Interfacing the HI7188 to a Microcontroller

Application Note | April 1996 | AN9610

Authors: John Kornblum and Stephen LaJeunesse

Introduction

The Intersil HI7188 is a monolithic eight channel 16-bit sigma delta instrumentation A/D subsystem suitable for applications such as Process Control and Measurement, Industrial Weigh Scales, Motion Control and Medical Equipment. The HI7188 serial I/O port is compatible with most serial bus protocols including the Motorola 6805/11 series SPI and Intel 8051 series SSR busses. This application note discusses the HI7188 Serial Interface Port and details two application circuits and pseudo code useful in demonstrating how to interface the HI7188 to a microcontroller. For further information review the HI7188 product data sheet, file number 4016.

HI7188 Serial Port Signals

The HI7188 Sigma-Delta A/D converter communicates to a controlling device over either a 2-wire or 3-wire serial interface. Data is transmitted or received synchronous to a port clock. The Serial Clock (SCLK) line is the synchronous data clock used to strobe the serial data in or out of the HI7188 A/D converter. The serial clock can be generated by the converter or can be supplied to the converter. When the HI7188 is the clock master, that mode is referred to as the Self Clocking mode. When the HI7188 is a clock slave, that mode is referred to as the External Clocking mode. The serial port also contains a status flag (End of Scan, $\overline{EOS}$) that signals a controller that the HI7188 has completed a conversion scan and the results are now available for reading from the device. The End of Scan flag is cleared by reading the data out of the HI7188's Data RAM.

The Chip Select ($\overline{CS}$) line allows multiple devices to reside on the serial bus and each in turn to be selected individually. The HI7188 is selected, enabling an I/O operation, whenever the $\overline{CS}$ input is asserted low.

The HI7188 has 2 data lines that can be used with 2-wire or 3-wire serial bus interfaces. The Serial Data I/O (SDIO) line is a bidirectional data line that can be used as a dedicated input or a bidirectional data path. The Serial Data Out (SDO) line is a dedicated output pin for use in 3-wire interfaces where there must be a separate path for data in and data out. In a 2-wire interface, such as that used with Intel microcontrollers, the SDIO line is used exclusively for bidirectional data transfers.

In addition to the SCLK, $\overline{CS}$ and data lines, the HI7188 has a serial interface reset function via the $\overline{IORST}$ pin. The $\overline{IORST}$ pin, when active low resets the serial interface state machine. The $\overline{IORST}$ input is useful for re-synchronizing the HI7188 serial interface to the microcontroller if synchronization ever becomes lost. Chip configuration and conversion operation is not affected by applying an $\overline{IORST}$ to the HI7188.

HI7188 Serial Protocol

When communicating with the HI7188 a specific sequence must be satisfied. During the first phase of a transfer an instruction byte must be written to the device. The instruction byte provides the HI7188 with information regarding the data transfer in phase 2 of the communication cycle. A typical communication cycle would involve an Instruction Cycle and a Data Cycle as shown in Figure 1.

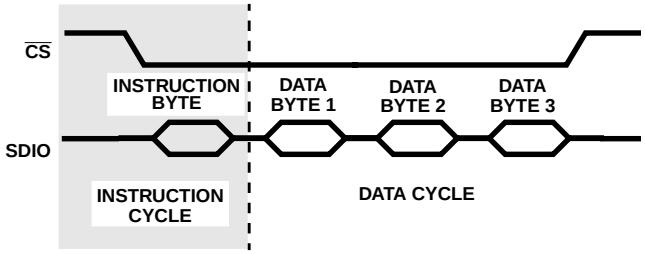

FIGURE 1. HI7188 COMMUNICATION CYCLE

The instruction byte allows access to the following storage elements internal to the HI7188:

Control Register
Channel Configuration Register #2
Channel Configuration Register #1
Data RAM
Offset Calibration RAM
Positive Full Scale Calibration RAM
Negative Full Scale Calibration RAM

Interfacing to RAMs Versus Registers

Accessing RAM storage is different than accessing registers on the HI7188. The fundamental difference is that the RAM is accessible only as a total entity, whereas the number of bytes transferred to/from a register is contained in the IR byte.

When accessing a RAM, the internal address generator will increment through the entire active RAM block starting at the address location corresponding to the first logical channel and ending with the last active logical channel. The number of active channels is set by the CR<7:5> bits, which therefore determine the number of bytes transferred during a RAM I/O access. For example, if the Offset Calibration RAM is the source/destination and the chip is configured to convert on 4 logical channels, 12 data bytes will be transferred, since an Offset Calibration RAM word is 3 bytes. With the same configuration, reading the Data RAM would transfer 8 bytes of data, since a Data RAM word is only 2 bytes.

Instruction Byte

The instruction byte is organized as follows:

MSB	6	5	4	3	2	1	LSB
$\overline{R}$/W	NB1	NB0	RB	A3	A2	A1	A0

$\overline{R}$/W - Bit 7 of the Instruction Byte determines whether phase 2 of the communication cycle will be a read or write operation. If $\overline{R}$/W is logic 1, a write transfer will occur in phase 2 of the communication cycle. If $\overline{R}$/W is logic 0, a read transfer will occur in phase 2 of the communication cycle.

NB1, NB0 - Bits 6 and 5 of the Instruction Byte determine the number of bytes that will be transferred during phase 2 of the communication cycle if a register is selected for I/O access. If a RAM is selected for I/O access, these bits are ignored. Any number of bytes from 1 to 4 is allowed. See the HI7188 Data Sheet for specific bit decodes.

RB - Bit 4 is used to determine the byte order when accessing a RAM address. When accessing a RAM address, if RB = 1, the data format is most significant byte first to least significant byte (descending). When accessing a RAM address, if RB = 0, the data format is least significant byte first to most significant byte (ascending). When accessing a register address, this bit is ignored.

A3, A2, A1, A0 - Bits 3 and 2 (A3 and A2) of the Instruction Byte determine which of the three internal registers will be accessed or if both bits are set (11b), that a RAM access is active. For register addresses, bits 1 and 0 (A1 and A0) determine which byte of that register will be accessed first. For RAM access (A3 = 1, A2 = 1), bits 1 and 0 (A1 and A0) determine which RAM is the source or destination. See the HI7188 data sheet for complete addressing information.

TABLE 2. INTERNAL DATA ACCESS DECODE STARTING BYTE

RB	A3	A2	A1	A0	DESCRIPTION
x	0	0	0	0	Control Register Byte0
x	0	0	0	1	Control Register Byte1
x	1	0	0	0	CCR #2 Byte0
x	1	0	0	1	CCR #2 Byte1
x	1	0	1	0	CCR #2 Byte2
x	1	0	1	1	CCR #2 Byte 3
x	0	1	0	0	CCR #1 Byte0
x	0	1	0	1	CCR #1 Byte1
x	0	1	1	0	CCR #1 Byte2
x	0	1	1	1	CCR #1 Byte3
0	1	1	0	0	Data RAM, least significant byte first, READ ONLY
1	1	1	0	0	Data RAM, most significant byte first, READ ONLY
0	1	1	0	1	Offset Calibration RAM, least significant byte first
1	1	1	0	1	Offset Calibration RAM, most significant byte first
0	1	1	1	0	Positive Gain Calibration RAM, least significant byte first
1	1	1	1	0	Positive Gain Calibration RAM, most significant byte first
0	1	1	1	1	Negative Gain Calibration RAM, least significant byte first
1	1	1	1	1	Negative Gain Calibration RAM, most significant byte first

TABLE 2. INTERNAL DATA ACCESS DECODE STARTING BYTE (Continued)

Interfacing to the 8X51 SSR Protocol

The HI7188 can interface to microcontrollers that use a 2 or 3-wire serial hardware interface. A 2-wire interface involves a tightly coupled system where a single converter is connected to a single microcontroller. In this mode only the Serial Clock Line (SCLK) and the Bidirectional Data Line (SDIO) are used to communicate between the A/D and the microcontroller. Figure 2 shows a 2-wire interface to an 8X51 style microcontroller.

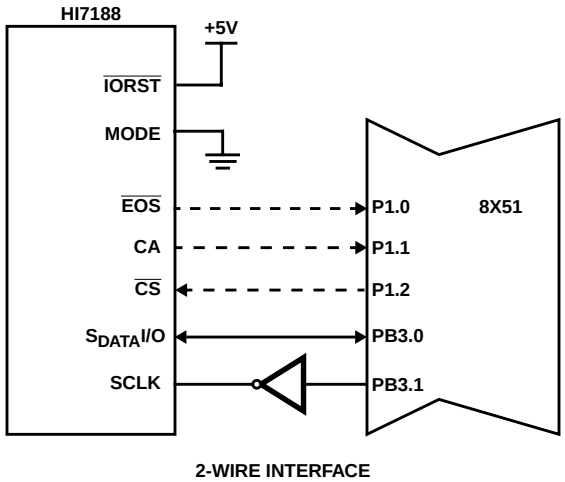

FIGURE 2. HI7188 INTERFACE TO 8X51

8051 Setup

Mode 0 of the 8X51 uses RXD (Port 3 line 0) as the data port and TXD (Port 3 line 1) as the shift clock. Data is shifted with LSB being the first bit in the sequence. The baud rate is fixed to 1/12 the microcontroller oscillator frequency. The 8X51 is the serial shift clock master, therefore the HI7188 is placed in external clocking mode by grounding the MODE pin. The HI7188 can be setup in polled mode where the status of the $\overline{EOS}$ line is read into the 8X51. When $\overline{EOS}$ is low, the HI7188 is ready to be accessed. In a multi-converter application the $\overline{CS}$ line can be used to address each individual A/D in the system. In a single converter application the $\overline{CS}$ may be grounded and an access is started by writing the instruction byte. The HI7188 should be reset to ensure proper power-up state. On power-up the HI7188 is configured for MSB first transfers and descending byte mode.

Since the 8X51 Intel Microcontrollers use an ascending or little endian data structure, the HI7188 should be programmed for LSB first and ascending byte mode. Ascending byte mode will sequence through multiple bytes from least significant byte to most significant byte. The HI7188 expects data to be valid for the rising edge of the shift clock and changed

on the falling edge of the shift clock. The 8X51 microcontroller expects just the opposite, so an inverter can be used on the serial clock line if the user wants to maintain approximately 1/2 clock cycle setup and hold times at the HI7188. Eliminating the inverter would give approximately a full clock cycle of setup time and zero hold time. The HI7188 will work in either design. Figure 3 shows the HI7188 port timing.

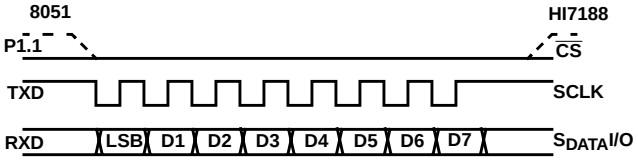

FIGURE 3A. DATA SEND/WRITE HI7188

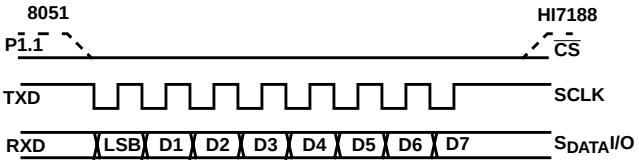

FIGURE 3B. DATA RECEIVE/READ HI7188

FIGURE 3. HI7188 SERIAL PORT TIMING

Programming the HI7188 with the 8X51

The serial port of the 8X51 and the HI7188 need to be configured after power-up or a hardware reset. The HI7188 Control Register must be set to comply with the 8X51 data format. Also, the Control Register is written to configure other aspects of the HI7188, such as number of channels to convert on, enabling the line noise rejection function, etc. The following program initializes the 8X51 serial port and HI7188. Data is read in a polled fashion instead of interrupt driven.

8X51Microcode Example

```
;Power-up/Reset, Port initialization
;Set-Up Port 1 for reading status bits
;Polled Data, (No interrupts)
;Set P1.0 for End of Scan (EOS)
;Set P1.1 for Calibration Active (CA)
;Set P1.2 for Chip Select (CS)
SSRINIT:  SCON, #0000 0000B;
          SETB   93H; Configure Port 1
          CLRB   P3.1; Deassert Clock Line
          CLRB   P1.2: CS Active
          MOV    R2,#001H; EOS Mask Value
          MOV    R3,#002H; CA Mask Value

;SSR Clock Subroutine runs the serial clock.
;Inactive high and asserted low with inverter on SCLK
SSRCLK    MOV    R4, #008H: Reset Clock Count
CLKLOOP   SETB   P3.1; Assert Clock Line
          DEC    R4; Decrement Clock Count
          CLRB   P3.1; Deassert Clock Line
          JNZ    CLKLOOP; Another clock if not finished
          RET

;Configure the HI7188 for:
;Ascending byte direction, LSB first
;Convert on 8 logical channels
;Suppress EOS during calibration
;Enable 60 Hz line noise rejection
Note: HI7188 expects MSB first format until after ADINIT is
complete.
ADINIT:   MOV    SBUF, # 10000101;  IR byte for CR
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, # 00110000;  Write CR<15:8>
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, # 01100111;  Write CR<7:0>
          CALL   SSRCLK;            Serial Clock

;Write Channel Configuration Registers for:
;8 unique physical channels,
;Offset Calibration Mode, gain of 1.
OFFCAL    MOV    SBUF, #11101000;   IR byte for CCR#2
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #01110100;   logical channel 4
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #01010100;   logical channel 3
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #00110100;   logical channel 2
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #00010100;   logical channel 1
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #11100100;   IR byte for CCR#1
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #11110100;   logical channel 8
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #11010100;   logical channel 7
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #10110100;   logical channel 6
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #10010100;   logical channel 5
          CALL   SSRCLK;            Serial Clock

;Test Calibrate Active output. If low, offset calibration is
;complete for all 8 channels, jump to POSCAL code
;segment
OCAL_DN   MOV A,P1;
          ANL A,R3;
          JZ POSCAL;
          SJMP OCAL_DN;

;Write Channel Configuration Registers as
;before except write mode bits for positive
;full scale calibration.
POSCAL    MOV    SBUF, #11101000;   IR byte for CCR#2
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #01111000;   logical channel 4
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #01011000;   logical channel 3
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #00111000;   logical channel 2
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #00011000;   logical channel 1
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #11100100;   IR byte for CCR#1
          CALL   SSRCLK;            Serial Clock
          MOV    SBUF, #11111000;   logical channel 8
          CALL   SSRCLK;            Serial Clock
```

```
                MOV     SBUF, #11011000;    logical channel 7
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #10111000;    logical channel 6
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #10011000;    logical channel 5
                CALL    SSRCLK;             Serial Clock

;Test Calibrate Active output. If low, positive gain calibration
;is complete for all 8 channels, jump to NEGCAL code
;segment
PCAL_DN         MOV A,P1;
                ANL A,R3;
                JZ NEGCAL;
                SJMP PCAL_DN;

;Write Channel Configuration Registers as
;before except write mode bits for negative
;full scale calibration.
NEGCAL          MOV     SBUF, #11101000;    IR byte for CCR#2
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #01111100;    logical channel 4
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #01011100;    logical channel 3
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #00111100;    logical channel 2
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #00011100;    logical channel 1
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #11100100;    IR byte for CCR#1
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #11111100;    logical channel 8
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #11011100;    logical channel 7
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #10111100;    logical channel 6
                CALL    SSRCLK;             Serial Clock
                MOV     SBUF, #10011100;    logical channel 5
                CALL    SSRCLK;             Serial Clock
;Test Calibrate Active output. If low, negative gain calibration
;is complete for all 8 channels completing full calibration
;of the HI7188.
POL_N_DNMOV A,P1;
                ANL A,R3;
                JZ CALDONE;
                SJMP POL_N_DN;
CALDONE         NOP

;Poll EOS Signal for End of Scan and read 8 words of data.
ADRUN:          MOV     R1, #007H;
                MOV     R0, START_ADDRESS;
                MOV     R5, DATA_STREAM_SIZE;
POLL_EOS        MOV     A, P1;
                ANL     A,R2;
                JZ      READ_DATA;
                SJMP    POLL_EOS;
READ_DATA       SETB    P1.0;
                MOV     SBUF,#00001100;IR data RAM read
                CALL    SSRCLK; Serial Clock
DATA_LOOP       MOV     A,SBUF;
                MOV     @R0, A;
                INC     R0;
                DEC     R1;
                JZ      DATA_LOOP;
                DEC     R5;
                JZ      FINISHED;
                SJMP    POLL_EOS;
                RET;
```

The initialization routine (SSRINIT) configures the Serial Port in Mode 0 operation where the shift clock is generated by the SSRCLK subroutine under Program Control. The baud rate should not exceed the HI7188's specification of 5Mbps. Port 1 bit 2 is the control bit for Chip Select that enables the HI7188's serial port.

The SSRCLK module generates 8 low to high edges on the HI7188 SCLK line. For this example, it is intended that the HI7188 SCLK stall high between byte transfers. This is true if the inverter shown in Figure 2 is included.

The ADINIT module configures the HI7188 operating mode. After a power-up the HI7188's Control Register is initialized for conversion on one logical channel, line noise filtering off, chopper stabilizer circuitry enabled, and offset binary data coding. The byte sequencing on port accesses is descending (2..1..0), the MSB is the first bit shifted in serial transfer and the serial data out line is disabled. The ADINIT module changes the byte sequencing to ascending where the least significant byte is sent first (0.. 1.. 2) to match the Intel little endian data structure. The shift order is also changed from the MSB first to the LSB first in the serial transfer. In addition, the module configures the HI7188 to convert on 8 logical channels with line noise rejection filtering enabled and the $\overline{EOS}$ interrupt is suppressed during calibration. Please note that the HI7188 does not recognize the switch from MSB to LSB bit positioning until after the lower Control Register byte CR<7:0> write is complete. For this reason the instruction byte and data bytes shown in this module are written in reverse order.

The OFFCAL module writes the CCR registers, thereby programming the HI7188 physical channel information. The physical channel configuration is ordered such that the first logical channel converted uses the physical channel 1 inputs, the second logical uses the physical channel 2 inputs, etc. All channels are programmed for bipolar input levels, offset calibration mode and a gain of 1.

The OCAL_DN module checks for the Calibrate Active (CA) pin to be inactive (logic 0). When CA is inactive, offset calibration is complete and the POSCAL module is invoked.

The POSCAL module is similar to the OFFCAL module in that it writes the CCR registers, this time changing the operating mode to positive full scale calibration while maintaining the other physical to logical channel relationships. Again, the code monitors the CA pin and jumps to the NEGCAL module when CA is inactive. CA inactive indicates the positive gain calibration is complete for all channels.

The NEGCAL module writes the operating mode for each channel to negative full scale calibration while maintaining the other physical to logical channel relationships as written in OFFCAL and POSCAL. When the CA output is detected low, negative gain calibration is complete. This completes the HI7188 calibration sequence.

The CA signal is used to indicate that calibration is complete for

all channels. The $\overline{EOS}$ output could have been used because the suppress $\overline{EOS}$ during calibration bit was set. A down side to using $\overline{EOS}$ in the above example is that a data RAM read would have to occur to deassert the end of scan interrupt.

If the suppress $\overline{EOS}$ bit is not set, the $\overline{EOS}$ signal cannot reliably be used to indicate that calibration is complete. This is due to the inherent asynchronosity between the converter and the serial interface. For example, for a given 8-channel conversion scan, assume that the HI7188 has completed conversions on the first three logical channels when the user programs logical channel 1 for calibration. The $\overline{EOS}$ output will still go active at the end of the current channel scan, but the first logical channel has not been calibrated. It gets calibrated on the ensuing channel scan. Generally, if the user intends to calibrate on some channels while converting others, the $\overline{EOS}$ signal should not be suppressed and the CA pin can be used to indicate when calibration is complete. The order of which the CCR registers is written is inconsequential. What is important is that the analog input for the channel(s) being calibrated is setup before the CCR write occurs.

The ADRUN module initializes the byte count for data transfers into the R1 register, while the starting address for the incoming data storage is set as well as the data buffer size. R2 is set with the mask value for the $\overline{EOS}$ flag which can be read or Port 1 bit 0.

The POLL_$\overline{EOS}$ module checks the status of the $\overline{EOS}$ flag from the HI7188 A/D converter, upon detecting $\overline{EOS}$ being low the READ_DATA module is called.

The READ_DATA module will assert the $\overline{CS}$ signal for the HI7188 serial port low and write the instruction byte to the A/D. Sixteen bytes (2 bytes/channel) of data will be read from the A/D which comprises the entire conversion data for the entire scan. If the data buffer is full the routine will return to the main calling routine.

Interfacing to the SPI Bus Protocol

The Serial Peripheral Interface (SPI Bus) is a serial bus using a 3-wire hardware interface. The three lines used to transfer data from one device to the other are the Serial Clock (SCK) line, the Master In Slave Out (MISO) data line and the Master Out Slave In (MOSI) data line. Data is shifted MSB first, and byte sequencing is in descending order (2.. 1..0). The clock is typically inactive low. Port D, line 4 is used as SCK. The shift clock is generated by the bus master which can be either a microcontroller or a peripheral. Data is routed either to PD2 (Master In Slave Out) or PD3 (Master Out Slave In) depending on software initialization. The Slave Select (SS) line determines if the 68HC11 microcontroller is a Master or Slave on the SPI Bus.

The Serial Peripheral Data I/O register in the microcontroller initiates transmission/reception of a byte. The SPI port on the microcontroller is configured using the Serial Peripheral Control Register. Many devices contain SPI ports, such as the 6805 and 6802, but this discussion will center on the 68HC11. When connecting an HI7188 Sigma-Delta A/D Converter to the SPI Port of the 68HC11, the user has many configuration options available. The serial clock generation can be generated by the HI7188 using Self Clocking mode or by the 68HC11. In Figure 4 the HI7188 is configured as the clock master for the SPI port. This is accomplished by pulling the HI7188 MODE pin high ('1') and grounding the 68HC11 Slave Select ($\overline{SS}$) pin of the microcontroller. Conversely, if the microcontroller was to be the clock master then the $\overline{SS}$ line would be tied high and the MODE pin grounded. The $\overline{EOS}$ line of the A/D converter can be monitored via an interrupt scheme or by using simple polling. The programming example uses a polled status scheme.

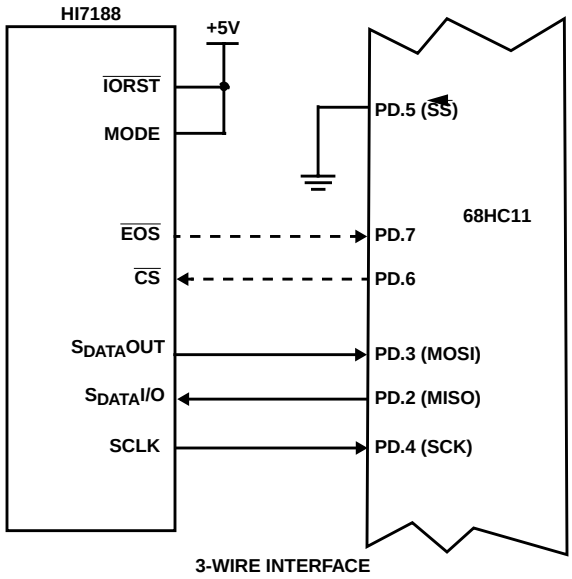

FIGURE 4. HI7188 INTERFACE TO 68HC11

Programming the HI7188 with the 68HC11

The serial ports of the HI7188 A/D converter and the 68HC11 must be configured after power-up or a hardware reset. The SPINIT module configures the SPI Control Register as follows: 1) no interrupt, 2) system enable, 3) normal CMOS outputs, 4) slave mode, 5) SCK idle hi, 6) clock phase hi and 7) clock divider = 2.

68HC11 Microcode Example

```
*Configuration of the SPI Control Register in no
*interrupt, system enable, normal CMOS outputs,
*slave mode, SCK idle hi, clock phase hi, clock divider = 2.
SPINIT  CLRA
        LDAA    #%x1xx xxxx Bit 6 Port D drives CS
        STAA    PORTD       CS inactive
        LDAA    #$4C        Init Serial Port
        STAA    SPCR        Load SPI Control Reg
        LDAA    SPDR        Read to clear port

*SPI Port Load/Write Subroutine called when writing data or
*control out to the HI7188 Serial Port. The SPI Port operates
*automatically once loaded with data.
SPILOAD STAA    SPDR        Load SPI Data Reg
IRWAIT  LDAA    SPSR        Check Port Status
        BPL     IRWAIT      Wait for port to Empty
        RET                 Return from Subroutine
```

*SPI Port Read Subroutine called to capture data from the

HI7188.

```
SPIREAD  LDAA   SPSR       Check Port Status
         BPL    SPIREAD    Wait for Port Ready
         STAA   SPDR       Read SPI Data Reg
         RET               Return from Subroutine

*Initialize the HI7188 Control Register, Convert on 8 logical
*channels, line noise rejection enabled, Descending
*Byte direction, MSB First, Serial Data Out Enabled.
IRCR     LDAA   #%x0xx xxxx Bit 6 Port D drives CS
         STAA   PORTD       CS active
         LDAA   #$A1        Instruction Byte
         CALL   SPILOAD     Call SPI Load Subroutine
         LDAB   #$01        Set Number of Bytes
         LDAA   #$04        Write CR<15:8>
         CALL   SPILOAD     Call SPI Load Subroutine
         LDAA   #$E0        Write CR<7:0>
         CALL   SPILOAD     Call SPI Load Subroutine
         JMP    IRCCR2

* Write the CCR2 register to configure the first four logical
* channels for physical channel ID and operating modes.
* Write 4 bytes.
IRCCR2   LDY    DATAPTR    Buff Pointer for cal data
         LDAA   #$FB       Instruction Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         LDAB   #$03       Set Number of Bytes
CCR2     LDA    A,DATAPTR  Write CCR2 byte
         CALL   SPILOAD    Call SPI Load Subroutine
         DECB              Decrement Pointer
         CMPB   #$00       Compare B to 0
         BEQ    IRCCR1
         INCY              Increment Data Buffer Ptr
         JMP    CCR2
* Write the CCR1 register to configure the final four logical
* channels for physical channel ID and operating modes.
* Write 4 bytes.
IRCCR1   LDAA   #$F7       Instruction Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         LDAB   #$03       Set Number of Bytes
CCR1     LDA    A,DATAPTR  Write CCR1 byte
         CALL   SPILOAD    Call SPI Load Subroutine
         DECB              Decrement Pointer
         CMPB   #$00       Compare B to 0
         BEQ    IROFF
         INCY              Increment Data Buffer Ptr
         JMP    CCR1

* Write Offset Calibration Coefficients.
* HI7188 configured for converting on 8 logical channels,
* therefore a single IR requires 24 bytes of data to be written.
* Three bytes per logical channel.
IROFF    LDAA   #$9D       Instruction Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         LDAB   #$23       Set Number of Bytes
OFFCAL   LDA    A,DATAPTR  Offset Cal Data Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         DECB              Decrement Pointer
         CMPB   #$00       Compare B to 0
         BEQ    IRPOS
         INCY              Increment Data Buffer Ptr
         JMP    OFFCAL

* Write Positive Gain Calibration Coefficients:
* HI7188 configured for converting on 8 logical channels,
* therefore a single IR requires 24 bytes of data to be written.
* Three bytes per logical channel.
IRPOS    LDAA   #$9E       Instruction Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         LDAB   #$23       Set Number of Bytes
POSCAL   LDA    A,DATAPTR  Positive Cal Data Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         DECB              Decrement Reg Pointer
         CMPB   #$00       Compare B to 0
         BEQ    IRNEG
         INCY              Increment Data Buffer Ptr
         JMP    POSCAL

* Write Negative Gain Calibration Coefficients.
* HI7188 configured for converting on 8 logical channels,
* therefore a single IR requires 24 bytes of data to be written.
* Three bytes per logical channel.
IRNEG    LDAA   #$9F       Instruction Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         LDAB   #$23       Set Number of Bytes
NEGCAL   LDA    A,DATAPTR  Negative Cal Data Byte
         CALL   SPILOAD    Call SPI Load Subroutine
         DECB              Decrement Pointer
         CMPB   #$00       Compare B to 0
         BEQ    ADRUN      Go to A/D Run
         INCY              Increment Data Buffer Ptr
         JMP    NEGCAL

*The HI7188 is now fully configured and converting analog
* inputs.


*This Subroutine will collect data from the HI7188
*Sigma-Delta Converter, Poll EOS Signal for Data
*Ready. When ready, write instruction byte and read 16
*data bytes.
IRCR     LDAA   #%x0xx xxxx Bit 6 Port D drives CS
         STAA   PORTD       CS active
ADRUN    LDY    STRT_ADD   Data Buffer Pointer
         LDX    BUFF_SIZE  Data Buffer Size
EOS      LDAA   PORTD      Poll End of Scan
         ANDA   EOSMASK    80H for Port D MSB
         BNE    EOS        EOS Cleared?
RD_DATA  LDAA   #$1C       Instruction Byte
         CALL   SPIREAD    Call SPI Read Subroutine
         STAA   STRT_ADD   Store in Memory
         INCY
         CMPY   BUFF_SIZE  Test Byte Counter
         BNE    RD_DATA    Read another byte
         LDAA   #%x1xx xxxx Bit 6 Port D drives CS
         STAA   PORTD       CS inactive
         RTS                Done Return from
                            subroutine
```

In the example above, after completing the Control Register write, the channel configuration data and the calibration coefficients are downloaded to the HI7188 from a contiguous memory buffer that uses DATAPTR as the handle for the beginning of the buffer. The data is arranged (for logical channels 1 to 8) in order: physical channel configuration, offset calibration coefficients, positive full scale coefficients and negative full

scale coefficients. Although the calibration coefficients were originally generated by the HI7188 in a system calibration procedure, the coefficients are downloaded in this example to demonstrate how to configure the HI7188 without performing a system calibration. This is useful in applications where calibration procedures are maintained by a regulatory body such as for weight scales where calibration must be certified and only done at select times. The routines to load the configuration and the calibration coefficients retrieve a byte from the memory pointer (DATAPTR) and load them into the HI7188 via subroutine SPILOAD. Each successive routine increments through the buffer until the memory is exhausted and the HI7188 is fully configured and calibrated.

The IRCR module configures the HI7188 chip level operating conditions by writing to the HI7188 Control Register. After a power-up the HI7188's Control Register is initialized for conversion on one logical channel, line noise filtering off, chopper stabilizer circuitry enabled and offset binary data coding. The byte sequencing on port accesses is descending (2..1..0), the MSB first bit positioning and the serial data out line is disabled. The IRCR module configures the HI7188 for conversion on 8 logical channels with 60Hz line noise rejection filtering enabled. The I/O configuration maintains MSB first bit positioning and descending byte order but is configured for three wire interface. The IRCR module jumps to the IRCCR2 module when complete.

The IRCCR2 and CCR2 modules together perform a single communication cycle which writes 4 data bytes to the CCR#2 Register. The CCR#2 Register contents describe the physical channel operating conditions for the first 4 logical channels to be converted. The CCR2 module jumps to the IRCCR1 module after the 4 data bytes have been written.

The IRCCR1 and CCR1 modules together perform a single communication cycle which writes 4 data bytes to the CCR#1 Register. The CCR#1 Register contents describe the physical channel operating conditions for the last 4 logical channels to be converted. The CCR1 module jumps to the IROFF module after the 4 data bytes have been written.

The IROFF and OFFCAL modules together perform a single communication cycle write to the Offset Calibration RAM. Since the HI7188 is configured for conversion on 8 logical channels, 24 data bytes are written after the single instruction byte. During this communication cycle the HI7188 is not converting analog inputs because I/O access to the calibration RAMs take precedence over the converters need for the coefficients. When this I/O access is complete, conversion automatically restarts. The OFFCAL module jumps to the IRPOS module after the 24 data bytes have been written.

The IRPOS and POSCAL modules together perform a single communication cycle write to the Positive Gain Calibration RAM. Since the HI7188 is configured for conversion on 8 logical channels, 24 data bytes are written after the single instruction byte. During this communication cycle the HI7188 is not converting analog inputs because I/O access to the calibration RAMs take precedence over the converters need for the coefficients. When this I/O access is complete, conversion automatically restarts. The POSCAL module jumps to the IRNEG module after the 24 data bytes have been written.

The IRNEG and NEGCAL modules together perform a single communication cycle write to the Negative Gain Calibration RAM. Since the HI7188 is configured for conversion on 8 logical channels, 24 data bytes are written after the single instruction byte. During this communication cycle the HI7188 is not converting analog inputs because I/O access to the calibration RAMs take precedence over the converters need for the coefficients. When this I/O access is complete conversion automatically restarts. The NEGCAL module jumps to the ADRUN module after the 24 data bytes have been written.

The ADRUN module polls the HI7188 $\overline{EOS}$ output and reads the conversion results after $\overline{EOS}$ is activated low. Finally, the $\overline{CS}$ line is de-asserted.

Conclusion

This application note has described two typical application circuits with microcode segment examples.

The first circuit is designed with the 8X51 microcontroller(s) in a configuration such that the 8x51 is the clock master in a two line interface and data transfers are LSB to MSB format. The pseudo code configured the part for converting on 8 logical channels with line noise rejection filtering enabled. 8 unique physical channels were converted, all with a gain of 1, all with bipolar inputs. In addition, the code demonstrated full system calibration requirements and a data RAM read for 8 channels.

The second circuit is designed with the 68HCxx microcontroller(s) in a configuration such that the HI7188 is the clock master in a three line interface and data transfers are MSB to LSB format. The pseudo code configured the part for converting on 8 logical channels with line noise rejection filtering enabled. This code segment demonstrated writing the calibration coefficients instead of programming the part to perform system calibration. Finally, the code completed a data RAM read for the 8 logical channels.