MicroConverter® Products

ADuC812 User's Manual

Revision Draft.05

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.

Tel: 617-329-4700 Fax: 617-326-8703

CONTENTS:

Preface

Analog Devices Inc. designs and manufacturers a broad line of linear, mixed signal and digital signal processing integrated circuits. Analog Devices is recognized as an industry leader in the innovation and design of high-performance analog products and digital signal processing solutions. Find out more about Analog Devices from our web site - www.analog.com.

Analog Devices' new MicroConverter products feature a unique integration of technologies combining high-resolution data conversion with on-chip microcontroller and nonvolatile memory functionality. Together, these features make up the world's first high precision data acquisition system on a chip.

This User's Manual details both the hardware and software features of the ADuC812 MicroConverter. The intended audience is both the software programmer and hardware/system designer.

Additional documentation, including technical datasheets, technical notes and quick reference guides should be used to supplement this text. This additional documentation can be found at the Analog Devices MicroConverter web site. - www.analog.com/microconverter.

Customer Service and Technical Support:

For product marketing information or technical support, contact any Analog Devices sales office or authorized distributor.

For applications engineering assistance, please use the following contacts:

North America:	Call	800-ANALOG-D (800-262-5643)
	Email	linear.apps@analog.com
Europe:	Call	353-61-495969
	Email	euro.linear@analog.com
Rest of World:	Call	781-937-1428
	Email	linear.apps@analog.com

Chapter 1 Hardware Design Guide

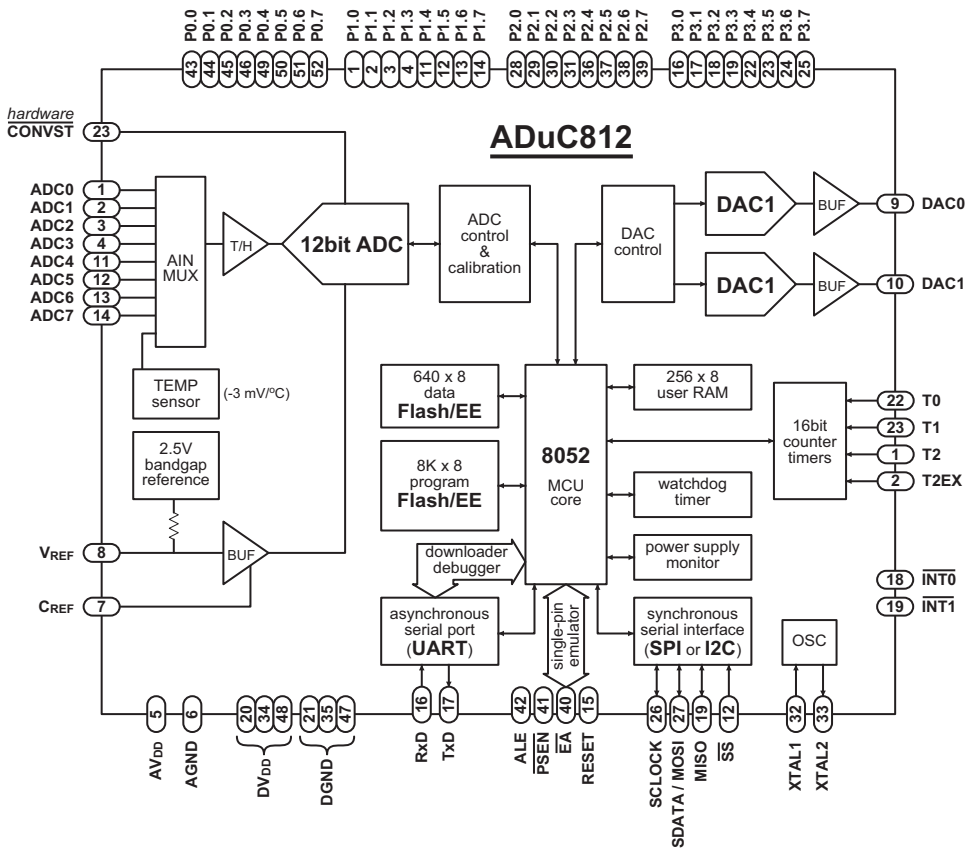


Figure 1 – Functional Block Diagram

This chapter is more than an architectural overview. It, along with the product datasheet, should be thought of as a hardware designer's primary source of information on the ADuC812. Future chapters will focus on integrated functions largely from a software configuration perspective. This chapter details the significant hardware design considerations required to facilitate successful integration of the ADuC812 into any hardware system.

1.1 Brief Overview of the ADuC812

The ADuC812 is a fully integrated 12-bit data acquisition system incorporating a high performance self-calibrating multichannel ADC, two 12-bit DACs, and a programmable 8-bit (8051-compatible) MCU on a single chip.

The Programmable 8051-compatible core is supported by 8K bytes Flash/EE program memory, 640 bytes Flash/EE data memory, and 256 bytes data SRAM on-chip.

Additional MCU support functions include Watchdog Timer, Power Supply Monitor, and ADC DMA functions. 32 Programmable I/O lines plus I2C-compatible, SPI, and UART Serial Port I/O are provided for multiprocessor interfaces and I/O expansion.

Normal, idle, and power-down operating modes for both the MCU core and analog converters allow for flexible power management schemes. The part is specified for 3V and 5V operation over the industrial temperature range (-40°C to +85°C) and is available in a 52-lead, plastic quad flatpack package.

The ADC conversion block incorporates a 5µs, 12-bit, A/D converter. Multiplexer and track/hold functions are integrated into the ADC section to allow accurate sampling of up to eight external inputs plus a ninth input from the on-chip temperature sensor. Trigger sources for the ADC converter can come from an external source or from one of three software triggers from the MCU.

Two integrated 12-bit DACs provide rail-to-rail buffered analog outputs and can be individually configured for 0-to-V_{DD} or 0-to-V_{REF} output voltage range.

The reference source for the ADC and DACs can come from an external voltage reference or from the on-chip 2.5V bandgap reference.

All analog peripherals are fully configurable through the on-chip MCU via a simple SFR interface.

1.2 Driving the A/D Converter

The ADC incorporates a successive approximation (SAR) architecture involving a charge-sampled input stage. Figure 2 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 2. 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 (see Chapter 3).

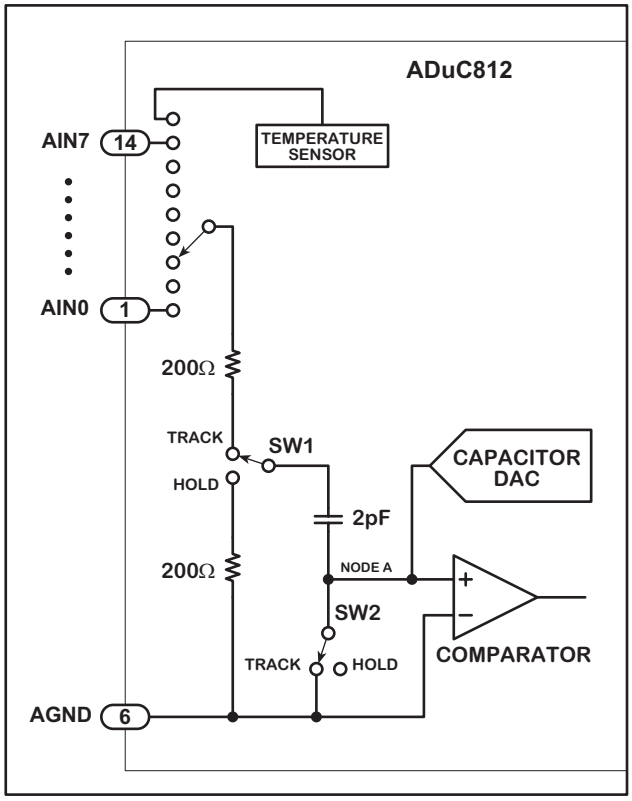


Figure 2 – Analog Input Equivalent Circuit

Note that whenever a new input channel is selected, a residual charge from the 2pF 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 300ns in order to guarantee adequate settling under all software configurations. A better solution, recommended for use with any amplifier, is shown in Figure 3.

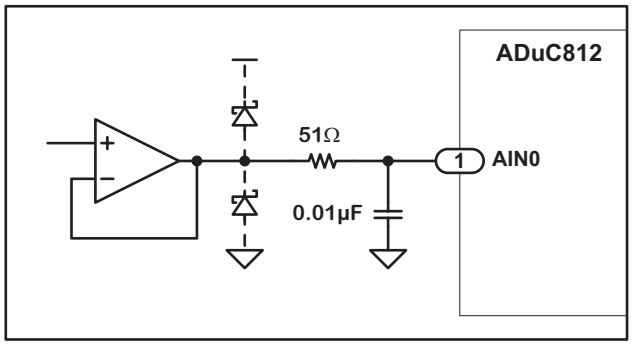


Figure 3 – Driving the Analog Inputs

Though at first glance the circuit in Figure 3 may look like a simple anti-aliasing filter, it actually serves no such purpose since its corner frequency is well above the Nyquist frequency, even at a 200kHz 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 does so by providing a capacitive bank from which the 2pF sampling capacitor can draw its charge. Since the 0.01µF capacitor in Figure 3 is more than 4096 times the size of the 2pF sampling capacitor, its voltage will not change by more than one count (1/4096) of the 12-bit transfer function when the 2pF 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 3 may be necessary to limit the voltage applied to the analog input pin as

per the datasheet absolute maximum ratings. They are not necessary if the op amp is powered from the same supply as the ADuC812 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 ADuC812's analog inputs can cause measurable DC errors with external source impedances as little as 100Ω or so. To ensure accurate ADC operation, keep the total source impedance at each analog input less than 61Ω. Table 1 illustrates examples of how source impedance can affect DC accuracy.

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

Table 1 – DC Error due to Source Impedance (assuming V_{REF} = 2.5V)

Although Figure 3 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 2. Check Analog Devices literature (CD ROM data book, etc.) for details on these and other op amps and instrumentation amps.

Op Amp Model	Characteristics
OP181/281/481	micropower
OP191/291/491	I/O good up to V_{DD}, low cost
OP196/296/496	I/O to V_{DD}, micropwr, low cost
OP183/283	high gain-bandwidth product
OP162/262/462	high GBP, micro package
AD820/822/824	FET input, low cost
AD823	FET input, high GBP

Table 2 – Some single-supply op amps

Keep in mind that the ADC's transfer function is 0 to V_{REF}, and any signal range lost to amplifier saturation near ground will impact dynamic range. Though the op amps in Table 2 are capable of delivering output signals very closely *approaching* ground, no amplifier can deliver signals all the way *to* ground when powered by a single supply. Therefore, if a negative supply is available, you might consider using it to power the front-end amplifiers. If you do, however, be sure to include the Schottky diodes shown in Figure 3 (or at least the lower of the two diodes) to protect the analog input from under-voltage conditions.

To summarize this section, use the circuit of Figure 3 to drive the analog input pins of the ADuC812. Functional details of the ADC, including timing specifics and software configuration, will be covered in Chapter 3 of this manual.

1.3 Voltage Reference Connections

The on-chip 2.5V bandgap voltage reference can be used as the reference source for the ADC and DACs. In order to ensure the accuracy of the voltage reference you must decouple both the V_{REF} pin and the C_{REF} pin to ground with 0.1µF ceramic chip capacitors as shown in Figure 4.

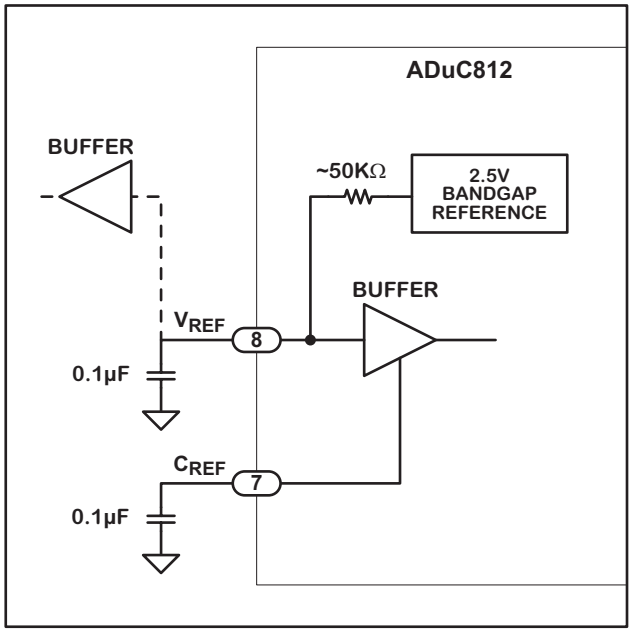


Figure 4 – Using the Internal Voltage Reference

The internal voltage reference can also be tapped directly from the V_{REF} pin, if desired, to drive external circuitry. However, a buffer must be used in this case to ensure that no current is drawn from the V_{REF} pin itself. The voltage on the C_{REF} pin is

that of an internal node within the buffer block, and its voltage is critical to ADC and DAC accuracy. Do not connect anything to this pin except the capacitor, and be sure to keep trace-lengths short on the C_{REF} capacitor, decoupling the node straight to the underlying ground plane.

The ADuC812 powers up with its internal voltage reference in the "off" state. The voltage reference turns on automatically whenever the ADC or either DAC gets enabled in software. Once enabled, the voltage reference requires approximately 65ms to power up and settle to its specified value. Be sure that your software allows this time to elapse before initiating any conversions.

If an external voltage reference is preferred, simply connect it to the V_{REF} pin as shown in Figure 5 to overdrive the internal voltage reference.

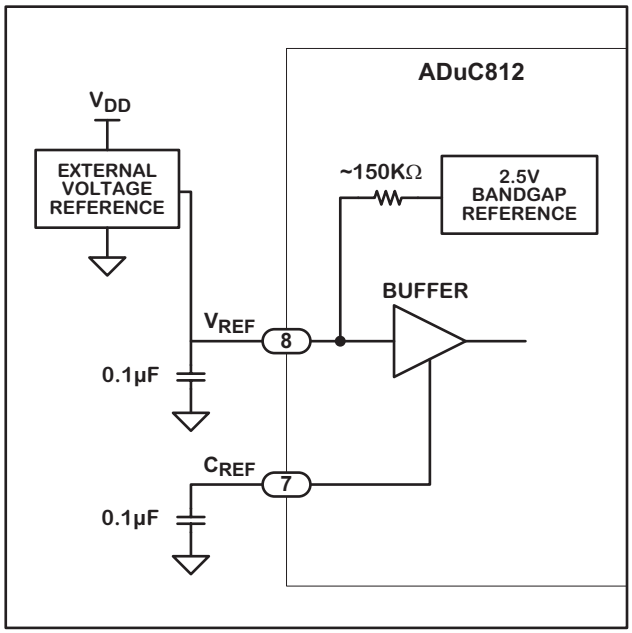


Figure 5 – Using an External Voltage Reference

To ensure accurate ADC operation, the voltage applied to V_{REF} must be between 2.3V and AV_{DD}. In situations where analog input signals are proportional to the power supply (such as some strain-gage applications) it can be desirable to connect the V_{REF} pin directly to AV_{DD}. In such a configuration you must *also* connect the C_{REF} pin directly to AV_{DD} to circumvent internal buffer headroom limitations. This allows the ADC input transfer function to accurately span the full range 0 to AV_{DD}.

Operation of the ADC or DACs with a reference voltage below 2.3V, however, may incur loss of accuracy eventually resulting in missing codes or non-monotonicity. For that reason, do not use a reference voltage less than 2.3V.

1.4 The D/A Converter Outputs

The on-chip D/A converter architecture consists of a resistor string DAC followed by an output buffer amplifier, the functional equivalent of which is illustrated in Figure 6. Details of the actual DAC architecture can be found in U.S. patent number 5969657 (www.uspto.gov). Features of this architecture include inherent guaranteed monotonicity and excellent differential linearity.

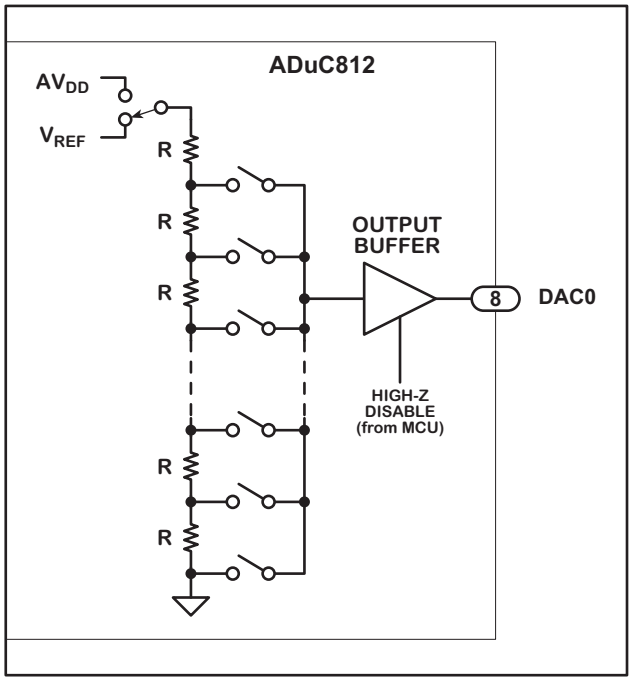


Figure 6 – Resistor String DAC Functional Equivalent

As illustrated in Figure 6, the reference source for each DAC is user selectable in software. It can be either AV_{DD} or V_{REF}. In 0-to-AV_{DD} mode, the DAC output transfer function spans from 0V to the voltage at the AV_{DD} pin. In 0-to-V_{REF} mode, the DAC output transfer function spans from 0V to the voltage at the V_{REF} pin.

The DAC output buffer amplifier features a true rail-to-rail output stage implementation. This means that, unloaded, each output is capable of swinging to

within less than 100mV of both AV_{DD} and ground. Moreover, the DAC's linearity specification (when driving a 10KΩ resistive load to ground) is guaranteed[1] through the full transfer function *except* codes 0 to 48, and, in 0-to-AV_{DD} mode only, codes 3995 to 4095. Linearity degradation near ground and V_{DD} is caused by saturation of the output amplifier, and a general representation of its effects (neglecting offset & gain error) is illustrated in Figure 7. The dotted line in Figure 7 indicates the *ideal* transfer function, and the solid line represents what the transfer function might look like with endpoint non-linearities due to saturation of the output amplifier. Note that Figure 7 represents a transfer function in 0-to-V_{DD} mode only. In 0-to-V_{REF} mode (with $V_{REF} < V_{DD}$) the lower non-linearity would be similar, but the upper portion of the transfer function would follow the "ideal" line right to the end (V_{REF} in this case, not V_{DD}), showing no signs of endpoint linearity errors.

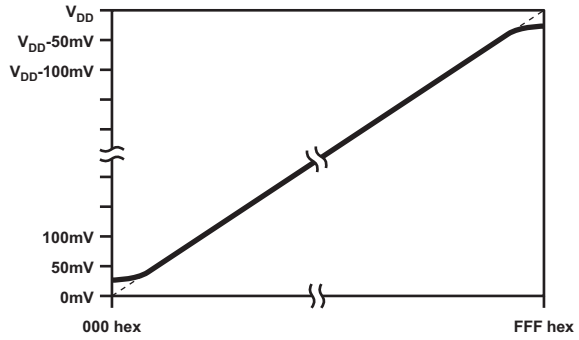


Figure 7 – Endpoint Non-linearities due to Amplifier Saturation

The endpoint non-linearities conceptually illustrated in Figure 7 get worse as a function of output loading. Most of the ADuC812's datasheet specifications assume a 10KΩ resistive load to ground at the DAC output. As the output is forced to source or sink more current, the nonlinear regions at the top or bottom (respectively) of Figure 7 become larger. With larger current demands, this can significantly limit output voltage swing. Figure 8 & Figure 9 illustrate this behavior. It should be noted that the upper trace in each of these figures is only valid for an output range selection of 0-to-AV_{DD}. In 0-to-V_{REF} mode, DAC loading will not cause high-side voltage drops as long as the reference voltage remains below the upper trace in the corresponding figure. For example, if AV_{DD}=3V & V_{REF}=2.5V, the high-side voltage will not be affected by loads less than 5mA. But somewhere around 7mA the upper curve in Figure 9 drops below 2.5V (V_{REF}) indicating that at these higher currents the output will not be capable of reaching V_{REF}.

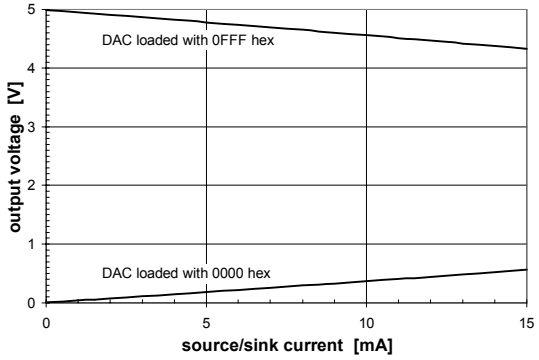


Figure 8 – Source & Sink Current Capability with $V_{REF} = V_{DD} = 5V$

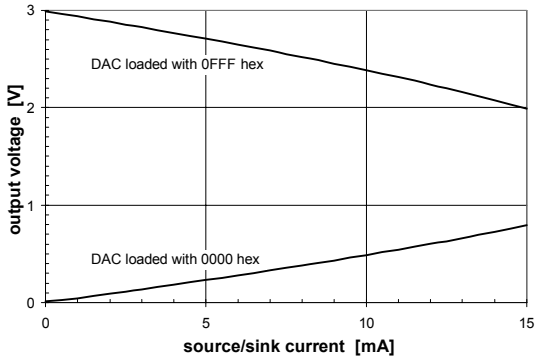


Figure 9 – Source & Sink Current Capability with $V_{REF} = V_{DD} = 3V$

The DAC output buffer also features a high-impedance disable function. In the chip's default power-on state, both DACs are disabled, and their outputs are in a high-impedance state (or "tri-state") where they remain inactive until enabled in software. This means that if a zero output is desired during power-up or power-down transient conditions, then a pull-down resistor must be added to each DAC output. Assuming this resistor is in place, the DAC outputs will remain at ground potential whenever the DAC is disabled. However, each DAC output will still spike briefly when you first apply power to the chip, and again when each DAC is first enabled in

[1] Refer to ADuC812 datasheet.

software. Typical scope shots of these spikes are given in Figure 10 and Figure 11 respectively.

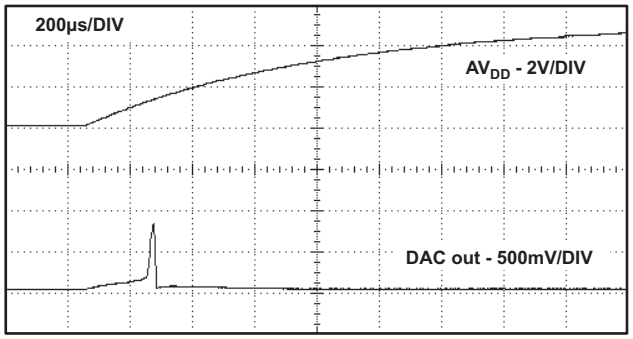


Figure 10 – DAC Output Spike at Chip Power-Up

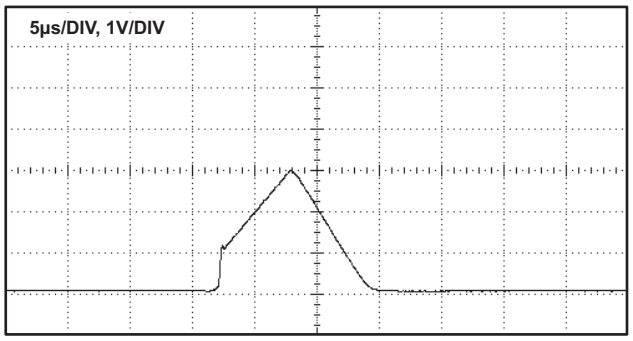


Figure 11 – DAC Output Spike at DAC Enable

1.5 Clock Oscillator

The clock source for the ADuC812 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 pins 32 & 33, and connect a capacitor from each pin to ground, as shown in Figure 12. The value of the capacitors should be the value recommended by the crystal manufacturer for use with that specific crystal.

To use an external clock source to drive the ADuC812, simply connect the clock source to pin 32 (XTAL1) and leave pin 33 (XTAL2) open. The logic levels required at the XTAL1 input are specified in the product datasheet under "Digital Inputs".

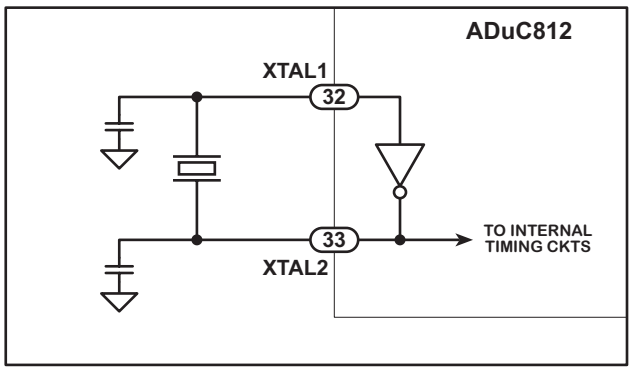


Figure 12 – Using a Parallel Resonant Crystal

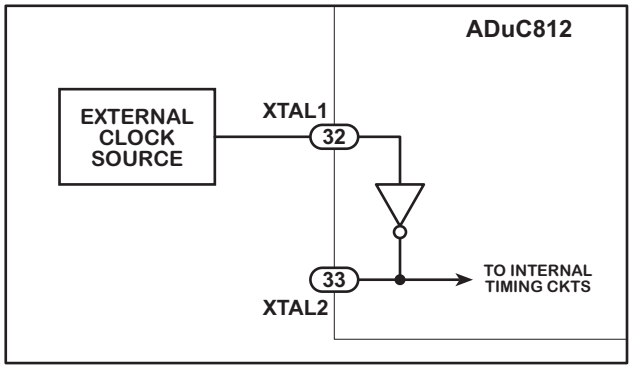


Figure 13 – Connecting an External Clock Source

Whether using the internal oscillator or an external clock source, the ADuC812's specified operational clock speed range is 400kHz to 16MHz. The core itself is static, and will function all the way down to DC. But at clock speeds slower than 400kHz, the ADC will no longer function correctly. Therefore, to ensure specified operation, use a clock frequency of at least 400kHz, and no more than 16MHz.

1.6 External Memory Interface

In addition to its internal program and data memories, the ADuC812 can access up to 64K bytes of external program memory (ROM/PROM/etc.) and up to 16M bytes 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* 8K 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. In

either case, addresses above 1FFF hex (8K) are mapped to the external space, as shown in Figure 14.

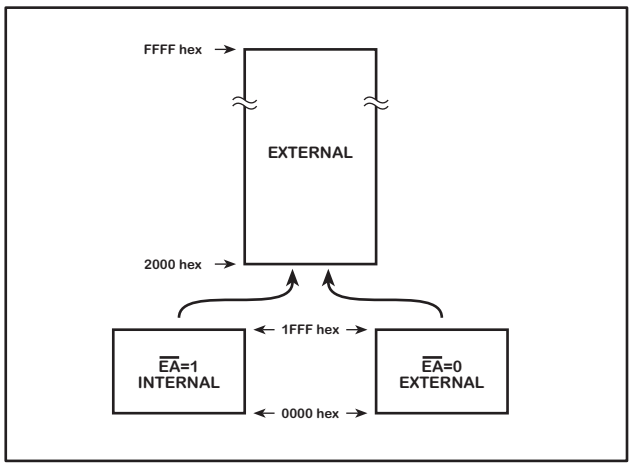


Figure 14 – Program Memory Selection Internal / External

Note that a second very important function of the $\overline{EA}$ pin is described in the section on "Other Hardware Considerations" near the end of this chapter.

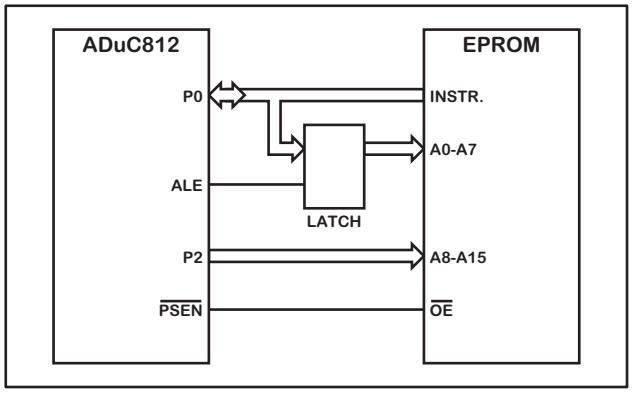


Figure 15 – External Program Memory Interface

External program memory (if used) must be connected to the ADuC812 as illustrated in Figure 15. Note that 16 I/O lines (Ports 0 & 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 $\overline{PSEN}$ strobes the EPROM and the code byte is read into the ADuC812. A detailed diagram of external program memory read sequence timing is given in the ADuC812 product datasheet.

Note that program memory addresses are always 16 bits wide, even in cases where the actual amount of program memory used is less than 64K bytes. 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.

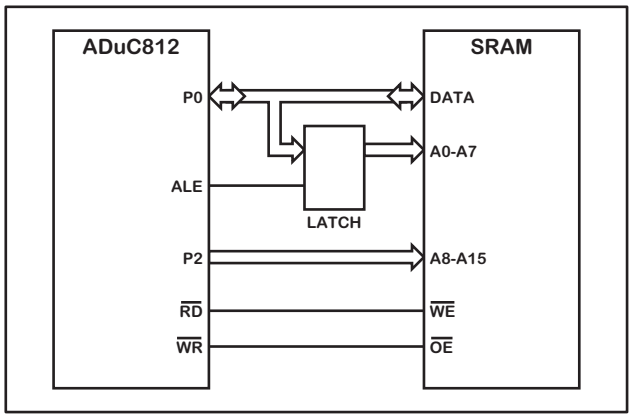


Figure 16 – External Data Memory Interface for up to 64K Addressing

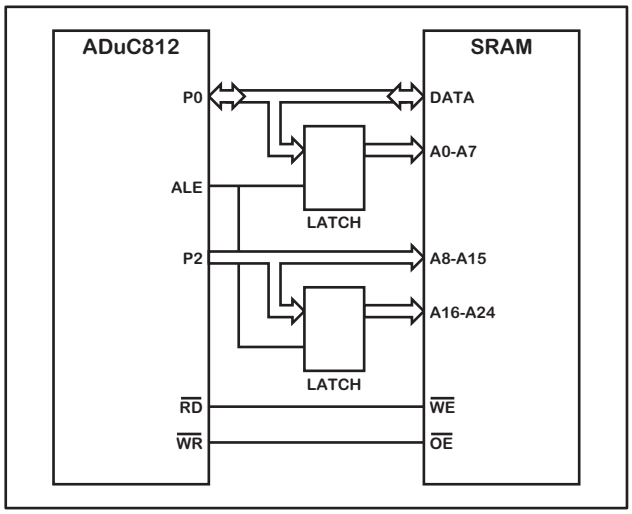


Figure 17 – External Data Memory Interface for up to 16Meg Addressing

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 16 shows a hardware configuration for accessing up to 64K bytes of external RAM. This interface is standard to any 8051 compatible MCU. If access to more than 64K bytes of RAM is desired, a feature unique to the ADuC812 allows addressing up to 16M bytes of external RAM by simply adding a latch as illustrated in Figure 17.

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 gets latched by a pulse of ALE prior to data being placed on the bus by the ADuC812 (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, then DPP is ignored by the SRAM, and the 8051 standard of 64K byte external data memory access is maintained.

Detailed timing diagrams of external program and data memory read and write access can be found in the ADuC812 product datasheet.

1.7 I/O Ports

Each of the four I/O ports on the ADuC812 features a different implementation of drive/receive circuitry. All ports except Port 1 are bi-directional, featuring standard 8051/8052 functionality. The operation of Port 1 is unique to the ADuC812.

As described briefly in the previous section, Ports 0 and 2 are used to access external memory. When not accessing external memory, Ports 0 and 2 are available for general-purpose input and output functions, controlled by the special function registers P0 and P2. The different types of external memory accesses disrupt the standard I/O functions of Ports 0 and 2 in different ways. Specifically:

- "MOVX @Ri" instructions require only an 8-bit address bus, so only Port 0 is used. The P2 SFR goes untouched, and the Port 2 pins retain their general-purpose I/O states. The contents of the P0 SFR, however, are annihilated and replaced by the power-on default value of FF hex.
- "MOVX @DPTR" or "MOVC" instructions use both Port 0 and Port 2. Just as above, the contents of the P0 SFR are annihilated and replaced with FF hex. The contents of the P2 SFR, however, are retained, and the Port 2 pins will return to the state defined by P2 as soon as the instruction is complete (assuming that the next instruction cycle doesn't access external memory).
- Execution from external program memory uses both Port 0 and Port 2. Again, the contents of the P0 SFR get forced to FF hex. Again the contents of the P2 SFR are retained, and Port 2 pins will return to the state defined by P2 upon the first machine cycle that doesn't access external memory.

All the Port 3 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed in Table 3.

The alternate functions of Port 3 pins can only be activated if the corresponding bit latch in the P3 SFR contains a 1. Otherwise, the port pin is stuck at 0.

Pin	Alternate Function
P3.0	RxD (UART input port) (or serial data I/O in mode 0)
P3.1	TxD (UART output port) (or serial clock output in mode 0)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1) MISO (SPI port "master-in / slave-out")*
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input) $\overline{\text{CONVST}}$ (ADC hardware convert-start)*
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

* The MISO and $\overline{\text{CONVST}}$ functions are specific to the ADuC812. All other functions are standard to the 8052.

Table 3 – Port 3 Alternate Pin Functions

Port 3 pins are tested for greater sink current (8mA) than other port pins, allowing direct drive of LED or optocoupler devices. However, avoid sinking this full current into more than a couple of pins at a time during ADC conversions to prevent performance degradation.

The primary function of Port 1 pins on the ADuC812 is as analog inputs. Writing ones to the P1 SFR bits (the power-on default condition) enables the Port 1 pins for analog input mode. Any Port 1 pins not used for analog inputs can alternatively be used for digital inputs by writing zeros to the corresponding P1 SFR bits. Port 1 pins have no output drivers and therefore cannot be used as digital outputs.

Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 external input)
P1.1	T2EX (Timer/Counter 2 capture/reload trigger)
P1.5	$\overline{SS}$ (SPI port "slave-select")*

* The $\overline{SS}$ function is specific to the ADuC812. The other two functions are standard to the 8052.

Table 4 – Port 1 Alternate Pin Functions

Three Port 1 pins are multifunctional. In addition to providing analog or digital inputs, they serve the special functions listed in Table 4.

Bit Latches & I/O Buffers

Figure 18 through Figure 22 show a typical bit latch and I/O buffer in each of the four ports. 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. More about that in Chapter 2.

As shown in Figure 18 and Figure 19, the output drivers of Ports 0 and 2 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 P2 SFR remains unchanged, but the P0 SFR gets 1s written to it (i.e. all of its bit latches become 1).

In general purpose I/O port mode, Port 0 pins feature "open-drain" style output drivers. This is represented in Figure 18 by the NAND gate who's 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. When accessing external memory, however, the CONTROL signal in Figure 18 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.

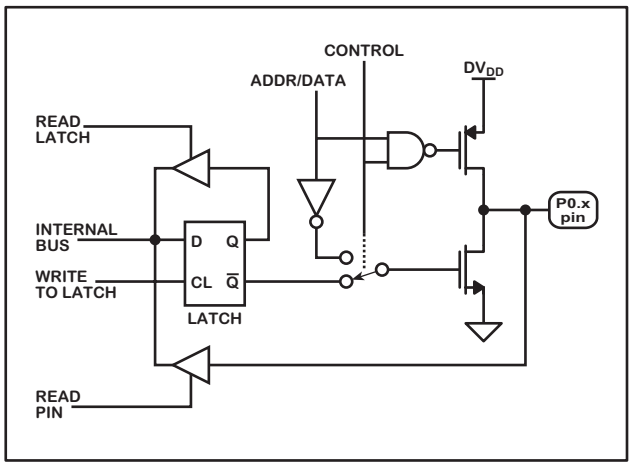
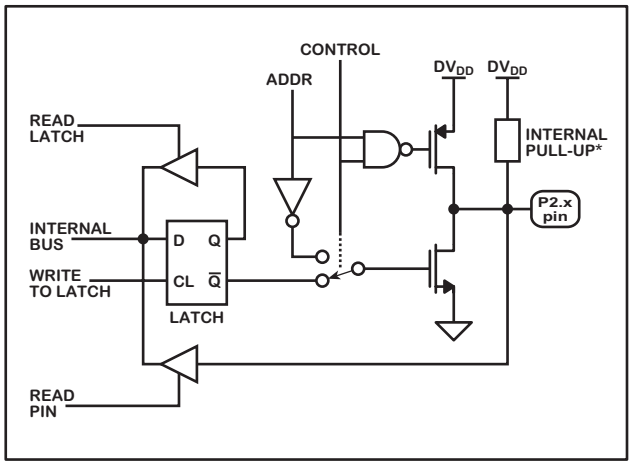


Figure 18 – Port 0 Bit Latch & I/O Buffer

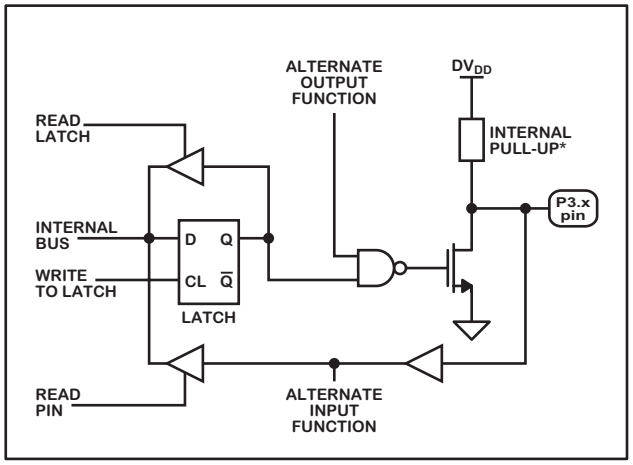


*See Figure 21 for details of internal pull-up.

Figure 19 – Port 2 Bit Latch & I/O Buffer

Similarly, Port 2 has two modes of operation as dictated by the CONTROL signal from the core. In normal mode (CONTROL=0) the top FET is disabled, but in external memory addressing mode (CONTROL=1) the port pins feature push/pull operation controlled by the internal address bus (ADDR line).

Unlike Port 0, however, both Port 2 and Port 3 feature an internal pull-up function. The internal pull-up (represented as a single block in Figure 19 and Figure 20) actually consists of active circuitry as shown in Figure 21.



*See Figure 21 for details of internal pull-up.

Figure 20 – Port 3
Bit Latch & I/O Buffer

Whenever a Port 2 or Port 3 bit latch transitions from low to high, Q1 in Figure 21 turns on for 2 oscillator periods to quickly pull the pin to a logic high state. Once there, the weaker Q3 turns on, thereby latching the pin to a logic high. If the pin is momentarily pulled low externally, Q3 will turn off, but the very weak Q2 will continue to source some current into the pin, attempting to restore it to a logic high.

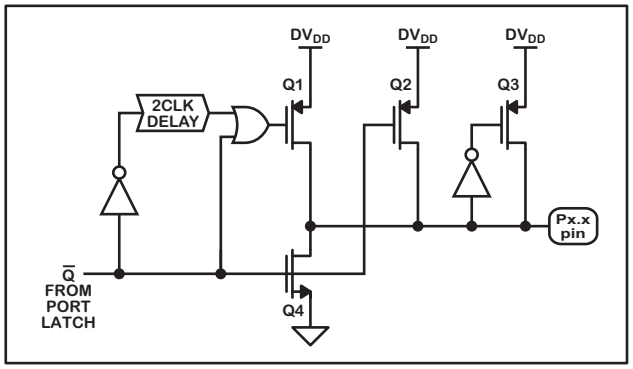


Figure 21 – Internal Pull-Up Configuration

One exception to the Figure 20 representation of Port 3 pins applies to the P3.3 pin only, which doubles as the MISO pin in SPI mode. Whenever SPI mode is enabled, the SPI hardware takes full control of P3.3. Software write operations to the P3.3 bit will then cause no changes on the pin, nor will reads from P3.3 return the correct state of the pin.

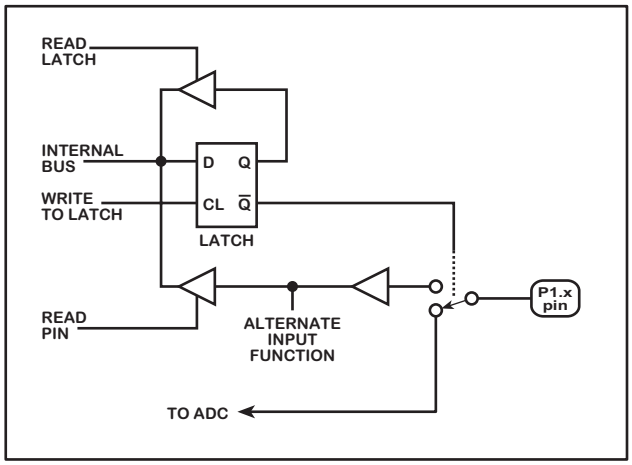


Figure 22 – Port 1
Bit Latch & I/O Buffer

As mentioned previously, Port 1 of the ADuC812 is primarily used for analog inputs. Any Port 1 pins *not* used as analog inputs can be configured as digital inputs by writing zeros to the corresponding SFR bit latches. Figure 22 illustrates this function. Note that there are no output drivers for Port 1 pins, and they therefore cannot be used as outputs.

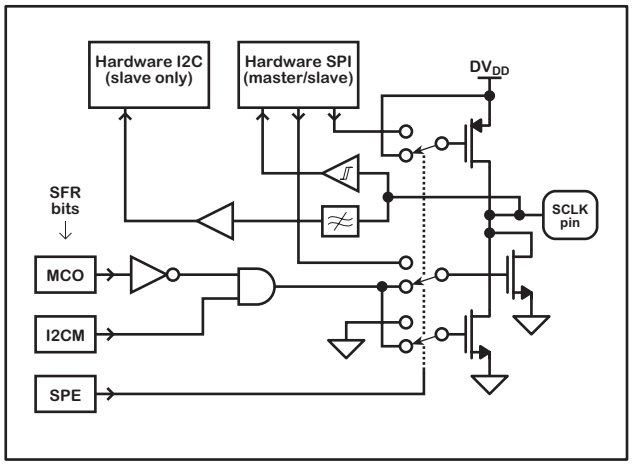


Figure 23 – SCLOCK pin I/O
functional equivalent

In addition to the port pins, the dedicated SPI/I^2C pins (SCLOCK and SDATA/MOSI) also feature both input and output functions. Their equivalent I/O architectures are illustrated in Figure 23 and Figure 24 respectively. The blocks at the left of these diagrams represent SFR bits in the SPI and I^2C

control registers (SPICON & I2CCON). Notice that in I^2C mode (SPE=0) the upper FET in each diagram is held in its "off" state, thereby giving both pins "open drain" type drivers with no internal pull-up, as per standard I^2C. By contrast, in SPI mode (SPE=1) the upper FET is controlled directly by SPI hardware, giving the pin push/pull capability. Also, in I^2C mode (SPE=0) two pull-down FETs operate in parallel in order to provide an extra 60% or 70% of current sinking capability. In SPI mode, however, (SPE=1) the bottom FET in each diagram is disabled and only one pull-down FET operates on each pin, resulting in sink capabilities identical to that of port 0 and port 2 pins.

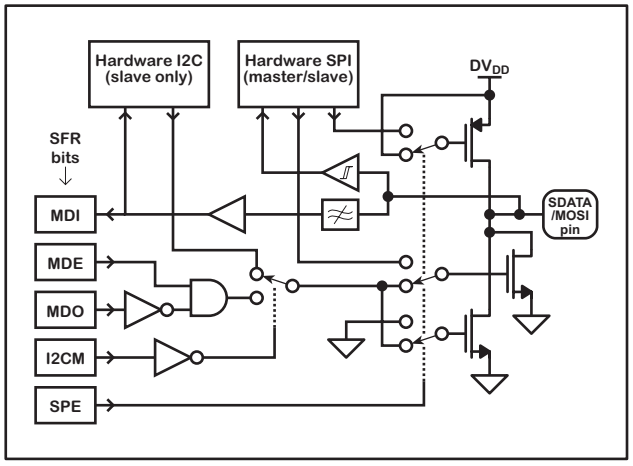


Figure 24 – SDATA/MOSI pin I/O functional equivalent

On the input path of SCLOCK and SDATA/MOSI, notice that a Schmitt trigger conditions the signal going to the SPI hardware to prevent false triggers (double triggers) on slow incoming edges. For incoming signals from these pins going to I^2C hardware, a filter conditions the signals in order to reject glitches of up to 50ns in duration.

The functions of the "Hardware I2C" and "Hardware SPI" blocks of the above diagrams will be described in Chapter 3 of this manual. However, note that direct access to the SCLOCK and SDATA/MOSI pins is afforded through the SFR interface in I^2C master mode. Therefore, if you are not using the SPI or I^2C functions, you can use these two pins for general purpose I/0. The SDATA/MOSI pin can be used for both input and output, while the SCLOCK pin can only be used as an output in this mode.

1.8 Reset & POR (Power-On Reset)

You must implement external POR (power-on reset) circuitry to drive the RESET pin of the ADuC812. Your circuit must hold the RESET pin asserted (high) whenever the power supply (AV_{DD} & DV_{DD}) is below 2.5V. Furthermore, V_{DD} must remain above 2.5V for at least 10ms before the RESET signal is de-asserted (low). The external POR circuit must be operational down to 1.2 volts or less. The timing diagram of Figure 25 illustrates this functionality under three separate events: power-up, brownout, and power-down. Notice that when RESET is asserted (high) it tracks the voltage on V_{DD}.

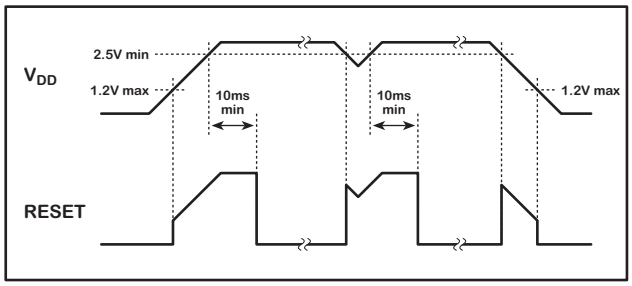


Figure 25 – POR timing requirements

The best way to implement an external POR function to meet the above requirements involves the use of a dedicated POR chip, such as the ADM181x family of SOT-23 packaged PORs from Analog Devices. Recommended connection diagrams for both active-high and active-low type PORs are shown in Figure 26 and Figure 27 respectively. The POR chip's output can also be used to provide reset signals to additional circuitry on your board that may need a POR, but be sure to observe the POR chip's output loading limitations.

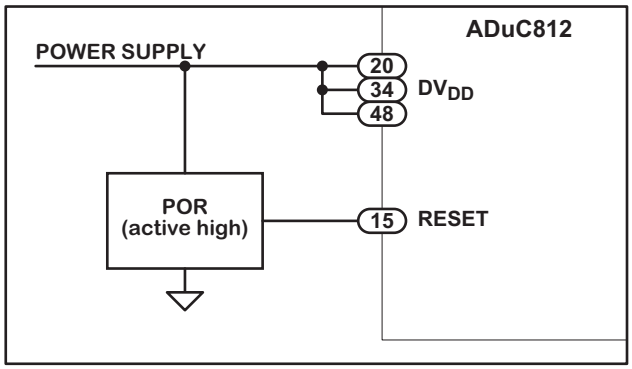


Figure 26 – Recommended POR Circuit for Active-High POR chip

Some active-low POR chips, such as the ADM1813 & ADM1818, can be used with a manual pushbutton as an additional reset source as illustrated by the dashed line connection in Figure 27.

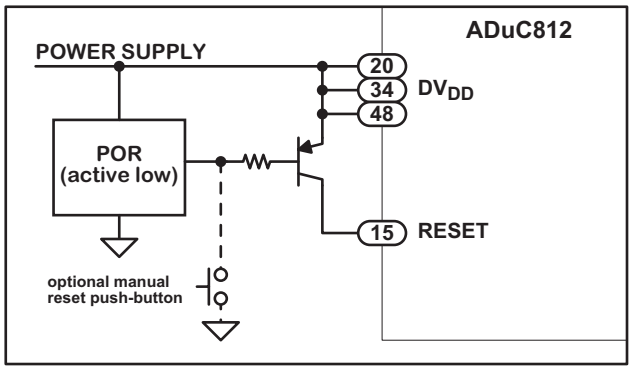


Figure 27 – Recommended POR Circuit for Active-Low POR chip

The recommended POR implementations of Figure 26 or Figure 27 will ensure correct startup and shutdown under virtually any power-cycling scenarios, including incomplete shutdown and brown-out conditions.

1.9 Power Supplies

The ADuC812's operational power supply voltage range is 2.7V to 5.5V. Though the guaranteed datasheet specifications are given only for power supplies within ±10% of nominal 3V or 5V levels, the chip will function equally well at any power supply level between 2.7V and 5.5V.

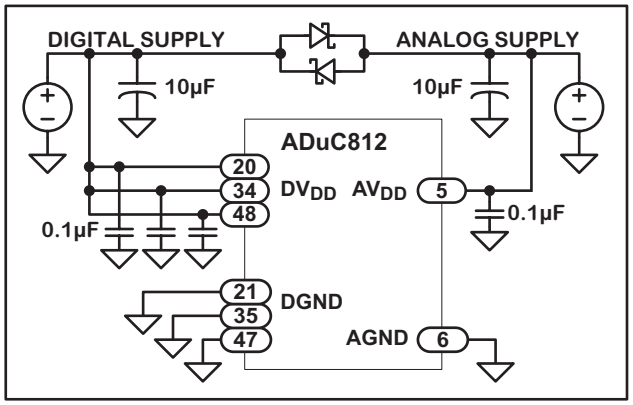


Figure 28 – Powering from Separate Analog & Digital Supplies

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 ±0.3V of one another at all times in order to avoid damaging the chip (as per the absolute maximum ratings in the product datasheet). 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 28.

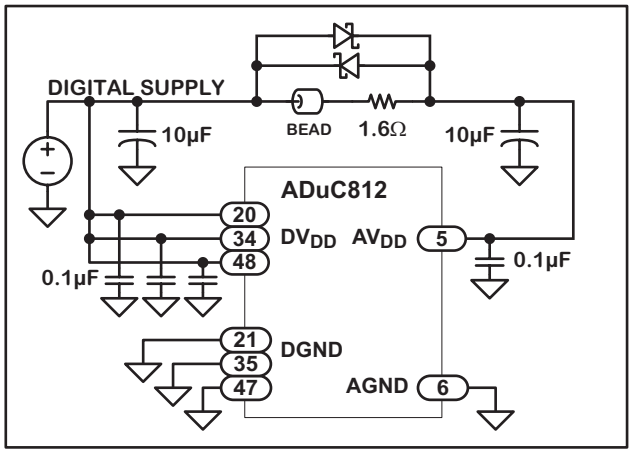


Figure 29 – "Filtering" the Digital Supply to Generate Quiet AV_{DD}

As an alternative to providing two separate power supplies, you can help keep AV_{DD} quiet by placing a small series resistor and/or ferrite bead between it and DV_{DD}, and then decoupling AV_{DD} separately to ground. An example of this configuration is shown in Figure 29. Do not, however, use a series inductor between AV_{DD} and DV_{DD} to avoid forming a tuned circuit with the decoupling capacitance on the AV_{DD} line. Also, avoid choosing a resistor value greater than a few ohms to prevent a significant increase in the impedance at AV_{DD}. With this configuration (Figure 29) you can of course power other analog circuitry (such as op amps, voltage reference, etc.) from the AV_{DD} supply line as well. You will still want to include back-to-back Schottky diodes between AV_{DD} and DV_{DD} in order to protect from power-up and power-down transient conditions that could separate the two supply voltages momentarily.

Notice that in both Figure 28 and Figure 29, a large value reservoir capacitor sits on DV_{DD} and a separate one sits on AV_{DD}. Also, local small value ceramic chip caps are located at each V_{DD} pin of the chip. As

per standard design practice, be sure to include all of these capacitors, and keep the ceramic chip caps very close to each AV_{DD} pin with trace lengths as short as possible. Connect the ground terminal of each of these caps directly to the underlying ground plane. More on ground planes in the following section.

Power Consumption

The currents consumed by the various sections of the ADuC812 are shown in Table 5. The "CORE" values given represent the current drawn by DV_{DD}, while the rest ("ADC", "DAC", "voltage ref") are pulled by the AV_{DD} pin and can be disabled in software when not in use. The other on-chip peripherals (watchdog timer, power supply monitor, etc.) consume negligible current and are therefore lumped in with the "CORE" operating current here. Of course, you must add any currents sourced by the parallel and serial I/O pins, and that sourced by the DAC, in order to determine the *total* current needed at the ADuC812's supply pins. Also, current draw from the DV_{DD} supply will increase by approximately 10mA during Flash/EE erase and program cycles.

	V_{DD} = 5V	V_{DD} = 3V
CORE: (normal mode)	1.6nAs·M_{CLK} + 5mA	0.8nAs·M_{CLK} + 1.5mA
CORE: (idle mode)	0.75nAs·M_{CLK} + 5mA	0.25nAs·M_{CLK} + 1.5mA
ADC:	1.3mA	1.0mA
DAC (each)**:**	250µA	200µA
voltage ref:	200µA	150µA

Table 5 – Typical I_{DD} of Core and Peripherals

Since operating DV_{DD} current is primarily a function of clock speed, the expressions for "CORE" supply current in Table 5 are given as functions of M_{CLK}, the oscillator frequency. Plug in a value for M_{CLK} in Hertz to determine the current consumed by the core at that oscillator frequency. Since the ADC and DACs can be enabled or disabled in software, add only the currents from the peripherals you expect to use. The internal voltage reference is automatically enabled whenever either the ADC or at least one DAC is enabled. And again, don't forget to include current sourced by I/O pins, serial port pins, DAC outputs, etc., plus the additional current drawn during Flash/EE erase and program cycles.

A software switch allows the chip to be switched from *normal mode* into *idle mode*, and also into full *powerdown mode*. Details of these modes will be given in Chapter 2 of this manual, but below are brief descriptions of powerdown and idle modes.

In *idle mode*, the oscillator continues to run, but is gated off to the core only. The on-chip peripherals continue to receive the clock, and remain functional. Port pins and DAC output pins retain their states in this mode. The chip will recover from idle mode upon receiving any enabled interrupt, or on receiving a hardware reset.

In full *powerdown mode*, the on-chip oscillator stops, and all on-chip peripherals are shut down. Port pins retain their logic levels in this mode, but the DAC output goes to a high-impedance state (tri-state). The chip will only recover from powerdown mode upon receiving a hardware reset or when power is cycled. During full powerdown mode, the ADuC812 consumes a total of approximately 5µA.

1.10 Grounding, Board Layout, Etc.

As with all high resolution data converters, special attention must be paid to grounding and PC board layout of ADuC812 based designs in order to achieve optimum performance from the ADC and DACs. Though each system is unique, attention to some common guidelines can be helpful.

Although the ADuC812 has separate pins for analog and digital ground (AGND & DGND), you must *not* tie these to two separate ground planes *unless the two ground planes are connected together very close to the ADuC812*, as illustrated in the simplified example of Figure 30 Diagram A. 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 ADuC812 since a ground loop would result. In these cases, tie the ADuC812's AGND and DGND pins *all* to the *analog* ground plane, as illustrated in Figure 30 Diagram B. In systems with only one ground plane, ensure that you physically separate digital and analog components onto separate halves of the board such that digital return

currents don't flow near analog circuitry and vice versa. The ADuC812 can then be placed *between* the digital and analog sections, as illustrated in Figure 30 Diagram C.

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 get to their destinations. For example, do not power components on the analog side of Figure 30 Diagram B 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, such as would happen if you placed a noisy digital chip on the left half of Board C in Figure 30. 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.

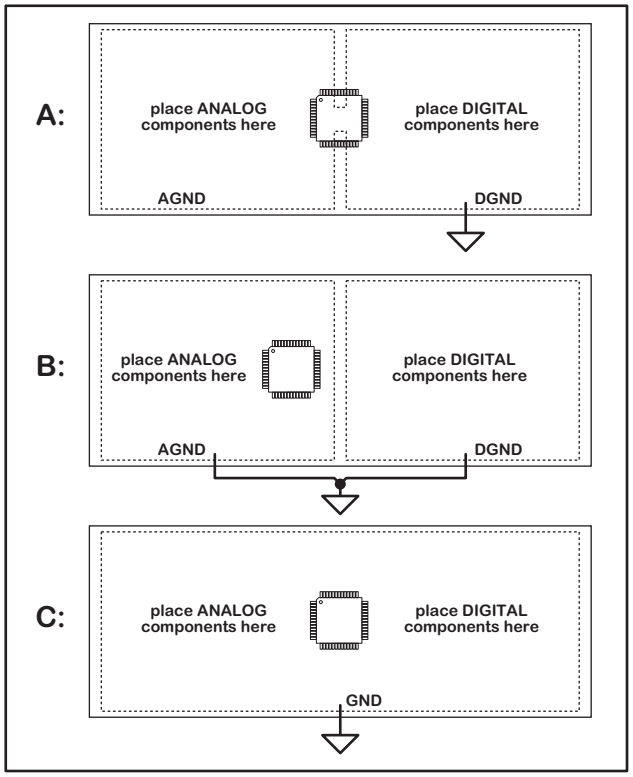


Figure 30 – Three Different Grounding Schemes

If you plan to connect fast logic signals (rise/fall time < 5ns) to any of the ADuC812's digital inputs, then add a series resistor to each relevant line to keep rise and fall times longer than 5ns at the ADuC812 input pins. A value of 100 or 200 ohms is usually sufficient to prevent high-speed signals from coupling capacitively into the ADuC812 and affecting the accuracy of ADC conversions.

1.11 Other Hardware Considerations

To facilitate in-circuit programming, plus in-circuit debug and emulation options, you'll want to implement some simple connection points in your hardware that will allow easy access to download, debug, and emulation modes. Details of these modes will be given in Chapter 4 of this manual, but first let's take a quick look at some ways that you can tailor your hardware design to allow easy entry and access to them.

Figure 31 shows an example connection diagram of a typical end circuit implementation. Included is all

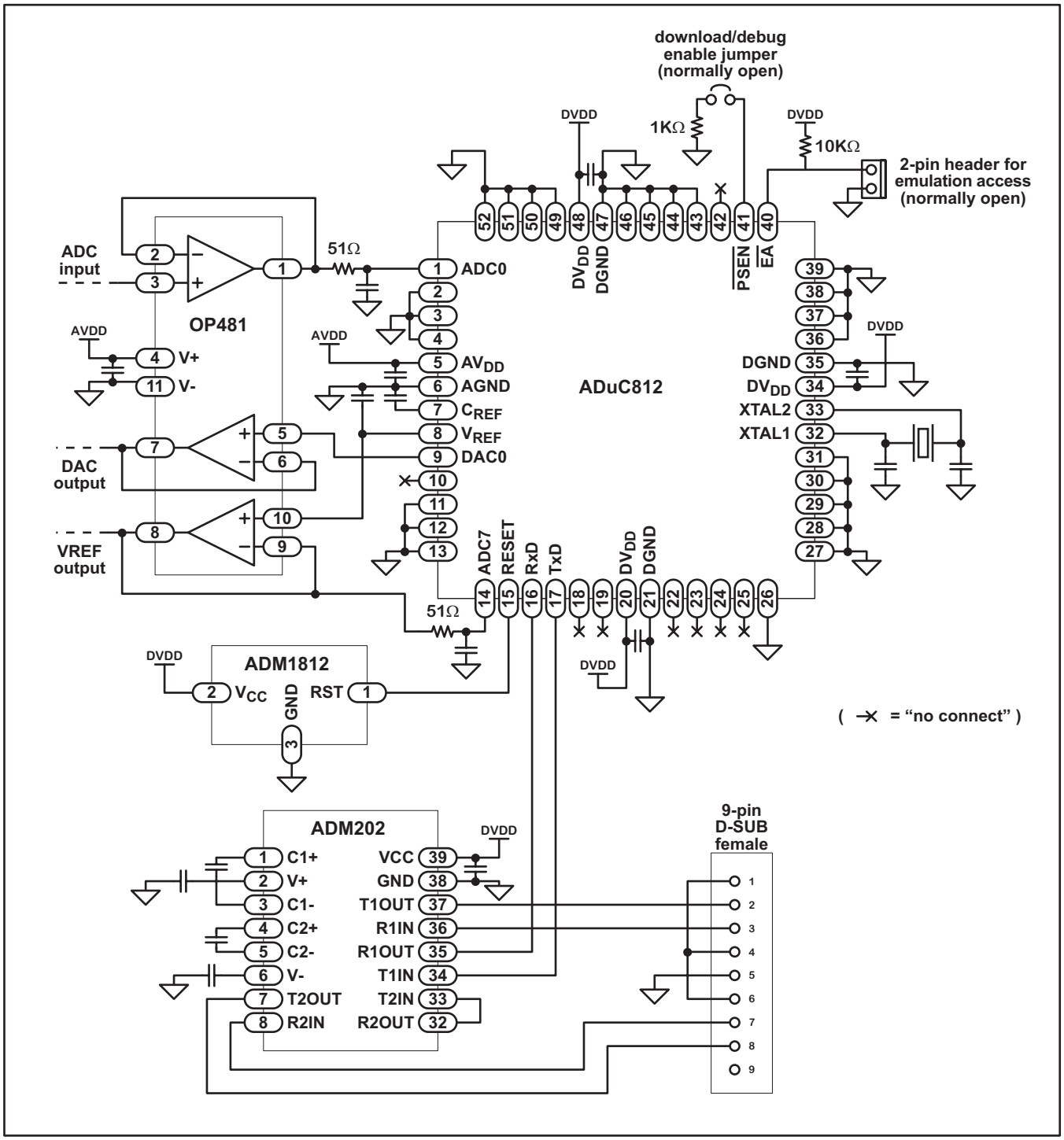


Figure 31 – A Typical Connection Diagram

the hardware needed for access to all of the below operating modes, plus all the basic hardware needed to operate the chip (crystal, POR, etc), plus a quad op amp for buffering ADC and DAC signals. Notice also that an op amp is used to buffer the V_{REF} signal and its output is fed back to an ADC input. This, along with another ADC input connected to ground, can be used for ADC endpoint calibration as will be discussed in Chapter 3 of this manual, and in application note uC005[2] on the web.

The power supply is omitted from Figure 31 for simplicity.

In-Circuit Serial Download Access

Nearly all ADuC812 designs will want to take advantage of the in-circuit re-programmability of the chip. This is accomplished by a connection to the ADuC812'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 31 with a simple ADM202-based circuit. If you would rather not design an RS-232 chip onto your board, then refer to the application note "uC006 – A 4-Wire UART-to-PC interface"[3] for a simple (and zero-cost-per-board) method of gaining in-circuit serial download access to the ADuC812.

In addition to the basic UART connections, you'll also need a way to trigger the chip into download mode. This is accomplished via a 1K pull-down resistor that can be jumpered onto the $\overline{PSEN}$ pin, as shown in Figure 31. To get the chip into download mode, simply connect this jumper and power-cycle the chip (or manually reset it if you have a manual reset button) and it will then be ready to receive a new program serially. With the $\overline{PSEN}$ jumper removed, the chip will come up in normal mode (and run your program) whenever power is cycled or RESET is toggled.

Note that $\overline{PSEN}$ is normally an output (as described in the "External Memory Interface" section) and it is sampled as an input *only* on the falling edge of RESET (i.e. at power-up or upon a manual reset). Note also that if any external circuitry unintentionally pulls $\overline{PSEN}$ low during power-up or reset events, then 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 $\overline{PSEN}$ low, except for the jumper itself.

Embedded Serial Port Debugger

From a hardware perspective, serial port debug access is 100% identical to serial download access. In fact, later chapters will refer to "download/debug mode" since it 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 ADuC812 chip, (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 ADuC812 is a dedicated controller for single-pin in-circuit emulation (ICE) using production chips. This emulation access is gained by connection to a single pin, the $\overline{EA}$ pin. Normally, you'd hard-wire this pin either high or low to select execution from internal or external program memory space (as described in the "External Memory Interface" section). To enable single-pin emulation mode, however, you'll need to pull the $\overline{EA}$ pin high through a 10K resistor as shown in Figure 31. The emulator will then connect to the 2-pin header also shown in Figure 31. 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 31, 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).

[2] Application note uC005 is available at www.analog.com/microconverter/technotes_code.html

[3] Application note uC006 is available at www.analog.com/microconverter/technotes_code.html

Note that single-pin emulation is only available when executing from *internal* code space ($\overline{EA}$ high). Note also that in normal operation, a change of state on the $\overline{EA}$ pin will immediately trigger the chip into emulation mode, effectively halting the processor. You therefore must ensure that the $\overline{EA}$ pin remains constantly in one state during normal chip operation. In a very noisy environment, this might mean replacing the 10K pull-up with a very small resistance or direct connection to DV_{DD} in production units. The 10K resistor can then be substituted back in on individual units when emulation access is desired.

Enhanced-Hooks Emulation Mode

In addition to single-pin emulation mode, the ADuC812 also supports enhanced-hooks emulation mode. An enhanced-hooks based emulator is available from Metalink Corporation (www.metaice.com). No special hardware support for these emulators needs to be designed onto your board since these are "pod style" emulators where you must replace the chip on your board with a header device that the emulator pod plugs into. The only hardware concern is then one of determining if adequate space is available for the emulator pod to fit into your system enclosure.

Once again, all of the above modes will be documented in detail in Chapter 4 of this manual.

1.12 Pin Functions

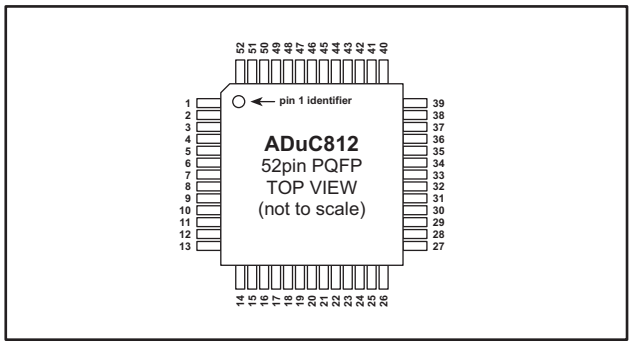


Figure 32 – Pin Diagram

The ADuC812 is packaged in a 52-pin PQFP. Many of the pins have multiple functions. Figure 32 shows the pin-numbering scheme of the package, and Table 6 lists the name(s) of each pin, in order of pin number. Following that is a description of each pin, sorted by function.

1	P1.0 / ADC0 / T2	27	SDATA / MOSI
2	P1.1 / ADC1 / T2EX	28	P2.0 / A8 / A16
3	P1.2 / ADC2	29	P2.1 / A9 / A17
4	P1.3 / ADC3	30	P2.2 / A10 / A18
5	AV_{DD}	31	P2.3 / A11 / A19
6	AGND	32	XTAL1 (in)
7	C_{REF}	33	XTAL2 (out)
8	V_{REF}	34	DV_{DD}
9	DAC0	35	DGND
10	DAC1	36	P2.4 / A12 / A20
11	P1.4 / ADC4	37	P2.5 / A13 / A21
12	P1.5 / ADC5 / $\overline{SS}$	38	P2.6 / A14 / A22
13	P1.6 / ADC6	39	P2.7 / A15 / A23
14	P1.7 / ADC7	40	$\overline{EA}$
15	RESET	41	$\overline{PSEN}$
16	P3.0 / RxD	42	ALE
17	P3.1 / TxD	43	P0.0 / AD0
18	P3.2 / $\overline{INT0}$	44	P0.1 / AD1
19	P3.3 / $\overline{INT1}$ / MISO	45	P0.2 / AD2
20	DV_{DD}	46	P0.3 / AD3
21	DGND	47	DGND
22	P3.4 / T0	48	DV_{DD}
23	P3.5 / T1 / $\overline{CONVST}$	49	P0.4 / AD4
24	P3.6 / $\overline{WR}$	50	P0.5 / AD5
25	P3.7 / $\overline{RD}$	51	P0.6 / AD6
26	SCLOCK	52	P0.7 / AD7

Table 6 – Pin Configuration

DV_{DD} *(power supply)*
Digital supply voltage. Connect to +3V or +5V power supply.

AV_{DD} *(power supply)*
Analog supply voltage. Connect to +3V or +5V power supply.

DGND *(ground)*
Digital ground. Ground reference point for digital circuitry.

AGND *(ground)*
Analog ground. Ground reference point for analog circuitry.

XTAL1 *(input)*
Crystal input. Input to the inverting oscillator amplifier.

XTAL2 *(output)*
Crystal output. Output of the inverting oscillator amplifier.

$\overline{\text{EA}}$ *(input)*
External access enable. Hard-wire this pin (through a resistor if desired) to DGND or DV_{DD}. Connection to DV_{DD} configures the chip to fetch code from internal program memory addresses 0000hex to 1FFFhex, and from external memory for all other addresses. Connection to ground configures the chip to fetch all instructions from external program memory.

ALE *(output)*
Address latch enable. Used to trigger an external latch for use in accessing external code and data memory. Leave open if unused.

$\overline{\text{PSEN}}$ *(output)*
Program strobe enable. A logic output used to access external program memory. This pin can also be used to enable serial download/debug mode when pulled low through a resistor on power-up or reset. Leave open for normal operation if unused.

RESET *(input)*
Active-high reset input. Assert logic-high for at least 24 master oscillator cycles to reset the chip.

P0.0–P0.7 *(I/O)*
Port 0 pins. Can be used as general-purpose digital inputs or outputs. Pull to ground if unused.

P1.0–P1.7 *(input)*
Port 1 pins. Can be used as analog or digital inputs only, not as digital outputs. Pull to ground if unused.

P2.0–P2.7 *(I/O)*
Port 2 pins. Can be used as general-purpose digital inputs or outputs. Pull to ground if unused.

P3.0–P3.7 *(I/O)*
Port 3 pins. Can be used as general-purpose digital inputs or outputs. Leave open if unused.

C_{REF}
Decoupling pin for on-chip voltage reference. Connect 0.1µF between this pin and AGND. Do not make any other connections to this pin.

V_{REF} *(I/O)*
Voltage reference input/output. The internal reference voltage is available on this pin, or you can feed an external voltage reference to this pin. Connect a 0.1µF capacitor between this pin and AGND.

DAC0 *(output)*
Voltage output from DAC 0. Leave open if unused.

DAC1 *(output)*
Voltage output from DAC 1. Leave open if unused.

AD0–AD7 *(I/O)*
Address/data bus (alternate function of Port 2). Bus is time-multiplexed between data byte and low-order address byte during access to external program and data memories.

A8–A15 / A16–A23 *(output)*
High-order address bus (alternate function of Port 0). Bus outputs the address high-byte during access to external program memory. Bus is time-multiplexed between the address high-byte and address page-byte during access to external data memory.

$\overline{\text{WR}}$ *(output)*
Write control signal (alternate function of P3.6). Used for write access to external data memory.

$\overline{\text{RD}}$ *(output)*
Read control signal (alternate function of P3.7). Used for read access to external data memory.

ADC0–ADC7 *(inputs)*
Analog inputs (alternate function of Port 1 pins). Apply voltage signals to these pins for ADC conversion.

$\overline{\text{CONVST}}$ *(input)*
Hardware convert start (alternate function of P3.8). Input pin for external ADC conversion trigger when external convert start function is enabled in software.

$\overline{\text{SS}}$ *(input)*
Slave select input for SPI interface (alternate function of P1.5).

MISO *(I/O)*
SPI master input / slave output (alternate function of P3.3). Data input pin in SPI master mode, and data output pin in SPI slave mode.

SDATA / MOSI *(I/O)*
I^2C data pin & SPI master output / slave input pin. Data input and output pin in I^2C mode. Data output pin in SPI master mode, and data input pin in SPI slave mode. Pull this pin to ground if unused.

SCLOCK *(I/O)*
SPI/I^2C serial clock pin. Serial clock input pin in SPI or I^2C slave modes. Serial clock output pin in SPI or I^2C master modes. Pull this pin to ground if unused.

RxD *(I/O)*
Receiver data input (alternate function of P3.0). Receiver data input pin of UART serial port in asynchronous mode. Data input/output pin of same in synchronous mode.

TxD *(output)*
Transmitter data output (alternate function of P3.1). Transmitter data output pin of UART serial port in asynchronous mode. Clock output pin of same in synchronous mode.

T0 *(input)*
Timer/Counter 0 input (alternate function of P3.4).

T1 *(input)*
Timer/Counter 1 input (alternate function of P3.5).

T2 *(input)*
Timer/Counter 2 input (alternate function of P1.0). Can be used to connect an external clocking source to Timer/Counter 2.

T2EX *(input)*
Timer/Counter 2 capture/reload trigger (alternate function of P1.1). Can be used to connect an external capture or reload signal to Timer/Counter 2.

$\overline{\text{INT0}}$ *(input)*
Interrupt 0 (alternate function of P3.2). Can be configured to trigger an interrupt from an external signal. Can also be used as a gate control input to Timer 0.

$\overline{\text{INT1}}$ *(input)*
Interrupt 1 (alternate function of P3.3). Can be configured to trigger an interrupt from an external signal. Can also be used as a gate control input to Timer 1.