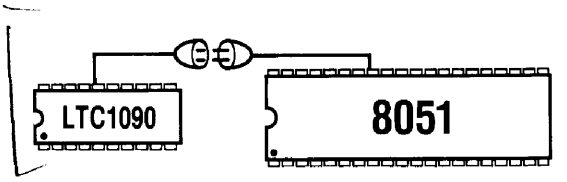
Interfacing the LTC1090 to the 8051 MCU



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.

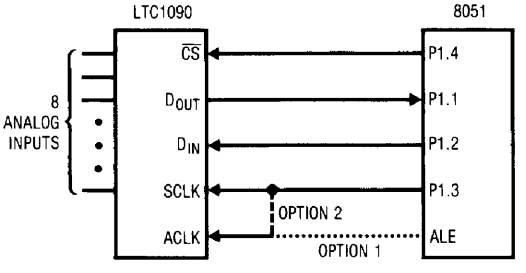


Figure 1. Schematic

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.

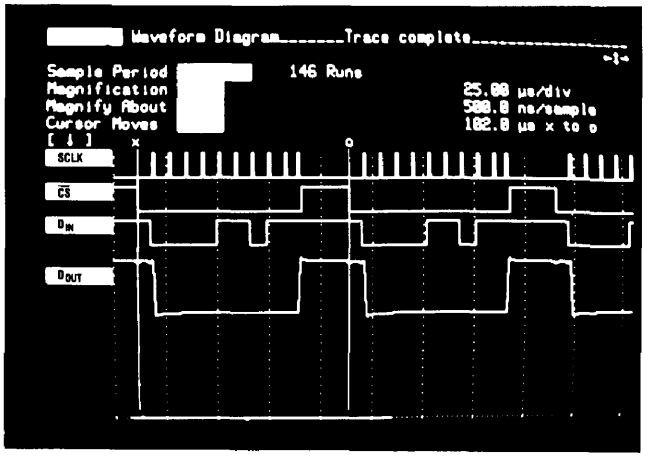


Figure 2. Timing Diagram for Option 1

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

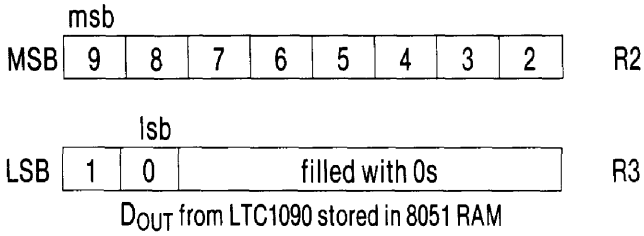


Figure 4. Memory Map

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	$\overline{CS}$ GOES HIGH
CONT	MOV	A, #0DH	D_{IN} WORD FOR LTC1090
	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 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	$\overline{CS}$ 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)

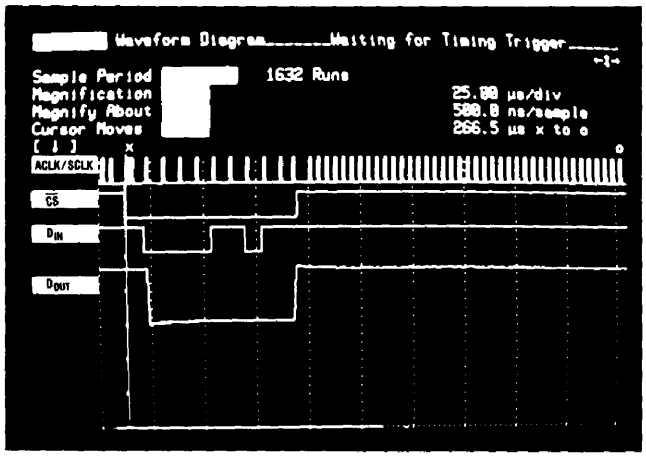


Figure 7. Timing Diagram for Option 2