‘C6x McBSP Initialization

APPLICATION REPORT: PRELIMINARY

Shaku Anjanaiah

Digital Signal Processing Solutions
June 1998

TEXAS
INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI’s standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage (“Critical Applications”).

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer’s applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

‘C6x McBSP Initialization

Abstract

The ‘C6x Multi-channel Buffered Serial Port (McBSP) can be programmed to operate in different modes as per application requirements. For proper operation, the serial port has to be initialized in a certain order after device power up.

This document describes the initialization steps necessary when either the DMA or the CPU is used to service the McBSP data. Typically the DMA is used to perform read/write transfers from/to the McBSP. The DMA transfers are read/write synchronized and the McBSP provides these sync events. In cases where the CPU is used to read from DRR and write to DXR, either the polled or interrupt method can be used. The McBSP transmit and receive interrupts can be programmed to trigger on any one of the four conditions that are allowed.

Product Support

Related Documentation

The following list specifies product names, part numbers, and literature numbers of relevant TI documentation.

- ☐ *TMS320C6201 Digital Signal Processor* data sheet, March 1998, Literature number SPRS051C
- ☐ *TMS320C6201/C6701 Peripherals* Reference Guide, March 1998, Literature number SPRU190B

World Wide Web

Our World Wide Web site at **www.ti.com** contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to **dsph@ti.com**. Questions receive prompt attention and are usually answered within one business day.

Design Problem

How do I initialize the McBSP for correct frame sync generation and data exchange? What are the important initialization steps and their order of execution?

McBSP Introduction

The main functional blocks of the McBSP are shown in Figure 1. They are:

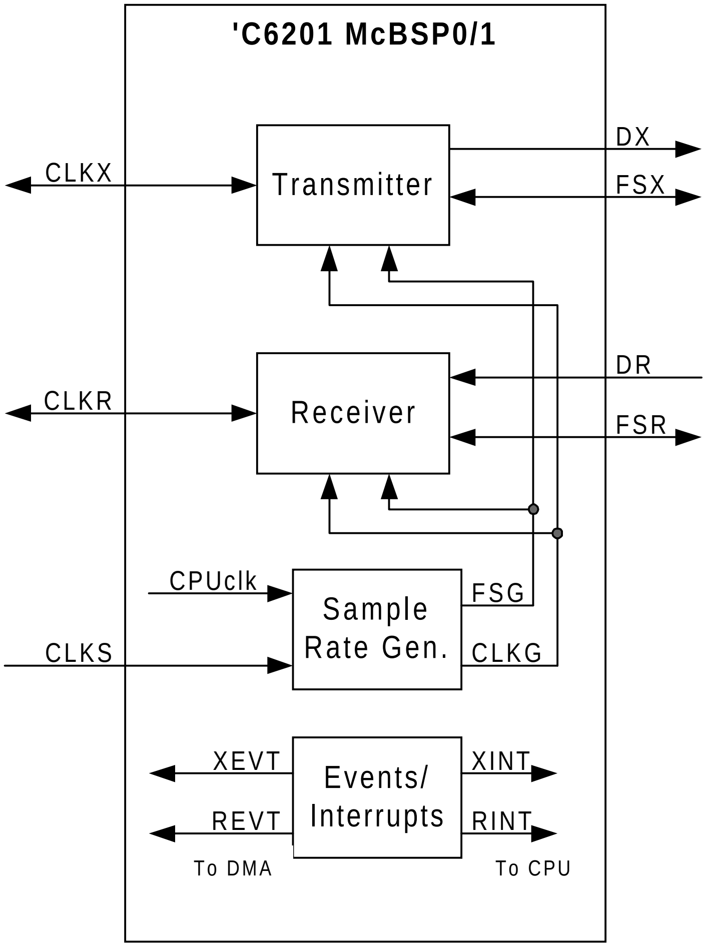
- **Transmitter:** The transmitter section is responsible for the serial transmission of data that is written in DXR. The transfer starts as soon as the transmit frame sync (FSX) is detected. One bit of data is transmitted on every transmit clock CLKX.

- **Receiver:** The data received on the DR pin is shifted into the Receive Shift Register (RSR) on every receive clock (CLKR). Again, the actual shifting in of data begins after detection of a receive frame sync (FSR).

- **Sample Rate Generator:** As the name implies, this module generates control signals such as the transmit/receive clocks and frame sync signals necessary for data transfer to and from the McBSP. Clock generation circuitry allows user to choose either the CPU clock or an external source via CLKS to generate CLKR/X. Frame sync signal properties such as frame period and frame width are also programmable. FSR/X, CLKR/X are bi-directional pins, and therefore can be inputs or outputs.

- **Events/Interrupt Generation:** The McBSP generates sync events to the DMA to indicate that data is ready in DRR or that DXR is ready for new data. They are read sync event REVT, and write sync event XEVT. Similarly the CPU can read/write to the McBSP based on interrupts (RINT and XINT) generated by the McBSP.

Figure 1. McBSP Functional Block Diagram



Servicing the McBSP

The McBSP data has to be read/written as and when required. The ‘C6x can service the McBSP via the CPU or the DMA. All control registers have to be programmed via the CPU. But, the data registers, DXR and DRR can be accessed either by the CPU or the DMA. Typically, DMA channel(s) is used to read/write to the data registers, thus relieving the CPU from servicing a slow peripheral. For more details of DMA servicing a peripheral, please refer *TMS320C6201 DMA Applications* application note.

- **Via DMA:** DMA write accesses to DXR require write synchronization event, XEVT, which is provided by the McBSP. Similarly a DMA read of DRR is synchronized by the internal REVT signal from the McBSP. Therefore, the DMA will read from or write to the McBSP for every serial element in the frame. Once the DMA completes the required number of element transfers, it can be programmed to generate a channel complete interrupt to the CPU if required.

- **Via CPU:** For cases where the CPU is used to service the McBSP, it can be done in either interrupt-driven mode or the polled method. Polled method ties up the CPU while waiting for data to be transmitted or received. The (R/X)RDY bits are polled for receive/transmit ready condition of the McBSP.

  In the interrupt-driven case, the CPU can do useful work while the interrupts from the McBSP signal the CPU when it needs to be served. The default value of (R/X)INTM = 00b causes the McBSP to interrupt the CPU via (R/X)INT on every element transfer (if the CPU interrupts are enabled). The XINT is generated when DXR is ready to accept new data, and RINT is generated when a new serial element has been received in DRR. Other interrupt modes are not meant for servicing the McBSP for data reads/writes, but for diagnostic, and tracking purposes.

Initialization Requirements

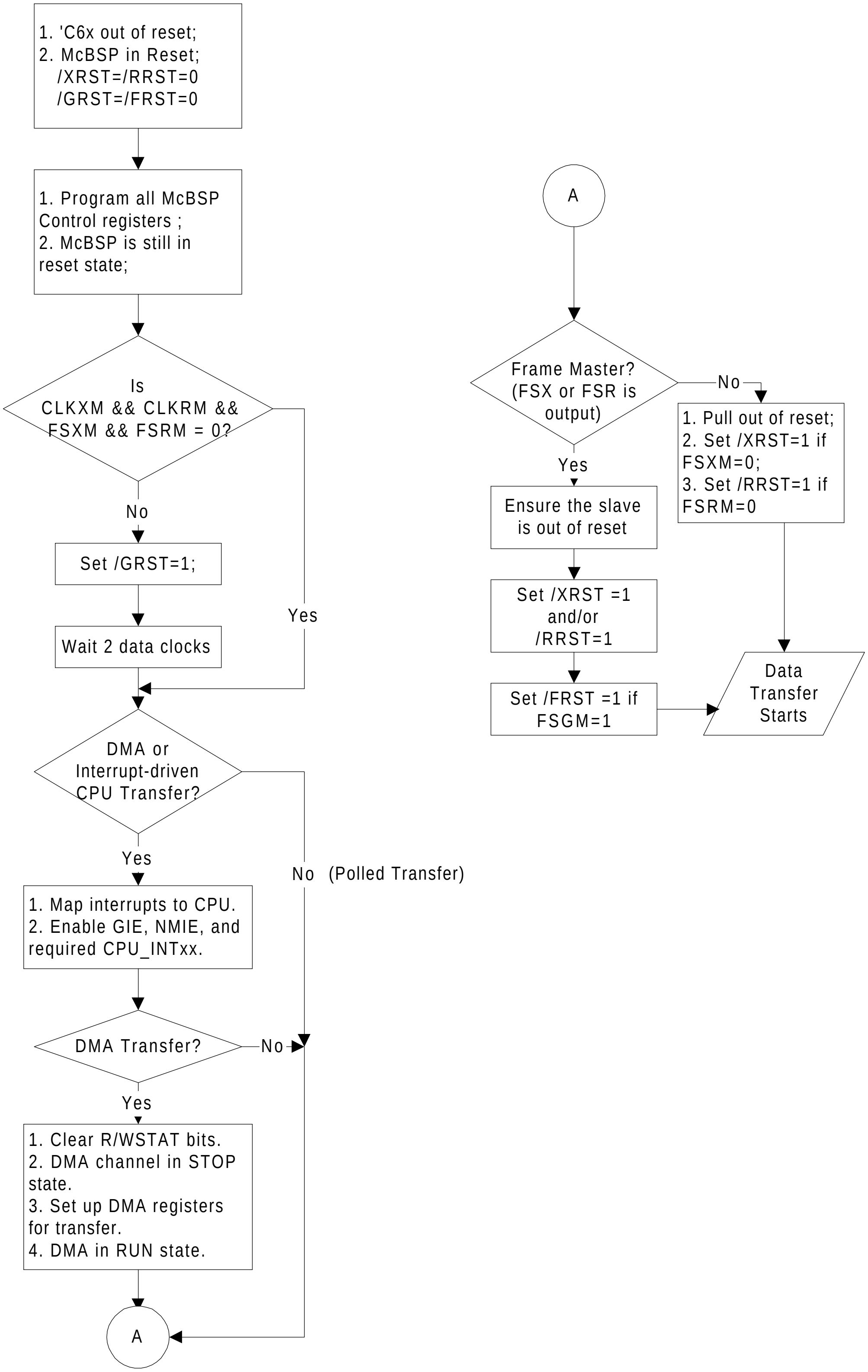
Generation of control signals such as the clock, frame sync, clock source in the McBSP is programmable. The order in which the respective modules are activated is important for correct operation of the McBSP.

For example, consider the case when the transmitter is the clock and frame master meaning it is responsible for generation of clocks and frames for itself and to the device (receiver) it is communicating to. The first step is to ensure that the slave (in this case the receiver) is awake (taken out of reset) and is ready to receive frame and data from the transmitter. This is followed by taking the transmitter out of reset, and then activating the frame sync generator in the transmitter. This ensures that the receiver does not lose the first frame and its data.

McBSP Initialization Cases

This section describes the step-by-step procedure for McBSP initialization based on CPU or DMA data transfers. The following three methods and the initialization procedure is summarized in a flowchart shown in Figure 2.

Figure 2. McBSP Initialization Flowchart


1. 'C6x out of reset;
2. McBSP in Reset;
/XRST=/RRST=0
/GRST=/FRST=0
1. Program all McBSP Control registers ;
2. McBSP is still in reset state;
Is CLKXM && CLKRM && FSXM && FSRM = 0?
No
Yes
Set /GRST=1;
Wait 2 data clocks
DMA or Interrupt-driven CPU Transfer?
Yes
No (Polled Transfer)
1. Map interrupts to CPU.
2. Enable GIE, NMIE, and required CPU_INTxx.
DMA Transfer?
No
Yes
1. Clear R/WSTAT bits.
2. DMA channel in STOP state.
3. Set up DMA registers for transfer.
4. DMA in RUN state.
A
A
Frame Master? (FSX or FSR is output)
No
Yes
1. Pull out of reset;
2. Set /XRST=1 if FSXM=0;
3. Set /RRST=1 if FSRM=0
Ensure the slave is out of reset
Set /XRST =1 and/or /RRST=1
Set /FRST =1 if FSGM=1
Data Transfer Starts

- **DMA Transfers:** The following steps describe the setup of interrupts, DMA, and the McBSP in the required order.

1. Set /GRST=/XRST=/RRST=/FRST=0. If coming out of device reset, this is not required.

2. Program the Sample Rate Generator Register (SRGR), Serial Port Control Register (SPCR), Pin Control Register (PCR), and Receive Control Register (RCR) to the required values.

   Caution: Do not set the bits described in Step 1 while setting up registers.

3. Take the sample rate generator out of reset by setting /GRST=1 in the SPCR. Internal clock CLKG is now driven by the chosen clock source and as per programmed clock divide-down.

   Note: If frame syncs and clocks are inputs to both transmit and receive sections of the McBSP, this step is not required.

4. Wait 2 bit clocks (CLKR/X). A simple formula to arrive at this number in terms of CPU clock cycles is:

   (a) For CPU clock as source: Let P=(1/CPUclock).

   The number of CPU clocks equal to 2 data bit-clocks is,

   N = (1+CLKGDV) * 2, where min. value of CLKGDV=1

   (b) For CLKS as clock source: Let P_s=(1/CLKS frequency) and P=(1/CPUclock).

   The number of CPU clocks equal to 2 data bit-clocks is,

   N = (1+CLKGDV) * 2 * (P_s /P), where min. value of CLKGDV=0 and CLKS is not greater than (CPUclock/2)

   In general, the following formula can be used depending on the clock source:

   N = ((1+CLKGDV) * 2 * CLKSM) +

   ((1+CLKGDV) * 2 * (P_S /P) * (!CLKSM))

5. Enabling Interrupts: To use interrupts, you have to set the Global Interrupt Enable (GIE), and Non-Maskable Interrupt Enable (NMIE) bits in the IER.

   Select the DMA channel you want to use. Enable CPU interrupts that correspond to the DMA channel that will be used to service the McBSP. The default mapping of DMA channel-complete interrupts to CPU is as follows:

DMA channel 0	➔	CPU interrupt 8
DMA channel 1	➔	CPU interrupt 9
DMA channel 2	➔	CPU interrupt 11
DMA channel 3	➔	CPU interrupt 12

6. Put the DMA in a stop condition. Clear any previous R/WSTAT bits so that unwanted transfers do not occur.

7. DMA initialization: Program the DMA channel for required operation. Following would be a typical set up:

   Source address = DRR for reads OR memory location for writes.
   Destination address = memory location for reads OR DXR for writes.
   Transfer counter = number of elements to be transferred.
   Receive synchronization event, *R/WSYNC* = REVT from McBSP for reads.
   Transmit synchronization event, *R/WSYNC* = XEVT from McBSP for writes.
   DMA channel complete interrupt bit, *TCINT* = enabled
   Priority bit, *PRI* = 1; optional, but recommended.

8. Instruct the DMA to run. For example, set START=01b in the DMA channel's primary control register to start the DMA without auto-initialization.

9. Take the section (transmitter/receiver) that is not a frame master (frame sync is an input) out of reset by setting /XRST or /RRST=1. Now the slave is ready to accept a frame sync and start data transfer.

   Alternatively, a new frame sync interrupt ((R/X)INTM=10b) can be used to wake up the transmitter/receiver. This case is discussed in another application note titled, *'C6x McBSP Interface to a Single rate ST bus Device*.

10. Pull the frame master (transmitter or receiver) out of reset (/XRST or /RRST = 1).

11. If FSGM=1 (frame sync generated by sample rate generator), enable frame sync generator by setting /FRST=1. The first frame sync will be output after 8 CLKG clocks. If FSGM=0, frames are generated on every DXR to XSR copy and therefore /FRST is not used. In any case, the master now starts data transfer.

- **Interrupt-driven CPU Transfers:** Setting (R/X)INTM=00b in the SPCR allows the McBSP to interrupt the CPU whenever data is ready in DRR or when data can be written to DXR.

The initialization steps are similar to DMA driven transfers except that DMA is not used. Therefore, replace steps 5 through 8 under *DMA Transfers* with the following:

- Map the required XINT0/1 and/or RINT0/1 interrupts to the CPU via the Interrupt Multiplexer Registers.
- Enable the mapped interrupts.

Once the McBSP is initialized (after step 11 above), each element in the transfer will cause the execution of an ISR that writes to DXR or reads from DRR.

- **Polled CPU Transfers:** The transmit and receive ready, (R/X)RDY bits in SPCR is polled to determine the readiness of the transmitter and receiver. Here, the CPU has to constantly check for this condition thus preventing it from doing any useful work. The McBSP initialization process has some differences compared to the previous two methods of McBSP service.

  Since neither DMA nor interrupt-driven transfer is applicable for this case, steps 5 through 8 are not required. The steps are therefore 1 to 4, 9 to 11 followed by the polling loop.