



## STATUS SHEET

Type: TCM 320C  
Step Code: DA  
Date: 08. April 2010

---

### Product Status

- Series production

### Customer Documentation

- TCM3xx User Manual V1.00
- TCM3xy Data Sheet March 2010
- Dolphin Core Description V0.80
- EO3000I\_API\_V2.0.0.0 (API Manual)

### Spec Restrictions

- DAC: small offset, inaccuracy at small values, not uniformly continuous
- VDD current peak of several 100mA for 2-4  $\mu$ s after powerup/wakeup

## Errata and Workarounds

### Malfunction of internal pull-downs at PROG\_EN and RESET

#### Description:

Under certain circumstances it can happen that the internal pull-downs at PROG\_EN and RESET are not working properly. In case the pull-down at PROG\_EN does not work it can happen, e.g. under bad EMI or temperature conditions, that at start-up of the module the bootloader jumps into programming mode instead of operating mode. The module then hangs in programming mode and a reset or new start-up is required. A malfunction of the pull-down at RESET pin can lead to an unwanted reset of the module.

#### Workaround:

Please connect an external 10k $\Omega$  pull-down resistor to these pins!

## 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 accidentally 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 accidentally 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.