Microcontroller Based Data Acquisition Using the TLC2543 12-Bit Serial-Out ADC

SLAA012
July 1995

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

List of Illustrations

Introduction

Scope of this Application Report

This application report describes how to construct 12-bit data acquisition systems using the TLC2543 serial-out analog-to-digital converter (ADC) in conjunction with a range of four popular microcontrollers.

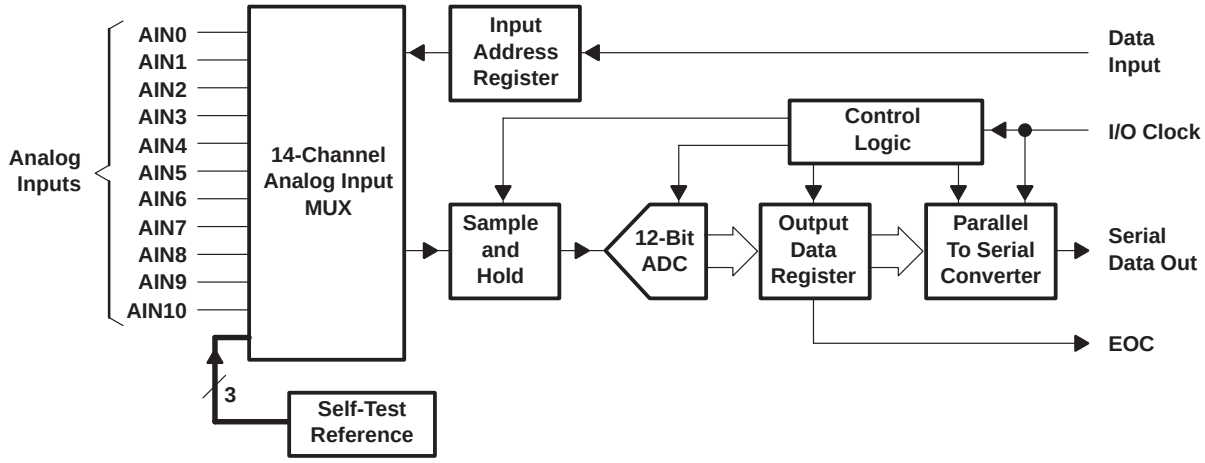
The four microcontrollers used are the TMS370, H8/300, 68HC11 and 80C51.

The TLC2543

The TLC2543 is a 12-bit ADC which uses the switched capacitor successive approximation technique to perform the conversion process and provides a maximum sampling rate of 66k samples per second (KSPS) while using only 1 mA (typical) of supply current.

The block diagram of the TLC2543 is shown in Figure 1. Any one of eleven analog input channels can be selected by programming the four most significant bits (MSBs) of the eight bit channel/mode control byte applied serially to the DATA INPUT terminal of the device. In addition three internal test voltages [V_{ref-}, V_{ref+} and $(V_{ref+} - V_{ref-})/2$] can be applied to the converter for calibration or other purposes by applying the appropriate code to the same four MSBs.

The four least significant bits (LSBs) of the channel/mode control byte are used to select the output data length (8, 12 or 16 bits), the output data order (MSB first or LSB first) and whether unipolar (binary) or bipolar (2's complement around $(V_{ref+} - V_{ref-})/2$) format is required.



- 12-Bit Resolution ADC
- 66-KSPS Sampling Rate
- 11 Analog Input Channels
- Low Supply Current – 1 mA (Typ)
- Power-Down Mode – 4 μA (Typ)
- SPI Compatible Serial Interface

Figure 1. TLC2543 Block Diagram

Interface Timing

Four transfer methods are available for obtaining the full 12 bits of resolution from the TLC2543. Either 12 or 16 clock cycles can be used for each conversion and data transfer.

A chip select ($\overline{CS}$) pulse can be inserted at the start of each conversion or only once at the beginning of each sequence of conversions with $\overline{CS}$ remaining low until the sequence is completed.

Figure 2 shows the timing for the method which uses 16 clock cycles for each conversion and data transfer cycle and which inserts $\overline{CS}$ between each of these transfer cycles. Figure 3 shows the timing for the method which uses 16 clock cycles for each conversion and data transfer cycle but inserts $\overline{CS}$ only once at the start of each sequence of conversions.

This application report describes various microcontroller interfaces, each of which uses 16 clock cycles for each conversion data transfer. $\overline{CS}$ is applied at the start of each conversion and data transfer. This method allows for the general case where one or more conversions may be required. It also simplifies the required software.

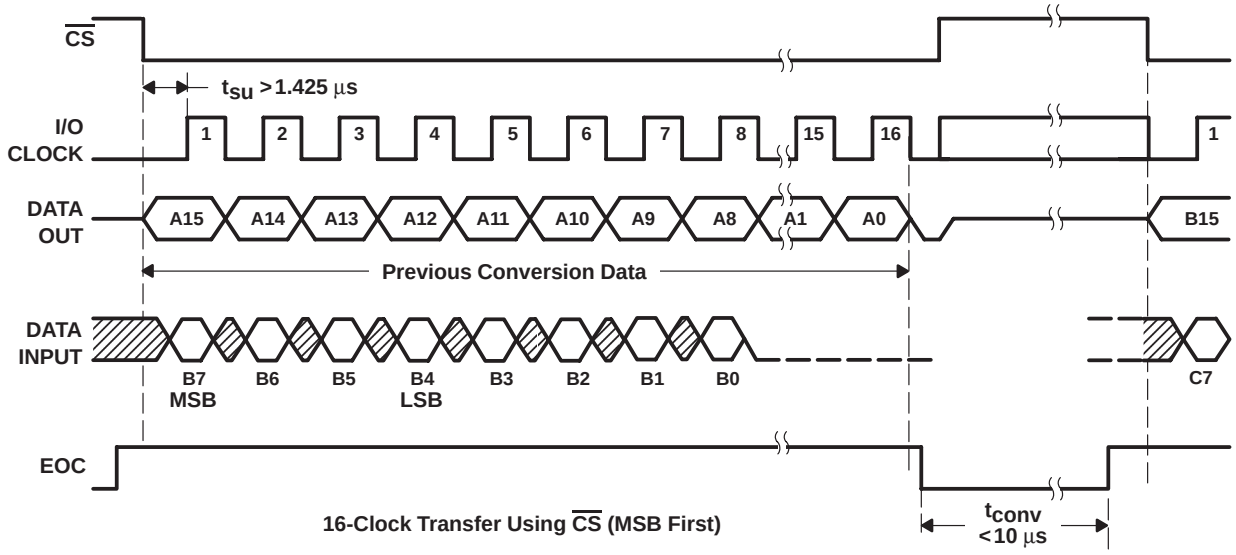


Figure 2. Timing for 16-Clock Transfer Using $\overline{CS}$ With MSB First

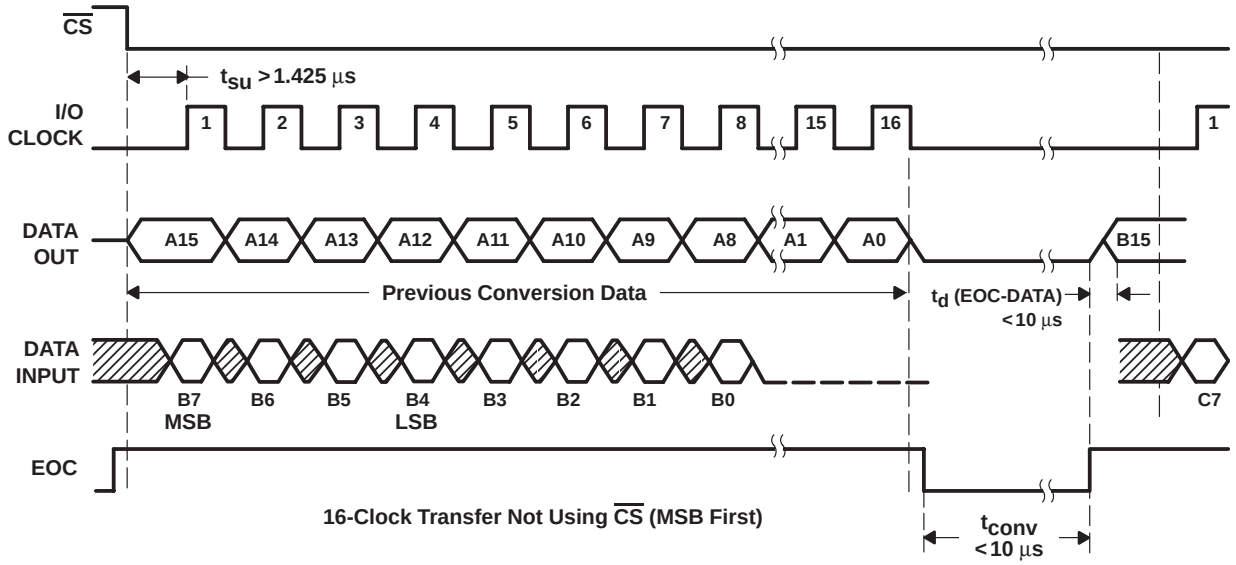


Figure 3. Timing for 16-Clock Transfer Not Using $\overline{CS}$ With MSB First

Minimum Number of Data Transfers per Channel

It should be noted that in any single data transfer cycle between the TLC2543 and the chosen microcontroller the data output from the ADC is the result of the previous conversion. The software listings included in this application report have been written for the general case where the conversion results may be required for any individual channel or sequence of channels. In this case the program included for each microcontroller interface must be run at least twice per channel so that valid data corresponding to the required analogue input channel and ADC mode is delivered.

Software can be written to implement the consecutive channel scanning mode of operation of the TLC2543. In this case the result from the first analog-to-digital conversion should be ignored or overwritten.

Serial Peripheral Interface (SPI)

The fastest and most efficient method of implementing a data transfer between the TLC2543 and a microcontroller is to use the serial peripheral interface (SPI) of the microcontroller, if this is available.

The TMS370 (Texas Instruments), H8/300 (Hitachi) and MC68HC11 (Motorola) all offer SPIs (or the equivalent) in a subset of each of these families of microcontrollers. The H8/300 offers a serial communications interface (SCI) which can be configured to operate in a similar way to that of the standard SPI's offered by the TMS370 and MC68HC11.

The principle features of the SPI are:

- Simultaneous serial data input and output
- Synchronous operation
- Provision of frequency programmable serial clock
- Data transfer complete flag

Figure 4 shows the structure of the SPI. In this case the TMS370C010 is used to illustrate the main elements of the interface.

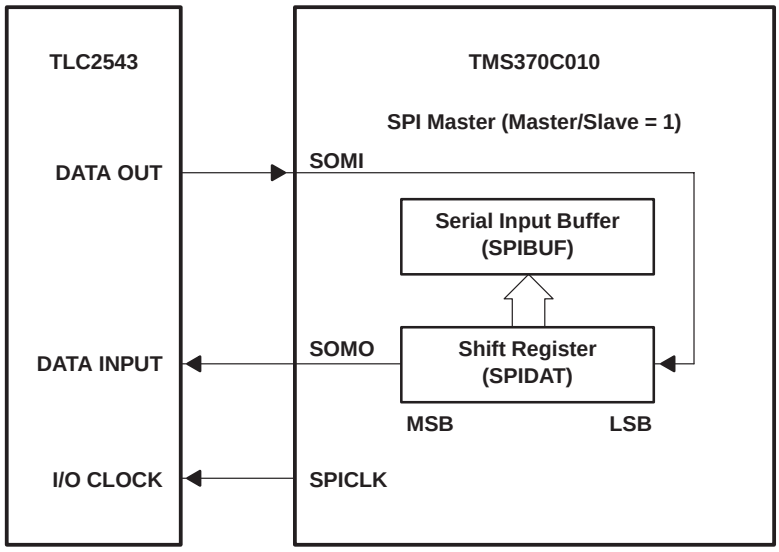


Figure 4. Serial Peripheral Interface – Internal Structure and Data Flow

The microcontroller can be configured by software to perform as the SPI master or slave. When operating as the master, data is input to the SPI shift register (SPIDAT) via the slave out master in (SOMI) terminal. At the same time data is output from the SPIDAT via the slave in master out (SIMO) terminal.

The SPI functions as follows. The SPIDAT should be loaded with the first byte of data to be sent. This automatically initiates the transmission of this byte. During this transmission time data is received at the other end of the SPIDAT shift register. The SPI INT FLAG is regularly checked. As soon as the last bit of the input data byte is received the SPI INT FLAG is set to 1. This then signals that the received byte can be read from the serial input buffer (SPIBUF) and that the SPIDAT is ready to accept the next byte of data to be transmitted.

Additional SPI features which apply to the specific microcontrollers are described in their respective sections which follow.

TLC2543 to SPI Interface Timing

The timing digram for the 16 clock transfer TLC2543 to SPI interface is shown in Figure 5. The channel select/mode data is read into the TLC2543 on the positive going edges of the I/O clock and analog-to-digital conversion results are read into the microcontroller on the negative going edges of the I/O clock.

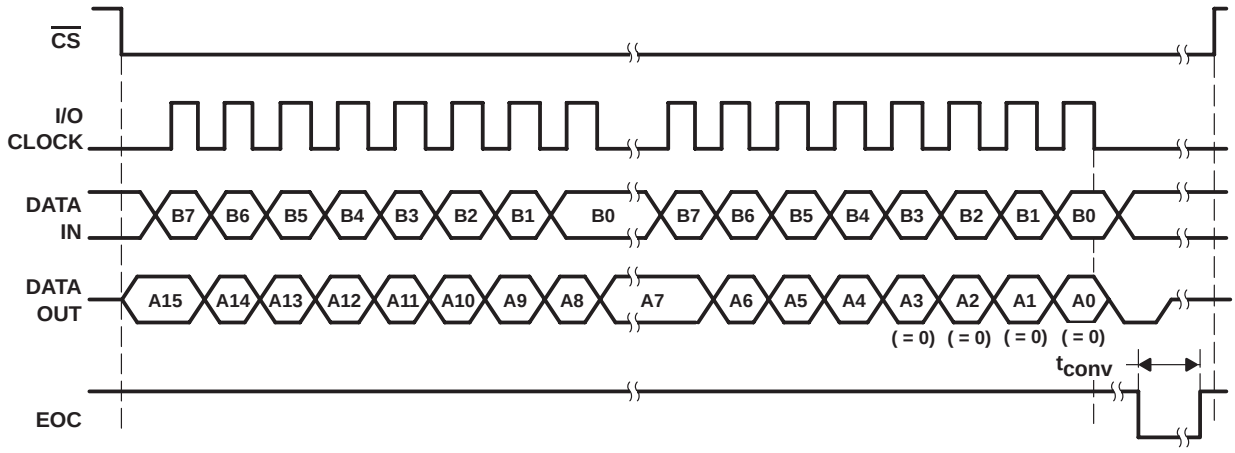


Figure 5. TLC2543 to SPI Interface Timing

Software Flowcharts

Figures 6, 7, and 8 show the flow charts for the main program and subroutines used in the TLC2543 to TMS370C010 interface software shown in this application report. The same program structure also applies to the other three interfaces included in this report.

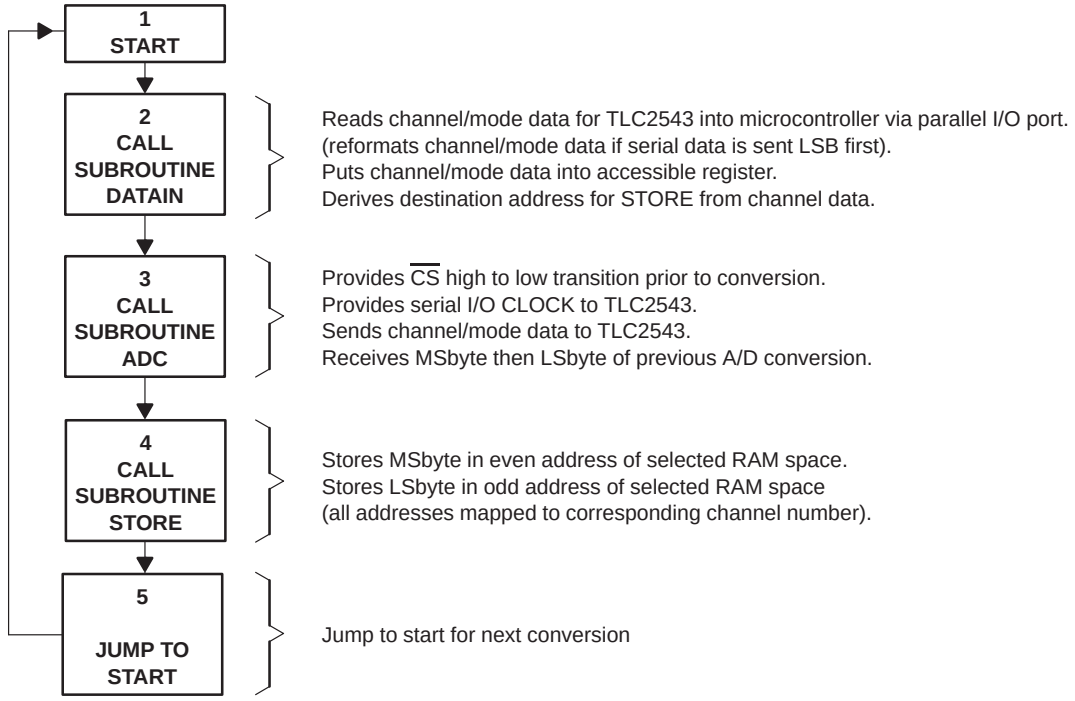


Figure 6. Flowchart for Main Program of TLC2543 to TMS370C010

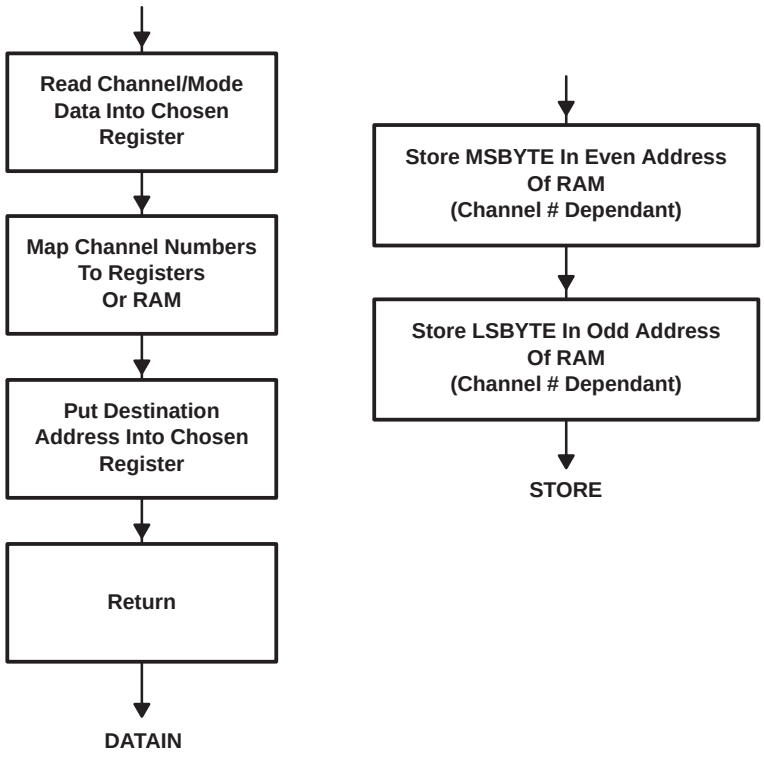


Figure 7. Flowcharts of Subroutine DATAIN and STORE for TLC2543 to TMS370C010 Interface Software

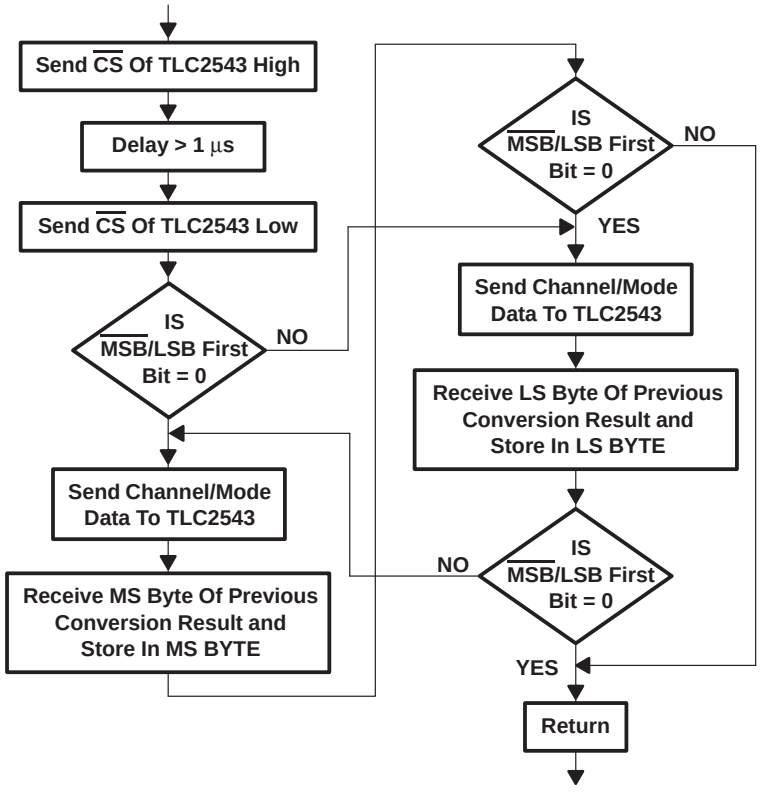


Figure 8. Flowcharts of Subroutine ADC for TLC2543 to TMS370C010 Interface Software

TLC2543 TO TMS370 Microcontroller Interface

Microcontroller Features

Within the family of TMS370 microcontrollers there are several versions which contain a serial peripheral interface (SPI) facility. One of these versions should be chosen to implement the interface method described below. One such version is the TMS370C010 which is used to illustrate the method.

Interface Circuit

Figure 9 shows the circuit interconnections for the TLC2543 to TMS370C010 microcontroller interface. Note that no extra logic is required to implement this interface.

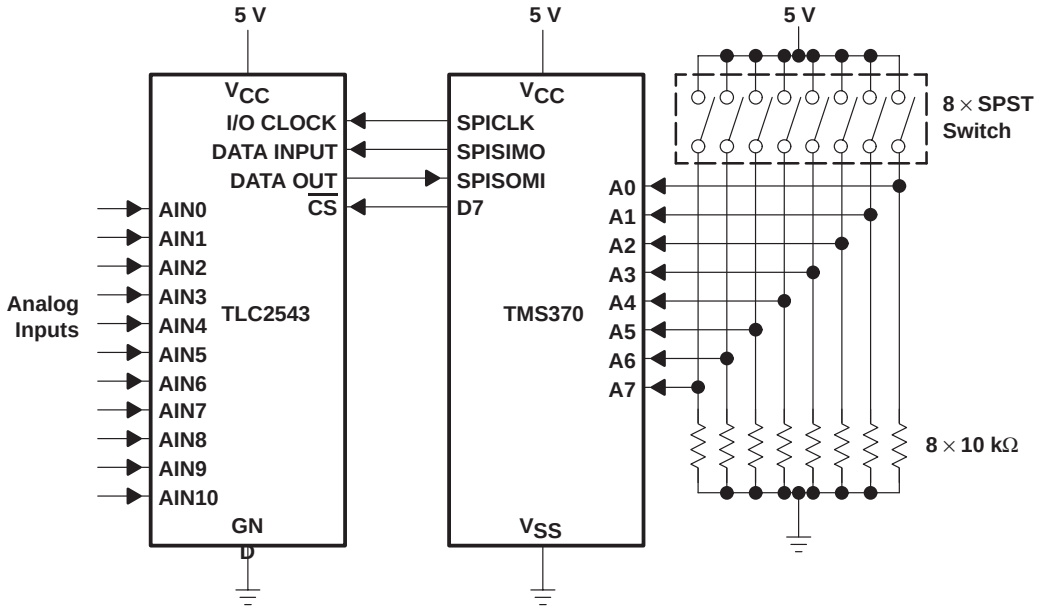


Figure 9. TLC2543 to TMS370C010 Interface Circuit

Depending upon the layout of the particular printed circuit board used it may be necessary to insert a small value capacitor of between 50 and 100 pF between the I/O CLOCK input of the TLC2543 and ground. This has the effect of ensuring that data applied to the DATA INPUT terminal of the TLC2543 is valid before the positive going transition of the I/O CLOCK.

The positive reference, REF+, to the TLC2543 is provided directly from the V_{CC+} supply.

The four digital interface terminals, I/O CLOCK, DATA INPUT, DATA OUT, and $\overline{CS}$, of the TLC2543 connect directly to the SPICLK, SPISIMO, SPISOMI and D7 terminals respectively of the TMS370C010.

The operation mode and channel number of the TLC2543 is determined by the serial data which is sent to its DATA INPUT terminal.

Software

List 1 contains the software listing for the program which controls the interface illustrated in Figure 5. The software consists of the main program and three subroutines called DATA IN, ADC and STORE. DATAIN reads in the channel select and mode control data into a holding register and maps the channel select number to a corresponding pair of registers between R64 and R91. The mapping vector is held in register R10. ADC provides the chip select pulse, controls the SPI operation, and puts the MSByte and LSByte of each conversion result into registers R20 and R21 respectively. STORE puts the MSByte into the even number register and the LSByte into the odd number register mapped by the contents of register R10.

The user can put channel select and ADC mode control data into the holding register within the microcontroller, via the 8-bit wide port A bidirectional I/O port, using a bank of eight toggle switches as shown in Figure 9. Alternatively, the mode and channel data can be sent to the microcontroller holding register via the asynchronous serial communications interface (SCI). This option is available only on those versions of the TMS370, such as the TMS370C020, which include both SPI and SCI interfaces. Additional software to control the SCI must be appended to the software shown in List 1 to provide this method of control.

List 1

```
LINE   LOC   OBJ              SOURCE

   1                         ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
   2                         ;*                                                 *
   3                         ;* TLC2543 to TMS 370Cx10 Interface Program        *
   4                         ;*                                                 *
   5                         ;* This program reads channel select and mode     *
   6                         ;* control data (provided by toggle switches)     *
   7                         ;* into the microcontroller, using subroutine     *
   8                         ;* DATAIN.                                        *
   9                         ;* It then provides the control signals to        *
  10                         ;* the TLC2543 to perform a 12 bit analog to      *
  11                         ;* conversion, using subroutine ADC. It           *
  12                         ;* finally stores the MSByte and LSByte of        *
  13                         ;* each conversion in consecutive even and        *
  14                         ;* odd number registers respectively starting     *
  15                         ;* at R64 corresponding to channel 0, using       *
  16                         ;* subroutine STORE.                              *
  17                         ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
  18        0030             SPICCR  .EQU P030       ;*  *  *  *  *  *  *  *  *
  19        0031             SPICTL  .EQU P031       ;*                       *
  20        0037             SPIBUF  .EQU P037       ;*                       *
  21        0039             SPIDAT  .EQU P039       ;*                       *
  22        003d             SPIPC1  .EQU P03D       ;*                       *
  23        003e             SPIPC2  .EQU P03E       ;* Name Peripheral        *
  24        003f             SPIPRI  .EQU P03F       ;* Registers              *
  25        0021             APORT2  .EQU P021       ;*                       *
  26        0022             ADATA   .EQU P022       ;*                       *
  27        0023             ADIR    .EQU P023       ;*                       *
  28        002c             DPORT1  .EQU P02C       ;*                       *
  29        002d             DPORT2  .EQU P02D       ;*                       *
  30        002e             DDATA   .EQU P02E       ;*                       *
  31        002f             DDIR    .EQU P02F       ;*  *  *  *  *  *  *  *  *
  32        7ffe             RESET   .EQU 7FFEH      ;Reset vector named
  33        2e               CSBIT   .DBIT 7,DDATA   ;TLC2543 Chip Select bit
  34                                                 ;named CSBIT
  35                         ;
  36                         ;
  37
  38 4000                                 .TEXT 4000H     ;Start program at 4000H
  39                         ;Main Program
  40 4000   5260             START   MOV #60H,B      ;
  41 4002   fd                       LDSP            ;Set SP to address 60H
  42 4003 * 88400000                 MOVW #4000H,A
  43 4007   8b7ffe                   MOV A,RESET     ;Set reset vector
  44 400a   b5                       CLR A
  45 400b   2121                     MOV A,APORT2
  46 400d   2123                     MOV A,ADIR
  47 400f   f7802f                   MOV #080H,DDIR
  48 4012   2280                     MOV #80H,A
  49 4014   2130                     MOV A,SPICCR
  50 4016   2207                     MOV #07,A       ;Configure SPI for 8–bit
  51 4018   2130                     MOV A,SPICCR    ;character length.
  52 401a   2203                     MOV #03,A       ;Configure SPICLK
  53 401c   213d                     MOV A,SPIPC1    ;function and direction.
  54 401e   2222                     MOV #22H,A      ;Configure SPISOMI and
  55 4020   213e                     MOV A,SPIPC2    ;SPISIMO pin functions.
  56        31               SPIF    .DBIT 6,SPICTL  ;SPI INT FLAG named SPINTF
  57        0b               MSLSB   .DBIT 1,R11     ;Bit 1 of R11 named MSLSB
  58 4022  ’8e402d                   CALL DATAIN
  59 4025  ’8e403e                   CALL ADC
  60 4028  ’8e409d                   CALL STORE
```

List 1 (Continued)

```
LINE   LOC   OBJ              SOURCE

  61 402b  '00d3                    JMP START
  62
  63                         ;
  64                         ;Subroutine :- DATAIN
  65                         ;
  66 402d   9122             DATAIN  MOV ADATA,B     ;Read ADC mode/channel
  67 402f   d10b                     MOV B,R11       ;Put ADC mode/channel
  68                                                 ;in R11
  69 4031   53f0                     AND #0F0H,B     ;Retain channel number
  70 4033   cc                       RR B            ;*  *  *  *  *  *  *  *  *
  71 4034   cc                       RR B            ;* Map channel numbers   *
  72 4035   cc                       RR B            ;* to registers R64-R91  *
  73 4036   cc                       RR B            ;* R10 contains storage  *
  74 4037   5c02                     MPY #002,B      ;* address               *
  75 4039   5840                     ADD #40H,B      ;* Even numbers - MS Byte*
  76 403b   d10a                     MOV B,R10       ;* Odd numbers - LS Byte *
  77                                                 ;*  *  *  *  *  *  *  *  *
  78 403d   f9                       RTS
  79                         ;
  80                         ;Subroutine :- ADC
  81                         ;
  82 403e   2203             ADC     MOV #003H,A
  83 4040   a4802e                   SBIT1 CSBIT     ;Set ADC Chip Select high.
  84 4043   b2               LOOP1   DEC A           ;Chip Select stays high
  85 4044  '06fd                     JNE LOOP1       ;while A is not 0.
  86 4046   c5                       CLR B
  87 4047   512e                     MOV B,DDATA     ;CS goes low
  88 4049   2207                     MOV #7,A
  89 404b   2131                     MOV A,SPICTL    ;Enable SPI transmission
  90 404d  '76020b19                 JBIT1 MSLSB,LS1ST
  91 4051   120b                     MOV R11,A
  92 4053   2139                     MOV A,SPIDAT
  93                         ;       MOV R11,SPIDAT  ;Send mode/channel data
  94                                                 ;to TLC2543
  95 4055  'a74031fc         FLAG1   JBIT0 SPIF,FLAG1;If SPIF=0, repeat check.
  96 4059   a21437                   MOV SPIBUF,R20  ;Put received MS Byte
  97                                                 ;in R20
  98 405c   71390b                   MOV R11,SPIDAT  ;Send mode/channel data
  99                                                 ;to TLC2543
 100 405f  'a74031fc         FLAG2   JBIT0 SPIF,FLAG2;If SPIF=0, repeat check.
 101 4063   a21537                   MOV SPIBUF,R21  ;Put received LS Byte
 102                                                 ; in R21
 103 4066  '77020b32                 JBIT0 MSLSB,RETURN ;If MSLSB=0, go
 104                                                    ;to RETURN
 105 406a   120b             LS1ST   MOV R11,A
 106 406c   2139                     MOV A,SPIDAT
 107 406e  'a74031fc         FLAG3   JBIT0 SPIF,FLAG3;If SPIF=0, repeat check.
 108 4072   a21537                   MOV SPIBUF,R21  ;Put received LS Byte
 109                                                 ;in R21
 110 4075   120b                     MOV R11,A
 111 4077   2139                     MOV A,SPIDAT
 112 4079  'a74031fc         FLAG4   JBIT0 SPIF,FLAG4;If SPIF=0, repeat check.
 113 407d   a21437                   MOV SPIBUF,R20  ;Put received MS Byte
 114                                                 ;in R20
 115 4080   2208                     MOV #08,A       ;*  *  *  *  *  *  *  *
 116 4082   d516                     CLR R22         ;*                     *
 117 4084   dd14             LOOP2   RRC R20         ;*  Reformat MS Byte  *
 118 4086   df16                     RLC R22         ;*                     *
 119 4088   b2                       DEC A           ;*  Put result in R20 *
 120 4089  '06f9                     JNZ LOOP2       ;*                     *
 121 408b   421614                   MOV R22, R20    ;*  *  *  *  *  *  *  *
 122 408e   2208                     MOV #08,A       ;*  *  *  *  *  *  *  *
 123 4090   d517                     CLR R23         ;*                     *
 124 4092   dd15             LOOP3   RRC R21         ;*  Reformats LS Byte *
 125 4094   df17                     RLC R23         ;*                     *
 126 4096   b2                       DEC A           ;*  Put result in R21 *
 127 4097  '06f9                     JNZ LOOP3       ;*                     *
 128 4099   421715                   MOV R23,R21     ;*  *  *  *  *  *  *  *
```

List 1 (Continued)

```
LINE   LOC   OBJ             SOURCE
 129 409c   f9              RETURN  RTS
 130                        ;
 131                        ;Subroutine :– STORE
 132                        ;
 133 409d   1214            STORE   MOV R20,A       ;Put MS Byte into even
 134 409f   9b0a                    MOV A,@R10      ;address contained in R10
 135 40a1   d30a                    INC R10         ;(R10)+1
 136 40a3   1215                    MOV R21,A       ;Put LS Byte into odd
 137 40a5   9b0a                    MOV A,@R10      ;address contained in R10
 138 40a7   f9                      RTS
 139                                .END
```

Opto-Isolated 12-Bit Data Acquisition System

The serial nature of the data flow between the TLC2543 analog-to-digital converter and the accompanying microcontroller makes this ADC an ideal choice for isolated 12-bit data acquisition. Figure 10 shows an opto-isolated system which uses four optocouplers to provide a 3-kV isolation barrier.

Note that the optocoupler which routes conversion result data from the TLC2543 to the microcontroller is a single device and does not share the same piece of silicon with any of the other optocouplers used. This ensures that the full 3 kV of isolation is maintained between the ADC and microcontroller.

The choice of VP0610 P-channel enhancement MOSFETs avoids the use of an extra inverter stage for each optocoupler driver. In addition, the relatively low input capacitance of the VP0610 (typically 15 pF) allows data rates up to 100 kHz to be achieved without the need for external buffers to be added at the outputs of the TLC2543 and TMS370.

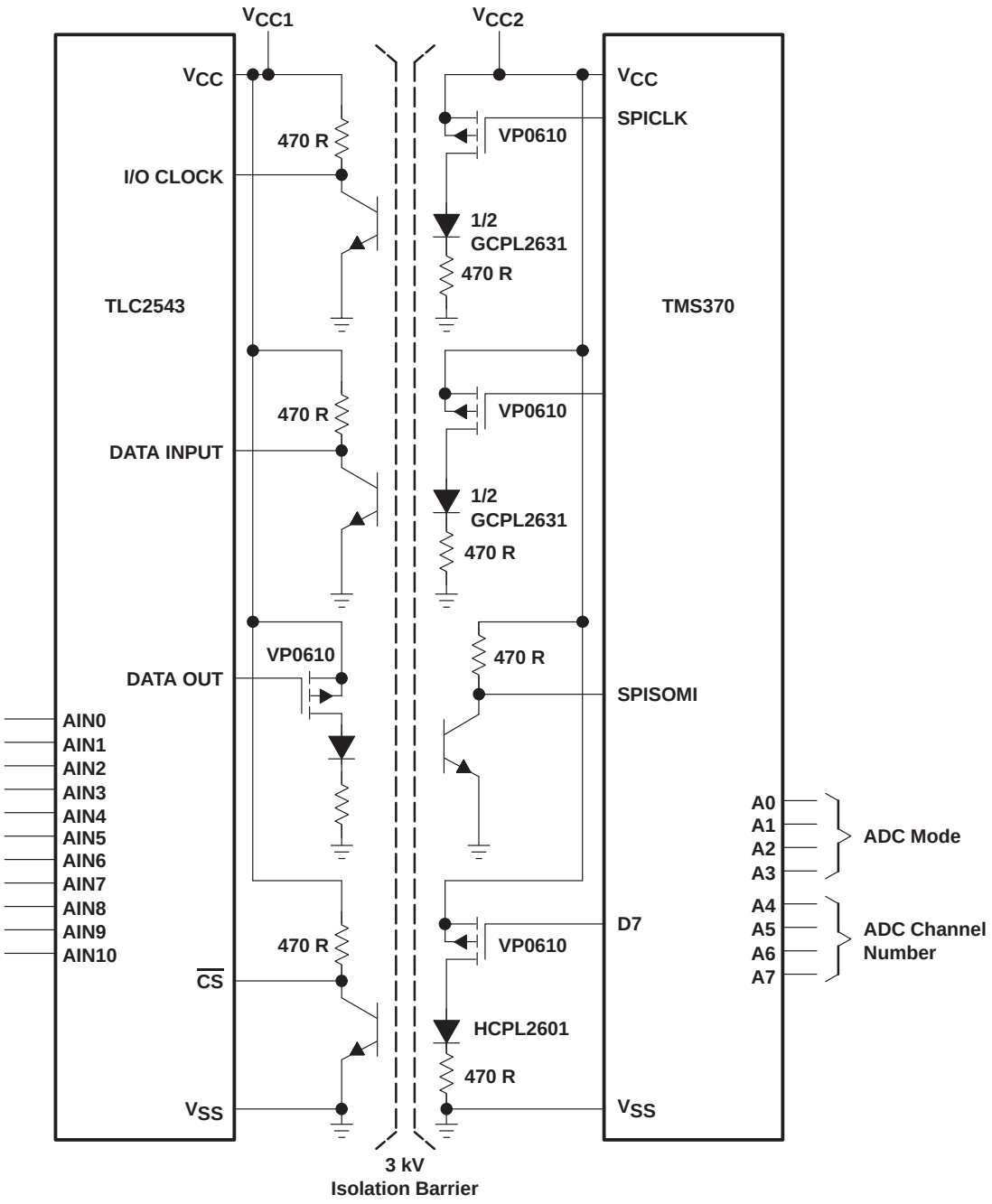


Figure 10. Opto-Isolated 12-Bit Data Acquisition System

TLC2543 to H8/325 Microcontroller Interface

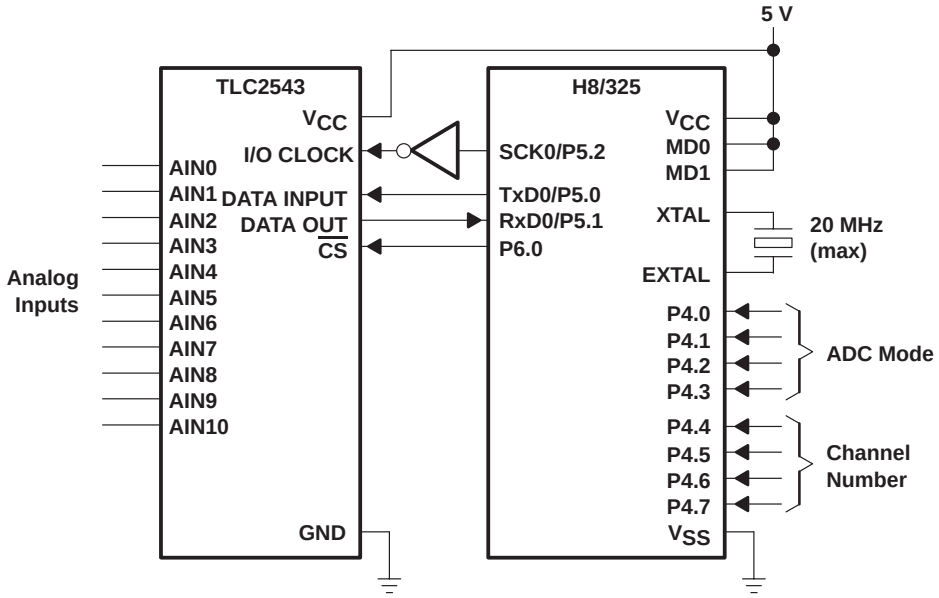
Microcontroller Features

The individual members of the H8 family of microcontrollers can be differentiated by various criteria, for example the inclusion or otherwise of an on-board 8-bit resolution analog-to-digital converter (ADC). Those members which include an ADC generally cost between 10 and 20 percent more than their counterparts which do not.

System requirements such as ADC resolution, remote location of ADC, flexibility, and total cost all influence the final choice of microcontroller architecture. The H8/325, used for this application report, does not include an on-board ADC but provides 1K of RAM, 32K of ROM, and two serial I/O ports. It is therefore well suited to interfacing with the TLC2543 serial output ADC.

Interface Circuit

Figure 11 illustrates a typical 12-bit data acquisition system which uses the H8/325 microcontroller to coordinate the operation of the TLC2543 ADC via one of its serial (SCI) ports. The circuit uses the H8's 8-bit parallel I/O port 4 to route ADC channel and mode information into the microcontroller. This information could be provided by a host system data bus or, as in Figure 6, by a bank of eight manually operated toggle switches situated on the same printed circuit board as the microcontroller.



NOTE: Single Chip Mode (MD0 = MD1 = 1)

Figure 11. TLC2543 to H8/300 Microcontroller Interface Circuit

Software

List 2 shows the program which was written to coordinate the interface. It uses three subroutines to implement the overall interface to the TLC2543. The first of these is called DATAIN which reads ADC channel and mode information into the microcontroller. It also maps converter channel numbers to corresponding addresses in RAM where conversion results can be stored. In this case the addresses from 0040H to 0067H were chosen to store the results. The most significant byte of each result is placed in an even address and the least significant byte of each result is placed in the corresponding adjacent odd address.

The conversion result of each channel is stored in left justified format and therefore occupies the upper 12 bits of the 16-bit words which occupy even addresses from 0040H up to 0066H.

The second subroutine to be used is ADC. This begins by producing a chip-select high pulse. The trailing negative edge of this pulse is rapidly followed by the transmission of channel and mode information to the converter.

List 2

```
 LINE   LOC   OBJ   SOURCE

  1                ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
  2                ;*                                                                     *
  3                ;*     TLC2543 to H8 Microcontroller Interface Program.                *
  4                ;*                                                                     *
  5                ;*  This program contains subroutines DATAIN, ADC,                     *
  6                ;*  FORMAT and STORE.                                                  *
  7                ;*  DATAIN reads channel number and mode data into the                 *
  8                ;*  microcontroller via Port4. "ADC" controls A to D                   *
  9                ;*  conversion. "FORMAT" changes conversion results from               *
 10                ;*  LSB first format to MSB first format. "Store" places               *
 11                ;*  results in memory addresses 40 to 5B (MS Bytes in even             *
 12                ;*  addresses, LS Bytes in odd addresses) according to                 *
 13                ;*  channel number.                                                    *
 14                ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 15                ;
 16                ; * Define special function register names *
 17                ;
 18  FFDD          RDR     EQU H'FFDD              ;Receive Data Register – RDR
 19  FFDB          TDR     EQU H'FFDB              ;Transmit Data Register – TDR
 20  FFD8          SMR     EQU H'FFD8              ;Serial Mode Register – SMR
 21  FFDA          SCR     EQU H'FFDA              ;Serial Control Register – SCR
 22  FFDC          SSR     EQU H'FFDC              ;Serial Status Register – SSR
 23  FFD9          BRR     EQU H'FFD9              ;Bit Rate Register – BRR
 24  FFB8          P5DDR   EQU H'FFB8              ;Port5 Data Direction Register
 25  FFBA          P5DR    EQU H'FFBA              ;Port5 Data Register – P5DR
 26  FFB5          P4DDR   EQU H'FFB5              ;Port4 Data Direction Register
 27  FFB7          P4DR    EQU H'FFB7              ;Port4 Data register – P4DR
 28  FFB9          P6DDR   EQU H'FFB9              ;Port6 Data Direction register
 29  FFBB          P6DR    EQU H'FFBB              ;Port6 Data Register
 30
 31
 32  1000                  ORG H'1000              ;Sets start of program
 33                ;
 34                ; * Main Program *
 35                ;
 36  1000 79001000         MOV.W #H'1000, R0
 37                ;       MOV.W R0, @H'0000       ;Sets reset vector to START
 38  1004 7907FD00         MOV.W #H'FD00, SP       ;Sets contents of Stack Pointer
 39  1008 F984     START   MOV.B #H'84, R1L        ;* * * * * * * * * * * * * * * *
 40  100A 39D8             MOV.B R1L, @SMR:8       ;* Sets up serial port         *
 41  100C F931             MOV.B #H'31,R1L         ;* registers for simultaneous  *
 42  100E 39DA             MOV.B R1L, @SCR:8       ;* transmit and receive        *
 43  1010 F901             MOV.B #H'01,R1L         ;*                              *
 44  1012 39D9             MOV.B R1L, @BRR:8       ;* * * * * * * * * * * * * * * *
 45  1014 F901             MOV.B #H'01, R1L        ;Sets R1(Low Byte) to 01H
 46  1016 6A89FFB9         MOV.B R1L, @P6DDR       ;Set Bit0 of Port6 as Output
 47  101A 5E001082         JSR @DATAIN:16          ;Read in ADC channel/mode data
 48  101E 5E00102C         JSR @ADC:16             ;Do A/D conversion
 49  1022 5E0010B4         JSR @FORMAT:16          ;Reformat A/D conversion result
 50  1026 5E0010AC         JSR @STORE:16           ;Store A/D conversion result
 51  102A 40DC             BRA START               ;Repeat above routine.
 52                ;
 53                ; * Subroutine ADC which controls the conversion process *
 54                ;
 55  102C FA05     ADC     MOV.B #H'05, R2L        ;Put 05 in R2L
 56  102E 7FBB7000         BSET #0, @P6DR:8        ;TLC2543 chip select goes high
 57  1032 1A0A     CSHIGH  DEC R2L                 ;(R2L) – 1
 58  1034 46FC             BNE CSHIGH              ;If not zero, CS stays high
 59  1036 7FBB7200         BCLR #0, @P6DR:8        ;TLC2543 chip select goes low
 60  103A 6A0CFFB7         MOV.B @P4DR, R4L        ;Puts channel/mode data in R4L
 61  103E 731C             BTST #1, R4L            ;Is LSBF of channel/mode data
 62  1040 461E             BNE LSBYTE              ;If not, do LSBYTE first
 63  1042 7EDC7370 MSBYTE  BTST #7, @SSR:8         ;Is TDR empty ?
 64  1046 47FA             BEQ MSBYTE              ;If not empty, repeat check.
 65  1048 34DB             MOV.B R4H, @TDR:8       ;Put channel/mode data in TDR
 66  104A 7FDC7270         BCLR #7, @SSR:8         ;Reset TDRE bit of SSR to 0
 67  104E 7EDC7360 TESTB61 BTST #6, @SSR:8         ;Is receive shift reg. empty ?
```

List 2 (Continued)

```
  LINE   LOC   OBJ              SOURCE

 68  1052 47FA              BEQ TESTB61             ;If not empty, repeat check
 69  1054 23DD              MOV.B @RDR:8, R3H       ;Put MS Byte of conversion
 70                                                 ;result in R3H
 71  1056 7FDC7260          BCLR #6, @SSR:8         ;Reset RDRF bit of SSR to 0
 72  105A 68D3              MOV.B R3H, @R5          ;Put MS Byte in even address
 73                                                 ;mapped by channel number.
 74  105C 731C              BTST #1, R4L            ;Is LSBF of channel/mode data 0
 75  105E 4620              BNE RETURN              ;If not, return from subroutine
 76  1060 7EDC7370  LSBYTE  BTST #7, @SSR:8         ;Is TDR empty ?
 77  1064 47FA              BEQ LSBYTE              ;If not empty, repeat check
 78  1066 34DB              MOV.B R4H, @TDR:8       ;Put channel/mode data in TDR
 79  1068 7FDC7270          BCLR #7, @SSR:8         ;Reset TDRE bit of SSR to 0
 80  106C 7EDC7360  TESTB62 BTST #6, @SSR:8         ;Is receive shift reg. empty ?
 81  1070 47FA              BEQ TESTB62             ;If not empty, repeat check
 82  1072 0A0D              INC R5L                 ;(R5L) + 1
 83  1074 2BDD              MOV.B @RDR:8, R3L       ;Put LS Byte of conversion
 84                                                 ;result in R3L
 85  1076 7FDC7260          BCLR #6, @SSR:8         ;Reset RDRF bit of SSR to 0
 86  107A 68DB              MOV.B R3L, @R5          ;Put LS Byte in odd address
 87                                                 ;mapped by channel number.
 88  107C 731C              BTST #1, R4L            ;Is LSBF of channel/mode data 0
 89  107E 46C2              BNE MSBYTE              ;If not, go to MSBYTE
 90  1080 5470      RETURN  RTS                     ;Return from subroutine
 91                 ;
 92                 ; * Subroutine "DATAIN" which reads in ADC channel/mode data *
 93                 ;
 94  1082 79040000  DATAIN  MOV.W #H'0000, R4
 95  1086 79050000          MOV.W #H'0000, R5
 96  108A 6A0CFFB7          MOV.B @P4DR, R4L        ;Puts channel/mode data in R4L
 97  108E 0CCD              MOV.B R4L, R5L          ;Puts (R4L) in R5L
 98  1090 110D              SHLR R5L                ;*  *  *  *  *  *  *  *  *  *
 99  1092 110D              SHLR R5L                ;* Retain only channel      *
100  1094 110D              SHLR R5L                ;* number in R5L             *
101  1096 110D              SHLR R5L                ;*  *  *  *  *  *  *  *  *  *
102  1098 79060002          MOV.W #0002, R6         ;*  *  *  *  *  *  *  *  *  *
103  109C 50E5              MULXU R6L, R5           ;* Maps channel numbers to  *
104  109E 8D40              ADD.B #H'40, R5L        ;* even addresses 40H to 5AH*
105                                                 ;* Put address in R5L       *
106                                                 ;*  *  *  *  *  *  *  *  *  *
107  10A0 F008              MOV.B #H'08, R0H        ;Put 08 in R0H
108  10A2 110C      NEXTBIT SHLR R4L                ;*  *  *  *  *  *  *  *  *  * *
109  10A4 1204              ROTXL R4H               ;* Reformats channel/mode data*
110  10A6 1A00              DEC R0H                 ;* from MSB first to LSB first*
111  10A8 46F8              BNE NEXTBIT             ;*  *  *  *  *  *  *  *  *  * *
112  10AA 5470              RTS
113                ;
114                ; * Subroutine "STORE" stores A/D conversion results in RAM *
115                ;
116  10AC 68D3      STORE   MOV.B R3H, @R5          ;Store MS Byte in even address
117                                                 ;corresponding to channel
118                                                 ;number
119  10AE 0A0D              INC R5L                 ;(R5) + 1
120  10B0 68DB              MOV.B R3L, @R5          ;Store LS Byte in odd address
121                                                 ;corresponding to channel
122                                                 ;number
123  10B2 5470              RTS                     ;Return from subroutine
124                ;
125                ; * Subroutine "FORMAT" changes received data format   *
126                ; * ( LSB first ) into MSB first format                *
127                ;
128  10B4 F008      FORMAT  MOV.B #H'08, R0H        ;Put 08 in R0H
129  10B6 1103      LOOP1   SHLR R3H                ;*  *  *  *  *  *  *  *  *
130  10B8 1207              ROTXL R7H               ;*                        *
131  10BA 1A00              DEC R0H                 ;*   Reformats MSBYTE     *
132  10BC 46F8              BNE LOOP1               ;*                        *
133  10BE 0C73              MOV.B R7H, R3H          ;*  *  *  *  *  *  *  *  *
134  10C0 F008              MOV.B #H'08, R0H        ;Put 08 in R0H
135  10C2 110B      LOOP2   SHLR R3L                ;*  *  *  *  *  *  *  *  *
```

List 2 (Continued)

```
   LINE   LOC   OBJ             SOURCE

136  10C4 120F            ROTXL R7L               ;*                      *
137  10C6 1A00            DEC R0H                 ;*   Reformats LSBYTE   *
138  10C8 46F8            BNE LOOP2               ;*                      *
139  10CA 0CFB            MOV.B R7L, R3L          ;*  *  *  *  *  *  *  *  *
140  10CC 5470            RTS                     ;Return from subroutine
141  10CE                 END
```

TLC2543 to MC68HC11 Microcontroller Interface

Microcontroller Features

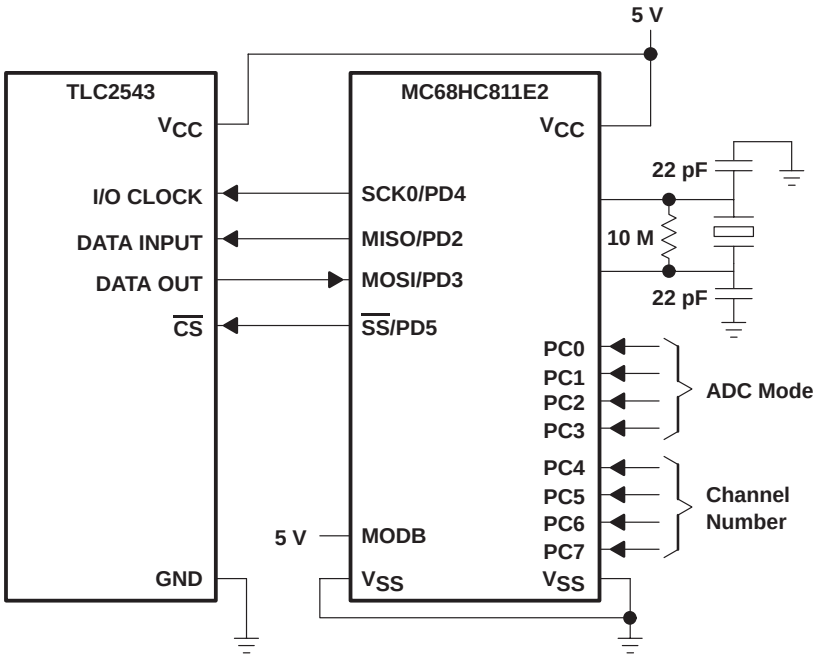
All members of the MC68HC11 family of microcontrollers contain an SPI. As is the case for the TMS370, the user is able to set the idle level of the serial clock of the 68HC11. This eliminates the need for an external inverter to be used to invert the microcontroller's serial clock output prior to its arrival at the TLC2543's serial clock input.

The 68HC11D0, 68HC11D3 and 68HC711D3 versions do not contain an on-board ADC. One of these three devices may prove to be the most cost effective choice when used with the TLC2543. All other versions contain either an 8- or 10-bit resolution ADC.

Interface Circuit

Figure 12 shows the circuit diagram of the interface between the 68HC11 and the TLC2543. The microcontroller device type used to illustrate this interface is the 48-pin dual-in-line version of the MC68HC811E2.

The master in slave out (MISO), master out slave in (MOSI) and serial clock (SCK) terminals of the SPI are available as the alternative, user selectable, functions of port D pins PD2, PD3, and PD4 respectively. When the SPI is configured to operate as a master, the $\overline{SS}$/PD5 terminal can be used as an output to drive the chip select ($\overline{CS}$) terminal of the TLC2543. This leaves all other bidirectional I/O ports of the microcontroller uncommitted and available for other uses. Note that no extra glue logic is required to implement the interface.



NOTES: A. Configured for single chip mode of operation
B. Maximum SPI data rate = crystal frequency/8

Figure 12. TLC2543 to MC68HC811E2 Microcontroller Interface

Software

The listing of the program which was written to coordinate and control the interface between the TLC2543 and the 68HC811E2 is shown in List 3. The software consists of the main program and two subroutines named TLC2543 and STORE. TLC2543 begins by providing the ADC's chip select pulse. It then reads in channel/mode data via the port C parallel I/O port and subsequently sends this data to the TLC2543 via the MOSI terminal of the SPI. At the same time, the first byte of the result from the previous analog-to-digital conversion is received at the MISO terminal of the SPI.

List 3

```
   LINE   LOC   OBJ   SOURCE

 1 A                 *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 2 A                 *                                                                     *
 3 A                 * TLC2543 12-bit Serial Out ADC to MC68HC11 Microcontroller *
 4 A                 *     Interface Program           *
 5 A                 *                                                                     *
 6 A                 * This program contains subroutines "TLC2543" and "STORE".  *
 7 A                 * "TLC2543" reads in ADC mode control and channel select    *
 8 A                 * data via Port C. It then sends this data to the TLC2543   *
 9 A                 * and at the same time receives the result of the previous  *
10 A                 * conversion.                                                        *
11 A                 * "STORE" puts the results into addresses $30 to $4B with   *
12 A                 * MSBytes in even addresses and LSBytes in odd addresses.   *
13 A                 * Channel 0 result in addresses $30 and $31      *
14 A                 * Channel 1 result in addresses $32 and $33 etc.     *
15 A                 *                                                                     *
16 A                 *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
17 A
18 A       1000      BASEADD   EQU       $1000                Register block start
19 A       0000      PORTA     EQU       $00                  Port A Data Register
20 A       0003      PORTC     EQU       $03                  Port C Data Register
21 A       0007      DDRC      EQU       $07                  Port C Data Dir Reg
22 A       0008      PORTD     EQU       $08                  Port D Data Register
23 A       0009      DDRD      EQU       $09                  Port D Data Dir Reg
24 A       0028      SPCR      EQU       $28                  SPI Control Register
25 A       0029      SPSR      EQU       $29                  SPI Status Register
26 A       002A      SPDR      EQU       $2A                  SPI Data Register
27 A       00F0      MSBYTE    EQU       $F0                  MSBYTE address
28 A       00F1      LSBYTE    EQU       $F1                  LSBYTE address
29 A       00F2      TEMP      EQU       $F2                  TEMP address
30 A 0000
31 A       F800                ORG       $F800                Start @ $F800
32 A                 * LDS #$0070 Set Stack Pointer
33 A F800 863E       START     LDAA      #$3E                 Load 3EH in AA
34 A F802 A709                 STAA      DDRD,X               Store 3EH in DDRD
35 A F804 8650                 LDAA      #$50                 Load 50H into AA
36 A F806 A728                 STAA      SPCR,X               Set SPI as master
37 A F808 8600                 LDAA      #$00                 Load 00 into AA
38 A F80A A707                 STAA      DDRC,X               Set PORTC - all I/Ps
39 A F80C 1C0820               BSET      PORTD,X#$20          ;ADC CS high
40 A F80F BDF817               JSR       TLC2543              Do A/D conversion
41 A F812 BDF84A               JSR       STORE                Store results
42 A F815 20E9                 BRA       START                Repeat above
43 A
44 A F817 CE1000     TLC2543   LDX       #BASEADD
45 A F81A 8602                 LDAA      #02
46 A F81C 1C0820     CSHIGH    BSET      PORTD,X#$20
47 A                 *  Set Port A bit 6 (TLC2543 CS) high
48 A F81F 4A                   DECA
49 A F820 26FA                 BNE       CSHIGH
50 A F822 1D0820               BCLR      PORTD,X#$20          ADC CS low
51 A F825 1E030210             BRSET     PORTC,X#$02 LSB      Do LSB first
52 A                 *     if LSBF set.
53 A F829 A603       MSB       LDAA      PORTC,X              Load Chan/mode data
54 A F82B A72A                 STAA      SPDR,X               Send the data to ADC
55 A                 *     and receive MSByte
56 A F82D 1F2980FC   LOOP1     BRCLR     SPSR,X#$80 LOOP1     If SPIF=0,
57 A                 *    repeat check
58 A F831 A62A                 LDAA      SPDR,X
59 A F833 97F0                 STAA      MSBYTE               Store MSByte
60 A F835 1E030210             BRSET     PORTC,X#$02 RETURN
61 A                 * If MSB/LSB-first bit = 1, return
62 A F839 A603       LSB       LDAA      PORTC,X              Load chan/mode data
63 A F83B A72A                 STAA      SPDR,X               Send the data to ADC
64 A                 *     and receive LSByte
65 A F83D 1F2980FC   LOOP2     BRCLR     SPSR,X#$80 LOOP2     If SPIF=0,
66 A F841 A62A                 LDAA      SPDR,X               repeat check
67 A F843 97F1                 STAA      LSBYTE               Store LSByte
68 A F845 1E0302E0             BRSET     PORTC,X#$02 MSB
```

List 3 (Continued)

```
   LINE   LOC   OBJ   SOURCE

69 A                 * If MSB/LSB–first bit = 1 go to MSB
70 A F849 39         RETURN    RTS
71 A                 *
72 A                 *  S'routine stores MSByte in even address
73 A                 *                   LSByte in odd address
74 A                 *  Channel 0 result in $30 and $31
75 A                 *  Channel 1 result in $32 and $33 etc.
76 A                 *  (Reserve addresses $30–$4B for results
77 A                 *   of all channels)
78 A F84A A603       STORE     LDAA      PORTC,X
79 A F84C CE0030               LDX       #$30
80 A F84F 97F2                 STAA      TEMP
81 A F851 86F0                 LDAA      #$F0
82 A F853 94F2                 ANDA      TEMP
83 A F855 46                   RORA
84 A F856 46                   RORA
85 A F857 46                   RORA
86 A F858 46                   RORA
87 A F859 16                   TAB
88 A F85A 8602                 LDAA      #$02
89 A F85C 3D                   MUL
90 A F85D DDF2                 STD       TEMP
91 A F85F 96F0                 LDAA      MSBYTE
92 A F861 A7F2                 STAA      TEMP,X
93 A F863 08                   INX
94 A F864 96F1                 LDAA      LSBYTE
95 A F866 A7F2                 STAA      TEMP,X
95 A F866 A7F2                 STAA   TEMP,X
96 A F868 39                   RTS
97 A                           END
```

TLC2543 to 80C51 Microcontroller Interface

Microcontroller Features

The 80C51 microcontroller family does not provide an SPI or equivalent facility. In order to implement the interface with the TLC2543 analog-to-digital converter, it is necessry to use software to synthesize the operation of an SPI. This results in a slower data transfer rate which is governed by the microcontroller's instruction cycle times. These are, in turn, influenced by the clock frequency of the microcontroller. The highest clock frequency possible should therefore be selected for the microcontroller to minimize instruction cycle times and thus optimize the data transfer rate of the interface.

Interface Circuit

Figure 13 shows the circuit for the interface of the TLC2543 to the 80C51 microcontroller. The I/O CLOCK, DATA INPUT and $\overline{CS}$ inputs to the TLC2543 are provided via the bidirectional parallel port 1 terminals P1.0, P1.1, and P1.3 respectively. Conversion result data from the TLC2543 is received by the 80C51 through the P1.2 terminal of port 1. The channel select/mode data is input to the microcontroller via port 3.

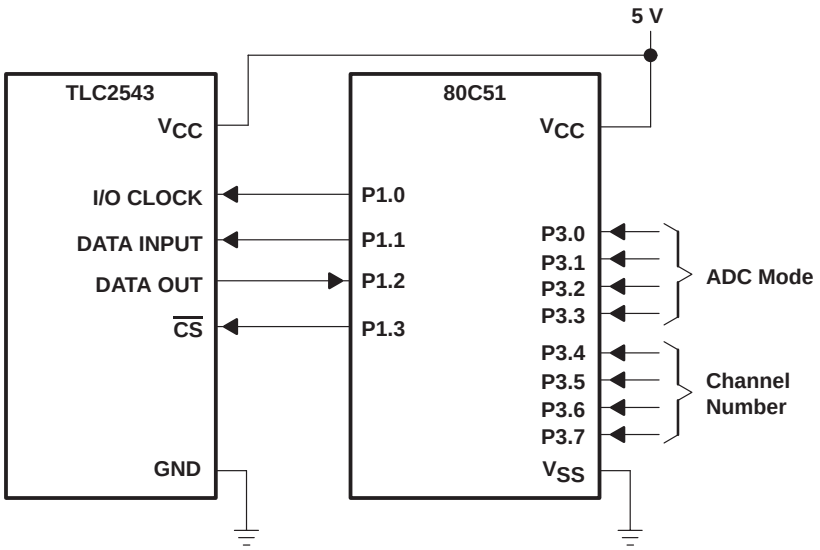


Figure 13. TLC2543 to 80C51 Microcontroller Interface

Software

The listing for the program used to control the interface circuit mentioned above is shown in List 4. As for the other microcontroller interface programs, it consists of a main program and two subroutines – TLC2543 and STORE.

The main program initializes the directions of the port 1 I/O terminals. P1.2 is configured as an input. P1.0, P1.1, and P1.3 are all programmed to perform as outputs. The chip select terminal of the TLC2543 is set high by setting P1.3. TLC2543 is then called. This subroutine contains the instructions which synthesize the SPI function and controls the exchange of data between the microcontroller and the TLC2543. The least significant bit first (LSBF) flag which is bit 1 of the channel select/mode data byte is checked to determine which byte (most significant – MSByte, least significant – LSByte) of the conversion result is to be expected first.

The SPI function is synthesized by using the accumulator in conjunction with the rotate left through carry (RLC) instruction to act as the SPI shift register. The following sequence provides a slow motion version of the SPI function.

The first bit of the first byte of the conversion result is read into the carry (C) bit. The contents of the accumulator are rotated left through carry and the first bit of the channel select/mode data is then output from P1.1. The first pulse of the serial clock is then provided by toggling the P1.0 bit of port 1 first high and then low. This sequence is repeated seven more times to complete the transfer of the first byte of data.

The second byte of data is tranferred between the TLC2543 and the 80C51 by repeating the entire sequence of eight sets of data transfer and clock pulse. The MSByte is placed in register 2 (R2) and the LSByte is placed in register 3 (R3). The subroutine STORE is used to map the MSByte and LSByte conversion results into even and odd number RAM addresses corresponding to the particular channel number which has been selected.

List 4

```
     LINE   LOC   OBJ   SOURCE

                        1     ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
                        2     ;*                                                *
                        3     ;*  TLC2543 12-bit Serial Out ADC to 80C51     *
                        4     ;*                                                *
                        5     ;*  Microcontroller Interface Program           *
                        6     ;*                                                *
                        7     ;*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
                        8     ;This program reads mode/channel select data into the
                        9     ;80C51 via Port 4 and transmits this data to the
                       10     ;TLC2543 at the same time as reading the result from
                       11     ;the previous conversion and storing the result in an
                       12     ;adjacent pair of memory locations from 30H to 4CH.
                       13     ;MSByte - Even Address  LSByte - Odd Address
                       14     ;MSByte Channel 0 in 30H, MSByte Channel 1 in 32H etc.
                       15     ;
0100                   16             ORG 100H
0100 758150            17     START:  MOV SP,#50H     ;Initialise Stack Pointer
0103 759004            18             MOV P1,#04H     ;Initialize port 1 I/O Pins
0106 C290              19             CLR P1.0        ;Set I/O clock low
0108 D293              20             SETB P1.3       ;Set chip select high
010A 74FF              21             MOV A,#00FFH
010C 3112              22             ACALL TLC2543
010E 313F              23             ACALL STORE
0110 80EE              24             JMP START
                       25
0112 ACB0              26     TLC2543:MOV R4,P3       ;Read mode/channel data into R4
0114 EC                27             MOV A,R4        ;and A
0115 C293              28             CLR P1.3        ;Set chip select low
0117 20E112            29             JB ACC.1,LSB    ;If bit 1 of A is 1,
                       30                             ;do LSByte first
011A 7D08              31     MSB:    MOV R5,#08      ;Load MS bit counter
011C A292              32     LOOP1:  MOV C,P1.2      ;Read data bit into carry
011E 33                33             RLC A           ;Rotate into accumulator
011F 9291              34             MOV P1.1,C      ;Output mode/channel bit
0121 D290              35             SETB P1.0       ;Set I/O clock high
0123 C290              36             CLR P1.0        ;Set I/O clock low
0125 DDF5              37             DJNZ R5,LOOP1   ;Get/send another bit
0127 FA                38             MOV R2,A        ;Put MSByte in R2
0128 EC                39             MOV A,R4        ;Put mode/channel data in A
0129 20E112            40             JB ACC.1,RETURN ;
012C 7D08              41     LSB:    MOV R5,#08      ;Load LS bit counter
012E A292              42     LOOP2:  MOV C,P1.2      ;Read data bit into carry
0130 33                43             RLC A           ;Rotate into accumulator
0131 9291              44             MOV P1.1,C      ;Output mode/channel bit
0133 D290              45             SETB P1.0       ;Set I/O clock high
0135 C290              46             CLR P1.0        ;Set I/O clock low
0137 DDF5              47             DJNZ R5,LOOP2   ;Get/send another bit
0139 FB                48             MOV R3,A        ;Put LSByte in R3
013A EC                49             MOV A,R4        ;Put mode/channel data in A
013B 20E1DC            50             JB ACC.1,MSB    ;If bit 1 of R4 is 1,
                       51                             ;do MSbyte next
013E 22                52     RETURN: RET
                       53
013F EC                54     STORE:  MOV A,R4        ;Put mode/channel data in A
0140 54F0              55             ANL A,#0F0H     ;Retain only channel number
0142 C4                56             SWAP A          ;Swap high and low nibble of A
0143 75F002            57             MOV B,#02
```

List 4 (Continued)

```
     LINE   LOC   OBJ   SOURCE

0146 A4              58            MUL AB
0147 2430            59            ADD A,#030H     ;Add 30H to accumulator
0149 F9              60            MOV R1,A
014A EA              61            MOV A,R2
014B F7              62            MOV @R1,A       ;Put MSByte in corresponding
                     63                            ;even number address :–
                     64                            ;Channel 0 in address 30H,
                     65                            ;Channel 1 in address 32H etc.
014C 09              66            INC R1
014D EB              67            MOV A,R3
014E F7              68            MOV @R1,A       ;Put LSByte in corresponding
                     69                            ;odd number address :–
                     70                            ;Channel 0 in address 31H,
                     71                            ;Channel 1 in address 33H etc.
014F 22              72            RET
                     73            END
```

Analog Considerations

Power Supply Decoupling

Care should be taken with the design of the printed circuit board when using 12-bit devices such as the TLC2543. The power supply terminal of each analog integrated circuit should be separately decoupled to the analog ground using a 0.1 μF ceramic capacitor. The inclusion of a 10 μF tantalum capacitor in parallel with the ceramic capacitor at each device supply terminal is also recommended, particularly in noisy environments.

Grounding

Separate ground return paths for analog and digital components back to the power supply should be used to prevent any noise currents induced by digital components from passing through the analog ground return path. These noise currents can induce noise voltages to occur in the analog ground return and thus corrupt the analog signal. Remember that, for a 5-V full scale signal, only 600 microvolts represents approximately half an LSB for a 12-bit ADC.

The important point to remember is that all ground return paths have a finite impedance. This impedance should be kept to a minimum by the use of wide printed circuit board tracks or ground planes where possible. A separate star connected ground topology is recommended for the analog components. This involves connecting each analog component's ground terminal to a central star point, which can then be connected via a wide printed circuit track to the power supply ground connection.

Board Layout

Digital devices and power switching elements should be kept as far away physically from analog components, such as the TLC2543, as possible. Particular attention should be paid to the use of switching power supplies. The high frequency switching currents which flow in the ground return paths of these space saving power blocks can introduce several LSBs of noise into 12-bit analog circuits. Linear regulated power supplies should be used or, if essential, switching regulators should be as far as possible from the analog circuitry with their outputs decouple.

Judicious use of ground planes can help to reduce analog ground impedances.

Figure 14 illlustrates a typical bypassing scheme for the TLC2543-to-TMS370 microcontroller interface.

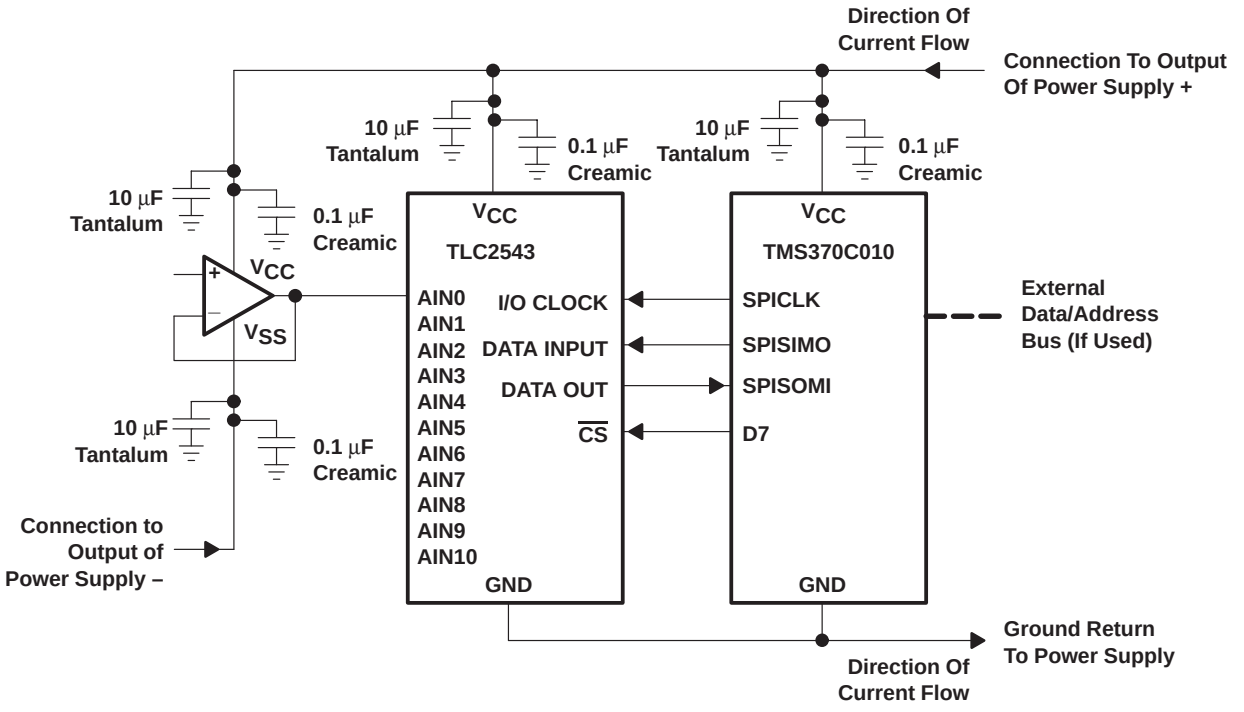


Figure 14. TLC2543 to Microcontroller Interface: Grounding and Decoupling Scheme

Appendix A

References

H8/325 Hardware User's Manual	Hitachi
H8/300 Series Programming Manual	Hitachi
Embedded Microcontrollers and Processors Vol 1	Intel Corporation
M68HC11 Reference Manual (1991)	Motorola Inc.
TMS370 Family Data Manual (1993)	Texas Instruments Incorporated
TLC2543 Data Sheet (Dec. '93)	Texas Instruments Incorporated

Acknowledgement

I wish to express my thanks to Mike Williams (Microcontroller Field Applications Engineer – Northern European Industrial Segment) for his useful comments on the TMS370 interface.