

3.2 TLC320AC01 Analog Interface Circuit

The 'AC01 is an audio-band processor. It integrates a band-pass switched-capacitor antialiasing input filter, 14-bit A/D and D/A converters, a low-pass switched-capacitor output reconstruction filter, $\sin x/x$ compensation, and a serial port for data and control transfers. The A/D and D/A channels operate synchronously, and data is transferred in 2s-complement format. It has nine internal registers that can be programmed for an application.

There are three basic modes of operation:

- 1) Stand-alone analog interface mode: The 'AC01 generates the shift clock and frame synchronization signals for data transfers; it is the only AIC used.
- 2) Master-slave mode: One 'AC01 serves as the master, generating the master shift clock and frame synchronization signals; the other 'AC01s serve as slaves.
- 3) Linear codec mode: The shift clock and frame synchronization signals are externally generated by the host.

Software control of the AIC allows you to program the sampling frequency, cut-off frequencies for the low- and high-pass filters, and analog input and output gains at any time. No programming is needed when the default values of the 'AC01 are satisfactory.

Data transfers between the DSP and the 'AC01 are categorized as primary and secondary serial communications. During primary communication, the 14 most significant bits (MSBs), bits 2 through 15, specify sample data. If the two least significant bits (LSBs), bits 1 and 0, are 11, a subsequent 16 bits of control information is received by the 'AC01 as part of the secondary serial communication interval. This control information programs one of the nine internal registers. During the secondary communication interval, the bits transmitted to the 'AC01 are defined as follows:

- ☐ Bits 15 and 14, which control phase shifting in certain applications, are usually 0s.
- ☐ Bit 13 decides whether the data is written to or read from a register.
- ☐ Bits 12 through 8 define the address of the register to be programmed.
- ☐ Bits 7 through 0 contain the data to be stored in a register for write operations.

The bits received from the 'AC01 during the secondary communication interval are 0s for bits 15 through 8, and the value of the register data for bits 7 through 0.

The three programs that follow show the use of the 'AC01 in different circumstances: Example 3–1 assumes that the user is satisfied with the default configuration of the 'AC01 and proceeds to receive and send data, Example 3–2 on page 3-12 shows how to program the 'AC01 for a particular configuration before data transfer, and Example 3–3 on page 3-17 demonstrates serial communications while in a master-slave configuration. All three programs consist of a main routine which, apart from initializing the serial port and the interrupt, also resets the 'AC01.

Example 3–1 initiates a reset of the 'AC01, initializes the serial port, and sets up interrupts. To reset the 'AC01 properly, it must be held in reset for at least one MCLK cycle. With the 'AC01 operating at 10.368 MHz, one MCLK cycle equals 96.45 ns. In all the examples, the total time spent between initiating a reset and pulling the 'AC01 out of reset is 12 cycles. For a '54x DSP operating at 40 MHz, one cycle is 25 ns. This means that the 'AC01 spends 300 ns in reset. This is roughly three times the specification's recommended value and is safe, since it allows a proper reset.

If a 50-MHz DSP is used, one cycle equals 20 ns; the same code keeps the 'AC01 in reset for 240 ns. This is also safe, since it is approximately 2.5 times the specification's recommended value.

The 'AC01 can operate at a minimum clock frequency of 5.184 MHz. This implies that it must be held in reset for at least one MCLK cycle, or 192.9 ns. Assuming a processor speed of 50 MHz, the 'AC01 spends 12 cycles multiplied by 20 ns = 240 ns in reset.

A safe margin holds the 'AC01 in reset for at least twice the specification's recommended time. To allow the necessary time in reset, use a dedicated timer function or insert useful instructions to fill the time slot. After the 'AC01 is reset, the main program sits in a loop, waiting for a serial port interrupt.

Example 3–1 initializes the serial port and uses the default initialization of the 'AC01, which is in the master mode and transfers the data upon a receive interrupt. It is not necessary to initiate a secondary communication interval to program the 'AC01. The serial port is configured between the cycles that the AIC is in reset.

Example 3–1. Default Initialization of 'AC01

```

;TEXAS INSTRUMENTS INCORPORATED
    .mmregs
    .include      "interrpt.inc"
    .include      "init_ser.inc"
AIC_VAR_DP      .usect "aic_vars",0
aic_in_rst      .usect "aic_vars",1
aic_out_of_rst  .usect "aic_vars",1
    .def          serial_init
;-----
; Functional Description
; This routine initializes the serial port 1 of 541. The serial port is put
; in reset by writing 0's to Rrst and Xrst bits and pulled out of reset by
; writing 1's to both Rrst and Xrst bits. This routine also puts the AC01
; in reset and after 12 cycles the AC01 is pulled out of reset. The serial
; port initialization is done during the 12 cycle latency of the AC01 init.
;-----
    .sect          "ser_cnfg"
serial_init:
    LD              #AIC_VAR_DP,DP          ; initialize DP for aic_reset
    ST              #K_0, aic_in_rst        ; bit 15 = 0 of TCR resets AIC
    PORTW          aic_in_rst,K_TRGCR_ADDR ;write to address 14h (TCR)
*****
*We need at least 12 cycles to pull the AIC out of reset.
*****
    STM              #K_SERIAL_RST, SPC1      ;reset the serial port with
                                           ;0000 0000 0000 1000
    STM              #K_SERIAL_OUT_RST, SPC1  ;bring ser.port out of reset
                                           ;0000 0000 1100 1000

    RSBX            INTM
    LD              #0,DP
    ORM              #(K_RINT1|K_INT1),IMR    ; Enable RINT1,INT1
                                           ; 0000 0000 0100 0010
    LD              #AIC_VAR_DP,DP          ; restore DP
    STM              #(K_RINT1),IFR          ; clear RINT1
    STM              #K_0,DXR1              ; 0000 0000 0100 0000
; Pull the AC01 out of reset - the AC01 requires that it be held in reset for
; 1 MCLK, which is equivalent to 96.45ns (based on an MCLK of 10.368MHz)
    ST              #K_8000, aic_out_of_rst  ; bit 15 = 1 brings AIC from reset
    RETD
    PORTW          aic_out_of_rst, K_TRGCR_ADDR ;AIC out of reset
    .end
* This include file includes the SPC1 register configuration
*****

```

Example 3–1. Default Initialization of 'AC01 (Continued)

```

* FILENAME: "INIT_SER.INC"
; SPC Register Organization
*
*  -----
*  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
*  |----|----|----|----|----|----|----|
*  | FREE | SOFT | RSRFULL | XSREMPY | XRDY | RRDY | IN1 | IN0 |
*  -----
*
*  -----
*  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
*  |----|----|----|----|----|----|----|
*  | RRST | XRST | TXM | MCM | FSM | FO | DLB | RES |
*  -----
*
* This include file includes the SPC1 register configuration
*****
;Bit      Name      Function
;0         Reserved  Always read as 0
;1         DLB       Digital loop back : 0 -> Disabled, 1 -. Enabled
;2         FO        Format bit: 0 - > data transferred as 8 bit bytes, 1 -> 16 bit
                        words
;3         FSM       Frame sync pulse: 0 -> serial port in continuous mode, 1 -> FSM
                        is required
;4         MCM       Clock mode bit: 0 -> CLKX obtained from CLKX pin 1-> CLKX
                        obtained from CLKX
;5         TXM       Transmit mode bit: 0 -> Frame sync pulses generated externally
                        and supplied on FSX pin, 1-> Internally generated frame sync
                        pulses out on FSX pin
;6         XRST      Transmit reset bit: 0 -> reset the serial port, 1-> bring
                        serial port out of reset
;7         RRST      Receive reset bit: 0 -> reset the serial port, 1-> bring
                        serial port out of reset
;8         IN0       Read-only bit reflecting the state of the CLKR pin
;9         IN1       Read-only bit reflecting the state of the CLKX pin
;10        RRDY      Transition from 0 to 1 indicates data is ready to be read
;11        XRDY      Transition from 0 to 1 indicates data is ready to be sent
;12        XSREMPY   Transmit shift register empty ( Read-only) 0 -> transistor has
                        experienced underflow, 1-> has not experienced underflow
;13        RSRFUL    Receive shift register full flag (Read-only): 0 -> Receiv-
                        er has experienced overrun, 1-> receiver has not experienced
                        overrun
;14        SOFT      Soft bit - 0 -> immediate stop, 1-> stop after word completion
;15        FREE      Free run bit: 0 -> behavior depends on SOFT bit, 1-> free run
                        regardless of SOFT bit

```

Example 3–1. Default Initialization of 'AC01 (Continued)

```

; The system has the following configuration:
; Uses 16-bit data => F0 = 0
; Operates in burst mode => FSM = 1
; CLKX is derived from CLKX pin => MCM = 0
; Frame sync pulses are generated externally by the AIC => TXM = 0
; Therefore, to reset the serial port, the SPC field would have
; 0000 0000 0000 1000
; To pull the serial port out of reset, the SPC field would have
; 0000 0000 1100 1000
K_0      .set      00000000b << 8      ; bits 15–8 to 0 at reset
K_RRST   .set      0b << 7              ; First write to SPC1 is 0
                                              ; second write is 1
K_XRST   .set      0b << 6              ; First write to SPC1 is 0
                                              ; second write is 1

K_TXM    .set      0b << 5
K_MCM    .set      0b << 4
K_FSM    .set      1b << 3              ; Frame Sync mode
K_ZERO   .set      000b << 0
K_SERIAL_RST .set    K_0|K_RRST|K_XRST|K_TXM|K_MCM|K_FSM|K_ZERO
                                              ; first write to SPC1 register
K_RRST1   .set      1b << 7              ; second write to SPC1
K_XRST1   .set      1b << 6              ; second write to SPC1
K_SERIAL_OUT_RST .set  K_0|K_RRST1|K_XRST1|K_TXM|K_MCM|K_FSM|K_ZERO
K_TRGCR_ADDR .set    14h                ; Target/Status I/O address
K_0       .set      0h
K_8000    .set      8000h                ; set bit 15 to pull AIC out
                                              ; of reset

* FILENAME: INTERRUPT.INC
; 541 Interrupt Mask Register (IMR) Organization
*
* | 15 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
* |-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
* | Reserved | INT3 | XINT1 | RINT1 | XINT0 | RINT0 | TINT | INT2 | INT1 | INT0 |
*
* This file includes the IMR and IFR configuration
*****
K_IMR_RESR .set      0000000b << 9      ; reserved space
K_INT3     .set      1b << 8              ; disable INT3
K_XINT1    .set      1b << 7              ; disable transmit interrupt 1
K_RINT1    .set      1b << 6              ; enable receive interrupt 1
K_XINT0    .set      1b << 5              ; disable transmit interrupt 0
K_RINT0    .set      1b << 4              ; disable receive interrupt
K_TINT     .set      1b << 3              ; disable timer interrupt
K_INT2     .set      1b << 2              ; disable INT2
K_INT1     .set      1b << 1              ; disable INT1
K_INT0     .set      1b << 1              ; enable INT0

```

Example 3–2 on page 3-12 demonstrates initiation of a secondary communication interval for programming the 'AC01. The configuration of the 'AC01 can be changed via its nine internal registers. To understand this, consider registers 1 and 2 in the 'AC01.

The data content of register 1, also referred to as the register A, determines the divisions of the master clock frequency that produce the internal filter clock, FCLK. The default value of this register is 18 (decimal). The filter clock frequency is calculated using the equation:

$$\text{FCLK frequency} = \text{master clock frequency} / (\text{register A contents} \times 2)$$

This means that for an MCLK of 10.368 MHz, the default value of the filter clock frequency is

$$\text{FCLK} = 288 \text{ kHz}$$

This FCLK determines the –3 dB corner frequency of the low-pass filter which is given by:

$$f_{LP} = \text{FCLK}/40$$

The default value of f_{LP} is equal to 7.2 kHz.

The data content of register 2 of the 'AC01, also referred to as register B, determines the division of FCLK to generate the conversion clock, and is given by the equation:

$$\text{Conversion frequency} = \text{FCLK}/(\text{register B contents})$$

The default value of register B is equal to 18 (decimal). Hence, the default value of the conversion (sampling) frequency is equal to:

$$288 \text{ kHz} / 18 = 16 \text{ kHz}$$

This register also determines the –3 dB corner frequency of the high-pass filter, which is given by:

$$f_{HP} = \text{sampling frequency}/200$$

Hence, the default value of f_{HP} is equal to 80 Hz.

For a system that processes speech signals, the following parameters are desirable:

$$\text{Sampling frequency } f_s = 8.0 \text{ kHz}$$

$$\text{Low-pass filter corner frequency, } f_{LP} = 3.6 \text{ kHz}$$

Assume that the 'AC01 uses a master clock frequency, MCLK, of 10.368 MHz. The 'AC01's parameters are:

$$f_{LP} = FCLK/40 \text{ gives } FCLK = 144 \text{ kHz}$$

$$FCLK = MCLK/(\text{register A} \times 2) \text{ gives register A} = 36 \text{ (decimal)}$$

$$f_s = FCLK/B \text{ gives register B} = 18 \text{ (decimal)}$$

This also means that with the specified sampling frequency, the -3 dB corner frequency of the high-pass filter changes to $8.0 \text{ kHz} / 200 = 48 \text{ Hz}$.

After the 'AC01 has been reset in the main program, it makes a call to the routine `wrt_cnfg`, which programs the new values of registers A and B as calculated above. To prevent the occurrence of an interrupt while programming the 'AC01, `wrt_cnfg` disables all interrupts until the end of programming. After each word has been sent to the serial port, the code waits for the data to be copied from the data transmit register to the transmit shift register before it sends the next data. After the 'AC01 has been programmed, the main routine waits for an interrupt. The service routines that transfer data between the memory buffers and the serial port transmit and receive registers remain the same.

Example 3–2. Initiation of a Secondary Communication Interval for Programming the 'AC01

```

;TEXAS INSTRUMENTS INCORPORATED
    .mmregs
    .include      "interrpt.inc"
    .include      "init_ser.inc"
AIC_VAR_DP      .usect "aic_vars",0
aic_in_rst      .usect "aic_vars",1
aic_out_of_rst  .usect "aic_vars",1
    .def      serial_init
;-----
; Functional Description
; This routine initializes the serial port 1 of 541. The serial port is put
; in reset by writing 0's to RRST and XRST bits and pulled out of reset by
; writing 1's to both RRST and XRST bits. This routine also puts the AC01
; in reset and after 12 cycles the AC01 is pulled out of reset. The serial
; port initialization is done during the 12 cycle latency of the AC01 init.
;-----
    .sect      "ser_cnfg"
serial_init:
    LD      #AIC_VAR_DP,DP      ; initialize DP for aic_reset
    ST      #K_0, aic_in_rst    ; bit 15 = 0 of TCR resets AIC
    PORTW   aic_in_rst,K_TRGCR_ADDR ;write to address 14h (TCR)
*****
*We need at least 12 cycles to pull the AIC out of reset.
*****
    STM      #K_SERIAL_RST, SPC1      ;reset the serial port with
                                        ;0000 0000 0000 1000
    STM      #K_SERIAL_OUT_RST, SPC1  ;bring ser.port out of reset
                                        ;0000 0000 1100 1000

    RSBX     INTM
    LD      #0,DP
    ORM      #(K_RINT1|K_INT1),IMR    ; Enable RINT1,INT1
                                        ; 0000 0000 0100 0010
    LD      #AIC_VAR_DP,DP          ; restore DP
    STM      #(K_RINT1),IFR          ; clear RINT1
    STM      #K_0,DXR1              ; 0000 0000 0100 0000
; Pull the AC01 out of reset - the AC01 requires that it be held in reset for
; 1 MCLK, which is equivalent to 96.45ns (based on an MCLK of 10.368MHz)
    ST      #K_8000, aic_out_of_rst  ; bit 15 = 1 brings AIC from reset
    RETD
    PORTW   aic_out_of_rst, K_TRGCR_ADDR ;AIC out of reset
    .end
; TEXAS INSTRUMENTS INCORPORATED
    .mmregs
    .include      "interrpt.inc"
    .ref      wrt_cnfg      ; initializes AC01
    .def      aic_init
;-----
; Functional Description
; This routine disables IMR and clears any pending interrupts before
; initializing AC01. The wrt_cnfg function configures the AC01
;-----
    .sect      "aic_cnfg"

```

Example 3–2. Initiation of a Secondary Communication Interval for Programming the 'AC01(Continued)

```

aic_init:
    CALLD wrt_cnfg ; initialize AC01
    ANDM  #(~K_RINT1|K_INT1),IMR ; disable receive_int1,INT1
    ORM   #(K_RINT1|K_INT1),IMR ; enable the RINT1, INT1
    RETD
    STM   #(K_RINT1),IFR ; service any pending interrupt
    .end
*****
* This file includes the AC01 registers initialization
* All registers have 2 control bits that initiates serial communication
* There are 2 communication modes - primary and secondary communications
* During primary communication the control bits D00 and D01 are 11 to request
* for a secondary communication. In the secondary serial communications the
* control bits D15 and D14 perform same control function as primary.
* The R/W~ bit at reset is set to 0 placing the device in write mode.
*****
K_NOP_ADDR      .set  0 << 8
K_REG_0          .set  K_NOP_ADDR
K_A_ADDR        .set  1 << 8 ; REG 1 address
K_A_REG         .set  36
K_REG_1          .set  K_A_ADDR|K_A_REG ; FCLK = 144KHz => A =24h
K_B_ADDR        .set  2 << 8 ; REG 2 address
K_B_REG         .set  18
K_REG_2          .set  K_B_ADDR|K_B_REG ; Sampling rate = 8KHz
K_AA_ADDR       .set  3 << 8 ; Register 3 address
K_AA_REG        .set  0
K_REG_3          .set  K_AA_ADDR|K_AA_REG ; ; no shift
K_GAIN_ADDR     .set  4 << 8 ; Register 4 address
K_MONITOR_GAIN  .set  00b << 4 ; Monitor output gain = squelch
K_ANLG_IN_GAIN  .set  01b << 2 ; Analog input gain = 0dB
K_ANLG_OUT_GAIN .set  01b << 0 ; Analog output gain = 0dB
K_REG_4          .set  K_GAIN_ADDR|K_MONITOR_GAIN|K_ANLG_IN_GAIN|K_ANLG_OUT_GAIN
K_ANLG_CNF_ADDR .set  5 << 8 ; Register 5 address
K_ANLG_RESRV    .set  0 << 3 ; Must be set to 0K_HGH_FILTER
                .set  0 << 2 ; High pass filter is enabled
K_ENBL_IN       .set  01b << 0 ; Enables IN+ and IN-
K_REG_5          .set  K_ANLG_CNF_ADDR|K_ANLG_RESRV|K_HGH_FILTER|K_ENBL_IN
K_DGTL_CNF_ADDR .set  6 << 8 ; Register 6 address
K_ADC_DAC       .set  0 << 5 ; ADC and DAC is inactive
K_FSD_OUT        .set  0 << 4 ; Enabled FSD output
K_16_BIT_COMM   .set  0 << 3 ; Normal 16-bit mode
K_SECND_COMM    .set  0 << 2 ; Normal secondary communication
K_SOFT_RESET    .set  0 << 1 ; Inactive reset
K_POWER_DWN     .set  0 << 0 ; Power down external
K_REG_HIGH_6    .set  K_DGTL_CNF_ADDR|K_ADC_DAC|K_FSD_OUT|K_16_BIT_COMM
K_REG_LOW_6     .set  K_SECND_COMM|K_SOFT_RESET|K_POWER_DWN
K_REG_6         .set  K_REG_HIGH_6|K_REG_LOW_6
K_FRME_SYN_ADDR .set  7 << 8 ; Register 7 address
K_FRME_SYN      .set  0 << 8 ;
K_REG_7         .set  K_FRME_SYN_ADDR|K_FRME_SYN
K_FRME_NUM_ADDR .set  8 << 8 ; Register 8 address
K_FRME_NUM      .set  0 << 8 ;

```

Example 3–2. Initiation of a Secondary Communication Interval for Programming the 'AC01(Continued)

```

K_REG_8      .set    K_FRME_NUM_ADDR|K_FRME_NUM
; primary word with D01 and D00 bits set to 11 will cause a
; secondary communications interval to start when the frame
; sync goes low next
K_SCND_CONTRL.set    11b << 0 ; Secondary comm.bits
AIC_REG_START_LIST .sect    "aic_reg" ; includes the aic table
    .word    AIC_REG_END_LIST-AIC_REG_START_LIST-1
    .word    K_REG_1
    .word    K_REG_2
    .word    K_REG_3
    .word    K_REG_4
    .word    K_REG_5
    .word    K_REG_6
    .word    K_REG_7
    .word    K_REG_8
AIC_REG_END_LIST
K_XRDY      .set      0800h ; XRDY bit in SPC1
; TEXAS INSTRUMENTS INCORPORATED
    .mmregs
    .include    "interrpt.inc"
    .ref    wrt_cnfg ; initializes AC01
    .def    aic_init
;-----
; Functional Description
; This routine disables IMR and clears any pending interrupts before
; initializing AC01. The wrt_cnfg function configures the AC01
;-----
    .sect    "aic_cnfg"
aic_init:
    CALLD    wrt_cnfg ; initialize AC01
    ANDM    #(~K_RINT1),IMR ; disable receive_int1
    ORM     #(K_RINT1|K_INT1),IMR ; enable the RINT1, INT1
    RETD
    STM     #(K_RINT1),IFR ; service any pending interrupt
    .end
; TEXAS INSTRUMENTS INCORPORATED
    .mmregs
    .include    "aic_cfg.inc"
aic_reg_tble .usect    "aic_vars",10
    .def    wrt_cnfg
;
; Functional Description
; Writes new configuration data into the AC01. Assuming a system
; which processes speech signals and * requires the following parameters
; Low pass filter cut-off frequency = 3.6 kHz
; Sampling rate = 8000 Hz
; Assume the Master clock MCLK = 10.368 MHz
; This example demonstrates how to program these parameters -
; the registers affected are:
; Register A which determines the division of the MCLK frequency
; to generate the internal filter clock FCLK.
; It also determines the -3 dB corner frequency of the low-pass filter

```

Example 3–2. Initiation of a Secondary Communication Interval for Programming the 'AC01(Continued)

```

;      Register B  which determines the division of FCLK to generate
;      the sampling (conversion) frequency
;      It also determines the -3dB corner frequency of the high-pass filter
;-----
      .asg  AR1,AIC_REG_P
      .sect "aic_cnfg"
wrt_cnfg:
      STM   #aic_reg_tble,AIC_REG_P           ; init AR1
      RPT   #AIC_REG_END_LIST-AIC_REG_START_LIST
      MVPD  #AIC_REG_START_LIST,*AIC_REG_P+   ; move the table
      STM   #aic_reg_tble,AIC_REG_P           ; init AR1
      STM   #K_REG_0,DXR1                     ; primary data word -
                                              ; a jump start!

wait_xrdy
      BITF  SPC1,K_XRDY                       ;test XRDY bit in SPC1
      BC    wait_xrdy,NTC                     ;loop if not set
      STM   #K_SCND_CONTRL,DXR1               ;send primary word with
                                              ; D01-D00 = 11 to
                                              ; signify secondary communication

      LD    *AIC_REG_P+,A
      STLM  A,BRC                             ; gives the # of registers to be
      NOP                                     ; initialized

      RPTB  aic_cfg_complte-1
wait_xrdy1
      BITF  SPC1,K_XRDY                       ;test XRDY bit in SPC1
      BC    wait_xrdy1,NTC                    ;loop if not set
      LD    *AIC_REG_P+,A                     ; Read the register contents
      STLM  A,DXR1
wait_xrdy2
      BITF  SPC1,K_XRDY                       ;test XRDY bit in SPC1
      BC    wait_xrdy2,NTC                    ;loop if not set
      STM   #K_SCND_CONTRL,DXR1               ; set to read the next register
aic_cfg_complte                             ; contents
wait_xrdy3
      BITF  SPC1,K_XRDY                       ;test XRDY bit in SPC1
      BC    wait_xrdy3,NTC                    ;loop if not set
      RET

```

Example 3–3 on page 3-17 uses two 'AC01s in master-slave configuration. After the 'AC01s have been reset, the 'AC01s make a call to the routine `wrt_cnfg`. This routine programs the low- and high-pass filter cutoff frequencies, the number of slave devices attached, and sets up the frame-sync delay register. During the programming mode, the master and its slaves are programmed with the same control information. The last register programmed is frame-sync delay, which determines the lag between the frame-sync and delayed frame-sync signals. The minimum value for the frame-sync delay register is 18 (decimal). This is because every data transfer requires 16 clock periods, and the output data, DOUT, is placed in the high-impedance state on the 17th rising edge of the clock.

Example 3–3 uses a minimum value of 18, which means that the time delay between the frame sync and the delayed frame sync is equal to 18 multiplied by the period of SCLK. With $SCLK = 2.592\text{ MHz}$ based on an MCLK of 10.368 MHz , the time delay is $6.94\mu\text{s}$. The hardware schematic indicates that the delayed frame-sync signal from the master is the frame-sync signal for the slave. The master, therefore, generates a frame-sync signal for itself and a delayed frame-sync signal for its slave. It then synchronizes the host serial port for data transfers to itself and its slaves. All interrupts on the host are disabled during the time the 'AC01s are programmed. This disallows the host from receiving or sending any data to or from the 'AC01s.

The output data (DOUT) from the 'AC01 operating in the master mode is available at the falling edge of the master frame-sync signal. In the primary communication interval, this data represents the 14-bit A/D conversion result. The two least significant bits (LSBs), D01 and D00, determine whether the data is generated by the master or its slave as follows:

	D01	D00
Master mode	0	0
Slave mode	0	1

In the secondary communication interval, the data available at DOUT represents the contents of the register read with the eight MSBs set to 0 and the read/write bit set to logic 1 during the primary interval. If no register read is requested, the second word is all 0s.

The output data at DOUT from the 'AC01 operating in slave mode is available at either the falling edge of the external frame sync signal (which is the delayed frame sync from the master) or the rising edge of the external SCLK signal, whichever comes first.

When the AC01 receives an interrupt, the service routine first determines whether the received data is from the master or the slave by checking the LSBs of the received data at bit positions D01 and D00. If the LSBs are 00, the received data is stored in a memory buffer, which holds all data received from the master. The service routine also sends data from a memory buffer, which holds all data going the master. If the LSBs of the received data are 01, data is received from and sent to the slave 'AC01, using dedicated memory buffers. This can be used with double buffering or circular buffers to allow the ISR to notify the main routine. This occurs when new input data is available for processing and/or space is available to accept processed data for output to or from the master and the slave.

Example 3–3. Master-Slave Mode

```
*****
* This file includes the AC01 registers initialization
* All registers have 2 control bits that initiates serial communication
* There are 2 communication modes - primary and secondary communications
* During primary communication the control bits D00 and D01 are 11 to request
* for a secondary communication. In the secondary serial communications the
* control bits D15 and D14 perform same control function as primary.
* The R/W~ bit at reset is set to 0 placing the device in write mode.
* DS7-DS00 -> Number of SCLK's between FS~ and FSD~. When slaves are used,
* this should be the last register to be programmed The minimum value for this
* register should be decimal 18. This means that the time interval between the
* FS~ and FSD~ with an SCLK * frequency of 2.592 MHz (with an MCLK of 10.368 MHz)
* is 18 1/2.592 MHz = 6.94 us
; DS7-DS00 -> # of frame sync signals generated by AC01
; Therefore, number of frame sync signals number of slave = devices + 1
; In programming the AC01 REG7 is the last register to be programmed if the
; configuration is master slave mode configuration
*****
K_NOP_ADDR      .set    0 << 8
K_REG_0         .set    K_NOP_ADDR
K_A_ADDR        .set    1 << 8           ; REG 1 address
K_A_REG         .set    36
K_REG_1         .set    K_A_ADDR|K_A_REG ; FCLK = 144KHz => A =24h
K_B_ADDR        .set    2 << 8           ; REG 2 address
K_B_REG         .set    18
K_REG_2         .set    K_B_ADDR|K_B_REG ; Sampling rate = 8KHz
K_AA_ADDR       .set    3 << 8           ; Register 3 address
K_AA_REG        .set    0
K_REG_3         .set    K_AA_ADDR|K_AA_REG ; no shift
K_GAIN_ADDR     .set    4 << 8           ; Register 4 address
K_MONITOR_GAIN  .set    00b << 4         ; Monitor output gain = squelch
K_ANLG_IN_GAIN  .set    01b << 2         ; Analog input gain = 0dB
K_ANLG_OUT_GAIN .set    01b << 0         ; Analog output gain = 0dB
K_REG_4         .set    K_GAIN_ADDR|K_MONITOR_GAIN|K_ANLG_IN_GAIN|K_ANLG_OUT_GAIN
K_ANLG_CNF_ADDR .set    5 << 8           ; Register 5 address
K_ANLG_RESRV    .set    0 << 3           ; Must be set to 0
K_HGH_FILTER    .set    0 << 2           ; High pass filter is enabled
K_ENBL_IN       .set    01b << 0         ; Enables IN+ and IN-
```

Example 3–3. Master-Slave Mode (Continued)

```

K_REG_5      .set    K_ANLG_CNF_ADDR|K_ANLG_RESRV|K_HGH_FILTER|K_ENBL_IN
K_DGTL_CNF_ADDR .set    6 << 8                ; Register 6 address
K_ADC_DAC    .set    0 << 5                ; ADC and DAC is inactive
K_FSD_OUT    .set    0 << 4                ; Enabled FSD output
K_16_BIT_COMM .set    0 << 3                ; Normal 16-bit mode
K_SECND_COMM .set    0 << 2                ; Normal secondary communication
K_SOFT_RESET .set    0 << 1                ; Inactive reset
K_POWER_DWN  .set    0 << 0                ; Power down external
K_REG_HIGH_6 .set    K_DGTL_CNF_ADDR|K_ADC_DAC|K_FSD_OUT|K_16_BIT_COMM
K_REG_LOW_6  .set    K_SECND_COMM|K_SOFT_RESET|K_POWER_DWN
K_REG_6      .set    K_REG_HIGH_6|K_REG_LOW_6
K_FRME_SYN_ADDR .set    7 << 8                ; Register 7 address
K_FRME_SYN    .set    18 << 8                ;
K_REG_7      .set    K_FRME_SYN_ADDR|K_FRME_SYN
K_FRME_NUM_ADDR .set    8 << 8                ; Register 8 address
K_FRME_NUM    .set    2 << 8                ;
K_REG_8      .set    K_FRME_NUM_ADDR|K_FRME_NUM
; primary word with D01 and D00 bits set to 11 will cause a
; secondary communications interval to start when the frame
; sync goes low next
K_SCND_CONTRL .set    11b << 0                ; Secondary comm.bits
AIC_REG_START_LIST .sect    "aic_reg"        ; includes the aic table
    .word    AIC_REG_END_LIST-AIC_REG_START_LIST-1
    .word    K_REG_1
    .word    K_REG_2
    .word    K_REG_3
    .word    K_REG_4
    .word    K_REG_5
    .word    K_REG_6
    .word    K_REG_8
    .word    K_REG_7
; this should be the last
; register to be programmed
AIC_REG_END_LIST
K_XRDY      .set    0800h                ; XRDY bit in SPC1

```