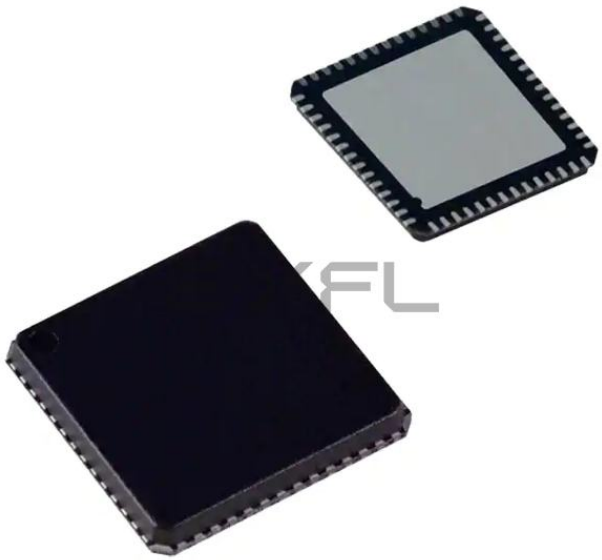

Welcome to E-XFL.COM

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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	56-VFQFN Exposed Pad, CSP
Supplier Device Package	56-LFCSP-VQ (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/aduc831bcpz-reel

SPECIFICATIONS (continued)

Parameter	V_{DD} = 5 V	V_{DD} = 3 V	Unit	Test Conditions/Comments
DAC CHANNEL SPECIFICATIONS[12, 13] Internal Buffer Disabled				
DC ACCURACY[10]				
Resolution	12	12	Bits	
Relative Accuracy	±3	±3	LSB typ	
Differential Nonlinearity[11]	–1	–1	LSB max	Guaranteed 12-bit Monotonic
	±1/2	±1/2	LSB typ	
Offset Error	±5	±5	mV max	V_{REF} Range
Gain Error	–0.3	–0.3	% typ	V_{REF} Range
Gain Error Mismatch[4]	0.5	0.5	% max	% of Full-Scale on DAC1
ANALOG OUTPUTS				
Voltage Range_0	0 to V_{REF}	0 to V_{REF}	V typ	DAC V_{REF} = 2.5 V
REFERENCE INPUT/OUTPUT				
REFERENCE OUTPUT[14]				
Output Voltage (V_{REF})	2.5	2.5	V	
Accuracy	±2.5	±2.5	% max	Of V_{REF} Measured at the C_{REF} Pin
Power Supply Rejection	47	57	dB typ	
Reference Temperature Coefficient	±100	±100	ppm/°C typ	
Internal V_{REF} Power-On Time	80	80	ms typ	
EXTERNAL REFERENCE INPUT[15]				
Voltage Range (V_{REF})[4]	0.1	0.1	V min	V_{REF} and C_{REF} Pins Shorted
	V_{DD}	V_{DD}	V max	
Input Impedance	20	20	kΩ typ	
Input Leakage	1	1	µA max	Internal Band Gap Deselected via ADCCON1.6
POWER SUPPLY MONITOR (PSM)				
DV_{DD} Trip Point Selection Range	2.63		V min	Four Trip Points Selectable in
	4.37		V max	This Range Programmed via TPD1–0 in PSMCON
DV_{DD} Power Supply Trip Point Accuracy	±3.5		% max	
WATCHDOG TIMER (WDT)[4]				
Time-out Period	0	0	ms min	Nine Time-out Periods
	2000	2000	ms max	Selectable in This Range
FLASH/EE MEMORY RELIABILITY CHARACTERISTICS[16]				
Endurance[17]	100,000	100,000	Cycles min	
Data Retention[18]	100	100	Years min	
DIGITAL INPUTS				
Input High Voltage (V_{INH})[4]	2.4	2	V min	
Input Low Voltage (V_{INL})[4]	0.8	0.4	V max	
Input Leakage Current (Port 0, $\overline{EA}$)	±10	±10	µA max	V_{IN} = 0 V or V_{DD}
	±1	±1	µA typ	V_{IN} = 0 V or V_{DD}
Logic 1 Input Current				
(All Digital Inputs)	±10	±10	µA max	V_{IN} = V_{DD}
	±1	±1	µA typ	V_{IN} = V_{DD}
Logic 0 Input Current (Port 1, 2, 3)	–75	–25	µA max	
	–40	–15	µA typ	V_{IL} = 450 mV
Logic 1-0 Transition Current (Port 2, 3)	–660	–250	µA max	V_{IL} = 2 V
	–400	–140	µA typ	V_{IL} = 2 V

Parameter	V_{DD} = 5 V	V_{DD} = 3 V	Unit	Test Conditions/Comments
SCLOCK and RESET Only[4] (Schmitt-Triggered Inputs)				
V_{T+}	1.3	0.95	V min	
	3.0	2.5	V max	
V_{T-}	0.8	0.4	V min	
	1.4	1.1	V max	
$V_{T+} - V_{T-}$	0.3	0.3	V min	
	0.85	0.85	V max	
CRYSTAL OSCILLATOR				
Logic Inputs, XTAL1 Only				
V_{INL}, Input Low Voltage	0.8	0.4	V typ	
V_{INH}, Input High Voltage	3.5	2.5	V typ	
XTAL1 Input Capacitance	18	18	pF typ	
XTAL2 Output Capacitance	18	18	pF typ	
MCU CLOCK RATE	16	16	MHz max	
DIGITAL OUTPUTS				
Output High Voltage (V_{OH})	2.4		V min	V_{DD} = 4.5 V to 5.5 V
	4.0		V typ	I_{SOURCE} = 80 μA
		2.4	V min	V_{DD} = 2.7 V to 3.3 V
		2.6	V typ	I_{SOURCE} = 20 μA
Output Low Voltage (V_{OL})				
ALE, Ports 0 and 2	0.4	0.4	V max	I_{SINK} = 1.6 mA
	0.2	0.2	V typ	I_{SINK} = 1.6 mA
Port 3	0.4	0.4	V max	I_{SINK} = 4 mA
SCLOCK/SDATA	0.4	0.4	V max	I_{SINK} = 8 mA, I^2C Enabled
Floating State Leakage Current[4]	±10	±10	μA max	
	±1	±1	μA typ	
Floating State Output Capacitance	10	10	pF typ	
START UP TIME				MCLKIN = 16 MHz
At Power-On	500	500	ms typ	
From Idle Mode	100	100	μs typ	
From Power-Down Mode				
Wakeup with $\overline{INT0}$ Interrupt	150	400	μs typ	
Wakeup with SPI/I^2C Interrupt	150	400	μs typ	
Wakeup with External RESET	150	400	μs typ	
After External RESET in Normal Mode	30	30	ms typ	
After WDT Reset in Normal Mode	3	3	ms typ	Controlled via WDCON SFR

MEMORY ORGANIZATION

The ADuC831 contains four different memory blocks:

- 62 kBytes of On-Chip Flash/EE Program Memory
- 4 kBytes of On-Chip Flash/EE Data Memory
- 256 Bytes of General-Purpose RAM
- 2 kBytes of Internal XRAM

Flash/EE Program Memory

The ADuC831 provides 62 kBytes of Flash/EE program memory to run user code. The user can choose to run code from this internal memory or run code from an external program memory.

If the user applies power or resets the device while the $\overline{EA}$ pin is pulled low, the part will execute code from the external program space, otherwise the part defaults to code execution from its internal 62 kBytes of Flash/EE program memory. Unlike the ADuC812, where code execution can overflow from the internal code space to external code space once the PC becomes greater than 1FFFH, the ADuC831 does not support the rollover from F7FFH in internal code space to F800H in external code space. Instead the 2048 bytes between F800H and FFFFH will appear as NOP instructions to user code.

This internal code space can be downloaded via the UART serial port while the device is in-circuit. 56 kBytes of the program memory can be reprogrammed during runtime thus the code space can be upgraded in the field using a user defined protocol or it can be used as a data memory. This will be discussed in more detail in the Flash/EE Memory section.

Flash/EE Data Memory

4 kBytes of Flash/EE Data Memory are available to the user and can be accessed indirectly via a group of control registers mapped into the Special Function Register (SFR) area. Access to the Flash/EE data memory is discussed in detail later as part of the Flash/EE Memory section.

General-Purpose RAM

The general-purpose RAM is divided into two separate memories, namely the upper and the lower 128 bytes of RAM. The lower 128 bytes of RAM can be accessed through direct or indirect addressing. The upper 128 bytes of RAM can only be accessed through indirect addressing as it shares the same address space as the SFR space, which can only be accessed through direct addressing.

The lower 128 bytes of internal data memory are mapped as shown in Figure 2. The lowest 32 bytes are grouped into four banks of eight registers addressed as R0 through R7. The next 16 bytes (128 bits), locations 20H through 2FH above the register banks, form a block of directly addressable bit locations at bit addresses 00H through 7FH. The stack can be located anywhere in the internal memory address space, and the stack depth can be expanded up to 2048 bytes.

Reset initializes the stack pointer to location 07H and increments it once before loading the stack to start from locations 08H which is also the first register (R0) of register bank 1. Thus, if one is going to use more than one register bank, the stack pointer should be initialized to an area of RAM not used for data storage.

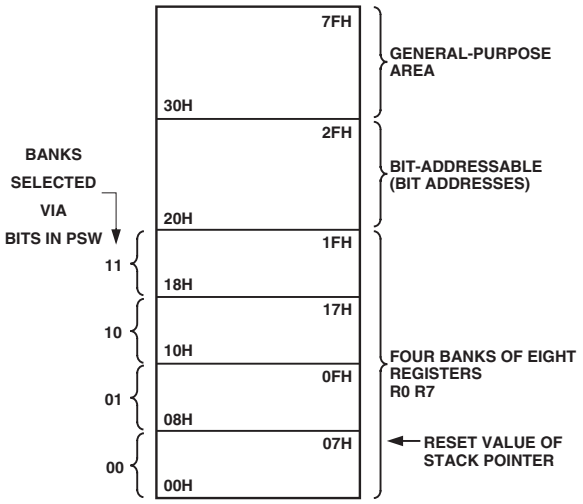

Figure 2. Lower 128 Bytes of Internal Data Memory

The ADuC831 contains 2048 bytes of internal XRAM, 1792 bytes of which can be configured to be used as an extended 11-bit stack pointer.

By default, the stack will operate exactly like an 8052 in that it will roll over from FFH to 00H in the general-purpose RAM. On the ADuC831 however, it is possible (by setting CFG831.7) to enable the 11-bit extended stack pointer. In this case, the stack will roll over from FFH in RAM to 0100H in XRAM.

The 11-bit stack pointer is visible in the SP and SPH SFRs. The SP SFR is located at 81H as with a standard 8052. The SPH SFR is located at B7H. The 3 LSBs of this SFR contain the three extra bits necessary to extend the 8-bit stack pointer into an 11-bit stack pointer.

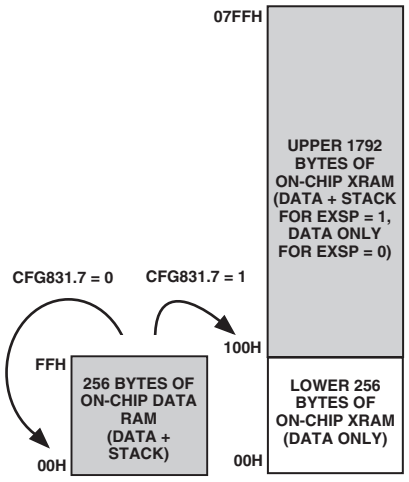

Figure 3. Extended Stack Pointer Operation

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.

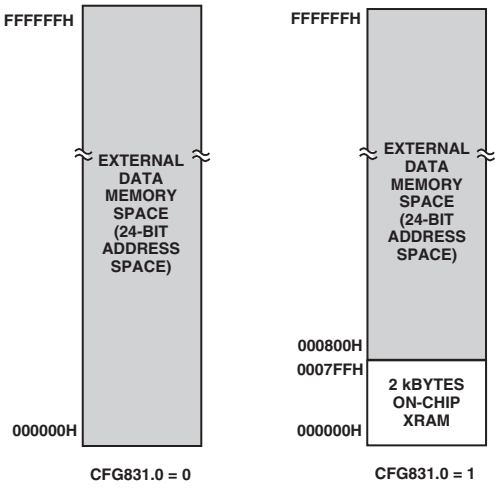

Figure 4. Internal and External XRAM

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.

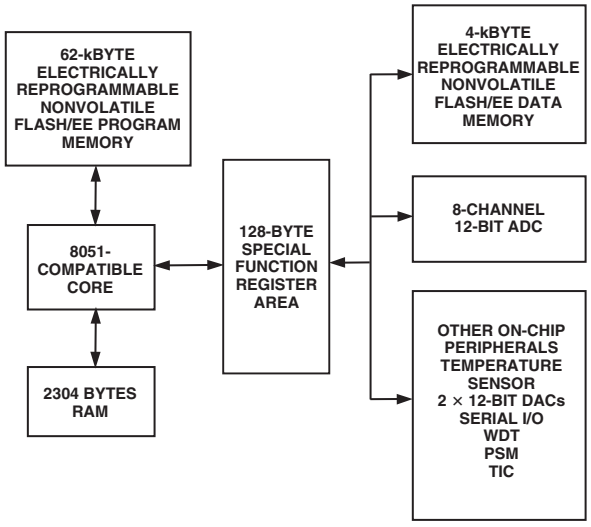

Figure 5. Programming Model

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.

ADC CIRCUIT INFORMATION

General Overview

The ADC conversion block incorporates a fast, 8-channel, 12-bit, single supply ADC. This block provides the user with multichannel mux, track/hold, on-chip reference, calibration features, and ADC. All components in this block are easily configured via a 3-register SFR interface.

The ADC consists of a conventional successive-approximation converter based around a capacitor DAC. The converter accepts an analog input range of 0 to V_{REF}. A high precision, low drift, and factory calibrated 2.5 V reference is provided on-chip. An external reference can be connected as described later. This external reference can be in the range of 1 V to AV_{DD}.

Single step or continuous conversion modes can be initiated in software or alternatively by applying a convert signal to an external pin. Timer 2 can also be configured to generate a repetitive trigger for ADC conversions. The ADC may be configured to operate in a DMA Mode whereby the ADC block continuously converts and captures samples to an external RAM space without any interaction from the MCU core. This automatic capture facility can extend through a 16 MByte external data memory space.

The ADuC831 is shipped with factory programmed calibration coefficients that are automatically downloaded to the ADC on power-up ensuring optimum ADC performance. The ADC core contains internal offset and gain calibration registers, that can be hardware calibrated to minimize system errors.

A voltage output from an on-chip band gap reference proportional to absolute temperature can also be routed through the front end ADC multiplexor (effectively a ninth ADC channel input) facilitating a temperature sensor implementation.

ADC Transfer Function

The analog input range for the ADC is 0 V to V_{REF}. For this range, the designed code transitions occur midway between successive integer LSB values (i.e., 1/2 LSB, 3/2 LSBs, 5/2 LSBs, . . ., FS –3/2 LSBs). The output coding is straight binary with 1 LSB = FS/4096 or 2.5 V/4096 = 0.61 mV when V_{REF} = 2.5 V. The ideal input/output transfer characteristic for the 0 to V_{REF} range is shown in Figure 7.

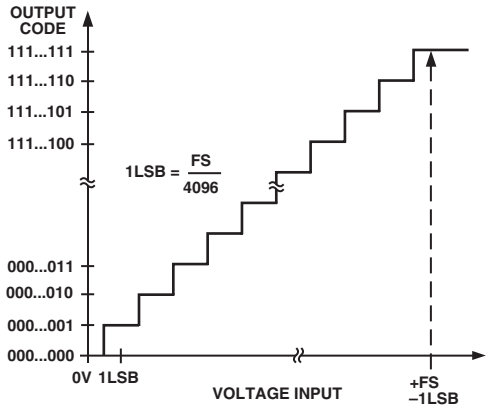

Figure 7. ADC Transfer Function

Typical Operation

Once configured via the ADCCON 1-3 SFRs the ADC will convert the analog input and provide an ADC 12-bit result word in the ADCDATAH/L SFRs. The top four bits of the ADCDATAH SFR will be written with the channel selection bits so as to identify the channel result. The format of the ADC 12 bit result word is shown in Figure 8.

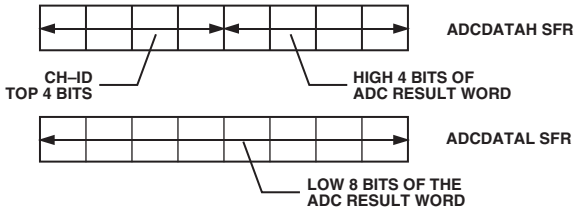

Figure 8. ADC Result Format

USER INTERFACE TO OTHER ON-CHIP ADuC831 PERIPHERALS

The following section gives a brief overview of the various peripherals also available on-chip. A summary of the SFRs used to control and configure these peripherals is also given.

DAC

The ADuC831 incorporates two 12-bit, voltage output DACs on-chip. Each has a rail-to-rail voltage output buffer capable of driving 10 kΩ/100 pF. Each has two selectable ranges, 0 V to V_{REF} (the internal band gap 2.5 V reference) and 0 V to AV_{DD}. Each can operate in 12-bit or 8-bit mode. Both DACs share a control register, DACCON, and four data registers, DAC1H/L, DAC0H/L. It should be noted that in 12-bit asynchronous mode, the DAC voltage output will be updated as soon as the DACL data SFR has been written; therefore, the DAC data registers should be updated as DACH first, followed by DACL. Note: for correct DAC operation on the 0 to V_{REF} range, the ADC must be switched on. This results in the DAC using the correct reference value.

DACCON	**DAC Control Register**
SFR Address	FDH
Power-On Default Value	04H
Bit Addressable	No

Table IX. DACCON SFR Bit Designations

Bit	Name	Description
7	MODE	The DAC MODE bit sets the overriding operating mode for both DACs. Set to "1" = 8-Bit Mode (Write 8 Bits to DACxL SFR). Set to "0"= 12-Bit Mode.
6	RNG1	DAC1 Range Select Bit. Set to "1" = DAC1 Range 0–V_{DD}. Set to "0" = DAC1 Range 0–V_{REF}.
5	RNG0	DAC0 Range Select Bit. Set to "1" = DAC0 Range 0–$V_{DD.}$ Set to "0" = DAC0 Range 0–$V_{REF.}$
4	CLR1	DAC1 Clear Bit. Set to "0" = DAC1 Output Forced to 0 V. Set to "1" = DAC1 Output Normal.
3	CLR0	DAC0 Clear Bit. Set to "0" = DAC1 Output Forced to 0 V. Set to "1" = DAC1 Output Normal.
2	SYNC	DAC0/1 Update Synchronization Bit. When set to "1" the DAC outputs update as soon as DACxL SFRs are written. The user can simultaneously update both DACs by first updating the DACxL/H SFRs while SYNC is "0." Both DACs will then update simultaneously when the SYNC bit is set to "1."
1	$\overline{PD1}$	DAC1 Power-Down Bit. Set to "1" = Power-On DAC1. Set to "0" = Power-Off DAC1.
0	PD0	DAC0 Power-Down Bit. Set to "1" = Power-On DAC0. Set to "0" = Power-Off DAC0.

DACxH/L	**DAC Data Registers**			
Function	DAC Data Registers, written by user to update the DAC output.			
SFR Address	DAC0L (DAC0 Data Low Byte)	→ F9H; DAC1L (DAC1 Data Low Byte)	→ FBH	
	DAC0H (DAC0 Data High Byte)	→ FAH; DAC1H(DAC1 Data High Byte)	→ FCH	
Power-On Default Value	00H	→ All four Registers		
Bit Addressable	No	→ All four Registers		

The 12-bit DAC data should be written into DACxH/L right-justified such that DACxL contains the lower eight bits, and the lower nibble of DACxH contains the upper four bits.

PULSEWIDTH MODULATOR (PWM)

The PWM on the ADuC831 is highly flexible PWM offering programmable resolution and input clock, and can be configured for any one of six different modes of operation. Two of these modes allow the PWM to be configured as a Σ-Δ DAC with up to 16 bits of resolution. A block diagram of the PWM is shown in Figure 26.

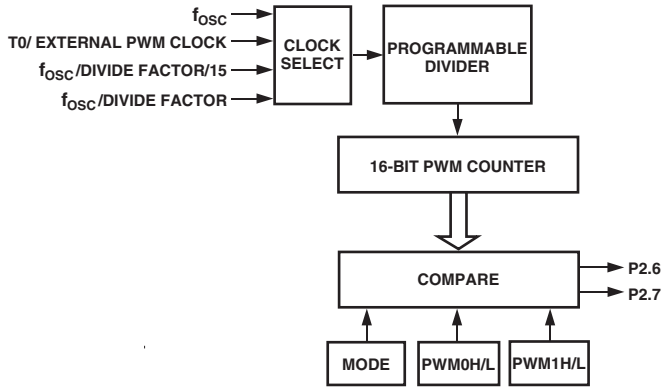

Figure 26. PWM Block Diagram

The PWM uses five SFRs: the control SFR (PWMCON), and four data SFRs (PWM0H, PWM0L, PWM1H, and PWM1L). PWMCON (as described below) controls the different modes of operation of the PWM as well as the PWM clock frequency. PWM0H/L and PWM1H/L are the data registers that determine the duty cycles of the PWM outputs. The output pins that the PWM uses are determined by the CFG831 register and they can be either P2.6 and P2.7 or P3.4 and P3.3. In this section of the data sheet, it is assumed that P2.6 and P2.7 are selected as the PWM outputs.

To use the PWM user software, first write to PWMCON to select the PWM mode of operation and the PWM input clock. Writing to PWMCON also resets the PWM counter. In any of the 16-bit modes of operation (modes 1, 3, 4, 6), user software should write to the PWM0L or PWM1L SFRs first. This value is written to a hidden SFR. Writing to the PWM0H or PWM1H SFRs updates both the PWMxH and the PWMxL SFRs but does not change the outputs until the end of the PWM cycle in progress. The values written to these 16-bit registers are then used in the next PWM cycle.

PWMCON	**PWM Control SFR**
SFR Address	AEH
Power-On Default Value	00H
Bit Addressable	No

Table X. PWMCON SFR Bit Designations

Bit	Name	Description
7	SNGL	Turns Off PWM output at P2.6 or P3.4 Leaving Port Pin Free for Digital I/O.
6	MD2	PWM Mode Bits
5	MD1	The MD2/1/0 bits choose the PWM mode as follows:
4	MD0	MD2 MD1 MD0 Mode 0 0 0 Mode 0: PWM Disabled 0 0 1 Mode 1: Single variable resolution PWM on P2.7 or P3.3 0 1 0 Mode 2: Twin 8-bit PWM 0 1 1 Mode 3: Twin 16-bit PWM 1 0 0 Mode 4: Dual NRZ 16-bit Σ-Δ DAC 1 0 1 Mode 5: Dual 8-bit PWM 1 1 0 Mode 6: Dual RZ 16-bit Σ-Δ DAC 1 1 1 Reserved for future use
3	CDIV1	PWM Clock Divider
2	CDIV0	Scale the clock source for the PWM counter as shown below: CDIV1 CDIV0 Description 0 0 PWM Counter = Selected Clock/1 0 1 PWM Counter = Selected Clock/4 1 0 PWM Counter = Selected Clock/16 1 1 PWM Counter = Selected Clock/64
1	CSEL1	PWM Clock Divider
0	CSEL0	Select the clock source for the PWM as shown below: CSEL1 CSEL0 Description 0 0 PWM Clock = $f_{OCS/DIVIDE\ FACTOR}$/15 (see CFG831 register) 0 1 PWM Clock = $f_{OCS/DIVIDE\ FACTOR}$ (see CFG831 register) 1 0 PWM Clock = External input at P3.4/T0 1 1 PWM Clock = f_{OSC}

PWM MODES OF OPERATION

MODE 0: PWM Disabled

The PWM is disabled, allowing P2.6 and P2.7 to be used as normal.

MODE 1: Single Variable Resolution PWM

In Mode 1, both the pulse length and the cycle time (period) are programmable in user code, allowing the resolution of the PWM to be variable.

PWM1H/L sets the period of the output waveform. Reducing PWM1H/L reduces the resolution of the PWM output but increases the maximum output rate of the PWM.

(For example, setting PWM1H/L to 65536 gives a 16-bit PWM with a maximum output rate of 244 Hz (16 MHz/65536). Setting PWM1H/L to 4096 gives a 12-bit PWM with a maximum output rate of 3906 Hz (16 MHz/4096).)

PWM0H/L sets the duty cycle of the PWM output waveform, as shown in the diagram below.

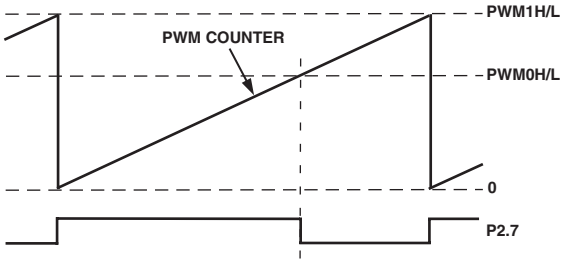

Figure 27. ADuC831 PWM in Mode 1

MODE 2: Twin 8-Bit PWM

In Mode 2, the duty cycle of the PWM outputs and the resolution of the PWM outputs are both programmable. The maximum resolution of the PWM output is eight bits.

PWM1L sets the period for both PWM outputs. Typically, this will be set to 255 (FFH) to give an 8-bit PWM, although it is possible to reduce this as necessary. A value of 100 could be loaded here to give a percentage PWM (i.e., the PWM is accurate to 1%).

The outputs of the PWM at P2.6 and P2.7 are shown in the diagram below. As can be seen, the output of PWM0 (P2.6) goes low when the PWM counter equals PWM0L. The output of PWM1 (P2.7) goes high when the PWM counter equals PWM1H, and goes low again when the PWM counter equals PWM0H. Setting PWM1H to 0 ensures that both PWM outputs start simultaneously.

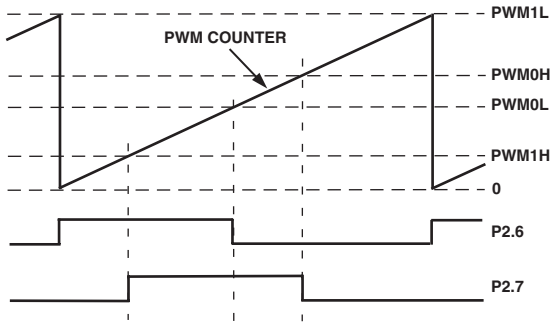

Figure 28. PWM Mode 2

MODE 3: Twin 16-Bit PWM

In Mode 3, the PWM counter is fixed to count from 0 to 65536 giving a fixed 16-bit PWM. Operating from the 16 MHz core clock results in a PWM output rate of 244 Hz. The duty cycle of the PWM outputs at P2.6 and P2.7 are independently programmable.

As shown in Figure 29, while the PWM counter is less than PWM0H/L, the output of PWM0 (P2.6) is high. Once the PWM counter equals PWM0H/L, then PWM0 (P2.6) goes low and remains low until the PWM counter rolls over.

Similarly while the PWM counter is less than PWM1H/L, the output of PWM1 (P2.7) is high. Once the PWM counter equals PWM1H/L, then PWM1 (P2.7) goes low and remains low until the PWM counter rolls over.

In this mode, both PWM outputs are synchronized. Once the PWM counter rolls over to 0, both PWM0 (P2.6) and PWM1 (P2.7) will go high.

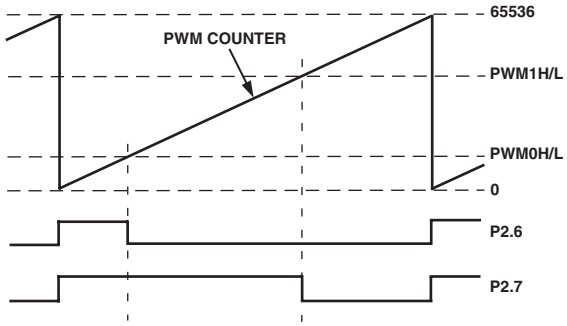

Figure 29. PWM Mode 3

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.

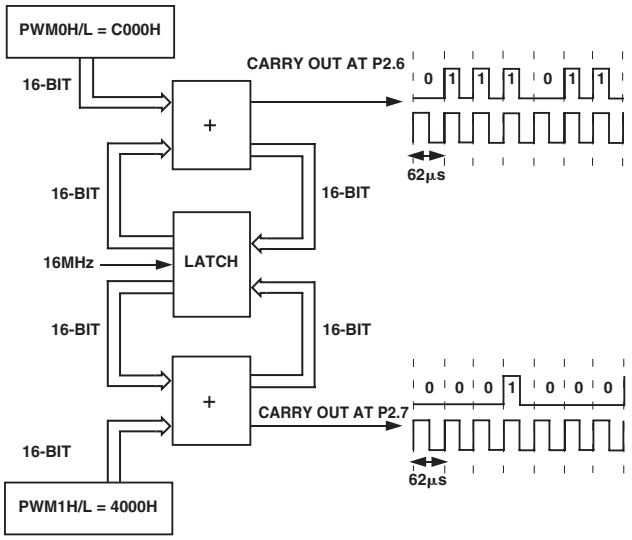

Figure 30. PWM Mode 4

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.

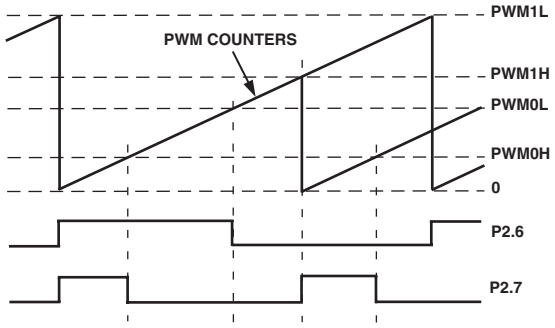

Figure 31. PWM Mode 5

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.

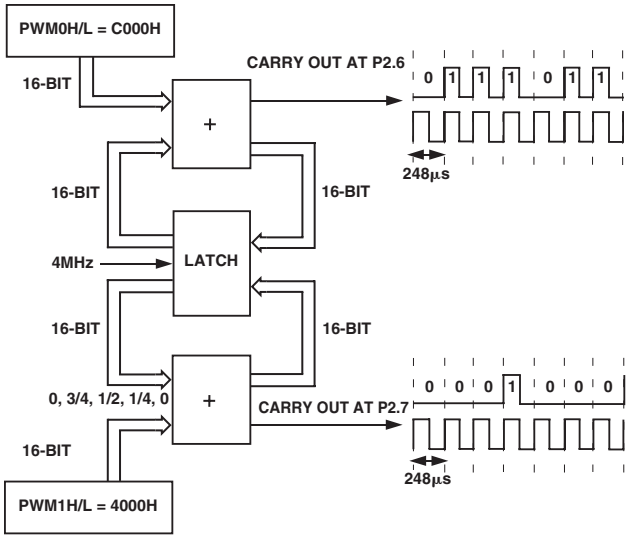

Figure 32. PWM Mode 6

The main features of the MicroConverter I²C interface are:

- Only two bus lines are required; a serial data line (SDATA) and a serial clock line (SCLOCK).
- An I²C master can communicate with multiple slave devices. Because each slave device has a unique 7-bit address, single master/slave relationships can exist at all times even in a multislave environment (Figure 34).
- On-Chip filtering rejects <50 ns spikes on the SDATA and the SCLOCK lines to preserve data integrity.

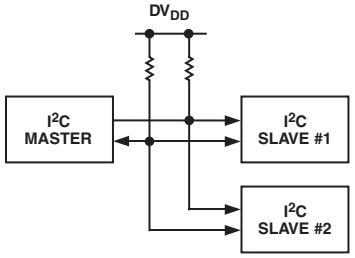

Figure 34. Typical I²C System

Software Master Mode

The ADuC831 can be used as an I²C master device by configuring the I²C peripheral in master mode and writing software to output the data bit by bit. This is referred to as a software master. Master mode is enabled by setting the I2CM bit in the I2CCON register.

To transmit data on the SDATA line, MDE must be set to enable the output driver on the SDATA pin. If MDE is set, then the SDATA pin will be pulled high or low depending on whether the MDO bit is set or cleared. MCO controls the SCLOCK pin and is always configured as an output in master mode. In master mode the SCLOCK pin will be pulled high or low depending on the whether MCO is set or cleared.

To receive data, MDE must be cleared to disable the output driver on SDATA. Software must provide the clocks by toggling the MCO bit and read the SDATA pin via the MDI bit. If MDE is cleared MDI can be used to read the SDATA pin. The value of the SDATA pin is latched into MDI on a rising edge of SCLOCK. MDI is set if the SDATA pin was high on the last rising edge of SCLOCK. MDI is cleared if the SDATA pin was low on the last rising edge of SCLOCK.

Software must control MDO, MCO, and MDE appropriately to generate the START condition, slave address, acknowledge bits, data bytes, and STOP conditions appropriately. These functions are provided in technical note uC001.

Hardware Slave Mode

After reset the ADuC831 defaults to hardware slave mode. The I²C interface is enabled by clearing the SPE bit in SPICON. Slave mode is enabled by clearing the I2CM bit in I2CCON. The ADuC831 has a full hardware slave. In slave mode the I²C address is stored in the I2CADD register. Data received or to be transmitted is stored in the I2CDAT register.

Once enabled in I²C slave mode the slave controller waits for a START condition. If the ADuC831 detects a valid start condition, followed by a valid address, followed by the R/$\overline{W}$ bit, the I2CI interrupt bit will get set by the hardware automatically.

The I²C peripheral will only generate a core interrupt if the user has preconfigured the I²C interrupt enable bit in the IEIP2 SFR as well as the global interrupt bit EA in the IE SFR.

```
; Enabling I2C Interrupts for the ADuC831
MOV  IEIP2,#01H     ; enable I2C interrupt
SETB EA
```

On the ADuC831 an autoclear of the I2CI bit is implemented so this bit is cleared automatically on a read or write access to the I2CDAT SFR.

```
MOV     I2CDAT, A   ; I2CI auto cleared
MOV     A, I2CDAT   ; I2CI auto cleared
```

If for any reason the user tries to clear the interrupt more than once, i.e., access the data SFR more than once per interrupt then the I²C controller will halt. The interface will then have to be reset using the I2CRS bit.

The user can choose to poll the I2CI bit or enable the interrupt. In the case of the interrupt, the PC counter will vector to 003BH at the end of each complete byte. For the first byte when the user gets to the I2CI ISR, the 7-bit address and the R/$\overline{W}$ bit will appear in the I2CDAT SFR.

The I2CTX bit contains the R/$\overline{W}$ bit sent from the master. If I2CTX is set then the master would like to receive a byte. Thus the slave will transmit data by writing to the I2CDAT register. If I2CTX is cleared, the master would like to transmit a byte. Therefore, the slave will receive a serial byte. Software can interrogate the state of I2CTX to determine whether it should write to or read from I2CDAT.

Once the ADuC831 has received a valid address, hardware will hold SCLOCK low until the I2CI bit is cleared by software. This allows the master to wait for the slave to be ready before transmitting the clocks for the next byte.

The I2CI interrupt bit will be set every time a complete data byte is received or transmitted, provided it is followed by a valid ACK. If the byte is followed by a NACK an interrupt is NOT generated. The ADuC831 will continue to issue interrupts for each complete data byte transferred until a STOP condition is received or the interface is reset.

When a STOP condition is received, the interface will reset to a state where it is waiting to be addressed (idle). Similarly, if the interface receives a NACK at the end of a sequence it also returns to the default idle state. The I2CRS bit can be used to reset the I²C interface. This bit can be used to force the interface back to the default idle state.

It should be noted that there is no way (in hardware) to distinguish between an interrupt generated by a received START + valid address and an interrupt generated by a received data byte. User software must be used to distinguish between these interrupts.

8052 COMPATIBLE ON-CHIP PERIPHERALS

This section gives a brief overview of the various secondary peripheral circuits that are also available to the user on-chip. These remaining functions are mostly 8052 compatible (with a few additional features) and are controlled via standard 8052 SFR bit definitions.

Parallel I/O

The ADuC831 uses four input/output ports to exchange data with external devices. In addition to performing general-purpose I/O, some ports are capable of external memory operations while others are multiplexed with alternate functions for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general-purpose I/O pin.

Port 0

Port 0 is an 8-bit, open-drain, bidirectional I/O port that is directly controlled via the Port 0 SFR. Port 0 is also the multiplexed low-order address and data bus during accesses to external program or data memory.

Figure 36 shows a typical bit latch and I/O buffer for a Port 0 port pin. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. See the following Read-Modify-Write Instructions section for more details.

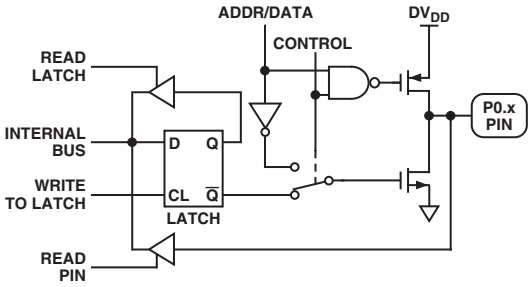

Figure 36. Port 0 Bit Latch and I/O Buffer

As shown in Figure 36, the output drivers of Port 0 pins are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses the P0 SFR gets 1s written to it (i.e., all of its bit latches become 1). When accessing external memory, the CONTROL signal in Figure 36 goes high, enabling push-pull operation of the output pin from the internal address or data bus (ADDR/DATA line). Therefore, no external pull-ups are required on Port 0 in order for it to access external memory.

In general-purpose I/O port mode, Port 0 pins that have 1s written to them via the Port 0 SFR will be configured as "open drain" and will therefore float. In this state, Port 0 pins can be used as high impedance inputs. This is represented in Figure 36 by the NAND gate whose output remains high as long as the CONTROL signal is low, thereby disabling the top FET. External pull-up resistors are therefore required when Port 0 pins are used as general-purpose outputs. Port 0 pins with 0s written to them will drive a logic low output voltage (V_{OL}) and will be capable of sinking 1.6 mA.

Port 1

Port 1 is also an 8-bit port directly controlled via the P1 SFR. Port 1 digital output capability is not supported on this device. Port 1 pins can be configured as digital inputs or analog inputs.

By (power-on) default these pins are configured as analog inputs, i.e., 1 written in the corresponding Port 1 register bit. To configure any of these pins as digital inputs, the user should write a 0 to these port bits to configure the corresponding pin as a high impedance digital input.

These pins also have various secondary functions described in Table XVII.

Table XVII. Port 1, Alternate Pin Functions

Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)
P1.5	$\overline{SS}$ (Slave Select for the SPI Interface)

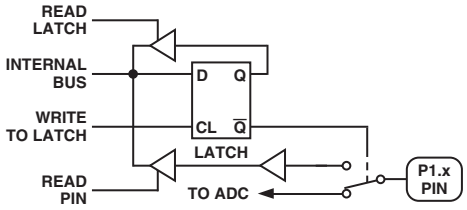

Figure 37. Port 1 Bit Latch and I/O Buffer

Port 2

Port 2 is a bidirectional port with internal pull-up resistors directly controlled via the P2 SFR. Port 2 also emits the high order address bytes during fetches from external program memory and middle and high order address bytes during accesses to the 24-bit external data memory space.

As shown in Figure 38, the output drivers of Ports 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses (as for Port 0). In external memory addressing mode (CONTROL = 1) the port pins feature push-pull operation controlled by the internal address bus (ADDR line). However, unlike the P0 SFR during external memory accesses, the P2 SFR remains unchanged.

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.

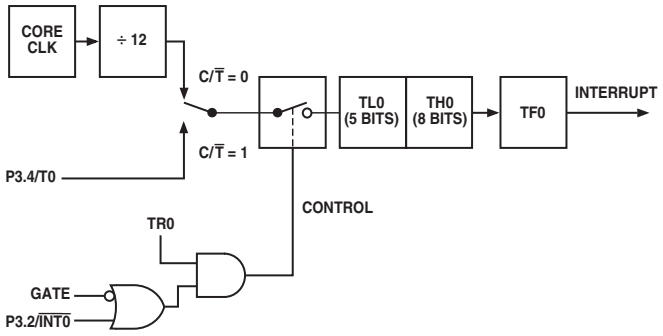

Figure 45. Timer/Counter 0, Mode 0

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.

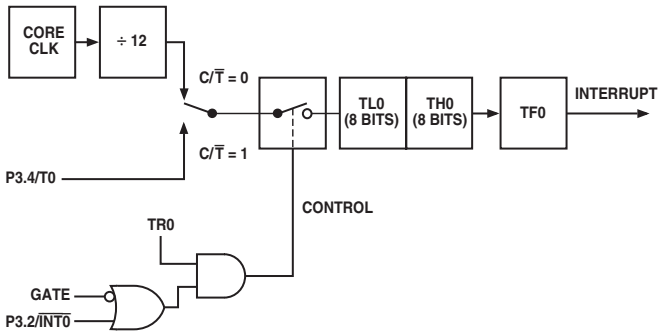

Figure 46. Timer/Counter 0, Mode 1

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.

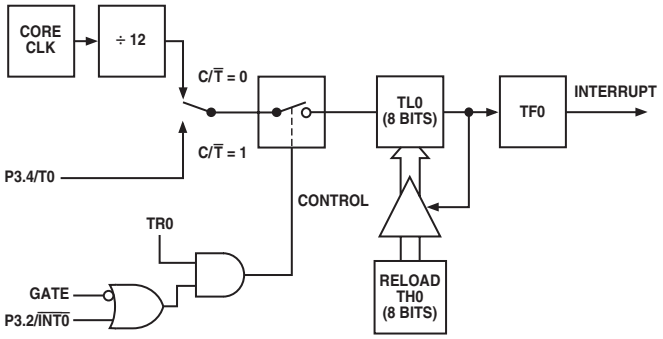

Figure 47. Timer/Counter 0, Mode 2

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.

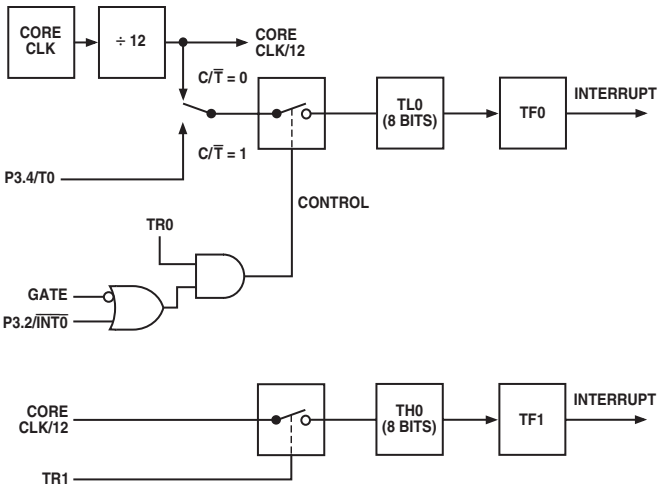

Figure 48. Timer/Counter 0, Mode 3

Timer 1 Generated Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$Modes\ 1\ and\ 3\ Baud\ Rate = (2^{SMOD}/32) \times (Timer\ 1\ Overflow\ Rate)$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either timer or counter operation, and in any of its three running modes. In the most typical application, it is configured for timer operation in the Autoreload mode (high nibble of TMOD = 0010 binary). In that case, the baud rate is given by the formula:

$$Modes\ 1\ and\ 3\ Baud\ Rate = (2^{SMOD}/32) \times (Core\ Clock/(12 \times [256 - TH1]))$$

Table XXIV shows some commonly used baud rates and how they might be calculated from a core clock frequency of 11.0592 MHz and 12 MHz. Generally speaking, a 5% error is tolerable using asynchronous (start/stop) communications.

Table XXIV. Commonly-Used Baud Rates, Timer 1

Ideal Baud	Core CLK (MHz)	SMOD Value	TH1-Reload Value	Actual Baud	% Error
9600	12	1	–7 (F9H)	8929	7
19200	11.0592	1	–3 (FDH)	19200	0
9600	11.0592	0	–3 (FDH)	9600	0
2400	11.0592	0	–12 (F4H)	2400	0

Timer 2 Generated Baud Rates

Baud rates can also be generated using Timer 2. Using Timer 2 is similar to using Timer 1 in that the timer must overflow 16 times before a bit is transmitted/received. Because Timer 2 has a 16-bit Autoreload mode, a wider range of baud rates is possible using Timer 2.

$$Modes\ 1\ and\ 3\ Baud\ Rate = (1/16) \times (Timer\ 2\ Overflow\ Rate)$$

Therefore, when Timer 2 is used to generate baud rates, the timer increments every two clock cycles and not every core machine cycle as before. Thus, it increments six times faster than Timer 1, and therefore baud rates six times faster are possible. Because Timer 2 has 16-bit autoreload capability, very low baud rates are still possible.

Timer 2 is selected as the baud rate generator by setting the TCLK and/or RCLK in T2CON. The baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode as shown in Figure 53.

In this case, the baud rate is given by the formula:

$$Modes\ 1\ and\ 3\ Baud\ Rate = (Core\ Clk)/(32 \times [65536 - (RCAP2H,\ RCAP2L)])$$

Table XXV shows some commonly used baud rates and how they might be calculated from a core clock frequency of 11.0592 MHz and 12 MHz.

Table XXV. Commonly Used Baud Rates, Timer 2

Ideal Baud	Core CLK (MHz)	RCAP2H Value	RCAP2L Value	Actual Baud	% Error
19200	12	–1 (FFH)	–20 (ECH)	19661	2.4
9600	12	–1 (FFH)	–41 (D7H)	9591	0.1
2400	12	–1 (FFH)	–164 (5CH)	2398	0.1
1200	12	–2 (FEH)	–72 (B8H)	1199	0.1
19200	11.0592	–1 (FFH)	–18 (EEH)	19200	0
9600	11.0592	–1 (FFH)	–36 (DCH)	9600	0
2400	11.0592	–1 (FFH)	–144 (70H)	2400	0
1200	11.0592	–2 (FFH)	–32 (E0H)	1200	0

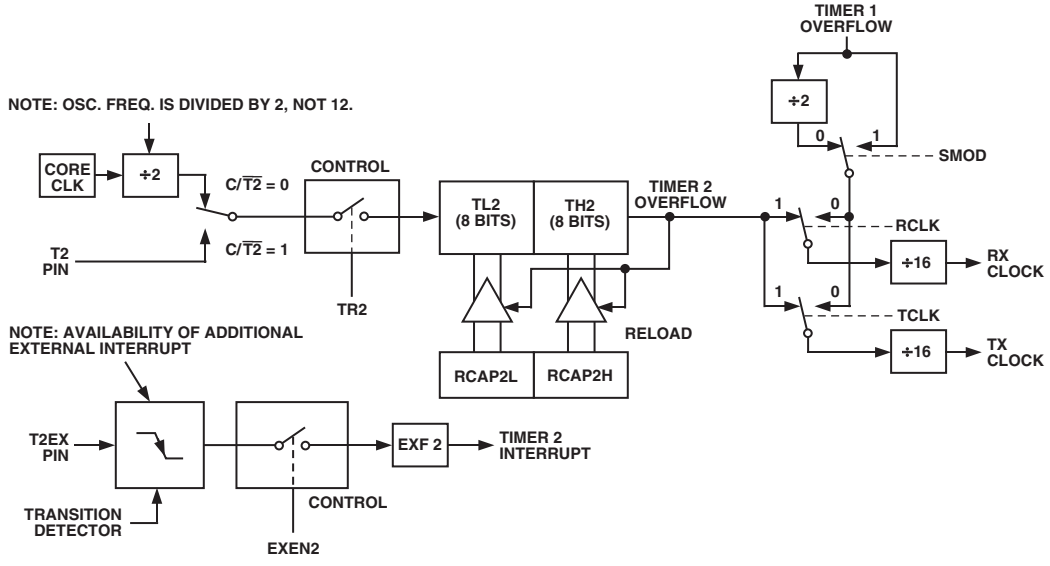

Figure 53. Timer 2, UART Baud Rates

Interrupt Priority

The Interrupt Enable registers are written by the user to enable individual interrupt sources, while the Interrupt Priority registers allow the user to select one of two priority levels for each interrupt. An interrupt of a high priority may interrupt the service routine of a low priority interrupt, and if two interrupts of different priority occur at the same time, the higher level interrupt will be serviced first. An interrupt cannot be interrupted by another interrupt of the same priority level. If two interrupts of the same priority level occur simultaneously, a polling sequence is observed as shown in Table XXXI.

Table XXXI. Priority within an Interrupt Level

Source	Priority	Description
PSMI	1 (Highest)	Power Supply Monitor Interrupt
WDS	2	Watchdog Timer Interrupt
IE0	2	External Interrupt 0
ADCI	3	ADC Interrupt
TF0	4	Timer/Counter 0 Interrupt
IE1	5	External Interrupt 1
TF1	6	Timer/Counter 1 Interrupt
I2CI + ISPI	7	SPI Interrupt
RI + TI	8	Serial Interrupt
TF2 + EXF2	9 (Lowest)	Timer/Counter 2 Interrupt
TII	11 (Lowest)	Time Interval Counter Interrupt

Interrupt Vectors

When an interrupt occurs, the program counter is pushed onto the stack and the corresponding interrupt vector address is loaded into the program counter. The Interrupt Vector Addresses are shown in Table XXXII.

Table XXXII. Interrupt Vector Addresses

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
ADCI	0033H
I2CI + ISPI	003BH
PSMI	0043H
TII	0053H
WDS	005BH

ADuC831 HARDWARE DESIGN CONSIDERATIONS

This section outlines some of the key hardware design considerations that must be addressed when integrating the ADuC831 into any hardware system.

Clock Oscillator

The clock source for the ADuC831 can come either from an external source or from the internal clock oscillator. To use the internal clock oscillator, connect a parallel resonant crystal between XTAL1 and XTAL2, and connect a capacitor from each pin to ground as shown below.

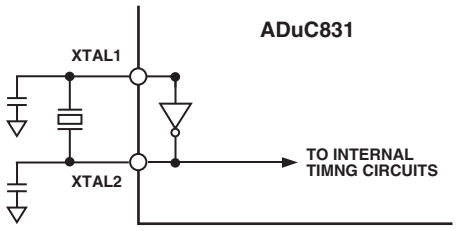

Figure 55. External Parallel Resonant Crystal Connections

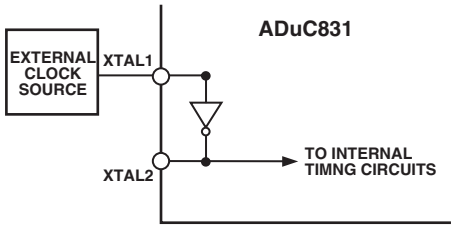

Figure 56. Connecting an External Clock Source

Whether using the internal oscillator or an external clock source, the ADuC831's specified operational clock speed range is 400 kHz to 16 MHz. The core itself is static, and will function all the way down to dc. But at clock speeds slower that 400 kHz the ADC will no longer function correctly. Therefore, to ensure specified operation, use a clock frequency of at least 400 kHz and no more than 16 MHz. Note: the Flash/EE memory may not program correctly at a clock frequency of less than 2 MHz.

External Memory Interface

In addition to its internal program and data memories, the ADuC831 can access up to 64 kBytes of external program memory (ROM/PROM/etc.) and up to 16 MBytes of external data memory (SRAM).

To select from which code space (internal or external program memory) to begin executing instructions, tie the $\overline{EA}$ (external access) pin high or low, respectively. When $\overline{EA}$ is high (pulled up to V_{DD}), user program execution will start at address 0 of the internal 62 kBytes Flash/EE code space. When $\overline{EA}$ is low (tied to ground) user program execution will start at address 0 of the external code space.

A second very important function of the $\overline{EA}$ pin is described in the Single Pin Emulation Mode section.

External program memory (if used) must be connected to the ADuC831 as illustrated in Figure 57. Note that 16 I/O lines

(Ports 0 and 2) are dedicated to bus functions during external program memory fetches. Port 0 (P0) serves as a multiplexed address/data bus. It emits the low byte of the program counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the program memory. During the time that the low byte of the program counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2) emits the high byte of the program counter (PCH), then $\overline{\text{PSEN}}$ strobes the EPROM and the code byte is read into the ADuC831.

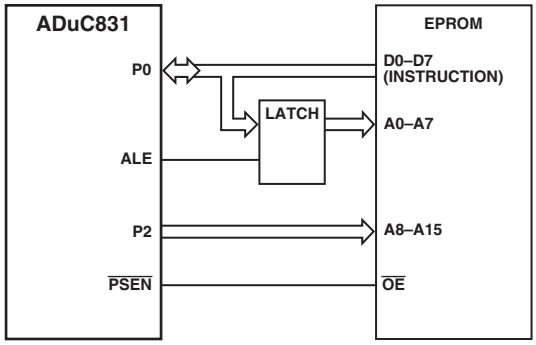

Figure 57. External Program Memory Interface

Note that program memory addresses are always 16 bits wide, even in cases where the actual amount of program memory used is less than 64 kBytes. External program execution sacrifices two of the 8-bit ports (P0 and P2) to the function of addressing the program memory. While executing from external program memory, Ports 0 and 2 can be used simultaneously for read/write access to external data memory, but not for general-purpose I/O.

Though both external program memory and external data memory are accessed by some of the same pins, the two are completely independent of each other from a software point of view. For example, the chip can read/write external data memory while executing from external program memory.

Figure 58 shows a hardware configuration for accessing up to 64 kBytes of external RAM. This interface is standard to any 8051 compatible MCU.

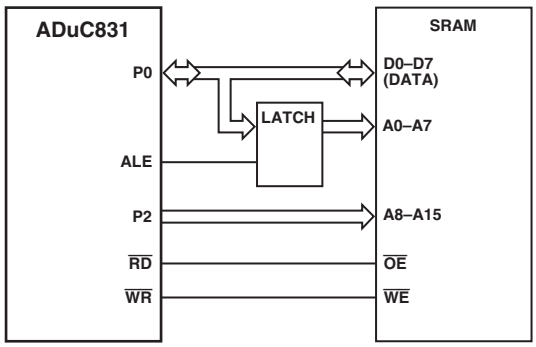

Figure 58. External Data Memory Interface (64 K Address Space)

If access to more than 64 kBytes of RAM is desired, a feature unique to the ADuC831 allows addressing up to 16 MBytes of external RAM simply by adding an additional latch as illustrated in Figure 59.

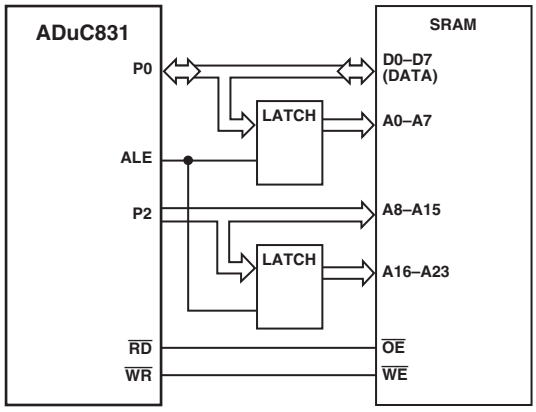

Figure 59. External Data Memory Interface (16 MBytes Address Space)

In either implementation, Port 0 (P0) serves as a multiplexed address/data bus. It emits the low byte of the data pointer (DPL) as an address, which is latched by a pulse of ALE prior to data being placed on the bus by the ADuC831 (write operation) or the SRAM (read operation). Port 2 (P2) provides the data pointer page byte (DPP) to be latched by ALE, followed by the data pointer high byte (DPH). If no latch is connected to P2, DPP is ignored by the SRAM, and the 8051 standard of 64 kBytes external data memory access is maintained.

Power Supplies

The ADuC831's operational power supply voltage range is 2.7 V to 5.25 V. Although the guaranteed data sheet specifications are given only for power supplies within 2.7 V to 3.6 V or ±10% of the nominal 5 V level, the chip will function equally well at any power supply level between 2.7 V and 5.5 V.

Note: Figures 60 and 61 refer to the PQFP package, for the CSP package connect the extra DV_{DD}, DG_{ND}, AV_{DD}, and AG_{ND} in the same manner. Note: for the CSP package, the bottom paddle should be left unconnected.

Separate analog and digital power supply pins (AV_{DD} and DV_{DD}, respectively) allow AV_{DD} to be kept relatively free of noisy digital signals often present on the system DV_{DD} line. However, though you can power AV_{DD} and DV_{DD} from two separate supplies if desired, you must ensure that they remain within ±0.3 V of one another at all times in order to avoid damaging the chip (as per the Absolute Maximum Ratings section). Therefore, it is recommended that unless AV_{DD} and DV_{DD} are connected directly together, you connect back-to-back Schottky diodes between them as shown in Figure 60.

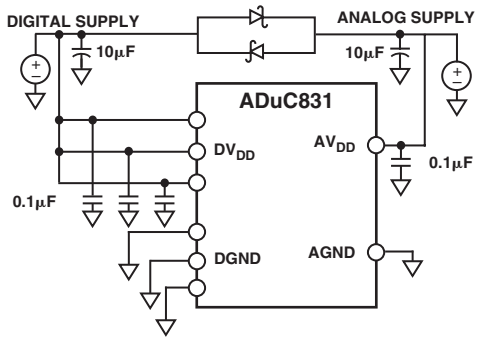

Figure 60. External Dual-Supply Connections

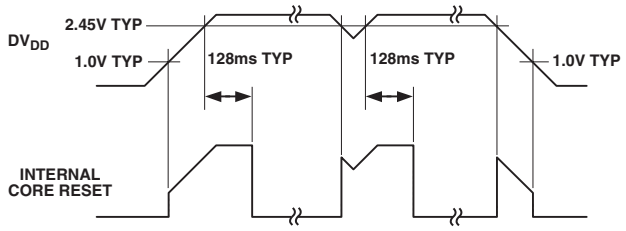

Figure 62. Internal POR Operation

Grounding and Board Layout Recommendations

As with all high resolution data converters, special attention must be paid to grounding and PC board layout of ADuC831-based designs in order to achieve optimum performance from the ADC and DACs.

Although the ADuC831 has separate pins for analog and digital ground (AGND and DGND), the user must not tie these to two separate ground planes unless the two ground planes are connected together very close to the ADuC831, as illustrated in the simplified example of Figure 63a. In systems where digital and analog ground planes are connected together somewhere else (at the system's power supply for example), they cannot be connected again near the ADuC831 since a ground loop would result. In these cases, tie the ADuC831's AGND and DGND pins all to the analog ground plane, as illustrated in Figure 63b. In systems with only one ground plane, ensure that the digital and analog components are physically separated onto separate halves of the board such that digital return currents do not flow near analog circuitry and vice versa. The ADuC831 can then be placed between the digital and analog sections, as illustrated in Figure 63c.

In all of these scenarios, and in more complicated real-life applications, keep in mind the flow of current from the supplies and back to ground. Make sure the return paths for all currents are as close as possible to the paths the currents took to reach their destinations. For example, do not power components on the analog side of Figure 63b with DV_{DD} since that would force return currents from DV_{DD} to flow through AGND. Also, try to avoid digital currents flowing under analog circuitry, which could happen if the user placed a noisy digital chip on the left half of the board in Figure 63c. Whenever possible, avoid large discontinuities in the ground plane(s) (such as are formed by a long trace on the same layer), since they force return signals to travel a longer path. And of course, make all connections to the ground plane directly, with little or no trace separating the pin from its via to ground. Note that the bottom paddle of the CSP package should not be connected to ground. It should be left unconnected.

If the user plans to connect fast logic signals (rise/fall time < 5 ns) to any of the ADuC831's digital inputs, add a series resistor to each relevant line to keep rise and fall times longer than 5 ns at the ADuC831 input pins. A value of 100 Ω or 200 Ω is usually sufficient to prevent high speed signals from coupling capacitively into the ADuC831 and affecting the accuracy of ADC conversions.

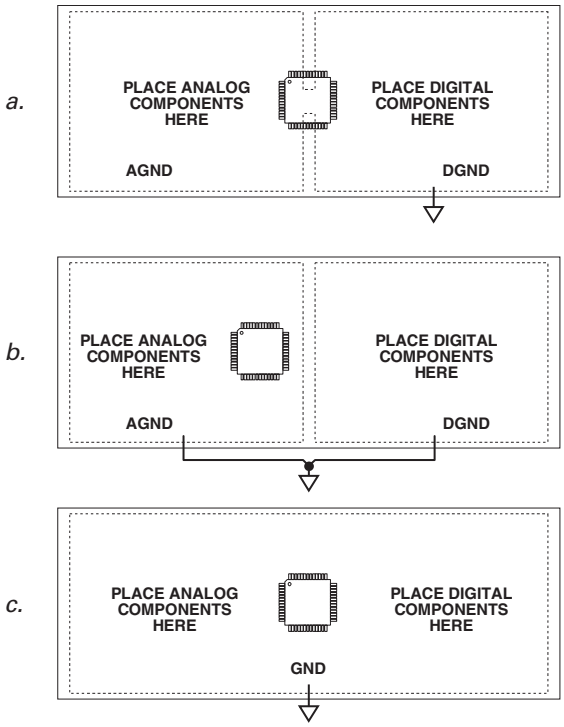

Figure 63. System Grounding Schemes

OTHER HARDWARE CONSIDERATIONS

To facilitate in-circuit programming, plus in-circuit debug and emulation options, users will want to implement some simple connection points in their hardware that will allow easy access to download, debug, and emulation modes.

In-Circuit Serial Download Access

Nearly all ADuC831 designs will want to take advantage of the in-circuit reprogrammability of the chip. This is accomplished by a connection to the ADuC831's UART, which requires an external RS-232 chip for level translation if downloading code from a PC. Basic configuration of an RS-232 connection is illustrated in Figure 66 with a simple ADM202-based circuit. If users would rather not design an RS-232 chip onto a board, refer to the application note "uC006–A 4-Wire UART-to-PC Interface"* for a simple (and zero-cost-per-board) method of gaining in-circuit serial download access to the ADuC831.

In addition to the basic UART connections, users will also need a way to trigger the chip into download mode. This is accomplished via a 1 kΩ pull-down resistor that can be jumpered onto the $\overline{\text{PSEN}}$ pin, as shown in Figure 64. To get the ADuC831 into download mode, simply connect this jumper and power-cycle the device (or manually reset the device, if a manual reset button is available) and it will be ready to receive a new program serially. With the jumper removed, the device will come up in normal mode (and run the program) whenever power is cycled or RESET is toggled.

*Application Note uC006 is available at www.analog.com/microconverter

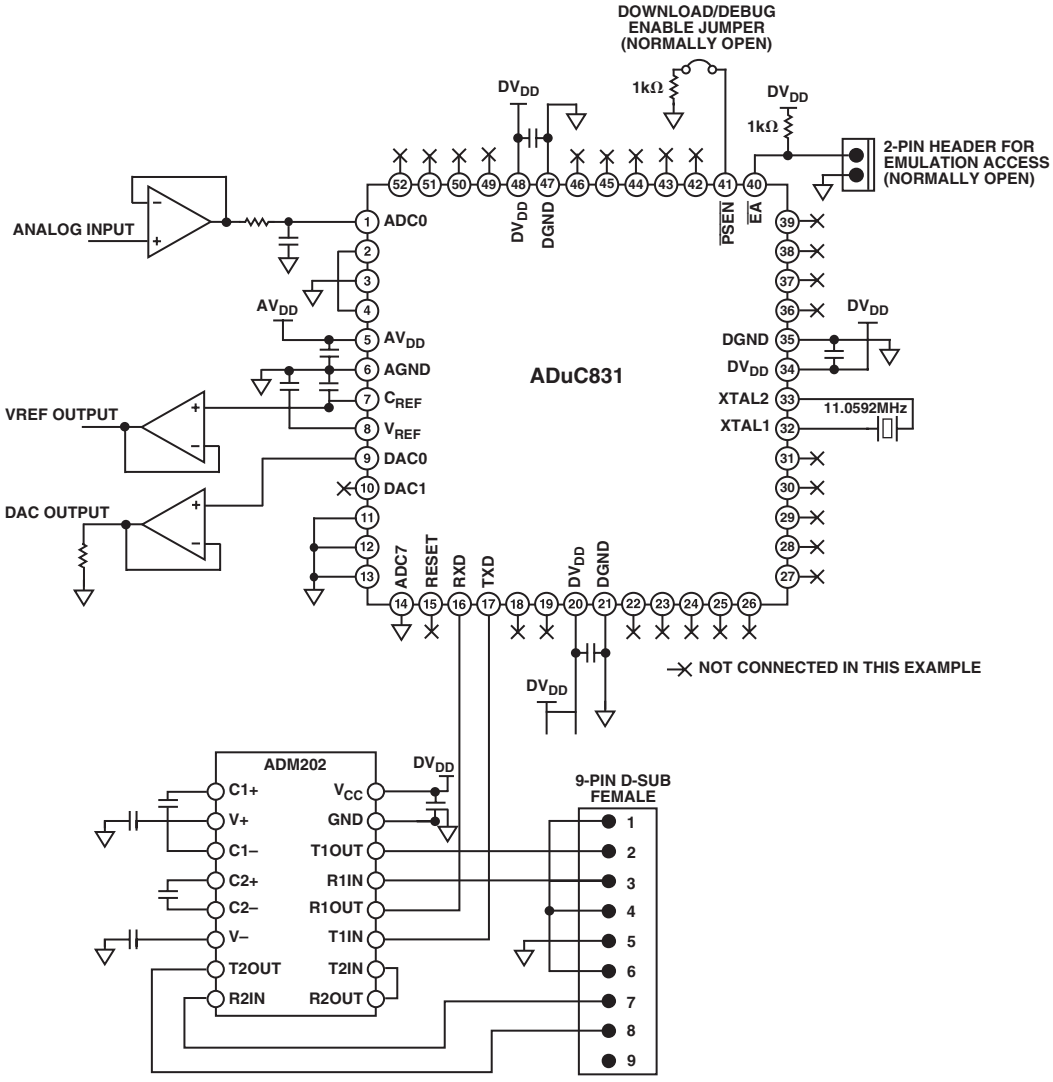

Figure 64. Example ADuC831 System (PQFP Package)

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 READ CYCLE							
t_{RLRH}	$\overline{RD}$ Pulsewidth	400		$6t_{CK} - 100$		ns	71
t_{AVLL}	Address Valid after ALE Low	43		$t_{CK} - 40$		ns	71
t_{LLAX}	Address Hold after ALE Low	48		$t_{CK} - 35$		ns	71
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CK} - 165$	ns	71
t_{RHDX}	Data and Address Hold after $\overline{RD}$	0		0		ns	71
t_{RHDZ}	Data Float after $\overline{RD}$		97		$2t_{CK} - 70$	ns	71
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CK} - 150$	ns	71
t_{AVDV}	Address to Valid Data In		585		$9t_{CK} - 165$	ns	71
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CK} - 50$	$3t_{CK} + 50$	ns	71
t_{AVWL}	Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CK} - 130$		ns	71
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns	71
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$t_{CK} - 40$	$6t_{CK} - 100$	ns	71

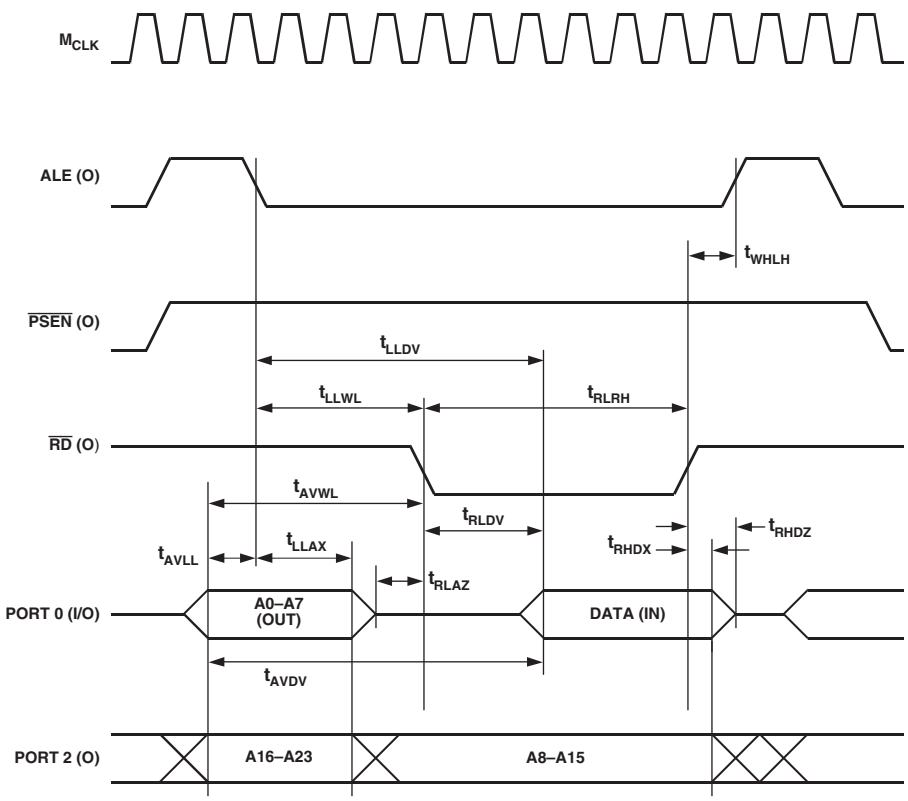

Figure 71. External Data Memory Read Cycle

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.

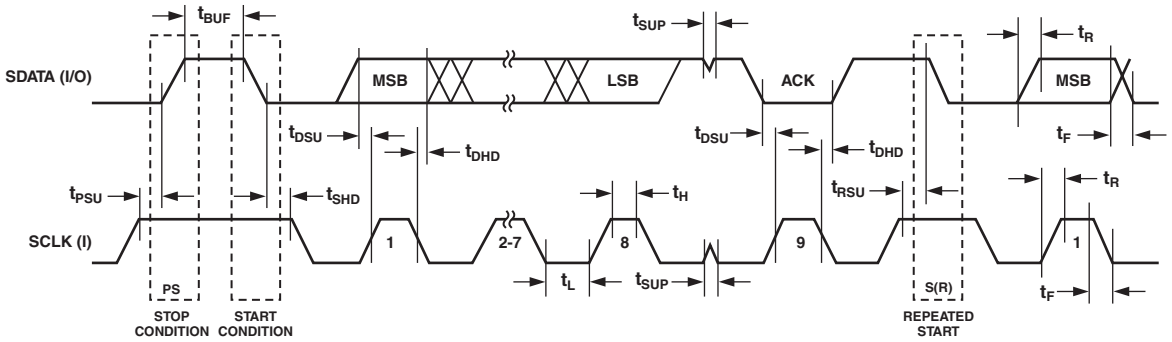

Figure 74. I²C Compatible Interface Timing

REVISION HISTORY

5/16—Rev. 0 to Rev. A

D02986-0-5/16(A)