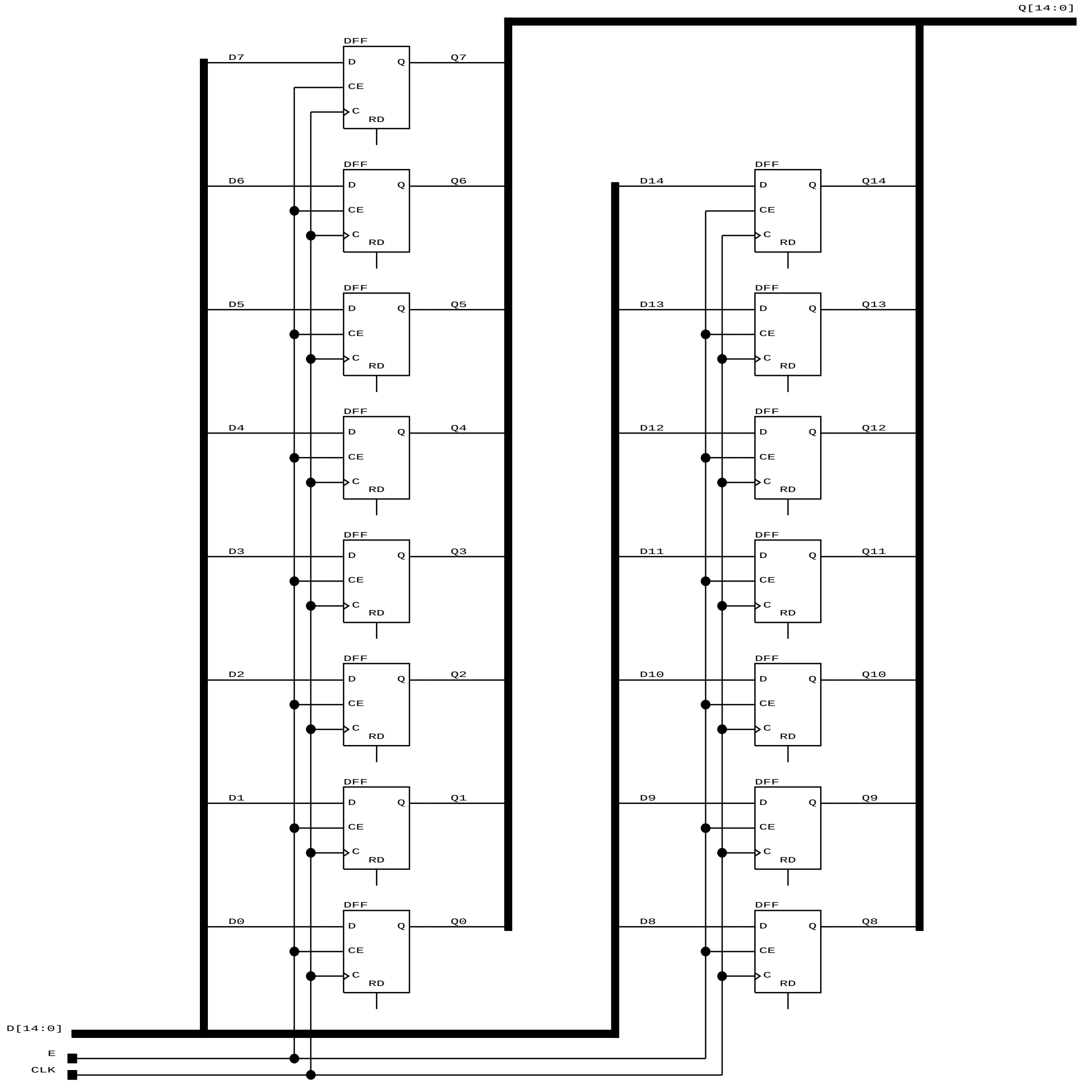
2.3.1.3 YREG



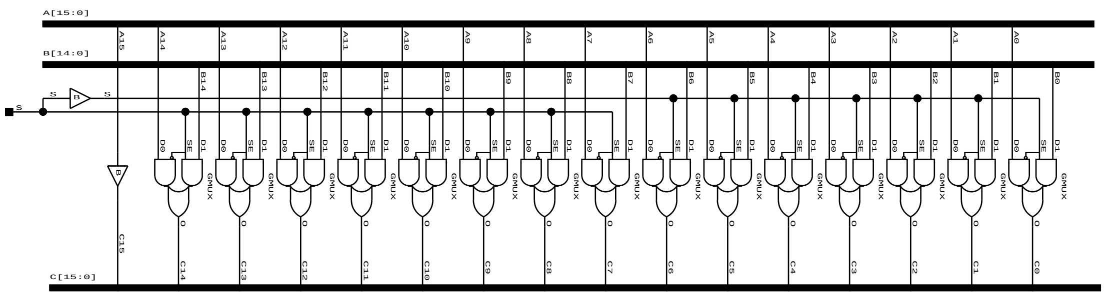
2.3.1.4 YGATE



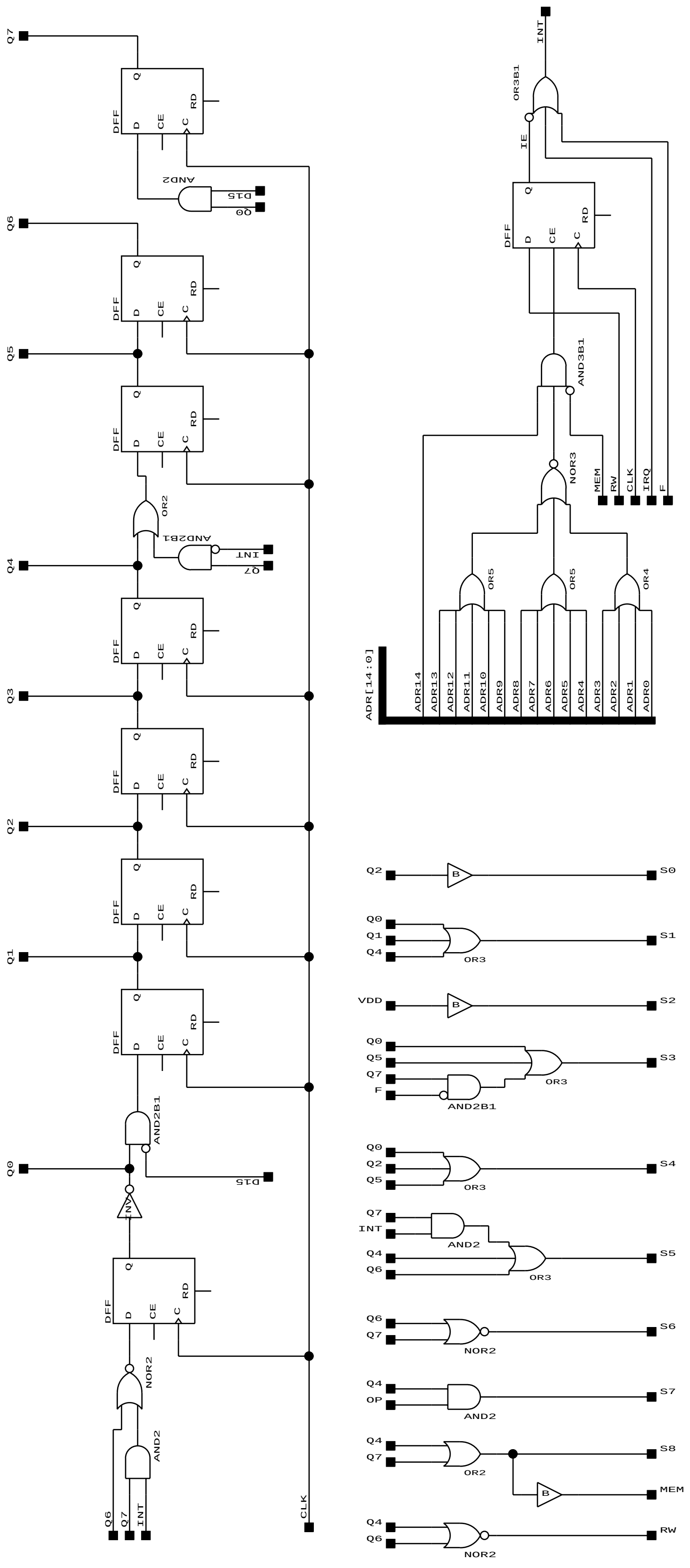
2.3.1.5 PC



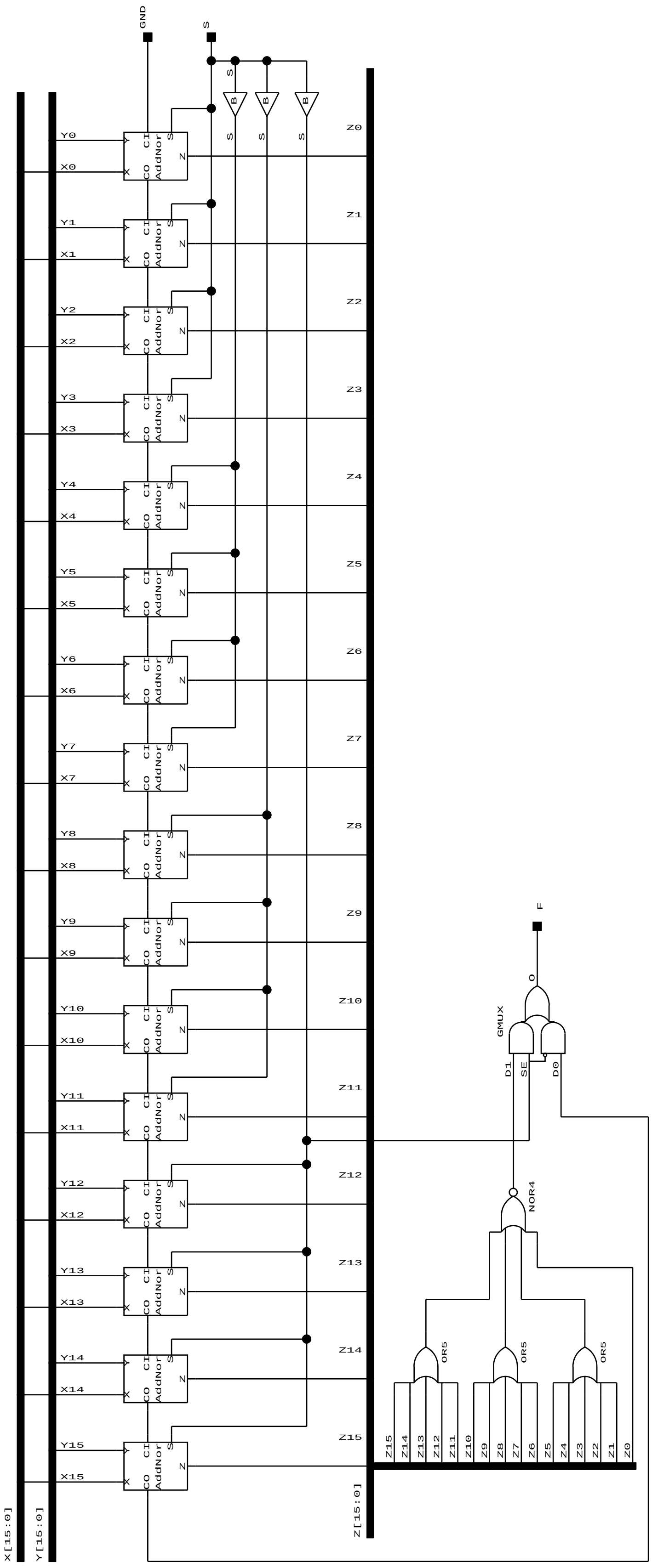
2.3.1.6 AMUX



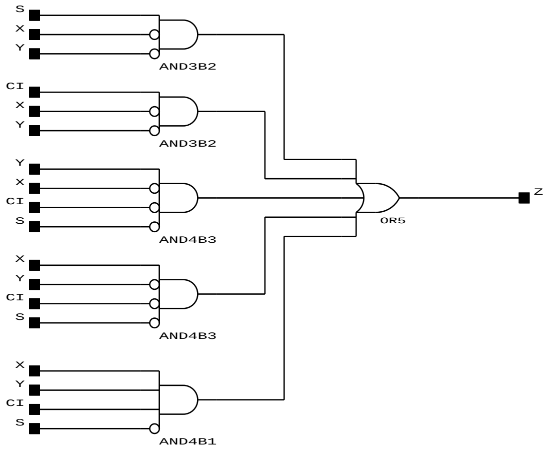
2.3.1.7 STEU

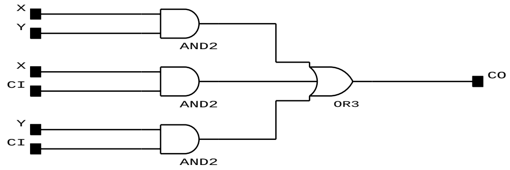


2.3.1.8 ALU

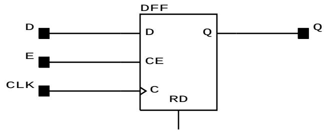

GND
S
B
AddNor
CO
CI
Y0
X0
Z0
Z15
X[15:0]
Y[15:0]
Z[15:0]
F
GMUX
D1
SE
D0
O
NOR4
OR5

2.3.1.9 ADDNOR

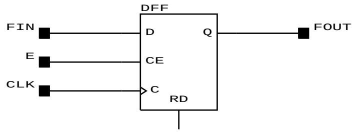




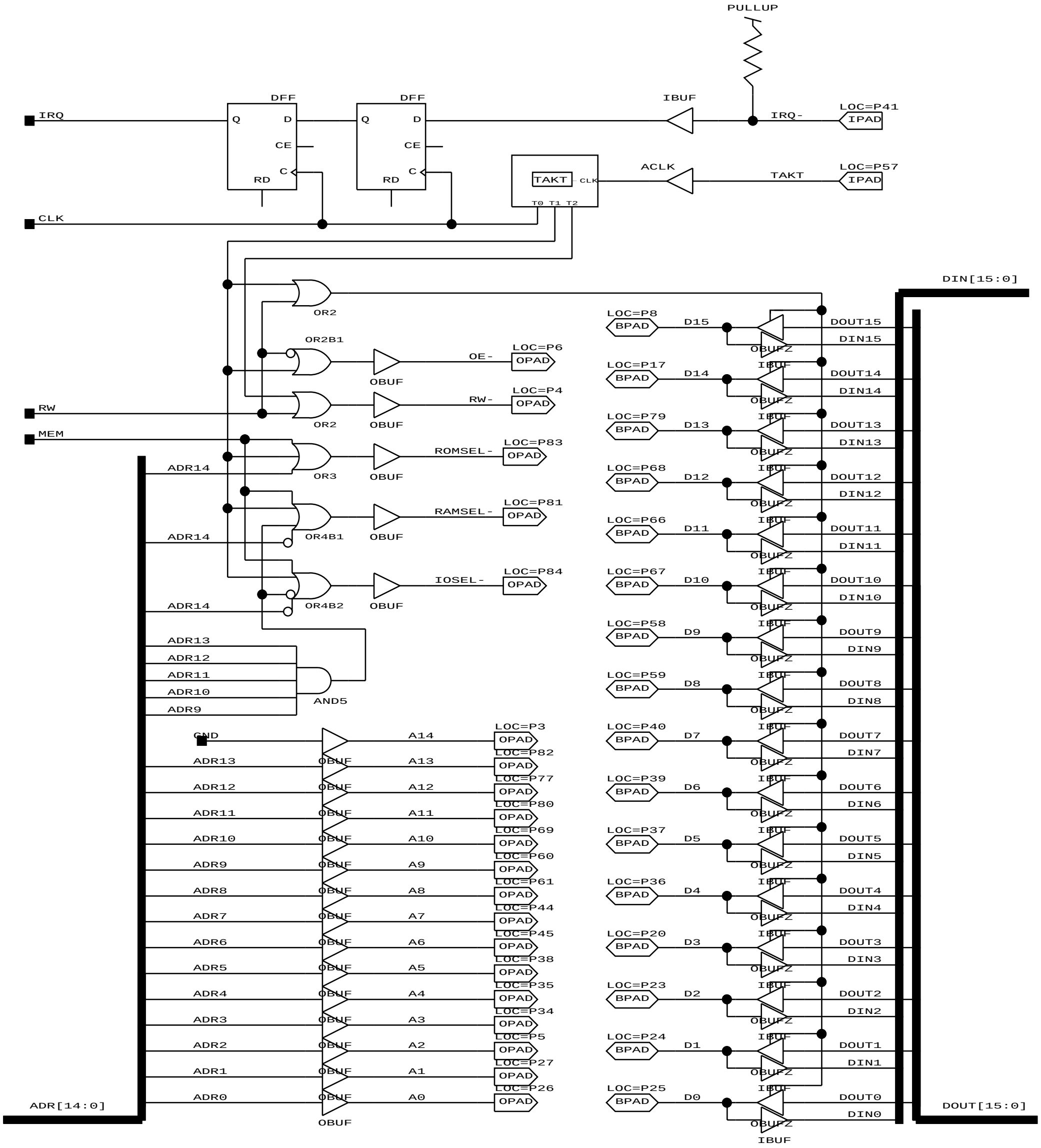
2.3.1.10 OP



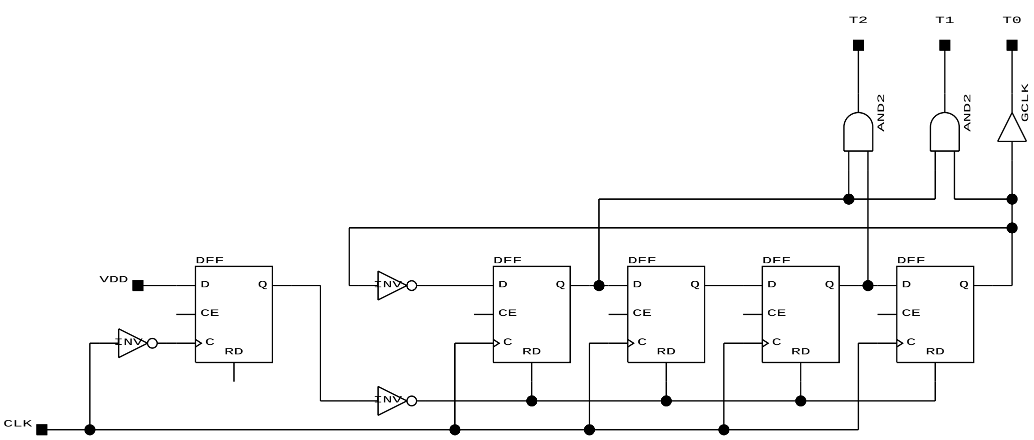
2.3.1.11 FLAG



2.3.2 Interface



2.3.2.1 TAKT



3. Assembler

The assembler supports beyond the three MPROZ instructions (add, nor, br) also the macro instructions move and bsr (branch subroutine) and indirect addressing. For better readability the synonyms bcc and bne can be used instead of br :

```
        add     r0,r1
        add     #20,r0
        add     (r0),r1
        add     r0,(r1)
        add     (r0),(r1)
        bcc     lab1            ; branch to lab1 if F=carry=0
        br      lab1            ; branch to lab1 (now F is always zero)
lab2:   nor     r0,r1
        nor     #20,r0
        nor     (r0),r1
        nor     r0,(r1)
        nor     (r0),(r1)
        bne     lab1            ; branch to lab1 if F=zero=0
        br      lab1            ; branch to lab1 (now F is always zero)
lab3:   move    r0,r1
        move    #20,r0
        move    (r0),r1
        move    r0,(r1)
        move    (r0),(r1)
        br      lab1            ; branch to lab1 (F =0 after a move)
        .
lab1:   bsr     sub
        .
        .
sub:    dcw     0       ; memory location to save a jump to the return address
        .
        .
        .
        br      sub     ; return from subroutine

r0:             dc.w    0
r1:             dc.w    0
#20:            dc.w    20
#$ffff:         dc.w    #$ffff
#$7fff:         dc.w    #$7fff
```

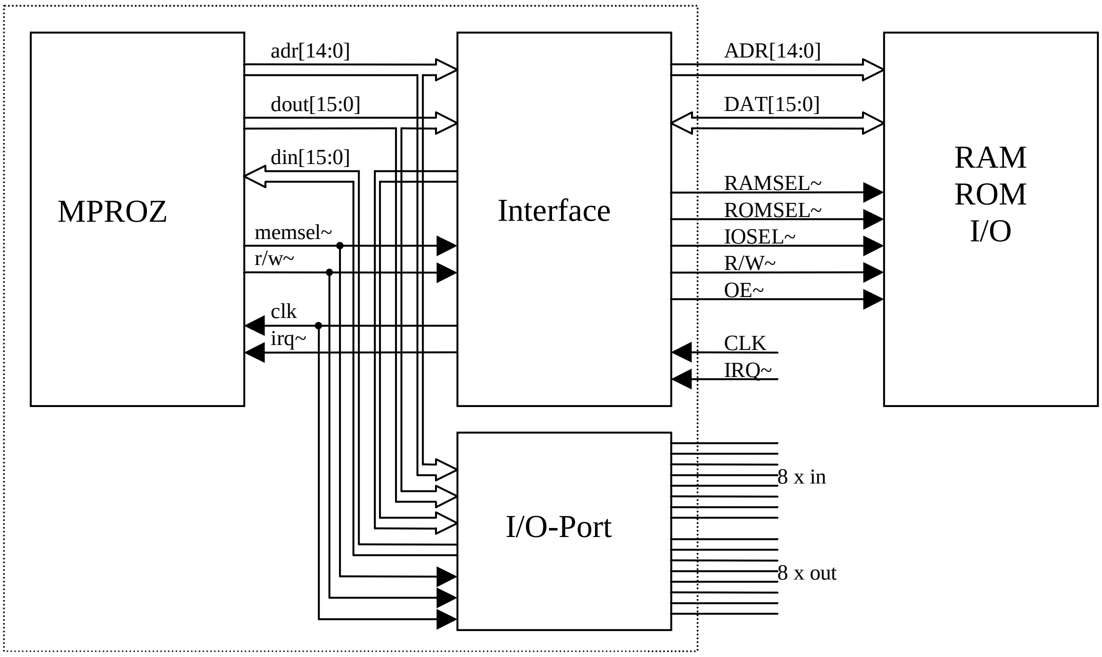
Because this macros are realized with selfmodifying Code, they can be used only in programs stored in RAM but not in ROM. If these macros are used, the following to lines must be inserted into the program:

```
#$ffff:         dc.w    #$ffff
#$7fff:         dc.w    #$7fff
```

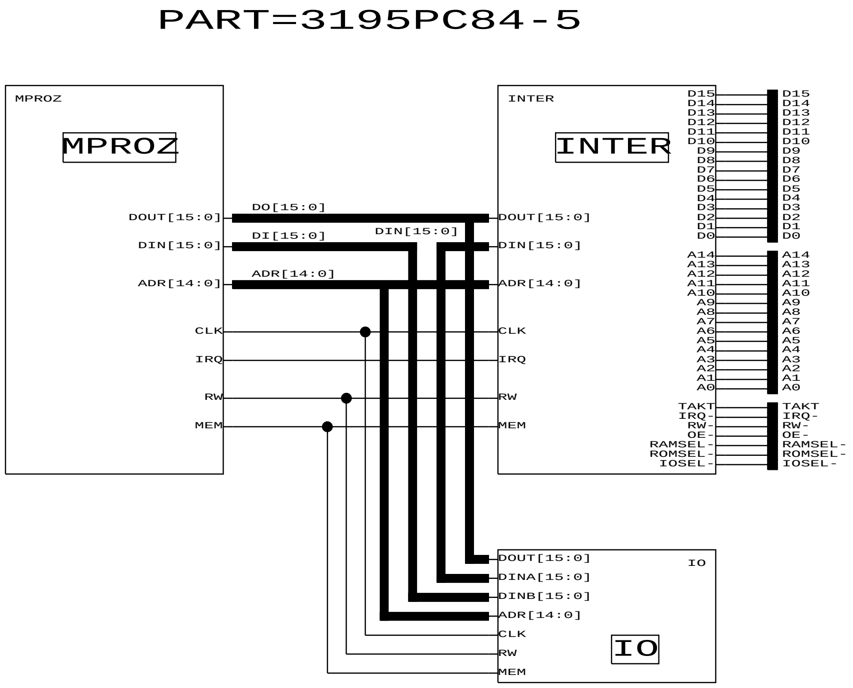
4. Extentions

4.1 In- / Output

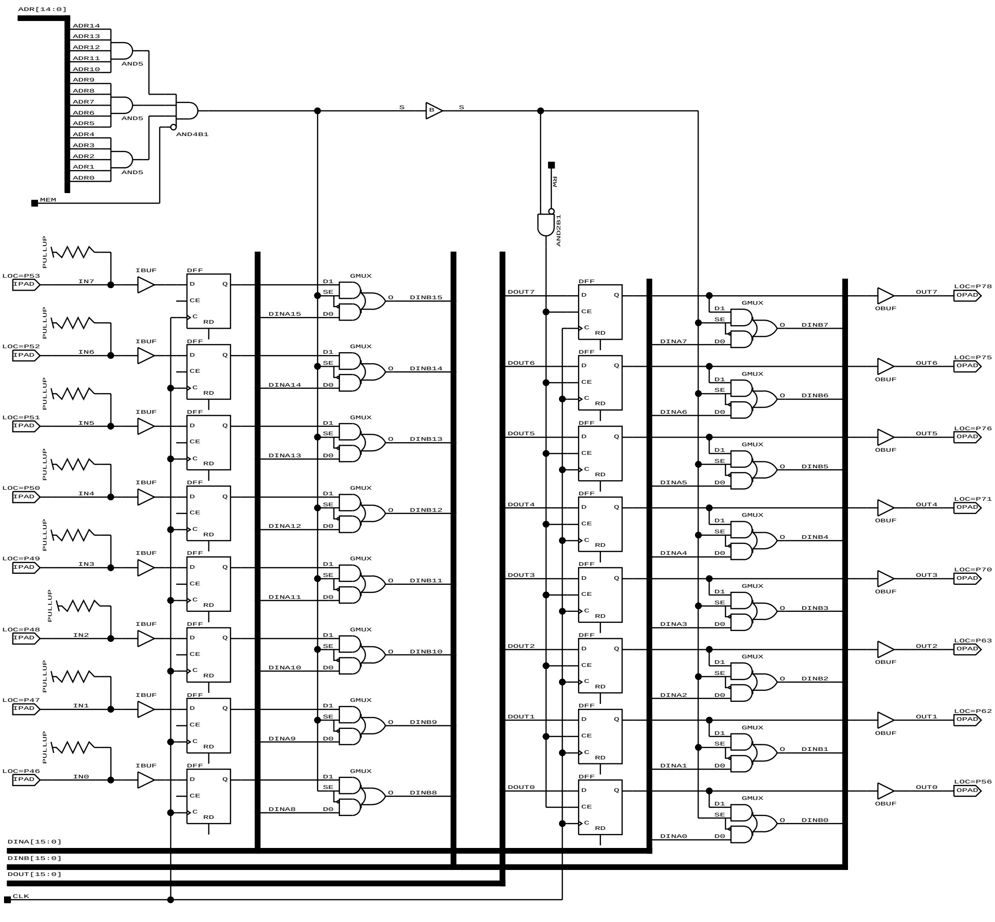
In order to do simple IO without external hardware, an 8 bit input and an 8 bit output port was included in MPROZ. This port is addressed by $7fff . Bit 7-0 belongs to the output and bit 15-8 to the input lines.



4.1.1 Schematics



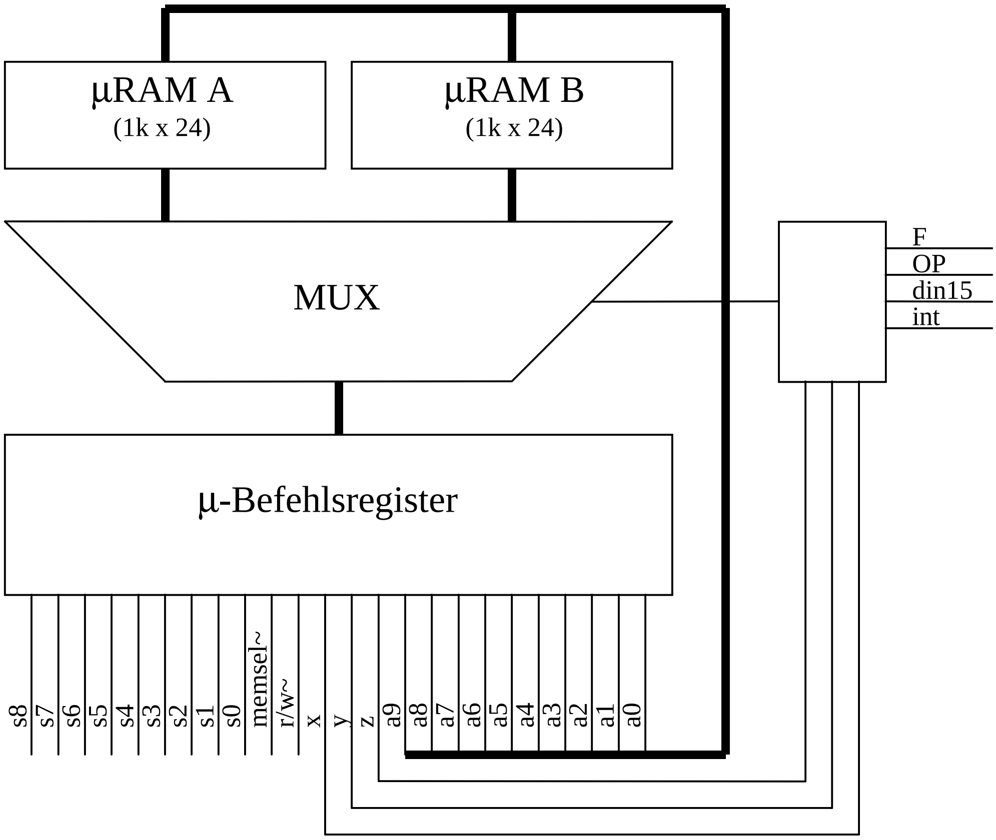
4.1.1 IO



4.2 Microprogramming

4.2.1 Hardware

To show the principles of microprogramming, the control unit of MPROZ is also design in a microprogrammed version :



The three bits x,y,z determine, whether microword A or B is stored into the micro instruction register:

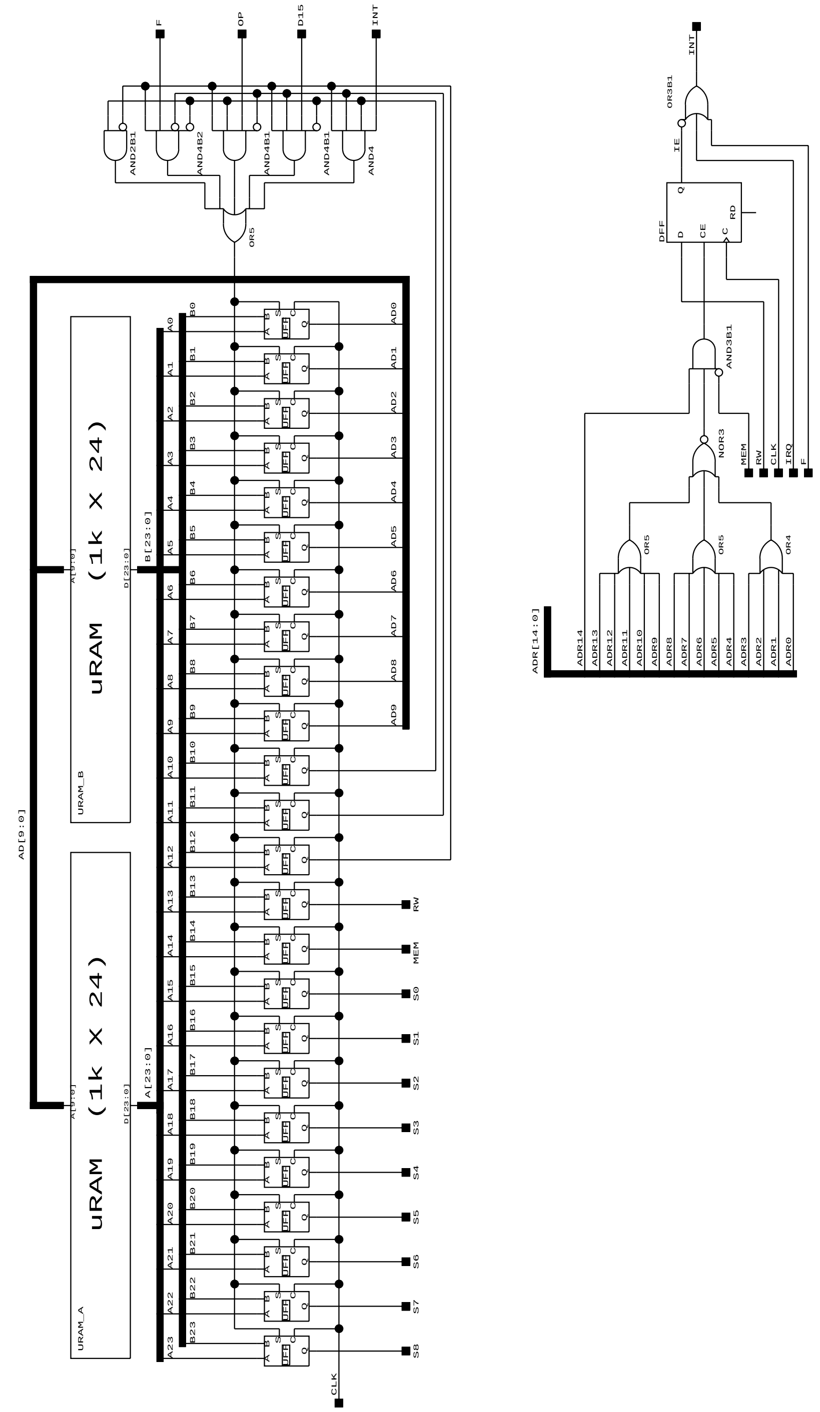
x y z		
0 – 0	A	
0 – 1	B	
1 0 0	A if F=0	B if F=1
1 0 1	A if OP=0	B if OP=1
1 1 0	A if D15=0	B if D15=1
1 1 1	A if INT=0	B if INT=1

4.2.1 Software

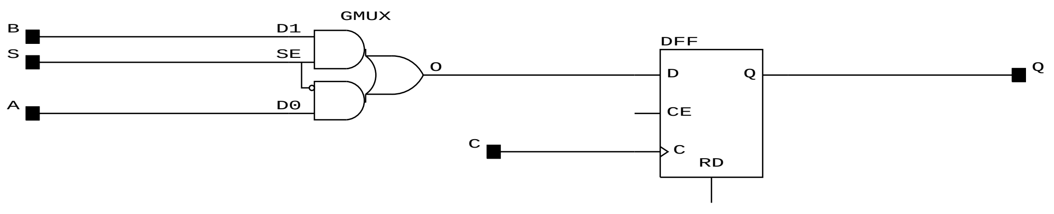
The microprogram for control unit of MPROZ:

	microword A						microword B							
Adresse	s8 s7 s6 s5	s4 s3 s2 s1	s0 memsel~ r/w~ x	y z A9 A8	A7 A6 A5 A4	A3 A2 A1 A0	s8 s7 s6 s5	s4 s3 s2 s1	s0 memsel~ r/w~ x	y z A9 A8	A7 A6 A5 A4	A3 A2 A1 A0	A	B
0	0010	1111	0011	1000	0000	0001	0010	1111	0011	1000	0000	0010	2f3801	2f3802
1	0000	0001	0010	0000	0000	0011	1001	0100	0111	1100	0000	0110	012003	947c06
2	0000	0001	0010	0000	0000	0011	1001	0000	0110	0000	0000	0000	012003	906000
3	0000	1010	1010	0100	0000	0011	0000	0010	0011	0100	0000	0100	0aa403	023404
4	1011	0001	0100	0000	0000	0101	1111	0001	0100	0000	0000	0101	b14005	f14005
5	0010	1110	0010	0100	0000	0101	0001	0000	0001	0000	0000	0000	2e2405	101000
6	1000	0100	0110	0000	0000	0101	0010	1111	0011	1000	0000	0001	846005	2f3801

4.2.3 Schematics

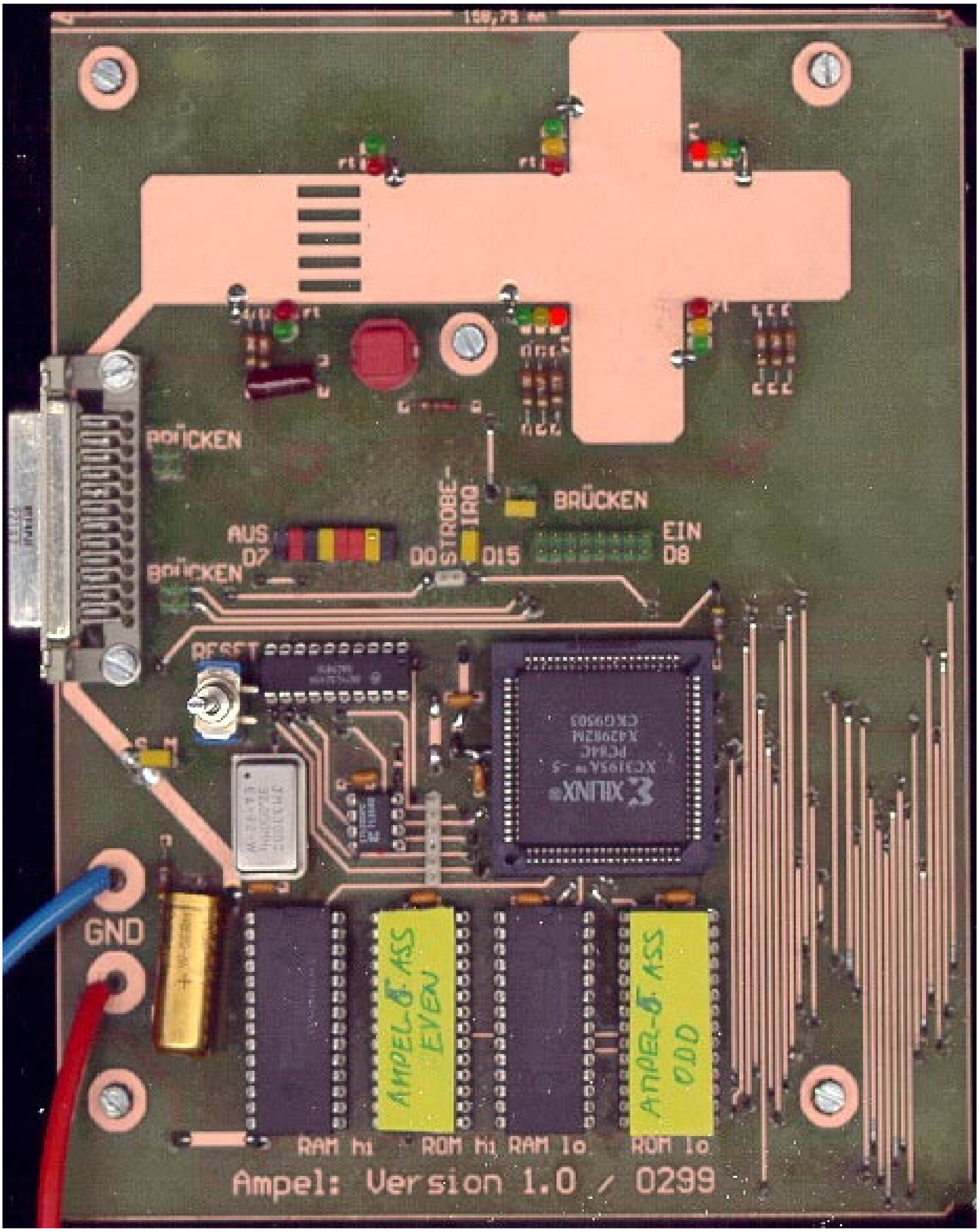


4.2.3.1 UFF



5. Test hardware

To demonstrate that MPROZ works properly, a simple traffic-light was realized.



Program for the traffic-light:

```
;******
;******    ampel_5.ass vom 10.März 1999    ******
;******    mit Fußgängerampel (Interrupt) ******
;
     pc=$0
        br      auto_amp
        dcw     $4000       ; Adresse zum retten des PC bei einem Interrupt
;
;****** Interruptroutine ab Adresse 0002 !!!    ******
;
;        br      irq         ; Adresse der Interruptroutine
irq:
        move    io,r0   ;
        nor     #ff3f,r0    ; Abfrage, ob FUSS AUS
        bne     set_fuss    ; FUSS AUS ---> FUSS auf rot
        move    io,r0
        add     #00c0,r0    ; FUSS auf ROT
        move    r0,io
set_fuss:
        br      $4000       ; Rücksprung aus Interrupt
        br      $4000       ; Rücksprung aus Interrupt
;
;********** Variablen **********
;
#$7fff: dcw     $7fff
#$ffff: dcw     $ffff
#$0001: dcw     $0001
#0000 : dcw     $0000
#0001 : dcw     $0001
;
;********** Die Zeitschleife "zeit" für 22 Sekunden hat im äußeren
;********** Schleifenregister den Wert $96 und im inneren den Wert
;********** $ffff bei einem 32MHz-Quarz (Teiler 8=250µsec)
;
#02s  : dcw     $0001       ; Zeitwert für 0,2 Sekunden
#05s  : dcw     $0003       ; Zeitwert für 0,5 Sekunden
#1s   : dcw     $0007       ; Zeitwert für 1 Sekunden
#2s   : dcw     $000d       ; Zeitwert für 2 Sekunden
#4s   : dcw     $001b       ; Zeitwert für 4 Sekunden
#6s   : dcw     $0029       ; Zeitwert für 6 Sekunden
#8s   : dcw     $0036       ; Zeitwert für 8 Sekunden
#22s  : dcw     $0096       ; Zeitwert für 22 Sekunden
;
#0076: dcw  $0076
#00c0: dcw  $00c0
#00db: dcw  $00db
#00eb: dcw  $00eb
#00f1: dcw  $00f1
#00f5: dcw  $00f5
#00f6: dcw  $00f6
#00e3: dcw  $00e3
#005b :dcw  $005b
#ff00 :dcw  $ff00
#ff3f :dcw  $ff3f
#ffbf: dcw  $ffbf
#feff: dcw  $feff
#ff7f: dcw  $ff7f
#ffff: dcw  $ffff
;
;
;********** Programmstart und Initialisierung **********
;
auto_amp:   move    #ff00,io    ; Alle LEDs AN
            move    #1s,r3
            move    auto_ini1,zeit_ret
```

```
            br      zeit
auto_ini1:  dcw $8000+auto_ini2
auto_ini2:  move    #ffff,io   ; Alle LEDs AUS
            move    #1s,r3
            move    auto_ini3,zeit_ret
            br      zeit
auto_ini3:  dcw $8000+auto_ini3a

auto_ini3a: move    #1s,r6
            nor     #0000,r6
auto_ini4:  move    #0001,r2   ; LED's nacheinander AN
auto_ini5:  move    r2,r4
            nor     #0000,r4
            move    r4,io       ; Ausgabe
            move    #02s,r3
            move    auto_ini6,zeit_ret
            br      zeit
auto_ini6:  dcw $8000+auto_ini7
auto_ini7:  move    r2,r5
            add     r5,r2
            move    r2,r5
            nor     #feff,r5
            bne     auto_ini5
            add     #0001,r6
            bcc     auto_ini4
auto_ini8:  move    #ffff,io   ; Alle LEDs AUS
            move    #2s,r3
            move    auto_ini9,zeit_ret
            br      zeit
auto_ini9:  dcw $8000+auto_ini10

auto_ini10: add     $4000,tmp   ; Lesezugriff: Interrupt enable
;
;********** Ampelsteuerung **********
;
auto_1: ;aampel_1: rt    - 6 sec / aampel_2: gn    - 6 sec; Wert:$xxdb

        move    io,r4   ;
        nor     #ff3f,r4    ; FUSS maskieren
        move    #00db,r5
        nor     #0000,r5
        nor     r5,r4
        move    r4,io

        move    #6s,r3
        move    a1_s,zeit_ret
        br      zeit
a1_s:   dcw $8000+auto_2

auto_2: ;aampel_1: rt    - 2 sec / aampel_2: ge    - 2 sec; Wert:$xxeb

        move    io,r4   ;
        nor     #ff3f,r4    ; FUSS maskieren
        move    #00eb,r5
        nor     #0000,r5
        nor     r5,r4
        move    r4,io

        move    #2s,r3
        move    a2_s,zeit_ret
        br      zeit
a2_s:   dcw $8000+auto_3


auto_3: ;aampel_1: rt/ge - 2 sec / aampel_2: rt    - 2 sec; Wert:$xxf1

        move    io,r4   ;
```

```
        nor     #ff3f,r4    ; FUSS maskieren
        move    #00f1,r5
        nor     #0000,r5
        nor     r5,r4
        move    r4,io
        move    #2s,r3
        move    a3_s,zeit_ret
        br      zeit
a3_s:   dcw $8000+auto_4

auto_4: ;aampel_1: gn    - 6 sec / aampel_2: rt    - 6 sec; Wert:$xxf6

        move    io,r4   ;
        nor     #ffbf,r4    ; Abfrage, ob FUSS rot
        bne     a4_1        ; FUSS ROT?
        move    #00f6,io    ; Normalbetrieb
        br      a4_2
a4_1:   move    #0076,io    ; FUSS auf grün setzen!
a4_2:   move    #6s,r3
        move    a4_s,zeit_ret
        br      zeit
a4_s:   dcw $8000+auto_5

        move    #6s,r3
        move    a4_s,zeit_ret
        br      zeit
a4_s:   dcw $8000+auto_5

auto_5: ;aampel_1: ge    - 2 sec / aampel_2: rt    - 2 sec; Wert:$xxf5

        move    io,r4
        nor     #ff7f,r4    ; FUSS grün?
        bne     a5_1
        move    io,r4       ; FUSS nicht grün
        nor     #ff3f,r4
        move    #00f5,r5
        nor     #0000,r5
        nor     r5,r4
        move    r4,io
        br      a5_2
a5_1:   move    #00f5,io    ; FUSS AUS
a5_2:   move    #2s,r3
        move    a5_s,zeit_ret
        br      zeit
a5_s:   dcw $8000+auto_6

auto_6: ;aampel_1: rt    - 2 sec / aampel_2: rt/ge - 2 sec; Wert:$xxe3

        move    io,r4   ;
        nor     #ff3f,r4    ; FUSS maskieren
        move    #00e3,r5
        nor     #0000,r5
        nor     r5,r4
        move    r4,io
        move    #2s,r3
        move    a6_s,zeit_ret
        br      zeit
a6_s:   dcw $8000+auto_1

;************   Zeitschleife         *************
;************   Zählerwert in r3     ************

zeit:   nor #$ffff,r1
        nor #0000,r3
        add #$0001,r3
w1i:    add #$0001,r1
        bcc w1i
```

```
        add #$0001,r3
        bcc w1i
        br  zeit_ret

;************   Umschalten auf RAM   ************
;
     pc=$4000
        dcw     0       ; Speicherzelle zum retten des PC bei einem Interrupt


r0:     dcw     0
r1:     dcw     0
r2:     dcw     0
r3:     dcw     0
r4:     dcw     0
r5:     dcw     0
r6:     dcw     0
doio_ret:dcw    0
zeit_ret:dcw    0
tmp:    dcw     0
io=$7fff

;****** Die Fußgängerampel schaltet mit aampel_2 (Querrichtung)
;
;auto_1=01xx: ;aampel_1: rt    - 6 sec / aampel_2: gn    - 6 sec; Wert:$xxdb
;auto_2=02xx: ;aampel_1: rt    - 2 sec / aampel_2: ge    - 2 sec; Wert:$xxeb
;auto_3=03xx: ;aampel_1: rt/ge - 2 sec / aampel_2: rt    - 2 sec; Wert:$xxf1
;auto_4=04xx: ;aampel_1: gn    - 6 sec / aampel_2: rt    - 6 sec; Wert:$xxf6
;auto_5=05xx: ;aampel_1: ge    - 2 sec / aampel_2: rt    - 2 sec; Wert:$xxf5
;auto_6=06xx: ;aampel_1: rt    - 2 sec / aampel_2: rt/ge - 2 sec; Wert:$xxe3
;fuss_1=07xx: ;fussampel: rot  - Beginn in auto_3 bis auto_6
;fuss_2=08xx: ;fussampel: grün - 8 sec in Phase auto_1 und auto_2

end:_
```