X36SKD

lab

Display

Miroslav Skrbek, Josef Hlaváč

Literature

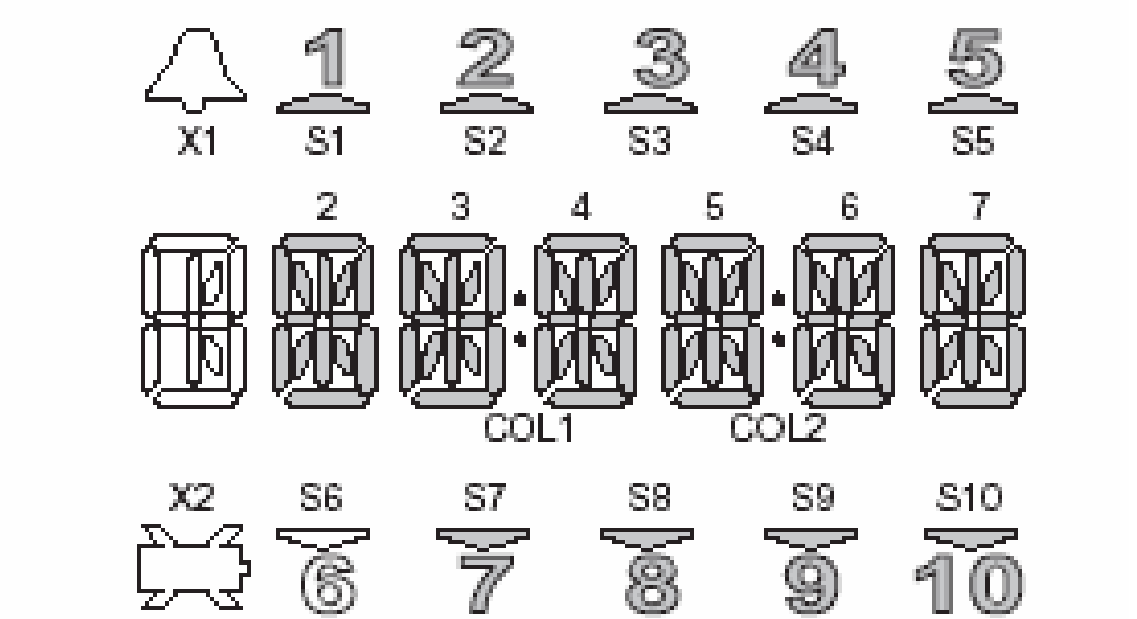
[1] *8-bit AVR® Instruction Set*, Technical literature of ATMEL Corporation, 2002. http://www.atmel.com

[2] *8-bit AVR® Microcontroller width 16K Bytes In-System Programmable Flash ATmega169*. Datasheet. Technical literature of ATMEL Corporation, 2003. http://www.atmel.com

[3] *AVR Assembler User Guide*. Technical literature of ATMEL Corporation, 2002. http://www.atmel.com

[4] *AVR065: LCD Driver for the STK502 and AVR Butterfly.* Application note. Technical literature of ATMEL Corporation, 2004. http://www.atmel.com

Display



Only the segments shown in gray can be controlled.

Display initialization

Display init:

```
init_disp:
  ldi r16, 0xB7
  sts LCDCRB, r16
  ldi r16, 0x10
  sts LCDFRR, r16
  ldi r16, 0x0F
  sts LCDCCR, r16
  ldi r16, 0x80
  sts LCDCRA, r16
  ret
```

This sequence must be executed at the beginning of every program that uses the display.

The init sequence is necessary to set up the display controller. The constants written to registers are given by the design of the controller and the display.

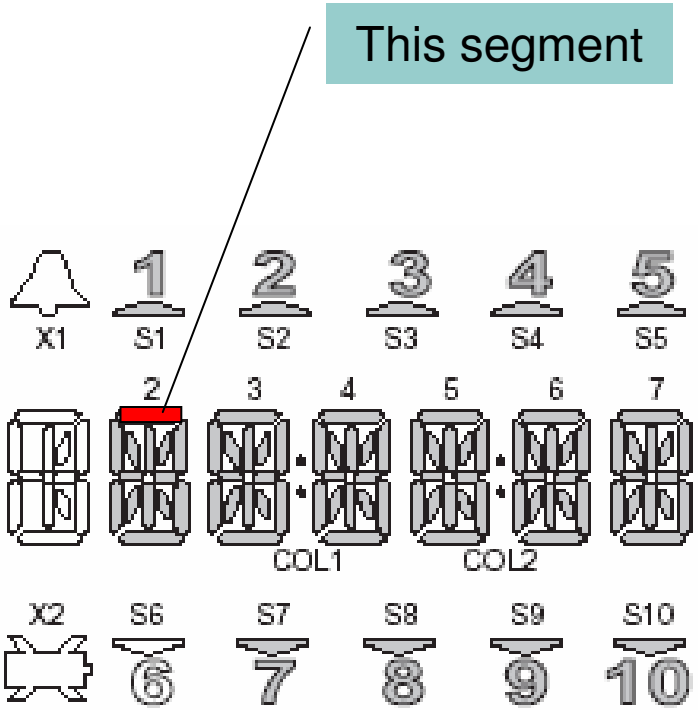
The LCD controller function is outside of the scope of this course. If interested, study the recommended literature.

Controlling the segments

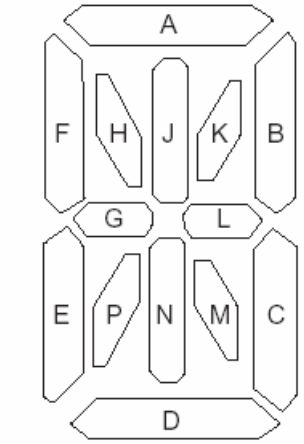
Each segment corresponds to one bit in the display data registers.

E.g. writing a ‘1’ to bit 4 of LCDDR0 causes the segment A of the second digit to light up (=turn black).

```
.include "m169def.inc"

          .org 0x0000
          ldi      r16, low(RAMEND)
          out      SPL, r16
          ldi      r16, high(RAMEND)
          out      SPH, r16

          call init_disp
          ldi      r16, 0x01
          sts      LCDDR0, r16
end:      jmp end
```



Display



Register	Digit	Address
LCDDR0	2(L) and 3(H)	ECh
LCDDR1	4(L) and 5(H)	EDh
LCDDR2	6(L) and 7(H)	EEh
LCDDR3		EFh

	H				L			
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDDRx	K	-	-	A	K	-	-	A
LCDDRx+5	J	F	H	B	J	F	H	B
LCDDRx+10	L	E	G	C	L	E	G	C
LCDDRx+15	M	P	N	D	M	P	N	D

Task: Write a subroutine that clears the entire display

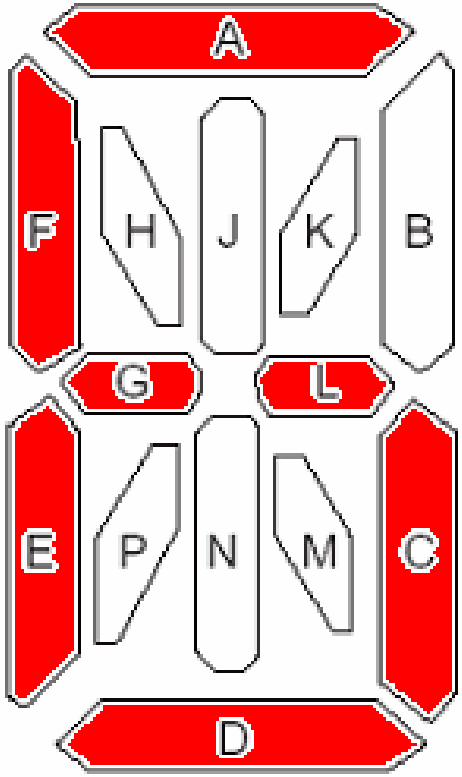
Hint: Clear (=store a zero into) all LCDDRx registers.

Solution

```
.include "m169def.inc"

        .org 0x0000
        ldi    r16, 0x00
        out    SPL, r16
        ldi    r16, 0x04
        out    SPH, r16
        call init_disp
        call clear_disp
end:    jmp end

clear_disp:

  ldi  r16, 0x00
  sts  LCDDR0   , r16
  sts  LCDDR0+5 , r16
  sts  LCDDR0+10, r16
  sts  LCDDR0+15, r16
```

```
  sts    LCDDR1   , r16
  sts    LCDDR1+5 , r16
  sts    LCDDR1+10, r16
  sts    LCDDR1+15, r16
  sts    LCDDR2   , r16
  sts    LCDDR2+5 , r16
  sts    LCDDR2+10, r16
  sts    LCDDR2+15, r16
  sts    LCDDR3   , r16
  sts    LCDDR3+5 , r16
  sts    LCDDR3+10, r16
  sts    LCDDR3+15, r16
  ret
```

Task: Write a subroutine that displays “6” at position 2.



We’ll write to the following registers:

LCDDR0
LCDDR0+5
LCDDR0+10
LCDDR0+15

Solution

```
.include "m169def.inc"

        .org 0x000
        ldi     r16, 0x00
        out     SPL, r16
        ldi     r16, 0x04
        out     SPH, r16
        call    init_disp
        call show_6
end:    jmp end
```

```
show_6:
    ldi r16, 0x01
    sts LCDDR0, r16
    ldi r16, 0x04
    sts LCDDR0+5, r16
    ldi r16, 0x0F
    sts LCDDR0+10, r16
    ldi r16, 0x01
    sts LCDDR0+15, r16
    ret

init_disp:
...
```

Task: Write a subroutine that can display any character at any display position.

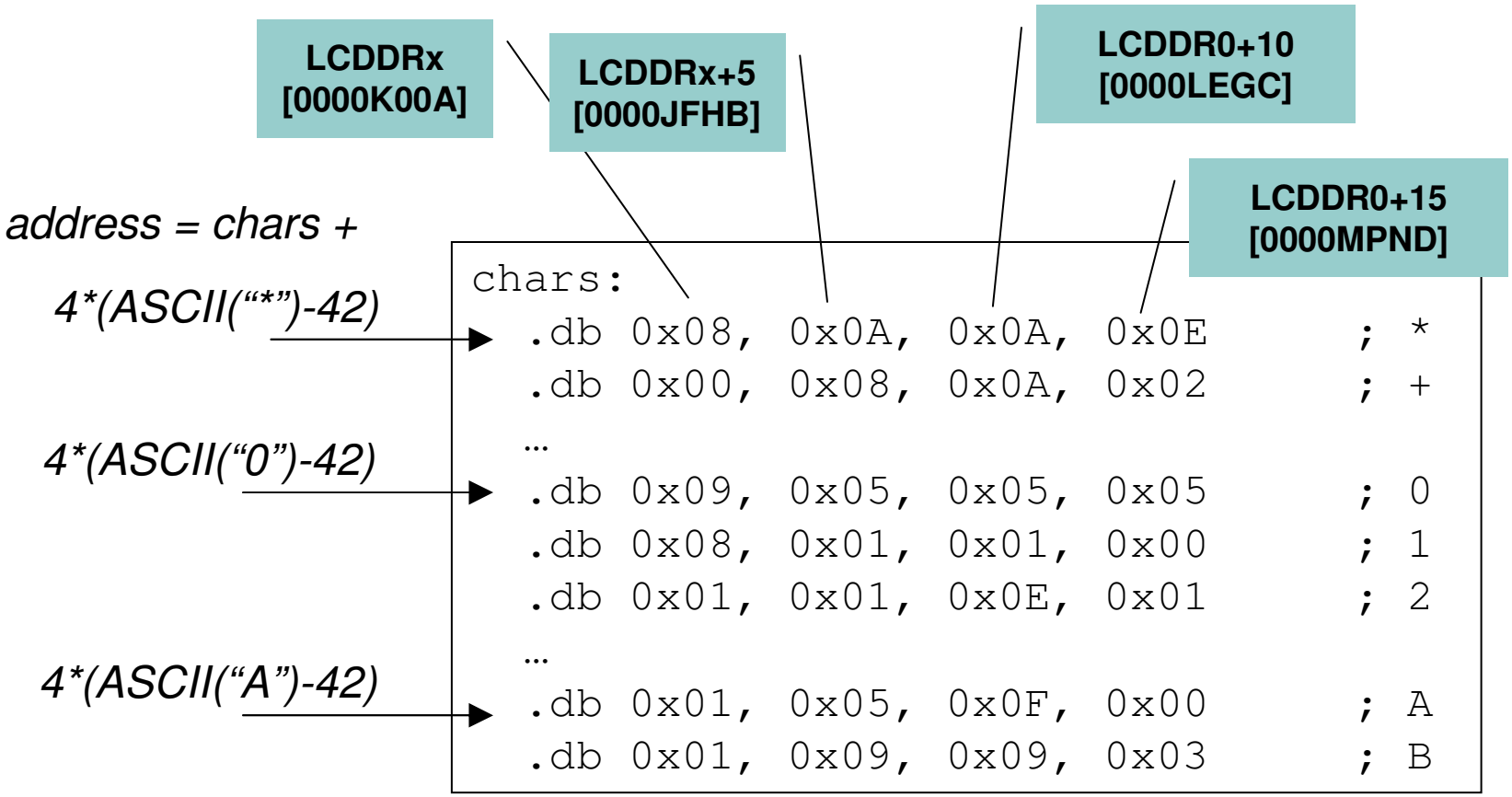
- Familiarize yourself with the format of the character shape table.
- Write a program portion that retrieves character shape by its ASCII code.
- Write a program portion that determines the base address of the display registers according to the requested position.
- Put all portions together into a single subroutine (`show_char`).

Example call

```
        .include "m169def.inc"

        .org 0x0000

start:  ldi     r16, low(RAMEND)
        out     SPL, r16
        ldi     r16, high(RAMEND)
        out     SPH, r16
        call    init_disp


        ldi r16, 'A'
        ldi r17, 2          ; position
        call show_char
end:    jmp end

show_char:

… write instructions here …
```

Note: Position 2 is the leftmost one, position 7 the rightmost one.

Character shape table



Task: Write a subroutine that displays a given decimal number (range 0-255) on the display.

```
        .include "m169def.inc"

        .org 0x0000

start:  ldi    r16, 0x00
        out    SPL, r16
        ldi    r16, 0x04
        out    SPH, r16

        ldi r16, 123
        call show_dec
end:    jmp end

… write instructions here …
```

Reuse your "show_char" and division subroutines.