

Interfacing the LTC1290 to the 8051 MCU

Sammy Lum
Tim Rust

Introduction

This application note describes the hardware and software required for communication between the LTC1290 12-bit data acquisition system and the MCS-51 family of microcontrollers (e.g., 8051). The four wire interface is capable of completing a 12-bit conversion and transferring the data to the 8051 in 116 μ s. Configuration of the 8051 and the LTC1290 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 LTC1290. 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 LTC1290 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 LTC1290 (option 1). Alternatively, tying the clocks together saves one line that has to go between the LTC1290 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.

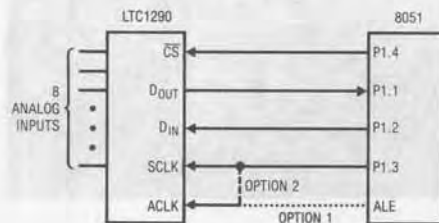


Figure 1. Schematic

Application Note 36A

Hardware Description

The code for this interface was developed on an 8051 evaluation board.

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 LTC1290 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 LTC1290 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 \$0E is then loaded into the ac-

cumulator. An examination of Figure 3 and the data sheet will show that this configures the LTC1290 for CH0 with respect to CH1, unipolar, MSB first and a 12-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 LTC1290) 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 LTC1290).

After the eight MSBs have been shifted, the contents of the accumulator are stored in R2. The final four 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 (52 ACLK cycles) for the LTC1290 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 LTC1290 52 times and this free time is then lost as shown in Figure 7. An example of this routine is labeled LOOP 1.

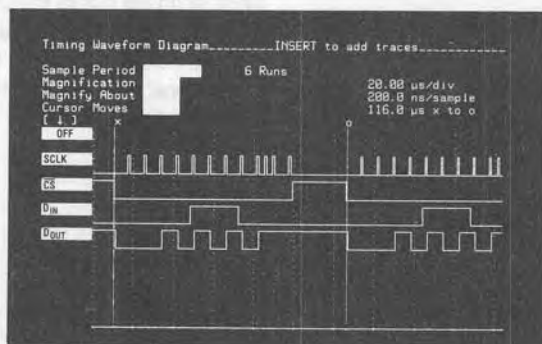


Figure 2. Timing Diagram for Option 1

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 \$0A). Also it would be necessary to swap the rotate left and rotate right instructions.

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

Figure 3. D_{IN} Word for LTC1290

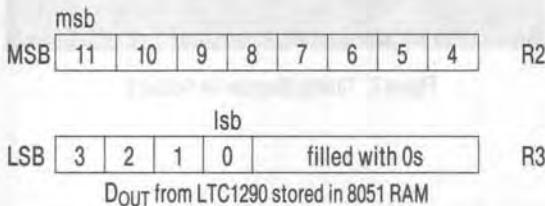


Figure 4. Memory Map

Power Shutdown

Figure 8 shows what occurs during a power shutdown and subsequent power up of the LTC1290. In ① a dummy conversion is performed prior to power shutdown in order to obtain the data from the previous conversion. In this example we are requesting a 12-bit word length MSB first. In ② power shutdown is requested by inputting the appropriate D_{IN} word (\$0D). The bottom trace is the LTC1290 supply current which shows the current going from a nominal value of 5mA to 5µA. In ③ power up occurs and the device is ready for conversion. The D_{OUT} word is not valid until the next cycle, ④. The analog input to the LTC1290 was a constant DC voltage. As one would expect the D_{OUT} word in ④ is the same as in ①.

Summary

A four wire interface between the LTC1290 and the 8051 with a combined data conversion and transfer time of 116µs was demonstrated. It was shown that the ACLK of the LTC1290 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 LTC1290 data format. The code shown applies to all MCS-51 family members. The same technique can be used on any parallel port processor.

Reference

Hoover, Guy and Rempfer, William, "Interfacing the LTC1090 to the 8051 MCU," Application Note 26A, Linear Technology Corp.

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, #0EH	D_{IN} WORD FOR LTC1290
	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 D_{IN} BIT TO LTC1290
	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
	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
	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, #0BH	LOAD COUNTER
DELAY	DJNZ R5, DELAY	GO TO DELAY IF NOT DONE

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

Application Note 36A

LABEL	MNEMONIC	COMMENTS
	MOV P1, #02H	BIT 1 PORT 1 SET AS INPUT
	CLR P1.3	SCLK GOES LOW
	SETB P1.4	\overline{CS} GOES HIGH
CONT	MOV A, #0EH	D _{IN} WORD FOR LTC1290
	CLR P1.4	\overline{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 D _{IN} BIT TO LTC1290
	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
	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
	RLC A	ROTATE DATA BIT INTO ACC
	RRC A	ROTATE RIGHT INTO ACC
	RRC A	ROTATE RIGHT INTO ACC
	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	\overline{CS} GOES HIGH
LOOP 1	MOV R4, #34H	LOAD COUNTER
	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)

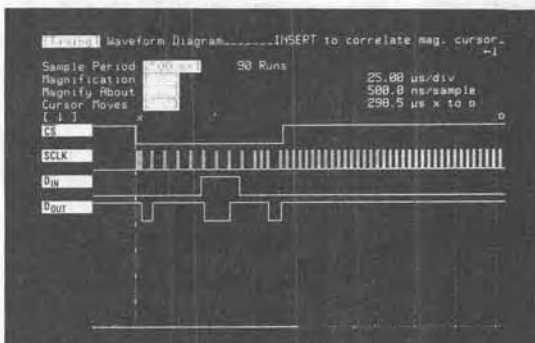


Figure 7. Timing Diagram for Option 2

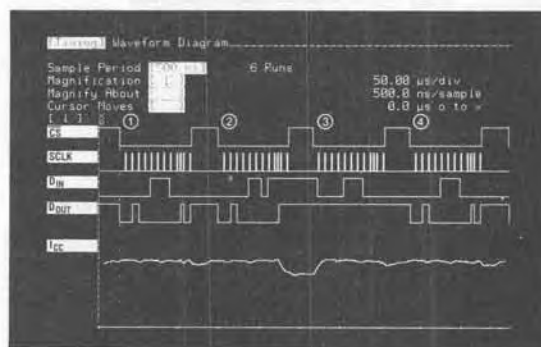


Figure 8. Power Shutdown

Interfacing the LTC1290 to the MC68HC05 MCU

Sammy Lum
 Tim Rust

Introduction

This application note describes an interface between the LTC1290 12-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 12-bit conversion and shifting the data to the 68HC05 in 40 μ s. Configuration of the LTC1290 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 LTC1290 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 LTC1290. SPI provides clock, data in and data out lines that are compatible with the LTC1290. The only additional line required is one programmable output pin (CO) to control \overline{CS} on the LTC1290. The schematic of Figure 1 shows how the two devices are connected.

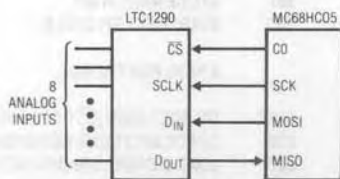
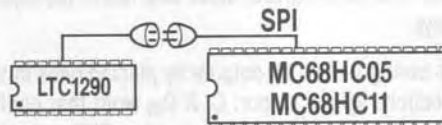


Figure 1. Schematic



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 4MHz 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 LTC1290 please see the data sheet.

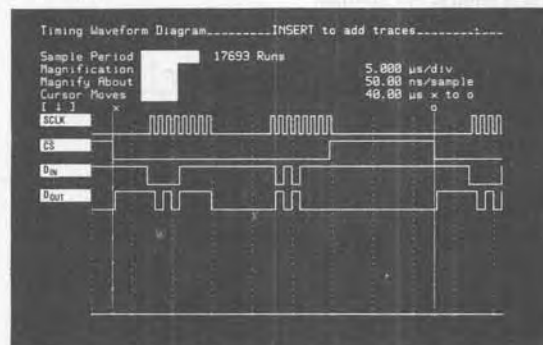


Figure 2. Timing Diagram

Software Description

The software configures and controls the SPI of the 68HC05. Additionally, the software manipulates CO (\overline{CS} of the LTC1290) and generates a delay during which time the LTC1290 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

Application Note 36B

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 LTC1290 for CH0 with respect to CH1, unipolar, MSB first and a 16-bit word length is stored in memory location \$50. Figure 3 shows how the D_{IN} word is composed.

CO is made to go low. D_{IN} for the LTC1290 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 LTC1290 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	WLO

Figure 3. D_{IN} Word for LTC1290

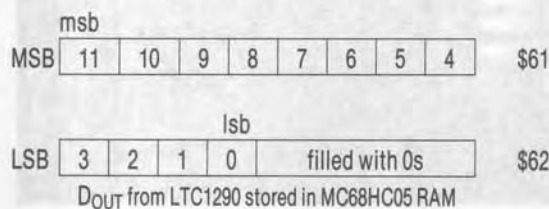


Figure 4. Memory Map

At this time 52 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.

Power Shutdown

The LTC1290 can be shutdown by inputting the appropriate D_{IN} word (0D). A dummy conversion prior to a request for power shutdown is required because the data from the previous conversion will be shifted out as a 10-bit word during the power shutdown request. Upon power up the LTC1290 is ready for conversion and the D_{OUT} word will be valid on the second request for conversion.

Summary

A four wire interface between the LTC1290 and the 68HC05 with a combined data conversion and transfer time of 40 μ s was demonstrated. The interface used the serial (SPI) port of the 68HC05. The 12 data bits of the LTC1290 are shifted MSB first in two 8-bit transfers. The data is stored left justified in the 68HC05's internal RAM.

Reference

Hoover, Guy and Rempfer, William, "Interfacing the LTC1090 to the MC68HC05," Application Note 26B, Linear Technology Corp.

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 LTC1290 D_{IN} DATA INTO ACC
	STA \$50	LOAD LTC1290 D_{IN} DATA INTO \$50
START	BCLR 0,\$02	CO GOES LOW (\overline{CS} GOES LOW)
	LDA \$50	LOAD D_{IN} INTO ACC FROM \$50
	STA \$0C	LOAD D_{IN} INTO SPI. START SCK
	NOP	8 NOPs FOR TIMING
	LDA \$0B	CHECK SPI STATUS REG
	LDA \$0C	LOAD LTC1290 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 (\overline{CS} GOES HIGH)
	LDA \$0B	CHECK SPI STATUS REGISTER
	LDA \$0C	LOAD LTC1290 LSBs INTO ACC
	STA \$62	STORE LSBs IN \$62

Figure 5. 68HC05 Code

Interfacing the LTC1290/LTC1090 to the TMS370 MCU

Guy Hoover
 Sammy Lum
 Tim Rust

Introduction

This application note describes an interface between the LTC1290 12-bit data acquisition system and the TMS370 family of microcontrollers (e.g., TMS370C050). The simple four wire interface is capable of completing a 12-bit conversion and shifting data to the TMS370C050 in 42 μ s. Configuration of the LTC1290 and the TMS370C050 will be discussed as it applies to this interface. Schematics, code, and timing diagrams will be shown. The LTC1090 10-bit data acquisition system is also compatible with this interface. Next the power shutdown feature of the LTC1290 will be discussed and a summary of the key points of this interface will be given, including data throughput rates.

Interface Details

The LTC1290 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}.

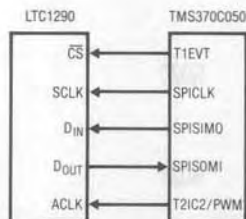
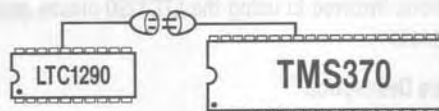


Figure 1. Schematic



The TMS370C050 has a Serial Peripheral Interface (SPI) which is a synchronous, full duplex, serial port that allows the user to construct a simple communication path to the LTC1290. SPI provides clock, data in and data out lines that are compatible with the LTC1290. The only additional line required is one programmable output pin (T1EVT) to control \overline{CS} on the LTC1290. The TMS370C050 has two on board timers and one of them can be used to provide ACLK through pin T2IC2/PWM. The schematic of Figure 1 shows how the two devices are connected.

Hardware Description

The code for this interface was developed on a TMS370 application board. The TMS370C050 was used in the micro-processor mode which requires MC (pin 6) be tied high. External memory was used to store the program.

The timing diagram of Figure 2 was obtained with an HP1631A logic analyzer using a 2.5MHz ACLK. The TMS370C050 clock was 20MHz.

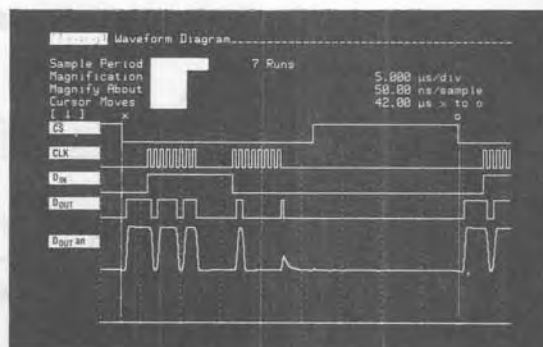


Figure 2. Timing Diagram

Application Note 36C

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 LTC1290 please see the data sheet.

Software Description

The software configures and controls the SPI of the TMS370C050. Additionally, the software manipulates T1EVT (\overline{CS} of the LTC1290), generates ACLK via pin T2IC2/PWM and generates a delay during which time the LTC1290 performs a conversion.

The System Control and Configuration Control Register 2 (SCCR2) is first configured so the system is operating in the privilege mode. Next the code configures the Timer 2 module to generate a 2.5MHz ACLK. The T2 Compare Registers are loaded with a one. This will generate an ACLK with a frequency equal to one half the internal processor clock (one fourth the crystal frequency). Next the T2IC2/PWM pin is enabled to toggle. Then the pin is configured as an output with the T2IC2/PWM function.

Next the SPI clock is enabled then the SOMI and SIMO functions are enabled. The SPI Configuration Control Register (SPICCR) is set for eight bit character length, an

SPI clock that is 1/16 of the crystal frequency, input data transfer on the rising edge, output data transfer on the falling edge and initialization of the SPI. The SPI Operation Control Register (SPICTL) is set to disable the SPI interrupt, enable transmission and place the SPI in the master mode. Finally the SPI is enabled.

The T1EVT pin on Timer Module 1 is configured as an output pin and \overline{CS} is made to go low or high by placing a "0" or "1" in the T1EVT DATA OUT bit of the Timer 1 Port Control Register 1 (T1PC1). A D_{IN} word that configures the LTC1290 for CH7 with respect to COM, unipolar, MSB first and a 16-bit word length is shown in Figure 3.

T1EVT is made to go low. D_{IN} for the LTC1290 is loaded into the serial data register (SPIDAT). Storing D_{IN} in SPIDAT causes the transfer to begin. After waiting for the first eight bits to be transferred (3 NOP's) the first eight bits containing the MSB's of the LTC1290 are read from the receive data buffer register (SPIBUF) and stored in register 20 as shown in Figure 4. The SPI INT FLAG in the SPICTL register is reset when the data is read from the SPIBUF. The LSB's are transferred in the same manner and are stored in register 21. The data at this point is stored left justified.

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

Figure 3. D_{IN} Word for LTC1290

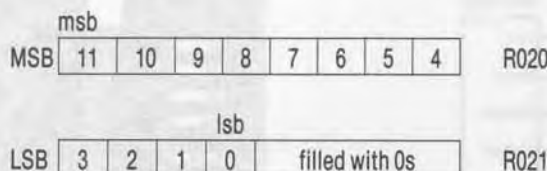


Figure 4. Memory Map

At this time 52 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 NOP's or a simple delay loop can be used to generate this delay. The code is shown in Figure 5 and a delay loop has been chosen for the A/D conversion.

LABEL	MNEMONIC	COMMENTS
	MOV #001h,P012	;CONFIGURATION DATA FOR SCCR2
		;CONFIGURE T2IC2/PWM PIN FOR ACLK
	MOV #000h,P062	;LOAD MSB DATA FOR COMPARE REG
	MOV #001h,P063	;LOAD LSB DATA FOR COMPARE REG
	MOV #070h,P06C	;CONFIGURATION DATA FOR T2CTL3
	MOV #030h,P06E	;CONFIGURATION DATA FOR T2PC2
	MOV #002h,P03D	;ENABLE SPI CLOCK
	MOV #032h,P03E	;ENABLE SOMI AND SIMO
	MOV #0CFh,P030	;CONFIGURATION DATA FOR SPICCR
	MOV #006h,P031	;CONFIGURATION DATA FOR SPICTL
	MOV #00Fh,P030	;ENABLE SPI
START	MOV #001h,P04D	;T1EVT GOES LOW (\overline{CS} GOES LOW)
	MOV #0FFh,P039	;LOAD D_{IN} INTO SPIDAT START SCLK
	NOP	;3 NOP'S FOR TIMING
	MOV P037,R020	;STORE MSB'S IN REGISTER 20
	MOV #000h,P039	;START NEXT SPI CYCLE
	NOP	;3 NOP'S FOR TIMING
	MOV P037,R021	;STORE LSB'S IN REGISTER 21
	MOV #005h,P04D	;T1EVT GOES HIGH (\overline{CS} GOES HIGH)
		;A/D CONVERSION DELAY LOOP
	MOV #002h,R023	;LOAD DATA IN DELAY COUNTER
DELAY	DEC R023	;DECREMENT COUNTER
	CMP #000h,R023	;COMPARE COUNTER WITH ZERO
	JNZ DELAY	;IF NOT ZERO RETURN TO DELAY
	JMPL START	;RETURN FOR NEXT SPI CYCLE

Figure 5. TMS370C050 Code

Power Shutdown

The TMS370C050 can be placed in one of three power reduction modes by configuring the SCCR2 register and issuing the IDLE command. Placing the TMS370C050 in the HALT mode reduces the supply current to its minimum power down value of 50 μ A. Therefore incorporating this with the power shutdown feature of the LTC1290 (typically 5 μ A in shutdown) it is possible to build a system that draws very low current when not in use. Figure 6 shows such a sequence. An external interrupt is required for the TMS370C050 to exit from the HALT mode. Then three conversion cycles are required to obtain the required data from a measurement and then enter the power shutdown mode again. The data is valid on the second conversion.

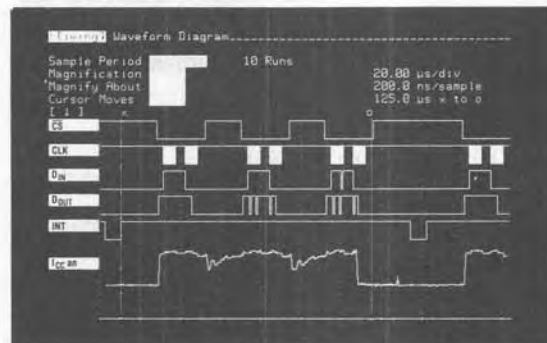


Figure 6. Power Shutdown

Application Note 36C

The third conversion is required because when power shutdown is requested ($D_{IN} = \#0FD$) the data from the previous conversion is output as a 10-bit word. The bottom trace shows the supply current for the LTC1290 during shutdown and then during conversion. Note there is no wait required for the LTC1290 to start a conversion after exiting from the shutdown mode. The time required for applying an external interrupt, making a measurement, storing the data and entering the shutdown mode is approximately 125 μ s. The code is shown in Figure 7.

LABEL	MNEMONIC	COMMENTS
.DATA	7FF8h	;SET INTERRUPT 3 VECTOR WHEN
.BYTE	70h,00h	;PROCESSOR COMES OUT OF HALT MODE
.TEXT	7000h	
		;INTERRUPT SERVICE ROUTINE EXECUTED
		;WHEN PROCESSOR COMES OUT OF HALT
		;MODE
MOV	#005h,P019	;CLEAR INTERRUPT FLAG
RTI		;RETURN FROM INTERRUPT
BEGIN		;MAIN PROGRAM
EINT		;ENABLE INTERRUPTS
MOV	#005h,P04D	;CS GOES HIGH
MOV	#001h,P012	;CONFIGURATION DATA FOR SCCR2
		;CONFIGURE T2IC2/PWM PIN FOR ACLK
MOV	#000h,P062	;LOAD MSB DATA FOR COMPARE REG.
MOV	#001h,P063	;LOAD LSB DATA FOR COMPARE REG.
MOV	#070h,P06C	;CONFIGURATION DATA FOR T2CTL3
MOV	#030h,P06E	;CONFIGURATION DATA FOR T2PC2
MOV	#002h,P03D	;ENABLE SPI CLOCK
MOV	#032h,P03E	;ENABLE SOMI AND SIMO
MOV	#0CFh,P030	;CONFIGURATION DATA FOR SPICCR
MOV	#006h,P031	;CONFIGURATION DATA FOR SPICTL
MOV	#00Fh,P030	;ENABLE SPI
START	EINT	;ENABLE INTERRUPTS
	MOV	#002h,R025 ;SET LOOP REG. TO 2 (WILL LOOP 2 TIMES)
LOOP	MOV	#001h,P04D ;T1EVT GOES LOW (\overline{CS} GOES LOW)
	MOV	#0FFh,P039 ;LOAD D_{IN} INTO SPIDAT START SCLK
	NOP	;3 NOP'S FOR TIMING
	MOV	P037,R020 ;STORE MSB'S IN REGISTER 20
	MOV	#000h,P039 ;START NEXT SPI CYCLE

Summary

A four wire interface between the LTC1290 and the TMS370C050 with a combined data conversion and data transfer rate of 42 μ s was demonstrated. The interface used the serial (SPI) port of the TMS370C050. The 12 data bits of the LTC1290 are shifted MSB first in two 8-bit transfers. The data is stored left justified in the TMS370C050's internal registers. By using the power reduction mode of the TMS370C050 and the power shutdown feature of the LTC1290 a low power system was shown to make a measurement in 125 μ s when active.

LABEL	MNEMONIC	COMMENTS
	NOP	;3 NOP'S FOR TIMING
	MOV	P037,R021 ;STORE LSB'S IN REGISTER 21
	MOV	#005,P04D ;T1EVT GOES HIGH (\overline{CS} GOES HIGH)
		;A/D CONVERSION DELAY LOOP
	MOV	#002h,R023 ;LOAD DATA IN DELAY COUNTER
DELAY	DEC	R023 ;DECREMENT COUNTER
	CMP	#000h,R023 ;COMPARE COUNTER WITH ZERO
	JNZ	DELAY ;IF NOT ZERO RETURN TO DELAY
	DJNZ	R025,LOOP ;RETURN FOR NEXT CONVERSION
		;START POWER SHUTDOWN CONVERSION
	MOV	#001h,P04D ;T1EVT GOES LOW (\overline{CS} GOES LOW)
	MOV	#0FDh,P039 ;LOAD D_{IN} FOR POWER SHUTDOWN
	NOP	;3 NOP'S FOR TIMING
	MOV	P037,R022 ;CLEAR SPIBUF
	MOV	#000,P039 ;START NEXT SPI CYCLE
	NOP	;3 NOP'S FOR TIMING
	MOV	P027,R022 ;CLEAR SPIBUF
	MOV	#005h,P04D ;T1EVT GOES HIGH (\overline{CS} GOES HIGH)
	MOV	#0C0h,P012 ;PUT TMS370C050 IN HALT MODE
	IDLE	
	JMPL	START ;AFTER EXTERNAL INTERRUPT JUMP TO ;START

Figure 7. Power Shutdown Code

Interfacing the LTC1290 to the COP820C MCU

Sammy Lum
Tim Rust

Introduction

This application note describes the hardware and software required for communication between the LTC1290 12-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 12-bit conversion and shifting the data in $37\mu\text{s}$. Configuration of the LTC1290 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 LTC1290 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 LTC1290. MICROWIRE/PLUS provides clock, data in and data out lines that are compatible with the LTC1290. One additional line (G1) is required to control the \overline{CS} pin on the LTC1290. The schematic of Figure 1 shows how the two devices are connected.

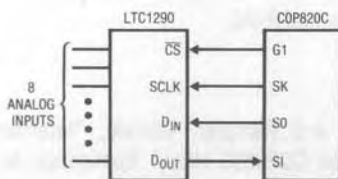
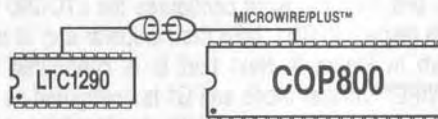


Figure 1. Schematic



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 using a 4MHz ACLK. The COP820C clock was 10MHz. To obtain a $37\mu\text{s}$ 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 LTC1290 please see the data sheet.

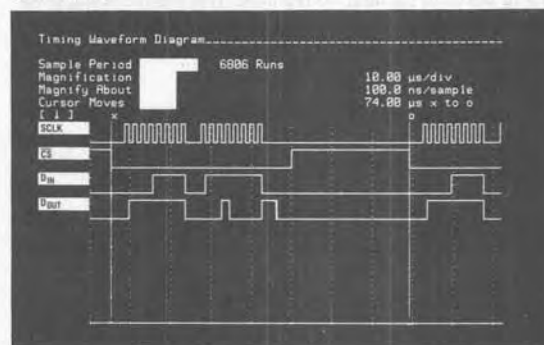


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE/PLUS serial interface of the COP820C. Additionally, the software manipulates G1 (\overline{CS} of the LTC1290) and generates a delay during which time the LTC1290 performs a conversion.

Application Note 36D

The code first loads the LTC1290 D_{IN} word into memory location \$F0. This D_{IN} word configures the LTC1290 for CH0 with respect to CH1, MSB first, unipolar and 16 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 LTC1290 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 LTC1290 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. G1 (\overline{CS}) is set and the contents of the MICROWIRE shift register are swapped with those 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.

52 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.

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

Figure 3. D_{IN} Word for LTC1290

								MSB	
MSB	11	10	9	8	7	6	5	4	F3
								LSB	
LSB	3	2	1	0	0	0	0	0	F4

D_{OUT} from LTC1290 stored in COP820C RAM

Figure 4. Memory Map

LABEL	MNEMONIC	COMMENTS
	LD (F0)—0F	LOAD 0F 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
	NOP	15 NOPs FOR TIMING
	X (A)—(F3)	LOAD D_{OUT} IN ADDR F3
	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
	X (A)—(F4)	LOAD D_{OUT} IN ADDR F4

Figure 5. COP820C Code

Power Shutdown

The LTC1290 can be shutdown by inputting the appropriate D_{IN} word (0D). A dummy conversion prior to a request for power shutdown is required because the data from the previous conversion will be shifted out as a 10-bit word during the power shutdown request. Upon power up the LTC1290 is ready for conversion and the D_{OUT} word will be valid on the second request for conversion.

Summary

A four wire interface between the LTC1290 and the COP820C with a combined data conversion and transfer time of 37 μ s was demonstrated. The interface used the MICROWIRE/PLUS serial port of the COP820C. The 16 data bits of the LTC1290 are shifted MSB first in two 8-bit transfers. The data is stored left justified in the COP820C's internal RAM.

Reference

Hoover, Guy and Rempfer, William, "Interfacing the LTC1090 to the COP820C MCU," Application Note 26D, Linear Technology Corp.

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

Interfacing the LTC1290 to the TMS7742 MCU

Sammy Lum
Tim Rust

Introduction

This application note describes an interface between the LTC1290 12-bit data acquisition system and the TMS7000 family of microcomputers (e.g., TMS7742). The simple four wire interface is capable of completing a 12-bit conversion and shifting the data to the TMS7742 in 102 μ s. Configuration of the LTC1290 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 LTC1290 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 LTC1290. The serial port provides clock, transmit, and receive lines that are compatible with the LTC1290. The only additional line required is one programmable output pin (A0) to control \overline{CS} on the LTC1290. 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

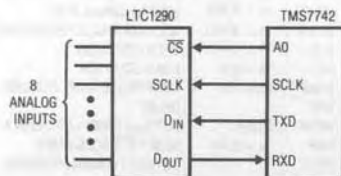
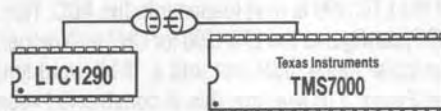


Figure 1. Schematic



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 LTC1290 was 4MHz and the TMS7742 clock was 5MHz.

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 LTC1290 please see the data sheet.

Software Description

The software configures and controls the serial port of the TMS7742. Additionally, the software manipulates A0 (\overline{CS} of the LTC1290) and generates a delay during which time the LTC1290 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 eight bits, no parity, and one stop bit. The SCLK rate is

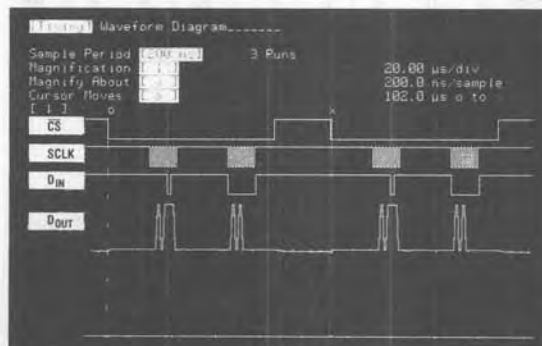


Figure 2. Timing Diagram

Application Note 36E

set to the processor clock frequency divided by 4. The D_{IN} word of the LTC1290 is next loaded into the ACC. This D_{IN} word (\$DF) configures the LTC1290 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 LTC1290 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 four 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 52 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. Branching back to the label LOOP starts the next conversion.

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

Figure 3. D_{IN} Word for LTC1290 Stored in TMS7742 RAM

								LSB	
LSB	7	6	5	4	3	2	1	0	R5
fill with zeroes				MSB					
MSB	0	0	0	0	11	10	9	8	R6

D_{OUT} from LTC1290 stored in TMS7742 RAM

Figure 4. Memory Map

Power Shutdown

The LTC1290 can be shutdown by inputting the appropriate D_{IN} word (9F). A dummy conversion prior to a request for power shutdown is required because the data from the previous conversion will be shifted out as a 10-bit word during the power shutdown request. Upon power up the LTC1290 is ready for conversion and the D_{OUT} word will be valid on the second request for conversion.

Summary

A four wire interface between the LTC1290 and the TMS7742 with a combined data conversion and transfer time of 102 μ s 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 LTC1290. The 12 data bits of the LTC1290 are shifted LSB-first in two 8-bit transfers. The data is stored right justified in the TMS7742's internal RAM.

Reference

Hoover, Guy and Rempfer, William, "Interfacing the LTC1090 to the TMS7742 MCU," Application Note 26E, Linear Technology Corp.

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 LTC1290 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
	BR @LOOP	NEXT CONVERSION
SXTNBIT	ANDP % > FE, P4	A0 CLEARED (\overline{CS} GOES LOW)
	MOVP A, P23	PUT LTC1290 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 LTC1290 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 LTC1290 IN A
	ORP % > 01, P4	A0 SET (\overline{CS} GOES HIGH)
	RETS	RETURN TO MAIN PROGRAM

Figure 5. TMS7742 Code

Interfacing the LTC1290 to the COP402N MCU

Sammy Lum
 Tim Rust

Introduction

This application note describes the hardware and software required for communication between the LTC1290 12-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 12-bit conversion and shifting the data in 100 μ s. Configuration of the LTC1290 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 LTC1290 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 LTC1290. MICROWIRE provides clock, data in and data out lines that are compatible with the LTC1290. One additional line (G0) is required to control the \overline{CS} pin on the LTC1290. The schematic of Figure 1 shows how the two devices are connected.

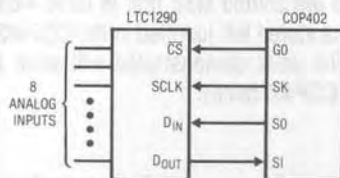
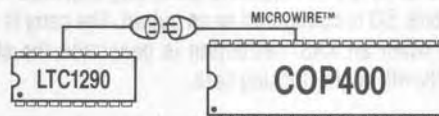


Figure 1. Schematic



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 4MHz 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 LTC1290 please see the data sheet.

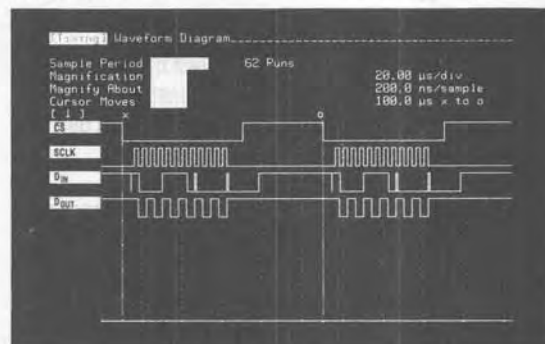


Figure 2. Timing Diagram

Software Description

The software configures and controls the MICROWIRE serial interface of the COP402N. Additionally, the software manipulates G0 (\overline{CS} of the LTC1290) and generates a delay during which time the LTC1290 performs a conversion.

The code first initializes the B register and then loads the LTC1290 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

Application Note 36F

LTC1290 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 LTC1290 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 LTC1290 D_{OUT} word containing the LSBs 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 LTC1290 is now in a left justified format as shown in Figure 4.

52 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.

\$10				\$11			
1	0	0	0	1	1	1	0
S/D	O/S	S1	S2	UNI	MSBF	WL1	WLO

Figure 3. D_{IN} Word for LTC1290

MSB				
11	10	9	8	\$13
7	6	5	4	\$14
LSB				
3	2	1	0	\$15

D_{OUT} from LTC1290 stored in COP402 RAM

Figure 4. Memory Map

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	0 $G0$ (\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	1 $G0$ (\overline{CS}) SET
	XIS	0 THIRD NIBBLE D_{OUT} TO \$15
	LBI	1,3 SET B REG. FOR NEXT LOOP

Figure 5. COP402 Code

Power Shutdown

The LTC1290 can be shutdown by inputting the appropriate D_{IN} word (D for the second nibble). A dummy conversion prior to a request for power shutdown is required because the data from the previous conversion will be shifted out as a 10-bit word during the power shutdown request. Upon power up the LTC1290 is ready for conversion and the D_{OUT} word will be valid on the second request for conversion.

Summary

A four wire interface between the LTC1290 and the COP402N with a combined data conversion and transfer time of 100 μ s was demonstrated. The interface used the MICROWIRE serial port of the COP402N. The 12 data bits of the LTC1290 are shifted MSB first in three 4-bit transfers. The data is stored left justified in the COP402N's internal RAM. The code demonstrated will work on any member of the COP400 family.

Reference

Hoover, Guy and Rempfer, William, "Interfacing the LTC1090 to the COP402N MCU," Application Note 26F, Linear Technology Corp.

MICROWIRE is a trademark of National Semiconductor Corp.

Interfacing the LTC1290 to the Z-80 MPU

Sammy Lum
Tim Rust

Introduction

This application note describes an interface between the LTC1290 12-bit data acquisition system and the Z-80 microcomputer. The interface is capable of completing a 12-bit conversion and shifting the data to Z-80 in 260 μ s. Configuration of the LTC1290 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 LTC1290 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.

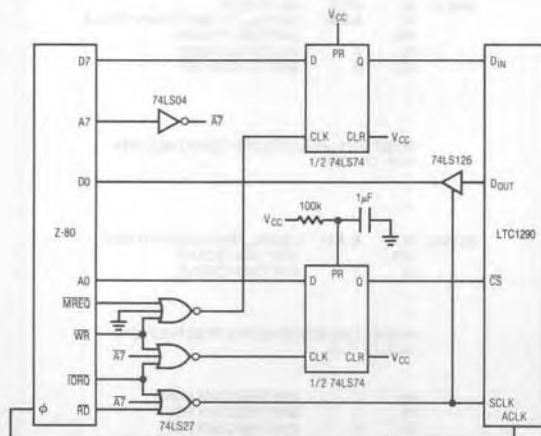
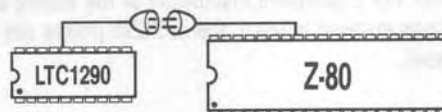


Figure 1. Serial Interface Requires Four 74LS Chips



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 LTC1290 SCLK is generated by reading from an I/O port that has an address greater than 128. Data is clocked into the LTC1290 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 LTC1290 is fed into D0 of the Z-80 through the 74LS126. The 74LS126 prevents the LTC1290 from writing to the data bus of the Z-80 except when the microprocessor requires data from the A/D. The ACLK of the LTC1290 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 100 μ s. This would require generating ACLK externally or dividing down the ϕ signal.

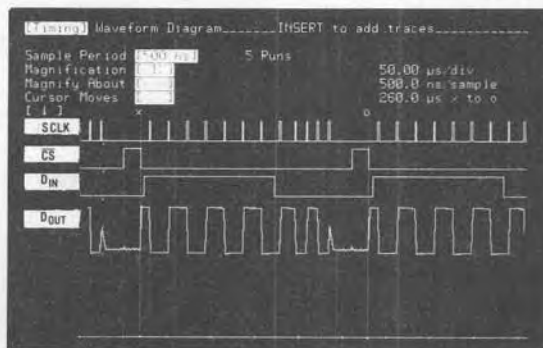


Figure 2. Throughput Time is Limited by the Z-80 MPU. A 12-Bit Conversion Result is Transmitted Every 260 μ s.

Application Note 360

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 LTC1290 please see the data sheet.

Software Description

The software serially shifts the D_{IN} configuration word to the LTC1290 while simultaneously reading the previous data back. Additionally, the software waits while the LTC1290 performs its next conversion before attempting the next data exchange cycle.

The Z-80 code is shown in Figure 5. First the C register is cleared. The D register is loaded with the D_{IN} word (FEH) for the LTC1290. This word as shown in Figure 3 configures the input MUX of the LTC1290 to accept a signal on CH7 with respect to COM, perform a unipolar conversion and shift the data out MSB-first as a 12-bit word. Next \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 the output on bit 7 of the data bus of the Z-80. The first bit of the LTC1290 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. This process is repeated seven more times until the D_{IN} bits have been shifted out and the 8 MSBs of the D_{OUT} word have been shifted into the B register. The last four bits of the D_{OUT} word are then shifted through the carry bit into the LSB position of the C register. Then it is rotated right through the carry bit into the four MSB positions of the C register. \overline{CS} is then brought high. The 12-bit D_{OUT} word is now stored left justified in the Z-80 as shown in Figure 4.

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

Figure 3. D_{IN} Word for LTC1290 Stored in D Register of Z-80

MSB

11	10	9	8	7	6	5	4	REG B
----	----	---	---	---	---	---	---	-------

LSB

3	2	1	0	FILLED WITH ZEROS	REG C
---	---	---	---	-------------------	-------

D_{OUT} from LTC1290 stored in Z-80 registers

Figure 4. Memory Map of Z-80

After the last SCLK pulse is ended, 52 ACLK cycles must be allowed for the LTC1290 to perform the desired A/D conversion. During this time \overline{CS} is taken high. The software must ensure that this occurs.

Power Shutdown

The LTC1290 can be shutdown by inputting the appropriate D_{IN} word (FDH). A dummy conversion prior to a request for power shutdown is required because the data from the previous conversion will be shifted out as a 10-bit word during the power shutdown request. Upon power up, the LTC1290 is ready for conversion and the D_{OUT} word will be valid on the second request for conversion.

Summary

An interface between the LTC1290 12-bit data acquisition system and the Z-80 microprocessor with a combined data conversion and transfer time of 260 μ s was demonstrated. The interface used four 74LS chips to interface the two devices. The 12 data bits of the LTC1290 are shifted MSB-first one bit at a time. The data is stored left justified in the Z-80's internal registers.

Reference

Hoover, Guy, "Interfacing the LTC1090 to the Z-80," Application Note 260, Linear Technology Corp.

LABEL	MNEMONIC	COMMENTS
BEGIN	LD C,00H	INITIALIZE REG C
	LD D,FE	LOAD D_{IN} IN REG D
	OUT (80H),A	\overline{CS} GOES LOW
BLOCK1	LD IN (HL),D	OUTPUT D_{IN} BIT
	IN A,(80H)	READ D_{OUT} BIT AND OUTPUT ONE SCLK
	RRA	SHIFT DATA TO CARRY
	RL B	SHIFT DATA TO REG B
	RLC D	SHIFT D_{IN} WORD LEFT
	.	.
	.	.
	.	REPEAT CODE LABELLED BLOCK1 SEVEN TIMES FOR A TOTAL OF EIGHT
	.	.
	.	.
BLOCK2	IN IN A,(80H)	READ D_{OUT} BIT AND OUTPUT ONE SCLK
	RRA	SHIFT DATA TO CARRY
	RL C	SHIFT DATA TO REG C
	.	.
	.	.
	.	REPEAT CODE LABELLED BLOCK2 THREE TIMES FOR A TOTAL OF FOUR
	.	.
	.	.
	RR C	SHIFT REG C DATA RIGHT
	RR C	SHIFT REG C DATA RIGHT
	RR C	SHIFT REG C DATA RIGHT
	RR C	SHIFT REG C DATA RIGHT
	OUT (81H)	\overline{CS} GOES HIGH

Figure 5. Z-80 Code