Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8052
Core Size	8-Bit
Speed	16MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	PSM, Temp Sensor, WDT
Number of I/O	34
Program Memory Size	62KB (62K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	2.25K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	52-QFP
Supplier Device Package	80-PQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/aduc831bsz

The typical performance plots presented in this section illustrate typical performance of the ADuC831 under various operating conditions.

TPC 1 and TPC 2 below show typical ADC Integral Nonlinearity (INL) errors from ADC code 0 to code 4095 at 5 V and 3 V supplies respectively. The ADC is using its internal reference (2.5 V) and operating at a sampling rate of 152 kHz and the typically worst-case errors in both plots is just less than 0.3 LSBs.

TPC 3 and TPC 4 below show the variation in Worst Case Positive (WCP) INL and Worst Case Negative (WCN) INL versus external reference input voltage.

TPC 5 and TPC 6 show typical ADC differential nonlinearity (DNL) errors from ADC code 0 to code 4095 at 5 V and 3 V supplies, respectively. The ADC is using its internal reference (2. V) and operating at a sampling rate of 152 kHz and the typically worst case errors in both plots is just less than 0.2 LSBs.

TPC 7 and TPC 8 show the variation in worst case positive (WCP) DNL and worst-case negative (WCN) DNL versus external reference input voltage.

TPC 9 shows a histogram plot of 10,000 ADC conversion results on a dc input with V_{DD} = 5 V. The plot illustrates an excellent code distribution pointing to the low noise performance of the on-chip precision ADC.

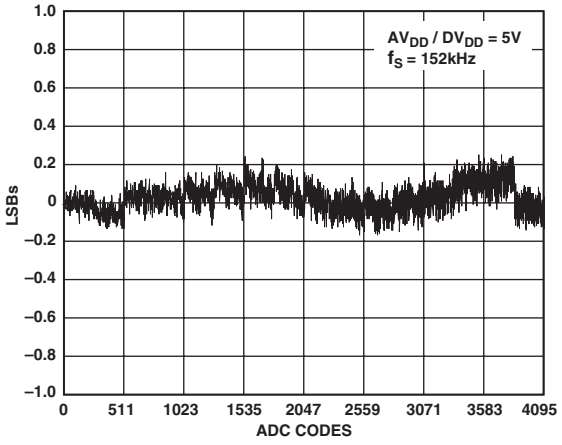
TPC 10 shows a histogram plot of 10,000 ADC conversion results on a dc input for V_{DD} = 3 V. The plot again illustrates a very tight code distribution of 1 LSB with the majority of codes appearing in one output bin.

TPC 11 and TPC 12 show typical FFT plots for the ADuC831. These plots were generated using an external clock input. The ADC is using its internal reference (2.5 V) sampling a full-scale, 10 kHz sine wave test tone input at a sampling rate of 149.79 kHz. The resultant FFTs shown at 5 V and 3 V supplies illustrate an excellent 100 dB noise floor, 71 dB Signal-to-Noise Ratio (SNR) and THD greater than –80 dB.

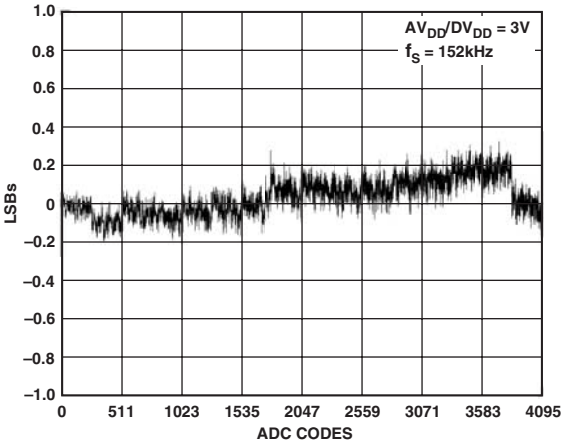
TPC 13 and TPC 14 show typical dynamic performance versus external reference voltages. Again excellent ac performance can be observed in both plots with some roll-off being observed as V_{REF} falls below 1 V.

TPC 15 shows typical dynamic performance versus sampling frequency. SNR levels of 71 dBs are obtained across the sampling range of the ADuC831.

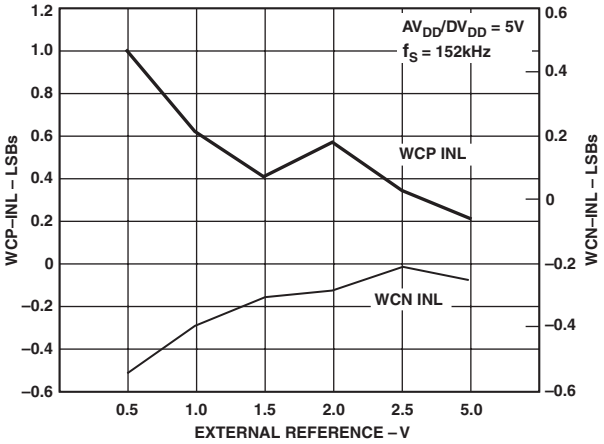
TPC 16 shows the voltage output of the on-chip temperature sensor versus temperature. Although the initial voltage output at 25°C can vary from part to part, the resulting slope of –2 mV/°C is constant across all parts.



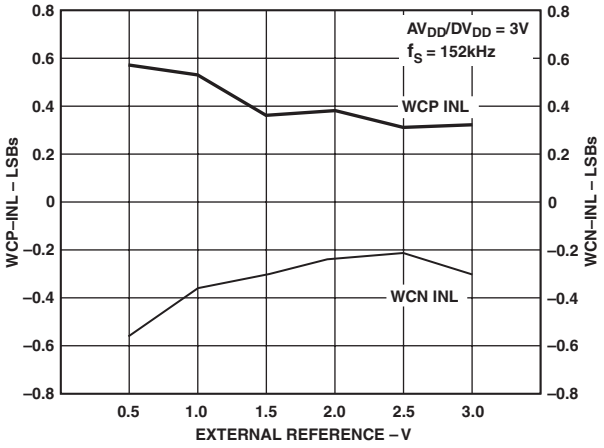
TPC 1. Typical INL Error, V_{DD} = 5 V



TPC 2. Typical INL Error, V_{DD} = 3 V



TPC 3. Typical Worst Case INL Error vs. V_{REF}, V_{DD} = 5 V



TPC 4. Typical Worst Case INL Error vs. V_{REF}, V_{DD} = 3 V

External Data Memory (External XRAM)
Just like a standard 8051 compatible core, the ADuC831 can access external data memory using a MOVX instruction. The MOVX instruction automatically outputs the various control strobes required to access the data memory.

The ADuC831, however, can access up to 16 MBytes of external data memory. This is an enhancement of the 64 kBytes external data memory space available on a standard 8051 compatible core.

The external data memory is discussed in more detail in the ADuC831 Hardware Design Considerations section.

Internal XRAM
2 kBytes of on-chip data memory exist on the ADuC831. This memory, although on-chip, is also accessed via the MOVX instruction. The 2 kBytes of internal XRAM are mapped into the bottom 2 kBytes of the external address space if the CFG831 bit is set. Otherwise, access to the external data memory will occur just like a standard 8051. When using the internal XRAM, ports 0 and 2 are free to be used as general-purpose I/O.

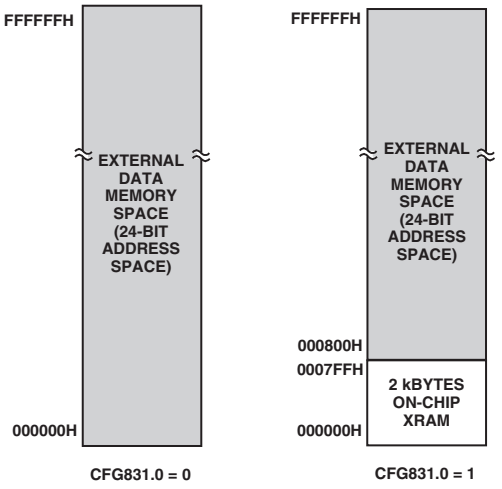


Figure 4. Internal and External XRAM

SPECIAL FUNCTION REGISTERS (SFRS)
The SFR space is mapped into the upper 128 bytes of internal data memory space and accessed by direct addressing only. It provides an interface between the CPU and all on-chip peripherals. A block diagram showing the programming model of the ADuC831 via the SFR area is shown in Figure 5.

All registers, except the Program Counter (PC) and the four general-purpose register banks, reside in the SFR area. The SFR registers include control, configuration, and data registers that provide an interface between the CPU and all on-chip peripherals.

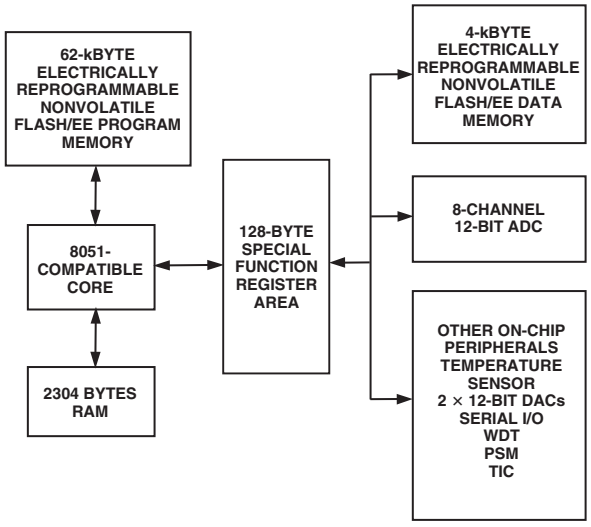


Figure 5. Programming Model

Accumulator SFR (ACC)
ACC is the Accumulator register and is used for math operations including addition, subtraction, integer multiplication and division, and Boolean bit manipulations. The mnemonics for accumulator-specific instructions refer to the Accumulator as A.

B SFR (B)
The B register is used with the ACC for multiplication and division operations. For other instructions it can be treated as a general-purpose scratchpad register.

Stack Pointer (SP and SPH)
The SP SFR is the stack pointer and is used to hold an internal RAM address that is called the *top of the stack*. The SP register is incremented before data is stored during PUSH and CALL executions. While the Stack may reside anywhere in on-chip RAM, the SP register is initialized to 07H after a reset. This causes the stack to begin at location 08H.

As mentioned earlier, the ADuC831 offers an extended 11-bit stack pointer. The three extra bits to make up the 11-bit stack pointer are the 3 LSBs of the SPH byte located at B7H.

Driving the A/D Converter

The ADC incorporates a successive approximation (SAR) architecture involving a charge-sampled input stage. Figure 9 shows the equivalent circuit of the analog input section. Each ADC conversion is divided into two distinct phases as defined by the position of the switches in Figure 9. During the sampling phase (with SW1 and SW2 in the "track" position) a charge proportional to the voltage on the analog input is developed across the input sampling capacitor. During the conversion phase (with both switches in the "hold" position) the capacitor DAC is adjusted via internal SAR logic until the voltage on node A is zero, indicating that the sampled charge on the input capacitor is balanced out by the charge being output by the capacitor DAC. The digital value finally contained in the SAR is then latched out as the result of the ADC conversion. Control of the SAR, and timing of acquisition and sampling modes, is handled automatically by built-in ADC control logic. Acquisition and conversion times are also fully configurable under user control.

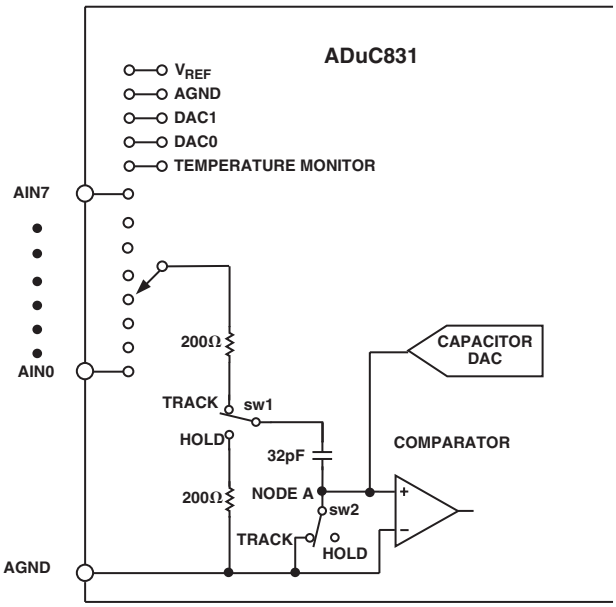


Figure 9. Internal ADC Structure

Note that whenever a new input channel is selected, a residual charge from the 32 pF sampling capacitor places a transient on the newly selected input. The signal source must be capable of recovering from this transient before the sampling switches click into "hold" mode. Delays can be inserted in software (between channel selection and conversion request) to account for input stage settling, but a hardware solution will alleviate this burden from the software design task and will ultimately result in a cleaner system implementation. One hardware solution would be to choose a very fast settling op amp to drive each analog input. Such an op amp would need to fully settle from a small signal transient in less than 300 ns in order to guarantee adequate settling under all software configurations. A better solution, recommended for use with any amplifier, is shown in Figure 10.

Though at first glance the circuit in Figure 10 may look like a simple antialiasing filter, it actually serves no such purpose since its corner frequency is well above the Nyquist frequency, even at a 200 kHz sample rate. Though the R/C does helps to reject some incoming high-frequency noise, its primary function is to ensure that the transient demands of the ADC input stage are met. It

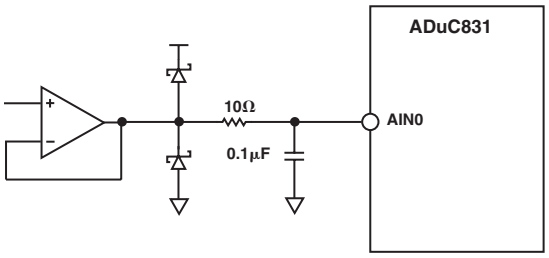


Figure 10. Buffering Analog Inputs

does so by providing a capacitive bank from which the 32 pF sampling capacitor can draw its charge. Its voltage will not change by more than one count (1/4096) of the 12-bit transfer function when the 32 pF charge from a previous channel is dumped onto it. A larger capacitor can be used if desired, but not a larger resistor (for reasons described below).

The Schottky diodes in Figure 10 may be necessary to limit the voltage applied to the analog input pin as per the data sheet absolute maximum ratings. They are not necessary if the op amp is powered from the same supply as the ADuC831 since in that case the op amp is unable to generate voltages above V_{DD} or below ground. An op amp of some kind is necessary unless the signal source is very low impedance to begin with. DC leakage currents at the ADuC831's analog inputs can cause measurable dc errors with external source impedances as little as 100 Ω or so. To ensure accurate ADC operation, keep the total source impedance at each analog input less than 61 Ω. The table below illustrates examples of how source impedance can affect dc accuracy.

Source Impedance	Error from 1 μA Leakage Current	Error from 10 μA Leakage Current
61 Ω	61 μV = 0.1 LSB	610 μV = 1 LSB
610 Ω	610 μV = 1 LSB	6.1 mV = 10 LSB

Although Figure 10 shows the op amp operating at a gain of 1, you can, of course, configure it for any gain needed. Also, you can just as easily use an instrumentation amplifier in its place to condition differential signals. Use any modern amplifier that is capable of delivering the signal (0 to V_{REF}) with minimal saturation. Some single-supply rail-to-rail op amps that are useful for this purpose include, but are certainly not limited to, the ones given in Table VI. Check Analog Devices literature (CD ROM data book, and so on) for details on these and other op amps and instrumentation amps.

Table VI. Some Single-Supply Op Amps

Op Amp Model	Characteristics
OP281/OP481	Micropower
OP191/OP291/OP491	I/O Good up to V_{DD}, Low Cost
OP196/OP296/OP496	I/O to V_{DD}, Micropower, Low Cost
OP183/OP283	High Gain-Bandwidth Product
OP162/OP262/OP462	High GBP, Micro Package
AD820/AD822/AD824	FET Input, Low Cost
AD823	FET Input, High GBP

Configuring the ADC

The ADuC831's successive approximation ADC is driven by a divided down version of the master clock. To ensure adequate ADC operation, this ADC clock must be between 400 kHz and 6 MHz, and optimum performance is obtained with ADC clock between 400 kHz and 4.5 MHz. Frequencies within this range can easily be achieved with master clock frequencies from 400 kHz to well above 16 MHz with the four ADC clock divide ratios to choose from. For example, with a 12 MHz master clock, set the ADC clock divide ratio to 4 (i.e., ADCCLK = MCLK/4 = 3 MHz) by setting the appropriate bits in ADCCON1 (ADCCON1.5 = 1, ADCCON1.4 = 0).

The total ADC conversion time is 15 ADC clocks, plus 1 ADC clock for synchronization, plus the selected acquisition time (1, 2, 3, or 4 ADC clocks). For the example above, with three clocks acquisition time, total conversion time is 19 ADC clocks (or 6.3 μs for a 3 MHz ADC clock).

In continuous conversion mode, a new conversion begins each time the previous one finishes. The sample rate is then simply the inverse of the total conversion time described above. In the example above, the continuous conversion mode sample rate would be 157.8 kHz.

If using the temperature sensor as the ADC input, the ADC should be configured to use an ADCCLK of MCLK/16 and four acquisition clocks.

Increasing the conversion time on the temperature monitor channel improves the accuracy of the reading. To further improve the accuracy, an external reference with low temperature drift should also be used.

ADC DMA Mode

The on-chip ADC has been designed to run at a maximum conversion speed of 4 μs (247 kHz sampling rate). When converting at this rate, the ADuC831 MicroConverter has 4 μs to read the ADC result and store the result in memory for further postprocessing, otherwise the next ADC sample could be lost. In an interrupt driven routine the MicroConverter would also have to jump to the ADC Interrupt Service routine, which will also increase the time required to store the ADC results. In applications where the ADuC831 cannot sustain the interrupt rate, an ADC DMA mode is provided.

To enable DMA mode, Bit 6 in ADCCON2 (DMA) must be set. This allows the ADC results to be written directly to a 16 MByte external static memory SRAM (mapped into data memory space) without any interaction from the ADuC831 core. This mode allows the ADuC831 to capture a contiguous sample stream at full ADC update rates (247 kHz).

A Typical DMA Mode Configuration Example

To set the ADuC831 into DMA mode a number of steps must be followed:

1. The ADC must be powered down. This is done by ensuring MD1 and MD0 are both set to 0 in ADCCON1.
2. The DMA address pointer must be set to the start address of where the ADC results are to be written. This is done by writing to the DMA mode address pointers DMAL, DMAH, and DMAP. DMAL must be written to first, followed by DMAH, and then by DMAP.
3. The external memory must be preconfigured. This consists of writing the required ADC channel IDs into the top four bits of every second memory location in the external SRAM starting at the first address specified by the DMA address pointer. As the ADC DMA mode operates independent from the ADuC831 core, it is necessary to provide it with a stop command. This is done by duplicating the last channel ID to be converted followed by "1111" into the next channel selection field. A typical preconfiguration of external memory is as follows.

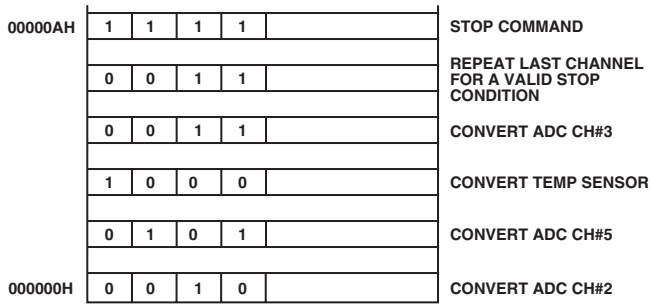


Figure 14. Typical DMA External Memory Preconfiguration

4. The DMA is initiated by writing to the ADC SFRs in the following sequence:
   a. ADCCON2 is written to enable the DMA mode, i.e., MOV ADCCON2, #40H; DMA mode enabled.
   b. ADCCON1 is written to configure the conversion time and power-up of the ADC. It can also enable Timer 2 driven conversions or external triggered conversions if required.
   c. ADC conversions are initiated. This is done by starting single conversions, starting Timer 2 running for Timer 2 conversions or by receiving an external trigger.

When the DMA conversions are completed, the ADC interrupt bit ADCI, is set by hardware and the external SRAM contains the new ADC conversion results as shown below. It should be noted that no result is written to the last two memory locations.

When the DMA mode logic is active, it takes the responsibility of storing the ADC results away from both the user and ADuC831 core logic. As it writes the results of the ADC conversions to external memory, it takes over the external memory interface from the core. Thus, any core instructions that access the external memory while DMA mode is enabled will not get access to it. The core will execute the instructions and they will take the same time to execute but they will not gain access to the external memory.

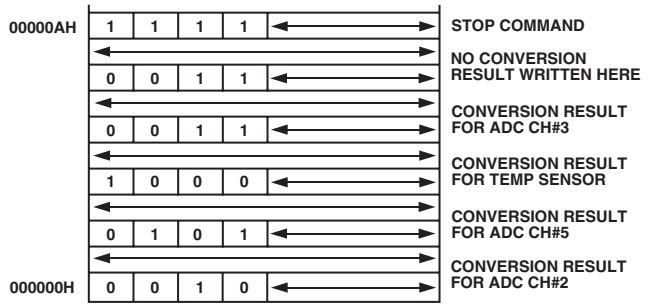


Figure 15. Typical External Memory Configuration Post ADC DMA Operation

The DMA logic operates from the ADC clock and uses pipelining to perform the ADC conversions and access the external memory at the same time. The time it takes to perform one ADC conversion is called a DMA cycle. The actions performed by the logic during a typical DMA cycle are shown in the following diagram.

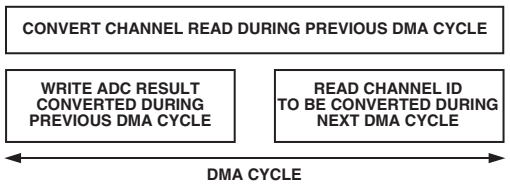


Figure 16. DMA Cycle

From the previous diagram, it can be seen that during one DMA cycle the following actions are performed by the DMA logic:

1. An ADC conversion is performed on the channel whose ID was read during the previous cycle.
2. The 12-bit result and the channel ID of the conversion performed in the previous cycle is written to the external memory.
3. The ID of the next channel to be converted is read from external memory.

For the previous example, the complete flow of events is shown in Figure 16. Because the DMA logic uses pipelining, it takes three cycles before the first correct result is written out.

Micro Operation during ADC DMA Mode

During ADC DMA mode the MicroConverter core is free to continue code execution, including general housekeeping and communication tasks. However, note that MCU core accesses to Ports 0 and 2 (which of course are being used by the DMA controller) are gated "OFF" during ADC DMA mode of operation. This means that even though the instruction that accesses the external Ports 0 or 2 will appear to execute, no data will be seen at these external ports as a result. Note that during DMA the internally contained XRAM Ports 0 and 2 are available for use.

The only case in which the MCU will be able to access XRAM during DMA, is when the internal XRAM is enabled and the section of RAM to which the DMA ADC results are being written to lies in an external XRAM. Then the MCU will be able to access the internal XRAM only. This is also the case for use of the extended stack pointer.

The MicroConverter core can be configured with an interrupt to be triggered by the DMA controller when it had finished filling the requested block of RAM with ADC results, allowing the service routine for this interrupt to postprocess data without any real-time timing constraints.

ADC Offset and Gain Calibration Coefficients

The ADuC831 has two ADC calibration coefficients, one for offset calibration and one for gain calibration. Both the offset and gain calibration coefficients are 14-bit words, and are each stored in two registers located in the Special Function Register (SFR) area. The offset calibration coefficient is divided into ADCOFSH (six bits) and ADCOFSL (eight bits) and the gain calibration coefficient is divided into ADCGAINH (six bits) and ADCGAINL (eight bits).

The offset calibration coefficient compensates for dc offset errors in both the ADC and the input signal. Increasing the offset coefficient compensates for positive offset, and effectively pushes the ADC transfer function down. Decreasing the offset coefficient compensates for negative offset, and effectively pushes the ADC transfer function up. The maximum offset that can be compensated is typically ±5% of V_{REF}, which equates to typically ±125 mV with a 2.5 V reference.

Similarly, the gain calibration coefficient compensates for dc gain errors in both the ADC and the input signal. Increasing the gain coefficient compensates for a smaller analog input signal range and scales the ADC transfer function up, effectively increasing the slope of the transfer function. Decreasing the gain coefficient, compensates for a larger analog input signal range and scales the ADC transfer function down, effectively decreasing the slope of the transfer function. The maximum analog input signal range for which the gain coefficient can compensate is $1.025 \times V_{REF}$ and the minimum input range is $0.975 \times V_{REF}$ which equates to typically ±2.5% of the reference voltage.

CALIBRATING THE ADC

There are two hardware calibration modes provided which can be easily initiated by user software. The ADCCON3 SFR is used to calibrate the ADC. Bit 1 (TYPICAL) and the CS3 to CS0 (ADCCON2) set up the calibration modes.

Device calibration can be initiated to compensate for significant changes in operating conditions frequency, analog input range, reference voltage and supply voltages. In this calibration mode, offset calibration uses internal AGND selected via ADCCON2 register bits CS3–CS0 (1011) and gain calibration uses internal V_{REF} selected by CS3–CS0 (1100). Offset calibration should be executed first, followed by gain calibration.

System calibration can be initiated to compensate for both internal and external system errors. To perform system calibration using an external reference, tie system ground and reference to any two of the six selectable inputs. Enable external reference mode (ADCCON1.6). Select the channel connected to AGND via CS3–CS0 and perform system offset calibration. Select the channel connected to V_{REF} via CS3–CS0 and perform system gain calibration.

The ADC should be configured to use settings for an ADCCLK of divide by 16 and 4 acquisition clocks.

INITIATING CALIBRATION IN CODE

When calibrating the ADC, using ADCCON1 the ADC should be set up into the configuration in which it will be used. The ADCCON3 register can then be used to set the device up and calibrate the ADC offset and gain.

```
MOV ADCCON1,#08CH     ;ADC on; ADCCLK set
                      ;to divide by 16, 4
                      ;acquisition clock
```

To calibrate device offset:

```
MOV ADCCON2,#0BH      ;select internal AGND
MOV ADCCON3,#25H      ;select offset calibration,
                      ;31 averages per bit,
                      ;offset calibration
```

To calibrate device gain:

```
MOV ADCCON2,#0CH      ;select internal VREF
MOV ADCCON3,#27H      ;select offset calibration,
                      ;31 averages per bit,
                      ;offset calibration
```

To calibrate system offset:
Connect system AGND to an ADC channel input (0).

```
MOV ADCCON2,#00H      ;select external AGND
MOV ADCCON3,#25H      ;select offset calibration,
                      ;31 averages per bit
```

To calibrate system gain:
Connect system V_{REF} to an ADC channel input (1).

```
MOV ADCCON2,#01H      ;select external VREF
MOV ADCCON3,#27H      ;select offset calibration,
                      ;31 averages per bit,
                      ;offset calibration
```

The calibration cycle time, T_{CAL}, is calculated by the following equation assuming a 16 MHz crystal:

$$T_{CAL} = 14 \times ADCCLK \times NUMAV \times (16 + T_{ACQ})$$

For an ADCCLK/F_{CORE} divide ratio of 16, a T_{ACQ} = 4 ADCCLK, NUMAV = 15, the calibration cycle time is:

$$T_{CAL} = 14 \times (1/1000000) \times 15 \times (16 + 4)$$
$$T_{CAL} = 4.2\ ms$$

In a calibration cycle the ADC busy flag (Bit 7), instead of framing an individual ADC conversion as in normal mode, will go high at the start of calibration and only return to zero at the end of the calibration cycle. It can therefore be monitored in code to indicate when the calibration cycle is completed. The following code can be used to monitor the BUSY signal during a calibration cycle:

```
WAIT:
   MOV A, ADCCON3    ;move ADCCON3 to A
   JB ACC.7, WAIT    ;If Bit 7 is set jump to
                      WAIT else continue
```

USING THE FLASH/EE DATA MEMORY

The 4 kBytes of Flash/EE data memory is configured as 1024 pages, each of four bytes. As with the other ADuC831 peripherals, the interface to this memory space is via a group of registers mapped in the SFR space. A group of four data registers (EDATA1–4) are used to hold the four bytes of data at each page. The page is addressed via the two registers EADRH and EADRL. Finally, ECON is an 8-bit control register that may be written with one of nine Flash/EE memory access commands to trigger various read, write, erase, and verify functions.

A block diagram of the SFR interface to the Flash/EE data memory array is shown in Figure 20.

ECON—Flash/EE Memory Control SFR

Programming of either the Flash/EE data memory or the Flash/EE program memory is done through the Flash/EE memory control SFR (ECON). This SFR allows the user to read, write, erase, or verify the 4 kBytes of Flash/EE data memory or the 56 kBytes of Flash/EE program memory.

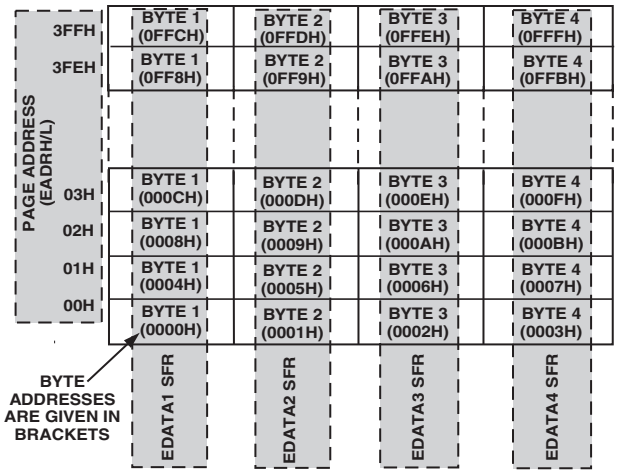


Figure 20. Flash/EE Data Memory Control and Configuration

Table VII. ECON—Flash/EE Memory Commands

ECON VALUE	COMMAND DESCRIPTION (NORMAL MODE) (Power-On Default)	COMMAND DESCRIPTION (ULOAD MODE)
01H READ	Results in 4 bytes in the Flash/EE data memory, addressed by the page address EADRH/L, being read into EDATA1–4.	Not Implemented. Use the MOVC instruction.
02H WRITE	Results in four bytes in EDATA1–4 being written to the Flash/EE data memory, at the page address given by EADRH/L (0 ≤ EADRH/L < 0400H. Note: The four bytes in the page being addressed must be pre-erased.	Results in bytes 0-255 of internal XRAM being written to the 256 bytes of Flash/EE program memory at the page address given by EADRH. (0 ≤ EADRH < E0H) Note: The 256 bytes in the page being addressed must be pre-erased.
03H	Reserved Command	Reserved Command
04H VERIFY	Verifies if the data in EDATA1–4 is contained in the page address given by EADRH/L. A subsequent read of the ECON SFR will result in a 0 being read if the verification is valid, or a nonzero value being read to indicate an invalid verification.	Not Implemented. Use the MOVC and MOVX instructions to verify the WRITE in software.
05H ERASE PAGE	Results in the Erase of the 4-byte page of Flash/EE data memory addressed by the page address EADRH/L.	Results in the 64-byte page of Flash/EE program memory, addressed by the byte address EADRH/L being erased. EADRL can equal any of 64 locations within the page. A new page starts whenever EADRL is equal to 00H, 40H, 80H, or C0H.
06H ERASE ALL	Results in the erase of entire 4 kBytes of Flash/EE data memory.	Results in the Erase of the entire 56 kBytes of ULOAD Flash/EE program memory.
81H READBYTE	Results in the byte in the Flash/EE data memory, addressed by the byte address EADRH/L, being read into EDATA1. (0 ≤ EADRH/L ≤ 0FFFH).	Not Implemented. Use the MOVC command.
82H WRITEBYTE	Results in the byte in EDATA1 being written into Flash/EE data memory, at the byte address EADRH/L.	Results in the byte in EDATA1 being written into Flash/EE program memory, at the byte address EADRH/L (0 ≤ EADRH/L ≤ DFFFH).
0FH EXULOAD	Leaves the ECON instructions to operate on the Flash/EE data memory.	Enters NORMAL mode directing subsequent ECON instructions to operate on the Flash/EE data memory.
F0H ULOAD	Enters ULOAD mode, directing subsequent ECON instructions to operate on the Flash/EE program memory.	Leaves the ECON instructions to operate on the Flash/EE program memory.

MODE 4: Dual NRZ 16-Bit Σ-Δ DAC
Mode 4 provides a high speed PWM output similar to that of a Σ-Δ DAC. Typically, this mode will be used with the PWM clock equal to 16 MHz.

In this mode P2.6 and P2.7 are updated every PWM clock (62 ns in the case of 16 MHz). Over any 65536 cycles (16 bit PWM) PWM0 (P2.6) is high for PWM0H/L cycles and low for (65536 - PWM0H/L) cycles. Similarly PWM1 (P2.7) is high for PWM1H/L cycles and low for (65536 - PWM1H/L) cycles.

For example, if PWM1H was set to 4010H (slightly above one quarter of FS) then typically P2.7 will be low for three clocks and high for one clock (each clock is approximately 80 ns). Over every 65536 clocks the PWM will compensate for the fact that the output should be slightly above one quarter of full scale by having a high cycle followed by only two low cycles.

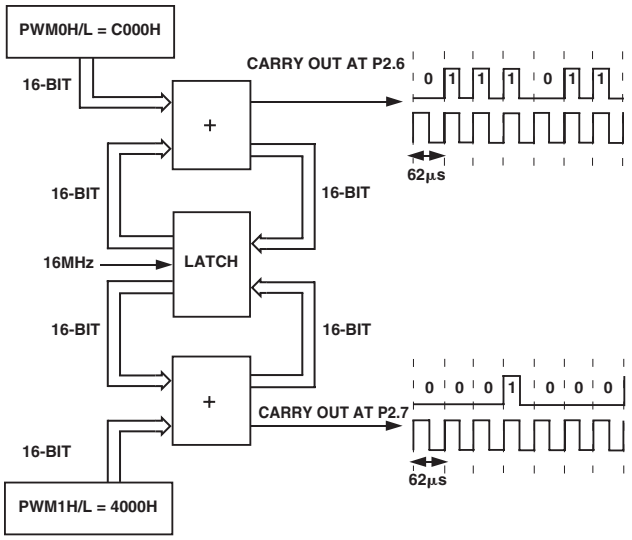


Figure 30. PWM Mode 4

For faster DAC outputs (at lower resolution) write 0s to the LSBs that are not required. If for example only 12-bit performance is required then write 0s to the 4LSBs. This means that a 12-bit accurate Σ-Δ DAC output can occur at 3.906 kHz. Similarly, writing 0s to the 8 LSBs gives an 8-bit accurate Σ-Δ DAC output at 62 kHz.

MODE 5: Dual 8-Bit PWM
In Mode 5, the duty cycle of the PWM outputs and the resolution of the PWM outputs are individually programmable. The maximum resolution of the PWM output is eight bits. The output resolution is set by the PWM1L and PWM1H SFRs for the P2.6 and P2.7 outputs, respectively. PWM0L and PWM0H sets the duty cycles of the PWM outputs at P2.6 and P2.7, respectively. Both PWMs have same clock source and clock divider.

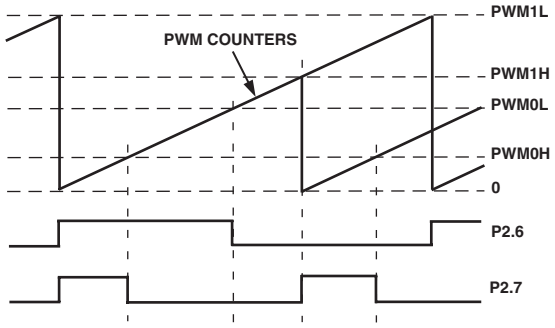


Figure 31. PWM Mode 5

MODE 6: Dual RZ 16-Bit Σ-Δ DAC
Mode 6 provides a high speed PWM output similar to that of a Σ-Δ DAC. Mode 6 operates very similarly to Mode 4. However, the key difference is that Mode 6 provides return to zero (RZ) Σ-Δ DAC output. Mode 4 provides non-return-to-zero Σ-Δ DAC outputs. The RZ mode ensures that any difference in the rise and fall times will not effect the Σ-Δ DAC INL. However, the RZ mode halves the dynamic range of the Σ-Δ DAC outputs from 0–AV_{DD} down to 0–AV_{DD}/2. For best results, this mode should be used with a PWM clock divider of four.

If PWM1H was set to 4010H (slightly above one quarter of FS), then typically P2.7 will be low for three full clocks (3 × 62 ns), high for half a clock (31 ns) and then low again for half a clock (31 ns) before repeating itself. Over every 65536 clocks the PWM will compensate for the fact that the output should be slightly above one quarter of full scale by leaving the output high for two half clocks in four every so often.

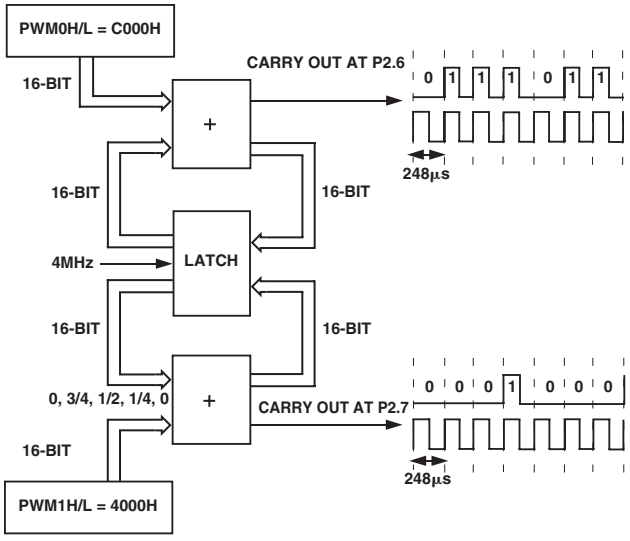


Figure 32. PWM Mode 6

SERIAL PERIPHERAL INTERFACE

The ADuC831 integrates a complete hardware Serial Peripheral Interface (SPI) on-chip. SPI is an industry standard synchronous serial interface that allows eight bits of data to be synchronously transmitted and received simultaneously, i.e., full duplex. It should be noted that the SPI pins are shared with the I^2C interface pins. Therefore, the user can only enable one or the other interface at any given time (see SPE in Table XI below). The SPI Port can be configured for Master or Slave operation, and typically consists of four pins, namely:

MISO (Master In, Slave Out Data I/O Pin)

The MISO (master in slave out) pin is configured as an input line in master mode and an output line in slave mode. The MISO line on the master (data in) should be connected to the MISO line in the slave device (data out). The data is transferred as byte wide (8-bit) serial data, MSB first.

MOSI (Master Out, Slave In Pin)

The MOSI (master out slave in) pin is configured as an output line in master mode and an input line in slave mode. The MOSI line on the master (data out) should be connected to the MOSI line in the slave device (data in). The data is transferred as byte wide (8-bit) serial data, MSB first.

SCLOCK (Serial Clock I/O Pin)

The master serial clock (SCLOCK) is used to synchronize the data being transmitted and received through the MOSI and MISO data lines. A single data bit is transmitted and received in each SCLOCK period. Therefore, a byte is transmitted/received after eight SCLOCK periods. The SCLOCK pin is configured as an output in master mode and as an input in slave mode. In master mode the bit-rate, polarity and phase of the clock are controlled by the CPOL, CPHA, SPR0, and SPR1 bits in the SPICON SFR (see Table XI). In slave mode the SPICON register will have to be configured with the phase and polarity (CPHA and CPOL) of the expected input clock. In both master and slave modes, the data is transmitted on one edge of the SCLOCK signal and sampled on the other. It is important therefore, that the CPHA and CPOL are configured the same for the master and slave devices.

Slave Select Input Pin ($\overline{SS}$)

The Slave Select ($\overline{SS}$) input pin is shared with the ADC5 input. In order to configure this pin as a digital input, the bit must be cleared, e.g., CLR P1.5.

This line is active low. Data is only received or transmitted in slave mode when the $\overline{SS}$ pin is low, allowing the ADuC831 to be used in single master, multislave SPI configurations. If CPHA = 1 then the $\overline{SS}$ input may be permanently pulled low. With CPHA = 0, the $\overline{SS}$ input must be driven low before the first bit in a byte wide transmission or reception and return high again after the last bit in that byte wide transmission or reception. In SPI slave mode, the logic level on the external $\overline{SS}$ pin can be read via the SPR0 bit in the SPICON SFR.

The following SFR registers are used to control the SPI interface.

SPICON	**SPI Control Register**
SFR Address	F8H
Power-On Default Value	OOH
Bit Addressable	Yes

Table XI. SPICON SFR Bit Designations

Bit	Name	Description
7	ISPI	SPI Interrupt Bit. Set by MicroConverter at the end of each SPI transfer. Cleared directly by user code or indirectly by reading the SPIDAT SFR.
6	WCOL	Write Collision Error Bit. Set by MicroConverter if SPIDAT is written to while an SPI transfer is in progress. Cleared by user code.
5	SPE	SPI Interface Enable Bit. Set by user to enable the SPI interface. Cleared by user to enable the I^2C interface.
4	SPIM	SPI Master/Slave Mode Select Bit. Set by user to enable Master Mode operation (SCLOCK is an output). Cleared by user to enable Slave Mode operation (SCLOCK is an input).
3	CPOL	Clock Polarity Select Bit. Set by user if SCLOCK idles high. Cleared by user if SCLOCK idles low.
2	CPHA	Clock Phase Select Bit. Set by user if leading SCLOCK edge is to transmit data. Cleared by user if trailing SCLOCK edge is to transmit data.
1	SPR1	SPI Bit-Rate Select Bits.
0	SPR0	These bits select the SCLOCK rate (bit-rate) in Master Mode as follows: SPR1 SPR0 Selected Bit Rate 0 0 $f_{OSC}/2$ 0 1 $f_{OSC}/4$ 1 0 $f_{OSC}/8$ 1 1 $f_{OSC}/16$ In SPI Slave Mode, i.e., SPIM = 0, the logic level on the external $\overline{SS}$ pin can be read via the SPR0 bit.

The CPOL and CPHA bits should both contain the same values for master and slave devices.

SPIDAT	SPI Data Register
Function	The SPIDAT SFR is written by the user to transmit data over the SPI interface or read by user code to read data just received by the SPI interface.
SFR Address	F7H
Power-On Default Value	00H
Bit Addressable	No

Using the SPI Interface

Depending on the configuration of the bits in the SPICON SFR shown in Table XI, the ADuC831 SPI interface will transmit or receive data in a number of possible modes. Figure 33 shows all possible ADuC831 SPI configurations and the timing relationships and synchronization between the signals involved. Also shown in this figure is the SPI interrupt bit (ISPI) and how it is triggered at the end of each byte-wide communication.

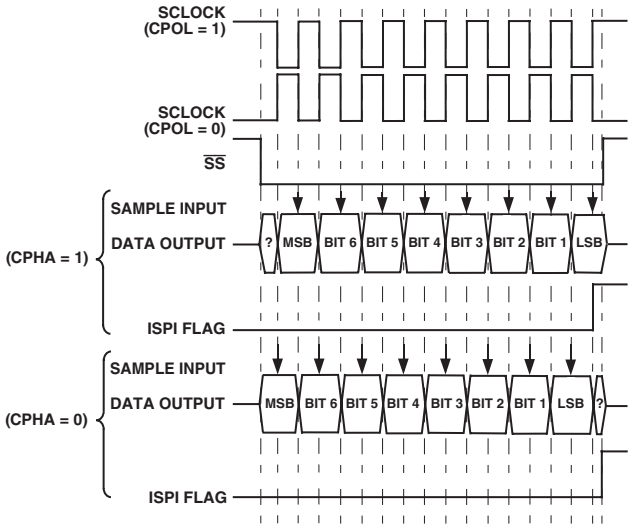


Figure 33. SPI Timing, All Modes

SPI Interface—Master Mode

In master mode, the SCLOCK pin is always an output and generates a burst of eight clocks whenever user code writes to the SPIDAT register. The SCLOCK bit rate is determined by SPR0 and SPR1 in SPICON. It should also be noted that the $\overline{SS}$ pin is not used in master mode. If the ADuC831 needs to assert the $\overline{SS}$ pin on an external slave device, a port digital output pin should be used.

In master mode, a byte transmission or reception is initiated by a write to SPIDAT. Eight clock periods are generated via the SCLOCK pin and the SPIDAT byte being transmitted via MOSI. With each SCLOCK period a data bit is also sampled via MISO. After eight clocks, the transmitted byte will have been completely transmitted and the input byte will be waiting in the input shift register. The ISPI flag will be set automatically and an interrupt will occur if enabled. The value in the shift register will be latched into SPIDAT.

SPI Interface—Slave Mode

In slave mode the SCLOCK is an input. The $\overline{SS}$ pin must also be driven low externally during the byte communication.

Transmission is also initiated by a write to SPIDAT. In slave mode, a data bit is transmitted via MISO and a data bit is received via MOSI through each input SCLOCK period. After eight clocks, the transmitted byte will have been completely transmitted and the input byte will be waiting in the input shift register. The ISPI flag will be set automatically and an interrupt will occur if enabled. The value in the shift register will be latched into SPIDAT only when the transmission/reception of a byte has been completed. The end of transmission occurs after the eighth clock has been received if CPHA = 1, or when $\overline{SS}$ returns high if CPHA = 0.

WATCHDOG TIMER

The purpose of the watchdog timer is to generate a device reset or interrupt within a reasonable amount of time if the ADuC831 enters an erroneous state, possibly due to a programming error or electrical noise. The watchdog function can be disabled by clearing the WDE (Watchdog Enable) bit in the Watchdog Control (WDCON) SFR. When enabled, the watchdog circuit will generate a system reset or interrupt (WDS) if the user program fails to set the watchdog (WDE) bit within a predetermined amount of time (see PRE3–0 bits in WDCON). The watchdog timer itself is a 16-bit counter that is clocked at 32 kHz by the internal R/C oscillator. The watchdog time out interval can be adjusted via the PRE3–0 bits in WDCON. Full control and status of the watchdog timer function can be controlled via the watchdog timer control SFR (WDCON). The WDCON SFR can only be written by user software if the double write sequence described in WDWR below is initiated on every write access to the WDCON SFR.

WDCON	**Watchdog Timer Control Register**
SFR Address	C0H
Power-On Default Value	10H
Bit Addressable	Yes

Table XV. WDCON SFR Bit Designations

Bit	Name	Description
7	PRE3	Watchdog Timer Prescale Bits.
6	PRE2	The Watchdog timeout period is given by the equation: $t_{WD} = (2^{PRE} \times (2^9/f_{R/C\ OSC}))$
5	PRE1	$(0 \le PRE \le 7; f_{R/C\ OSC} = 32\ kHz \pm 10\%$ at 25°C)
4	PRE0	PRE3 PRE2 PRE1 PRE0 Timeout Period (ms) Action 0 0 0 0 15.6 Reset or Interrupt 0 0 0 1 31.2 Reset or Interrupt 0 0 1 0 62.5 Reset or Interrupt 0 0 1 1 125 Reset or Interrupt 0 1 0 0 250 Reset or Interrupt 0 1 0 1 500 Reset or Interrupt 0 1 1 0 1000 Reset or Interrupt 0 1 1 1 2000 Reset or Interrupt 1 0 0 0 0.0 Immediate Reset PRE3–0 > 1000 Reserved
3	WDIR	Watchdog Interrupt Response Enable Bit. If this bit is set by the user, the watchdog will generate an interrupt response instead of a system reset when the watchdog timeout period has expired. This interrupt is not disabled by the CLR EA instruction and it is also a fixed, high-priority interrupt. If the watchdog is not being used to monitor the system, it can alternatively be used as a timer. The prescaler is used to set the timeout period in which an interrupt will be generated.
2	WDS	Watchdog Status Bit. Set by the Watchdog Controller to indicate that a watchdog timeout has occurred. Cleared by writing a "0" or by an external hardware reset. It is not cleared by a watchdog reset.
1	WDE	Watchdog Enable Bit. Set by the user to enable the watchdog and clear its counters. If this bit is not set by the user within the watch dog timeout period, the watchdog will generate a reset or interrupt, depending on WDIR. Cleared under the following conditions, user writes "0," Watchdog Reset (WDIR = "0"); Hardware Reset; PSM Interrupt.

0	WDWR	Watchdog Write Enable Bit.
		To write data into the WDCON SFR involves a double instruction sequence. The WDWR bit must be set and the very next instruction must be a write instruction to the WDCON SFR. For example:
		`CLR EA ;disable interrupts while writing`
		`;to WDT`
		`SETB WDWR ;allow write to WDCON`
		`MOV WDCON, #72H ;enable WDT for 2.0s timeout`
		`SETB EA ;enable interrupts again (if rqd)`

INTVAL — **User Time Interval Select Register**

Function	User code writes the required time interval to this register. When the 8-bit interval counter is equal to the time interval value loaded in the INTVAL SFR, the TII bit (TIMECON.2) is set and generates an interrupt if enabled.
SFR Address	A6H
Power-On Default Value	00H
Bit Addressable	No
Valid Value	0 to 255 decimal

HTHSEC — **Hundredths Seconds Time Register**

Function	This register is incremented in 1/128 second intervals once TCEN in TIMECON is active. The HTHSEC SFR counts from 0 to 127 before rolling over to increment the SEC time register.
SFR Address	A2H
Power-On Default Value	00H
Bit Addressable	No
Valid Value	0 to 127 decimal

SEC — **Seconds Time Register**

Function	This register is incremented in 1-second intervals once TCEN in TIMECON is active. The SEC SFR counts from 0 to 59 before rolling over to increment the MIN time register.
SFR Address	A3H
Power-On Default Value	00H
Bit Addressable	No
Valid Value	0 to 59 decimal

MIN — **Minutes Time Register**

Function	This register is incremented in 1-minute intervals once TCEN in TIMECON is active. The MIN counts from 0 to 59 before rolling over to increment the HOUR time register.
SFR Address	A4H
Power-On Default Value	00H
Bit Addressable	No
Valid Value	0 to 59 decimal

HOUR — **Hours Time Register**

Function	This register is incremented in 1-hour intervals once TCEN in TIMECON is active. The HOUR SFR counts from 0 to 23 before rolling over to 0.
SFR Address	A5H
Power-On Default Value	00H
Bit Addressable	No
Valid Value	0 to 23 decimal

TCON **Timer/Counter 0 and 1 Control Register**
SFR Address 88H
Power-On Default Value 00H
Bit Addressable Yes

Table XX. TCON SFR Bit Designations

Bit	Name	Description
7	TF1	Timer 1 Overflow Flag. Set by hardware on a Timer/Counter 1 overflow. Cleared by hardware when the Program Counter (PC) vectors to the interrupt service routine.
6	TR1	Timer 1 Run Control Bit. Set by user to turn on Timer/Counter 1. Cleared by user to turn off Timer/Counter 1.
5	TF0	Timer 0 Overflow Flag. Set by hardware on a Timer/Counter 0 overflow. Cleared by hardware when the PC vectors to the interrupt service routine.
4	TR0	Timer 0 Run Control Bit. Set by user to turn on Timer/Counter 0. Cleared by user to turn off Timer/Counter 0.
3	IE1*	External Interrupt 1 ($\overline{INT1}$) Flag. Set by hardware by a falling edge or zero level being applied to external interrupt Pin $\overline{INT1}$, depending on bit IT1 state. Cleared by hardware when the PC vectors to the interrupt service routine only if the interrupt was transition-activated. If level-activated, the external requesting source controls the request flag, rather than the on-chip hardware.
2	IT1*	External Interrupt 1 (IE1) Trigger Type. Set by software to specify edge-sensitive detection (i.e., 1-to-0 transition). Cleared by software to specify level-sensitive detection (i.e., zero level).
1	IE0*	External Interrupt 0 ($\overline{INT0}$) Flag. Set by hardware by a falling edge or zero level being applied to external interrupt Pin $\overline{INT0}$, depending on bit IT0 state. Cleared by hardware when the PC vectors to the interrupt service routine only if the interrupt was transition-activated. If level-activated, the external requesting source controls the request flag, rather than the on-chip hardware.
0	IT0*	External Interrupt 0 (IE0) Trigger Type. Set by software to specify edge-sensitive detection (i.e., 1-to-0 transition). Cleared by software to specify level-sensitive detection (i.e., zero level).

*These bits are not used in the control of timer/counter 0 and 1, but are used instead in the control and monitoring of the external $\overline{INT0}$ and $\overline{INT1}$ interrupt pins.

Timer/Counter 0 and 1 Data Registers
Each timer consists of two 8-bit registers. These can be used as independent registers or combined to be a single 16-bit register, depending on the timer mode configuration.

TH0 and TL0
Timer 0 high byte and low byte.
SFR Address = 8CH, 8AH respectively.

TH1 and TL1
Timer 1 high byte and low byte.
SFR Address = 8DH, 8BH respectively.

TIMER/COUNTER 0 AND 1 OPERATING MODES

The following paragraphs describe the operating modes for Timer/Counters 0 and 1. Unless otherwise noted, it should be assumed that these modes of operation are the same for Timer 0 as for Timer 1.

Mode 0 (13-Bit Timer/Counter)

Mode 0 configures an 8-bit Timer/Counter with a divide-by-32 prescaler. Figure 45 shows mode 0 operation.

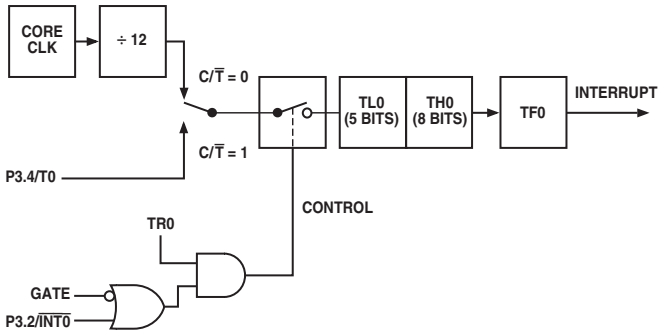


Figure 45. Timer/Counter 0, Mode 0

In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer overflow flag TF0. The overflow flag, TF0, can then be used to request an interrupt. The counted input is enabled to the timer when TR0 = 1 and either Gate = 0 or $\overline{INT0}$ = 1. Setting Gate = 1 allows the timer to be controlled by external input $\overline{INT0}$, to facilitate pulsewidth measurements. TR0 is a control bit in the special function register TCON; Gate is in TMOD. The 13-bit register consists of all eight bits of TH0 and the lower five bits of TL0. The upper three bits of TL0 are indeterminate and should be ignored. Setting the run flag (TR0) does not clear the registers.

Mode 1 (16-Bit Timer/Counter)

Mode 1 is the same as Mode 0, except that the timer register is running with all 16 bits. Mode 1 is shown in Figure 46.

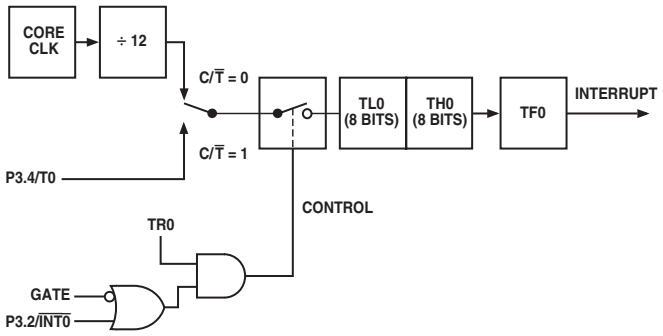


Figure 46. Timer/Counter 0, Mode 1

Mode 2 (8-Bit Timer/Counter with Autoreload)

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reload, as shown in Figure 47. Overflow from TL0 not only sets TF0, but also reloads TL0 with the contents of TH0, which is preset by software. The reload leaves TH0 unchanged.

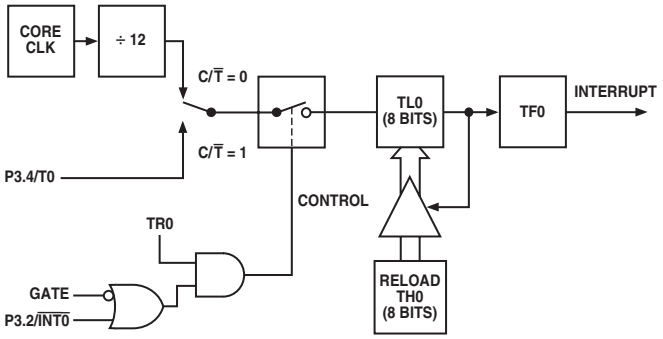


Figure 47. Timer/Counter 0, Mode 2

Mode 3 (Two 8-Bit Timer/Counters)

Mode 3 has different effects on Timer 0 and Timer 1. Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. This configuration is shown in Figure 50. TL0 uses the Timer 0 control bits: C/T, Gate, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the Timer 1 interrupt. Mode 3 is provided for applications requiring an extra 8-bit timer or counter.

When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of, and into, its own Mode 3, or can still be used by the serial interface as a *baud rate generator*. In fact, it can be used in any application not requiring an interrupt from Timer 1 itself.

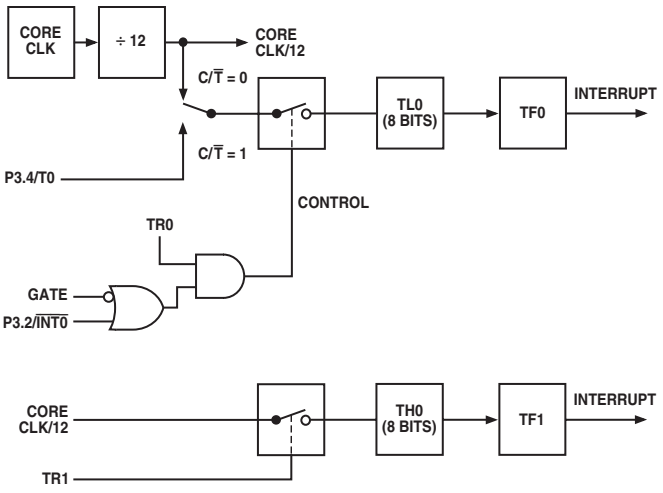


Figure 48. Timer/Counter 0, Mode 3

Timer/Counter Operation Modes

The following paragraphs describe the operating modes for Timer/Counter 2. The operating modes are selected by bits in the T2CON SFR as shown in Table XXII.

Table XXII. T2CON Operating Modes

RCLK (or) TCLK	CAP2	TR2	Mode
0	0	1	16-Bit Autoreload
0	1	1	16-Bit Capture
1	X	1	Baud Rate
X	X	0	OFF

16-Bit Autoreload Mode

In Autoreload mode, there are two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2, but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still performs the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2. The Autoreload mode is illustrated in Figure 49.

16-Bit Capture Mode

In the Capture mode, there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which, upon overflowing, sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still performs the above, but a l-to-0 transition on external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt. The Capture mode is illustrated in Figure 50.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1.

In either case, if Timer 2 is being used to generate the baud rate, the TF2 interrupt flag will not occur. Therefore, Timer 2 interrupts will not occur so they do not have to be disabled. In this mode the EXF2 flag, however, can still cause interrupts and this can be used as a third external interrupt.

Baud rate generation will be described as part of the UART serial port operation in the following pages.

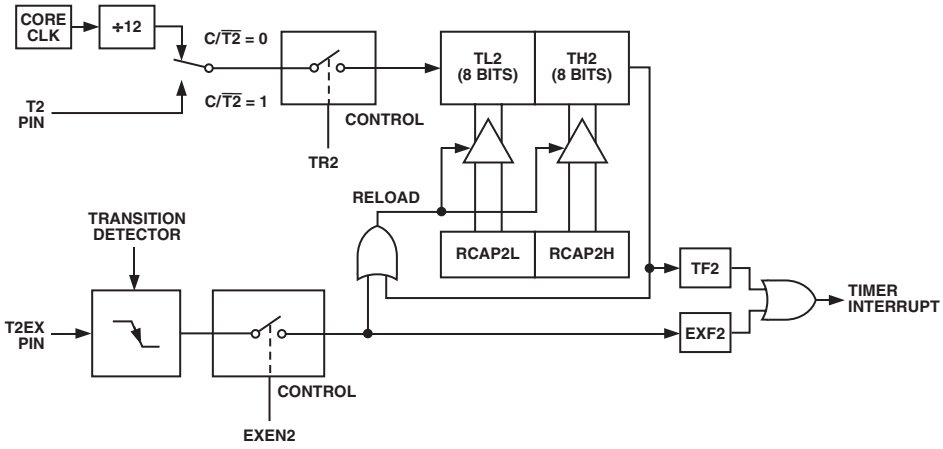


Figure 49. Timer/Counter 2, 16-Bit Autoreload Mode

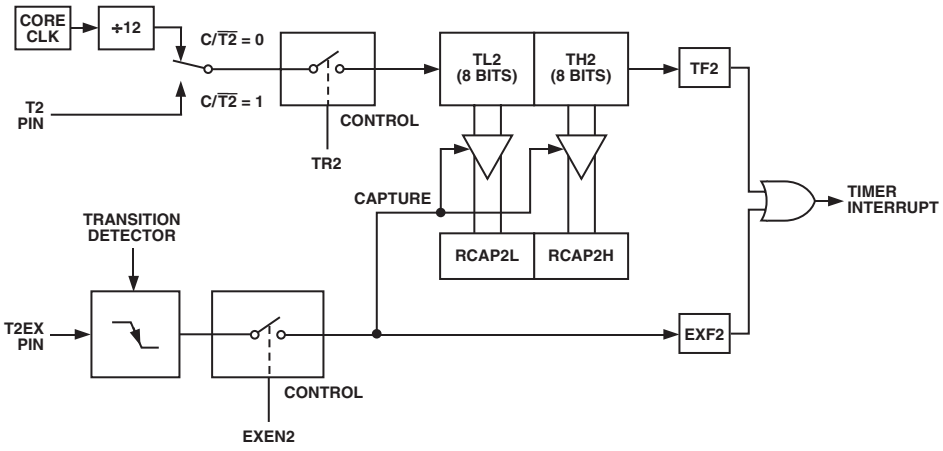


Figure 50. Timer/Counter 2, 16-Bit Capture Mode

INTERRUPT SYSTEM

The ADuC831 provides a total of nine interrupt sources with two priority levels. The control and configuration of the interrupt system is carried out through three Interrupt-related SFRs.

IE	Interrupt Enable Register
IP	Interrupt Priority Register
IEIP2	Secondary Interrupt Enable Register

IE **Interrupt Enable Register**

SFR Address	A8H
Power-On Default Value	00H
Bit Addressable	Yes

Table XXVIII. IE SFR Bit Designations

Bit	Name	Description
7	EA	Written by User to Enable "1" or Disable "0" All Interrupt Sources
6	EADC	Written by User to Enable "1" or Disable "0" ADC Interrupt
5	ET2	Written by User to Enable "1" or Disable "0" Timer 2 Interrupt
4	ES	Written by User to Enable "1" or Disable "0" UART Serial Port Interrupt
3	ET1	Written by User to Enable "1" or Disable "0" Timer 1 Interrupt
2	EX1	Written by User to Enable "1" or Disable "0" External Interrupt 1
1	ET0	Written by User to Enable "1" or Disable "0" Timer 0 Interrupt
0	EX0	Written by User to Enable "1" or Disable "0" External Interrupt 0

IP **Interrupt Priority Register**

SFR Address	B8H
Power-On Default Value	00H
Bit Addressable	Yes

Table XXIX. IP SFR Bit Designations

Bit	Name	Description
7	----	Reserved for Future Use
6	PADC	Written by User to Select ADC Interrupt Priority ("1" = High; "0" = Low)
5	PT2	Written by User to Select Timer 2 Interrupt Priority ("1" = High; "0" = Low)
4	PS	Written by User to Select UART Serial Port Interrupt Priority ("1" = High; "0" = Low)
3	PT1	Written by User to Select Timer 1 Interrupt Priority ("1" = High; "0" = Low)
2	PX1	Written by User to Select External Interrupt 1 Priority ("1" = High; "0" = Low)
1	PT0	Written by User to Select Timer 0 Interrupt Priority ("1" = High; "0" = Low)
0	PX0	Written by User to Select External Interrupt 0 Priority ("1" = High; "0" = Low)

IEIP2 **Secondary Interrupt Enable Register**

SFR Address	A9H
Power-On Default Value	A0H
Bit Addressable	No

Table XXX. IEIP2 SFR Bit Designations

Bit	Name	Description
7	----	Reserved for Future Use
6	PTI	Priority for Time Interval Interrupt
5	PPSM	Priority for Power Supply Monitor Interrupt
4	PSI	Priority for SPI/I^2C Interrupt
3	----	This Bit Must Contain Zero
2	ETI	Written by User to Enable "1" or Disable "0" Time Interval Counter Interrupt
1	EPSMI	Written by User to Enable "1" or Disable "0" Power Supply Monitor Interrupt
0	ESI	Written by User to Enable "1" or Disable "0" SPI/I^2C Serial Port Interrupt

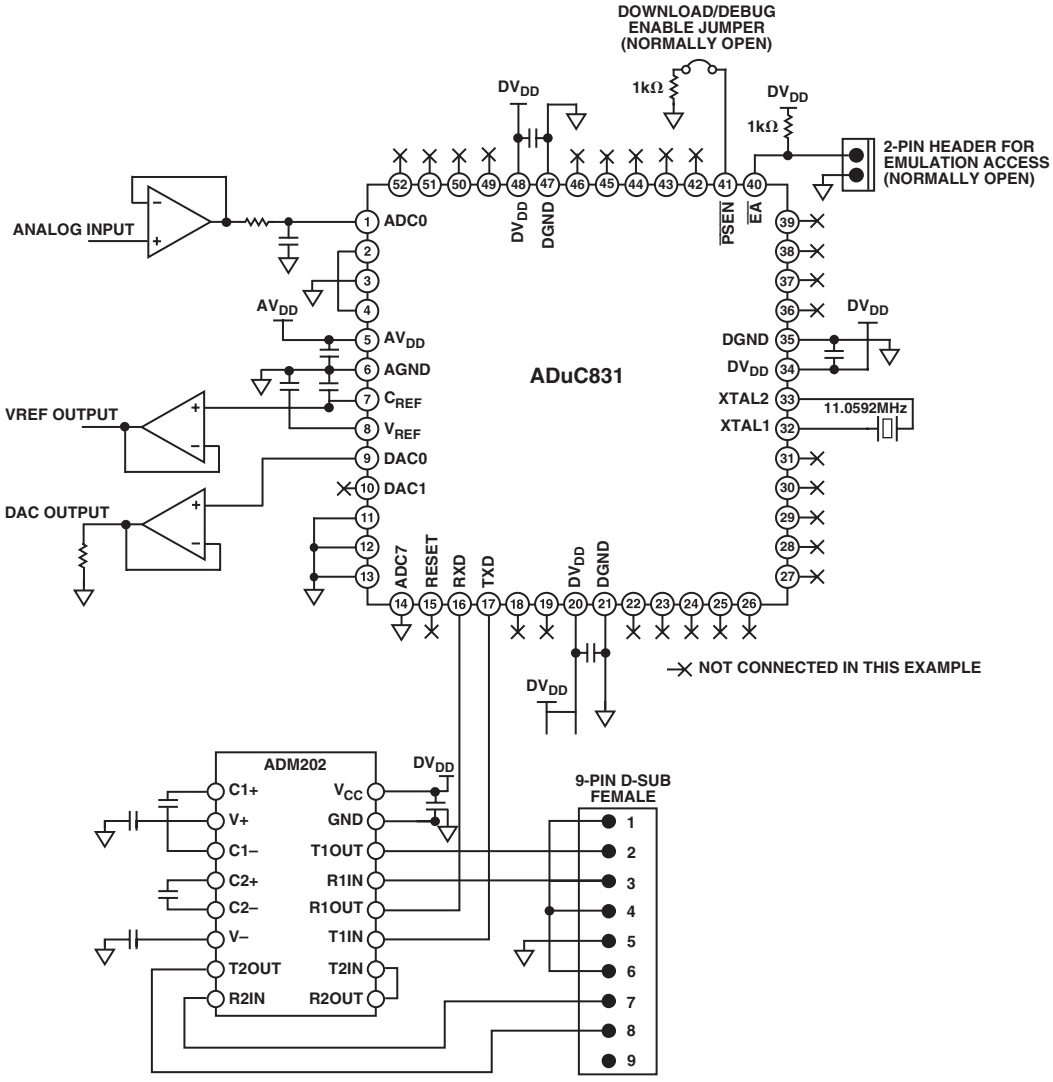


Figure 64. Example ADuC831 System (PQFP Package)

Note that $\overline{\text{PSEN}}$ is normally an output (as described in the External Memory Interface section) and is sampled as an input only on the falling edge of RESET (i.e., at power-up or upon an external manual reset). Note also that if any external circuitry unintentionally pulls $\overline{\text{PSEN}}$ low during power-up or reset events, it could cause the chip to enter download mode and therefore fail to begin user code execution as it should. To prevent this, ensure that no external signals are capable of pulling the $\overline{\text{PSEN}}$ pin low, except for the external $\overline{\text{PSEN}}$ jumper itself.

Embedded Serial Port Debugger

From a hardware perspective, entry into serial port debug mode is identical to the serial download entry sequence described above. In fact, both serial download and serial port debug modes can be thought of as essentially one mode of operation used in two different ways.

Note that the serial port debugger is fully contained on the ADuC831 device, (unlike ROM monitor type debuggers) and therefore no external memory is needed to enable in-system debug sessions.

Single-Pin Emulation Mode

Also built into the ADuC831 is a dedicated controller for single-pin in-circuit emulation (ICE) using standard production ADuC831 devices. In this mode, emulation access is gained by connection to a single pin, the $\overline{\text{EA}}$ pin. Normally, this pin is hardwired either high or low to select execution from internal or external program memory space, as described earlier. To enable single-pin emulation mode, however, users will need to pull the $\overline{\text{EA}}$ pin high through a 1 kΩ resistor as shown in Figure 64. The emulator will then connect to the 2-pin header also shown in Figure 64. To be compatible with the standard connector that comes with the single-pin emulator available from Accutron Limited (www.accutron.com), use a 2-pin 0.1-inch pitch "Friction Lock" header from Molex (www.molex.com) such as their part number 22-27-2021. Be sure to observe the polarity of this header. As represented in Figure 64, when the Friction Lock tab is at the right, the ground pin should be the lower of the two pins (when viewed from the top).

Typical System Configuration

A typical ADuC831 configuration is shown in Figure 64. It summarizes some of the hardware considerations discussed in the previous paragraphs.

Parameter		12 MHz Min	12 MHz Max	Variable Clock Min	Variable Clock Max	Unit	Figure
EXTERNAL DATA MEMORY WRITE CYCLE							
t_{WLWH}	$\overline{WR}$ Pulsewidth	400		$6t_{CK} - 100$		ns	72
t_{AVLL}	Address Valid after ALE Low	43		$t_{CK} - 40$		ns	72
t_{LLAX}	Address Hold after ALE Low	48		$t_{CK} - 35$		ns	72
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CK} - 50$	$3t_{CK} + 50$	ns	72
t_{AVWL}	Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CK} - 130$		ns	72
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	33		$t_{CK} - 50$		ns	72
t_{QVWH}	Data Setup before $\overline{WR}$	433		$7t_{CK} - 150$		ns	72
t_{WHQX}	Data and Address Hold after $\overline{WR}$	33		$t_{CK} - 50$		ns	72
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$t_{CK} - 40$	$6t_{CK} - 100$	ns	72

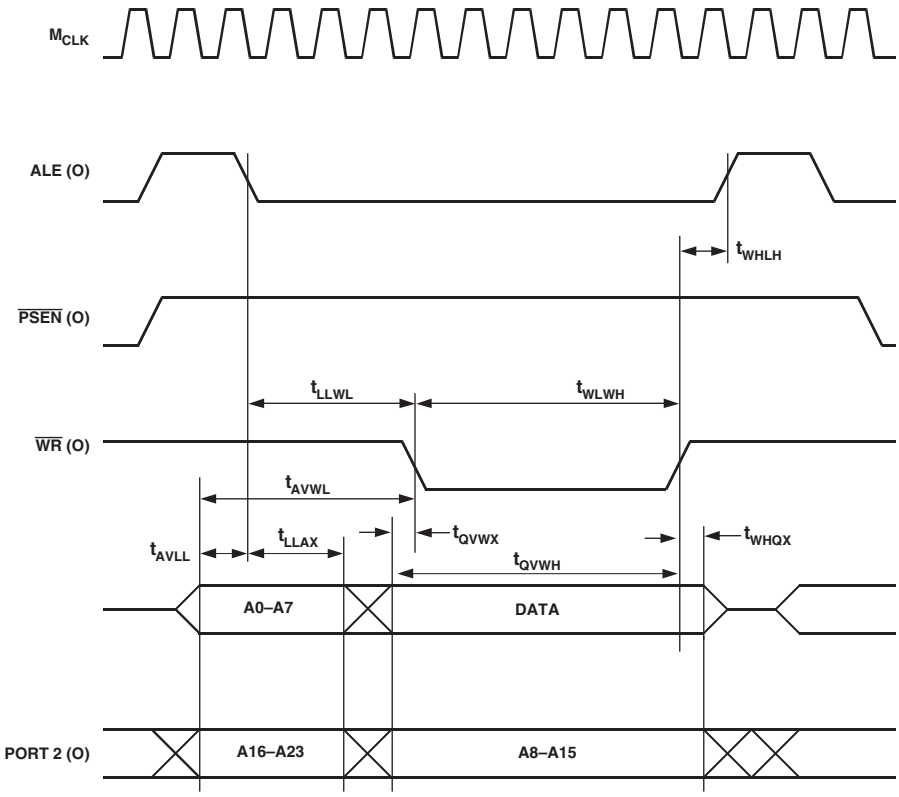


Figure 72. External Data Memory Write Cycle

Parameter		Min	Max	Unit	Figure
I²C COMPATIBLE INTERFACE TIMING					
t_L	SCLOCK Low Pulsewidth	4.7		μs	74
t_H	SCLOCK High Pulsewidth	4.0		μs	74
t_{SHD}	Start Condition Hold Time	0.6		μs	74
t_{DSU}	Data Setup Time	100		μs	74
t_{DHD}	Data Hold Time		0.9	μs	74
t_{RSU}	Setup Time for Repeated Start	0.6		μs	74
t_{PSU}	Stop Condition Setup Time	0.6		μs	74
t_{BUF}	Bus Free Time Between a STOP Condition and a START Condition	1.3		μs	74
t_R	Rise Time of Both SCLOCK and SDATA		300	ns	74
t_F	Fall Time of Both SCLOCK and SDATA		300	ns	74
t_{SUP}*	Pulsewidth of Spike Suppressed		50	ns	74

*Input filtering on both the SCLOCK and SDATA inputs suppresses noise spikes less than 50 ns.

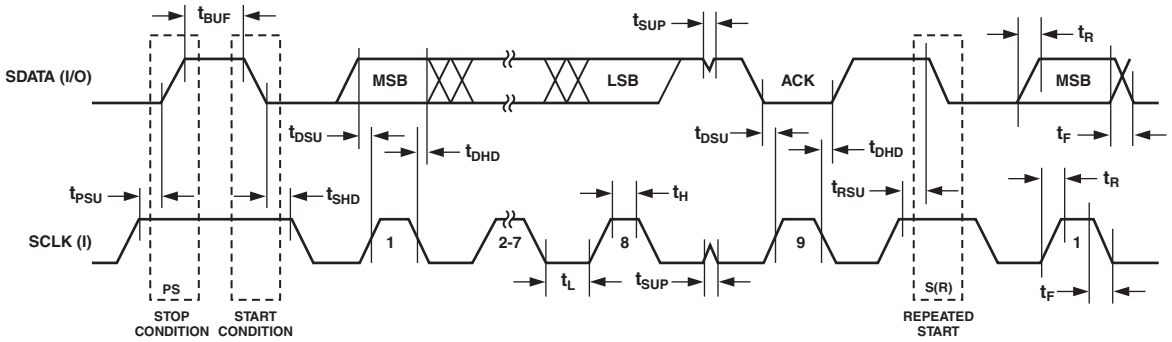


Figure 74. I²C Compatible Interface Timing

Parameter		Min	Typ	Max	Unit	Figure
SPI MASTER MODE TIMING (CPHA = 1)						
t_{SL}	SCLOCK Low Pulsewidth		330		ns	75
t_{SH}	SCLOCK High Pulsewidth		330		ns	75
t_{DAV}	Data Output Valid after SCLOCK Edge			50	ns	75
t_{DSU}	Data Input Setup Time before SCLOCK Edge	100			ns	75
t_{DHD}	Data Input Hold Time after SCLOCK Edge	100			ns	75
t_{DF}	Data Output Fall Time		10	25	ns	75
t_{DR}	Data Output Rise Time		10	25	ns	75
t_{SR}	SCLOCK Rise Time		10	25	ns	75
t_{SF}	SCLOCK Fall Time		10	25	ns	75

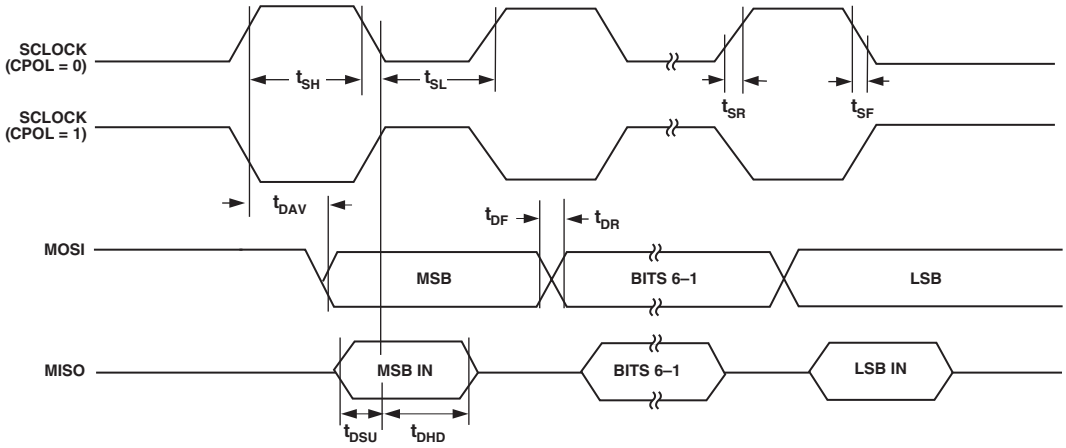


Figure 75. SPI Master Mode Timing (CPHA = 1)