Interfacing the TLC32040 Family to the TMS320 Family

SLAU001A
July 1995

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 applications 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.

Contents

Appendices

1 Introduction

The TLC32040 and TLC32041 analog interface circuits are designed to provide a high level of system integration and performance. The analog interface circuits combine high resolution A/D and D/A converters, programmable filters, digital control and timing circuits as well as programmable input amplifiers and multiplexers. Emphasis is placed on making the interface to digital signal processors (the TMS320 family) and most microprocessors as simple as possible. This application report describes the software and circuits necessary to interface to numerous members of the TMS320 family. It presents three circuits for interfacing the TLC32040 Analog Interface Circuit to the TMS320 family of digital signal processors. Details of the hardware and software necessary for these interfaces are provided.

To facilitate the discussion of the software the following definitions and naming conventions are used:

1. >nnnn – a number represented in hexadecimal.
2. Interrupt service routine – a subroutine called in direct response to a processor interrupt.
3. Interrupt subroutine – any routine called by the interrupt service routine.
4. Application program (application routine) – the user’s application dependent software (e.g., digital filtering routines, signal generation routines, etc.)

2 TLC32040 Interface to the TMS32010/E15

2.1 Hardware

Because the TLC32040 (Analog Interface Circuit) is a serial-I/O device, the interface to the TMS32010, which has no serial port, requires a small amount of glue-logic. The circuit shown in Figure 1 accomplishes the serial-to-parallel conversion for the AIC operating in synchronous mode.

2.1.1 Parts List

The interface circuit for the TMS32010 uses the following standard logic circuits:

1. One SN74LS138 3-to-8-line address decoder
2. One SN74LS02 Quad NOR-Gate
3. One SN74LS00 Quad NAND-Gate
4. One SN74LS04 Hex Inverter
5. One SN74LS74 Dual D-Flip-Flop
6. Two SN74LS299 8-bit Shift Registers

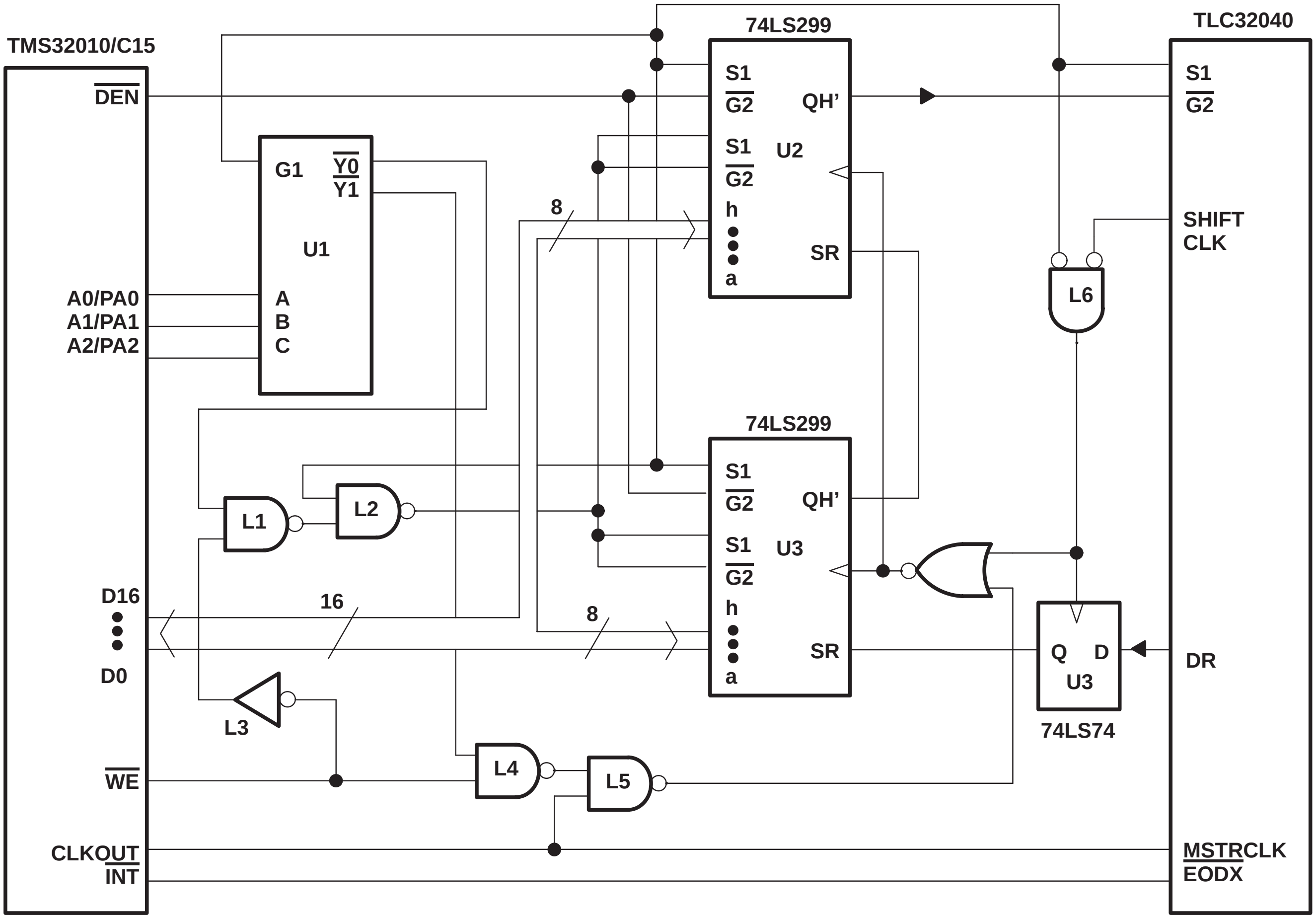


Figure 1. AIC Interface to TMS32010/E15

2.1.2 Hardware Description

The SN74LS138 is used to decode the addresses of the ports to which the TLC32040 and the interface logic have been mapped. If no other ports are needed in the development system, this device may be eliminated and the address lines of the TMS32010 used directly in place of $\overline{\text{Y1}}$ and $\overline{\text{Y0}}$ (see Figure 1).

Since the interface circuits are only addressed when the TMS32010 executes an IN or an OUT instruction, gates L1, L2, L3, L4, and L5 are required to enable reading and writing to the shift registers only on these instructions. The TBLW instruction is prohibited because it has the same timing as the OUT instruction. Flip-flop U4 ensures that the setup and hold times of SN74LS299 shift registers are met.

Although not shown in the circuit diagram, it is recommended that the $\overline{\text{CLR}}$ pins of the SN74LS299 shift registers as well as the $\overline{\text{RESET}}$ pin of the AIC be tied to the power-up reset circuit shown in the AIC data sheet. This ensures that the registers are clear when the AIC begins to transfer data and decrease the possibility that the AIC will shift in bad data which could cause the AIC to shut down or behave in an unexpected manner.

2.2 Software

The flowcharts for the communication program along with the TMS32010 program listing are presented in Appendix A. If this software is to be used, and application program that moves data into and out of the transmit and receive registers must be supplied.

2.2.1 Initializing the Digital Signal Processor

As shown in the flowcharts in Appendix A, the program begins with an initialization routine which clears both the transmit/receive-end flag and the secondary communication flag, and stores the addresses of the interrupt subroutines. The program uses the MPYK...PAC instruction sequence to load data memory locations with the 12-bit address of the subroutines. This sequence is only necessary if the subroutines are to reside in program memory locations larger than >00FF. Otherwise, the instructions LACK and SACL may be used to initialize the subroutine-address storage locations.

2.2.2 Communicating with the TLC32040

After the storage registers and status register have been initialized, the interrupt is enabled and control is passed to the user’s application routine (i.e., the system-dependent software that processes received data and prepares data for transmission). The program ignores the first interrupt that occurs after interrupts are enabled (page 22, line 207, IGINT routine), allowing the AIC to stabilize after a reset. The application routine should not write to the shift registers while data is moving into (and out of) them. In addition, it should ensure that no primary data is written to the shift registers between a primary and secondary data-communication pair. The first objecive can be accomplished by writing to the SN74LS299 shift registers as quickly as possible after the receive interrupt. The number of instruction cycles between the data transfers can be calculated from the conversion frequency. By counting instruction cycles in the application program, it is possible to determine whether the data transfer will conflict with the OUT instruction to the shift register. The second objective can be accomplished by monitoring SNDFLG in the application program. If SNDFLG is true (>00FF), secondary communication has not been completed.

When the processors receives an interrupt, the program counter is pushed onto the hardware stack and then the program counter is set to >0002, the location of the interrupt service routine, INTSVC (page 19, line 46). The interrupt service routine then saves the contents of the accumulator and the status register and calls the interrupt subroutine to which XVECT points. If secondary communication is to follow the upcoming primary communication, XVECT, is set by the application program to refer to SINT1, otherwise, XVECT defaults to NINT (i.e., the normal interrupt routine).

Because the interrupt subroutine makes one subroutine call and uses two levels of the hardware stack, the application program can only use two levels of nesting (i.e., if stack extension is not used). This means that any subroutine called by the application program can only call subroutines containing no instructions that use the hardware stack (e.g., TBLW) and that make no other subroutine calls. In addition, if the application program and communication program are being implemented on an XDS series emulator, the emulator consumes one level of the hardware stack and allows the application program only one level of nesting (i.e., one level of subroutine calls).

As shown in the flowcharts in Appendix A, the normal interrupt routine reads the A/D data from the shift registers and then sets the receive/transmit end-flag (RXEFLG). The application program must write the outgoing D/A data word to the shift registers at a time convenient to the application routine. It should have the restriction that the data be written before the next data transfer.

2.2.3 TLC32040 Secondary Communication

If it is necessary to write to the control register of the AIC or configure any of the AIC internal counters, the application program must initiate a primary/secondary communication pair. This can be accomplished by placing a data word in which bits 0 and 1 are both high into DXMT, placing the secondary control word (see program listing page 19) in D2ND, and placing the address of the secondary communication subroutine, SINT1, in XVECT. When the next interrupt occurs, the interrupt subroutine will call routine SINT1. SINT1 reads the A/D information from the shift registers and writes the secondary communication word to the shift registers.

3 TLC32040 Interface to the TMS32020/C25

3.1 Hardware Description

Because the TLC32040 is designed specifically to interface with the serial port of the TMS32020/C25, the interface requires no external hardware. Except for CLKR and CLKX, there is a one-to-one correspondence between the serial port control and data pins of TMS32020 and TLC32040. CLKR and CLKX are tied together since both the transmit and the receive operations are synchronized with SHIFT CLK of the TLC32040. The interface circuit, along with the communication program (page 26), allows the AIC to communicate with the TMS32020/C25 in both synchronous and asynchronous modes. See Figures 2, 3, and 4.

3.2 Software

The program listed in Appendix B allows the AIC to communicate with the TMS32020 in synchronous or asynchronous mode. Although originally written for the TMS32020, it will work just as well for the TMS320C25.

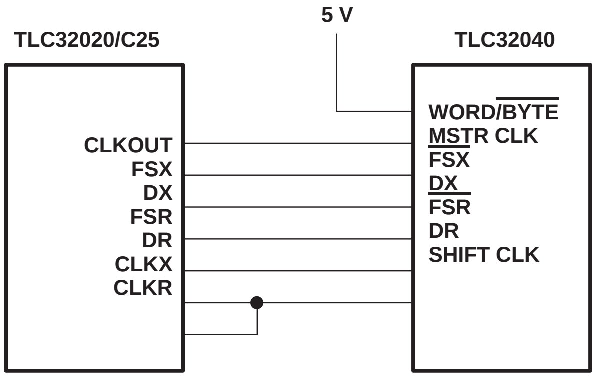
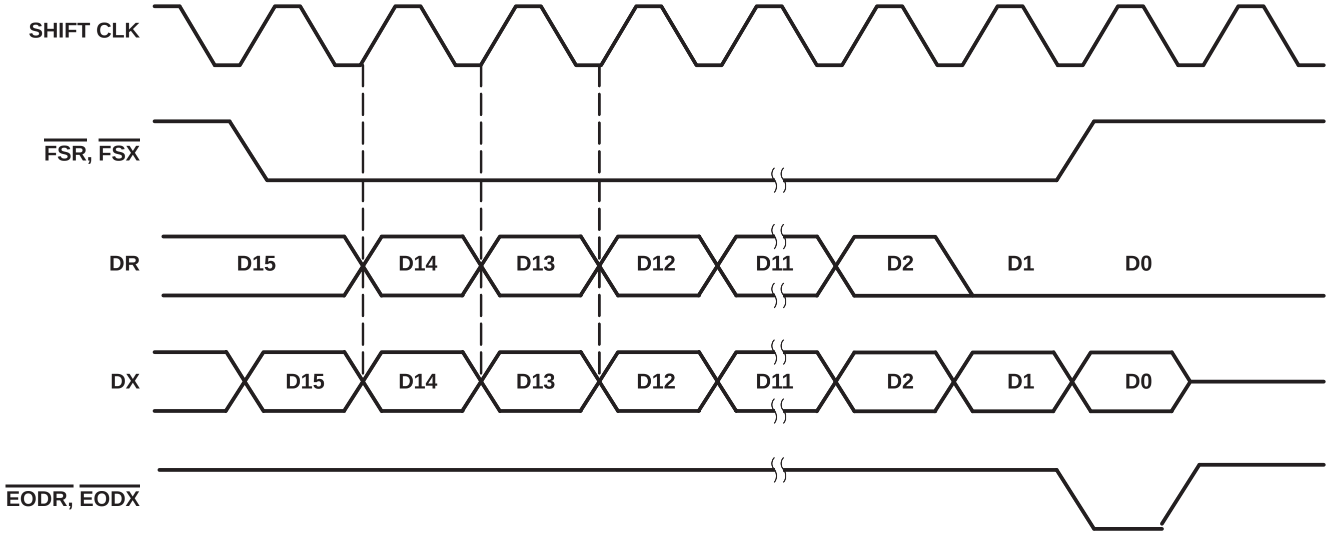


Figure 2. AIC Interface to TMS32020/C25



The sequence of operation is:
1. The FSX or FSR pin is brought low.
2. One 16-bit word is transmitted or one 16-bit byte is received.
3. The FSX or FSR pin is brought high.
4. The EODX or EODR pin emits a low-going pulse as shown.

Figure 3. Operating Sequence for AIC-TMC32020/C25 Interface

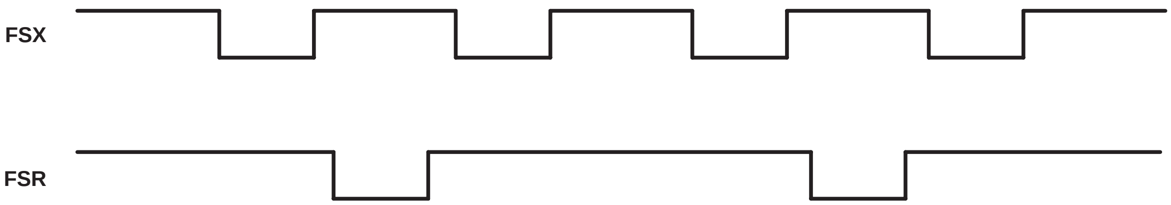


Figure 4. Asynchronous Communication AIC-TMS32020/C25 Interface

3.2.1 Initializing the TMS32020/C25

This program starts by calling the initialization routine. The working storage registers for the communication program and the transmit and receive registers of the DSP are cleared, and the status registers and interrupt mask register of the TMS32020/C25 are set (see program flow charts in Appendix B). The addresses of the transmit and receive interrupt subroutines are placed in their storage locations, and the addresses of the routines which ignore the first transmit and receive interrupts are placed in the transmit and receive subroutine pointers (XVECT and RVECT). The TMS32020/C25 serial port is configured to allow transmission of 16-bit data words (FO), the serial port format bit of the TMS32020/C25 must be set to zero) with an externally generated frame synchronization ($\overline{\text{FSX}}$ and $\overline{\text{FXR}}$ are inputs, TXM bit is set to 0).

3.2.2 Communicating with the TLC32040

After the TMS32020/C25 has been initialized, interrupts are enabled and the program calls subroutine IGR. The processor is instructed to wait for the first transmit and receive interrupts (XINT and RINT) and ignore them. After the TMS32020 has received both a receive and a transmit interrupt, the IGR routine will transfer control back to the main program and IGR will not be called again.

If the transmit interrupt is enabled, the processor branches to location 28 in program memory at the end of a serial transmission. This is the location of the transmit interrupt service routine. The program context is saved by storing the status registers and the contents of the accumulator. Then the interrupt service routine calls the interrupt subroutine whose address is stored in the transmit interrupt pointer (XVECT).

A similar procedure occurs on completion of a serial receive. If the receive interrupt is enabled, the processor branches to location 26 in program memory. As with the transmit interrupt service routine (XINT, page 30, line 226), the receive interrupt service routine (page 30, line 194) saves context and then calls the interrupt subroutine whose address is stored in the receive interrupt pointer (RVECT). It is important that during the execution of either the receive or transmit interrupt service routines, all interrupts are disabled and must be re-enabled when the interrupt service routine ends.

The main program is the application program. Procedures such as digital filtering, tone-generation and detection, and secondary communication judgment can be placed in the application program. In the program listing shown in Appendix B, a subroutine (C2ND) is provided which will prepare for secondary communication. If secondary communication is required, the user must first write the data with the secondary code to the DXMT register. This data word should have the two least significant bits set high (e.g., >0003). The first 14 bits transmitted will go to the D/A converter and the last two bits indicate to the AIC that secondary communication will follow. After writing to the SXMT register, the secondary communication word should be written to the D2ND register.

This data may be used to program the AIC internal counters or to reconfigure the AIC (e.g., to change from synchronous to asynchronous mode or to bypass the bandpass filter). After both data words are stored in their respective registers, the application program can then call the subroutine C2ND which will prepare the TMS32020 to transmit the secondary communication word immediately after primary communication.

3.2.3 Secondary Communicating – Special Considerations

This communication program disables the receive interrupt (RINT) when secondary communication is requested. Because of the critical timing between the primary and secondary communication words and because RINT carries a higher priority than the transmit interrupt, the receive interrupt cannot be allowed to interrupt the processor before the secondary data word can be written to the data-transmit register. If this situation were to occur, the AIC would not receive the correct secondary control word and the AIC could be shut down.

In many applications, the AIC internal registers need only be set at the beginning of operation, (i.e., just after initialization). Thereafter, the DSP only communicates with the AIC using primary communication. In cases such as these, the communication program can be greatly simplified.

4 TMS32040 Interface to the TMS320C17

4.1 Hardware Description

As shown in Figure 5, the TMS320C17 interfaces directly with the TLC32040. However, because the TMS320C17 responds more slowly to interrupts than the TMS32010/E15 or the TMS32020/C25, additional circuit connections are necessary to ensure that the TMS320C17 can respond to the interrupt, accomplish the context-switching that is required when an interrupt is serviced, and proceed with the interrupt vector. This must all be accomplished within the strict timing requirements imposed by the TLC32040. To meet these requirements, FSX of the TLC32040 is connected to the EXINT pin of the TMS320C17. This allows the TMS320C17 to recognize the transmit interrupt before the transmission is complete. This allows the interrupt service routine to complete its context-switching while the data is being transferred. The interrupt service routine branches to the interrupt subroutines only after the FSX flag bit has been set. This signals the end of data transmission.

The other hardware modification involves connecting the EODX pin of the TLC32040 to the BIO pin of the TMS320C17. Because the TMS320C17 serial port accepts data in 8-bit bytes (see Figure 6) and the TLC32040 controls the byte sequence (i.e., which byte is transmitted first, the high-order byte or the low-order byte) it is important that the TMS320C17 be able to distinguish between the two transmitted bytes. The EODX signal is asserted only once during each transmission pair, making it useful for marking the end of a transmission pair and synchronizing the TMS320C17 with the AIC byte sequence. After synchronization has been established, the BIO line is no longer needed by the interface program and may be used elsewhere.

Because the TMS320C17 serial port operates only in byte mode, 16-bit transmit data should be separated into two 8-bit bytes and stored in separate registers before a transmit interrupt is acknowledged. Alternatively, the data can be prepared inside the interrupt service routine before the interrupt subroutine is called. From the time that the interrupt is recognized to the end of the data transmission is equivalent to 28 TMS320C17 instruction cycles.

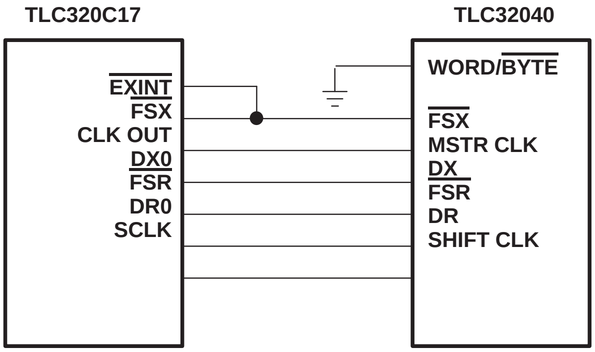
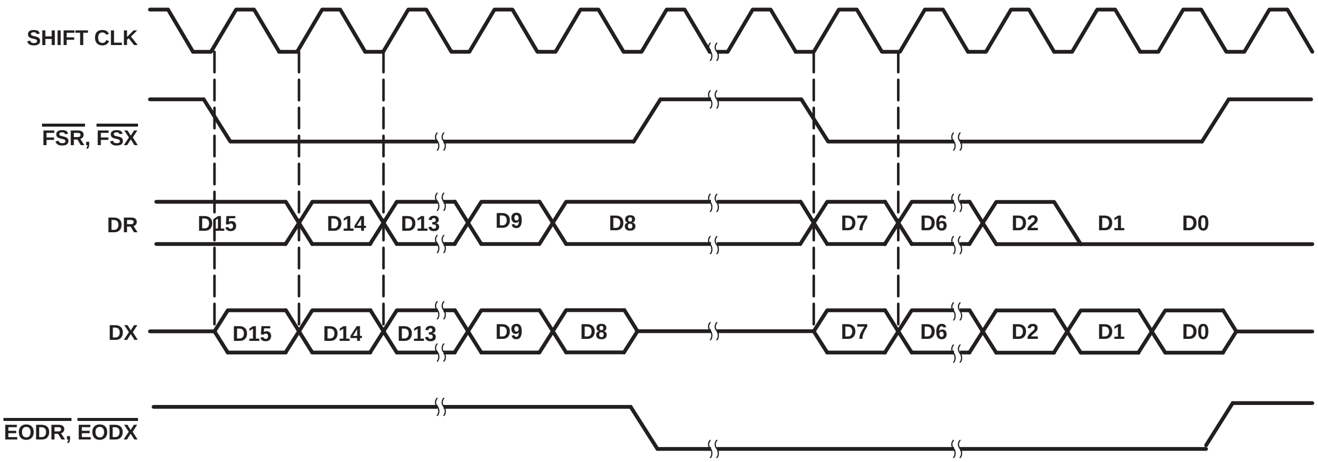


Figure 5. AIC Interface to TMS320C17


SHIFT CLK
$\overline{\text{FSR}}$, $\overline{\text{FSX}}$
DR
D15
D14
D13
D9
D8
D7
D6
D2
D1
D0
DX
D15
D14
D13
D9
D8
D7
D6
D2
D1
D0
$\overline{\text{EODR}}$, $\overline{\text{EODX}}$

The sequence of operation is:
1. The $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ pin is brought low.
2. One 8-bit word is transmitted or one 8-bit byte is received.
3. The $\overline{\text{EODX}}$ or $\overline{\text{EODR}}$ pins are brought low.
4. The $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ emit a positive frame-sync pulse that is four shift clock cycles wide.
5. One 8-bit byte is transmitted and one 8-bit byte is received.
6. The $\overline{\text{EODX}}$ and $\overline{\text{EODR}}$ pins are brought high.
7. The $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ pins are brought high.

Figure 6. Operating Sequence for AIC-TMS320C17

4.2 Software

The software listed in Appendix C only allows the AIC to communicate with the TMS320C17 in synchronous mode. This communication program is supplied with an application routine, DLB (Appendix C, program listing line 253), which returns the most recently received data word back to the AIC (digital loopback).

4.2.1 Initializing the TMS320C17

The program begins with an initialization routine (INIT, page 40, line 120). Interrupts are disabled and all the working storage registers used by the communication program are cleared. Both transmit registers are cleared, the constants used by the program are initialized and the addresses of the subroutines called by the program are placed in data memory. This enables the interrupt service routine to call subroutines located in program-memory addresses higher than 255. After the initialization is complete, the TMS320C17 monitors the $\overline{\text{FSX}}$ interrupt flag in the control register to establish synchronization with the AIC.

4.2.2 AIC Communications and Interrupt Management

Because the AIC $\overline{\text{FSX}}$ pin is tied to the $\overline{\text{EXINT}}$ line of the TMS320C17 and the delay through the interrupt multiplexer, the interrupt service routine is called four instruction cycles after the falling edge of $\overline{\text{FSX}}$. The interrupt service routine (INTSVC, Appendix C, program listing, line 90) completes its context switching and then monitors the lower control register, polling the FSX flag bit that indicates the end of the 8-bit serial data transfer. If the $\overline{\text{FSX}}$ flag bit is set, the transfer is complete. After this bit is set, control is transferred to the interrupt subroutine whose address is stored in VECT. The serial communication must be complete before data is read from the data receive register.

When no secondary communication is to follow, the interrupt subroutines, NINT1 and NINT2, are called. If data has been stored in DXMT2 (the low-order eight bits of the transmit data word), which does not indicate that secondary communication is to follow, the interrupt service routine calls NINT1 when the first 8-bit serial transfer is complete. NINT1 immediately writes the second byte of transmit data, (i.e., the contents of DXMT2) to transmit data register 0 (TR0). It then moves the first byte of the received data (i.e., the high-order byte of the A/D conversion result) into DRCV1. NINT1 then stores in VECT the address of NINT2. NINT2 is called at the end of the next 8-bit data transfer and resets the $\overline{\text{FSX}}$ interrupt flag bit by writing a logic high to it. The next interrupt (a falling edge of $\overline{\text{EXINT}}$) occurs before the interrupt service routine returns control to the main program. This is an acceptable situation since the TMS320C17, on leaving the interrupt service routine, recognizes that an interrupt has occurred and immediately responds by servicing the interrupt.

The interrupt subroutine NINT2 is similar in operation to NINT1. It stores the low-order byte of receive data (bits 7 through 0 of the A/D conversion result) and stores the address of the next interrupt subroutine in VECT. NINT2 does not write to the transmit data register, TR0. This task has been left to the application program. After the transmit data has been prepared by the main program and the data has been stored in DXMT1 and DXMT2, the main program stores the first byte of the transmit data in transmit data register 0 (TR0).

4.2.3 Secondary Communications

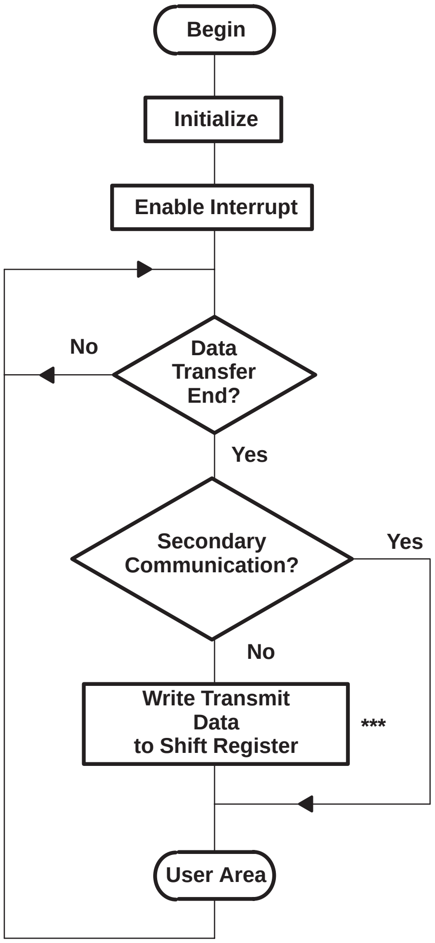
The interrupt subroutines SINT1 through SINT4 are called when secondary communication is required. For secondary communication, DXMT1 and DXMT2 will hold the primary communication word. DXMT3 and DXMT4 will hold the secondary communication word. VECT, the subroutine pointer should then be initialized to the address of SINT1. As with the normal (primary communication only) interrupt subroutines (i.e., NINT1 and NINT2), the secondary communication routines will change VECT to point to the succeeding routine (e.g., SINT1 will point to SINT2, SINT2 will point to SINT3, etc.).

5 Summary

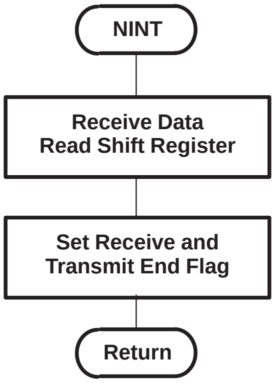
The TLC32040 is an excellent choice for many digital signal processing applications such as speech recognition/storage systems and industrial process control. The different serial modes of the AIC (synchronous, asynchronous, 8- and 16-bit) allow it to interface easily with all of the serial port members of the TMS320 family as well as other processors.

A TLC32040 and TMS32010 Flowcharts and Communication Program

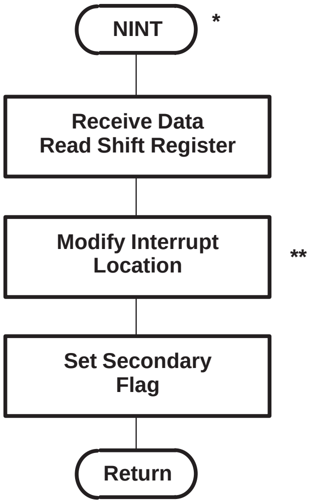
A.1 Flowcharts



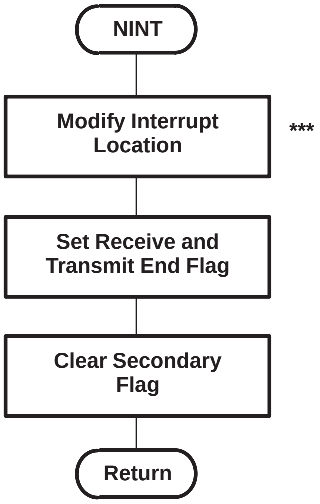
a. MAIN


NINT
Receive Data Read Shift Register
Set Receive and Transmit End Flag
Return

b. PRIMARY INTERRUPT ROUTINE



c. SECONDARY DATA COMMUNICATIONS 1



d. SECONDARY DATA COMMUNICATIONS 2

* Set, if need secondary.
** Modify to call SINT2.
*** Modify to call NINT.
**** Must execute before transfer beginning.

A.2 Communication Program List

```
            *********************************************************
            * When using this program, the circuit in the TLC32040  *
            * data sheet or its equivalent circuit must be fused    *
            * port 1 are reserved for data receiving and data       *
            * transmitting. The TBLW command is prohibited because  *
            * it has the same timing as the OUT command. TLC32040 is*
            * used only in synchronous mode.                        *
            *********************************************************
            *
      0002  RXEFLG    EQU   >02   receive and xmit end flag.
      0003  SNDFLG    EQU   >03   secondary communication flag.
      0004  DRCV      EQU   >04   receive data storage.
      0005  DXMT      EQU   >05   xmit data storage.
      0006  D2ND      EQU   >06   secondary data storage.
      0007  XVECT     EQU   >07   interrupt address storage.
      0008  ACHSTK    EQU   >08   ACCH stack.
      0009  ACLSTK    EQU   >09   ACCL stack.
      000A  SSTSTK    EQU   >0A   Status stack.
      000C  ANINT     EQU   >0C   interrupt address 1
      000D  ASINT1    EQU   >0D   interrupt address 2
      000E  ASINT2    EQU   >0E   interrupt address 3
      000F  TMP0      EQU   0F    temporary register.
            *
      00FF  SET       EQU   >FF
      0001  ONE       EQU   >01
            *    =================
            *       Reset vector.
                      =================
0000                  AORG  >0000 program start address.
0000  F900            B     EPIL  jump to initialization.
0001  000D

            *********************************************************
            * =================                                     *
            * Interrupt vector.                                     *
            * =================                                     *
            * When secondary communication, modify the content of   *
            * XVECT to the address of secondary communication and   *
            * store secondary data in D2ND.                         *
            * ex.                                                   *
            *         LAC   ASINT1,0  modify XVECT                  *
            *         SACL  XVECT,0                                 *
            *             |                                         *
            *         LAC   D2ND,0    store secondary data.         *
            *********************************************************
```

```
0002                  AORG  >0002     interrupt vector.
0002
0002  7C0A  INTSVC    SST   SSTSTK    push status register.
0003  6E01            LDPK  ONE       set data pointer one.
0004  5808            SACH  ACHSTK    push ACCH.
0005  5009            SACL  ACLSTK    push ACCL.
0006  2007            LAC   XVECT,0   load interruput address.
0007  7F8C            CALA            branch to interruupt routine.
0008  6508            ZALH  ACHSTK    pop ACCH
0009  7A09            OR    ACLSTK    pop ACCL.
000A  7B0A            LST   SSTSTK    pop stack register.
000B  7F82            EINT            enable interrupt.
000C  7F8D            RET             return from interrupt routine.
            *********************************************************
            *         ============================                  *
            *         Initialization after reset.                   *
            *         ===========================                   *
            *                                                       *
            *    Data RAM locations 82H(130) through 8FH(143),      *
            *    12 words of page 1, are reserved for this          *
            *    program. The user must set the status register     *
            *    by adding the SST command at the end of the        *
            *    the initialization routine.                        *
            *********************************************************
            *
            *
            *
000D                  AORG  $         initial program.
000D
000D  6E01  EPIL      LDPK  ONE       set data page pointer one.
000E
000E  7E01            LACK  ONE       save normal communication
000F  500F            SACL  TMP0      address to its storage.
0010  6A0F            LT    TMP0
0011  802C            MPYK  NINT
0012  7F8E            PAC
0013  500C            SACl  ANINT
0014
0014  8030            MPYK  SINT1     save secondary communication
0015  7F8E            PAC             address1 to its storage.
0016  500D            SACL  ASINT1
0017
0017  8037            MPYK  SINT2     save secondary communication
0018  7F8E            PAC             address2 to its storage.
0019  500E            SACL  ASINT2
001A
001A  803A            MPYK  IGINT     ignore interrupt once after
001B  7F8E            PAC             master reset.
001C  5007            SACL  XVECT
001D
001D  7F89            ZAC             clear flags.
001E  5002            SACL  RXFLG,0
001F
001F  5003            SACL  SNDFLG,0
```

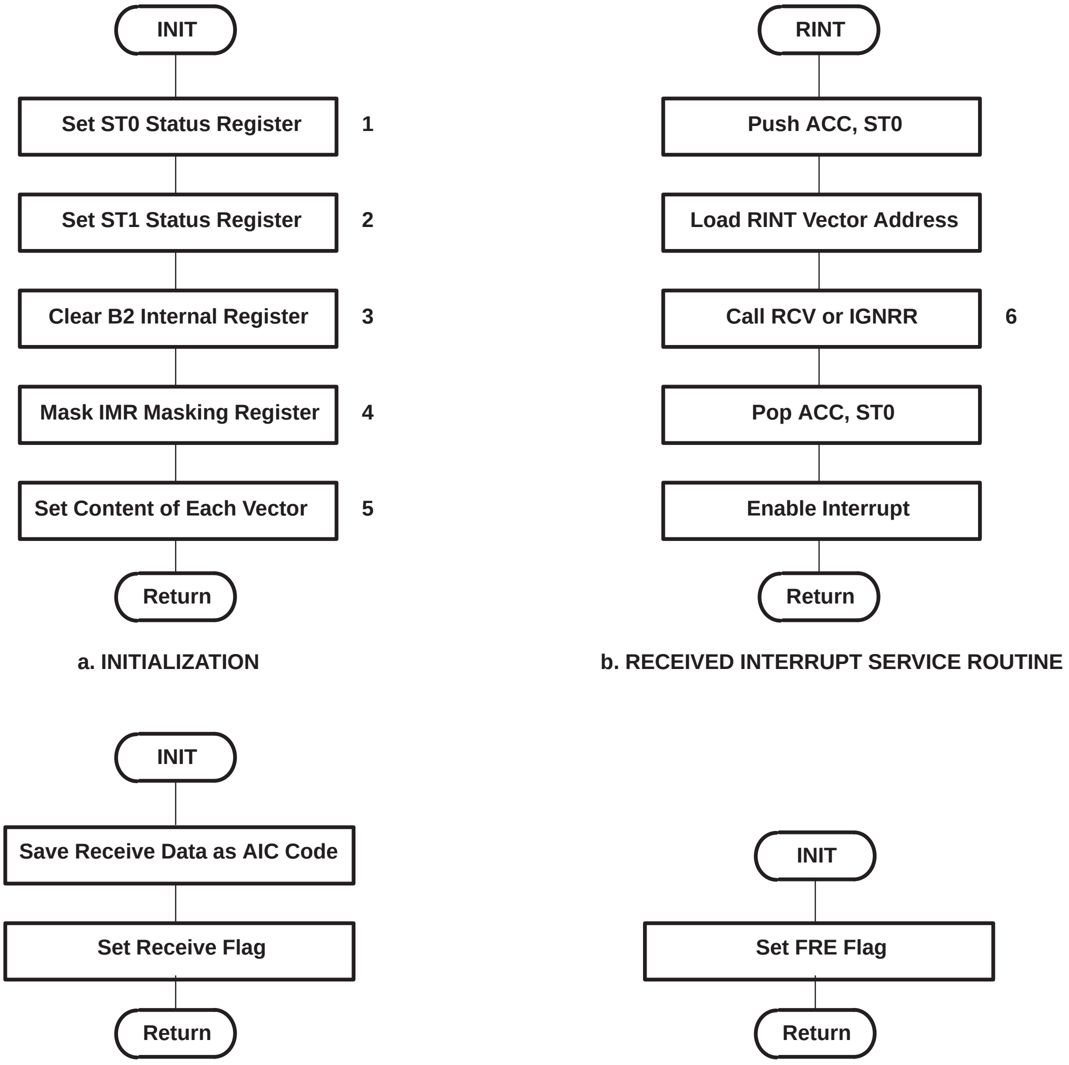
```
0020
0020
0020
0020  7F82        EINT            enable interrupt.
            *
            *********************************************************
            *           ==============                              *
            *           Main program.                               *
            *           =============                               *
            *                                                       *
            *  This program allows the user two levels of nesting   *
            *  since one level is used as stack for the interrupt.  *
            *  When the RXEFLG flag is false then no data transfer  *
            *  has ocurred, if it is true then data transfer has    *
            *  finished. User rountines such as digital filter,     *
            *  secondary-data-communication judgement etc., must be *
            *  placed in this location. Depending on the sampling   *
            *  rate (conversion rate), these user routines must     *
            *  write the xmit data to the shift registers within    *
            *  approximately 500 instruction cycles. If the user    *
            *  requires secondary communication, it will be         *
            *  necessary to delay the OUT instruction until the     *
            *  secondary data transfer has finished.                *
0021        *********************************************************
0021  2002  MAIN  LAC   RXEFLG,0  wait for interrupt.
0022  FF00        BZ    MAIN
0023  0021
0024
0024  2003        LAC   SNDFLG,0  skip OUT instruction during
0025  FE00        BNZ   MAIN1     secondary communication.
0026  0028
0027
0027  4905        OUT   DXMT,PA1  write xmit data to shift register.
0028
0028  7F89  MAIN1 ZAC             clear flags.
0029  5002        SACL  RXEFLG
002A
002A  F900        B     MAIN      loop.
002B  0021
            *
            *********************************************************
            *           ===========================                 *
            *           Normal interrupt rountine.                  *
            *           ==========================                  *
            *                     destroy ACC, DP.                  *
            *                                                       *
            *  Write the contents of DXMT to the ’LS299s, receive   *
            *  DAC data in DRCV, and set RXEFLG flag.               *
            *********************************************************
002C
002C  4004  NINT  IN    DRCV,PA0 receive data from shift register.
002D
```

```
0147 002D  7EFF        LACK  SET       set receive and xmit ended flag.
0148 002E  5002        SACL  RXEFLG
0149 002F
0150 002F  7F8D        RET             return.
0151             *
0152             ********************************************************
0153             *     ==============================================    *
0154             *     Secondary communication interrupt routine 1.      *
0155             *     ==============================================    *
0156             *                     destroy ACC,DP                    *
0157             *                                                       *
0158             *   Write the contents of D2ND to the ’LS299s, receive  *
0159             *   data in DRCV, and modify XVECT for secondary        *
0160             *   communication interrupt.                            *
0161             ********************************************************
0162 0030
0163 0030  4004  SINT1 IN    DRCV,PA0 receive data from shift register.
0164 0031
0165 0031  4906        OUT   D2ND,PA1 write secondary data to shift
0166             *           register.
0167 0032  200E        LAC   ASINT2,0  modify interrupt location.
0168 0033  5007        SACL  XVECT     secondary communication 2
0169 0034
0170 0034  7EFF        LACK  SET       set secondary communication flag.
0171 0035  5003        SACL  SNDFLG,0
0172 0036
0173 0036  7F8D        RET             return.
0174 0037
0175             ********************************************************
0176             *     ==============================================    *
0177             *     Secondary communication interrupt routine 2.      *
0178             *     ==============================================    *
0179             *                     destroy ACC,DP                    *
0180             *                                                       *
0181             *   Modify XVECT for normal communication, and set      *
0182             *   RXEFLG flag.                                        *
0183             ********************************************************
0184 0037
0185 0037  200C  SINT2 LAC   ANINT     modify interrupt location
0186 0038  5007        SACL  XVECT     normal communication.
0187 0039
0188 0039  7EFF        LACK  SET       set receive and xmit ended flag.
0189 003A  SACL  RXEFLG
0190 003B
0191 003B  7F89        ZAC             clear secondary communication flag.
0192 003C  5003        SACL  SNDFLG,0
0193 003D
0194 003D  7F8D        RET             return.
0195 003E
```

```
            **********************************************************
            *     ===========================================        *
            *     Ignoring the first interrupt after reset.          *
            *     ===========================================        *
            *                    destroy ACC,DP.                     *
            *                                                        *
            *   Ignore the first interrupt after reset. the TLC32040 *
            *   receives zero as DAC data but no ADC data in DRCV.   *
            *                                                        *
            **********************************************************
003E
003E  200C  IGINT LAC   ANINT     modify interrupt location
003F  5007        SACL  XVECT     normal communication.
0040
0040  7F8D        RET             return.
0041
                  END
NO ERRORS, NO WARNINGS
```

B TLC32040 and TMS32020 Flowcharts and Communication Program

B.1 Flowcharts



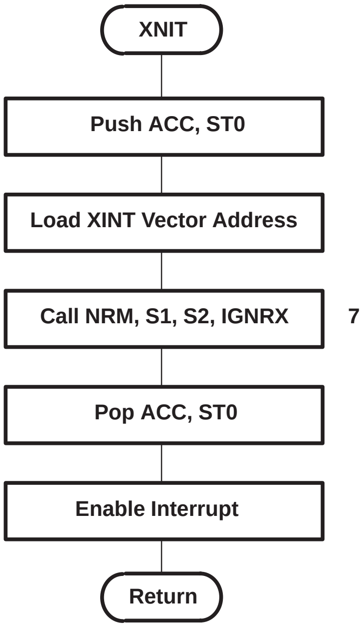
a. INITIALIZATION

b. RECEIVED INTERRUPT SERVICE ROUTINE

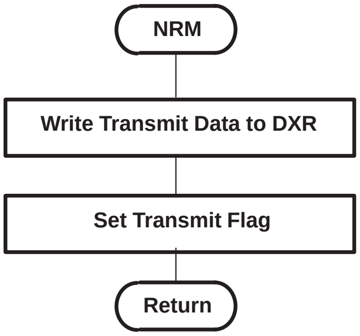
c. RECEIVE SUBROUTINE

d. IGNORE INTERRUPT

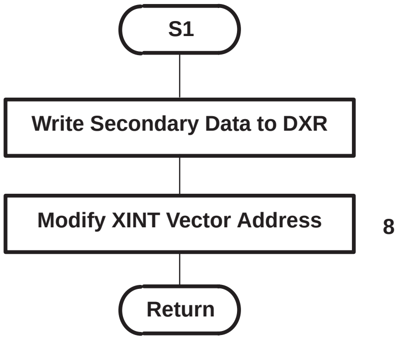
1 – Alterable AR pointer and OVM.
2 – Alterable CNF, SXM and XF.
3 – Must clear at least 108 through 127, 19 of internal RAM.
4 – If IMR is changed by user program. INST must be changed.
5 – Their contents will be changed by their routine locations.
6 – IGNRR is executed only once after reset.



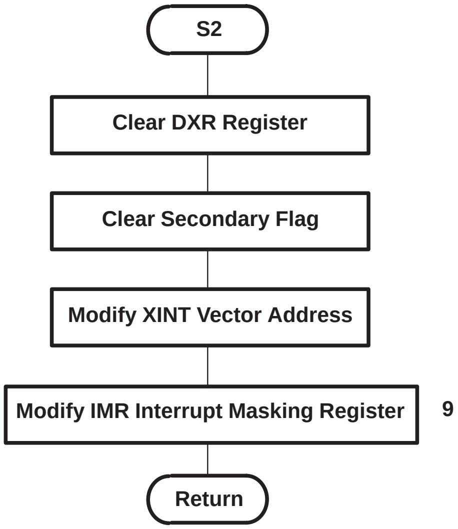
e. TRANSMIT INTERRUPT SERVICE ROUTINE



f. PRIMARY TRANSMISSION ROUTINE

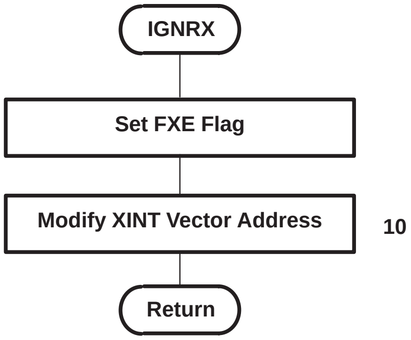


g. PRIMARY-SECONDARY COMMUNICATIONS 1

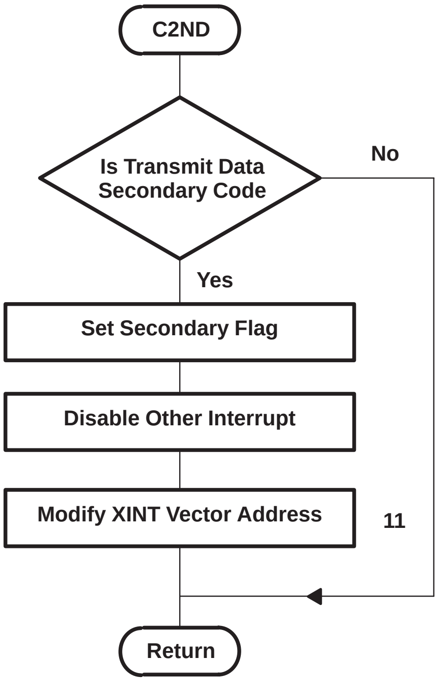

S2
Clear DXR Register
Clear Secondary Flag
Modify XINT Vector Address
Modify IMR Interrupt Masking Register
9
Return

h. PRIMARY-SECONDARY COMMUNICATIONS 2

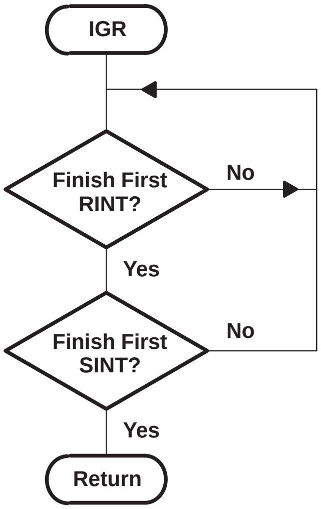
7 – IGNRX is executed only once after reset.
8 – Modify to S2 address.
9 – Modify to NRM address.



I. IGNORE TRANSMIT INTERRUPT



j. SECONDARY COMMUNICATION JUDGEMENT



k. IGNORE FIRST INTERRUPTS

10 – Modify to NRM address.
11 – Modify to S1 address.

B.2 Communication Program List

```
         *************************************************************
         * ===========================================               *
         *  TLC32040 & TMS32020 communication program.               *
         * ===========================================               *
         *            by H.Okubo & W.Rowand                          *
         *           version 1.1    7/22/88.                         *
         *                                                           *
         * This is a TMS32020 – TLC32040 communication program       *
         * that can be used in many systems. To use this program     *
         * the TMS32020 and the TLC32040 (AIC) must be connected     *
         * as shown in the publication: Linear and Interface         *
         * Circuit Applications, Volume 3. The program reserves      *
         * TMS32020 internal data memory 108 through 127 (B2) as     *
         * flags and storage. When secondary communication is        *
         * needed, every maskable interrupt except XINT is           *
         * disabled until that communication finishes.               *
         *                                                           *
         * If you have any questions, please let us know.            *
         *************************************************************
         *
         *
         * =======================
         * Memory mapped register.
         * =======================
         *
0000     DRR  EQU0           * data receive register address.
0001     DXR  EQU       1     * data xmit register address.
0004     IMR  EQU       4     * interrupt mask register address.
         *
         * ==============================================
         *    Reserved onchip RAM as flags and storages.
         *           (block B2 108 through 127.)
         * ==============================================
         *
006C     FXE  EQU       108   * ignore first XINT flag.
006D     FRE  EQU       109   * ignore first RINT flag.
006F     TMPO EQU       111   * temporary register.
0070     ACCHST EQU     12    * stack for ACCH.
0071     CCLST  EQU     113   * stack for ACCL.
0072     SSTST  EQU     114   * stack for STO register.
0073     INTST  EQU     115   * stack for IMR register.
0074     RVECT  EQU     116   * vector for RINT.
0075     XVECT  EQU     117   * vector for XINT.
0076     VRCV EQU       118   * RINT vector storage.
0077     VNRM EQU       119   * XINT vector storage.
0078     VS1  EQU       120   * secondary vector storagel.
0079     VS2  EQU       121   * secondary vector storage2.
007A     DRCV EQU       122   * receive data storage.
007B     DXMT EQU       123   * xmit data storage.
007C     D2ND EQU       124   * secondary data storage.
007D     FRCV EQU       125   * receive flag.
007E     FXMT EQU       126   * xmit flag.
007F     F2ND EQU       127   * secondary communication flag.
         *
```

```
   0055             *******************************************************
   0056             * Processor starts at this address after reset.       *
   0057             *                                                     *
   0058 0000               AORG     0   * program start address.          *
   0059 0000   FF80        B     STRT   * jump to initialization routine. *
        0001   0020
   0060             *******************************************************
   0061             *
   0062             *******************************************************
   0063             * Receive interrupt location.                         *
   0064             *                                                     *
   0065 001A               AORG     26  * Rint vector.                    *
   0066 001A   FF80        B   RINT     * jump to receive interrupt       *
        00IB   004A                     * routine.                        *
   0067             *******************************************************
   0068             *
   0069             *******************************************************
   0070             * Transmit interrupt location.                        *
   0071             *                                                     *
   0072 ODIC               AORG     28  * Xint vector.                    *
   0073 001C   FF80        B      XINT  * jump to xmit interrupt routine. *
        001D   005A
   0074             *******************************************************
   0075             *
   0076 0020               AORG     32  * start initial program.
   0077             *
   0078             *******************************************************
   0079             * User must initialize DSP with the routine INIT.     *
   0080             * The user may modify this routine to suit his        *
   0081             * requirements as he likes.                           *
   0082             *******************************************************
   0083 0020   FE80 STRT   CALL INIT    *
        0021   0025
   0084 0022   CE00        EINT         * enable interrupt.
   0085 0023   FE80        CALL IGR
        0024   008D
0086                *
```

```
      ****************************************************************
      *                ================                             *
      *                   User area                                 *
      *                ================                             *
      *                           *
      * This program allows the user two levels of nesting,         *
      * since two levels are used as stack for the interrupt.       *
      * When the FXMT flag is false no data has occurred             *
      * When the FRCV flag is false, no data has been                *
      * received. As those flags are not reset by any                *
      * routine in this program, the user must reset the             *
      * flags if he chooses to use them and note that >00ff          *
      * means true, >0000 means false. User routines such as         *
      * digital filtering, FFTs etc. must be placed in this          *
      * location. Depending on the sampling rate (conver-            *
      * sion rate), these user routines must write the xmit          *
      * data to the DXMT registers within approximately 500          *
      * instruction cycles. If the user requires secondary           *
      * communication, data with the secondary code (xxx             *
      * xxxx xxxx xx11) should first be written to DXMT and          *
      * then secondary data should be written to D2ND. Next,         *
      * a call should be made to C2ND to set up SVECT and the        *
      * F2ND flag to perform the secondary communication.            *
      * Note that all maskable interrupts except XINT are            *
      * disabled until secondary communication has completed.        *
      ****************************************************************
      *
      ****************************************************************
      *           ======================                             *
      *           Initialization routine.                            *
      *           ======================                             *
      * This routine initializes the status registers, flags,        *
      * vector storage contents and internal data locations          *
      * 96 through 107. Note that the user can modify these          *
      * registers (i.e., STO ST1 IMR), as long as the contents       *
      *do not conflict with the operation of the AIC.                *
      ***********************************************************
0025  C800 INIT  LDPK  0       * set status0 register.
0026  D00I LALK  >OE00,0       * 0000 1110 0000 D000B
 0027 OE00
0028  606F SACL  TMPO,0        * ARP=0 AR pointer 0
0029  506F LST   TMPO          * OV =0 (Overflow reg–clear)
      *                        * OVM=1 (Overflow mode set to 1)
      *                        * ? =1 Not affected.
      *                        * INTM=1 Not affected
      *                        * DP 000000000 page 0
      *
      *                        * set statusl register.
      *
002A  D00I LALK  >03F0         * 0000 0011 1111 0000B
 002B 03F0
002C  606F SACL  TMPO,0        * APB=0
002D  516F LST1  TMPO          * CNF=0 (Set B0 data memory)
002E
```

```
0139           *                   * TC =0
0140           *                   * SXM=1 (enable sign extend mode.)
0141           *                   * D9–D5=111111 not affected.
0142           *                   * F=1 (XF pin status.)
0143           *                   * F0=0 (16–bit data transfer mode.)
0144           *                   * TXM=0 (FSX input)
0145           *
0146           *
0147           *
0148 002E CA00       ZAC           * clear registers
0149 002F 6001       SACL  DXR,0   *
0150 0030 6000       SACL  DRR,0   *
0151 0031 C060       LARK  AR0,96  * clear Block B2.
0152 0032 CBIF       RPTK 31       *
0153 0033 6OA0       SACL  +,0     *
0154           *
0155           *  Interrupt masking
0156           *
0157 0034 CA30       LACK  >30     * 0000 0000 0011 0000B
0158 0035 6004       SACL  IMR,0   * INT ________|| ||||
0159 0036 6073       SACL  INTST,0 * RINT_________| ||||
0160                               * TINT___________||||
0161           *                   * INT2____________|||
0162           *                   * INT1_____________||
0163                               * INTO______________|
0164           *
0165 0037 D001       LALK  NRM,0   * normal xint routine address.
     0038 0067
0166 0039 6077       SACL  VNRM,0  *
0167
0168 003A D001       LALK  Sl,0    * secondary xint routine address 1.
     003B 006C
0169 003C 6078       SACL  VS1,0   *
0170           *
0171 003D D001       LALK  S2,0    * secondary xint routine address 2.
     003E 0071
0172 003F 6079       SACL  VS2,0   *
0173           *
0174 0040 D001       LALK  RCV,0   * rint routine address.
     0041 0055
0175 0042 6076       SACL  VRCV,0
0176           *
0177 0043 D001       LALK  IGNRR,0 * set ignore first rint address.
     0044 0094
0178 0045 6074       SACL  RVECT,0
0179           *
0180 0046 D001       LALK  IGNRX,0 * set ignore first xint address.
     0047 0099
0181 0048 6075       SACL  XVECT,0
0182 0049 CE26       RET           * return.
0183 004A
```

```
          *
          ***********************************************************
          *           ============================              *
          *            Receive interrupt routine.               *
          *           ============================              *
          * This routine stores receive data in its storage DRCV  *
          * (112 page0) and sets the receive flag FRCV (125 page0)*
          * As two levels of nesting are used, this routine       *
          * allows the user two levels, without stack extension . *
          ***********************************************************
004A 7872 RINT  SST   SSTST       * push STO register.
004B C800       LDPK 0            * data pointer page 0.
004C 6071       SACL  ACCLST,0    * push ACCL.
004D 6870       SACH  ACCHST,0    * push ACCH.
004E 2074       LAC   RVECT,0     * load ACC vector address.
004F CE24       CALA
0050 4171       ZALS  ACCLST      * pop ACC
0051 4870       ADDH  ACCHST
0052 5072       LST   SSTST       * pop ST register.
0053 CE00       EINT              * enable interrupts.
0054 CE26       RET               * return.
          *
0055 2000 RCV   LAC   DRR,0       * load data from DRR.
0056 607A       SACL  DRCV,0      * save it to its storage.
0057 CAFF       LACK  >FF         * set receive flag.
0058 607D       SACL  FRCV        *
0059 CE26       RET               * return.
          *
          ***********************************************************
          *           ===========================               *
          *             xmit interrupt routine.                 *
          *           ===========================               *
          * This routine writes xmit data C%the contents of DXMT  *
          * (123 page0)) to the DXR register according to the type*
          * of communication, i.e. normal communication or secondary *
          * communication. For normal communication, call the normal *
          * communication routine (NRM). For secondary, call the  *
          * secondary communication routines (Sl and S2). Because *
          * these routines use two levels of nesting, the user is *
          * allowed two levels of nesting if stack extension is   *
          * not used.                                             *
          ***********************************************************
005A 7872 XINT  SST   SSTST       * push ST register.
005B C800       LDPK  0           * data pointer page 0.
005C 6071       SACL  ACCLST,0    * push ACCL.
005D 6870       SACH  ACCHST,0    * push ACCH.
005E 207C       LAC   D2ND,0      * preload dxr with secondary
005F 6001       SACL  DXR,0       * communication data.
0060 2075       LAC   XVECT,0     * load vector address.
0061 CE24       CALA              * call xmit routine.
0062 4171       ZALS  ACCLST      * POP ACC
0063 4870       ADDH  ACCHST
0064 5072       LST   SSTST       * pop ST register.
0065 CE00       EINT              * enable interrupt.
0066 CE26       RET               * return.
```

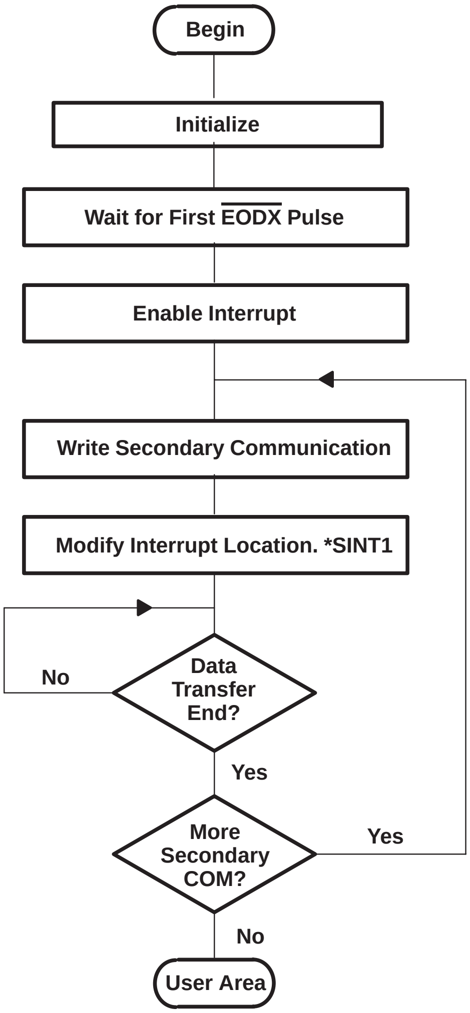
```
          ***********************************************************
          *           =============================               *
          *             Normal data write routine.                *
          *           =============================               *
          * This routine is called when normal communication occurs.*
          * This routine writes xmit data to DXR, and sets the    *
          * transmit flag (126 page0).                            *
          ***********************************************************
          *
0067 207B NRM    LAC  DXMT,0      * write DXR data.
0068 6001        SACL DXR,0
0069 CAFF        LACK >FF         * set flag.
006A 607E        SACL FXMT
006B CE26       RET               * return.
          ***********************************************************
          *           ==================================          *
                       Secondary data write routine 1.            *
          *           ==================================          *
          * This routine is called when secondary communication   *
          * occurs. It writes secondary data to DXR, and modifies *
          * the content of XVECT(117 page0) for continuing secondary*
          * communication.                                        *
          ***********************************************************
006C 207  S1     LAC  D2ND,0      * write DXR 2nd data.
006D 6001        SACL DXR,0
006E 2079        LAC  VS2,0       * modify for next XINT.
006F 6075        SACL XVECT,0
0070 CE26        RET              * return.
          *
          ***********************************************************
          *           ==================================          *
          *           Secondary data writing routine 2.           *
          *           ==================================          *
          *
          * This routine is called when secondary communication   *
          * occurs. It writes dummy data to DXR to ensure that    *
          * secondary communication is not inadvertently          *
          * initiated on the next XINT. It also modifies the      *
          * content of XVECT for normal communication.            *
          ***********************************************************
0071 CA00 S2     ZAC              * clear data for protection.
0072 6001        SACL DXR,0       * of double secondary communication.
0073 607F        SACL F2ND        * clear secondary flag.
0074 CAFF        LACK >FF         * set xmit end flag.
0075 607E        SACL FXMT,0
0076 2077        LAC  VNRM,0      * set normal communication vector.
0077 6075        SACL XVECT,0
0078 2073        LAC  INTST,0     * enable all interrupts.
0079 6004        SACL IMR,0
007A CE26        RET              * return.
```

```
          *
          ***********************************************************
          *           ======================                    *
          *            Check secondary code.  destroy DP pointer. *
          *           ======================        ACC.        *
          *                                                     *
          * This routine checks whether the data in DXMT (123 pageo)*
          * has secondary code or not. If secondary code exists,  *
          * then disable maskable interrupts except XINT, modify the*
          * contents of XVECT(117 page0) for secondary communi-   *
          * cation, and set secondary flag. Note that we recommend*
          * calling this routine to send control words to the AIC.*
          ***********************************************************
007B C800 C2ND    LDPK 0         * data page pointer 0.
007C CA03         LACK 03
007D 606F         SACL TMPO
007E 207B         LAC  DXMT,0    * is this data secondary code
007F 4E6F         AND  TMPO
0080 106F         SUB  TMPO,0
0081 F680         BZ   C2ND1     * if yes, then next.
0082 0084
0083 CE26         RET            * else return.
          *
0084 CAFF C2NDI   LACK >FF       * set secondary flag.
0085 607F         SACL F2ND,0
0086 CA20         LACK >20       * enable only XINT.
0087 6004         SACL IMR,0
0088 2078         LAC  VSI,0     * modify vector address for secondary
0089 6075         SACL XVECT,0   * communication.
00BA 207B         LAC  DXMT,0    * write primary data to DXR.
008B 6001         SACL DXR,0
00BC CE26         RET            * return.
          *
          ***********************************************************
          *                                                     *
          *           ======================                    *
          *           Check first interrupt                     *
          *           ======================                    *
          *                                                     *
          * This routine checks if both first interrupts have     *
          * occurred. If this routine is called after reset, it   *
          * waits for both interrupts then returns.               *
          ***********************************************************
008D 206D IGR     LAC  FRE,0     * check first interrupt after
008E F680         BZ   IGR       * master reset.
008F 008D
0090 206C         LAC  FXE,0
0091 F680         BZ   IGR
0092 008D
0093 CE26         RET
0094
```

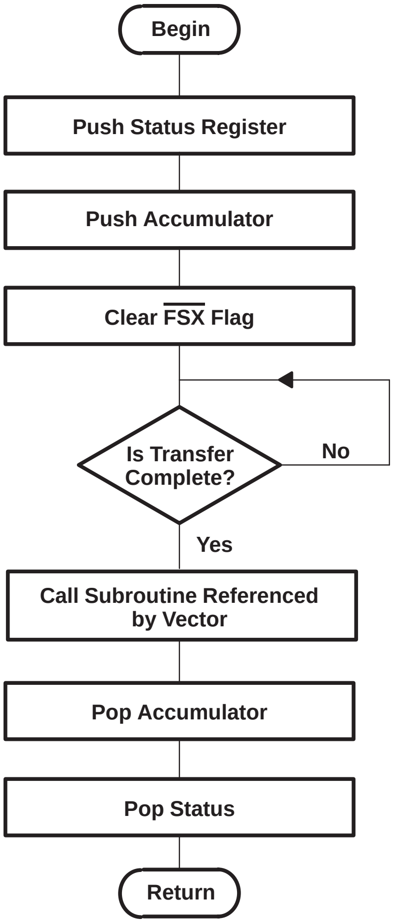
```
          *
          ***********************************************************
          *           ==========================                  *
          *            Ignore interrupt routine.                  *
          *           ==========================                  *
          * These routines are used so that the first RINT and    *
          * XINT after the %DSP reset can be ignored. They set    *
          * flags and modify each vector address to the normal    *
          * interrupt address but do not read or write to the     *
          * serial ports. Note that the first data that the AIC will*
          * receive after the DSP reset is >0000.                 *
          ***********************************************************
0094 CAFF IGNRR  LACK >FF
0095 606D        SACL FRE,0
0096 2076        LAC  VRCV,0     * set normal receive address.
0097 6074        SACL RVECT,0    *
0098 CE26        RET             * return.
          *
0099 CAFF IGNRX  LACK >FF
009A 606C        SACL FXE,0
009B 2077        LAC  VNRM,0     * set normal xmit address.
009C 6075        SACL XVECT,0    *
009D CE26        RET             * return.
          *
                 END
NO ERRORS, NO WARNINGS
```

C TLC32040 and TMS320C17 Flowcharts and Communication Program

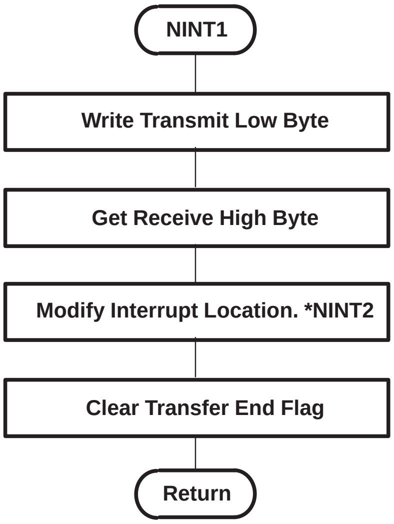
C.1 Flowcharts


Begin
Initialize
Wait for First $\overline{\text{EODX}}$ Pulse
Enable Interrupt
Write Secondary Communication
Modify Interrupt Location. *SINT1
No
Data Transfer End?
Yes
More Secondary COM?
Yes
No
User Area

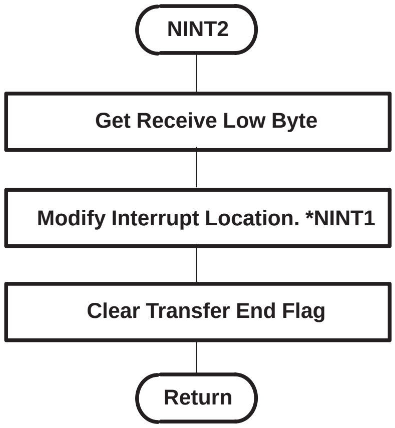
a. MAIN



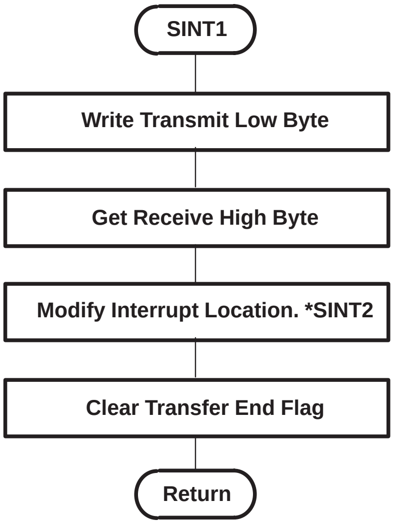
b. INTERRUPT SERVICE ROUTINE



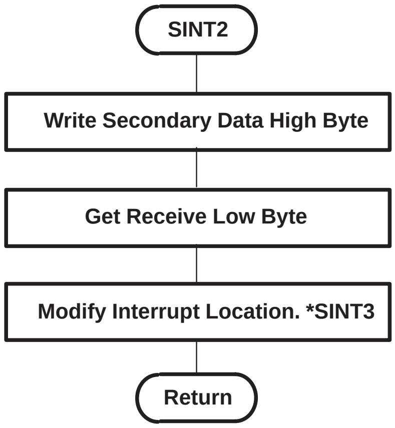
c. PRIMARY COMMUNICATION 1



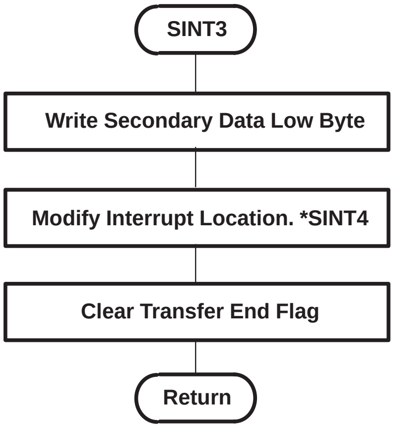
d. PRIMARY COMMUNICATION 2



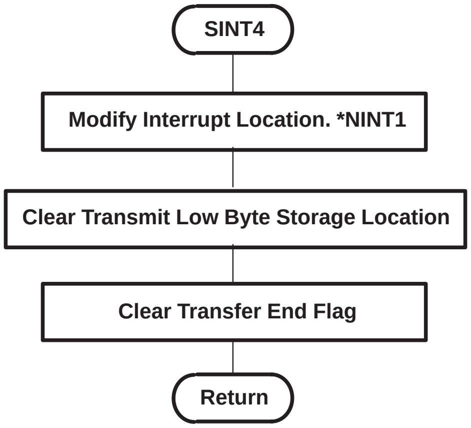
e. PRIMARY-SECONDARY COMMUNICATION 1


SINT2
Write Secondary Data High Byte
Get Receive Low Byte
Modify Interrupt Location. *SINT3
Return

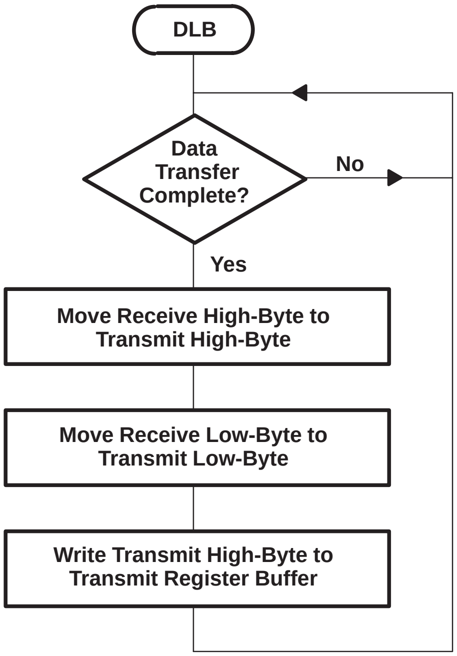
f. PRIMARY-SECONDARY COMMUNICATION 2


SINT3
Write Secondary Data Low Byte
Modify Interrupt Location. *SINT4
Clear Transfer End Flag
Return

g. PRIMARY-SECONDARY COMMUNICATION 3


SINT4
Modify Interrupt Location. *NINT1
Clear Transmit Low Byte Storage Location
Clear Transfer End Flag
Return

h. PRIMARY-SECONDARY COMMUNICATION 4


DLB
Data Transfer Complete?
No
Yes
Move Receive High-Byte to Transmit High-Byte
Move Receive Low-Byte to Transmit Low-Byte
Write Transmit High-Byte to Transmit Register Buffer

i. DIGITAL LOOPBACK

C.2 Communication Program List

```
     ***********************************************************
     *                                                       *
     *=========================================================*
     *      TLC32040 to TMS32OC17 Communication Program      *
     *                  version 1.2                          *
     *                revised 7/22/88                        *
     *                                                       *
     *          by Hironori Okubo and Woody Rowand           *
     *                  Texas Instruments                    *
     *                    (214) 997–3460                     *
     *=========================================================*
     *                                                       *
     * This program uses the circuit published in the Volume *
     * 3 of the Linear and Interface Circuit Applications    *
     * book with the following modification:                 *
     *       *
     *   1. INT– of the TMS32OC17 must be connected to       *
     *      EODX– of the TLC32040.                           *
     *                                                       *
     *                                                       *
     * In this configuration, the program will allow the     *
     * TLC32040 to communicate with the TLC32OC17 with the   *
     * restriction that all interrupts except INT– are       *
     * prohibited and only synchronous communication can     *
     * occur. The program allows the user two levels of      *
     * nesting in the main program; the remaining two levels *
     * are reserved for the interrupt vector and subroutines.*
     *                                                       *
     * If desired, this program may be used with the TMS32011*
     * digital signal processor with the following change.   *
     * Since the TMS32011 has only sixteen words of data RAM *
     * on data page 1, all of the registers used by this     *
     * program should be moved to data page 0, except for    *
     * SSTSTK (the temporary storage location for the status *
     * register) which must remain on page %I (since the     *
     * SST instruction always addresses page 1).             *
     *                                                       *
     ***********************************************************
0000 SSTSTK  EQU  >00   stack for status (SST) register.
0001 ACHSTK  EQU  >01   stack for accumulator high (ACCH).
0002 ACLSTK  EQU  >02   stack for accumulator low (ACCL).
0003 RXEFLG  EQU  >03   xmit/receive in progress.
0004 DRCV1   EQU  >04   storage for high byte receive data.
0005 DRCV2   EQU  >05   storage for low byte receive data.
0006 DXMT1   EQU  >06   storage for high byte xmit data.
0007 DXMT2   EQU  >07   storage for low byte xmit data.
0008 DXMT3   EQU  >08   storage for high byte secndry data.
0009 DXMT4   EQU  >09   storage for low byte secndry data.
D00A VECT    EQU  >OA   storage for interrupt vector addr.
000B ANINT1  EQU  >OB   storage for normal xmit/rcv vect 1.
000C ANINT2  EQU  >OC   storage for normal xmit/rcv vect 2.
000D ASINTI  EQU  >OD   storage for secndry xmit/rcv vect 1.
000E ASINT2  EQU  >OE   storage for secndry xmit/rcv vect 2.
000F ASINT3  EQU  >OF   storage for secndry xmit/rcv vect 3.
```

```
0055 0000
0056      0010   ASINT4  EQU   >10   storage for secndry xmit/rcv vect 4.
0057      0011   CNTREG  EQU   >11   storage for control register.
0058      0012   MXINT   EQU   >12   storage for xmit interrupt mask.
0059      0013   CLRX    EQU   >13   storage for xmit interrupt clear
0060      0014   CLRX1   EQU   >14   storage for xmit intrpt clear/mask.
0061      0015   TEMP    EQU   >15   temporary register.
0062      00FF   FLAG    EQU   >FF   flag set.
0063             *    =========================
0064             *        Branch to initialization routine.
0065             *   =========================
0066 0000            AORG      >0000
0067 0000 F900       B   INIT  branch to initialization routine.
     0001 0013
0068   00020069  ***********************************************************
0070             *  =======================                             *
0071             *   Interrupt service routine.                         *
0072             * =======================                              *
0073             *                                                      *
0074             *   To initiate secondary communication, change the    *
                 *   contents of VECT to the address of the secondary   *
0075             *   communication subroutine and store the             *
0076             *   in DXMT3 and DXMT4.                                *
0077             *                                                      *
0078             *                                                      *
0079             * e.g.                    *
0080             *   LAC ASINTI      modify VECT.                       *
0081             *   SACL      VECT                                     *
0082             *      |                                               *
0083             *   LAC Hl          store high-byte of                 *
0084             *   SACL      DXMT3       secondary information in     *
0085             *   LAC H2          DXMT4 store low-byte in DXMT4.     *
0086             *   SACL      DXMT4                                    *
0087             *                                                      *
0088             *******************************************************
0089 0002               AORG  >02
0090 0002 6EOI   INTSVC LDPK  1
0091 0003 7C00          SST   SSTSTK      push status register.
0092 0004 5801          SACH  ACHSTK      push accumulator high.
0093 0005 5002          SACL  ACLSTK      push accumulator low.
0094 0006 4813          OUT   CLRXPAO     make sure FSX–flag is clear.
0095 0007 4011   WAIT1  IN    CNTREG,PAO  read control register.
0096 0008 2011          LAC   CNTREG,0    load accumulator with control
0097 0009 7912          AND   MXINT       reg mask-off xmit interrupt
0098 000A FF00          BZ    WAIT1       flag loop until xmit interrupt
     000B 0007                            flag is recognized.
0099 000C
0100             *
0101 000C 200A          LAC   VECT        load acc with interrupt vector.
0102 000D 7F8C          CALA              call appropriate xmit/rcv
                                          routines
0103 000E 6501          ZALH  ACHSTK      pop accumulator high.
0104 000F 7A02          OR    ACLSTK      pop accumulator low.
0105 0010 7B00          LST   SSTSTK      pop status register.
0106 0011 7F82          EINT              enable interrupts.
0107 0012 7F8D          RET               return to main program.
```

```
0108  0013
0109            ***********************************************************
0110            *     ==============================                      *
0111            *       Initialization after reset.                       *
0112            *     ==============================                      *
0113            *                                                         *
0114            *     Data RAM locations >80 through >92 are reserved     *
0115            *     by this program. The user must set the status       *
0116            *     register at the end of this program with the SST    *
0117            *     command or a combination of SOVM, LDPK etc.         *
0118            *                                                         *
0119            ***********************************************************
0120  0013 7F81 INIT  DINT                   disable interrupts.
0121  0014 6EOl       LDPK    1              set data page pointer one.
0122  0015 7F89       ZAC                    clear registers.
0123  0016 6880       LARP    0
0124  0017 7083       LARK    0,RXEFLG+>80
0125  0018 5OA8       SACL    *+
0126  0019 5OA8       SACL    *+
0127  001A 5OA8       SACL    *+
0128  001B 5OA8       SACL    *+
0129  001C 5OA8       SACL    *+
0130  001D 5OA8       SACL    *+
0131  001E 5OA8       SACL    *+
0132  001F 5088       SACL    *
0133  0020 4906       OUT     DXMT1,PAl      clear transmit registers.
0134  0021 4906       OUT     DXMT1,PAI
0135  0022 7EO4       LACK    ?00000100
0136  0023 5012       SACL    MXINT          initialize xmit–int mask.prepare
0137  0024 7EOl       LACK    1              for serial port initialization
0138  0025 5015       SACL    TEMP           and initialization of registers
0139  0026 6Al5       LT      TEMP           containing 16-bit constants.
0140  0027 80A1       MPYK    CLX1           initialize interrupt flag clear.
0141  0028 7F8E       PAC
0142  0029 6713       TBLR    CLRX
0143  002A 8OA2       MPYK    CLX2           initialize interrupt flag clear
0144  002B 7F8E       PAC                    with interrupts disabled.
0145  00IC 6714       TBLR    CLRX1
0146  002D 809D       MPYK    IGN
0147  002E 7F8E       PAC
0148  002F 500A       SACL    VECT           initialize interrupt vector.
0149  0030 8077       MPYK    NINT1          save normal communication
0150  0031 7F8E       PAC                    address to its storage.
0151  0032 500B       SACL    ANINT1
0152  0033 807D       MPYK    NINT2          save normal communication
0153  0034 7F8E       PAC                    address 2 to its storage.
0154  0035 500C       SACL    ANINT2
0155  0036 8084       MPYK    SINTI          save secondary communication
0156  0037 7F8E       PAC                    address 1 to its storage.
0157  0038 500D       SACL    ASINT1
0158  0039 808A       MPYK    SINT2          save secondary communication
0159  003A 7F8E       PAC                    address 2 to its storage.
0160  003B 500E       SACL    ASINT2
0161  003C 8090       MPYK    SINT3          save secondary communication
0162  003D 7FBE       PAC                    address 3 to its storage.
0163  003E 600F      SACL     ASINT3
0164  003F A095      MPYK     SINT4      save secondary communication
```

```
0165 0040 CE14      PAC        address 4 to its storage.
0166 0041 6010      SACL     ASINT4
0167 0042
0168           ***********************************************************
0169           *  ==========================================            *
0170           *    Synchronize high/low byte transmission.             *
0171           *  ==========================================            *
0172           *                                                        *
0173           * The time between FSX– interrupts is approximately      *
0174           * ten microseconds (50 cycles). Wait for first if        *
0175           * FSX–, this is the first interrupt, delay 60 cycles     *
0176           * (past the second interrupt). If it is the second       *
0177           * interrupt, no harm done.                               *
0178           *                                                        *
0179           ***********************************************************
0180 0042 E014         OUT   CLRX1,PAO  clear interrupt flags,disable int.
0181 0043 8011 IGNOR   IN    CNTREG,PAO read control register.
0182 0044 2011         LAC   CNTREG     wait
0183 0045 4E12         AND   MXINT         for
0184 0046 F680         BZ    IGNOR            FSX– flag.
     0047 0043
0185 0048
0186 0048 C014         LARK  0,20       wait 60 cycles (20 × 3 cycles) in
0187 0049 5500 IGNOR1  NOP              case FSX– int. is first of the
                                        pair.
0188 004A FB90         BANZ  IGNOR1     if FSXI– int was the second, delay
     004B 0049
0189 004C
0190 004C E013         OUT CLRX,PAO     anyway.
0191 004D
0192 004D CEDO         EINT             enable interrupt.
0193           ***********************************************************
0194           *    ==============================                      *
0195           *        Main program (user area)                        *
0196           *    ==============================                      *
0197           *                                                        *
0198           * This program allows the user two levels of nesting,    *
0199           * since one level is used as stack for the interrupt and *
0200           * the interrupt service routine makes one subroutine     *
0201           * call. User routines such as digital filtering, FFTS,   *
0202           * and secondary communication judgement may be placed    *
0203           * here. The number of instruction cycles between         *
0204           * interrupts varies with the sampling rate. In the       *
0205           * power-up condition this is approximately 500 cycles.   *
0206           *                                                        *
0207           * In the example below, the first two transmissions send *
0208           * secondary information to the AIC. The first configures *
0209           * the TB and RB registers. The second configures the     *
0210           * control register.                                      *
0211           *                                                        *
0212           ***********************************************************
0213 004E CA00 MAIN    ZAC              prepare first control word.
0214 004F 6006         SACL  DXMTI
0215 0050 CA03         LACK  >03
0216 0051 5007      SACL     DXMT2                   should be xxxx xx11.
0217 0052 7E24      LACK     >24
```

```
0218  0053  5008     SACL  DXMT3
0219  0054  7E92     LACK  >92
0220  0055  5009     SACL  DXMT4
0221  0056  200D     LAC   ASINT1      set VECT for secondary
0222  0057
0223  0057  500A     SACL  VECT     communications.
0224  0058  4906     OUT   DXMTI,PAI   store first transmit byte in
0225           *              transmit buffer.
0226  0059  7F89     ZAC
0227  005A  5003     SACL  RXEFLG      clear xmit/rcv end flag.
0228  005B  2003  MAIN1 LAC   RXEFLG
0229  005C  FF00  BZ       MAINI    wait for data transfer to
      005D  005B                 complete.
0230  005E
0231  005E  7F89     ZAC         prepare second control word.
0232  005F  5006     SACL  DXMT1
0233  0060  7EO3     LACK  >03
0234  0061  5007     SACL  DXMT2
0235  0062  7E00     LACK  >00
0236  0063  5008     SACL  DXMT3
0237  0064  7E67     LACK  >67
0238  0065  5009     SACL  DXMT4
0239  0066  200D     LAC   ASINTI
0240  0067  500A     SACL  VECT
0241  0068  4906     OUT   DXMTI,PAL
0242  0069  7F89     ZAC
0243  006A  5003     SACL  RXEFLG      clear xmit/rcv end flag.
0244           **********************************************************
0245           *  =========================  *
0246           *     Digital loop–back program  *
0247           *  =========================  *
0248           *              *
0249           *  This program serves as an example of what can   *
0250           *  be done in the user area.  *
0251           *              *
0252           ***********************************************************
0253  006B  2003  DLB   LAC   RXEFLG   wait for data transfer to complete.
0254  006C  FF00     BZ    DLB
      006D  006B
0255  006E
0256  006E  2004     LAC   DRCV1    move receive data to transmit
0257  006F  5006     SACL  DXMT1    registers.
0258  0070  2005     LAC   DRCV2
0259  0071  5007     SACL  DXMT2
0260  0072  4906     OUT   DXMTI,PAL   write first transmit byte to
0261           *              transmit buffer.
0262  0073  7F89     ZAC
0263  0074  5003     SACL  RXEFLG      clear rcv/xmit-end flag.
0264  0075  F900     B     DLB
0076  006B
0265  0077
```

```
               **********************************************************
                    ===================================             *
               *       Normal interrupt routines.                   *
               *    ===================================             *
               *                                                    *
               *    These routines destroy the contents of the      *
               *    accumulator and the data page pointer, making it  *
               *    necessary to save these before the routines begin *
               *                                                    *
               *    Write the contents of DXMT2 to the transmit buffer*
               *    and read the receive buffer into DRCV1.         *
               *                                                    *
               **********************************************************
0077
0077 4907 NINT1 OUT     DXMT2,PAI     write xmit-low to xmit register.
0078 4104       IN      DRCVI,PAI     read rcv-data-high from rcv reg.
0079 200C       LAC     ANINT2        prepare next interrupt vector.
007A 500A SACL  VECT
007B 4813 OUT   CLRX,PAO                  clear xmit interrupt flag.
007C 7F8D RET
007D 4105 NINT2 IN      DRCV2,PAI     read receive-data-low from rcv
                                      reg.
007E 200B LAC   ANINTI                prepare next interrupt vector.
007F 500A SACL  VECT
0080 4813 OUT   CLRX, PAO             clear xmit interrupt flag.
0081 7EFF LACK  FLAG
0082 5003 SACL  RXEFLG                set xmi%t/rcv end flag.
0083 7F8D RET
               **********************************************************
               *    ===================================             *
               *       Secondary interrupt routines                 *
               *    ===================================             *
               *       These routines destroy the contents of the   *
               *       accumulator and the data page pointer.
               *                                                    *
               *       The following routines write the low byte of *
               *       the primary data word and the high and low byte *
               *       of the secondary data word. They also read the *
               *       A/D information in the receive registers.    *
               **********************************************************
0084 4907 SINT1 OUT     DXMT2,PAl     write xmit-data-low to xmit reg.
0085 4104       IN      DRCVI,PA1     read receive-data-high from rcv
                                      reg
0086 200E       LAC     ASINT2        prepare next interrupt vector.
0087 500A       SACL    VECT
0088 4813       OUT     CLRX,PAO      clear xmit interrupt flag.
0089 7F8D       RET
008A 4908 SINT2 OUT     DXMT3,PAI     write secondary-data-high to
                                      xmit.
008B 4105       IN      DRCV2,PAl     read receive-data-low from rcv.
008C 200F       LAC     ASINT3        prepare next interrupt vector.
008D 500A       SACL    VECT
008E 4813       OUT     CLRX,PAO          clear xmit interrupt flag.
008F 7F8D       RET
0090 4909 SINT3 OUT     DXMT4,PAI     write secondary-data-low to xmit
0091 2010       LAC     ASINT4        prepare next interrupt vector.
```

```
   0319 0092 500A SACL  VECT
   0320 0093 4813       OUT     CLRX,PAO      clear xmit interrupt flag.
   0321 0094 7F8D       RET
   0322 0095 200B SINT4 LAC     ANINT1        prepare next interrupt vector.
   0323 0096 500A       SACL    VECT
   0324 0097 4813       OUT     CLRX,PAO          clear xmit interrupt flag.
   0325 0098 7F89       ZAC
   0326 0099 5007       SACL    DXMT2         clear DXMT2 immediately to
eliminate
   0327 009A 7EFF       LACK    FLAG          unnexpected secondary
communications
   0328 009B 5003       SACL    RXEFLG        set xmit/rcv end flag.
   0329 009C 7FBD       RET
   0330           ***********************************************************
   0331           *  ===================================                  *
   0332           *  Ignore first interrupt.                              *
   0333           *  ===================================                  *
   0334           *     This routine is used to ignore the first data     *
   0335           *     transmission and also to synchronize the AIC      *
   0336           *     with the processor.
   0337           ***********************************************************
   0338 009D 200B IGN   LAC     ANINT1
   0339 009E 500A       SACL    VECT
   0340 009F 4813       OUT     CLRX,0
   0341 00AO 7F8D       RET
   0342           ***********************************************************
   0343           *                                                       *
   0344           *     CONTROL REGISTER INFORMATION                      *
   0345           *                                                       *
   0346           *     SERIAL–PORT CONFIG.  INT. MASK INT. FLAG          *
   0347           *     | 1 0 0 0 1 1 1 0 |  | 0 0 0 1 0 1 0 0 |          *
   0348           *      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0            *
   0349           *                    |                |  |  |  |___INT  *
   0350           *                    |_XF status      |  |  |_____ FSR  *
   0351           *                                     |__________FSX    *
   0352           *                                     |__________FR     *
   0353           *                                                       *
   0354           *                                (write l’s to clear)   *
   0355           ***********************************************************
   0356 00Al 8ElF CLXI  DATA    >8EIF
   0357 00A2 8EOF CLX2  DATA    >8EOF
   0358                 END
   NO ERRORS, NO WARNINGS
```

```
0317 009D 200B IGN   LAC     ANINT1
0318 009E 500A       SACL    VECT
0319 009F 4813       OUT     CLRX,0
0320 00AO 7F8D       RET
0321           ***********************************************************
0322           *                                                       *
0323           *     CONTROL REGISTER INFORMATION                      *
0324           *                                                       *
0325           *     SERIAL–PORT CONFIG.  INT. MASK INT. FLAG          *
0326           *     | 1 0 0 0 1 1 1 0 |  | 0 0 0 1 0 1 0 0 |          *
0327           *      15 14 13 12 11 10 9  8 7 6 5 4 3 2 1 0           *
0328           *                     |               | | | |___INT     *
0329           *                     |_XF status     | | |_____ FSR    *
0330           *                                     |___________FSX   *
0331           *                                     |___________FR    *
0332           *                                                       *
0333           *                                (write l’s to clear)   *
0334           ***********************************************************
0335 00Al 8ElF CLXI  DATA    >8EIF
0336 00A2 8EOF CLX2  DATA    >8EOF
0337                 END
NO ERRORS, NO WARNINGS
```