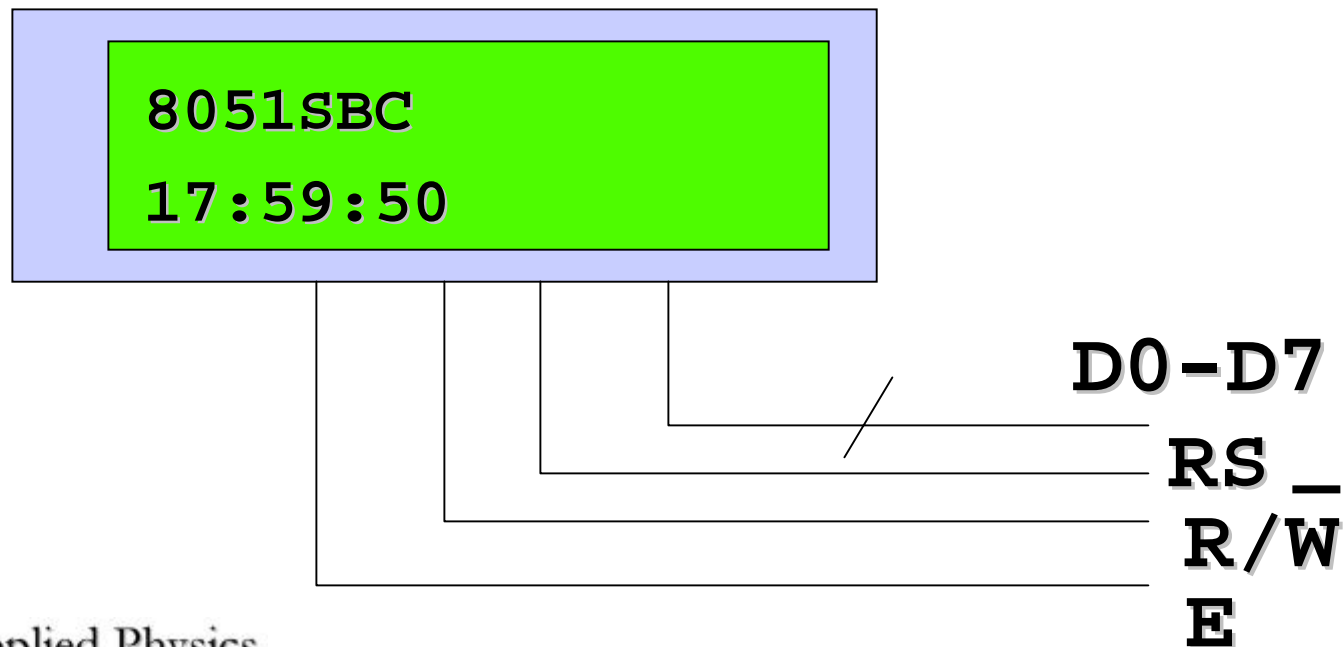


# Onboard LCD driver: hardware

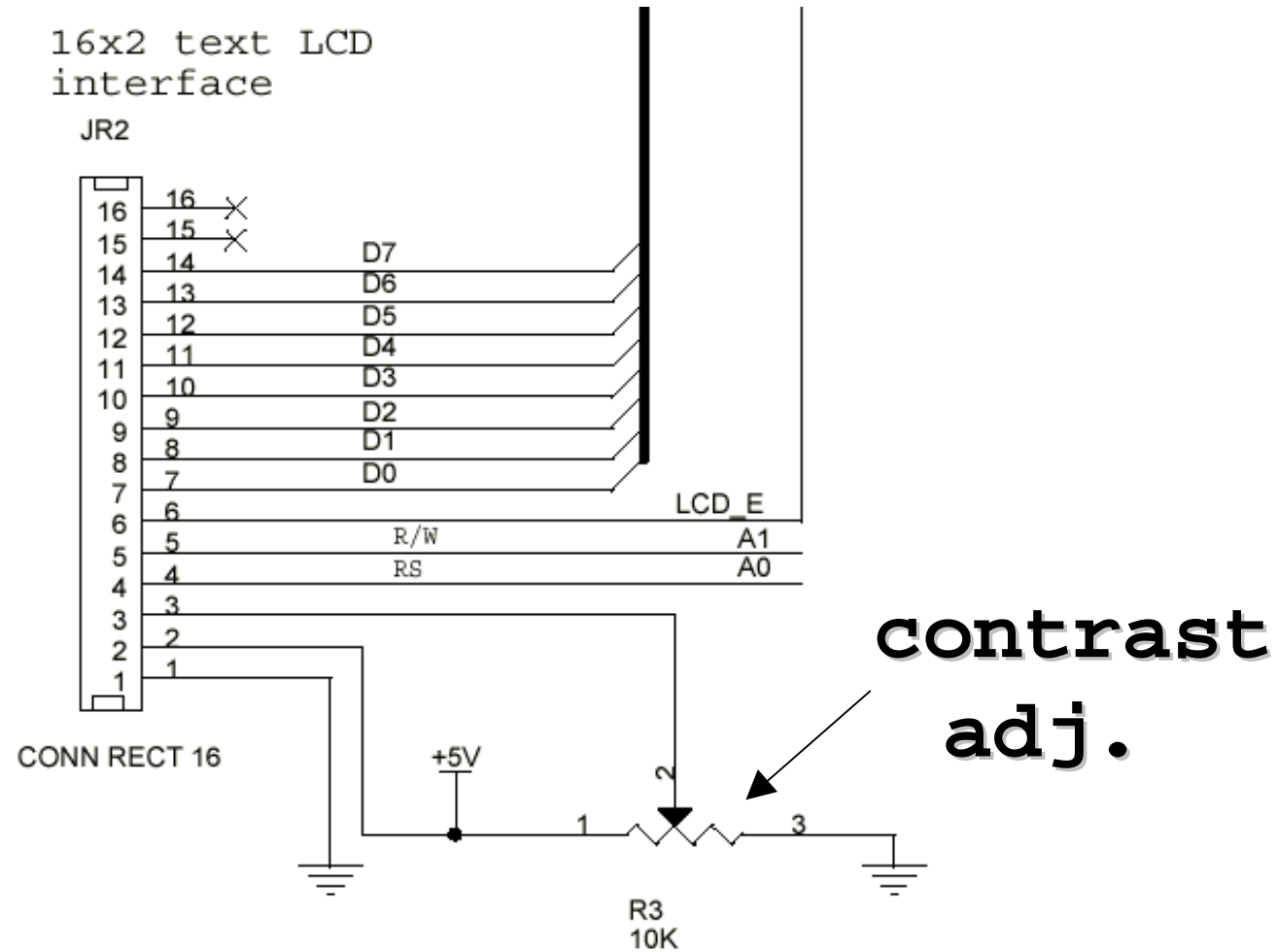
---

8-bit CPU bus interface

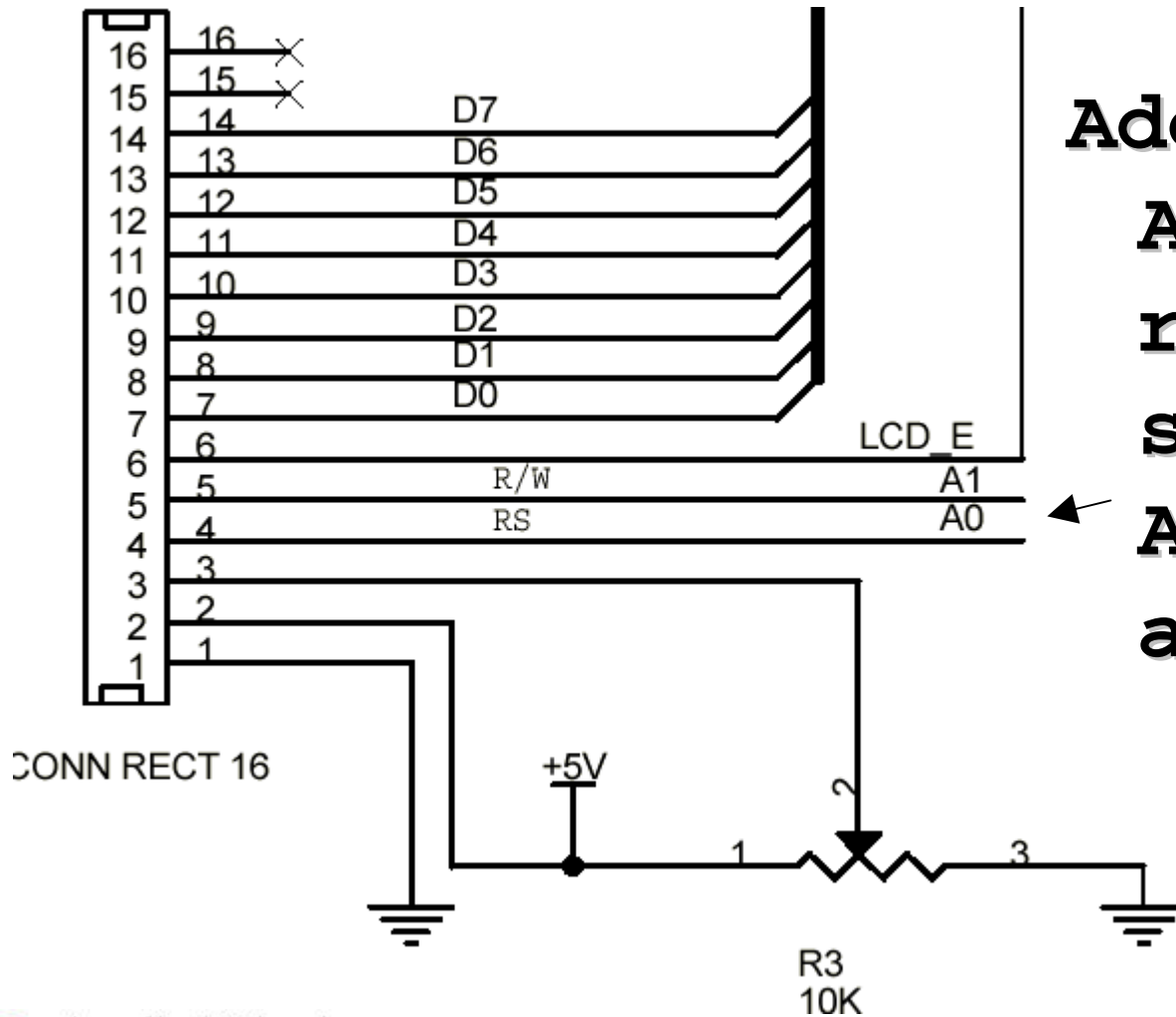
16x2 characters Liquid Crystal  
Display (LCD) HD4478 controller



# Onboard LCD driver: hardware



# Onboard LCD driver: A0 and A1



Address line  
A0 ties to  
register  
select and  
A1 to read  
and write.

# Onboard LCD driver: Registers MAP

---

memory mapped I/O  
using external  
data memory

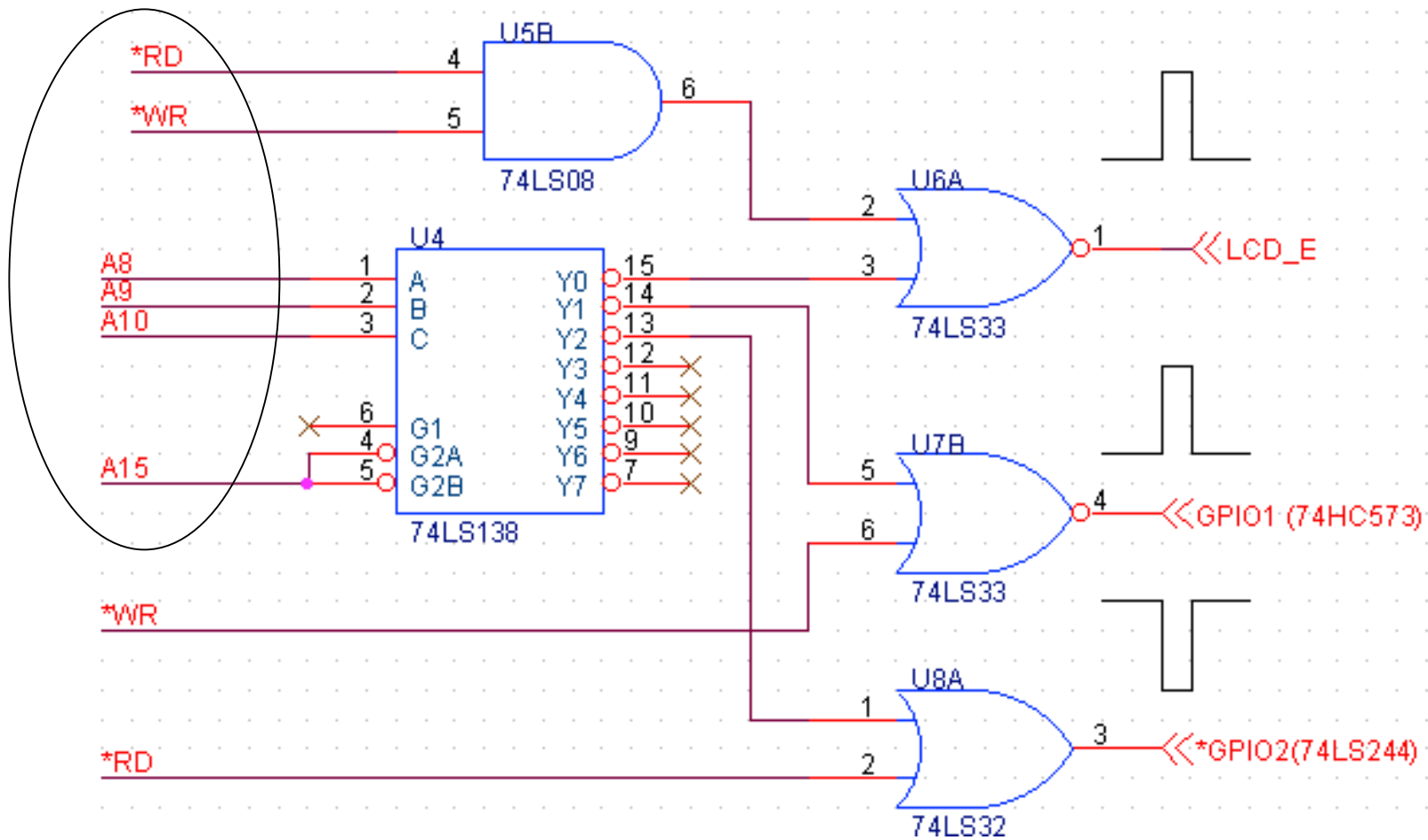
**0x0000-0x0003**

0x0000	lcd command write
0x0001	lcd data write
0x0002	lcd command read
0x0003	lcd data read

0x0100	GPIO1
0x0200	GPIO2
0x0300	expansion i/o space
0x07FF	

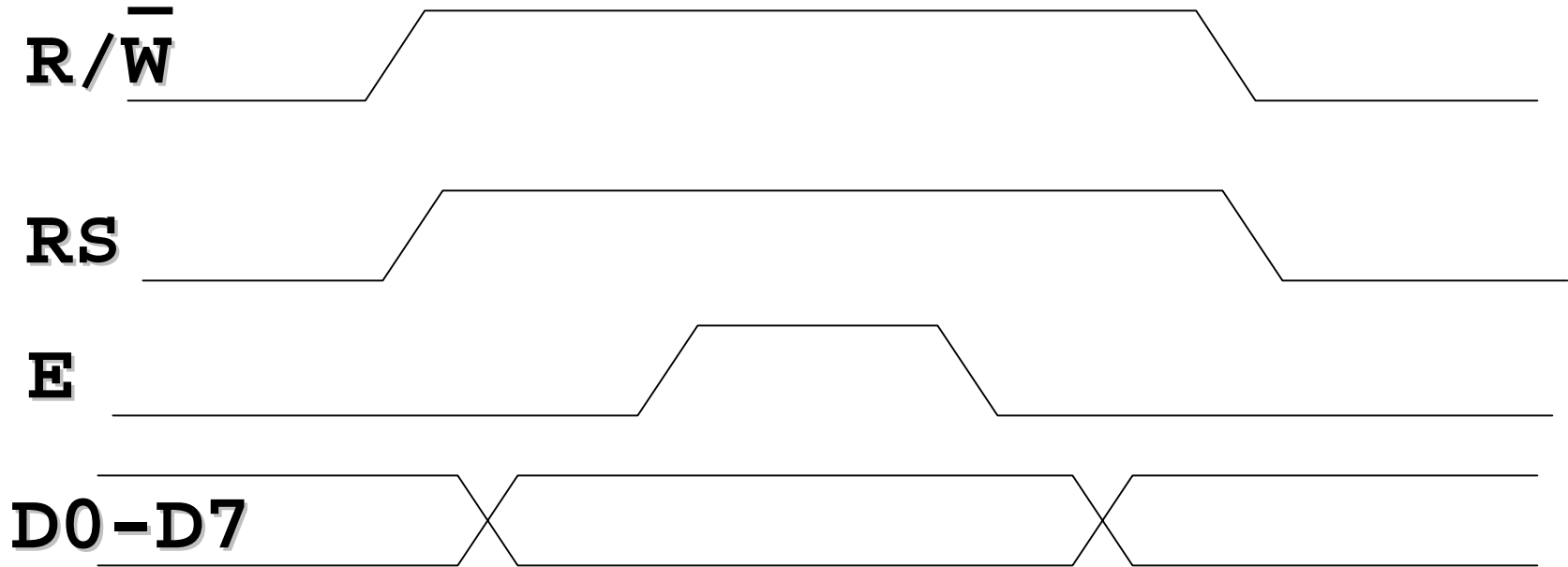
**A0=0 command reg.**  
**A0=1 data register**  
**A1=0 write**  
**A1=1 read**

# Onboard LCD: Enable signal



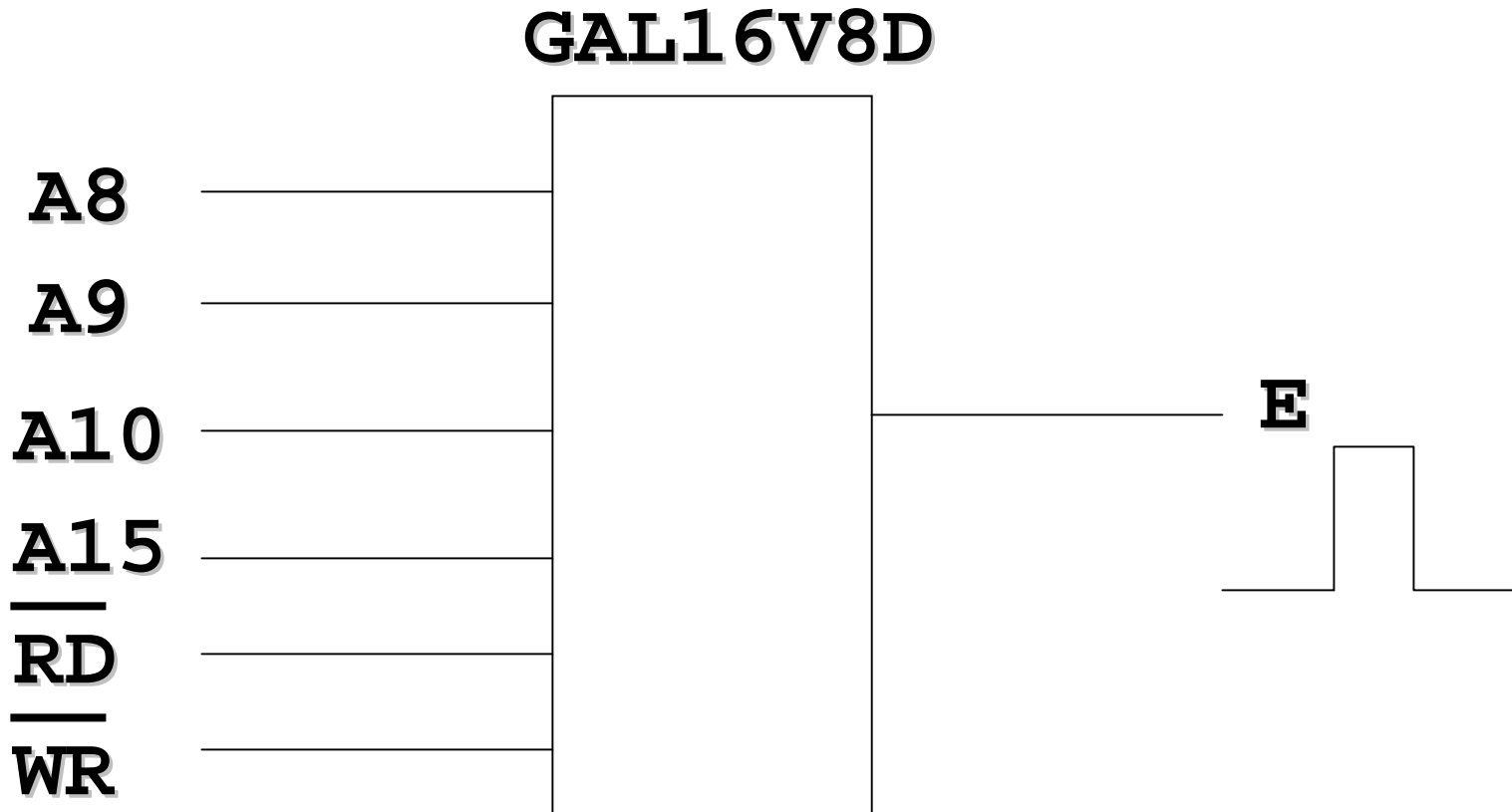
# Onboard LCD: Bus Timing

---



# PLD Decoder: GAL16V8D

---

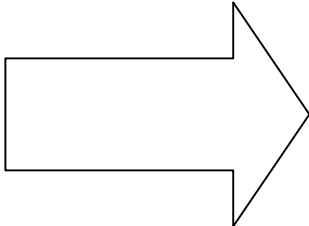
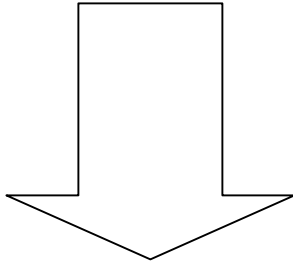


**Reprogrammable Generic  
Array Logic**

# PLD Assembler: OPALjr

---

Logic Equation



JEDEC File

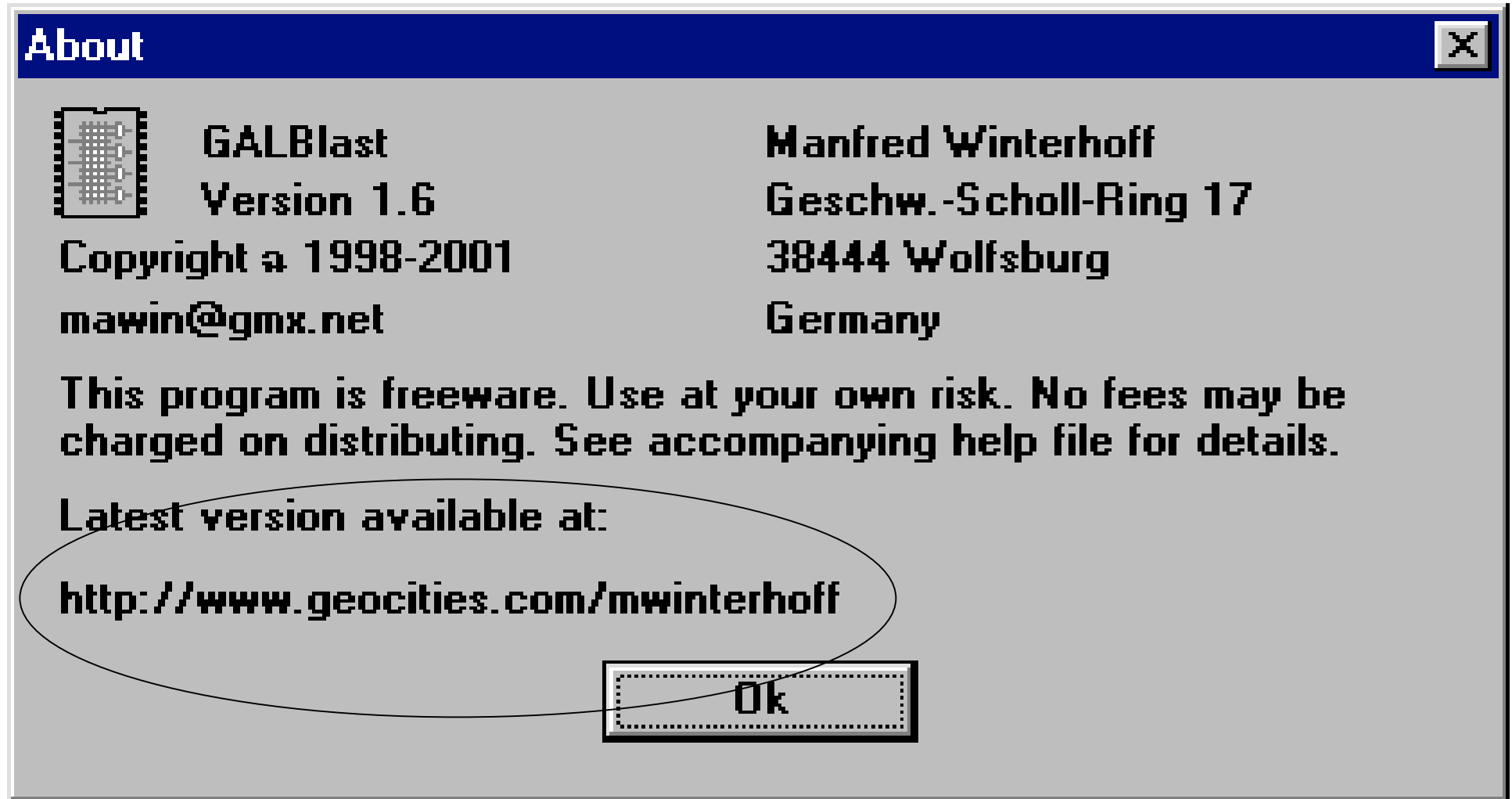
National Semiconductor Corp.





# GAL Programmer: GALBlast

---



# PLD Equation for 8051SBC

---

**; 8051SBC PLD Equation**  
**; Memory and I/O decoder for 8051**  
**; Single Board Computer**  
**; Wichit Sirichote,**  
**; kswichit@kmitl.ac.th**  
**; May 1, 2003**  
**; PLD: Lattice GAL16V8D**  
**CHIP 8051SBC GAL16V8**

# PLD Equation: pin designation

---

a15=1 rd=2 wr=3 a8=4 a9=5 a10=6

rs232=7 rs485=8 psen=9

rom\_ce=12

input pins

ram\_ce=13 ram\_oe=14 rom\_oe=15

lcd\_e=16 gpio1=17 gpio2=18

rx=19

output pins

# PLD Equations:

---

## EQUATIONS

`ram_oe = rd * psen`

`rxd = rs232 * rs485`

`rom_ce = a15`

`ram_ce = /a15`

`rom_oe = psen`

`/lcd_e = rd * wr + a8 + a9 + a10  
+ a15`

# JEDEC File: 8051sbc.jed

---

GAL16V8

EQN2JED - Boolean Equations to JEDEC file assembler (Version V101)

Copyright (c) National Semiconductor Corporation 1990-1993

Assembled from "D:\C52EVBV3\O\8051SBC.EQN". Date: 5-1-103

\*

NOTE PINS a15:1 rd:2 wr:3 a8:4 a9:5 a10:6 rs232:7 rs485:8  
psen:9\*

NOTE PINS rom\_ce:12 ram\_ce:13 ram\_oe:14 rom\_oe:15 lcd\_e:16  
gpio1:17\*

NOTE PINS gpio2:18 rxd:19\*

NOTE GALMODE SMALL\*

QF2194\*QP20\*F0\*

L0000

1111111111111111111111110111011111111\*

L0256

111111110111111111111111111111111111

111111111111110111111111111111111111

111111111111111111011111111111111111

110111111111111111111111111111111111

011111111111111111111111111111111111\*

L0512

# LCD Programming Model:

---

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

**Location of display  
characters 16x2 lines**

# LCD Programming Model:

---

command write register= 0000H

command read register = 0001H

data write register = 0002H

data read register = 0003H

# LCD Programming Model:

---

- Initialize LCD for our applications,
- set address for ASCII code to be loaded into LCD display RAM,

▶ • write ASCII code,   
Auto increment

# LCD Programming: LCD's Registers

---

**; LCD driver for 8051SBC**

**BUSY EQU 80H**

**; LCD's registers are mapped**

**; into external data memory**

**command\_write EQU 0000H**

**data\_write EQU 0001H**

**command\_read EQU 0002H**

**data\_read EQU 0003H**

# LCD Programming: Busy Flag

---

```
;wait until LCD busy bit clear
```

```
LcdReady: PUSH ACC
```

```
MOV DPTR, #command_read
```

```
?ready: MOVX A, @DPTR
```

```
JB ACC.7, ?ready
```

```
POP ACC
```

```
RET
```

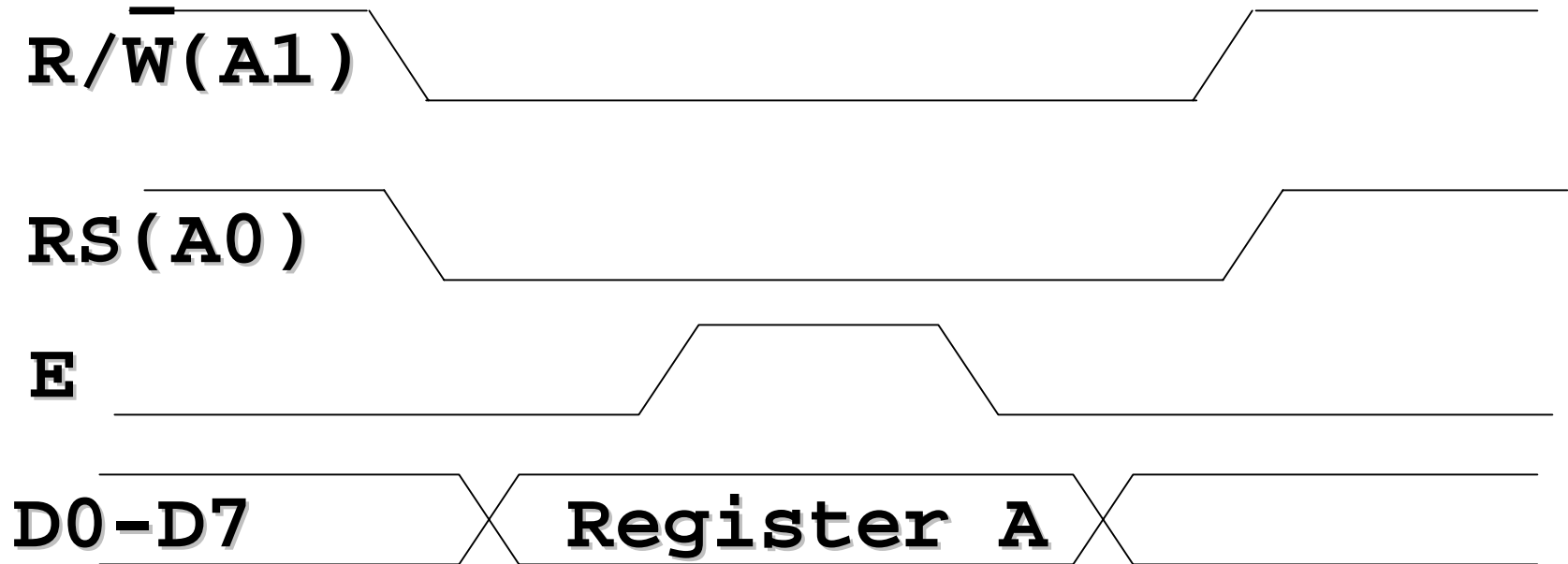
# LCD Programming: command write

---

```
; write command to LCD  
; LCD_command_write:  
    CALL LcdReady  
    MOV DPTR,#command_write  
    MOVX @DPTR,A  
  
    RET
```

# LCD Programming: command write

---



**A= command or location**

# LCD Programming: location

---

Command = clear screen, set  
location, cursor, set mode

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

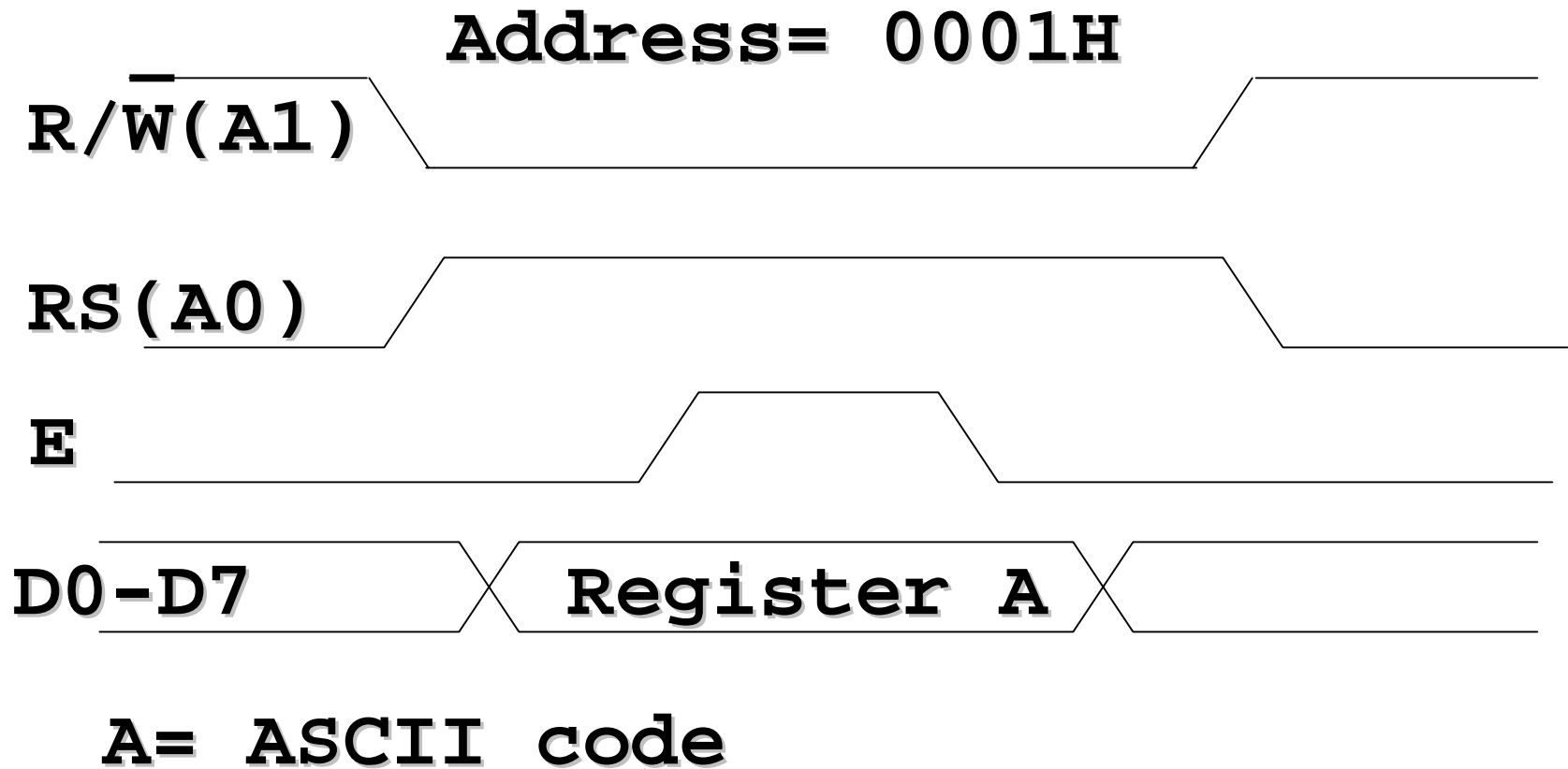
# LCD Programming: data write

---

```
; LCD_data_write: PUSH DPL  
    PUSH DPH  
    CALL LcdReady  
    MOV DPTR,#data_write  
    MOVX @DPTR,A  
    POP DPH  
    POP DPL  
    RET
```

# LCD Programming: data write

---



# LCD Programming: putch\_lcd

---

```
; write ASCII code to LCD at  
; current position  
; entry: A  
putch_lcd: CALL LcdReady  
           CALL LCD_data_write  
           RET
```

# LCD Programming: putch\_lcd

---

```
; send_string
```

```
; entry: DPTR
```

```
send_string: MOVX A,@DPTR
```

```
        CJNE A,#0,send_string1
```

```
        RET
```

```
send_string1:CALL LCD_data_write
```

```
        INC DPTR
```

```
        JMP send_string
```

# LCD Programming: init LCD

---

```
InitLcd:  MOV A,#38H
          CALL LCD_command_write
          MOV A,#0CH
          CALL LCD_command_write
          CALL clr_screen
          MOV A,#00H   ; A = Y
          MOV B,#00H   ; B = X
          CALL goto_xy
          RET
```

---

# LCD Programming: lcd.asm

---

```
main:    call initlcd
         mov  dptr,#hello
         call send_string
         mov  a,#1
         mov  b,#0
         call goto_xy
         mov  dptr,#sawasdee
         call send_string
         jmp  monitor
```

---

# LCD Programming: init LCD

---

hello: DB '8051SBC V1.0',0

sawasdee: DB 'Sawasdee people',0



8051SBC V1.0  
Sawasdee people

Download lcd driver and  
sample program from my  
class and lab page