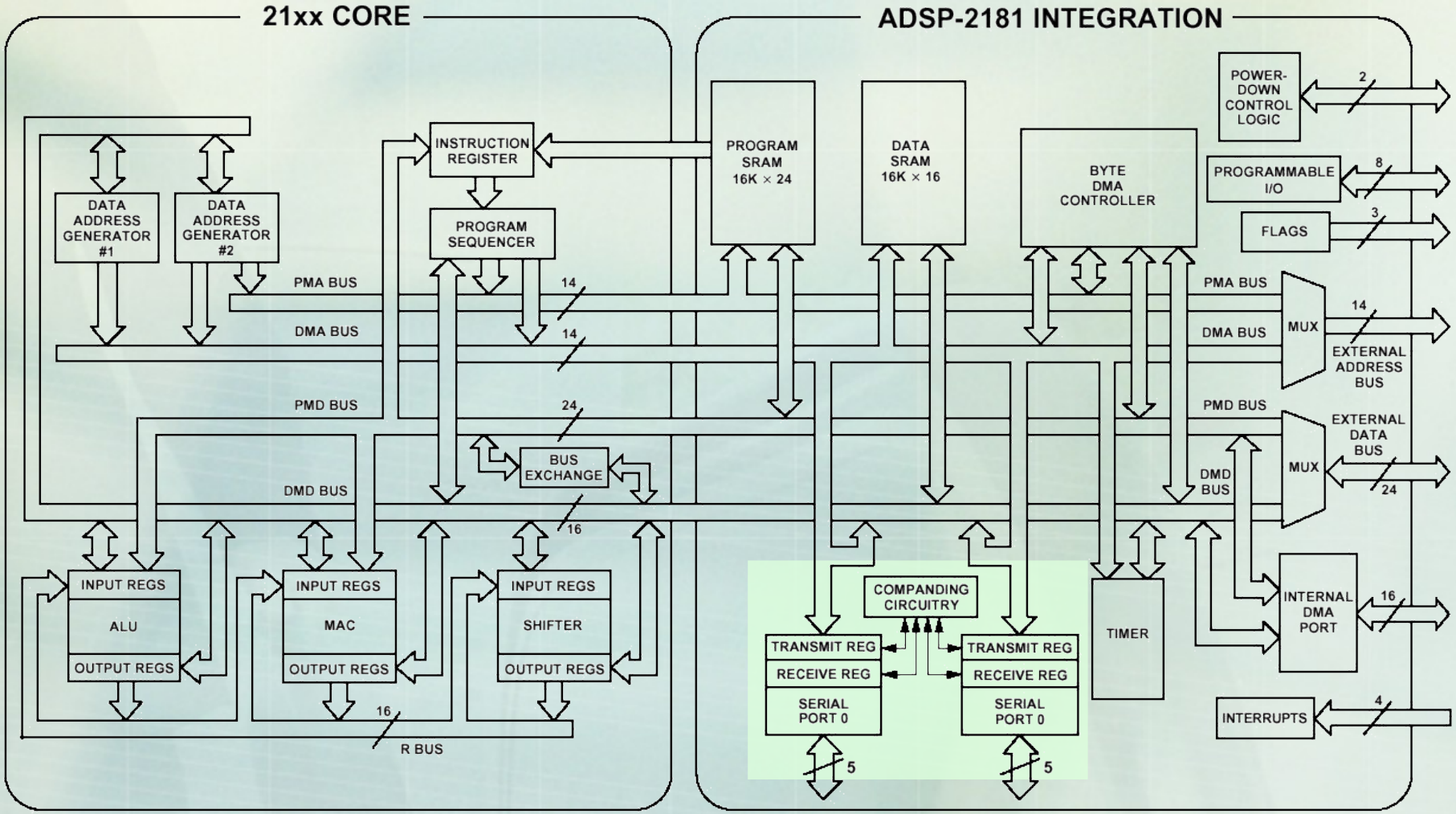
Real time DSP

Professors:

- Eng. Diego Barral
- Eng. Mariano Llamedo Soria
- Julian Bruno

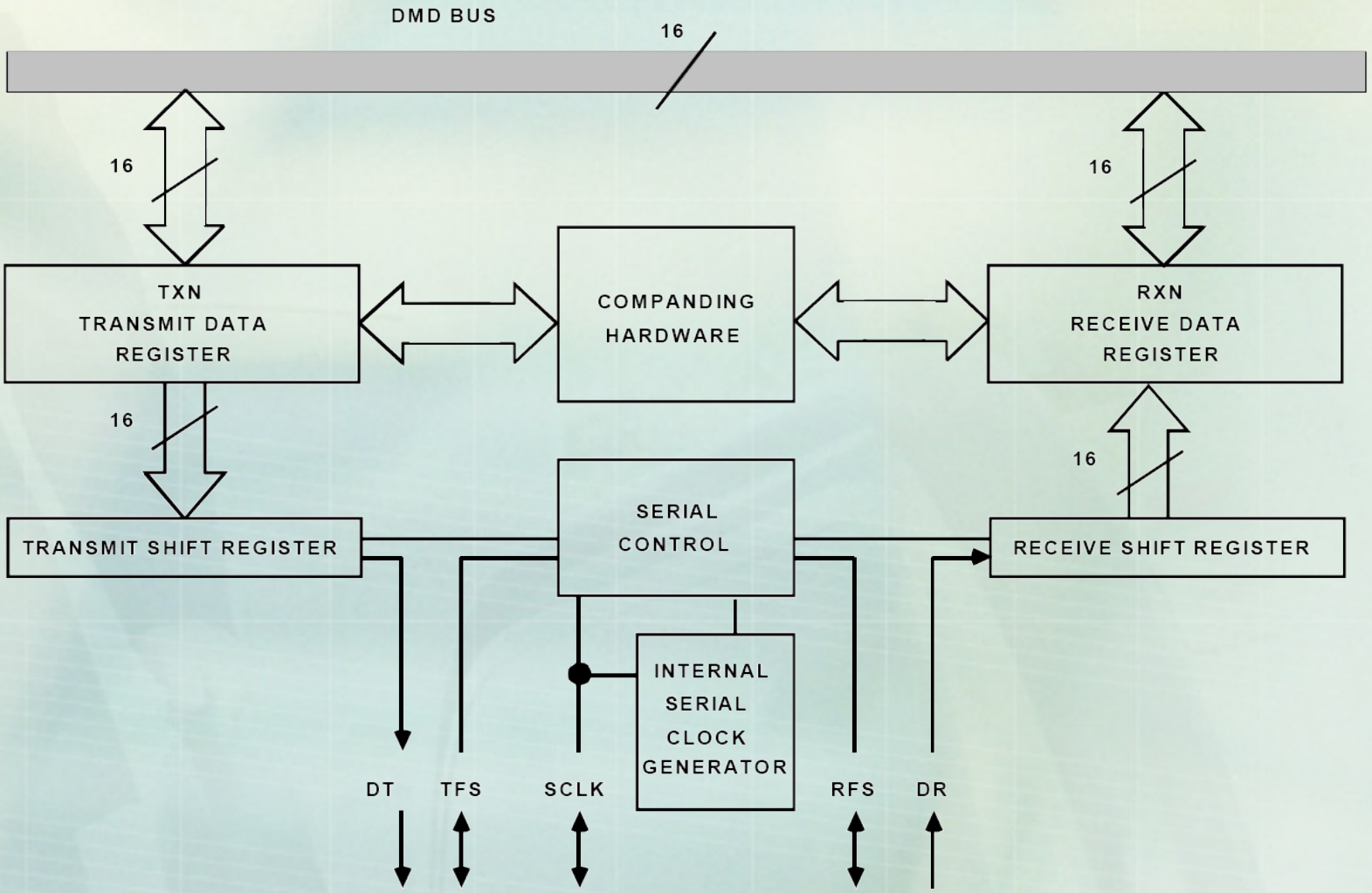
ADSP 2181


21xx CORE
ADSP-2181 INTEGRATION
DATA ADDRESS GENERATOR #1
DATA ADDRESS GENERATOR #2
INSTRUCTION REGISTER
PROGRAM SEQUENCER
PROGRAM SRAM 16K × 24
DATA SRAM 16K × 16
BYTE DMA CONTROLLER
POWER-DOWN CONTROL LOGIC
PROGRAMMABLE I/O
FLAGS
PMA BUS
DMA BUS
PMD BUS
DMD BUS
BUS EXCHANGE
MUX
EXTERNAL ADDRESS BUS
EXTERNAL DATA BUS
INPUT REGS
ALU
OUTPUT REGS
MAC
SHIFTER
R BUS
COMPANDING CIRCUITRY
TRANSMIT REG
RECEIVE REG
SERIAL PORT 0
TIMER
INTERNAL DMA PORT
INTERRUPTS

Serial Ports

- 2 Serial Ports
  - SPORT0 (Multichannel & Alternate configuration)
  - SPORT1
- Pin Name Function
  - SCLK Serial clock
  - RFS Receive frame synchronization
  - TFS Transmit frame synchronization
  - DR Serial data receive
  - DT Serial data transmit

Serial Port Block Diagram



SPORT characteristics

- Bidirectional: full duplex
- Double-buffered: receive and transmit
- Clocking: int or ext
- Word length: 3 to 16 bits
- Frame synchronization
- Companding in hardware (A-law and μ-law)
- Autobuffering: Using the DAGs, each SPORT can automatically receive and/or transmit an entire circular buffer of data with an overhead of only one cycle per data word
- Interrupts: receive and transmit
- Multichannel: time-division multiplexed into 24 or 32 channels
- Alternate configuration: IRQ0, IRQ1, Flag In, Flag Out

Interrupts

- Priority
  - Highest
    - SPORT0 Transmit
    - SPORT0 Receive
    - SPORT1 Transmit
  - Lowest
    - SPORT1 Receive
- Programming
  - Control registers - memory mapped
  - Data section - register Tx & Rx

Configuration Registers

Address	Contents
0x3FFA	SPORT0 multichannel receive word enables (HI)
0x3FF9	SPORT0 multichannel receive word enables (LO)
0x3FF8	SPORT0 multichannel transmit word enables (HI)
0x3FF7	SPORT0 multichannel transmit word enables (LO)
0x3FF6	SPORT0 control register
0x3FF5	SPORT0 serial clock divide modulus (freq)
0x3FF4	SPORT0 receive frame sync divide modulus (freq)
0x3FF3	SPORT0 autobuffer control register
0x3FF2	SPORT1 control register
0x3FF1	SPORT1 serial clock divide modulus (freq)
0x3FF0	SPORT1 receive frame sync divide modulus (freq)
0x3FEF	SPORT1 autobuffer control register

Receiving And Transmitting Data

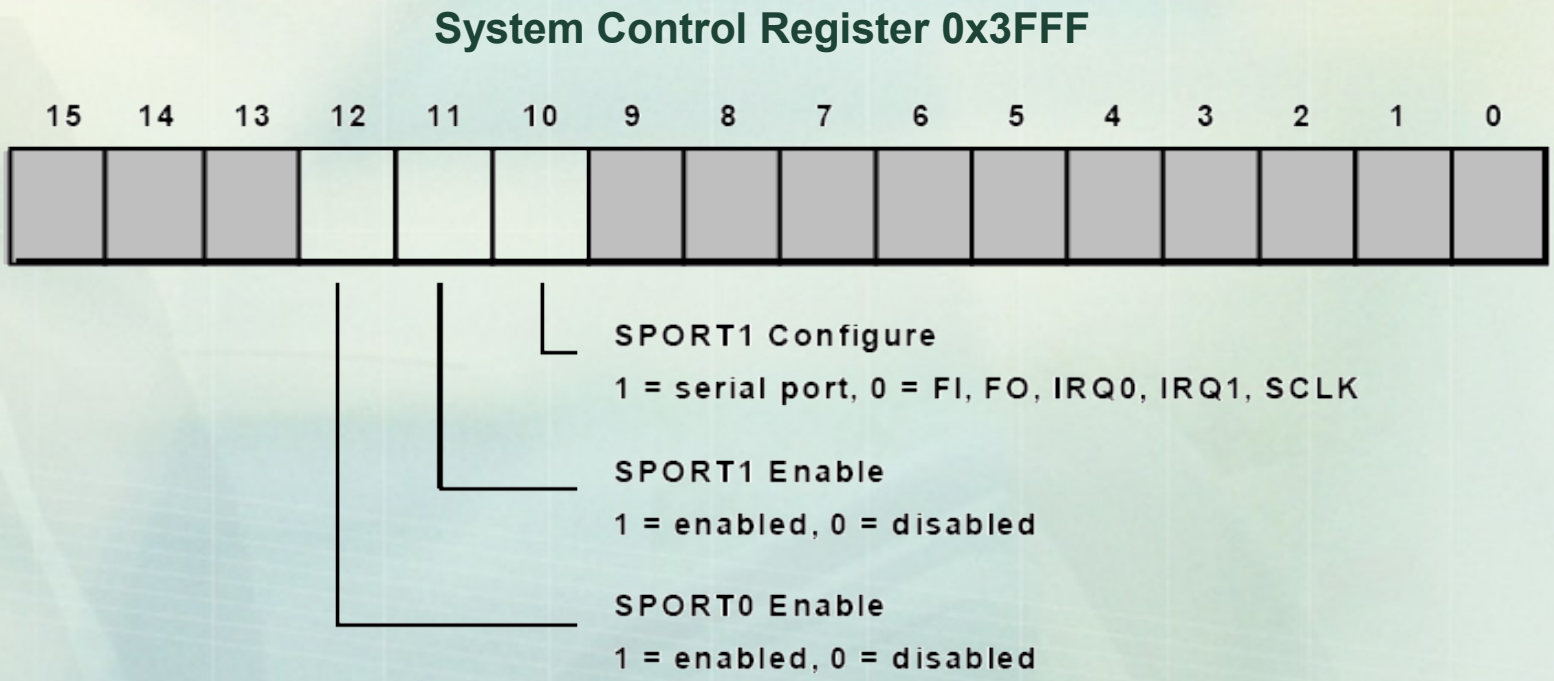
- Each SPORT has a receive register and a transmit register
  - RX0, TX0, RX1, TX1

```
TX1 = AX0;      /* the contents of AX0 are transmitted on SPORT1
*/

AY0 = RX0;      /* the contents of SPORT0 receive register is
                   transferred to AY0 */
```

(typically within an interrupt service routine)

Sport Enable

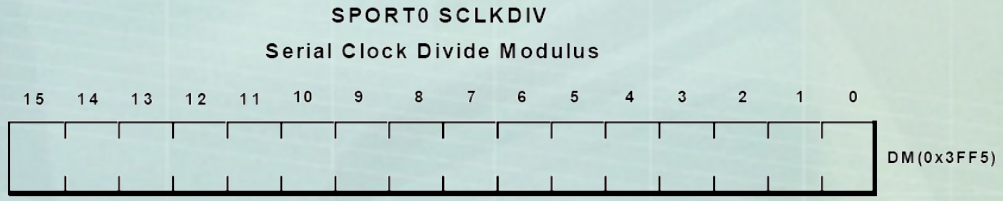


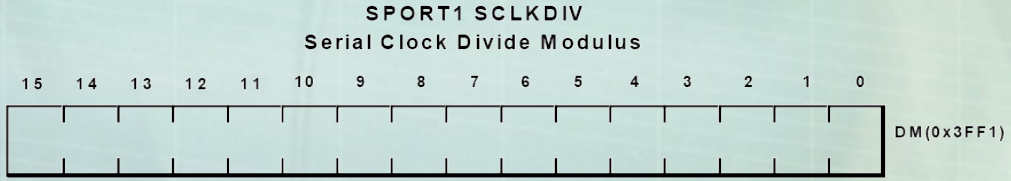
Pin Name	Alternate Name	Alternate Function
RFS1	IRQ0	External interrupt 0
TFS1	IRQ1	External interrupt 1
DR1	FI	Flag input
DT1	FO	Flag output
SCLK1	Same	Same

Serial Clock

- External serial clock frequencies may be as high as the processor's cycle rate, up to a maximum of 13.824 MHz
- Internal clock frequencies may be as high as one-half the processor's clock rate

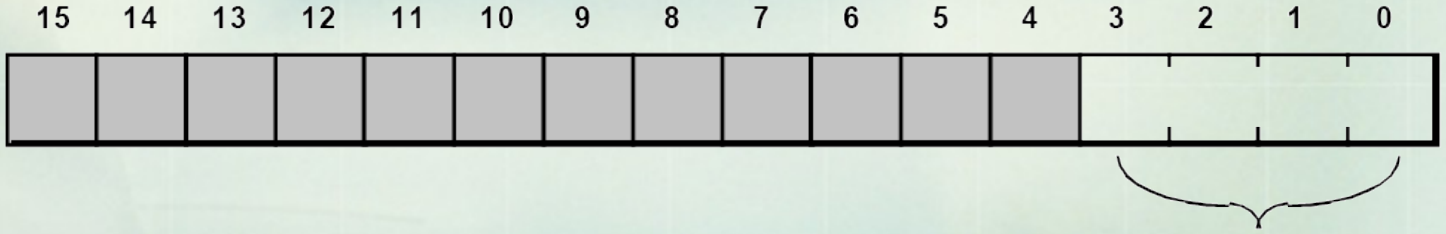
$$\text{SCLK frequency} = \frac{\text{CLKOUT frequency}}{2 \times (\text{SCLKDIV} + 1)}$$





Word Length & Frame Synchronization

SPORT0 Control Register: 0x3FF6
SPORT1 Control Register: 0x3FF2



SLEN (Serial Word Length – 1)

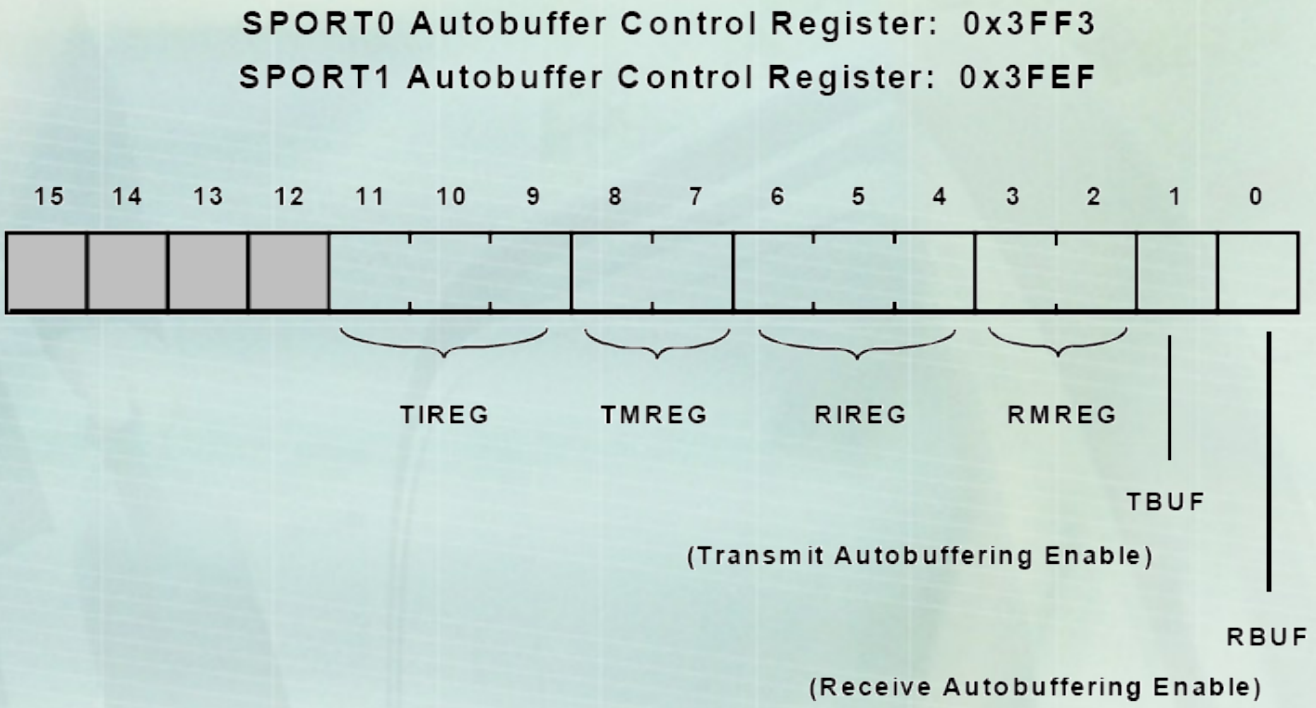
SPORT0 Control Register: 0x3FF6
SPORT1 Control Register: 0x3FF2

TFSR 0 = Transmit Frame Sync Required 1st Word
1 = Transmit Frame Sync Required Every Word

RFSR 0 = Receive Frame Sync Required 1st Word
1 = Receive Frame Sync Required Every Word

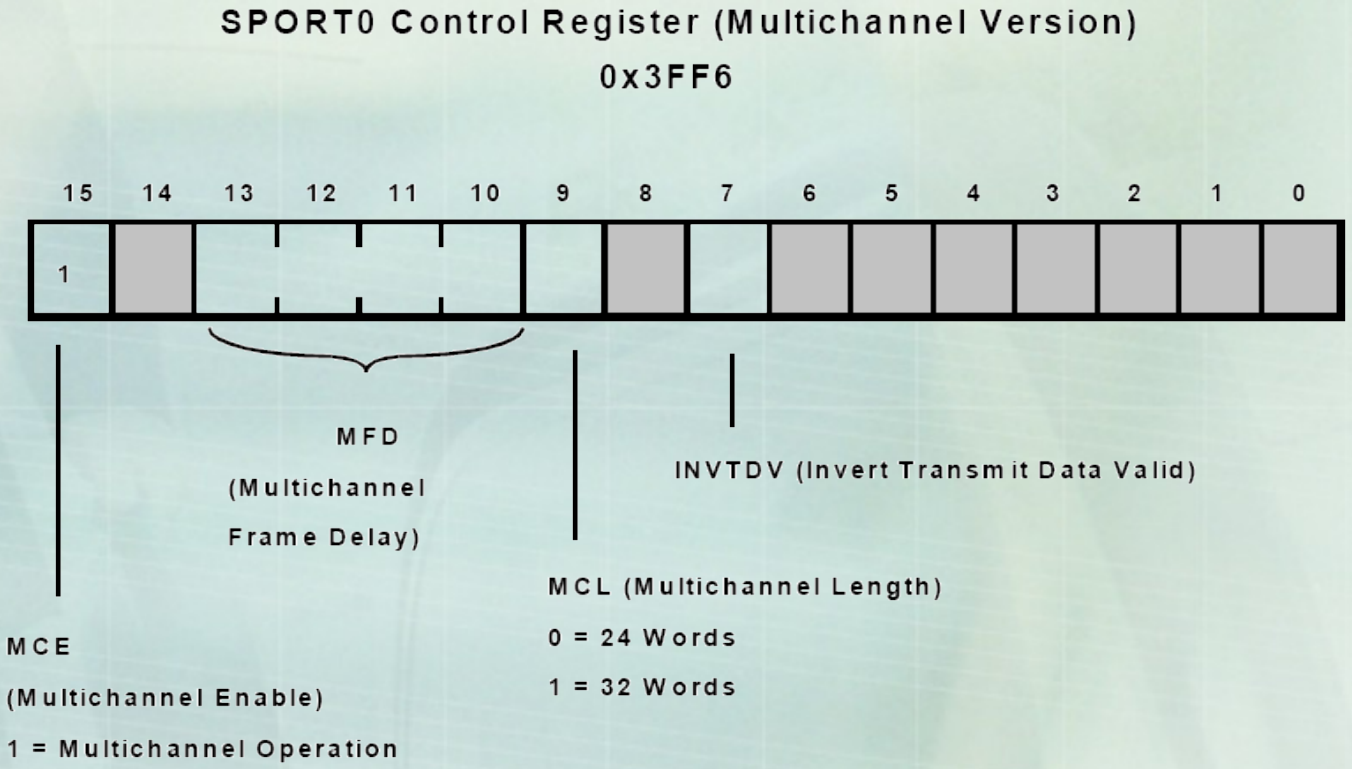
Autobuffering

- Receiving or transmitting an entire block of serial data before an interrupt is generated
- Uses the circular buffer (in Data Memory) addressing capability
- Interrupts on “wrap around”



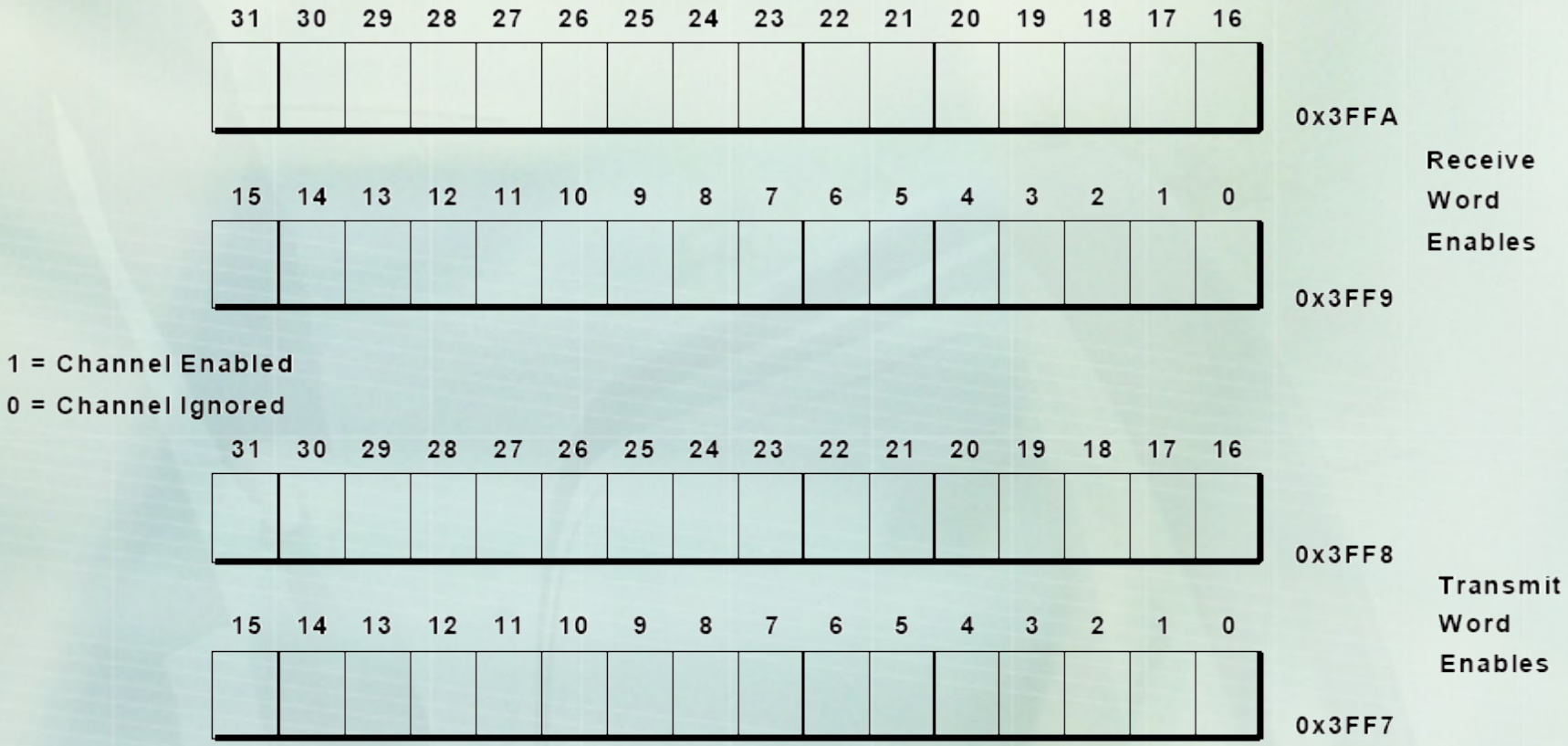
Multichannel

- serial data is time-division multiplexed
- SPORT0
- 32 or 24 channels



Multichannel

The multichannel frame delay (MFD) is a 4-bit field specifying (in binary) the number of serial clock cycles between the frame sync signal and the first data bit.



Autobuffering Example Configuration Code (I)

```
/* Initialization code for autobuffer */

.SECTION/DM    data1;
        .VAR/CIRC tx_buffer[10];
        .VAR/CIRC rx_buffer[10];
        .SECTION/PM program;

.global sport1_inits;

sport1_inits:
      I0 = tx_buffer;                 /* I0 contains address of tx_buffer */
      M0 = 1;                         /* fill every location */
      L0 = length (tx_buffer);        /* L0 set to length of tx_buffer */
      I1 = rx_buffer;                 /* I1 points to rx_buffer */
      L1 = length (rx_buffer);        /* L1 set to length of rx_buffer */

      /* set up SPORT1 for autobuffering */
      AX0 = 0x0013;                   /* TX uses I0, M0; RX uses I1, M0 */
      DM(0x3FEF) = AX0;               /* autobuffering enabled */
```

Autobuffering Example Configuration Code (II)

```
/* set up SPORT1 for 8 kHz sampling and 2.048 MHz SCLK */
AX0 = 255;                      /* set RFSDIV to 255 for 8 kHz */
DM(0x3FF0) = AX0;
AX0 = 17;                       /* set SCLKDIV to 17 for 2.048 MHz SCLK */
DM(0x3FF5) = AX0;

/* set up SPORT1 for normal required framing, internal SCLK internal generated framing */
AX0 = 0x6B27;                   /* normal framing, 8 bit mu-law */
DM(0x3FF2) = AX0;               /* internal clock, framing */

/* set up interrupts */
IFC = 6;                        /* clear any extraneous SPORT interrupts*/
ICNTL = 0;                      /* interrupt nesting disabled */
IMASK = 6;                      /* enable SPORT1 interrupts */

/* enable SPORT1 */
AX0 = 0x0C1F;                   /* enable SPORT1 leave */
DM(0x3FFF) = AX0;               /* PWAIT, BWAIT as default */

/* Place first transfer value into TX1 */
AX0 = DM(I0,M0);
TX1 = AX0;

RTS;
```