Chapter 6

Multi-channel Buffered Serial Port (McBSP)
Objectives

- **Definition of Terms:**
  - Bit, word or channel, frame and phase.

- Understand basic serial port operation.

- Understand clock generation.

- Pin polarity.

- Serial port interrupts.

- Describe multi-channel operation.

- Programming the serial port.
Basic Definitions: Bits, Words?

- **Bit** - one data bit per SP clock period.
- **Word** or **channel** contains #bits specified by WDLEN1 (8, 12, 16, 20, 24, 32).

Serial Port
- SP Ctrl (SPCR)
- Rcv Ctrl (RCR)
- Xmt Ctrl (XCR)
- Rate (SRGR)
- Pin Ctrl (PCR)
**Basic Definitions: Frame?**

- **Frame** - contains one or multiple words
- **FRLEN1** specifies #words per frame (1-128)

![Diagram showing basic definitions and examples involving frames and words.](image)
Basic Definitions - Phase

FS

Data

- Note: dual-phase used in Audio Codec97 (AC97) Std

- Each FRAME can contain **only 1 or 2 PHASES** (PHASE).
- Each PHASE can contain different #bits (WDLEN1/2) and #words (FRLEN1/2).

![Diagram of data phases and frames](image-url)
Basic Definitions - Phase

Each FRAME can contain 1 or 2 PHASES (\textit{PHASE}).
Each PHASE can contain different \#bits (\textit{WDLEN1}/2) and \#words (\textit{FRLEN1}/2).

From above example some of the bit fields of RCR and XCR can be initialised as shown below.
Exercise

Fill in the control values for the example above.

<table>
<thead>
<tr>
<th>PHASE</th>
<th>RFRLEN2</th>
<th>RWDLEN2</th>
<th>RFRLEN1</th>
<th>RWDLEN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>24</td>
<td>23</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PHASE</th>
<th>XFRLEN2</th>
<th>XWDLEN2</th>
<th>XFRLEN1</th>
<th>XWDLEN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>24</td>
<td>23</td>
<td>21</td>
</tr>
</tbody>
</table>
Objectives

- **Definition of Terms:**
  - Bit, word or channel, frame and phase.

- **Understand basic serial port operation.**

- Understand clock generation.

- Pin polarity.

- Serial port interrupts.

- Describe multi-channel operation.

- Programming the serial port.
McBSP Block Diagram (Read)

CPU

Peripheral Bus

DMA

DR

RSR

32

RR

DMA

RING

CLKR

FSR

REV
McBSP Block Diagram (Configuration)

Multi-Channel Buffered Serial Port (McBSP)

- DR
- DX
- CLKR
- CLKX
- FSR
- FSX

Serial Port Control Logic
- SPCR
- RCR
- XCR

Peripheral Bus

Peripheral Bus

CPU

DMA

Dr. Naim Dahnoun, Bristol University, (c) Texas Instruments 2004
Serial Port - Basic Operation

Multi-Channel Buffered Serial Port (McBSP)

Peripheral Bus

CPU

DMA

Peripheral Bus

RBR

RSR

DRR

DXR

XSR

DR

DX

CLKR

CLKX

FSR

FSX

SPCR

RCR

SRGR

XCR

PCR

Chapter 6, Slide 13

Dr. Naim Dahnoun, Bristol University, (c) Texas Instruments 2004
# McBSP Registers (1)

<table>
<thead>
<tr>
<th>category</th>
<th>register</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive</td>
<td>RSR</td>
<td>Receive Shift Reg</td>
</tr>
<tr>
<td></td>
<td>RBR</td>
<td>Receive Buffer Reg</td>
</tr>
<tr>
<td></td>
<td>DRR</td>
<td>Data Receive Reg</td>
</tr>
<tr>
<td>Transmit</td>
<td>XSR</td>
<td>Transmit Shift Reg</td>
</tr>
<tr>
<td></td>
<td>DXR</td>
<td>Data Transmit Reg</td>
</tr>
<tr>
<td>Control</td>
<td>SPCR</td>
<td>Serial Port Control Reg</td>
</tr>
<tr>
<td></td>
<td>RCR</td>
<td>Receive Control Reg</td>
</tr>
<tr>
<td></td>
<td>XCR</td>
<td>Transmit Control Reg</td>
</tr>
</tbody>
</table>
Objectives

- Definition of Terms:
  - Bit, word or channel, frame and phase.

- Understand basic serial port operation.

- Understand clock generation.

- Pin polarity.

- Serial port interrupts.

- Describe multi-channel operation.

- Programming the serial port.
Configure CLK and FS as inputs or outputs

![Diagram of Multi-Channel Buffered Serial Port (McBSP)]

- FSR, FSX, CLKR and CLKX can be configured either as inputs or outputs, depending on the application.
Configure CLK and FS as inputs or outputs

- **Multi-Channel Buffered Serial Port (McBSP)**
- **Serial Port Control Logic**
  - **SPCR**
  - **RCR**
  - **SRGR**
  - **XCR**
  - **PCR**

**CLK/FS Mode**
- 0: Input
- 1: Output

**Serial Port**
- SP Ctrl (SPCR)
- Rcv Ctrl (RCR)
- Xmt Ctrl (XCR)
- Rate (SRGR)
- Pin Ctrl (PCR)

Chapter 6, Slide 17
Generating CLK and FS as output

'C6000

CLKR → FSRM → FSX → CLKX

\[ \begin{array}{cccc}
0 & 1 & 0 & 0 \\
\end{array} \]

Serial Port
- SP Ctrl (SPCR)
- Rcv Ctrl (RCR)
- Xmt Ctrl (XCR)
- Rate (SRGR)
- Pin Ctrl (PCR)

CLK/FS Mode
0: Input
1: Output
Generating the CLK as output

---

**‘C6000**

**Sample Rate Generator (SRGR)**

- **CLKSM** – selects clock src (CLKOUT1 or CLKS)
- **CLKGDV** - divide down (1-255)
- **CLKG** = (input clock) / (1 + CLKGDV)
- Max transfer rate = CLKG = 150 MHz/2 = 75 Mb/s

---

Serial Port

- SP Ctrl (SPCR)
- Rcv Ctrl (RCR)
- Xmt Ctrl (XCR)
- Rate (SRGR)
- Pin Ctrl (PCR)

---

Dr. Naim Dahnoun, Bristol University, (c) Texas Instruments 2004

Chapter 6, Slide 19
Generating the FS as output

'C6000

Sample Rate Generator (SRGR)

CLKSM

CLKGDV

CLKG

CLKX

CLKR

CLKS

CLKOUT1

FSR

FSX

FSG

FSH

FPER

FWID

FSGM

Serial Port

SP Ctrl (SPCR)
Rcv Ctrl (RCR)
Xmt Ctrl (XCR)
Rate (SRGR)
Pin Ctrl (PCR)

FSGM: 0 - FS gen’d on every DXR → XSR copy
       1 - FS gen’d by FSG

FPER: frame sync period (12 bits)

FWID: frame sync pulse width (8 bits)

29 28 27 16 15 8 7 0

CLKSM  FSGM  FPER  FWID  CLKGDV
### McBSP Registers (2)

<table>
<thead>
<tr>
<th>Receive</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RSR</td>
<td>Receive Shift Reg</td>
<td></td>
</tr>
<tr>
<td>RBR</td>
<td>Receive Buffer Reg</td>
<td></td>
</tr>
<tr>
<td>DRR</td>
<td>Data Receive Reg</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transmit</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XSR</td>
<td>Transmit Shift Reg</td>
<td></td>
</tr>
<tr>
<td>DXR</td>
<td>Data Transmit Reg</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SPCR</td>
<td>Serial Port Control Reg</td>
<td></td>
</tr>
<tr>
<td>RCR</td>
<td>Receive Control Reg</td>
<td></td>
</tr>
<tr>
<td>XCR</td>
<td>Transmit Control Reg</td>
<td></td>
</tr>
<tr>
<td>SRGR</td>
<td>Sample Rate Generator</td>
<td></td>
</tr>
</tbody>
</table>
Objectives

- **Definition of Terms:**
  - Bit, word or channel, frame and phase.
- **Understand basic serial port operation.**
- **Understand clock generation.**
- **Pin polarity.**
- **Serial port interrupts.**
- **Describe multi-channel operation.**
- **Programming the serial port.**
Configure CLK and FS pin polarity

Multi-Channel Buffered Serial Port (McBSP)

CLKR  CLGX  FSR  FSX

Serial Port Control Logic

SPCR  RCR  SRGR  XCR  PCR

CLK/FS Polarity
0: Falling edge
1: Rising Edge

Serial Port

SP Ctrl (SPCR)
Rcv Ctrl (RCR)
Xmt Ctrl (XCR)
Rate (SRGR)
Pin Ctrl (PCR)
Objectives

- Definition of Terms:
  - Bit, word or channel, frame and phase.
- Understand basic serial port operation.
- Understand clock generation.
- Pin polarity.
- Serial port status and interrupts.
- Describe multi-channel operation.
- Programming the serial port.
RRDY/XRDY Status and Interrupts

- **RRDY/XRDY** displays the “status” of the read and transmit ports:
  - 0: not ready.
  - 1: ready to read/write.

- There are 3 methods for detecting if data is ready:
  - Poll **SPCR** bits via s/w.
  - Config CPU ints (RINT/XINT).
  - Program DMA sync events.

**Serial Port**

- SP Ctrl (SPCR)
- Rcv Ctrl (RCR)
- Xmt Ctrl (XCR)
- Rate (SRGR)
- Pin Ctrl (PCR)
Other sources of Interrupts (R/XINT)

“Trigger Event”

RRDY (RINTM=00b)
End of Block (RCV) (RINTM=01b)
New FSR (frame begin) (RINTM=10b)
Receive Sync Error (RINTM=11b)

CPU

XRDY (XINTM=00b)
End of Block (XMT) (XINTM=01b)
New FSX (frame begin) (XINTM=10b)
Transmit Sync Error (XINTM=11b)

XINT

Serial Port

SP Ctrl (SPCR)
Rcv Ctrl (RCR)
Xmt Ctrl (XCR)
Rate (SRGR)
Pin Ctrl (PCR)

21 20 17 5 4 1

XINTM XRDY RINTM RRDY
Objectives

- Definition of Terms:
  - Bit, word or channel, frame and phase.
- Understand basic serial port operation.
- Understand clock generation.
- Pin polarity.
- Serial port status and interrupts.
- Describe multi-channel operation.
- Programming the serial port.
How do you enable/disable each channel?
Multi-Channel operation

◆ You can **enable** or **disable** any channel.

**RCER/XCER Enable Bits**

<table>
<thead>
<tr>
<th>Enable</th>
<th>Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>[0]</td>
</tr>
</tbody>
</table>

Chapter 6, Slide 29
Multi-channel example

- Allows multiple channels (words) to be independently selected for transmit and receive.
Multi-channel and EDMA combination used for channel sorting

- EDMA’s can sort each channel into separate buffers!

Dr. Naim Dahnoun, Bristol University, (c) Texas Instruments 2004
EDMA’s flexible (indexed) addressing allows it to sort each channel into separate buffers!

How do you select channels? ...
Enable/Disable Channels

- RCER / XCER registers allow you to enable or disable only 32-channels.
- So how does the C6000 support 128 channels?
To be able to support 128 channels the following applies:

- Channels are broken into BLOCK’s (16 contiguous channels).
- Up to 32 channels (2 BLOCK’s) can be enabled at any one time.
- Channels are enabled via _CER registers and _BLK bits in MCR.
- After 16 channels, McBSP issues \textit{END\_OF\_BLOCK} interrupt.
- CPU ISR re-programs RCER (or XCER) for channels 32-47 and so on.
## McBSP Registers (3)

<table>
<thead>
<tr>
<th>Receive</th>
<th>RSR</th>
<th>Receive Shift Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBR</td>
<td>Receive Buffer Reg</td>
</tr>
<tr>
<td></td>
<td>DRR</td>
<td>Data Receive Reg</td>
</tr>
<tr>
<td>transmit</td>
<td>XSR</td>
<td>Transmit Shift Reg</td>
</tr>
<tr>
<td></td>
<td>DXR</td>
<td>Data Transmit Reg</td>
</tr>
<tr>
<td>control</td>
<td>SPCR</td>
<td>Serial Port Control Reg</td>
</tr>
<tr>
<td></td>
<td>RCR</td>
<td>Receive Control Reg</td>
</tr>
<tr>
<td></td>
<td>XCR</td>
<td>Transmit Control Reg</td>
</tr>
<tr>
<td></td>
<td>SRGR</td>
<td>Sample Rate Generator</td>
</tr>
<tr>
<td></td>
<td>PCR</td>
<td>Pin Control Reg</td>
</tr>
<tr>
<td></td>
<td>MCR</td>
<td>Multi-Channel Ctrl Reg</td>
</tr>
<tr>
<td></td>
<td>RCER</td>
<td>Rcv Channel Enable Reg</td>
</tr>
<tr>
<td></td>
<td>XCER</td>
<td>Xmit Channel Enable Reg</td>
</tr>
</tbody>
</table>
Objectives

- **Definition of Terms:**
  - Bit, word or channel, frame and phase.

- Understand basic serial port operation.

- Understand clock generation.

- Pin polarity.

- Serial port status and interrupts.

- Describe multi-channel operation.

- Programming the serial port.
Programming the Serial Port

- There are three methods available for programming the serial port:
  1. Writing directly to the serial port registers.
  2. Using the Chip Support Library (CSL).
  3. Graphically using the DSP/BIOS GUI configuration tool.
(A) Writing directly to the serial port registers:

- Although this method is straightforward, it relies on a good understanding of the serial port functionality.
- This method can be tedious and is prone to errors.

```c
#include <c6211dsk.h>

void mcbsp0_init()
{
    *(unsigned volatile int *)McBSP0_SPCR = 0;
    *(unsigned volatile int *)McBSP0_PCR = 0;
    *(unsigned volatile int *)McBSP0_RCR = 0x10040;
    *(unsigned volatile int *)McBSP0_XCR = 0x10040;
    *(unsigned volatile int *)McBSP0_DXCR = 0;
    *(unsigned volatile int *)McBSP0_SPCR = 0x12001;
}
```
(B) Using the Chip Support Library:

- The CSL provides a C language interface for configuring and controlling the on-chip peripherals, in this case the Serial Ports.
- The library is modular with each module corresponding to a specific peripheral. This has the advantage of reducing the code size.
- Some modules rely on other modules also being included, for example the IRQ module is required when using the EDMA module.
CSL programming procedure:

1. Create handles for the serial ports:

```
MCBSP_Handle hMcbsp;
```

2. Open the serial port:

```
hMcbsp = MCBSP_open(MCBSP_DEV1, MCBSP_OPEN_RESET);
```
CSL programming procedure:

3. Create a configuration structure for serial port:
   - \Links\McBSP_Config_Struct.pdf
CSL programming procedure (cont):

(4) Configure the serial port:

```
MCBSP_config(hMcbsp,&ConfigLoopback);
```

(5) Close the Serial Port after use:

```
MCBSP_close(hMcbsp);
```
Practical example on DSP Code 6711

- **Project name:** mcbsp_dynamiccfg.pjt
- **Location:** \Code\Chapter 06 - McBSP\Dynamic_CSL_Config\
Programming the Serial Port using the DSP/BIOS GUI

(C) DSP/BIOS GUI Interface:

- With this method the configuration structure is created graphically and the setup code is generated automatically.
Programming the Serial Port using the DSP/BIOS GUI

Procedure:

1. Create a configuration using the MCBSP Configuration manager (e.g., mcbspCfg0).
Programming the Serial Port using the DSP/BIOS GUI

Procedure:

(2) Right click on mcbspCfg0 and select “Properties”, see figures below, and then select “Advanced” and fill all parameters as shown below:
Programming the Serial Port using the DSP/BIOS GUI

Procedure:

3) Select the serial port you would like to use from the MCBSP Resource manager (eg. Mcbsp_Port1).

Right click and select properties. Select the mcbspCfg0 configuration just created.
Programming the Serial Port using the DSP/BIOS GUI

Procedure:

(4) A file is then generated that contains the configuration code. The file generated for this example is shown on the next slide.
Programming the Serial Port using the DSP/BIOS GUI

/* Do *not* directly modify this file. It was
   generated by the Configuration Tool; any
   changes risk being overwritten. */

/* INPUT mcbsp1.cdb */

/* Include Header File */
#include "mcbsp1cfg.h"

/* Config Structures */
MCBSP_Config mcbspCfg0 = {
  0x00008000,        /*  Serial Port Control Reg. (SPCR) */
  0x000000A0,        /*  Receiver Control Reg. (RCR) */
  0x000000A0,        /*  Transmitter Control Reg. (XCR) */
  0x203F1F0F,        /*  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) */
};

/* Handles */
MCBSP_Handle hMcbsp1;

/*
  * ------ CSL_cfgInit() ------
  */
void CSL_cfgInit()
{
  hMcbsp1 = MCBSP_open(MCBSP_DEV1, MCBSP_OPEN_RESET);
  MCBSP_config(hMcbsp1, &mcbspCfg0);
}
Programming the Serial Port using the DSP/BIOS GUI

Few remarks:

(1) Notice that values in the code generated are the same as the values inserted using the GUI interface.

```c
/* Do *not* directly modify this file. It was */
/* Config Structures */
MCBSP_Config mcbspCfg0 = {
  0x00008000, /* Serial Port Control Reg. (SPCR) */
  0x000000A0, /* Receiver Control Reg. (RCR) */
  0x000000A0, /* Transmitter Control Reg. (XCR) */
  0x203F1F0F, /* 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) */
};
```
Few remarks:

(2) Do not forget to close the serial port after use.

(3) To visualise the output of the logprintf() function make sure that the Message Log window is open (DSP/BIOS > Message Log).
Project name: mcbsp_staticcfg.pjt
Location: \Code\Chapter 06 - McBSP\Static_CSL_Config\ Extra Topic: Digital Loopback
Chapter 6
Multi-channel Buffered Serial Port (McBSP)
- End -
 Allows testing of the Serial Port code without the need of an external device.

 Digital Loopback internally connects the rcv/xmt ports together as shown.

 No hardware (pin connections) necessary.

 Interrupts are generated as normal (as programmed).
Digital Loopback (DLB)

- You can set the digital loop back by setting a bit in the SPCR or graphically using the GUI interface as shown: