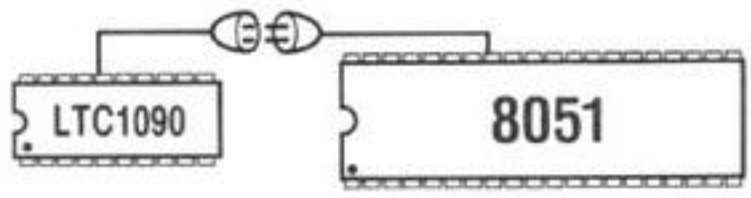
LINEAR TECHNOLOGY

Application Note 26A

January 1988

Interfacing the LTC1090 to the 8051 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes the hardware and software required for communication between the LTC1090 10-bit data acquisition system and the MCS-51 family of microcontrollers (e.g., 8051). The four wire interface is capable of completing a 10-bit conversion and transferring the data to the 8051 in 102μs. Configuration of the 8051 and the LTC1090 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be discussed. Finally, a summary of results including data throughput rates will be provided.

Interface Details

The serial port of the 8051 does not support the synchronous, full duplex format used by the LTC1090. Therefore it is necessary for the user to construct a serial port using four lines from one of the parallel ports available on the 8051. The lines are set or cleared using the bit manipulation features of the 8051. This provides a very flexible serial port but the data shift rate is three to four times slower than that available from microcontrollers with dedicated serial ports.

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. These lines may be tied together or run separately. The 8051 provides a pin (ALE) which can be used to drive the ACLK of the LTC1090 (option 1). Alternatively, tying the clocks together saves one line that has to go between the LTC1090 and the 8051 (option 2). However, this implementation slows the data throughput rate due to additional code. The schematic of Figure 1 shows both of these options.

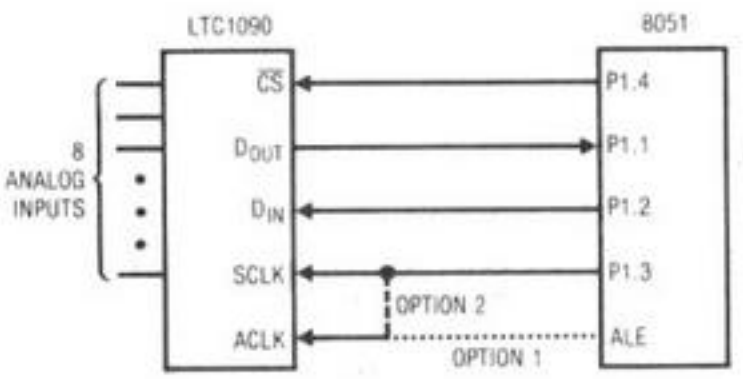


Figure 1. Schematic

Hardware Description

The 8051 was simulated and the code for this interface was developed on an Intel ICE 252 emulator.

Due to the weak pullups of the 8051, excess loading should be avoided when examining the output of the microcontroller.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using separate ACLK and SCLK (option 1). The 8051 clock rate was 12MHz, producing a 2.0MHz clock on the ALE pin.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

Software Description

The software simulates a serial port through bit manipulation instructions of the 8051. Additionally, the software generates a delay during which time the A/D conversion takes place.

The code sets up bit one of port one as an input by setting it high. (Due to the weak pullup of the 8051, the D_{OUT} pin of the LTC1090 can then drive the pin high or low.)

SCLK is initialized to a low state and $\overline{CS}$ is initialized to a high state. A D_{IN} word of $0D is then loaded into the accumulator. An examination of Figure 3 and the data sheet will show that this configures the LTC1090 for CH0 with respect to CH1, unipolar, MSB first and a 10-bit word length. Next $\overline{CS}$ goes low. If the user is tying ACLK and SCLK together (option 2) it is then necessary to generate two clock pulses to meet the deglitcher requirements. With separate clocks (option 1) the NOP is necessary to allow sufficient time for the deglitcher before starting to shift the data. Data is moved from the P1.1 pin (D_{OUT} of the LTC1090) to the carry register and shifted one bit at a time into the accumulator. At the same time, the 8-bit D_{IN} word is shifted from the accumulator into the carry register and output on P1.2 (D_{IN} of the LTC1090).

After the eight MSBs have been shifted, the contents of the accumulator are stored in R2. The final two bits are then shifted into the accumulator, placed in the most significant bits and stored in R3. The data is left justified at this point with the MSBs in R2 and the LSBs in R3. $\overline{CS}$ is then raised and time (44 ACLK cycles) for the LTC1090 to do its next conversion must be allowed before the next read can be performed. If separate clocks are being used (option 1), quite often the microcontroller will have other tasks to accomplish and this time can be used productively. Otherwise, a routine such as the one labeled DELAY can be used. With the clocks tied together (option 2), it is necessary for the 8051 to manually clock the LTC1090 44 times and this free time is then lost as shown in Figure 7. An example of this routine is labeled LOOP 1.

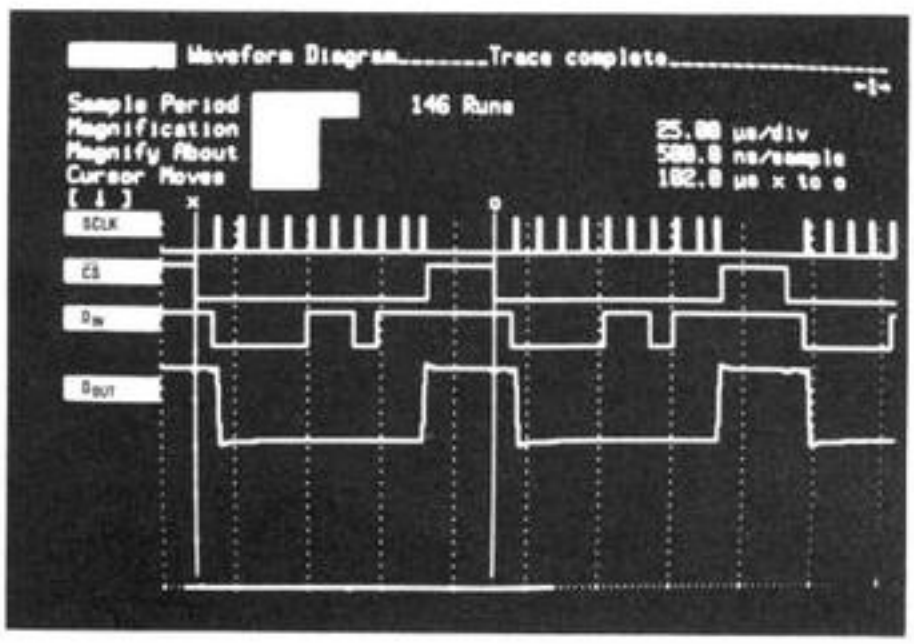


Figure 2. Timing Diagram for Option 1

If right justified data is required, the MSBF bit of the D_{IN} word could be cleared and the bits reversed (in this case producing a D_{IN} word of $90). Also it would be necessary to swap the rotate left and rotate right instructions.

0	0	0	0	1	1	0	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 3. D_{IN} Word for LTC1090

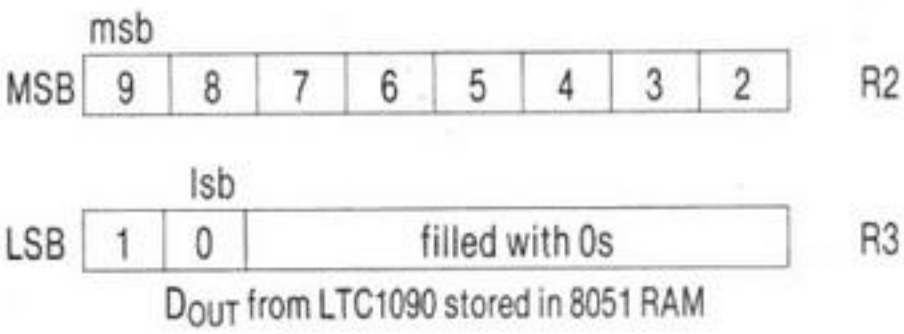


Figure 4. Memory Map

Summary

A four wire interface between the LTC1090 and the 8051 with a combined data conversion and transfer time of 102μs was demonstrated. It was shown that the ACLK of the LTC1090 can be run separately from the SCLK by tying the ACLK to the ALE of the 8051. Alternatively, the two clock pins can be tied together (saving one line at the expense of speed). The data can be either left justified or right justified in the microcontroller's memory through the proper choice of software and LTC1090 data format. The code shown applies to all MCS-51 family members. The same technique can be used on any parallel port processor.

```
LABEL   MNEMONIC        COMMENTS
        MOV     P1, #02H        BIT 1 PORT 1 SET AS INPUT
        CLR     P1.3            SCLK GOES LOW
        SETB    P1.4            CS GOES HIGH
CONT    MOV     A, #0DH         DIN WORD FOR LTC1090
        CLR     P1.4            CS GOES LOW
        MOV     R4, #08H        LOAD COUNTER
        NOP                     DELAY FOR DEGLITCHER
LOOP    MOV     C, P1.1         READ DATA BIT INTO CARRY
        RLC     A               ROTATE DATA BIT INTO ACC
        MOV     P1.2, C         OUTPUT DIN BIT TO LTC1090
        SETB    P1.3            SCLK GOES HIGH
        CLR     P1.3            SCLK GOES LOW
        DJNZ    R4, LOOP        NEXT BIT
        MOV     R2, A           STORE MSBs IN R2
        MOV     C, P1.1         READ DATA BIT INTO CARRY
        CLR     A               CLEAR ACC
        RLC     A               ROTATE DATA BIT INTO ACC
        SETB    P1.3            SCLK GOES HIGH
        CLR     P1.3            SCLK GOES LOW
        MOV     C, P1.1         READ DATA BIT INTO CARRY
        RRC     A               ROTATE RIGHT INTO ACC
        RRC     A               ROTATE RIGHT INTO ACC
        MOV     R3, A           STORE LSBs IN R3
        SETB    P1.3            SCLK GOES HIGH
        CLR     P1.3            SCLK GOES LOW
        SETB    P1.4            CS GOES HIGH

        MOV     R5, #07H        LOAD COUNTER
DELAY   DJNZ    R5, DELAY       GO TO DELAY IF NOT DONE
```

Figure 5. 8051 Code for Option 1 (ACLK Tied to ALE)

```
LABEL     MNEMONIC          COMMENTS
          MOV     P1, #02H  BIT 1 PORT 1 SET AS INPUT
          CLR     P1.3      SCLK GOES LOW
          SETB    P1.4      CS GOES HIGH
CONT      MOV     A, #0DH   DIN WORD FOR LTC1090
          CLR     P1.4      CS GOES LOW
          SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          MOV     R4, #08H  LOAD COUNTER
LOOP      MOV     C, P1.1   READ DATA BIT INTO CARRY
          RLC     A         ROTATE DATA BIT INTO ACC
          MOV     P1.2, C   OUTPUT DIN BIT TO LTC1090
          SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          DJNZ    R4, LOOP  NEXT BIT
          MOV     R2, A     STORE MSBs IN R2
          MOV     C, P1.1   READ DATA BIT INTO CARRY
          CLR     A         CLEAR ACC
          RLC     A         ROTATE DATA BIT INTO ACC
          SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          MOV     C, P1.1   READ DATA BIT INTO CARRY
          RRC     A         ROTATE RIGHT INTO ACC
          RRC     A         ROTATE RIGHT INTO ACC
          MOV     R3, A     STORE LSBs IN R3
          SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          SETB    P1.4      CS GOES HIGH

          MOV     R4, #2CH  LOAD COUNTER
LOOP 1    SETB    P1.3      SCLK GOES HIGH
          CLR     P1.3      SCLK GOES LOW
          DJNZ    R4, LOOP 1 GO TO LOOP 1 IF NOT DONE
```

Figure 6. 8051 Code for Option 2 (ACLK Tied to SCLK)

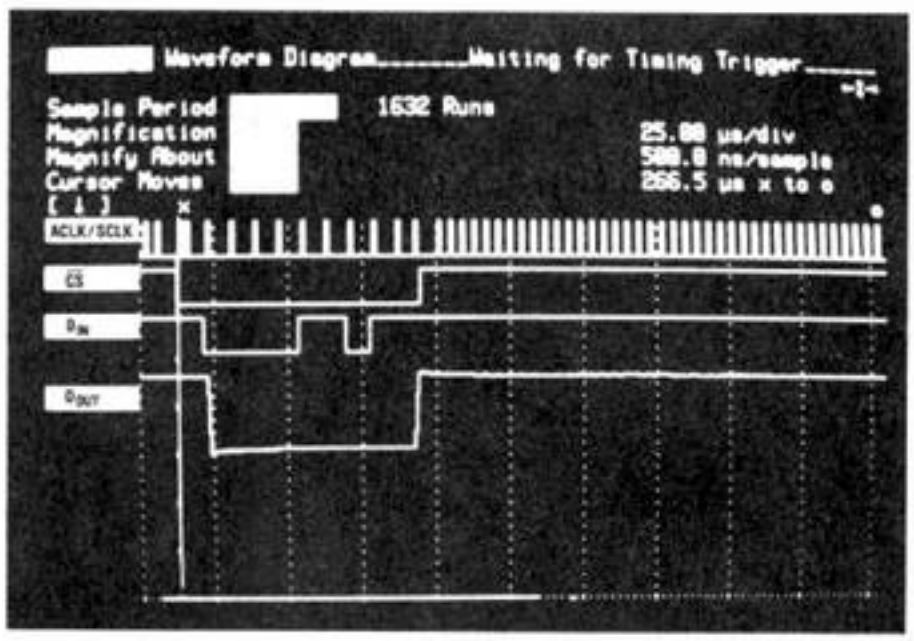


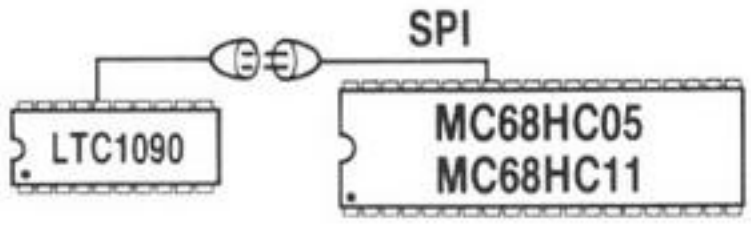
Figure 7. Timing Diagram for Option 2

LINEAR TECHNOLOGY

Application Note 26B

December 1987

Interfacing the LTC1090 to the MC68HC05 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1090 10-bit data acquisition system and the Motorola SPI family of single chip microcomputers (e.g., 68HC05). The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 68HC05 in 49μs. Configuration of the LTC1090 and the 68HC05 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given, including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a synchronous, full duplex format over D_{IN} and D_{OUT}.

The Motorola Serial Peripheral Interface (SPI) is a synchronous, full duplex, serial port built into the 68HC05 that allows the user to construct a simple communication path to the LTC1090. SPI provides clock, data in and data out lines that are compatible with the LTC1090. The only additional line required is one programmable output pin (CO) to control $\overline{CS}$ on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

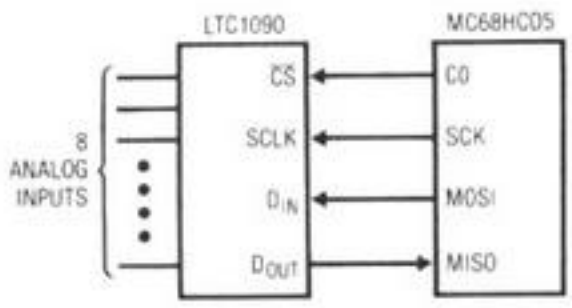


Figure 1. Schematic

Hardware Description

The 68HC05 was emulated and the code for this interface was developed on a Motorola M68HC05 EVM.

$\overline{SS}$ (Pin 34) of the 68HC05 must be held high to enable the SPI properly for this interface.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using a 2MHz ACLK. The 68HC05 clock was 4MHz.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

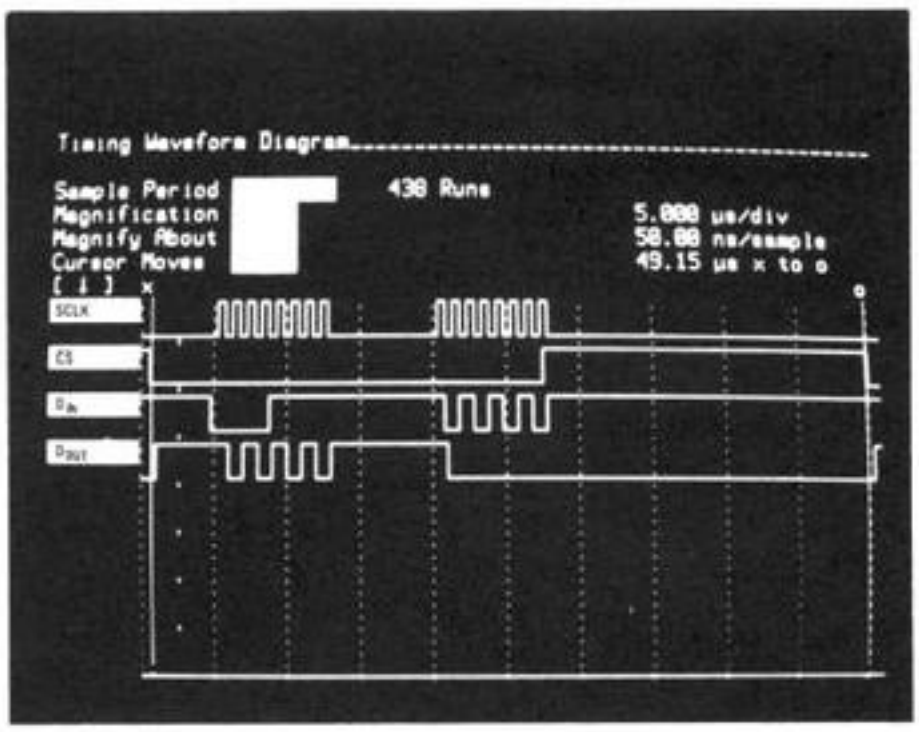


Figure 2. Timing Diagram

Software Description

The software configures and controls the SPI of the 68HC05. Additionally, the software manipulates CO ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion.

The code first configures the Serial Peripheral Control Register (SPCR) of the SPI. The SPI interrupt is disabled. The SPI outputs are enabled. The SPI is configured as a master. Finally, the SPI clock is set to normally low, for data transfer on the rising edge and for a frequency equal to half the internal processor clock (one fourth the crystal frequency).

Port C is configured as all outputs by placing ones in the data direction register of port C. A D_{IN} word that configures the LTC1090 for CH0 with respect to CH1, unipolar, MSB first and a 16-bit word length is stored in $50. Figure 3 shows how the D_{IN} word is composed.

CO is made to go low. D_{IN} for the LTC1090 is loaded into the SPI data register. Storing D_{IN} in the data register causes the transfer to begin. After waiting for the first eight bits to be transferred (8 NOPs) the status register of the SPI is examined. This clears the SPIF bit of the status register and allows the data register to be read, which is the next step. The first eight bits containing the MSBs from the LTC1090 are then stored in $61 of the 68HC05 as shown in Figure 4. The LSBs are transferred in the same manner and stored in $62 of the 68HC05. Notice in Figure 5 that only 6 NOPs are used in transferring the LSBs. This is because after 6 NOPs, time is consumed by the BSET command which sets the CO pin of the 68HC05. The data at this point is left justified.

0	0	0	0	1	1	1	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 3. D_{IN} Word for LTC1090

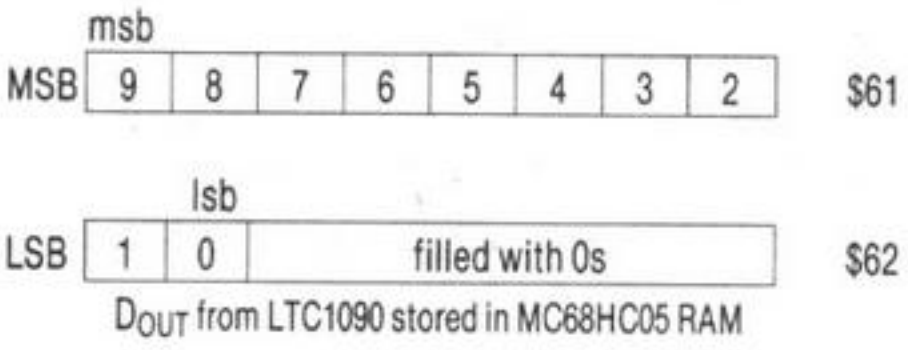


Figure 4. Memory Map

At this time 44 ACLK cycles must be allowed for the A/D to perform its next conversion. Usually the processor will have other tasks to perform during this time. If this is not the case a string of NOPs or a simple delay loop can be used to generate this delay.

The code was written for the 68HC05. By changing the addresses of the special function registers however, the code should run on all of Motorola's SPI processors including the 68HC11.

Summary

A four wire interface between the LTC1090 and the 68HC05 with a combined data conversion and transfer time of 49μs was demonstrated. The interface used the serial (SPI) port of the 68HC05. The 10 data bits of the LTC1090 are shifted MSB first in two eight bit transfers. The data is stored left justified in the 68HC05's internal RAM.

```
LABEL    MNEMONIC          COMMENTS
         LDA    #$50       CONFIGURATION DATA FOR SPCR
         STA    $0A        LOAD DATA INTO SPCR ($0A)
         LDA    #$FF       CONFIG. DATA FOR PORT C DDR
         STA    $06        LOAD DATA INTO PORT C DDR
         LDA    #$0F       LOAD LTC1090 DIN DATA INTO ACC
         STA    $50        LOAD LTC1090 DIN DATA INTO $50
START    BCLR   0,$02      CO GOES LOW (CS GOES LOW)
         LDA    $50        LOAD DIN INTO ACC FROM $50
         STA    $0C        LOAD DIN INTO SPI. START SCK

         NOP               8 NOPs FOR TIMING

         LDA    $0B        CHECK SPI STATUS REG
         LDA    $0C        LOAD LTC1090 MSBs INTO ACC
         STA    $61        STORE MSBs IN $61
         STA    $0C        START NEXT SPI CYCLE

         NOP               6 NOPs FOR TIMING

         BSET   0,$02      CO GOES HIGH (CS GOES HIGH)
         LDA    $0B        CHECK SPI STATUS REGISTER
         LDA    $0C        LOAD LTC1090 LSBs INTO ACC
         STA    $62        STORE LSBs IN $62
```

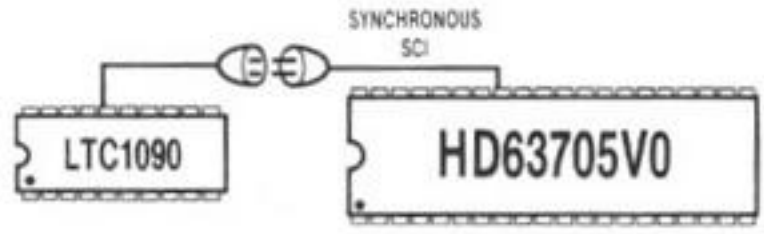
Figure 5. 68HC05 Code

LINEAR TECHNOLOGY

Application Note 26C

January 1988

Interfacing the LTC1090 to the HD63705V0 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1090 10-bit data acquisition system and the Hitachi 63705 microcomputer. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 63705 in 45μs. Configuration of the LTC1090 and the 63705 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a full duplex format over D_{IN} and D_{OUT}.

The Hitachi Serial Communication Interface (SCI) is a synchronous, full duplex, serial port built into the 63705 that allows the user to construct a simple communication path to the LTC1090. SCI provides clock, transmit and receive lines that are compatible with the LTC1090. The only additional line required is one programmable output pin (C0) to control $\overline{CS}$ on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

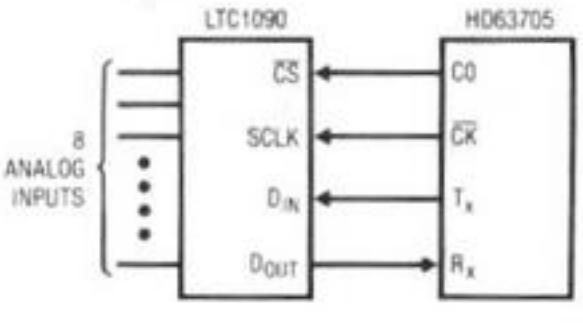


Figure 1. Schematic

Hardware Description

The 63705 was emulated and the code for this interface was developed on a Hitachi H35MIX3 emulator.

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. ACLK of the LTC1090 was 2 MHz and the 63705 clock was 4 MHz.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

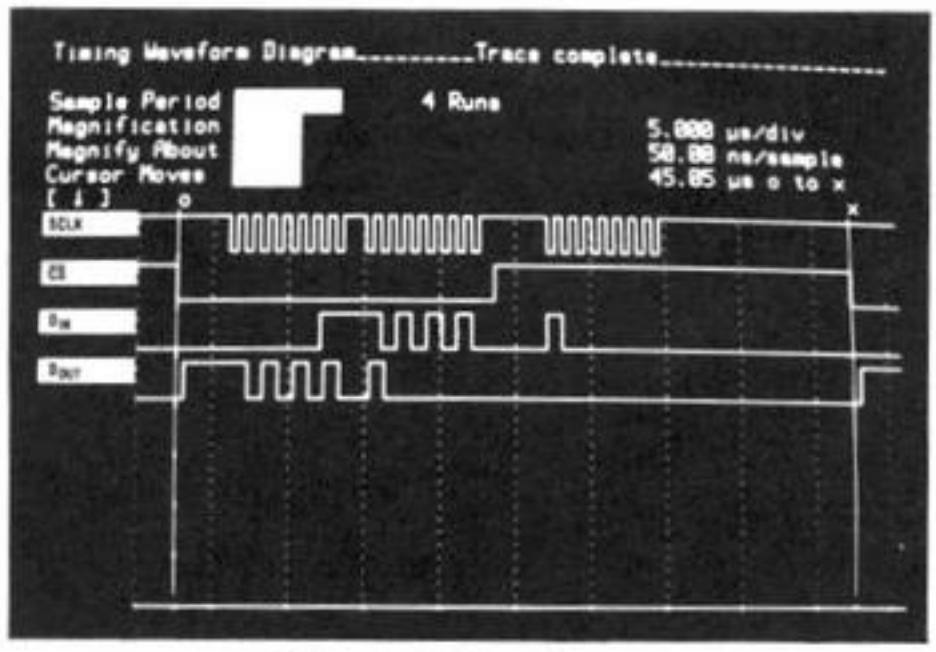


Figure 2. Timing Diagram

Software Description

The software configures and controls the SCI of the 63705. Additionally, the software manipulates C0 ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion.

The code first configures the SCI control register (SCR). D3 and D4 are set as the SCI output and input respectively. D5 is selected as a clock output with a frequency one fourth the crystal frequency. Next, the SCI status register (SSR) is configured so that the interrupts are disabled. Data is transmitted on the falling edge of the clock and received on the rising edge of the clock.

Port C is configured as all outputs by setting the data direction register (address $06). A D_{IN} word that configures the LTC1090 for CH0 with respect to CH1, bipolar, LSB first and a 16-bit word length is stored in $50. Figure 3 shows how the D_{IN} word is composed. Note, that for LSB first format the D_{IN} word must be constructed opposite from MSB first format. This is so that each bit of the D_{IN} word is always shifted into the LTC1090 in the same order.

C0 is made to go low. D_{IN} for the LTC1090 is loaded into the SCI data register (SDR). Storing D_{IN} in the SDR causes the transfer to begin. After waiting for the first eight bits to be transferred (6 NOPs) the data containing the LSBs from the LTC1090 is loaded into the accumulator and then stored in $61 of the 63705 RAM. The act of reading the LSBs into the ACC causes the next SCI transfer to begin. C0 is set and the MSBs from the LTC1090 are loaded into the ACC and then stored in $62 of the 63705 RAM.

1	1	0	0	0	0	0	0	
WL0	WL1	MSBF	UNI	S2	S1	O/S	S/D	$50

Figure 3. D_{IN} Word for LTC1090 Stored in 63705 RAM

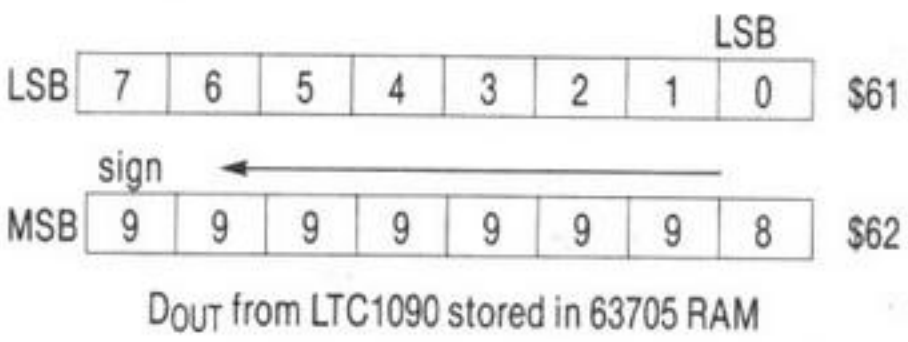


Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
	LDA	#$E0	CONFIGURATION DATA FOR SCR
	STA	$10	LOAD DATA INTO SCR ($10)
	LDA	#$30	CONFIGURATION DATA FOR SSR
	STA	$11	LOAD DATA INTO SSR ($11)
	LDA	#$FF	CONFIG. DATA FOR PORT C DDR
	STA	$06	LOAD DATA INTO PORT C DDR
	LDA	#$C0	LOAD LTC1090 D_{IN} DATA INTO ACC
	STA	$50	LOAD LTC1090 D_{IN} DATA INTO $50
START	LDA	$50	LOAD D_{IN} INTO ACC FROM $50
	BCLR	0,$02	C0 GOES LOW ($\overline{CS}$ GOES LOW)
	STA	$12	LOAD D_{IN} INTO SDR. START SCK
	NOP		6 NOPs FOR TIMING
	LDA	$12	LOAD LSBs. START NEXT CYCLE
	STA	$61	STORE LSBs IN $61
	NOP		1 NOP FOR TIMING
	BSET	0,$02	C0 GOES HIGH ($\overline{CS}$ GOES HIGH)
	LDA	$12	LOAD MSBs
	STA	$62	STORE MSBs IN $62

Figure 5. 63705 Code

The data at this point is right justified with the sign bit (B9) being extended into B1-B7 of $62 as shown in Figure 4.

At this time 44 ACLK cycles must be allowed for the A/D to perform its next conversion. Usually the processor will have other tasks to perform during this time (you have to do something with the data once the processor has it) . If this is not the case, a string of NOPs or a simple delay loop can be used to generate this delay.

Summary

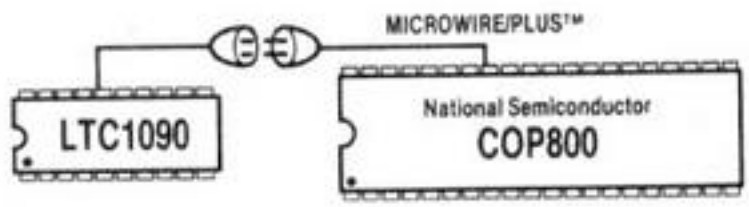
A four wire interface between the LTC1090 and the Hitachi 63705 with a combined data conversion and transfer time of 45μs was demonstrated. The interface used the serial (SCI) port of the 63705. Because the SCI port transfers data LSB first, care must be taken to properly construct the D_{IN} word so that the bits are transmitted in the proper order to the LTC1090. The 10 data bits of the LTC1090 are shifted LSB first in two eight bit transfers. The data is stored right justified in the 63705s internal RAM.

LINEAR TECHNOLOGY

Application Note 26D

January 1988

Interfacing the LTC1090 to the COP820C MCU



Guy Hoover
William Rempfer

Introduction

This application note describes the hardware and software required for communication between the LTC1090 10-bit data acquisition system and the National Semiconductor COP800 microcontroller family which uses the MICROWIRE/PLUS serial interface. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data in 56μs. Configuration of the LTC1090 and the COP820C will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a full duplex format over D_{IN} and D_{OUT}.

The National Semiconductor MICROWIRE/PLUS is a synchronous, full duplex, serial port built into the COP800 family that allows easy interface to the LTC1090. MICROWIRE/PLUS provides clock, data in and data out lines that are compatible with the LTC1090. One additional line (G1) is required to control the $\overline{CS}$ pin on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The actual interface was done using the COP820C, a member of the COP800 family. All code shown here should work with any of the COP800 family.

The code for this interface was developed on a COP820 evaluation board operated in the emulation mode.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using a 2MHz ACLK. The COP820C clock was 5MHz. To obtain a 56μs transfer time it is necessary to run the COP820C at 20MHz which requires a high speed version of the part.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

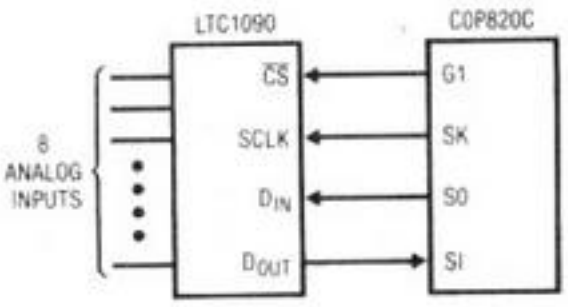


Figure 1. Schematic

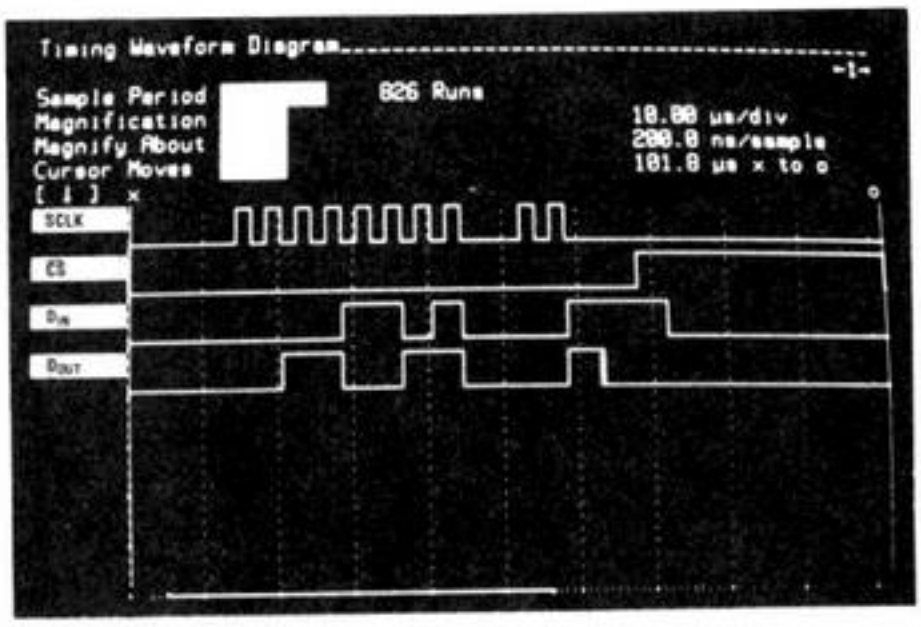


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE/PLUS serial interface of the COP820C. Additionally, the software manipulates G1 ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion.

The code first loads the LTC1090 D_{IN} word into memory location. $F0. This D_{IN} word configures the LTC1090 for CH0 with respect to CH1, MSB first, unipolar and 10 bits as shown in Figure 3. Next port G is configured for MICROWIRE™ master mode and G1 is configured as an output. The control register is initialized so that SO and SK are outputs. The port G data register address is loaded into the B register. At this point the COP820C is initialized and the data transfer process is ready to begin.

The D_{IN} word for the LTC1090 is then loaded into the ACC from location $F0. G1 ($\overline{CS}$) is cleared and D_{IN} is transferred into the MICROWIRE shift register. The BUSY bit of the PSW register is set which starts the transfer of the first eight bits. A delay consisting of 15 NOPs waits for the data shift to finish at which time the D_{OUT} word from the LTC1090 is loaded into the ACC. The busy bit is set again which causes the transfer to continue. Then, the D_{OUT} word in the ACC is stored in location $F3. The busy bit is cleared which halts the transfer. Two more bits have been shifted at this point. G1 ($\overline{CS}$) is set and the contents of the MICROWIRE shift register are swapped with those of the ACC. The carry is cleared and the data in the ACC is shifted right, through the carry bit three times. This puts the two LSBs of the D_{OUT} word in the MSBs of the ACC. The contents of the ACC are then stored in $F4. The data at this point is left justified as shown in Figure 4.

0	0	0	0	1	1	0	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 3. D_{IN} Word for LTC1090

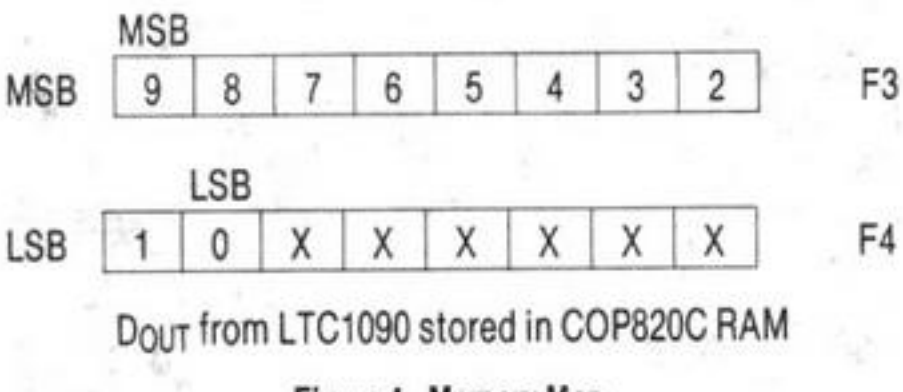


Figure 4. Memory Map

LABEL	MNEMONIC	COMMENTS
	LD (F0) – 0D	LOAD 0D INTO F0 (D_{IN})
	LD (D5) – 32	CONFIGURE PORT G
	LD (EE) – 8	CONFIGURE CONTROL REG.
	LD (B) – D4	PORT G DATA REG. INTO B
LOOP	LD (A) – (F0)	LOAD D_{IN} INTO ACC
	RBIT 1	G1 RESET ($\overline{CS}$ GOES LOW)
	X (A) – – (E9)	LOAD D_{IN} INTO SHIFT REG.
	LD (B) – EF	LOAD PSW REG ADDR IN B
	SBIT 2	TRANSFER BEGINS
	NOP	15 NOPs FOR TIMING
	X (A) – – (E9)	LOAD D_{OUT} INTO ACC
	SBIT 2	TRANSFER CONTINUES
	X (A) – – (F3)	LOAD D_{OUT} IN ADDR F3
	RBIT 2	STOP TRANSFER
	LD (B) – D4	PUT PORT G ADDR IN B
	SBIT 1	G1 SET ($\overline{CS}$ GOES HIGH)
	X (A) – – (E9)	LOAD D_{OUT} INTO ACC
	RC	CLEAR CARRY
	RRCA	SHIFT RIGHT THRU CARRY
	RRCA	SHIFT RIGHT THRU CARRY
	RRCA	SHIFT RIGHT THRU CARRY
	X (A) – – (F4)	LOAD D_{OUT} IN ADDR F4

Figure 5. COP820C Code

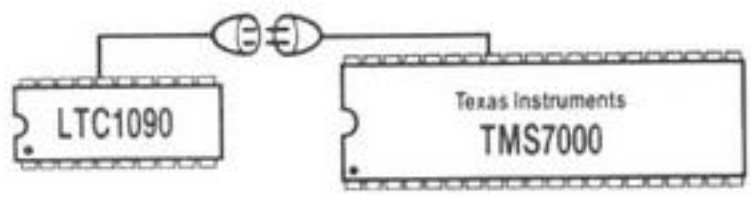
44 ACLK cycles must be allowed between transfers for the A/D to perform its next conversion. The instructions, after G1 is set, take enough time so that no additional delay is required by this program.

Summary

A four wire interface between the LTC1090 and the COP820C with a combined data conversion and transfer time of 56μs was demonstrated. The interface used the MICROWIRE/PLUS serial port of the COP820C. The 10 data bits of the LTC1090 are shifted MSB first in one eight bit and one two bit transfer. The data is stored left justified in the COP820C's internal RAM.

MICROWIRE and MICROWIRE/PLUS are trademarks of National Semiconductor Corp.

Interfacing the LTC1090 to the TMS7742 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1090 10-bit data acquisition system and the TMS7000 family of microcomputers (e.g., TMS7742). The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the TMS7742 in 103μs. Configuration of the LTC1090 and the TMS7742 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a full duplex format over D_{IN} and D_{OUT}.

The TMS7742 contains a synchronous, full duplex, serial port that allows the user to construct a simple communication path to the LTC1090. The serial port provides clock, transmit and receive lines that are compatible with the LTC1090. The only additional line required is one programmable output pin (A0) to control $\overline{CS}$ on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The TMS7742 was chosen because it contains 4k of EPROM which can be programmed using a standard EPROM programmer. Any member of the TMS7000 family which contains a serial port should be able to use this code with only modifications to the peripheral register numbers.

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. ACLK of the LTC1090 was 2 MHz and the TMS7742 clock was 5 MHz.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

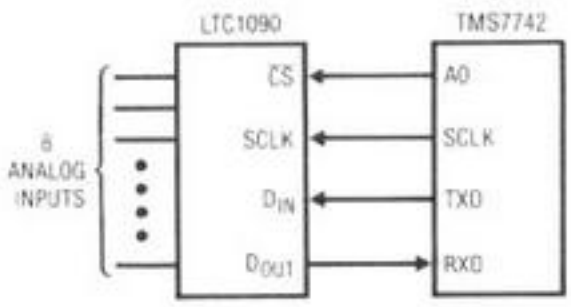


Figure 1. Schematic

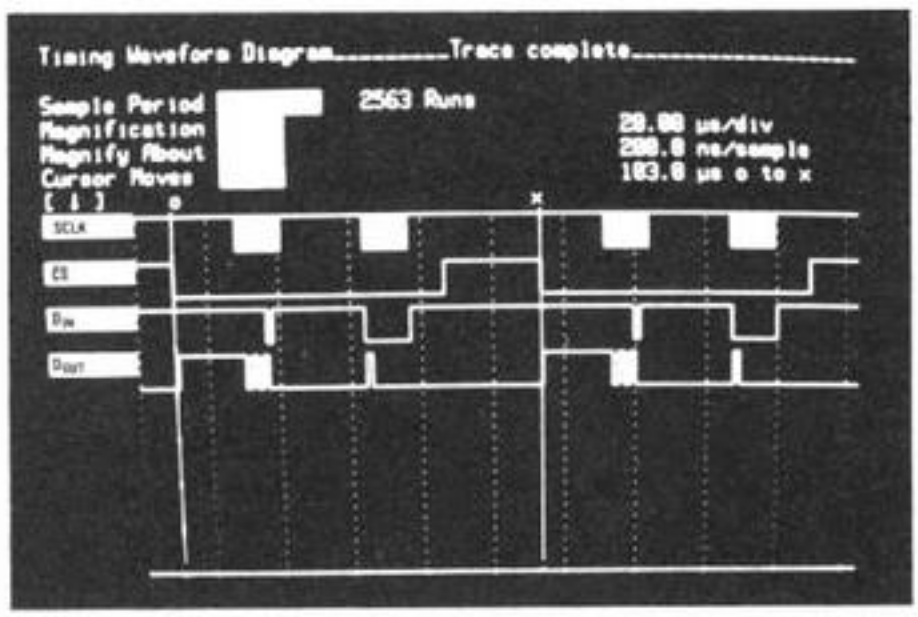


Figure 2. Timing Diagram

Software Description

The software configures and controls the serial port of the TMS7742. Additionally, the software manipulates A0 ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion.

The code first disables all interrupts and initializes the stack. Next the serial port is configured. Tx is enabled, the serial port is reset, and the SMODE register is configured for 8 bits, no parity, and one stop bit. The SCLK rate is set to the processor clock frequency divided by 4. The D_{IN} word of the LTC1090 is next loaded into the ACC. This D_{IN} word ($DF) configures the LTC1090 for CH7 with respect to COM, unipolar mode, LSB first and a 16-bit word length. Examine Figure 3 to see how this is constructed keeping in mind that the TMS7742 transmits data LSB first.

A subroutine SXTNBIT is called next. This is a routine that does the actual data shifting. A0 ($\overline{CS}$) is cleared. Then, the LTC1090 D_{IN} word is placed into the transmit buffer. The serial port is turned on and the data is shifted while the processor idles in a loop. The first eight bits containing the LSBs are then placed in the B register. The procedure is repeated for the next eight bits which contain the two MSBs and the result is placed in the A register. A0 ($\overline{CS}$) is then set and the subroutine returns to the original program. The data in the A and B registers is then stored in R5 and R6.

At this time 44 ACLK cycles must be allowed for the A/D to perform its next conversion. Enough time is consumed by this program however that no additional delay for the conversion is required.

1	1	0	1	1	1	1	1	
WL0	WL1	MSBF	UNI	S2	S1	O/S	S/D	P5

Figure 3. D_{IN} Word for LTC1090 Stored in TMS7742 RAM

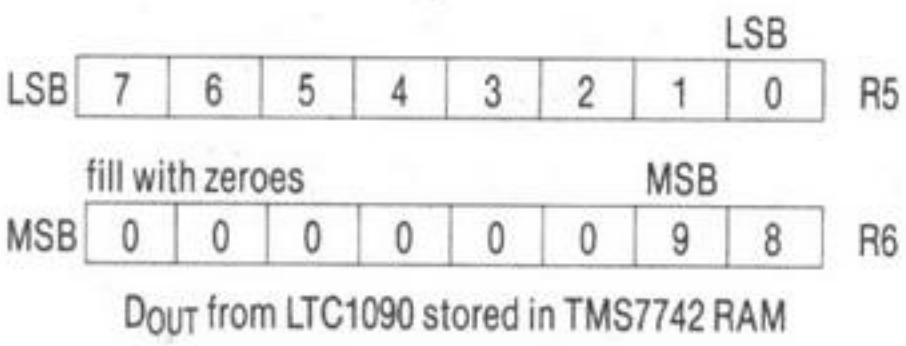


Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
START	DINT		DISABLES ALL INTERRUPTS
	MOVP	%>2A,P0	DISABLE INTERRUPT FLAGS
	MOVP	%>02,P16	DISABLE INTERRUPT FLAGS
	MOV	%>60,B	ADDRESS OF STACK
	LDSP		PUT ADDRESS INTO POINTER
	MOVP	%>DF,P5	CONFIGURE PORT A
	MOVP	%>08,P6	ENABLE Tx BY SETTING B3 = 1
	MOVP	%>00,P17	P17 POINTS TO SCTL0
	MOVP	%>40,P17	RESET THE SERIAL PORT
	MOVP	%>0C,P17	CONFIGURE THE SERIAL PORT
	MOVP	%>00,P21	TURN START BIT OFF
	MOVP	%>00,P17	ENABLE THE SERIAL PORT
	MOVP	%>00,P20	SET SCLK RATE (TIMER 3)
	MOVP	%>C0,P21	START TIMER
LOOP	MOV	%>DF,A	LOAD LTC1090 D_{IN} WORD IN A
	CALL	SXTNBIT	ROUTINE THAT SHIFTS DATA
	MOV	B,R5	PUT FIRST 8 LSBs IN R5
	MOV	A,R6	PUT MSBs IN R6
SXTNBIT	ANDP	%>FE,P4	A0 CLEARED ($\overline{CS}$ GOES LOW)
	MOVP	A,P23	PUT LTC1090 D_{IN} INTO TXBUF
	MOVP	%>40,P21	SCLK OFF (TIMER 3 DISABLED)
	MOVP	%>17,P17	ENABLE SERIAL PORT
	MOVP	%>C0,P21	SCLK ON (TRANSFER BEGINS)
	MOVP	%>14,P17	TXEN GOES LOW
	MOV	%>02,A	LOAD COUNTER
WAIT1	DJNZ	A,WAIT1	LOOP WHILE SHIFT OCCURS
	NOP		DELAY
	MOVP	P22,B	PUT D_{OUT} FROM LTC1090 IN B
	MOVP	A,P23	LOAD TXBUF
	MOVP	%>40,P21	SCLK OFF (TIMER 3 DISABLE)
	MOVP	%>17,P17	ENABLE SERIAL PORT
	MOVP	%>C0,P21	SCLK ON (TRANSFER BEGINS)
	MOVP	%>14,P17	TXEN GOES LOW
	MOV	%>02,A	LOAD COUNTER
WAIT2	DJNZ	A,WAIT2	LOOP WHILE SHIFT OCCURS
	NOP		DELAY
	MOVP	P22,A	PUT D_{OUT} FROM LTC1090 IN A
	ORP	%>01,P4	A0 SET ($\overline{CS}$ GOES HIGH)
	RETS		RETURN TO MAIN PROGRAM

Figure 5. TMS7742 Code

Summary

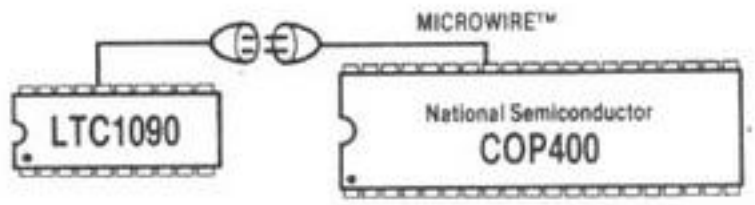
A four wire interface between the LTC1090 and the TMS7742 with a combined data conversion and transfer time of 103μs was demonstrated. The interface used the serial port of the TMS7742. Because the serial port transfers data LSB first, care must be taken to properly construct the D_{IN} word so that the bits are transmitted in the proper order to the LTC1090. The 10 data bits of the LTC1090 are shifted LSB first in two eight bit transfers. The data is stored right justified in the TMS7742's internal RAM.

LINEAR TECHNOLOGY

Application Note 26F

January 1988

Interfacing the LTC1090 to the COP402N MCU



Guy Hoover
William Rempfer

Introduction

This application note describes the hardware and software required for communication between the LTC1090 10-bit data acquisition system and the National Semiconductor COP400 microcontroller family which uses the MICROWIRE serial interface. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data in 100μs. Configuration of the LTC1090 and the COP402N will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a full duplex format over D_{IN} and D_{OUT}.

The National Semiconductor MICROWIRE interface is a synchronous, full duplex, serial port built into the COP400 family that allows the user to easily interface to the LTC1090. MICROWIRE provides clock, data in and data out lines that are compatible with the LTC1090. One additional line (G0) is required to control the $\overline{CS}$ pin on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The actual interface will involve using the COP402N, a member of the COP400 family. All code shown here should work with any of the COP400 family.

The code for this interface was developed on a COP400 evaluation board which allows an external EPROM to be used in place of the internal processor ROM.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using a 2MHz ACLK. The COP402N clock was 4MHz.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

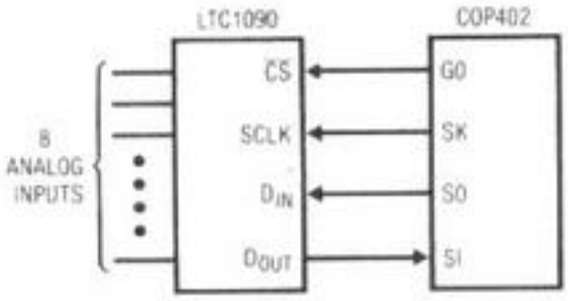


Figure 1. Schematic

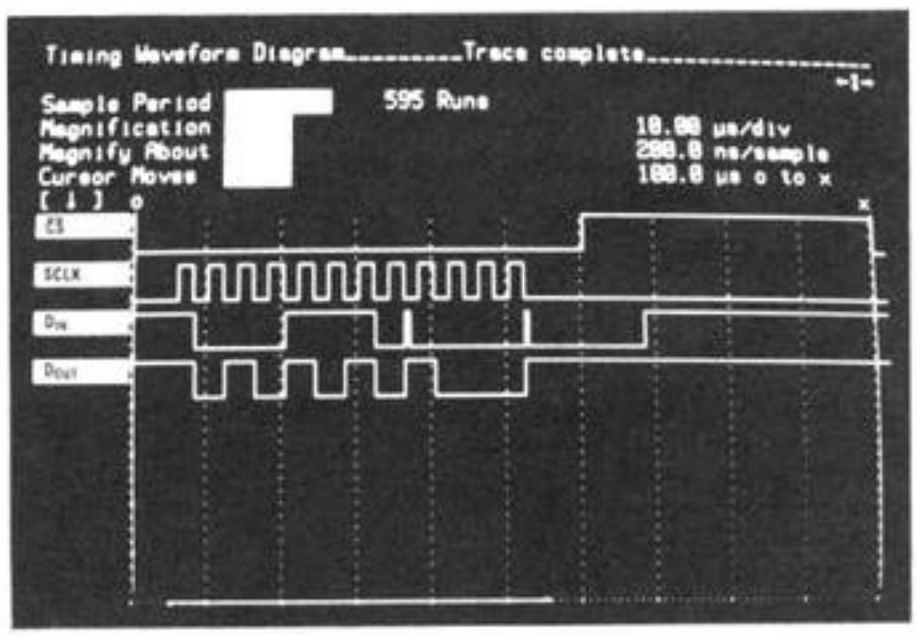


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE serial interface of the COP402N. Additionally, the software manipulates G0 ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion.

The code first initializes the B register and then loads the LTC1090 D_{IN} word into the RAM of the COP402 one nibble at a time. As shown in Figure 3 the D_{IN} word configures the LTC1090 for CH0 with respect to COM, unipolar, MSB first, and 12 bits. SO is configured as an output. The carry is set so that when an XAS instruction is generated the shift clock (SK) will begin clocking data.

The first nibble of the D_{IN} word is loaded into the ACC and G0 ($\overline{CS}$) is cleared. The D_{IN} nibble is loaded into the shift register and the data begins to shift. The second nibble of the D_{IN} word is loaded into the ACC. One NOP is allowed for timing and then the contents of the ACC are swapped with those of the shift register. The MSBs of the LTC1090 D_{OUT} word are now in the ACC. This data is then stored in memory location $13. The ACC is loaded with null data from RAM and another swap between the ACC and the shift register is executed. The next D_{OUT} nibble is stored in $14. The carry is cleared so that on the next XAS instruction the shift clock will stop. The XAS instruction is executed and the final nibble of the LTC1090 D_{OUT} word containing the two LSBs and two zeroes is loaded into the ACC. G0 ($\overline{CS}$) is taken high and the A/D begins its next conversion cycle. The third D_{OUT} nibble is stored in location $15. The B register is then reinitialized so that when the loop is run again the data will always be stored in the same memory locations. The D_{OUT} data from the LTC1090 is now in a left justified format as shown in Figure 4.

LABEL	MNEMONIC		COMMENTS
	CLRA		MUST BE FIRST INSTRUCTION
	LBI	1,0	BR = 1 BD = 0 INITIALIZE B REG.
	STII	8	FIRST D_{IN} NIBBLE IN $10
	STII	E	SECOND D_{IN} NIBBLE IN $11
	STII	0	NULL DATA IN $12, B = $13
	LEI	C	SET EN TO (1100) BIN
LOOP	SC		CARRY SET
	LDD	1,0	LOAD FIRST D_{IN} NIBBLE IN ACC
	OGI	50	G0 ($\overline{CS}$) CLEARED
	XAS		ACC TO SHIFT REG. BEGIN SHIFT
	LDD	1,1	LOAD NEXT D_{IN} NIBBLE IN ACC
	NOP		TIMING
	XAS		NEXT NIBBLE, SHIFT CONTINUES
	XIS	0	FIRST NIBBLE D_{OUT} TO $13
	LDD	1,2	PUT NULL DATA IN ACC
	XAS		SHIFT CONTINUES, D_{OUT} TO ACC
	XIS	0	NEXT NIBBLE D_{OUT} TO $14
	RC		CLEAR CARRY
	CLRA		CLEAR ACC
	XAS		THIRD NIBBLE D_{OUT} TO ACC
	OGI	51	G0 ($\overline{CS}$) SET
	XIS	0	THIRD NIBBLE D_{OUT} TO $15
	LBI	1,3	SET B REG. FOR NEXT LOOP

Figure 5. COP402 Code

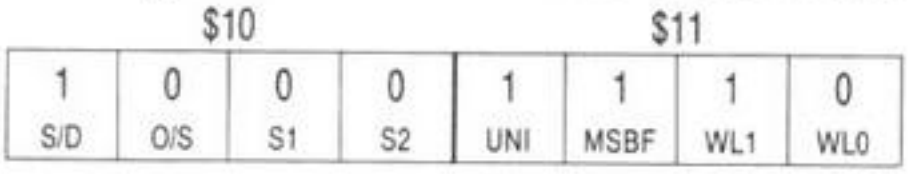


Figure 3. D_{IN} Word for LTC1090

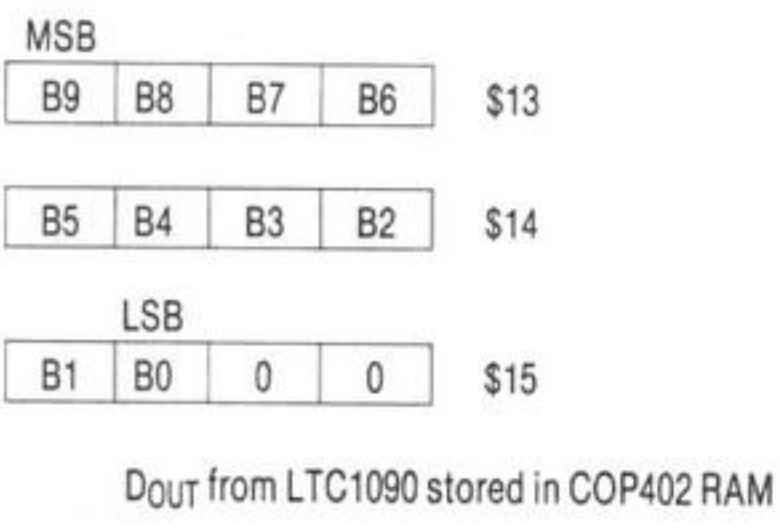


Figure 4. Memory Map

44 ACLK cycles must be allowed between transfers for the A/D to perform its next conversion. The instructions, after G0 is set, take enough time so that no additional delay is required by this program.

Summary

A four wire interface between the LTC1090 and the COP402N with a combined data conversion and transfer time of 100µs was demonstrated. The interface used the MICROWIRE serial port of the COP402N. The 10 data bits of the LTC1090 are shifted MSB first in three four bit transfers with the last two bits filled with zeroes. The data is stored left justified in the COP402N's internal RAM. The code demonstrated will work on any member of the COP400 family.

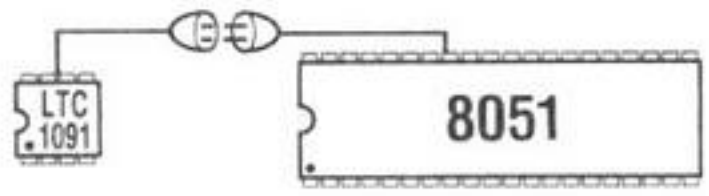
MICROWIRE is a trademark of National Semiconductor Corp.

LINEAR TECHNOLOGY

Application Note 26G

February 1988

Interfacing the LTC1091 to the 8051 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes the hardware and software required for communication between the LTC1091 10-bit data acquisition system and the MCS-51 family of microcontrollers (e.g., 8051). The three wire interface is capable of completing a 10-bit conversion and transferring the data to the 8051 in 57μs. Configuration of the 8051 and the LTC1091 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be discussed. Finally, a summary of results including data throughput rates will be provided.

Interface Details

The serial port of the 8051 does not support the synchronous, half duplex format used by the LTC1091. Therefore it is necessary for the user to construct a serial port using three lines from one of the parallel ports available on the 8051. The lines are set or cleared using the bit manipulation features of the 8051. This provides a very flexible serial port.

The data lines (D_{IN} and D_{OUT}) of the LTC1091 can be tied together as shown in Figure 1. This can be done because the I/O pins on the 8051 can be configured as either inputs or outputs independently of one another during the data transfer.

Hardware Description

The 8051 was simulated and the code for this interface was developed on an Intel ICE252 emulator.

Due to the weak pullups of the 8051, excess loading should be avoided when examining the output of the microcontroller.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer. The 8051 clock rate was 2MHz. The clock could be run as high as 12MHz yielding the minimum conversion and transfer time of 57μs.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

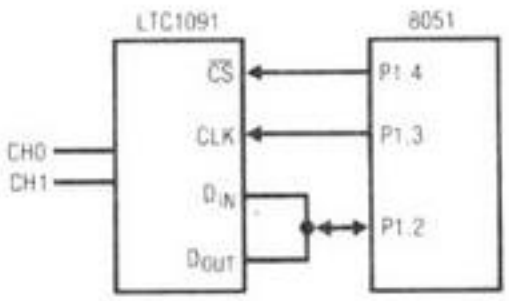


Figure 1. Schematic

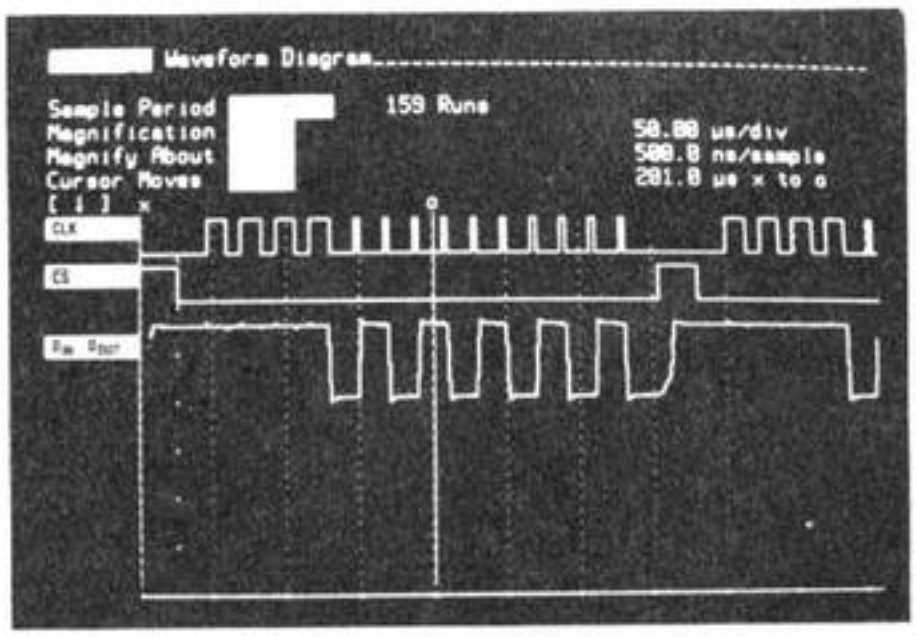


Figure 2. Timing Diagram

Software Description

The software simulates a serial port through bit manipulation instructions of the 8051.

$\overline{CS}$ is initialized by setting it high. Next the D_{IN} word for the LTC1091 is loaded into the accumulator. The LTC1091 is configured for MSB first and CH1 with respect to ground. Note that only the first four most significant bits of the byte loaded into the accumulator are sent to the LTC1091. The four LSBs of the D_{IN} word are don't cares.

$\overline{CS}$ is then cleared and a counter is set to four (the number of bits in the D_{IN} word). For each of the four D_{IN} bits the accumulator is shifted left with the MSB going into the carry bit. SCLK is cleared. The carry bit is output to D_{IN} and SCLK is set. The counter is decremented and checked and if all four bits have been output, P1.2 (D_{IN} and D_{OUT}) is set. This allows the pin to be used as an input now, to read the data in from the LTC1091.

SCLK is then cleared and the counter is set to nine (the first bit shifted out from the LTC1091 is a dummy, so nine shifts are required to fill the first byte.). The eight MSBs are read in the same manner as the D_{IN} word was output. The eight MSBs are stored in R2 and the remaining two LSBs are read into the accumulator. The LSBs are shifted into the MSBs of the accumulator and the remainder of the accumulator is masked to zeroes. The LSBs are then stored in R3 as shown in Figure 4. $\overline{CS}$ is then set and the process is completed. At this point the data is left justified.

1	1	1	1
Start	Sgl/Dif	O/S	MSBF

Figure 3. D_{IN} Word for LTC1091

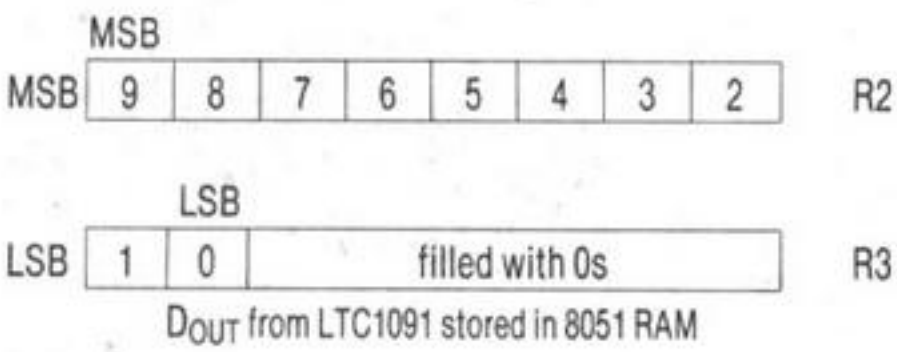


Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
BEGIN	SETB	P1.4	$\overline{CS}$ GOES HIGH
AGAIN	MOV	A, #FFH	D_{IN} WORD FOR LTC1091
	CLR	P1.4	$\overline{CS}$ GOES LOW
	MOV	R4, #04H	LOAD COUNTER
LOOP1	RLC	A	ROTATE D_{IN} BIT INTO CARRY
	CLR	P1.3	SCLK GOES LOW
	MOV	P1.2, C	OUTPUT D_{IN} BIT TO LTC1091
	SETB	P1.3	SCLK GOES HIGH
	DJNZ	R4, LOOP 1	NEXT BIT
	MOV	P1, #04H	BIT 2 BECOMES AN INPUT
	CLR	P1.3	SCLK GOES LOW
	MOV	R4, #09H	LOAD COUNTER
LOOP	MOV	C, P1.2	READ DATA BIT INTO CARRY
	RLC	A	ROTATE DATA BIT INTO CARRY
	SETB	P1.3	SCLK GOES HIGH
	CLR	P1.3	SCLK GOES LOW
	DJNZ	R4, LOOP	NEXT BIT
	MOV	R2, A	STORE MSBs IN R2
	MOV	C, P1.2	READ DATA BIT INTO CARRY
	SETB	P1.3	SCLK GOES HIGH
	CLR	P1.3	SCLK GOES LOW
	CLR	A	CLEAR ACC
	RLC	A	ROTATE DATA BIT TO ACC
	MOV	C, P1.2	READ DATA BIT INTO CARRY
	RRC	A	ROTATE RIGHT INTO ACC
	RRC	A	ROTATE RIGHT INTO ACC
	ANL	A, #C0H	MASK UNUSED BITS
	MOV	R3, A	STORE LSBs IN R3
	SETB	P1.4	$\overline{CS}$ GOES HIGH

Figure 5. Code

Summary

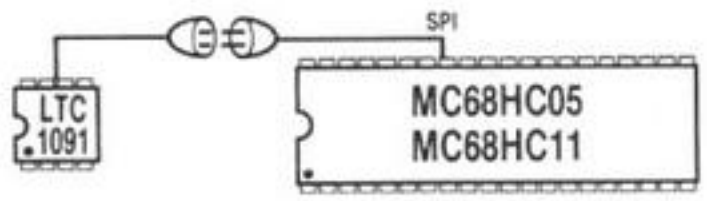
A three wire interface between the LTC1091 and the 8051 with a combined data conversion and transfer time of 57μs was demonstrated. The data is transferred MSB first and resides in two bytes of the 8051 RAM in a left justified format. The code shown applies to all MCS-51 family members. The same technique can be used on any parallel port processor which allows individual bits to be programmed as inputs or outputs.

LINEAR TECHNOLOGY

Application Note 26H

February 1988

Interfacing the LTC1091 to the MC68HC05 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1091 10-bit data acquisition system and the Motorola SPI family of single chip microcomputers (e.g., 68HC05). The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 68HC05 in 58μs. Configuration of the LTC1091 and the 68HC05 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given, including data throughput rates.

Interface Details

The LTC1091 has one clock line which controls the A/D conversion rate and the data shift rate. Data is transferred in a half duplex, synchronous format over D_{IN} and D_{OUT}.

The Motorola Serial Peripheral Interface (SPI) is a synchronous, full duplex, serial port built into the 68HC05 that allows the user to construct a simple communication path to the LTC1091. SPI provides clock, data in and data out lines that are compatible with the LTC1091. The only additional line required is one programmable output pin (C0) to control $\overline{CS}$ on the LTC1091. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The 68HC05 was emulated and the code for this interface was developed on a Motorola M68HC05 EVM.

$\overline{SS}$ (Pin 34) of the 68HC05 must be held high to enable the SPI properly for this interface.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using a 4MHz clock for the 68HC05.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

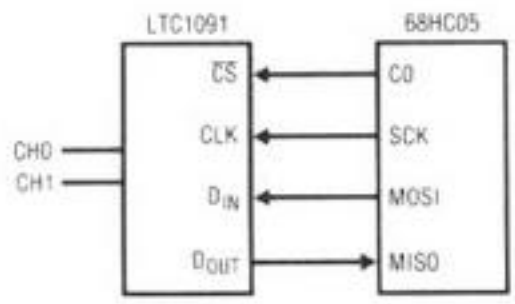


Figure 1. Schematic

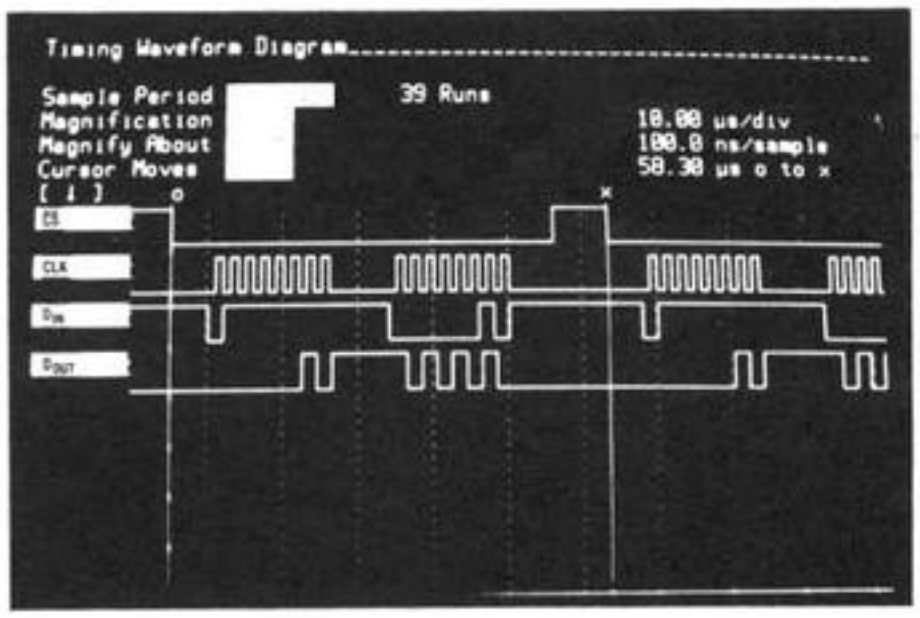


Figure 2. Timing Diagram

Software Description

The software configures and controls the SPI of the 68HC05. Additionally, the software manipulates C0 ($\overline{CS}$ of the LTC1091).

The code first configures the Serial Peripheral Control Register (SPCR) of the SPI. The SPI interrupt is disabled. The SPI outputs are enabled. The SPI is configured as a master. Finally, the SPI clock is set to normally low, for data transfer on the rising edge and for a frequency equal to one fourth the internal processor clock (one eighth the crystal frequency).

Port C is configured as all outputs by placing ones in the data direction register of port C. A D_{IN} word that configures the LTC1091 for CH1 with respect to ground and MSB first is stored in $50. Figure 3 shows how the D_{IN} word is composed. Leading zeroes in the D_{IN} word are ignored. This makes it easy to position the D_{OUT} word on exact byte boundaries so that shifting the data to right justify it is not necessary.

C0 is made to go low. D_{IN} for the LTC1091 is loaded into the SPI data register. Storing D_{IN} in the data register causes the transfer to begin. The status register of the SPI is tested until the SPIF bit is set which indicates the transfer is finished. Reading the SPI status register clears the SPIF bit and allows the data register to be read, which is the next step. The first eight bits containing the MSBs from the LTC1091 are then stored in $60 of the 68HC05 as shown in Figure 4. The LSBs are transferred in the same manner and stored in $61 of the 68HC05.

0	1	1	1	1	1	1	1
Ignore	Start	S/D	O/S	MSBF	don't care	don't care	don't care

Figure 3. 4-Bit D_{IN} Word for LTC1091 in $50

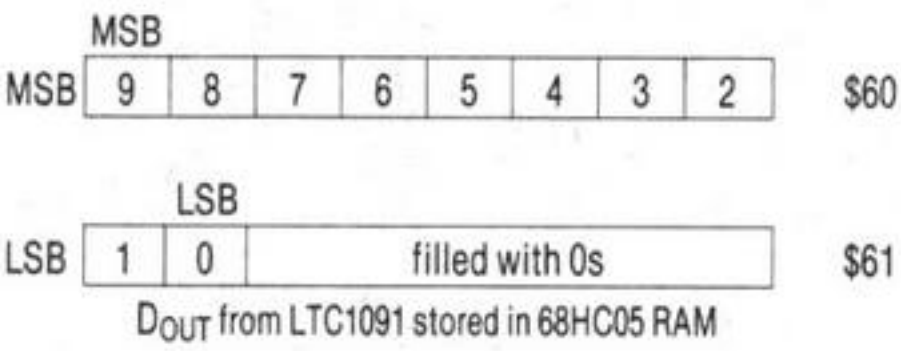


Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
	LDA	#$51	DATA FOR SPCR
	STA	$0A	LOAD DATA INTO SPCR
	LDA	#$FF	DATA FOR DDR
	STA	$06	CONFIGURE PORT C DDR
	LDA	#$7F	D_{IN} WORD FOR LTC1091
	STA	$50	PUT D_{IN} WORD IN $50
START	BCLR	0, $02	C0 ($\overline{CS}$) GOES LOW
	LDA	$50	PUT D_{IN} WORD IN ACC
	STA	$0C	START TRANSFER
TEST	TST	$0B	TEST IF DONE
	BPL	TEST	IF NOT TRY AGAIN
	LDA	$0C	LOAD MSBs IN ACC
	STA	$0C	START NEXT TRANSFER
	AND	#$03	MASK UNUSED BITS
	STA	$60	STORE MSBs IN $60
TEST1	TST	$0B	TEST IF DONE
	BPL	TEST1	IF NOT TRY AGAIN
	BSET	0, $02	C0 ($\overline{CS}$) GOES HIGH
	LDA	$0C	PUT LSBs IN ACC
	STA	$61	PUT LSBs IN $61

Figure 5. 68HC05 Code

The code was written for the 68HC05. By changing the addresses of the special function registers however, the code should run on all of Motorola's SPI processors including the 68HC11.

Summary

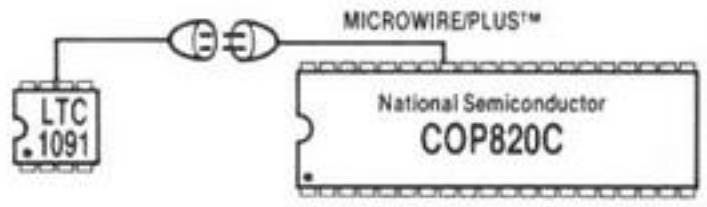
A four wire interface between the LTC1091 and the 68HC05 with a combined data conversion and transfer time of 58μs was demonstrated. The interface used the serial (SPI) port of the 68HC05. The 10 data bits of the LTC1091 are shifted MSB first in two eight bit transfers. The data is stored left justified in the 68HC05's internal RAM.

LINEAR TECHNOLOGY

Application Note 26I

February 1988

Interfacing the LTC1091 to the COP820C MCU



Guy Hoover
William Rempfer

Introduction

This application note describes the hardware and software required for communication between the LTC1091 10-bit data acquisition system and the National Semiconductor COP800 microcontroller family which uses the MICROWIRE/PLUS serial interface. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data in 45μs. Configuration of the LTC1091 and the COP820C will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a half duplex, synchronous, format over D_{IN} and D_{OUT}.

The National Semiconductor MICROWIRE/PLUS is a synchronous, full duplex, serial port built into the COP800 family that allows easy interface to the LTC1091. MICROWIRE/PLUS provides clock, data in and data out lines that are compatible with the LTC1091. One additional line (G1) is required to control the $\overline{CS}$ pin on the LTC1091. The schematic of Figure 1 shows how the two devices are connected.

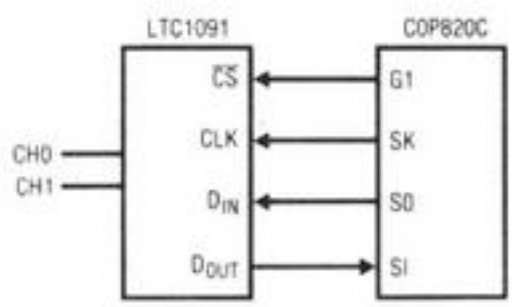


Figure 1. Schematic

Hardware Description

The actual interface was done using the COP820C, a member of the COP800 family. All code shown here should work with any of the COP800 family.

The code for this interface was developed on a COP820 evaluation board operated in the emulation mode.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer. A 5MHz COP820C clock (2μs instruction cycle) was used. By operating the MCU with a 1μs instruction cycle (high speed option) the minimum conversion and transfer time of 45μs is achieved.

The analog section of the schematic in Figure 1 is omitted for clarity. for a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

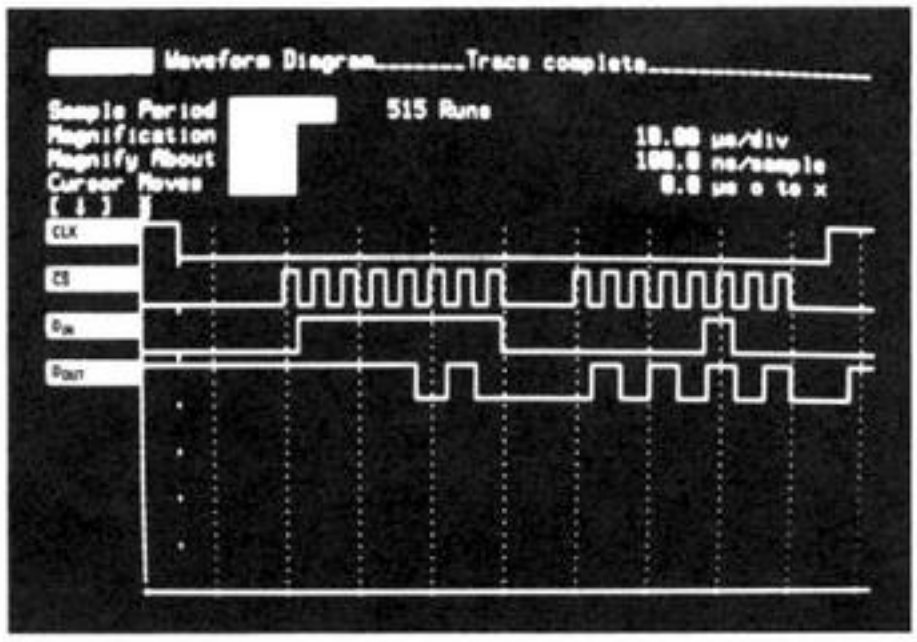


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE/PLUS serial interface of the COP820C. Additionally, the software manipulates G1 ($\overline{CS}$ of the LTC1091).

The code first loads the D_{IN} word of the LTC1091 into the memory address $F0. The D_{IN} word ($7F) contains a leading zero which is ignored, followed by a start bit. The next two bits configure the LTC1091 for CH1 with respect to ground. The fifth bit configures the A/D for MSB first mode and the remaining three LSBs are ignored as shown in Figure 3.

Next port G is configured such that pin G1 is an output and the MICROWIRE/PLUS serial port is a master. The control register is configured to enable SO and SK. Also the SK divide by is set up in such a way that the SK clock rate is equal to the crystal frequency divided by 20. The address of the port G data register is put into the B register so that the individual bits of port G can be manipulated. G1 ($\overline{CS}$) is then initialized by setting it high.

The D_{IN} word is then loaded into the accumulator. G1 is cleared and the LTC1091 D_{IN} word is loaded into the MICROWIRE™ shift register. The busy bit is set which begins the data transfer. 16 NOPs are used as a timer to allow the transfer to be completed. After the transfer is complete the D_{OUT} information is loaded into the accumulator and the next eight bits start to shift. The six MSBs in the accumulator are set to zeroes and the result is stored in $F3. Nine NOPs are then used to wait until the second eight bits have been shifted. G1 ($\overline{CS}$) is then set and the

0	1	1	1	1	1	1	1
Ignore	Start	S/D	O/S	MSBF	don't care	don't care	don't care

Figure 3. 4-Bit D_{IN} Word for LTC1091 in $F0

MSB

filled with zeroes						9	8	F3

LSB

7	6	5	4	3	2	1	0	F4

D_{OUT} from LTC1091 stored in COP820C RAM

Figure 4. Memory Map

LABEL	MNEMONIC	COMMENTS
	LD (F0)–$7F	LOAD D_{IN} WORD INTO $F0
	LD (D5)–$32	G1 IS OUT, MICROWIRE MASTER
	LD (EE)–$08	ENABLE SO, SK
	LD (B)–$D4	PORT G DATA ADDR INTO B
	SBIT 1	G1 SET (CS GOES HIGH)
LOOP	LD (A)–(F0)	PUT LTC1091 D_{IN} WORD IN ACC
	RBIT 1	G1 CLEARED (CS GOES LOW)
	X (A)– –(E9)	D_{IN} IN MICROWIRE SHIFT REG.
	LD (B)–$EF	PUT PSW REG. ADDR. INTO B
	SBIT 2	BUSY BIT SET TRANSFER START
	NOP	16 NOPs FOR TIMING
	X (A)– –(E9)	LOAD D_{OUT} INTO ACC
	SBIT 2	BUSY BIT SET TRANSFER START
	AND #03	MASK OUT UNUSED BITS
	X (A)– –(F3)	LOAD D_{OUT} INTO ADDR F3
	NOP	
	LD (B)–$D4	PORT G DATA REG. ADDR. IN B
	SBIT 1	G1 SET (CS GOES HIGH)
	X (A)– –(E9)	LOAD D_{OUT} INTO ACC
	X (A)– –(F4)	LOAD LSB INTO ADDR. F4

Figure 5. Code

data is loaded into the accumulator. The LSBs are then loaded into memory location $F4. The data at this point is right justified. With the appropriate shift routine the data can be easily left justified.

Summary

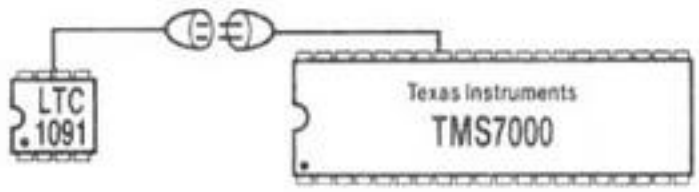
A four wire interface between the LTC1091 and the COP820C with a combined data conversion and transfer time of 45μs was demonstrated. The interface used the MICROWIRE/PLUS serial port of the COP820C. The 10 data bits of the LTC1091 are shifted MSB first in two eight bit transfers. The data is stored right justified in the COP820C's internal RAM.

LINEAR TECHNOLOGY

Application Note 26J

January 1988

Interfacing the LTC1091 to the TMS7742 MCU



Guy Hoover

Introduction

This application note describes an interface between the LTC1091 10-bit data acquisition system and the TMS7000 family of microcomputers (e.g., TMS7742). The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the TMS7742 in 99μs. Configuration of the LTC1091 and the TMS7742 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a synchronous, half duplex format over D_{IN} and D_{OUT}.

The TMS7742 contains a synchronous, full duplex, serial port that allows the user to construct a simple communication path to the LTC1091. The serial port provides clock, transmit and receive lines that are compatible with the LTC1091. The only additional line required is one programmable output pin (A0) to control $\overline{CS}$ on the LTC1091.

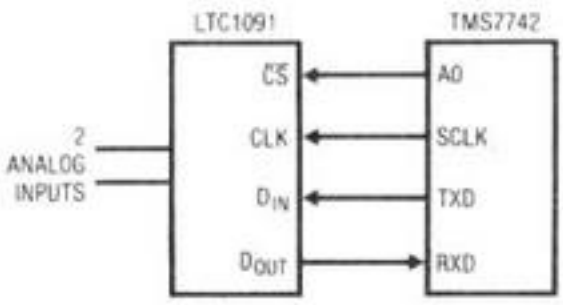


Figure 1. Schematic

The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The TMS7742 was chosen because it contains 4k of EPROM which can be programmed using a standard EPROM programmer. Any member of the TMS7000 family which contains a serial port should be able to use this code with only modifications to the peripheral register numbers.

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. The TMS7742 clock was 5 MHz.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

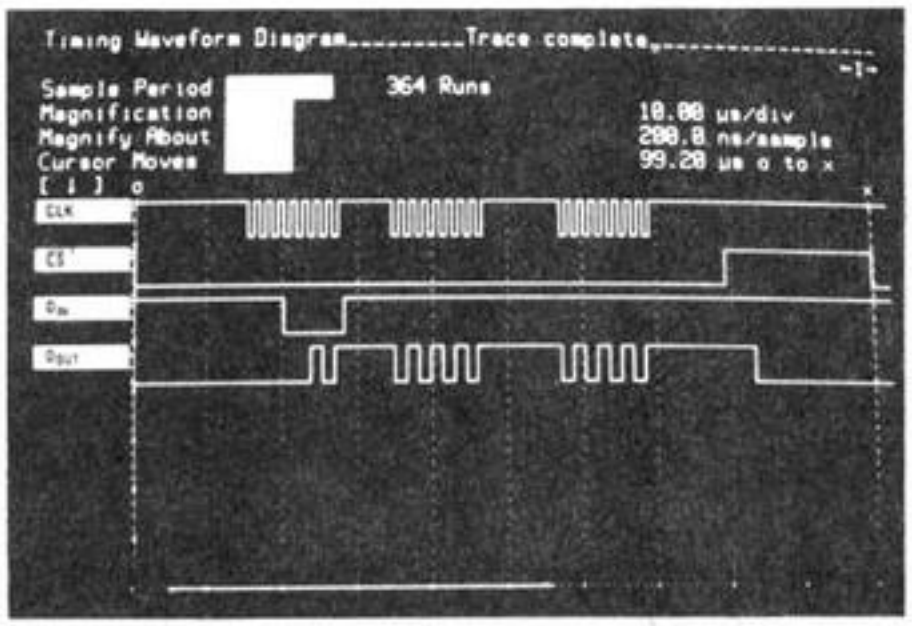


Figure 2. Timing Diagram

Software Description

The software configures and controls the serial port of the TMS7742. Additionally, the software manipulates A0 ($\overline{CS}$ of the LTC1091).

The code first disables all interrupts and initializes the stack. The data direction register for the A port then sets A5 (RXD) as an input and all other bits including A0 ($\overline{CS}$) as outputs. Next the serial port is configured. Tx is enabled, the serial port is reset, and the SMODE register is configured for 8 bits, no parity, and one stop bit. The SCLK rate is set to the processor clock frequency divided by 4.

The SCLK is turned off and A0 ($\overline{CS}$) is cleared. The D_{IN} word of the LTC1091 is loaded into the TXBUF. This D_{IN} word (07) configures the LTC1091 for CH1 with respect to ground and LSB first. Examine Figure 3 to see how this is constructed keeping in mind that the TMS7742 transmits data LSB first. The serial port and SCLK are turned on and the data is shifted while the processor idles in a loop. The first eight bits contain the D_{IN} word and the first three MSBs of the D_{OUT} word. (The LTC1091 clocks out the data MSB first and then LSB first when in the LSB first mode.) The serial port is turned on again and the next eight bits containing the rest of the MSB first data and the first two bits of the LSB first data are shifted while the processor idles in a loop. The data containing the LSBs is then placed in the B register. The procedure is repeated for the next eight bits which contain the MSBs and the result is placed in the A register. A0 ($\overline{CS}$) is then set. The data in the B register is stored in R5. If desired the lowest six bits of the B register can be cleared by adding them with $C0. The data in the A register is stored in R6. The data is now stored left justified as shown in Figure 4.

0	0	0	0	0 MSBF	1 O/S	1 S/D	1 START	P23

Figure 3. D_{IN} Word for LTC1091 Stored in TMS7742 TXBUF

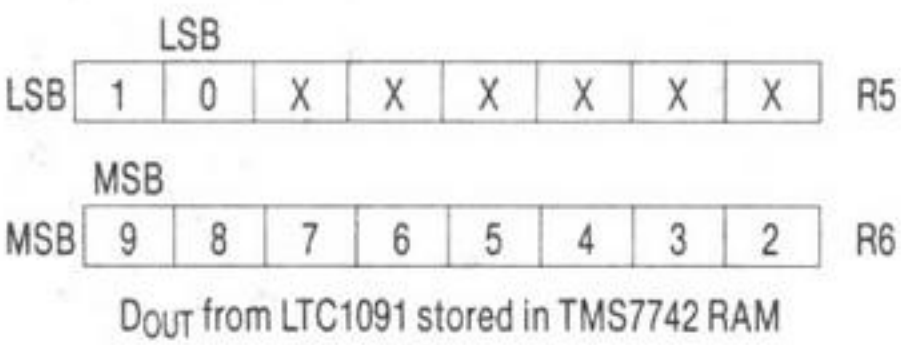


Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
START	DINT		DISABLES ALL INTERRUPTS
	MOVP	%>2A,P0	DISABLE INTERRUPT FLAGS
	MOVP	%>02,P16	DISABLE INTERRUPT FLAGS
	MOV	%>60,B	ADDRESS OF STACK
	LDSP		PUT ADDRESS INTO POINTER
	MOVP	%>DF,P5	CONFIGURE PORT A
	MOVP	%>08,P6	ENABLE Tx BY SETTING B3 = 1
	MOVP	%>00,P17	P17 POINTS TO SCTL0
	MOVP	%>40,P17	RESET THE SERIAL PORT
	MOVP	%>0C,P17	CONFIGURE THE SERIAL PORT
	MOVP	%>00,P21	TURN START BIT OFF
	MOVP	%>00,P17	ENABLE THE SERIAL PORT
	MOVP	%>00,P20	SET SCLK RATE (TIMER 3)
LOOP	MOVP	%>40,P21	SCLK OFF
	ANDP	%>FE,P4	A0 CLEARED ($\overline{CS}$ GOES LOW)
	MOVP	%>07,P23	PUT LTC1091 D_{IN} INTO TXBUF
	MOVP	%>17,P17	ENABLE SERIAL PORT
	MOVP	%>C0,P21	SCLK ON (TRANSFER BEGINS)
	MOVP	%>14,P17	TXEN GOES LOW
	MOV	%>02,A	LOAD COUNTER
WAIT	DJNZ	A,WAIT	LOOP WHILE SHIFT OCCURS
	NOP		MORE DELAY
	MOVP	%>17,P17	ENABLE SERIAL PORT
	MOVP	%>14,P17	TXEN GOES LOW
	MOV	%>02,A	LOAD COUNTER
WAIT1	DJNZ	A,WAIT1	LOOP WHILE SHIFT OCCURS
	NOP		DELAY
	MOVP	P22,B	PUT D_{OUT} FROM LTC1091 IN B
	MOVP	%>17,P17	ENABLE SERIAL PORT
	MOVP	%>14,P17	TXEN GOES LOW
	MOV	%>02,A	LOAD COUNTER
WAIT2	DJNZ	A,WAIT2	LOOP WHILE SHIFT OCCURS
	NOP		DELAY
	MOVP	P22,A	PUT D_{OUT} FROM LTC1091 IN A
	ORP	%>01,P4	A0 SET ($\overline{CS}$ GOES HIGH)
	MOV	B,R5	PUT FIRST 2 LSBs IN R5
	MOV	A,R6	PUT MSBs IN R6

Figure 5. TMS7742 Code

Summary

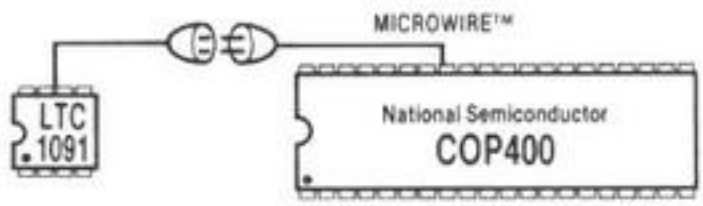
A four wire interface between the LTC1091 and the TMS7742 with a combined data conversion and transfer time of 99μs was demonstrated. The interface used the serial port of the TMS7742. Because the serial port transfers data LSB first, care must be taken to properly construct the D_{IN} word so that the bits are transmitted in the proper order to the LTC1091. The 10 data bits of the LTC1091 are shifted LSB first in three eight bit transfers. The data is stored left justified in the TMS7742's internal RAM.

LINEAR TECHNOLOGY

Application Note 26K

January 1988

Interfacing the LTC1091 to the COP402N MCU



Guy Hoover

Introduction

This application note describes the hardware and software required for communication between the LTC1091 10-bit data acquisition system and the National Semiconductor COP400 microcontroller family which uses the MICROWIRE serial interface. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data in 116μs. Configuration of the LTC1091 and the COP402N will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a half duplex format over D_{IN} and D_{OUT}.

The National Semiconductor MICROWIRE interface is a synchronous, full duplex, serial port built into the COP400 family that allows the user to easily interface to the LTC1091. MICROWIRE provides clock, data in and data out lines that are compatible with the LTC1091. One additional line (G0) is required to control the $\overline{CS}$ pin on the LTC1091. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The actual interface will involve using the COP402N, a member of the COP400 family. All code shown here should work with any of the COP400 family.

The code for this interface was developed on a COP400 evaluation board which allows an external EPROM to be used in place of the internal processor ROM.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer. The COP402N clock was 4MHz.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

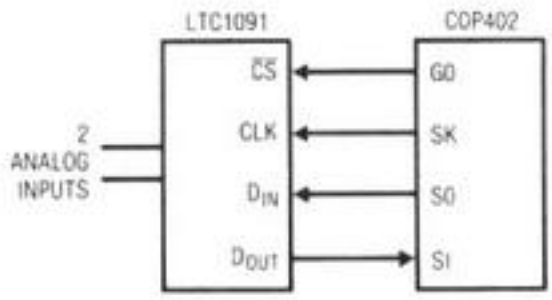


Figure 1. Schematic

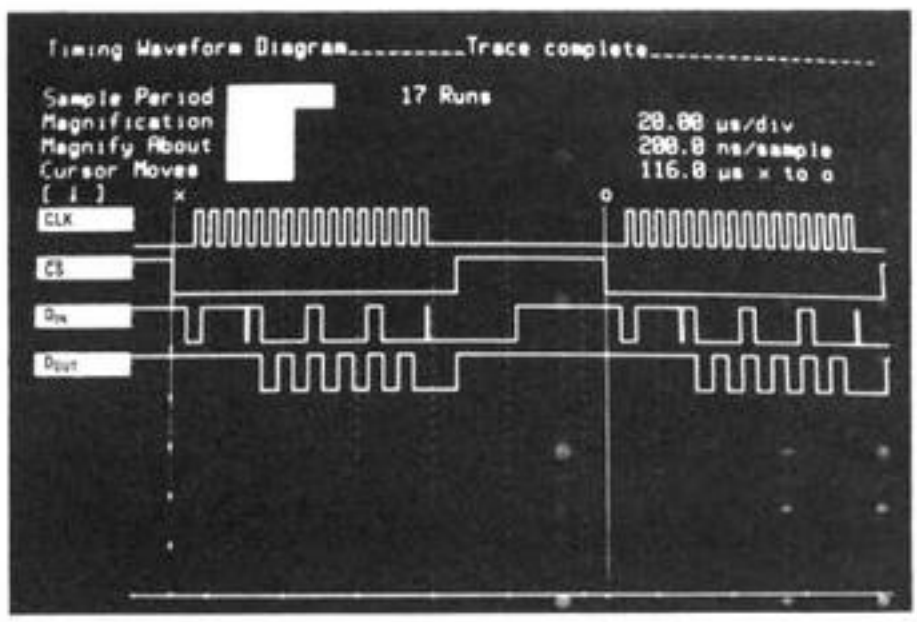


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE serial interface of the COP402N. Additionally, the software manipulates G0 ($\overline{CS}$ of the LTC1091).

The code first initializes the B register and then loads the LTC1091 D_{IN} word into the RAM of the COP402 one nibble at a time. As shown in Figure 3 the D_{IN} word configures the LTC1091 for CH1 with respect to GND and MSB first. SO is configured as an output. The carry is set so that when an XAS instruction is generated the shift clock (SK) will begin clocking data.

The first nibble of the D_{IN} word is loaded into the ACC and G0 ($\overline{CS}$) is cleared. The D_{IN} nibble is loaded into the shift register and the data begins to shift. The second nibble of the D_{IN} word is loaded into the ACC. One NOP is allowed for timing and then the contents of the ACC are swapped with those of the shift register. The transfer continues. One NOP is allowed for timing. The second D_{IN} nibble is loaded into the ACC again and the contents of the ACC are swapped with those of the shift register. The MSBs of the LTC1091 D_{OUT} word are now in the ACC. This data is then stored in memory location $13. The ACC is loaded with the second D_{IN} nibble from RAM and another swap

	$11					$12	
0	1 START	1 S/D	1 O/S	1 MSBF	0	0	0

Figure 3. D_{IN} Word for LTC1090

	MSB			
X	0	B9	B8	$13
B7	B6	B5	B4	$14
		LSB		
B3	B2	B1	B0	$15

D_{OUT} from LTC1091 stored in COP402 RAM

Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
	CLRA		MUST BE FIRST INSTRUCTION
	LBI	1,1	BR = 1 BD = 1 INITIALIZE B REG.
	STII	7	FIRST D_{IN} NIBBLE IN $11
	STII	8	SECOND D_{IN} NIBBLE IN $12
	LEI	8	SET EN TO (1000) BIN
	SC		CARRY SET
	LDD	1,1	LOAD FIRST D_{IN} NIBBLE IN ACC
	OGI	50	G0 ($\overline{CS}$) CLEARED
	XAS		ACC TO SHIFT REG. BEGIN SHIFT
	LDD	1,2	LOAD NEXT D_{IN} NIBBLE IN ACC
	NOP		TIMING
	XAS		NEXT NIBBLE, SHIFT CONTINUES
	NOP		TIMING
	LDD	1,2	LOAD NULL DATA IN ACC
	XAS		NEXT NIBBLE, SHIFT CONTINUES
	XIS	0	FIRST NIBBLE D_{OUT} TO $13
	LDD	1,2	LOAD NULL DATA IN ACC
	XAS		SHIFT CONTINUES, D_{OUT} → ACC
	XIS	0	NEXT NIBBLE D_{OUT} TO $14
	RC		CLEAR CARRY
	CLRA		CLEAR ACC
	XAS		THIRD NIBBLE D_{OUT} TO ACC
	OGI	51	G0 ($\overline{CS}$) SET
	XIS	0	THIRD NIBBLE D_{OUT} TO $15
	LBI	1,3	SET B REG. FOR NEXT LOOP

Figure 5. COP402 Code

between the ACC and the shift register is executed. The next D_{OUT} nibble is stored in $14. The carry is cleared so that on the next XAS instruction the shift clock will stop. The XAS instruction is executed and the final nibble of the LTC1091 D_{OUT} word containing the LSBs is loaded into the ACC. G0 ($\overline{CS}$) is taken high. The third D_{OUT} nibble is stored in location $15. The B register is then reinitialized so that when the loop is run again the data will always be stored in the same memory locations. The D_{OUT} data from the LTC1091 is now in a right justified format as shown in Figure 4.

Summary

A four wire interface between the LTC1091 and the COP402N with a combined data conversion and transfer time of 116μs was demonstrated. The interface used the MICROWIRE serial port of the COP402N. The 10 data bits of the LTC1091 are shifted MSB first in four four bit transfers. The data is stored right justified in the COP402N's internal RAM. The code demonstrated will work on any member of the COP400 family.

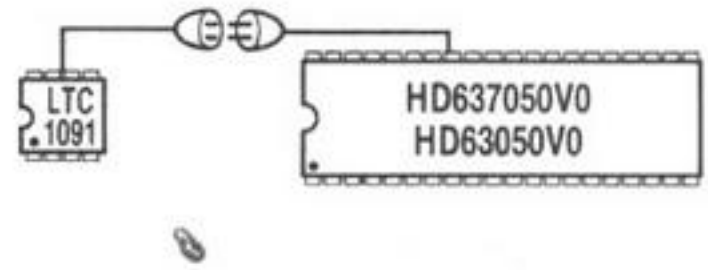
MICROWIRE is a trademark of National Semiconductor Corp.

LINEAR TECHNOLOGY

Application Note 26L

January 1988

Interfacing the LTC1091 to the HD637050 MCU



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1091 10-bit data acquisition system and the Hitachi 63705 microcomputer. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 63705 in 84μs. Configuration of the LTC1091 and the 63705 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a half duplex synchronous format over D_{IN} and D_{OUT}.

The Hitachi Serial Communication Interface (SCI) is a synchronous, full duplex, serial port built into the 63705 that allows the user to construct a simple communication path to the LTC1091. SCI provides clock, transmit and receive lines that are compatible with the LTC1091. The only additional line required is one programmable output pin (C0) to control $\overline{CS}$ on the LTC1091. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The 63705VOC was chosen for this application because it contains 4k bytes of EPROM which can be programmed by a 27256 EPROM programmer. The code shown will work on the 6305VO without modification.

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. The 63705 clock was 4 MHz.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

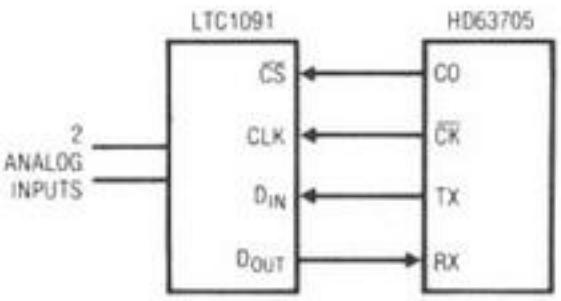


Figure 1. Schematic

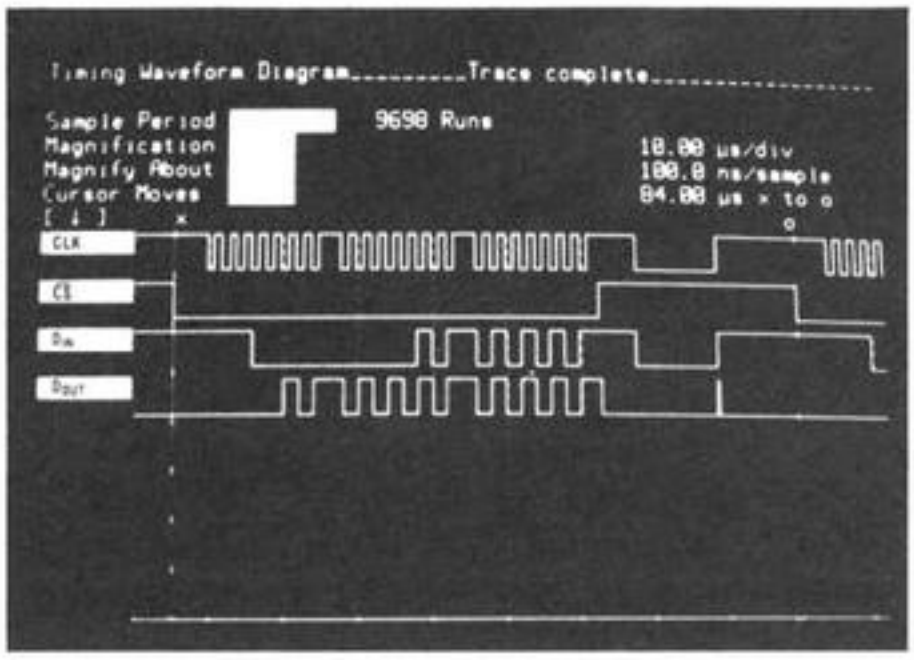


Figure 2. Timing Diagram

Software Description

The software configures and controls the SCI of the 63705. Additionally, the software manipulates C0 ($\overline{CS}$ of the LTC1091).

The code first configures the SCI control register (SCR). D3 and D4 are set as the SCI output and input respectively. D5 is selected as a clock output with a frequency one eighth the crystal frequency. Next, the SCI status register (SSR) is configured so that the interrupts are disabled. Data is transmitted on the falling edge of the clock and received on the rising edge of the clock.

Bit 0 of Port C is configured as an output by setting the first bit of the data direction register (address $06) to one. A D_{IN} word that configures the LTC1091 for CH1 with respect to ground and LSB first is stored in $50. Figure 3 shows how the D_{IN} word is composed. Note, that for LSB first format the D_{IN} word must be constructed opposite from MSB first format.

C0 is made to go low. D_{IN} for the LTC1091 is loaded into the SCI data register (SDR). Storing D_{IN} in the SDR causes the transfer to begin. When LSB first is selected, the LTC1091 first clocks out the data MSB first and then clocks out the data LSB first. After waiting for the first eight bits to be transferred the data containing the MSBs is loaded into the ACC. This starts the next transfer. The next eight bits are then shifted out and the first two LSBs from the LTC1091 are loaded into the accumulator and then stored in $61 of the 63705 RAM. The act of reading the

0	0	0	0	0 MSBF	1 O/S	1 S/D	1 START	$50

Figure 3. D_{IN} Word for LTC1091 Stored in 63705 RAM

LSB

LSB	1	0	X	X	X	X	X	X	$61

MSB

MSB	9	8	7	6	5	4	3	2	$62

D_{OUT} from LTC1091 stored in 63705 RAM

Figure 4. Memory Map

LABEL	MNEMONIC		COMMENTS
	LDA	#$E1	CONFIGURATION DATA FOR SCR
	STA	$10	LOAD DATA INTO SCR ($10)
	LDA	#$30	CONFIGURATION DATA FOR SSR
	STA	$11	LOAD DATA INTO SSR ($11)
	LDA	#$01	CONFIG. DATA FOR PORT C DDR
	STA	$06	LOAD DATA INTO PORT C DDR
	LDA	#$07	LOAD LTC1091 D_{IN} DATA INTO ACC
	STA	$50	LOAD LTC1091 D_{IN} DATA INTO $50
LOOP	LDA	$50	LOAD D_{IN} INTO ACC FROM $50
	BCLR	0,$02	C0 GOES LOW ($\overline{CS}$ GOES LOW)
	STA	$12	LOAD D_{IN} INTO SDR. START SCK
	BCLR	1,$06	FOR TIMING
	BCLR	1,$06	FOR TIMING
	BCLR	1,$06	FOR TIMING
	LDA	$12	LOAD DATA START NEXT CYCLE
	BCLR	1,$06	FOR TIMING
	BCLR	1,$06	FOR TIMING
	BCLR	1,$06	FOR TIMING
	LDA	$12	LOAD LSBs START NEXT CYCLE
	STA	$61	STORE LSBs IN $61
	BCLR	1,$06	FOR TIMING
	BCLR	1,$06	FOR TIMING
	BSET	0,$02	C0 GOES HIGH. ($\overline{CS}$ GOES HIGH)
	LDA	#$00	CONFIGURATION DATA FOR SCR
	STA	$10	DISABLE SCI
	LDA	$12	LOAD MSBs
	STA	$62	STORE MSBs IN $62
	LDA	#$E1	CONFIGURATION DATA FOR SCR
	STA	$10	TURN SCI ON

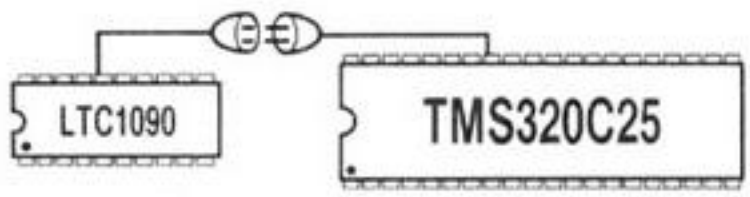
Figure 5. 63705 Code

LSBs into the ACC causes the next SCI transfer to begin. After the next eight bits are transferred, then C0 is set and the SCI port is disabled. The MSBs from the LTC1091 are loaded into the ACC and then stored in $62 of the 63705 RAM. The SCI port is then enabled. The data at this point is left justified as shown in Figure 4.

Summary

A four wire interface between the LTC1091 and the Hitachi 63705 with a combined data conversion and transfer time of 84μs was demonstrated. The interface used the serial (SCI) port of the 63705. Because the SCI port transfers data LSB first, care must be taken to properly construct the D_{IN} word so that the bits are transmitted in the proper order to the LTC1091. The 10 data bits of the LTC1091 are shifted MSB first and then LSB first in three eight bit transfers. The data is stored left justified in the 63705's internal RAM.

Interfacing the LTC1090 to the TMS320C25 DSP



Guy Hoover

Introduction

This application note describes the hardware and software required for communication between the LTC1090 10-bit data acquisition system and the TMS320C25 digital signal processor (DSP). In particular two interfaces will be demonstrated. The first interface requires only one inverter in addition to the LTC1090 and the TMS320C25. The second interface, which is optimized for speed of transfer and minimum processor supervision, can complete a conversion and shift the data in only 32μs. Configuration of the TMS320C25 and LTC1090 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be presented. Finally, a summary of results including data throughput rates will be provided.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a synchronous, full duplex format over D_{IN} and D_{OUT}.

The serial port of the TMS320C25 is not directly compatible with that of the LTC1090. The data shift clock lines (CLKR, CLKX) are inputs only. Therefore the data shift clock must be externally generated. Inverting the shift clock is also necessary because the LTC1090 and the TMS320C25 clock data on opposite edges. This prevents a race condition. The framing pulse width of the TMS320C25 is not long enough to be used as a chip select for the A/D directly. It must somehow be stretched. This can be done with additional hardware or through software control of the shift clock which controls the framing pulse timing.

The schematic of Figure 1 has the shift clock generated by the XF pin of the TMS320C25. The signal is inverted with a 74HC04 and fed into the SCLK pin of the LTC1090. The framing pulse is properly generated by delaying the SCLK edge which causes FSX to fall until the A/D conversion is finished. This method results in the simplest hardware configuration but has the drawbacks of requiring more processor supervision and a slower data shift time.

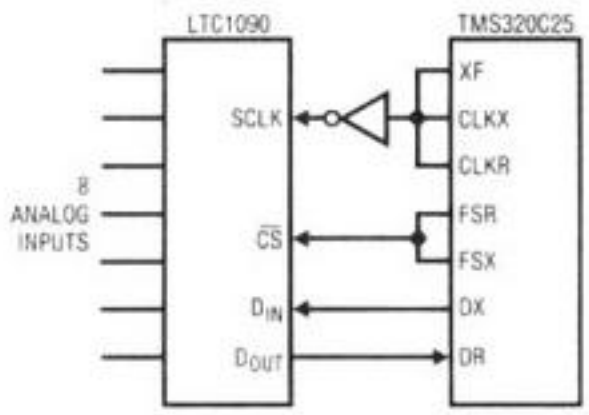


Figure 1. Circuit 1: Minimum Hardware Interface

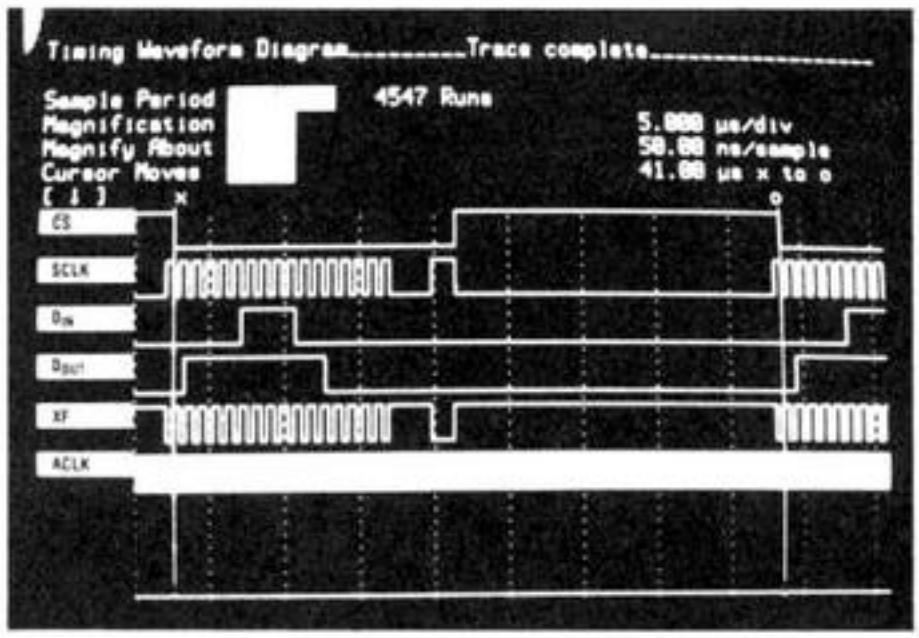


Figure 2. Timing for Circuit 1 Shows 41μs Throughput Rate

The schematic of Figure 7 has the shift clock generated by dividing down the processor clock with a 74HC163 counter. The signal is inverted with a 74HC04 and fed into the CLKX and CLKR pins of the TMS320C25. The $\overline{CS}$ signal is generated with another 74HC163 and two inverters. The obvious disadvantage of this method is that it requires considerably more hardware to implement but it provides a faster data shift rate and requires less processor supervision.

Hardware Description

The DSP was emulated and the code for this interface was developed on a TMS320C25 Software Development System (SWDS). The SWDS requires a PC compatible computer to run.

The timing diagram of Figure 2 was obtained using the circuit of Figure 1. The timing diagram of Figure 8 was obtained using the circuit of Figure 7. Both pictures were taken with an HP1631 logic analyzer. ACLK was 2.5MHz and SCLK was 1.25MHz. The TMS320C25 clock rate was 40MHz.

The analog sections of the schematics of Figure 2 and Figure 7 are omitted for clarity. For a complete discussion of analog considerations involved in using the LTC1090 please see the data sheet.

B14							B7
0	0	0	0	1	1	1	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 3. D_{IN} Word in ACC of TMS320C25 for Circuit 1

MSB									LSB		
9	8	7	6	5	4	3	2	1	0	filled with 0s	>200

D_{OUT} from LTC1090 stored in TMS320C25 RAM

Figure 4. Memory Map for Circuit 1

```
LABEL    MNEMONIC          COMMENTS
         AORG   0          ON RST CODE STARTS AT 0
         B      INIT       BRANCH TO INIT ROUTINE

         AORG   >26        ADDRESS OF RINT VECTOR
         B      RINT       BRANCH TO RINT ROUTINE

         AORG   >32        MAIN PROGRAM ADDRESS
INIT     DINT               DISABLE INTERRUPTS
         LDPK   >0          DATA PAGE POINTER IS 0
         LARP   >1          AUX. REG. POINTER IS 1
         LRLK   AR1,>200    AUX. REG. 1 = >200
         LACK   >10         CONFIG. WORD FOR IMR
         SACL   >4          PUT CONFIG. WORD IN IMR
         STXM               CONFIGURE FSX AS OUTPUT
         FORT   0           SET SERIAL PORT TO 16 BITS

TXRX     RXF                RESET XF
         STC                SET TC BIT (FIRST TIME FLG)
         LACK   >F0         LOAD D_IN WORD INTO ACC
         SFSM               FSX STARTS ON XSR LOAD
         RPTK   2           REPEAT 3 TIMES
         SFL                SHIFT D_IN TO LEFT
         SACL   >1          D_IN PUT IN TX REGISTER
         EINT               ENABLE INTERRUPTS

TIMER    RXF                RESET XF (SCLK)
         RPTK   2           REPEAT 3 TIMES
         NOP                TIMING
         SXF                SET XF (SCLK)
         BBZ    TIMER       SCLKS UNTIL RINT
         RPTK   >D0         DELAY FOR CONVERSION
         NOP
         RTC                RESET TC (NOT FIRST TIME)
         B      TIMER       NEXT SCLK

RINT     ZALS   >0          STORE D_OUT WORD IN ACC
         SFL                SHIFT ACC LEFT 1 BIT
         SACL   *,0         STORE ACC IN >200
         B      TXRX        GO TO TXRX ROUTINE
         END
```

Figure 5. TMS320C25 Code for Circuit 1

Software Description

The software configures and controls the serial port of the TMS320C25. Additionally, the software generates a delay during which time the LTC1090 performs a conversion.

The first 13 lines of code are the same for circuit 1 and circuit 2. The code first sets up the interrupt and reset vectors. On reset the TMS320C25 starts executing code at the label INIT. Upon completion of a 16-bit data transfer, an interrupt is generated and the DSP will begin executing code at the label RINT.

Next, the code initializes registers in the TMS320C25 that will be used in the transfer routine. The interrupts are temporarily disabled. The data memory page pointer register is set to zero. The auxiliary register pointer is loaded with one and auxiliary register one is loaded with the value 200 hexadecimal. This is the data memory location where the data from the LTC1090 will be stored. The interrupt mask register (IMR) is configured to recognize the RINT interrupt, which is generated after receiving the last of 16 bits on the serial port. This interrupt is still disabled at this time however. The transmit framing synchronization pin (FSX) is configured to be an output. The F0 bit of the status register ST1, is initialized to zero which sets up the serial port to operate in the 16-bit mode.

The code for transmitting and receiving data is different for the two circuits. For circuit 1 the XF bit is first initialized to 0. The TC bit is set (TC is used as a flag to determine whether the processor is waiting for the A/D to perform a conversion, TC set, or whether the processor is shifting data, TC cleared). Next, the D_{IN} word is loaded into the ACC and shifted left three times so that it appears as in Figure 3. This D_{IN} word configures the LTC1090 for CH0 with respect to CH1, unipolar, MSB first, and 16-bit length. The D_{IN} word is then put in the transmit register and the RINT interrupt is enabled. For circuit 2 the code is similar except that XF and TC are not used. Also the D_{IN} word for circuit 2 configures the LTC1090 for 10 bits instead of 16 bits (Figure 6) because the additional hardware of circuit 2 allows fewer bits to be shifted which speeds up the transfer process. The circuit 1 code then causes the DSP to put out one SCLK cycle on the XF pin. This causes the FSX pin ($\overline{CS}$) to go high. The FSX pin stays high until the DSP goes through 208 NOPs during which time the LTC1090 performs a conversion. The XF bit is then reset and set with a 0.8μs period until the RINT interrupt is generated. For the circuit 2 code the timer consists of only one instruction that loops upon itself until the RINT interrupt is generated. All clocking and $\overline{CS}$ functions are performed by the hardware. This time could be used to do some simple processing of the data.

For circuit 1 once RINT is generated the code begins execution at the label RINT. This code stores the D_{OUT} word from the LTC1090 in the ACC, shifts it left one bit to position it properly and then stores it in location 200 hex. The data appears in location 200 hex left justified as shown in Figure 4. The code is set up to continually loop, so at this point the code jumps to label TXRX and repeats from there. The circuit 2 code handles the RINT interrupt in a similar fashion except that the data is shifted right five bits and is stored right justified as shown in Figure 10. Also the circuit 2 code has the delay for the LTC1090 in the RINT routine instead of during the TIMER routine.

Summary

Two interfaces between the LTC1090 10-bit data acquisition and the TMS320C25 DSP were demonstrated. The first interface required only one inverter in addition to the A/D and the DSP. The combined data conversion and transfer time of this interface was 41μs. The data was placed in the internal RAM of the TMS320C25 in a left justified format. The second circuit, which required two counters and three inverters to implement, was able to perform a conversion and shift the data to the processor in only 32μs. The data also was placed in the RAM of the TMS320C25 except that it was in a right justified format.

B14							B7
0	0	0	0	1	1	0	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 6. D_{IN} Word in ACC of TMS320C25 for Circuit 2

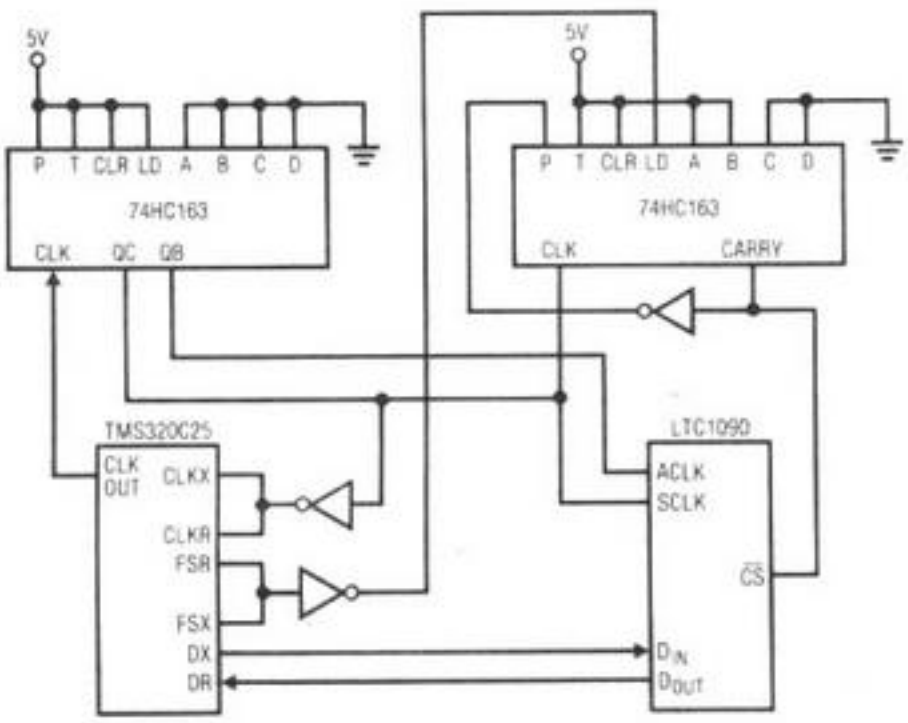


Figure 7. Circuit 2: Minimum Software Interface

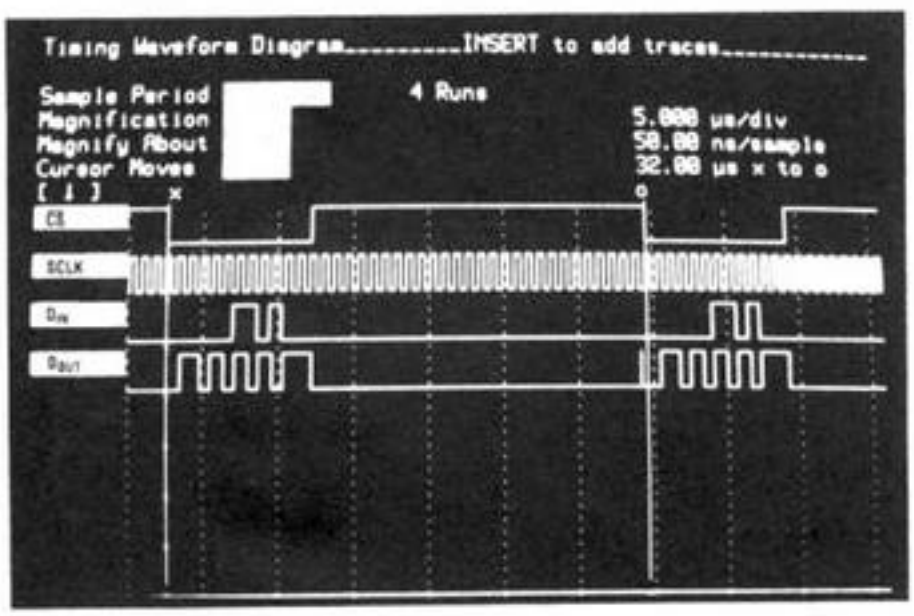


Figure 8. Timing for Circuit 2 shows 32µs Throughput Rate

```
LABEL    MNEMONIC          COMMENTS
         AORG   0          ON RST CODE STARTS AT 0
         B      INIT       BRANCH TO INIT ROUTINE

         AORG   >26        ADDRESS OF RINT VECTOR
         B      RINT       BRANCH TO RINT ROUTINE

         AORG   >32        MAIN PROGRAM ADDRESS
INIT     DINT               DISABLE INTERRUPTS
         LDPK   >0          DATA PAGE POINTER IS 0
         LARP   >1          AUX. REG. POINTER IS 1
         LRLK   AR1,>200    AUX. REG. 1 = >200
         LACK   >10         CONFIG. WORD FOR IMR
         SACL   >4          PUT CONFIG. WORD IN IMR
         STXM               CONFIGURE FSX AS OUTPUT
         FORT   0           SET SERIAL PORT TO 16 BITS

TXRX     LACK   >D0         LOAD DIN WORD INTO ACC
         SFSM               FSX STARTS ON XSR LOAD
         RPTK   2           REPEAT 3 TIMES
         SFL                SHIFT DIN TO LEFT
         SACL   >1          DIN PUT IN TX REGISTER
         EINT               ENABLE INTERRUPTS

TIMER    B      TIMER       LOOP UNTIL FINISHED

RINT     ZALS   >0          STORE DOUT WORD IN ACC
         RPTK   >4          REPEAT 5 TIMES
         SFR                SHIFT ACC RIGHT 1 BIT
         SACL   *,0         STORE ACC IN >200
         RPTK   127         DELAY
         NOP                22µs FOR
         RPTK   3           NEXT
         NOP                CONVERSION
         B      TXRX        GO TO TXRX ROUTINE
         END
```

Figure 9. TMS320C25 Code for Circuit 2

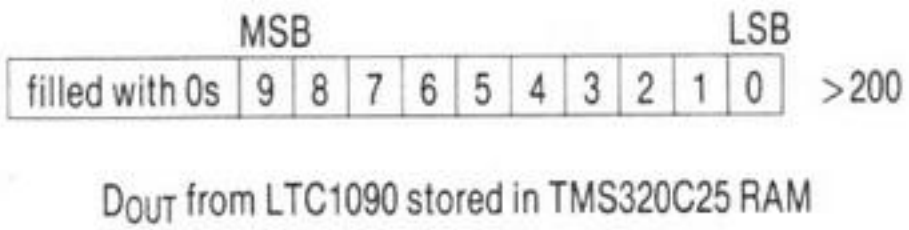
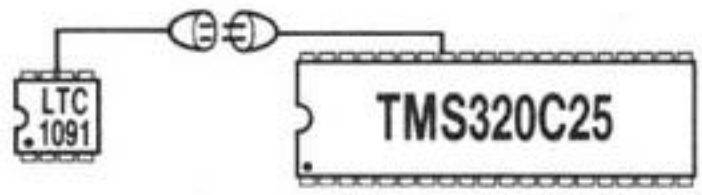


Figure 10. Memory Map for Circuit 2

Interfacing the LTC1091/92 to the TMS320C25 DSP



Guy Hoover

Introduction

This application note describes the hardware and software required for communication between the LTC1091 10-bit data acquisition system and the TMS320C25 digital signal processor (DSP). In particular two interfaces will be demonstrated. The first interface requires only one inverter in addition to the LTC1091 and the TMS320C25. The second interface, which is optimized for speed of transfer and minimum processor supervision, can complete a conversion and shift the data in only 32μs. Configuration of the TMS320C25 and LTC1091 will be discussed as it applies to this interface as well as the minor modifications required for the interface to work with the LTC1092. Schematics, code, and timing diagrams will be presented. Finally, a summary of results including data throughput rates will be provided.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a synchronous, half duplex format over D_{IN} and D_{OUT}. The serial port of the TMS320C25 is not directly compatible with that of the LTC1091. The data shift clock lines (CLKR, CLKX) are inputs only. Therefore the data shift clock must be externally generated. Inverting the shift clock is also necessary because the LTC1091 and the TMS320C25 clock data on opposite edges. This prevents a race condition.

The schematic of Figure 1 has the shift clock generated by the XF pin of the TMS320C25. The signal is inverted with a 74HC04 and fed into the CLK pin of the LCT1091. The framing pulse of the TMS320C25 is fed directly to the $\overline{CS}$ of the LTC1091. DX and DR are tied directly to D_{IN} and D_{OUT} respectively. This method results in the simplest hardware configuration but has the drawbacks of requiring more processor supervision and a slower data shift time.

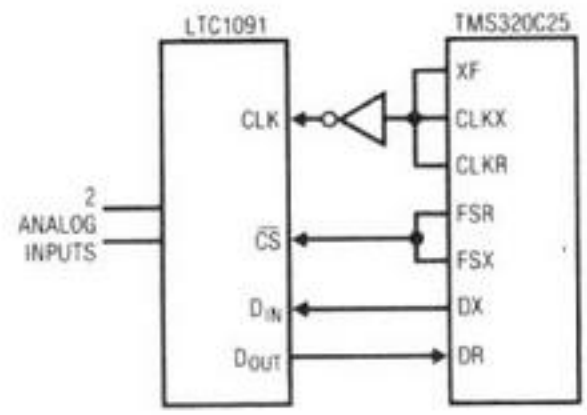


Figure 1. Circuit 1: Minimum Hardware Interface

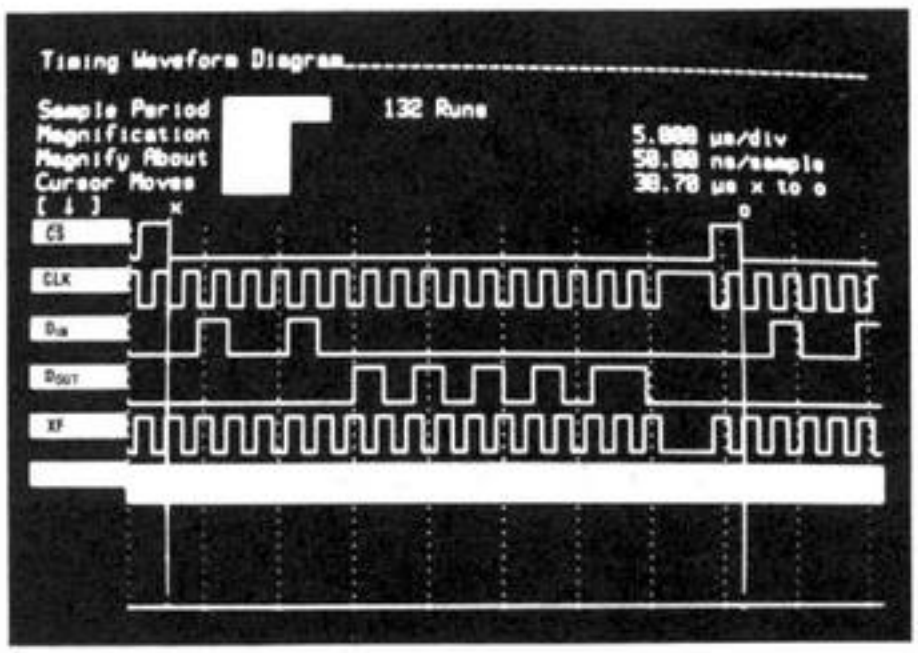


Figure 2. Timing for Circuit 1 Shows 39μs Throughput Rate

The schematic of Figure 6 has the clock generated by dividing down the CLK OUT pin by a factor of 16 with a 74HC163 counter. The signal is inverted with a 74HC04 and fed into the CLK pin of the LTC1091. The $\overline{CS}$ signal is generated directly from the FSX pin of the TMS320C25. The obvious disadvantage of this method is that it requires considerably more hardware to implement but it provides a faster data shift rate and requires less processor supervision.

Hardware Description

The DSP was emulated and the code for this interface was developed on a TMS320C25 Software Development System (SWDS). The SWDS requires a PC compatible computer to run.

The timing diagram of Figure 2 was obtained using the circuit of Figure 1. The timing diagram of Figure 7 was obtained using the circuit of Figure 6. Both pictures were taken with an HP1631 logic analyzer. The CLK was 500kHz for the timing diagram of Figure 2 and 625kHz for the timing diagram of Figure 7. The TMS320C25 clock rate was 40MHz.

The analog sections of the schematics of Figure 1 and Figure 6 are omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

B15							B8
0	1 START	0 S/D	0 O/S	1 MSBF	0	0	0

Figure 3. D_{IN} Word in ACC of TMS320C25 for Circuits 1 and 2

	MSB														LSB	
	X	X	X	X	X	X	9	8	7	6	5	4	3	2	1	0	>200

D_{OUT} from LTC1091 stored in TMS320C25 RAM

Figure 4. Memory Map for Circuits 1 and 2

```
LABEL   MNEMONIC        COMMENTS
        AORG    0       ON RST CODE STARTS AT 0
        B       INIT    BRANCH TO INIT ROUTINE

        AORG    >26     ADDRESS OF RINT VECTOR
        B       RINT    BRANCH TO RINT ROUTINE

        AORG    >32     MAIN PROGRAM ADDRESS
INIT    DINT            DISABLE INTERRUPTS
        LDPK    >0      DATA PAGE POINTER IS 0
        LARP    >1      AUX. REG. POINTER IS 1
        LRLK    AR1,>200 AUX. REG. 1 = >200
        LACK    >10     CONFIG. WORD FOR IMR
        SACL    >4      PUT CONFIG. WORD IN IMR
        STXM            CONFIGURE FSX AS OUTPUT
        FORT    0       SET SERIAL PORT TO 16 BITS
        RXF             RESET XF

TXRX    LACK    >48     LOAD D_IN WORD INTO ACC
        SFSM            FSX STARTS ON XSR LOAD
        RPTK    7       REPEAT 8 TIMES
        SFL             SHIFT D_IN TO LEFT
        SACL    >1      D_IN PUT IN TX REGISTER
        EINT            ENABLE INTERRUPTS

TIMER   SXF             SET XF (CLK)
        RPTK    5       REPEAT 6 TIMES
        NOP             TIMING
        RXF             RESET XF (CLK)
        RPTK    3       REPEAT 4 TIMES
        NOP             NOP FOR TIMING
        B       TIMER   CLKS UNTIL RINT

RINT    ZALS    >0      STORE D_OUT WORD IN ACC
        SACL    *,0     STORE ACC IN >200
        B       TXRX    GO TO TXRX ROUTINE
        END
```

Figure 5. TMS320C25 Code for Circuit 1

Software Description

The software configures and controls the serial port of the TMS320C25.

The first 13 lines of code are the same for circuit 1 and circuit 2. The code first sets up the interrupt and reset vectors. On reset the TMS320C25 starts executing code at the label INIT. Upon completion of a 16-bit data transfer, an interrupt is generated and the DSP will begin executing code at the label RINT.

Next, the code initializes registers in the TMS320C25 that will be used in the transfer routine. The interrupts are temporarily disabled. The data memory page pointer register is set to zero. The auxiliary register pointer is loaded with one and auxiliary register one is loaded with the value 200 hexadecimal. This is the data memory location where the data from the LTC1091 will be stored. The interrupt mask register (IMR) is configured to recognize the RINT interrupt, which is generated after receiving the last of 16 bits on the serial port. This interrupt is still disabled at this time however. The transmit framing synchronization pin (FSX) is configured to be an output. The F0 bit of the status register ST1, is initialized to zero which sets up the serial port to operate in the 16-bit mode. For circuit 1 the XF bit is first initialized to zero.

The code for transmitting and receiving data is the same for the two circuits except for the section of code labelled TIMER. The D_{IN} word is loaded into the ACC and shifted left eight times so that it appears as in Figure 3. This D_{IN} word configures the LTC1091 for CH0 with respect to CH1 and MSB first. The D_{IN} word is then put in the transmit register and the RINT interrupt is enabled. For circuit 1 the XF bit is set which causes the FSX pin to generate a $\overline{CS}$ signal which is fed into the $\overline{CS}$ pin of the LTC1091 and the FSR pin of the TMS320C25.

The XF bit is then reset and set with a 2.0μs period until the RINT interrupt is generated. For the circuit 2 code the timer consists of only one instruction that loops upon itself until the RINT interrupt is generated. All clocking and $\overline{CS}$ functions are performed by the hardware. This time could be used to do some simple processing of the data.

Once RINT is generated the code begins execution at the label RINT. This code stores the D_{OUT} word from the LTC1091 in the ACC and then stores it in location 200 hex. The data appears in location 200 hex right justified as shown in Figure 4. The code is set up to continually loop, so at this point the code jumps to label TXRX and repeats from there.

The code for circuits 1 and 2 can be made to work with the LTC1092 as well with only minor modifications. It is not necessary to use a D_{IN} word for the LTC1092, which reduces the number of lines required by the interface. After the data has been shifted into the TMS320C25 it must be shifted twice to the left for left justified data or shifted four times to the right for right justified data.

Summary

Two interfaces between the LTC1091 10-bit data acquisition and the TMS320C25 DSP were demonstrated. The first interface required only one inverter in addition to the A/D and the DSP. The combined data conversion and transfer time of this interface was 39μs. The data was placed in the internal RAM of the TMS320C25 in a right justified format. The second circuit, which required a counter and an inverter to implement, was able to perform a conversion and shift the data to the processor in only 32μs. The data again was placed in the RAM of the TMS320C25 in a right justified format. With only minor modifications these interfaces can also be used with the LTC1092.

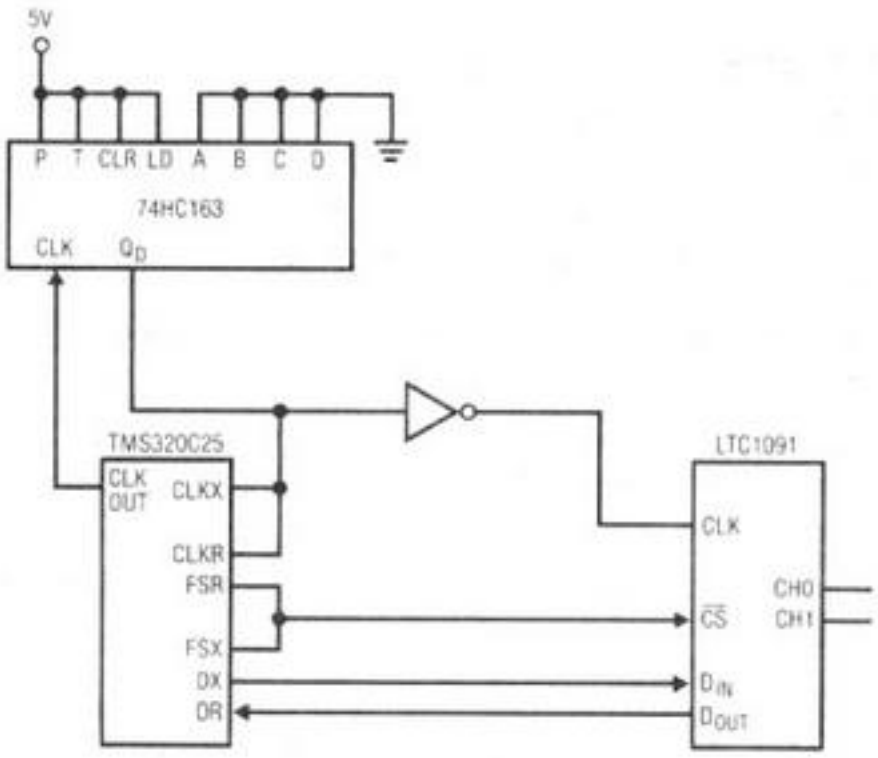


Figure 6. Circuit 2: Minimum Software Interface

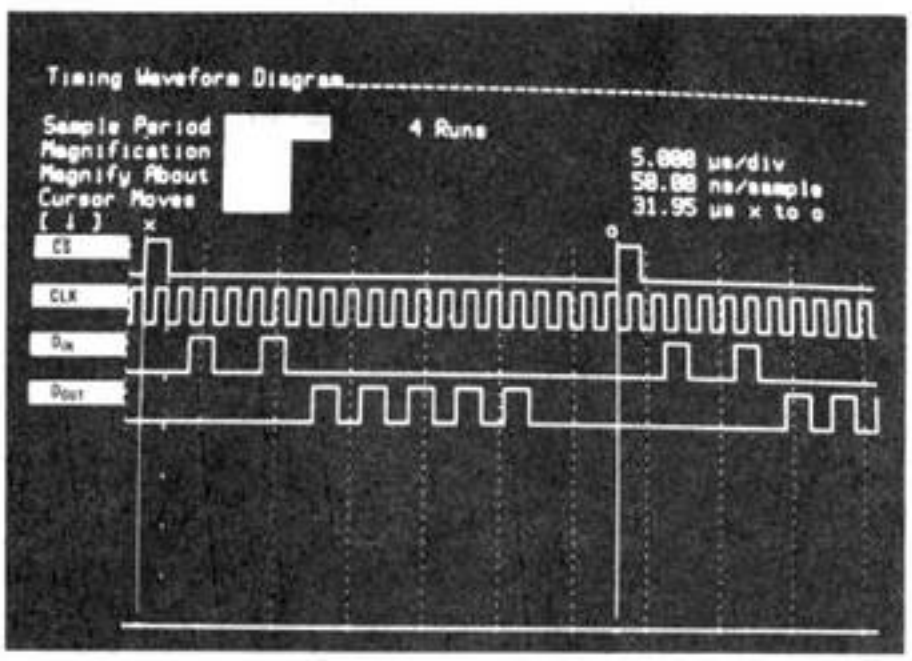
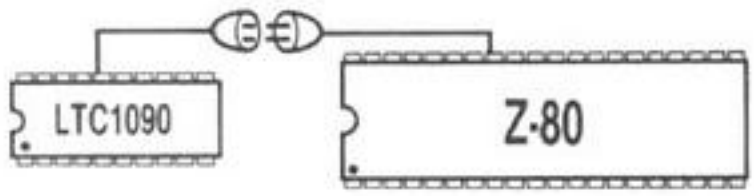


Figure 7. Timing for Circuit 2 Shows 32μs Throughput Rate

```
LABEL     MNEMONIC          COMMENTS
          AORG    0         ON RST CODE STARTS AT 0
          B       INIT      BRANCH TO INIT ROUTINE

          AORG    >26       ADDRESS OF RINT VECTOR
          B       RINT      BRANCH TO RINT ROUTINE

          AORG    >32       MAIN PROGRAM ADDRESS
INIT      DINT              DISABLE INTERRUPTS
          LDPK    >0        DATA PAGE POINTER IS 0
          LARP    >1        AUX. REG. POINTER IS 1
          LRLK    AR1,>200  AUX. REG. 1 = >200
          LACK    >10       CONFIG. WORD FOR IMR
          SACL    >4        PUT CONFIG. WORD IN IMR
          STXM              CONFIGURE FSX AS OUTPUT
          FORT    0         SET SERIAL PORT TO 16 BITS

TXRX      LACK    >48       LOAD D_IN WORD INTO ACC
          SFSM              FSX STARTS ON XSR LOAD
          RPTK    7         REPEAT 8 TIMES
          SFL               SHIFT D_IN TO LEFT
          SACL    >1        D_IN PUT IN TX REGISTER
          EINT              ENABLE INTERRUPTS

TIMER     B       TIMER     LOOP UNTIL FINISHED

RINT      ZALS    >0        STORE D_OUT WORD IN ACC
          SACL    *,0       STORE ACC IN >200
          B       TXRX      GO TO TXRX ROUTINE
          END
```

Figure 8. TMS320C25 Code for Circuit 2

Interfacing the LTC1090 to the Z-80 MPU



Guy Hoover

Introduction

This application note describes an interface between the LTC1090 10-bit data acquisition system and the Z-80 microcomputer. The interface is capable of completing a 10-bit conversion and shifting the data to Z-80 in 288μs. Configuration of the LTC1090 and the Z-80 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given, including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred serially in a synchronous, full duplex format over D_{IN} and D_{OUT}.

The Z-80 does not have a serial port. Therefore it is necessary for the user to construct a serial port with TTL gates as shown in the schematic of Figure 1.

Hardware Description

$\overline{CS}$ is set or cleared by placing a 1 or a 0 on address line A0 and writing to an I/O port that has an even address of 128 or higher. The LTC1090 SCLK is generated by reading from an I/O port that has an address greater than 128. Data is clocked into the LTC1090 one bit at a time by placing the desired bit on D7 of the Z-80 and writing to any memory location. The serial data output of the LTC1090 is fed into D0 of the Z-80 through the 74LS126. The 74LS126 prevents the LTC1090 from writing to the data bus of the Z-80 except when the microprocessor requires data from the A/D. The ACLK of the LTC1090 is also the clock for the Z-80.

The code for this interface was developed on a Multitech MPF-1 single board development system.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer. The Z-80 clock rate was 1.79MHz. Using a Z-80B and running it at a 6MHz clock rate, it is possible to reduce this time to approximately 100μs. This would require generating ACLK externally or dividing down the ϕ signal.

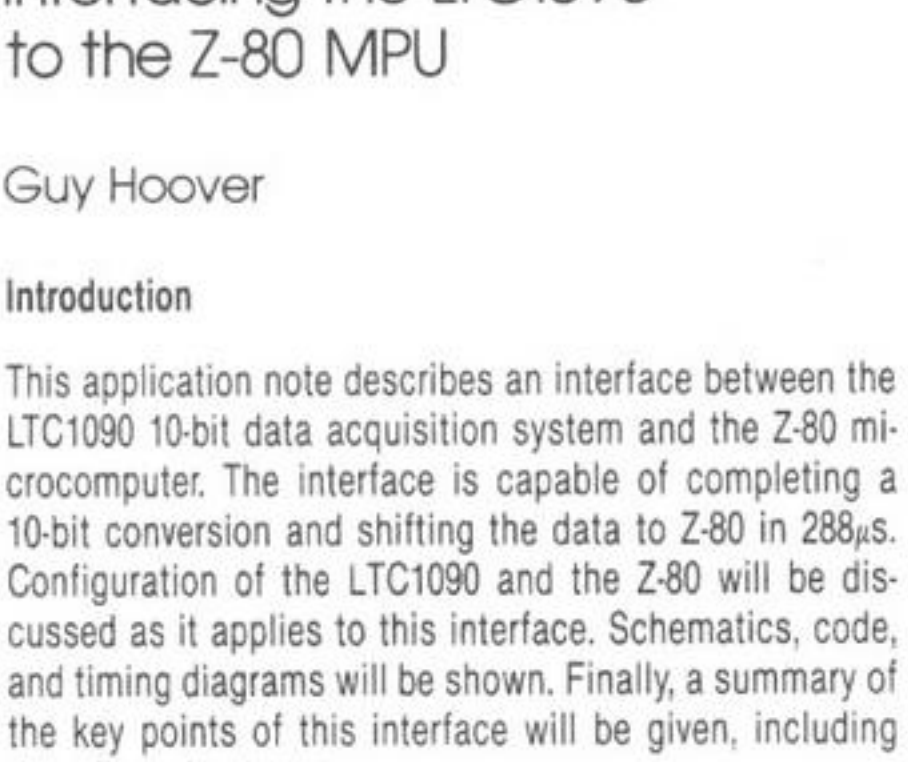


Figure 1. Serial Interface Requires Four 74LS Chips

Figure 2. Throughput Time is Limited by the Z-80 MPU. A 10 Bit Conversion Result is Transmitted Every 288μs.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

Software Description

The software serially shifts the D_{IN} configuration word to the LTC1090 while simultaneously reading the previous data back. Additionally, the software waits while the LTC1090 performs its next conversion before attempting the next data exchange cycle.

The code, Figure 5, first clears the C register. Next the E register, which is used as a counter, is loaded with the value 8. The D register is loaded with the D_{IN} word for the LTC1090. This word as shown in Figure 3 configures the LTC1090 for channel 7 with respect to common. D_{IN} also sets up the LTC1090 for unipolar mode, MSB first and tells the A/D to shift out 10 bits of data. $\overline{CS}$ is brought low by writing to I/O port 128 (80H). The MSB of the D register containing the D_{IN} word is then output on bit 7 of the data bus of the Z-80. The first bit of the LTC1090 D_{OUT} word is then read into the A register. The act of reading this bit also generates an SCLK pulse. The D_{OUT} bit is then shifted into the carry bit and from there it is rotated into the LSB of the B register. The next bit of the D_{IN} word is shifted into the MSB of the D register. The E register counter is decremented. At this point a test is made to determine if the first eight bits have been shifted. If not, another D_{IN} bit is output and D_{OUT} bit is read until eight bits have been shifted. The two LSBs of the D_{OUT} word are similarly shifted into the C register. These two bits are then shifted right through the carry until they are in the two MSB positions of the C register. $\overline{CS}$ is then brought high. The 10 bit LTC1090 D_{OUT} word is stored left justified in the Z-80 at this time as shown in Figure 4.

After the last SCLK pulse is ended 44 ACLK cycles must be allowed for the LTC1090 to perform the desired the A/D conversion. During this time $\overline{CS}$ is taken high. The software must ensure that this occurs.

Summary

An interface between the LTC1090 10 bit data acquisition system and the Z-80 microprocessor with a combined data conversion and transfer time of 288μs was demonstrated. The interface used four 74LS chips to interface the two devices. The 10 data bits of the LTC1090 are shifted MSB first one bit at a time. The data is stored left justified in the Z-80's internal registers.

1	1	1	1	1	1	0	1
S/D	O/S	S1	S2	UNI	MSBF	WL1	WL0

Figure 3. D_{IN} Word for LTC1090 Stored in D Register of Z-80

MSB

9	8	7	6	5	4	3	2	REG B

LSB

1	0	FILLED WITH 0's	REG C

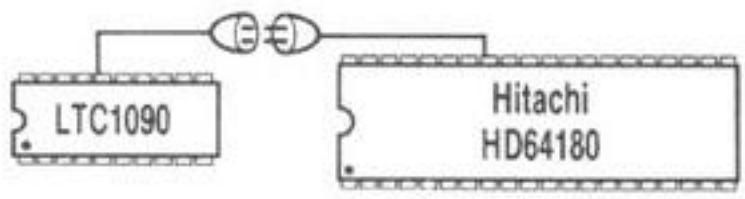
D_{OUT} from LTC1090 stored in Z-80 registers

Figure 4. Memory Map of Z-80

LABEL	MNEMONIC		COMMENTS
BEGIN	LD	C,00H	INITIALIZE REG C
	LD	E,08H	INITIALIZE REG E
	LD	D,FDH	PUT D_{IN} IN REG D
	OUT	(80H),A	$\overline{CS}$ GOES LOW
LOOP	LD	(HL),D	OUTPUT D_{IN} BIT
	IN	A,(80H)	READ D_{OUT} BIT
	RRA		SHIFT DATA TO CARRY
	RL	B	SHIFT DATA TO REG B
	RLC	D	SHIFT D_{IN} WORD LEFT
	DEC	E	DECREMENT COUNTER
	JP	NZ,LOOP	GET NEXT BIT IF NOT 0
	IN	A,(80H)	READ BIT 1 OF D_{OUT}
	RRA		SHIFT BIT INTO CARRY
	RL	C	SHIFT DATA TO REG C
	IN	A,(80H)	READ BIT 0 OF D_{OUT}
	RRA		SHIFT BIT INTO CARRY
	RR	C	REG C SHIFTS RIGHT
	RR	C	REG C SHIFTS RIGHT
	OUT	(81H),A	$\overline{CS}$ GOES HIGH

Figure 5. Z-80 Code

Interfacing the LTC1090 to the HD64180



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1090 10-bit data acquisition system and the Hitachi 64180 microprocessor. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 64180 in 96μs. Configuration of the LTC1090 and the 64180 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1090 has two clock lines: ACLK and SCLK. ACLK controls the A/D conversion rate while SCLK controls the data shift rate. Data is transferred in a synchronous full duplex format over D_{IN} and D_{OUT}.

The 64180 has a clocked serial I/O port (CSIO) that allows the user to construct a simple communication path to the LTC1090. The serial port provides clock, transmit and receive lines that are compatible with the LTC1090. The only additional line required is one programmable output pin (RTSO) to control $\overline{CS}$ on the LTC1090. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. ACLK of the LTC1090 was 2MHz and the 64180 crystal frequency was 4MHz. This produced a transfer time of 479μs. A version of the 64180 can be run at a 20MHz crystal frequency so the times shown can be reduced by a factor of five yielding a total transfer time of 96μs.

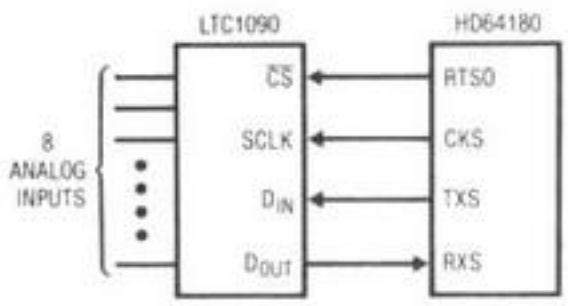


Figure 1. LTC1090 Transmits Data to HD64180 Using 4 Wires

At crystal frequencies up to 4MHz ACLK can be generated directly from the ϕ pin of the 64180. Above 4MHz a divider must be used to generate ACLK or it must be externally generated to ensure it remains below 2MHz.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1090 please see the data sheet.

Software Description

The software configures and controls the CSIO of the 64180. Additionally, the software manipulates RTSO ($\overline{CS}$ of the LTC1090) and generates a delay during which time the LTC1090 performs a conversion. Because the CSIO of the 64180 communicates in a half duplex format it is necessary for the software to first write a configuration word to

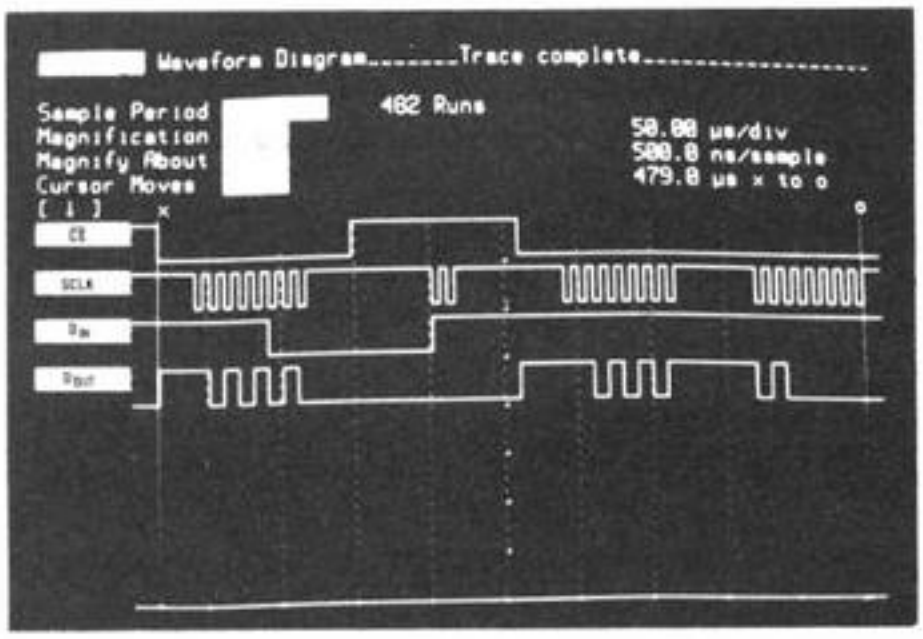


Figure 2. Timing Diagram. Transfer Times as Short as 96μs are Possible.

the LTC1090 and then read back the data. Normally with the LTC1090 the D_{IN} and D_{OUT} words are transferred simultaneously.

The software first disables all interrupts and enables the receive (RXS) pin. The D_{IN} word is loaded into the Transmit Receive Data Register. The D_{IN} word programs the LTC1090 for channel 7 with respect to common, LSB first, unipolar and eight bits as shown in Figure 3. Note, that for LSB first format processors the D_{IN} word must be constructed opposite from MSB first format. This is because the bits forming the D_{IN} word of the LTC1090 must always be shifted in the same order regardless of whether MSB first or LSB first is chosen. B4 of CNTLA0 is cleared which causes $\overline{CS}$ of the LTC1090 to go low. B4 of the CSIO control register is set which causes the D_{IN} word for the LTC1090 to begin transmitting. B7 of the CSIO control register is polled until a 1 is detected. B4 of CNTLA0 is then set which causes $\overline{CS}$ to go high.

Forty-four ACLK cycles must pass before $\overline{CS}$ can be taken low so that the A/D can perform a conversion. During this time a transmit is started and stopped so that the TXS line will stop high. The transmit is not allowed to finish to save time. It is desirable to have the TXS line high so that the proper word length will be clocked into the LTC1090 when D_{OUT} is read.

$\overline{CS}$ of the LTC1090 is again cleared. The CSIO control register is set up to receive this time. The transmit line is held high so that all ones are clocked into the LTC1090 D_{IN} pin while the LSBs of D_{OUT} are being clocked into the 64180. The same polling method is used as before. After the first eight bits are received the data is stored in Register L. The two MSBs are then clocked in and placed in Register H. The data at this point is right justified with the unused bits being set to 0s as shown in Figure 4. $\overline{CS}$ of the LTC1090 is then set again.

Because the D_{IN} word received by the LTC1090 was a dummy word, it is not necessary to wait 44 ACLK cycles again at this point. The $\overline{CS}$ line can be brought low immediately and another cycle begun at this time.

0	0	0	1	1	1	1	1	
WL0	WL1	MSBF	UNI	S2	S1	O/S	S/D	REG D

Figure 3. D_{IN} Word for LTC1090 Stored in Reverse Order in 64180 Internal Registers

								LSB	
LSB	7	6	5	4	3	2	1	0	REG L
							MSB		
MSB	0	0	0	0	0	0	9	8	REG H

Figure 4. D_{OUT} from LTC1090 Stored in 64180 Internal Registers

Summary

A four wire interface between the LTC1090 and the Hitachi 64180 with a combined data conversion and transfer time of 96μs was demonstrated. The interface used the CSIO port of the 64180. Because the CSIO port transfers data LSB first care must be taken in constructing the D_{IN} word so that the bits are transmitted in the proper order to the LTC1090. A configuration word is written to the LTC1090 in one eight bit transfer and then the 10 data bits of the LTC1090 are shifted LSB first to the 64180 in two eight bit transfers. The data is stored right justified in the 64180's internal registers.

ADDR	LABEL	CODE	MNEMONIC	COMMENTS
0	BEGIN	F3	DI	DISABLE INTERRUPTS
1		1E 00	LD E, 00H	DATA FOR ASCI STATUS REG
3		ED 19 05	OUT0 (05H), E	ENABLE RXS
6	LOOP	16 1F	LD D, 1FH	LOAD D_{IN} IN REG D
8		ED 11 0B	OUT0 (0BH), D	LOAD D_{IN} IN TRDR
B		1E 00	LD E, 00H	DATA FOR CNTLA0
D		ED 19 00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ RESET
10		1E 10	LD E, 10H	DATA FOR CSIO CONTROL REG
12		ED 19 0A	OUT0 (0AH), E	START TRANSMIT OF D_{IN}
15		0E 0A	LD C, 0AH	ADDR OF CSIO CONTROL REG
17	TX1	ED 74 80	TSTIO 80H	
1A		28 FB	JR Z TX1	WAIT FOR TRANSFER TO END
1C		1E 10	LD E, 10H	DATA FOR CNTLA0
1E		ED 19 00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ SET
21		16 FF	LD D, FFH	DUMMY DATA WORD
23		ED 11 0B	OUT0 (0BH), D	LOAD DUMMY IN TRDR
26		1E 10	LD E, 10H	DATA FOR CSIO CONTROL REG
28		ED 19 0A	OUT0 (0AH), E	START TRANSMIT OF DUMMY
2B		1E 00	LD E, 00H	DATA FOR CSIO CONTROL REG
2D		ED 19 0A	OUT0 (0AH), E	STOP TRANSMIT OF DUMMY
30		ED 20 0B	IN0 H, (0BH)	CLEAR EF OF CSIO CNTRL REG
33		1E 00	LD E, 00H	DATA FOR CNTLA0
35		ED 19 00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ RESET
38		1E 20	LD E, 20H	DATA FOR CSIO CNTRL REG
3A		ED 19 0A	OUT0 (0AH), E	RECEIVE D_{OUT} LSBs
3D	RX1	ED 74 80	TSTIO 80H	
40		28 FB	JR Z RX1	WAIT FOR TRANSFER TO END
42		ED 28 0B	IN0 L, (0BH)	PUT LSBs IN REG L
45		ED 19 0A	OUT0 (0AH), E	RECEIVE D_{OUT} MSBs
48	RX2	ED 74 80	TSTIO 80H	
4B		28 FB	JR Z RX2	WAIT FOR TRANSFER TO END
4D		ED 20 0B	IN0 H, (0BH)	PUT MSBs IN REG H
50		1E 10	LD E, 10H	DATA FOR CNTLA0
52		ED 19 00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ SET
55		C3 06 00	JP LOOP	DO NEXT CONVERSION

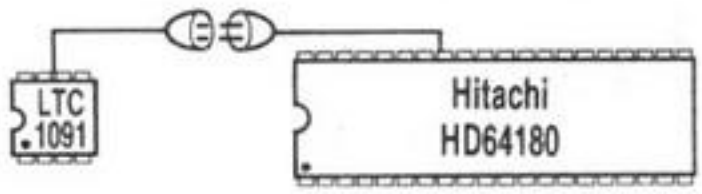
Figure 5. HD64180 Code Transfers Data to and from the LTC1090

LINEAR TECHNOLOGY

Application Note 26Q

November 1988

Interfacing the LTC1091 to the HD64180



Guy Hoover
William Rempfer

Introduction

This application note describes an interface between the LTC1091 10-bit data acquisition system and the Hitachi 64180 microprocessor. The simple four wire interface is capable of completing a 10-bit conversion and shifting the data to the 64180 in 91μs. Configuration of the LTC1091 and the 64180 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. Finally, a summary of the key points of this interface will be given including data throughput rates.

Interface Details

The LTC1091 clock line controls the A/D conversion rate and the data shift rate. Data is transferred in a synchronous half duplex format over D_{IN} and D_{OUT}.

The 64180 has a clocked serial I/O port (CSIO) that allows the user to construct a simple communication path to the LTC1091. The serial port provides clock, transmit and receive lines that are compatible with the LTC1091. The only additional line required is one programmable output pin (RTSO) to control $\overline{CS}$ on the LTC1091. The schematic of Figure 1 shows how the two devices are connected.

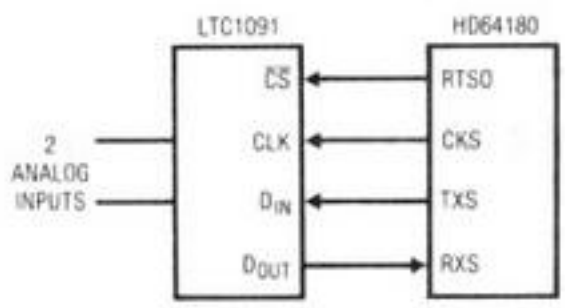


Figure 1. LTC1091 Transmits Data to HD64180 Using 4 Wires

Hardware Description

The timing diagram of Figure 2 was obtained using an HP1631A logic analyzer. The 64180 crystal frequency was 4MHz. This produced a transfer time of 455μs. A version of the 64180 can be run at 20MHz crystal frequency so the times shown can be reduced by a factor of five yielding a total transfer time of 91μs.

The analog section of the schematic of Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1091 please see the data sheet.

Software Description

The software configures and controls the CSIO of the 64180. Additionally, the software manipulates RTSO ($\overline{CS}$ of the LTC1091).

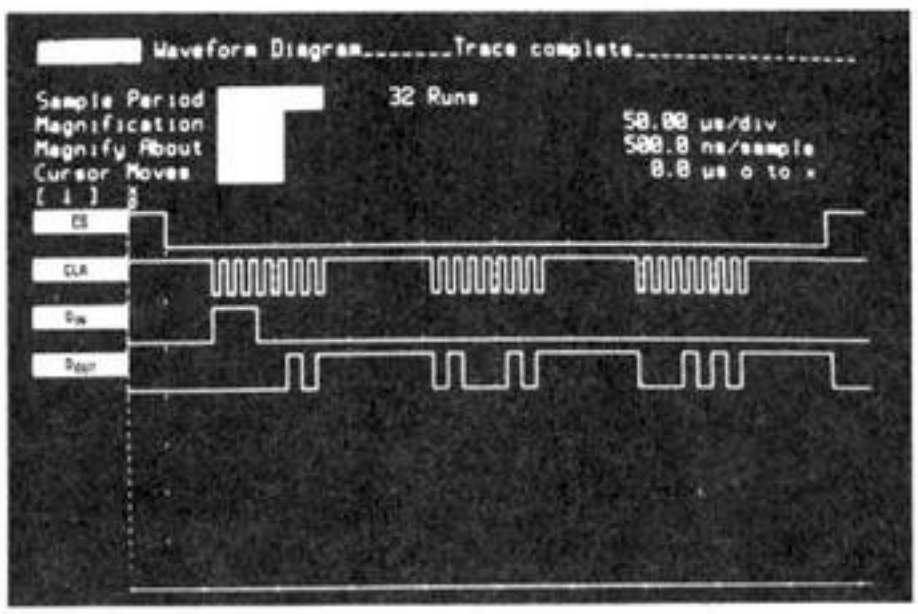


Figure 2. Timing Diagram. Throughput Times as Short as 91μs are Possible.

The software first disables all interrupts and enables the receive (RXS) pin. The D_{IN} word is loaded into the Transmit Receive Data Register. The D_{IN} word programs the LTC1091 for channel 1 with respect to ground and LSB first as shown in Figure 3. Note, that for LSB first format processors the D_{IN} word must be constructed opposite from MSB first format. This is because the bits forming the D_{IN} word into the LTC1091 must always be shifted in the same order regardless of whether MSB first or LSB first is chosen. B4 of CNTLA0 is cleared which causes $\overline{CS}$ of the LTC1091 to go low. B4 of the CSIO control register is set which causes the D_{IN} word for the LTC1091 to begin transmitting. After receiving the start bit, channel information, and LSB first format data the LTC1091 starts transmitting the results of the conversion back to the 64180 in MSB first format. The first three bits of this data are ignored by the 64180 because it is still in the transmit mode. B7 of the CSIO control register is polled until a 1 is detected signifying the end of the transfer.

The CSIO control register is set up to receive this time. The same polling method is used as before. After the first eight bits are received the data is stored in Register A. The last two bits received were transmitted in LSB first format. These two bits end up in the two MSBs of REG A, where they are ANDed with C0H to clear the LSBs of REG A. The D_{OUT} LSBs in REG A are then stored in the L register. The eight MSBs of D_{OUT} are then clocked in and placed in Register H. The data at this point is left justified as shown in Figure 4. $\overline{CS}$ of the LTC1091 is then set again. The $\overline{CS}$ line can be brought low immediately and another cycle begun at this time.

0	0	0	0	0 MSBF	1 O/S	1 S/D	1 START	REG D

Figure 3. D_{IN} Word for LTC1091 Stored in Reverse Order in 64180 Internal Registers

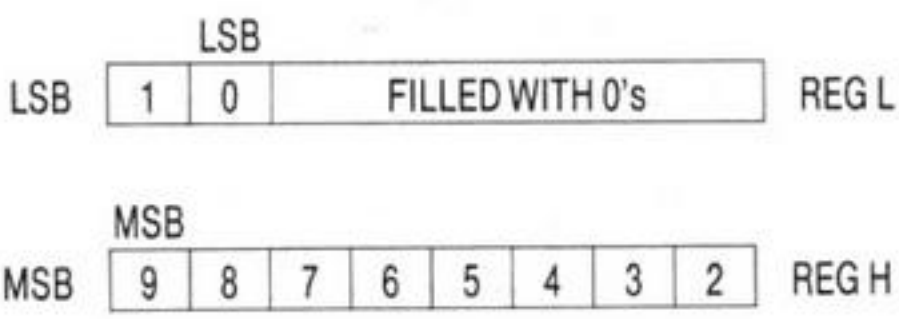


Figure 4. D_{OUT} from LTC1091 Stored in 64180 Internal Registers

Summary

A four wire interface between the LTC1091 and the Hitachi 64180 with a combined data conversion and transfer time of 91μs was demonstrated. The interface used the CSIO port of the 64180. Because the CSIO port transfers data LSB first care must be taken in constructing the D_{IN} word so that the bits are transmitted in the proper order to the LTC1091. A configuration word is written to the LTC1091 in one eight bit transfer and then the 10 data bits of the LTC1091 are shifted LSB first to the 64180 in two eight bit transfers. The data is stored left justified in the 64180's internal registers.

ADDR	LABEL	CODE			MNEMONIC	COMMENTS
0	BEGIN	F3			DI	DISABLE INTERRUPTS
1		1E	00		LD E, 00H	DATA FOR ASCI STATUS REG
3		ED	19	05	OUT0 (05H), E	ENABLE RXS
6	LOOP	16	07		LD D, 07H	LOAD D_{IN} IN REG D
8		ED	11	0B	OUT0 (0BH), D	LOAD D_{IN} IN TRDR
B		1E	00		LD E, 00H	DATA FOR CNTLA0
D		ED	19	00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ RESET
10		1E	10		LD E, 10H	DATA FOR CSIO CONTROL REG
12		ED	19	0A	OUT0 (0AH), E	START TRANSMIT OF D_{IN}
15		0E	0A		LD C, 0AH	ADDR OF CSIO CONTROL REG
17	TX1	ED	74	80	TSTIO 80H	
1A		28	FB		JR Z TX1	WAIT FOR TRANSFER TO END
1C		ED	20	0B	IN0 H, (0BH)	CLEAR EF OF CSIO CNTRL REG
1F		1E	20		LD E, 20H	DATA FOR CSIO CNTRL REG
21		ED	19	0A	OUT0 (0AH), E	RECEIVE D_{OUT} LSBs
24	RX1	ED	74	80	TSTIO 80H	
27		28	FB		JR Z RX1	WAIT FOR TRANSFER TO END
29		ED	38	0B	IN0 A, (0BH)	PUT LSBs IN REG A
2C		E6	C0		AND C0H	MASK OUT LSBs
2E		6F			LD L, A	PUT D_{OUT} IN REG L
2F		ED	19	0A	OUT0 (0AH), E	RECEIVE D_{OUT} MSBs
32	RX2	ED	74	80	TSTIO 80H	
35		28	FB		JR Z RX2	WAIT FOR TRANSFER TO END
37		ED	20	0B	IN0 H, (0BH)	PUT MSBs IN REG H
3A		1E	10		LD E, 10H	DATA FOR CNTLA0
3C		ED	19	00	OUT0 (00H), E	DATA IN CNTLA0. $\overline{CS}$ SET

Figure 5. HD64180 Code Transfers Data to and from the LTC1091

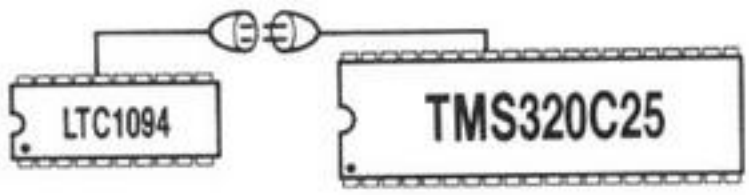
LINEAR TECHNOLOGY

Application Note 26R

January 1989

Interfacing the LTC1094 to a Parallel Bus

Guy Hoover
William Rempfer



Introduction

This application note describes the hardware and software required to interface the LTC1094 10-bit data acquisition system to the TMS320C25 digital signal processor. The circuitry shown can be used to interface any member of the LTC1090 family to the bus of virtually any processor with only minor modifications. The software provided is specific to the TMS320 family. The interface shown can be either interrupt driven or polled by the processor after a convert command has been given to the LTC1094. The interface is capable of completing a 10-bit conversion and transferring the data to the TMS320C25 in 40μs. Configuration of the LTC1094 and the TMS320C25 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be discussed. Finally, a summary of results will be provided.

Interface Details

The LTC1094 is a serial 10-bit data acquisition system. It uses a half duplex synchronous serial interface. It uses a D_{IN} word to configure the A/D for channel number, unipolar or bipolar, and MSB first or LSB first. Data is shifted out on the D_{OUT} line. Both D_{IN} and D_{OUT} are synchronous with the CLK line.

Many processors do not have a compatible serial port. It then becomes necessary to construct an interface circuit that will allow the LTC1094 to hang directly on the data bus of the processor. The circuit of Figure 1 interfaces directly to the TMS320C25 bus. It latches the D_{IN} word provided by the processor and then shifts it to the LTC1094. The LTC1094 then clocks out the D_{OUT} word to a shift register where the data is latched and the conversion complete signal is sent to the processor. When the processor requests the D_{OUT} word the interface circuit comes out of tri-state and the data is present on the bus. The circuit generates the clock for the LTC1094 and automatically shuts it off after the conversion has been done. The processor must provide chip select, read/write and interrupt lines as well as a 16-bit data bus. (For an 8-bit data bus a two byte read would be required.) When the read/write line is LOW and the chip select line is LOW the D_{IN} word is latched into the 74LS165. When the read/write line is HIGH and the chip select line is LOW the D_{OUT} word is read from the 74LS365 tri-state buffers.

Hardware Description

The DSP was emulated and the code for this interface was developed on a TMS320C25 Software Development System (SWDS).

The timing diagram of Figure 2 was obtained with an HP1631 logic analyzer with an LTC1094 CLK frequency of 500kHz.

The analog section of the schematic in Figure 1 is omitted for clarity. For a complete discussion of the analog considerations involved in using the LTC1094 please see the data sheet.

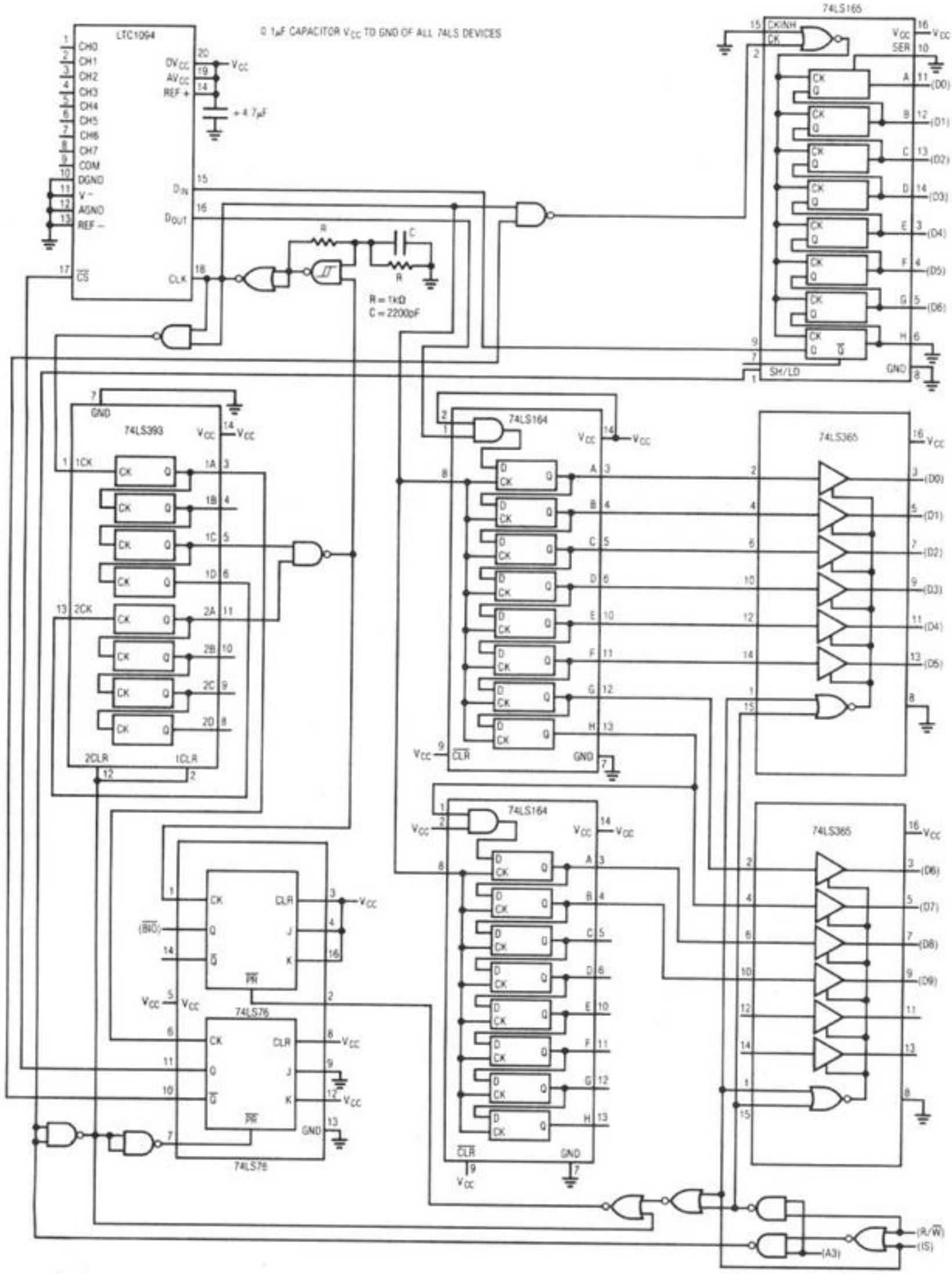


Figure 1. Circuit Allows LTC1094 to Interface Directly with the TMS320C25 Data Bus

Software Description

The software outputs a D_{IN} word from the TMS320C25 to the interface and when informed by the interface that the resulting data is present reads in the D_{OUT} word of the LTC1094.

The code of Figure 5 first disables all interrupts. Next, a D_{IN} word is placed in >60 of the TMS320C25 as shown in Figure 3. This D_{IN} word configures the LTC1094 for CH7 with respect to COM, unipolar, and MSB first. This D_{IN} word is then output to Port 8 of the TMS320C25. The software then polls the $\overline{BIO}$ pin until the interface pulls it LOW indicating that the conversion is complete and that the data is ready to be received by the TMS320C25. (The interface could just as easily connect to one of the maskable interrupt pins so that the processor could be performing some task while waiting for the conversion to be completed.) The data is then read into the TMS320C25 and placed into >61. The data at this point is right justified as shown in Figure 4.

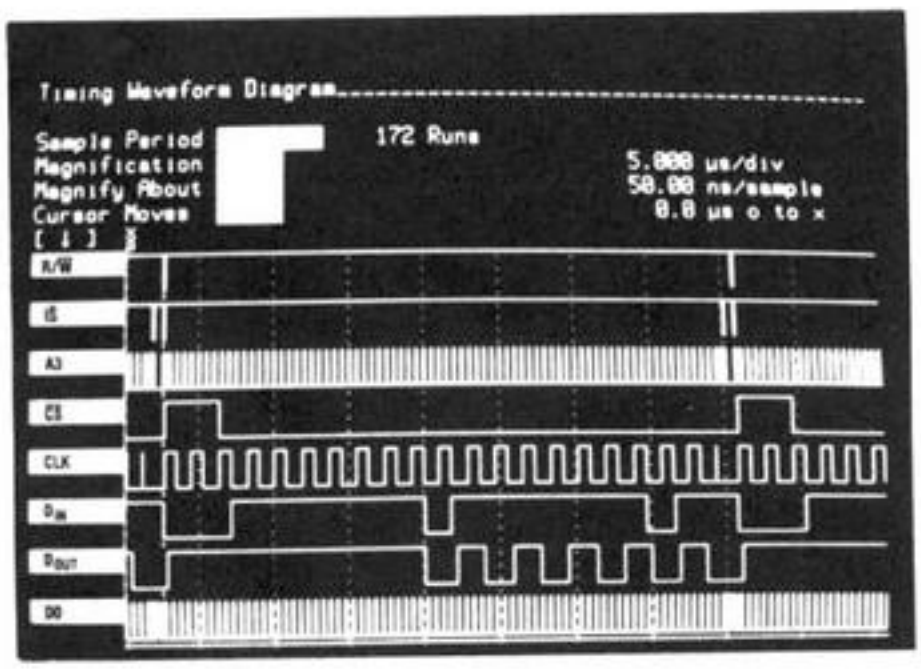


Figure 2. Timing Diagram Shows 25kHz Throughput Rate

Summary

An interface between the LTC1094 and the TMS320C25 with a combined data conversion and transfer rate of 40μs was demonstrated. This inexpensive interface (about $2.00 in production quantities) uses ten 74LS chips to allow the LTC1094 to hang directly on the data bus of the TMS320C25. The circuit shown here should work with any 16-bit processor (an 8-bit processor would require two reads per conversion) that has a read/write line, and external interrupt capability.

0	1	1	1	1	1	1	1	
NULL	START	S/D	O/S	SEL1	SEL0	UNI	MSBF	>60

Figure 3. D_{IN} Word for LTC1094

	MSB														LSB
X	X	X	X	X	X	9	8	7	6	5	4	3	2	1	0

Figure 4. D_{OUT} of LTC1094 Stored in >61 of TMS320C25

LABEL	CODE	MNEMONIC	COMMENTS
		AORG >20	
START	CE01	DINT	DISABLE INTERRUPTS
LOOP	CAFF	LACK >7F	D_{IN} FOR LTC1094
	6060	SACL >60	PUT D_{IN} IN >60
	E860	OUT >60.8	OUTPUT D_{IN} TO PORT 8
WAIT1	FA80005F	BIOZ READ	IF DONE GO TO READ
	FF800024	B WAIT1	IF NOT DONE GO TO WAIT1
		AORG >5F	
READ	8861	IN >61.8	PUT D_{OUT} IN >61

Figure 5. TMS320C25 Code for Interfacing to LTC1094