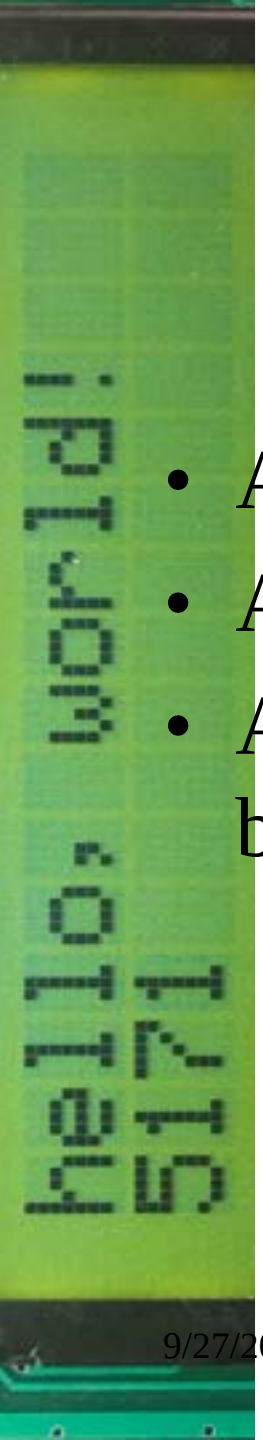
1DT301, Computer Technology

- Index registers
- Addressing modes
- Introduction to lab 5

INDEX REGISTERS



Memory

- A memory location has an address
- An address is composed of 2 bytes (16 bits)
- A location’s content has a size of 1 byte (8 bits)

Index Registers

- Refers to a memory location
- Named X, Y and Z
- 16 bits (2 bytes)
  - X → r27:r26, Y→ r29:r28 Z→r31:r30
  - XH:XL, YH;YL, ZH:ZL
- Compare to the concept of pointers in the C language.

Uses

- Referring to a memory location.
- Accessing data in lists or sequences.
- Some instructions
  - LD, ST
  - ADIW, SBIW, MUL

Example

```
.equ AMemoryLocation = 0x0060

; Setup so that X refers to AMemoryLocation
ldi XH, HIGH(AMemoryLocation)
ldi XL, LOW(AMemoryLocation)

ld r2, X ; Whatever X refers to load it into r2
st X, r3 ; Store content of r3 to the memory location X refers to.
```

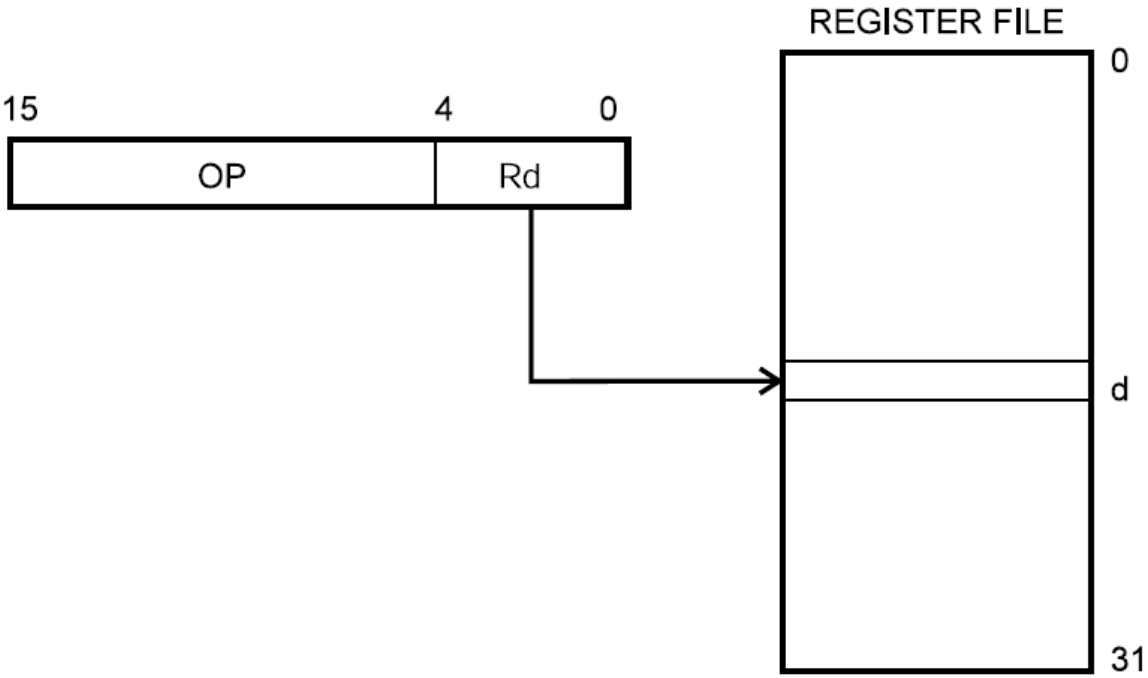
(Remember this slide when we are talking about pointers in C!)

ADRESSING MODES

Register Direct, Single Register Rd

Register Direct, Single Register Rd

Figure 1. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Single Register Rd:

COM – One's Complement

Description:

This instruction performs a One's Complement of register Rd.

Operation:

(i) Rd ← $FF - Rd

	Syntax:	Operands:	Program Counter:
(i)	COM Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0000

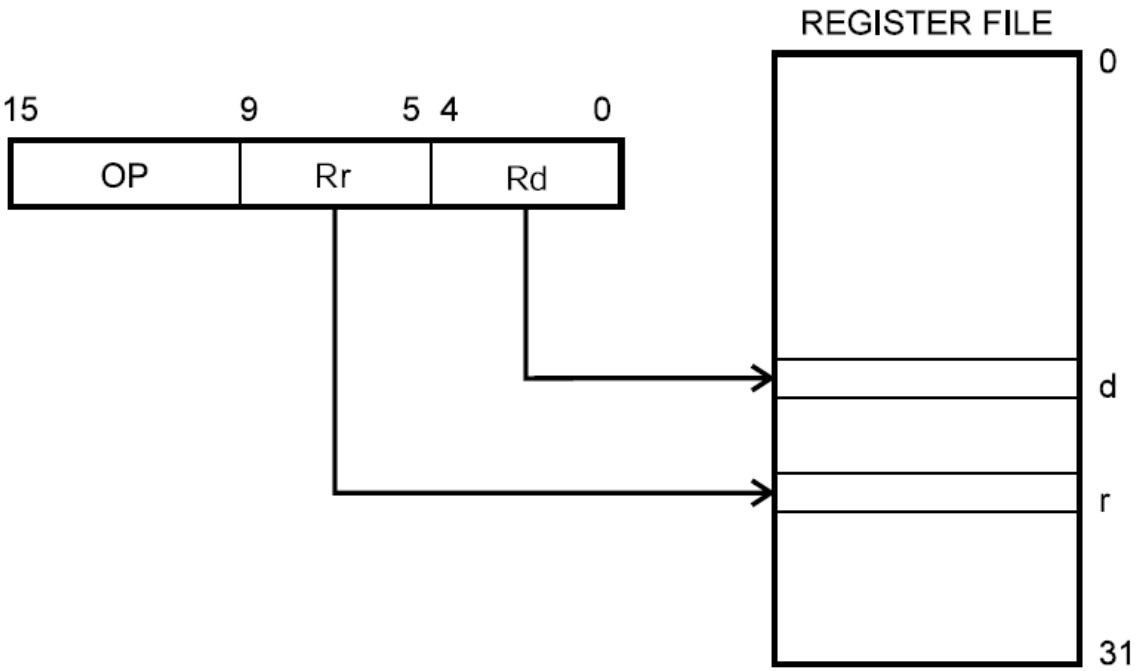
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	1

Register Direct, Two Registers Rd and Rr

Register Direct, Two Registers Rd and Rr

Figure 2. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

Register Direct, Two Registers Rd and Rr:

ADD – Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd + Rr

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

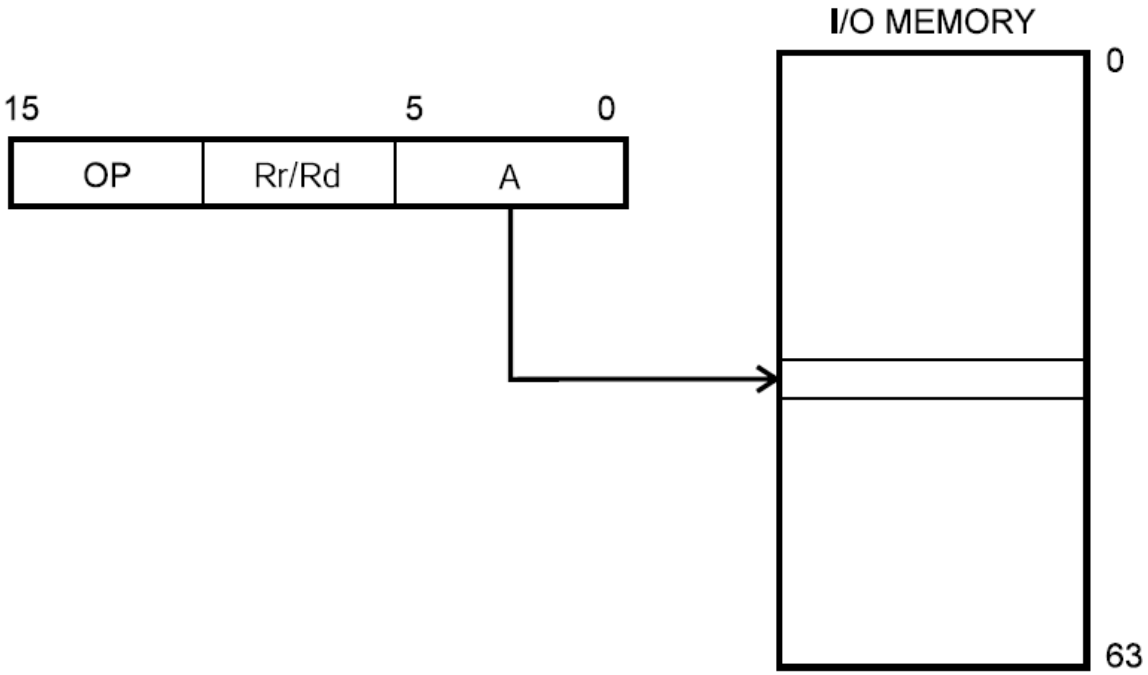
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

I/O Direct Addressing

I/O Direct

Figure 3. I/O Direct Addressing



I/O Direct Addressing.

OUT – Store Register to I/O Location

Description:

Stores data from register Rr in the Register File to I/O Space (Ports, Timers, Configuration Registers etc.).

Operation:

(i) I/O(A) ← Rr

	Syntax:	Operands:	Program Counter:
(i)	OUT A,Rr	$0 \le r \le 31, 0 \le A \le 63$	PC ← PC + 1

16-bit Opcode:

1011	1AAr	rrrr	AAAA

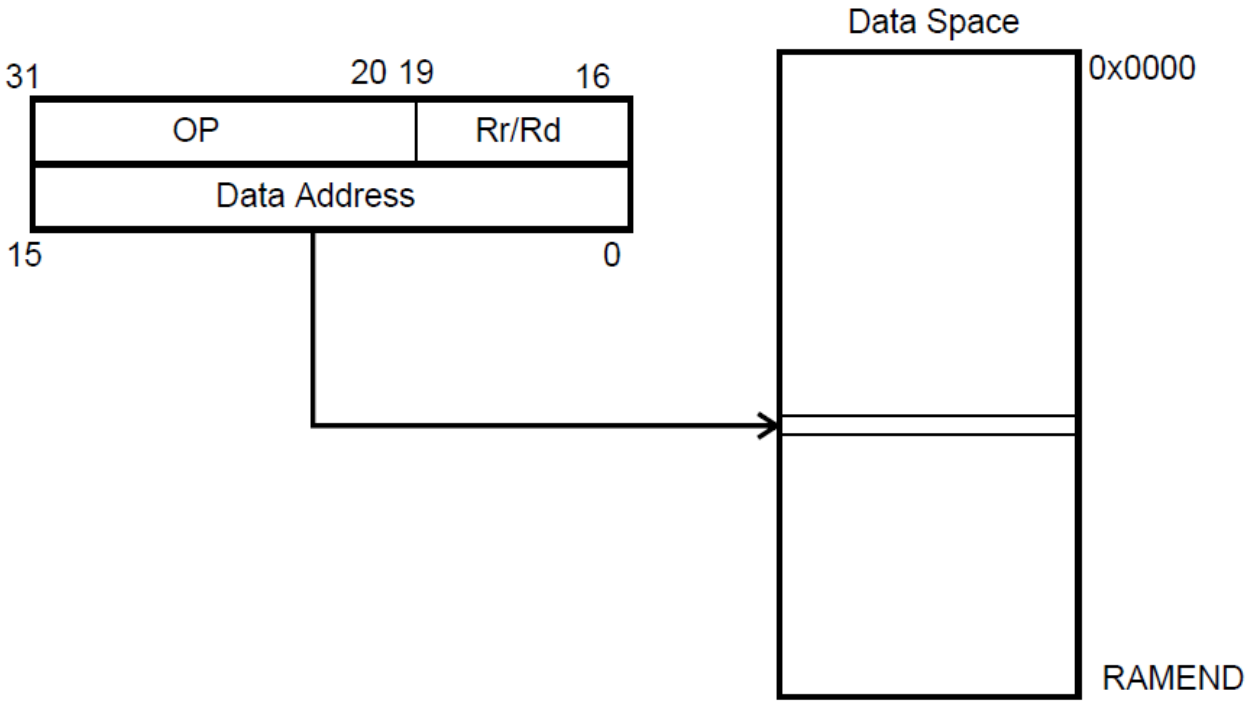
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Direct Data Addressing

Data Direct

Figure 4. Direct Data Addressing



Direct Data Addressing:

STS – Store Direct to Data Space

Description:

Stores one byte from a Register to the data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64K bytes. The STS instruction uses the RAMPD Register to access memory above 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPD in register in the I/O area has to be changed.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

Operation:

(i) (k) ← Rr

	Syntax:	Operands:	Program Counter:
(i)	STS k,Rr	$0 \le r \le 31, 0 \le k \le 65535$	PC ← PC + 2

32-bit Opcode:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

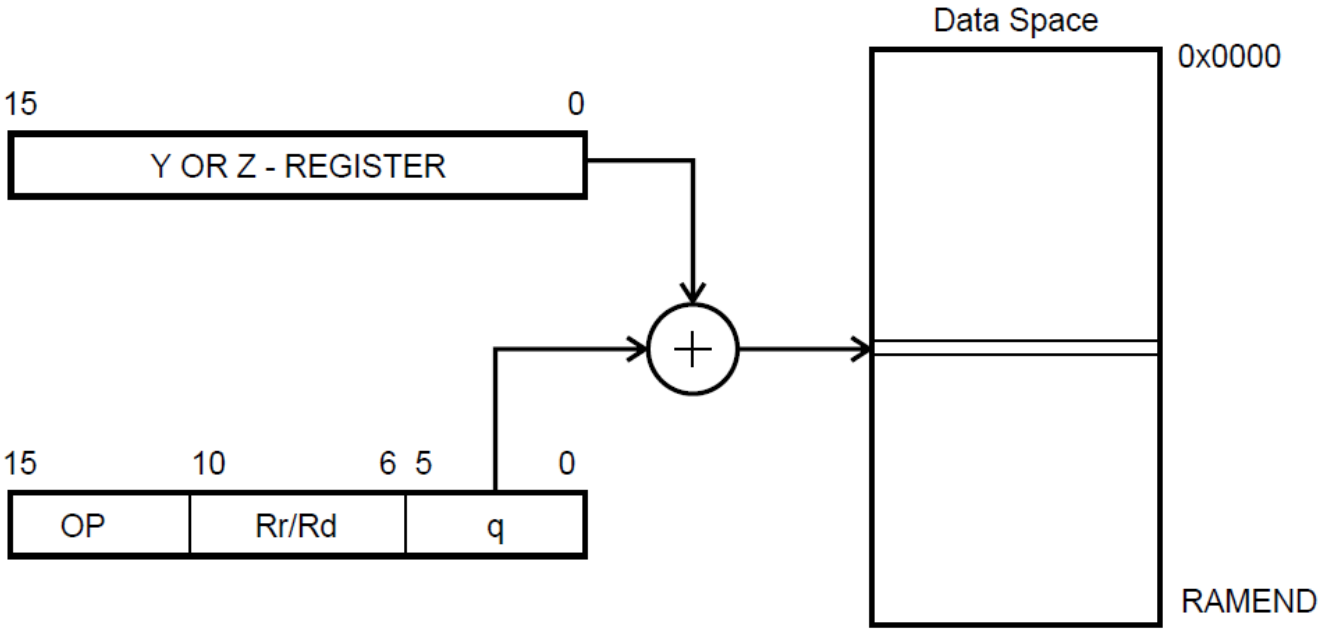
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Data Indirect with Displacement

Data Indirect with Displacement

Figure 5. Data Indirect with Displacement



Data Indirect with Displacement

ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

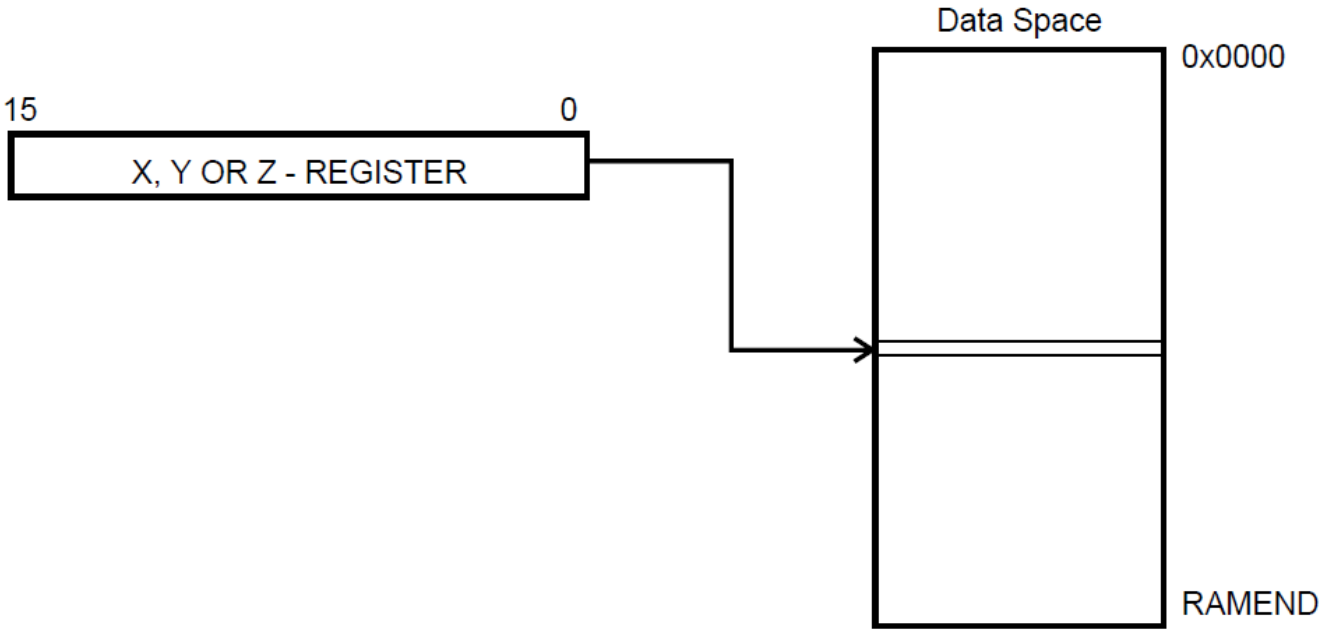
Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

(iiii)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	$PC \leftarrow PC + 1$

Data Indirect

Data Indirect

Figure 6. Data Indirect Addressing



Data Indirect

ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

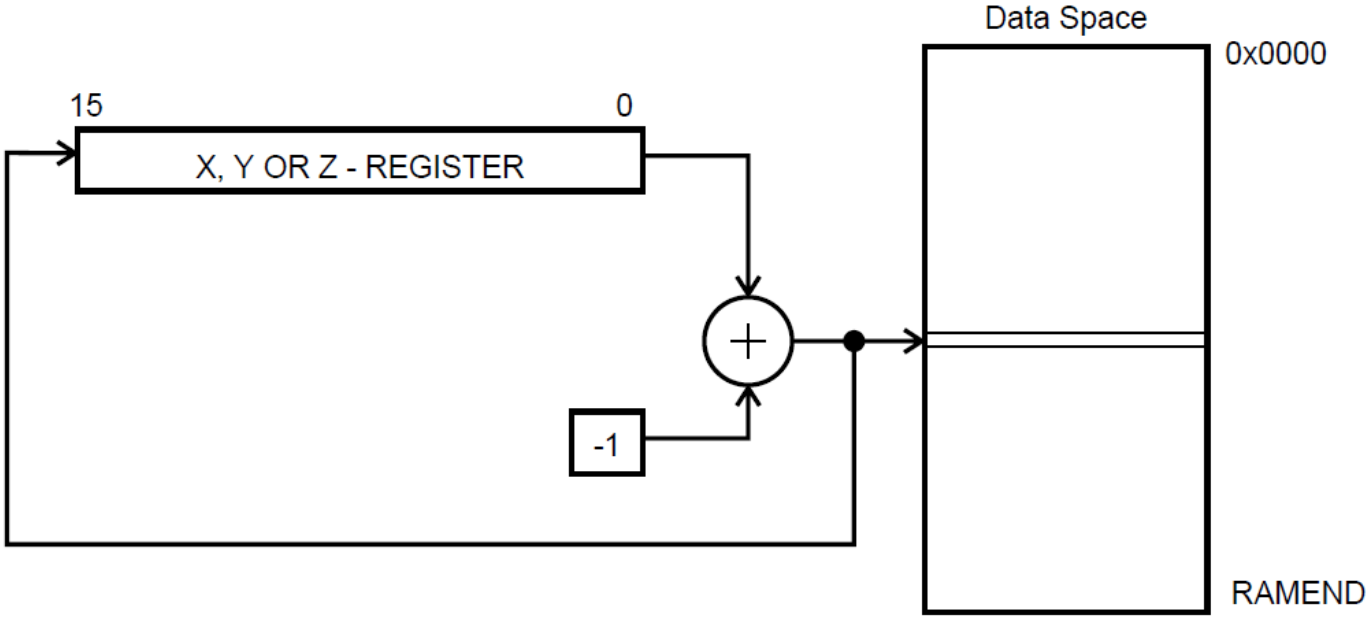
Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1

Data Indirect with Pre-decrement

Data Indirect with Pre-decrement

Figure 7. Data Indirect Addressing with Pre-decrement



Data Indirect with Pre-decrement

ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

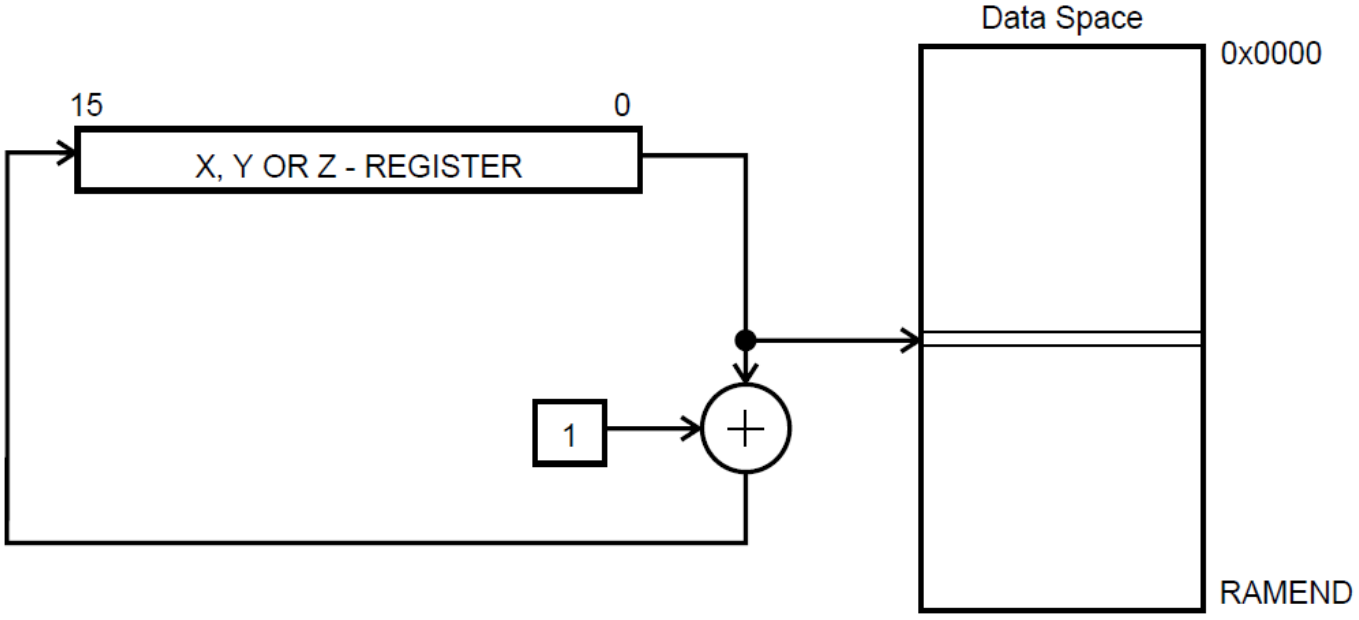
Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

	Syntax:	Operands:	Program Counter:
(iii)	ST -Z, Rr	$0 \le r \le 31$	$PC \leftarrow PC + 1$

Data Indirect with Post-increment

Data Indirect with Post-increment

Figure 8. Data Indirect Addressing with Post-increment



Data Indirect with Pre-decrement

ST (STD) – Store Indirect From Register to Data Space using Index Z

Description:

Stores one byte indirect with or without displacement from a register to data space. For parts with SRAM, the data space consists of the Register File, I/O memory and internal SRAM (and external SRAM if applicable). For parts without SRAM, the data space consists of the Register File only. The EEPROM has a separate address space.

The data location is pointed to by the Z (16 bits) Pointer Register in the Register File. Memory access is limited to the current data segment of 64K bytes. To access another data segment in devices with more than 64K bytes data space, the RAMPZ in register in the I/O area has to be changed.

The Z-pointer Register can either be left unchanged by the operation, or it can be post-incremented or pre-decremented. These features are especially suited for Stack Pointer usage of the Z-pointer Register, however because the Z-pointer Register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y-pointer as a dedicated Stack Pointer. Note that only the low byte of the Z-pointer is updated in devices with no more than 256 bytes data space. For such devices, the high byte of the pointer is not used by this instruction and can be used for other purposes. The RAMPZ Register in the I/O area is updated in parts with more than 64K bytes data space or more than 64K bytes Program memory, and the increment/decrement/displacement is added to the entire 24-bit address on such devices.

Not all variants of this instruction is available in all devices. Refer to the device specific instruction set summary.

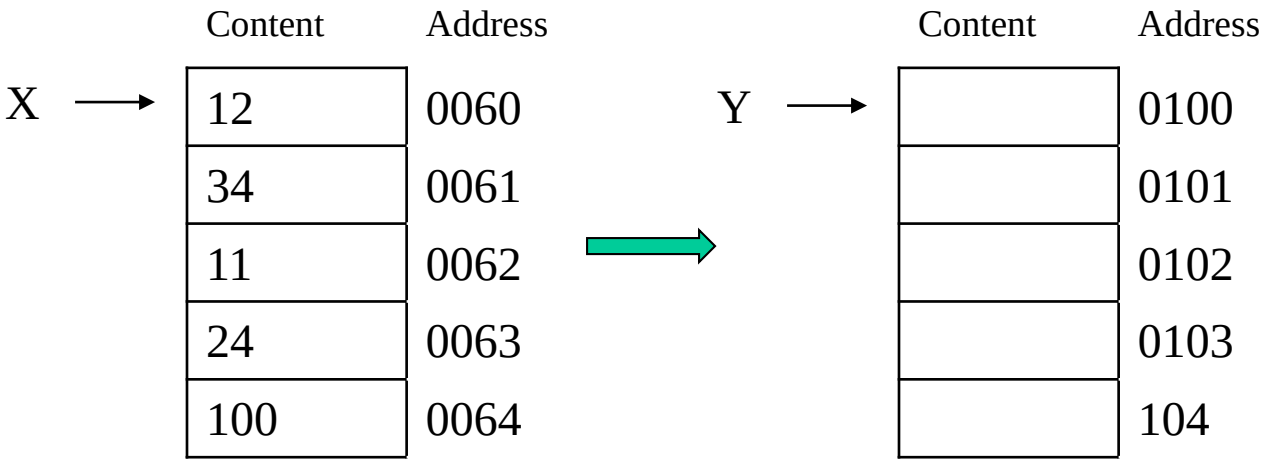
	Syntax:	Operands:	Program Counter:
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1

Example

Copy 10 bytes from

```
.equ SourceMemoryLocation = 0x0060
.equ DestinationMemoryLocation = 0x00100
.def Temp = r1
.def Counter = r16

; Setup so that X refers to SourceMemoryLocation
ldi XH, HIGH(SourceMemoryLocation)
ldi XL, LOW(SourceMemoryLocation)

; Setup so that Y refers to DestinationMemoryLocation
ldi YH, HIGH(DestinationMemoryLocation)
ldi YL, LOW(DestinationMemoryLocation)

ldiCounter, 10; The items to copy

; Copy from X to Y
CopyMemory:
    ld Temp, X+
    st Y+, Temp
    dec Counter
    brne CopyMemory
```

Example (cont.)

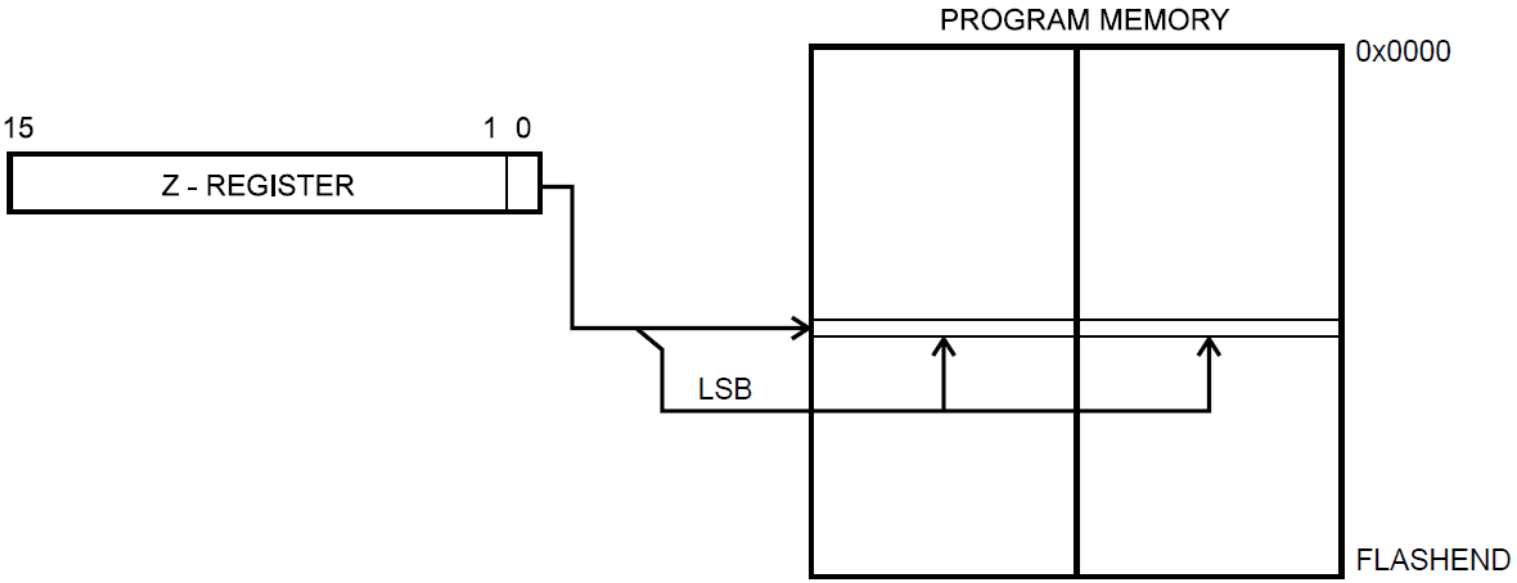



```
CopyMemory:
    ld Temp, X+
    st Y+, Temp
    dec Counter
    brne CopyMemory
```

Program memory Constant Addressing using the LPM, ELPM and SPM Instructions.

Program Memory Constant Addressing using the LPM, ELPM, and SPM Instructions

Figure 9. Program Memory Constant Addressing



Program memory Constant Addressing using the LPM, ELPM and SPM Instructions.

LPM – Load Program Memory

Description:

Loads one byte pointed to by the Z-register into the destination register Rd. This instruction features a 100% space effective constant initialization or constant data fetch. The Program memory is organized in 16-bit words while the Z-pointer is a byte address. Thus, the least significant bit of the Z-pointer selects either low byte (Z_{LSB} = 0) or high byte (Z_{LSB} = 1). This instruction can address the first 64K bytes (32K words) of Program memory. The Z-pointer Register can either be left unchanged by the operation, or it can be incremented. The incrementation does not apply to the RAMPZ Register.

Devices with Self-Programming capability can use the LPM instruction to read the Fuse and Lock bit values. Refer to the device documentation for a detailed description.

Not all variants of the LPM instruction are available in all devices. Refer to the device specific instruction set summary. The LPM instruction is not implemented at all in the AT90S1200 device.

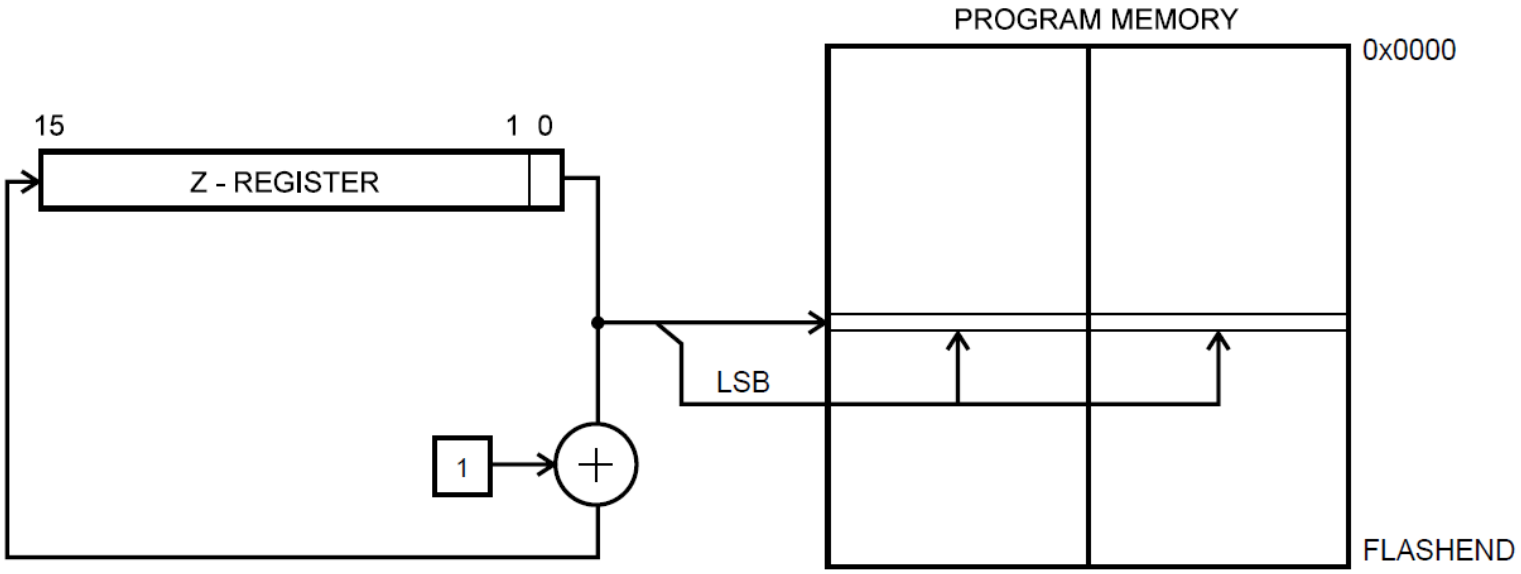
The result of these combinations is undefined:

	Syntax:	Operands:	Program Counter:
(i)	LPM	None, R0 implied	PC ← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Program memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction.

Program Memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction

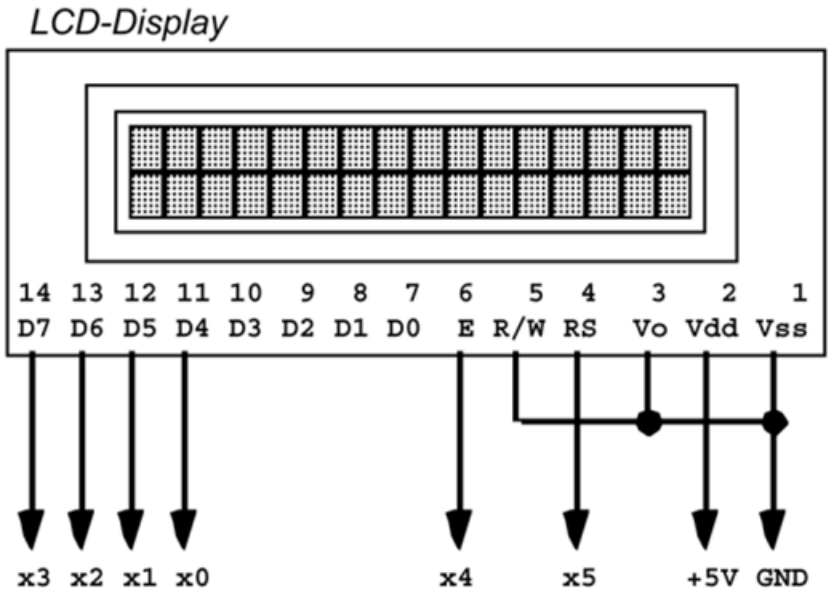
Figure 10. Program Memory Addressing with Post-increment



	Syntax:	Operands:	Program Counter:
(i)	LPM	None, R0 implied	PC ← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

LAB 5

Lab 5, LCD Display



Lab 5, LCD Display

Task 1: Write a program that displays a character on the display.

Write a program in Assembly that displays the character %. Look in the data sheet how to initiate the display. The data sheet you'll find on MyMoodle. The display will be connected as in the figure above. 4-bit-mode should be used, since only RS, E, D7, D6, D5 and D4 are connected to I/O-pins on the STK500 or STK600.
(The program *lab5_init_display.asm* gives you a good start…)

Lab 5, LCD Display

Task 2: Electronic bingo machine.

You should create an electronic bingo generator. The generator should create random numbers between 1 and 75. The numbers should be displayed on the display. Clear the display before a new value is displayed. Use interrupt and a pushbutton for the input.

Lab 5, LCD Display

Task 3: Serial communication and display.

Use program modules from lab 4 and write a program that receives a character on the serial port and displays each character on the display.

Lab 5, LCD Display

Task 4: Modify the program in task 3.

Modify the program in task 3 so 4 lines of text can be displayed. Each textline should be displayed during 5 seconds, after that the text on line 1 should be moved to line 2 and so on. The text should be entered from the terminal program, via the serial port.

JHD202A SERIES

■CHARACTERISTICS：

DISPLAY CONTENT：20 CHAR x 2ROW

CHAR. DOTS： 5 x 8

DRIVING MODE： 1/16D

AVAILABLE TYPES：

TN， STN(YELLOW GREEN、GREY、B/W)

REFLECTIVE、WITH EL OR LED BACKLIGHT

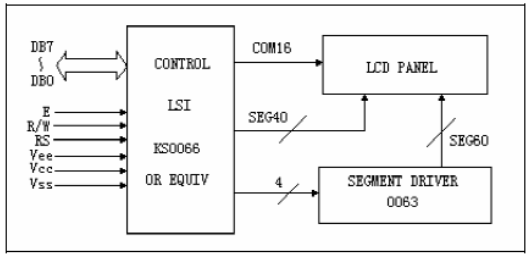
EL/100VAC， 400HZ

LED/4.2VDC

■PARAMETER （V_{DD}=5. 0V±10%, V_{SS}=0V, T_a=25℃）

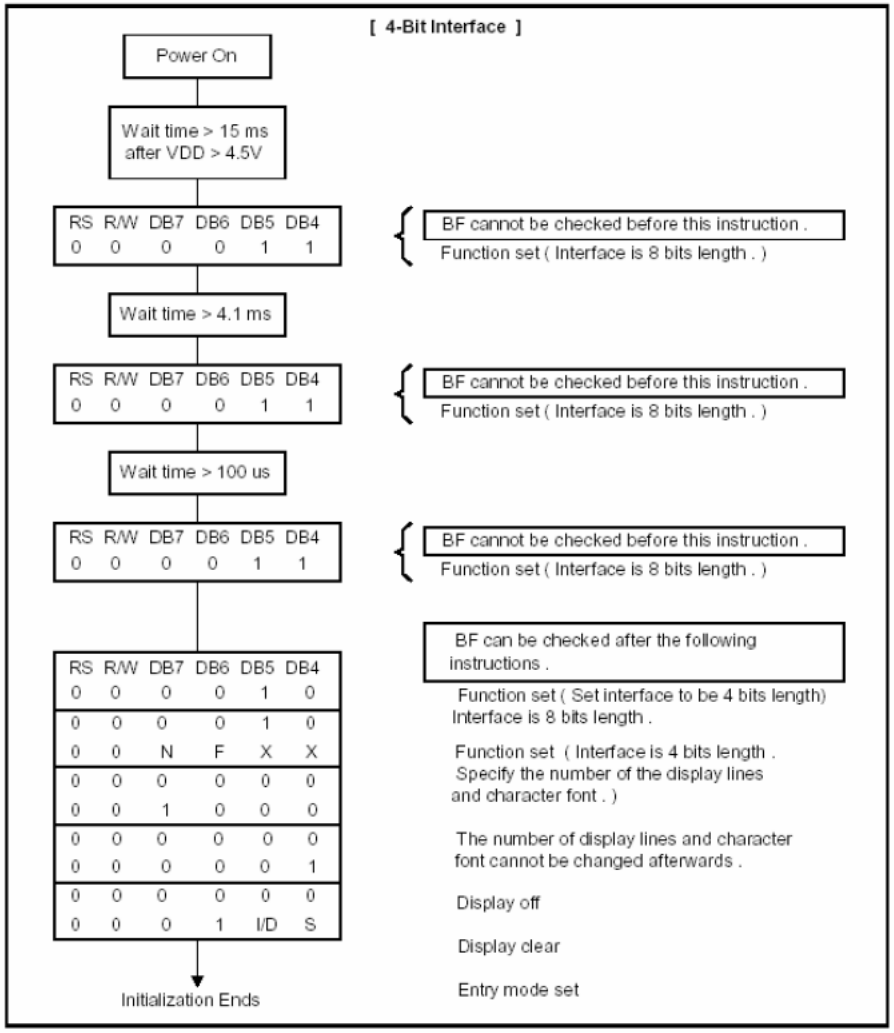
Parameter	Symbol	Testing Criteria	Standard Values			Unit
			Min.	Typ.	Max	
Supply voltage	V_{DD}-V_{SS}	-	4.5	5.0	5.5	V
Input high voltage	V_{IH}	-	2.2	-	V_{DD}	V
Input low voltage	V_{IL}	-	-0.3	-	0.6	V
Output high voltage	V_{OH}	-I_{OH}=02mA	2.4	-	-	V
Output low voltage	V_{OL}	I_{OL}=1.2mA	-	-	0.4	V
Operating voltage	I_{DD}	V_{DD}=5.0V	-	1.8	3.0	mA

■APPLICATION CIRCUIT



■DIMENSIONS/DISPLAY CONTENT

12. INITIALIZATION SEQUENCE



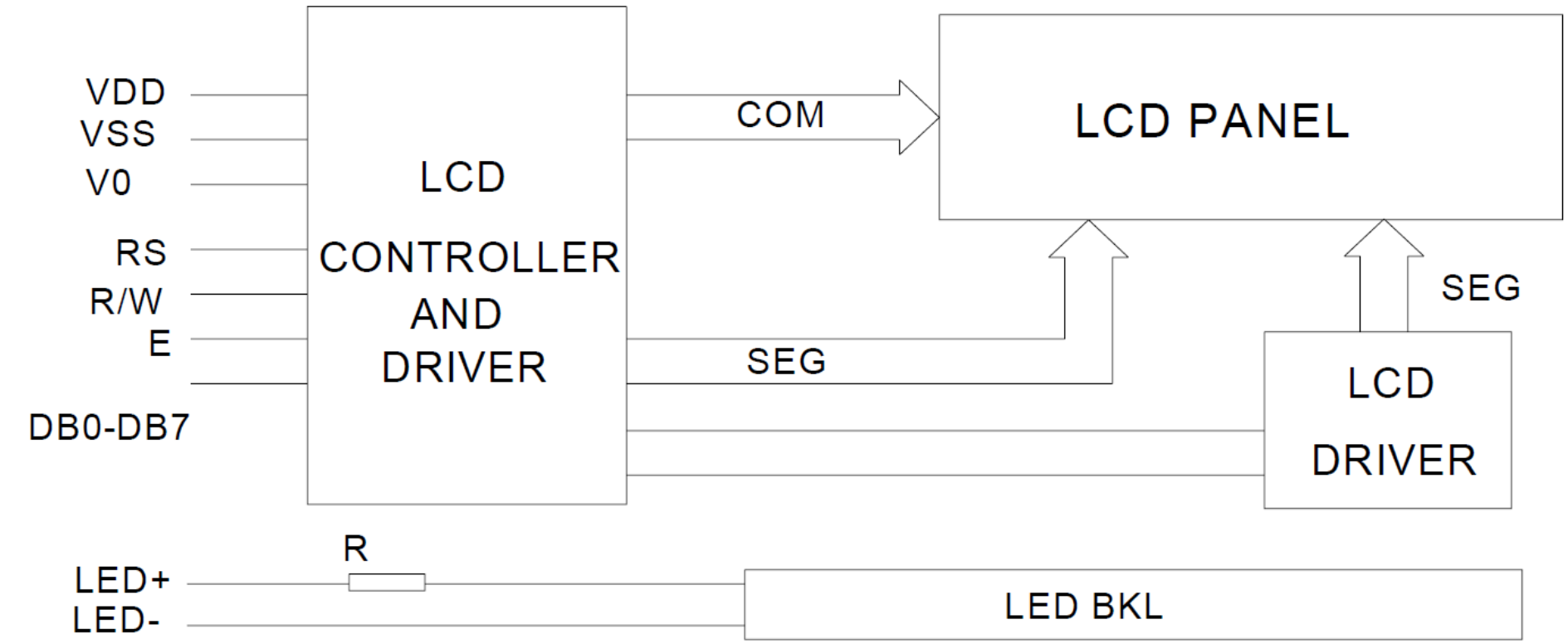
GDM1602A,

Interface Pin Description

Pin no.	Symbol	External connection	Function
1	V_{SS}	Power supply	Signal ground for LCM (GND)
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL (GND)

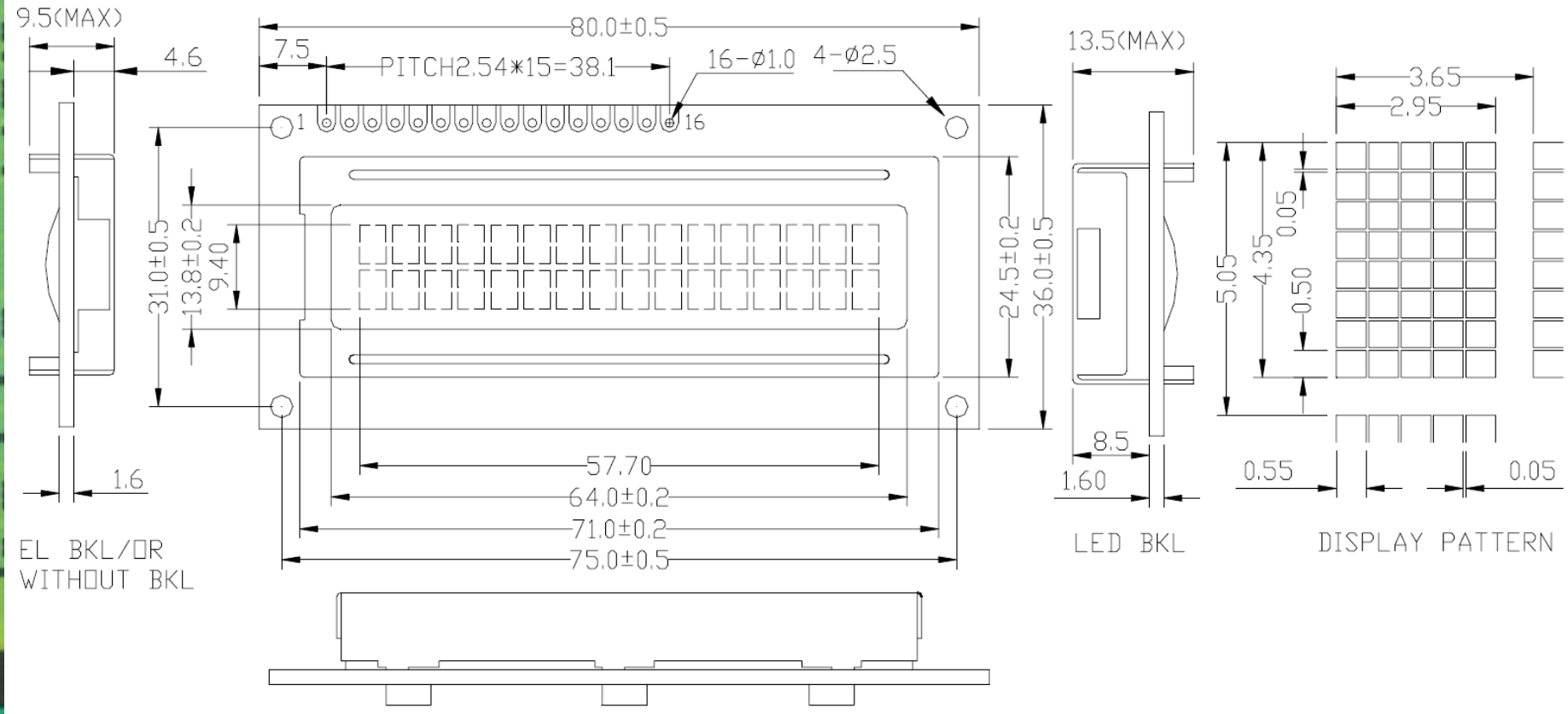
Block diagram

Block diagram

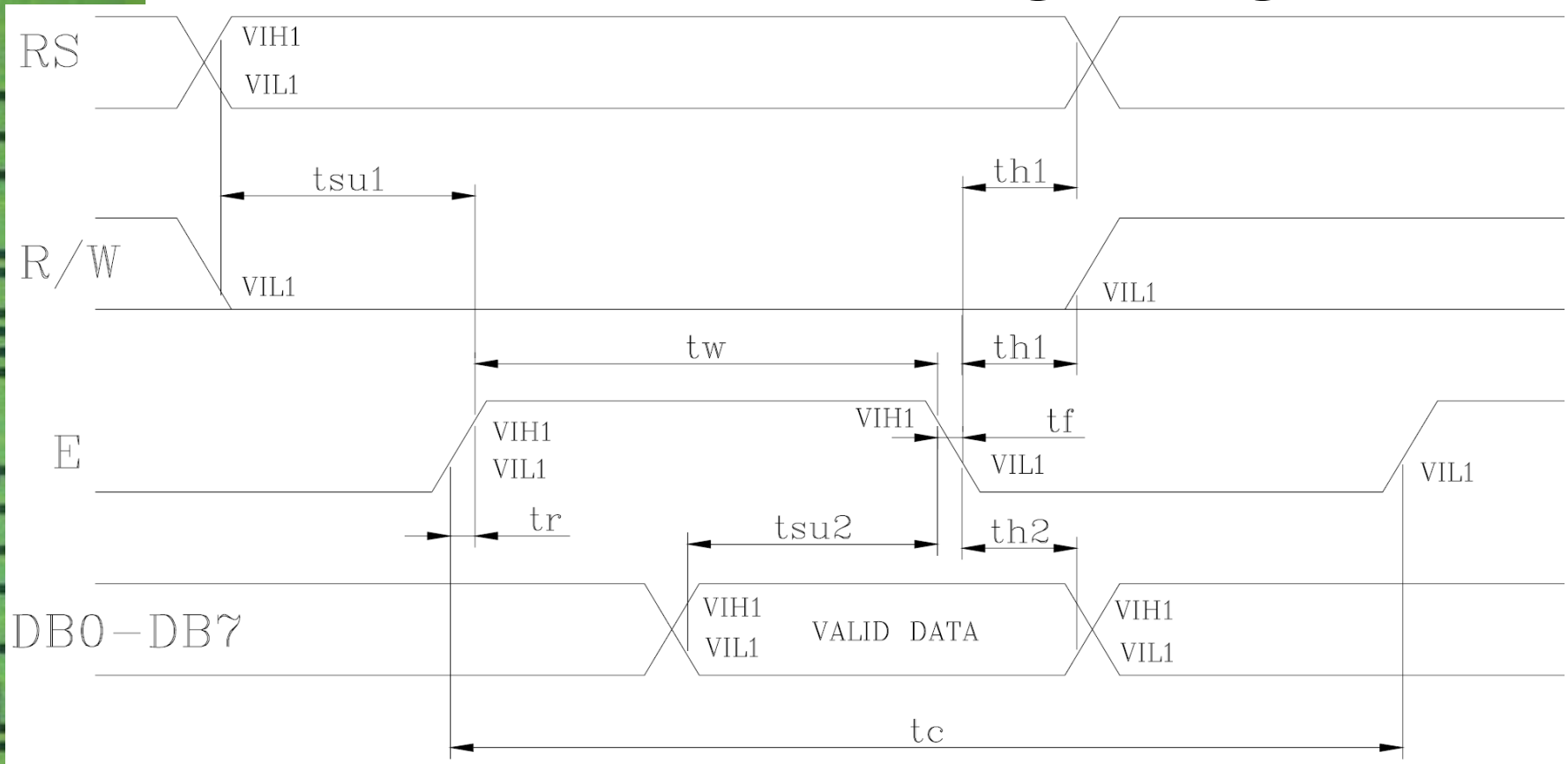


Physical dimension

Outline dimension



Write mode timing diagram



RS – Instruction/Data Register Selection.

RS = 0: Instruction Register, RS =1: Data Register

E = Enable Signal

Instruction table

Instruction	Instruction code										Description	Execution time (fosc= 270 KHZ
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRA and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us

Instruction table

Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Wikipedia:

Instruction set [edit]

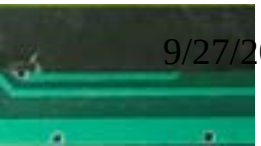
The HD44780 instruction set is shown below:[3]

HD44780U based instruction set

Instruction	Code										Description	Execution time (max) (when f_{cp} = 270 kHz)
	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.52 ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position. Also returns display being shifted to the original position. DDRAM content remains unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D); specifies to shift the display (S). These operations are performed during data read/write.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets on/off of all display (D), cursor on/off (C), and blink of cursor position character (B).	37 µs
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM content remains unchanged.	37 µs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data are sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data are sent and received after this setting.	37 µs
Read busy flag & address counter	0	1	BF	CGRAM/DDRAM address							Reads busy flag (BF) indicating internal operation being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0 µs
Write CGRAM or DDRAM	1	0	Write Data								Write data to CGRAM or DDRAM.	37 µs
Read from CG/DDRAM	1	1	Read Data								Read data from CGRAM or DDRAM.	37 µs

Instruction bit names —

I/D - 0 = decrement cursor position, 1 = increment cursor position; **S** - 0 = no display shift, 1 = display shift; **D** - 0 = display off, 1 = display on; **C** - 0 = cursor off, 1 = cursor on; **B** - 0 = cursor blink off, 1 = cursor blink on ; **S/C** - 0 = move cursor, 1 = shift display; **R/L** - 0 = shift left, 1 = shift right; **DL** - 0 = 4-bit interface, 1 = 8-bit interface; **N** - 0 = 1/8 or 1/11 duty (1 line), 1 = 1/16 duty (2 lines); **F** - 0 = 5×8 dots, 1 = 5×10 dots; **BF** - 0 = can accept instruction, 1 = internal operation in progress.



Write data to display

10) Write data to RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write binary 8-bit data to DDRAM/CGRAM.

The selection of RAM from DDRAM, and CGRAM, is set by the previous address set

C:\Document__Kurser\D2_E2\Datorteknik\HT_2016\Program_examples\Lab_5_2016\Lab_5_init_HT_2016.asm

```
;   1DT301
;   Lab 5, Initialize display JHD202A.
;
;   Date: 2016-09-30
;   Author, Anders Haggren
;   Modified:
;
;   Function
;   --------
;   Initialize display JHD202 connected to PORTE
;
;   (run @ 1.8432 MHz clk frequency)
;

.include    "m2560def.inc"
.def    Temp    = r16
.def    Data    = r17
.def    RS  = r18

.equ    BITMODE4    = 0b00000010        ; 4-bit operation
.equ    CLEAR   = 0b00000001            ; Clear display
.equ    DISPCTRL    = 0b00001111        ; Display on, cursor on, blink on.

.cseg
.org    0x0000              ; Reset vector
    jmp reset

.org    0x0072

reset:

    ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
    out SPH, Temp           ; sph = Temp
    ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
    out SPL, Temp           ; spl = Temp

    ser Temp                ; r16 = 0b11111111
    out DDRE, Temp          ; port E = outputs ( Display JHD202A)
    clr Temp                ; r16 = 0
    out PORTE, Temp

loop:   nop
    rjmp loop           ; loop forever

; **
; ** init_display
; **
init_disp:
    rcall power_up_wait     ; wait for display to power up
```

C:\Document__Kurser\D2_E2\Datorteknik\HT_2016\Program_examples\Lab_5_2016\Lab_5_init_HT_2016.asm

```
; **
; ** init_display
; **
init_disp:
    rcall power_up_wait      ; wait for display to power up

    ldi Data, BITMODE4       ; 4-bit operation
    rcall write_nibble       ; (in 8-bit mode)
    rcall short_wait         ; wait min. 39 us
    ldi Data, DISPCTRL       ; disp. on, blink on, curs. On
    rcall write_cmd          ; send command
    rcall short_wait         ; wait min. 39 us
clr_disp:
    ldi Data, CLEAR          ; clr display
    rcall write_cmd          ; send command
    rcall long_wait          ; wait min. 1.53 ms
    ret

; **
; ** write char/command
; **

write_char:
    ldi RS, 0b00100000       ; RS = high
    rjmp write
write_cmd:
    clr RS                   ; RS = low
write:
    mov Temp, Data           ; copy Data
    andi Data, 0b11110000    ; mask out high nibble
    swap Data                ; swap nibbles
    or Data, RS              ; add register select
    rcall write_nibble       ; send high nibble
    mov Data, Temp           ; restore Data
    andi Data, 0b00001111    ; mask out low nibble
    or Data, RS              ; add register select

write_nibble:
    rcall switch_output      ; Modify for display JHD202A, port E
    nop                      ; wait 542nS
    sbi PORTE, 5             ; enable high, JHD202A
    nop
    nop                      ; wait 542nS
    cbi PORTE, 5             ; enable low, JHD202A
    nop
    nop                      ; wait 542nS
    ret

; **
; ** busy_wait loop
```

C:\Document__Kurser\D2_E2\Datorteknik\HT_2016\Program_examples\Lab_5_2016\Lab_5_init_HT_2016.asm

```
    nop                         ; wait 542nS
    ret

; **
; ** busy_wait loop
; **
short_wait:
    clr zh                      ; approx 50 us
    ldi zl, 30
    rjmp wait_loop
long_wait:
    ldi zh, HIGH(1000)          ; approx 2 ms
    ldi zl, LOW(1000)
    rjmp wait_loop
dbnc_wait:
    ldi zh, HIGH(4600)          ; approx 10 ms
    ldi zl, LOW(4600)
    rjmp wait_loop
power_up_wait:
    ldi zh, HIGH(9000)          ; approx 20 ms
    ldi zl, LOW(9000)

wait_loop:
    sbiw z, 1                   ; 2 cycles
    brne wait_loop              ; 2 cycles
    ret

; **
; ** modify output signal to fit LCD JHD202A, connected to port E
; **

switch_output:
    push Temp
    clr Temp
    sbrc Data, 0                ; D4 = 1?
    ori Temp, 0b00000100        ; Set pin 2
    sbrc Data, 1                ; D5 = 1?
    ori Temp, 0b00001000        ; Set pin 3
    sbrc Data, 2                ; D6 = 1?
    ori Temp, 0b00000001        ; Set pin 0
    sbrc Data, 3                ; D7 = 1?
    ori Temp, 0b00000010        ; Set pin 1
    sbrc Data, 4                ; E = 1?
    ori Temp, 0b00100000        ; Set pin 5
    sbrc Data, 5                ; RS = 1?
    ori Temp, 0b10000000        ; Set pin 7 (wrong in previous version)
    out porte, Temp
    pop Temp
    ret
```