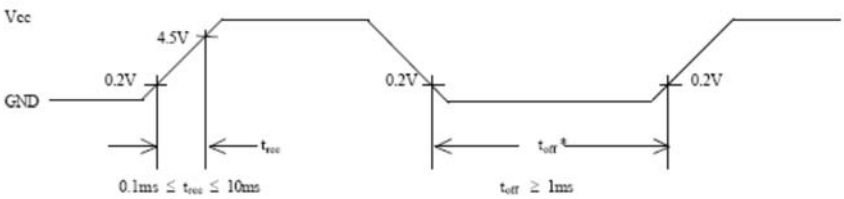
LCD Software

Prof. Ken Short

LCD Module Initialization Using Internal Circuitry

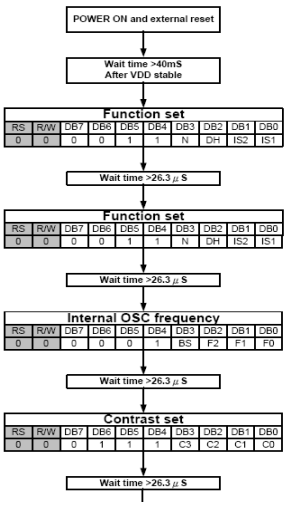
- LCD module must be initialized at power on
- The display can be initialized using the internal reset circuit if the Internal Power Supply Reset timing below is met



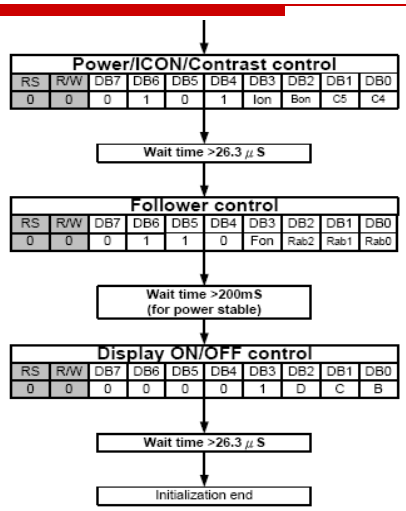
LCD Module Initialization Using Software

- Software initialization is preferred because result is not dependent on characteristics of power supply used
- Flowchart for initialization is provided in Optrex User's Manual (or HD44780 data sheet) or in the manual for the DOG driver IC ST7036
- Flowchart is different for 4-bit parallel, 8-bit parallel, SPI and I2C.

Serial Interface Initialization DOG Module



Serial Interface Initialization (cont.)



Discussion of Initialization Routine

- In the software for Module 1 initialization is done by the assembly language routines in the file lcd_dog_iar_driver.asm
- The display is connected to Port B and is operated in the SPI mode
- The Atmel AVR assembler and the AVR IAR Assembler use the same mnemonics for instructions but do not use the same assembler directives (see pages 9 – 12 of the AVR IAR Assembler Reference Guide)
- With the exception of differences in the directives the lcd_dog_iar_driver.asm code should already be understandable to you

Comparison of Assembler Directives

Atmel AVR Assembler format	AVR IAR Assembler format	Comments
label: .BYTE size	label: DS8 size	
.CSEG	RSEG segment name:CODE: segment flags	1
.DB data1,data2,data3	DB data1,data2,data3	
.DEF name = value	#define name value	2
.DSEG	RSEG segment name:DATA:segment flags	1
.DW data1,data2,data3	DW data1,data2,data3	
.ENDMACRO	ENDM	
.EQU label = expression	label EQU expression	
.ESEG	RSEG segment name:XDATA:segment flags	1
.INCLUDE file	#include file	2
.LIST	LSTOUT+	
.LISTMAC	LSTEXP+	
.MACRO macroname	macroname MACRO arguments…	3

Table 7: Migrating from Atmel AVR Assembler to AVR IAR Assembler

Comparison of Assembler Directives (cont.)

Atmel AVR Assembler format	AVR IAR Assembler format	Comments
.NOLIST	LSTOUT-	
.ORG expression	ORG expression	
.SET label = expression	label VAR expression	

Table 7: Migrating from Atmel AVR Assembler to AVR IAR Assembler (Continued)

Multimodule and Mixed Language Programs

- Large programs are comprised of multiple files which can be separately assembled or compiled
- Each file can contain one or more modules
- If some of the modules comprising a single program are written in C and others are written in assembler the result is a mixed language program
- The linker links together the relocatable object code from the separate compilations and assemblys

Multimodule Programs

- A function in a multimodule program may be a local function. Such a function can only be called from within the module where it is defined
- Other functions in a multimodule program may be public. Public functions can be called from any module where they have been declared
- Variables in a multimodule program may be local or public. Local variables can only be accessed from the module in which the memory has been allocated. Public variables may be accessed from any module where they have been declared

Functions in Module lcd in file lcd_dog_iar_driver.asm

Public functions – can be called from outside the module

- init_lcd_dog
- update_lcd_dog

Local functions – can only be called from within the module

- delay_30uS
- v_delay
- delay_40mS
- init_spi_lcd
- and many others

Public Memory in Module lcd in File lcd.asm

- Three public arrays are declared in module lcd
- dsp_buff_1, dsp_buff_2, and dsp_buff_3 are each 16 byte buffers that provide an image in the AVR of the characters to be written to the DDRAM of the LCD

Module and Public Directives

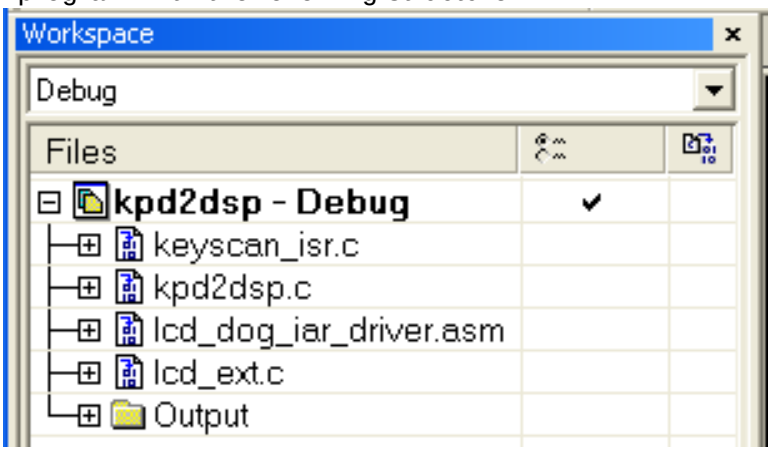
```
46 ; Module must have a name for the linker
47 NAME lcd
48
49 ;Include ATmega128 definitions
50 #include <iom128.h>
51
52 ; All functions used esternally must be declared public
53 PUBLIC init_lcd_dog
54 PUBLIC update_lcd_dog
55
56 PUBLIC dsp_buff_1
57 PUBLIC dsp_buff_2
58 PUBLIC dsp_buff_3
```


Segment and Memory Allocation Directives

```
69 ;***  DATA Segment  ************************************
70 ;Puts display buffers in near initialize to zero segment.
71 ;See chapter on segments in IAR reference guide
72 RSEG NEAR_Z
73 dsp_buff_1   DS8 16
74 dsp_buff_2   DS8 16
75 dsp_buff_3   DS8 16
76
77 ;***  CODE Segment Subroutines  ***************************
78 ;Puts object code in CODE segment.
79 RSEG CODE
80
81 ;************************
```


Structure of kpd2dsp Program

- The program kpd2dsp is a multimodule mixed language program with the following structure



Interface Between kpd2dsp.c and lcd_dog_iar_driver.asm

- Function main in kpd2dsp.c calls the function init_lcd_dog in lcd_dog_iar_driver.asm to initialize the display
- Function main writes character codes for characters it wishes to display into the memory buffers dsp_buff_1, dsp_buff_2, and dsp_buff_3
- To cause the characters in memory buffers dsp_buff_1, dsp_buff_2, and dsp_buff_3 to actually be displayed, function main calls update_lcd_dog

Mixed Language Program kpd2dsp

- kpd2dsp is a simple example of a mixed language program
- The C function kpd2dsp calls assembly language functions in the module lcd in lcd_dog_iar_driver.asm and writes memory locations in module lcd.
- No parameters are passed in the function calls. Parameter passing is the more interesting aspect of mixed language programming and will be discussed later in the course
- Also there are no calls from assembly code to C functions in this program

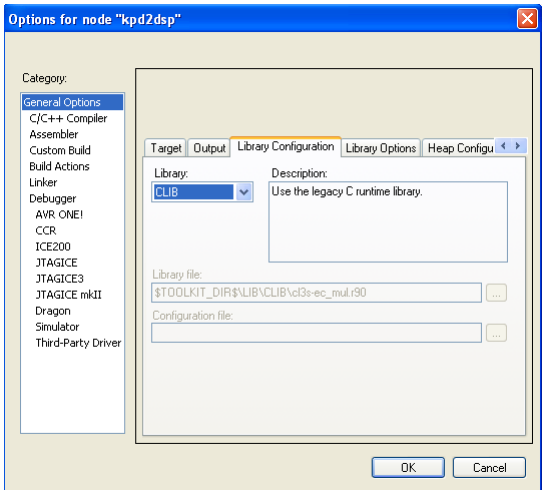
Subroutines Provided

- init_lcd_dog
  Initializes ST7036
- update_lcd_dog
  Transfers contents of three buffers to ST7036's DDRAM

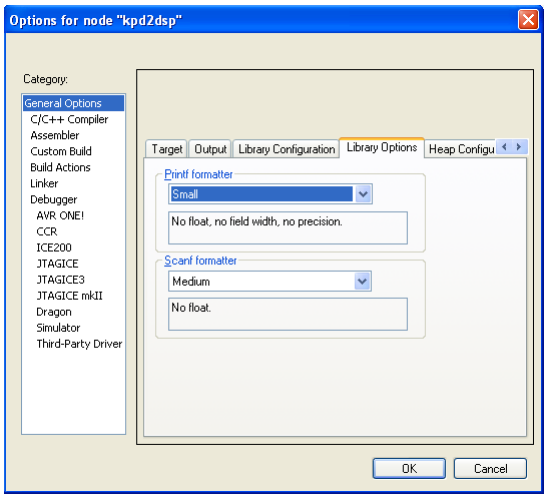
Data Buffers Created in Include File

```
69 ;***  DATA Segment  ***********************************
70 ;Puts display buffers in near initialize to zero segment.
71 ;See chapter on segments in IAR reference guide
72 RSEG NEAR_Z
73 dsp_buff_1   DS8 16
74 dsp_buff_2   DS8 16
75 dsp_buff_3   DS8 16
```

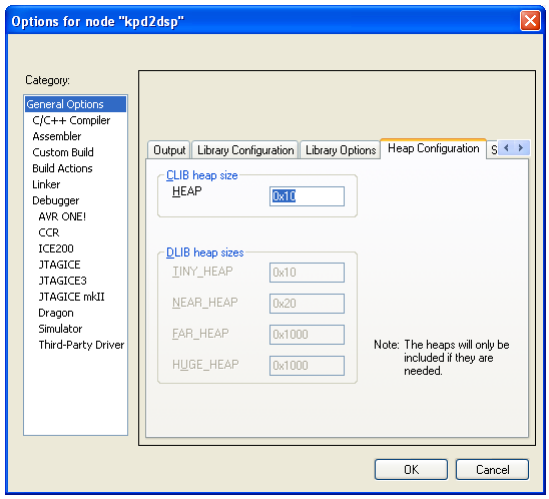

Library Selection



Printf Formatter Selection



Heap Allocation



Stack Allocation – CSTACK and RSTACK