Capstone Design Course

Lecture-1: Parallel I/O

By

Syed Masud Mahmud, Ph.D.

Copyright © 2002 by Syed Masud Mahmud
BCLR          Clear Bit(s) in Memory

Syntax: BCLR (opr) (msk)

Example: BCLR 10,X $C0

Operation: Clear multiple bits in the operand. The bit(s) to be cleared are specified by ONEs in the mask byte. The other bits remain unchanged.

Assume that X = $1000.

The address of operand = 10+$1000 = $100A

Mask = $C0 = 1100 0000

• The most significant two bits of the byte at location $100A are cleared.
Valid Addressing Modes for BCLR

<table>
<thead>
<tr>
<th>Cycle</th>
<th>BCLR (DIR)</th>
<th>BCLR (IND,X)</th>
<th>BCLR (IND,Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addr</td>
<td>Data</td>
<td>R/W</td>
</tr>
<tr>
<td>1</td>
<td>OP</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>OP+1</td>
<td>dd</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>00dd</td>
<td>(00dd)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>OP+2</td>
<td>MM</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>FFFF</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>00dd</td>
<td>result</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>X+ff</td>
<td>result</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BSET          Set Bit(s) in Memory

Syntax: BSET (opr) (msk)
Example: BSET 20,X $07

Operation: Set multiple bits in the operand.
   The bit(s) to be set are specified by ONEs in the mask byte. The other bits remain unchanged.

BSET 20,X $07
Assume that X = $1000.
The address of operand = 20+$1000 = $1014
Mask = $07 = 0000 0111
• The least significant three bits of the byte at location $1014 are set.
BRCLR  Branch if Bit(s) Clear

Syntax:  BRCLR (opr) (msk) (rel)

Example:  BRCLR  10,X $80  NEXT

Operation:  If all operand bits corresponding to the 1s in the mask byte are ZEROs, then a branch will occur.

Assume that X = $1000.
The address of operand = 10+$1000 = $100A
Mask = $80 = 1000 0000

• If the most significant bit of the byte at location $100A is a zero, then the processor will branch to the instruction at label NEXT
Valid Addressing Modes for BRCLR

<table>
<thead>
<tr>
<th>Cycle</th>
<th>BRCLR (DIR)</th>
<th>BRCLR (IND,X)</th>
<th>BRCLR (IND,Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addr</td>
<td>Data</td>
<td>R/W</td>
</tr>
<tr>
<td>1</td>
<td>OP</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>OP+1</td>
<td>dd</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>00dd</td>
<td>(00dd)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>OP+2</td>
<td>mm</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>OP+3</td>
<td>rr</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>FFFF</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>FFFF</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>FFFF</td>
<td>—</td>
<td>1</td>
</tr>
</tbody>
</table>
BRSET Branch if Bit(s) Set

Syntax: BRSET (opr) (msk) (rel)

Example: BRSET 10,X $80 NEXT

Operation: If all operand bits corresponding to the 1s in the mask byte are ONEs, then a branch will occur.

BRSET 10,X $80 NEXT

Assume that X = $1000.

The address of operand = 10+$1000 = $100A

Mask = $80 = 1000 0000

• If the most significant bit of the byte at location $100A is a ONE, then the processor will branch to the instruction at label NEXT
Modes of 68HC11

- Single Chip Mode
- Expanded Mode

**Single Chip Mode**
- Only the internal RAM and ROM are used to keep the code and data.
- Five Input/Output ports are available in this mode.

**Expanded Mode**
- In addition to the internal RAM and ROM some external memory can also be used.
- Only three Input/Output ports are available in this mode.

**FOR ECE4600 COURSE, YOU WILL BE USING THE CHIP IN THIS MODE.**
Ports of 68HC11

- Port-A: It has 8 input/output lines
- Port-B: It has 8 output lines
- Port-C: It has 8 input/output lines
- Port-D: It has 6 input/output lines
- Port-E: It has 8 input lines

- Port-B and Port-C are not available in the Expanded Mode of the chip.
Port-A

PORT A

PA7/PAI/OC1
PA6/OC2/OC1
PA5/OC3/OC1
PA4/OC4/OC1
PA3/OC5/OC1
PA2/IC1
PA1/IC2
PA0/IC3
Port-B (available only in Single Chip Mode)
Port-C (available only in Single Chip Mode)
Port-D

CONTROL

PORT D

PD5/SS
PD4/SCK
PD3/MOSI
PD2/MISO

PD1/TxD
PD0/RxD
Port-E

PE7/AN7  →
PE6/AN6  →
PE5/AN5  →
PE4/AN4  →
PE3/AN3  →
PE2/AN2  →
PE1/AN1  →
PE0/AN0  →
PORT E
<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1000</td>
<td>PORTA</td>
<td>Port-A Data Reg.</td>
</tr>
<tr>
<td>$1003</td>
<td>PORTC</td>
<td>Port-C Data Reg.</td>
</tr>
<tr>
<td>$1004</td>
<td>PORTB</td>
<td>Port-B Data Reg.</td>
</tr>
<tr>
<td>$1007</td>
<td>DDRC</td>
<td>Port-C Data Direction Reg.</td>
</tr>
<tr>
<td>$1008</td>
<td>PORTD</td>
<td>Port-D Data Reg.</td>
</tr>
<tr>
<td>$1009</td>
<td>DDRD</td>
<td>Port-D Data Direction Reg.</td>
</tr>
<tr>
<td>$100A</td>
<td>PORTE</td>
<td>Port-E Data Reg.</td>
</tr>
</tbody>
</table>
DDRC: Port-C Data Direction Reg.

- If the 68HC11 is in single chip mode, then this register controls the direction of the lines of Port-C.
- If a bit of this register is a 1, then the corresponding line of Port-C is an output line. Otherwise, the line is an input line.

**Example:**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

PC7, PC4, PC2 and PC1 are output lines. PC6, PC5, PC3 and PC0 are input lines.
DDRD: Port-D Data Direction Reg.

- This register controls the direction of the lines of Port-D.
- If a bit of this register is a 1, then the corresponding line of Port-D is an output line. Otherwise, the line is an input line.

**Example:**

<table>
<thead>
<tr>
<th>DDRD</th>
</tr>
</thead>
<tbody>
<tr>
<td>7    6  5  4  3  2  1  0</td>
</tr>
<tr>
<td>x    x  0  0  0  0  1  1  0</td>
</tr>
</tbody>
</table>

PD2 and PD1 are output lines.
PD5, PD4, PD3 and PD0 are input lines.
Controlling an LED using a Switch

LDX #$1000 ; X = $1000 = PortA Addr.

LOOP BRCLR 0,X 1 ON ; If PA0 = 0 then LED ON
BCLR 0,X $10 ; LED is OFF
BRA LOOP ; Repeat
ON BSET 0,X $10 ; LED is ON
BRA LOOP ; Repeat

Function Table

<table>
<thead>
<tr>
<th>Switch</th>
<th>LED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open (PA0 = 1)</td>
<td>Off (PA4 = 0)</td>
</tr>
<tr>
<td>Closed (PA0 = 0)</td>
<td>On (PA4 = 1)</td>
</tr>
</tbody>
</table>
Controlling an LED using two Switches

LDX #$1000 ; X = $1000 = PortA Addr.

LOOP BRCLR 0,X 3 ON ; If PA0=0 then LED ON
BCLR 0,X $10 ; LED is OFF
BRA LOOP ; Repeat

ON BSET 0,X $10 ; LED is ON
BRA LOOP ; Repeat

<table>
<thead>
<tr>
<th>Function Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sw2</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Open (PA1 = 1)</td>
</tr>
<tr>
<td>Closed (PA1 = 0)</td>
</tr>
<tr>
<td>Closed (PA1 = 0)</td>
</tr>
</tbody>
</table>
Keyboard Scanning Program

![Diagram showing a keyboard scanning program with rows and columns labeled and numbers indicating connections.](image-url)
Scanning Col-0
Scanning Col-1
Keyboard Scanning Program (contd.)

- This subroutine scans a 4x4 keypad. If any key is down, then the subroutine returns the key number through Register B, otherwise $FF$ is returned through Register B.

```
REGBAS   EQU   $1000
PORTC   EQU   3
DDRC   EQU   7
ORG   $100

ROWNUM   RMB   1
COLNUM   RMB   1
```
Keyboard Scanning Program (contd.)

ORG   $2000
PSHX
PSHA
LDX   #REGBAS
LDAA  #$0F ; PC7 - PC4 are inputs
STAA  DDRC,X ; PC3 - PC0 outputs
* Scan a Column by sending a zero through the
* corresponding output line of Port-C
LDAA  #$FE
LOOP  STAA  PORTC,X
LDAB  PORTC,X ; Read keyboard
ORAB  #$0F ; Least sig. 4bits are 1
Keyboard Scanning Program (contd.)

CMPB #$FF ; Check if all bits are 1
BNE KEYDWN ; If not eq, a key down
LSLA
ORAA #1 ; Set the least significant bit to 1
CMPA #$EF ; Check if all cols scanned
BEQ NOKEY
BRA LOOP ; Go back, check next col.

NOKEY LDAB #$FF ; $FF is code for No Key
JMP DONE

* Determine the Column Number of the key
KEYDWN CLR COLNUM ; COLNUM=0
Keyboard Scanning Program (contd.)

CLOOP LSRA
BCC NEXT1 ; If C=0, go to NEXT1
INC COLNUM ; Inc. COLNUM by 1
BRA CLOOP

* Determine the Row Number of the key

NEXT1 LDAA #3
STA ROWNUM ; ROWNUM = 3

RLOOP LSLB
BCC NEXT2
DEC ROWNUM ; Dec. ROWNUM by 1
BRA RLOOP
Keyboard Scanning Program (contd.)

* Determine the key number

```
NEXT2   LDAA   COLNUM
LDAB    #4
MUL     ; B = COLNUM*4
ADDB    ROWNUM

* B = COLNUM*4 + ROWNUM = KEY NUMBER

DONE    PULA
PULX
RTS
```