Document No.:D-SLD-M40_0020_01

Internal Specification

E2x/RLIN3 model for M40PF

(V1.0)

Summary :

This document describes the Detail Specification of E2x/RLIN3 model.

Relative Document

Reference Manuals				
No.	Title name	Document number	Description	Path
1	REQ-SLD-12010_M40PF_Common.ppt	REQ-SLD-12-010_v1.6	M40PF common requirement	Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/REQ/2012
2	E2xFCC1_RLIN3_target_specification.docx	-	Hardware manual of E2x/RLIN3	-
3	M40/Verify on SC-HEAP phase2 Requirement	REQ-SLD-12029	The M40 verify on SC_HEAP phase 2 requirement (version 1.3)	Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/REQ/2012/REQ-SLD-12029_M40_Verify_on_SCHEAP_ph2.ppt

Contents

Index of Figure

Index of Tables

1. Model summary

(1) As a Serial Communication Interface RLIN (Renesas Local Interconnect Network) of M40 platform, is a concept for low cost automotive networks, which complements the existing portfolio of automotive multiplex networks. RLIN will be the enabling factor for the implementation of a hierarchical vehicle network in order to gain further quality enhancement and cost reduction of vehicles. The standardization will reduce the manifold of existing low-end multiplex solutions and will cut the cost of development, production, service, and logistics in vehicle electronics.

(2) In this design, the following features are supported :

(2.1) A TLM target I/F in both AT and LT mode.

(2.2) handleCommand function with parameters such as DumpInterrupt, DumpStatInfo to control dumping the message during operation of RLIN model.

(2.3) High level pulse protocol for the interrupt request signals: transmit-data-empty interrupt and receive error interrupt.

(2.4) Support 32 bits bus-width socket.

2. Supported features

Table 2.1: Features of RLIN3 model

Feature item		Hardware manual	Hardware specification	Implementation (Yes/No)
LIN mode		*HW chapter* 17.2.1	master mode	Yes
			slave mode	Yes
			V1.3, V2.0, V2.1, V2.2	Yes
			SAEJ2062	Yes
UART mode		*HW chapter* 17.2.1	half duplex	Yes
			full duplex	Yes
Baud Rate Select Function		*HW chapter* 17.2.1	2400, 4800, 9600, 10417, 19200, 38400,115200 bps	Yes
Data byte count in the response		*HW chapter* 17.2.1	0 - 8 byte	Yes
Check sum type		*HW chapter* 17.2.1	Classic	Yes
			Enhanced	Yes
Three points majority sampling		*HW chapter* 17.2.1		No
Possibility to read Checksum generated by IP during transmission & received checksum during reception		*HW chapter* 17.2.1		Yes
Multi-bytes response transmission or reception is supported		*HW chapter* 17.2.1		Yes
Self-Test mode		*HW chapter* 17.2.1 *HW chapter* 17.9		Yes
Status Flag write control mode		*HW chapter* 17.2.1		No
LIN master mode	Variable frame structure	*HW chapter* 17.7	Break Low transmission length：13 - 28 Tbit	Yes
			Break High (delimiter) : 1 - 4 Tbit	Yes
			Interbyte Header space : 0 - 7 Tbit	Yes
			Response Space : 0 - 7 Tbit	Yes
			Interbyte Space 0 - 3 Tbit	Yes
	Transmission mode selection	*HW chapter* 17.7	Transmits Header and Response with single Start Command.	Yes
			transmits Header and Response by individual Start Commands	Yes
			Transmission/Reception of Wake-up possible in LIN Wake-up	Yes
			Automatic baud rate selection possible for Wake up mode	Yes

<table>
<tr><th colspan="2">Feature item</th><th>Hardware manual</th><th>Hardware specification</th><th>Implementation (Yes/No)</th></tr>
<tr><td rowspan="7"></td><td rowspan="6">Status</td><td rowspan="6"><i>HW chapter</i> 17.7</td><td>Successful Transmission</td><td>Yes</td></tr>
<tr><td>Header Transmission</td><td>Yes</td></tr>
<tr><td>Successful Reception</td><td>Yes</td></tr>
<tr><td>One Byte Reception</td><td>Yes</td></tr>
<tr><td>Error SUM</td><td>Yes</td></tr>
<tr><td>LIN Mode Status : Reset/Normal/Wake up</td><td>Yes</td></tr>
<tr><td>Controllable error status, detection</td><td><i>HW chapter</i> 17.7</td><td>Error list table:
<table>
<tr><th>No</th><th>Error Type</th><th>Detect</th><th>Occur</th></tr>
<tr><td>1</td><td>Bit Error</td><td>Yes</td><td>No</td></tr>
<tr><td>2</td><td>Physical Bus Error</td><td>Yes</td><td>No</td></tr>
<tr><td>3</td><td>Response Preparation Error</td><td>Yes</td><td>No</td></tr>
<tr><td>4</td><td>Timeout Error</td><td>Yes</td><td>No</td></tr>
<tr><td>5</td><td>Framing Error</td><td>Yes</td><td>No</td></tr>
<tr><td>6</td><td>SYNC field Error</td><td>Yes</td><td>No</td></tr>
<tr><td>7</td><td>Check Sum Error</td><td>Yes</td><td>No</td></tr>
<tr><td>8</td><td>Identifier Parity Error</td><td>Yes</td><td>No</td></tr>
</table></td><td>Yes</td></tr>
<tr><td rowspan="7">LIN slave mode</td><td rowspan="3">Variable frame structure</td><td rowspan="3"><i>HW chapter</i> 17.7</td><td>Break Low reception length : 9.5, 10.5Tbit</td><td>Yes</td></tr>
<tr><td>Response Space (RS)</td><td>Yes</td></tr>
<tr><td>Inter byte Space (IBS)</td><td>Yes</td></tr>
<tr><td rowspan="2">Header reception mode selection</td><td rowspan="2"><i>HW chapter</i> 17.7</td><td>Automatic baud rate detection mode</td><td>Yes</td></tr>
<tr><td>Fixed baud rate mode</td><td>Yes</td></tr>
<tr><td rowspan="2">Wake-up transmission/reception</td><td rowspan="2"><i>HW chapter</i> 17.7</td><td>Transmission/Reception of Wake-up possible in LIN Wake-up</td><td>Yes</td></tr>
<tr><td>Automatic baud rate selection possible for Wake up mode</td><td>Yes</td></tr>
</table>

Feature item		Hardware manual	Hardware specification	Implementation (Yes/No)
	Status	*HW chapter* 17.7	Successful Transmission	Yes
			Header Reception	Yes
			Successful Reception	Yes
			One Byte Reception	Yes
			Error SUM	Yes
			LIN Mode Status : Reset/Normal/Wake up	Yes
	Controllable error status, detection	*HW chapter* 17.7	Bit error	Yes
			ID Parity error	Yes
			Sync Field error	Yes
			Framing error	Yes
			Frame Timeout error/Response Timeout error	Yes
			Checksum error	Yes
			Response preparation error	Yes
UART mode	Full duplex communication	*HW chapter* 17.8		Yes
	Configurable Data length	*HW chapter* 17.8	7 / 8 / 9bits	Yes
	Configurable number of Stop bit	*HW chapter* 17.8	1 / 2bits	Yes
	Parity configuration	*HW chapter* 17.8	Even / Odd parity / 0 (parity bit is always "0") / None	Yes
	Parity generation/judgment switch	*HW chapter* 17.8	Disable/Enable	Yes
	Multi byte communication	*HW chapter* 17.8	possibility to handle up to 8 bytes with no CPU load	Yes
	Transmission/Reception inversion	*HW chapter* 17.8	No inversion / Inversion	Yes
	Bit order configuration	*HW chapter* 17.8	LSB / MSB first	Yes
	Status	*HW chapter* 17.8	Successful Transmission / Reception	Yes
			UART Transmission / Reception status	Yes
			ID match	Yes
			Expansion bit detection	Yes
			Error SUM	Yes
			UART mode state : Reset / Normal	Yes

Feature item		Hardware manual	Hardware specification	Implementation (Yes/No)
	Controllable Error status detection	*HW chapter* 17.8	Bit error	Yes
			Framing error	Yes
			Parity error	Yes
			Overrun error	Yes
	Baud Rate Select	*HW chapter* 17.8	Supports bit rate up to 5.33 Mbits/s	Yes
Interrupt	LIN mode	*HW chapter* 17.4	Successful Transmission interrupt request	Yes
			Successful Reception interrupt request	Yes
			Error Detection interrupt request	Yes
			Disable/Enable individually	Yes
	UART mode	*HW chapter* 17.4	Transmission interrupt request	Yes
			Reception completion interrupt request	Yes
			Status interrupt request	Yes
			pulse signal (clear automatically after 1 CLK_LSB clock)	Yes
			Can not disable	Yes

3. Block diagram

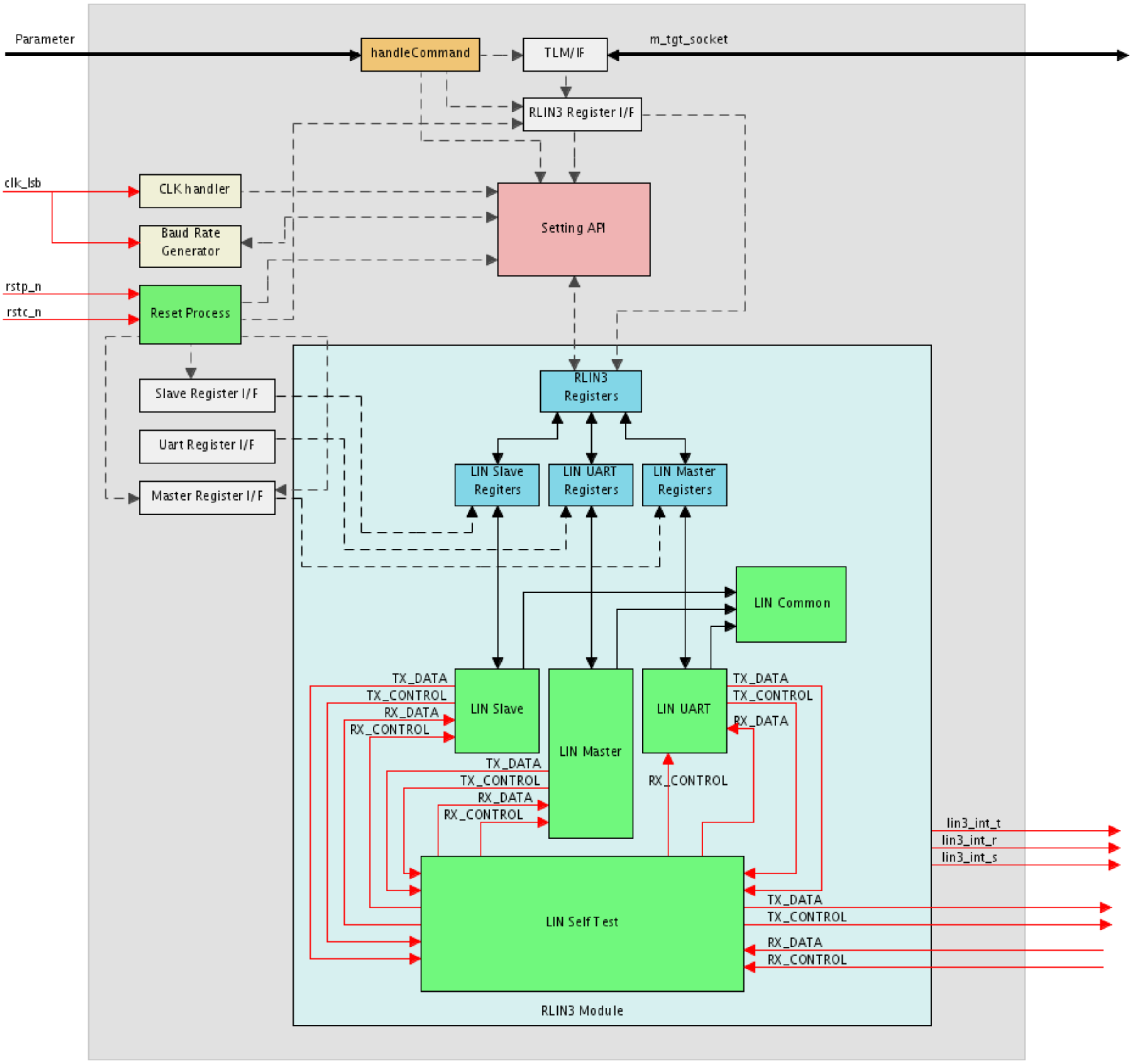


Figure 3.1: Block diagram of RLIN3 model

Color Explanation:

- 🟦 RLIN3 module
- 🟩 Sub Modules/Functions
- 🟥 API functions
- 🟦 Register
- 🟧 handleCommand functions
- 🟨 Clock functions
- ⬜ Interface

Symbol explanation :

Port	→
APIs call	⇢
Parameter setting	➡
Directly call	→
TLM socket	⟺

Explanation:

(1) RLIN3 model has three sub modules LIN Master, LIN Slave and UART corresponding three modes Master. Slave and Uart. Each module has a registers separately.

(2) Additionally, RLIN3 has Self Test module in LIN Master mode and LIN Slave mode which is implemented for Self Test mode. In the Self Test mode, the data input ports will be connected to data output ports for self test operations. The RLIN3 Self Test module controls this action refer to the section 7.43 for details.

(3) The RLIN3 register bank is used for setting by users. Registers in RLIN3 register bank can be updated value among LIN Master, LIN Slave and UART modes. Depended on mode configuration, data can be set/gotten between RLIN3 register bank and LIN Master/LIN Slave/UART private registers.

(4) Functions of LIN Master/LIN Slave/UART modes (LIN Common) can use data in LIN Master/LIN Slave/UART private registers correspondingly. These functions are contained in LIN Master, LIN Slave and LIN Uart modules.

(5) Data are transferred to RLIN3 via m_tgt_sockets[0] port, through TML I/F and Register I/F they are stored at the general register bank. These data can be updated to LIN Master/LIN Slave/UART private registers corresponding to mode configuration.

(6) handleCommand is used for setting output messages and the period of clock. The setting for handleCommand is done through outside parameter.

(7) Registers in RLIN3 register bank set up control variables, transfer time, transmit and receive conditions and interrupt conditions via setting APIs.

(8) When *rstp_n* signal or *rstc_n* signal de-asserted, registers, ports and all control variables are reseted. In this model, the *rstp_n* is named as *preset_n*

(9) CLK handler is used for setting the period of APB bus clock.

(10) A baud rate generator is used for setting frequency of serial transfer clock and output serial transfer baud rate clock frequency.

(11) The Self Test module will transfer data to RLIN3 module to handle main operations if Self Test mode is selected. Refer to section 7.43 for main self test process operations.

(12) When RLIN3 transmits/receives data, a transaction information also is sent via TX_CONTROL/RX_CONTROL ports.

4. List of implemented registers

Table 4.1: List of implemented registers of RLIN3 model

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
1	LWBR *HW chapter* *17.3.2.1* *17.3.3.1* *17.3.4.1*	H'00	Master	[0]	LWBR0	0: Baud rate clock is based on System Clock configuration in Wake-up mode 1: Baud rate clock source is automatically set to "fa" in Wake-up mode.	Yes
			Master/ Slave/UART	[3:1]	LPRS	These bits configure the prescaler "000": 1/1 "001": 1/2 "010": 1/4 "011": 1/8 "100": 1/16 "101": 1/32 "110": 1/64 "111": 1/128	Yes
				[7:4]	NSPB	These bits configure the value for number of samples in 1 Bit time period. "0000": 16 samples per bit "0001": Prohibited "0010": Prohibited "0011": 4 samples per bit "0100": Prohibited "0101": 6 samples per bit "0110": 7 samples per bit "0111": 8 samples per bit "1000": 9 samples per bit "1001": 10 samples per bit "1010": 11 samples per bit "1011": 12 samples per bit "1100": 13 samples per bit "1101": 14 samples per bit "1110": 15 samples per bit "1111": 16 samples per bit	Yes
2	LBRP0 *HW chapter* *17.3.2.2* *17.3.3.2* *17.3.4.2*	H'00	LIN Master	[7:0]	LBRP0	Baud Rate prescaler 0. The value in this register is used to control the "fa", "fb" and "fc" baud rate source clock frequencies.	Yes
			LIN Slave or	[7:0]	BRP	The frequency division value for the BRP counter.	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
			UART				
3	LBRP1 *HW chapter* *17.3.2.3* *17.3.3.2* *17.3.4.2*	H'00	LIN Master	[7:0]	LBRP1	Selectable value: 00h ˜ Ffh Baud Rate prescaler 1. The value in this register is used to control the "fd" baud rate source clock frequency.	Yes
			LIN Slave or UART	[15:8]	BRP	The frequency division value for the BRP counter.	Yes
4	LSTC *HW chapter* *17.3.2.3* *17.3.3.4*	H'00	Master/Slave	[0]	LSTM	0: Self Test mode disabled 1: Self Test mode enabled	Yes
				[7:1]	LSTME	The test mode key values for configuring the RLIN3 module in Test mode.	Yes
5	LMD *HW chapter* *17.3.2.5* *17.3.3.4* *17.3.4.3*	H'00	Master/Slave/UART	[1:0]	LMD	00: LIN Master mode 01: UART mode 10: LIN Slave mode with automatic baud rate detection 11: LIN Slave mode with fixed baud rate	Yes
			Master	[3:2]	LCKS	00: fa 01: fb 10: fc 11: fd	Yes
			Master/Slave	[4]	LIOS	0: Not supported, setting is prohibited. 1: RLIN3 transmission interrupt, RLIN3 successful reception interrupt, and RLIN3 status interrupt are used.	Yes
			Master/Slave/UART	[5]	LRDNFS	0: 3-bit majority voting logic for sampling RX data is enabled. 1: 3-bit majority voting logic for sampling RX data is disabled.	No
6	LBFC *HW chapter* *17.3.2.6* *17.3.3.5* *17.3.4.4*	H'00	LIN Master	[3:0]	BLT	0h = Break Low width is 13 Tbits 1h = Break Low width is 14 Tbits : : : : Fh = Break Low width is 28 Tbits	Yes
				[5:4]	BDT	00: Break Delimiter width is 1 Tbit 0 1: Break Delimiter width is 2 Tbits 10: Break Delimiter width is 3 Tbits 11: Break Delimiter width is 4 Tbits	Yes
			LIN	[0]	LBLT	0 = Break Low width is 9.5 Tbits or	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
			Slave			10 Tbits 1 = Break Low width is 10.5 Tbits or 11 Tbits	
			UART	[0]	UBLS	0: UART 8-bit communication 1: UART 7-bit communication	Yes
				[1]	UBOS	0: LSB first 1: MSB first	Yes
				[2]	USBLS	0: Stop Bit 1 bit 1: Stop Bit 2 bits	Yes
				[4:3]	UPS	00:Parity Disabled 01:Even Parity 10:0 Parity 11:Odd Parity	Yes
				[5]	URPS	0: Without inversion 1: With inversion	Yes
				[6]	UTPS	0: Without inversion 1: With inversion	Yes
7	LSC *HW chapter 17.3.2.7 17.3.3.6 17.3.4.5*	H'00	Master/ Slave	[2:0]	IBHS	0h: 0 Tbits 1h: 1 Tbit 2h: 2 Tbits 3h: 3 Tbits 4h: 4 Tbits 5h: 5 Tbits 6h: 6 Tbits 7h: 7 Tbits	Yes
			Master/ Slave/UART	[5:4]	IBS	00b: 0 Tbits 01b: 1 Tbit 10b: 2 Tbits 11b: 3 Tbits	Yes
8	LWUP *HW chapter 17.3.2.8 17.3.3.7*	H'00	Master/ Slave	[7:4]	WUTL	0h = Low pulse transmission 1Tbit 1h = Low pulse transmission 2Tbits : : : Fh = Low pulse transmission 16Tbits	Yes
9	LIE *HW chapter 17.3.2.9 17.3.3.8*	H'00	Master/ Slave	[0]	FTCIE	0: LIN Response or LIN Wake-up successful Transmission Interrupt Disabled 1: LIN Response or LIN Wake-up successful transmission Interrupt Enabled	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
				[1]	FRCIE	0: LIN Response or LIN Wake-up Successful Reception Interrupt Disabled 1: LIN Response or LIN Wake-up Successful Reception Interrupt Enabled	Yes
				[2]	ERRIE	0: Error Detection Interrupt Disabled 1: Error Detection Interrupt Enabled	Yes
				[3]	SHIE	0: LIN Successful Header interrupt disabled 1: LIN Successful Header interrupt enabled	Yes
10	LEDE *HW chapter 17.3.2.10 17.3.3.9 17.3.4.6*	H'00	LIN Master	[0]	BERE	0: Bit Error Detection Disabled 1: Bit Error Detection Enabled	Yes
				[1]	PBERE	0: Physical Bus Error Detection Disabled 1: Physical Bus Error Detection Enabled	Yes
				[2]	FTERE	0: Frame / Response Timeout Error Detection Disabled 1: Frame / Response Timeout Error Detection Enabled	Yes
				[3]	FERE	0: Framing Error Detection Disabled 1: Framing Error Detection Enabled	Yes
				[7]	LTES	0: Frame Timeout error is selected 1: Response Timeout error is selected	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
			LIN Slave	[0]	BERE	0: Bit Error Detection Disabled 1: Bit Error Detection Enabled	Yes
				[2]	TERE	0: Frame / Response Timeout Error Detection Disabled 1: Frame / Response Timeout Error Detection Enabled	Yes
				[3]	FERE	0: Framing Error Detection Disabled 1: Framing Error Detection Enabled	Yes
				[4]	SFERE	0: SYNC field Error Detection Disabled 1: SYNC field Error Detection Enabled	Yes
				[6]	IPERE	0: Identifier Parity Error Detection Disabled 1: Identifier Parity Error Detection Enabled	Yes
				[7]	LTES	0: Frame Timeout error is selected 1: Response Timeout error is selected	Yes
			UART	[0]	BERE	0: Bit Error Detection Disabled 1: Bit Error Detection Enabled	Yes
				[2]	OERE	0: Overrun Error Detection Disabled 1: Overrun Error Detection Enabled	Yes
				[3]	FERE	0: Framing Error Detection Disabled 1: Framing Error Detection Enabled	Yes
11	LCUC *HW chapter* *17.3.2.11* *17.3.3.10* *17.3.4.7*	H'00	Master/ Slave/UART	[0]	OM0	0: SW Reset request is active. 1: SW Reset request is inactive	Yes
			Master/ Slave	[1]	OM1	0: LIN Wake-up mode enabled 1: LIN Normal Communication mode enabled	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
12	LTRC *HW chapter* *17.3.2.12* *17.3.3.11* *17.3.4.8*	H'00	LIN Master	[0]	FTS	0: Frame Communication is stopped 1: Frame Communication is started	Yes
				[1]	RTS	0: Response transmission or reception 1: Response transmission or reception start	Yes
			LIN Slave	[0]	FTS	0: Frame Communication is stopped 1: Frame Communication start is enabled	Yes
				[1]	RTS	0: Response transmission or reception 1: Response transmission or reception start	Yes
				[2]	LNRR	0: Response for received ID is present 1: Response for received ID is absent	Yes
			UART	[1]	RTS	0: UART buffer mode transmission is stopped 1: UART buffer mode start is enabled	Yes
13	LMST *HW chapter* *17.3.2.13* *17.3.3.12* *17.3.4.9*	H'00	Master/ Slave/UART	[0]	OMM0	0: Module is in Reset state. 1: Module is not in Reset state	Yes
			Master/ Slave	[1]	OMM1	0: LIN Wake-up mode enabled 1: LIN Normal Communication mode enabled	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
14	LST *HW chapter 17.3.2.14 17.3.3.13 17.3.4.10*	H'00	LIN Master	[0]	FTC	0: Response or Wake-up transmission not completed 1: Response or Wake-up transmitted successfully	Yes
				[1]	FRC	0: Response or Wake-up reception not completed 1: Response or Wake-up received successfully	Yes
				[3]	ERR	0: No error detected in LIN mode 1: Errors detected in LIN mode	Yes
				[6]	D1RC	0: One Byte reception not completed 1:One Byte reception completed	Yes
				[7]	HTRC	0: LIN Header (Tx or Rx) not completed 1: LIN Header (Tx or Rx) completed successfully	Yes
			LIN Slave	[0]	FTC	0: Response or Wake-up transmission not completed 1: Response or Wake-up transmitted successfully	Yes
				[1]	FRC	0: Response or Wake-up reception not completed 1: Response or Wake-up received successfully	Yes
				[3]	ERR	0: No error detected in LIN mode 1: Errors detected in LIN mode	Yes
				[6]	D1RC	0: One Byte reception not completed 1:One Byte reception completed	Yes
				[7]	HTRC	0: LIN Header (Tx or Rx) not completed 1: LIN Header (Tx or Rx) completed successfully	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
			UART	[0]	FTC	0: Frame transmission not completed 1: Frame transmitted successfully	Yes
				[3]	ERR	0: No changes in Error status detected in UART mode. 1: Change in Error status detected in UART mode.	Yes
				[4]	UTS	0: A transmit operation is not in progress. 1: A transmit operation is in progress.	Yes
				[5]	URS	0: A receive operation is not in progress. 1: A receive operation is in progress.	Yes
15	LEST *HW chapter 17.3.2.15 17.3.3.14 17.3.4.11*	H'00	LIN Master	[0]	BER	0: Bit error not detected 1: Bit error detected	Yes
				[1]	PBER	0: Physical Bus error not detected 1: Physical Bus error detected	Yes
				[2]	FTER	0: LIN Timeout error not detected 1: LIN Timeout error detected	Yes
				[3]	FER	0: LIN Framing error not detected 1: LIN Framing error detected	Yes
				[5]	CSER	0: LIN Checksum error not detected 1: LIN Checksum error detected	Yes
				[7]	RPER	0: Response Preparation error not detected 1: Response Preparation error detected	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
			LIN Slave	[0]	BER	0: Bit error not detected 1: Bit error detected	Yes
				[2]	TER	0: LIN Timeout error not detected 1: LIN Timeout error detected	Yes
				[3]	FER	0: LIN Framing error not detected 1: LIN Framing error detected	Yes
				[4]	SFER	0: SYNC field Error not detected 1: SYNC field Error detected	Yes
				[5]	CSER	0: LIN Checksum error not detected 1: LIN Checksum error detected	Yes
				[6]	IPER	0: Identifier Parity Error not detected 1: Identifier Parity Error detected	Yes
				[7]	RPER	0: Response Preparation error not detected 1: Response Preparation error detected	Yes
			UART	[0]	BER	0: Bit error not detected 1: Bit error detected	Yes
				[2]	OER	0: UART Overrun error not detected 1: UART Overrun error detected	Yes
				[3]	FER	0: UART Framing error not detected 1: UART Framing error detected	Yes
				[4]	EXBT	0: Expansion bit is not detected 1: Expansion bit is detected	Yes
				[5]	IDMT	0: Received byte does not match ID value 1: Received byte matches ID value.	Yes
				[6]	UPER	0: UART Parity Error not detected 1: UART Parity Error detected	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
16	LDFC *HW chapter 17.3.2.16 17.3.3.15 17.3.4.12*	H'00	LIN Master	[3:0]	RFDL	0h = 0byte + Checksum 1h = 1byte + Checksum 2h = 2bytes + Checksum : : : 8h = 8bytes + Checksum Other are prohibited	Yes
				[4]	RFT	0: Response Reception 1: Response Transmission	Yes
				[5]	CSM	0: Classic Checksum 1: Enhanced Checksum	Yes
				[6]	FSM	0: Frame Combined Mode 1: Frame Separate Mode	Yes
				[7]	LSS	0: Last Data group to be transmitted or received 1: Not the last data group	Yes
			LIN Slave	[3:0]	RFDL	0h = 0byte + Checksum 1h = 1byte + Checksum 2h = 2bytes + Checksum : : : 8h = 8bytes + Checksum Other are prohibited	Yes
				[4]	RCDS	0: Response Reception 1: Response Transmission	Yes
				[5]	LCS	0: Classic Checksum 1: Enhanced Checksum	Yes
				[7]	LSS	0: Last Data group to be transmitted or received 1: Not the last data group	Yes
			UART	[3:0]	MDL	0h = 9bytes 1h = 1byte 2h = 2bytes : : : 8h = 8bytes 9h = 9bytes Other are prohibited	Yes
				[5]	UTSW	0: Starts transmission immediately when multi-byte data transmission is requested 1: Delays starting of transmission until completion of stop bit of reception when multi-byte data transmission is requested	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
17	LIDB *HW chapter 17.3.2.17 17.3.3.16 17.3.4.13*	H'00	LIN Master	[5:0]	ID	value of ID to be transmitted within the ID Field	Yes
				[6]	IDP0	Value of the parity (P0) bit to be transmitted with ID	Yes
				[7]	IDP1	Value of the parity (P1) bit to be transmitted with ID	Yes
			LIN Slave	[5:0]	ID	value of ID in the ID Field	Yes
				[6]	IDP0	Value of the parity (P0) bit	Yes
				[7]	IDP1	Value of the parity (P1) bit	Yes
			UART	[7:0]	ID	Reference value of ID for comparison with received value	Yes
18	LCBR *HW chapter 17.3.2.18 17.3.3.17*	H'00	LIN Master/ Slave	[7:0]	CKSM	value of Checksum in the Response field	Yes
19	LUDB0 *HW chapter 17.3.4.14*	H'00	UART	[7:0]	UDB	value of UART data	Yes
20	LDBRn *HW chapter 17.3.2.19 17.3.3.18 17.3.4.15*	H'00	Master/ Slave/UART	[7:0]	LDBn	value of LIN / UART data	Yes
21	LUOER *HW chapter 17.3.4.16*	H'00	UART	[0]	UTOE	0:Stops transmission operation 1:Enables transmission operation	Yes
				[1]	UROE	0:Stops reception operation 1:Enables reception operation	Yes
22	LUOR1 *HW chapter 17.3.4.17*	H'00	UART	[0]	UEBE	0:Disables expansion bit operation 1:Enables expansion bit operation	Yes
				[1]	UEBDL	0:Selects expansion bit value "0" as expansion bit detection level. 1:Selects expansion bit value "1" as expansion bit detection level	Yes
				[2]	UEBDCE	0:No comparison 1:Compares UART 7bits/8bits/9bits Receive Data Register and LIN / UART Identifier Buffer Register	Yes

No.	Register	Default value	Mode	Bit	Bit Name	Description	Supported (Yes/No)
						when the level selected for UART Expansion Bit Detection Level Select Bit has been detected as the expansion bit	
				[3]	UTIGTS	0:Outputs transmission interrupt request upon transmission start 1:Outputs transmission interrupt request upon transmission completion	Yes
				[4]	UECD	0: Expansion bit comparison enable 1: Expansion bit comparison disable	Yes
23	LUTDR *HW chapter 17.3.4.18*	H'00	UART	[8:0]	UTD	Value of transmit data in UART mode	Yes
24	LURDR *HW chapter 17.3.4.19*	H'00	UART	[8:0]	URD	Value of received data in UART mode	Yes
25	LUWTDR *HW 17.3.4.20*	H'00	UART	[8:0]	UWTD	Value of transmit data in UART mode with STOP bit reception	Yes

- The following features are based on the hardware manual:

1. Address offset
2. Access Size
3. Initial value
4. Reserve bit

5. List of implemented ports

Table 5.1: List of implemented ports

HW port name	Model	I/O	Type	Initial	Description
CLK_LSB	*clk_lsb*	I	sc_dt::uint64	-	APB clock (Hz) LIN clock (Hz)
rstp_n	*preset_n*	I	bool	0	Asynchronous reset (Active 0)
rstc_n	*rstc_n*	I	bool	0	LIN reset (Active 0)
paddr	*m_tgt_sockets[0]*	I/O	tlm:tlm_target_socket	-	APB I/F
penable					
psel					
pwrite					
pstrb					
pwdata					
prdata					
pready					
INTRLIN3nUR0	*lin3_int_t*	O	bool	0	Transmit-start/end interrupt
INTRLIN3nUR1	*lin3_int_r*	O	bool	0	Receive-end interrupt
INTRLIN3nUR2	*lin3_int_s*	O	bool	0	Status interrupt
RLIN3nRX	*RX_CONTROL*	I	unit	-	Receive control
	RX_DATA	I	unit	-	Receive data
RLIN3nTX	*TX_CONTROL*	O	unit	0x00000108	Transmit control
	TX_DATA	O	unit	0xFFFFFFFF	Transmit data

5.1. TX_CONTROL and RX_CONTROL behavior

5.1.1. Data structure

The data structure of RX_CONTROL and TX_CONTROL is as the following table:

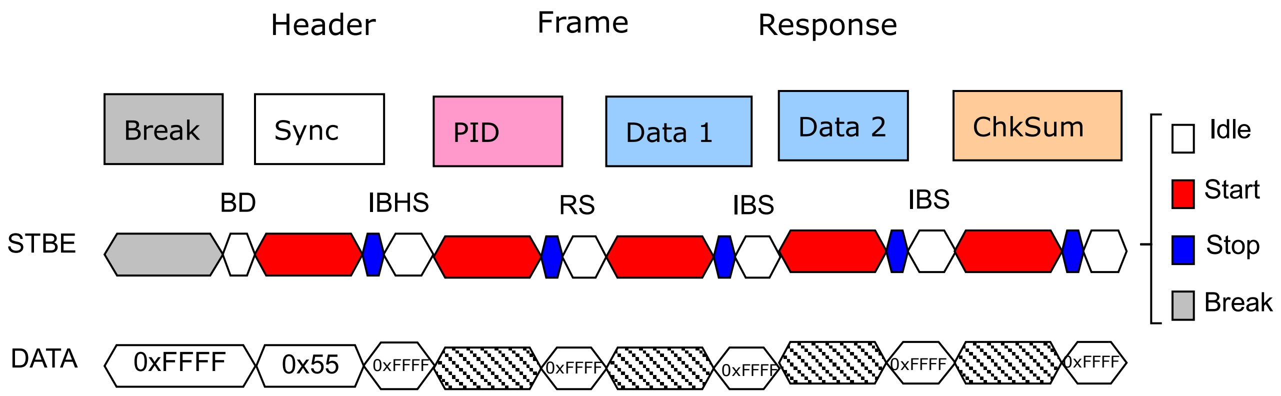
Table 5.2: The data structure of TX_CONTROL and RX_CONTROL

Bit	Definition	Initial value	Using in LIN mode	Using in UART mode	Explanation
[31:16]	BITT	0x0	Yes	Yes	Contains the time of bit on a transaction. The unit is 100ns. The bit time is calculated by (1/(baud rate))*10e7
[15]	TE	0x0	No	No	Transmitter is enable.
[14]	TC	0x0	No	No	Transmitter complete.
[13]	Reserved Bit	0x0	-	-	This bit is not used.

Bit	Definition	Initial value	Using in LIN mode	Using in UART mode	Explanation
[12:9]	NUM[3:0]	0x0	Yes	Yes	Number of byte data transmit.
[8]	DIR	0x1	Fixed to 1	Yes	The data direction: 0: Data is transferred MSB first. 1: Data is transferred LSB first.
[7:6]	STBE	0x0	Yes	Yes	Data strobe: 0x00: Idle 0x1: Start bit 0x2: Stop bit 0x3: Break Character
[5:4]	Reserved Bits	0x0	-	-	These bits are not used.
[3:0]	SIZE	0x8	Fixed to 8	Yes	The data size. The unit is bit.

5.1.2. Behaviors of serial I/F

The behaviors of LIN I/F are illustrated as figure below:



BD : break delimiter
IBHS : Inter-Byte Header Space
RS : Response Space
IBS : Inter-Byte Space

Figure 5.1: LIN I/F behaviors

Explanation :

(1) When RLIN3 sends data on the TX_DATA port, RLIN3 also sends current transaction information via TX_CONTROL port.

(2) A data frame consists of Header and Response. SYNC field and ID field in the Header have a start/stop bit. Data field and Check Sum field have a start/stop bit too.

(3) The start/stop bit information needs to be sent when SYNC field, ID field, Data field and Check Sum field are sent. RLIN3 sends each start/stop bit information via TX_CONTROL at the beginning of start bit, stop bit and the end of stop bit.

The behaviors of UART I/F are illustrated as figure below:

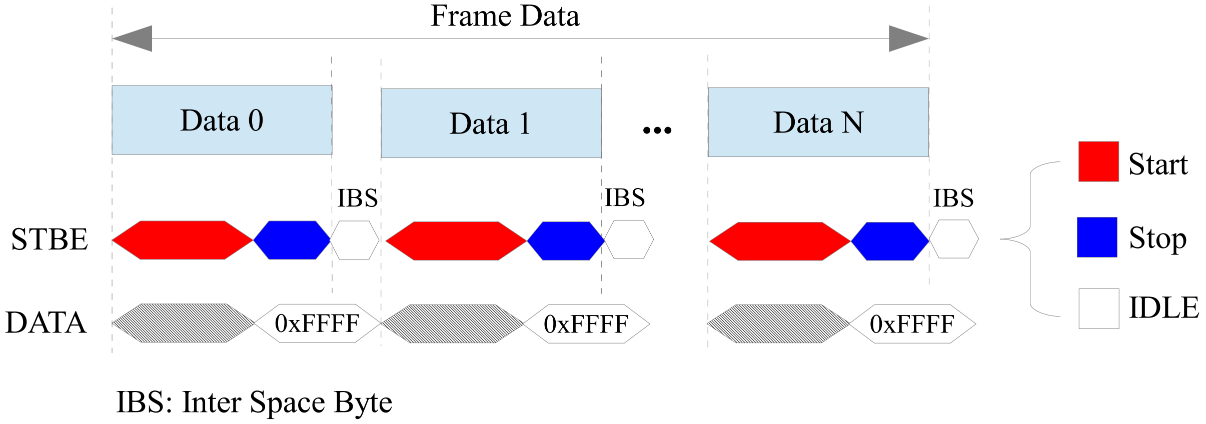


Figure 5.2: UART I/F behaviors

Explanation :

(1) In UART mode, RLIN3 sends frame data on the TX_DATA port and current transaction information on the TX_CONTROL port.

(2) A data can be 7bits / 8bits / 9bits. If data are 7bits, there is one parity bit is inserted into data. If data is 9bits, the ninth bit is expansion bit. Each data has a start bit and 1 or 2 stop bit.

(3) The information of start/stop bit is sent when Data field is sent. RLIN3 sends each information of start/stop bit via TX_CONTROL at the beginning of the start bit, stop bit and the end of stop bit.

5.2. CLK_LSB behavior

(1) The LIN clock *clk_lsb* is used to generate baud rate clock source for LIN communication. The table 5.3 describes the selection and the formula of baud rate clock in LIN Master/Slave/ UART mode.

(2) The baud rate clock source is selected by LCKS[1:0] bits. In master mode, if LCKS[1:0] = 0x0, 0x1, 0x2 and 0x3 the baud rate clock source is fa, fb, fc and fd which are calculated by the formula in the table. In Slave/UART mode, there is one baud rate clock source only that is fa.

(3) In LIN Wake-up mode, if LWBR.LWBR0 is set to 0, the clock specified in the LCKS bit of the RLN3nLMD register is used. (for LIN1.3)

(4) In LIN wake-up mode, if LWBR.LWBR0 is set to 1, the clock fa is used regardless of the setting in the LCKS bit of the RLN3nLMD register. (for LIN2.x)

Table 5.3: Baud rate behavior

Mode	LMD.LCKS[1:0]	Baud rate clock source
Master	0x0	fa = (clk_lin) / ((LBRP0 value)+1)
	0x1	fb = fa / 2
	0x2	fc = fa / 8
	0x3	fd = [(clk_lin) / ((LBRP1 value)+1)] / 2
Slave and UART	-	fa = (clk_lin) / ((BRP value)+1)

(*1) – is “don’t care”.

6. Direction for user

6.1. File structures

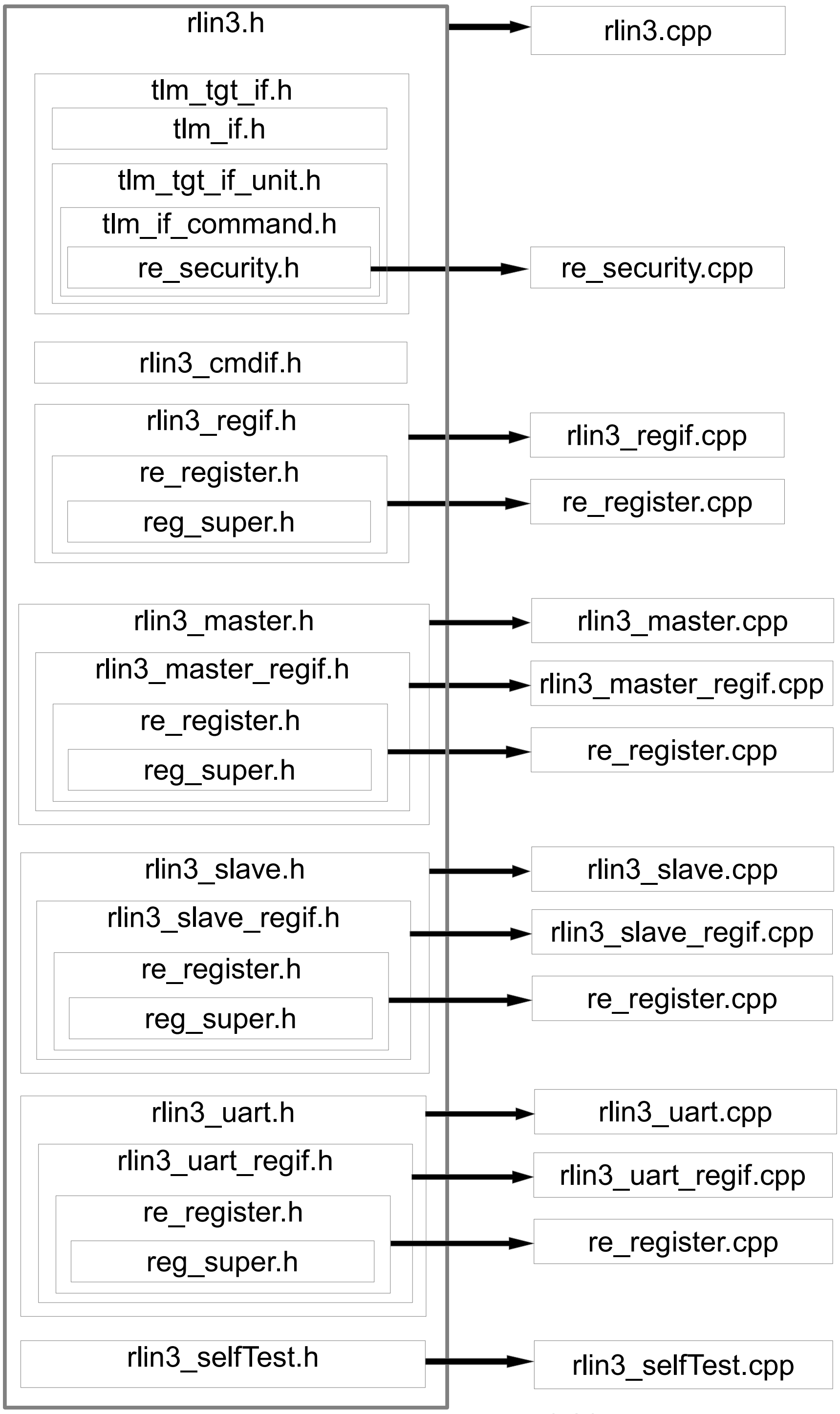


Figure 6.1: File structures (1/2)

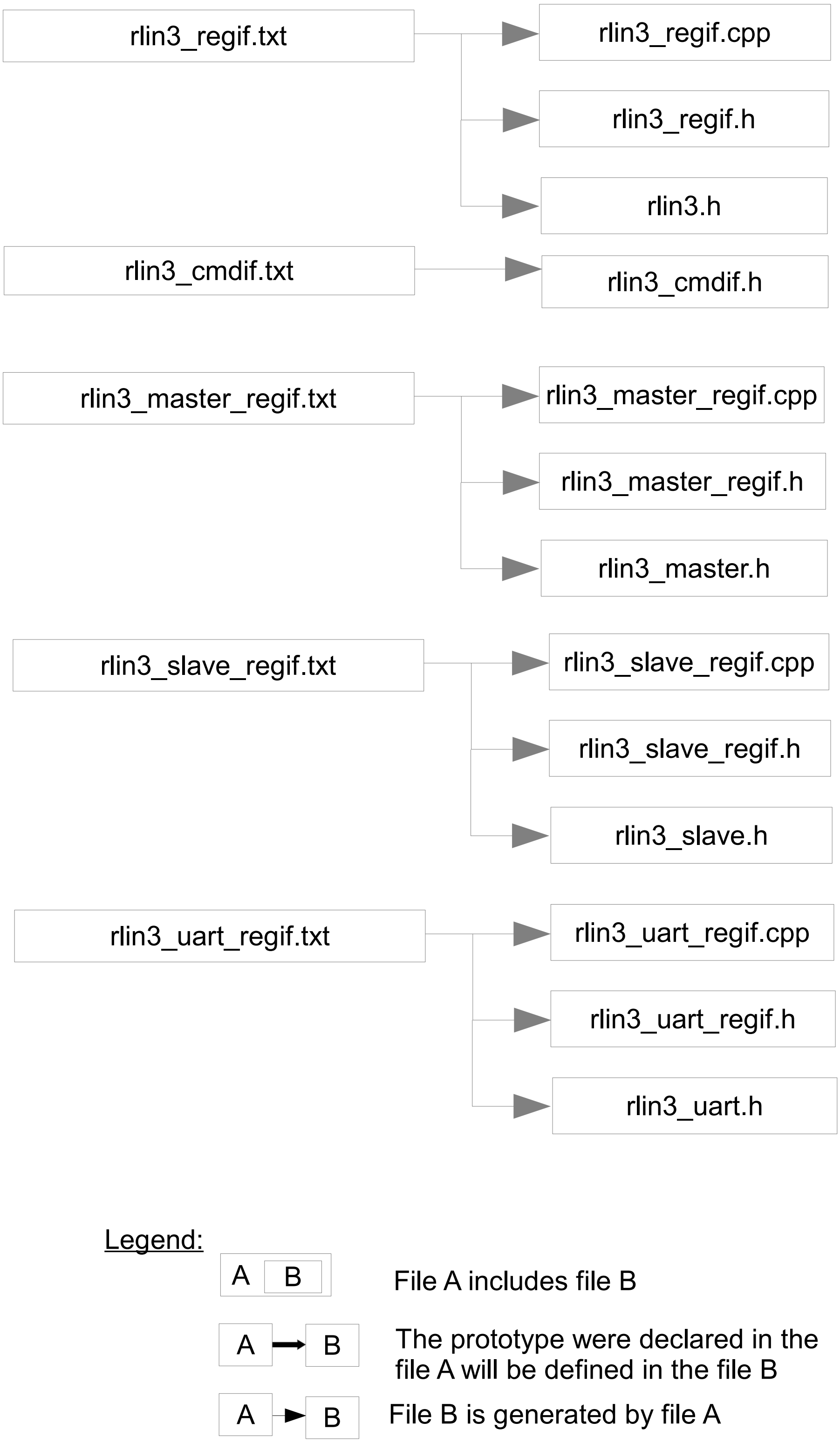


Figure 6.2: File structures (2/2)

Table 6.1: File description

No.	File name	Version	Developed / Reused	Description
1	*re_register.h*	v2013_05_28	Reused	Header file of the re_register class
2	*re_register.cpp*		Reused	Implements the attributes and the operations of common register class
3	*reg_super.h*		Reused	General class for models to access to the memory array
4	*tlm_tgt_if.h*	v2014_04_02	Reused	Header file of the tlm_tgt_if class
5	*tlm_if.h*		Reused	Header file of the tlm_if class
6	*tlm_tgt_if_unit.h*		Reused	Header file of the tlm_tgt_if_unit class
7	*tlm_if_command.h*		Reused	Header file of the tlm_if_command class
8	*rlin3_regif.txt*		Developed	Input file of register IF generator for Crlin3 class
9	*rlin3_master_regif.txt*		Developed	Input file of register IF generator for Crlin3_master class
10	*rlin3_slave_regif.txt*		Developed	Input file of register IF generator for Crlin3_slave class
11	*rlin3_uart_regif.txt*		Developed	Input file of register IF generator for Crlin3_uart class
9	*rlin3_regif.h*		Generated*	Header file of Register IF of RLIN3 model for all modes.
10	*rlin3_regif.cpp*		Generated*	Implementation file of Register IF of RLIN3 model for all modes.
11	*rlin3_master_regif.h*		Generated*	Header file of Register IF of RLIN3 model for master mode.
12	*rlin3_master_regif.cpp*		Generated*	Implementation file of Register IF of RLIN3 model for master mode.
13	*rlin3_slave_regif.h*		Generated*	Header file of Register IF of RLIN3 model for slave mode.
14	*rlin3_slave_regif.cpp*		Generated*	Implementation file of Register IF of RLIN3 model for slave mode.
15	*rlin3_uart_regif.h*		Generated*	Header file of Register IF of RLIN3 model for UART mode.
16	*rlin3_uart_regif.cpp*		Generated*	Implementation file of Register IF of RLIN3 model for UART mode.
17	*rlin3.h*		Developed	Header file of RLIN3 model
18	*rlin3.cpp*		Developed	Implementation file of RLIN3 model
19	*rlin3_master.h*		Developed	Header file of master class of RLIN3 model

No.	File name	Version	Developed / Reused	Description
20	*rlin3_master.cpp*		Developed	Implementation file of master class of RLIN3 model
21	*rlin3_slave.h*		Developed	Header file of slave class of RLIN3 model
22	*rlin3_slave.cpp*		Developed	Implementation file of slave class of RLIN3 model
23	*rlin3_uart.h*		Developed	Header file of master UART of RLIN3 model
24	*rlin3_uart.cpp*		Developed	Implementation file of UART class of RLIN3 model
25	*rlin3_selftest.h*		Developed	Header file of self test class of RLIN3 model
26	*rlin3_selftest.cpp*		Developed	Implementation file of self test class of RLIN3 model
27	*rlin3_cmdif.txt*		Developed	Input file of command IF generator
28	*rlin3_cmdif.h*		Generated*	Command interface
29	*re_security.h*	v100419	Reused	Additional file of tlm_ini_if class and tlm_tgt_if class
30	*re_security.cpp*		Reused	Additional file of tlm_ini_if class and tlm_tgt_if class

(*)Note: **_regif.h* and file **_regif.cpp* are generated from Register IF Generator **v2014_12_01**.
rlin*3_cmdif.h* is generated from command IF generator by version **v2012_05_02**.

6.2. Input/Output file

There is no input or output file.

6.3. How to connect Verification Environment

(1) Connect RLIN3 each input/output port to each relevant signal.

(2) Users need to register RLIN3 and RLIN3's handleCommand pointers to commandHandler.

(3) A RLIN3 users need to build with "IS_RESET_ACTIVE_LOW" to define the active level (active low) for the reset port and “REGIF_SC_REPORT” for dumping sc_report messages.

6.4. handleCommand

Table 6.2: List of parameters of handleCommand API

No.	Parameters	Type	Default	Description
1	MessageLevel	string	fatal|error	Select debug message level from “fatal”, “error”, “warning”, “info” (for register access only). One or more than levels can be connected by a vertical bar. Example “fatal|error”.
2	DumpInterrupt	bool	false	Dump interrupt information when an interrupt is asserted. This message is info level. false ... Not dump interrupt information true ... Dump interrupt information
3	EnableTransInfo	bool	false	Enable/disable information display RLIN3 transmits/receives data: - Simulation time - Instance name. - Frame name. - The following registers value when it turned on: LIDB[5:0], LCBR, LDBRn. - Receive or Transmit.

Table 6.3: List of commands

No.	Parameters	Type	Argument	Description
1	DumpStatInfo	void	-	Dump the statistical information about transmitting/receiving activity. When this command is called, RLIN3 model dumps the following information : - The total amount of data transmission/reception. Clear it after dumping message - Status of the channel (Enable/Disable, waiting data/just transferring)
2	AssertReset	void	start-time, period	Assert and deassert reset signal to the RLIN3 model. <start-time> : the time until asserting reset signal from current time. The unit is "ns". <period> : the time from asserting reset signal to deasserting it. The unit is "ns"
3	SetCLKfreq	void	clk_name, clk_freq	Set clock frequency (Hz) to clk_lsb port specified by clock name. After calling this function, a setting by *clk_lsb* port enables to overwrite and vice versa. <clk_name> : The clock name (clk_lsb). <clk_freq> : The clock frequency which is set to clk_lsb. The unit is “Hz”.
4	help	void	-	Dump the direction how to use handleCommand parameters and commands.

6.4.1. Interrupt condition messages style

Table 6.4: Dump Interrupt condition message description

<table>
<tr><td colspan="2"><b>Condition</b></td><td>This message is dumped out when RLIN3 interrupt is assert and parameter DumpInterrupt is set to true value.</td></tr>
<tr><td colspan="2"><b>Output</b></td><td>This message is printed to standard output (console).</td></tr>
<tr><td colspan="3"><b>Format:</b>
Info [<time>ns] (hier_instance_name) INT [RLIN3: interrupt_name] Assert
<b>Example:</b>
Info   [2010ns] (HARDWARE.....RLIN3) INT [RLIN3: lin3_int_t] Assert</td></tr>
<tr><td><b>No</b></td><td><b>Tag name</b></td><td><b>Description</b></td></tr>
<tr><td>1</td><td>Info</td><td>Always "Info" because this message is info level.</td></tr>
<tr><td>2</td><td>time</td><td>Simulation time. The time unit depends on sc_time_resolution setting.</td></tr>
<tr><td>3</td><td>hier_instance_name</td><td>Hierarchy instance name of RLIN3 model is being used.</td></tr>
<tr><td>4</td><td>interrupt_name</td><td>Interrupt factor: lin3_int_t, lin3_int_r, lin3_int_s</td></tr>
</table>

6.4.2. DumpStatInfo messages style

Table 6.5: Statistical information message description

Condition	This message is dumped out when “DumpStatInfo” is transferred to handleCommand.
Output	This message is printed to standard output (console).

Format:

```
PROFILE(StatInfo): RLIN3 <mode>: Info [<time>ns] ( hier_instance_name ):
PROFILE(StatInfo): RLIN3 <mode>: RLIN3 transfer information:
PROFILE(StatInfo): RLIN3 <mode>:   Current state:   <state>
PROFILE(StatInfo): RLIN3 <mode>:   Total data transmitted: %d byte(s)
PROFILE(StatInfo): RLIN3 <mode>:   Total data received:    %d byte(s)
PROFILE(StatInfo): RLIN3 <mode>: EndInfo.
```

Example:

```
PROFILE(StatInfo): RLIN3 MASTER: Info [2010ns] (HARWARE...RLIN3)  :
PROFILE(StatInfo): RLIN3 MASTER: RLIN3 transfer information:
PROFILE(StatInfo): RLIN3 MASTER:   Current State:   Idle
PROFILE(StatInfo): RLIN3 MASTER:   Total data transmitted: 4 bytes
PROFILE(StatInfo): RLIN3 MASTER:   Total data received:    10 bytes
PROFILE(StatInfo): RLIN3 MASTER: EndInfo.
```

No	Tag name	Description
1	Info	Always "Info" because this message is info level.
2	time	Simulation time. The time unit depends on sc_time_resolution setting.
3	hier_instance_name	Hierarchy instance name of RLIN3 model is being used.
4	state	The current state of RLIN3: - Idle: RLIN3 is in idle state. - Reset: RLIN3 is in reset state. - Header Communication: RLIN3 is transmitting a frame header. - Response Transmission: RLIN3 is transmitting a response transmission. - Response Reception: RLIN3 is receiving a response reception.
5	Total data transmitted	The total bytes of transactions are transmitted.
6	Total data received	The total bytes of transactions are received.

6.4.3. EnableTransInfo messages style

Table 6.6: Transmit/receive debug message description

Condition	This message is dumped out when RLIN3 transmits/receives a data and parameter 'EnableTransInfo' is set to true value.
Output	This message is printed to standard output (console).

Format:

```
PROFILE(TransInfo): RLIN3 <mode>: Info [<time>ns] ( hier_instance_name ):
PROFILE(TransInfo): RLIN3 <mode>: RLIN3 model transmit a frame <frame_name>
PROFILE(TransInfo): RLIN3 <mode>:   Operation:     <operation>
PROFILE(TransInfo): RLIN3 <mode>:   ID value:      <ID>
PROFILE(TransInfo): RLIN3 <mode>:   Data transfer value:  <DataTransferValue>
PROFILE(TransInfo): RLIN3 <mode>:   Checksum value:  <ChecksumValue>
PROFILE(TransInfo): RLIN3 <mode>: EndInfo.
```

Example:

```
PROFILE(TransInfo): RLIN3 MASTER: Info  [2011ns] (HARDWARE.... RLIN3):
PROFILE(TransInfo): RLIN3 MASTER: RLIN3 model transmit a frame Header
PROFILE(TransInfo): RLIN3 MASTER:   Operation:     Transmitting data
PROFILE(TransInfo): RLIN3 MASTER:   ID value:      0x0F2A
PROFILE(TransInfo): RLIN3 MASTER: EndInfo.

PROFILE(TransInfo): RLIN3 MASTER: Info  [3011ns] (HARDWARE.... RLIN3):
PROFILE(TransInfo): RLIN3 MASTER: RLIN3 model transmit a frame Response
PROFILE(TransInfo): RLIN3 MASTER:   Operation:     Transmitting data
PROFILE(TransInfo): RLIN3 MASTER:   Data transfer value:   0x0F2A
PROFILE(TransInfo): RLIN3 MASTER: EndInfo.

PROFILE(TransInfo): RLIN3 MASTER: Info  [3011ns] (HARDWARE.... RLIN3):
PROFILE(TransInfo): RLIN3 MASTER: RLIN3 model transmit a frame Response
PROFILE(TransInfo): RLIN3 MASTER:   Operation:     Transmitting data
PROFILE(TransInfo): RLIN3 MASTER:   Checksum value:   0x0F2A
PROFILE(TransInfo): RLIN3 MASTER: EndInfo.
```

No	Tag name	Description
1	Info	Always "Info" because this message is info level.
2	time	Simulation time. The time unit depends on sc_time_resolution setting.
3	mode	The RLIN3 mode: - "MASTER": RLIN3 model is in Master mode. - "SLAVE": RLIN3 model is in Slave mode. - "UART": RLIN3 model is in Uart mode.
4	hier_instance_name	Hierarchy instance name of the RLIN3 model is being used.
5	frame_name	The data frame name: - “Header”: the packet data are frame header. - “Response”: the packet data are a response.
6	ID	The value of ID field in the register LIDB. This information is dumped when frame name is Header.
7	DataTransferValue	The value of transferred data. This information is dumped when frame name is Response and transferred data are not Checksum value.
8	ChecksumValue	The value of checksum data. This information is dumped when frame name is Response and transferred data are not Data value.
9	operation	RLIN3 operation: - “Transmitting data” if the RLIN3 model is transmitting data - “Receiving data” if the RLIN3 model is receiving data

6.4.4. Help messages

Table 6.7: Dump help command message description

Condition	This message is dumped out when “help” is transferred to handleCommand.
Output	The help message is used for handleCommand.

```
    --- command ---
  help                                              Show direction
  MessageLevel   <fatal|error|warning|info> Select debug message level (Default: fatal,error)
  AssertReset <start_time> <period>          Assert and deassert reset signal to a target model
  DumpInterrupt <arg>                             Enable/disable interrupt information display when an interrupt is sent
( Default:false  )
  EnableTransInfo <arg>                          Enable/disable information display ( Default:false  )
  DumpStatInfo                                      Dumps statistical information of RLIN3
  SetCLKfreq <clk_name> <clk_freq>        Set clocks to RLIN3
```

Table 6.8: Dump tgt help command message description

Condition	This message is dumped out when “tgt help” is transferred to handleCommand.
Output	The help message is used for handleCommand.

```
  Command                          Description
  -------------------------------------------------------------------------------------------
  set_param <term> <value> : Set simulation information about access to target.
                 <term>  : m_bus_clk | m_bus_gnt | m_bus_rgnt | m_buf_size
                           | m_wr_latency | m_rd_latency | m_phase_mode
                           | m_p_log_file | m_wr_log | m_rd_log | m_msg_out_lvl
                 <value> : Please see tlm_common_class spec sheet.
  get_param <term>   : Get simulation information about access to target.
  init_param             : Initialize simulation information.
```

Table 6.9: Dump reg help command message description

Condition	This message is dumped out when “tgt help” is transferred to handleCommand.
Output	The help message is used for handleCommand.

```
  --- reg ---
  reg MessageLevel <fatal|error|warning|info>  Select debug message level (Default: fatal,error)
  reg DumpRegisterRW <true/false>               Select dump register access information  (Default: false)
  reg DumpFileNameLineNum <true/false>      Select dump information about file name and line number (Default:
false)
  reg <register_name> MessageLevel <fatal|error|warning|info> Select debug message level for register (Default:
fatal,error)
  reg <register_name> force <value>      Force register with setting value
  reg <register_name> release              Release register from force value
  reg <register_name> <value>              Write a value into register
  reg <register_name>                         Read value of register
  reg help                                          Show a direction
```

6.5. Error and debugging messages

6.5.1. Error and debugging messages style

Table 6.10: Dump error message description

Condition		This kind of message is output when error occurs or some important events occur. Detailed conditions are described in the "Description" column of table 6.12.
Output		This kind of message is printed to standard output (console).
Format: <Severity> [<time>ns] (<hier_instance_name>) [<port><handleCommand>] [Message content] **Example:** Error [1230ns] (HARDWARE...RLIN3) Cannot write to LWBR0 while LMST[0] is 1. Error [1240ns] (HARDWARE...RLIN3) [handleCommand] clk_lsb must be greater than 0. Error [1254ns] (HARDWARE...RLIN3) [clk_lsb port] clk_lsb must be greater than 0.		
No	**Tag name**	**Description**
1	Severity	Kind of severity of the message including Error,Warning,Info.
2	time	Simulation time. The time unit depends on sc_time_resolution setting.
3	hier_instance_name	Hierarchy instance name of RLIN3 model is being used.
4	port	Message is outputted against port access.
5	handleCommand	Message is outputted against handleCommand access.

Table 6.11: Dump error message description for handleCommand message

Condition		This kind of message is output when error occurs or some important events occur when using command of handleCommand.
Output		This kind of message is printed to standard output (console).
Format: <Severity> (<hier_instance_name>) [Message content] **Example:** Error (HARDWARE...RLIN3) wrong number of arguments (clk_lsb invalid_value) : Type reslx.rlin3 help		
No	**Tag name**	**Description**
1	Severity	Kind of severity of the message including Error, Warning, Info.
2	hier_instance_name	Hierarchy instance name of RLIN3 model is being used.

6.5.2. Error and debugging messages

Table 6.12: Error and debugging message

No.	Type	Severity	Message	Description
1	User	Error	Clock name is invalid.	Dump this message when the setting clock name is invalid.
2	User	Info	Break low is detected, start header receiving process.	Dump this message when receiving break low signal.
3	User	Info	Expansion bit is matched with data comparison.	Dump this message when the expansion bit is matched with data comparison
4	User	Info	Expansion bit is matched.	Dump this message when the expansion bit is matched
5	User	Info	RLIN3 will reset for %f ns after %f ns.	Dump this message when AssertReset command is called.

No.	Type	Severity	Message	Description
6	User	Info	Reset signal is asserted.	Dump this message when rstp_n and rstc_n is low or AssertReset command is called.
7	User	Info	Reset signal is negated.	Dump this message when rstp_n and rstc_n is high or AssertReset command is ended.
8	User	Info	Software reset is asserted	Dump this message when software reset is asserted.
9	User	Info	Software reset is de-asserted	Dump this message when software reset is deasserted.
10	User	Info	The %s is set with a frequency as %f.	Dump this message when clk_lsb clock is set new value.
11	User	Warning	Bit LCUC.OM1 is not allowed to access when communication bit (LTRC.FTS) is set in Selftest mode.	Dump this message when access to LCUC.OM1 when communication bit (LFRC.FTS) is set in Selftest mode
12	User	Warning	Bit LDFC.LSS can not be set in Frame Combined Mode.	Dump this message when bit LDFC.LSS is set in Frame Combined Mode.
13	User	Warning	Bit error occurs, Master transferring is stopped.	Dump this message when bit error occurs.
14	User	Warning	Bit error occurs, Slave transferring is stopped.	Dump this message when Bit error occurs in Slave mode.
15	User	Warning	Break delimiter is detected unsuccessfully.	Dump this message when detect invalid break delimiter control and value data.
16	User	Warning	Break low is detected unsuccessfully.	Dump this message when detect invalid break low control and data value.
17	User	Warning	Break low period is less than configuration value.	Dump this message when break low period is less than predefined value.
18	User	Warning	Checksum error occurs, Master receiving is stopped.	Dump this message when checksum error occurs.
19	User	Warning	Checksum error occurs, Slave receiving is stopped.	Dump this message when Checksum error occurs in Slave mode.
20	User	Warning	Classic checksum should be used when response data bytes is 0.	Dump this message when enhance checksum is set with response data bytes is 0.
21	User	Warning	Data register LDBR%d should not be written when RTS is 1.	Dump this message when written data to LDB register while RTS is 1.
22	User	Warning	Data registers LDBR%d should not be written when FTS is 1 in Frame Combined mode transmission.	Dump this message when user write to LDBRN when FTS is 1 in Frame Combined mode transmission in Master mode.
23	User	Warning	Data registers LDBR%d should not be written when FTS is 1 in reception.	Dump this message when user write to LDBRN when FTS is 1 in reception in Master mode.
24	User	Warning	Data registers LDBR%d should not be written when RTS is 1 in Frame Separate mode transmission.	Dump this message when user write to LDBRN when RTS is 1 in Frame Separate mode transmission in Master mode.
25	User	Warning	Data registers LDBR%d should not be written when RTS is 1.	Dump this message when LDBRn is written when RTS is 1.
26	User	Warning	Frame error occurs, Master receiving is stopped.	Dump this message when frame error occurs.
27	User	Warning	Frame error occurs, Slave receiving is stopped.	Dump this message when Frame error occurs in Slave mode.
28	User	Warning	Frame timeout error occurs during header reception process.	Dump this message when frame timeout error occurs during header reception process.

No.	Type	Severity	Message	Description
29	User	Warning	In Expansion bit mode, LEDE.OERE should not be set when LUOR1.UEBDCE is set.	Dump this message when in enable overrun check with expansion bit mode.
30	User	Warning	Inter header space is detected unsuccessfully.	Dump this message when detect invalid Inter byte space control and value data.
31	User	Warning	LEDE.BERE should be written to 1 in Master mode.	Dump this message when clear LEDE.BERE to 0 in Master mode
32	User	Warning	LEDE.BERE should be written to 1 in Slave mode.	Dump this message when clear LEDE.BERE to 0 in Slave mode
33	User	Warning	LEDE.FERE should be written to 1 in Master mode.	Dump this message when clear LEDE.FERE to 0 in Master mode
34	User	Warning	LEDE.FERE should be written to 1 in Slave mode.	Dump this message when clear LEDE.FERE to 0 in Slave mode
35	User	Warning	LEDE.IPERE should be written to 1 in Slave mode.	Dump this message when clear LEDE.IPERE to 0 in Slave mode
36	User	Warning	LIE is set to initial value in UART mode	Dump this message when set value to LIE in UART mode
37	User	Warning	LIN Slave Auto Baud Rate mode is NOT supported in Self-test mode.	Dump this message when set auto baud rate mode in Self-test mode
38	User	Warning	LIN Slave mode should be operated with fa only.	Dump this message when the setting clock is not fa in Slave mode.
39	User	Warning	LSC.IBHS is set to initial value in LIN mode reception	Dump this message when set 1 to LSC.IBHS in LIN mode reception
40	User	Warning	LSC.IBHS is set to initial value in UART mode	Dump this message when set 1 to LSC.IBHS in UART mode
41	User	Warning	LSC.IBHS should be set 3'b001 in LIN Slave Self test mode.	Dump this message when set value other than 3'b001 in LIN Slave Self-test mode
42	User	Warning	LTRC.LNRR bit can not be cleared by CPU access.	Dump this message when user clears LNRR bit in Slave mode.
43	User	Warning	LTRC.RTS bit can not be cleared by CPU access.	Dump this message when user clears RTS bit.
44	User	Warning	LTRC.RTS should not be set if LIN no response is set or Receive Header is not completed.	Dump this message when set LTRC.RTS = 1 when LNRR is not set or header is not received completed
45	User	Warning	LTRC.RTS should not be set if LUOER.UTOE is not set to 1.	Dump this message when set LTRC.RTS = 1 when LUOER.UTOE Is not set in UART mode
46	User	Warning	LUOER is set to initial value in LIN mode	Dump this message when set value to LUOER in LIN mode
47	User	Warning	LUOER.UROE can not be set during multi-byte transmission.	Dump this message when set 1 to LUOER.UROE in multi-byte transmission in UART mode
48	User	Warning	LUOR1 is set to initial value in LIN mode	Dump this message when set value to LUOR1 in LIN mode
49	User	Warning	LUOR1.UEBDCE should not be set to 1 for multi-byte communication.	Dump this message when set 1 to LUOR1.UEBDCE in multi-byte transmission in UART mode
50	User	Warning	LUOR1.UEBDCE should not be set when LUOR1.UEBE is not set.	Dump this message when set 1 to LUOR1.UEBDCE with LUOR1.UEBE = 0
51	User	Warning	LUOR1.UEBDCE should not be set when LUOR1.UECD is set.	Dump this message when set 1 to LUOR1.UEBDCE with LUOR1.UECD = 0
52	User	Warning	LUOR1.UEBDL should not be set to 1 for multi-byte communication.	Dump this message when set 1 to LUOR1.UEBDL in multi-byte transmission in UART mode

No.	Type	Severity	Message	Description
53	User	Warning	LUOR1.UEBE should not be set to 1 for multi-byte communication. Transmission will not start.	Dump this message when set 1 to LUOR1.UEBE in multi-byte transmission in UART mode
54	User	Warning	LUOR1.UECD should not be set to 1 for multi-byte communication.	Dump this message when set 1 to LUOR1.UECD in multi-byte transmission in UART mode
55	User	Warning	LUTDR is set to initial value in LIN mode	Dump this message when set value to LUTDR in LIN mode
56	User	Warning	LUWTDR is set to initial value in LIN mode	Dump this message when set value to LUWTDR in LIN mode
57	User	Warning	LWBR.LWBR0 is set to initial value in UART mode	Dump this message when set value to LWBR.LWBR0 in UART mode
58	User	Warning	LWUP.WUTL is set to initial value in UART mode	Dump this message when set value to LWUP.WUTL in UART mode
59	User	Warning	Operation can not perform because setting Bit Time is invalid .	Dump this message when setting Bit Time is invalid
60	User	Warning	Operation can not perform because setting Bit Time is invalid .	Dump this message when setting Bit Time is invalid
61	User	Warning	PID error occurs, Slave receiving is stopped.	Dump this message when PID error occurs in Slave mode.
62	User	Warning	PID start bit is detected unsuccessfully.	Dump this message when detect invalid PID start bit control and value data.
63	User	Warning	Physical error occurs, Master transferring is stopped.	Dump this message when physical error occurs.
64	User	Warning	Reception operation is invalid when FTS is equal 0.	Can not receive a data when FTS = 0.
65	User	Warning	Register %s is not allowed to access when communication bit (LTRC.FTS) is set in Selftest mode.	Dump this message when access to any register (except LCUC.OM0) when communication bit is set in Self-test mode
66	User	Warning	Register LURDR should not be read by access size 8 bit when 9 bit communication is in progress.	Dump this message when LURDR is read by access size 8 bit when 9 bit communication.
67	User	Warning	Register LUTDR should not be written by access size 8 bit when 9 bit communication is in progress.	Dump this message when LUTDR is written by access size 8 bit when 9bit communication.
68	User	Warning	Register LUTDR should not be written when LUWTDR is already written.	Dump this message when LUTDR is written when LUWTDR is already written.
69	User	Warning	Register LUTDR should not be written when multi-byte communication is in progress.	Dump this message when LUTDR is written when multi-byte communication.
70	User	Warning	Register LUWTDR should not be written by access size 8 bit when 9 bit communication is in progress.	Dump this message when LUWTDR is written by access size 8 bit when 9bit communication.
71	User	Warning	Register LUWTDR should not be written when LUTDR is already written.	Dump this message when LUWTDR is written when LUTDR is already written.
72	User	Warning	Register LUWTDR should not be written when multi-byte communication is in progress.	Dump this message when LUWTDR is written when multi-byte communication.
73	User	Warning	Register LUWTDR should not be written when receiving data in Half duplex mode.	Dump this message when LUWTDR is written during receiving data.
74	User	Warning	Reset is in progress.	Dump this message when reset is in progress.
75	User	Warning	Respond preparation error occurs, Master operation is stopped.	Dump this message when response preparation error error occurs.
76	User	Warning	Respond preparation error occurs, Slave operation is stopped.	Dump this message when Respond preparation error in Slave mode.
77	User	Warning	SYNC field error occurs, Slave receiving is stopped.	Dump this message when SYNC field error occurs in Slave mode.
78	User	Warning	SYNC start bit is detected unsuccessfully.	Dump this message when detect invalid SYNC start bit control and value data.

No.	Type	Severity	Message	Description
79	User	Warning	SYNC stop bit is detected unsuccessfully.	Dump this message when detect invalid SYNC stop bit control and value data.
80	User	Warning	Samples per bit should be fixed 16 in LIN Master mode.	Dump this message when user set Samples per bit differrent from 16 in LIN Master mode.
81	User	Warning	Samples per bit should be fixed 16 in LIN Slave Fix baud rate mode.	Dump this message when user set Samples per bit differrent from 16 in LIN Slave Fix baud rate mode.
82	User	Warning	Samples per bit should be fixed to 4 or 8 in LIN Slave Auto baud rate mode.	Dump this message when user set Samples per bit differrent from 4 or 8 in LIN Slave Auto baud rate mode.
83	User	Warning	Samples per bit should more than 6 in UART mode.	Dump this message when user set Samples per bit less than 6 in UART mode.
84	User	Warning	Should not set LMD.LIOS bit in UART mode.	Dump this message when user set LIOS bit in UART mode.
85	User	Warning	The %s period is less than 1 unit time of system.	Dump this message when the setting clk_lsb is less than 1 unit time of system.
86	User	Warning	The Bit error occurs in Uart mode.	Dump this message when Bit error occurs in UART modes.
87	User	Warning	The Frame error occurs in Uart mode.	Dump this message when Frame error occurs in UART modes.
88	User	Warning	The Overrun error occurs in Uart mode.	Dump this message when Overrun error occurs in UART modes.
89	User	Warning	The Parity error occurs in Uart mode.	Dump this message when ID parity error occurs in UART modes.
90	User	Warning	The bit %s.%s is read 0 only.	Dump this message when a bit with access mode read 0 only is written.
91	User	Warning	The bit %s.%s is read only.	Dump this message when a bit with access mode read only is written.
92	User	Warning	The bit %s.%s is written 0 only.	Dump this message when a bit with access mode write 0 only is written or read.
93	User	Warning	The clk_lsb period is equal 0.	Dump this message when input signals change value while clk_lsb is equal 0.
94	User	Warning	The frame timeout error occurs . Operation of LIN master is stopped!.	Dump this message when frame timeout error occurs.
95	User	Warning	The respond timeout error occurs . Operation of LIN master is stopped!.	Dump this message when response timeout error occurs.
96	User	Warning	The timeout error occurs during response process.	Dump this message when timeout error occurs during response process.
97	User	Warning	Timeout error should be disabled for Auto baud rate LIN Slave mode operation.	Dump this message when timeout error check is enabled in Auto baud rate LIN Slave mode
98	User	Warning	Timeout error should be disabled for data group communication.	Dump this message when timeout error check is enabled in data group communication in LIN mode
99	User	Warning	UART mode should be operated with fa only.	Dump this message when the setting clock is not fa in UART mode.
100	User	Warning	Unlock sequence key is wrong.	Dump this message when unlock sequence key is wrong
101	User	Warning	User should not access to LUDB0 register in LIN mode.	Dump this message when users access to LUDB0 register in LIN mode.
102	User	Warning	LTRC.LNRR and LTRC.RTS should not be set to 1 simultaneously.	Dump this message when users set 1 to both LTRC.LNRR and LTRC.RTS
103	User	Warning	Setting value to LDFC.RFDL must be equal or less than 0x8.	Dump this message when users set value greater than 0x8 into LDFC.RFDL
104	User	Warning	Register LUTDR should not be written before the generation of transmission interrupt.	Dump this message when users write value to LUTDR in UART mode during transmission process

6.6. Defined macro and template

(1) There is no macro in this model.

(2) This model supports only 32 bits bus width socket.

7. Flow diagram

Summary:

(1) Chapter 7.1 describes sequence diagram of the RLIN3 model.

(2) Activity of RLIN3 model has 4 independent state machines for LIN normal operation, UART operation, WAKEUP operation and SELF TEST operation. Their relation is described in chapter 7.2, state diagram.

Table 7.1: Features and diagram reference table

Model Features	Hardware manual chapter	Diagram	Description	Figure
Sequence flow	-	Sequence flow	The sequence diagram of RLIN3	7.1 7.2 7.3 7.4
State diagram	-	State diagram	State information of model	7.5 7.6 7.7 7.8 7.9
Bit time calculating	*17.10*	Bit time calculating process	Bit time calculating for a transaction	7.10
DeAssertIntrMethod	*17.4*	DeAssertIntrMethod	De-assert interrupts ports.	7.11
ResetMethod	-	ResetMethod	Handle the reset operations.	7.12
EnableReset	-	EnableReset	Reset all variables and output ports if reset is selected.	7.13
GetRegBitsVal	-	GetRegBitsVal	Get value of a number of bits in register.	7.14
SetRegBitsVal	-	SetRegBitsVal	Set value to a number of bits in register.	7.15
UpdateAllRegs	-	UpdateAllRegs	Update all register values from register bank of rlin3 class to register bank of master/slave/UART classes.	7.16
RegisterAccessCheck	-	RegisterAccessCheck	Check access mode for all registers.	7.17

Model Features	Hardware manual chapter	Diagram	Description	Figure
TransmitDataMethod	*17.7.1.1* *17.7.1.2* *17.7.2.2* *17.7.3.1* *17.7.4.1* *17.7.4.3* *17.7.5.1*	TransmitDataMethod	Handle main operations for transmitting a data in RLIN3 model.	7.18
TransmitProcess	*17.7.1.1* *17.7.1.2* *17.7.2.2* *17.7.3.1* *17.7.4.1* *17.7.4.3* *17.7.5.1*	TransmitProcess	Transmit a data. This function is called by Master/Slave/Uart classes.	7.19
ReceptionProcess	*3.3.5*	ReceptionProcess	Handle main operations for reception data in RLIN3 model.	7.20
TransmitWakeup	*17.8.1*	TransmitWakeup	Transmit a wakeup signal.	7.21
TransmitHeaderLoop	*17.7.1.1* *17.7.4.1*	TransmitHeaderLoop	Transmit a frame header data.	7.22
TransmitRespLoop	*17.7.1.2* *17.7.2.2* *17.7.4.1*	TransmitRespLoop	Transmit a response data.	7.23
RespReception	*17.7.1.3* *17.7.2.3* *17.7.3.2* *17.7.4.2*	RespReception	Receive a response data.	7.24 7.25
HeaderReception	*17.7.2.1*	HeaderReception	Receive a frame header data.	7.26
CalcNumOfByte	-	CalcNumOfByte	Calculate a number of byte for a transaction.	7.27
CalcBitBoundary	*17.10*	CalcBitBoundary	Calculate amount of time for waiting before notify bit error event.	7.28
OutputData process in Master mode	-	Output Data process in Master mode	Performs output data process in Master Mode.	7.30
UpdateStatus in Master mode	-	UpdateStatus in Master mode	Update current status to LST register also notify an interrupt.	7.31
UpdateErrorStatus in Master/Slave/Uart modes	-	UpdateErrorStatus in Master/Slave/Uart modes	Update current Error status to LEST register.	7.32

Model Features	Hardware manual chapter	Diagram	Description	Figure
UpdateRegisters in Master mode	-	UpdateRegisters in Master mode	Update data value into data register in Master mode.	7.33
ReceiveMethod in Master mode	*17.7.3.2*	Receive Method in Master mode	Receive a data in Master mode.	7.34
cb_LSTC_LSTM function of RLIN3 Master/Slave classes	*17.3.2.4* *17.3.3.3*	cb_LSTC_LSTM function of RLIN3 Master/Slave classes	Callback function for LSTC register in Master/Slave classes.	7.35
cb_LCUC_OM1 function of RLIN3 Master/Slave/Uart classes	*17.3.2.11* *17.3.3.10* *17.3.4.7*	cb_LCUC_OM1 function of RLIN3 Master/Slave/Uart classes	Callback function for LCUC register in Master/Slave/Uart classes.	7.36
cb_LTRC_FTS function of RLIN3 Master class	*17.3.2.12*	cb_LTRC_FTS function of RLIN3 Master class	Callback function for LTRC register in Master class.	7.37
OutputData in Slave mode	-	OutputData in Slave mode	Performs output data process in Slave Mode.	7.38
ReceiveMethod in Slave mode	*17.7.3.2*	Receive Method in Slave mode	Receive a data in Master mode.	7.39
UpdateStatus in Slave mode	-	UpdateStatus in Slave mode	Update current status to LST register also notify an interrupt.	7.40
UpdateRegisters in Slave mode	-	UpdateRegisters in Slave mode	Update data value into data register in Slave mode.	7.41
cb_LTRC_FTS in RLIN3 Slave class*17.3.3.11 17.3.4.8*	*17.3.3.11*	cb_LTRC_FTS in RLIN3 Slave class	Callback function for LTRC register in Slave class.	7.42
AddParity	-	AddParity	Add parity bit to data before transmitting.	7.43
ReceptionMethod in UART mode	*17.8.2*	ReceptionMethod in UART mode	Receive a data in UART mode.	7.44
UpdateRegisters in UART mode	-	UpdateRegisters in UART mode	Update data value into data register in UART mode.	7.45
UpdateStatus in UART mode	-	UpdateStatus in UART mode	Update current status to LST register also notify an interrupt.	7.46
cb_LTRC_RTS in UART mode	*17.3.4.8*	cb_LTRC_RTS in UART mode	Callback function for LTRC register in UART class.	7.47
cb_LUOER_UTOE in UART mode	*17.3.4.16*	cb_LUOER_UTOE in UART mode	Callback function for LUOER register in UART class.	7.48

Model Features	Hardware manual chapter	Diagram	Description	Figure
cb_LUTDR_UTD in UART mode	*17.3.4.18*	cb_LUTDR_UTD in UART mode	Callback function for LUTDR register in UART class.	7.49
cb_LUWTDR_UWTD in UART mode	*17.3.4.20*	cb_LUWTDR_UWTD in UART mode	Callback function for LUWTDR register in UART class.	7.50
Self Test process	*17.9*	Self Test process	Handle the connection port in the Self Test operations.	7.51
Timeout handling process	*17.7.7*	Timeout handling process	Handle the timeout functions in Master and Slave mode.	7.52
SW reset handling process	*17.6*	SW reset handling process	Handle the SW reset function in the all modes.	7.53
handleCommand	-	handleCommand process	handleCommand processing flow	7.54

7.1. Sequence flow

(1) RLIN3 model is described via Crlin3 class, Crlin3_master class, Crlin3_slave, Crlin3_common and Crlin3_uart class. Crlin3 class uses Crlin3_master class, Crlin3_slave class and Crlin3_uart class as instances. Crlin3_master class, Crlin3_slave class and Crlin3_uart class contain registers, call back functions and APIs to access registers for Master, Slave and UART modes.

(2) The relationship of Crlin3 class, Crlin3_master class, Crlin3_slave class and Crlin3_uart class are described more details in figure 7.1.

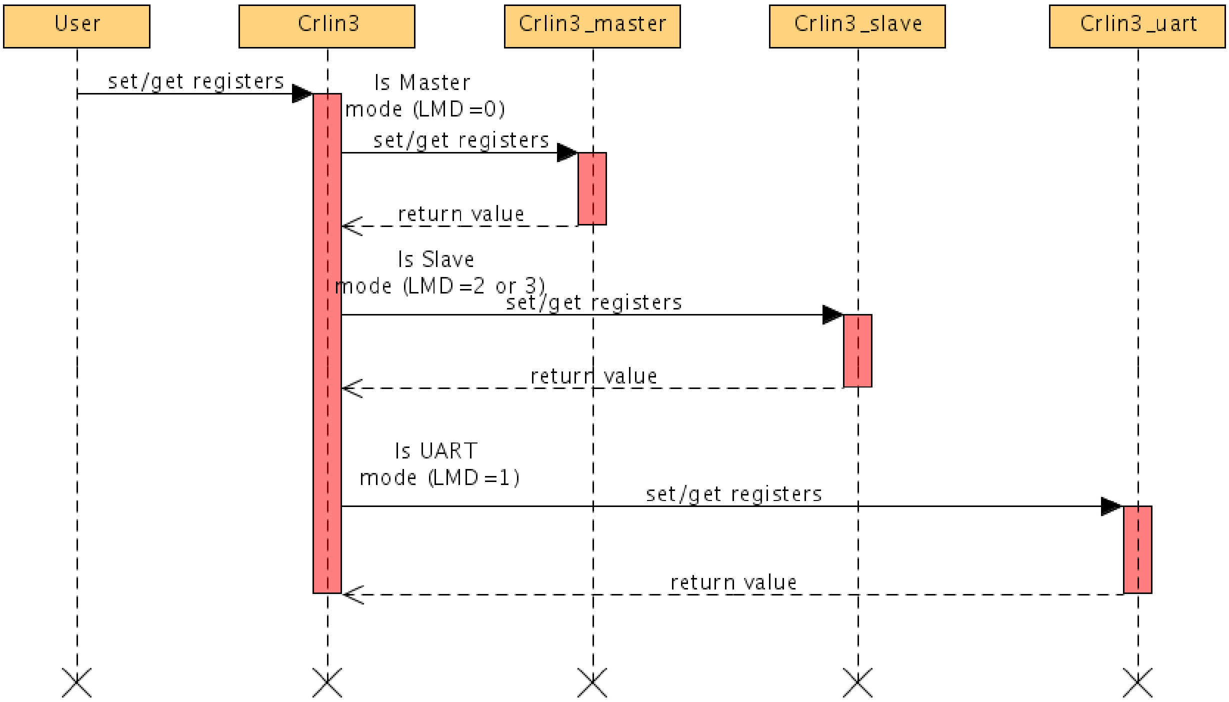


Figure 7.1: Sequence diagram for register update

Explanation :

(1) When users write/read a value to registers of the RLIN3 model. Depended on the value of the LMD bit, the RLIN3 model will update corresponding classes.

(2) If the LMD equals 0x0, then registers of the Master will be updated.

(3) If the LMD equals 0x2 or 0x3, then registers of the Slave will be updated.

(4) If the LMD equals 0x1, then registers of the UART will be updated.

(5) For RLIN3 registers, they will be updated in Crlin3 class when a register of Crlin3_master or Crlin3_slave or Crlin3_uart class are read or written. This means that master_reg_wr/master_reg_rd or slave_reg_wr/slave_reg_rd and uart_reg_wr/uart_reg_rd functions are called.

(6) The LMD bit can be changed only in the SW reset progress.

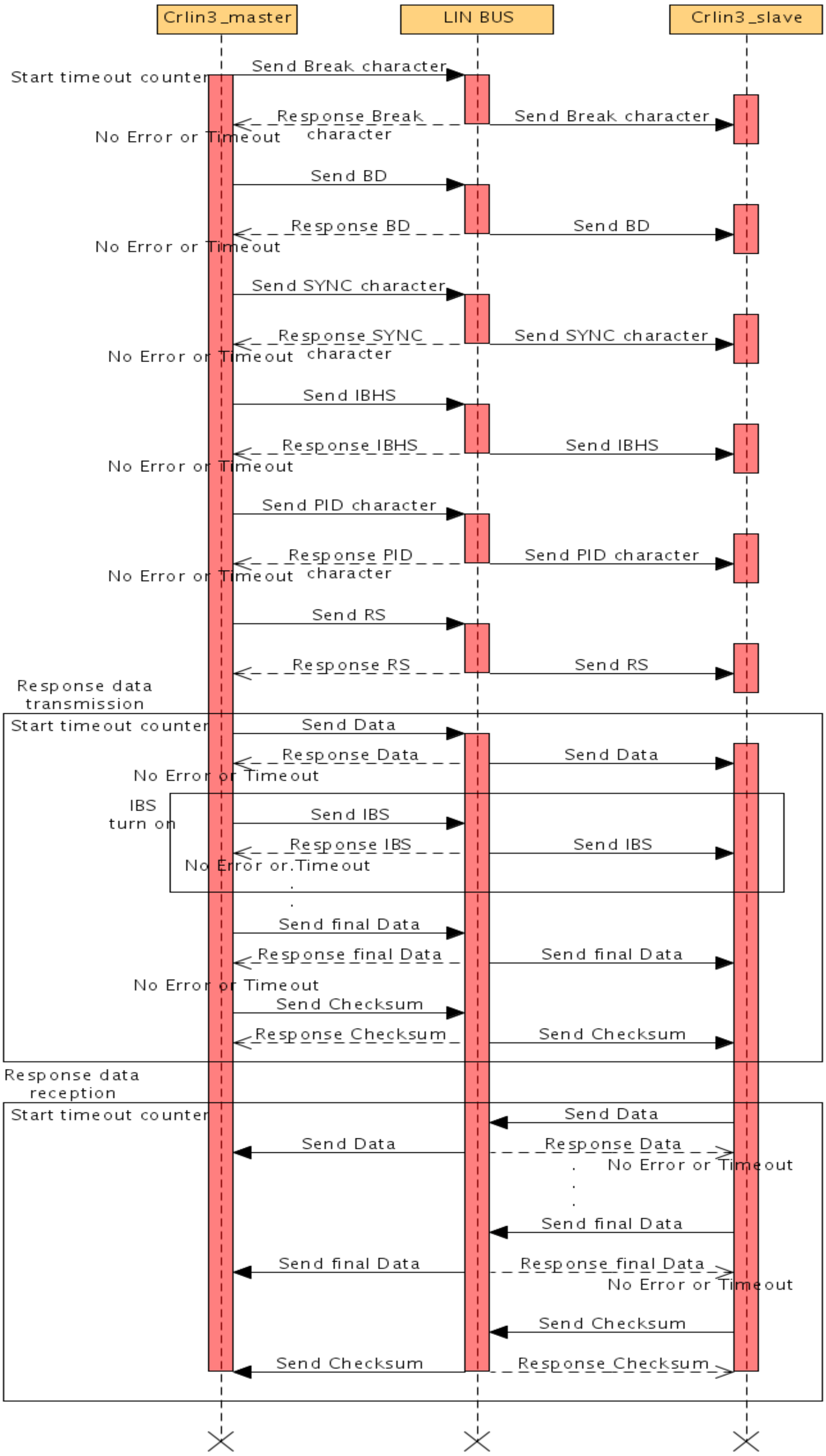


Figure 7.2: Sequential diagram for LIN mode transaction

Explanation :

(1) In the LIN transfer normal mode, for transmitting data from a master to a slave, the master first must transmits the header to slave according to the sequence such as BREAK/BD/SYN/IBHS/PID/RS respectively. During transmitting each field, the master must wait a response the same data from the LIN BUS to check whether the sent data is correct not. The LIN BUS will broadcast data to all slave nodes.

(2) If an error occurs, then transmit will be terminated. In reserve, the next field will be transmitted.

(3) When the header transmission finishes successfully, the master will start to transfer or receive data to/from the slave. If IBS is turned on, the IBS bits will be transferred between data. Similarly to transfer each header field, each data will be checked its correctness by comparing the sent data and the response data from LIN BUS.

(4) There will be three time out processes of the master and slaver for a transmit transaction.

(5) The first time out will be started when starting to transfer the first field of the header in the master while the slaver will start to count second time out when it receives the first field. For the third time out process, this time out will be started whenever users begin to create data transmission process.

(6) If the time out counting competes before finishing transferring the header or data. The transaction will be terminated.

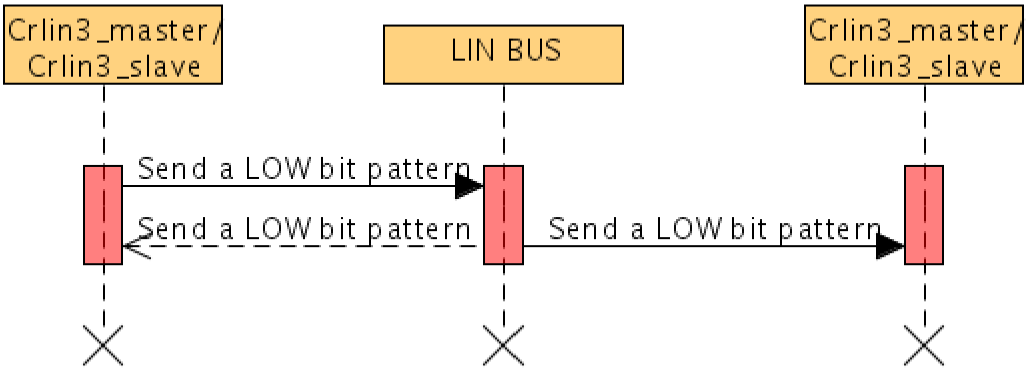


Figure 7.3: Wakeup transaction

Explanation :

(1) In the wake up mode, first, the a LIN node will send a pattern of bits to wake up the another node. While sending this pattern of bits, the LIN node will listen the response from the bus to check whether the LIN node sends data correctly or not. When the LIN node finishes sending, it will notify an interrupt for an user and stop to send. If an user wants to start to wake up another LIN node again, users must reconfigure the LIN to wakeup mode again.

(2) In the LIN node side which is waked up, when this node completes to recognize the pattern bits, the slave will assert an interrupt for users. The node will exit the wakeup mode when users reset and configure the mode.

(3) There is no time out in the wake up processing.

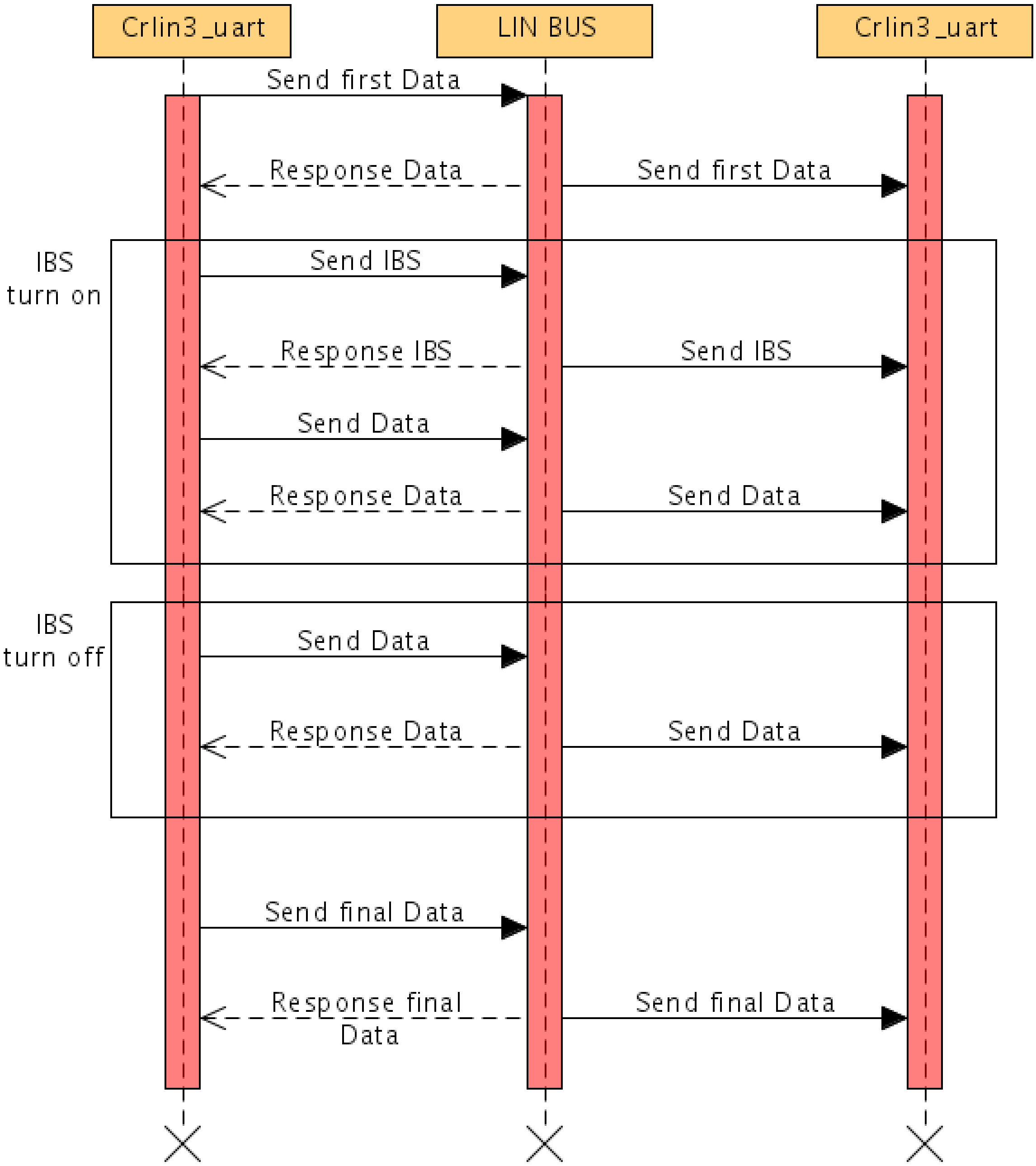


Figure 7.4: UART transaction

Explanation :

(1) In the UART mode, when an user wants to transfer data, this user can choose between two modes : single byte or multi-bytes.

(2) In the single byte, then when the register LUTDR or LUWTDR is updated, the UART will transfer this data. After it finishes transferring, if the IBS mode is turned on, then IBS will be transferred. In reserver, the transaction will be completed.

(3) In the multi-bytes, then when the UTOE is turned on, the UART will transfer data in LDBRn (n = 1 -> 8)

(4) In between each data, if the IBS is turned on, the IBS will be sent. In case, an user wants to transfer 9 bytes, the data in the LDBR0 will be sent. If UTOE won't be cleared, this process will begin again.

7.2. State diagram

7.2.1.Main state diagram

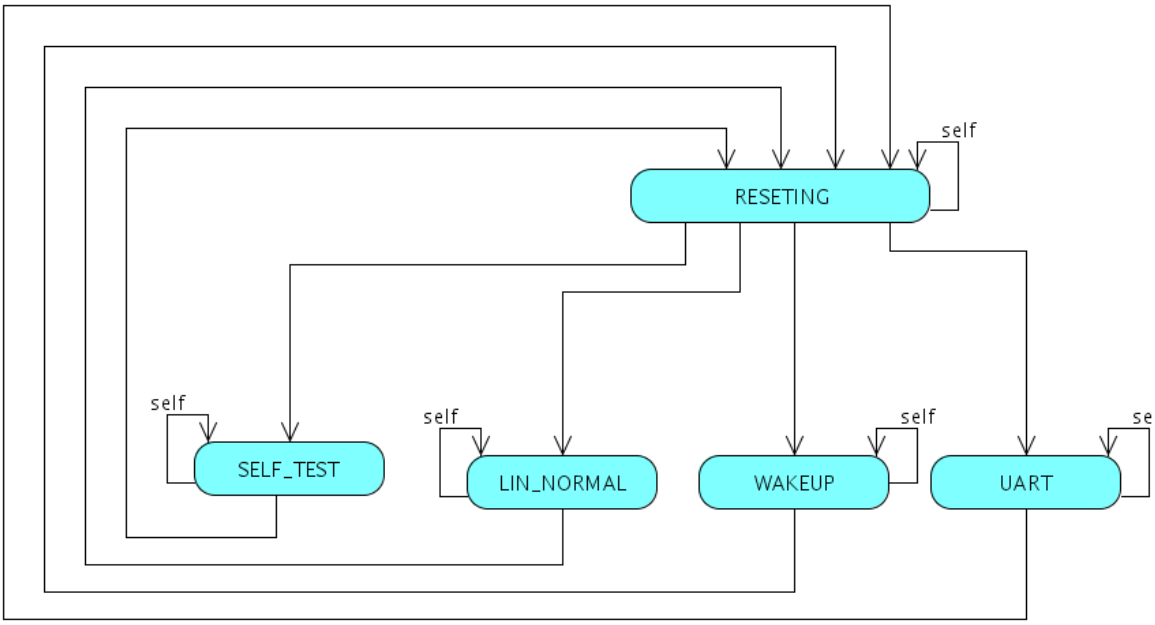


Figure 7.5: Main state diagram of the RLIN3 model

The state transition conditions are described as the table below:

Table 7.2: State transition conditions of main state diagram.

No	State	Actions	Transition	Condition
1	RESETING	RLIN3 is reseting.	SELF_TEST	- Write sequence key 0xA7, 0x58 and 0x01 to LSCR[7:0] in reset mode. - LCUC.OM0 = 1. - LCUC.OM1 = 1
			LIN_NORMAL	- LMD=0x00 or 0x02 or 0x03 - LCUC.OM0 = 1 - LCUC.OM1 = 1
			WAKEUP	- LCUC.OM0= 1
			UART	- LMD.LMD=0x1 - LCUC.OM0 = 1
			RESETING	- Other
2	SELF_TEST	Self test operations.	RESETING	- preset_n and rstc_n are asserted - AssertReset is called

No	State	Actions	Transition	Condition
				- LCUC.OM0 = 0
			SELF_TEST	- Other
3	LIN_NORMAL	LIN normal operations.	RESETING	- preset_n and rstc_n are asserted - AssertReset is called - LCUC.OM0 = 0
			LIN_NORMAL	- Other
4	WAKEUP	Wakeup operations.	RESETING	- preset_n and rstc_n are asserted - AssertReset is called - LCUC.OM0 = 0
			WAKEUP	- Other
5	UART	UART operations.	RESETING	- preset_n and rstc_n are asserted - AssertReset is called - LCUC.OM0 = 0
			UART	- Other

Explanation:

(1) RESETTING: RLIN3 is in reseting.

(2) SELF_TEST: This state processes operations in Self Test mode including transmission/reception operations in LIN Master/LIN Slave. RLIN3 issues a transmission/reception successful interrupt for each transmission/ reception data. The figure 7.9 describes the state machine of this state.

(3) LIN_NORMAL: When the RLIN3 model moves to this state, the transactions of LIN Master/LIN Slave modes is progressed. This state processes all operations in LIN mode including transmission/reception operations in LIN Master/LIN Slave modes. The figure 7.6 describes the state machine of this state.

(4) WAKEUP: When the RLIN3 model moves to this state, the transmission/reception wakeup signal to TX_DATA/RX_DATA ports is progressed. The figure 7.7 describes the state machine of this state.

(5) UART: When the RLIN3 model moves to this state, the transmission/reception data between UART nodes in UART mode is progressed. The figure 7.8 describes the state machine of this state.

7.2.2. LIN mode state diagram

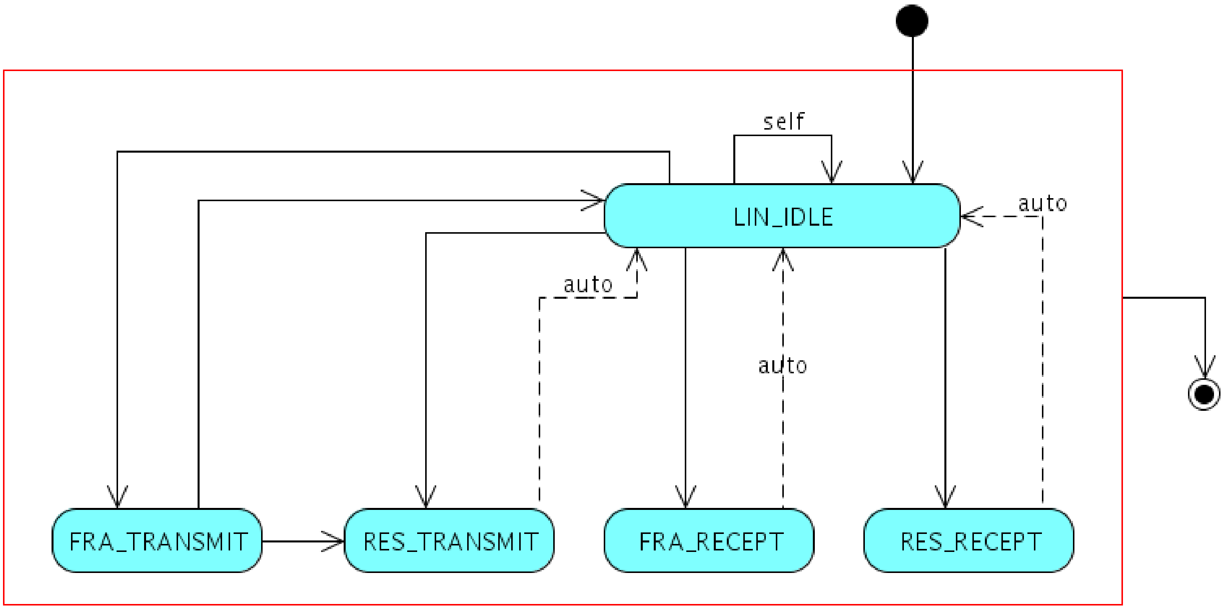


Figure 7.6: State diagram in LIN_NORMAL mode of the RLIN3 model

The state transition conditions are described as the table below:

Table 7.3: State transition conditions of LIN_NORMAL mode state diagram.

No	State	Actions	Transition	Condition
1	LIN_IDLE	Wait for new transaction.	FRA_TRANSMIT	- LTRC.FTS = 1 - LMD = 0
			RES_TRANSMIT	- LTRC.RTS=1 - LDFC[4]=1 - LMD = 0
			FRA_RECEPT	- LTRC.FTS = 1 - LMD = 2 or 3
			RES_RECEPT	- LTRC.RTS=1 - LTRC.FTS=1 - LMD = 0,2,3
			LIN_IDLE	- Other
2	FRA_TRANSMIT	Frame header transmission.	LIN_IDLE	- LDFC.FSM=1
			RES_TRANSMIT	- LDFC.FSM=0
3	RES_TRANSMIT	Response data transmission	LIN_IDLE	- Auto
4	FRA_RECEPT	Frame header reception.	LIN_IDLE	- Auto
5	RES_RECEPT	Response data reception.	LIN_IDLE	- Auto

Explanation:

(1) FRA_TRANSMIT: When the RLIN3 model moves to this state, the frame header is transmitted. Refer to figure 7.22 for frame header transmission process. After finishing, if frame combine mode is selected (FSM in LDFC equal to 0), the RLIN3 model moves to RES_TRANSMIT state for transmitting response data. If frame separate is selected, RLIN3 move to LIN_IDLE state and waiting for new transaction.

(2) FRA_RECEPT: When RLIN3 move to this state the frame header reception process is progressed. Refer to the figure 7.26 for frame header reception process. RLIN3 moves to LIN_IDLE state and waits for new transaction after frame header reception process is finished.

(3) RES_TRANSMIT: When the RLIN3 model moves to this state, response data are transmitted. Refer to the figure 7.23 for response data transmission process. After finishing, the RLIN3 model moves to LIN_IDLE state and waiting for new transaction.

(4) RES_RECEPT: When the RLIN3 model moves to this state, response data will be received. Refer to the figures 7.24 and 7.25 for response data reception process. After finishing receiving data, the RLIN3 model moves to LIN_IDLE state and waiting for new transaction.

(5) LIN_IDLE: When The RLIN3 model moves to this state, RLIN3 waits for a new transfer.

7.2.3. Wakeup mode state diagram

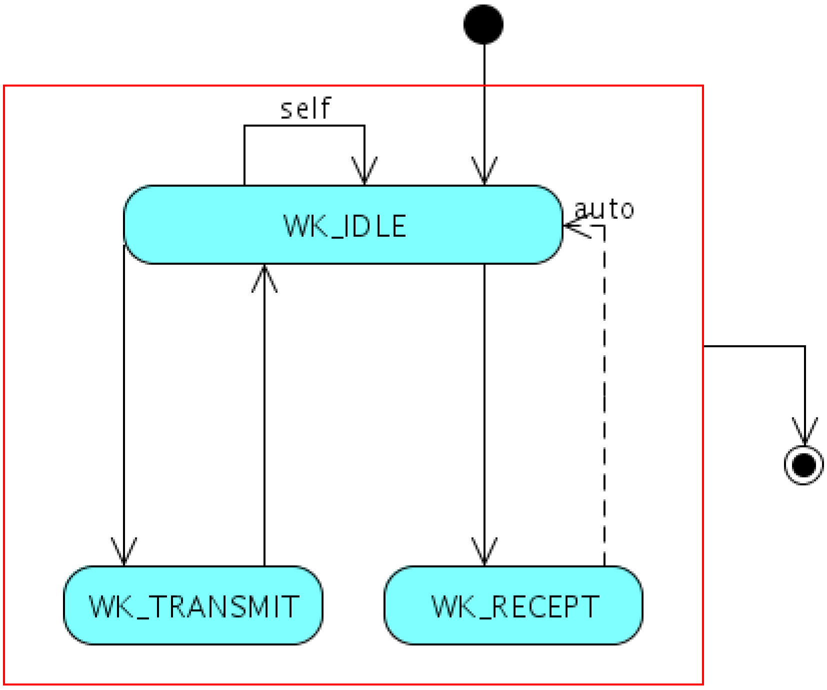


Figure 7.7: State diagram in WAKEUP mode of the RLIN3 model

The state transition conditions are described as the table below:

Table 7.4: State transition conditions of WAKEUP mode state diagram.

No	State	Actions	Transition	Condition
1	WK_IDLE	Wait for new wakeup transaction	WK_TRANSMIT	- LDFC[4]=1
			WK_RECEPT	- LDFC[4]=0
			WK_IDLE	- Other
2	WK_TRANSMIT	Wakeup transmission operations	WK_IDLE	- LTRC.FTS = 0
3	WK_RECEPT	Wakeup reception operations	WK_IDLE	- Auto

Explanation:

(1) WK_TRANSMIT: When the RLIN3 model moves to this state, wakeup signal is transmitted. Refer to the figure 7.21 for wakeup signal transmission process. After finishing, RLIN3 move to WK_IDLE state and waits for a new wakeup signal transmission.

(2) WK_RECEPT: When the RLIN3 model moves to this state, the wakeup signal reception process is progressed. Refer to the figure 7.20 for wakeup reception process. After wakeup signal reception process is finished, if data are recognized as a wake up signal, the RLIN3 model set FTC bit in LST to 1 and issues successful reception interrupt then the RLIN3 moves to WK_IDLE state.

(3) WK_IDLE: When the RLIN3 model moves to this state, the RLIN3 waits for a new transaction wakeup signal.

7.2.4. UART mode state diagram

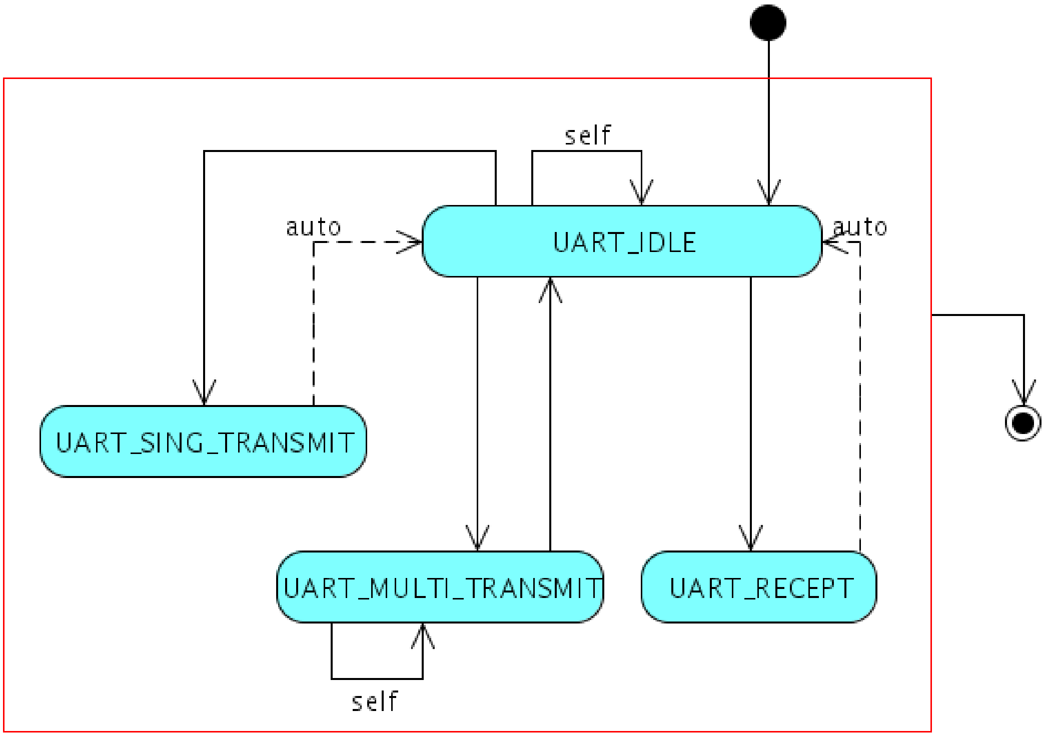


Figure 7.8: State diagram in UART mode of the RLIN3 model

The state transition conditions are described as the table below:

Table 7.5: State transition conditions of UART mode state diagram.

No	State	Actions	Transition	Condition
1	UART_IDLE	Wait for new UART transaction.	UART_SING_TRANSMIT	- LUOER.UTOE=1 - data is written to LUTDR or LUWTDR register
			UART_MULTI_TRANSMIT	- LUOER.UTOE=1 - LTRC.RTS = 1
			UART_RECEPT	- LUOER.UROE=1
			UART_IDLE	- Other
2	UART_SING_TRANSMIT	UART single byte transmission operations.	UART_IDLE	- Auto
3	UART_MULTI_TRANSMIT	UART multi byte transmission operations.	UART_IDLE	- LUOER.UTOE=0
			UART_MULTI_TRANSMIT	- Other
4	UART_RECEPT	UART reception operations.	UART_IDLE	- Auto

Explanation:

(1) UART_MULTI_TRANSMIT: When the RLIN3 model moves to this state and data in the register buffer is transmitted. Refer to the figures 7.18 and 7.23 for UART multi-byte

transmission process. After finishing data transmission, the RLIN3 model moves to UART_IDLE state and waits for new transaction.

(2) UART_SING_TRANSMIT: When RLIN3 moves to this state, the data in the register LUTDR or LUWTDR are transmitted depended on the selection stop bit completion before transferring new data. Refer to the figures 7.18 and 7.23 for the UART single byte transmission process. After finishing data transmission, the RLIN3 model moves to UART_IDLE state and waits for new transaction.

(3) UART_RECEPT: When the RLIN3 model moves to this state and data are received. Refer to the figures 7.24 and 7.25 for the UART reception process . After finishing data reception, the RLIN3 model moves to UART_IDLE state and waits for new transaction.

(4) UART_IDLE: When the RLIN3 model moves to this state, the RLIN3 waits for a new transfer.

7.2.5. Self Test mode state diagram

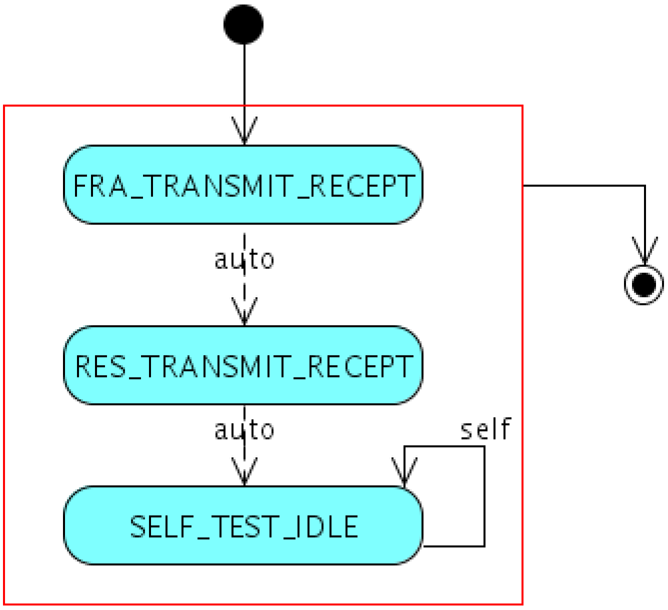


Figure 7.9: State diagram in SELF_TEST mode of the RLIN3 model

The state transition conditions are described as the table below:

Table 7.6: State transition conditions of Self Test mode state diagram.

No	State	Actions	Transition	Condition
1	FRA_TRANSMIT_RECEPT	Frame header transmission / reception operations	RES_TRANSMIT_RECEPT	- Auto
2	RES_TRANSMIT_RECEPT	Response data transmission / reception operations	SELF_TEST_IDLE	- Auto
3	SELF_TEST_IDLE	Wait for new Self Test transaction.	SELF_TEST_IDLE	- Other

Explanation:

(1) In Self Mode, the transmission data are loop backed to reception data. Users can not

access to registers. There are two main sub states FRA_TRANSMIT_RECEPT and RES_TRANSMIT_RECEPT in this state.

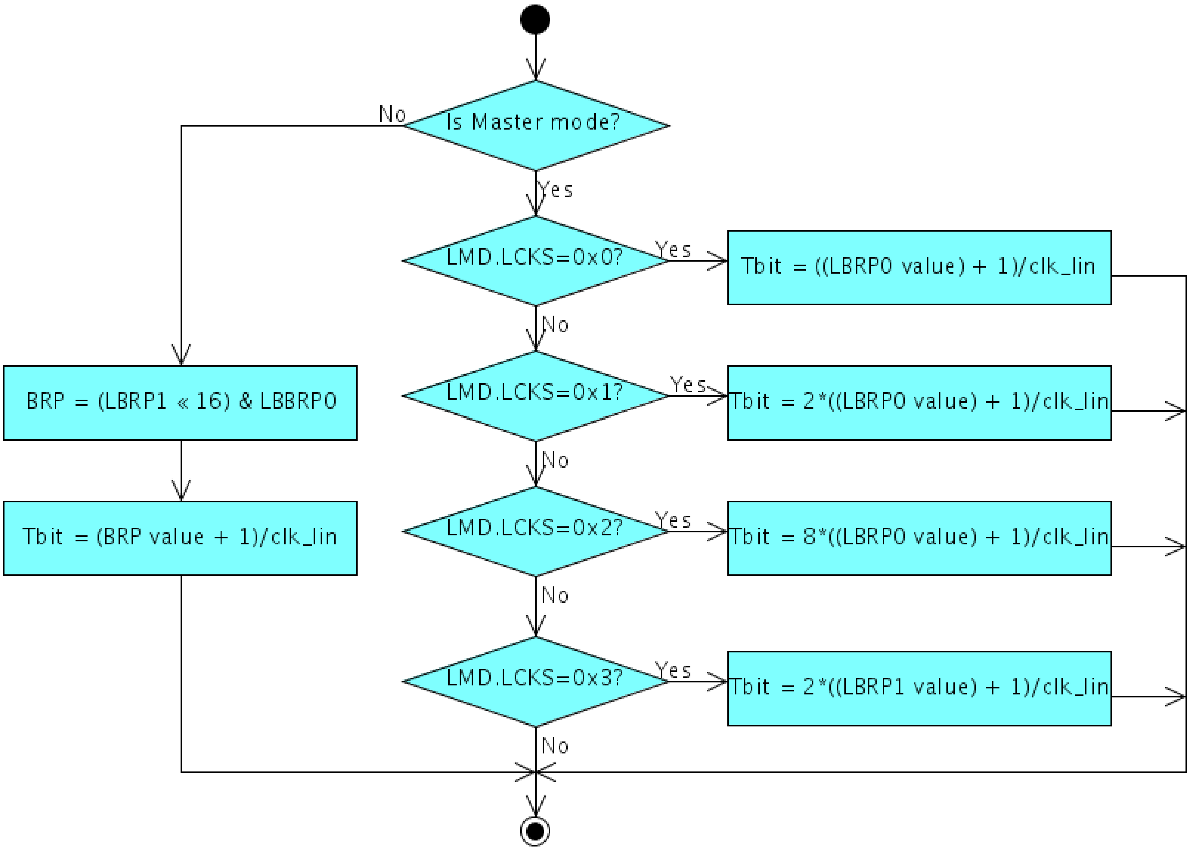
(2) FRA_TRANSMIT_RECEPT: When RLIN3 moves to this state, the frame header transmission process and frame header reception process are progressed in parallel. Refer to the figure 7.22 for the frame header transmission process and refer to the figure 7.26 for the frame header reception process. After finishing frame header transmission and reception processes, RLIN3 moves to RES_TRANSMIT_RECEPT and continues to do response data operations.

(3) RES_TRANSMIT_RECEPT: When RLIN3 moves to this state, the response data transmission process and response data reception process are progressed in parallel. Refer to the figure 7.23 for the response data transmission process and refer to the figures 7.24 and 7.25 for the response data reception process. This state will be finished until all the data are transmitted. After finishing response data transmission and reception processes, RLIN3 moves to SELF_TEST_IDLE state and waits for next actions by users.

(4) SELF_TEST_IDLE: When the RLIN3 model moves to this state, the RLIN3 waits for a action from users.

(5) To exit Self Test mode, users turn on the SW reset and write 1 to LSTM bit in LSTC register, the RLIN3 return to LIN normal mode after reseting.

7.3. Bit time calculating process



Is Master mode?
No
Yes
LMD.LCKS=0x0?
Yes
Tbit = ((LBRP0 value) + 1)/clk_lin
No
LMD.LCKS=0x1?
Yes
Tbit = 2*((LBRP0 value) + 1)/clk_lin
No
LMD.LCKS=0x2?
Yes
Tbit = 8*((LBRP0 value) + 1)/clk_lin
No
LMD.LCKS=0x3?
Yes
Tbit = 2*((LBRP1 value) + 1)/clk_lin
No
BRP = (LBRP1 « 16) & LBBRP0
Tbit = (BRP value + 1)/clk_lin

Figure 7.10: Bit time calculation process flow

Explanation:

(1) This function is called after reset is de-asserted.

(2) The bit time for LIN communication is calculated base on Baud rate setting on the bits LCKS[1:0] in LMD register. The values of two registers LBRP0 and LBRP1 are used to scale the baud rate clock source. If LBRP0 and LBRP1 register is set to 0, this function is not run and the operations are stopped.

(3) There are 4 baud rate clock sources available in Master mode corresponding to 4 formulas to calculate bit time value. The selection of LMD.LCKS[1:0] will determine which bit time's formula is used.

(4) In Slave mode and UART mode, there is only one baud rate clock source available. The bit time is calculated base on value of BRP register. The BRP register is combined by LBPRP0 and LBRP1 registers.

7.4. DeAssertIntrMethod

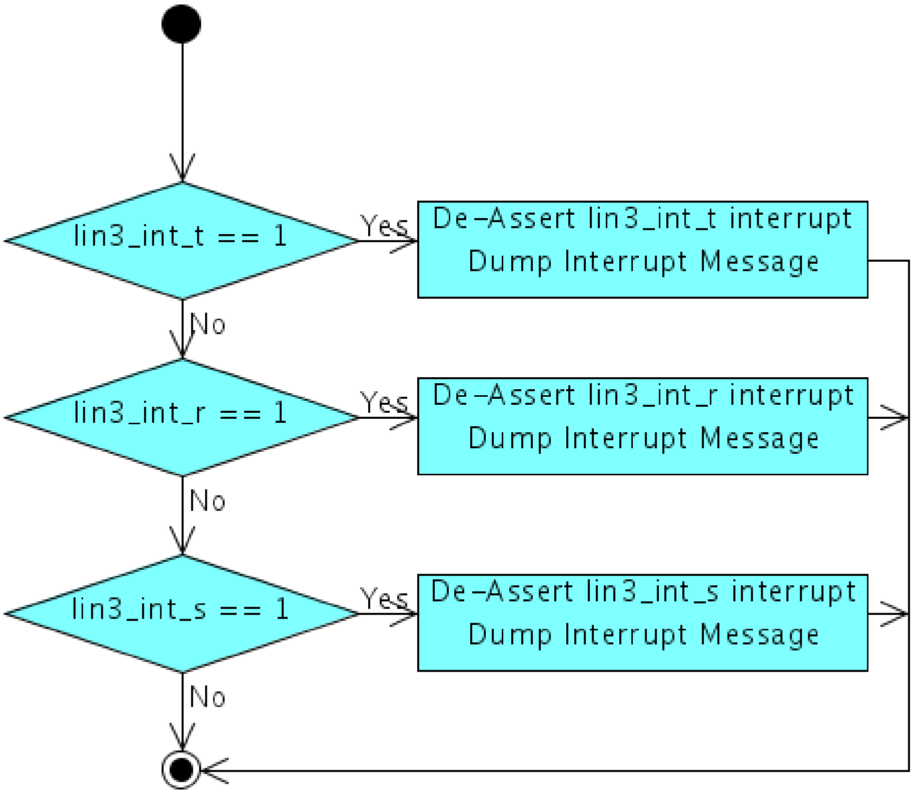


Figure 7.11: De-assert interrupt Flow

Explanation:

(1) This function is called after interrupt pins are asserted after one clock circle.

(2) Any interrupt pins lin3_int_t, lin3_int_r and lin3_int_s are equal 1, the interrupt pin will be

clear to 0 and dump interrupt information.

7.5. ResetMethod

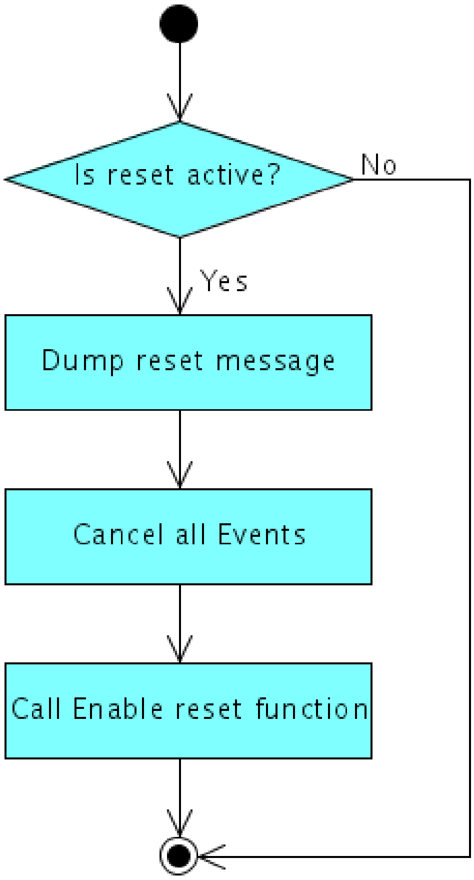


Figure 7.12: Reset method flow

Explanation:

(1) If *rstc_n* and *preset_n* ports is asserted, ResetMethod will be called.

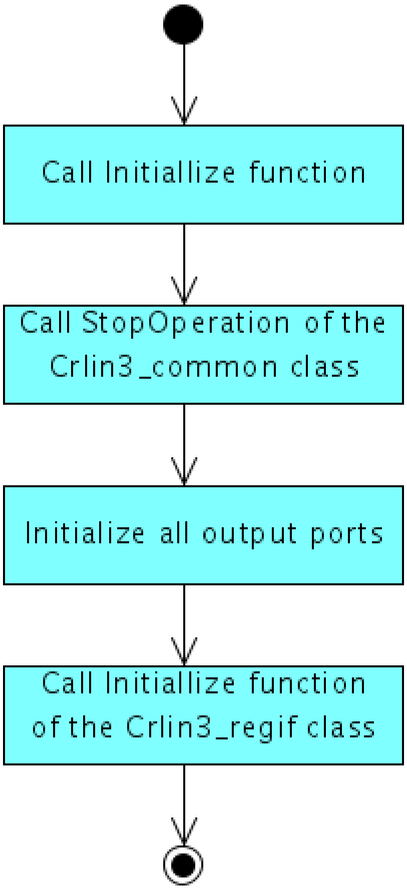
(2) If AssertReset command is called, mCmdResetEvent will be notified. CmdResetMethod will be called.

(3) ResetMethod and CmdResetMethod will call EnableReset function to reset the RLIN3 model.

(4) EnableReset will initialize all data members and output ports, cancel all events, notify mAssertResetEvent to reset all operations threads. At last, it calls the reset function of Crlin3 class and Crlin3_regif class.

(5) However, the parameters set by commandIF will not be cleared.

7.6. EnableReset



Call Initiallize function
Call StopOperation of the Crlin3_common class
Initialize all output ports
Call Initiallize function of the Crlin3_regif class

Figure 7.13: Enable Reset flow

Explanation:

(1) This functions is called when preset_n and rstc_n ports are asserted or handleCommand reset is asserted.

(2) When this function is called, all variables, ports, registers are initialized and all operations are stopped.

7.7. GetRegBitsVal

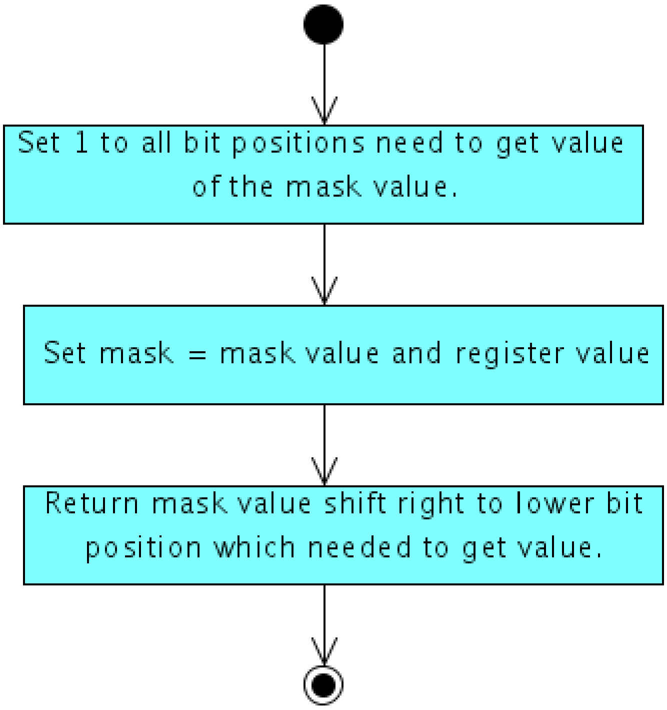

Set 1 to all bit positions need to get value of the mask value.
Set mask = mask value and register value
Return mask value shift right to lower bit position which needed to get value.

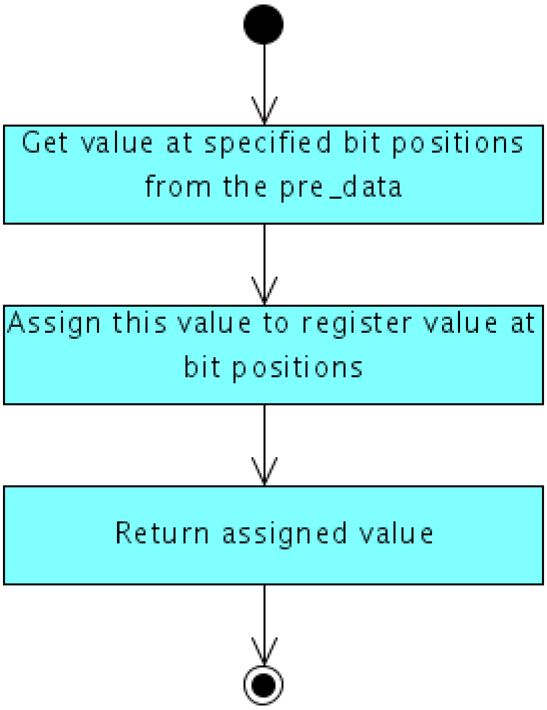
Figure 7.14: Get register bits value flow

Explanation:

(1) This function is called when users want to get the value of number of bits in the register.

(2) The mask is created depended on the bit position which users want to get the value. For producing the gotten value, this mask will be AND with the register value and shift the lower bit position which needed to get the value.

7.8. SetRegBitsVal



Get value at specified bit positions from the pre_data
Assign this value to register value at bit positions
Return assigned value

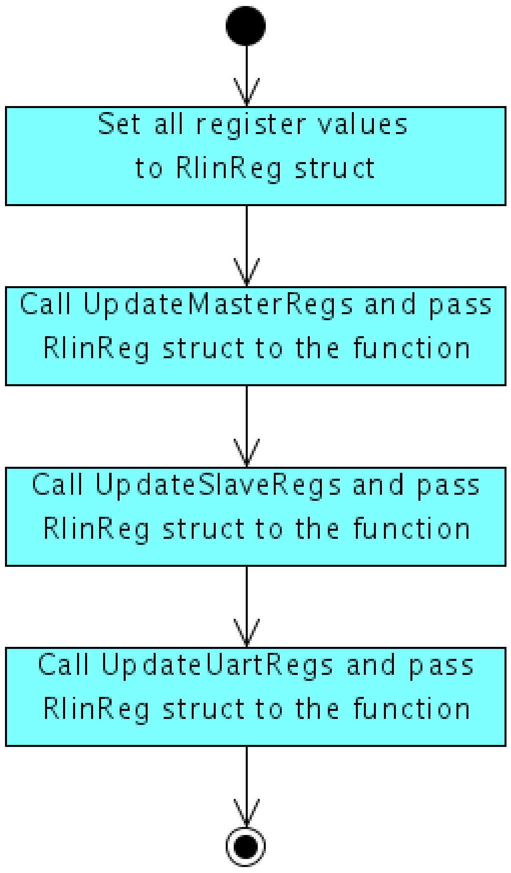
Figure 7.15: Set register bit value flow

Explanation:

(1) This function is called when users want to set the value of number of bits in the register.

(2) The set value is gotten in pre_data variable at the specified bit positions. This set value is assigned to register value at the same specified bit positions.

7.9. UpdateAllRegs



Set all register values to RlinReg struct
Call UpdateMasterRegs and pass RlinReg struct to the function
Call UpdateSlaveRegs and pass RlinReg struct to the function
Call UpdateUartRegs and pass RlinReg struct to the function

Figure 7.16: Update all registers flow

Explanation:

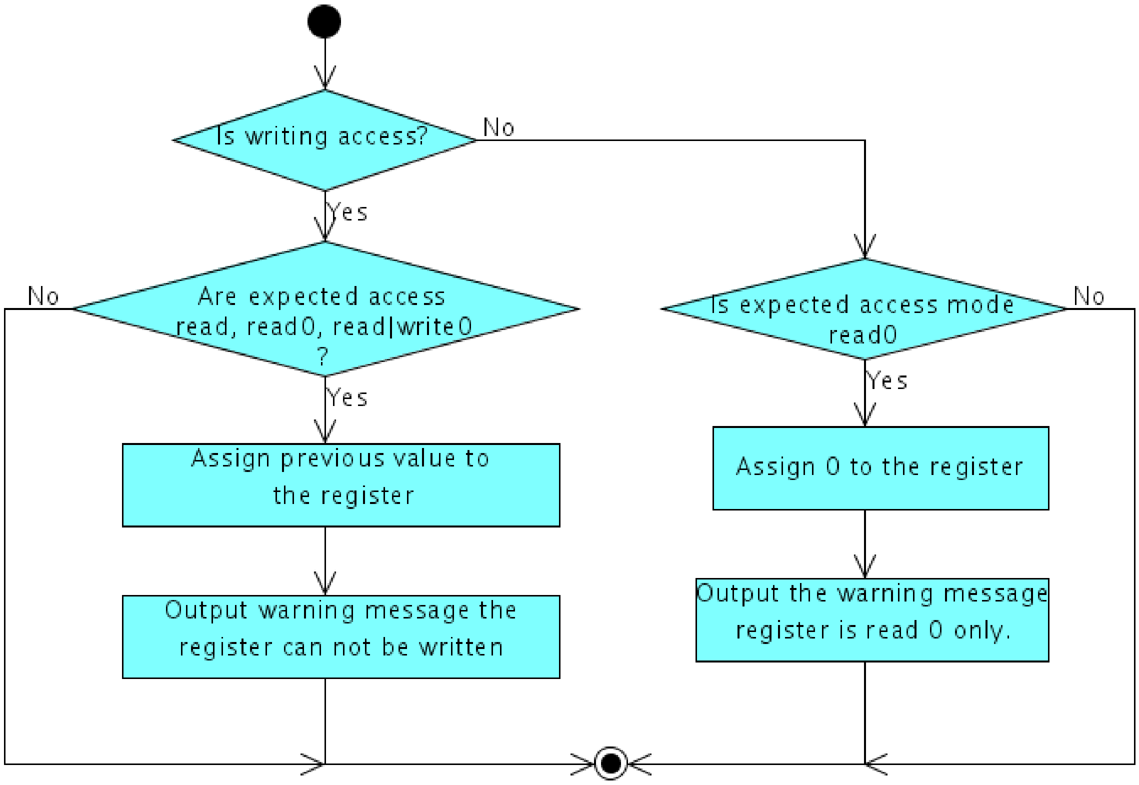
(1) This function is called when reset is active.

(2) The RlinReg struct contains all register values. The struct is used to update value of all registers.

(3) Users get the value of all registers and update to RlinReg struct.

(4) User call UpdateMasterRegs, UpdateSlaveRegs and UpdateUartRegs functions and pass this struct as argument for updating register values of Master,Slave and Uart classes.

7.10. RegisterAccessCheck



Is writing access?
No
Yes
No
Are expected access read, read0, read|write0 ?
Is expected access mode read0
No
Yes
Yes
Assign previous value to the register
Assign 0 to the register
Output warning message the register can not be written
Output the warning message register is read 0 only.

Figure 7.17: Register access mode check flow

Explanation:

(1) This function is called when users want to check register access mode in Reset, IDLE and Operation modes.

(2) If register access is writing and expected access mode is Read/Read0/Read-Write0. The Write operation is prohibited and a message is dumped.

(3) If register access is reading and expected access mode is Read0. The read value is assigned to 0 and a message is dumped.

7.11. TransmitDataMethod

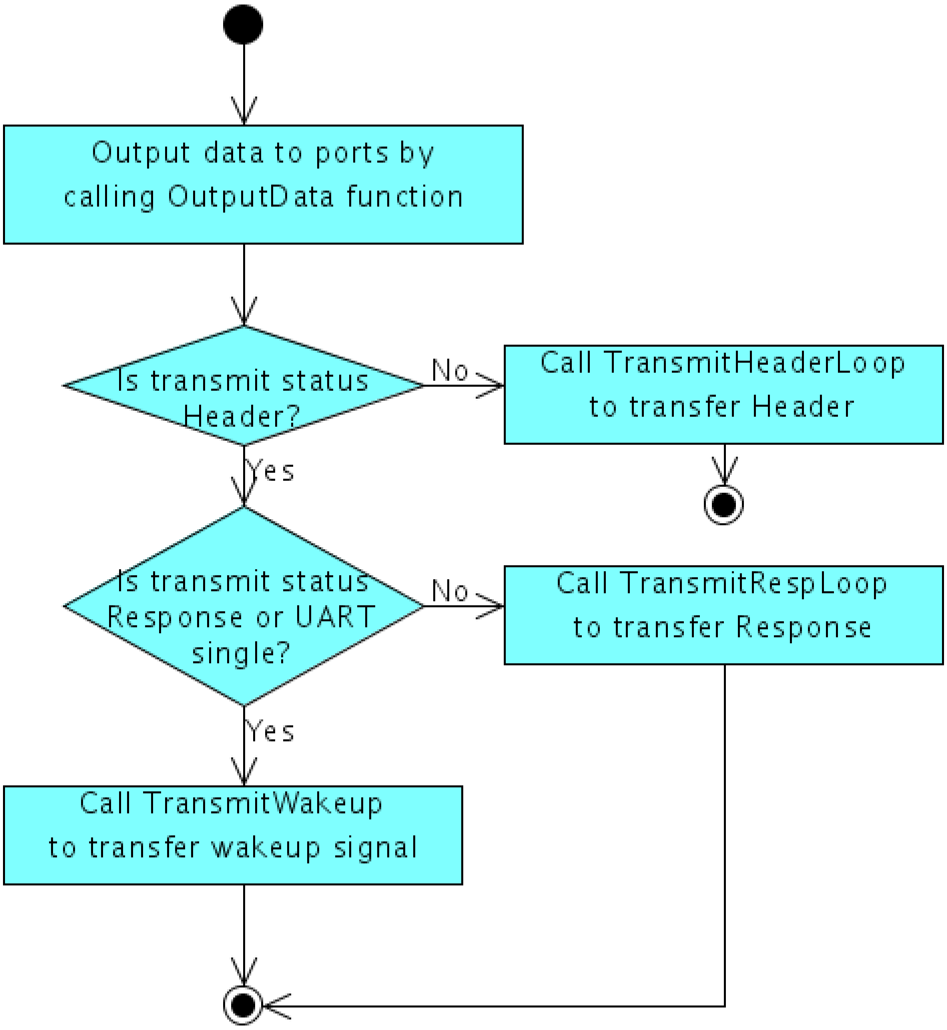

Output data to ports by calling OutputData function
Is transmit status Header?
No
Call TransmitHeaderLoop to transfer Header
Yes
Is transmit status Response or UART single?
No
Call TransmitRespLoop to transfer Response
Yes
Call TransmitWakeup to transfer wakeup signal

Figure 7.18: Transmit data method flow

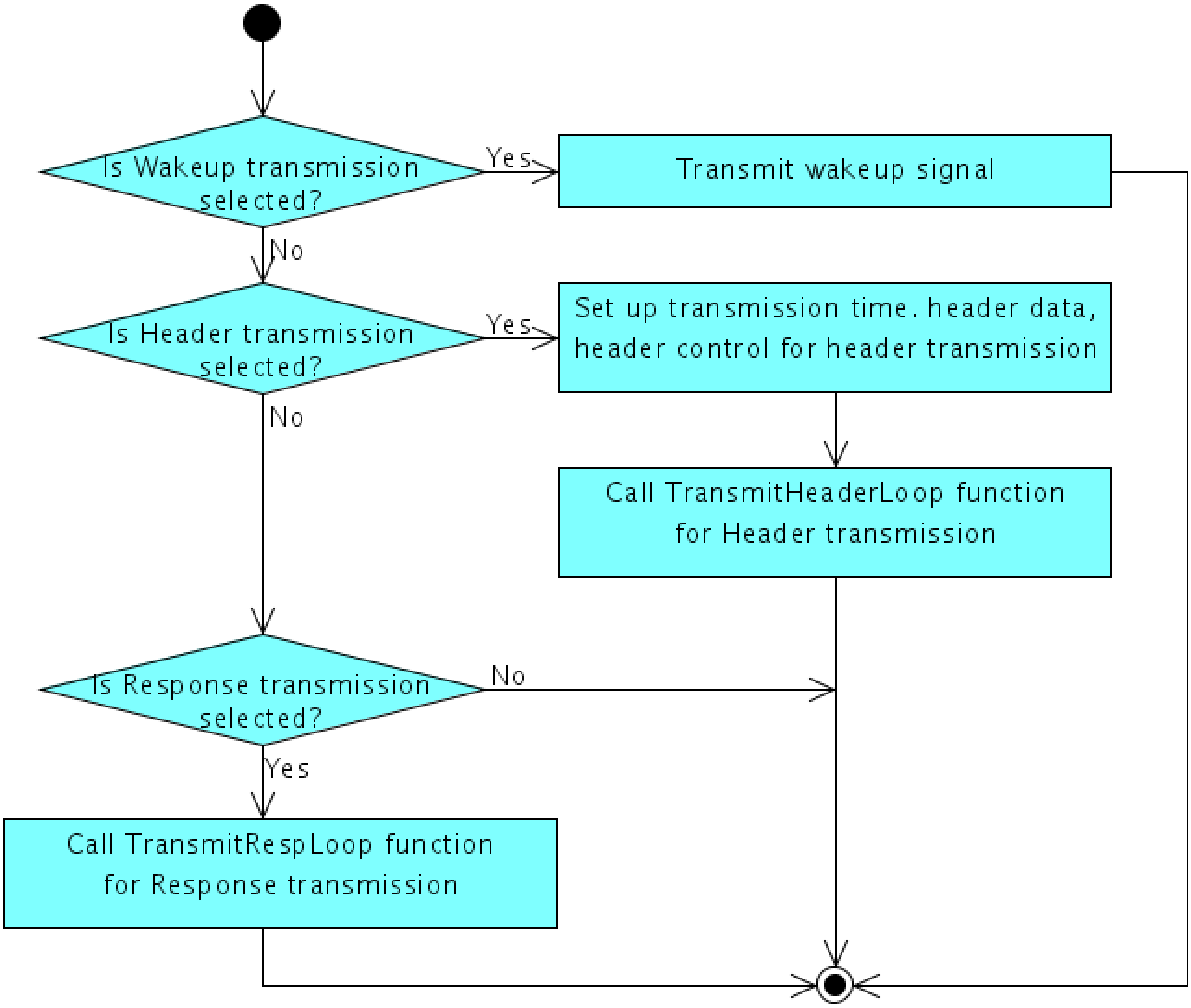
Explanation:

(1) This method sensitive with mTransmitEvent. This function is notified by TransmitHeaderLoop, TransmitRespLoop and TransmitWakeup functions to output the data to ports.

(2) The function check the transmitted status to determine which kind of data are transferred to the output ports.

(3) After transmitted the data to output ports, the TransmitHeaderLoop function is called for Header transmission. The TransmitRespLoop is called for Response transmission. The TransmitWakeup is called for Wakeup signal transmission.

7.12. TransmitProcess



Is Wakeup transmission selected?
Yes
Transmit wakeup signal
No
Is Header transmission selected?
Yes
Set up transmission time. header data, header control for header transmission
No
Call TransmitHeaderLoop function for Header transmission
Is Response transmission selected?
No
Yes
Call TransmitRespLoop function for Response transmission

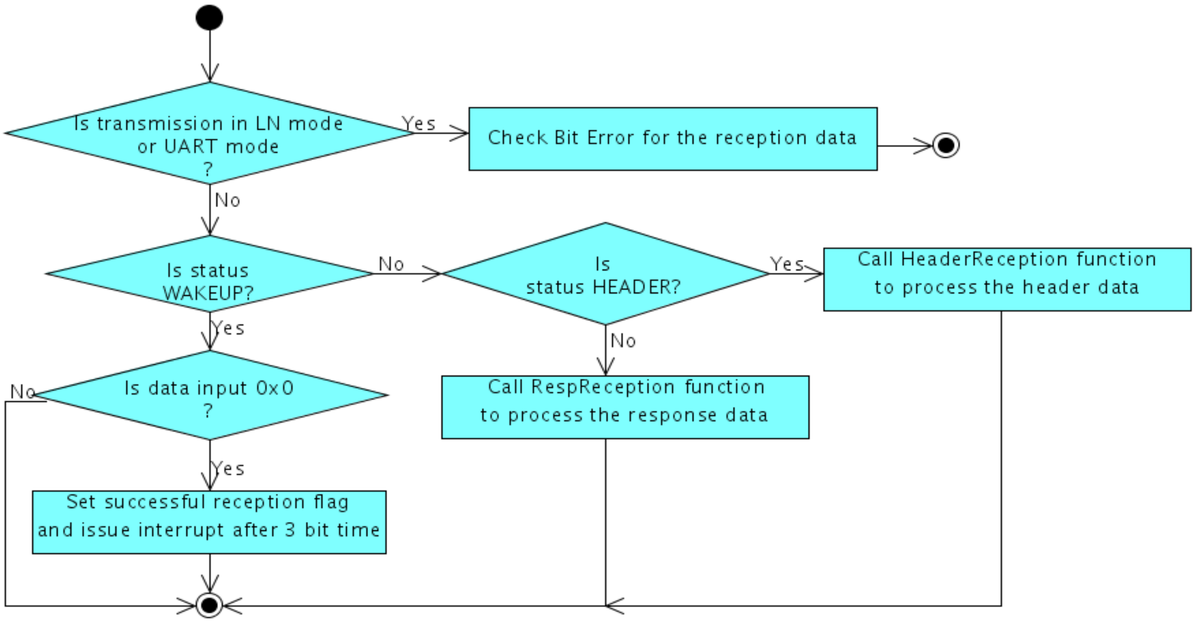
Figure 7.19: Transmit process flow

Explanation:

(1) This method is called when users want to transfer a frame Header, Response data or Wakeup signal.

(2) If the Wakeup transmission is selected, the TransmitWakeup is called for wakeup signal transmission. If the Header transmission is selected, the TransmitHeaderLoop is called for frame Header transmission. If the Response transmission is selected, the TransmitRespLoop is called for Response data transmission.

7.13. ReceptionProcess



Is transmission in LN mode or UART mode ?
Yes
Check Bit Error for the reception data
No
Is status WAKEUP?
No
Is status HEADER?
Yes
Call HeaderReception function to process the header data
Yes
No
Is data input 0x0 ?
No
Call RespReception function to process the response data
Yes
Set successful reception flag and issue interrupt after 3 bit time

Figure 7.20: Reception process flow

Explanation:

(1) This function is called when data arrive at RX_DATA and RX_CONTROL ports.

(2) If the transmission is selected, the received data is compared with transmitted data for Bit Error checking.

(3) If the status is Wakeup and wakeup signal is received successfully, the reception complete flag and interrupt will assert after 3 bit time.

(4) The HeaderReception function is callled if transmission status is Header. Otherwise, the RespReception is called.

7.14. TransmitWakeup

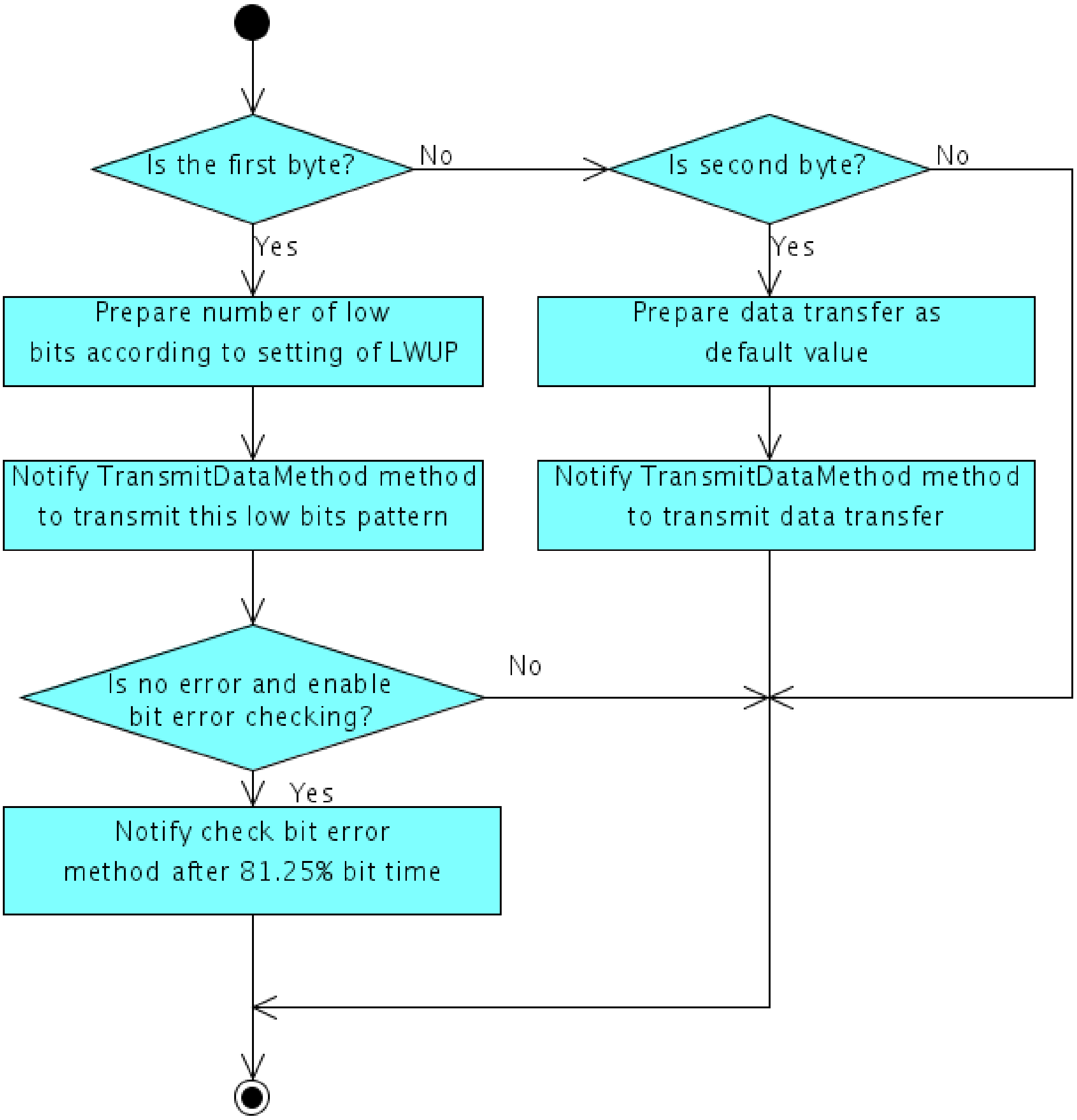

Is the first byte?
No
Is second byte?
No
Yes
Yes
Prepare number of low bits according to setting of LWUP
Prepare data transfer as default value
Notify TransmitDataMethod method to transmit this low bits pattern
Notify TransmitDataMethod method to transmit data transfer
Is no error and enable bit error checking?
No
Yes
Notify check bit error method after 81.25% bit time

Figure 7.21: Transmit Wakeup flow

Explanation:

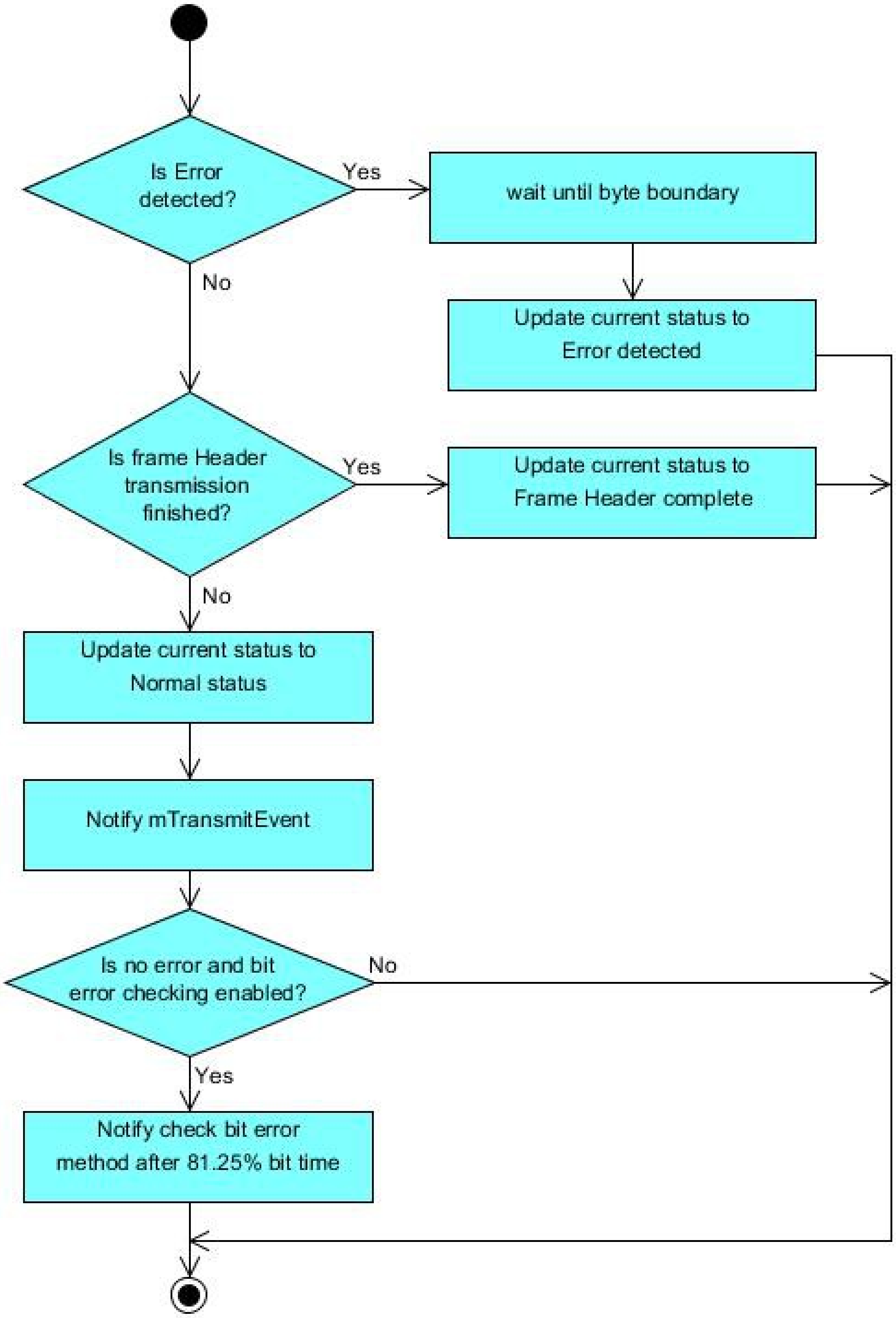
(1) This functions is called by TransmitProcess and TransmitDataMethod functions.

(2) In the first byte of wakeup signal transmission, a number of LOW bit pattern is transferred depended on the setting of LWUP.

(3) In the second byte of wakeup signal transmission, the default data 0xFFFFFFFF is transferred.

(4) If there is no error and checking bit error is enable, check bit error method will be notified after 81.25% bit time.

7.15. TransmitHeaderLoop



Is Error detected?
Yes
wait until byte boundary
No
Update current status to Error detected
Is frame Header transmission finished?
Yes
Update current status to Frame Header complete
No
Update current status to Normal status
Notify mTransmitEvent
Is no error and bit error checking enabled?
No
Yes
Notify check bit error method after 81.25% bit time

Figure 7.22: Transmit Header loop flow

Explanation:

(1) This functions is called by TransmitProcess and TransmitDataMethod functions.

(2) Frame Header is transferred in this function. If frame Header is transferred successfully,

the current status is updated to Frame Header complete.

(3) The mTransmitEvent is notified to activate TransmitDataMethod.

(4) If an Error occurred, the current status is updated to Error detected.

(5) If there is no error and checking bit error is enable, check bit error method will be notified after 81.25% bit time.

7.16. TransmitRespLoop

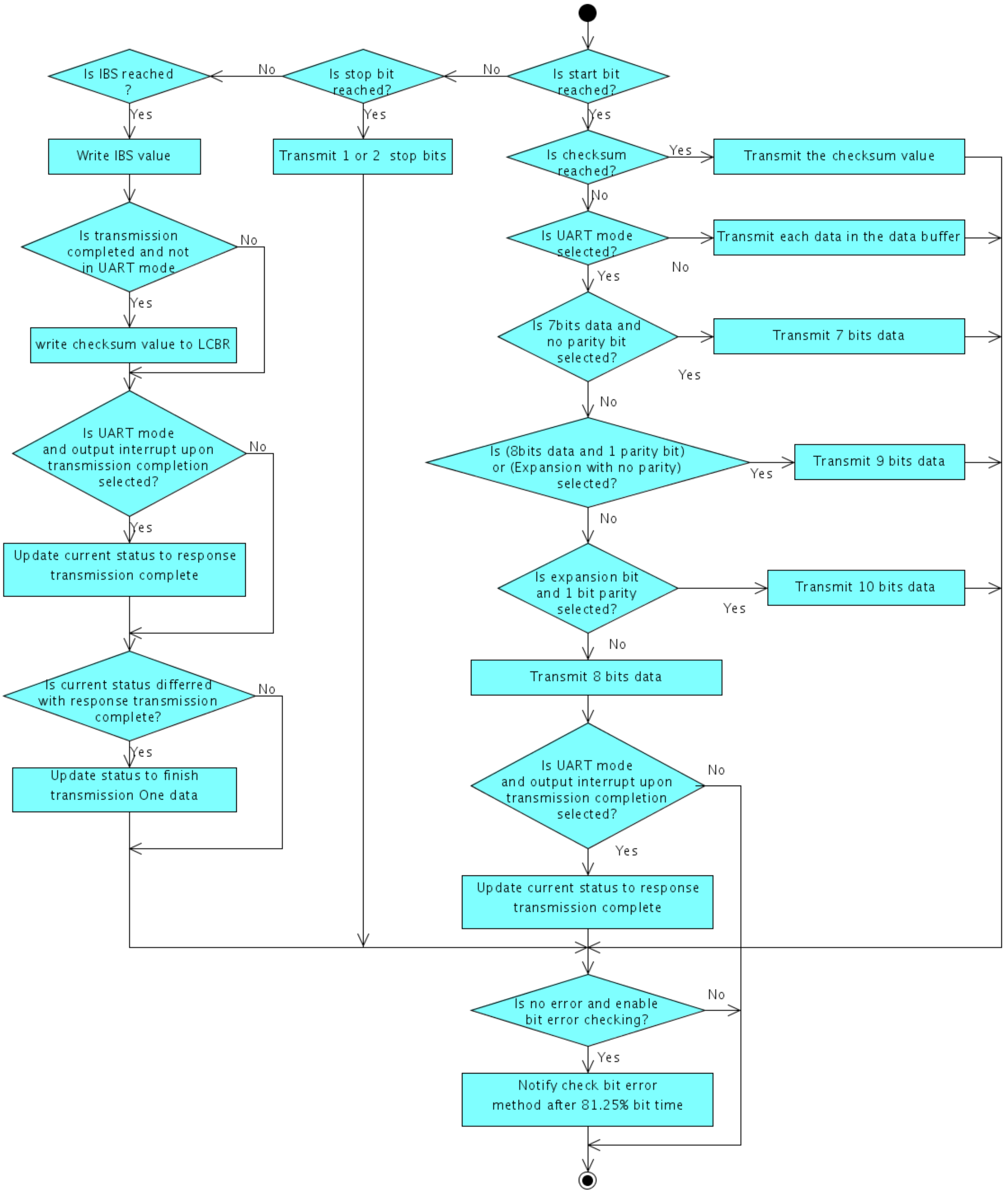

Is start bit reached?
No
Is stop bit reached?
No
Is IBS reached ?
Yes
Write IBS value
Yes
Transmit 1 or 2 stop bits
Yes
Is checksum reached?
Yes
Transmit the checksum value
No
Is UART mode selected?
No
Transmit each data in the data buffer
Yes
Is 7bits data and no parity bit selected?
Yes
Transmit 7 bits data
No
Is (8bits data and 1 parity bit) or (Expansion with no parity) selected?
Yes
Transmit 9 bits data
No
Is expansion bit and 1 bit parity selected?
Yes
Transmit 10 bits data
No
Transmit 8 bits data
Is UART mode and output interrupt upon transmission completion selected?
No
Yes
Update current status to response transmission complete
Is transmission completed and not in UART mode
No
Yes
write checksum value to LCBR
Is UART mode and output interrupt upon transmission completion selected?
No
Yes
Update current status to response transmission complete
Is current status differred with response transmission complete?
No
Yes
Update status to finish transmission One data
Is no error and enable bit error checking?
No
Yes
Notify check bit error method after 81.25% bit time

Figure 7.23: Transmit Response data loop flow

Explanation:

(1) This functions is called by TransmitProcess and TransmitDataMethod functions.

(2) At the beginning transmitting start bit, the checksum, data in the data buffer are transmitted in the LIN mode. In normal mode and self-test transmission, checksum value

will be calculated and transfer at end of last data group. In self-test reception, value of checksum buffer LCBR will be transferred instead of auto calculated value.

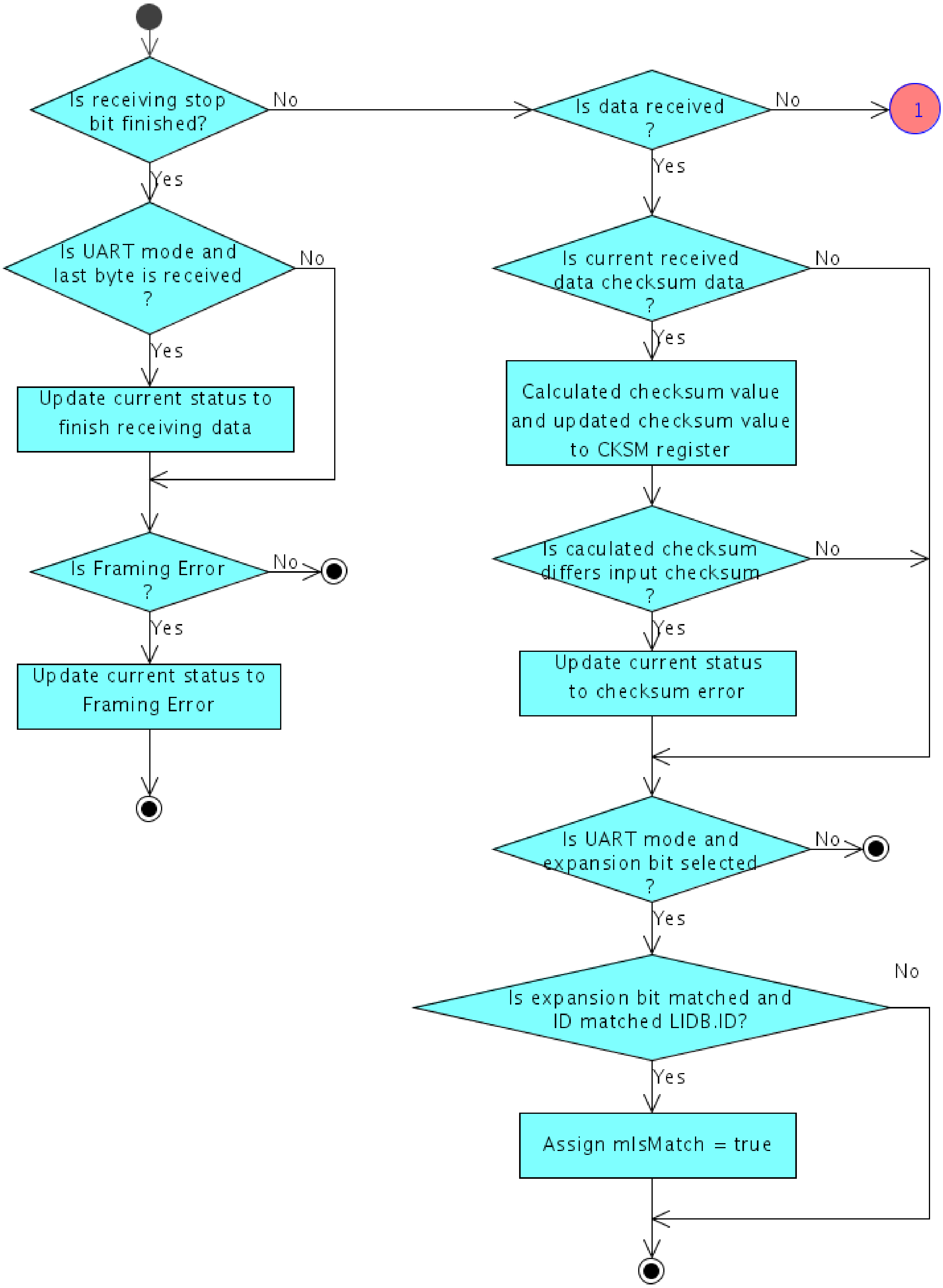
(3) In the UART mode, the 7bits/8bits/9bits data with parity and without parity bit are transmitted depended on the configurations in the control registers.

(4) At the beginning transmitting stop bit, 1 or 2 stop bits are transmitted depended on USBLS bit in LBFC register.

(5) At the beginning transmitting IBS bits, the checksum value is write to LCBR register in the LIN mode. Depended on the status of transmission process, the current status variable is updated to transmission complete or transmission One data complete.

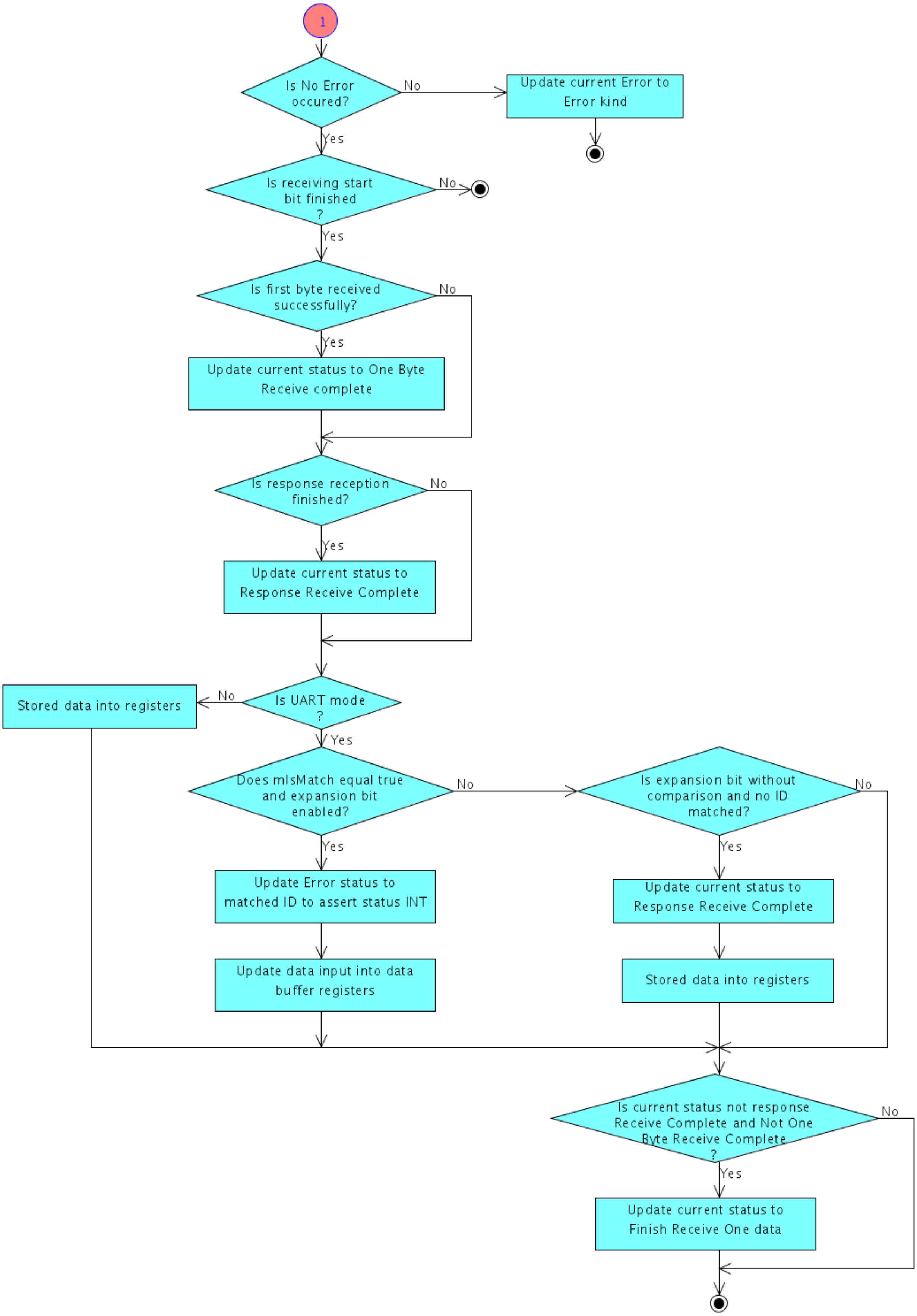
(6) If there is no error and checking bit error is enable, check bit error method will be notified after 81.25% bit time.

7.17. RespReception



Is receiving stop bit finished?
No
Yes
Is data received ?
No
1
Yes
Is UART mode and last byte is received ?
No
Yes
Update current status to finish receiving data
Is current received data checksum data ?
No
Yes
Calculated checksum value and updated checksum value to CKSM register
Is Framing Error ?
No
Yes
Update current status to Framing Error
Is caculated checksum differs input checksum ?
No
Yes
Update current status to checksum error
Is UART mode and expansion bit selected ?
No
Yes
Is expansion bit matched and ID matched LIDB.ID?
No
Yes
Assign mIsMatch = true

Figure 7.24: Response Reception flow (1/2)



1
Is No Error occured?
No
Update current Error to Error kind
Yes
Is receiving start bit finished ?
No
Yes
Is first byte received successfully?
No
Yes
Update current status to One Byte Receive complete
Is response reception finished?
No
Yes
Update current status to Response Receive Complete
Is UART mode ?
No
Stored data into registers
Yes
Does mIsMatch equal true and expansion bit enabled?
No
Is expansion bit without comparison and no ID matched?
No
Yes
Update Error status to matched ID to assert status INT
Update data input into data buffer registers
Yes
Update current status to Response Receive Complete
Stored data into registers
Is current status not response Receive Complete and Not One Byte Receive Complete ?
No
Yes
Update current status to Finish Receive One data

Figure 7.25: Response Reception flow (2/2)

Explanation:

(1) This function is called when received data need to handle in Master/Slave/UART modes.

(2) When stop bit is received successfully, the current status is updated to Finish Receive data if UART mode is selected and received byte is the last byte. The stop bit is also checked to determine Framing Error occurs or not.

(3) If received data is checksum data. The received checksum is compared with calculated checksum. If the result is not matched, the current Error status is updated to checksum Error.

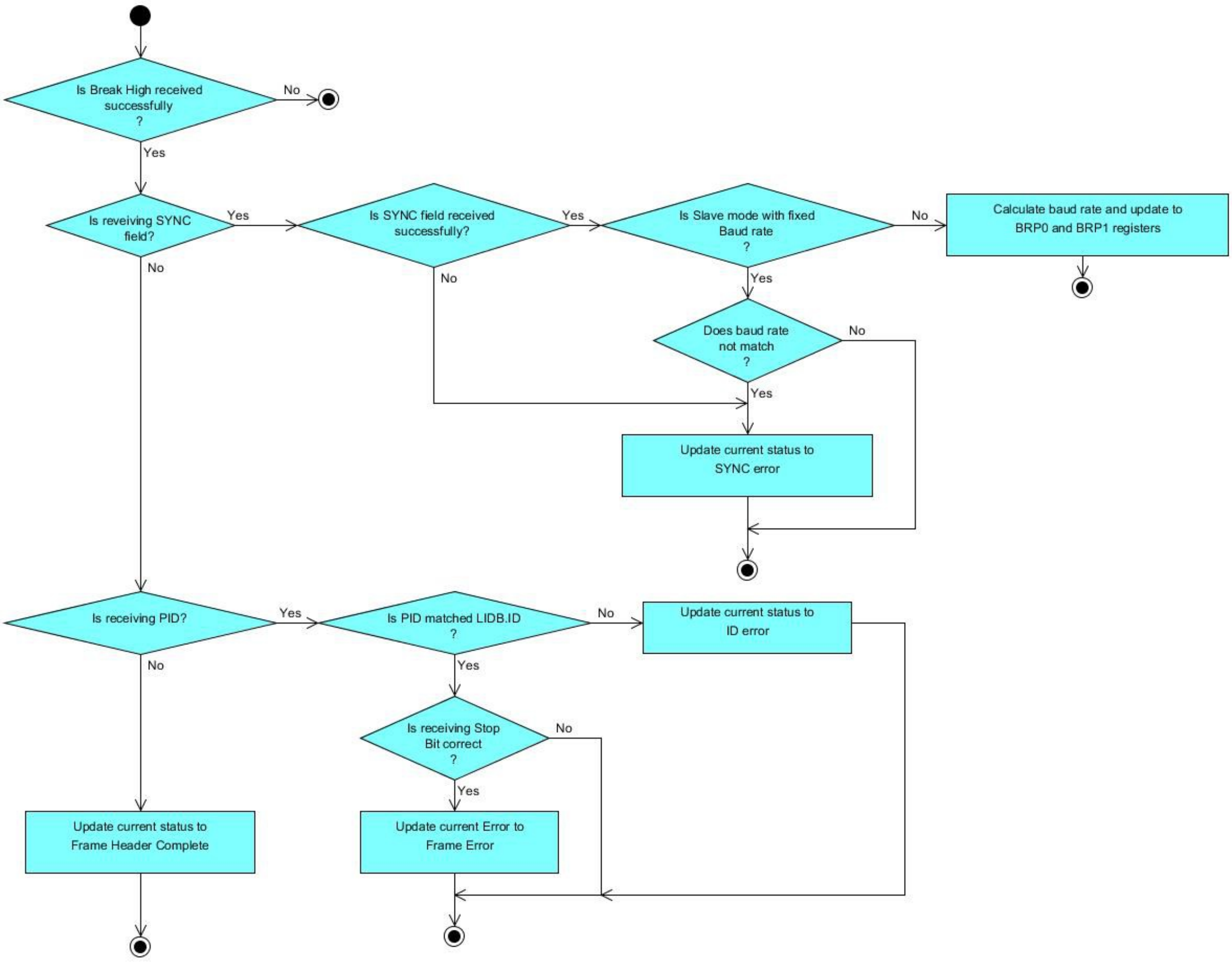
(4) When received data is transfer data, if 9th bit of received data matched the value in UEDBL bit, the expansion bit detection bit is set to 1 and assert the status interrupt. If data comparison is selected, the received data are compared to the value in ID register. If received data matched the value in ID register the matched ID bit specified by LEST.IDMT is set to 1. mIsMatch is assigned true for the updating Error status to ID matched and assert the status interrupt.

(5) When the expansion bit is enable but Expansion Bit Comparison is disabled, the receive data is stored and receive interrupt is asserted

(6) When start bit received successfully, the current status is updated to One Byte Receive Complete if first byte is received successfully. The current status is updated to Response Receive Complete if response reception is finished.

(7) In the UART mode, the function of expansion bit is executed as mention by (4). The received data is stored to data registers. Finally, if the current status is not Response Receive Complete and not One Byte Receive Complete, the current status is updated to Finish Receive One Data.

7.18. HeaderReception



Is Break High received successfully ?
No
Yes
Is reveiving SYNC field?
Yes
No
Is SYNC field received successfully?
Yes
No
Is Slave mode with fixed Baud rate ?
No
Yes
Calculate baud rate and update to BRP0 and BRP1 registers
Does baud rate not match ?
No
Yes
Update current status to SYNC error
Is receiving PID?
Yes
No
Is PID matched LIDB.ID ?
No
Yes
Update current status to ID error
Is receiving Stop Bit correct ?
No
Yes
Update current status to Frame Header Complete
Update current Error to Frame Error

Figure 7.26: Header Reception flow

Explanation:

(1) This function is called when data arrived at input ports and Header is not completed.

(2) When the Break High field is detected successfully, the SYNC field is recognized. If the SYNC field is detected successfully, the baud rate registers BRP0 and BRP1 are updated new baud rate value if Slave mode with auto baud rate is selected.

(3) When SYNC field is received successfully, the recognizing ID field is executed. If the receiving parity bit P0 or P1 is not matched, the current Error status is updated to ID Error. If the receiving stop bit is not correct, current Error status is updated to Framing Error.

(4) At the end, the current status is updated to Frame Header Complete.

7.19. CalcNumOfByte

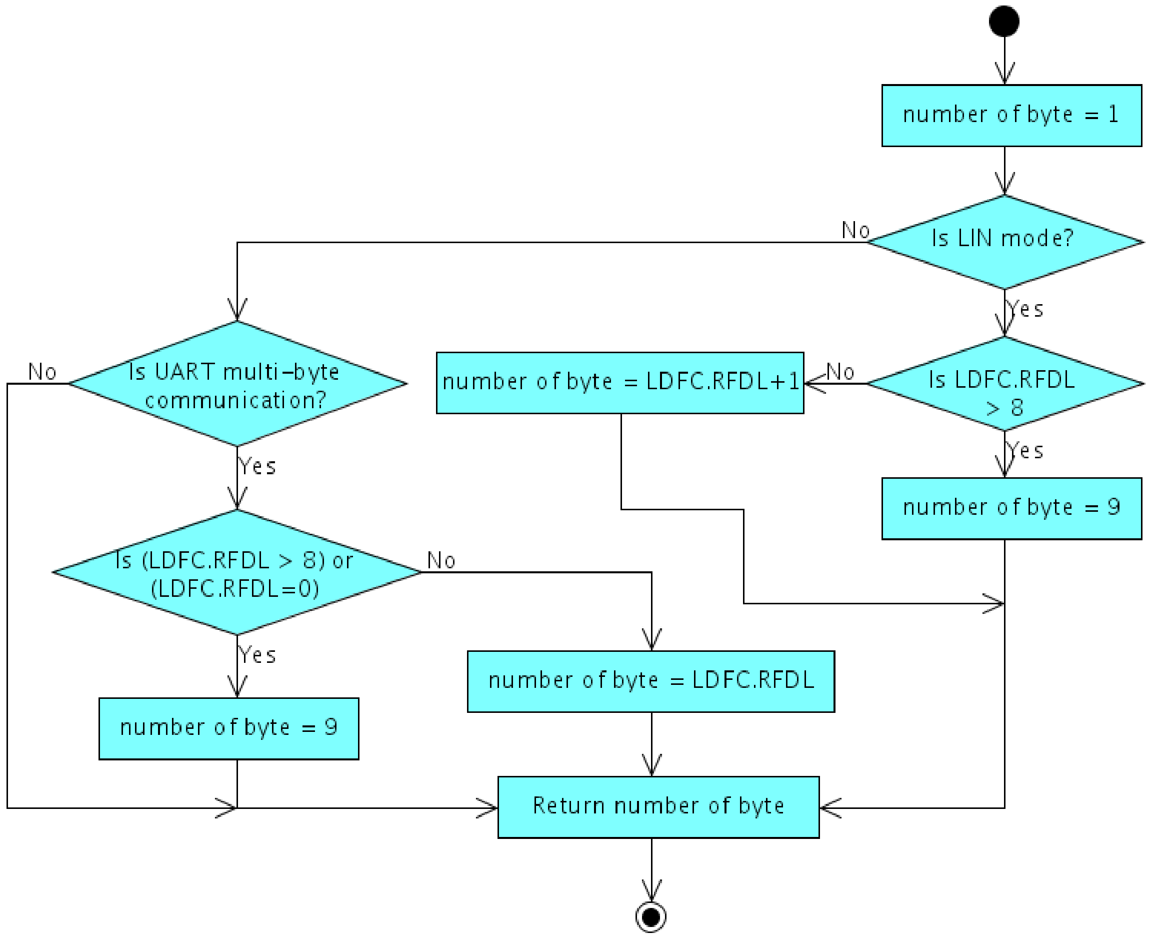


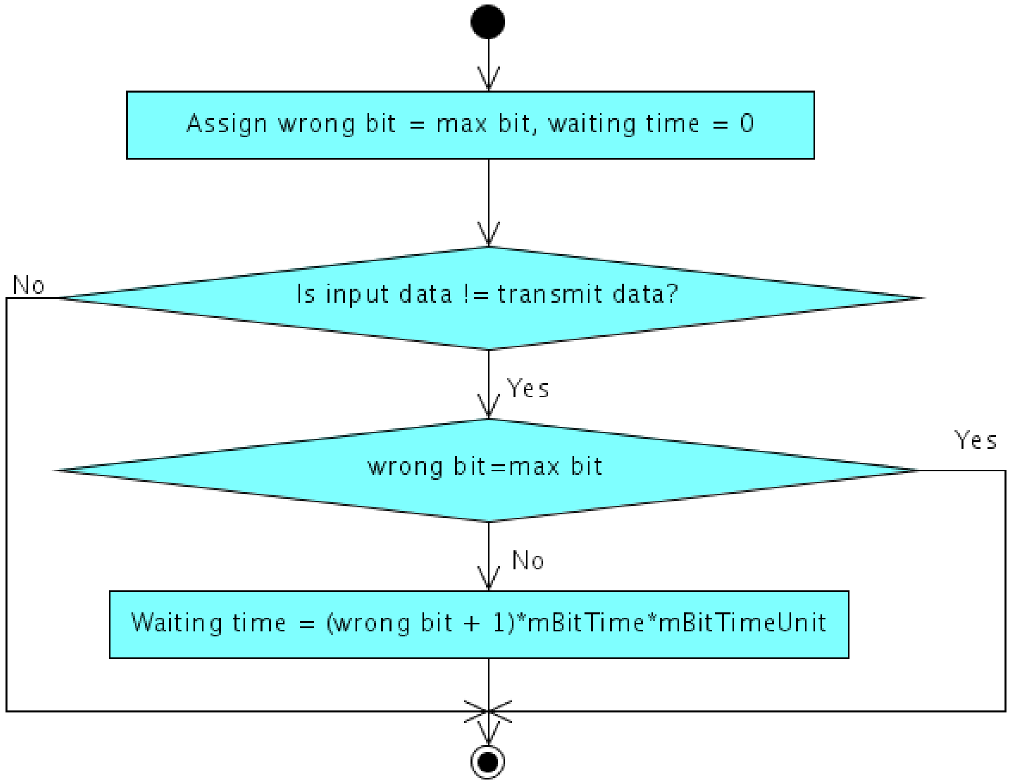
Figure 7.27: Calculate the number of bytes

Explanation:

(1) This function is used to calculate the number of communication bytes based on the value of LDFC.RFDL.

(2) In the LIN mode, number of bytes = 9 when LDFC.RFDL > 8. In the UART mode, number of bytes = 9 when LDFC.RFDL > 8 or LDFC.RFDL = 0. Otherwise, number of bytes = LDFC.RFDL.

7.20. CalcBitBoundary



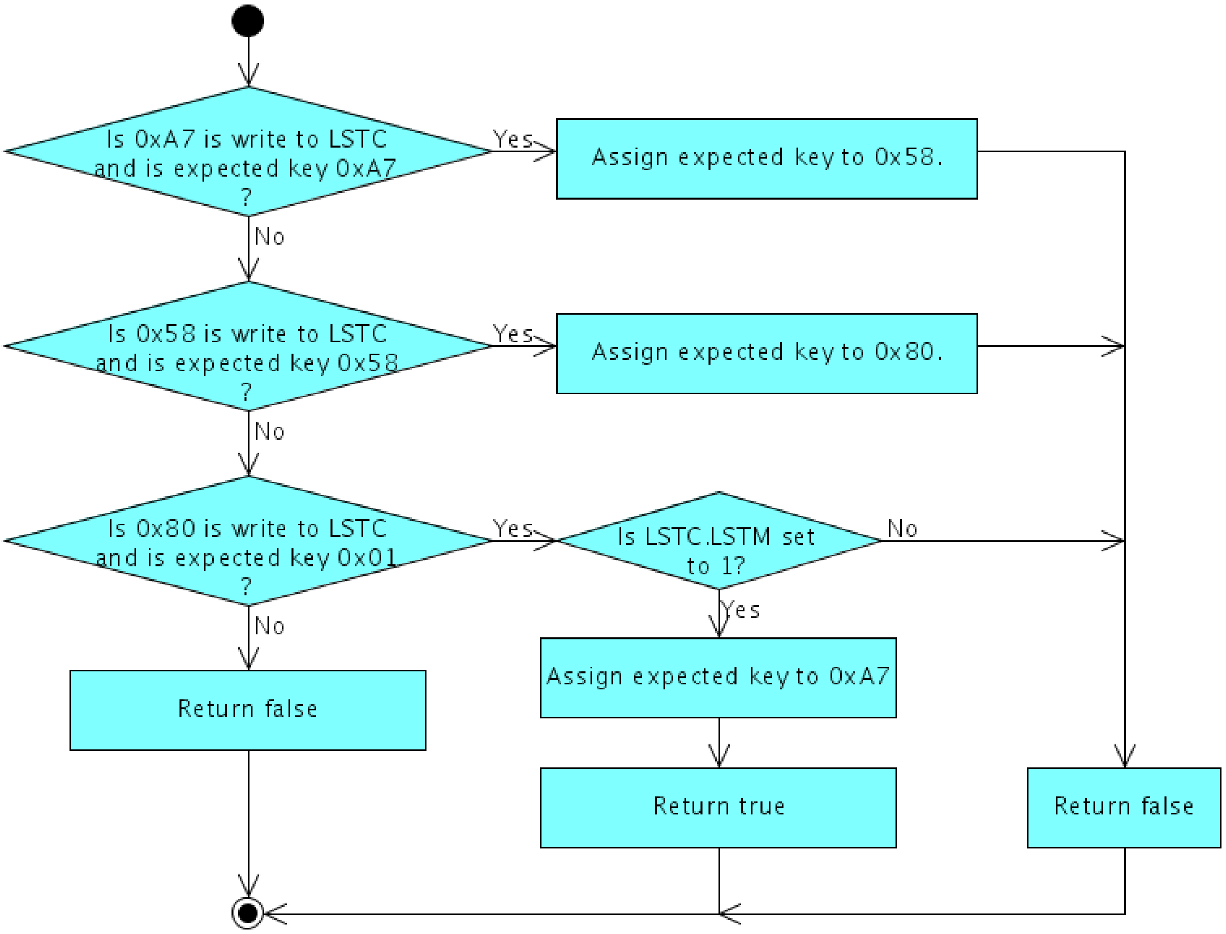
Assign wrong bit = max bit, waiting time = 0
Is input data != transmit data?
No
Yes
wrong bit=max bit
Yes
No
Waiting time = (wrong bit + 1)*mBitTime*mBitTimeUnit

Figure 7.28: Calculate the Bit Boundary flow

Explanation:

(1) This function is used to calculate the delay time before interrupt and flag is asserted in case of Bit Error occurred.

7.21. CheckEnterSelfTest



Is 0xA7 is write to LSTC and is expected key 0xA7 ?
Yes
Assign expected key to 0x58.
No
Is 0x58 is write to LSTC and is expected key 0x58 ?
Yes
Assign expected key to 0x80.
No
Is 0x80 is write to LSTC and is expected key 0x01 ?
Yes
Is LSTC.LSTM set to 1?
No
Yes
Assign expected key to 0xA7
No
Return false
Return true
Return false

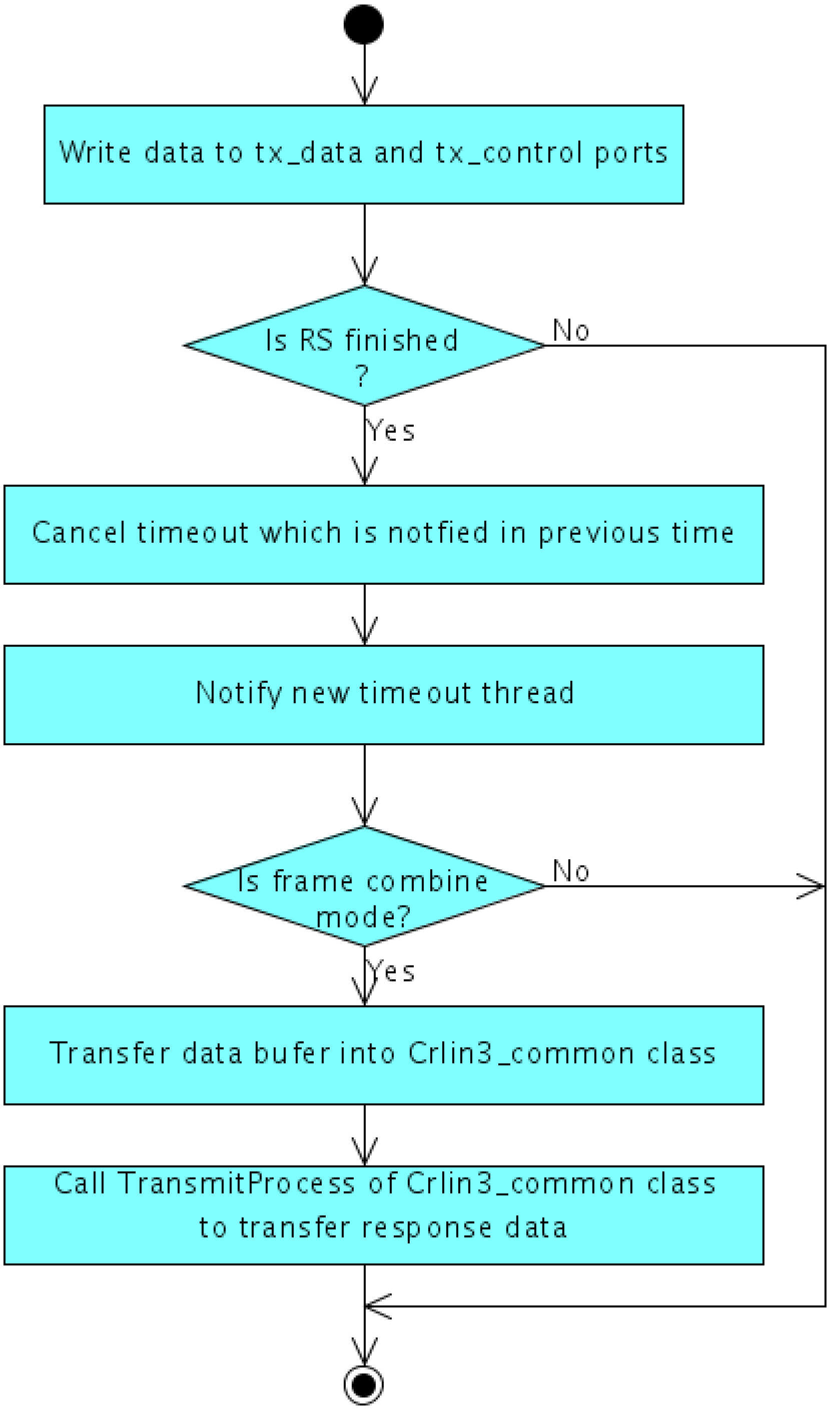
Figure 7.29: Check entering Self Test mode flow

Explanation:

(1) This function is used to check the key sequence which is written by users is correct or not.

(2) If users write a sequence 0xA7, 0x58 and 0x01 into LSTC.LSTME and LSTC.LSTM = 1 the function will return true. Otherwise, the function will return false.

7.22. OutputData process in Master mode



Write data to tx_data and tx_control ports
Is RS finished ?
No
Yes
Cancel timeout which is notfied in previous time
Notify new timeout thread
Is frame combine mode?
No
Yes
Transfer data bufer into Crlin3_common class
Call TransmitProcess of Crlin3_common class to transfer response data

Figure 7.30: Output data in Master mode flow

Explanation:

(1) This function is used to write the data to output ports.

(2) This function will notify Timeout process when the RS pattern is transferred successfully.

(3) In the frame combine mode, the data registers are transferred into RLIN3 common class

and the TransmitProcess of RLIN3 common class is called for Response data transmission.

7.23. UpdateStatus in Master mode

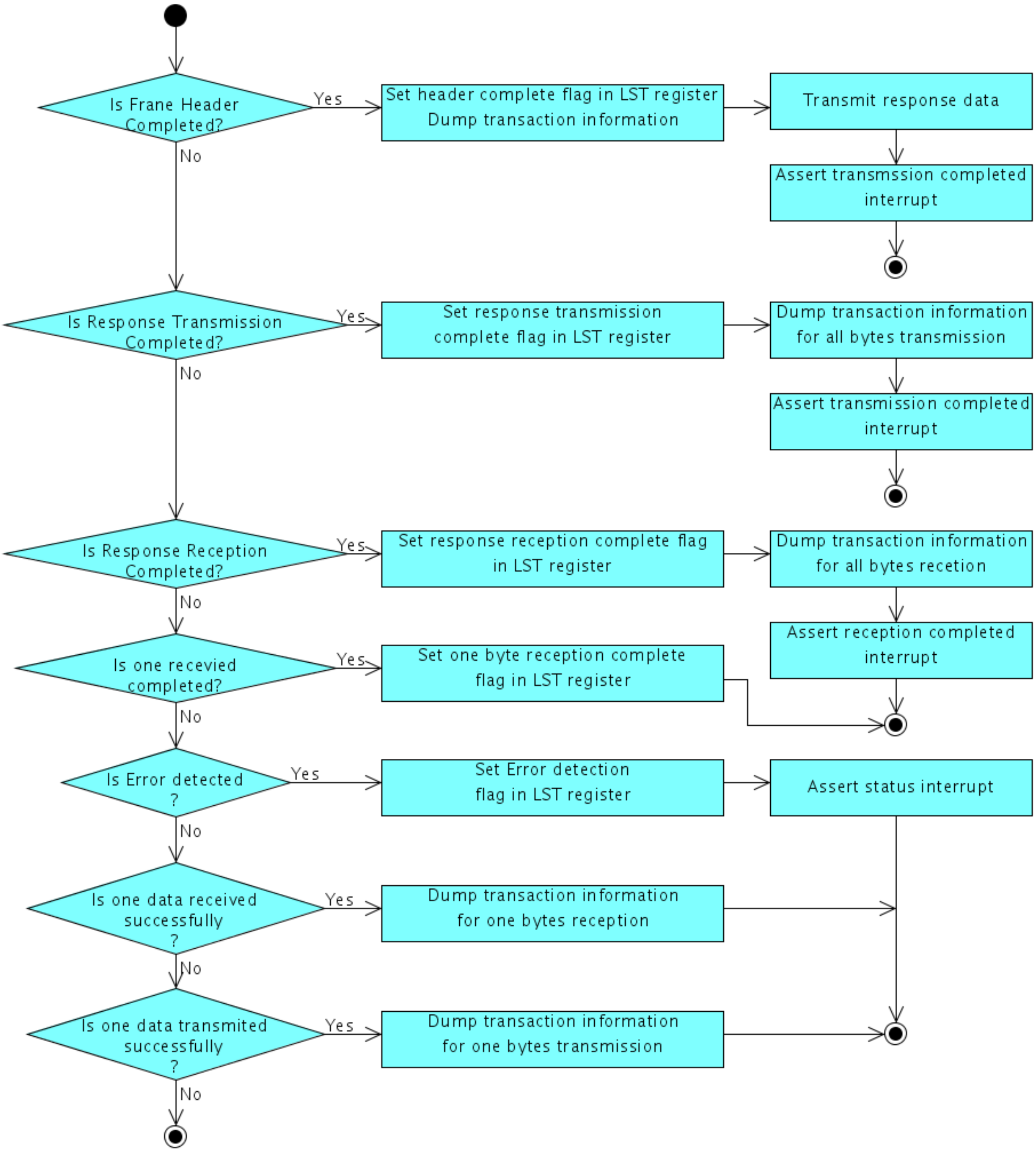

Is Frane Header Completed?
Yes
No
Set header complete flag in LST register
Dump transaction information
Transmit response data
Assert transmssion completed interrupt
Is Response Transmission Completed?
Yes
No
Set response transmission complete flag in LST register
Dump transaction information for all bytes transmission
Assert transmission completed interrupt
Is Response Reception Completed?
Yes
No
Set response reception complete flag in LST register
Dump transaction information for all bytes recetion
Assert reception completed interrupt
Is one recevied completed?
Yes
No
Set one byte reception complete flag in LST register
Is Error detected ?
Yes
No
Set Error detection flag in LST register
Assert status interrupt
Is one data received successfully ?
Yes
No
Dump transaction information for one bytes reception
Is one data transmited successfully ?
Yes
No
Dump transaction information for one bytes transmission

Figure 7.31: Update current status in Master mode flow

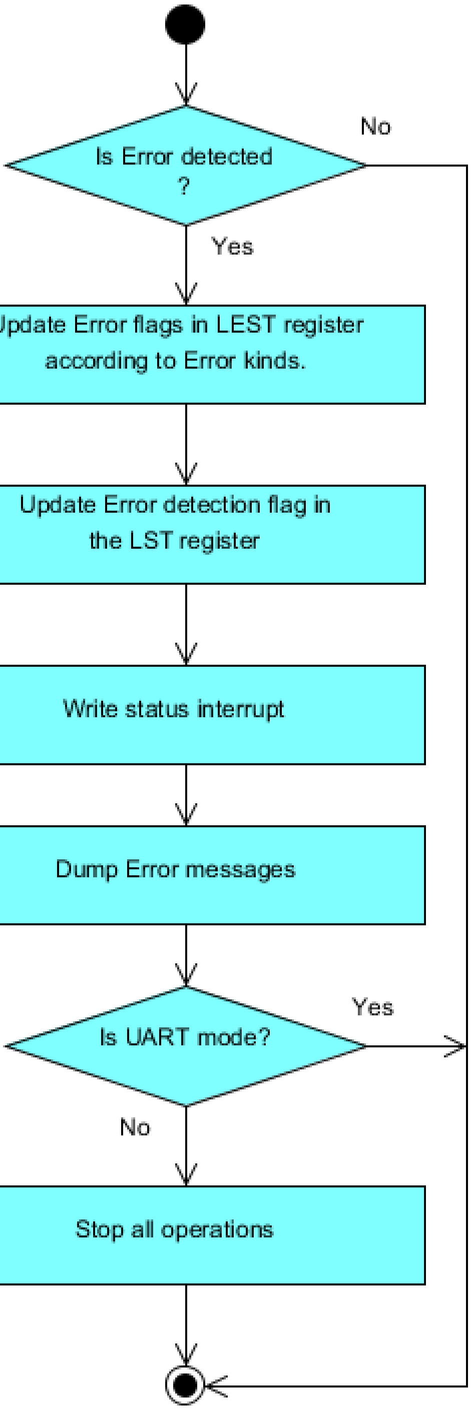
Explanation:

(1) This function is used to update the current status into LST register. The interrupts are asserted and transaction information are dumped at here.

(2) In this function, if frame Header is completed, the Response data are transmitted.

(3) Depended on transaction status, the flags in LST register are updated, interrupts are asserted and transaction information are dumped.

7.24. UpdateErrorStatus in Master/Slave/Uart modes



Is Error detected ?
No
Yes
Update Error flags in LEST register according to Error kinds.
Update Error detection flag in the LST register
Write status interrupt
Dump Error messages
Is UART mode?
Yes
No
Stop all operations

Figure 7.32: Update Error Status in Master/Slave/Uart mode flow

Explanation:

(1) This function used to update Error flags into LEST register, update Error detection flag into LST register and stop all operation when Error is detected.

(2) In Slave mode, when receiving SYNC field error, the RLIN3 stop reception regardless of SYNC field error detection enable bit (LEDE.SFERE). SYNC field error detection enable bit (LEDE.SFERE) only affect to SYNC filed error flag and error interrupt.

(3) In Uart mode, the operation is not stopped even when errors is detected.

7.25. UpdateRegisters in Master mode

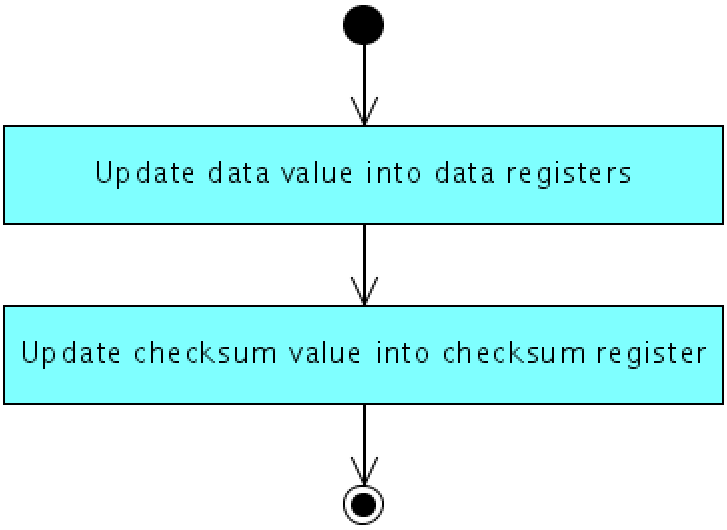
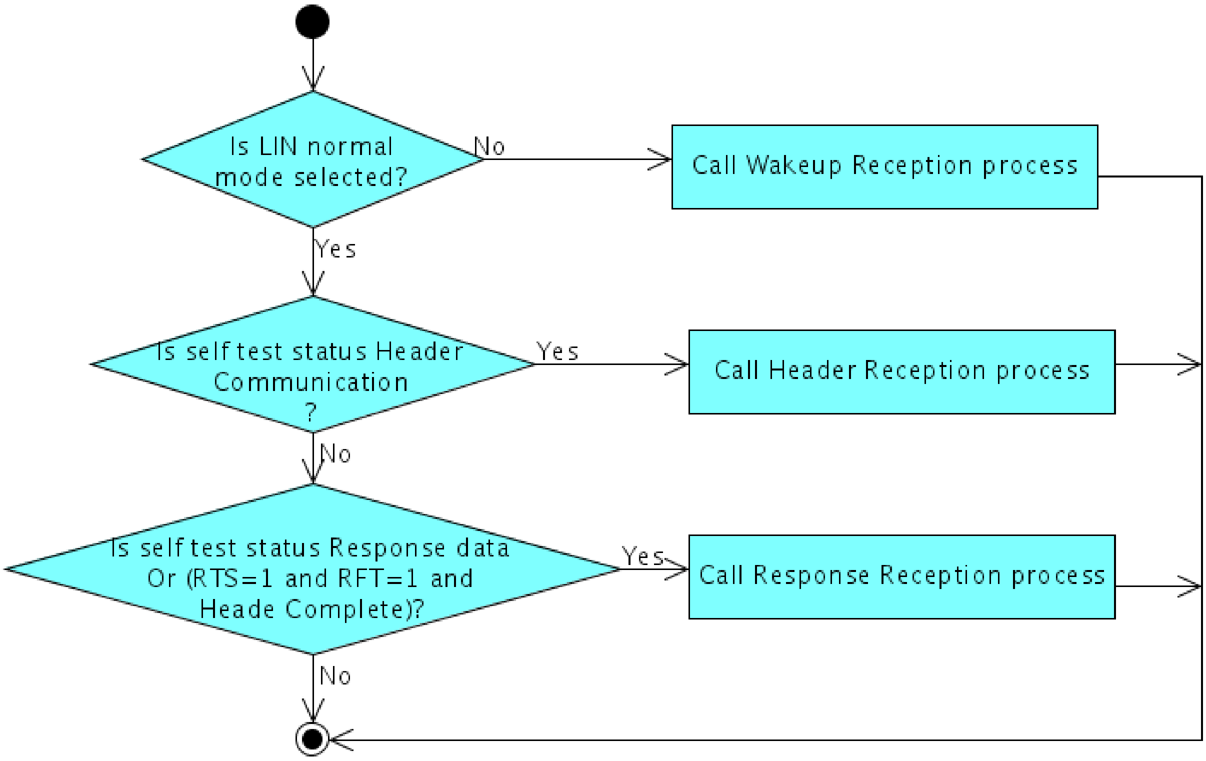

Update data value into data registers
Update checksum value into checksum register

Figure 7.33: Update registers in Master mode flow

Explanation:

(1) This function is used to update data values into data registers of RLIN3 Master class and update checksum value into LCBR register of RLIN3 Master class.

7.26. ReceiveMethod in Master mode



Is LIN normal mode selected?
No
Call Wakeup Reception process
Yes
Is self test status Header Communication ?
Yes
Call Header Reception process
No
Is self test status Response data Or (RTS=1 and RFT=1 and Heade Complete)?
Yes
Call Response Reception process
No

Figure 7.34: Receive Method in Master mode flow

Explanation:

(1) This thread is called when data arrived at RX_CONTROL port.

(2) If LIN normal is not selected, the Wakeup reception process is called. The Header reception process is called if the status of Self Test is Header Communication.

(3) The Response Reception process is called when the status of Self Test is Response Communication, Transmission is selected, RTS = 1 and Header is completed.

7.27. cb_LSTC_LSTM function of RLIN3 Master/Slave classes

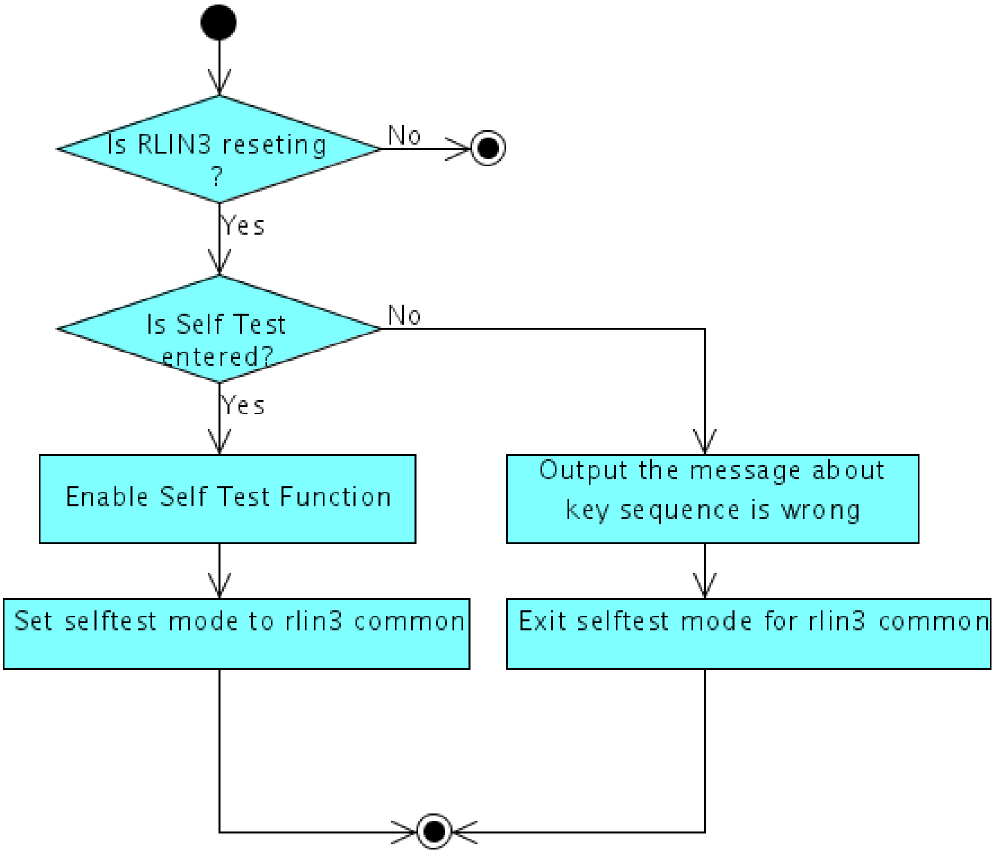

Is RLIN3 reseting ?
No
Yes
Is Self Test entered?
No
Yes
Enable Self Test Function
Output the message about key sequence is wrong
Set selftest mode to rlin3 common
Exit selftest mode for rlin3 common

Figure 7.35: Callback function of LSTC register flow

Explanation:

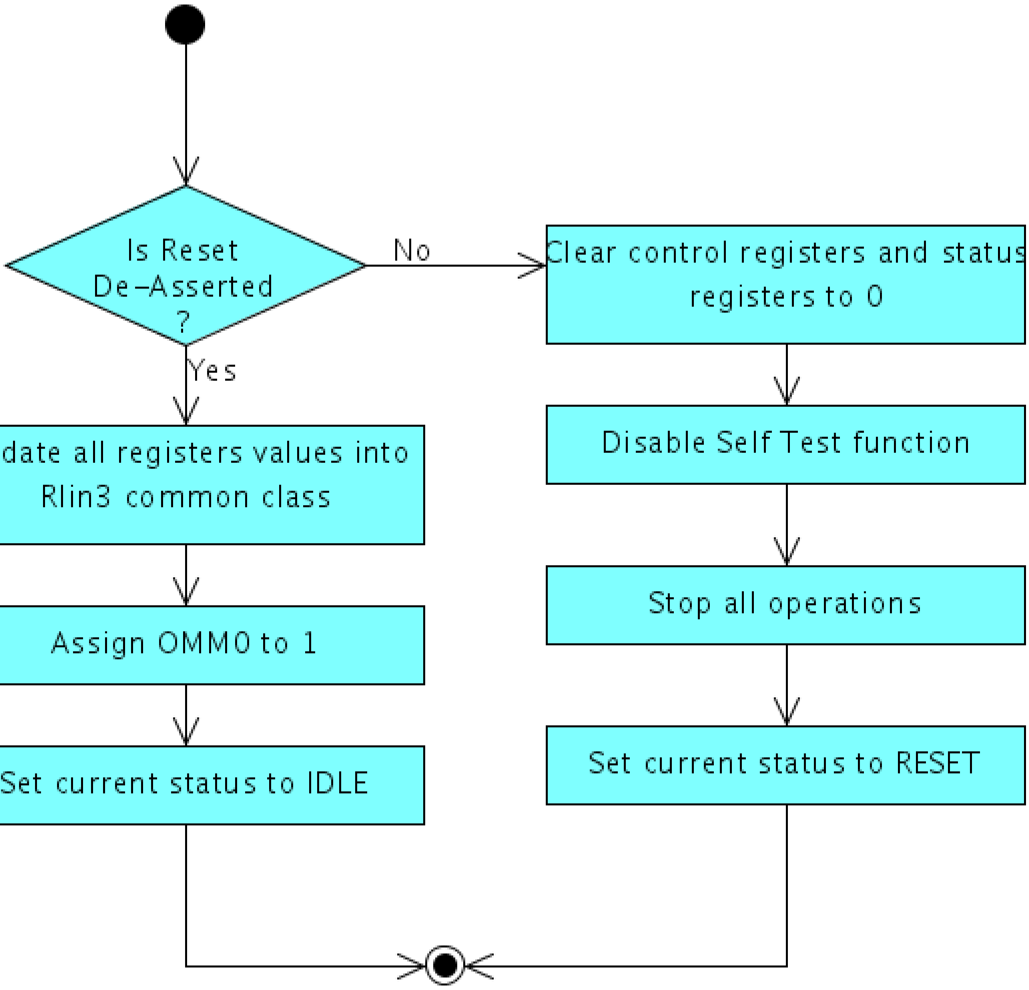
(1) This function is called when users write data to LSTC register.

(2) When RLIN3 is in Reset mode, the function CheckEnterSelfTest is called to check the key sequence for entering Self Test is correct or not. If key sequence is correct, the Self Test function is enabled. Otherwise, the message about wrong key sequence is outputted.

(3) When RLIN3 enter self-test mode successfully, set self-test mode to RLIN3 common class.

When RLIN3 exit from self-test mode, exit self-test mode for RLIN3 common class.

7.28. cb_LCUC_OM1 function of RLIN3 Master/Slave/Uart classes



Is Reset De-Asserted ?
No
Yes
Clear control registers and status registers to 0
Disable Self Test function
Stop all operations
Set current status to RESET
Update all registers values into Rlin3 common class
Assign OMM0 to 1
Set current status to IDLE

Figure 7.36: Callback function of LCUC register flow

Explanation:

(1) This function is called when users write data to LCUC register.

(2) The reset is asserted when LCUC.OM0 = 0, otherwise the reset is de-asserted.

(3) When reset asserted, control registers and status registers are clear to 0. The Self test function is disabled, all operations are stopped and current status is updated to Reset state.

(4) When reset de-asserted, all register values are updated into RLIN3 common class, OMM0 is assigned to 1 and current status is updated to IDLE state.

7.29. cb_LTRC_FTS function of RLIN3 Master class

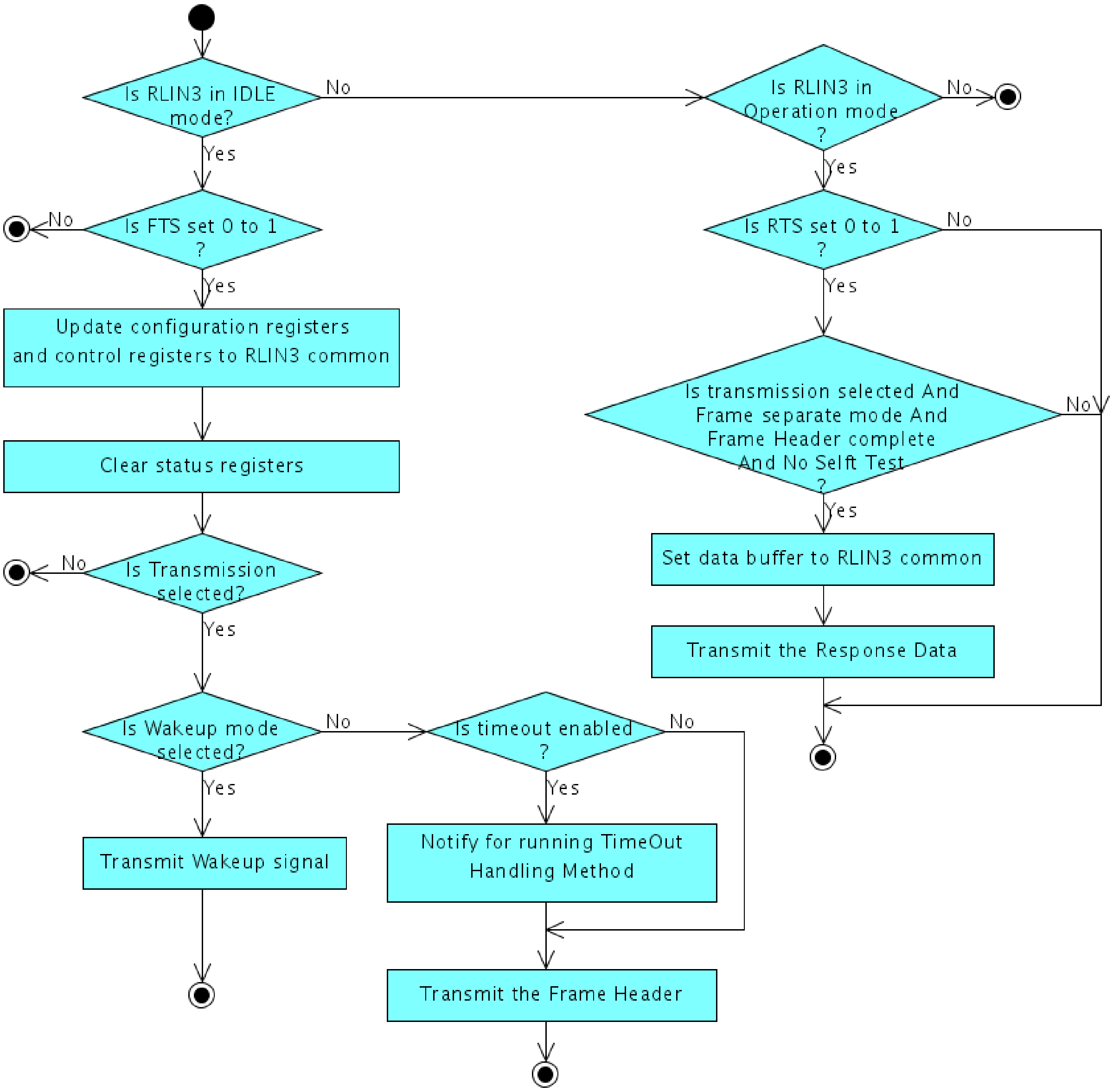

Is RLIN3 in IDLE mode?
No
Yes
Is RLIN3 in Operation mode ?
No
Yes
Is FTS set 0 to 1 ?
No
Yes
Is RTS set 0 to 1 ?
No
Yes
Update configuration registers and control registers to RLIN3 common
Clear status registers
Is transmission selected And Frame separate mode And Frame Header complete And No Selft Test ?
No
Yes
Set data buffer to RLIN3 common
Transmit the Response Data
Is Transmission selected?
No
Yes
Is Wakeup mode selected?
No
Yes
Is timeout enabled ?
No
Yes
Transmit Wakeup signal
Notify for running TimeOut Handling Method
Transmit the Frame Header

Figure 7.37: Callback function of LTRC register flow

Explanation:

(1) This function is called when users write data to LTRC register.

(2) When the current status is IDLE mode, if the LTRC.FTS is change 0 to 1, the values of configuration registers and control registers are updated to RLIN3 common class, the status registers are cleared. When transmission is selected, the Wakeup signal or Frame Header is transmitted depended on LCUC.OM1.

(3) The timeout process is notified before Header transmission if timeout function is enabled.

7.30. OutputData in Slave mode

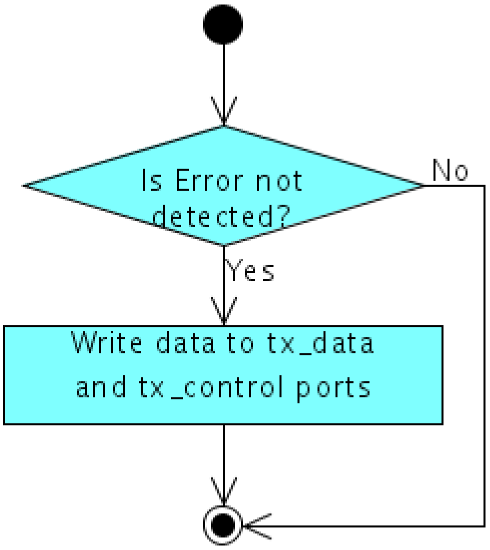


Figure 7.38: Output data in Slave mode flow

Explanation:

(1) This function is called when data need to write output ports.

(2) When Error is not detected, the data are written to output ports.

7.31. ReceiveMethod in Slave mode

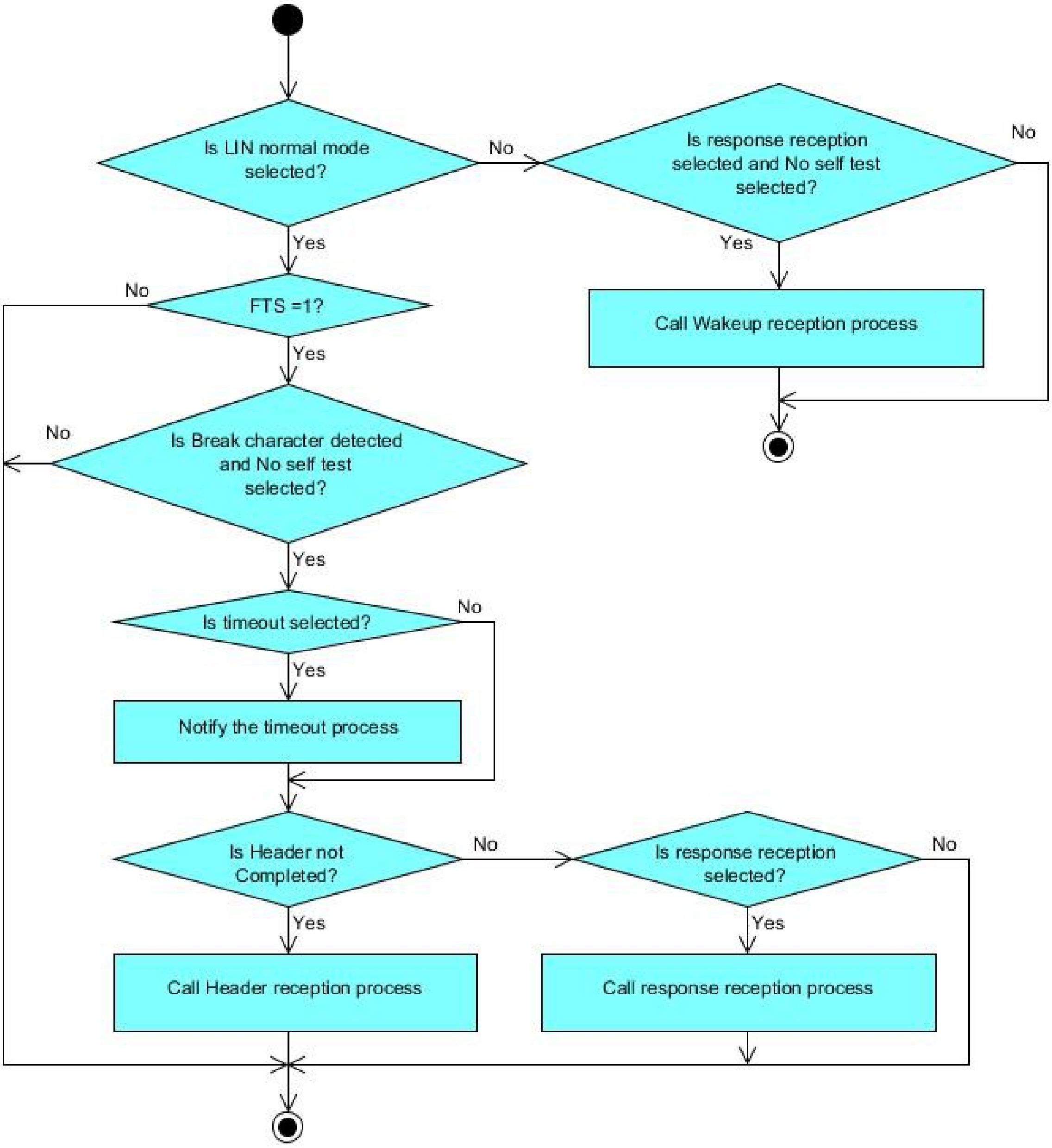

Is LIN normal mode selected?
No
Is response reception selected and No self test selected?
No
Yes
Call Wakeup reception process
Yes
No
FTS =1?
Yes
No
Is Break character detected and No self test selected?
Yes
Is timeout selected?
No
Yes
Notify the timeout process
Is Header not Completed?
No
Is response reception selected?
No
Yes
Yes
Call Header reception process
Call response reception process

Figure 7.39: Receive method in Slave mode flow

Explanation:

(1) This function is called when data are arrived at input ports.

(2) In this function, if Wakeup mode is selected and No Self Test, Wakeup reception process is called.

(3) In the LIN normal mode, if LTRC.FTS = 0, the reception is stopped.

(4) After receiving break character, if timeout is enabled, RLIN3 notifies the timeout event to check the timeout process after a period of time out. The Header Reception process is called when Header Complete flag = 0. The Response Reception process is called when Header Complete flag = 1 and Response reception is selected.

7.32. UpdateStatus in Slave mode

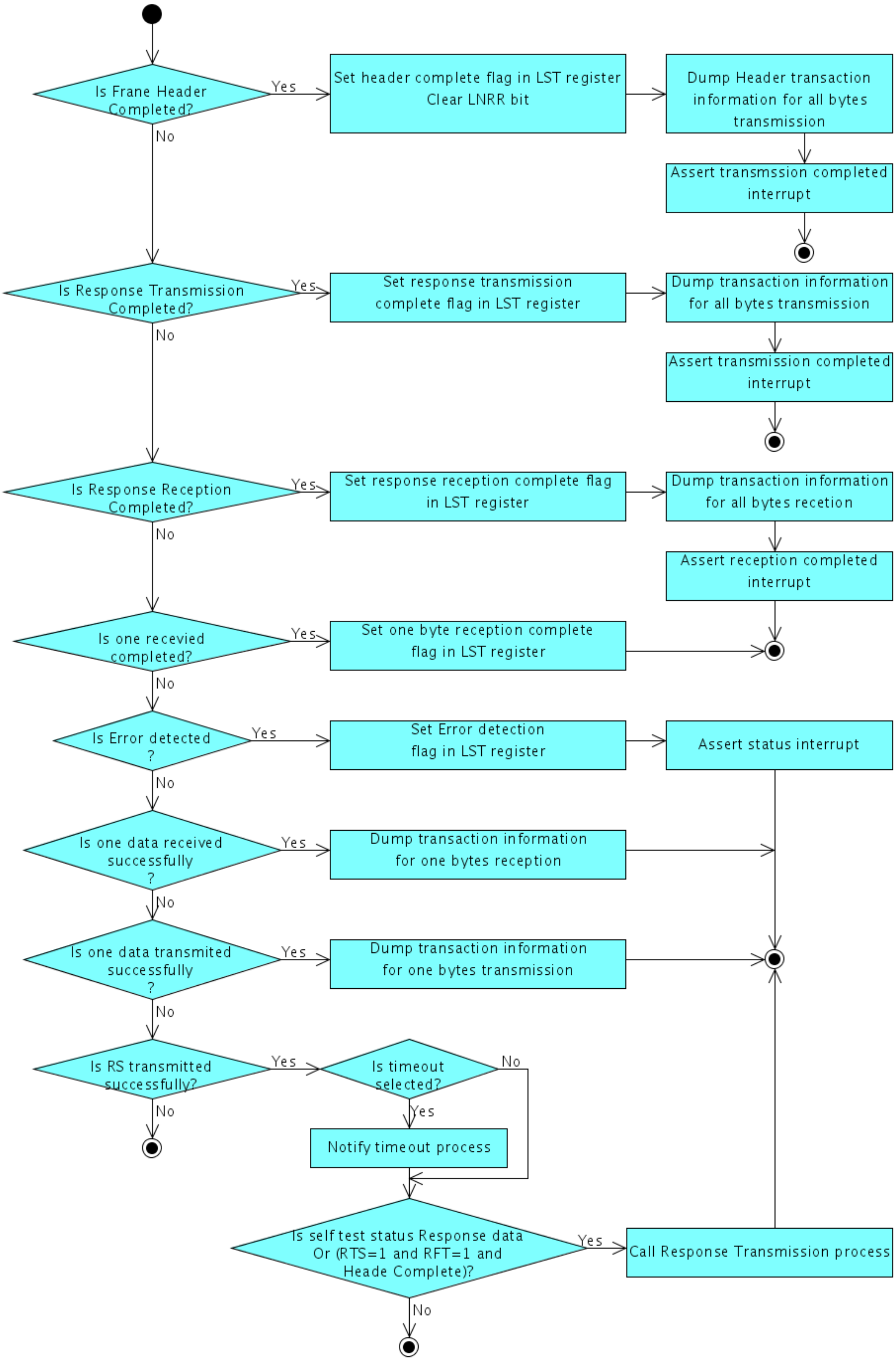

Is Frane Header Completed?
Yes
No
Set header complete flag in LST register
Clear LNRR bit
Dump Header transaction information for all bytes transmission
Assert transmssion completed interrupt
Is Response Transmission Completed?
Yes
No
Set response transmission complete flag in LST register
Dump transaction information for all bytes transmission
Assert transmission completed interrupt
Is Response Reception Completed?
Yes
No
Set response reception complete flag in LST register
Dump transaction information for all bytes recetion
Assert reception completed interrupt
Is one recevied completed?
Yes
No
Set one byte reception complete flag in LST register
Is Error detected ?
Yes
No
Set Error detection flag in LST register
Assert status interrupt
Is one data received successfully ?
Yes
No
Dump transaction information for one bytes reception
Is one data transmited successfully ?
Yes
No
Dump transaction information for one bytes transmission
Is RS transmitted successfully?
Yes
No
Is timeout selected?
No
Yes
Notify timeout process
Is self test status Response data Or (RTS=1 and RFT=1 and Heade Complete)?
Yes
No
Call Response Transmission process

Figure 7.40: Update current status in Slave mode flow

Explanation:

(1) This function is called when the current status need to update in status register LST.

(2) Depended on the current status, the status flags LST register are updated, the interrupts are asserted and transaction information are also dumped.

(3) If Response Space is transmitted successfully, the timeout process is called when timeout function is enabled. The response transmission process is called when Self Test Response is selected or in case RTS = 1 and RFT = 1 and Header complete.

7.33. UpdateRegisters in Slave mode

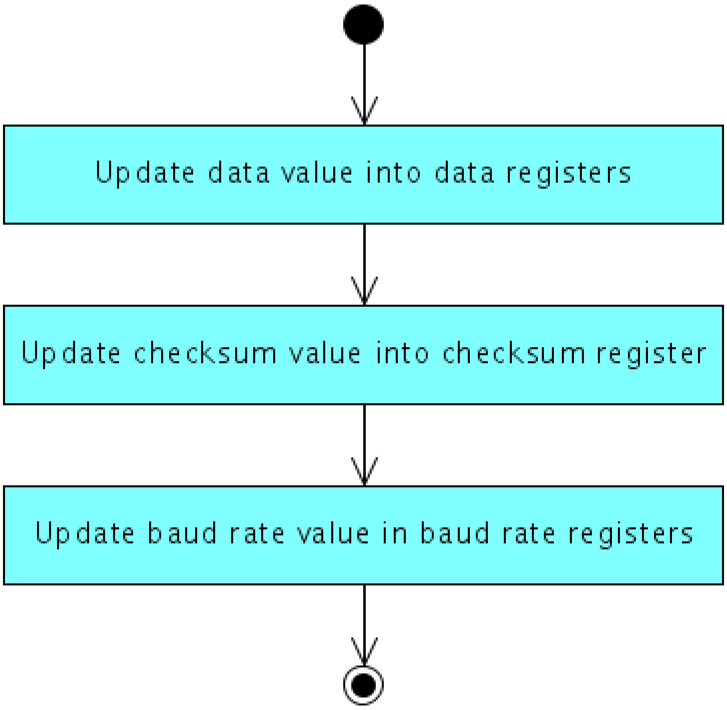

Update data value into data registers
Update checksum value into checksum register
Update baud rate value in baud rate registers

Figure 7.41: Update registers value in Slave mode flow

Explanation:

(1) This function is used to update data values into data registers of RLIN3 Slave class and update checksum value into LCBR register of RLIN3 Slave class.

(2) The baud rate values are also updated to LBRP0 and LBRP1 registers in Slave mode with auto baud rate.

7.34. cb_LTRC_FTS in RLIN3 Slave class

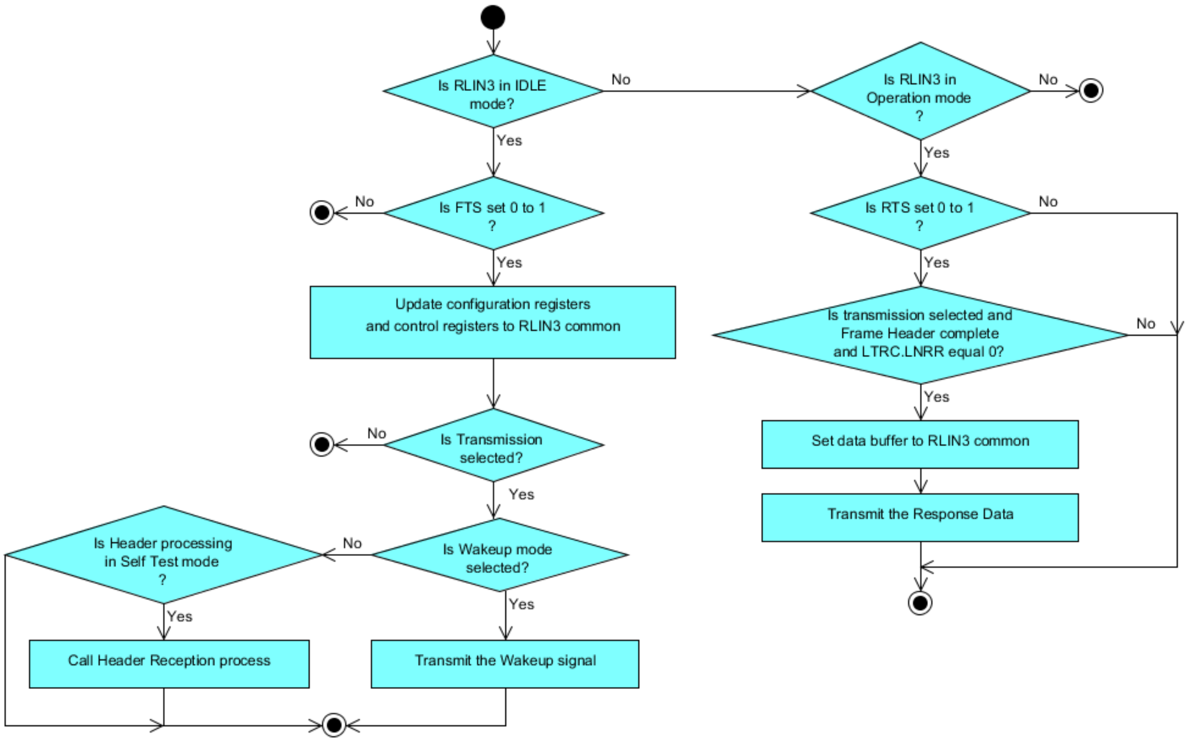

Is RLIN3 in IDLE mode?
No
Yes
Is RLIN3 in Operation mode ?
No
Yes
Is FTS set 0 to 1 ?
No
Yes
Is RTS set 0 to 1 ?
No
Yes
Update configuration registers and control registers to RLIN3 common
Is transmission selected and Frame Header complete and LTRC.LNRR equal 0?
No
Yes
Is Transmission selected?
No
Yes
Set data buffer to RLIN3 common
Transmit the Response Data
Is Header processing in Self Test mode ?
No
Is Wakeup mode selected?
Yes
Yes
Call Header Reception process
Transmit the Wakeup signal

Figure 7.42: Callback function of LTRC register in Slave mode flow

Explanation:

(1) This function is called when users write data to LTRC register in Slave mode.

(2) When the current status is IDLE mode, if the LTRC.FTS is change 0 to 1, the values of configuration registers and control registers are updated to RLIN3 common class, the status registers are cleared. When transmission is selected, the Wakeup signal is transmitted depended on LCUC.OM1. In the LIN normal mode, if Self Test status is Header processing, the Header Reception process is called.

(3) In the Operation mode, when RTS is set to 1, transmission is selected and Frame Header is completed, and LTRC.LNRR is equal to 0, the Response data transmission process is called.

7.35. AddParity

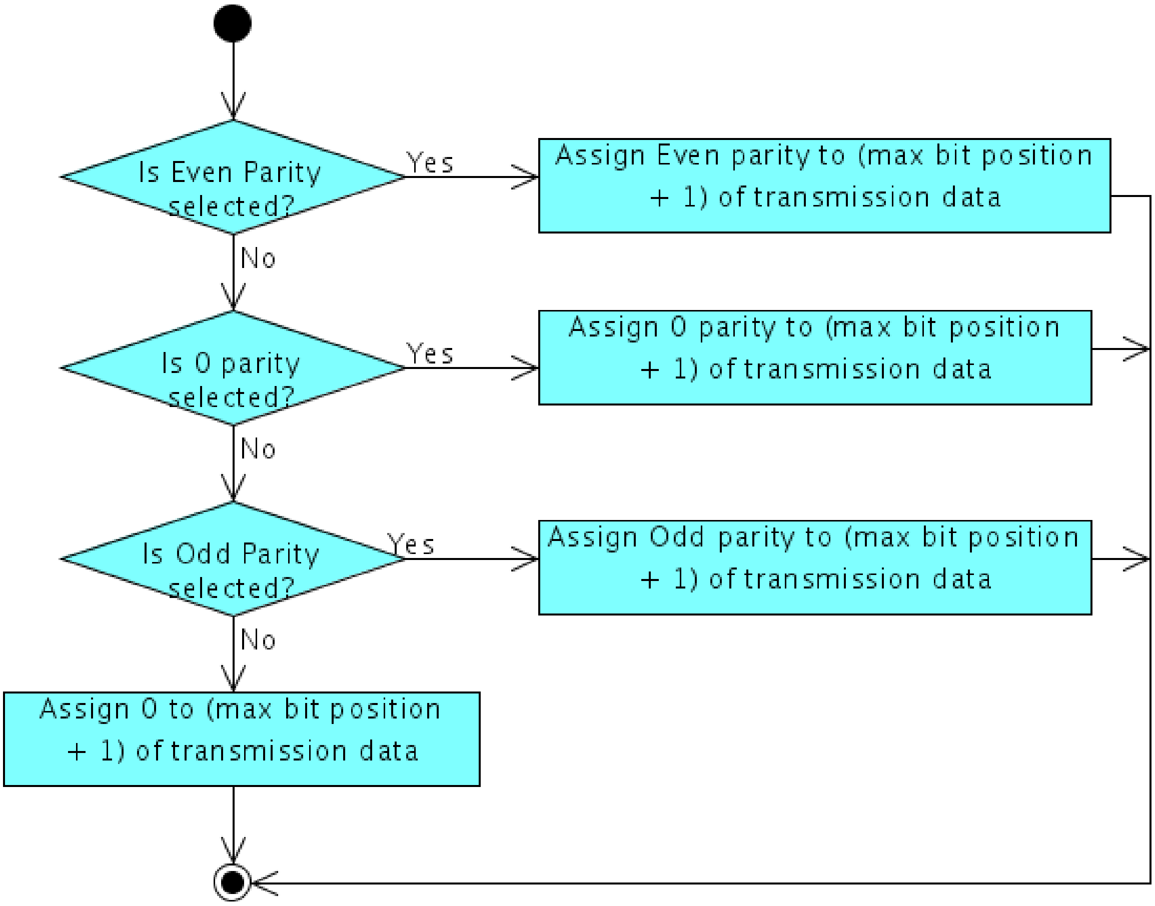

Is Even Parity selected?
Yes
Assign Even parity to (max bit position + 1) of transmission data
No
Is 0 parity selected?
Yes
Assign 0 parity to (max bit position + 1) of transmission data
No
Is Odd Parity selected?
Yes
Assign Odd parity to (max bit position + 1) of transmission data
No
Assign 0 to (max bit position + 1) of transmission data

Figure 7.43: Add parity process flow

Explanation:

(1) This function is called when transferred data are needed to add parity bit.

(2) There are three kind of parity bit that is even parity bit, 0 parity bit and odd parity bit. Depended kind of parity bit, the parity bit is added to max bit position of data + 1 before transferring.

7.36. ReceptionMethod in UART mode

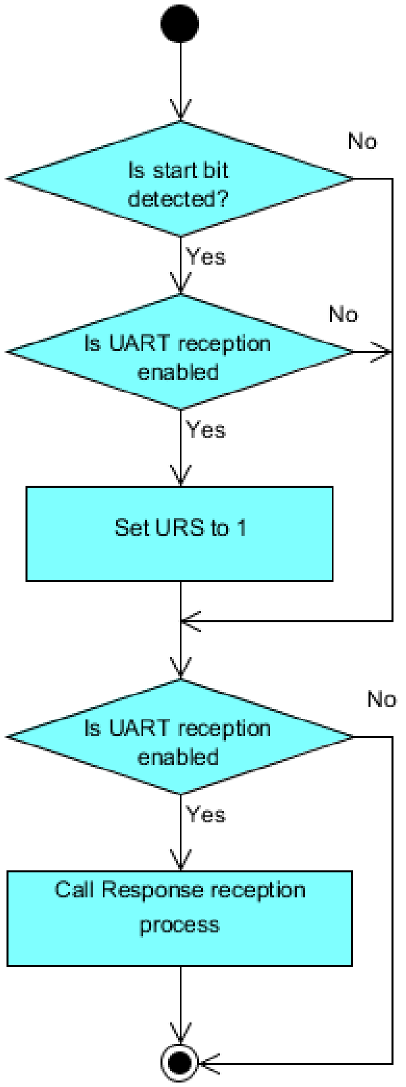

Is start bit detected?
No
Yes
Is UART reception enabled
No
Yes
Set URS to 1
Is UART reception enabled
No
Yes
Call Response reception process

Figure 7.44: Reception Method in UART mode flow

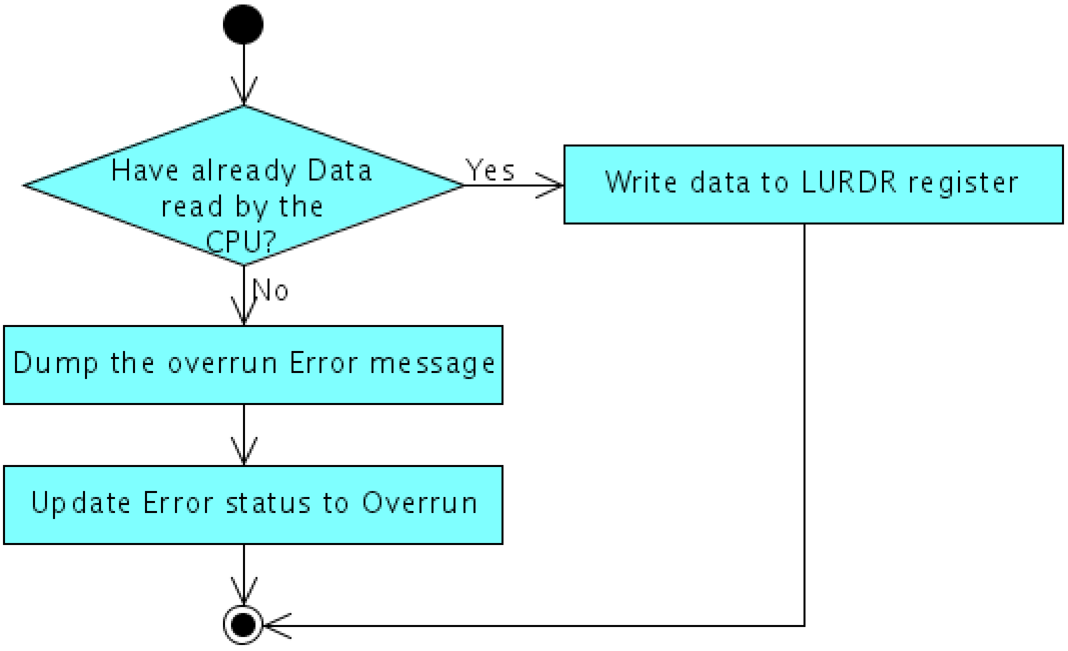
Explanation:

(1) This function is called when data are arrived at input ports.

(2) When start bit is detected, the URS is set to 1 if UART reception is enabled. In case any error occurred during reception process, the reception still continue regardless of error.

(3) The Response reception process is called when UART reception is enabled (UROE = 1).

7.37. UpdateRegisters in UART mode



Have already Data read by the CPU?
Yes
Write data to LURDR register
No
Dump the overrun Error message
Update Error status to Overrun

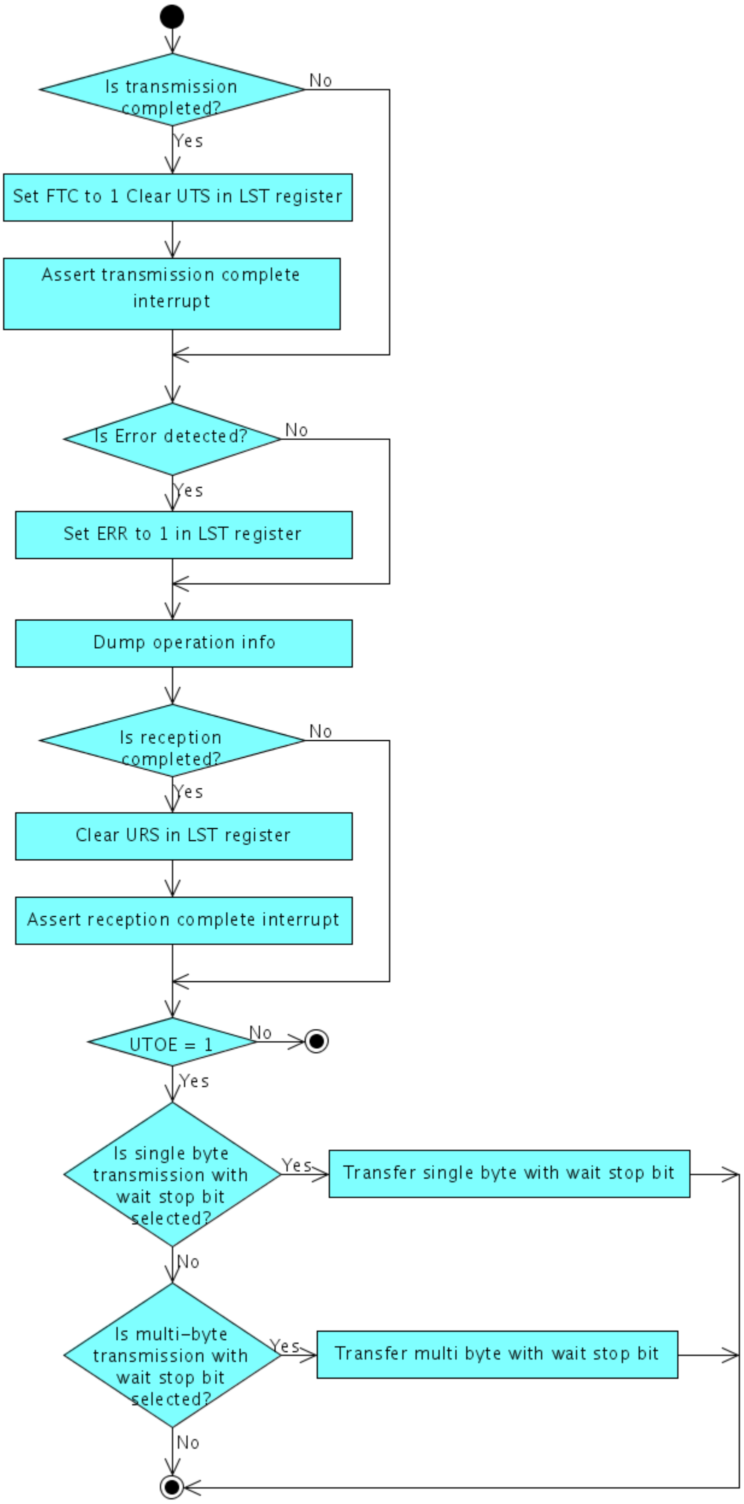
Figure 7.45: Update registers process in UART mode flow

Explanation:

(1) This function is called when data value need to update to LURDR register.

(2) If the data in LURDR is not read by the CPU, the Error status is updated to Overrun Error. Otherwise, the new data is updated to LURDR register.

7.38. UpdateStatus in UART mode



Is transmission completed?
No
Yes
Set FTC to 1 Clear UTS in LST register
Assert transmission complete interrupt
Is Error detected?
No
Yes
Set ERR to 1 in LST register
Dump operation info
Is reception completed?
No
Yes
Clear URS in LST register
Assert reception complete interrupt
UTOE = 1
No
Yes
Is single byte transmission with wait stop bit selected?
Yes
Transfer single byte with wait stop bit
No
Is multi-byte transmission with wait stop bit selected?
Yes
Transfer multi byte with wait stop bit
No

Figure 7.46: Update current status process in UART mode flow

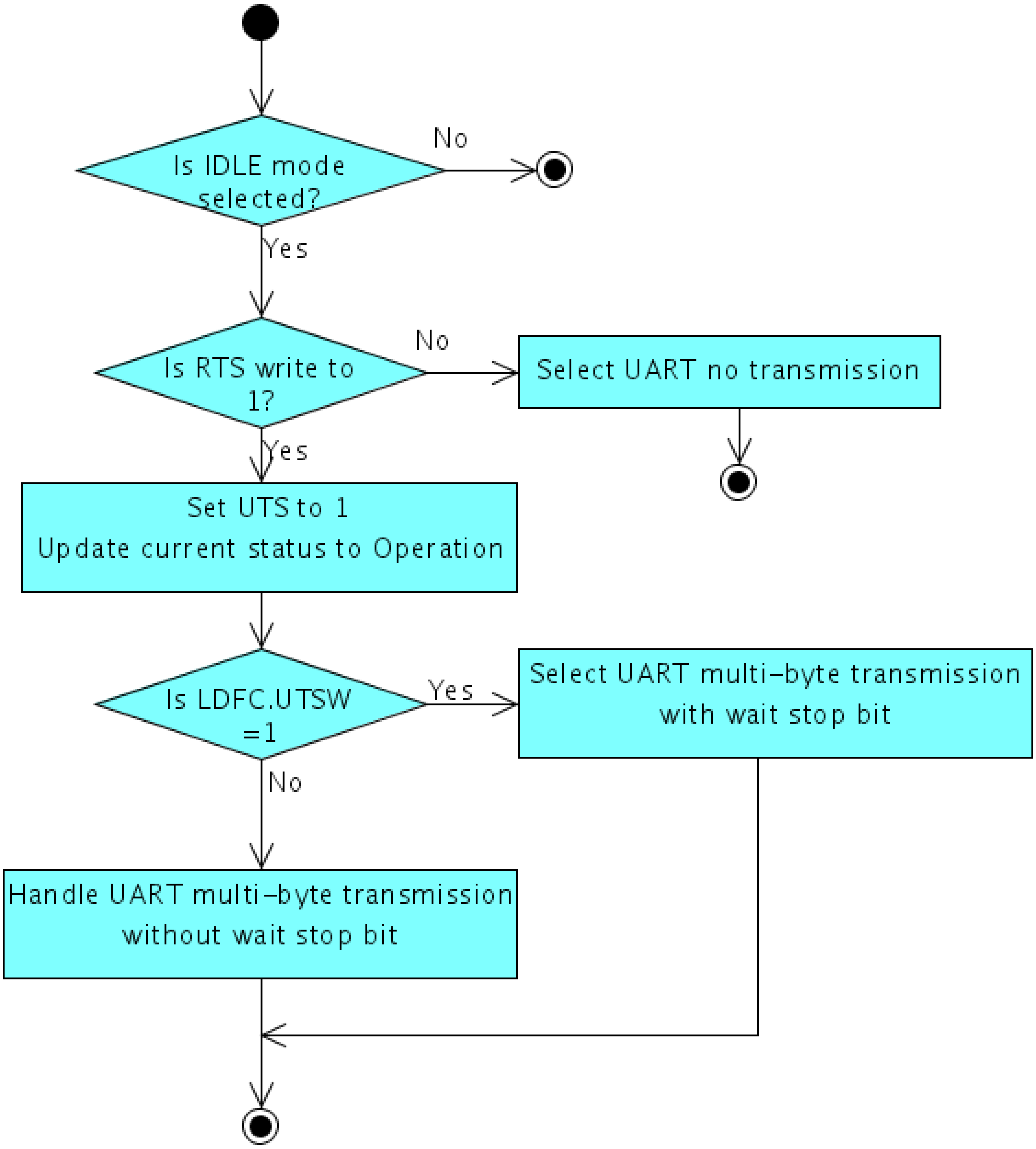
Explanation:

(1) This function is used to update the current status into LST register. The transaction information are dumped at here.

(2) When transmission is completed the UTS is cleared and transmission complete interrupt is asserted. When reception is completed the URS is cleared and reception complete interrupt is asserted.

(3) Depended on the transmission status, the single byte transmission with wait stop bit, multi-byte transmission with wait stop bit is executed when transmission is enabled (UTOE = 1).

7.39. cb_LTRC_RTS in UART mode



Is IDLE mode selected?
No
Yes
Is RTS write to 1?
No
Select UART no transmission
Yes
Set UTS to 1
Update current status to Operation
Is LDFC.UTSW =1
Yes
Select UART multi-byte transmission with wait stop bit
No
Handle UART multi-byte transmission without wait stop bit

Figure 7.47: Callback function of LTRC register in UART mode

Explanation:

(1) This function is called when users write data to LTRC register in UART mode.

(2) In the IDLE mode, when RTS is written to 1, the UTS is set to 1 and the current status is updated to Operation. If the transmission with wait stop bit is selected (UTSW = 1), the transmission status is updated to multi-byte transmission with wait stop bit. Otherwise, the transmission status is updated to multi-byte transmission without wait stop bit and multi-byte transmission without wait stop bit process is proceed.

7.40. cb_LUOER_UTOE in UART mode

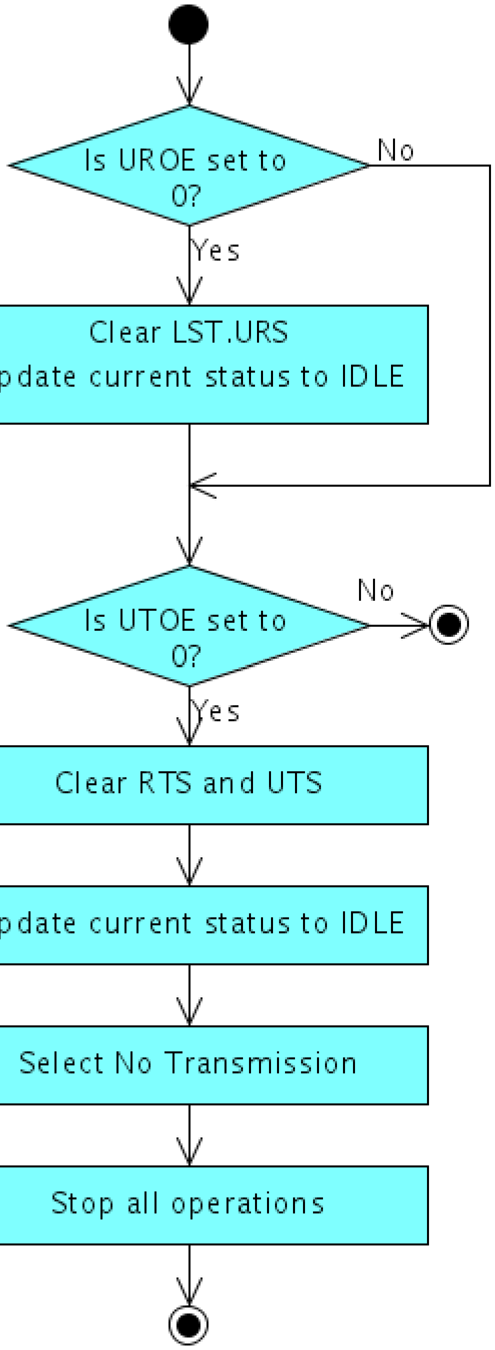


Figure 7.48: Callback function of LUOER register in UART mode flow

Explanation:

(1) This function is called when users write data to LUOER register in UART mode.

(2) When UROE = 1, the current status is updated to ILDE and URS is cleared to 0.

(3) When UTOE = 1, the current status is updated to ILDE and UTS is cleared to 0. All

operations are stopped.

7.41. cb_LUTDR_UTD in UART mode

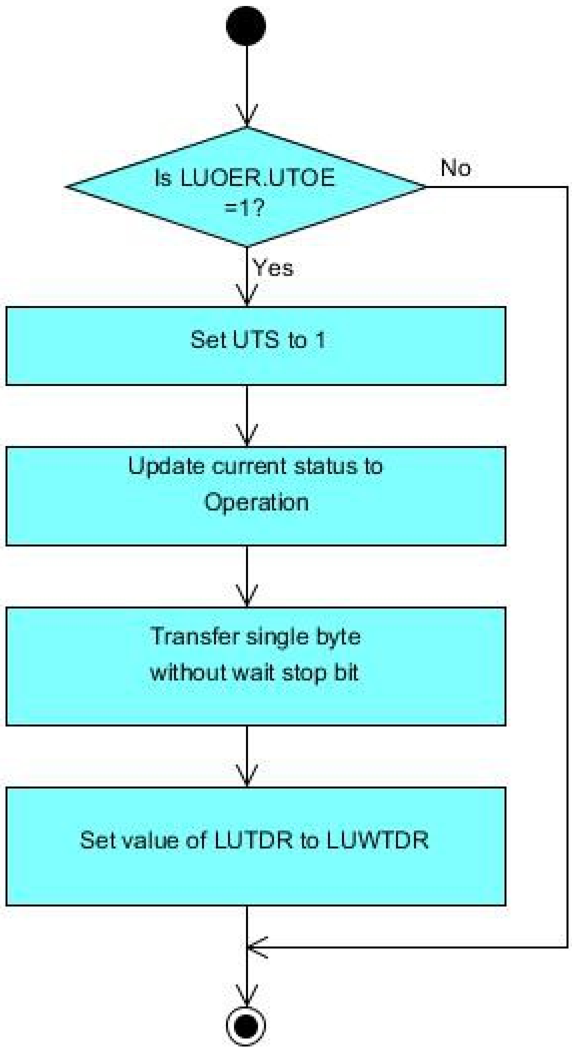

Is LUOER.UTOE =1?
No
Yes
Set UTS to 1
Update current status to Operation
Transfer single byte without wait stop bit
Set value of LUTDR to LUWTDR

Figure 7.49: Callback function of LUTDR register in UART mode flow

Explanation:

(1) This function is called when users write data to LUTDR register in UART mode.

(2) When UTOE = 1, the UTS is set to 1, the current status is updated to Operation state and the transmission status is updated to Single-byte transmission without wait stop bit and Single-byte transmission without wait stop bit process is proceed.

(3) Update the value of LUTDR into LUWTDR

7.42. cb_LUWTDR_UWTD in UART mode

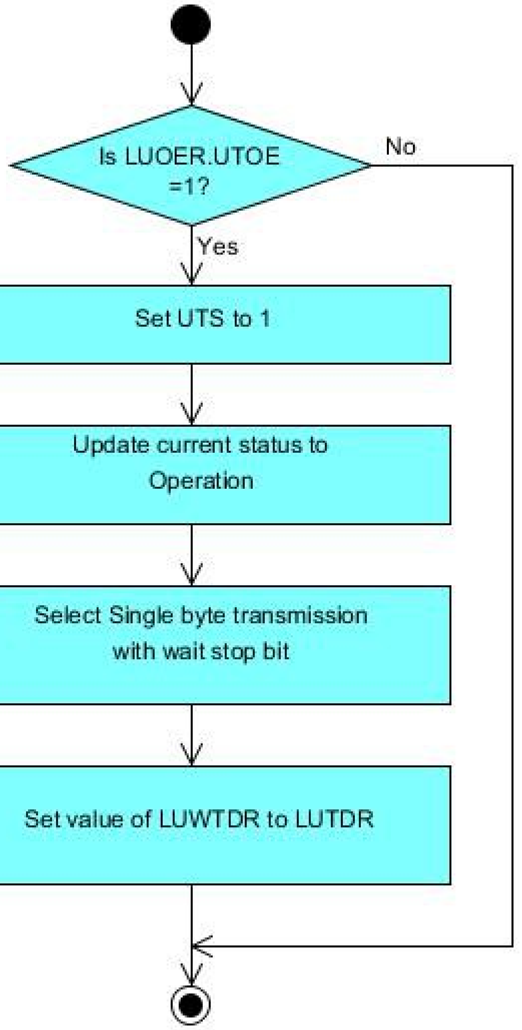

Is LUOER.UTOE =1?
No
Yes
Set UTS to 1
Update current status to Operation
Select Single byte transmission with wait stop bit
Set value of LUWTDR to LUTDR

Figure 7.50: Callback function of LUWTDR register in UART mode flow

Explanation:

(1) This function is called when users write data to LUWTDR register in UART mode.

(2) When UTOE = 1, the UTS is set to 1, the current status is updated to Operation state and the transmission status is updated to Single-byte transmission with wait stop bit.

(3) Update the value of LUWTDR into LUTDR

7.43. Self Test process

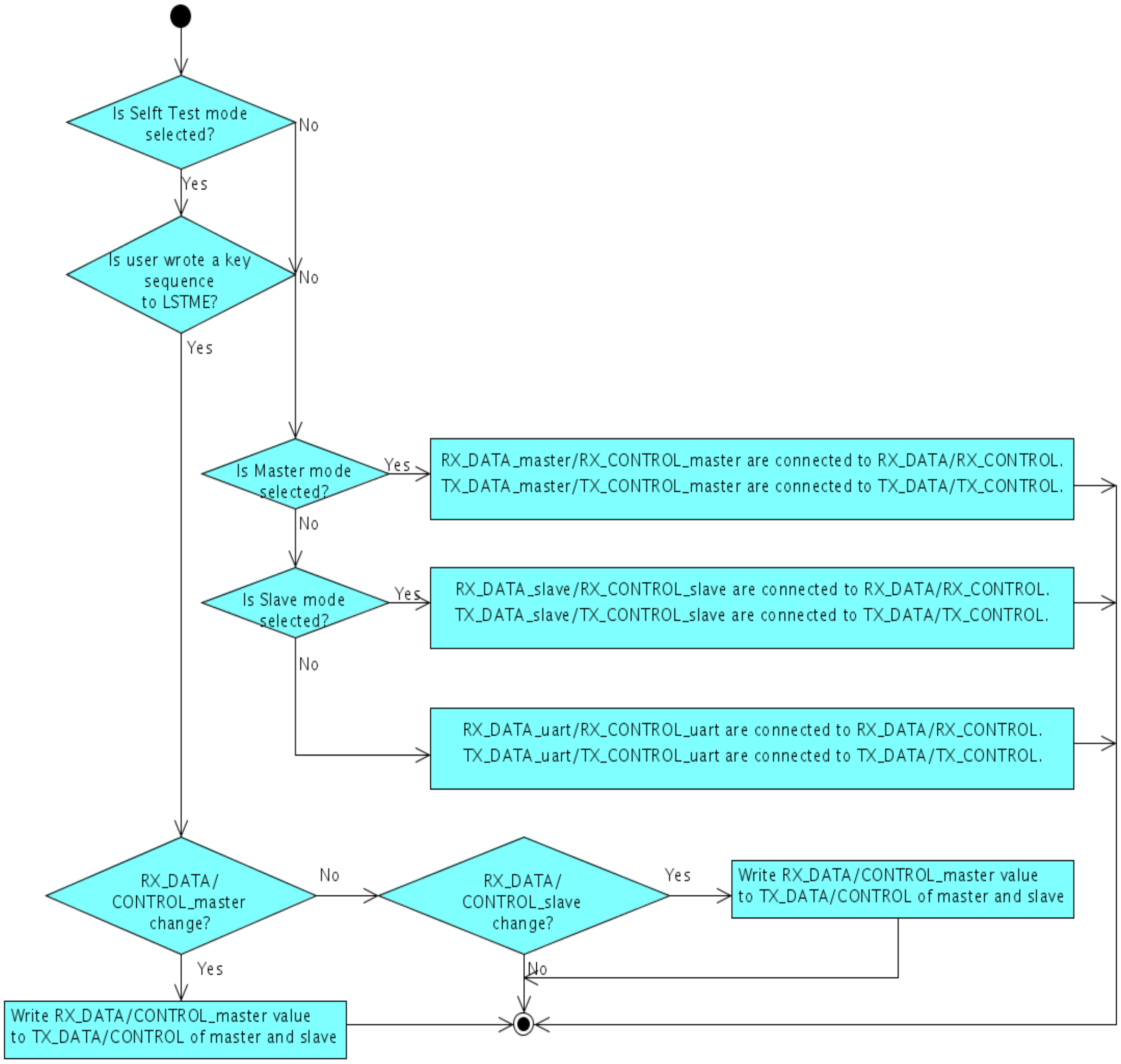

Is Selft Test mode selected?
No
Yes
Is user wrote a key sequence to LSTME?
No
Yes
Is Master mode selected?
Yes
RX_DATA_master/RX_CONTROL_master are connected to RX_DATA/RX_CONTROL.
TX_DATA_master/TX_CONTROL_master are connected to TX_DATA/TX_CONTROL.
No
Is Slave mode selected?
Yes
RX_DATA_slave/RX_CONTROL_slave are connected to RX_DATA/RX_CONTROL.
TX_DATA_slave/TX_CONTROL_slave are connected to TX_DATA/TX_CONTROL.
No
RX_DATA_uart/RX_CONTROL_uart are connected to RX_DATA/RX_CONTROL.
TX_DATA_uart/TX_CONTROL_uart are connected to TX_DATA/TX_CONTROL.
RX_DATA/ CONTROL_master change?
No
RX_DATA/ CONTROL_slave change?
Yes
Write RX_DATA/CONTROL_master value to TX_DATA/CONTROL of master and slave
Yes
No
Write RX_DATA/CONTROL_master value to TX_DATA/CONTROL of master and slave

Figure 7.51: Self Test process flow

Explanation:

(1) The operations in Self mode if Self Test mode is selected and users already write key sequence 0xA7, 0x58 and 0x01 to LSTME bits of LSTC register.

(2) Main operations of Self Test class is handled by this function. Refer to the figure 3.1 for the detail of connection among Self Test module and Master/Slave/Uart modules. The ports RX_DATA_master/RX_CONTROL_master are connected to RX_DATA/RX_CONTROL ports of the Master module. The ports TX_DATA_master/TX_CONTROL_master are connected to TX_DATA/TX_CONTROL ports of the Master module. The ports RX_DATA_slave/RX_CONTROL_slave are connected to RX_DATA/RX_CONTROL ports of the Slave module. The ports TX_DATA_slave/TX_CONTROL_slave are connected to TX_DATA/TX_CONTROL ports of the Slave module.

(3) In Self Test mode, LSTC.LSTM = 1, RX_DATA_master/RX_CONTROL_master ports of

Self Test class and RX_DATA_slave/RX_CONTROL_slave ports of Self Test class will be checked whenever they change. If RX_DATA_master/RX_CONTROL_master ports change, value of them will be written to TX_DATA_master/TX_CONTROL_master and TX_DATA_slave/TX_CONTROL_slave. If RX_DATA_slave/RX_CONTROL_slave ports change, value of them will be written to TX_DATA_master/TX_CONTROL_master and TX_DATA_slave/TX_CONTROL_slave.

(4) In Normal mode, LSTC.LSTM = 0, Master mode is selected, RX_DATA_master/RX_CONTROL_master ports of Self Test class are connected to RX_DATA/RX_CONTROL ports of Self Test class. TX_DATA_master/TX_CONTROL_master ports of Self Test class are connected to TX_DATA/TX_CONTROL ports of Self Test class.

(5) In Normal mode, LSTC.LSTM = 0, Slave mode is selected, RX_DATA_slave/RX_CONTROL_slave ports of Self Test class are connected to RX_DATA/RX_CONTROL port of Self Test class. TX_DATA_slave/TX_CONTROL_slave ports of Self Test class are connected to TX_DATA/TX_CONTROL ports of Self Test class.

(6) In Normal mode, LSTC.LSTM = 0, Uart mode is selected, RX_DATA_uart/RX_CONTROL_uart ports of Self Test class are connected to RX_DATA/RX_CONTROL ports of Self Test class. TX_DATA_uart/TX_CONTROL_uart ports of Self Test class are connected to TX_DATA/TX_CONTROL ports of Self Test class.

7.44. Timeout handling process

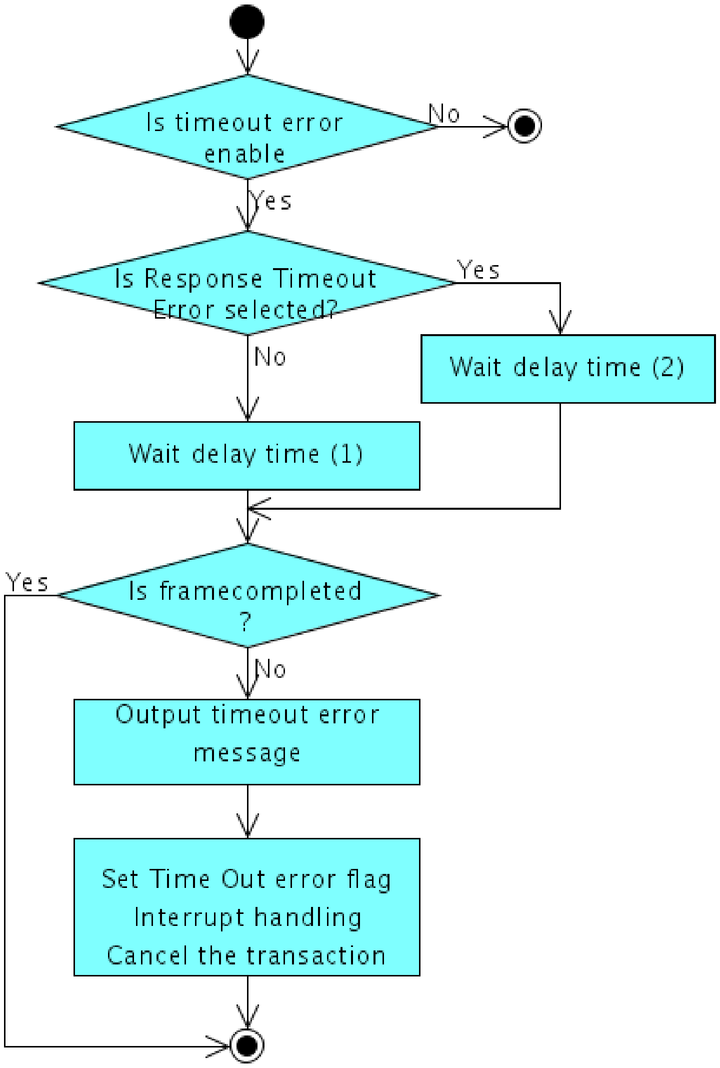


Figure 7.52: Timeout handling process flow

Table 7.7: The waiting conditions for Timeout Handling

Waiting condition	Meaning
1	(48 or 49)+14*(communication byte count +1)*Tbit
2	14*(communication byte count +1)*Tbit

Explanation:

(1) This function is called when Master mode handling, Slave mode handling are called.

(2) The value 49 is selected for classic checksum and the value 48 is chosen for enhance checksum in the waiting condition (1). Tbit is the bit time for the waiting conditions (1) and (2).

(3) If response timeout error is not selected, the duration of timeout is waiting condition (1). If response timeout error is selected, the duration of timeout is waiting condition (2).

(4) After the duration of timeout, if frame is not completed the timeout error message is outputted and the transaction is canceled. The timeout error flag is set to 1 and interrupt handling is called to assert status interrupt.

(5) If the transmission/ reception is finished successfully, the timeout process is canceled.

7.45. SW reset handling process

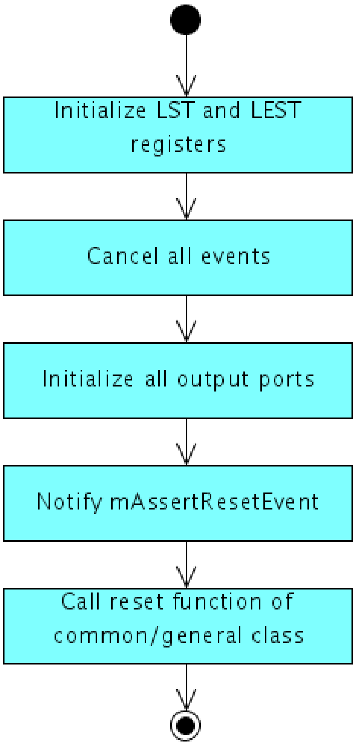

Initialize LST and LEST registers
Cancel all events
Initialize all output ports
Notify mAssertResetEvent
Call reset function of common/general class

Figure 7.53: SW reset handling flow

Explanation:

(1) If LCUC.OM0 bit is set to 1, SWResetMethod will be called. The operations are same as reset handling process excepted two registers LST and LEST are cleared only.

7.46. handleCommand process

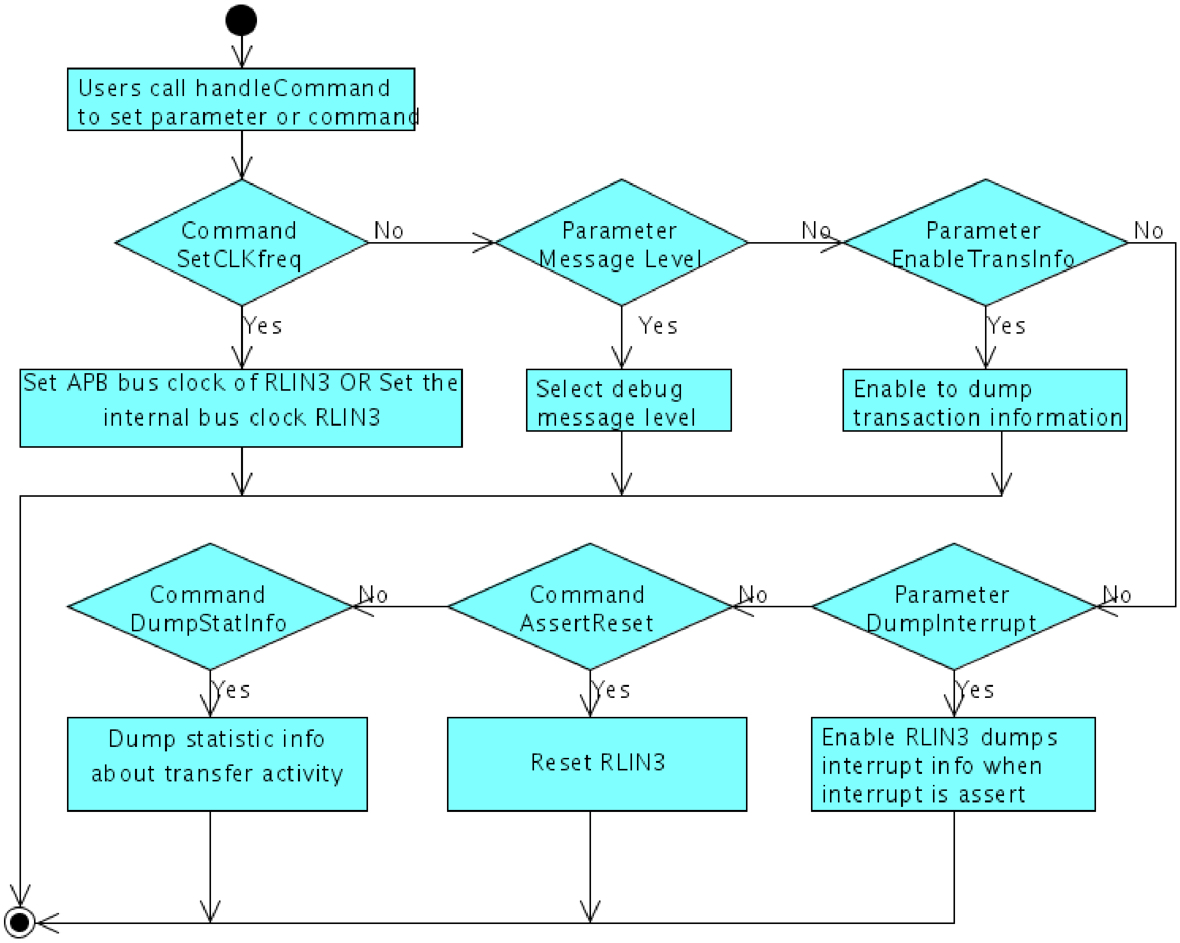

Users call handleCommand to set parameter or command
Command SetCLKfreq
No
Parameter Message Level
No
Parameter EnableTransInfo
No
Yes
Yes
Yes
Set APB bus clock of RLIN3 OR Set the internal bus clock RLIN3
Select debug message level
Enable to dump transaction information
Command DumpStatInfo
No
Command AssertReset
No
Parameter DumpInterrupt
No
Yes
Yes
Yes
Dump statistic info about transfer activity
Reset RLIN3
Enable RLIN3 dumps interrupt info when interrupt is assert

Figure 7.54: handleCommand operation flow of RLIN3 model

Explanation:

(1) Users call handleCommand to set parameter or command.

(2) If parameter or command is:

(2.1) SetCLKfreq, it sets the frequency of clock APB bus clock(Hz) or the frequency of internal bus clock RLIN3 (Hz). Refer to table 6.3.

(2.2) MessageLevel, it selects debug message level of the RLIN3 model.

(2.3) EnableTransInfo, it enables dumping the information of the RLIN3 model when it is turned on/off. Refer to table 6.6.

(2.4) DumpInterrupt, it enables dumping interrupt information when interrupt is asserted. Refer to table 6.4.

(2.5) AssertReset, it resets the RLIN3 model.

(2.6) DumpStatInfo, it dumps information about the RLIN3 model. When this command is called, the statistical information about transfer activity is dumped. Refer to table 6.5.

8. Class explanation

8.1. Class relationships

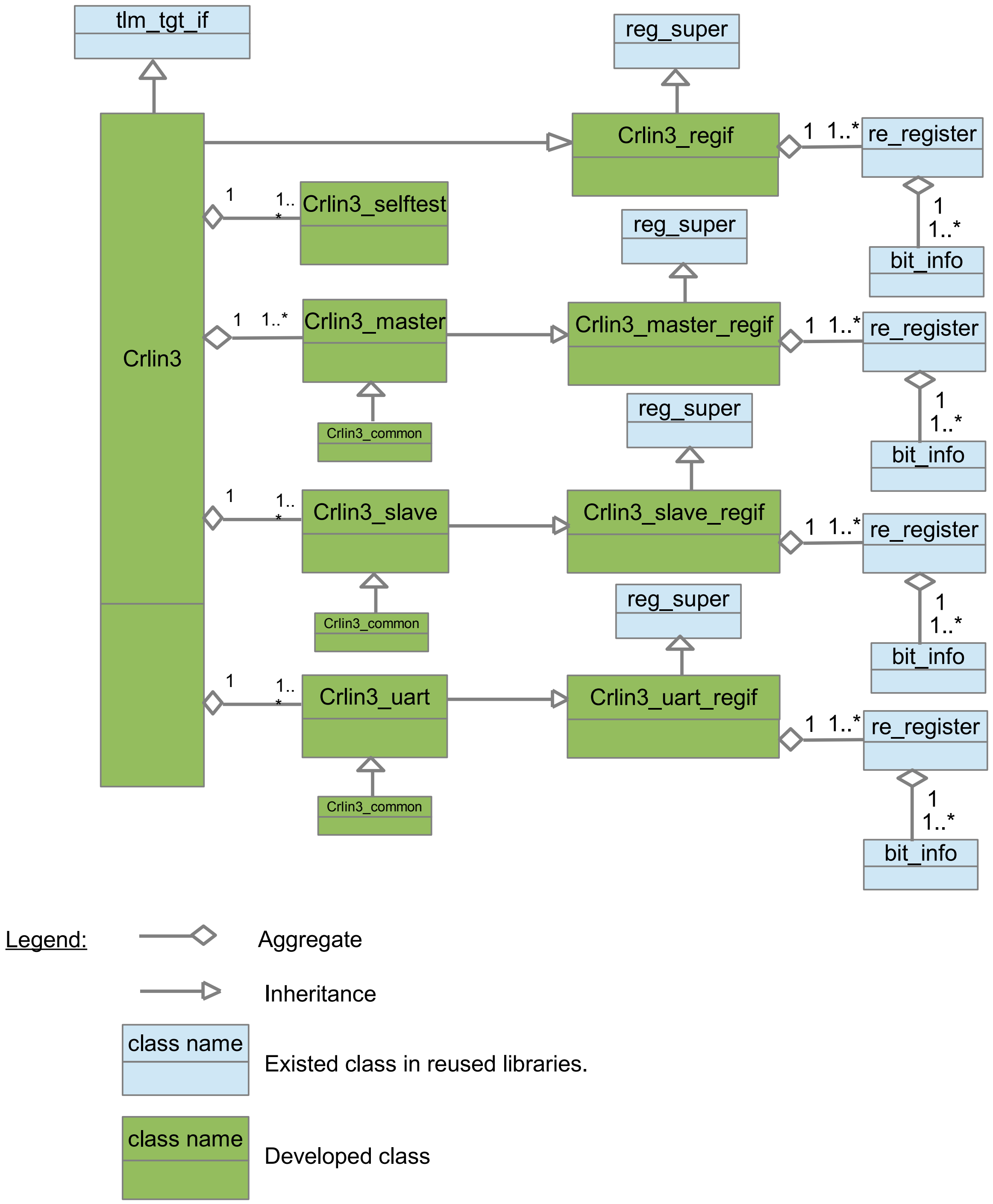


Figure 8.1: Relationship of classes

Table 8.1: Class explanation

No.	Class name	Explanation
1	sc_module	Existed class in SystemC library is used to implement a module.
2	bit_info	Existed class in General register library is used to implement a bit in a register.
3	re_register	Existed class in General register library is used to implement a register. A register can have one or many bit_info instances.
4	reg_super	Existed interface class in General register library is inherited by modules that use re_register class.
5	tlm_tgt_if	Existed interface class in TLM common library is inherited by modules that have a target socket.
6	Crlin3_regif	Crlin3_regif has registers for all modes. This class uses the re_register class so that it must inherit reg_super class.
7	Crlin3_master_regif	Crlin3_master_regif has registers for master mode. This class uses the re_register class so that it must inherit reg_super class.
8	Crlin3_slave_regif	Crlin3_slave_regif has registers for slave mode. This class uses the re_register class so that it must inherit reg_super class.
9	Crlin3_uart_regif	Crlin3_uart_regif has registers for uart mode. This class uses the re_register class so that it must inherit reg_super class.
10	Crlin3_master	Crlin3_master is implemented to represent register bank of RLIN3 model in Master mode.
11	Crlin3_slave	Crlin3_slave is implemented to represent register bank of RLIN3 model in Slave mode.
12	Crlin3_uart	Crlin3_uart is implemented to represent register bank of RLIN3 model in UART mode.
13	Crlin3	Crlin3 class represents the RLIN3 model. This class inherits tlm_tgt_if, Crlin3_regif, and sc_module class. Besides, it instantiates Crlin3_master, Crlin3_slave and Crlin3_uart inside as register banks for three modes.
14	Crlin3_selftest	Crlin3_selftest is implemented to represent input/output ports control of RLIN3 model in Self Test mode.

8.2. Class Crlin3_slave

(1) This section is generated by the doxygen

Table 8.2: List of implemented functions in Crlin3_slave class

No.	Function	Level	Description
1	Crlin3_slave(sc_module_name name, Crlin3 *parent)	Public	Constructor of Crlin3_slave class
2	~Crlin3_slave()		Destructor of Crlin3_slave class
3	void ResetSlave(bool is_active)		Reset the slave module
4	void SetSlaveClock(double clk_lsb)		This function is used to set up new frequency for clock of slave module

No.	Function	Level	Description
5	void UpdateSlaveRegs(RlinRegs rlin_reg)		This function is used to update the new value for register in slave module.
6	std::string GetCurrentStatus()		This function is used to get the information of slave module
7	void Lin3EnterSelfTest()		This function is used to active self test function in the slave module
8	std::string slave_reg_command(const std::vector<std::string>& args)		This function is used to handle the parameters/commands in slave module.
9	bool slave_reg_rd(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of slave module
10	bool slave_reg_rd_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of slave module in debug mode
11	bool slave_reg_wr(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of slave module
12	bool slave_reg_wr_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of slave module in debug mode
13	void SoftwareReset(unsigned int om0_val)		This function is used to trigger software reset
14	void TimeoutChecking()	Private	This function is used to calculate the wait time to handle error status.
15	void ReceiveMethod(void)		This method is used to trigger reception process to handle received data in slave module.
16	void TimeoutHandlingMethod(void)		This method is used to handle the error status
17	void StartRespondMethod(void)		This method is used to start the transmit process of slave module
18	void OutputData(unsigned int tx_control, unsigned int tx_data)		This function is used to export data to output ports of slave module
19	void UpdateStatus(eSTATUS_FLAG flag)		This method is used to update RLIN3's current status to status registers. Dump operation information, assert interrupts and transfer data according to current status of RLIN3 model.
20	void UpdateErrorStatus (eERROR_FLAG error_kind)		This function is used to export error report in slave module
21	void UpdateRegisters(eREG_KIND reg_kind, unsigned int value)		Update data registers with the new value. The new values are gotten from receiving process

No.	Function	Level	Description
22	void _re_printf(const std::string group, const char *message, ...)		This function is used to dump the debug message
23	void UpdateRegsOfLin3()		This function is used to update the new value into the registers of slave module
24	void CheckWriteLDBRN(RegCBstr str, vpcl::re_register *reg, unsigned int index)		This function is use to check the writing condition for LDBRn (n: 1~8) registers
25	void cb_LWBR_LWBR0(RegCBstr str)		Callback function of register LWBR
26	void cb_LBRP0_BRP(RegCBstr str)		Callback function of register LBRP0
27	void cb_LBRP1_BRP(RegCBstr str)		Callback function of register LBRP1
28	void cb_LSTC_LSTM(RegCBstr str)		Callback function of register LSTC
29	void cb_LMD_LMD(RegCBstr str)		Callback function of register LMD
30	void cb_LBFC_LBLT(RegCBstr str)		Callback function of register LBFC
31	void cb_LSC_IBHS(RegCBstr str)		Callback function of register LSC
32	void cb_LWUP_WUTL(RegCBstr str)		Callback function of register LWUP
33	void cb_LIE_FTCIE(RegCBstr str)		Callback function of register LIE
34	void cb_LEDE_BERE(RegCBstr str)		Callback function of register LEDE
35	void cb_LCUC_OM0(RegCBstr str)		Callback function of register LCUC
36	void cb_LTRC_FTS(RegCBstr str)		Callback function of register LTRC
37	void cb_LMST_OMM0(RegCBstr str)		Callback function of register LMST
38	void cb_LST_FTC(RegCBstr str)		Callback function of register LST
39	void cb_LEST_BER(RegCBstr str)		Callback function of register LEST
40	void cb_LDFC_RFDL(RegCBstr str)		Callback function of register LDFC
41	void cb_LIDB_ID(RegCBstr str)		Callback function of register LIDB
42	void cb_LCBR_CKSM(RegCBstr str)		Callback function of register LCBR
43	void cb_LUDB0_UDB(RegCBstr str)		Callback function of register LUDB0
44	void cb_LDB1_LDB(RegCBstr str)		Callback function of register LDB1
45	void cb_LDB2_LDB(RegCBstr str)		Callback function of register LDB2
46	void cb_LDB3_LDB(RegCBstr str)		Callback function of register LDB3
47	void cb_LDB4_LDB(RegCBstr str)		Callback function of register LDB4
48	void cb_LDB5_LDB(RegCBstr str)		Callback function of register LDB5
49	void cb_LDB6_LDB(RegCBstr str)		Callback function of register LDB6
50	void cb_LDB7_LDB(RegCBstr str)		Callback function of register LDB7
51	void cb_LDB8_LDB(RegCBstr str)		Callback function of register LDB8
52	void cb_LUOER_UTOE(RegCBstr str)		Callback function of register LUOER
53	void cb_LUOR1_UEBE(RegCBstr str)		Callback function of register LUOR1

No.	Function	Level	Description
54	void cb_LUTDR_UTD(RegCBstr str)		Callback function of register LUTDR
55	void cb_LURDR_URD(RegCBstr str)		Callback function of register LURDR
56	void cb_LUWTDR_UWTD(RegCBstr str)		Callback function of register LUWTDR

8.3. Class Crlin3_master

(1) This section is generated by the doxygen

Table 8.3: List of implemented functions in Crlin3_master class

No.	Function	Level	Description
1	Crlin3_master(sc_module_name name, Crlin3 *parent)	Public	Constructor of Crlin3_master class
2	virtual ~Crlin3_master()		Destructor of Crlin3_master class
3	void ResetMaster(bool is_active)		Reset the master module
4	void SetMasterClock(double clkc, double pclk)		This function is used to set up new frequency for clock of master module
5	void UpdateMasterRegs(RlinRegs rlin_reg)		This function is used to update the new value for register in master module.
6	std::string GetCurrentStatus()		This function is used to get the information of master module
7	void Lin3EnterSelfTest()		This function is used to active self test function in the master module
8	std::string master_reg_command(const std::vector<std::string>& args)		This function is used to handle the parameters/commands in master module.
9	bool master_reg_rd(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of master module
10	bool master_reg_rd_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of master module in debug mode
11	bool master_reg_wr(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of master module
12	bool master_reg_wr_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of master module in debug mode
13	void SoftwareReset(unsigned int om0_val)		This function is used to trigger software reset
14	void TimeoutChecking()	Private	This function is used to calculate the wait time to handle error status.

No.	Function	Level	Description
15	void ReceiveMethod(void)		This method is used to trigger reception process to handle received data in slave module.
16	void TimeoutHandlingMethod(void)		This method is used to handle the error status
17	void TransRespondMethod(void)		This method is used to start the transmit process of slave module
18	void OutputData(unsigned int tx_control, unsigned int tx_data)		This function is used to export data to output ports of slave module
19	void UpdateStatus(eSTATUS_FLAG flag)		This method is used to update RLIN3's current status to status registers. Dump operation information, assert interrupts and transfer data according to current status of RLIN3 model.
20	void UpdateErrorStatus (eERROR_FLAG error_kind)		This function is used to export error report in slave module
21	void UpdateRegisters(eREG_KIND reg_kind, unsigned int value)		Update data registers with the new value. The new values are gotten from receiving process
22	void _re_printf(const std::string group, const char *message, ...)		This function is used to dump the debug message
23	void UpdateRegsOfLin3()		This function is used to update the new value into the registers of slave module
24	void CheckWriteLDBRN(RegCBstr str, vpcl::re_register *reg, unsigned int index)		This function is use to check the writing condition for LDBRn (n: 1~8) registers
25	void cb_LWBR_LWBR0(RegCBstr str)		Callback function of register LWBR
26	void cb_LBRP0_LBRP0(RegCBstr str)		Callback function of register LBRP0
27	void cb_LBRP1_LBRP1(RegCBstr str)		Callback function of register LBRP1
28	void cb_LSTC_LSTM(RegCBstr str)		Callback function of register LSTC
29	void cb_LMD_LMD(RegCBstr str)		Callback function of register LMD
30	void cb_LBFC_BLT(RegCBstr str)		Callback function of register LBFC
31	void cb_LSC_IBHS(RegCBstr str)		Callback function of register LSC
32	void cb_LWUP_WUTL(RegCBstr str)		Callback function of register LWUP
33	void cb_LIE_FTCIE(RegCBstr str)		Callback function of register LIE
34	void cb_LEDE_BERE(RegCBstr str)		Callback function of register LEDE
35	void cb_LCUC_OM0(RegCBstr str)		Callback function of register LCUC
36	void cb_LTRC_FTS(RegCBstr str)		Callback function of register LTRC
37	void cb_LMST_OMM0(RegCBstr str)		Callback function of register LMST
38	void cb_LST_FTC(RegCBstr str)		Callback function of register LST
39	void cb_LEST_BER(RegCBstr str)		Callback function of register LEST
40	void cb_LDFC_RFDL(RegCBstr str)		Callback function of register LDFC

No.	Function	Level	Description
41	void cb_LIDB_ID(RegCBstr str)		Callback function of register LIDB
42	void cb_LCBR_CKSM(RegCBstr str)		Callback function of register LCBR
43	void cb_LUDB0_UDB(RegCBstr str)		Callback function of register LUDB0
44	void cb_LDB1_LDB(RegCBstr str)		Callback function of register LDB1
45	void cb_LDB2_LDB(RegCBstr str)		Callback function of register LDB2
46	void cb_LDB3_LDB(RegCBstr str)		Callback function of register LDB3
47	void cb_LDB4_LDB(RegCBstr str)		Callback function of register LDB4
48	void cb_LDB5_LDB(RegCBstr str)		Callback function of register LDB5
49	void cb_LDB6_LDB(RegCBstr str)		Callback function of register LDB6
50	void cb_LDB7_LDB(RegCBstr str)		Callback function of register LDB7
51	void cb_LDB8_LDB(RegCBstr str)		Callback function of register LDB8
52	void cb_LUOER_UTOE(RegCBstr str)		Callback function of register LUOER
53	void cb_LUOR1_UEBE(RegCBstr str)		Callback function of register LUOR1
54	void cb_LUTDR_UTD(RegCBstr str)		Callback function of register LUTDR
55	void cb_LURDR_URD(RegCBstr str)		Callback function of register LURDR
56	void cb_LUWTDR_UWTD(RegCBstr str)		Callback function of register LUWTDR

8.4. Class Crlin3_uart

(1) This section is generated by the doxygen

Table 8.4: List of implemented functions in Crlin3_uart class

No.	Function	Level	Description
1	Crlin3_uart(sc_module_name name, Crlin3 *parent)	Public	Constructor of Crlin3_uart class
2	~Crlin3_uart()		Destructor of Crlin3_uart class
3	void ResetUart(bool is_active)		It initializes setting and variables of Crlin3_uart class
4	void SetUartClock(double clkc, double pclk)		This function is used to set up new frequency for clock of uart module
5	void UpdateUartRegs(RlinRegs rlin_reg)		This function is used to update the new value for register in uart module
6	std::string GetCurrentStatus()		This function is used to get the information of uart module
7	std::string uart_reg_command(const std::vector<std::string>& args)		This function is used to handle the parameters/commands in uart module.
8	bool uart_reg_rd(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of uart module

No.	Function	Level	Description
9	bool uart_reg_rd_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to read registers of uart module in debug mode
10	bool uart_reg_wr(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of uart module
11	bool uart_reg_wr_dbg(unsigned int addr,unsigned char *p_data,unsigned int size)		This function is used to write to registers of uart module in debug mode
12	void SoftwareReset(unsigned int om0_val)		This function is used to trigger software reset
13	void ReceptionMethod(void)	Private	This method is used to trigger the receive process to handle data from input ports
14	void SingleTransMethod(void)		This method is used to trigger transmit process in single mode
15	void MultiTransMethod(void)		This method is used to trigger transmit process in multiple mode
16	void MultiTransWaitStopBitMethod(void)		This method is used to check the conditions to trigger transmit process in multiple mode
17	unsigned int AddParity(unsigned int data)		This function is used to add the parity bit to transmit data before triggering the transmit process
18	unsigned int ChangeDataDirection (bool is_msb, unsigned int data, unsigned int bit_length)		This function change the direction from MSB to LSB Or LSB to MSB the data before transmit process
19	void OutputData(unsigned int tx_control, unsigned int tx_data)		This function is used to export data to output ports of uart module
20	void UpdateStatus(eSTATUS_FLAG flag)		This method is used to update RLIN3's current status to status registers. Dump operation information, assert interrupts and transfer data according to current status of RLIN3 model
21	void UpdateErrorStatus(eERROR_FLAG flag)		This function is used to export error report in uart module
22	void UpdateRegisters(eREG_KIND reg_kind, unsigned int value)		Update data registers with the new value. The new values are gotten from receiving process
23	void _re_printf(const std::string group, const char *message, ...)		This function is used to dump the debug message
24	void UpdateRegsOfLin3()		This function is used to
25	void CheckWriteLDBRN(RegCBstr str, vpcl::re_register *reg, unsigned int index)		This function is use to check the writing conditions for LDBRn (n: 1~8) registers

No.	Function	Level	Description
26	void cb_LWBR_LWBR0(RegCBstr str)		Callback function of register LWBR
27	void cb_LBRP0_BRP(RegCBstr str)		Callback function of register LBRP0
28	void cb_LBRP1_BRP(RegCBstr str)		Callback function of register LBRP1
29	void cb_LSTC_LSTM(RegCBstr str)		Callback function of register LSTC
30	void cb_LMD_LMD(RegCBstr str)		Callback function of register LMD
31	void cb_LBFC_UBLS(RegCBstr str)		Callback function of register LBFC
32	void cb_LSC_IBHS(RegCBstr str)		Callback function of register LSC
33	void cb_LWUP_WUTL(RegCBstr str)		Callback function of register LWUP
34	void cb_LIE_FTCIE(RegCBstr str)		Callback function of register LIE
35	void cb_LEDE_BERE(RegCBstr str)		Callback function of register LEDE
36	void cb_LCUC_OM0(RegCBstr str)		Callback function of register LCUC
37	void cb_LTRC_RTS(RegCBstr str)		Callback function of register LTRC
38	void cb_LMST_OMM0(RegCBstr str)		Callback function of register LMST
39	void cb_LST_FTC(RegCBstr str)		Callback function of register LST
40	void cb_LEST_BER(RegCBstr str)		Callback function of register LEST
41	void cb_LDFC_MDL(RegCBstr str)		Callback function of register LDFC
42	void cb_LIDB_ID(RegCBstr str)		Callback function of register LIDB
43	void cb_LCBR_CKSM(RegCBstr str)		Callback function of register LCBR
44	void cb_LUDB0_UDB(RegCBstr str)		Callback function of register LUDB0
45	void cb_LDB1_LDB(RegCBstr str)		Callback function of register LDB1
46	void cb_LDB2_LDB(RegCBstr str)		Callback function of register LDB2
47	void cb_LDB3_LDB(RegCBstr str)		Callback function of register LDB3
48	void cb_LDB4_LDB(RegCBstr str)		Callback function of register LDB4
49	void cb_LDB5_LDB(RegCBstr str)		Callback function of register LDB5
50	void cb_LDB6_LDB(RegCBstr str)		Callback function of register LDB6
51	void cb_LDB7_LDB(RegCBstr str)		Callback function of register LDB7
52	void cb_LDB8_LDB(RegCBstr str)		Callback function of register LDB8
53	void cb_LUOER_UTOE(RegCBstr str)		Callback function of register LUOER
54	void cb_LUOR1_UEBE(RegCBstr str)		Callback function of register LUOR1
55	void cb_LUTDR_UTD(RegCBstr str)		Callback function of register LUTDR
56	void cb_LURDR_URD(RegCBstr str)		Callback function of register LURDR
57	void cb_LUWTDR_UWTD(RegCBstr str)		Callback function of register LUWTDR

8.5. Class Crlin3_common

(1) This section is generated by the doxygen

Table 8.5: List of implemented functions in Crlin3_common class

No.	Function	Level	Description
1	Crlin3_common(sc_module_name name)	Public	Constructor of Crlin3_common class
2	~Crlin3_common()		Destructor of Crlin3_common class
3	void SetSelftestMode(bool is_selftest)		Using this API to configure self-test mode for RLIN3 common
4	void StopOperation(bool is_hw_reset)		This function is used to stop the operation of the RLIN3.
5	void SetDataTransfer(unsigned int index, unsigned int data)		This function is used to update transferred data for the Crlin3_common class
6	void SetLinClock(double clkc, double pclk)		This function is used to set the RLIN3 clock
7	bool CheckZeroClock()		Using this function in call back functions and trigger methods to check zero clock before processing
8	void SetConfigFactors(RlinRegs config_struct)		This function is used to set the RLIN3 register's values
9	void TransmitProcess(eTRANS_STATUS status)		This function is used to invoke the transmit process of RLIN3
10	void ReceptionProcess(unsigned int data_input, unsigned int control_input, eTRANS_STATUS status)		This function is used to invoke the reception process of RLIN3
11	unsigned int CalcBitTime()		Using the RLIN3 clock and registers setting, this function calculates the bit time value of the RLIN3's operation
12	virtual void OutputData(unsigned int tx_control, unsigned int tx_data) = 0		This function is a pure virtual function which is for outside classes to update the value on output ports
13	virtual void UpdateStatus(eSTATUS_FLAG flag) = 0		This function is a pure virtual function which is for outside classes to update the current status of RLIN3
14	virtual void UpdateErrorStatus(eERROR_FLAG flag) = 0		This function is a pure virtual function which is for outside classes to update the current error of RLIN3
15	virtual void UpdateRegisters(eREG_KIND reg_kind, unsigned int value) = 0		This function is a pure virtual function which is for outside classes to update the value for corresponding registers
16	virtual void _re_printf(const std::string group, const char *message, ...) = 0		This function is a pure virtula function which is for outside classes to dump the debug message

No.	Function	Level	Description
17	void TransmitDataMethod()	Protected	This method is used to control all phases of transmission process of RLIN3
18	void WaitToBitErrorMethod()		This method is used to check the bit error of RLIN3 process
19	void WaitToPhyscBusErrorMethod()		Tis method is waited to bit boundary to assert error flag
20	void WriteOutputMethod()		This method is used to output data on ports
21	void CheckBitErrorMethod()		This method is used to check the bit error
22	void ReceptionCompleteMethod()		This method use to handle reception complete wake-up
23	void Initialize()		This function is used to initialize internal variables of the Crlin3_common
24	void TransmitWakeup()		This function is used to transmit in the wake up mode
25	void TransmitHeaderLoop()		This function is used to transfer the header of RLIN3 protocol
26	void TransmitRespLoop()		This function is used to transmit the response data
27	void RespReception(unsigned int data_input, unsigned int control_input, unsigned int mode)		This function is used to set the data for the response reception process
28	void HeaderReception(unsigned int data_input, unsigned int control_input, unsigned int mode)		This function is used to set the data for the header reception process
29	void FinishReceiveHandle(unsigned int mode, unsigned int index)		This function is used to handle when finish receiving data
30	unsigned int CalcChecksumValue (unsigned int pre_checksum_val)		This function is used to calculate the check sum value
31	unsigned int CalcBaudRate(unsigned int bit_time)		This function is used to calculate the baud rate value
32	bool CheckIDParity (unsigned int PID)		This function is used to check the validation of the ID
33	unsigned int GetParity (unsigned int data,unsigned int bit_length)		This function is used to get the parity value
34	unsigned int CalcNumOfByte(eTRANS_STATUS status)		This function is used to get the number of transferred bytes
35	double CalcBitBoundary(unsigned int max_bit, unsigned int input_data, unsigned int expected_data)		This function is used to calculate wait time for the bit error case

No.	Function	Level	Description
36	bool CheckBreakLowPeriod(unsigned int mode)		This function is used to check the break low period constraint

8.6. Class Crlin3_selftest

(1) This section is generated by the doxygen

Table 8.6: List of implemented functions in Crlin3_selftest class

No.	Function	Level	Description
1	Crlin3_selftest(sc_module_name name)	Public	Constructor of Crlin3 _selfTestclass
2	~Crlin3_selftest()		Destructor of Crlin3_selftest class
3	void SetSelfTestMode(bool is_selfTest)		Set the status of the self test mode
4	void SetOptMode(unsigned int opt_mode)		Set the current operation mode of RLIN3
5	void SetAssertReset(bool is_reset)		Set the reset of the Crlin3_selfTest
6	void SelfTestHandlingMethod()	Private	This method is used to switch ports for RLIN during is self test mode

8.7. Class Crlin3

(1) This section is generated by the doxygen

Table 8.7: List of implemented functions in Crlin3 class

No.	Function	Level	Description
1	Crlin3(sc_module_name name)	Public	Constructor of Crlin3 class
2	~Crlin3()		Destructor of Crlin3 class
3	void DeAssertIntrMethod(void)	Private	This method is used to deasserted RLIN3's interrupts and dump interrupt message
4	void ResetMethod (void)		This method is used to control the reset progress of both the port reset and command reset.
5	void CmdResetMethod(void)		This method is used to control the reset progress by the command reset
6	void CancelCmdResetMethod(void)		This method is used to cancel the reset progress of the reset command.
7	void CLK_LSBPeriodMethod(void)		Whenever the clk_lsb port has a change, this method will be invoked to call SetCLKfreq to set the new frequency for the internal variable.
8	void WriteLin3IntMethod(void)		This method is used to write the values to RLIN3's interrupt ports and ump interrupt messages
9	void Initialize(void)		This function is called in the constructor and whenever the reset progress takes place to initialize internal variables and registers of RLIN3

No.	Function	Level	Description
10	void EnableReset(const bool is_active)		This function will initialize internal variables, registers and ports of RLIN3
11	void AssertReset(const double delay,const double period)		This function will delay the reset a period containing in the delay variable and process the reset for a period containing in the period variable
12	void DumpInfo(const char *type, const char *message, ...)		This message will dump message according to the type and current mode of RLIN3
13	void DumpStatInfo(void)		This function is used to dump the operation status of RLIN3
14	void SetCLKfreq(std::string clk_name, double clk_freq)		This function is called when users want to use the handle command or when clock ports are changed to update new frequencies for corresponding clocks
15	void DumpInterruptMsg(Crlin3_common::eINTERRUPT_KIND interrupt_id, bool int_assert)		This function is called whenever users set the DumpIntMsg true and an interrupt occurs
16	unsigned int GetRegBitsVal(unsigned int reg, unsigned int lower_index,unsigned int upper_index)		This function is used to mask particular set bits and clear other bits outside the mask
17	unsigned int SetRegBitsVal(unsigned int reg, unsigned int pre_reg, unsigned int lower_index, unsigned int upper_index)		This register will be used to set the previous register value for the masked bits
18	double GetTimeResolution(void)		This function is used to get the time unit
19	void SetLatency_TLM(void)		Whenever the SetCLKfreq is called, the bus latency of the TLM common class will be calculated again by this function
20	void DumpOperationInfo(const char *frame_name, const char *operation, unsigned int id_val, unsigned int data_val, const char *no_cksum, unsigned int cksum_val)		This function is called whenever users want to dump the operation info of RLIN3
21	void UpdateRlin3Reg(Crlin3_common::RlinRegs rlin_reg)		Whenever the sub-class wants to update the register values from it to the Crlin3 class, this function will be called
22	void SetSelfTestFunc(bool is_selftest)		Whenever users unlock the self-test mode, this function is called to activate the self test mode
23	void UpdateAllRegs()		This function is called to update the register's value fro Crlin3 class to sub-classes such as Master, Slave and Uart

No.	Function	Level	Description
24	bool SelftestAccessCheck (vpcl::re_register *reg, bool is_wr, uint pre_data, uint status, bool is_selftest)		This function is used to check access register in Selftest mode
25	void RegisterAccessCheck (vpcl::re_register *reg, bool is_wr, uint pre_data, std::string bit_name, Crlin3_common::eACCESS_MODE expected_access, unsigned int lower_index,unsigned int upper_index)		This function is used to update value of a register in some special cases such as read-return-zero
26	void WriteLin3IntT(bool value)		This function is called to write the value for the transmission interrupt
27	void WriteLin3IntR(bool value)		This function is called to write the value for the response interrupt
28	void WriteLin3IntS(bool value)		This function is called to write the value for the status interrupt
29	void SoftwareReset(unsigned int om0_val)		This function is used to trigger software reset
30	std::string RegIfCommand(const std::vector<std::string>& args)		This function is used to set reg command for sub classes
31	void tgt_acc(tlm::tlm_generic_payload &trans, sc_time &t)		This function is used to process the normal TLM transaction
32	unsigned int tgt_acc_dbg(tlm::tlm_generic_payload &trans)		This function is used to process the debug TLM transaction

Revision History					
Rev.	**Modified Contents**	**Agreed by Customer**	**Approved by RVC**	**Checked by**	**Created by**
1.0	- Create based on INT-SLD-12006 (C1x/RLIN3) + Update to reflect HWM chapter of E2x + pclk and clkc clock are merged to clk_lsb clock + Remove LSTC.LSFWC bit in LIN mode + Update bit range of LSTC.LSTME bit in LIN mode + Update description of LMD.LIOS bit in LIN mode + Remove LWBR.LWBR0, LMD.LCKC bit in Slave mode + Remove LWBR.LWBR0, LMD.LIOS, LMD.LCKS, LSC.IBHS, LCUC.OM1 in UART mode + Remove register LURDE in all modes + Update message list in Table 6.12 + Change file name from rlin3_selfTest into rlin3_selftest			Uyen Le 11/26/2015	Son Tran 11/26/2015