Errata Sheet

MOTIX™ TLE989x/8x - AK step

Overview

This document lists the errata of the TLE989x and TLE988x family.

It is strongly recommended that the device behavior and proposed workarounds are considered in the application.

Table 1 Reference documents

Document type	Document reference	Issue date	Number
Datasheet	Infineon-TLE989x_8x-DS-v01_10-EN	2023-06-19	Z8F80164852
User manual	Infineon-TLE988x_9x-UM-v01_00-EN	2023-05-16	Z8F69685615
Firmware UM	Infineon-TLE988x_9x-FWUM-v01_00-EN	2023-05-16	Z8F80177275

Table 2 List of affected products

Device	Orderable part number (OPN)	Design step
TLE9893-2QKW62S	TLE98932QTW62SXUMA1	AK
TLE9893QKW62S	TLE9893QKW62SXUMA1	AK
TLE9893-2QTW62S	TLE98932QTW62SXUMA1	AK
TLE9891-2QTW61	TLE98912QTW61XUMA1	AK
TLE9891-2QTW60	TLE98912QTW60XUMA1	AK
TLE9893-2QTA62S	TLE98932QTA62SXUMA1	AK
TLE9893-2QTA62	TLE98932QTA62XUMA1	AK
TLE9891QTA61	TLE9891QTA61XUMA1	AK
TLE9883-2QTW62S	TLE98832QTW62SXUMA1	AK
TLE9881-2QTW60	TLE98812QTW60XUMA1	AK
TLE9883QTA62	TLE9883QTA62XUMA1	AK

Table of contents

Table of contents **2**
1 Product errata **3**
1.1 Limited support of UART0/1 for LIN (0000064543-770) 3
1.2 SYSTICK prescaler not mentioned in user manual (0000064543-769) 10
1.3 VCAN comment in the data sheet (0000064543-768) 11
1.4 PLL accumulated jitter note in the data sheet (0000064543-767) 12
1.5 Default UBSL size in the firmware Usermanual (0000060210-443) 14
2 Revision history **15**

1 Product errata

This chapter lists the errata of the referenced products and documentation.

1.1 Limited support of UART0/1 for LIN (0000064543-770)

Introduction

The support of UART0/1 to use the device in a LIN application, i.e. with an external LIN transceiver, is limited.

Description

The UART module of the TLE989x/8x has the following hardware features to handle LIN frames:

- BREAK field detection, indicated in the status bit LINST.BRK. The interrupt status is indicated in UARTx.IS. If the feature is enabled, the user software can use this event to start the UART reception
- End of SYNCH field detection, indicated in the status bit IS.EOFSYN, it issues an interrupt request if enabled; the user software can use this event to verify the baud rate or adjust the baud rate accordingly
- SYNCH error detection, indicated in the status bit IS.ERRSYN, it issues an interrupt request if enabled; the user software can use this event to disable the reception of LIN frames.

The features described above can only be used once, i.e. with the first received LIN message. The detection hardware is not released after clearing the status flags.

Workaround

The handling and detection of the BREAK and SYNCH field must be implemented using the Timer20 or Timer 21 External Interrupt and UART0 or UART1 Receive Interrupt as well as an initial configuration of these modules.

This workaround requires BREAK and SYNCH field detection to be implemented in the Timer20 external interrupt as well as the UART0 receive interrupt.

The detection mechanism can be integrated in the general detection mechanism for PID field, data, and checksum. The proposed detection mechanism includes only the detection of a BREAK or SYNCH field, it does not include the PID field detection, message handling, checksum verification or error handling.

In the following pages, SW examples are proposed as workaround.

The detection mechanism requires global variables to determine its state and configuration.

Code example

```
**********************************************************************************************************************

enum LINStateMachine {Idle, SYNCH; PID; Message, Cksum}; // create state machine states

enum LINStateMachine LINSts = Idle; //set default detection state machine to Idle

volatile uint16 CapTbit; //variable to write captured Timer20 value to

uint16 BrkLgthLo =X; //set initial shortest accepted BREAK field length

uint16 BrkLgthUp =Y; //set initial longest accepted BREAK field length

uint16 BF_reload; //variable recalculate BREAK field length

static uint32 SYNCHMeasure = 0; //variable to count edges during SYNCH field detection

volatile uint16 SYNCHLength; //variable to write captured SYNCH field length to

**********************************************************************************************************************
```

The interrupts as well as the specific interrupt node pointer in the NVIC must be enabled and the specific callbacks must be defined.

Code example

```
**********************************************************************************************************************

#define UART0_RI_CALLBACK UART_RX_ISR // UART receive interrupt callback is named UART_RX_ISR

#define T20_EXF2_CALLBACK LIN_ISR //T20 external interrupt callback is named LIN_ISR

…

CPU->NVIC_ISER.bit.IRQEN9 = 1; // enable NVIC node pointer 9 (mapped to T20)

CPU->NVIC_ISER.bit.IRQEN22 = 1; // enable NVIC node pointer 22 (mapped to UART0)

…

T20->CON1.bit.EXF2EN = 1; //enable Timer20 external interrupt

UART0->EIN.bit.RIEN = 1; //enable UART0 receive interrupt

…

**********************************************************************************************************************
```

Before the LIN communication can be established, the UART0 as well as the UART0 module must be initialized.

Code example

```
**********************************************************************************************************************

UART0->SCON.bit.REN = 0; // disable UART0 receiver

UART0->SCON.bit.RMOD = 1; // UART0 buffer is updated, even if UART0->IS.bit.RI is still set

UART0->SCON.bit.SM1 = 1; // set UART0 to mode 1

UART0->ISC.reg = 0x303; // clear all UART0 interrupt status flags


//set initial Baudrate example baudrate to 19200 Baud

UART0->BCON.bit.BR_PRE = 0; // set UART0 prescaler to 1

UART0->BCON.bit.BG_BR_VALUE = 130; // set UART0 reload value to 130

UART0->BCON.bit.BG_FD_SEL = 7; // set UART0 fractional divider to 7

UART0->BCON.bit.BR_R = 1; // enable Baudrate generation


T20->CON.bit.TR2 = 0; // stop Timer20

T20->CNT.reg = 0u; // clear count value of Timer20

T20->CON.bit.CP_RL2 = 1u; // set Timer20 to capture mode

T20->CON.bit.C_T2 = 0u; //set Timer20 to timer

T20->CON.bit.EXEN2 = 1u; // enable Timer20 external events

T20->MOD.bit.EDGESEL = 1u; // set rising edge to triggerTimer20 external event


T20->MOD.bit.PREN = 1; // enable Timer20 prescaler

T20->MOD.bit.T2PRE = 2u; // set Timer20 prescaler to 4

T20->MOD.bit.T2RHEN = 1u; // enable external start for Timer20

T20->MOD.bit.T2REGS = 0u; // set faling edge to trigger Timer20 external start

**********************************************************************************************************************
```

The Timer20 external interrupt can be used to implement the LIN state machine to detect header fields as well as messages and checksums.

Code example

```
**********************************************************************************************************************

void LIN_ISR(void){
        switch (LINSts){
                case Idle:{
                        CapTbit = T20->RC.reg; //write T20 timer timer value to CapTbit
                        T20->CON.bit.TR2 = 0; // stop Timer20
                        T20->CNT.reg = 0u; // clear count value of Timer20
                        if (CapTbit >= BrkLgthLo && CapTbit <= BrkLgthUp){ //check captured BREAK field length
                                LINSts = SYNCH; //captured BREAK field is acceptable, next must be SYNCH field
                                SYNCHMeasure = 0; //clear variable for SYNCH field detection
                                UART0->SCON.bit.REN = 1; // enable UART0 receiver
                        }break;
                }
                case SYNCH:{
                        T20->MOD.bit.EDGESEL = 0u; // set falling edge to triggerTimer20 external event
                        SYNCHMeasure++; //increment SYNCH bit faling egde counter
                        if(SynchMeasure == 5){
                                SYNCHLength = T20->RC.reg; //write T20 timer timer value to CapTbit
                                T20->CON.bit.TR2 = 0; // stop Timer20
                                T20->CNT.reg = 0u; // clear count value of Timer20
                                T20->MOD.bit.EDGESEL = 1u; // set rising edge to triggerTimer20 external event
                                BF_reload = (SynchLength - 16) >> 5; //reduce captured 16-bit to 11-bit
                                UART1->BCON.bit.BG_BR_VALUE = BG_reload; //change UART1 baudrate value
                                BrkLgthLo = BF_reload * 11; // change lower break field detection threshold
                                BrkLgthUp = BF_reload * 18; // change upper break field detection threshold
                                LINSts = PID; //LIN slave is synchronized to SYNCH field, next must be PID field
                        }break;
                }
```

```
            case Message:{
                CapTbit = T20->RC.reg; //write T20 timer timer value to CapTbit
                T20->CON.bit.TR2 = 0; // stop Timer20
                T20->CNT.reg = 0u; // clear count value of Timer20
                if (CapTbit >= BrkLgthLo && CapTbit <= BrkLgthUp){ //check captured BREAK field length
                    LINSts = SYNCH; //BREAK field detected, slave must synchronize
                    SYNCHMeasure = 0; //clear variable for SYNCH field detection
                    UART0->SCON.bit.REN = 1; // enable UART0 receiver
            …
                }break;
…}
**********************************************************************************************************************
```

The UART0 receive interrupt can be used to support the detection of the SYNCH field.

Code example

```
**********************************************************************************************************************

void UART_RX_ISR(void){
        switch (LINSts){
                case SYNCH:{
                        if (UART1->RXBUF.reg == 0x155){ // correct SYNCH field detected
                                T20->MOD.bit.EDGESEL = 1u; // set rising edge to triggerTimer20 external event
                                LINSts = PID; // SYNCH field was correctly received, next field is PID
                …
                        }
                break;
        }
…}
```

For a productive LIN stack for TLE989x/8x and TLE8457 as external LIN transceiver, please contact IHR (www.ihr.de).

Planned fix

No fix is planned. Additional information is available from Infineon Technologies on request.

1.2 SYSTICK prescaler not mentioned in user manual (0000064543-769)

Description

Setting MCU->SYSTICK_CS.CLKSOURCE = 0, adds a fixed prescaler dividing the SYSTICK input clock (CPU_CLK) by 4. This is not mentioned in figure 77 of the user manual.

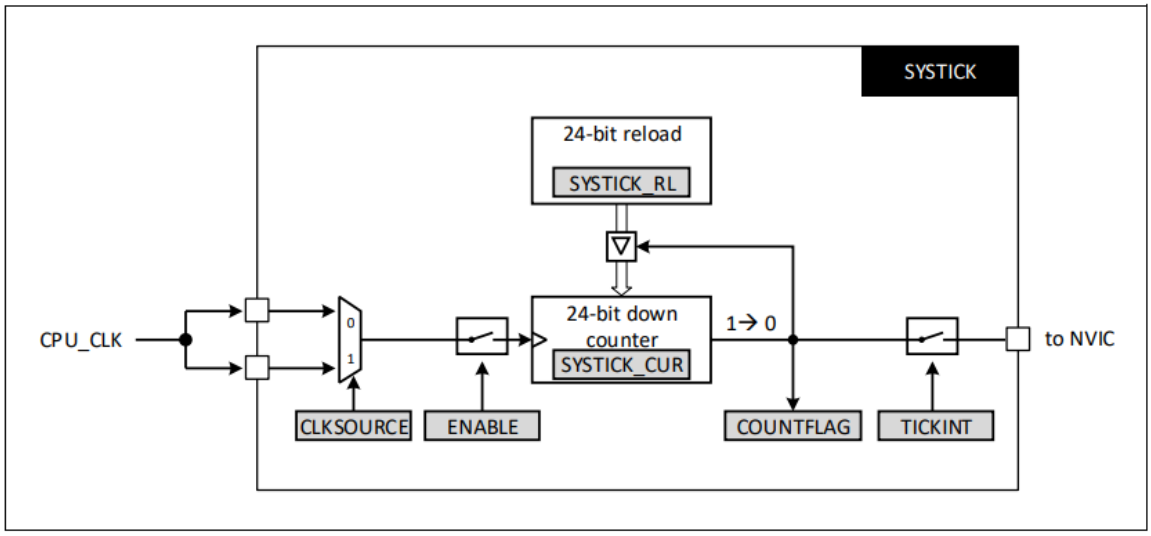

Figure 77 SYSTICK block diagram

Figure 1: figure 77 of TLE989x/8x user manual

Workaround

None.

Planned fix

Figure 77 will be modified accordingly in the next user manual update.

1.3 VCAN comment in the data sheet (0000064543-768)

Description

VCAN does not have to be connected necessarily to VDDP, as instead mentioned in table 3 of TLE989x/8x data sheet.

CAN interface

CANH	64	48	I/O	HiZ	CAN high bus	Connect resistor to CANL; Leave open if not used
CANL	63	47	I/O	HiZ	CAN low bus	Connect resistor to CANH; Leave open if not used
VCAN	61	45	P	–	Supply input for CAN transceiver	Connect to VDDP; Connect capacitor to GNDCAN
GNDCAN	62	46	P	–	Ground for CAN transceiver	Connect to GNDVSSP; Connect capacitor to VCAN

Figure 2: CAN interface in Table 3 of TLE989x/8x data sheet

Workaround

VCAN can be connected to a nominal supply voltage with range 4.75 V to 5.25 V, in order to fullfill specific accuracy requirements.

Planned fix

Table 3 will be modified accordingly in the next data sheet update.

1.4 PLL accumulated jitter note in the data sheet (0000064543-767)

Description

Notes in accumulated jitter with external oscillator describing the PLL0/1 divider settings (P_SCU_03_06 and P_SCU_04_06) are wrong. The setting for PDIV is 16 for both parameters.

Table 34 PLL0 (cont'd)

V_S = 5.5 V to 28 V, Grade 0 devices: T_j = -40°C to +175°C, Grade 1 devices: T_j = -40°C to +150°C, all voltages with respect to ground, positive current flowing into pin (unless otherwise specified)

Parameter	Symbol	Values			Unit	Note or Test Condition	Number
		Min.	Typ.	Max.			
Accumulated jitter with external oscillator	t_{jacc0}	-5	-	5	ns	1) accumulated over 300 cycles; @f_{PLL0} = 60 MHz, NDIV = 120, PDIV = 2; f_{XTAL} = 16 MHz	P_SCU_03_06
Lock-in time	t_{L0}	-	-	260	µs	1) from enable till lock	P_SCU_03_07

1) Not subject to production test, specified by design

Figure 3: PLL0 parameter Table 34 of TLE989x/8x data sheet

Table 35 PLL1

V_S = 5.5 V to 28 V, Grade 0 devices: T_j = -40°C to +175°C, Grade 1 devices: T_j = -40°C to +150°C, all voltages with respect to ground, positive current flowing into pin (unless otherwise specified)

Parameter	Symbol	Values			Unit	Note or Test Condition	Number
		Min.	Typ.	Max.			
VCO reference frequency range	f_{REF1}	0.8	1	1.27	MHz	1) fref1 = fin1/PDIV1	P_SCU_04_01
VCO frequency (tuning) range	f_{VCO1}	48	-	160	MHz	1)	P_SCU_04_02
Input frequency range	f_{in1}	4	-	40	MHz	1)	P_SCU_04_03
Output frequency range	f_{PLL1}	5	-	80	MHz	1)	P_SCU_04_04
Free-running frequency	$f_{VCOfree1}$	10	21.5	45	MHz	1)	P_SCU_04_05
Accumulated jitter with external oscillator	t_{jacc1}	-5	-	5	ns	1) accumulated over 300 cycles; @f_{PLL1} = 80 MHz, NDIV = 160, PDIV = 2; f_{XTAL} = 16 MHz	P_SCU_04_06
Lock-in time	t_{L1}	-	-	260	µs	1) from enable till lock	P_SCU_04_07

1) Not subject to production test, specified by design

Figure 4: PLL1 parameter Table 35 of TLE989x/8x data sheet

Workaround

None

Planned fix

Table 34 and 35 will be modified accordingly in the next data sheet update.

1.5 Default UBSL size in the firmware Usermanual (0000060210-443)

Description

The default UBSL size is 24kB.

Table 33 "Cmd 0x99 - UBSL size set" command frame parameter definition

Field	Description
Message type	UBSL size set command. Always set to 0x99.
UBSL size	Possible UBSL size options: 0x00: 4 kB 0x01: 8 kB 0x02: 12 kB 0x03: 16 kB 0x04: 20 kB 0x05: 24 kB 0x06: 28 kB (default) 0x07: 32 kB

Figure 5: UBSL size set command parameter definition table

Workaround

None

Planned fix

Table 33 will be modified accordingly in the next Firmware User Manual update.

2 Revision history

Document revision	Date	Description of changes
1.00	2024-03-07	First release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-03-07

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2024 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Z8F80576467

IMPORTANT NOTICE

The information contained in this document is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this document must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this document.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.