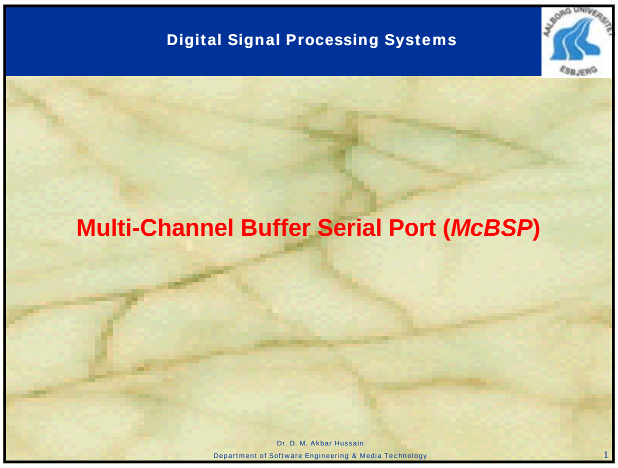
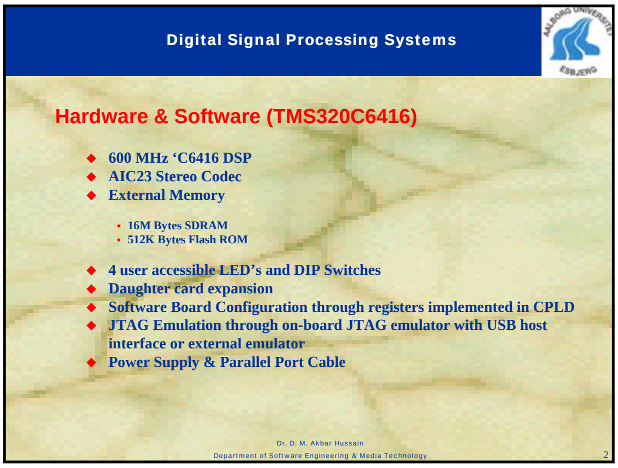


Digital Signal Processing Systems

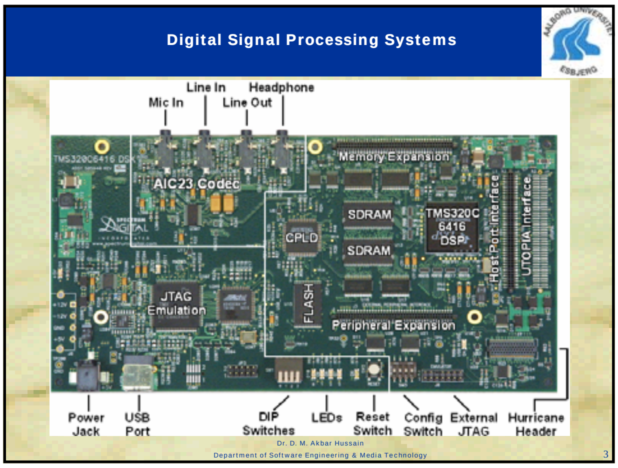
Multi-Channel Buffer Serial Port (*McBSP*)



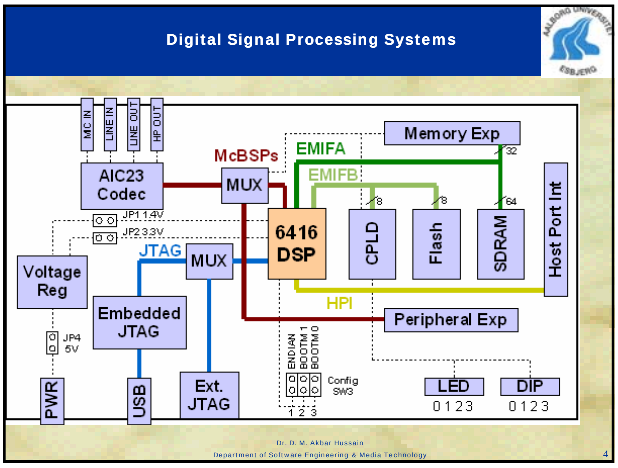
Digital Signal Processing Systems

Hardware & Software (TMS320C6416)

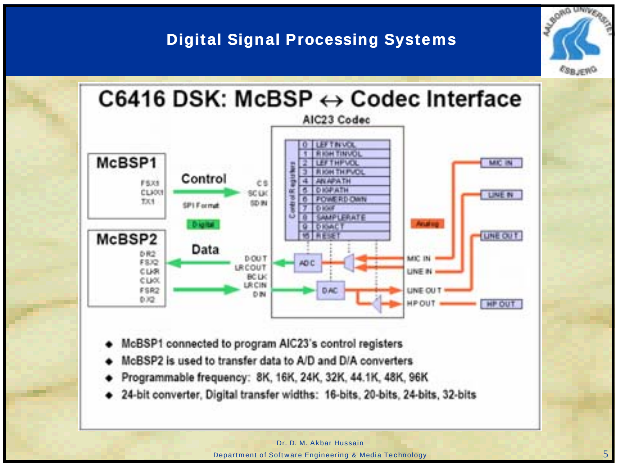
- 600 MHz ‘C6416 DSP
- AIC23 Stereo Codec
- External Memory
  - 16M Bytes SDRAM
  - 512K Bytes Flash ROM
- 4 user accessible LED’s and DIP Switches
- Daughter card expansion
- Software Board Configuration through registers implemented in CPLD
- JTAG Emulation through on-board JTAG emulator with USB host interface or external emulator
- Power Supply & Parallel Port Cable



Digital Signal Processing Systems



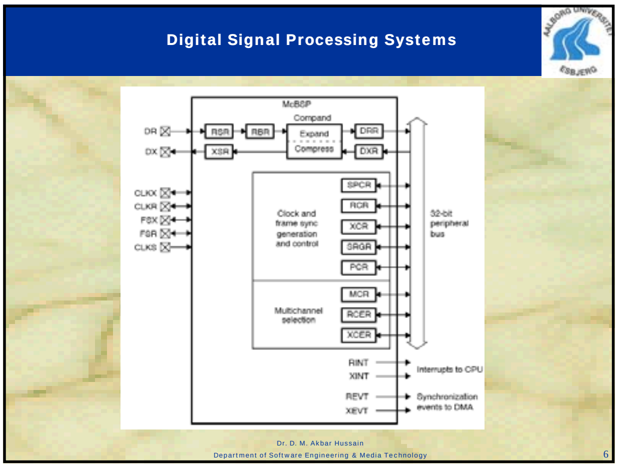
Digital Signal Processing Systems



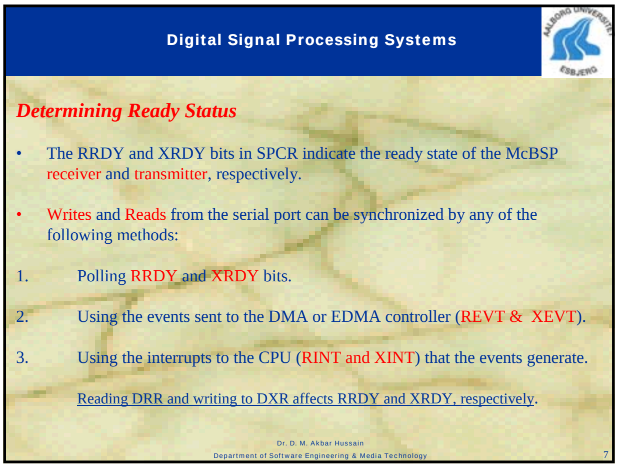
Digital Signal Processing Systems

C6416 DSK: McBSP ↔ Codec Interface

- McBSP1 connected to program AIC23’s control registers
- McBSP2 is used to transfer data to A/D and D/A converters
- Programmable frequency: 8K, 16K, 24K, 32K, 44.1K, 48K, 96K
- 24-bit converter, Digital transfer widths: 16-bits, 20-bits, 24-bits, 32-bits



Digital Signal Processing Systems



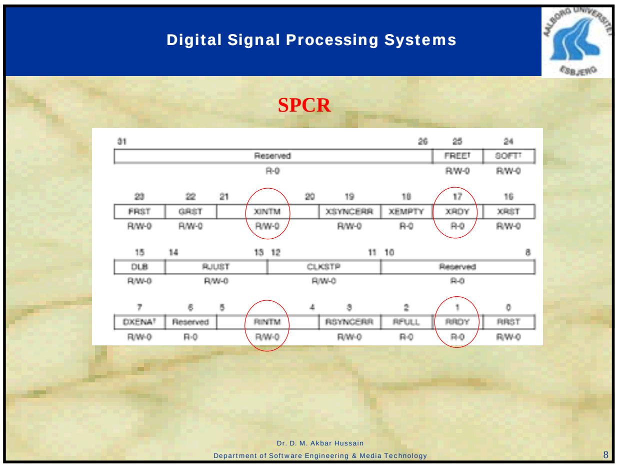
Determining Ready Status

- The RRDY and XRDY bits in SPCR indicate the ready state of the McBSP receiver and transmitter, respectively.

- Writes and Reads from the serial port can be synchronized by any of the following methods:

1. Polling RRDY and XRDY bits.
2. Using the events sent to the DMA or EDMA controller (REVT & XEVT).
3. Using the interrupts to the CPU (RINT and XINT) that the events generate.

Reading DRR and writing to DXR affects RRDY and XRDY, respectively.



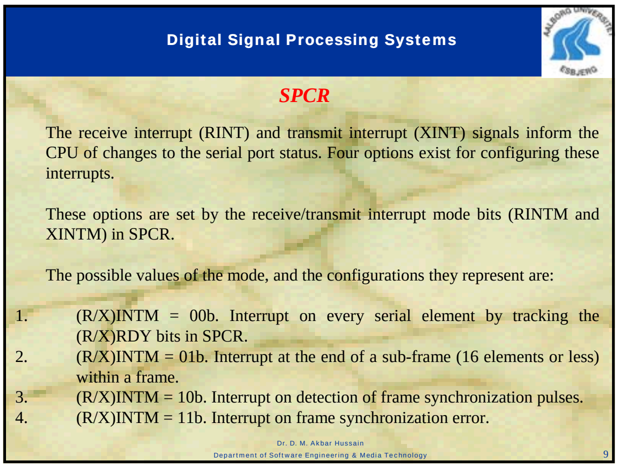
SPCR

31 26	25	24
Reserved	FREE†	SOFT†
R-0	R/W-0	R/W-0

23	22	21 20	19	18	17	16
FRST	GRST	XINTM	XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R/W-0

15	14 13	12 11	10 8
DLB	RJUST	CLKSTP	Reserved
R/W-0	R/W-0	R/W-0	R-0

7	6	5 4	3	2	1	0
DXENA†	Reserved	RINTM	RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0	R/W-0	R-0	R-0	R/W-0



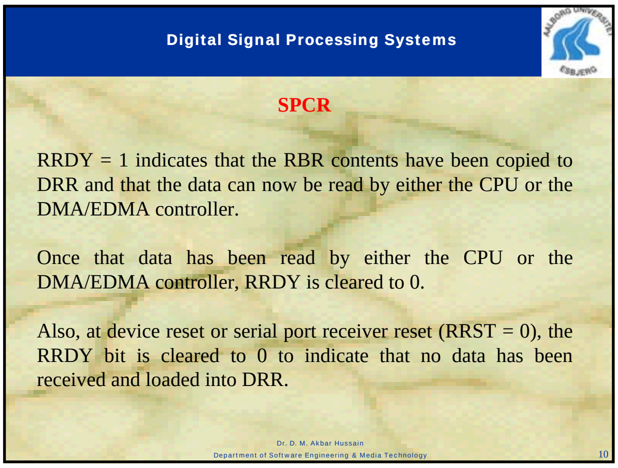
SPCR

The receive interrupt (RINT) and transmit interrupt (XINT) signals inform the CPU of changes to the serial port status. Four options exist for configuring these interrupts.

These options are set by the receive/transmit interrupt mode bits (RINTM and XINTM) in SPCR.

The possible values of the mode, and the configurations they represent are:

1. (R/X)INTM = 00b. Interrupt on every serial element by tracking the (R/X)RDY bits in SPCR.
2. (R/X)INTM = 01b. Interrupt at the end of a sub-frame (16 elements or less) within a frame.
3. (R/X)INTM = 10b. Interrupt on detection of frame synchronization pulses.
4. (R/X)INTM = 11b. Interrupt on frame synchronization error.

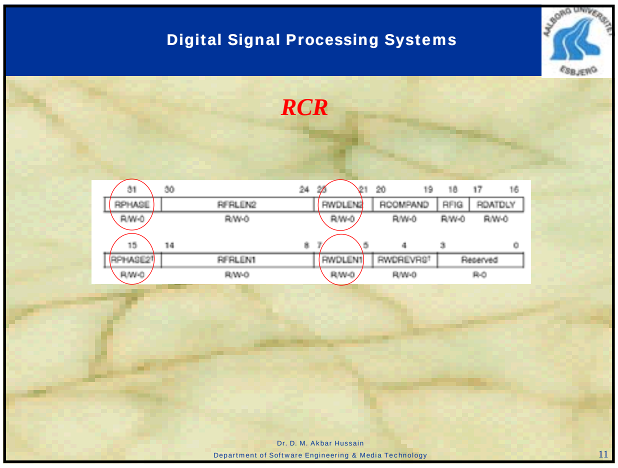


SPCR

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by either the CPU or the DMA/EDMA controller.

Once that data has been read by either the CPU or the DMA/EDMA controller, RRDY is cleared to 0.

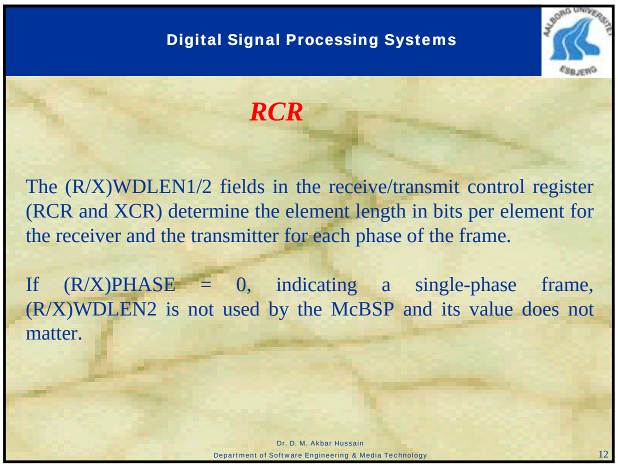
Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR.



RCR

31	30 24	23 21	20 19	18	17 16
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

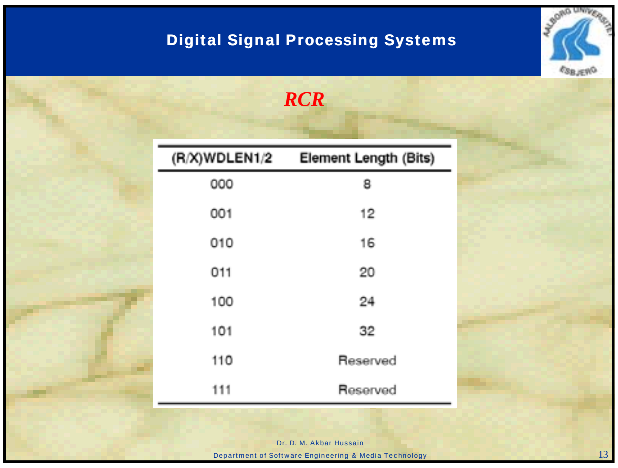
15	14 8	7 5	4	3 0
RPHASE2†	RFRLEN1	RWDLEN1	RWDREVRS†	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R-0



RCR

The (R/X)WDLEN1/2 fields in the receive/transmit control register (RCR and XCR) determine the element length in bits per element for the receiver and the transmitter for each phase of the frame.

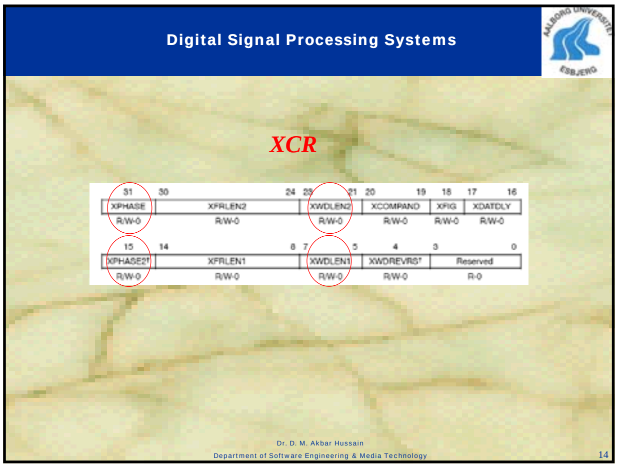
If (R/X)PHASE = 0, indicating a single-phase frame, (R/X)WDLEN2 is not used by the McBSP and its value does not matter.



Digital Signal Processing Systems

RCR

(R/X)WDLEN1/2	Element Length (Bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	Reserved
111	Reserved

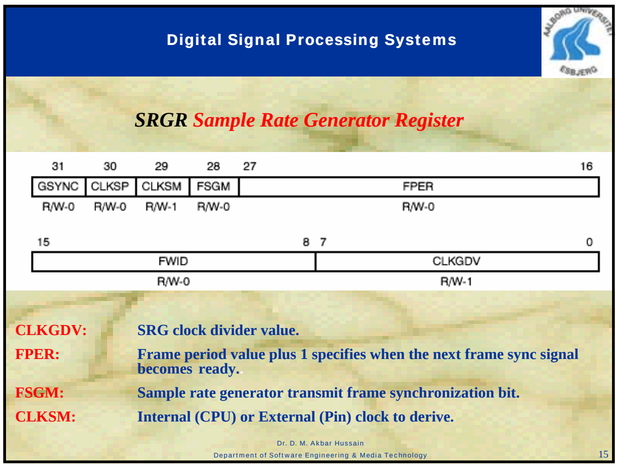


Digital Signal Processing Systems

XCR

31	30–24	23–21	20–19	18	17–16
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14–8	7–5	4	3–0
XPHASE2†	XFRLEN1	XWDLEN1	XWDREVRS†	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R-0



Digital Signal Processing Systems

SRGR *Sample Rate Generator Register*

31	30	29	28	27–16
GSYNC	CLKSP	CLKSM	FSGM	FPER
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0

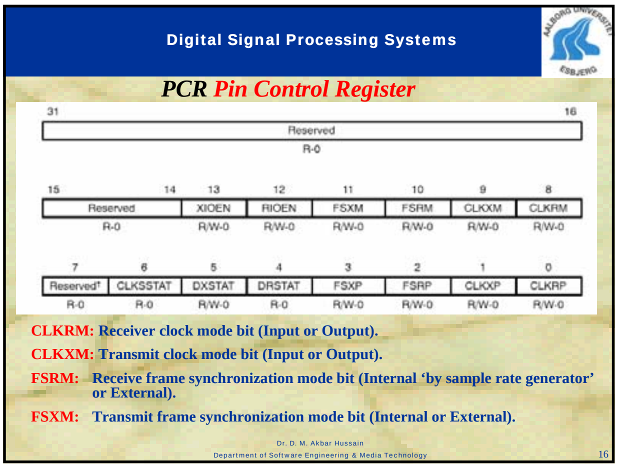
15–8	7–0
FWID	CLKGDV
R/W-0	R/W-1

CLKGDV: SRG clock divider value.

FPER: Frame period value plus 1 specifies when the next frame sync signal becomes ready.

FSGM: Sample rate generator transmit frame synchronization bit.

CLKSM: Internal (CPU) or External (Pin) clock to derive.



Digital Signal Processing Systems

PCR *Pin Control Register*

31–16
Reserved
R-0

15–14	13	12	11	10	9	8
Reserved	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

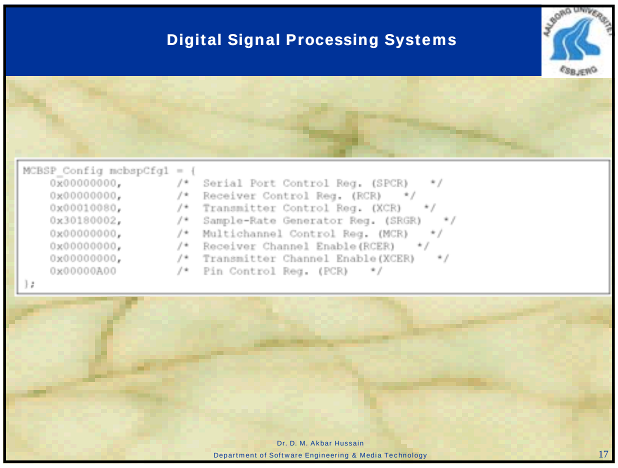
7	6	5	4	3	2	1	0
Reserved†	CLKSSTAT	DXSTAT	DRSTAT	FSXP	FSRP	CLKXP	CLKRP
R-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

CLKRM: Receiver clock mode bit (Input or Output).

CLKXM: Transmit clock mode bit (Input or Output).

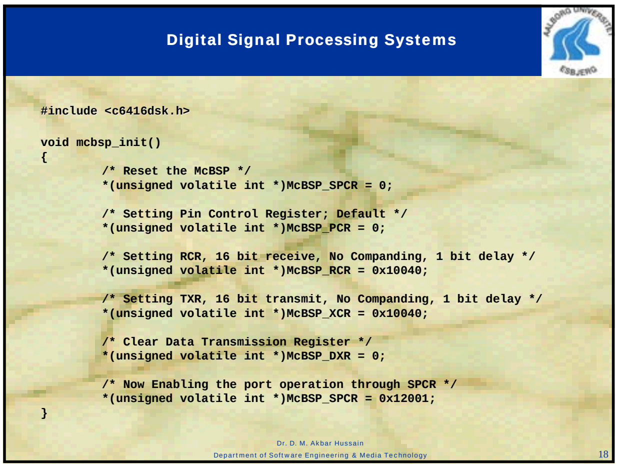
FSRM: Receive frame synchronization mode bit (Internal 'by sample rate generator' or External).

FSXM: Transmit frame synchronization mode bit (Internal or External).



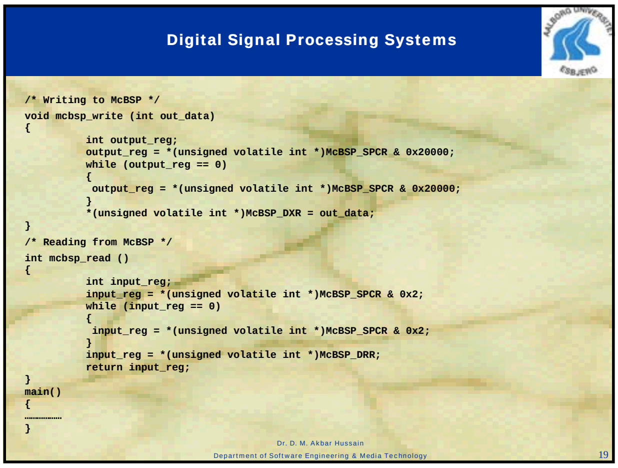
Digital Signal Processing Systems

```
MCBSP_Config mcbspCfg1 = {
    0x00000000,        /*  Serial Port Control Reg. (SPCR)   */
    0x00000000,        /*  Receiver Control Reg. (RCR)   */
    0x00010080,        /*  Transmitter Control Reg. (XCR)   */
    0x30180002,        /*  Sample-Rate Generator Reg. (SRGR)   */
    0x00000000,        /*  Multichannel Control Reg. (MCR)   */
    0x00000000,        /*  Receiver Channel Enable(RCER)   */
    0x00000000,        /*  Transmitter Channel Enable(XCER)   */
    0x00000A00         /*  Pin Control Reg. (PCR)   */
};
```



Digital Signal Processing Systems

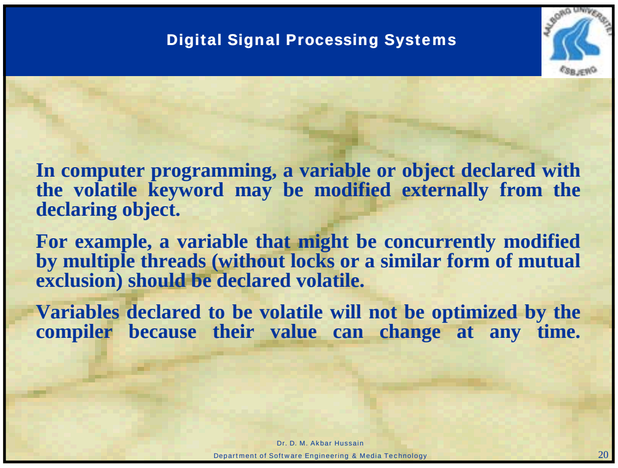
```
#include <c6416dsk.h>

void mcbsp_init()
{
        /* Reset the McBSP */
        *(unsigned volatile int *)McBSP_SPCR = 0;

        /* Setting Pin Control Register; Default */
        *(unsigned volatile int *)McBSP_PCR = 0;

        /* Setting RCR, 16 bit receive, No Companding, 1 bit delay */
        *(unsigned volatile int *)McBSP_RCR = 0x10040;

        /* Setting TXR, 16 bit transmit, No Companding, 1 bit delay */
        *(unsigned volatile int *)McBSP_XCR = 0x10040;

        /* Clear Data Transmission Register */
        *(unsigned volatile int *)McBSP_DXR = 0;

        /* Now Enabling the port operation through SPCR */
        *(unsigned volatile int *)McBSP_SPCR = 0x12001;
}
```



Digital Signal Processing Systems

```
/* Writing to McBSP */
void mcbsp_write (int out_data)
{
          int output_reg;
          output_reg = *(unsigned volatile int *)McBSP_SPCR & 0x20000;
          while (output_reg == 0)
          {
           output_reg = *(unsigned volatile int *)McBSP_SPCR & 0x20000;
          }
          *(unsigned volatile int *)McBSP_DXR = out_data;
}
/* Reading from McBSP */
int mcbsp_read ()
{
          int input_reg;
          input_reg = *(unsigned volatile int *)McBSP_SPCR & 0x2;
          while (input_reg == 0)
          {
           input_reg = *(unsigned volatile int *)McBSP_SPCR & 0x2;
          }
          input_reg = *(unsigned volatile int *)McBSP_DRR;
          return input_reg;
}
main()
{
………………
}
```

Dr. D. M. Akbar Hussain
Department of Software Engineering & Media Technology
19



Digital Signal Processing Systems

In computer programming, a variable or object declared with the volatile keyword may be modified externally from the declaring object.

For example, a variable that might be concurrently modified by multiple threads (without locks or a similar form of mutual exclusion) should be declared volatile.

Variables declared to be volatile will not be optimized by the compiler because their value can change at any time.

Dr. D. M. Akbar Hussain
Department of Software Engineering & Media Technology
20