Active Errata List

- **CAN – Sporadic Errors**
- **PCA – Incorect Behavior with CPU X2 Mode Bit of HSB**
- **Timer0/1 – Extra Interrupt**
- **Transmission after a 3 bit Intermessage**

Errata History

Lot Number	Errata List
all lots from A02543	1, 2, 3, 4

Errata Description

1. 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 resynchronization 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.

2. PCA – Incorect Behavior with CPU X2 Mode Bit of HSB

When starting the microcontroller in X2 mode upon reset with the X2 fuse bit of the HSB, the PCA may not work properly when configured with Timer0 in X1 mode as clock input.

Workaround

Set the CPU in X2 mode by software by writing CKCON register at the begin of the application.

3. Timer0/1 – Extra Interrupt

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

Workaround

Use the same mode for the two timers.

4. 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.

CAN Microcontrollers

AT89C51CC03
AT89C51CC03C
AT89C51CC03U

Errata Sheet

4293G–CAN–06/05

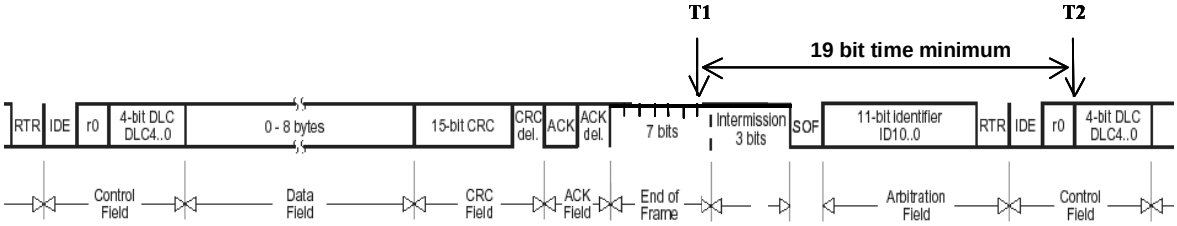
If the transmitter lost arbitration against another node, two conditions can occur :

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.

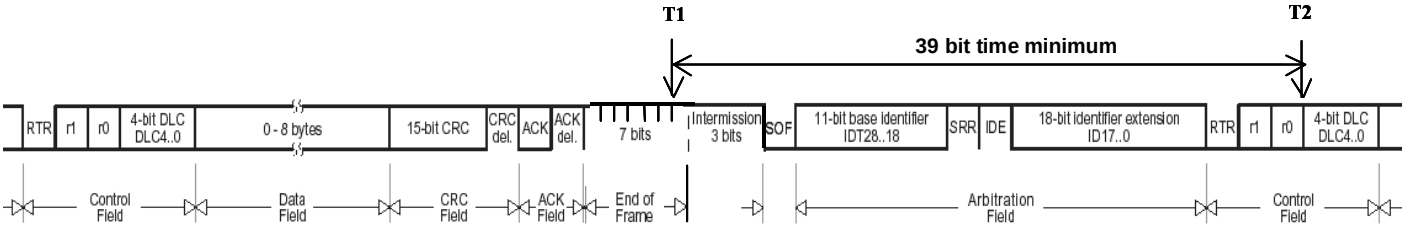
Workaround

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 */
  }
........
........
}
```

Active CAN Bootloader Errata List

- **Watchdog and Flash API Starting the Bootloader Execution**
- **Start application with security set**

CAN Bootloader Errata History

Version Number	Errata List	Date Code
1.0.0	1, 2, 3, 4, 5, 6, 7, 9	
1.0.2	**3, 8, 9**	
1.0.3	**3, 8,10**	
1.0.4	**3, 8**	**0601 and above**

CAN Bootloader Errata Description

1. Boot Process - SBV > 0x7F00 leads to Atmel bootloader execution

When BLJB bit is active and SBV > 0x7F00 the Atmel bootlader is executed instead of a custom bootloader.

Workaround

Update to bootloader revision 1.0.2

2. 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
```

3. 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.

4. 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.

5. 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 errors appear depending of the start address.

Workaround

Program with a range address higher than 16 bytes.

6. 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.

7. Start of the Bootloader from Application Using api_start_isp Doesn't Work

The API to start the bootloader directly in CAN communication open does not work.

Workaround

Start the bootloader using api_start_bootloader and send the CAN command to open the communication with the CAN bootloader.

8. 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.

9. Write column latches robustness improvement

The data bytes from the CAN messages are directly written in the column latches until the 128 bytes page is filled or the end of the program is reached. Afterward, the column latches are written in the flash page. If a CAN failure ocurs during the load process, it will leave the column latches partially written. A reset will be needed before starting over again.

Workaround

Use 1.0.3 revision: Incoming bytes are stored in RAM. Once the full page transfer is successfull, the 128 bytes RAM will be copied into the column latches, then the column latches copied into the flash. If a CAN failure occurs, nothing will be written in the column latches and flash page.

10. CAN autobaud

Due to uninitialized RAM cells, the CAN autobaud process may require additional time to synchronise properly with the In System Programing Interface (up to 30seconds).

Workarounds

Use 1.0.4 bootloader revision.

Or

When using ATMEL "Flip" In Sytem Programing Interface:

Update to 2.4.6 version and set timeout parameter to 30sec.

When using a custom In System Programing Interface:

Wait up to 30sec for the "Node Connect" frame to be acknowledged by the CAN bootloader.

Active UART Bootloader Errata List

- **No specific UART bootloader errata pending**

UART Bootloader Errata History

Version Number	Errata List
1.0.1	None

Atmel Corporation

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

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
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
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
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

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

4293G–CAN–06/05 0M