CPE 323 Introduction to Embedded Computer Systems:
DMA Controller, LCD Controller

Instructor: Dr Aleksandar Milenkovic
Lecture Notes
Outline

MSP430: System Architecture
- DMA Controller
- LCD Controller
DMA Controller Introduction

Direct memory access (DMA) controller transfers data from one address to another without CPU intervention, across the entire address range.

- Move data from the ADC12 conversion memory to RAM
- Move data from RAM to DAC12

Devices that contain a DMA controller may have one, two, or three DMA channels available

Using the DMA controller

- Can increase the throughput of peripheral modules
- Can reduce system power consumption by allowing the CPU to remain in a low-power mode without having to awaken to move data to or from a peripheral
MSP430 DMA Features

- Up to three independent transfer channels
- Configurable DMA channel priorities
- Requires only two MCLK clock cycles per transfer
- Byte or word and mixed byte/word transfer capability
- Block sizes up to 65535 bytes or words
- Configurable transfer trigger selections
- Selectable edge or level-triggered transfer
- Four addressing modes
- Single, block, or burst-block transfer modes

- Configured from software
DMA Block Diagram
DMA Addressing Modes

- Configured with the DMASRCINCRx and DMADSTINCRx control bits
  - DMASRCINCRx/ DMADSTINCRx bits select if the source/destination address is incremented, decremented, or unchanged after each transfer

- Transfers may be byte-to-byte, word-to-word, byte-to-word, or word-to-byte
  - Word-to-byte: only the lower byte of the source-word is transferred
  - Byte-to-word: the upper byte of the destination-word is cleared when the transfer occurs
DMA Transfer Modes

- Single/Repeated single modes: each byte/word transfer requires a separate trigger
- Block/Repeated block modes: a transfer of a complete block of data occurs after one trigger
  - CPU is halted until the complete block has been transferred
- Burst-block/Repeated burst-block modes: transfers are block transfers with CPU activity interleaved.
  - CPU executes 2 MCLK cycles after every four byte/word transfers of the block resulting in 20% CPU execution capacity

<table>
<thead>
<tr>
<th>DMADTx</th>
<th>Transfer Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Single transfer</td>
<td>Each transfer requires a trigger. DMAEN is automatically cleared when DMAxSZ transfers have been made.</td>
</tr>
<tr>
<td>001</td>
<td>Block transfer</td>
<td>A complete block is transferred with one trigger. DMAEN is automatically cleared at the end of the block transfer.</td>
</tr>
<tr>
<td>010, 011</td>
<td>Burst-block transfer</td>
<td>CPU activity is interleaved with a block transfer. DMAEN is automatically cleared at the end of the burst-block transfer.</td>
</tr>
<tr>
<td>100</td>
<td>Repeated single transfer</td>
<td>Each transfer requires a trigger. DMAEN remains enabled.</td>
</tr>
<tr>
<td>101</td>
<td>Repeated block transfer</td>
<td>A complete block is transferred with one trigger. DMAEN remains enabled.</td>
</tr>
<tr>
<td>110, 111</td>
<td>Repeated burst-block transfer</td>
<td>CPU activity is interleaved with a block transfer. DMAEN remains enabled.</td>
</tr>
</tbody>
</table>
DMA Trigger Operation

- DMAxTSELx bits select trigger
- Edge-sensitive or level-sensitive

Table 10-2. DMA Trigger Operation

<table>
<thead>
<tr>
<th>DMAxTSELx</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>A transfer is triggered when the DMAREQ bit is set. The DMAREQ bit is automatically reset when the transfer starts.</td>
</tr>
<tr>
<td>0001</td>
<td>A transfer is triggered when the TACC2 CCIFG flag is set. The TACC2 CCIFG flag is automatically reset when the transfer starts. If the TACC2 CCIE bit is set, the TACC2 CCIFG flag will not trigger a transfer.</td>
</tr>
<tr>
<td>0010</td>
<td>A transfer is triggered when the TBCCR2 CCIFG flag is set. The TBCCR2 CCIFG flag is automatically reset when the transfer starts. If the TBCCR2 CCIE bit is set, the TBCCR2 CCIFG flag will not trigger a transfer.</td>
</tr>
<tr>
<td>0011</td>
<td>Devices with USART0: A transfer is triggered when the URXIFG0 flag is set. URXIFG0 is automatically reset when the transfer starts. If URXIE0 is set, the URXIFG0 flag will not trigger a transfer. Devices with USCI_A0: A transfer is triggered when the UCA0RXIFG flag is set. UCA0RXIFG is automatically reset when the transfer starts. If UCA0RXIE is set, the UCA0RXIFG flag will not trigger a transfer.</td>
</tr>
<tr>
<td>0100</td>
<td>Devices with USART0: A transfer is triggered when the UTXIFG0 flag is set. UTXIFG0 is automatically reset when the transfer starts. If UTXIE0 is set, the UTXIFG0 flag will not trigger a transfer. Devices with USCI_A0: A transfer is triggered when the UCA0TXIFG flag is set. UCA0TXIFG is automatically reset when the transfer starts. If UCA0TXIE is set, the UCA0TXIFG flag will not trigger a transfer.</td>
</tr>
<tr>
<td>0101</td>
<td>A transfer is triggered when the DAC12_0CTL DAC12IFG flag is set. The DAC12_0CTL DAC12IFG flag is automatically cleared when the transfer starts. If the DAC12_0CTL DAC12IE bit is set, the DAC12_0CTL DAC12IFG flag will not trigger a transfer.</td>
</tr>
<tr>
<td>0110</td>
<td>A transfer is triggered by an ADC12IFGx flag. When single-channel conversions are performed, the corresponding ADC12IFGx is the trigger. When sequences are used, the ADC12IFGx for the last conversion in the sequence is the trigger. A transfer is triggered when the conversion is completed and the ADC12IFGx is set. Setting the ADC12IFGx with software will not trigger a transfer. All ADC12IFGx flags are automatically reset when the associated ADC12MEMx register is accessed by the DMA controller.</td>
</tr>
</tbody>
</table>
DMA Trigger Operation (cont’d)

0111 A transfer is triggered when the TACCR0 CCIFG flag is set. The TACCR0 CCIFG flag is automatically reset when the transfer starts. If the TACCR0 CCIE bit is set, the TACCR0 CCIFG flag will not trigger a transfer.

1000 A transfer is triggered when the TBCCR0 CCIFG flag is set. The TBCCR0 CCIFG flag is automatically reset when the transfer starts. If the TBCCR0 CCIE bit is set, the TBCCR0 CCIFG flag will not trigger a transfer.

1001 A transfer is triggered when the URXIFG1 flag is set. URXIFG1 is automatically reset when the transfer starts. If URXIE1 is set, the URXIFG1 flag will not trigger a transfer.

1010 A transfer is triggered when the UTXIFG1 flag is set. UTXIFG1 is automatically reset when the transfer starts. If UTXIE1 is set, the UTXIFG1 flag will not trigger a transfer.

1011 A transfer is triggered when the hardware multiplier is ready for a new operand.

1100 A transfer is triggered when the UCB0RXIFG flag is set. UCB0RXIFG is automatically reset when the transfer starts. If UCB0RXIE is set, the UCB0RXIFG flag will not trigger a transfer.

1101 A transfer is triggered when the UCB0TXIFG flag is set. UCB0TXIFG is automatically reset when the transfer starts. If UCB0TXIE is set, the UCB0TXIFG flag will not trigger a transfer.

1110 A transfer is triggered when the DMAxIFG flag is set. DMA0IFG triggers channel 1, DMA1IFG triggers channel 2, and DMA2IFG triggers channel 0. None of the DMAxIFG flags are automatically reset when the transfer starts.

1111 A transfer is triggered by the external trigger DMAE0.
DMA Channel Priorities

- Default DMA channel priorities are DMA0–DMA1–DMA2
  - If two or three triggers happen simultaneously or are pending, the channel with the highest priority completes its transfer (single, block or burst-block transfer) first, then the second priority channel, then the third priority channel.

- Transfers in progress are not halted if a higher priority channel is triggered
  - The higher priority channel waits until the transfer in progress completes before starting

- DMA channel priorities are configurable with the ROUNDROBIN bit (see below)

<table>
<thead>
<tr>
<th>DMA Priority</th>
<th>Transfer Occurs</th>
<th>New DMA Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA0 – DMA1 – DMA2</td>
<td>DMA1</td>
<td>DMA2 – DMA0 – DMA1</td>
</tr>
<tr>
<td>DMA2 – DMA0 – DMA1</td>
<td>DMA2</td>
<td>DMA0 – DMA1 – DMA2</td>
</tr>
<tr>
<td>DMA0 – DMA1 – DMA2</td>
<td>DMA0</td>
<td>DMA1 – DMA2 – DMA0</td>
</tr>
</tbody>
</table>
DMA Transfer Cycle Times

- DMA requires 1 or 2 MCLK cc to synchronize before each single transfer or complete block or burst-block transfer.
- Each byte/word transfer requires 2 MCLK after synchronization, and one cycle of wait time after the transfer.
- DMA cycle time is dependent on the MSP430 operating mode and clock system setup (use MCLK).

If the MCLK source is active, but the CPU is off, the DMA controller will use the MCLK source for each transfer, without re-enabling the CPU.

If the MCLK source is off, the DMA controller will temporarily restart MCLK, sourced with DCOCLK, for the single transfer or complete block or burst-block transfer.

The CPU remains off, and after the transfer completes, MCLK is turned off.

<table>
<thead>
<tr>
<th>CPU Operating Mode</th>
<th>Clock Source</th>
<th>Maximum DMA Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active mode</td>
<td>MCLK=DCOCLK</td>
<td>4 MCLK cycles</td>
</tr>
<tr>
<td>Active mode</td>
<td>MCLK=LFXT1CLK</td>
<td>4 MCLK cycles</td>
</tr>
<tr>
<td>Low-power mode LPM0/1</td>
<td>MCLK=DCOCLK</td>
<td>5 MCLK cycles</td>
</tr>
<tr>
<td>Low-power mode LPM3/4</td>
<td>MCLK=DCOCLK</td>
<td>5 MCLK cycles + 6 μs†</td>
</tr>
<tr>
<td>Low-power mode LPM0/1</td>
<td>MCLK=LFXT1CLK</td>
<td>5 MCLK cycles</td>
</tr>
<tr>
<td>Low-power mode LPM3</td>
<td>MCLK=LFXT1CLK</td>
<td>5 MCLK cycles</td>
</tr>
<tr>
<td>Low-power mode LPM4</td>
<td>MCLK=LFXT1CLK</td>
<td>5 MCLK cycles + 6 μs†</td>
</tr>
</tbody>
</table>

† The additional 6 μs are needed to start the DCOCLK. It is the τ_{h,LPM0} parameter in the data sheet.
DMA and Interrupts

- DMA transfers are not interruptible by system interrupts.
  - System interrupts remain pending until the completion of the transfer
  - NMI interrupts can interrupt the DMA controller if the ENNMI bit is set
- System interrupt service routines are interrupted by DMA transfers
  - If an interrupt service routine or other routine must execute with no interruptions, the DMA controller should be disabled prior to executing the routine
- Each DMA channel has its own DMAIFG flag
  - Each DMAIFG flag is set in any mode, when the corresponding DMAxSZ register counts to zero. If the corresponding DMAIE and GIE bits are set, an interrupt request is generated
DMA and Other Devices

- USCI_B I2C Module
- ADC12
- DAC12
- Writing to Flash
LCD Controller

- Liquid Crystal Display (LCD) controller
  - Included in several devices of the MSP430 families (’3xx and ’4xx)
  - Allows a rapid and simple way to interface with the program
- LCD controller commands the LCD panels generating voltage signals to the segments. It supports static, and multiplex rates up to 4 (2 mux, 3 mux and 4 mux) LCD panels
- Features
  - Display memory
  - Automatic signal generation
  - Configurable frame frequency
  - Blinking capability
  - Support for 4 types of LCDs:
    - Static
    - 2-mux, 1/2 bias
    - 3-mux, 1/3 bias
    - 4-mux, 1/3 bias
LCD Memory Map

- Each memory bit corresponds to one LCD segment, or is not used, depending on the mode.
- To turn on an LCD segment, its corresponding memory bit is set.
LCD Controller Operation

- LCD controller supports blinking
  - The LCDSON bit is ANDed with each segment’s memory bit.
    - When LCDSON = 1, each segment is on or off according to its bit value
    - When LCDSON = 0, each LCD segment is off

- Timing generation
  - Uses the $f_{\text{LCD}}$ signal from the Basic Timer1 to generate the timing for common and segment lines
    - Proper frequency $f_{\text{LCD}}$ depends on the LCD’s requirement for framing frequency and LCD multiplex rate.
    - See the Basic Timer1 chapter for more information on configuring the $f_{\text{LCD}}$ frequency
LCD Controller Operation

- LCD voltage generation
  - Voltages required for the LCD signals are supplied externally to pins R33, R23, R13, and R03
  - Using an equally weighted resistor divider ladder between these pins establishes the analog voltages as shown in Table 24-1
  - The resistor value R is typically 680 k
    - Values of R from 100k to 1M can be used depending on LCD requirements.
  - R33 is a switched-VCC output. This allows the power to the resistor ladder to be turned off eliminating current consumption when the LCD is not used.

<table>
<thead>
<tr>
<th>OSCOFF</th>
<th>LCDMX</th>
<th>LCDON</th>
<th>VA</th>
<th>VB</th>
<th>VC</th>
<th>VD</th>
<th>R33</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>xx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>xx</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>1</td>
<td>V5/V1</td>
<td>V1/V5</td>
<td>V5/V1</td>
<td>V1/V5</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>1</td>
<td>V5/V1</td>
<td>V1/V5</td>
<td>V3/V3</td>
<td>V1/V5</td>
<td>On</td>
</tr>
<tr>
<td>0</td>
<td>1x</td>
<td>1</td>
<td>V5/V1</td>
<td>V2/V4</td>
<td>V4/V2</td>
<td>V1/V5</td>
<td>On</td>
</tr>
</tbody>
</table>
Static Mode

- Each MSP430 segment pin drives one LCD segment
- One common line, COM0, is used.

![Diagram of Static Mode with waveforms]

Resulting Voltage for Segment a (COM0–SP1) Segment is On.

Resulting Voltage for Segment b (COM0–SP2) Segment is Off.
Static LCD Example
Static Mode Software Example

; All eight segments of a digit are often located in four
display memory bytes with the static display method.
;
a EQU 001h
b EQU 010h
c EQU 002h
d EQU 020h
e EQU 004h
f EQU 040h
g EQU 008h
h EQU 080h
; The register content of RX should be displayed.
; The Table represents the 'on'-segments according to the
; content of RX.

MOV.B Table (Rx),Ry ; Load segment information
; into temporary memory.
; (Ry) = 0000 0000 hidb gecas
MOV.B Ry, &LCDn
; Note:
; All bits of an LCD memory 'byte are written
ERA Ry
; (Ry) = 0000 0000 0hfd bgec
MOV.B Ry, &LCDn+1
; Note:
; All bits of an LCD memory 'byte are written
ERA Ry
; (Ry) = 0000 0000 00hf dbge
MOV.B Ry, &LCDn+2
; Note:
; All bits of an LCD memory 'byte are written
ERA Ry
; (Ry) = 0000 0000 000h fdhg
MOV.B Ry, &LCDn+3
; Note:
; All bits of an LCD memory 'byte are written

................
................

Table DB a+b+c+d+e+f ; displays "0"
DB b+c; ; displays "1"
................
................
DB

}
2-MUX Mode

- Each MSP430 segment pin drives two LCD segments
- Two common lines, COM0 and COM1, are used
- 2-mux example waveforms

\begin{align*}
a &= \text{COM1-SP1} \\
b &= \text{COM1-SP2} \\
c &= \text{COM1-SP3} \\
d &= \text{COM0-SP3} \\
e &= \text{COM0-SP4} \\
f &= \text{COM0-SP1} \\
g &= \text{COM1-SP4} \\
h &= \text{COM0-SP2}
\end{align*}
2-MUX LCD Example
2-MUX Software Example

; All eight segments of a digit are often located in two
display memory bytes with the 2mux display rate
;
a EQU 002h
b EQU 020h
c EQU 008h
d EQU 004h
e EQU 040h
f EQU 001h
g EQU 080h
h EQU 010h
; The register content of Rx should be displayed.
The Table represents the 'on'-segments according to the
content of Rx.
;
...........
MOV.B Table(Rx),Ry ; Load segment information into
 temporary memory.
MOV.B Ry, &LCDn ; (Rx) = 0000 0000 0geb 0cad
Note:
; All bits of an LCD memory byte
 are written
RAR Ry ; (Rx) = 0000 0000 00eb 00cd
RAR Ry ; (Rx) = 0000 0000 00ge 00cd
MOV.B Ry, &LCDn+1 ; Note:
; All bits of an LCD memory byte
 are written
...........
...........
...........
Table DE a+b+c+d+e+f ; displays "0"
...........
DE a+b+c+d+e+f+g+ ; displays "8"
...........
...........
DE
...........
..
3-MUX Mode Waverforms

- Each MSP430 segment pin drives three LCD segments
- Three common lines, COM0 and COM1, and COM2 are used
- 3-mux example waverforms
3-MUX LCD Example
; The 3mux rate can support nine segments for each digit. The nine segments of a digit are located in 1 1/2 display memory bytes.

a EQU 0040h
b EQU 0400h
c EQU 0200h
d EQU 0610h
e EQU 0011h
f EQU 0602h
g EQU 0020h
h EQU 0100h
Y EQU 0004h

; The LSDigit of register Rx should be displayed.
; The Table represents the 'on'-segments according to the LSDigit of register of Rx.
; The register Ry is used for temporary memory

; ODDDIGRLA Rx ; LCD in 3mux has 9 segments per digit; word table required for displayed characters.
MOV Table(Rx),Ry ; Load segment information to temporary mem.
(Ry) = 0000 0bch 0agd 0yfe
MOV.B Ry, &LCDn ; write 'a, g, d, y, f, e' of Digit n (LowByte)
SWPB Ry ; (Ry) = 0agd 0yfe 0000 0bch
BIC.B #07h, &LCDn+1 ; write 'b, c, h' of Digit n+1 (HighByte)
BIS.B Ry, &LCDn+1

; EVNDIGRLA Rx ; LCD in 3mux has 9 segments per digit; word table required for displayed characters.
MOV Table(Rx),Ry ; Load segment information to temporary mem.
(Ry) = 0000 0bch 0agd 0yfe
RLA Ry ; (Ry) = 0000 b0ch a0ad y0fe
RLA Ry ; (Ry) = 000b c0a 0doy f000
RLA Ry ; (Ry) = 00bc h0ag d0yf e000
RLA Ry ; (Ry) = 0bc8 0agd 0yfe 0000
BIC.B #070h, &LCDn+1
BIS.B Ry, &LCDn+1 ; write 'y, f, e' of Digit n+1 (LowByte)
SWPB Ry ; (Ry) = 0yfe 0000 0bch 0agd
MOV.B Ry, &LCDn+2 ; write 'b, c, h, a, g, d' of Digit n+1 (HighByte)

............

Table DW a+b+c+d+e+f ; displays "0"
DW b+c ; displays "1"
............
DW a+e+f+g ; displays "?"
4-MUX Mode Waverforms

- Each MSP430 segment pin drives four LCD segments
- Four common lines, COM0, COM1, COM2, and COM3 are used
- 4-mux example waverforms
4-MUX LCD
Example

Pinout and Connections

Display Memory

<table>
<thead>
<tr>
<th>COM1</th>
<th>COM2</th>
<th>COM3</th>
<th>COM4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digit 16</th>
<th>Digit 15</th>
<th>Digit 14</th>
<th>Digit 13</th>
<th>Digit 12</th>
<th>Digit 11</th>
<th>Digit 10</th>
<th>Digit 9</th>
<th>Digit 8</th>
<th>Digit 7</th>
<th>Digit 6</th>
<th>Digit 5</th>
<th>Digit 4</th>
<th>Digit 3</th>
<th>Digit 2</th>
<th>Digit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
4-MUX Software Example

; The 4mux rate supports eight segments for each digit.
; All eight segments of a digit can often be located in
; one display memory byte
a EQU 080h
b EQU 040h
c EQU 020h
d EQU 001h
e EQU 002h
f EQU 008h
g EQU 004h
h EQU 010h

; The LSDigit of register Rx should be displayed.
; The Table represents the 'on' - segments according to the
; content of Rx.
;
MOV.B Table(Rx),&LCr0 ; n = 1 ...... 15
; all eight segments are
; written to the display
; memory


Table DB a+b+c+d+e+f ; displays "0"
   DB b+c ; displays "1"


DB b+c+d+e+g ; displays "d"
DB a+d+e+f+g ; displays "E"
DB a+e+f+g ; displays "F"
# LCD Control Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Short Form</th>
<th>Register Type</th>
<th>Address</th>
<th>Initial State</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD control register</td>
<td>LCDCTL</td>
<td>Read/write</td>
<td>090h</td>
<td>Reset with PUC</td>
</tr>
<tr>
<td>LCD memory 1</td>
<td>LCDM1</td>
<td>Read/write</td>
<td>091h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 2</td>
<td>LCDM2</td>
<td>Read/write</td>
<td>092h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 3</td>
<td>LCDM3</td>
<td>Read/write</td>
<td>093h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 4</td>
<td>LCDM4</td>
<td>Read/write</td>
<td>094h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 5</td>
<td>LCDM5</td>
<td>Read/write</td>
<td>095h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 6</td>
<td>LCDM6</td>
<td>Read/write</td>
<td>096h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 7</td>
<td>LCDM7</td>
<td>Read/write</td>
<td>097h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 8</td>
<td>LCDM8</td>
<td>Read/write</td>
<td>098h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 9</td>
<td>LCDM9</td>
<td>Read/write</td>
<td>099h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 10</td>
<td>LCDM10</td>
<td>Read/write</td>
<td>09Ah</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 11</td>
<td>LCDM11</td>
<td>Read/write</td>
<td>09Bh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 12</td>
<td>LCDM12</td>
<td>Read/write</td>
<td>09Ch</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 13</td>
<td>LCDM13</td>
<td>Read/write</td>
<td>09Dh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 14</td>
<td>LCDM14</td>
<td>Read/write</td>
<td>09Eh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 15</td>
<td>LCDM15</td>
<td>Read/write</td>
<td>09Fh</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 16</td>
<td>LCDM16</td>
<td>Read/write</td>
<td>0Ah</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 17</td>
<td>LCDM17</td>
<td>Read/write</td>
<td>0Ah</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 18</td>
<td>LCDM18</td>
<td>Read/write</td>
<td>0A2h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 19</td>
<td>LCDM19</td>
<td>Read/write</td>
<td>0A3h</td>
<td>Unchanged</td>
</tr>
<tr>
<td>LCD memory 20</td>
<td>LCDM20</td>
<td>Read/write</td>
<td>0A4h</td>
<td>Unchanged</td>
</tr>
</tbody>
</table>
### LCD Control Register

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit</th>
<th>Description &amp; Details</th>
</tr>
</thead>
</table>
| 7-5          | LCDPx | LCD Port Select. These bits select the pin function to be port I/O or LCD function for groups of segments pins. These bits ONLY affect pins with multiplexed functions. Dedicated LCD pins are always LCD function.  
000 No multiplexed pins are LCD function  
001 S0-S15 are LCD function  
010 S0-S19 are LCD function  
011 S0-S23 are LCD function  
100 S0-S27 are LCD function  
101 S0-S31 are LCD function  
110 S0-S35 are LCD function  
111 S0-S39 are LCD function |
| 4-3          | LCDMx | LCD mux rate. These bits select the LCD mode.  
00 Static  
01 2-mux  
10 3-mux  
11 4-mux |
| 2            | LCDSON | Bit 2 LCD segments on. This bit supports flashing LCD applications by turning off all segment lines, while leaving the LCD timing generator and R33 enabled.  
0 All LCD segments are off  
1 All LCD segments are enabled and on or off according to their corresponding memory location |
| 1            | Unused | Unused |
| 0            | LCDON | Bit 0 LCD On. This bit turns on the LCD timing generator and R33,  
0 LCD timing generator and Ron are off  
1 LCD timing generator and Ron are on |
Softbaugh LCD SBLCDA4: Segment Description

SBLCDA4 Display