Active Errata List

- **Timer 2 – Baud Rate Generator – No IT When TF2 is Set by Software**
- **Timer 2 – Baud Rate Generator – Long Start Time**
- **UART – RB8 Lost with JBC on SCON Register**
- **ADC – Interrupt During Idle Conversion**
- **CAN – CANCONCH Harmless Corruption**
- **CAN – Sporadic Errors**
- **C51 Core – Bad Exit of Power-down in X2 Mode**
- **Timer0/1 – Extra Interrupt**
- **Timer1 - Mode 1 Does Not Generate Baud Rate Generator for UART**
- **EEPROM - Lock-up During ISP write**
- **Transmission after a 3 bit Intermessage**

Errata History

Lot Number	Errata List
A00151	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17,18
A00369	1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17,18
A00367, A00368, A00396 to A00529	1, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17,18
A00510, all lots from A00588 (included) to A03874 (excluded)	**1, 7, 8, 9,10, 11, 12, 13, 14, 15, 16, 17,18**
Lots A03874, A03875, A03876 and all lots from A04440 (excluded)	**7, 8, 9,10, 11, 13, 14, 15, 16, 17,18**

Note: For lots between A03876 (excluded) and A04440 (included) it is not determined wether they will have the latest revision errata (7, 8, 9, 10, 11, 13, 14, 15, 16) or the previous revision.

Errata Description

1. Flash/EEPROM – First Read after Write Disturbed

After a write of more than 32 bytes in the EEPROM and 16 bytes in the user Flash memory, the read of the first byte may be disturbed if it occurs just after the write.

Workaround

Do not load/write more than 32 bytes at a time for EEPROM memory.

Do not load/write more than 16 bytes at a time for user Flash memory.

Or wait 10 ms before reading the first byte.

2. Buffer Noise

Large bounces and high noise are generated when buffers are switching (both rising and falling edges).

Workaround

None.

3. Double IT on External Falling Edge on INT1 or INT0 in X2 Mode

When the CPU is in X2 mode and Timer 1 or Timer 0 in X1 mode (CKCON = 0x7F), IEx flag is not cleared by hardware after servicing interrupt. In this case, the CPU executes the ISR a second time.

CAN Microcontrollers

T89C51CC01
T89C51CC01UA
T89C51CC01CA

Errata Sheet

Workaround

The workaround is to clear IEx bit in Interrupt subroutine.

```
INT1_ISR :                    ; Interrupt sub routine
CLR IE1
....
```

4. Movc Instruction on Boot Memory from Boot Memory Does Not Work

No movc instruction is performed when a program running on the boot memory tries to read its own code using a movc instruction.

Workaround

None.

5. Power OFF Flag

Power OFF Flag does not work.

Workaround

None.

6. CAN – Lost CAN Error Interrupt

When a stuff error occurs during a CAN frame transmission on DPRAM write access, the controller does not generate the error interrupt and any received frame can generate a Receive interrupt.

Workaround

None.

7. Timer 2 – Baud Rate Generator – No IT When TF2 is Set by Software

When Timer 2 is used in baud rate generator mode, setting TF2 by software does not generate an interrupt.

Workaround

Use Timer 1 instead of Timer 2 to generate baud rate and interrupt.

8. Timer 2 – Baud Rate Generator – Long Start Time

When Timer 2 is used as a baud rate generator, TH2 is not loaded with RCAP2H at the beginning, then UART is not operational before 10000 machine cycles.

Workaround

Add the initialization of TH2 and TL2 in the initialization of Timer 2.

9. UART – RB8 Lost with JBC on SCON Register

May lose RB8 value, if RB8 changes from 1 to 0 during JBC instruction on SCON register.

Workaround

Clear RB8 at the beginning of the code and after each time it goes to 1.

10. ADC – Interrupt Controller/ADC Idle Mode/Loops In High Priority Interrupt

The problem occurs during an A/D conversion in idle mode,If a hardware interrupt occurs followed by a second interrupt with higher priority before the end of the A/D conversion. If the above configuration occurs, the highest priority interrupt is served immediately after the A/D conversion. At the end of the highest priority interrupt service, the processor will not serve the hardware reset interrupt pending. It will also not serve any new interrupt requests with a priority lower than the high level priority last served.

Workaround

Disable all interrupts (Interrupt Global Enablet Bit) before starting an A/D conversion in idle mode, then re-enable all interrupts immediately after.

11. CAN – CANCONCH Harmless Corruption

When the stuff error occurs (same condition than the errata 6), the CONCH1, CONCH0 bits in CANCONCH are corrupted. This corruption has no effect on the behavior of the Transmit channel.

Workaround

No workaround is required, re-writing CANCONCH to start a new message resolves this corruption.

12. Flash/EEPROM – First Read after Load Disturbed

In the 'In-Application Programming' mode from the Flash, if the User software application loads the Column Latch Area prior to calling the programming sequence in the CAN Bootloader.

The 'Read after load' issue leads to a wrong Opcode Fetch during the column latch load sequence.

Workaround

Update of the Flash API Library. A NOP instruction has to be inserted after the load instruction.

```
MOVX @DPTR,A ;Load Column latches
NOP ; ADDED INSTRUCTION
```

13. CAN – Sporadic Errors

When BRP = 0 or when BRP > 0 and SMP = 0, the CAN controller may desynchronize and send one error frame to ask for the retransmission of the incoming frame, even though it had no error.

This is likely to occur with BRP = 0 or after long inter frame periods without synchronization (low bus load). The CAN macro can still properly synchronize on frames following the error.

Workaround

Setting BRP greater than 0 in CANBT1 and SMP equals 1 in CANBT3 allows re-synchronization with the majority vote, and thus fixes the issue.
The sampling point might have to be slightly advanced for the majority vote to take place within the bit. Therefore, at maximum speed of 1Mbit/s, the sampling point should be at less than 80% (e.g. 75%) for XTAL = 16 MHz or less than 85% (e.g. 80%) for XTAL = 20 MHz.

14. C51 Core – Bad Exit of Power-down in X2 Mode

When exiting power-down mode by interrupt while CPU is in X2 mode, it leads to bad execution of the first instruction run when CPU restarts.

Workaround

Set the CPU in X1 mode diretly before entering power-down mode.

15. Timer0/1 – Extra interrupt.

When the Timer0 is in X1 mode and Timer1 in X2 mode and vice versa, extra interrupt may randomly occured for Timer0 or Timer1.

Workaround

Use the same mode for the two timers..

16. EEPROM – Lock-up during ISP write

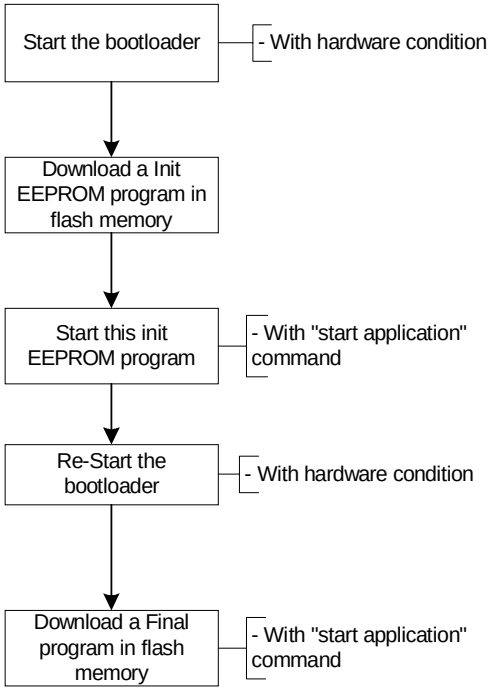
Program lock-up can be experienced when using Atmel FLIP software or a custom In system Pogramming tools to write in the internal EEPROM. This problem occurs with clock frequency > 12MHz in X2 mode or 24MHz in X1 mode. Neither the code Flash nor the EEPROM are corrupted by the lock-up. Although the EEPROM cannot be written by the bootloader.

This problem does not affect the capability to erase, read and write into the code Flash and the additional bytes with In-System Programming as Atmel FLIP.

This problem is bootloader related and doesn't affect the capability for the user to read and write into the EEPROM. This problem is technology dependant. The likelyhood of experiencing the problem is low at Clock frequency > 24 MHz in X1 mode and non-existent below. A reset can be applied to recover from the lock-up.

Workaround

The following process can be done to work around the problem.



C Init EEPROM program example:

```
#define SIZE_EEPROM 12

unsigned char code
tab_eep[SIZE_EEPROM]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA};

void main (void)
{
  unsigned char xdata * address;
  unsigned char cpt;

  address =0x00;
  for (cpt=0; cpt<SIZE_EEPROM; cpt++)
```

```
  {
    EECON = 0x02; // enable eeprom access
    *(address + cpt) = tab_eep[cpt];
    EECON = 0x50;
    EECON = 0xA0;
    while (EECON&0x01);
  }
}
```

Assembler Init EEPROM program example:

```
SIZE_EEPROM EQU 00Ah

CSEG AT 0000H

Mov  R1, #SIZE_EEPROM
Mov  DPTR, #0h

Load_eeprom:
  MovEECON, #02h    ; set bit EEE for access to the column latches
  MovA, #Tab_eep
  Movc   A, @A + DPTR
  Movx  @DPTR, A

  Mov EECON, #050h
  Mov EECON, #0A0h

WAIT_FBUSY:
  Mov  A, EECON
  AnlA, #001h
  Xrl A, #001h
  JzWAIT_FBUSY
  Inc   DPTR
  Djnz  R1, Load_eeprom

End_loop:
  jmp End_loop

  Tab_eep: DB     012h, 023h, 045h, 067h, 089h, 0ABh, 0CDh, 0EFh
```

17. Transmission after a 3 bit Intermessage

If a Transmit Message Object (MOB) is enabled while the CAN bus is busy with an on going message, the transmitter will wait for the 3-bit Intermission before starting its transmission. This is in full agreement with the CAN recommendation.

If the transmitter lost arbitration against another node, two conditions can occur :

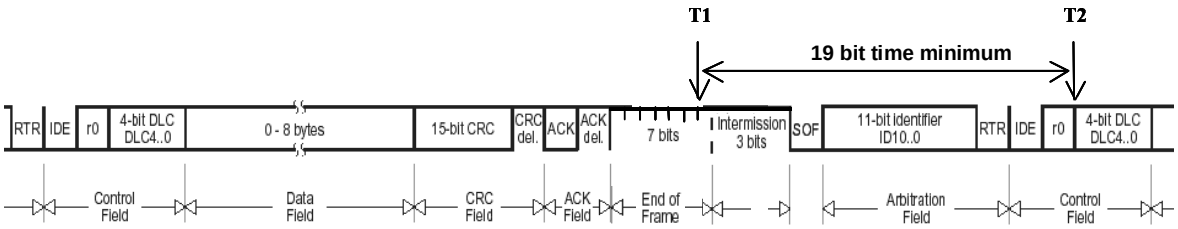
1. At least one Receive MOB of the chip are programmed to accept the incoming message. In this case, the transmitter will wait for the next 3-bit Intermission to retry its transmission.

2. No Receive MOB of the chip are programmed to accept the incoming message. In this case the transmitter will wait for a 4-bit Intermission to retry its transmission. In this case, any other CAN nodes ready to transmit after a 3-bit Intermission will start transmit before the chip transmitter, even if their messages have lower priority IDs.

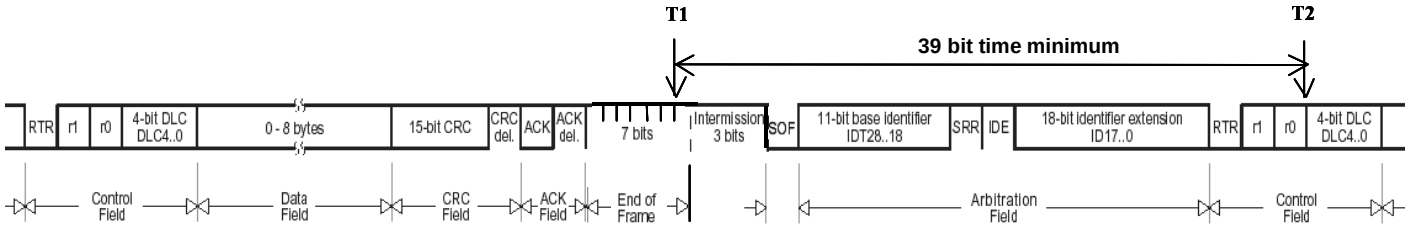
Workaround

Always have a Receive MOB enabled ready to accept any incoming messages. Thanks to the implementation of the CAN interface, a Receive MOB must be enable at latest, before the 1st bit of the DLC field. The Receive MOB status register is written (RXOK if message OK) immediately after the 6th bit of the End of Frame field. This will leave in CAN2.0A mode a minimum 19-bit time delay to respond to the end of message interrupt (RXOK) and re-enable the Receive MOB before the start of the DLC field of the next incoming message. This minimum delay will be 39-bit time in CAN2.0B. See CAN2.0A CAN2.0B frame timings below.

CAN2.0A



CAN2.0B



Workaround implementation

The workaround is to have the last MOb (MOb14) as "spy" enabled all the time; it is the MOb of lowest priority : If a Mob other than MOb14 is programmed in receive mode and its acceptance filter matches with the incoming message ID, this Mob will take the message. MOb14 will only take messages than no other MObs will have accepted. MOb14 will need to be re-enabled fast enough to manage back to back frames. The deadline to do this is the beginning of DLC slot of incoming frames as explained above.

Minimum code to insert in CAN interrupt routine:

```
__interrupt void can_int_handler(void)
{
if ((CANSIT1 & 0x40) == 0x40 )         /* MOb14 interrupt  (SIT14=1) */
  {
  CANPAGE = (0x0E << 4);      /* select MOb14 */
  CANSTCH = 0x00;              /* reset MOb14 status */
  CANCONCH = 0x88;        /* reception enable */
  }
........
........
}
```

18. Timer1 – in Mode 1 does not generate Baud rate to UART.

The timer1, when used as a baud rate generator in mode 1 (16 bits counter) for low baud rates, does not generate baud rate to UART.

Workaround

No.

Active UART Bootloader Errata List

- **Timer 2 and UART Are Not Stopped**
- **Watchdog and Flash API Starting the Bootloader Execution**
- **Autobaud False Start Bit Detection**
- **Boot Process Compatibility**
- **Flash API '__api_wr_code_page' with 0 Data in Length Parameter Field**

UART Bootloader Errata History

Version Number	Errata List
1.2 (1.1.2 displayed by FLIP)	1, 2, 3
1.4 (Current)	**1, 2, 3, 4,5**

UART Bootloader Errata Description

1. Timer 2 and UART Are Not Stopped

   When the bootloader receives the command 'Start Application' (LJMP 0), the Timer 2 and the UART are not stopped.

   Workaround

   The application must have in its setup function a reset of Timer 2 and UART.

```
mov SCON, #00h
mov T2CON, #00h
mov RCAP2L, #00h
mov RCAP2H, #00h
mov TL2, #00h
mov TH2, #00h
```

2. Watchdog and Flash API Starting the Bootloader Execution

When an application call '__api_start_bootloader' or '__api_start_isp' routines while the watchdog is enabled, when the watchdog overflows it will restart the application instead of the bootloader.

Workaround

Set BLJB(=1) before calling the __api_start_bootloader or __api_start_isp if the watchdog is used.

3. Autobaud False Start Bit Detection

UART autobaud sequence does not work in some special UARTs.

Some laptops have the UART TX line set to 0 when unused (COM port closed), this results in a false baud rate calculation in the 'U' character.

The autobaud sequence checkes for a '0' state (not a falling edge) on the Rx line of the UART microcontroller to detect the 'start' bit of the 'U' synchro character.

As this line is '0' by default when COM port is closed, the autobaud routine starts its baudrate calculation at the opening sequence of the UART.

Workaround

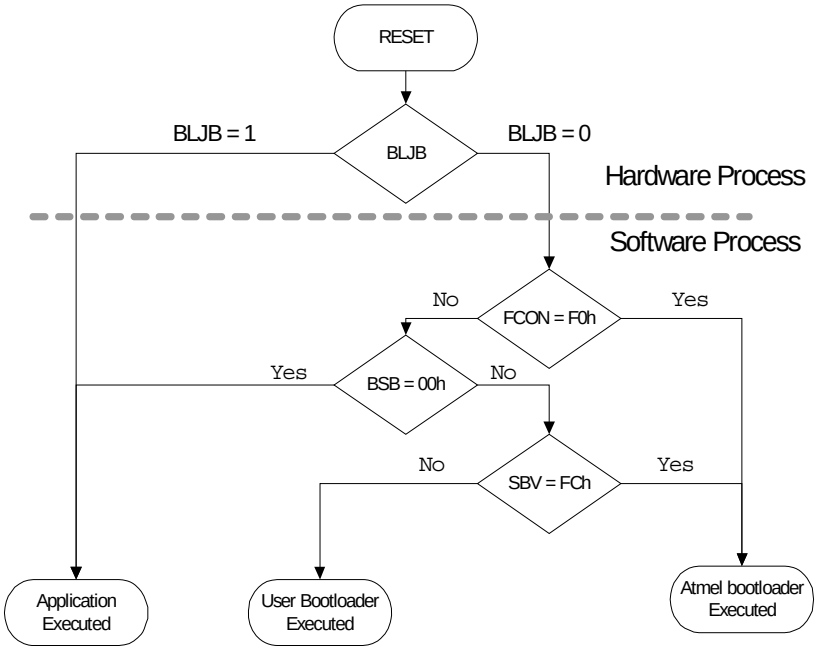
A 'Special Sync' can be used with 'FLIP' software.

In this case, the open port event and the 'U' sent are dissociated. The user must first open the COM port with the 'connect' button, then reset the hardware and finally push the 'sync' button.

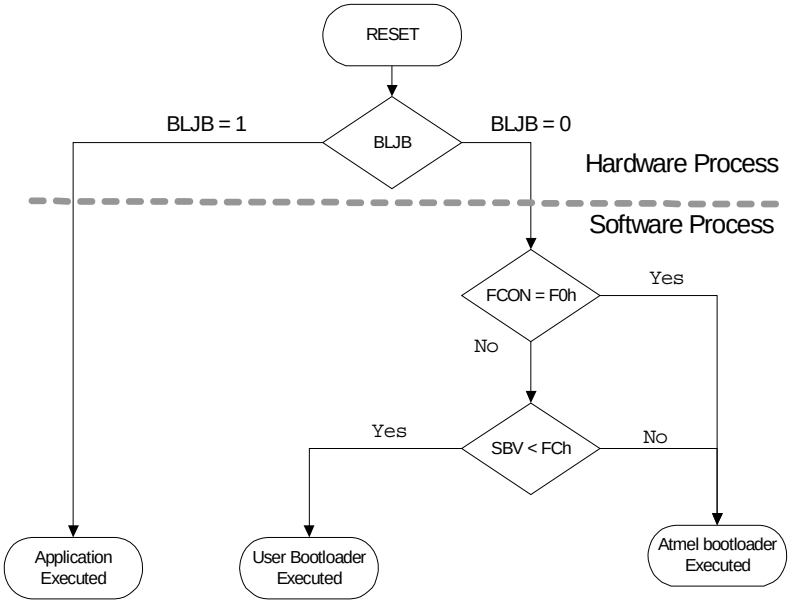
4. Boot Process Compatibility

There are some differences between Boot process of bootloader 1.2 and newer versions (see below).

Version 1.2:



Version 1.4:



Workaround

Use the fuse bit BLJB to start the application in both versions.

5. Flash API '__api_wr_code_page' with 0 Data in Length Parameter Field

When the Flash API '__api_wr_code_page' is called with the field 'nb_data' equal 0 then 255 data are written in Flash.

Workaround

Include a test on 'nb_data' before executing __api_wr_code_page routine.

Active CAN Bootloader Errata List

- **Watchdog and Flash API Starting the Bootloader Execution**
- **Start application with Software Security Set**

CAN Bootloader Errata History

Version Number	Errata List
1.0.4	1, 2, 3, 4, 5, 6
1.2.0 (Current)	2, 7

CAN Bootloader Errata Description

1. The CAN is Not Stopped

When the bootloader receives the command 'Start Application' (LJMP 0), the CAN is not stopped.

Workaround

The application must have in its setup function a reset of CAN macro.

```
mov CANGCON, #00h
```

2. Watchdog and Flash API Starting the Bootloader Execution

When an application call '__api_start_bootloader' or'__api_start_isp' routines while the watchdog is enabled, when the watchdog overflow it will restart the application instead of the bootloader

Workaround

Set BLJB(=0) before calling the '__api_start_bootloader' or '__api_start_isp' if the watchdog is used.

3. Flash API '__api_wr_code_page' with 0 Data in Length Parameter Field

When the Flash API '__api_wr_code_page' is called with the field 'nb_data' equals 0 then 255 data are written in Flash.

Workaround

Include a test on 'nb_data' before executing '__api_wr_code_page' routine.

4. Problem to program a hex file less than 16 bytes

When we try to program a hex file with a size size less than 16 bytes, some troubles appear depending of the start address.

Workaround

Program with a range address higher than 16 bytes.

5. Unexpected echo after start application command

When the command start application (with reset) is received by the CAN bootloader; the bootloader answers with a ramdom CAN frame before starting the application.

Workaround

The FLIP software is not impacted by this CAN frame.

6. CRIS Modification not applicable for In-application Usage

When the CAN Relocatable Identifier Segment (CRIS) is modified, this modification is not taken into account in the bootloader, when it is started by the application for In-application Programming usage.

Workaround

Use only CRIS=00h or reprogram with a parallel programmer the new CAN bootloader rev 1.2.0 available at http://www.atmel.com.
This new bootloader will correct also errata #1, 3, 4, 5, 6.

7. Start application with software security set

The start application with or without reset doesn't work if the software security are set.

Workaround

Use the SSB API to secure the device in the application.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

4131J–CAN–05/06