BURR-BROWN®
APPLICATION BULLETIN
Mailing Address: PO Box 11400, Tucson, AZ 85734 · Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 · Tel: (520
Telex: 066-6491 · FAX (520) 889-1510 · Product Info: (800) 548-6132 · Internet: www.burr-brown.com/ · FAXLine: (80

INTERFACING THE ADS1210 WITH AN 8xC51 MICROCONTROLLER

By Miaochen Wu

The ADS1210 is a 24-bit delta-sigma analog-to-digital converter. This application bulletin provides examples of how to interface the ADS1210 with an Intel 8xC51 microcontroller. The converter can be operated with the serial data clock SCLK, generated by itself or provided externally, such as by an 8xC51. The ADS1210 supports SPI (Serial Peripheral Interface) interfacing and SSI (Synchronized Serial Interface) interfacing. This bulletin focuses on interfacing the ADS1210 to an 8xC51, which provides the synchronous SCLK signal. The schematic and 8xC51 assembly code are provided for both of the interface protocols.

DATA STRUCTURE

The ADS1210 has five internal registers; the 24-bit Data Output Register (DOR), 8-bit Instruction Register (INSR), 32-bit Command Register (CMR), 24-bit Offset Calibration Register (OCR) and 24-bit Full-Scale Calibration Register (FCR). In the 8xC51, memory address from 060H to 06EH are used to duplicate these registers. This part of memory is called the ADS memory. The data stored in the ADS memory are from LSB (Least Significant Bit) to MSB (Most Significant Bit) for all the registers except INSR. Table I shows the organization of the ADS memory.

	MSB	1 Byte	LSB
060H	B16	DOR 2	B23
	B8	DOR 1	B15
	B0	DOR 0	B7
	B7	**INSR**	**B0**
	B24	CMR 3	B31
	B16	CMR 2	B23
	B8	CMR 1	B15
	B0	CMR 0	B7
	B16	OCR 2	B23
	B8	OCR 1	B15
	B0	OCR 0	B7
		RESERVED	
	B16	FCR 2	B23
	B8	FCR 1	B15
06EH	B0	FCR 0	B7

TABLE I. The Memory Organization Used to Map the ADS1210 to the 8xC51.

The INSR data is the code to instruct the ADS1210 what to do in the next operation. Typical instruction codes are the following:

INSR = C0H	Read 3 bytes of the Data Output Register of the ADS1210
INSR = 64H	Write 4 bytes of the Command Register of the ADS1210
INSR = E4H	Read 4 bytes of the Command Register of the ADS1210

This data structure is implemented in the following assembly code:

DATA STRUCTURE

```
; The ADS1210 Instruction Register code
RD_DATA     EQU     0C0H    ; INSR = Read 3 Bytes Data
WR_CMR      EQU     064H    ; INSR = Write 4 Bytes CMR
RD_CMR      EQU     0E4H    ; INSR = Read 4 Bytes CMR
RRD_DATA    EQU     003H    ; INSR = Read 3 Bytes Data Reversed
RRD_DLSB    EQU     043H    ; INSR = Read 3 Bytes Data Reversed
                            ; with LSByte first
; Definition for delays
D1210A      EQU     8       ; 256 XIN clocks with XIN=10MHz
D1210B      EQU     15      ; 512 XIN clocks
D1210C      EQU     30      ; 1024 XIN clocks
DSYNCD      EQU     100     ; _DSYNC=L for 120µs
; Pin connections to the ADS1210
DSYNC_      BIT     P2.5    ; the ADS1210 _DSYNC PIN
CS_         BIT     P2.4    ; the ADS1210 _CS PIN
M_SDIO      BIT     P3.0    ; SDIO by MC51
M_SCLK      BIT     P3.1    ; SCLK by MC51
; Definition for flags
DRDYINT     BIT     003H    ; _DRDY interrupt flag
LSByte      BIT     004H    ; LSbyte flag
SLEEP       BIT     005H    ; SLEEP flag
; Address for variables
AOFS        DATA    05AH    ; Address offset
ABIT        DATA    05BH    ; Address bit
NBYTE       DATA    05CH    ; INSR BYTE
NADDR       DATA    05DH    ; INSR ADDR
; Address for storing the ADS1210 register data
DOR2        DATA    060H    ; DATA REG BYTE2
DOR1        DATA    061H    ; DATA REG BYTE1
DOR0        DATA    062H    ; DATA REG BYTE0
INSR        DATA    063H    ; INSTRUCTION REG
CMR3        DATA    064H    ; COMMAND REG BYTE3
CMR2        DATA    065H    ; COMMAND REG BYTE2
CMR1        DATA    066H    ; COMMAND REG BYTE1
CMR0        DATA    067H    ; COMMAND REG BYTE0
OCR2        DATA    068H    ; OFFSET COEF REG BYTE2
OCR1        DATA    069H    ; OFFSET COEF REG BYTE1
OCR0        DATA    06AH    ; OFFSET COEF REG BYTE0
FCR2        DATA    06CH    ; GAIN COEF REG BYTE2
FCR1        DATA    06DH    ; GAIN COEF REG BYTE1
FCR0        DATA    06EH    ; GAIN COEF REG BYTE0
```

TABLE II. 8xC51 Assembly Listing—Constants and Definitions.

Printed in U.S.A. March, 1997

INITIALIZATION

After power on, the I/O pins and internal memory of the 8xC51 defined in Table II are initialized. The memory for CMR, OCR and FCR are initialized from LSB to MSB, based on the register definition from the ADS1210 data sheet. The memory for INSR is initialized from MSB to LSB with the value C0H. Port P3 of the 8xC51 is used for input and output of the 8xC51. The ADS1210 pins $\overline{DSYNC}$, $\overline{CS}$, SDIO and SCLK are connected to the 8xC51 pins P2.5, P2.4, P3.0 and P3.1 (see Figures 1 through 3). AOFS is for INSR address offset with a value of 01100000B. ABIT is for masking the address bits of INSR with a value of 00001111B. R7 holds the INSR data. R1 and R3 in register bank 00 and 01 are assigned with 60H and 02H. Register R1 is used to index the ADS memory, and R3 stores the number of bytes for each register in the ADS memory. For example, when reading the three bytes of the Data Output Register, R1 = 03H.

INITIALIZATION

```
; Interrupt Vectors
        ORG     0               ; Reset vector.
        AJMP    RESET

        ORG     13H             ; External interrupt 1
        CLR     DRDYINT         ; _DRDY=L
        RETI
; MC51 Initialization
RESET:
        MOV     IE,#00000101B   ; Disable all interrupts
        MOV     CCON,#00H       ; Disable PCA
        MOV     PSW,#00H        ; Select Bank0
        MOV     SP,#30H         ; Stack pointer
        MOV     8EH,#01H        ; EMI flag
        MOV     A,#11111111B    ; ACC=FFH
        MOV     P0,A            ; P0 input
        MOV     P1,A            ; P1 output H
        MOV     P2,A            ; P2 input
                                ; {NA,NA,DSYNC_,CS_,NA,NA,NA,NA}
        MOV     A,#10111111B    ; ACC=BFH
        MOV     P3,A            ; P3 inout
        MOV     AOFS,#01100000B ; INSR address offset
        MOV     ABIT,#00001111B ; INSR address bits
        MOV     R7,#RD_DATA     ; R7 gets INSR data
        MOV     IP,#00H         ; INT priority
        MOV     TCON,#00000101B ; EX0/1 edge trigger
        MOV     SCON,#00000000B ; serial port mode 0
                                ; 000REN00TIRI
; The ADS1210 Initialization
START:
        SETB    DRDYINT         ; _DRDY=H
        CLR     LSByte          ; LSByte=L
        CLR     SLEEP           ; SLEEP=L
        MOV     INSR,#RD_DATA   ; INSR=C0H
        MOV     CMR3,#10000010B ; CMR3=R41H
        MOV     CMR2,#00H       ; CMR2=R00H
        MOV     CMR1,#00H       ; CMR1=R00H
        MOV     CMR0,#01101000B ; CMR0=R16H
        MOV     OCR2,#00H       ; OCR2=R00H
        MOV     OCR1,#00H       ; OCR1=R00H
        MOV     OCR0,#00H       ; OCR0=R00H
        MOV     FCR2,#01H       ; FCR2=R80H
        MOV     FCR1,#00H       ; FCR1=R00H
        MOV     FCR0,#00H       ; FCR0=R00H
        MOV     R1,#60H         ; ADDR=60H
        MOV     R3,#02H         ; NBYTE=02H
        SETB    PSW.3           ; Rbank=01
        MOV     R1,#60H         ; ADDR=60H
        MOV     R3,#02H         ; NBYTE=02H
        CLR     PSW.3           ; Rbank=00
```

TABLE III. 8xC51 Assembly Listing—ADS1210 Initialization.

SPI-COMPATIBLE INTERFACE DESIGN

A typical configuration using SPI type interfacing is shown in Figure 1. In the schematic, the SCLK pin of the ADS1210 is connected to pin TXD of the 8xC51, SDIO is connected to RXD, and $\overline{DRDY}$ to INT1. The $\overline{CS}$ pin is hardwired to digital ground in this operating mode. The connection for $\overline{DSYNC}$ pin is optional. For simplicity, assume that the ADS1210 X_{IN} pin is at F_{XIN} = 10MHz frequency and 8xC51 XTAL1 pin is at F_{XTAL} = 10MHz frequency.

There are basically five routines that can be used to interface and control the ADS1210 (see Table IV). The routine SPISCWR performs writing the one byte INSR code followed by the writing operation defined in the INSR code. For example, with INSR=64H, SPISCWR will write 64H to the ADS1210 Instruction Register followed by writing four bytes of CMR data indexed by R1 into the ADS1210 Command Register.

The routine SPISCRD performs writing the one byte INSR code followed by the reading operation defined in the INSR code. For example, with INSR=E4H, SPISCRD will write E4H to the ADS1210 INSR followed by reading four bytes of CMR from the ADS1210. This data is placed into the ADS memory as indexed by R1.

The routine PSCONVT performs the basic data acquisition function. It writes the one byte INSR code C0H into the ADS1210 and reads back three bytes of data from the Data Output Register. This data is put into the ADS memory as indexed by R1.

Two routines, EWRITE and EREAD, are the core for the above operations. In EWRITE, the INSR content is copied to R5. The data to be written into the ADS1210 is indexed by R1. R3 controls the number of bytes sent to the ADS1210. In EREAD, R5 carries the INSR code. The data to be read back from the ADS1210 is put to the address indexed by R1. To be compatible with the approach described in the section Increasing Interface Speed, EWRITE and EREAD transmit data with LSB first. The reader can easily make it MSB first. In such a case, supporting routines BITINSR and REVINSR are not needed. The SCLK clock rate using these two routines is about 119kHz (F_{XTAL}/84).

The routine PCSYNC performs the synchronization function as defined in the ADS1210 data sheet. The pin $\overline{DSYNC}$ will be held LOW, approximately 120µs, and then returned HIGH. The routine PCRESET generates a pre-defined clock pattern at the SCLK pin to trigger a serial reset operation on the ADS1210. The main routine SPISC is just an example showing how to use the routines described above.

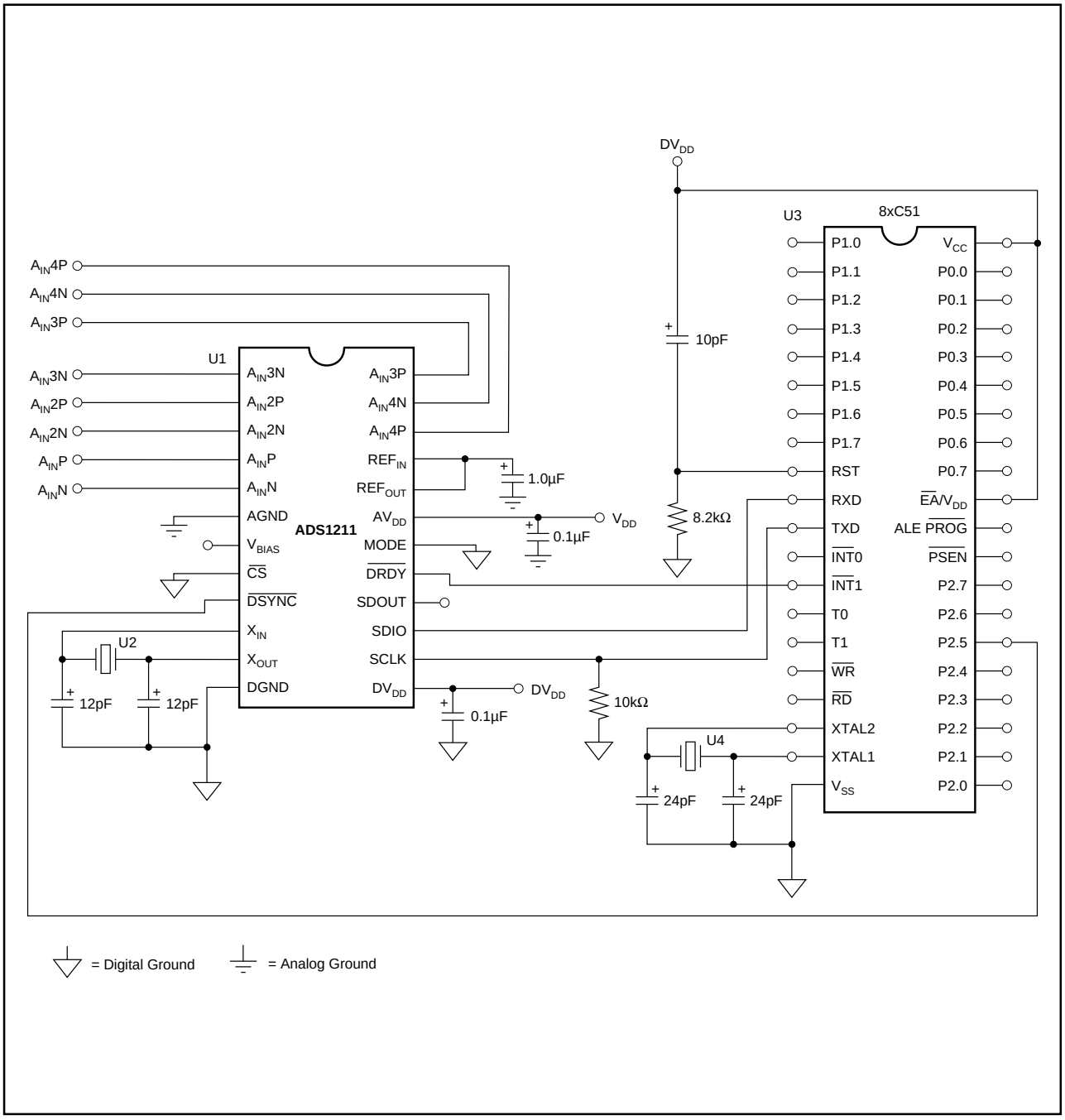

FIGURE 1. SPI Compatible Interface.

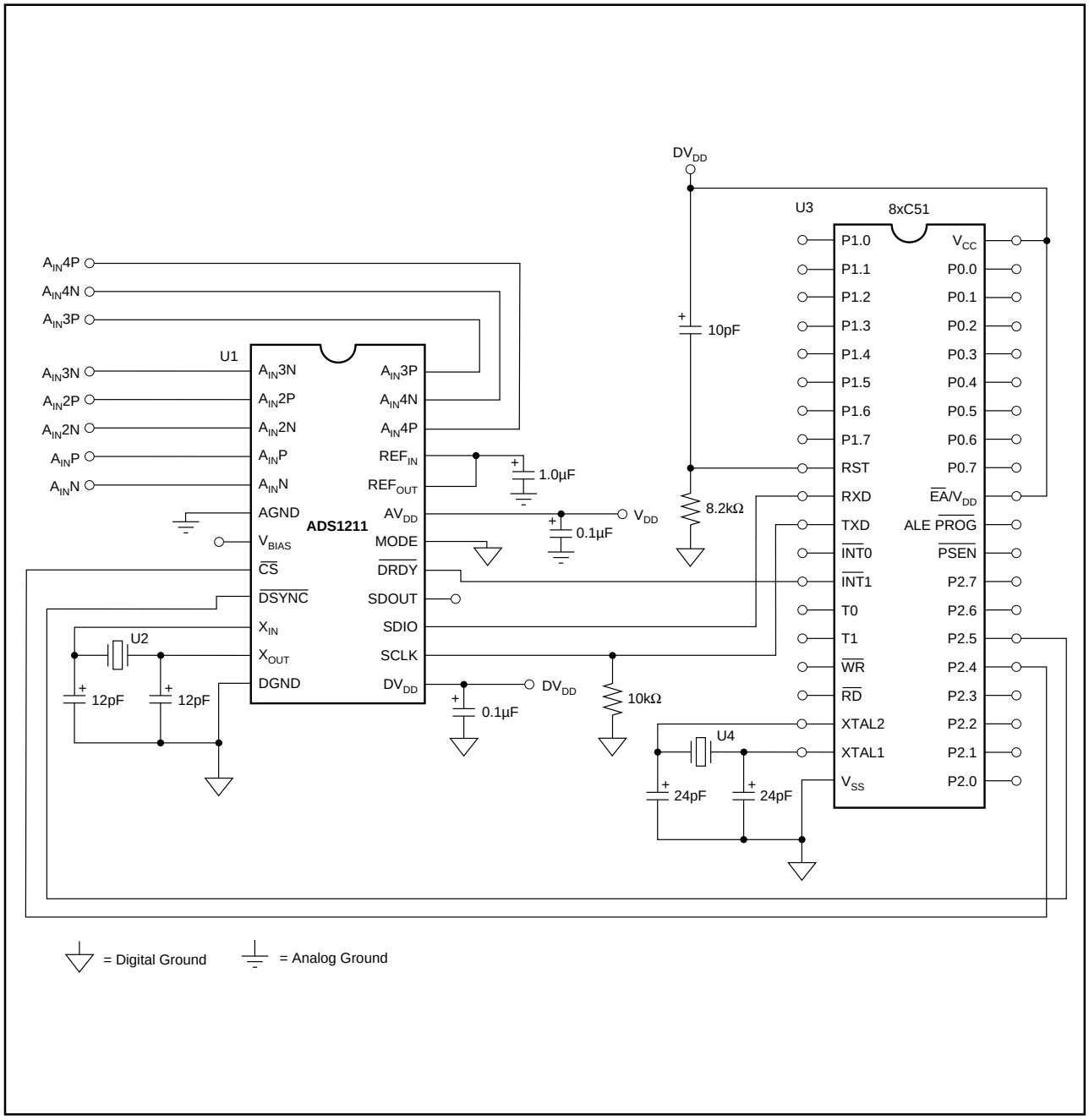

FIGURE 2. SSI Interface.

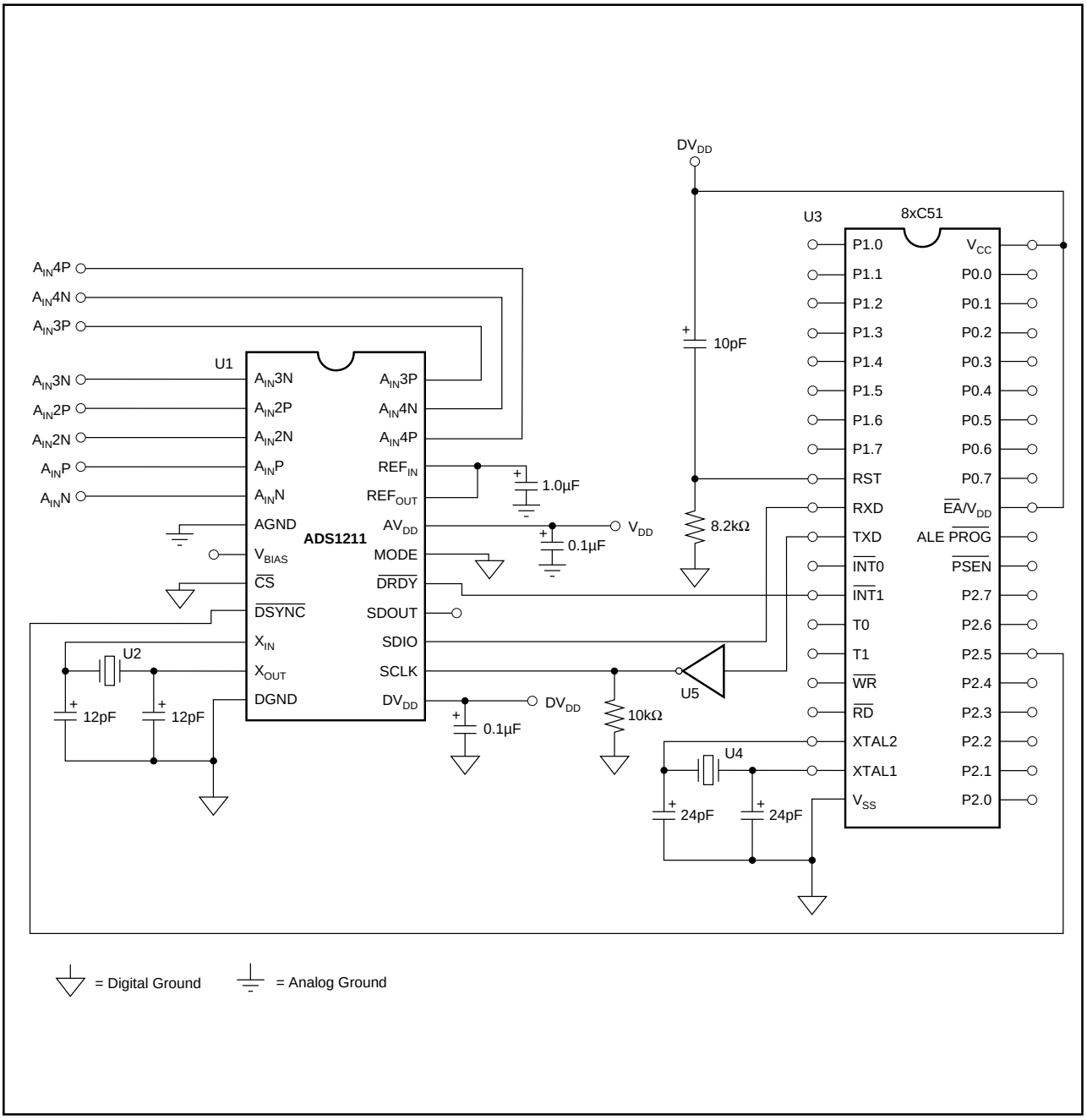

FIGURE 3. Increasing the Interface Speed.

SPI-COMPATIBLE INTERFACE DESIGN

```
; Routines for SPISC protocol
SPISC:
        MOV     INSR,#WR_CMR        ; get WR_CMR
        ACALL   SPISCWR             ; Write the ADS1210 CMR
        MOV     INSR,#RD_CMR        ; get RD_CMR
        ACALL   SPISCRD             ; Read back the ADS1210 CMR
        ACALL   PSCONVT             ; get the ADS1210 Data
        ACALL   PCSYNC              ; Sync the ADS1210 via _DSYNC Pin
        ACALL   PCRESET             ; Reset the ADS1210 via SCLK
        RET
; Routine for SPISC write the ADS1210
SPISCWR:
        ACALL   BITINSR             ; get ADDR NBYTE
        ACALL   REVINSR             ; reverse INSR
        JNB     SLEEP,PSDWH         ; not sleep
        CLR     SLEEP               ; SLEEP=L
        JMP     PSWKUP              ; wakeup
PSDWH:  JB      DRDYINT,PSDWH       ; sense _DRDY
        SETB    DRDYINT
PSWKUP: ACALL   EWRITE              ; serial write
        MOV     B,CMR3              ; get CMR3
        MOV     C,B.4               ; get LSByte
        MOV     LSByte,C            ; save LSByte
        MOV     A,#00000011B        ; mask CMR2
        ANL     A,CMR2              ; get sleep mode
        CJNE    A,#00000011B,PSEND  ; not sleep
        SETB    SLEEP               ; SLEEP=H
PSEND:  RET
; Routine for SPISC read the ADS1210
SPISCRD:
        ACALL   BITINSR             ; get ADDR NBYTE
        ACALL   REVINSR             ; reverse INSR
PSDRH:  JB      DRDYINT,PSDRH       ; sense _DRDY
        SETB    DRDYINT
        ACALL   EREAD               ; serial read
        RET
; Routine for SPISC data converting
PSCONVT:
        JB      DRDYINT,EPSCON      ; sense _DRDY
        SETB    DRDYINT
        MOV     R3,#02H             ; read 3 bytes
        MOV     R1,#60H             ; DOR2 address
        MOV     R5,#RRD_DATA        ; get RRD_DATA
        JNB     LSByte,EDINS2       ; goto EDINS2
        MOV     R5,#RRD_DLSB        ; get RRD_DLSB
```

SPI-COMPATIBLE INTERFACE DESIGN (cont)

```
EDINS2: ACALL   EREAD               ; read data
;       ACALL   MCWRITE             ; MC writes data to RAM
EPSCON: RET
; Routine for PCSYNC
PCSYNC:
        CLR     DSYNC_              ; _DSYNC=L
        MOV     R6,#DSYNCD          ; delays
LOOP:   DJNZ    R6,LOOP
        SETB    DSYNC_              ; _DSYNC=H
        RET
; Routine for PCRESET
PCRESET:
        CLR     M_SCLK              ; SCLK=L
        SETB    M_SCLK              ; SCLK=H
        MOV     R6,#D1210A          ; delays
LOOPA:  NOP
        DJNZ    R6,LOOPA
        CLR     M_SCLK              ; SCLK=L
        NOP
        NOP
        NOP
        NOP
        SETB    M_SCLK              ; SCLK=H
        MOV     R6,#D1210B          ; delays
LOOPB:  NOP
        DJNZ    R6,LOOPB
        CLR     M_SCLK              ; SCLK=L
        NOP
        NOP
        NOP
        NOP
        SETB    M_SCLK              ; SCLK=H
        MOV     R6,#D1210C          ; delays
LOOPC:  NOP
        DJNZ    R6,LOOPC
        CLR     M_SCLK              ; SCLK=L
        RET
; Routine for serial port external clock write
EWRITE:
        MOV     A,R5                ; get INSR
        ACALL   TXBIT               ; output 8 bits
ETION:  CJNE    R3,#00H,EWMORE      ; if not 0, more output
        MOV     A,@R1               ; output last one
        ACALL   TXBIT
        SJMP    EWEND
```

TABLE IV. 8xC51 Assembly Listing—Core ADS1210 Communication Functions (SPI Interface).

SPI-COMPATIBLE INTERFACE DESIGN

```
EWMORE:   MOV     A,@R1              ; output byte 1,2,3
          ACALL   TXBIT
          INC     R1                 ; ADDR+1
          DEC     R3                 ; NBYTE-1
          SJMP    ETION              ; TI send
EWEND:    RET
; Routine for TX data
TXBIT:    MOV     R6,#08H            ; 8 bits
TXLOOP:   RRC     A                  ; get LSB, can be modified MSB first
          SETB    M_SCLK             ; SCLK=H
          MOV     M_SDIO,C           ; output LSB
          CLR     M_SCLK             ; SCLK=L
          DJNZ    R6,TXLOOP          ; 8 bits out
          RET
; Routine for RX data
RXBIT:    MOV     R6,#08H            ; 8 bit
RXLOOP:   SETB    M_SCLK             ; SCLK=H
          MOV     C,M_SDIO           ; get data
          CLR     M_SCLK             ; SCLK=L
          RRC     A                  ; get LSB into A,  can be modified MSB first
          DJNZ    R6,RXLOOP          ; 8 bit in
          RET
; Routine for serial port external clock read
EREAD:
          MOV     A,R5               ; get INSR
          ACALL   TXBIT
ERISSI:   CJNE    R3,#00H,ERMORE     ; if not 0, more output
          ACALL   RXBIT
          MOV     @R1,A
          SJMP    EREND              ; goto end
ERMORE:   ACALL   RXBIT
          MOV     @R1,A              ; store data
          INC     R1                 ; ADDR+1
          DEC     R3                 ; NBYTE-1
          SJMP    ERISSI             ; more read
EREND:    RET
; Routine to get INSR bits for NBYTE and ADDR
BITINSR:
          MOV     A,INSR             ; get NBYTE
          ANL     A,AOFS             ; mask bits
          SWAP    A                  ; swap A
          RR      A                  ; rotate right A
          MOV     R3,A               ; save NBYTE
          MOV     A,INSR             ; get ADDR
```

SPI-COMPATIBLE INTERFACE DESIGN (cont)

```
          ANL     A,ABIT             ; mask bits
          ORL     A,AOFS             ; offset ADDR
          MOV     R1,A               ; save read ADDR
          MOV     NBYTE,R3           ; save NBYTE
          MOV     NADDR,R1           ; save NADDR
          SETB    PSW.3              ; Rbank=01
          MOV     R3,NBYTE           ; save NBYTE
          MOV     R1,NADDR           ; save NADDR
          CLR     PSW.3              ; Rbank=00
          RET
; Routine for reverse INSR
REVINSR:
          MOV     A,INSR             ; INSR to A
          RLC     A                  ; rotate bit7
          MOV     B.0,C
          RLC     A                  ; bit6
          MOV     B.1,C
          RLC     A                  ; bit5
          MOV     B.2,C
          RLC     A                  ; bit4
          MOV     B.3,C
          RLC     A                  ; bit3
          MOV     B.4,C
          RLC     A                  ; bit2
          MOV     B.5,C
          RLC     A                  ; bit1
          MOV     B.6,C
          RLC     A                  ; bit0
          MOV     B.7,C
          MOV     R5,B               ; save to R5
          RET
```

TABLE IV (cont). 8xC51 Assembly Listing—Core ADS1210 Communication Functions (SPI Interface).

SSI INTERFACE DESIGN

A typical configuration using SSI interfacing is shown in Figure 2. The difference between Figure 1 and Figure 2 is the $\overline{CS}$ pin is connected to P2.4. In this interface protocol, communication starts when $\overline{CS}$ is toggles from HIGH to LOW. This means that the communication is synchronized by the $\overline{CS}$ signal.

There are three routines for basic SSI interface operation: The routine SSIECWR performs writing one byte INSR code followed by the writing operation defined in the INSR code. The routine SSIECRD performs writing one byte INSR code followed by the reading operation defined in the INSR code. However, SSIECRD is also used to read Data Output Register from the ADS1210. If a single data point is collected from the ADS1210, SSIECRD will do the job.

The ADS1210 design supports continuous reading of the Data Output Register without issuing the INSR data for each read. The routine SECONVT performs this function. Once SSIECRD is executed to read back a single data point and $\overline{CS}$ pin stays LOW and is not toggled, SECONVT will

SSI INTERFACE DESIGN

```
; Routine for SSIEC protocol
SSIEC:      MOV     INSR,#WR_CMR          ; get WR_CMR
            ACALL   SSIECWR               ; Write the ADS1210 CMR
            MOV     INSR,#RD_CMR          ; get RD_CMR
            ACALL   SSIECRD               ; Read back the ADS1210 CMR
            ACALL   SECONVT               ; get the ADS1210 Data
            ACALL   PCSYNC                ; Sync the ADS1210 via DSYNC_ Pin
            ACALL   PCRESET               ; Reset the ADS1210 via SCLK
            RET
; Routine for SSIEC write the ADS1210
SSIECWR:
            SETB    CS_                   ; _CS=H
            ACALL   BITINSR               ; get ADDR NBYTE
            ACALL   REVINSR               ; reverse INSR
            JNB     SLEEP,SEDWH           ; no sleep
            CLR     SLEEP                 ; SLEEP=L
            JMP     SEWKUP                ; wakeup
SEDWH:      JB      DRDYINT,SEDWH         ; sense _DRDY
            SETB    DRDYINT
SEWKUP:     CLR     CS_                   ; _CS=L
            ACALL   EWRITE                ; serial write
            SETB    CS_                   ; _CS=H
            MOV     A,#00000011B          ; mask CMR2
            ANL     A,CMR2                ; get sleep mode
            CJNE    A,#00000011B,SEEND    ; not sleep
            SETB    SLEEP                 ; SLEEP=H
SEEND:      RET
; Routine for SSIEC read the ADS1210
SSIECRD:
            SETB    CS_                   ; _CS=H
            ACALL   BITINSR               ; get ADDR NBYTE
            ACALL   REVINSR               ; reverse INSR
SEDRH:      JB      DRDYINT,SEDRH         ; sense _DRDY
            SETB    DRDYINT
            CLR     CS_                   ; _CS=L
            ACALL   EREAD                 ; serial read
            CJNE    R5,#03H,ESERD3        ; not SSI read
            JMP     ESERD2                ; SSI read
ESERD3:     CJNE    R5,#43H,ESERD         ; not SSI read
            JMP     ESERD2                ; SSI read
ESERD:      SETB    CS_                   ; _CS=H
ESERD2:     RET
; Routine for SSIEC data converting
SECONVT:
            JB      DRDYINT,ESECON        ; sense _DRDY
            SETB    DRDYINT
            JB      CS_,ESECON            ; _CS=H
            MOV     R3,#02H               ; read 3 bytes
            MOV     R1,#60H               ; DOR2 address
            ACALL   ERISSI                ; read data
;           ACALL   MCWRITE               ; MC writes data to RAM
ESECON:     RET
```

TABLE V. 8xC51 Assembly Listing—Core ADS1210 Communication Functions (SSI Interface).

continuously read back three bytes data from the ADS1210. The data is put in the ADS memory indexed by R1. Routine MCWRITE (not shown in the code listing) can be inserted to copy the data from DOR2-DOR0 address to external memory.

Two routines, EWRITE and EREAD used in the SPI-compatible interface, are shared for the SSI interface. The routines PCSYNC and PCRESET can be modified by adding $\overline{CS}$ toggling to perform synchronization and serial reset operation for SSI operation. The main routine SSIEC is an example of how to use the routines described in Tables IV and V.

INCREASING THE INTERFACE SPEED

In the SPI-Compatible Design and SSI Interface Design sections of this application bulletin, the clock frequency in SCLK is approximately 119kHz or $F_{XTAL}/84$. The ADS1210 supports a maximum 2MHz or $F_{XIN}/5$ MHz SCLK clock frequency. One way to increase the interface speed is to increase the 8xC51 crystal frequency, F_{XTAL}. For example, with F_{XTAL} = 40MHz, the SCLK frequency is 476kHz. However, in order to have a 2MHz SCLK clock frequency, F_{XTAL} would have to be 168MHz! Another way to increase the interface speed is to use the 8xC51 Standard Serial Interface Mode 0. The clock frequency at the TXD pin is $F_{XTAL}/12$ in Mode 0. For example, with F_{XTAL} = 24MHz, the SCLK frequency is 2MHz.

A typical configuration using the 8xC51 Standard Serial Interface Mode 0 is shown in Figure 3. As TXD stays HIGH during idle, an inverter is required between SCLK and TXD. In Mode 0, serial data is transmitted/received with LSB first. Two core routines, EWRITE and EREAD, are modified and listed in Table VI. The data structure described at the beginning still supports these two new routines. Of course, the interrupt register IE should be set properly to enable the interrupt.

INCREASING THE INTERFACE SPEED

```
; Routine for serial port external clock write
EWRITE:
        MOV     SCON,#00000000B     ; serial mode=0
        MOV     SBUF,R5             ; output INSR
ETION:  JNB     SCON.1,ETION        ; TI is done ?
        CLR     SCON.1              ; clear TI
        CJNE    R3,#00H,EWMORE      ; if not 0, more output
        MOV     SBUF,@R1            ; output last one
ETION2: JB      SCON.1,EWEND        ; goto end
        SJMP    ETION2
EWMORE: MOV     SBUF,@R1            ; output byte 1,2,3
        INC     R1                  ; ADDR+1
        DEC     R3                  ; NBYTE-1
        JMP     ETION               ; TI send
EWEND:  RET
; Routine for serial port external clock read
EREAD:
        MOV     SCON,#00000000B     ; serial mode=0
        MOV     SBUF,R5             ; output INSR
ETION3: NB      SCON.1,ETION3       ; TI is done ?
ERISSI: OV      SCON,#00010001B     ; serial mode=0
ERIUSE: JNE     R3,#00H,ERMORE      ; if not 0, more output
        CLR     SCON.0              ; clear RI
ERION:  NB      SCON.0,ERION        ; RI is done ?
        MOV     @R1,SBUF            ; input last one
        SJMP    EREND               ; goto end
ERMORE: CLR     SCON.0              ; clear RI
ERION2: NB      SCON.0,ERION2       ; RI is done ?
        MOV     @R1,SBUF            ; input byte 1,2,3
        INC     R1                  ; ADDR+1
        DEC     R3                  ; NBYTE-1
        SJMP    ERIUSE              ; more read
EREND:  RET
```

TABLE VI. 8xC51 Assembly Listing—Mode 0 Interface Routines.

SUMMARY

This application bulletin discusses how to interface the ADS1210 with an 8xC51 microcontroller using either SPI-compatible interface or SSI interface. The routines for writing/reading the ADS1210 internal registers are provided. A modified version is given to increase the interface speed. The schematic for the different protocols are presented. The reader can customize the data structure and the writing/reading routines for the application. The routines, EWRITE and EREAD in the sections SPI-Compatible Interface Design and Increasing Interface Speed, serve for general purpose.