

## STATUS SHEET

**Type:** STM300  
**Step Code:** BB  
**Date:** 21. January 2010

---

### Product Status

- B-Sample (Engineering Samples)
- Module, API, and Tools not fully qualified and tested
- Hardware und built-in firmware is not fully qualified and tested and subject to changes
- Final Status of pinout and mechanical dimensions

### Customer Documentation

- STM300 User Manual (Preliminary V0.80)
- STM300 Data Sheet (Jan 2010)
- Dolphin Core Description V0.70
- EO3000I\_API\_V1.6.0.0 (API Specification)

### Spec Restrictions

- 868 MHz radio performance
  - Near field problem: Minimum distance between transmitter and receiver must be 1m
  - Receive sensitivity typ. -94 dBm
  - Transmit power steps: only +6 dBm supported currently
- Application note for antenna layout is preliminary
- Current consumption

Operation Mode	Current consumption measured typ. at 25°C
Receive	31 mA
TX 6dBm 50% Dutycycle	26 mA
CPU-Mode	4 mA
XTAL on, CPU Standby	1.4 mA
Deep-Sleep VDD=3,3V	220 nA
Flywheel Sleep Mode	720 nA
ShortTermSleep	10 µA
Analogue Measurement	14 mA

- DAC: small offset, inaccuracy at small values, not uniformly continuous
- VDD current peak of several 100mA for 2-4 µs after powerup/wakeup
- Internal pull down not working
- Digital Inputs: pull ups must be enabled for IOVDD<2,5V – will be configured automatically in Dolphin Studio

### Other known Restrictions

- Labels not temperature resistant (might be problem for reflow soldering)
- Tape & Reel packaging not yet available



### Important application notes

- RVDD: If RVDD is used in an application circuit a serial ferrite bead shall be used and wire length should be as short as possible (<3cm). The following ferrite beads have been tested: 74279266 (0603), 74279205 (0805) from Würth. During radio transmission and reception only small currents may be drawn ( $I < 100\mu\text{A}$ ). Pulsed current drawn from RVDD has to be avoided. If pulsed currents are necessary, sufficient blocking has to be provided.

## Errata and Workarounds

### UART hangs up (transmitting) or loses bytes (receiving) during full duplex operation

#### Description

Under certain circumstances it is possible that an UART transmit or receive interrupt is lost. This only happens during full duplex (simultaneous transmitting and receiving) serial operation. The transmit and receive interrupt flags are bit-addressable in the 8051 SFR (special function register) space and are located in the same 8 bit register. From software perspective clearing of an interrupt flag is an atomic operation, but on the hardware side, a read-modify-write occurs (since the register is a single 8-bit unit). The problem now occurs when a new transmit/receive interrupt occurs (respective interrupt flag is set) between the read and write cycles.

#### There are two possible scenarios resulting in two different effects:

1. During clearing of a receive interrupt flag the transmit interrupt flag is set (TI occurs within 2 clock cycles after an RI=0 instruction). Then the transmit interrupt flag is accidental cleared and transmitting of further bytes will hang up.
2. During clearing of a transmit interrupt flag the receive interrupt flag is set (RI occurs within 2 clock cycles after an TI=0 instruction). Then the receive interrupt flag is accidental cleared and the received byte will be lost.

#### Workaround

##### 1. Use half duplex communication

If possible use half duplex communication. This will completely avoid the behaviour described above.

##### 2. Avoid hanging up of transmitting bytes by adding a timeout mechanism

The Dolphin API (>= V1.4.0.0) has implemented a timeout mechanism using the system scheduler (timer interrupt). With every 1 ms tick the scheduler probes if UART transmission is ongoing and whether the communication was locked or not. If it was locked the scheduler unlocks this manually and the error log counter ERR\_UART\_TX\_INT\_LOST is incremented.

This will result in an inter-byte delay between 1 ms (best case) and 4 ms (worst case) of the transmitted bytes.

If the Application is using the API System log feature, it is possible to detect this unlock events (for more info read EO30001\_API.chm->Error Diagnostic->System Log).

#### Note:

This API workaround does **not** apply to ultra low power applications where the scheduler is not running!

##### 3. Handling of lost received bytes

There is no workaround completely avoiding losing a received byte.

The Dolphin API (>= V1.4.0.0) has implemented a mechanism where a newly received byte is also detected under the error condition if the new byte has a different value than the previous one (does not work if the bytes are identical). In such case the ERR\_UART\_RX\_INT\_LOST is incremented.

If the Application is using the API System log feature, it is possible to detect this rescue events (for more info read EO30001\_API.chm->Error Diagnostic->System Log).

Additionally using of a protocol, e.g. ESP2 (using the Serial Telegram Checksum) or ESP3 (using the Packet CRC8), allows to detect lost bytes and to request and resend the data.