

Interfacing TLC320AC01 to the TMS320C54x Serial Port

C5000 Applications Team

Digital Signal Processing Solutions

ABSTRACT

Most DSP systems transfer data through peripherals. These peripherals include parallel and serial ports. This application report describes how the serial ports are initialized and how the TLC320AC01 ('AC01) analog interface circuit (AIC) interfaces to the TMS320C54x[™] serial port. This application report also describes the various issues involved such as stack, context switching, interrupt priorities, and different addressing modes for collecting the samples during the interrupt processing.

Contents

1	Context Switching	1
2	Interrupt Handling	3
3	Interrupt Priority	5
4	Circular Addressing	6

List of Examples

Example 1.	Context Save and Restore for TMS320C54x	2
Example 2.	Receive Interrupt Service Routine	4
Example 3.	Interrupt Service Routine (ISR)	5
Example 4.	Circular Addressing Mode	6

1 Context Switching

Before you execute a routine, you must save its context and restore the context after the routine has finished. This procedure is called context switching, and involves pushing the PC onto the stack. Context switching is useful for subroutine calls, especially when making extensive use of the auxiliary registers, accumulators, and other memory-mapped registers.

Due to system and CPU requirements, the order of saving and restoring can vary. Some repeat instructions, such as RPTB, are interruptible. To nest repeat block instructions, you must ensure that the block-repeat counter (BRC), block-repeat start address (RSA), and block-repeat end address (REA) registers are saved and restored.

You must also ensure that the block-repeat active flag (BRAF) is properly set. Since the block-repeat flag can be deactivated by clearing the BRAF bit of the ST1 register, the order in which you push the block-repeat counter and ST1 is important. If the BRC register is pushed onto the stack prior to ST1, any PC discontinuity in RPTB can give a wrong result, since BRAF is cleared in ST1. Thus, you must restore BRC before restoring the ST1 register.

A context save complements the restored contents. To ensure the integrity of the code, determine what contents must be restored so that no sequencing is lost.

TMS320C54x is a trademark of Texas Instruments.



Example 1. Context Save and Restore for TMS320C54x

. Litle "CONTEXT SAVE/RESTORE on SUBROUTINE or INTERRUPT CONTEXT_RESTORE interro POPM PMST : Restore PMST register POPM REA : Restore block repeat end address POPM REA : Restore block repeat counter POPM BC : Restore interrupt mask register POPM BK : Restore circular size register POPM STI : Restore STI POPM ARO : Restore ARO POPM ARO : Restore ARA POPM ARO : Restore transition register POPM BL : Restore tower 16 bits of accB POPM BL : Restore 8 guard bits of accA POPM AG : Restore 8 guard bits of accA POPM AG : Save 8 guard bits of accA PSHM AG : Save ARO PSHM AG : Save ARO PSHM ARO : Save ARO PSHM ARO : Save ARA PSHM ARO : Save STI PSHM ARO : Save STI PSHM ARO : Save Interrupt mask register PSHM REA : Save block repeat counter PSHM REA : Save block repeat start address PSHM PMST ; Save EMST register .endm		
PORMPMST;Restore PMST registerPORMRSA;Restore block repeat end addressPORMRRC;Restore block repeat counterPORMBRC;Restore interrupt mask registerPORMRK;Restore circular size registerPORMST1;Restore ST0PORMST1;Restore AR1PORMAR0;Restore AR2PORMAR1;Restore AR3PORMAR2;Restore AR4PORMAR4;Restore AR6PORMAR4;Restore AR6PORMAR7;Restore AR6PORMAR7;Restore transition registerPORMAR7;Restore transition registerPORMBL;Restore lower 16 bits of accBPORMBL;Restore lower 16 bits of accAPORMAR4;Restore lower 16 bits of accAPORMAR4;Save 8 guard bits of accAPORMAR4;Save 8 guard bits of accAPORMAR4;Save 10 bits of accAPORMAR4;Save 8 guard bits of accAPORMAR4;Save 8 guard bits of accAPORMAR4;Save 10 bits of accAPORMAR4;Save 8 guard bits of accAPORMAR4;Save 8 guard bits of accAPSHMAG4;Save aguard bits of accA		
POPM RSA ;Restore block repeat end address POPM BRA ;Restore block repeat counter POPM BRC ;Restore bick repeat counter POPM IMR ;Restore interrupt mask register POPM ST1 ;Restore ST1 POPM ST0 ;Restore ST0 POPM AR0 ;Restore AR0 POPM AR1 ;Restore AR1 POPM AR2 ;Restore AR2 POPM AR3 ;Restore AR4 POPM AR4 ;Restore AR5 POPM AR6 ;Restore AR6 POPM AR6 ;Restore temporary register POPM TT ;Restore temporary register POPM BL ;Restore lower 16 bits of accB POPM BG ;Restore 8 guard bits of accA POPM AG ;Save 8 guard bits of accA PSHM AG ;Save 8 guard bits of accB PSHM AF5 ;Save AR7 PSHM AR6 ;Save AR6 PSHM AR7 ;Save transition register PSHM AR6 ;Save AR6 PSHM AR6 ;Save AR6 PSHM AR7 ;Save AR7 PSHM AR6 ;Save AR7 PSHM AR6 ;Save AR4 PSHM AR1 ;Save AR1 PSHM AR2 ;Save AR2 PSHM AR1 ;Save block repeat counter PSHM REA ;Save block repeat counter PSHM REA ;Save block repeat end address PSHM PMST ;Save block repeat start address PSHM PMST ;Save block repeat start address PSHM PMST ;Save block repeat start address	—	
POPM REA ;Restore block repeat end address POPM BRC ;Restore block repeat counter POPM IMR ;Restore interrupt mask register POPM ST1 ;Restore ST1 POPM ST0 ;Restore ST0 POPM AR0 ;Restore AR1 POPM AR1 ;Restore AR2 POPM AR2 ;Restore AR3 POPM AR4 ;Restore AR4 POPM AR5 ;Restore AR5 POPM AR5 ;Restore AR6 POPM AR7 ;Restore temporary register POPM T ;Restore temporary register POPM BH ;Restore lower 16 bits of accB POPM BL ;Restore 8 guard bits of accA POPM AG ;Save 8 guard bits of accA POPM BH ;Save upper 16 bits of accA POPM AG ;Save 8 guard bits of accA POPM AG ;Save 8 guard bits of accA POPM AG ;Save 8 guard bits of accA PSHM AG ;Save 8 guard bits of accA PSHM AG ;Save 100000 16 bits of accA PSHM AG ;Save 8 guard bits of accA PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AG ;Save 8 guard bits of accA PSHM AL ;Save upper 16 bits of accA PSHM AL ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accB PSHM BH ;Save lower 16 bits of accB PSHM AF ;Save AR7 PSHM AR7 ;Save AR7 PSHM AR7 ;Save AR7 PSHM AR7 ;Save AR7 PSHM AR7 ;Save AR7 PSHM AR3 ;Save AR8 PSHM AR1 ;Save AR8 PSHM AR1 ;Save AR8 PSHM AR1 ;Save AR1 PSHM AR1 ;Save AR1 PSHM AR1 ;Save Circular size register PSHM AR1 ;Save Circular size register PSHM AR1 ;Save block repeat end address PSHM RSA ;Save block repeat start address PSHM PMST ;Save PMST register		
POPMBRC;Restore block repeat counterPOPMIMR;Restore interrupt mask registerPOPMST0;Restore ST1POPM ST0;Restore ST0POPM AR0;Restore AR0POPM AR1;Restore AR2POPM AR2;Restore AR3POPM AR3;Restore AR4POPM AR4;Restore AR5POPM AR6;Restore AR7POPM AR6;Restore temporary registerPOPM AR7;Restore lemporary registerPOPM AR6;Restore lower 16 bits of accBPOPM BL;Restore lower 16 bits of accAPOPM AG;Restore lower 16 bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 16 bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Save 8 guard bits of accAPOPM AG;Save lower 16 bits of accAPSHM AG;Save lower 16 bits of accAPSHM AG;Save k guard bits of accAPSHM AT;Save lower 16 bits of accBPSHM AT;Save kernasition registerPSHM AF;Save AR6PSHM AF;Save AR7PSHM AR7;Save AR7PSHM AR3;Save AR3PSHM AR4;Save AR4PSHM AR5;Save AR6PSHM AR7;Save AR6PSHM AR4;Save AR6PSHM AR5;Save AR6PSHM AR2;Save AR6PSHM AR2;Save AR6PSHM AR4;Save AR6<	-	
POPMINR/Restore interrupt mask registerPOPMBK/Restore circular size registerPOPMST1;Restore ST0POPMST0;Restore ST0POPMAR0;Restore AR0POPMAR2;Restore AR1POPMAR2;Restore AR3POPMAR4;Restore AR4POPMAR5;Restore AR6POPMAR7;Restore temporary registerPOPMRT;Restore temporary registerPOPMBL;Restore lower 16 bits of accBPOPMBL;Restore lower 16 bits of accAPOPMBL;Restore lower 16 bits of accAPOPMBG;Restore lower 16 bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accBPSHMAG;Save 4R6PSHMSave 8 guard bits of accBPSHMAG;Save 4R6PSHMSave 2ac<	I I	
POPM BK 'Restore circular size register POPM ST1 'Restore ST1 POPM AR0 'Restore AR0 POPM AR1 'Restore AR1 POPM AR2 'Restore AR2 POPM AR3 'Restore AR4 POPM AR5 'Restore AR4 POPM AR5 'Restore AR7 POPM AR6 'Restore transition register POPM T 'Restore transition register POPM T 'Restore lower 16 bits of accB POPM BH 'Restore lower 16 bits of accB POPM BH 'Restore lower 16 bits of accA POPM AR4 'Restore lower 16 bits of accA POPM AR4 'Restore lower 16 bits of accA POPM AR4 'Restore 8 guard bits of accA POPM AR4 'Restore lower 16 bits of accA POPM AR4 'Save upper 16 bits of accA PSHM AF4 'Save upper 16 bits of accA PSHM AF4 'Save upper 16 bits of accA PSHM AF4 'Save upper 16 bits of accA PSHM AF5 'Save A 8 guard bits of accB PSHM BF4 'Save upper 16 bits of accB PSHM BF4 'Save upper 16 bits of accB PSHM AF5 'Save AR7 PSHM AF5 'Save AR7 PSHM AR7 'Save AR7 PSHM AR5 'Save AR7 PSHM AR5 'Save AR7 PSHM AR4 'Save AR7 PSHM AR4 'Save AR7 PSHM AR2 'Save AR7 PSHM AR4 'Save AR7 PSHM AR2 'Save AR7 PSHM AR6 'Save AR7 PSHM AR7 'Save AR7 PSHM AR6 'Save AR7 PSHM AR7 'Sa		
POPM ST1;Restore ST1POPM ST0;Restore ST0POPM AR0;Restore AR0POPM AR1;Restore AR1POPM AR2;Restore AR3POPM AR4;Restore AR4POPM AR5;Restore AR6POPM AR7;Restore temporary registerPOPM AR7;Restore temporary registerPOPM BL;Restore lower 16 bits of accBPOPM BL;Restore lower 16 bits of accAPOPM AR6;Restore lower 16 bits of accAPOPM AR1;Restore 8 guard bits of accAPOPM AR1;Save 8 guard bits of accAPOPM AR1;Save way per 16 bits of accAPSHM AR1;Save way per 16 bits of accAPSHM AR1;Save way per 16 bits of accAPSHM BG1;Save way transition registerPSHM BG2;Save 4R6PSHM T1;Save transition registerPSHM AR5;Save AR6PSHM AR3;Save AR1PSHM AR2;Save AR2PSHM AR3;Save AR1PSHM AR3;Save AR2PSHM AR3;Save AR1PSHM AR3;Save AR2PSHM AR3;Save AR2PSHM AR4;Save AR1PSHM AR5;Save AR2PSHM AR4;Save AR1PSHM AR5;Save AR2PSHM AR4;Save AR1PSH		
POPMSTO;Restore STOPOPMAR0;Restore AR1POPMAR2;Restore AR2POPMAR3;Restore AR3POPMAR4;Restore AR4POPMAR5;Restore AR6POPMAR6;Restore AR7POPMT;Restore lower 16 bits of accBPOPMBL;Restore lower 16 bits of accBPOPMBH;Restore lower 16 bits of accAPOPMBH;Restore lower 16 bits of accAPOPMBH;Restore lower 16 bits of accAPOPMBG;Restore upper 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAH;Restore 16 bits of accAPOPMAH;Save 8 guard bits of accAPOHMAH;Save 16 bits of accAPSHMAH;Save upper 16 bits of accAPSHMAH;Save lower 16 bits of accAPSHMBH;Save upper 16 bits of accBPSHMBH;Save upper 16 bits of accBPSHMBH;Save upper 16 bits of accBPSHMBH;Save tower 16 bits of accBPSHMBH;Save AR6PSHMSave AR6PSHMAR5;Save AR6PSHMAR6;Save AR4PSHMAR2;Save AR4PSHMAR2;Save AR1PSHMAR2;Save AR1PSHMAR1;Save AR1PSHMAR2;Save AR1PSHM<		
POPMAR0;Restore AR0POPMAR1;Restore AR1POPMAR3;Restore AR2POPMAR3;Restore AR3POPMAR4;Restore AR4POPMAR5;Restore AR5POPMAR7;Restore transition registerPOPMT;Restore transition registerPOPMT;Restore transition registerPOPMT;Restore transition registerPOPMBL;Restore transition registerPOPMBL;Restore transition registerPOPMBL;Restore transition registerPOPMBL;Restore transition registerPOPMBG;Restore transition registerPOPMBG;Restore tower 16 bits of accBPOPMAG;Restore tower 16 bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAG;Save 10 wer 16 bits of accAPSHMAH;Save 10 wer 16 bits of accBPSHMBH;Save 10 wer 16 bits of accBPSHMBH;Save argPSHMT;Save AR5PSHMAR5;Save AR5PSHMAR6;Save AR4PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMSave interrupt mask registerPSH		
POPMAR1;Restore AR1POPMAR2;Restore AR2POPMAR4;Restore AR3POPMAR5;Restore AR6POPMAR7;Restore AR6POPMT;Restore transition registerPOPMT;Restore lower 16 bits of accBPOPMBL;Restore upper 16 bits of accBPOPMBH;Restore upper 16 bits of accAPOPMBH;Restore upper 16 bits of accAPOPMBH;Restore upper 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAG;Save 8 guard bits of accAPOPMAG;Save 36 guard bits of accAPSHMAG;Save 4 guard bits of accAPSHMAL;Save upper 16 bits of accAPSHMAL;Save upper 16 bits of accAPSHMBG;Save 8 guard bits of accBPSHMBL;Save upper 16 bits of accBPSHMBL;Save upper 16 bits of accBPSHMBL;Save aR7PSHMSave are upper 16 bits of accBPSHMBL;Save are upper 16 bits of accBPSHMAG;Save AR7PSHMAR6;Save AR6PSHMAR7;Save AR7PSHMAR6;Save AR2PSHMAR1;Save AR2PSHMAR1;Save AR2PSHMAR1;Save AR1PSHMA		
POPMAR2;Restore AR2POPMAR3;Restore AR3POPMAR4;Restore AR4POPMAR5;Restore AR6POPMAR6;Restore AR7POPMT;Restore transition registerPOPMT;Restore lower 16 bits of accBPOPMBL;Restore lower 16 bits of accBPOPMBH;Restore lower 16 bits of accAPOPMBG;Restore lower 16 bits of accAPOPMBG;Restore lower 16 bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPOHMAG;Save 8 guard bits of accAPSHMAG;Save 10wer 16 bits of accAPSHMAG;Save 8 guard bits of accBPSHMBG;Save 8 guard bits of accBPSHMBG;Save 10wer 16 bits of accBPSHMBG;Save 10wer 16 bits of accBPSHMBG;Save 2000000000000000000000000000000000000		
POPMAR3;Restore AR3POPMAR4;Restore AR4POPMAR5;Restore AR5POPMAR7;Restore AR7POPMT;Restore transition registerPOPMT;Restore transition registerPOPMBL;Restore lower 16 bits of accBPOPMBH;Restore lower 16 bits of accAPOPMAH;Restore lower 16 bits of accAPOPMAH;Restore lower 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAH;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPOPMAH;Save 10wer 16 bits of accAPSHMAH;Save 10wer 16 bits of accAPSHMAH;Save 10wer 16 bits of accAPSHMAH;Save 10wer 16 bits of accBPSHMBH;Save 10wer 16 bits of accBPSHMBH;Save upper 16 bits of accBPSHMBH;Save transition registerPSHMBH;Save transition registerPSHMT;Save transition registerPSHMAR7;Save AR6PSHMAR4;Save AR6PSHMAR2;Save AR6PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR2;Save AR2PSHMAR1;Save ST0PSHMST1;Save ST0PSHMST1;Save ST0PSHMBC;Save block repeat counterPSHMBC <ts< td=""><td></td><td></td></ts<>		
POPMAR4;Restore AR4POPMAR5;Restore AR5POPMAR6;Restore AR7POPMT;Restore temporary registerPOPMTRN;Restore tomporary registerPOPMTRN;Restore lower 16 bits of accBPOPMBH;Restore lower 16 bits of accAPOPMBG;Restore lower 16 bits of accAPOPMAL;Restore lower 16 bits of accAPOPMAL;Restore lower 16 bits of accAPOPMAL;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPOHMAH;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accAPSHMAH;Save lower 16 bits of accAPSHMBG;Save 8 guard bits of accBPSHMBG;Save 8 guard bits of accBPSHMBH;Save lower 16 bits of accBPSHMBH;Save lower 16 bits of accBPSHMBH;Save lower 16 bits of accBPSHMBH;Save AR7PSHMAR7;Save AR7PSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR3;Save AR6PSHMAR1;Save AR2PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMST1;Save ST1PSHMBK;Save circular size registerPSHMBK <td>POPM AR2 ;Restore AR2</td> <td></td>	POPM AR2 ;Restore AR2	
POPMAR5;Restore AR5POPMAR6;Restore AR6POPMAR7;Restore AR7POPMT;Restore transition registerPOPMTN;Restore transition registerPOPMBL;Restore lower 16 bits of accBPOPMBL;Restore a guard bits of accAPOPMAL;Restore upper 16 bits of accAPOPMAL;Restore a guard bits of accAPOPMAL;Restore a guard bits of accAPOPMAG;Restore a guard bits of accAPOPMAG;Restore a guard bits of accAPOPMAG;Save 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAG;Save a guard bits of accAPSHMAH;Save lower 16 bits of accAPSHMBG;Save 8 guard bits of accBPSHMBH;Save lower 16 bits of accBPSHMBH;Save lower 16 bits of accBPSHMBH;Save transition registerPSHMBL;Save transition registerPSHMT;Save transition registerPSHMAR6;Save AR6PSHMAR4;Save AR4PSHMAR3;Save AR3PSHMAR1;Save AR1PSHMAR1;Save AR2PSHMAR1;Save AR2PSHMAR1;Save STIPSHMSC;Save STIPSHMBK;Save circular size registerPSHMBK;Save block repeat counter <tr< td=""><td>POPM AR3 ;Restore AR3</td><td></td></tr<>	POPM AR3 ;Restore AR3	
POPMAR6;Restore AR6POPMAR7;Restore AR7POPMT;Restore temporary registerPOPMTRN;Restore transition registerPOPMBL;Restore lower 16 bits of accBPOPMBH;Restore 8 guard bits of accBPOPMAL;Restore 10 wer 16 bits of accAPOPMAL;Restore 10 wer 16 bits of accAPOPMAL;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAH;Save upper 16 bits of accAPSHMAL;Save 10 wer 16 bits of accAPSHMBG;Save 8 guard bits of accBPSHMBL;Save lower 16 bits of accBPSHMBL;Save AR7PSHMAR7PSHMAR6PSHMAR7PSHMAR6PSHMAR1PSHMAR2PSHMAR1PSHMAR2PSHMAR1PSHMAR2PSHMAR1PSHMAR2PSHMAR1PSHMAR2PSHMAR2PSHMAR1PSHMSave AR0PSHMSAve AR1 <td>POPM AR4 ;Restore AR4</td> <td></td>	POPM AR4 ;Restore AR4	
POPMAR7;Restore AR7POPMT;Restore temporary registerPOPMTRN;Restore transition registerPOPMBL;Restore lower 16 bits of accBPOPMBH;Restore 0 guard bits of accBPOPMBG;Restore 10 bits of accAPOPMAL;Restore 10 bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accAPSHMAH;Save 10 bits of accAPSHMAH;Save 8 guard bits of accBPSHMBL;Save 8 guard bits of accBPSHMBL;Save 10 bits of accBPSHMBL;Save 10 bits of accBPSHMT;Save temporary registerPSHMT;Save aR7PSHMAR7;Save AR6PSHMAR2;Save AR4PSHMAR2;Save AR4PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR1;Save AR2PSHMAR1;Save ST0PSHMST1;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMBRC;Save block repeat counterPSHMBRC;Save block repeat at addressPSHMPMST;Save PMST register	POPM AR5 ;Restore AR5	
POPM T;Restore temporary registerPOPM BL;Restore transition registerPOPM BL;Restore lower 16 bits of accBPOPM BG;Restore upper 16 bits of accBPOPM AL;Restore lower 16 bits of accAPOPM AL;Restore upper 16 bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Save 8 guard bits of accAPSHM AG;Save 8 guard bits of accAPSHM AL;Save lower 16 bits of accAPSHM BG;Save 8 guard bits of accBPSHM BH;Save upper 16 bits of accBPSHM BL;Save lower 16 bits of accBPSHM BL;Save transition registerPSHM T;Save transition registerPSHM AR;Save AR7PSHM AR6;Save AR6PSHM AR3;Save AR4PSHM AR4;Save AR4PSHM AR2;Save AR2PSHM AR1;Save AR0PSHM ST0;Save ST0PSHM ST1;Save circular size registerPSHM BRC;Save block repeat counterPSHM BRC;Save block repeat counterPSHM REA;Save block repeat start addressPSHM REA;Save PMST register	POPM AR6 ;Restore AR6	
POPM TRN;Restore transition registerPOPM BL;Restore lower 16 bits of accBPOPM BG;Restore a guard bits of accBPOPM AL;Restore lower 16 bits of accAPOPM AL;Restore a guard bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 8 guard bits of accAPOPM AG;Restore 0 bits of accAPOPM AG;Restore 16 bits of accAPSHM AG;Save 8 guard bits of accAPSHM AH;Save 10wer 16 bits of accAPSHM BG;Save 8 guard bits of accBPSHM BG;Save 10wer 16 bits of accBPSHM BL;Save lower 16 bits of accBPSHM TN;Save transition registerPSHM TRN;Save transition registerPSHM AR7;Save AR7PSHM AR6;Save AR7PSHM AR7;Save AR7PSHM AR6;Save AR4PSHM AR3;Save AR4PSHM AR4;Save AR4PSHM AR5;Save AR1PSHM AR0;Save ST0PSHM ST1;Save ST1PSHM BK;Save circular size registerPSHM BRC;Save block repeat counterPSHM REA;Save block repeat counterPSHM REA;Save block repeat start addressPSHM REA;Save PMST register	POPM AR7 ;Restore AR7	
POPMBL;Restore lower 16 bits of accBPOPMBH;Restore upper 16 bits of accBPOPMBG;Restore 8 guard bits of accAPOPMAL;Restore upper 16 bits of accAPOPMAH;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 7 guard bits of accAPORMAG;Save 8 guard bits of accAPSHMAG;Save 10wer 16 bits of accAPSHMBG;Save 10wer 16 bits of accBPSHMBH;Save 10wer 16 bits of accBPSHMBH;Save upper 16 bits of accBPSHMBH;Save 10wer 16 bits of accBPSHMBH;Save upper 16 bits of accBPSHMBH;Save accPSHMBH;Save accPSHMAR7;Save transition registerPSHMAR7;Save AR7PSHMAR4;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR2PSHMAR1;Save AR2PSHMAR1;Save ST1PSHMAR0;Save ST1PSHMBK;Save circular size registerPSHMBRC;Save	POPM T ;Restore temporary register	
<pre>POPM BH ;Restore upper 16 bits of accB POPM AG ;Restore 8 guard bits of accA POPM AL ;Restore lower 16 bits of accA POPM AH ;Restore upper 16 bits of accA POPM AG ;Restore 8 guard bits of accA .endm CONTEXT_SAVE .macro PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BL ;Save upper 16 bits of accB PSHM BL ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TRN ;Save transition register PSHM TRN ;Save temporary register PSHM AR7 ;Save AR7 PSHM AR5 ;Save AR6 PSHM AR5 ;Save AR5 PSHM AR4 ;Save AR4 PSHM AR3 ;Save AR4 PSHM AR2 ;Save AR2 PSHM AR1 ;Save AR1 PSHM AR1 ;Save CN2 PSHM AR1 ;Save ST0 PSHM ST0 ;Save ST0 PSHM ST1 ;Save circular size register PSHM IMR ;Save interrupt mask register PSHM IMR ;Save block repeat counter PSHM REA ;Save block repeat start address PSHM REA ;Save PMST register</pre>	POPM TRN ; Restore transition register	
<pre>POPM BH ;Restore upper 16 bits of accB POPM AG ;Restore 8 guard bits of accA POPM AL ;Restore lower 16 bits of accA POPM AH ;Restore upper 16 bits of accA POPM AG ;Restore 8 guard bits of accA .endm CONTEXT_SAVE .macro PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BL ;Save upper 16 bits of accB PSHM BL ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TRN ;Save transition register PSHM TRN ;Save temporary register PSHM AR7 ;Save AR7 PSHM AR5 ;Save AR6 PSHM AR5 ;Save AR5 PSHM AR4 ;Save AR4 PSHM AR3 ;Save AR4 PSHM AR2 ;Save AR2 PSHM AR1 ;Save AR1 PSHM AR1 ;Save CN2 PSHM AR1 ;Save ST0 PSHM ST0 ;Save ST0 PSHM ST1 ;Save circular size register PSHM IMR ;Save interrupt mask register PSHM IMR ;Save block repeat counter PSHM REA ;Save block repeat start address PSHM REA ;Save PMST register</pre>		
POPMBG;Restore 8 guard bits of accBPOPMAL;Restore lower 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAG;Restore 8 guard bits of accA.endmCONTEXT_SAVE.macroPSHMAG;Save 8 guard bits of accAPSHMAH;Save upper 16 bits of accAPSHMAH;Save 10wer 16 bits of accAPSHMAL;Save 8 guard bits of accBPSHMBG;Save 8 guard bits of accBPSHMBH;Save upper 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMBL;Save transition registerPSHMT;Save temporary registerPSHMAR7;Save AR7PSHMAR5;Save AR6PSHMAR4;Save AR7PSHMAR2;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMAR1;Save ST0PSHMST1;Save ST1PSHMBRC;Save block repeat counterPSHMBRC;Save block repeat addressPSHMREA;Save block repeat addressPSHMREA;Save block repeat start addressPSHMPMST;Save PMST register		
POPMAL;Restore lower 16 bits of accAPOPMAH;Restore upper 16 bits of accAPOPMAG;Restore 8 guard bits of accAPOPMAG;Save 8 guard bits of accAPSHMAG;Save 8 guard bits of accAPSHMAH;Save lower 16 bits of accAPSHMAL;Save 8 guard bits of accBPSHMBG;Save 8 guard bits of accBPSHMBH;Save lower 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMBL;Save transition registerPSHMT;Save temporary registerPSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR3;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR2PSHMAR1;Save AR1PSHMAR0;Save AR0PSHMST1;Save interrupt mask registerPSHMBK;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat start addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
POPMAH;Restore upper 16 bits of accAPOPMAG;Restore 8 guard bits of accA.endm.CONTEXT_SAVE.macroPSHMAG;Save 8 guard bits of accAPSHMAH;Save lower 16 bits of accAPSHMAL;Save 8 guard bits of accBPSHMBG;Save 8 guard bits of accBPSHMBH;Save lower 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMT;Save transition registerPSHMT;Save transition registerPSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR7;Save AR7PSHMAR4;Save AR4PSHMAR2;Save AR3PSHMAR2;Save AR4PSHMAR1;Save AR1PSHMAR1;Save AR1PSHMST1;Save ST0PSHMBK;Save circular size registerPSHMBK;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat start addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
<pre>POPM AG ;Restore 8 guard bits of accA .endm CONTEXT_SAVE .macro PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BH ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TRN ;Save transition register PSHM TRN ;Save transition register PSHM AR7 ;Save temporary register PSHM AR7 ;Save AR7 PSHM AR6 ;Save AR6 PSHM AR5 ;Save AR6 PSHM AR4 ;Save AR4 PSHM AR3 ;Save AR3 PSHM AR2 ;Save AR2 PSHM AR1 ;Save AR1 PSHM AR0 ;Save AR0 PSHM ST0 ;Save ST0 PSHM ST1 ;Save circular size register PSHM BK ;Save circular size register PSHM BK ;Save interrupt mask register PSHM REA ;Save block repeat counter PSHM REA ;Save Dlock repeat address PSHM RSA ;Save PMST register</pre>		
.endm CONTEXT_SAVE .macro PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BG ;Save 8 guard bits of accB PSHM BH ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TT ;Save transition register PSHM TT ;Save transition register PSHM AR7 ;Save transition register PSHM AR6 ;Save AR7 PSHM AR6 ;Save AR7 PSHM AR5 ;Save AR8 PSHM AR4 ;Save AR4 PSHM AR2 ;Save AR2 PSHM AR1 ;Save AR1 PSHM AR1 ;Save AR0 PSHM ST0 ;Save ST0 PSHM ST1 ;Save circular size register PSHM BK ;Save interrupt mask register PSHM BK ;Save block repeat counter PSHM REA ;Save Dlock repeat address PSHM RSA ;Save PMST register		
CONTEXT_SAVE .macro PSHM AG ;Save 8 guard bits of accA PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BG ;Save 8 guard bits of accB PSHM BH ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TrN ;Save transition register PSHM T ;Save temporary register PSHM AR7 ;Save AR7 PSHM AR6 ;Save AR6 PSHM AR5 ;Save AR6 PSHM AR4 ;Save AR4 PSHM AR2 ;Save AR3 PSHM AR2 ;Save AR3 PSHM AR1 ;Save AR1 PSHM AR0 ;Save AR0 PSHM ST0 ;Save ST0 PSHM ST1 ;Save ST1 PSHM BK ;Save circular size register PSHM IMR ;Save interrupt mask register PSHM BC ;Save block repeat counter PSHM RSA ;Save PMST register		
PSHMAG;Save 8 guard bits of accAPSHMAH;Save upper 16 bits of accAPSHMAL;Save lower 16 bits of accAPSHMBG;Save 8 guard bits of accBPSHMBH;Save upper 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMTX;Save transition registerPSHMTX;Save transition registerPSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR4;Save AR4PSHMAR2;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMST0;Save ST0PSHMST1;Save interrupt mask registerPSHMBK;Save interrupt mask registerPSHMBRC;Save block repeat end addressPSHMREA;Save block repeat start addressPSHMRSA;Save PMST register		
<pre>PSHM AH ;Save upper 16 bits of accA PSHM AL ;Save lower 16 bits of accA PSHM BG ;Save 8 guard bits of accB PSHM BH ;Save upper 16 bits of accB PSHM BL ;Save lower 16 bits of accB PSHM TRN ;Save transition register PSHM T ;Save temporary register PSHM AR7 ;Save AR7 PSHM AR6 ;Save AR6 PSHM AR5 ;Save AR6 PSHM AR4 ;Save AR4 PSHM AR3 ;Save AR3 PSHM AR2 ;Save AR3 PSHM AR1 ;Save AR1 PSHM AR1 ;Save AR1 PSHM AR0 ;Save AR0 PSHM ST0 ;Save ST1 PSHM BK ;Save circular size register PSHM IMR ;Save interrupt mask register PSHM BK ;Save block repeat counter PSHM REA ;Save PMST register</pre>		
PSHMAL; Save lower 16 bits of accAPSHMBG; Save 8 guard bits of accBPSHMBH; Save upper 16 bits of accBPSHMBL; Save lower 16 bits of accBPSHMTRN; Save transition registerPSHMT; Save temporary registerPSHMAR7; Save AR7PSHMAR6; Save AR6PSHMAR5; Save AR6PSHMAR4; Save AR5PSHMAR2; Save AR4PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST1; Save ST0PSHMBK; Save circular size registerPSHMBK; Save block repeat counterPSHMREA; Save block repeat end addressPSHMREA; Save block repeat start addressPSHMRSA; Save PMST register		
PSHMBG;Save 8 guard bits of accBPSHMBH;Save upper 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMTRN;Save transition registerPSHMT;Save temporary registerPSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR5;Save AR5PSHMAR4;Save AR4PSHMAR2;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMST0;Save ST0PSHMST1;Save interrupt mask registerPSHMBK;Save block repeat counterPSHMREA;Save block repeat start addressPSHMRSA;Save PMST register		
PSHMBH;Save upper 16 bits of accBPSHMBL;Save lower 16 bits of accBPSHMTRN;Save transition registerPSHMT;Save temporary registerPSHMAR7;Save AR7PSHMAR6;Save AR6PSHMAR5;Save AR6PSHMAR4;Save AR4PSHMAR2;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR0;Save AR0PSHMST1;Save ST0PSHMBK;Save circular size registerPSHMBK;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat start addressPSHMRSA;Save PMST register		
PSHMBL; Save lower 16 bits of accBPSHMTRN; Save transition registerPSHMT; Save temporary registerPSHMAR7; Save AR7PSHMAR6; Save AR7PSHMAR6; Save AR6PSHMAR5; Save AR6PSHMAR4; Save AR4PSHMAR2; Save AR3PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST1; Save ST0PSHMST1; Save circular size registerPSHMBK; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat start addressPSHMRSA; Save PMST register		
PSHMTRN; Save transition registerPSHMT; Save temporary registerPSHMAR7; Save AR7PSHMAR6; Save AR6PSHMAR5; Save AR6PSHMAR4; Save AR4PSHMAR3; Save AR3PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save circular size registerPSHMBK; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat start addressPSHMRSA; Save PMST register		
PSHMT; Save temporary registerPSHMAR7; Save AR7PSHMAR6; Save AR6PSHMAR5; Save AR5PSHMAR4; Save AR4PSHMAR3; Save AR3PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save circular size registerPSHMBK; Save cinterrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save PMST register		
PSHMAR7; Save AR7PSHMAR6; Save AR6PSHMAR5; Save AR5PSHMAR4; Save AR4PSHMAR3; Save AR3PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save ST1PSHMBK; Save circular size registerPSHMIMR; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save PMST register		
PSHMAR6;Save AR6PSHMAR5;Save AR5PSHMAR4;Save AR4PSHMAR3;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR0;Save AR0PSHMST0;Save ST0PSHMST1;Save ST1PSHMBK;Save circular size registerPSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMAR5; Save AR5PSHMAR4; Save AR4PSHMAR3; Save AR3PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save ST1PSHMBK; Save circular size registerPSHMIMR; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save pMST register		
PSHMAR4;Save AR4PSHMAR3;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR0;Save AR0PSHMST0;Save ST0PSHMST1;Save ST1PSHMBK;Save circular size registerPSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMAR3;Save AR3PSHMAR2;Save AR2PSHMAR1;Save AR1PSHMAR0;Save AR0PSHMST0;Save ST0PSHMST1;Save ST1PSHMBK;Save circular size registerPSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMAR2; Save AR2PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save ST1PSHMBK; Save circular size registerPSHMIMR; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save block repeat start addressPSHMPMST; Save PMST register		
PSHMAR1; Save AR1PSHMAR0; Save AR0PSHMST0; Save ST0PSHMST1; Save ST1PSHMBK; Save circular size registerPSHMIMR; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save block repeat start addressPSHMPMST; Save PMST register		
PSHMAR0;Save AR0PSHMST0;Save ST0PSHMST1;Save ST1PSHMBK;Save circular size registerPSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMST0; SaveST0PSHMST1; SaveST1PSHMBK; Savecircular sizeregisterPSHMIMR; SaveinterruptmaskregisterPSHMBRC; SaveblockrepeatcounterPSHMREA; SaveblockrepeatendaddressPSHMRSA; SaveblockrepeatstartaddressPSHMPMST; SavePMSTregister		
PSHMST1; Save ST1PSHMBK; Save circular size registerPSHMIMR; Save interrupt mask registerPSHMBRC; Save block repeat counterPSHMREA; Save block repeat end addressPSHMRSA; Save block repeat start addressPSHMPMST; Save PMST register		
PSHMBK;Save circular size registerPSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMIMR;Save interrupt mask registerPSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHMBRC;Save block repeat counterPSHMREA;Save block repeat end addressPSHMRSA;Save block repeat start addressPSHMPMST;Save PMST register		
PSHM REA ;Save block repeat end address PSHM RSA ;Save block repeat start address PSHM PMST ;Save PMST register		
PSHM RSA ;Save block repeat start address PSHM PMST ;Save PMST register	-	
PSHM PMST ;Save PMST register		
.enam		
	.enam	

2 Interrupt Handling

The '54x CPU supports 16 user-maskable interrupts. The vectors for interrupts not used by a '54x device can function as software interrupts, using the INTR and TRAP instructions. TRAP and INTR allow you to execute any of the 32 available ISRs. You can define other locations in the interrupt vector table. The INTR instruction sets the INTM bit to 1, clears the corresponding interrupt flag to 0, and makes the IACK signal active, but the TRAP instruction does not. INTR and TRAP are nonmaskable interrupts.

When a maskable interrupt occurs, the corresponding flag is set to 1 in the interrupt flag register (IFR). Interrupt processing begins if the corresponding bit in IMR register is set to 1 and the INTM bit in the ST1 register is cleared. The IFR register can be read and action taken if an interrupt occurs. This is true even when the interrupt is disabled. This is useful when not using an interrupt-driven interface, such as in a subroutine call when INT1 has not occurred.

When interrupt processing begins, the PC is pushed onto the stack and the interrupt vector is loaded into the PC. Interrupts are then disabled by setting INTM = 1. The program continues from the address loaded in the PC. Since all interrupts are disabled, the program can be processed without any interruptions, unless the ISR reenables them. Except for very simple ISRs, it is important to save the processor context during execution of the routine.

During the time the 'AC01 is reset, the DSP initializes the serial port and sets up the interrupt. To set up the interrupts, it performs the following operations:

- Enables unmasked interrupts by clearing the interrupt mode bit (INTM)
- Clears prior receive interrupts by writing the current contents of the appropriate receive interrupt flag in the IFR back to the IFR
- Enables receive interrupts by setting the appropriate receive interrupt flag in the interrupt mask register (IMR)

The initialization of the IMR and IFR registers and the INTM bit is included in the serial port and the 'AC01 initialization.

Example 2 processes the receive interrupt 1 service routine. The routine collects 256 samples in the first buffer and changes the address to the second buffer for the next 256 samples while processing the first buffer.

Example 2. Receive Interrupt Service Routine

; TEXAS INSTRUMENTS INCORPORATED .mmregs .include "INTERRPT.INC" .include "main.inc" RCV_INT1_DP.usect"rcv_vars",0d_index_count.usect"rcv_vars",1d_rcv_in_ptr.usect"rcv_vars",1d_xmt_out_ptr.usect"rcv_vars",1 ; save/restore input bffr ptr ; save/restore output bffr ptr d_frame_flag .usect "rcv_vars",1 input_data .usect "inpt_buf",K_FRAME_SIZE*2 ; input data array output_data .usect "outdata",K_FRAME_SIZE*2 ; output data array .def receive_int1 .def d_frame_flag .def RCV_INT1_DP .cf input data.output_data ptr .def d xmt out ptr .def d_rcv_in_ptr ;-----Functional Description ; This routine services receive interrupt1. Accumulator A, AR2 and AR3 ; are pushed onto the stack since AR2 and AR3 are used in other applications. ; A 512 buffer size for both input and output. ; After every 256 collection of input samples a flag is set to process the ; data. No circular buffering scheme is used here. ; After collecting 256 samples in the 1st bffr, then the second buffer ; ; address is loaded and collect data in the second buffer while processing ; the first buffer and vice versa. ;------; get frame input data pointer .asgAR2,GETFRM_IN_P.asgAR3,GETFRM_OUT_P.asgAR2,SAVE_RSTORE_AR2.asgAR3,SAVE_RSTORE_AR3.sect"main_prg" ; get frame output data pointer receive_int1: PSHM AL AH PSHM PSHM AG PSHM BLPSHM BH PSHM BG ; AR2, AR3 are used in other routines, they need to be saved and restored ; since receive interrupt uses AR2 and AR3 as pointers PSHM SAVE RSTORE AR2 ; Since AR2 and AR3 are used PSHM SAVE_RSTORE_AR3 ; in other routines, they need PSHM BRC ; init. DP LD #RCV_INT1_DP,DP #RCV_INT1_DP,DP , INTC. Dr d_rcv_in_ptr,GETFRM_IN_P ; restore input bffr ptr d_xmt_out_ptr,GETFRM_OUT_P ; restore output bffr ptr #1.d index count ; increment the index count MVDK MVDK ADDM #K_FRAME_SIZE,A T'D SUB BC get_samples,AGT ; check for a frame of samples frame_flag_set ADDM #1,d_int_count ST #K_FRAME_FLAG,d_frame_flag ; set frame flag #K_FRAME_FLAG,d_Irame_____; reset the counter #0,d_index_count ; reset the counter #input_data+K_FRAME_SIZE,A ; second input bffr starting addr #output_data+K_FRAME_SIZE,B ; second output bffr starting addr d int count.2 ; check for 1st/2nd bffr STLD T'D BITF d_int_count,2 BC

Example 2. Receive Interrupt Service Routine (Continued)

SUB ; 1st input address #K_FRAME_SIZE,A SUB #K FRAME SIZE, B ; 1st output address ST#K 0,d int count reset buffer STLM A, GETFRM_IN_P ; input buffer address STLM B,GETFRM_OUT_P ; output buffer address get samples LDM DRR1,A ; load the input sample A,*GETFRM_IN_P+ *GETFRM_OUT_P+,A ; write to buffer STL ; if not true, then the filtered T'D AND #0fffch,A ; signal is send as output STLM A, DXR1 ; write to DXR1 MVKD GETFRM_IN_P,d_rcv_in_ptr MVKD GETFRM_OUT_P,d_xmt_out_ptr ; save input buffer ptr ; save out bffr ptr POPM BRC POPM SAVE_RSTORE_AR3 ; restore AR3 SAVE_RSTORE_AR2 POPM ; restore AR2 POPM BG POPM BH POPM BL POPM AG POPM AH POPM AL POPM ST1 POPM ST0 RETE ; return and enable interrupts .end

3 Interrupt Priority

Interrupt prioritization allows interrupts that occur simultaneously to be serviced in a predefined order. For instance, infrequent but lengthy ISRs can be interrupted frequently. In Example 3, the ISR for the INT1 bit includes context save and restore macros. When the routine has finished processing, the IMR is restored to its original state. Notice that the RETE instruction not only pops the next program counter address from the stack, but also clears the INTM bit to 0. This enables all interrupts that have their IMR bit set.

Example 3. Interrupt Service Routine (ISR)



There is a potential conflict between the INTM bit disable and context restore. If an interrupt 0 (INT0) occurs during context restore, the macro CONTEXT_RESTORE is executed before servicing INT0. This can trigger an INT0. If INTM is cleared during the context restore, it branches to the INT0 service routine. If you reenable the interrupts when INTM returns from INT0, a conflict occurs, because INTM is set to 0 and its original contents are lost. To preserve the contents of the INTM bit, do not enable the interrupts when INTM returns from the INTO service routine. During interrupt priorities, preserve the INTM and IMR bits for the system requirements.

4 Circular Addressing

Circular addressing is an important feature of the '54x instruction set. Algorithms for convolution, correlation, and FIR filters can use circular buffers in memory. In these algorithms, the circular buffers implement a sliding window that contains the most recent data. As new data comes in, it overwrites the oldest data. The size, the bottom address, and the top address of the circular buffer are specified by the block size register (BK) and a user-selected auxiliary register (ARn). A circular buffer size of R must start on a K-bit boundary (that is, the K LSBs of the starting address of the circular buffer must be 0), where K is the smallest integer that satisfies 2^K > R.

Circular addressing can be used for different functions of an application. For example, it can be used for collecting the input samples in a block. It can also be used in processing samples in blocks and data in the output buffer. In Example 4, a frame of 256 samples is collected from the serial port to process the data using the circular addressing mode. The output from the processed block is sent to the D/A converter through the serial port register using circular buffers. A ping-pong buffering scheme is used. While processing the first buffer, samples are collected in the second buffer, and vice versa. The real-time operation of the system is not disturbed and no data samples are lost.

Example 4.	Circular	Addressing	Mode
------------	----------	------------	------

```
; TEXAS INSTRUMENTS INCORPORATED
.include "INTERRPT.
.include "main.inc"
RCV_INT1_DP .user
d_inder
    .mmregs
                 "INTERRPT.INC"
RCV_INT1_DP .usect "rcv_vars",0
d_index_count .usect "rcv_vars",1
d_rcv_in ptr "
d_rcv_in_ptr .usect "rcv_vars",1
d_xmt_out_ptr .usect "rcv_vars",1
d_frame_flag .usect "rcv_vars",1
                                                              ; save/restore input bffr ptr
                                                              ; save/restore output bffr ptr
input_data .usect "inpt_buf",K_FRAME_SIZE*2 ; input data array
output_data .usect "outdata",K_FRAME_SIZE*2 ; output data array
    .def
                receive_int1
    .def
                 d_frame_flag
    .def
                RCV_INT1_DP
input_data,output_data
    .def
    .def
                 d_xmt_out_ptr
    .def
                d_rcv_in_ptr
; ------
                                          _____
       Functional Description
;
    This routine services receive interrupt1. Accumulator A, AR2 and AR3
;
   are pushed onto the stack since AR2 and AR3 are used in other applications.
;
   A 512 buffer size of both input and output uses circular addressing.
;
   After every 256 collection of input samples a flag is set to process the
;
   data. A PING/PONG buffering scheme is used such that upon processing
;
   PING buffer, samples are collected in the PONG buffer and vice versa.
;
```



.asg AR2,GETFRM_IN_P ; get frame input data pointer .asg AR3,GETFRM_OUT_P ; get frame output data pointer .asg AR2, SAVE_RSTORE_AR2 .asg AR3, SAVE_RSTORE_AR3 .sect "main_prg" receive intl: PSHM AL PSHM AH AG PSHM PSHM BL PSHM BHPSHM BG ; AR2, AR3 are used in other routines, they need to be saved and restored ; since receive interrupt uses AR2 and AR3 as pointers ; Since AR2 and AR3 are used PSHM SAVE_RSTORE_AR2 PSHM SAVE_RSTORE_AR3 ; in other routines, they need PSHM BRC STM #2*K_FRAME_SIZE,BK ; circular buffer size of in,out ; arrays LD#RCV_INT1_DP,DP ; init. DP MVDK d_rcv_in_ptr,GETFRM_IN_P ; restore input circular bffr ptr MVDK d_xmt_out_ptr,GETFRM_OUT_P ; restore output circular bffr ptr ADDM #1,d index count ; increment the index count LD#K_FRAME_SIZE,A SUB d_index_count, A BC get_samples,AGT ; check for a frame of samples frame flag set #K_FRAME_FLAG,d_frame_flag ; set frame flag ST#0,d_index_count ; reset the counter STget_samples LDM DRR1,A ; load the input sample STL A,*GETFRM IN P+% ; write to buffer *GETFRM_OUT_P+%,A ; if not true, then the filtered LD AND #0fffch,A ; signal is send as output STLM A, DXR1 ; write to DXR1 ; save input circular buffer ptr GETFRM_IN_P,d_rcv_in_ptr MVKD MVKD GETFRM_OUT_P,d_xmt_out_ptr ; save out circular bffr ptr POPM BRC POPM SAVE_RSTORE_AR3 ; restore AR3 POPM SAVE_RSTORE_AR2 ; restore AR2 POPM BG POPM BH POPM BL AG POPM AH POPM POPM AL POPM ST1 POPM ST0 RETE ; return and enable interrupts .end

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products 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.

Customers are responsible for their applications using TI components.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 2000, Texas Instruments Incorporated