ANOMALIES

1. Mode 0 UART Operation [er001]

Background:	UART Mode 0 allows the UART to function in an 8-bit shift register mode.
Issue:	UART Mode 0 is nonfunctional on the ADuC841/ADuC842/ADuC843.
Workaround:	None.
Related Issues:	None.

2. Use of the Extended Stack Pointer [er002]

Background:	The extended stack pointer allows the stack to overflow into internal XRAM.
Issue:	A PUSH onto the extended stack when it is the first instruction within a subroutine results in the return address being overwritten.
Workaround:	For Assembly code, insert a NOP as the first instruction in any subroutine. For C code, there is currently no workaround.
Related Issues:	None.

3. Use of I²C in Slave Mode with Stop Interrupt Enabled [er003]

Background:	In slave mode, the I²C interface can be configured to generate an interrupt due to a start, repeated start, data, or stop condition. The I²C interrupt decode bits (I2CID0 and I2CID1) in I2CCON indicate the source of the interrupt. If the stop interrupt is enabled via the I2CSI bit, an interrupt is generated when the slave receives a stop condition.
Issue A:	In the I²C interrupt service routine, if the I2CI bit or I2CDAT register is accessed during a stop interrupt, the I²C bus will fail to respond to further I²C communication.
Workaround A:	When a stop interrupt is detected, the user should reset the I²C bus by using the I2CRS bit.
Issue B:	When the stop interrupt is enabled, on occasion the I²C interrupt decode bits indicate that a start, repeated start, or DATA interrupt occurred when the source was in fact a stop interrupt. If this happens the user may try to clear I2CI or read I2CDAT, resulting in the bus failing to respond to further I²C communication.
Workaround B:	Tie the SCLOCK pin to an I/O pin; This allows the state of SCLOCK to be read. SCLOCK is high only during an interrupt if the source is a stop interrupt.
Related Issues:	er004: Use of I²C in slave mode with stop interrupt disabled.

4. Use Of I²C in Slave Mode with Stop Interrupt Disabled [er004]

Background:	In slave mode, the I²C interface can be configured to generate an interrupt due to a start, repeated start, data, or stop condition. The I²C interrupt decode bits (I2CID0 and I2CID1) in I2CCON indicate the source of the interrupt.
Issue:	Once one repeated start is detected by the I²C interface, all subsequent start conditions are detected as a repeated start even if a stop bit has been received between data transfers.
Workaround:	None.
Related Issues:	None.

5. I²C Data Transfer [er005]

Background:	The I2CDAT register is used to read or write data to the I²C bus. The I2CDAT register has an SFR address of 0x9A.
Issue A:	During an I²C transfer if a user accesses the RAM address 0x9A the contents of the I2CDAT SFR can be modified.
Workaround A:	For Assembly code: Do not use memory location 0x9A For C code: Assign a dummy variable to location 0x9A using the following code: `idata unsigned int ui32Dummy[2] _at_ 0x9A;`

Issue B:	During an I²C transfer, if a user executes either of the following instructions, the contents of the I2CDAT SFR can be modified.
	`MOV dest,#9AH`
	`SUBB A,R2`

Workaround B:	To prevent code from changing the contents of the I2CDAT SFR make sure that neither of these instructions are executed during an I²C transfer.
Issue C:	A small percentage (<3%) of I²C stop conditions received by a slave cause the I²C bus to enter a state where the subsequent START+matching address is not acknowledged.
Workaround C:	If a second START+matching address is issued following a NACK, this is detected correctly.
Related Issues:	None.

6. SPI Interface [er006]

Background:	The SPI can either be used on the standard pins or can be moved to P3.3, P3.4, and P3.5 by setting the MSPI bit in CFG841/CFG842. When the MSPI bit is set, P3.3 should be MISO, P3.4 MOSI, and P3.5 SCLOCK.
Issue A:	By setting the MSPI bit, the P3.3, P3.4, and P3.5 have the following configuration: P3.3 = MISO, P3.4 = SCLOCK, P3.5 = MOSI
Workaround A:	None.
Issue B:	When the ADuC841/ADuC842/ADuC843 is set up as an SPI slave, the device may receive or transmit bytes incorrectly.
Workaround B:	Incorporate checksums into all communication with the ADuC841/ADuC842/ADuC843 slave. This allows the master devices to retransmit if an error occurs.
Related Issues:	None.

7. Interrupts During Reading/Writing to Data FLASH/EE [er007]

Background:	There are 4 kB of DATAFLASH/EE that can be used for nonvolatile data storage.
Issue:	If an interrupt occurs during a DATAFLASH/EE read or write operation, code execution following the ISR may resume at a random program memory address.
Workaround:	Disable all interrupts prior to a read or write operation. This can be done by setting the EA bit to 0.
Related Issues:	None.

8. PWM Operation [er008]

Background:	The PWM output rate is determined by the PWMxH and PWMxL registers for the PWM0 and PWM1 outputs.	
Issue:	Modifying RAM Address 0x2E causes the PWM timer to be reset.	
Workaround:	For Assembly code:	Do not use memory location 0x2E.
	For C code:	Assign a dummy variable to location 0x2E using the following code: `idata unsigned int ui32Dummy[2] _at_ 0x2E;`
Related Issues:	None.	

9. Watchdog Timer [er009]

Background: The ADuC841, ADuC842, and ADuC843 incorporate a Watchdog Timer. The purpose of the WDT is to ensure the part is never stuck in an endless loop by generating either a hardware reset or an interrupt event that vectors to the WDT ISR.

Issue: If the WDT generates an interrupt as opposed to a hardware reset, and if the ISR subsequently sets up the WDT to time out to a hardware reset, the reset is ignored.

Workaround: Ensure that a double write to the WDCON is executed inside the ISR with the first write being a reset of the WDT. For example:

```
void isr_wdt( void ) interrupt 11
{
     WDWR = 1;     // This first WDT write is required to get the WDT to work inside the ISR.
     WDCON  = 0x60;// Reset WDT.
     WDWR = 1;     // Now set the WDT to the required 1s timeout
     WDCON  = 0x62;// select reset after 1000mS
     while(1);
}
void main(void)
{
     EA = 0;
     WDWR = 1;     // Allow write to WDCON
     WDCON = 0x6A; // timeout=1000mS, WDT enable, WDT ISR Interrupt
     while (1);
}
```

Related Issues: None.

10. Level Triggered Interrupt Operation [er010]

Background:	The ADuC841/ADuC842/ADuC843 incorporate two external interrupt sources (INT0 and INT1) that can be configured to respond to either an edge event or a level event.
Issue:	If an interrupt occurs on the INT0 or INT1 pins and is then removed within one core instruction cycle, the interrupt vector address that is generated may be incorrect resulting in a vector to 0x0000. This effectively restarts code execution.
Workaround:	To ensure that this does not occur the level triggered interrupt source must be kept low for a minimum of 9 core clock cycles.
Related Issues:	None.

11. Stack Pointer in ULOAD Mode [er011]

Background:	When starting user code, the stack pointer should, by default, be initialized to Address 0x07.
Issue:	In ULOAD mode, the stack pointer defaults to 0x03 causing conflict between RAM locations R4 to R7 and the stack.

Workaround:	Manually change, in code, the stack pointer address to 0x07 or to the address that is required upon entry to ULOAD mode, that is,
	`MOV SP, #07H`
	or
	`SP = 0x07;`

Related Issues:	None.

ADuC841/ADuC842/ADuC843 SILICON ANOMALIES

Anomaly No.	Description	Status
er001	Mode 0 UART Operation	Pending
er002	Use of the Extended Stack Pointer	Pending
er003	Use of I^2C in Slave Mode with Stop Interrupt Enabled	Pending
er004	Use Of I^2C in Slave Mode with Stop Interrupt Disabled	Pending
er005	I^2C Data Transfer	Pending
er006	SPI Interface	Pending
er007	Interrupts During Reading/Writing to Data FLASH/EE	Pending
er008	PWM Operation	Pending
er009	Watchdog Timer	Pending
er010	Level Triggered Interrupt Operation	Pending
er011	Stack Pointer in ULOAD Mode	Pending

S03260–0–7/04(A)

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Analog Devices Inc.:

EVAL-ADUC841QSPZ EVAL-ADUC841QSZ EVAL-ADUC842QSPZ EVAL-ADUC842QSZ ADUC843BSZ62-3 ADUC843BCPZ62-3 ADUC842BCPZ32-5 ADUC842BSZ62-5 ADUC841BCPZ62-3 ADUC842BCPZ8-5 ADUC842BSZ62-3 ADUC842BCPZ62-3 ADUC843BCPZ32-3 ADUC842BCPZ32-3 ADUC843BCPZ8-5 ADUC841BCPZ8-5 ADUC842BCPZ8-3 ADUC843BSZ62-5 USB-EA-CONVZ ADUC843BCPZ8-3 ADUC842BCPZ62-5 ADUC841BCPZ62-5 ADUC841BSZ62-3 ADUC843BCP32Z-5 ADUC843BCP62Z-5 ADUC841BCPZ8-3 ADUC841BSZ62-5