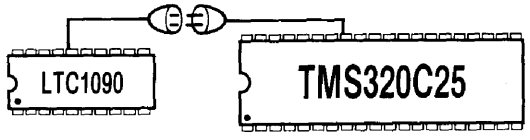
# Interfacing the LTC1090 to the TMS320C25 DSP




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 32$\mu$s. 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.

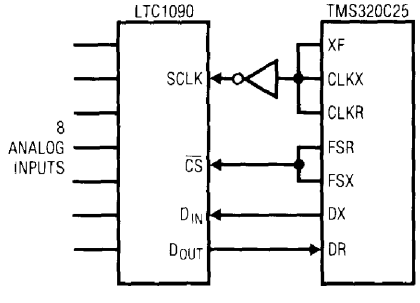



Figure 1. Circuit 1: Minimum Hardware Interface

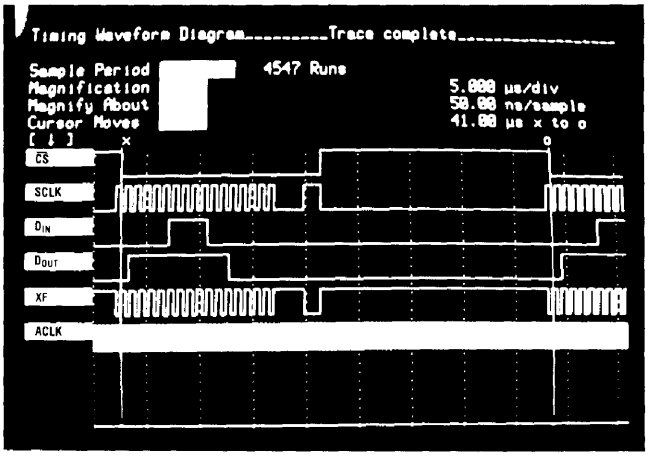



Figure 2. Timing for Circuit 1 Shows 41$\mu$s Throughput Rate

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 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    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 DOUT 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 0.8$\mu$s 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 41$\mu$s. 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 32$\mu$s. 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**

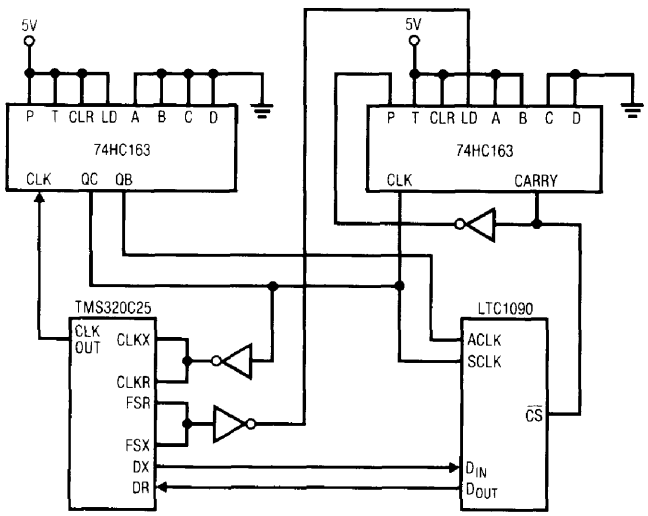



**Figure 7. Circuit 2: Minimum Software Interface**

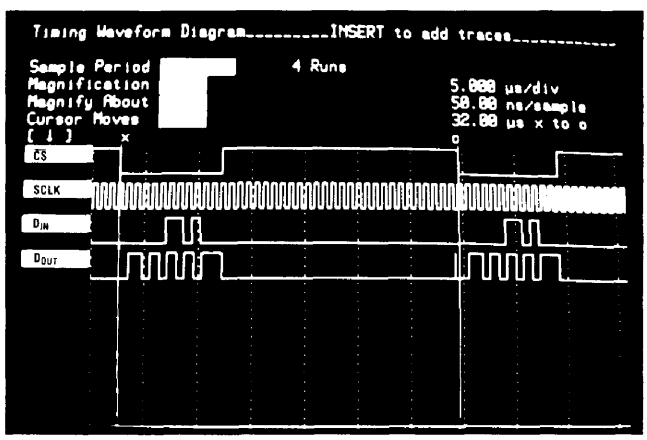
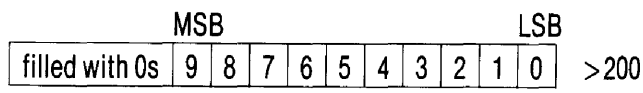



**Figure 8. Timing for Circuit 2 shows 32µs Throughput Rate**

| 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 $D_{IN}$ 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 | B | TIMER | LOOP UNTIL FINISHED |
| RINT | ZALS | >0 | STORE $D_{OUT}$ 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**




$D_{OUT}$ from LTC1090 stored in TMS320C25 RAM

**Figure 10. Memory Map for Circuit 2**