MICROCONVERTER GENERIC

LAYOUT AND DESIGN CONSIDERATIONS

Q: The MicroConverter has separate pins for **analog and digital grounds**. Should I connect these to two separate ground planes on my board?

A: No. At least, not unless the two ground planes are connected together very near the chip. In all other situations, it is best to keep the MicroConverter on a single ground plane. If you have two separate ground planes on your board, then place the MicroConverter on the quieter of the two (usually the analog plane) to obtain optimum ADC and DAC performance.

Q: I have some **fast logic edges** on some of the digital circuitry I plan to connect to the MicroConverter. Will these signals impact the analog performance of the chip?

A: Signals with rise and fall times of less than 5ns or so, when applied directly to a MicroConverter's digital input pins, can potentially feed through and degrade analog performance. A simple solution to this problem is found in the form of a series resistor. A series resistor of around 200Ω will slow an edge sufficiently that it won't affect the MicroConverter's analog performance.

MISCELLANEOUS

Q: When a MicroConverter DAC is disabled, what does it's output look like? Is there a **DAC "tri-state"** feature?

A: MicroConverter DACs default to a disabled state when the chip is powered up. In this state, the output appears as a high impedance node to the rest of the world. This is analogous to a "tri-state" type output in digital logic terms. You can place the DAC output into this high impedance state whenever you like simply by disabling the DAC again.

Q: I heard that the MicroConverter can address up to **16Mbytes of external data memory**. Is this true and how is it done?

A: Yes, the MicroConverter can address up to 16Mbytes of external data memory. If access to more than 64 Kbytes of RAM is desired, a feature unique to the MicroConverter allows addressing up to 16 Mbytes of external RAM simply by adding an additional latch on the Port 2 address bus.
On a standard 8051 as with the MicroConverter 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 MicroConverter (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 64kByte external data memory access is maintained.

Q: I would like to serially download new code to the MicroConverter using my host machine. What is the **serial download protocol** used to reprogram the MicroConverter?

A: MicroConverters can be serially reprogrammed in your system using Analog Devices' Windows based download program (WSD.exe) to communicate through the chip's UART. Alternatively, you can in-system-reprogram the MicroConverter from any other host processor using the same serial download protocol used by the WSD application. Refer to tech note uC004 available on the web or as part of the QuickStart™ Development System.

Q: What's this "**power-on configuration routine**" that I've heard about?

A: Every MicroConverter product features a "power-on configuration routine" that runs every time you apply power to the chip or reset it. Basically, this is a small piece of code that is executed prior to the execution of your code. It is used to configure some of the on-chip peripherals such as the ADC and Flash/EE memory with factory optimized calibration and timing parameters. Some of these you can see (for example, default ADC offset and gain calibration registers will be different from one chip to the next) and others you can't (for example, ADC linearity, PTAT coefficients are not visible to your code). For the MicroConverter devices that run off a 32kHz crystal the power on configuration will also wait for the part to 'lock' before it starts to run user code.
The power-on configuration routine is stored in a hidden area of program ROM. The address where the routine begins is FF00 hex. Although the power-on configuration routine is "mapped" to addresses FF00 through FFFF hex, this does not interfere with external code memory which shares these addresses. Up to 64K bytes of user code can be executed from an external PROM chip (or 8K from internal Flash/EE and 56K from external PROM).
The ALE output is automatically disabled during the execution of the power-on configuration routine so that you can tell when your code starts to execute. ALE only begins toggling when the first line of *your* code is executed.

Q: The $\overline{\text{PSEN}}$ pin is always an output on an 8051. How then can it be used as an input to **enter serial download mode?** What pulldown resistor should be used to be sure to enter serial download mode? Is the standard functionality of the $\overline{\text{PSEN}}$ pin affected?

A: On an 8051 the $\overline{\text{PSEN}}$ pin is always an output. It is used to access from external code memory. On the MicroConverter the $\overline{\text{PSEN}}$ pin is used to access external code memory and is used as an input to determine whether to run user code or run the on chip serial downloader.
The $\overline{\text{PSEN}}$ pin is normally configured as an output. When the RESET pin is pulled high, the $\overline{\text{PSEN}}$ pin becomes a digital input and the voltage at the $\overline{\text{PSEN}}$ pin is sampled every machine cycle. Once the RESET pin is released (low) the last sampled logic level of the $\overline{\text{PSEN}}$ pin is used to decide whether to run from user code or whether to run the on chip serial downloader.
To ensure that the $\overline{\text{PSEN}}$ pin is interpreted as a logic low it should be pulled low through a 1k resistor to ground. To ensure that the $\overline{\text{PSEN}}$ pin is interpreted as a logic high it can be left float as there is an internal pullup resistor.
Because the functionality of the $\overline{\text{PSEN}}$ pin on the MicroConverter only differs from that of an 8051 when RESET is pulled high (in this condition the $\overline{\text{PSEN}}$ pin on an 8051 is useless) the functionality of the $\overline{\text{PSEN}}$ pin on the MicroConverter is unaffected by its additional functionality.

Q: Do I have to enter "NOP" instructions to wait for the **Flash/EE erase and program times** to elapse before performing the next data Flash/EE operation?

A: Absolutely not. All timing for access to the data Flash/EE space is taken care of for you in hardware. When you perform a Flash/EE *erase* or *program* command, the MCU core will not move on to the next machine cycle until the Flash/EE operation is complete. Effectively this means that a Flash/EE erase or program command will only take one machine cycle, but that this machine cycle is stretched out over a 20ms/250μs period for a Flash/EE *erase/program* command on the ADuC812 or over a 2ms/250μs period for a Flash/EE *erase/program* command for the ADuC824.

Q: What happens if the power supply falls during a **program or erase of the Flash/EE data memory?**

A: If the power supply falls below 2.7V during a Flash/EE program or erase instruction then the core cannot guarantee that the instruction will execute correctly.

Also, because the Flash/EE program or erase instruction takes much longer than a typical instruction (250us for a program and 2ms for an erase (20ms for ADuC812), the time required to respond to interrupts can be greatly increased if the interrupt occurs during a flash/EE program or erase instruction.

For instance, if a power-supply-monitor (PSM) interrupt occurs during a Flash/EE program or erase instruction, it will only be processed after the instruction is complete. In this way, the PSM can be used to indicate when power has dropped below a specified threshold during a Flash/EE program or erase instruction, thereby indicating that the instruction may not have executed correctly.

Q: Are there any **sockets available for the52PQFP package**?

A: Yes, but only test or evaluation type sockets. They aren't suitable for production since they are ZIF types that cost many times the price of the MicroConverter products. They can, however, prove useful for test/programming heads or in some cases for debug/prototyping setups. A couple of different types include the below....

Enplas:
    OTQ-52-0.65-01:   "open-top" type
    FPQ-52-0.65-01A: "clam-shell" type
(sockets mount on through-hole pattern, useful for test/programming heads.)

Ironwood Electronics:
    CA-QFE52SB-L-Z-T-01: socket module
    SF-QFE52SB-L-01:   board header
(board header solders to 52PQFP footprint, and socket adapter plugs into it to accept MicroConverter package. also provides test point connections to all 52 pins. useful for debug/prototyping setups.)

ADUC831 SPECIFIC

ADUC831 ADC:

Q: Can I connect a **high impedance analog source** directly to one of the ADuC831's inputs? Or must I buffer the signal first?

A: The main limitation with high impedance sources is the input leakage current of the ADuC812's analog inputs, which is typically ±1µA. This current flowing through a source impedance of 610Ω will generate 610µV of error. With a 2.5V reference, this is 1LSB (or 2.5V/4096). Therefore, a source impedance of greater than 610Ω can potentially generate measurable DC errors.

Q: What **acquisition time** is required by the ADuC831? How do I choose the proper values for the acquisition time select bits in ADCCON1?

A: In nearly all cases, an acquisition time of 1 ADC clock (ADCCON1.2=0, ADCCON1.3=0) will provide plenty of time for the ADuC831 to acquire its signal before switching the internal track&hold amplifier in to hold mode. The only exception would be a high source impedance analog input, but these should be buffered first anyway since source impedances of greater than 610Ω can cause DC errors as well.

Q: How do I choose the **conversion time** in the ADuC831's ADCCON1 register? And what are these **ADC clock divide bits** anyway?

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

The total ADC conversion time is 15 ADC clocks, plus 1 ADC clock for synchronization, plus the selected acquisition time (1, 2, 3, or 4 ADC clocks). For the example above, with a 1 clock acquisition time, total conversion time is 17 ADC clocks (or 5.67µs for a 3MHz ADC clock).

Q: How do I determine the ADuC831's **sample rate in continuous conversion mode**?

A: In continuous conversion mode, a new conversion begins each time the previous one finishes. The sample rate is then simply the inverse of the total conversion time described above. In the example above, the continuous conversion mode sample rate would be 176.5KHz.

Q: What is the ADuC831's **aperture delay** in hardware CONVST mode? And what **aperture uncertainty** can I expect?

A: In hardware CONVST mode an external logic input is used to trigger ADC conversions. The *aperture delay* of the ADuC831 is the time from the rising edge of that external trigger to the moment when the sample & hold amplifier goes into hold mode. This time is equal to the acquisition time (selected via ADCCON1) plus a synchronization time of between 0.5 and 1.5 ADC clock periods.

When the CONVST trigger is asynchronous to the ADC clock, this results in an *aperture uncertainty* of 1 ADC clock period. This aperture uncertainty can be avoided by synchronizing the external CONVST signal with the ADC clock. Since the ADC clock is simply a divided down master clock it is not available to you directly. Therefore, to synchronize your CONVST signal with the ADC clock, you must synchronize it with a *divided* master clock, where the divide ratio is a direct multiple of the divide ratio used to generate the ADC clock (selected in ADCCON1).

Q: What happens if the ADuC831 receives a **second CONVST trigger** (software SCONV, Timer2 trigger, or hardware CONVST trigger) during a conversion that it hasn't yet completed?

A: The second CONVST trigger will be ignored. In this situation, the second trigger event is lost. To avoid this loss of samples, make sure your software checks the state of the ADC busy flag (ADCCON3.7) before starting a conversion. In timer driven or hardware driven conversion modes, make sure the timer overflow rate or the input edge rate is more that the ADC conversion time plus acquisition time (controlled by ADCCON1).

ADuC831 VOLTAGE REFERENCE:

Q: I'm using the ADuC831's **internal voltage reference**. What should I do with the V_{REF} and C_{REF} pins?

A: Decouple both to ground using 0.1µF chip capacitors with short trace lengths.

Q: I would like to **drive other circuitry with the internal voltage reference** of the ADuC831. Should I take this voltage from the V_{REF} pin or the C_{REF} pin? What are the current source/sink capabilities of these pins?

A: Use the C_{REF} pin. The C_{REF} value is the value used as the reference for the ADC and if your system requires the same reference for all parts this should be used. On the ADuC831 V_{REF} was recommended to be used. To keep compatability with ADuC831 designs this may also be used but there will be a gain error with respect to C_{REF}. With regard to the ability of the C_{REF} or V_{REF} pin to sink and source current, it really has none. It is effectively 2.5V with a nominally 50KΩ source impedance. You must buffer the voltage on these pins if you wish to use the voltage to drive other circuitry. Of course, decouple the V_{REF} and C_{REF} pins with 0.1µF chip capacitors to ground just as you would normally.

Q: Can I use an **external voltage reference** with the ADuC831?

A: Yes. Short the V_{REF} and C_{REF} pin together and drive both these pins with your external voltage reference. Bit 6 of ADCCON1 must be set to enable the use of the external voltage reference. V_{REF} and C_{REF} must be decoupled with a 0.1µF chip capacitor as in any other configuration.

Q: Can I use an **external voltage reference** in the same configuration as with the ADuC812

A: Yes. Simply drive the V_{REF} pin with your external voltage reference. This will look like a 50KΩ load to 2.5V. Depending on the voltage reference you choose, you may or may not want to capacitively decouple the V_{REF} node. The C_{REF} pin, however, must be decoupled to ground using a 0.1µF chip capacitor as in any other configuration.

An external voltage reference up to AV_{DD} is allowed although for $V_{REF} = AV_{DD}$ the V_{REF} and the C_{REF} pins should be tied together to bypass the internal buffer that connects them.

Note this introduces a gain error for the ADC which has to be calibrated out, thus the previous method is the recommended one for most users.

Q: How do I **enable and disable the internal voltage reference** of the ADuC831? How long after enabling the internal voltage reference should I allow for **reference voltage settling time**?

A: The ADuC831's internal voltage reference is automatically enabled once the ADC or DAC is powered on. To disable the internal reference both the DACs and ADC need to be powered down and the external reference selected in ADCCON1. Once enabled, the voltage reference will require 65ms to turn on and settle to an accurate value.

UART SERIAL PORT

Q: What **baud rates** can be achieved by the ADuC831? How is the **baud rate configured**?

A: The ADuC831 can achieve baud rates up to 500kbaud when using a 16MHz crystal. In terms of standard baud rates the highest baud rate obtainable by the ADuC834 is 460800 baud.
The ADuC831 includes a special baud rate timer that allows much more flexibility as regards achieving particular baud rates (see the datasheet for more details). There are two advantages why one should configure the baud rate using the UartBaud generator.
1) To achieve particular standard baud rates up to and including 230400 and 460800 baud.
2) To free up a timer. The user is free to use all 3 timers for timer functions while the baud rate is controlled via the UartBaud generator.

Q: How do I **configure the ADuC831 UART serial port at 9600/115200 baud?**

A: The UART serial port on the ADuC831 can be configured for 9600 baud using the following piece of code (assuming a 11.0592MHz clock)

```
MOV T3CON, #85h
MOV T3FD, #08
MOV SCON, #52h
```

The UART serial port on the ADuC831 can be configured for **115200 baud** using the following

piece of code (assuming a 11.0592MHz clock)

```
MOV T3CON, #81h
MOV T3FD, #20
MOV SCON, #52h
```

ADuC831 Miscellaneous

Q: Will I need to implement an external **reset generator** circuit for my ADuC831 design

A: No you will not have to implement an external reset generator. The ADuC831 has an integrated robust power on reset circuit. The pin can be left float for applications that will run from a power cycle or driven directly by another circuit.

Q: I want to **drive LEDs directly** from some digital outputs what pins should I use?

A: The high drive port pins are D0 and D1. Both of these port pins are tested to sink 8mA and keep the VOL below 0.4V which is easily capable of sinking enough current for an LED.

_____

Q: Port 1 on the ADuC831 is primarily used for analog inputs. Can I use **Port 1 pins for digital I/0?** If so, how?

A: Pins 1.0 to P1.7 inclusive are analog input/digital input pins. They have no digital output capabilities. The default configuration for these six Port 1 pins is as analog inputs. In the default state, the Port 1 register contains FFh. To configure any of these six Port 1 pins as a *digital* input, simply clear the corresponding bit to zero in the Port 1 register (P1).

_____

Q: The ADuC831 features **security bits** to restrict access to Flash/EE code and data space. Can I accidentally write to them in my code and prevent the part from entering serial download mode?

A: The ADuC831 facilitates three modes of Flash/EE program memory security. These modes can be independently activated, giving different levels of security to your code as described in the ADuC831 datasheet.
The security bits are read-only from user code and exist at byte address FFFEh. They can be read using the following piece of code.

```
MOV EADRH, #0FFh ; address the Sec Bits page
MOV EADRL, #0FEh
MOV ECON, #81H ; read byte command
```

The security bits can be only be set after parallel programming the ADuC831 or directly after a serial download by following the serial download protocol as described in uC004 (ver 2.1). This protocol is implemented in our Windows serial downloader (WSD) available as part of the QuickStart development system.

Q: The ADuC831 has two DACs. How can I **update the DACs simultaneously** to the new voltage?

A: The ADuC831 features a *synchronous update* bit (in DACCON) which can be used to ensure that both DAC outputs step to their new values at the same instant. To use this feature, clear the bit to 0 before reloading new values to both DAC registers. Once new data has been loaded into both DAC registers, set the bit to 1 again. Both DAC outputs will then simultaneously step to their new values.

Q: I want to initiate the **serial downloader through the SPI por**t, how can I do this?

A: A bootloader mode is implemented on the ADuC831 allowing re-programming of the flash/EE program memory using any peripherals or any protocol that the user wishes.
To download to this space you will have to write your own downloader in the 6kBytes of bootloader space (E000H -> F7FFh of flash/EE program memory) to download to the rest of the flash/EE program memory. The downloader can use any means to download, parallel, SPI, I2C, UART or any other protocol you wish.

_____

Q: Do I have to set the **ETIM registers** as in the ADuC812 for different Crystal frequencies?

A: No. There are no ETIM registers to be set on the ADuC831. Flash/EE erase and program timing is derived from the master clock. On power-up the ADuC831 automatically detects the speed of the master clock and adjusts the flash timing to the correct value. The flash timing is determined by the EPM bits in the CFG831 register. These bits should not be modified by the user.

Q: How do I use the on-chip **temperature sensor** feature of the ADuC831?

A: The temperature sensor on the ADuC831 outputs a voltage which is *inversely* proportional to chip temperature. At 25°C, this voltage is approximately 600mV. As temperature changes, the voltage changes by –3mV/°C. So as operating temperature *rises*, the temperature sensor output voltage *decreases*. And of course the ADC is used to convert the voltage on the temperature sensor to a digital value.
It should be noted that the two figures quoted above of 600mV @ 25degreesC and -3mV/°C are typical figures. While these figures will vary from part to part the temperature sensor has good linearity. Hence if a calibration routine is used to calibrate at two temperatures (one for offset and the other for temperature gradient) then fairly accurate temperatures can be measured.
And remember, the on-chip temperature sensor is measuring chip temperature, not ambient temperature, so don't forget to factor self-heating into the equation if you wish to determine the temperature of the ambient air.

Q: How do I calculate the **total power consumed** by the ADuC831 under a given set of conditions?

A: Below is a table to allow simple calculation of the ADuC831's total current draw. All of the below are typical values. Simply add the current draw of all enabled peripherals to the core (normal mode) current consumption at your given master clock frequency (M_{CLK}).

	V_{DD}=5V	V_{DD}=3V
Core (normal mode):	1.6µs•M_{CLK}+5mA	0.8µs•M_{CLK}+1.5mA
Core (idle mode):	0.75µs•M_{CLK}+5mA	0.25µs•M_{CLK}+1.5mA
ADC:	1.3mA	1.0mA
DAC:	250µA	200µA
Voltage reference:	200µA	150µA

Q: The ADuC831 has separate pins for **analog and digital power supplies**. Can I run from a split supply, for example a 3.3V DV_{DD} and a 5V AV_{DD}? What other implications should I be aware of about the supplies?

A: No. You cannot run the ADuC831 from split supplies. The absolute maximum rating for the difference between DV_{DD} and AV_{DD} is ±0.3V. For this reason, it is best to run the ADuC812 from a single supply. The most that should separate these supplies (and only if necessary) is a small value ferrite bead (0805 chip size) or a small value (1 or 2 Ω) resistor. Of course, decouple each supply pin to the ground plane with a 0.1µF chip cap connected with short trace lengths, and make sure there are larger electrolytic reservoir capacitors decoupling the supply to ground somewhere on the board. If you do choose to split the supplies with a ferrite bead or a small resistor, make sure there are local chip caps *and* an electrolytic on *each* side of the split.

Q: **How *fast* can I run the master clock?**

A: The ADuC831's 8052 core is capable of running at master clock speeds much greater than the specified maximum of 16MHz without any errors. However, running the ADuC831 master clock very much faster than this specification can degrade analog performance. Therefore, we do not recommend the use of crystal or clock oscillator frequencies of greater than 16MHz.

Q: Does the ADuC831 feature a static MCU core? **How *slow* can I run the master clock?**

A: Yes, the ADuC831 does feature a static 8052 core. The core will operate at master clock frequencies right down to DC. However, the ADC clock is derived from the master clock, and ADC performance will degrade at very slow clock speeds. The minimum ADC clock frequency for reliable ADC performance is 400KHz. The ADC clock is equal to the master clock divided by 16, 2, 4, or 8 (selected by ADCCON1), so the minimum *master* clock frequency for reliable ADC operation is also 400KHz.

Q: Port 1 on the ADuC831 is primarily used for analog inputs. Can I use **Port 1 pins for digital I/0**? If so, how?

A: Port 1 pins can be used as analog inputs or as digital inputs, but not as digital outputs. The default configuration for all Port 1 pins is as analog inputs. In this state, the Port 1 register contains FFhex (all ones). To configure any Port 1 pin as a *digital* input, simply clear the corresponding bit to zero in the Port 1 register (P1).