Datasheet for Telink 2.4GHz RF SoC TLSR8355

DS-TLSR8355-E1

Ver 0.1.0

2019/12/16

Keyword:

2.4GHz; Features; Typical Applications; Ordering Info; Pin Layout

Brief:

This datasheet is dedicated for Telink 2.4GHz RF SoC TLSR8355. In this datasheet, key features, ordering information and pin layout of the TLSR8355 are mainly introduced.

TELINK SEMICONDUCTOR

Published by
Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,
Zhangjiang Hi-Tech Park, Shanghai, China

© Telink Semiconductor
All Right Reserved

Legal Disclaimer

Telink Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein or in any other disclosure relating to any product.

Telink Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others

The products shown herein are not designed for use in medical, life-saving, or life-sustaining applications. Customers using or selling Telink Semiconductor products not expressly indicated for use in such applications do so entirely at their own risk and agree to fully indemnify Telink Semiconductor for any damages arising or resulting from such use or sale.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Revision History

Version	Major Changes	Date	Author
0.1.0	Preliminary release	2019/7	SY, SGJ, LY, YCQ, CJZ, YHL, MZD, SML, Cynthia, JF

1 Table of contents

2 Table of Figures

3 Table of Tables

1 Overview

The RoHS-compliant TLSR8355 series with internal Flash is dedicated to 2.4GHz RF System-On-Chip solution, such as wireless mouse and wireless USB dongle.

The TLSR8355 has hardware OTA upgrades support, allowing convenient product feature roll outs and upgrades.

The TLSR8355 also includes a full range of on-chip peripherals for interfacing with external components such as LEDs, sensors, keyboards, and motors. This makes it an ideal single-chip solution for HID (Human Interface Devices) application.

The TLSR8355 series have passed ETSI EN 300 328 and EN 300 440 Class 2 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T66 (Japan) certification.

1.1 Block diagram

The TLSR8355 is designed to offer high integration, ultra-low power application capabilities. The system's block diagram is as shown in Figure 1-1.

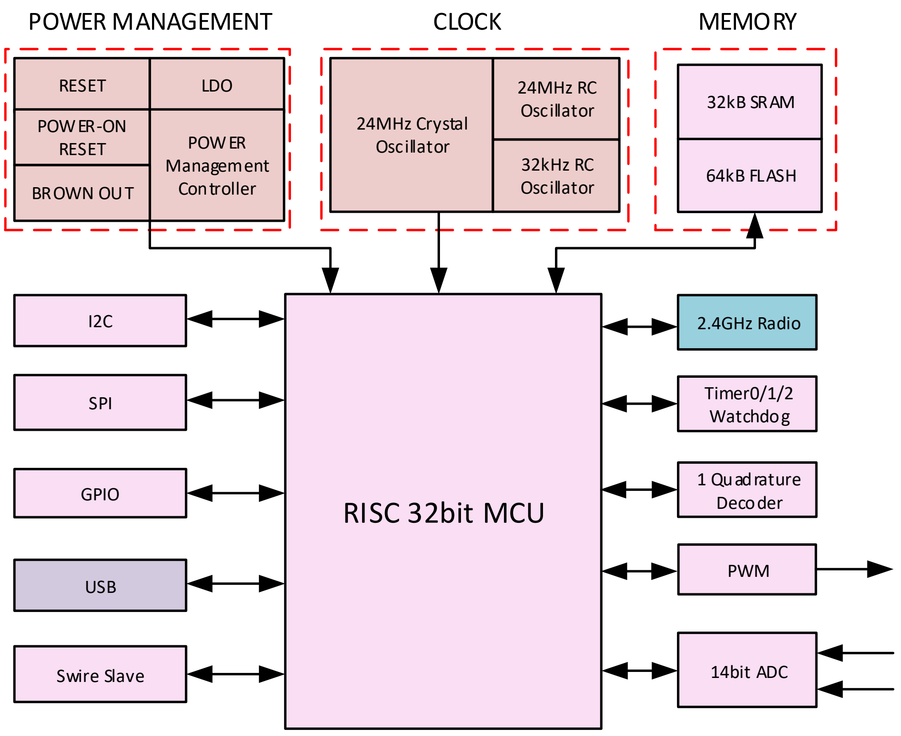


Figure 1- 1 Block diagram of the system

*Note:

1) Modules marked with different colors belong to different power domains. Power state of each power domain can be controlled independent of other power domains, for example, the 2.4GHz Radio can be independently powered on or powered down irrespective of other modules such as power management module, clock, and etc.
2) The 2.4GHz Radio and USB are powered down by default.
3) The power management module and clock should be always powered on, even in deep sleep.
4) In deep sleep, except for the power management and clock, all other modules should be powered down.

The TLSR8355 integrates a power-balanced 32-bit MCU, 2.4GHz Radio, 32kB (16k+16k) SRAM, 64kB internal Flash, 14bit ADC, 5-channel PWM (1-channel IR/IR FIFO/IR DMA FIFO), one quadrature decoder (QDEC), abundant and flexible GPIO interfaces, and nearly all the peripherals needed for 2.4GHz RF System-On-Chip application development.

The TLSR8355 also includes multi-stage power management design allowing ultra-low power operation and making it the ideal candidate for power-constraint applications.

With the high integration level of TLSR8355, few external components are needed to satisfy customers' ultra-low cost requirements.

1.2 Key features

1.2.1 General features

General features are as follows:

1) 4-byte Chip UID (Unique ID).
2) Embedded 32-bit proprietary microcontroller.
   - ✧ Better power-balanced performance than ARM M0
   - ✧ Instruction cache controller
   - ✧ Maximum running speed up to 48MHz
3) Internal 64kB Flash
4) 32kB on-chip SRAM with retention in deep sleep
5) RTC and other timers:
   - ✧ Clock source of 24MHz Crystal and 32kHz/24MHz embedded RC oscillator
   - ✧ Three general 32-bit timers with four selectable modes in active mode
   - ✧ Watchdog timer
   - ✧ A low-frequency 32kHz timer available in low power mode
6) A rich set of I/Os:
   - ✧ Up to 12 GPIOs. All digital IOs can be used as GPIOS.
   - ✧ SPI
   - ✧ I2C
   - ✧ USB
   - ✧ Swire Slave debug Interface
   - ✧ Manchester decoder interface selectable as wakeup source
7) Up to 6 channels of differential PWM:
   - ✧ PWM1~PWM5: 5-channel normal PWM output
   - ✧ PWM0: 1 channel with normal mode as well as additional IR/IR FIFO/IR DMA FIFO mode for IR generation

8) Sensor:
    - ✧ 14bit 2-channel (only GPIO input) SAR ADC
    - ✧ Temperature sensor
9) One quadrature decoder
10) Embedded hardware AES and AES-CCM
11) Embedded low power comparator
12) Embedded TRNG (True Random Number Generator)
13) Operating temperature range: -40℃~+85℃
14) Supports 2.4GHz proprietary technology.

1.2.2 RF Features

RF features include:

1) 2.4GHz RF transceiver embedded, working in worldwide 2.4GHz ISM band.
2) 2.4GHz proprietary 1Mbps/2Mbps/250kbps/500kbps mode
    - ✧ Support Adaptive Frequency Hopping feature
    - ✧ Support flexible GFSK/FSK modulation index configuration
    - ✧ Support 1-N receiver capability
3) Rx Sensitivity: -96.5dBm@2.4GHz 1Mbps mode, -94dBm @ 2.4GHz 2Mbps mode, -99dBm @ 2.4GHz 500kbps mode, -101dBm @ 2.4GHz 250kbps mode
4) Tx output power: up to +10dBm.
5) Single-pin antenna interface.
6) RSSI monitoring with +/-1dB resolution.
7) Auto acknowledgement, retransmission and flow control.
8) Integrated load inductor.

1.2.3 Features of power management module

Features of power management module include:

1) Embedded LDO.
    - ✧ 1.8V bypass LDO
    - ✧ 1.4V bypass LDO
    - ✧ USB LDO with power supply of 4.5V~5.5V
2) Battery monitor: Supports low battery detection.
3) Power supply: 1.8V~3.6V. USB 4.5V ~ 5.5V
4) Multiple stage power management to minimize power consumption.
5) Low power consumption:

- ✧ Whole Chip RX mode: 10mA with LDO
- ✧ Whole Chip TX mode @ 0dBm: 9.5mA with LDO
- ✧ Deep sleep with external wakeup (without SRAM retention): 0.4uA
- ✧ Deep sleep with SRAM retention: 1uA (with 16kB SRAM retention), 1.4uA (with 32kB SRAM retention)

1.2.4 USB features

USB features include:

1) Compatible with USB2.0 Full speed mode.
2) Supports 9 endpoints including control endpoint 0 and 8 configurable data endpoints.
3) Independent power domain.
4) Supports ISP (In-System Programming) via USB port.

1.2.5 Flash features

The TLSR8355 embeds Flash with features below:

1) Total 64kbytes.
2) Flexible architecture: 4kB per Sector, 64kB/32kB per block.
3) Up to 256 Bytes per programmable page.
4) Write protect all or portions of memory.
5) Sector erase (4kB).
6) Block erase (32kB/64kB).
7) Cycle Endurance: 100,000 program/erases.
8) Data Retention: typical 20-year retention.
9) Multi firmware encryption methods for anti-cloning protection.

1.3 Typical applications

The TLSR8355 is dedicated to 2.4GHz RF System-On-Chip solution.

Its typical applications include, but are not limited to the following:

- ✧ Wireless mouse
- ✧ Wireless USB Dongle

1.4 Ordering information

Table 1- 1 Order information of the TLSR8355*[1]

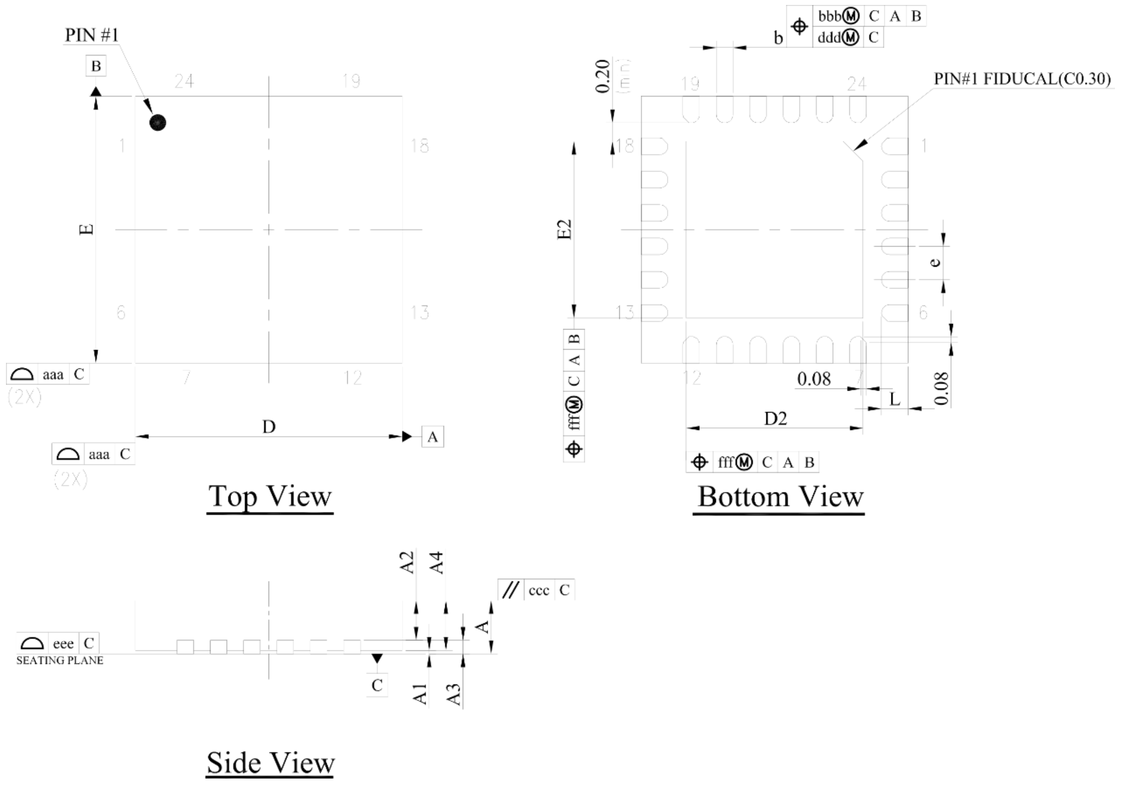
Product Series	Package Type	Temperature Range	Product Part No.	Packing Method *[2]	Minimum Order Quantity
TLSR8355F64	24-pin TQFN 4×4x0.75mm	-40℃~+85℃	TLSR8355F64 ET24	TR	3000

[1] MSL (Moisture Sensitivity Level): The 8355 series is applicable to MSL3 (Based on JEDEC Standard J-STD-020).
- ✧ After the packing opened, the product shall be stored at <30℃/ <60%RH and the product shall be used within 168 hours.
- ✧ When the color of the indicator in the packing changed, the product shall be baked before soldering.
- ✧ If baking is required, please refer to IPC/JEDEC J-STD-033 for baking procedure.

[2] Packing method "TR" means tape and reel. The tape and reel material DO NOT support baking under high temperature.

1.5 Package

Package dimensions for the TLSR8355F64ET24 is shown as below.



ITEM		Symbol	DIMENSION(mm)		
			MIN.	NOM.	MAX.
Total height		A	0.70	0.75	0.80
Stand off		A1	0	0.02	0.05
Mold thickness		A2	0.53	0.55	0.56
Leadframe thickness		A3	---	0.20REF	---
Mold+Leadframe thickness+Mold gap		A4	0.70	0.73	0.80
Lead width		b	0.18	0.25	0.30
Package size	X	D	3.90	4.00	4.10
	Y	E	3.90	4.00	4.10
E-PAD Size	X	D2	2.55	2.65	2.75
	Y	E2	2.55	2.65	2.75
Lead length		L	0.35	0.40	0.45
Lead pitch		e	---	0.50BSC	---
Package profile of a surface		aaa		0.15	
Lead position		bbb		0.10	
Paralleliam		ccc		0.10	
Lead position		ddd		0.05	
Package profile of a surface		eee		0.08	
Epad position		fff		0.10	

Figure 1- 2 Package dimension for TLSR8355F64ET24

1.6 Pin layout

1.6.1 Pin layout for TLSR8355F64ET24

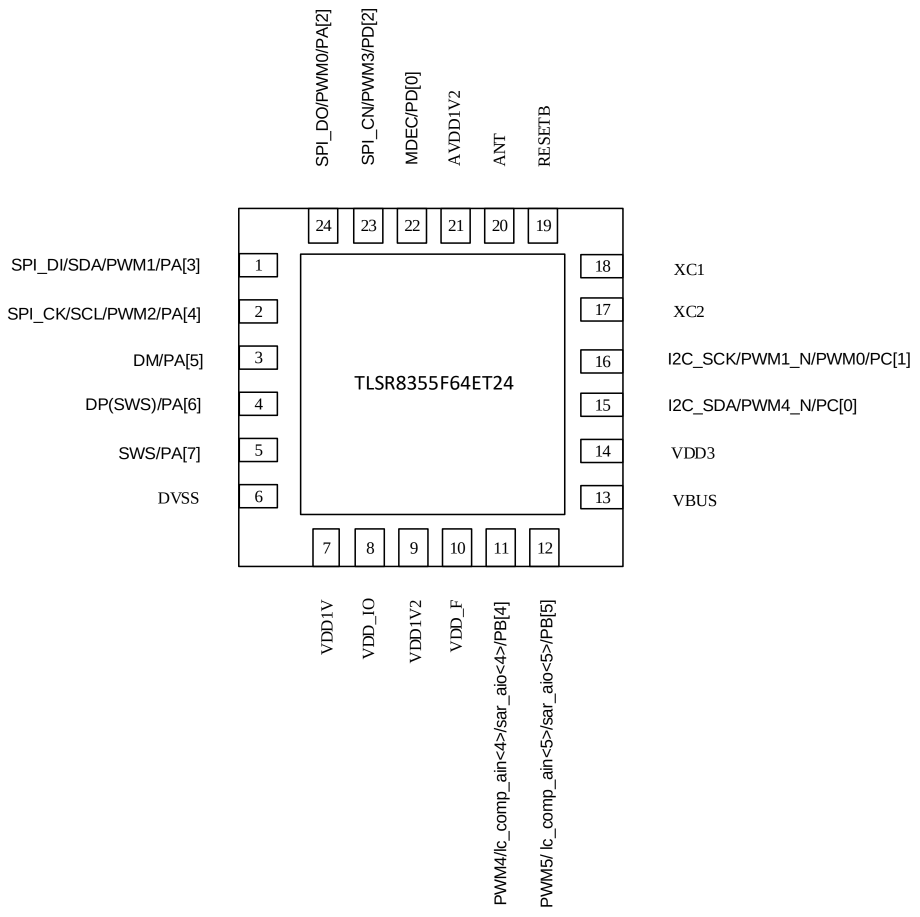


Figure 1- 3 Pin assignment for TLSR8355F64ET24

Table 1- 2 Pin functions for TLSR8355F64ET24

No.	Pin Name	Type	Description
1	SPI_DI/SDA/PWM1/PA[3]	Digital I/O	SPI data input (I2C_SDA) / PWM1 output / GPIO PA[3]
2	SPI_CK/SCL/PWM2/PA[4]	Digital I/O	SPI clock (I2C_SCK) / PWM2 output / GPIO PA[4]
3	DM/PA[5]	Digital I/O	USB data minus / GPIO PA[5]
4	DP(SWS)/PA[6]	Digital I/O	USB data positive (Single wire slave) / GPIO PA[6]
5	SWS/PA[7]	Digital I/O	Single wire slave/ GPIO PA[7]
6	DVSS	GND	Digital LDO ground
7	VDD1V	PWR	Internal LDO generated power supply input for digital core
8	VDD_IO	PWR	External 3.3V power supply input for IO
9	VDD1V2	PWR	Connect to GND via external capacitor. Route this 1.2V voltage power supply to AVDD1V2.
10	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor.
11	PWM4/lc_comp_ain<4>/ sar_aio<4>/PB[4]	Digital I/O	PWM4 output / Low power comparator input / SAR ADC input / GPIO PB[4]
12	PWM5/lc_comp_ain<5>/ sar_aio<5>/PB[5]	Digital I/O	PWM5 output / Low power comparator input / SAR ADC input / GPIO PB[5]
13	VBUS	PWR	USB 5V supply
14	VDD3	PWR	Connect to an external 3.3V power supply
15	I2C_SDA/PWM4_N/PC[0]	Digital I/O	I2C serial data / PWM4 inverting output / GPIO PC[0]
16	I2C_SCK/PWM1_N/PWM0/ PC[1]	Digital I/O	I2C serial clock / PWM1 inverting output / PWM0 output / GPIO PC[1]
17	XC2	Analog	Connect 24MHz crystal
18	XC1	Analog	Connect 24MHz crystal
19	RESETB	RESET	Power on reset, active low
20	ANT	Analog	RF antenna
21	AVDD1V2	PWR	Power supply input for internal RF Modules. Route from VDD1V2. Connect to GND via external capacitor.
22	MDEC/PD[0]	Digital I/O	Manchester decoder input / GPIO PD[0]
23	SPI_CN/PWM3/PD[2]	Digital I/O	SPI chip select (Active low) /PWM3 output / GPIO PD[2]
24	SPI_DO/PWM0/PA[2]	Digital I/O	SPI data output / PWM0 output / GPIO PA[2]

1.6.2 Notes

1) All digital IOs (including PA[2:7], PB[4:5], PC[0:1], PD[0,2]) can be used as GPIOs and have configurable pull-up/pull-down resistor.

2) SPI:

   ✧ PA[3]: SPI_DI, PA[4]: SPI_CK, PD[2]: SPI_CN, PA[2]: SPI_DO

3) I2C:

   ✧ PC[0] ~ PC[1] can be used as I2C. PC[0]: I2C_SDA, PC[1]: I2C_SCK

   ✧ I2C can also be multiplexed with SPI interface, i.e. I2C_SDA and I2C_SCK (SCL) can be multiplexed with PA[3]/SPI_DI and PA[4]/SPI_CK respectively.

4) USB:

   ✧ PA[5]: DM, PA[6]: DP

5) Single Wire debug interface:

   ✧ PA[7]: SWS

   ✧ PA[6]: SWS can also be multiplexed with DP.

6) Low power comparator input: PB[4]~ PB[5]. Please refer to section **13 Low Power Comparator**.

7) ADC GPIO input: PB[4]~ PB[5].

8) For register configuration to select pin multiplexed function, please refer to section **7.1.1.2 Multiplexed functions**.

9) For 24MHz crystal, the load capacitor range supported by design is 7.33pF~12.66pF. If the crystal needs load capacitor of 15pF, two external capacitors will be required.

10) Pin drive strength:

   ✧ PA[2:4], PC[0:1] and PD[0, 2] support drive strength up to 4mA (4mA when “DS”=1, 2mA when “DS”=0);

   ✧ PA[5:7] support drive strength up to 8mA (8mA when “DS”=1, 4mA when “DS”=0);

   ✧ PB[4:5] support drive strength up to 16mA (16mA when “DS”=1, 12mA when “DS”=0).

   ✧ “DS” configuration will take effect when the pin is used as output. Please refer to section **7.1.1 Basic configuration** for the corresponding “DS” register address and the default setting.

2 Memory and MCU

2.1 Memory

The TLSR8355 embeds 32kB SRAM with retention in deep sleep as data memory, and 64kB internal FLASH as program memory.

2.1.1 SRAM/Register

SRAM/Register memory map is shown as follows:

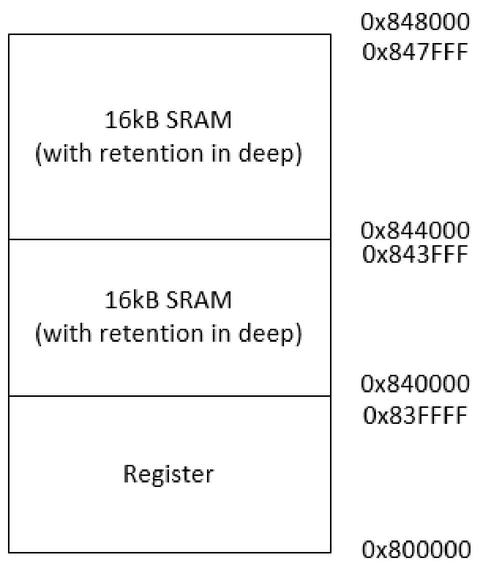


Figure 2- 1 Physical memory map

Register address: 0x800000 ~ 0x83FFFF.

Address for two independent 16kB SRAMs with retention in deep sleep: 0x840000 ~ 0x843FFF, 0x844000 ~ 0x847FFF.

Both register and SRAM address can be accessed (read or write) via debugging interface (SWS, SPI/I2C/USB interface).

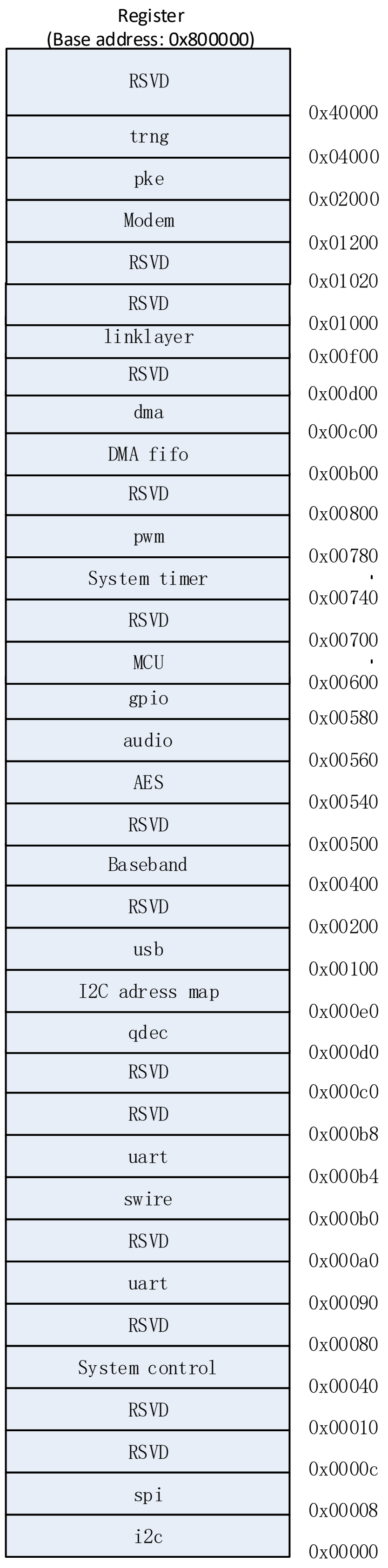


Figure 2- 2 Register space

2.1.2 Flash

The internal Flash mainly supports page program, sector/block/chip erase operations, and deep power down operation. Please refer to the corresponding SDK for Flash memory operation details.

For chip identification and traceability, the Flash is preloaded with Unique ID (UID). User is not allowed to modify this preloaded UID, but can read the UID via corresponding API interface.

MCU uses the system frequency to load instructions, and adopts flash driver to access (read/write) flash with the speed of half of the system clock.

2.1.3 E-Fuse

The non-volatile E-Fuse section is preloaded with 4-byte decryption key and 4-byte chip UID, as shown below.

Table 2- 1 E-Fuse information

E-Fuse bit information	Decryption key	Chip UID			
		Internal information	Wafer No.	Lot No.	Internal information
	Bit0~31	Bit32~47	Bit48~52	Bit53~55	Bit56~63

2.2 Firmware encryption

The TLSR8355 supports multiple firmware encryption methods to achieve the anti-cloning protection, including:

- UID-based authentication code generation method

  During firmware burning (e.g. via specific burning jig), user can use customized key and AES encryption algorithm to encrypt the UID read from the chip flash, generate unique ciphertext and write the ciphertext into E-Fuse section.

  During application, an encryption authentication procedure is added. User should use the same key and AES encryption algorithm to encrypt the UID read from the chip flash, and generate new ciphertext. Before running main application firmware, the new ciphertext will be compared with the ciphertext read from the E-Fuse section. Only when the authentication passes, i.e. the comparison result matches, the main firmware will be up and running, otherwise the chip will stop running the main firmware.

- Bootloader-based firmware encryption/decryption

  The firmware can be encrypted using a customer-provided security key. The customer security key is written into a specific secure register, and becomes unreadable. Any attempt to read the key will only result in either all 1's or all 0's.

  The encrypted firmware can be generated based on the plaintext firmware and the customer security key. The customer can burn the security key into the obscured memory area and also the encrypted firmware into Flash.

  The firmware is readable by all, but appears as garbled binaries to 3rd party.

2.3 MCU

The TLSR8355 integrates a powerful 32-bit MCU developed by Telink. The digital core is based on 32-bit RISC, and the length of instructions is 16 bits; four hardware breakpoints are supported.

2.4 Working modes

The TLSR8355 supports six working modes, including Active, Idle, Suspend, Deep sleep with SRAM retention, deep sleep without SRAM retention, and Shutdown.

- ✧ The Power Management (PM) module is always active in all working modes.
- ✧ For modules such as MCU, RF transceiver (Radio), and SRAM, the state depends on working mode, as shown below.

Table 2- 2 Working modes

Mode	**Active**	**Idle**	**Suspend**	**Deep sleep with SRAM retention**	**Deep sleep without SRAM retention**	**Shutdown**
MCU	active	stall	stall	off	off	off
Radio	available	available	off	off	off	off
USB	available	available	off	off	off	off
Audio	available	available	off	off	off	off
Wakeup time to Active mode	——	0us	100us	Shorter than deep sleep without retention, almost same as Suspend	1ms	10ms
(16k+16k) retention SRAMs (with retention in deep sleep)	full	full	full	full	off	off
Wakeup on RTC (32K Timer wakeup)	——	——	available	available	available	off
Wakeup on pin (IO wakeup)	——	——	available	available	available	off
Wakeup on interrupt	——	available	——	——	——	——
Wakeup on reset pin (RESETB)	——	available	available	available	available	on
Current	Please refer to section **15.3 DC characteristics**.					

***Notes:**

1) "active": MCU is at working state.
2) "stall": In Idle and Suspend mode, MCU does not work, while its clock is still running.
3) "available" for Modules: It's selectable to be at working state, or stall/be powered down if it does not need to work.
4) "available"/"on" for wakeup: Corresponding wakeup method is supported.
5) "off" for wakeup: Corresponding wakeup method is not supported.
6) "full"/"off" for SRAMs:

- ✧ "full": Full speed. In Active, Idle and Suspend mode, the two 16kB retention SRAMs are powered on and work normally (can be accessed); in Deep sleep with SRAM retention, the retention SRAMs are powered on, however, the contents of the retention SRAMs can be retained and cannot be accessed.
- ✧ "off": The retention SRAMs are powered down in Deep sleep without SRAM retention and Shutdown mode.

7) Current:

- ✧ In Deep sleep without SRAM retention, only the PM module is active, all digital and analog modules are powered down, thus the power consumption is largely decreased.
- ✧ In Deep sleep with SRAM retention, the PM module is active, all analog and digital modules except for the retention SRAMs are powered down, thus the power consumption is a little higher than in Deep sleep without SRAM retention, but much lower than in Suspend.

Table 2- 3 Retention analog registers in deep sleep

Address	R/W	Description	Reset Value
afe_0x35	R/W	buffer, clean at watch dog reset	0x20
afe_0x36	R/W	buffer, clean at watch dog reset	0x00
afe_0x37	R/W	buffer, clean at watch dog reset	0x00
afe_0x38	R/W	buffer, clean at watch dog reset	0x00
afe_0x39	R/W	buffer, clean at watch dog reset	0xff
afe_0x3a	R/W	buffer, clean at power on reset	0x00
afe_0x3b	R/W	buffer, clean at power on reset	0x00
afe_0x3c	R/W	buffer, clean at power on reset	0x0f

Analog registers (0x35 ~ 0x3c) as shown in Table 2- 3 are retained in deep sleep mode and can be used to store program state information across deep sleep cycles.

- ✧ Analog registers 0x3a~0x3c are non-volatile even when chip enters deep sleep or chip is reset by watchdog or software, i.e. the contents of these registers won't be changed by deep sleep or watchdog reset or chip software reset.
- ✧ Analog registers 0x35~0x39 are non-volatile in deep sleep, but will be cleared by watchdog reset or chip software reset.
- ✧ After POR (Power-On-Reset), all registers will be cleared to their default values, including these analog registers.

User can set flag in these analog registers correspondingly, so as to check the booting source by reading the flag.

For chip software reset, please refer to section **2.5 Reset**.

2.5 Reset

The chip supports three types of reset methods, including POR (Power-On-Reset), watchdog reset and software reset.

1) POR: After power on, the whole chip will be reset, and all registers will be cleared to their default values.

2) Watchdog reset: A programmable watchdog is supported to monitor the system. If watchdog reset is triggered, registers except for the retention analog registers 0x3a~0x3c will be cleared.

3) Software reset: It is also feasible to carry out software reset for the whole chip or some modules.

   ✧ Setting address 0x6f[5] as 1b'1 is to reset the whole chip. Similar to watchdog reset, the retention analog registers 0x3a~0x3c are non-volatile, while other registers including 0x35~0x39 will be cleared by chip software reset.

   ✧ Addresses 0x60~0x62 serve to reset individual modules: if some bit is set to logic "1", the corresponding module is reset.

Table 2- 4 Register configuration for software reset

Address	Mnemonic	Type	Description	Reset Value
0x60	RST0	R/W	Reset control, 1 for reset, 0 for clear [0]: SPI [1]: I2C [2]: RSVD (RS232, i.e. UART) [3]: USB [4]: PWM [5]: QDEC [6]: IR_LEARN [7]: Swire	0x7c
0x61	RST1	R/W	[0] ZB (i.e. Baseband) [1] System Timer [2] DMA [3] ALGM [4] AES [5] ADC [6] ALG [7] PKE (Public-Key Engine)	0xff
0x62	RST2	R/W	[0] AIF [1] RSVD (Audio) [2] DFIFO [3] TRNG (True Random Number Generator) [4] RISC [5] MCIC [6] RISC1 (R) [7] MCIC1 (R)	0xc7
0x6f	PWDNEN	R/W	[0]: suspend enable (RW) [4]: clear ramcrc enable (w1c) [5]: reset all (act as watchdog reset) [6]: rsvd (mcu low power mode) (W) [7]: stall mcu trig If bit[0] set 1, then system will go to suspend. Or only stall mcu (W)	0x00

2.6 Power Management

The multiple-stage Power Management (PM) module is flexible to control power state of the whole chip or individual functional blocks such as MCU, RF Transceiver, and peripherals.

2.6.1 Power-On-Reset (POR) and Brown-out detect

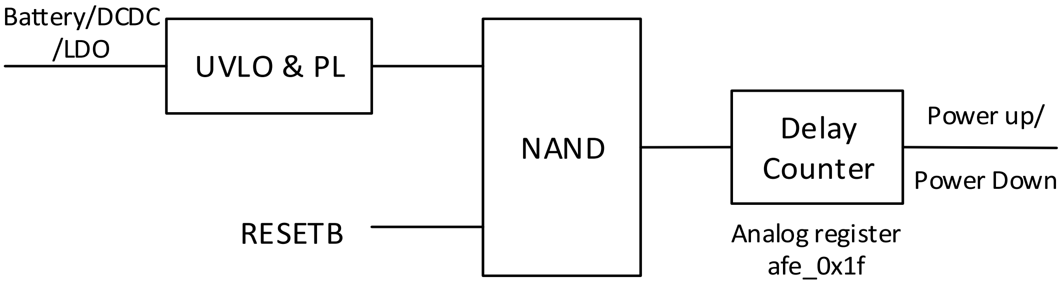


Figure 2- 3 Control logic for power up/down

The whole chip power up and down is controlled by the UVLO (Ultra-low Voltage Lockout) & PL (Power Logic) module and the external RESETB pin via the logic shown in the above diagram. UVLO takes the external power supply as input and releases the lock only when the power supply voltage is higher than a preset threshold. The RESETB pin has an internal pull-up resistor; an external Cap can be connected on the RESETB pin to control the POR delay.

After both UVLO and RESETB release, there is a further configurable delay before the system reset signal (“Sysrst”) is released. The delay is adjusted by analog register afe_0x1f. Since the content of afe_0x1f is reset to default only after power cycle, watchdog reset, or software reset, the delay change using afe_0x1f is only applicable when the chip has not gone through these reset conditions. For example, after deep sleep wakeup, the setting in afe_0x1f will take effect.

Table 2- 5 Analog register to control delay counters

Address	Name	Type	Description	Default
afe_0x1f	r_dly	R/W	wait for DCDC ready (based on 16kHz count decrement counter)	0x80

Initial Power up

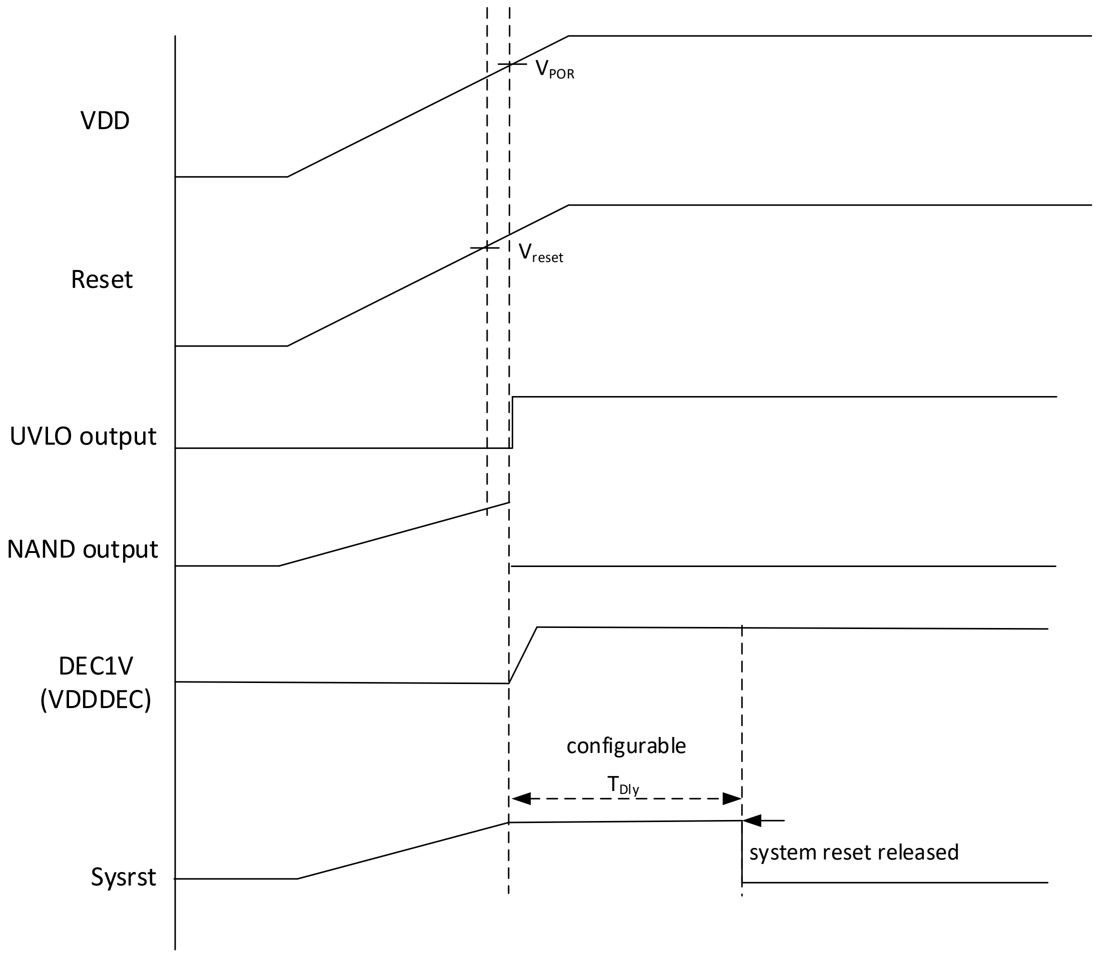


Figure 2- 4 Initial Power-up sequence

Power down

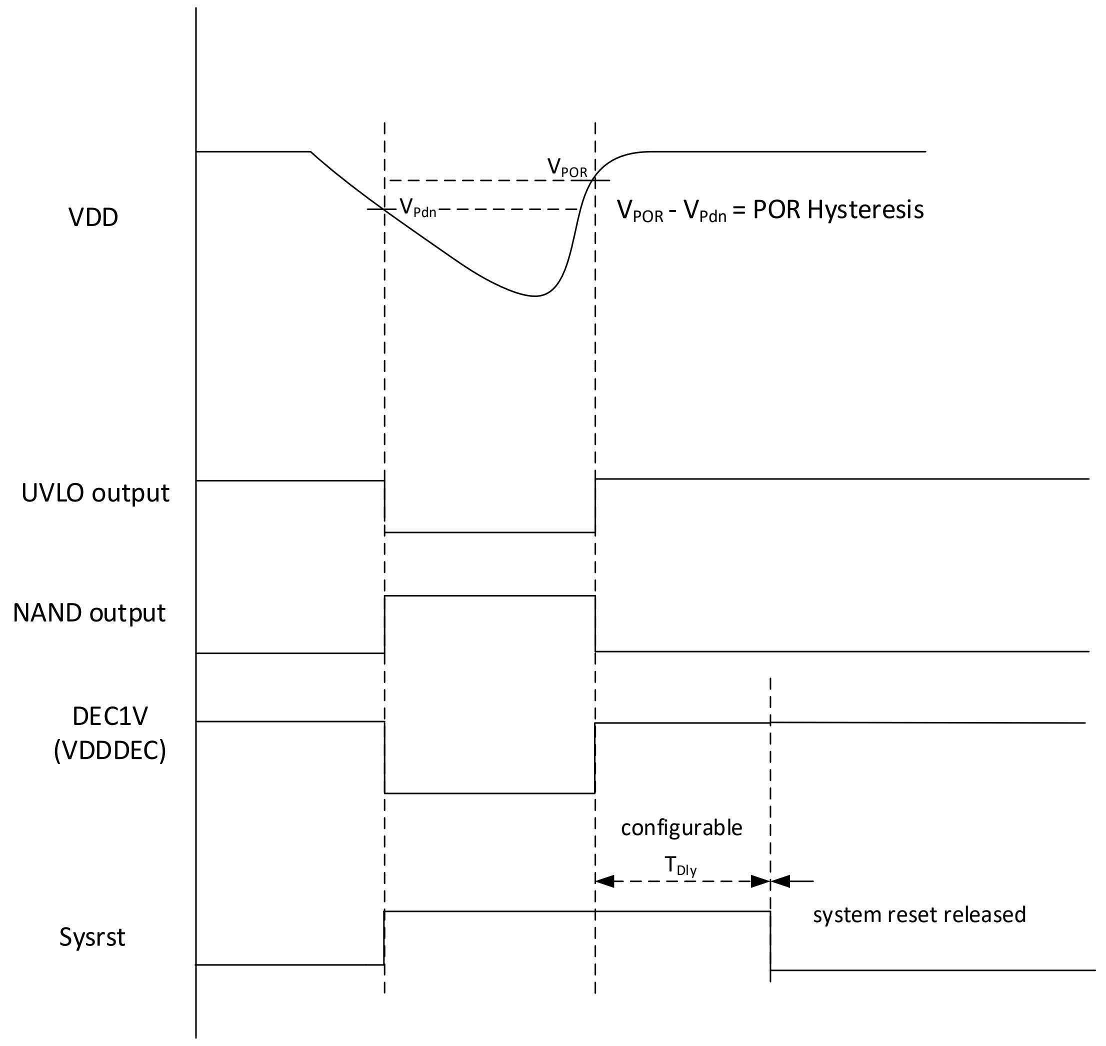


Figure 2- 5 Power-down sequence

Table 2- 6 Characteristics of Initial Power-up/ Power-down sequence

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{POR}	VDD voltage when V_{UVLO} turns to high level		1.62		V
V_{Pdn}	VDD voltage when V_{UVLO} turns to low level		1.55		V
T_{Dly}	Delay counter value	Configurable via analog register afe_0x1f			

2.6.2 Working mode switch

In Active mode, MCU is active, all SRAMs are accessible, and other modules are selectable whether to be at working state.

The chip can switch to Idle mode to stall the MCU. In this mode, all SRAMs are still accessible, modules such as RF transceiver, Audio and USB are still selectable whether to be at working state. The chip can be triggered to Active mode by interrupt or RESETB pin, and the time to switch to Active mode is negligible.

To decrease power consumption to different levels, the chip can switch to power saving mode (Suspend, Deep sleep with SRAM retention, Deep sleep without SRAM retention, Shutdown) correspondingly. (Please refer to Table 2-1.)

- ✧ In Suspend mode, MCU stalls, all SRAMs are still accessible, the PM module is active, modules such as RF transceiver, Audio and USB are powered down. The chip can be triggered to Active mode by 32K Timer, IO pin or RESETB pin. It takes 100us or so to switch from Suspend mode to Active mode.
- ✧ In Deep sleep with SRAM retention, the PM module is active, analog and digital modules except for the two 8kB and one 16kB retention SRAMs are powered down, while the retention SRAMs can be retained and not accessible. The chip can be triggered to Active mode by 32K Timer, IO pin or RESETB pin. The time to switch to Active mode is shorter than Deep sleep without SRAM retention and close to Suspend.
- ✧ In Deep sleep without SRAM retention, only the PM module is active, while analog and digital modules including the retention SRAMs are powered down. The chip can be triggered to Active mode by 32K Timer, IO pin or RESETB pin. The time to switch to Active mode is 1ms or so.
- ✧ In Shutdown mode, all digital and analog modules are powered down, and only the PM module is active. The chip can be triggered to Active mode by RESETB pin only. The time to switch to Active mode is 10ms or so.

User can directly invoke corresponding library function to switch working mode of the chip.

If certain module doesn't need to work, user can power down this module in order to save power.

2.6.3 LDO and DCDC

The chip embedded DCDC can generate 1.8V output voltage and supply power for the internal flash; the DCDC can also generate 1.4V output voltage.

The embedded LDO regulator takes the 1.4V voltage output from the DCDC, and generates 1.2V regulated voltage to supply power for 1.2V digital core and analog modules in Active/Idle/Suspend mode.

2.6.4 VBAT and VANT power-supply mode

The chip provides two power-supply modes including VBAT mode and VANT mode.

- ✧ In VBAT mode, the chip is directly supplied with power by its battery voltage. The maximum output power is related to power supply voltage, for example, the maximum power is 10dBm or so at 3.3V power supply.

✧ In VANT mode, the chip is supplied with 1.2V voltage by the embedded DCDC and LDO. In this mode, output power won't change with AVDD basically, and the maximum power is 5dBm or so. Corresponding to the VBAT mode, the VANT mode is more power-saving at the same Tx power.

2.7 Wakeup sources

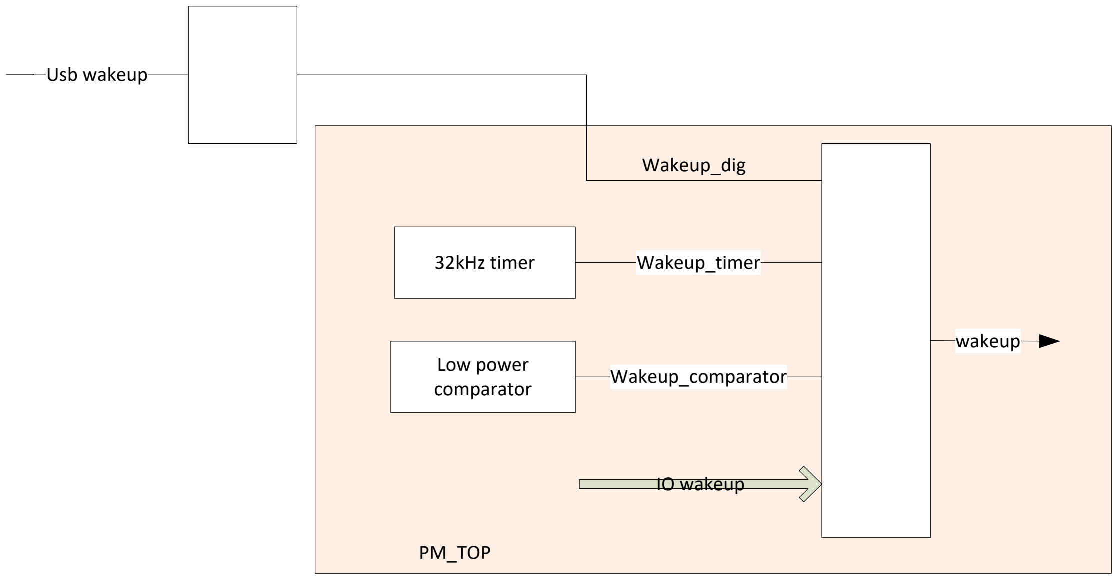


Figure 2- 6 Wakeup sources

2.7.1 Wakeup source - USB

This wakeup source can only wake up the system from suspend mode.

First, set the digital register 0x6e bit[2] as 1b'1.

To activate this mode, analog register afe_0x26[4] should also be set as 1b'1.

Once USB host sends out resuming signal, the system will be woke up.

2.7.2 Wakeup source – 32kHz timer

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

To enable the wakeup source from 32kHz timer, analog register afe_0x26[4] should be set as 1b'1.

2.7.3 Wakeup source – low power comparator

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

To enable the wakeup source from low power comparator, analog register 0x26[5] should be set as 1b’1. The low power comparator wakeup is active high.

2.7.4 Wakeup source – IO

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes. And IO wakeup supports high level or low level wakeup which is configurable via polarity control registers.

Analog register afe_0x26[3] should be set as 1b’1 to enable IO wakeup source.

Enabling control analog registers: PA[7:0] enabling control register is afe_0x27[7:0], PB[7:0] enabling control register is afe_0x28[7:0], PC[7:0] enabling control register is afe_0x29[7:0], and PD[7:0] enabling control register is afe_0x2a[7:0]. Total wakeup pin can be up to 32.

Polarity control registers: PA[7:0] polarity control register is afe_0x21[7:0], PB[7:0] polarity control register is afe_0x22[7:0], PC[7:0] polarity control register is afe_0x23[7:0], and PD[7:0] polarity control register is afe_0x24[7:0].

The corresponding driver is available so that user can directly invoke it to use IO wakeup source.

Analog register 0x44[3:0] indicates the wakeup source which triggers system wakeup. After wakeup, the corresponding wakeup status will be set as 1b’1 automatically, and it’s needed to write 1 to manually clean the status.

2.7.5 Wakeup source – MDEC

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

To enable the wakeup source from Manchester Decoder, analog register 0x26[7] should be set as 1b’1.

2.7.6 Register table

Table 2- 7 Analog registers for Wakeup

Address	Name	Type	Description	Default
afe_0x21	PA_polarity	R/W	Polarity control registers for IO wakeup 0: high level wakeup, 1: low level wakeup	0x00
afe_0x22	PB_polarity	R/W		0x00
afe_0x23	PC_polarity	R/W		0x00
afe_0x24	PD_polarity	R/W		0x00
afe_0x26	wkup_en	R/W	[7] MDEC wakeup enable	0x00
			[6] low power comparator wakeup enable	
			[5] 32kHz timer wakeup enable	
			[4] digital core (USB) wakeup enable	
			[3] IO (pad) wake up enable	
			[2] Enable/Mask filter for IO (Pad) wakeup 1: Select 16us filter to filter out jitter on IO PAD input.	

Address	Name	Type	Description	Default
			0: IO Pad combinational logic output (disable filter)	
afe_0x27	PA wake up enable	R/W	Enabling control registers for IO wakeup	0x00
afe_0x28	PB wake up enable	R/W		0x00
afe_0x29	PC wake up enable	R/W		0x00
afe_0x2a	PD wake up enable	R/W		0x00
afe_0x44	status	R	[7] rsvd	
			[6] rsvd	
			[5] rsvd	
			[4] MDEC wakeup status	
			[3] IO (pad) wakeup status	
			[2] digital core (USB) wakeup status	
			[1] 32k timer wakeup status	
			[0] low power comparator wakeup status	

Table 2- 8 Digital register for Wakeup

Address	Mnemonic	Type	Description	Reset Value
0x6e	WAKEUPEN	R/W	Wakeup enable [0]: enable wakeup from I2C host [1]: enable wakeup from SPI host [2]: enable wakeup from USB [3]: enable wakeup from gpio [4]: enable wakeup from I2C synchronous interface System resume control [5]: enable GPIO remote wakeup [6]: if set to 1, system will issue USB resume signal on USB bus [7] sleep wakeup reset system enable	0x1f

3 2.4GHz RF Transceiver

3.1 Block diagram

The TLSR8355 integrates an advanced 2.4GHz RF transceiver. The RF transceiver works in the worldwide 2.4GHz ISM (Industrial Scientific Medical) band.

The transceiver consists of a fully integrated RF synthesizer, a Power Amplifier (PA), a Low Noise Amplifier (LNA), a TX filter, a RX filter, a TX DAC, an ADC, a modulator and a demodulator. The transceiver can be configured to work in Proprietary 1Mbps, 2Mbps, 250kbps and 500kbps mode.

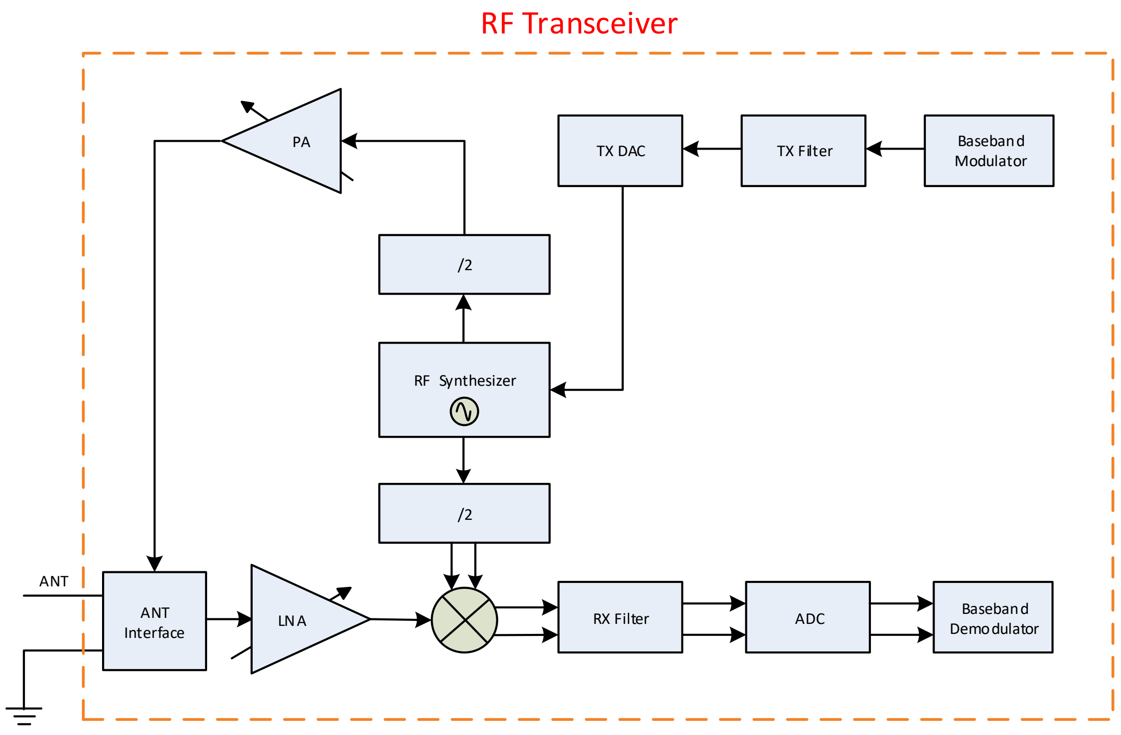


Figure 3- 1 Block diagram of RF transceiver

The internal PA can deliver a maximum 10dBm output power, avoiding the need for an external RF PA.

3.2 Air interface data rate and RF channel frequency

Air interface data rate, the modulated signaling rate for RF transceiver when transmitting and receiving data, is configurable via related register setting: 250kbps, 500kbps, 1Mbps, 2Mbps.

For the TLSR8355, RF transceiver can operate with frequency ranging from 2.400GHz to 2.4835GHz. The RF channel frequency setting determines the center of the channel.

3.3 Baseband

The baseband is disabled by default. The corresponding API is available for user to power on/down the baseband and enable/disable clock, so that the baseband can be turned on/off flexibly.

The baseband contains dedicated hardware logic to perform fast AGC control, access code correlation, CRC checking, data whitening, encryption/decryption and frequency hopping logic.

The baseband supports all features required by proprietary 2.4GHz specification.

3.3.1 Packet format

Packet format in 2.4GHz Proprietary mode is shown as Table 3- 2:

Table 3- 2 Packet format in Proprietary mode

LSB MSB

Preamble (8 bits)	Address code (configurable 3~5 bytes)	Packet Controller + Payload (1~33 bytes)	CRC (1~2 bytes)

3.3.2 RSSI and frequency offset

The TLSR8355 provides accurate RSSI (Receiver Signal Strength Indicator) and frequency offset indication.

- RSSI can be read from the 1byte at the tail of each received data packet.
- If no data packet is received (e.g. to perform channel energy measurement when no desired signal is present), real-time RSSI can also be read from specific registers which will be updated automatically.
- RSSI monitoring resolution can reach +/-1dB.
- Frequency offset can be read from the 2bytes at the tail of the data packet. Valid bits of actual frequency offset may be less than 16bits, and different valid bits correspond to different tolerance range.

Telink supplies corresponding drivers for user to read RSSI and frequency offset as needed.

4 Clock

4.1 Clock sources

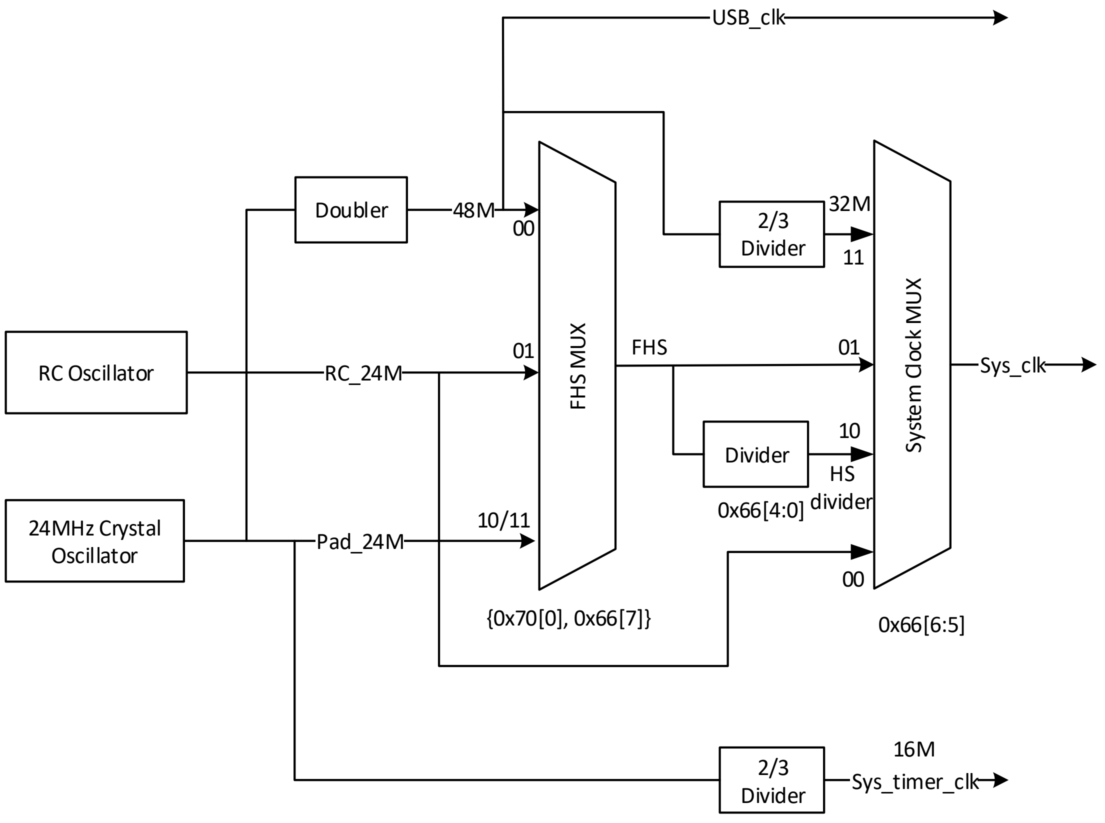


Figure 4- 1 Block diagram of clock

The TLSR8355 embeds a 24MHz RC oscillator which can be used as clock source for system, as well as a 32kHz RC oscillator to provide clock source for sleep state.

External 24MHz crystal is available via pin XC1 and XC2, which can provide a Pad_24MHz clock source for system and System Timer, and generate a 48M clock via a frequency doubler to provide clock source for USB.

4.2 System clock

There are four selectable clock sources for MCU system clock:

- ✧ **RC_24M** derived from 24MHz RC oscillator
- ✧ High speed clock "**FHS**"
- ✧ **HS divider clock** derived from "FHS" via a frequency divider
- ✧ **32MHz clock** derived from one 48MHz clock via a 2/3 frequency divider (The 48M clock is derived from 24MHz crystal oscillator via a frequency doubler).

The high speed clock (FHS) is selectable via address {0x70[0], 0x66[7]} from the following sources:

- ✧ **48MHz** clock derived from 24MHz crystal oscillator via a frequency doubler
- ✧ **RC_24M** derived from 24MHz RC oscillator
- ✧ **Pad_24M** derived from 24MHz crystal oscillator

The digital register CLKSEL (address 0x66) serves to set system clock: System clock source is selectable via bit[6:5].

- ✧ If address 0x66[6:5] is set to 2b'10 to select the HS divider clock, system clock frequency is adjustable via address 0x66[4:0]. The formula is shown as below:

  $F_{\text{System clock}} = F_{\text{FHS}}$ / (system clock divider value in address 0x66[4:0]).

  Note that address 0x66[4:0] should not be set as 0 or 1.

4.3 Module clock

Registers CLKEN0~CLKEN2 (address 0x63~0x65) are used to enable or disable clock for various modules. By disabling the clocks of unused modules, current consumption could be reduced.

4.3.1 System Timer clock

System Timer clock is derived from 24MHz crystal oscillator via a 2/3 frequency divider. The clock frequency is fixed as 16MHz.

4.3.2 USB clock

USB clock is derived from 48MHz clock. The 48MHz clock is derived from 24MHz crystal oscillator via a frequency doubler.

4.4 Register table

Table 4- 1 Register table related to clock

Address	Mnemonic	R/W	Description	Default
0x63	CLKEN0	R/W	Clock enable control: 1 for enable; 0 for disable [0]: SPI [1]: I2C [2]: RSVD (RS232, i.e. UART) [3]: USB [4]: PWM [5]: QDEC [6]: IR_LEARN [7]: Swire	0x83
0x64	CLKEN1	R/W	[0]: ZB (i.e. Baseband) [1]: System Timer [2]: DMA [3]: ALGM [4]: AES [5:6]: RSVD [7]: PKE (Public-Key Engine)	0x00
0x65	CLKEN2	R/W	[0]: AIF [1]: RSVD (Audio) [2]: DFIFO [3]: TRNG (True Random Number Generator) [4]: MC [5]: MCIC [6:7]: RSVD	0x30
0x66	CLKSEL	R/W	System clock select [4:0]: system clock divider (must exceed 1). If 0x66[6:5] is set as 2b'10, $F_{Sysclk} = F_{FHS}$ / (CLKSEL[4:0]). FHS: refer to 0x70 CLKSEH. [6:5]: select system clock source 2'b00: RC_24M from RC oscillator 2'b01: FHS 2'b10: HS divider (see 0x66[4:0]) 2'b11: 32M clock (48M * 2/3 divider) [7] FHS select (see 0x70[0])	0x06
0x67		R/W	RSVD	0x00
0x68		R/W	RSVD	0x02
0x6c		R/W	RSVD	0x01
0x6d		R/W	RSVD	0x02
0x70	CLKSEH	R/W	{0x70[0], 0x66[7]} FHS select 2'b00: 48M clock doubled from 24M crystal 2'b01: RC_24M from RC oscillator 2'b1x: Pad_24M from 24M crystal oscillator	0x00
0x73	SEL	R/W	[0]: clk32k select 0: select RC_32k from RC oscillator	0x04

Address	Mnemonic	R/W	Description	Default
			1: select Pad_32k from 32k crystal oscillator [1]: RSVD (MIC clock select 1: select 32k (see 0x73[0] to select 32k source) 0: rsvd (select MIC clk div))	

5 Timers

5.1 Timer0~Timer2

The TLSR8355 supports three timers: Timer0~ Timer2. The three timers all support four modes: Mode 0 (System Clock Mode), Mode 1 (GPIO Trigger Mode), Mode 2 (GPIO Pulse Width Mode) and Mode 3 (Tick Mode), which are selectable via the register TMR_CTRL0 (address 0x620) ~ TMR_CTRL1 (address 0x621).

Timer 2 can also be configured as "watchdog" to monitor firmware running.

5.1.1 Register table

Table 5- 1 Register configuration for Timer0~Timer2

Address	Mnemonic	Type	Description	Reset Value
0x72	Wd_status	R/W	[0] watch dog status: verify whether it is power reset (1'b0) or watch dog reset (1'b1), write 1 to clear.	00
0x620	TMR_CTRL0	RW	[0]Timer0 enable [2:1] Timer0 mode. 0: using sclk, 1: using gpio, 2: count width of gpi, 3: tick [3]Timer1 enable [5:4] Timer1 mode. [6]Timer2 enable [7]Bit of timer2 mode	00
0x621	TMR_CTRL1	RW	[0]Bit of timer2 mode [7:1]Low bits of watch dog capture	00
0x622	TMR_CTRL2	RW	[6:0]High bits of watch dog capture. It is compared with [31:18] of timer2 ticker [7]watch dog capture	00
0x623	TMR_STATUS	RW	[0] timer0 status, write 1 to clear [1] timer1 status, write 1 to clear [2] timer2 status, write 1 to clear [3] watch dog status, write 1 to clear (If Watchdog is enabled, need to clear it periodically to avoid triggering watchdog reset)	00
0x624	TMR_CAPT0_0	RW	Byte 0 of timer0 capture	00
0x625	TMR_CAPT0_1	RW	Byte 1 of timer0 capture	00
0x626	TMR_CAPT0_2	RW	Byte 2 of timer0 capture	00
0x627	TMR_CAPT0_3	RW	Byte 3 of timer0 capture	00
0x628	TMR_CAPT1_0	RW	Byte 0 of timer1 capture	00
0x629	TMR_CAPT1_1	RW	Byte 1 of timer1 capture	00
0x62a	TMR_CAPT1_2	RW	Byte 2 of timer1 capture	00
0x62b	TMR_CAPT1_3	RW	Byte 3 of timer1 capture	00
0x62c	TMR_CAPT2_0	RW	Byte 0 of timer2 capture	00
0x62d	TMR_CAPT2_1	RW	Byte 1 of timer2 capture	00

Address	Mnemonic	Type	Description	Reset Value
0x62e	TMR_CAPT2_2	RW	Byte 2 of timer2 capture	00
0x62f	TMR_CAPT2_3	RW	Byte 3 of timer2 capture	00
0x630	TMR_TICK0_0	RW	Byte 0 of timer0 ticker	00
0x631	TMR_TICK0_1	RW	Byte 1 of timer0 ticker	00
0x632	TMR_TICK0_2	RW	Byte 2 of timer0 ticker	00
0x633	TMR_TICK0_3	RW	Byte 3 of timer0 ticker	00
0x634	TMR_TICK1_0	RW	Byte 0 of timer1 ticker	00
0x635	TMR_TICK1_1	RW	Byte 1 of timer1 ticker	00
0x636	TMR_TICK1_2	RW	Byte 2 of timer1 ticker	00
0x637	TMR_TICK1_3	RW	Byte 3 of timer1 ticker	00
0x638	TMR_TICK2_0	RW	Byte 0 of timer2 ticker	00
0x639	TMR_TICK2_1	RW	Byte 1 of timer2 ticker	00
0x63a	TMR_TICK2_2	RW	Byte 2 of timer2 ticker	00
0x63b	TMR_TICK2_3	RW	Byte 3 of timer2 ticker	00

5.1.2 Mode0 (System Clock Mode)

In Mode 0, system clock is employed as clock source.

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive edge of system clock from preset initial Tick value. Generally the initial Tick value is set to 0.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), an interrupt is generated, Timer stops counting and Timer status is updated.

Steps of setting Timer0 for Mode 0 is taken as an example.

1st: Set initial Tick value of Timer0

Set Initial value of Tick via registers TMR_TICK0_0~TMR_TICK0_3 (address 0x630~0x633). Address 0x630 is lowest byte and 0x633 is highest byte. It's recommended to clear initial Timer Tick value to 0.

2nd: Set Capture value of Timer0

Set registers TMR_CAPT0_0~TMR_CAPT0_3 (address 0x624~0x627). Address 0x624 is lowest byte and 0x627 is highest byte.

3rd: Set Timer0 to Mode 0 and enable Timer0

Set register TMR_CTRL0 (address 0x620) [2:1] to 2b'00 to select Mode 0; Meanwhile set address 0x620[0] to 1b'1 to enable Timer0. Timer0 starts counting upward, and Tick value is increased by 1 on each positive edge of system clock until it reaches Timer0 Capture value.

5.1.3 Mode1 (GPIO Trigger Mode)

In Mode 1, GPIO is employed as clock source. The "**m0**"/"**m1**"/"**m2**" register specifies the GPIO which generates counting signal for Timer0/Timer1/Timer2.

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive/negative (configurable) edge of GPIO from preset initial Tick value. Generally the initial Tick value is set to 0. The "**Polarity**" register specifies the GPIO edge when Timer Tick counting increases.

Note: Refer to **Section 7.1.2** for corresponding “**m0**”, “**m1**”, “**m2**” and “**Polarity**” register address.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), an interrupt is generated and timer stops counting.

Steps of setting Timer1 for Mode 1 is taken as an example.

1st: Set initial Tick value of Timer1

Set Initial value of Tick via registers TMR_TICK1_0~TMR_TICK1_3 (address 0x634~0x637). Address 0x634 is lowest byte and 0x637 is highest byte. It’s recommended to clear initial Timer Tick value to 0.

2nd: Set Capture value of Timer1

Set registers TMR_CAPT1_0~TMR_CAPT1_3 (address 0x628~0x62b). Address 0x628 is lowest byte and 0x62b is highest byte.

3rd: Select GPIO source and edge for Timer1

Select certain GPIO to be the clock source via setting “m1” register.

Select positive edge or negative edge of GPIO input to trigger Timer1 Tick increment via setting “Polarity” register.

4th: Set Timer1 to Mode 1 and enable Timer1

Set address 0x620[5:4] to 2b’01 to select Mode 1; Meanwhile set address 0x620[3] to 1b’1 to enable Timer1. Timer1 starts counting upward, and Timer1 Tick value is increased by 1 on each positive/negative (specified during the 3rd step) edge of GPIO until it reaches Timer1 Capture value.

5.1.4 Mode2 (GPIO Pulse Width Mode)

In Mode 2, system clock is employed as the unit to measure the width of GPIO pulse. The “**m0**”/“**m1**”/“**m2**” register specifies the GPIO which generates control signal for Timer0/Timer1/Timer2.

After Timer is enabled, Timer Tick is triggered by a positive/negative (configurable) edge of GPIO pulse. Then Timer Tick (i.e. counting value) is increased by 1 on each positive edge of system clock from preset initial Tick value. Generally the initial Tick value is set to 0. The “**Polarity**” register specifies the GPIO edge when Timer Tick starts counting.

Note: Refer to **Section 7.1.2** for corresponding “**m0**”, “**m1**”, “**m2**” and “**Polarity**” register address.

While a negative/positive edge of GPIO pulse is detected, an interrupt is generated and timer stops counting. The GPIO pulse width could be calculated in terms of tick count and period of system clock.

Steps of setting Timer2 for Mode 2 is taken as an example.

1st: Set initial Timer2 Tick value

Set Initial value of Tick via registers TMR_TICK2_0~TMR_TICK2_3 (address 0x638~0x63b). Address 0x638 is lowest byte and 0x63b is highest byte. It’s recommended to clear initial Timer Tick value to 0.

2nd: Select GPIO source and edge for Timer2

Select certain GPIO to be the clock source via setting “m2” register.

Select positive edge or negative edge of GPIO input to trigger Timer2 counting start via setting “Polarity” register.

3rd: Set Timer2 to Mode 2 and enable Timer2

Set address 0x620[7:6] to 2b’01 and address 0x621 [0] to 1b’1.

Timer2 Tick is triggered by a positive/negative (specified during the 2nd step) edge of GPIO pulse. Timer2 starts counting upward and Timer2 Tick value is increased by 1 on each positive edge of system clock.

While a negative/positive edge of GPIO pulse is detected, an interrupt is generated and Timer2 tick stops.

4th: Read current Timer2 Tick value to calculate GPIO pulse width

Read current Timer2 Tick value from address 0x638~0x63b.

Then GPIO pulse width is calculated as follows:

$$\text{GPIO pulse width} = \text{System clock period} * (\text{current Timer2 Tick} - \text{intial Timer2 Tick})$$

For initial Timer2 Tick value is set to the recommended value of 0, then:

$$\text{GPIO pulse width} = \text{System clock period} * \text{current Timer2 Tick}.$$

5.1.5 Mode3 (Tick Mode)

In Mode 3, system clock is employed.

After Timer is enabled, Timer Tick starts counting upward, and Timer Tick value is increased by 1 on each positive edge of system clock.

This mode could be used as time indicator. There will be no interrupt generated. Timer Tick keeps rolling from 0 to 0xffffffff. When Timer tick overflows, it returns to 0 and starts counting upward again.

Steps of setting Timer0 for Mode 3 is taken as an example.

1st: Set initial Tick value of Timer0

Set Initial value of Tick via address 0x630~0x633. Address 0x630 is lowest byte and address 0x633 is highest byte. It’s recommended to clear initial Timer Tick value to 0.

2nd: Set Timer0 to Mode 3 and enable Timer0

Set address 0x620[2:1] to 2b’11 to select Mode 3, meanwhile set address 0x620[0] to 1b’1 to enable Timer0. Timer0 Tick starts to roll.

3rd: Read current Timer0 Tick value

Current Timer0 Tick value can be read from address 0x630~0x633.

5.1.6 Watchdog

Programmable watchdog could reset chip from unexpected hang up or malfunction.

Only Timer2 supports Watchdog.

Timer2 Tick has 32bits. Watchdog Capture has only 14bits, which consists of TMR_CTRL2 (address 0x622) [6:0] as higher bits and TMR_CTRL1 (address 0x621) [7:1] as lower bits. Chip will be reset when the Timer2 Tick[31:18] matches Watch dog capture.

1st: Clear Timer2 Tick value

Clear registers TMR_TICK2_0 ~TMR_TICK2_3 (address 0x638~0x63b). Address 0x638 is lowest byte and 0x63b is highest byte.

2nd: Enable Timer2

Set register TMR_CTRL0 (address 0x620) [6] to 1b'1 to enable Timer2.

3rd: Set 14-bit Watchdog Capture value and enable Watchdog

Set address 0x622[6:0] as higher bits of watchdog capture and 0x621[7:1] as lower bits. Meanwhile set address 0x622[7] to 1b'1 to enable Watchdog.

Then Timer2 Tick starts counting upwards from 0.

If bits[31:18] of Timer2 Tick value read from address 0x638~0x63b reaches watchdog capture, the chip will be reset, and the status bit in address 0x72[0] will be set as 1b'1 automatically. User can read the watchdog status bit after chip reset to check if the reset source is watchdog, and needs to write 1b'1 to this bit to manually clear the flag.

5.2 32K LTIMER

The TLSR8355 also supports a low frequency (32kHz) LTIMER in suspend mode or deep sleep mode. This timer can be used as one kind of wakeup source.

5.3 System Timer

The TLSR8355 also supports a System Timer. As introduced in section **4.3.1 System Timer clock**, the clock frequency for System Timer is fixed as 16MHz irrespective of system clock.

In suspend mode, both System Timer and Timer0~Timer2 stop counting, and 32K Timer starts counting. When the chip restores to active mode, Timer0~Timer2 will continue counting from the number when they stops; In contrast, System Timer will continue counting from an adjusted number which is a sum of the number when it stops and an offset calculated from the counting value of 32K Timer during suspend mode.

Table 5- 2 Register table for System Timer

Address	Mnemonic	R/W	Function	Default Value
0x740	SYS_TIMER[7:0]	R/W		0x00
0x741	SYS_TIMER[15:8]	R/W		0x00
0x742	SYS_TIMER[23:16]	R/W		0x00
0x743	SYS_TIMER[31:24]	R/W	System timer counter, write to set initial value.	0x00

Address	Mnemonic	R/W	Function	Default Value
			This is the sys timer counter	
0x744	IRQ_LEV[[7:0]	R/W		0xf0
0x745	IRQ_LEV[15:8]	R/W		0x0f
0x746	IRQ_LEV[23:16]	R/W		0x0f
0x747	IRQ_LEV[31:24]	R/W	System timer counter pulse irq trig value	0x0e
0x749	CAL_IRQ	R	[0]: calibration latch result update irq	
0x74a	SYS_TIMER_CTRL	R/W	[7:4]: 32KHz clock calibration mode (cycles of 32K clock) 4’h0: 65536 (2048ms) 4’h1: 32768 (1024ms) 4’h2: 16384 (512ms) 4’h3: 8192 (256ms) 4’h4: 4096 (128ms) 4’h5: 2048 (64ms) 4’h6: 1024 (32ms) 4’h7: 512 (16ms) 4’h8: 256 (8ms) 4’h9: 128 (4ms) 4’ha: 64 (2ms) 4’hb: 32 (1ms) 4’hc: 16 (500us) 4’hd: 8 (250us) 4’he: 4 (125us) 4’hf: 2 (62.5us) [3]: calibration enable [2]: rsvd [1]: enable of system timer [0]: write/read mode of 32KHz timer 1’b1: write; 1’b0: read	0xc1
0x74b	SYS_TIMER_ST	R	[6]: read busy status [5:0]: rsvd	
0x74c	32K_TIMER_SET[7:0]	R/W		0x00
0x74d	32K_TIMER_SET[15:8]	R/W		0x00
0x74e	32K_TIMER_SET[23:16]	R/W		0x00
0x74f	32K_TIMER_SET[31:24]	R/W	32KHz Timer write value	0x00
0x750	32K_TIMER_READ[7:0]	R		
0x751	32K_TIMER_READ[15:8]	R		
0x752	32K_TIMER_READ[23:16]	R		
0x753	32K_TIMER_READ[	R	32KHz Timer read value	

Address	Mnemonic	R/W	Function	Default Value
	31:24]			
0x754	CAL_LATCH[7:0]	R		
0x755	CAL_LATCH[15:8]	R		
0x756	CAL_LATCH[23:16]	R		
0x757	CAL_LATCH[31:24]	R	32KHz clock calibration result (representing 16MHz clock cycle number)	

***Note:** The lower three bits of address 0x740 is invalid, therefore, the resolution should be 0.5us.

6 Interrupt System

6.1 Interrupt structure

The interrupt function is applied to manage dynamic program sequencing based on real-time events triggered by timers, pins and etc.

For the TLSR8355, there are 24 interrupt sources in all: 16 types are level-triggered interrupt sources (listed in address 0x640~0x641), and 8 types are edge-triggered interrupt sources (listed in address 0x642).

When CPU receives an interrupt request (IRQ) from certain interrupt source, it will determine whether to respond to the IRQ. If CPU decides to respond, it pauses current routine and starts to execute interrupt service subroutine. Program will jump to certain code address and execute IRQ handling commands. After finishing interrupt service subroutine, CPU returns to the breakpoint and continues to execute main function.

6.2 Register configuration

Table 6- 1 Register table for Interrupt system

Address	Mnemonic	Type	Description	Reset Value
0x640	MASK_0	RW	Byte 0 interrupt mask, level-triggered type {irq_mix, irq_uart, irq_dfifo, irq_dma, usb_pwdn, time2, time1, time0} [7] irq_mix, i.e. irq_host_cmd [6] rsvd (irq_uart) [5] irq_dfifo [4] irq_dma [3] usb_pwdn [2] time2 [1] time1 [0] time0	0x00
0x641	MASK_1	RW	Byte 1 interrupt mask, level-triggered type {irq_pke, irq_pwm, irq_zb_rt, irq_udc[4:0]} [7] irq_pke [6] irq_pwm [5] irq_zb_rt [4] irq_udc[4] [3] irq_udc[3] [2] irq_udc[2] [1] irq_udc[1] [0] irq_udc[0]	0x00
0x642	MASK_2	RW	Byte 2 interrupt mask, edge-triggered type {rsvd, gpio2risc[1:0], irq_stimer, pm_irq, irq_gpio, usb_reset, usb_250us} [7] rsvd [6] gpio2risc[1] [5] gpio2risc[0] [4] irq_stimer	0x00

Address	Mnemonic	Type	Description	Reset Value
			[3] pm_irq_tm [2] irq_gpio [1] usb_reset [0] usb_250us	
0x643	IRQMODE	RW	[0] interrupt enable [1] reserved (Multi-Address enable)	0x00
0x644	PRIO_0	RW	Byte 0 of priority 1: High priority; 0: Low priority	0x00
0x645	PRIO_1	RW	Byte 1 of priority	0x00
0x646	PRIO_2	RW	Byte 2 of priority	0x00
0x648	IRQSRC_0	R	Byte 0 of interrupt source	0x00
0x649	IRQSRC_1	R	Byte 1 of interrupt source	0x00
0x64a	IRQSRC_2	R	Byte 2 of interrupt source	0x00

6.2.1 Enable/Mask interrupt sources

Various interrupt sources could be enabled or masked by the registers MASK_0~MASK_2 (address 0x640~0x642).

Interrupt sources of level-triggered type:

- ✧ irq_mix (0x640[7]): I2C Slave mapping mode or SPI Slave interrupt (irq_host_cmd)
- ✧ irq_dfifo (0x640[5]): DFIFO interrupt
- ✧ irq_dma (0x640[4]): DMA interrupt
- ✧ usb_pwdn (0x640[3]): USB Host has sent power down signal
- ✧ time2, time1, timer0 (0x640[2]~0x640[0]): Timer2~Timer0 interrupt
- ✧ irq_pke (0x641[7]): PKE (Public-Key Engine) interrupt
- ✧ irq_pwm (0x641[6]): PWM interrupt
- ✧ irq_zb_rt (0x641[5]): Baseband interrupt
- ✧ irq_udc[4:0] (0x641[4:0]): USB device interrupt

Interrupt sources of edge-triggered type:

- ✧ gpio2risc[1:0] (0x642[6]~0x642[5]): gpio2risc[1]~gpio2risc[0] interrupt, please refer to section **7.1.2**.
- ✧ irq_stimer (0x642[4]): System timer interrupt
- ✧ pm_irq_tm (0x642[3]): 32kHz timer wakeup interrupt
- ✧ irq_gpio (0x642[2]): GPIO interrupt, please refer to section **7.1.2**.
- ✧ usb_reset (0x642[1]): USB Host has sent reset command.
- ✧ usb_250us (0x642[0]): USB has been in idle status for 250us.

6.2.2 Interrupt mode and priority

Interrupt mode is typically-used mode. Register IRQMODE (address 0x643)[0] should be set as 1b’1 to enable interrupt function.

IRQ tasks could be set as High or Low priority via the registers PRIO_0~PRIO_2 (address 0x644~0x646). When two or more interrupt sources assert interrupt requests at the same time, CPU will respond depending on respective interrupt priority levels. It’s recommended not to modify priority setting.

6.2.3 Interrupt source flag

Three bytes in the registers IRQSRC_0~IRQSRC_2 (address 0x648~0x64a) serve to indicate IRQ sources. Once IRQ occurs from certain source, the corresponding IRQ source flag will be set as “1”. User could identify IRQ source by reading address 0x648~0x64a.

When handling edge-triggered type interrupt, the corresponding IRQ source flag needs to be cleared via address 0x64a. Take the interrupt source usb_250us for example: First enable the interrupt source by setting address 0x642 bit[0] as 1b’1; then set address 0x643 bit[0] as 1b’1 to enable the interrupt. In interrupt handling function, 24-bit data is read from address 0x648~0x64a to check which IRQ source is valid; if data bit[16] is 1, it means the usb_250us IRQ source is valid. Clear this interrupt source by setting address 0x64a bit[0] as 1b’1.

As for level-type interrupt, IRQ interrupt source status needs to be cleared by setting corresponding module status register. Take Timer0 IRQ interrupt source for example: First enable the interrupt source by setting address 0x640 bit[0] as 1b’1; then set address 0x643 bit[0] as 1b’1 to enable the interrupt. In interrupt handling function, 24-bit data is read from address 0x648~0x64a to check which IRQ source is valid; if data bit[0] is 1, it means the Timer0 IRQ source is valid. Register TMR_STATUS (address 0x623) [0] should be written with 1b’1 to manually clear Timer0 status (refer to section **5.1.1 Register table**).

7 Interface

7.1 GPIO

The TLSR8355F64ET24 supports up to 12 GPIOs respectively. All digital IOs can be used as general purpose IOs.

All GPIOs (including PA[2:7], PB[4:5], PC[0:1], PD[0,2]) have configurable pull-up/pull-down resistor. Please refer to **section 7.1.3 Pull-up/Pull-down resistor** for details.

7.1.1 Basic configuration

7.1.1.1 GPIO lookup table

Table 7- 1GPIO lookup table 1

Pin	Default function	Pad Function Mux					GPIO Setting						
		Register =3	Register =2	Register=1	Register=0	Register	Input (R)	IE	OEN	Output	Polarity	DS	Act as GPIO
SPI_DO/ PWM0/ PA[2]	GPIO	-	PWM0	/	SPI_DO	0x5a8[5:4]	0x580[2]	0x581[2]	0x582[2]	0x583[2]	0x584[2]	0x585[2]	0x586[2]
SPI_DI/ SDA/ PWM1/ PA[3]	GPIO	-	PWM1	/	SPI_DI/SDA	0x5a8[7:6]	0x580[3]	0x581[3]	0x582[3]	0x583[3]	0x584[3]	0x585[3]	0x586[3]
SPI_CK/ SCL/ PWM2/ PA[4]	GPIO	-	PWM2	/	SPI_CK/SCL	0x5a9[1:0]	0x580[4]	0x581[4]	0x582[4]	0x583[4]	0x584[4]	0x585[4]	0x586[4]
DM/ PA[5]	GPIO	/	/	/	DM	0x5a9[3:2]	0x580[5]	0x581[5]	0x582[5]	0x583[5]	0x584[5]	0x585[5]	0x586[5]
DP(SWS)/ PA[6]	GPIO	/	/	/	DP(SWS)	0x5a9[5:4]	0x580[6]	0x581[6]	0x582[6]	0x583[6]	0x584[6]	0x585[6]	0x586[6]
SWS/ PA[7]	SWS	/	/	/	SWS	0x5a9[7:6]	0x580[7]	0x581[7]	0x582[7]	0x583[7]	0x584[7]	0x585[7]	0x586[7]
PWM4/ PB[4]	GPIO	-	/	PWM4	/	0x5ab[1:0]	0x588[4]	0x589[4]	0x58a[4]	0x58b[4]	0x58c[4]	0x58d[4]	0x58e[4]
PWM5/ PB[5]	GPIO	-	/	PWM5	/	0x5ab[3:2]	0x588[5]	0x589[5]	0x58a[5]	0x58b[5]	0x58c[5]	0x58d[5]	0x58e[5]
I2C_SDA/ PWM4_N/ PC[0]	GPIO	-	/	PWM4_N	I2C_SDA	0x5ac[1:0]	0x590[0]	afe_0xc0 [0]	0x592[0]	afe_0xc1 [0]	0x594[0]	afe_0xc2 [0]	0x596[0]
I2C_SCK/ PWM1_N/ PWM0/ PC[1]	GPIO	-	PWM0	PWM1_N	I2C_SCK	0x5ac[3:2]	0x590[1]	afe_0xc0 [1]	0x592[1]	afe_0xc1 [1]	0x594[1]	afe_0xc2 [1]	0x596[1]
MDEC/ PD[0]	GPIO	-	/	/	/	0x5ae[1:0]	0x598[0]	0x599[0]	0x59a[0]	0x59b[0]	0x59c[0]	0x59d[0]	0x59e[0]
SPI_CN/ PWM3/ PD[2]	SPI_CN	-	PWM3	/	SPI_CN	0x5ae[5:4]	0x598[2]	0x599[2]	0x59a[2]	0x59b[2]	0x59c[2]	0x59d[2]	0x59e[2]

Notes:

(1) IE: Input enable, high active. 1: enable input, 0: disable input.

(2) OEN: Output enable, low active. 0: enable output, 1: disable output.

(3) Register: Configure multiplexed functions in “Pad Function Mux” column.

(4) Output: configure GPO output.

(5) Input: read GPI input.

(6) DS: Drive strength. 1: maximum DS level (Default), 0: minimal DS level.

(7) Act as GPIO: enable (1) or disable (0) GPIO function.

(8) Polarity: see section 7.1.2 Connection relationship between GPIO and related modules.

(9) Priority: "Act as GPIO" has the highest priority. To configure as multiplexed function, disable GPIO function first.

(10) afe_0xc0, afe_0xc1 and afe_0xc2 marked in red color are analog registers; others are digital registers.

(11) For all unused GPIOs, corresponding "IE" must be set as 0.

(12) When SWS/PA<7> "IE" is set as 1, this pin must be fixed as pull-up/pull-down state (float state is not allowed).

(13) To use SAR ADC/low power comparator pin function, please refer to corresponding module sections.

7.1.1.2 Multiplexed functions

Each pin listed in Table 7-1 acts as the function in the "**Default Function**" column by default.

✧ PA[7] acts as SWS function by default.

✧ PD[2] acts as SPI_CN function by default.

✧ The other digital IOs act as GPIO function by default.

If a pin with multiplexed functions does not act as GPIO function by default, to use it as GPIO, first set the bit in "**Act as GPIO**" column as 1b'1. After GPIO function is enabled, if the pin is used as output, both the bits in "**IE**" and "**OEN**" columns should be set as 1b'0, then set the register value in the "**Output**" column; if the pin is used as input, both the bits in "**IE**" and "**OEN**" columns should be set as 1b'1, and the input data can be read from the register in the "**Input**" column.

To use a pin as certain multiplexed function (neither the default function nor GPIO function), first clear the bit in "**Act as GPIO**" column to disable GPIO function, and then configure "**Register**" in "Pad Function Mux" column to enable multiplexed function correspondingly.

Example 1: SPI_DO/PWM0/PA[2].

(1) The pin acts as GPIO function by default.

✧ If the pin is used as general output, both address 0x581[2] (IE) and 0x582[2] (OEN) should be set as 1b'0, then configure address 0x583[2] (Output).

✧ If the pin is used as general input, both address 0x581[2] (IE) and 0x582[2] (OEN) should be set as 1b'1, and the input data can be read from address 0x580[2] (Input).

(2) To use the pin as SPI_DO function, address 0x586[2] (Act as GPIO) should be set as 1b'0, and 0x5a8[5:4] (Register) should be set as 2b'00.

(3) To use the pin as PWM0 function, address 0x586[2] (Act as GPIO) should be set as 1b'0, and 0x5a8[5:4] (Register) should be set as 2b'10.

Example 2: SPI_CN/PWM3/PD[2].

(1) The pin acts as SPI_CN function by default.

(2) To use it as GPIO function, first set address 0x59e[2] (Act as GPIO) as 1b’1.

✧ If the pin is used as general output, both address 0x599[2] (IE) and 0x59a[2] (OEN) should be set as 1b’0, then configure address 0x59b[2] (Output).

✧ If the pin is used as general input, both address 0x599[2] (IE) and 0x59a[2] (OEN) should be set to 1b’1, and the input data can be read from address 0x598[2] (Input).

(3) To use it as PWM3 function, set address 0x59e[2] (Act as GPIO) as 1b’0, and set 0x5ae[5:4] (Register) to 2b’10.

I2C can also be multiplexed with SPI interface, i.e. I2C_SDA/I2C_SCK can be multiplexed with SPI_DI/SPI_CK respectively.

To select multiplexed SPI/I2C function, please follow the steps below:

1) Disable GPIO function by setting corresponding “Act as GPIO” as 1b’0.

2) Select SPI/I2C function by setting corresponding “Register”.

3) Address 0x5b6[5:4] serve to select SPI or I2C output.

4) Address 0x5b7[5:4,1:0] serve to select SPI input or I2C input.

Table 7- 2Select multiplexed SPI/I2C

Pin with multiplexed SPI/I2C	Act as GPIO	Register	SPI Input Select	I2C Input Select	SPI/I2C Output Select
SPI_DI/SDA/PWM1/PA[3]	0x586[3]=0 Disable GPIO	0x5a8[7:6]=0 Select SPI_DI (I2C_SDA)	5b7[0] 1: as SPI input. 0: not as SPI input.	5b7[4] 1: as I2C input. 0: not as I2C input.	0x5b6[4] 1: as SPI/I2C output 0: not as SPI/I2C output
SPI_CK/SCL/PWM2/PA[4]	0x586[4]=0 Disable GPIO	0x5a9[1:0]=0 Select SPI_CK (I2C_SCK)	5b7[1] 1: as SPI input. 0: not as SPI input.	5b7[5] 1: as I2C input. 0: not as I2C input.	0x5b6[5] 1: as SPI/I2C output 0: not as SPI/I2C output

7.1.1.3 Drive strength

The registers in the “DS” column are used to configure the corresponding pin’s driving strength: “1” indicates maximum drive level, while “0” indicates minimal drive level.

The “DS” configuration will take effect when the pin is used as output. It’s set as the strongest driving level by default. In actual applications, driving strength can be decreased to lower level if necessary.

✧ PA[2:4], PC[0:1] and PD[0, 2]: maximum=4mA (“DS”=1), minimum=2mA (“DS”=0)

✧ PA[5:7]: maximum=8mA (“DS”=1), minimum=4mA (“DS”=0)

✧ PB[4:5]: maximum=16mA (“DS”=1), minimum=12mA (“DS”=0)

7.1.2 Connection relationship between GPIO and related modules

GPIO can be used to generate GPIO interrupt signal for interrupt system, counting or control signal for Timer/Counter module, or GPIO2RISC interrupt signal for interrupt system.

For the "Exclusive Or (XOR)" operation result for input signal from any GPIO pin and respective "Polarity" value, on one hand, it takes "And" operation with "irq" and generates GPIO interrupt request signal; on the other hand, it takes "And" operation with "m0/m1/m2", and generates counting signal in Mode 1 or control signal in Mode 2 for Timer0/Timer1/Timer2, or generates GPIO2RISC[0]/GPIO2RISC[1] interrupt request signal.

GPIO interrupt request signal = | ((input ^ polarity) & irq);

Counting (Mode 1) or control (Mode 2) signal for Timer0 = | ((input ^ polarity) & m0);

Counting (Mode 1) or control (Mode 2) signal for Timer1 = | ((input ^ polarity) & m1);

Counting (Mode 1) or control (Mode 2) signal for Timer2 = | ((input ^ polarity) & m2);

GPIO2RISC[0] interrupt request signal = | ((input ^ polarity) & m0);

GPIO2RISC[1] interrupt request signal = | ((input ^ polarity) & m1).

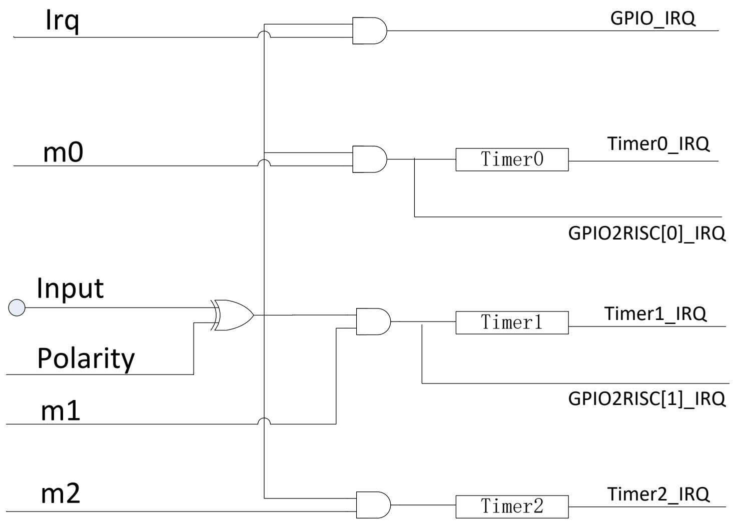


Figure 7- 1 Logic relationship between GPIO and related modules

Please refer to Table 7- 3 and Table 6- 1 to learn how to configure GPIO for interrupt system or Timer/Counter (Mode 1 or Mode 2).

(1) First enable GPIO function, enable IE and disable OEN. Please see section **7.1.1 Basic configuration** .

(2) GPIO IRQ signal:

Select GPIO interrupt trigger edge (positive edge or negative edge) via configuring "**Polarity**", and set corresponding GPIO interrupt enabling bit "**Irq**".

Then set address 0x5b5[3] (irq_enable) to enable GPIO IRQ.

Finally enable GPIO interrupt (irq_gpio) via address 0x642[2].

User can read addresses 0x5e0 ~ 0x5e3 to see which GPIO asserts GPIO interrupt request signal. Note: 0x5e0[7:0] --> PA<7>~PA<0>, 0x5e1[7:0] --> PB<7>~PB<0>, 0x5e2[7:0] --> PC<7>~PC<0>, 0x5e3[7:0] --> PD<7>~PD<0>.

(3) Timer/Counter counting or control signal:

Configure "**Polarity**". In Timer Mode 1, it determines GPIO edge when Timer Tick counting increases. In Timer Mode 2, it determines GPIO edge when Timer Tick starts counting.

Then set "**m0/m1/m2**" to specify the GPIO which generates counting signal (Mode 1)/control signal (Mode 2) for Timer0/Timer1/Timer2.

User can read addresses 0x5e8~0x5eb/0x5f0~0x5f3/0x5f8~0x5fb to see which GPIO asserts counting signal (in Mode 1) or control signal (in Mode 2) for Timer0/Timer1/Timer2. Note: Timer0: 0x5e8[7:0] --> PA<7>~PA<0>, 0x5e9[7:0] --> PB<7>~PB<0>, 0x5ea[7:0] --> PC<7>~PC<0>, 0x5eb[7:0] --> PD<7>~PD<0>; Timer1: 0x5f0[7:0] --> PA<7>~PA<0>, 0x5f1[7:0] --> PB<7>~PB<0>, 0x5f2[7:0] --> PC<7>~PC<0>, 0x5f3[7:0] --> PD<7>~PD<0>; Timer2: 0x5f8[7:0] --> PA<7>~PA<0>, 0x5f9[7:0] --> PB<7>~PB<0>, 0x5fa[7:0] --> PC<7>~PC<0>, 0x5fb[7:0] --> PD<7>~PD<0>.

(4) GPIO2RISC IRQ signal:

Select GPIO2RISC interrupt trigger edge (positive edge or negative edge) via configuring "**Polarity**", and set corresponding GPIO enabling bit **"m0"/"m1"**.

Enable GPIO2RISC[0]/GPIO2RISC[1] interrupt, i.e. "gpio2risc[0]" (address 0x642[5]) / "gpio2risc[1]"(address 0x642[6]).

Table 7- 3 GPIO lookup table2

Pin	**Input (R)**	**Polarity** 1: active low 0: active high	**Irq**	**m0**	**m1**	**m2**
PA[2]	0x580[2]	0x584[2]	0x587[2]	0x5b8[2]	0x5c0[2]	0x5c8[2]
PA[3]	0x580[3]	0x584[3]	0x587[3]	0x5b8[3]	0x5c0[3]	0x5c8[3]
PA[4]	0x580[4]	0x584[4]	0x587[4]	0x5b8[4]	0x5c0[4]	0x5c8[4]
PA[5]	0x580[5]	0x584[5]	0x587[5]	0x5b8[5]	0x5c0[5]	0x5c8[5]
PA[6]	0x580[6]	0x584[6]	0x587[6]	0x5b8[6]	0x5c0[6]	0x5c8[6]
PA[7]	0x580[7]	0x584[7]	0x587[7]	0x5b8[7]	0x5c0[7]	0x5c8[7]
PB[4]	0x588[4]	0x58c[4]	0x58f[4]	0x5b9[4]	0x5c1[4]	0x5c9[4]
PB[5]	0x588[5]	0x58c[5]	0x58f[5]	0x5b9[5]	0x5c1[5]	0x5c9[5]
PC[0]	0x590[0]	0x594[0]	0x597[0]	0x5ba[0]	0x5c2[0]	0x5ca[0]
PC[1]	0x590[1]	0x594[1]	0x597[1]	0x5ba[1]	0x5c2[1]	0x5ca[1]
PD[0]	0x598[0]	0x59c[0]	0x59f[0]	0x5bb[0]	0x5c3[0]	0x5cb[0]
PD[2]	0x598[2]	0x59c[2]	0x59f[2]	0x5bb[2]	0x5c3[2]	0x5cb[2]

7.1.3 Pull-up/Pull-down resistor

All GPIOs (including PA[0]~PD[7]) support configurable pull-up resistor of rank x1 and x100 or pull-down resistor of rank x10 which are all disabled by default. Analog registers afe_0x0e<7:0>~afe_0x15<7:0> serve to control the pull-up/pull-down resistor for each GPIO.

The DP pin also supports 1.5kΩ pull-up resistor for USB use. The 1.5kΩ pull up resistor is disabled by default and can be enabled by setting analog register afe_0x0b<7> as 1b'1. For the DP/PA[6] pin, user can only enable either 1.5kΩ pull-up resistor or pull-up resistor of rank x1/x100 / pull-down resistor of rank x10 at the same time. Please refer to Table 7- 4 for details.

Take the PA[3] for example: Setting analog register afe_0x0e<7:6> to 2b'01/2b'11/2b'10 is to respectively enable pull-up resistor of rank x100/pull-up resistor of rank x1/pull-down resistor of rank x10 for PA[3]; Clearing the two bits (default value) disables pull-up and pull-down resistor for PA[3].

Table 7- 4 Analog registers for pull-up/pull-down resistor control

Address	Mnemonic	Default	Description
afe_0x0b<7>	dp_pullup_res_3v	0	1.5k (Typ.) pull-up resistor for USB DP PAD 0: disable 1: enable
Rank x1 x10 x100	**Typical value (depend on actual application)** 18kohm 160kohm 1Mohm		
afe_0x0e<7:0>	a_sel<7:0>	00000000	PA[3:0] pull up and down select: <7:6>: PA[3] <5:4>: PA[2] <3:2>: rsvd (PA[1]) <1:0>: rsvd (PA[0]) 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x0f<7:0>	a_sel<15:8>	00000000	PA[7:4] pull up and down select: <7:6>: PA[7] <5:4>: PA[6] <3:2>: PA[5] <1:0>: PA[4] 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x10<7:0>	b_sel<7:0>	00000000	PB[3:0] pull up and down select: <7:6>: rsvd (PB[3]) <5:4>: rsvd (PB[2]) <3:2>: rsvd (PB[1]) <1:0>: rsvd (PB[0]) 00: Null 01: x100 pull up

Address	Mnemonic	Default	Description
			10: x10 pull down 11: x1 pull up
afe_0x11<7:0>	b_sel<15:8>	00000000	PB<7:4> pull up and down select: <7:6>: rsvd (PB[7]) <5:4>: rsvd (PB[6]) <3:2>: PB[5] <1:0>: PB[4] 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x12<7:0>	c_sel<7:0>	00000000	PC<3:0> pull up and down select: <7:6>: rsvd (PC[3]) <5:4>: rsvd (PC[2]) <3:2>: PC[1] <1:0>: PC[0] 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x13<7:0>	c_sel<15:8>	00000000	PC<7:4> pull up and down select: <7:6>: rsvd (PC[7]) <5:4>: rsvd (PC[6]) <3:2>: rsvd (PC[5]) <1:0>: rsvd (PC[4]) 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x14<7:0>	d_sel<7:0>	00000000	PD<3:0> pull up and down select: <7:6>: rsvd (PD[3]) <5:4>: PD[2] <3:2>: rsvd (PD[1]) <1:0>: PD[0] 00: Null 01: x100 pull up 10: x10 pull down 11: x1 pull up
afe_0x15<7:0>	d_sel<15:8>	00000000	PD<7:4> pull up and down select: <7:6>: rsvd (PD[7]) <5:4>: rsvd (PD[6]) <3:2>: rsvd (PD[5]) <1:0>: rsvd (PD[4]) 00: Null 01: x100 pull up

Address	Mnemonic	Default	Description
			10: x10 pull down 11: x1 pull up

7.2 Swire

The TLSR8355 supports Single Wire Slave interface. SWM (Single Wire Master) and SWS (Single Wire Slave) represent the master and slave device of the single wire communication system developed by Telink. The maximum data rate can be up to 2Mbps.

SWS usage is not supported in power-saving mode (deep sleep or suspend).

7.3 I2C

The TLSR8355 embeds I2C hardware module, which could act as Master mode or Slave mode. I2C is a popular inter-IC interface requiring only 2 bus lines, a serial data line (SDA) and a serial clock line (SCL).

7.3.1 Communication protocol

Telink I2C module supports standard mode (100kbps) and Fast-mode (400kbps) with restriction that system clock must be by at least 10x of data rate.

Two wires, SDA and SCL (SCK) carry information between Master device and Slave device connected to the bus. Each device is recognized by unique address (ID). Master device is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Slave device is the device addressed by a Master.

Both SDA and SCL are bidirectional lines connected to a positive supply voltage via a pull-up resister. It’s recommended to use external 3.3kohm pull-up resistor. For standard mode, the internal pull-up resistor of rank x1 can be used instead of the external 3.3kohm pull-up.

When the bus is free, both lines are HIGH. It’s noted that data in SDA line must keep stable when clock signal in SCL line is at high level, and level state in SDA line is only allowed to change when clock signal in SCL line is at low level.

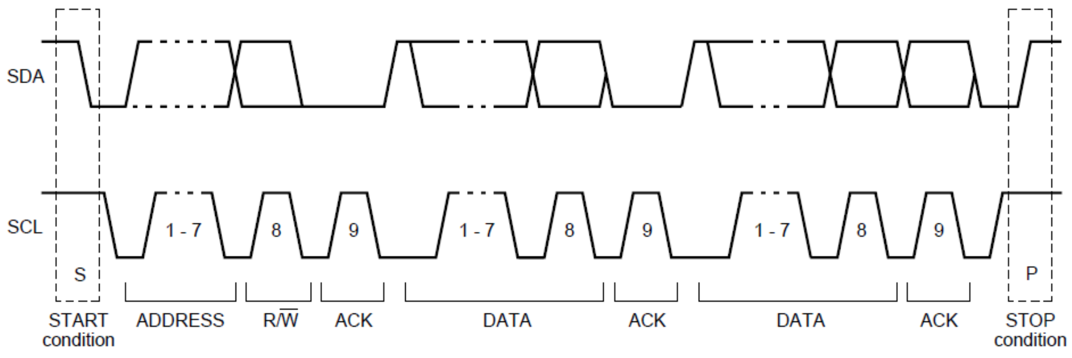


Figure 7- 2 I2C timing chart

7.3.2 Register table

Table 7- 5 Register configuration for I2C

Address	Name	R/W	Description	Reset Value
0x00	I2CSP	RW	I2C master clock speed	0x1f
0x01	I2C_ID	RW	[7:1] I2C ID	0x5c
0x02	I2CMST	RW	[0]: master busy [1]: master packet busy [2]: master received status 0 for ACK; 1 for NAK	0x00
0x03	I2CSCT0	RW	[0]: address auto increase enable [1]: I2C master enable [2]: enable Mapping Mode [3]: r_clk_stretch_en, suspend transmission by pulling SCL down to low level, and continue transmission after SCL is released to high level	0x01
0x04	I2CAD	RW	[7:0] data buffer in master mode	0x5a
0x05	I2CDW	RW	[7:0] Data buffer in master mode	0xf1
0x06	I2CDR	RW	[7:0] Data buffer for Read or Write in master mode	0x00
0x07	I2CSCT1	RW	[0]: launch ID cycle [1]: launch address cycle (send I2CAD data) [2]: launch data write cycle [3]: launch data read cycle For Master Write: 0: I2CAD&I2CDW, 1: I2CAD&I2CDW&I2CDR) To write 3 bytes: bit[3]=1; To write 2 bytes: bit[3]=0. For Master Read: always 1. [4]: launch start cycle [5]: launch stop cycle [6]: enable read ID [7]: enable ACK in read command	0x00
0xe0	I2CMAP_HADR	R	[6:0] I2C read address	0x00
0xe1	HOSR_ADR_L	RW	Low byte of Mapping mode buffer address	0x80
0xe2	HOSR_ADR_M	RW	Middle byte of Mapping mode buffer address	0xd7
0xe3	HOSR_ADR_H	RW	High byte of Mapping mode buffer address	0x00
0xe4	I2CMAP_HOST	RW	[0]: host_cmd_irq_o, I2C host operation has happened. Write 1 to clear. [1]: host_rd_tag_o, I2C host operation has happened and is read operation. Write 1 to clear.	0x00

7.3.3 I2C Slave mode

I2C module of the TLSR8355 acts as Slave mode by default. I2C slave address can be configured via register I2C_ID (address 0x01) [7:1].

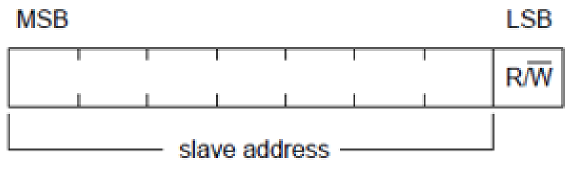


Figure 7- 3 Byte consisted of slave address and R/W flag bit

I2C slave mode supports two sub modes including Direct Memory Access (DMA) mode and Mapping mode, which is selectable via address 0x03[2].

In I2C Slave mode, Master could initiate transaction anytime. I2C slave module will reply with ACK automatically. To monitor the start of I2C transaction, user could set interrupt from GPIO for SCA or SCL.

7.3.3.1 DMA mode

In DMA mode, other devices (Master) could access (read/write) designated address in Register and/or SRAM of the TLSR8355 according to I2C protocol. I2C module of the TLSR8355 will execute the read/write command from I2C master automatically. But user needs to notice that the system clock shall be at least 10x faster than I2C bit rate.

The access address designated by Master is offset by 0x800000. In the TLSR8355, Register address starts from 0x800000 and SRAM address starts from 0x840000. For example, if Addr High (AddrH) is 0x04, Addr Middle (AddrM) is 0x00, and Addr Low (AddrL) is 0xcc, the real address of accessed data is 0x8400cc.

In DMA mode, Master could read/write data byte by byte. The designated access address is initial address and it supports auto increment by setting address 0x03[0] to 1b'1.

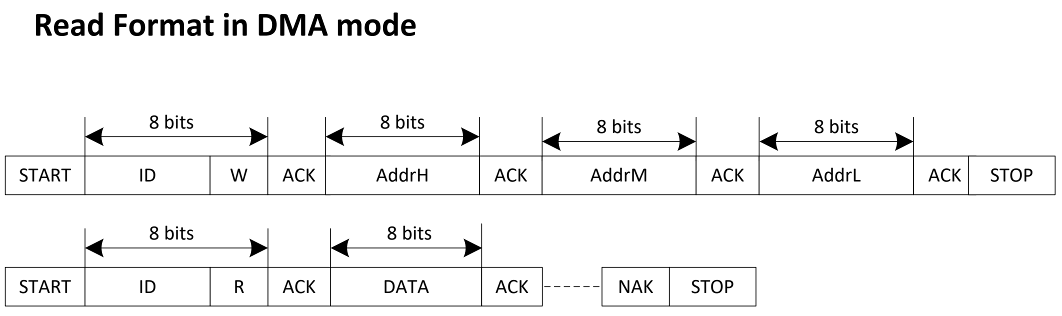


Figure 7- 4 Read format in DMA mode

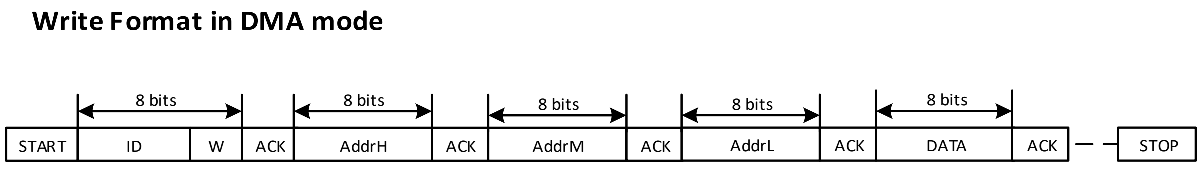


Figure 7- 5 Write format in DMA mode

7.3.3.2 Mapping mode

Mapping mode could be enabled via setting register I2CSCT0 (address 0x03)[2] to 1b'1.

In Mapping mode, data written and read by I2C master will be redirected to specified 128-byte buffer in SRAM. User could specify the initial address of the buffer by configuring registers HOSR_ADR_L (address 0xe1, lower byte), HOSR_ADR_M (address 0xe2, middle byte) and HOSR_ADR_H (address 0xe3, higher byte). The first 64-byte buffer is for written data and following 64-byte buffer is for read data. Every time the data access will start from the beginning of the Write-buffer/Read-buffer after I2C stop condition occurs. The last accessed data address could be checked in register I2CMAP_HADR (address 0xe0) [6:0] which is only updated after I2C STOP occurs.

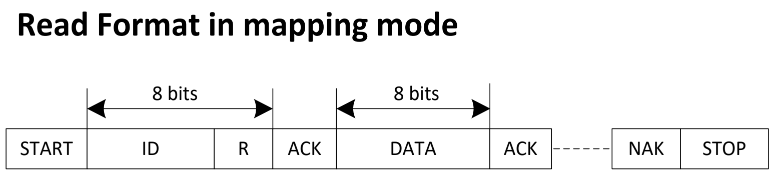


Figure 7- 6 Read format in Mapping mode

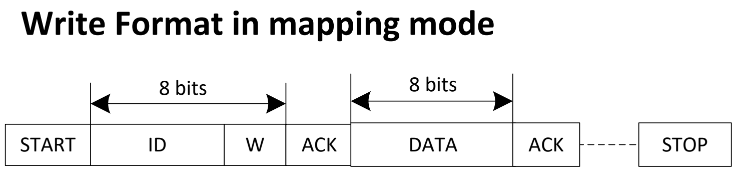


Figure 7- 7 Write format in Mapping mode

7.3.4 I2C Master mode

Address 0x03[1] should be set to 1b'1 to enable I2C master mode for the TLSR8355.

Address 0x00 serves to set I2C Master clock: F_{I2C} = (System Clock / (4 *clock speed configured in address 0x00).

A complete I2C protocol contains START, Slave Address, R/W bit, data, ACK and STOP. Slave address could be configured via address 0x01[7:1].

I2C Master (i.e. I2C module of the TLSR8355) could send START, Slave Address, R/W bit, data and STOP cycle by configuring address 0x07. I2C master will send enabled cycles in the correct sequence.

Address 0x02 serves to indicate whether Master/Master packet is busy, as well as Master received status. Bit[0] will be set to 1 when one byte is being sent, and the bit can be automatically cleared after a start signal/ address byte/acknowledge signal/data /stop signal is sent. Bit[1] is set to 1 when the start signal is sent, and the bit will be automatically cleared after the stop signal is sent. Bit[2] indicates whether to succeed in sending acknowledgement signal.

7.3.4.1 I2C Master Write transfer

I2C Master has 3-byte buffer for write data, which are I2CAD (0x04), I2CDW (0x05) and I2CDR (0x06). Write transfer will be completed by I2C master module.

For example, to implement an I2C write transfer with 3-byte data, which contains START, Slave Address, Write bit, ack from Slave, 1st byte, ack from slave, 2nd byte, ack from slave, 3rd byte, ack from slave and STOP, user needs to configure I2C slave address to I2C_ID (0x01) [7:1], 1st byte data to I2CAD, 2nd byte data to I2CDW and 3rd byte to I2CDR. To start I2C write transfer, I2CSCT1 (0x07) is configured to 0x3f (0011 1111). I2C Master will launch START, Slave address, Write bit, load ACK to I2CMST (0x02) [2], send I2CAD data, load ACK to I2CMST[2], send I2CDW data, load ACK to I2CMST[2], send I2CDR data, load ACK to I2CMST[2] and then STOP sequentially.

For I2C write transfer whose data are more than 3 bytes, user could split the cycles according to I2C protocol.

7.3.4.2 I2C Master Read transfer

I2C Master has one byte buffer for read data, which is I2CDR (0x06). Read transfer will be completed by I2C Master.

For example, to implement an I2C read transfer with 1 byte data, which contains START, Slave Address, Read bit, Ack from Slave, 1st byte from Slave, Ack by master and STOP, user needs to configure I2C slave address to I2C_ID (0x01) [7:1]. To start I2C read transfer, I2CSCT1 (0x07) is configured to 0xf9 (1111 1001). I2C Master will launch START, Slave address, Read bit, load ACK to I2CMST (0x02) [2], load data to I2CDR, reply ACK and then STOP sequentially.

For I2C read transfer whose data are more than 1 byte, user could split the cycles according to I2C protocol.

7.3.5 I2C and SPI Usage

I2C hardware and SPI hardware modules in the chip share part of the hardware, as a result, when both hardware interfaces are used, the restrictions listed within this section need to be taken into consideration.

I2C and SPI hardware cannot be used as Slave at the same time.

The other cases are supported, including:

- ✧ I2C Slave and SPI Master can be used at the same time.
- ✧ I2C Master and SPI Slave can be used at the same time.
- ✧ I2C and SPI can be used as Master at the same time.

Please refer to corresponding SDK instructions for details.

7.4 SPI

The TLSR8355 embeds SPI (Serial Peripheral interface), which could act as Master mode or Slave mode. SPI is a high-speed, full-duplex and synchronous communication bus requiring 4 bus lines including a chip select (CS) line, a data input (DI) line, a data output (DO) line and a clock (CK) line.

7.4.1 Register table

Table 7- 6 Register configuration for SPI

Address	Name	R/W	Description	Reset Value
0x08	SPIDAT	RW	[7:0]: SPI data access	0x00
0x09	SPICT	RW	[0]: mst_csn, control SPI_CSN output when SPI acts as Master [1]: enable master mode [2]: spi data output disable [3]: 1 for read command; 0 for write command [4]: address auto increase [5]: share_mode [6]: busy status	0x11
0x0a	SPISP	RW	[6:0]: SPI clock speed [7]: SPI function mode, p_csn, p_scl, p_sda and p_sdo function as SPI if 1	0x05
0x0b	SPIMODE	RW	[0]: inverse SPI clock output [1]: data delay half clk	0x00

7.4.2 SPI Master mode

SPI for the TLSR8355 supports both master mode and slave mode and acts as slave mode by default. Address 0x09 bit[1] should be set to 1b'1 to enable SPI Master mode. Register SPISP is to configure SPI pin and clock: setting address 0x0a bit[7] to 1 is to enable SPI function mode, and corresponding pins can be used as SPI pins; SPI clock = system clock/((clock speed configured in address 0x0a bit[6:0] +1)*2).

Address 0x08 serves as the data register. One reading/writing operation of 0x08 enables the SPI_CK pin to generate 8 SPI clock cycles.

Telink SPI supports four standard working modes: Mode 0~Mode 3. Register SPIMODE (address 0x0b) serves to select one of the four SPI modes:

Table 7- 7 SPI Master mode

SPI mode	**CPOL/CPHA**	**SPIMODE register (Address 0x0b)**
Mode 0	CPOL=0, CPHA=0	bit[0]=0, bit[1]=0
Mode 1	CPOL=0, CPHA=1	bit[0]=0, bit[1]=1
Mode 2	CPOL=1, CPHA=0	bit[0]=1, bit[1]=0
Mode 3	CPOL=1, CPHA=1	bit[0]=1, bit[1]=1
CPOL: Clock Polarity When CPOL=0, SPI_CLK keeps low level in idle state; When CPOL=1, SPI_CLK keeps high level in idle state. CPHA: Clock Phase When CPHA=0, data is sampled at the first edge of clock period When CPHA=1, data is sampled at the latter edge of clock period		

Address 0x09 bit[0] is to control the CS line: when the bit is set to 1, the CS level is high; when the bit is cleared, the CS level is low.

Address 0x09 bit[2] is the disabling bit for SPI Master output. When the bit is cleared, MCU writes data into address 0x08, then the SPI_DO pin outputs the data bit by bit during the 8 clock cycles generated by the SPI_CK pin. When the bit is set to 1b'1, SPI_DO output is disabled.

Address 0x09 bit[3] is the enabling bit for SPI Master reading data function. When the bit is set to 1b'1, MCU reads the data from address 0x08, then the input data from the SPI_DI pin is shifted into address 0x08 during the 8 clock cycles generated by the SPI_CK pin. When the bit is cleared, SPI Master reading function is disabled.

Address 0x09[5] is the enabling bit for share mode, i.e. whether SPI_DI and SPI_DO share one common line.

Users can read address 0x09 bit[6] to get SPI busy status, i.e. whether the 8 clock pulses have been sent.

7.4.3 SPI Slave mode

SPI for the TLSR8355 acts as slave mode by default. SPI Slave mode supports DMA. User could access registers of the TLSR8355 by SPI interface. It's noted that system clock of TLSR8355 shall be at least 5x faster than SPI clock for reliable connection. Address 0x0a should be written with data 0xa5 by the SPI host to activate SPI slave mode. SPI salve only supports Mode0 and Mode3.

Table 7- 8 SPI Slave mode

SPI slave mode	CPOL/CPHA
Mode 0	CPOL=0, CPHA=0
Mode 3	CPOL=1, CPHA=1
Receive data at positive edge of SPI MCLK clock. Send data at negative edge of SPI MCLK clock.	

Address 0x09[4] is dedicated for SPI Slave mode and indicates address auto increment. SPI write command format and read command format are illustrated in Figure 7-8:

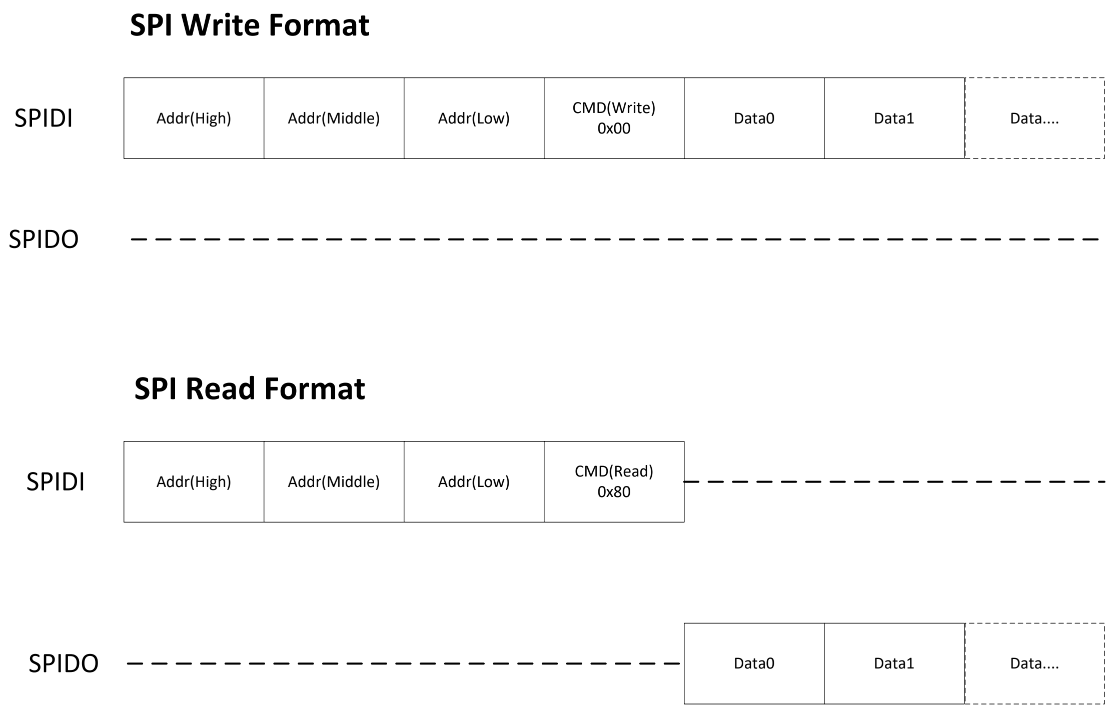


Figure 7- 8 SPI write/read command format

7.4.4 I2C and SPI Usage

I2C hardware and SPI hardware modules in the chip share part of the hardware, as a result, when both hardware interfaces are used, certain restrictions apply.

See **Section 7.3.5 I2C and SPI Usage** for detailed instructions.

7.5 USB

The TLSR8355 has a full-speed (12Mbps) USB interface for communicating with other compatible digital devices. The USB interface acts as a USB peripheral, responding to requests from a master host controller. The chip contains internal 1.5kohm pull up resistor for the DP pin, which can be enabled via analog register afe_0x0b<7>.

Telink USB interface supports the Universal Serial Bus Specification, Revision v2.0 (USB v2.0 Specification).

The chip supports 9 endpoints, including control endpoint 0 and 8 configurable data endpoints. Endpoint 1, 2, 3, 4, 7 and 8 can be configured as input endpoint, while endpoint 5 and 6 can be configured as output endpoint. In audio class application, only endpoint 6 supports iso out mode, while endpoint 7 supports iso in mode. In other applications, each endpoint can be configured as bulk, interrupt and iso mode. For control endpoint 0, the chip's hardware vendor command is configurable.

Optional suspend mode:

- ✧ Selectable as USB suspend mode or chip suspend mode, support remote wakeup.
- ✧ Current draw in suspend mode complied with USB v2.0 Specification.
- ✧ USB pins (DM, DP) can be used as GPIO function in suspend mode.
- ✧ Resume and detach detect: Recognize USB device by detecting the voltage on the DP pin with configurable 1.5K pull-up resistor.
- ✧ USB pins configurable as wakeup GPIOs.

The USB interface belongs to an independent power domain, and it can be configured to power down independently.

8 PWM

The TLSR8355 supports 6-channel PWM (Pulse-Width-Modulation) output. Each PWM#n (n=0~5) has its corresponding inverted output at PWM#n_N pin.

8.1 Register table

Table 8- 1 Register table for PWM

Address	Mnemonic	Type	Description	Reset Value
0x780	PWM_EN	R/W	[1]: 0--disable PWM1, 1--enable PWM1 [2]: 0--disable PWM2, 1--enable PWM2 [3]: 0--disable PWM3, 1--enable PWM3 [4]: 0--disable PWM4, 1--enable PWM4 [5]: 0--disable PWM5, 1--enable PWM5	0x00
0x781	PWM_EN0	R/W	[0]: 0--disable PWM0, 1--enable PWM0	0x00
0x782	PWM_CLKDIV	R/W	Set PWM_clk: (PWM_CLKDIV+1)*sys_clk	0x00
0x783	PWM_MODE	R/W	[3:0]: PWM0 mode select 0000-pwm0 normal mode 0001-pwm0 count mode 0011-pwm0 IR mode 0111-pwm0 IR FIFO mode 1111-pwm0 IR DMA FIFO mode	0x00
0x784	PWM_CC0	R/W	[5:0]:1‘b1 invert PWM output	0x00
0x785	PWM_CC1	R/W	[5:0]:1‘b1 invert PWM_INV output	0x00
0x786	PWM_CC2	R/W	[5:0]: Signal frame polarity of PWM5~PWM0 1b’0-high level first 1b’1-low level first	0x00
0x788~ 0x793	reserved			
0x794	PWM_TCMP0	R/W	[7:0] bits 7-0 of PWM0's high time or low time(if pola[0]=1)	0x00
0x795	PWM_TCMP0	R/W	[15:8] bits 15-8 of PWM0's high time or low time	0x00
0x796	PWM_TMAX0	R/W	[7:0] bits 7-0 of PWM0's cycle time	0x00
0x797	PWM_TMAX0	R/W	[15:8] bits 15-8 of PWM0's cycle time	0x00
0x798	PWM_TCMP1	R/W	[7:0] bits 7-0 of PWM1's high time	0x00

Address	Mnemonic	Type	Description	Reset Value
			or low time(if pola[1]=1)	
0x799	PWM_TCMP1	R/W	[15:8] bits 15-8 of PWM1's high time or low time	0x00
0x79a	PWM_TMAX1	R/W	[7:0] bits 7-0 of PWM1's cycle time	0x00
0x79b	PWM_TMAX1	R/W	[15:8] bits 15-8 of PWM1's cycle time	0x00
0x79c	PWM_TCMP2	R/W	[7:0] bits 7-0 of PWM2's high time or low time(if pola[2]=1)	0x00
0x79d	PWM_TCMP2	R/W	[15:8] bits 15-8 of PWM2's high time or low time	0x00
0x79e	PWM_TMAX2	R/W	[7:0] bits 7-0 of PWM2's cycle time	0x00
0x79f	PWM_TMAX2	R/W	[15:8] bits 15-8 of PWM2's cycle time	0x00
0x7a0	PWM_TCMP3	R/W	[7:0] bits 7-0 of PWM3's high time or low time(if pola[3]=1)	0x00
0x7a1	PWM_TCMP3	R/W	[15:8] bits 15-8 of PWM3's high time or low time	0x00
0x7a2	PWM_TMAX3	R/W	[7:0] bits 7-0 of PWM3's cycle time	0x00
0x7a3	PWM_TMAX3	R/W	[15:8] bits 15-8 of PWM3's cycle time	0x00
0x7a4	PWM_TCMP4	R/W	[7:0] bits 7-0 of PWM4's high time or low time(if pola[4]=1)	0x00
0x7a5	PWM_TCMP4	R/W	[15:8] bits 15-8 of PWM4's high time or low time	0x00
0x7a6	PWM_TMAX4	R/W	[7:0] bits 7-0 of PWM4's cycle time	0x00
0x7a7	PWM_TMAX4	R/W	[15:8] bits 15-8 of PWM4's cycle time	0x00
0x7a8	PWM_TCMP5	R/W	[7:0] bits 7-0 of PWM5's high time or low time(if pola[5]=1)	0x00
0x7a9	PWM_TCMP5	R/W	[15:8] bits 15-8 of PWM5's high time or low time	0x00
0x7aa	PWM_TMAX5	R/W	[7:0] bits 7-0 of PWM5's cycle time	0x00
0x7ab	PWM_TMAX5	R/W	[15:8] bits 15-8 of PWM5's cycle time	0x00
0x7ac	PWM_PNUM0	R/W	[7:0] bits 7-0 of PWM0 Pulse number in count mode and IR mode	0x00
0x7ad	PWM_PNUM0	R/W	[13:8] bits 13-8 of PWM0 Pulse number in count mode and IR mode	0x00
0x7ae~ 0x7af	reserved			
0x7b0	PWM_MASK0	R/W	INT mask [0] PWM0 Pnum int 0: disable 1: Enable [1] PWM0 ir dma fifo mode int	0x00

Address	Mnemonic	Type	Description	Reset Value
			0: disable 1: Enable [2] PWM0 frame int 0: disable 1: Enable [3] PWM1 frame int 0: disable 1: Enable [4] PWM2 frame int 0: disable 1: Enable [5] PWM3 frame int 0: disable 1: Enable [6] PWM4 frame int 0: disable 1: Enable [7] PWM5 frame int 0: disable 1: Enable	
0x7b1	PWM_INT0	R/W	INT status, write 1 to clear [0]: PWM0 pnum int (have sent PNUM pulses, PWM_NCNT==PWM_PNUM) [1]:PWM0 ir dma fifo mode int(pnum int &fifo empty in ir dma fifo mode) [2]: PWM0 cycle done int (PWM_CNT==PWM_TMAX) [3]: PWM1 cycle done int (PWM_CNT==PWM_TMAX) [4]: PWM2 cycle done int (PWM_CNT==PWM_TMAX) [5]: PWM3 cycle done int (PWM_CNT==PWM_TMAX) [6]: PWM4 cycle done int (PWM_CNT==PWM_TMAX) [7]: PWM5 cycle done int (PWM_CNT==PWM_TMAX)	0x00
0x7b2	PWM_MASK1	R/W	[0]: PWM0 fifo mode fifo cnt int mask 0: disable, 1: Enable	0x00
0x7b3	PWM_INT1	R/W	INT status, write 1 to clear [0]: fifo mode cnt int, when FIFO_NUM (0x7cd[3:0]) is less than FIFO_NUM_LVL (0x7cc[3:0])	0x00
0x7b4	PWM_CNT0	R	[7:0]PWM0 cnt value	0x00
0x7b5	PWM_CNT0		[15:8]PWM0 cnt value	0x00
0x7b6	PWM_CNT1	R	[7:0]PWM1 cnt value	0x00
0x7b7	PWM_CNT1		[15:8]PWM1 cnt value	0x00
0x7b8	PWM_CNT2	R	[7:0]PWM2 cnt value	0x00
0x7b9	PWM_CNT2		[15:8]PWM2 cnt value	0x00
0x7ba	PWM_CNT3	R	[7:0]PWM3 cnt value	0x00

Address	Mnemonic	Type	Description	Reset Value
0x7bb	PWM_CNT3		[15:8]PWM3 cnt value	0x00
0x7bc	PWM_CNT4	R	[7:0]PWM4 cnt value	0x00
0x7bd	PWM_CNT4		[15:8]PWM4 cnt value	0x00
0x7be	PWM_CNT5	R	[7:0]PWM5 cnt value	0x00
0x7bf	PWM_CNT5		[15:8]PWM5 cnt value	0x00
0x7c0	PWM_NCNT0	R	[7:0]PWM0 pluse_cnt value	0x00
0x7c1	PWM_NCNT0		[15:8]PWM0 pluse_cnt value	0x00
0x7c2 ~ 0x7c3	reserved			
0x7c4	PWM_TCMP0_SHADOW	R/W	[7:0] bits 7-0 of PWM0's high time or low time(if pola[0]=1),if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode	0x55
0x7c5	PWM_TCMP0_SHADOW	R/W	[15:8] bits 15-8 of PWM0's high time or low time ,if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode	0x55
0x7c6	PWM_TMAX0_SHADOW	R/W	[7:0] bits 7-0 of PWM0's cycle time, if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode	0x00
0x7c7	PWM_TMAX0_SHADOW	R/W	[15:8] bits 15-8 of PWM0's cycle time, if shadow bit(fifo frame[14]) is 1'b1 in ir fifo mode or dma fifo mode	0x00
0x7c8	FIFO_DAT0_ENTRY	R/W	Use in ir fifo mode	0x00
0x7c9	FIFO_DAT1_ENTRY	R/W	Use in ir fifo mode	0x00
0x7ca	FIFO_DAT2_ENTRY	R/W	Use in ir fifo mode	0x00
0x7cb	FIFO_DAT3_ENTRY	R/W	Use in ir fifo mode	0x00
0x7cc	FIFO_NUM_LVL	R/W	FIFO num int trigger level	0x00
0x7cd	FIFO_SR	R	[3:0]:FIFO DATA NUM(byte) [4]:FIFO EMPTY [5]:FIFO FULL	0x10
0x7ce	FIFO_CLR	W1	[0]: write 1 to clear data in FIFO	0x00

8.2 Enable PWM

Register PWM_EN (address 0x780)[5:1] and PWM_EN0 (address 0x781)[0] serves to enable PWM5~PWM0 respectively via writing “1” for the corresponding bits.

8.3 Set PWM clock

PWM clock derives from system clock. Register PWM_CLKDIV (address 0x782) serves to set the frequency dividing factor for PWM clock. Formula below applies:

$$F_{PWM}= F_{System\ clock} / (PWM\_CLKDIV+1)$$

8.4 PWM waveform, polarity and output inversion

Each PWM channel has independent counter and 2 status including “Count” and “Remaining”. Count and Remaining status form a signal frame.

8.4.1 Waveform of signal frame

When PWM#n is enabled, first PWM#n enters Count status and outputs High level signal by default. When PWM#n counter reaches cycles set in register PWM_TCMP#n (address 0x794~0x795, 0x798~0x799, 0x79c~0x79d, 0x7a0~0x7a1, 0x7a4~0x7a5, 0x7a8~0x7a9) / PWM_TCMP0_SHADOW (0x7c4~0x7c5), PWM#n enters Remaining status and outputs Low level till PWM#n cycle time configured in register PWM_TMAX#n (address 0x796~0x797, 0x79a~0x79b, 0x79e~0x79f, 0x7a2~0x7a3, 0x7a6~0x7a7, 0x7aa~0x7ab) / PWM_TMAX0_SHADOW (0x7c6~0x7c7) expires.

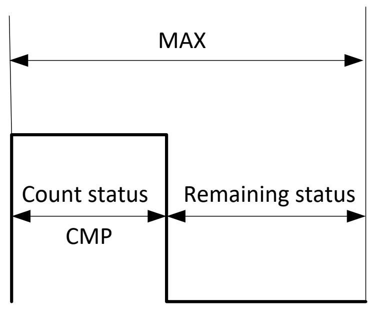


Figure 8- 1 A signal frame

An interruption will be generated at the end of each signal frame if enabled via register PWM_MASK (address 0x7b0[2:7]).

8.4.2 Invert PWM output

PWM#n and PWM#n_N output could be inverted independently via register PWM_CC0 (address 0x784) and PWM_CC1 (address 0x785). When the inversion bit is enabled, waveform of the corresponding PWM channel will be inverted completely.

8.4.3 Polarity for signal frame

By default, PWM#n outputs High level at Count status and Low level at Remaining status. When the corresponding polarity bit is enabled via register PWM_CC2 (address 0x786[5:0]), PWM#n will output Low level at Count status and High level at Remaining status.

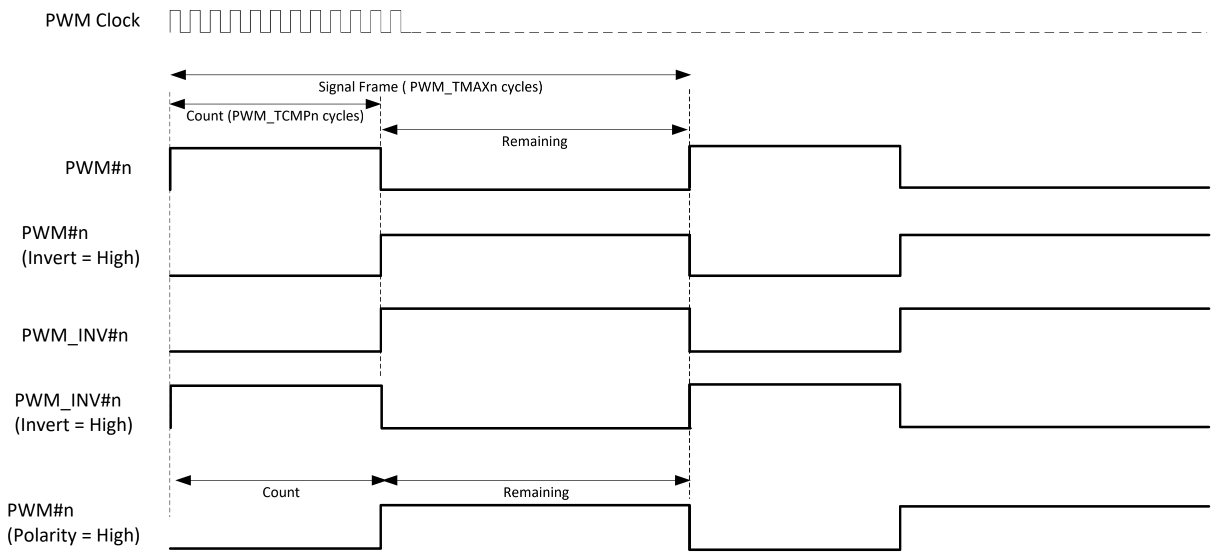


Figure 8- 2 PWM output waveform chart

8.5 PWM mode

8.5.1 Select PWM mode

PWM0 supports five modes, including Continuous mode (normal mode, default), Counting mode, IR mode, IR FIFO mode, IR DMA FIFO mode.

PWM1~PWM5 only support Continuous mode.

Register PWM_MODE (address 0x783) serves to select PWM0 mode.

8.5.2 Continuous mode

PWM0~PWM5 all support Continuous mode. In this mode, PWM#n continuously sends out signal frames. PWM#n should be disabled via address 0x780/0x781 to stop it; when stopped, the PWM output will turn low immediately.

During Continuous mode, waveform could be changed freely via PWM_TCMP#n and PWM_TMAX#n. New configuration for PWM_TCMP#n and PWM_TMAX#n will take effect in the next signal frame.

After each signal frame is finished, corresponding PWM cycle done interrupt flag bit (0x7b1[2:7]) will be automatically set to 1b’1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[2:7]) as 1b’1, a frame interruption will be generated. User needs to write 1b’1 to the flag bit to manually clear it.

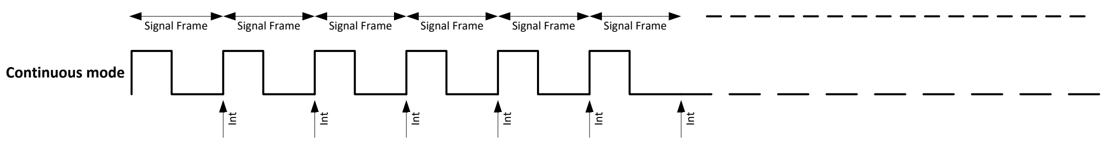


Figure 8- 3 Continuous mode

8.5.3 Counting mode

Only PWM0 supports Counting mode. Address 0x783[3:0] should be set as 4b'0001 to select PWM0 counting mode.

In this mode, PWM0 sends out specified number of signal frames which is defined as a pulse group. The number is configured via register PWM_PNUM0 (address 0x7ac~0x7ad).

After each signal frame is finished, PWM0 cycle done interrupt flag bit (0x7b1[2]) will be automatically set to 1b'1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[2]) as 1b'1, a frame interruption will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

After a pulse group is finished, PWM0 will be disabled automatically, and PWM0 pnum interrupt flag bit (0x7b1[0]) will be automatically set to 1b'1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[0]) as 1b'1, a Pnum interruption will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

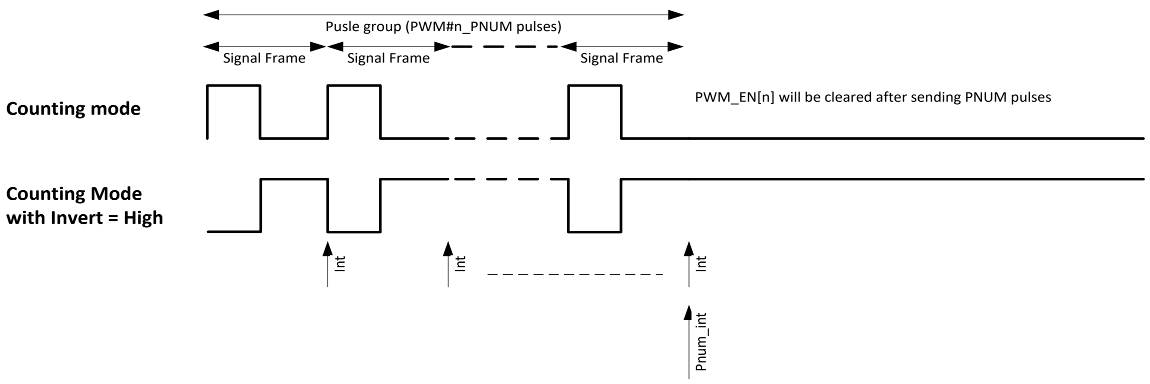

Pusle group (PWM#n_PNUM pulses)
Signal Frame
Signal Frame
Signal Frame
Counting mode
PWM_EN[n] will be cleared after sending PNUM pulses
Counting Mode with Invert = High
Int
Int
Int
Pnum_int

Figure 8- 4 Counting mode (n=0)

Counting mode also serves to stop IR mode gracefully. Refer to **section 8.5.4** for details.

8.5.4 IR mode

Only PWM0 supports IR mode. Address 0x783[3:0] should be set as 4b'0011 to select PWM0 IR mode.

In this mode, specified number of frames is defined as one pulse group. In contrast to Counting mode where PWM0 stops after first pulse group is finished, PWM0 will constantly send pulse groups in IR mode.

During IR mode, PWM0 output waveform could also be changed freely via WM_TCMP0, PWM_TMAX0 and PWM_PNUM0. New configuration for PWM_TCMP0, PWM_TMAX0 and PWM_PNUM0 will take effect in the next pulse group.

To stop IR mode and complete current pulse group, user can switch PWM0 from IR mode to Counting mode so that PWM0 will stop after current pulse group is finished. If PWM0 is disabled directly via PWM_EN0 (0x781[0]), PWM0 output will turn Low immediately despite of current pulse group.

After each signal frame/pulse group is finished, PWM0 cycle done interrupt flag bit (0x7b1[2])/PWM0 pnum interrupt flag bit (0x7b1[0]) will be automatically set to 1b’1. A frame interruption/Pnum interruption will be generated (if enabled by setting address 0x7b0[2]/0x7b0[0] as 1b’1).

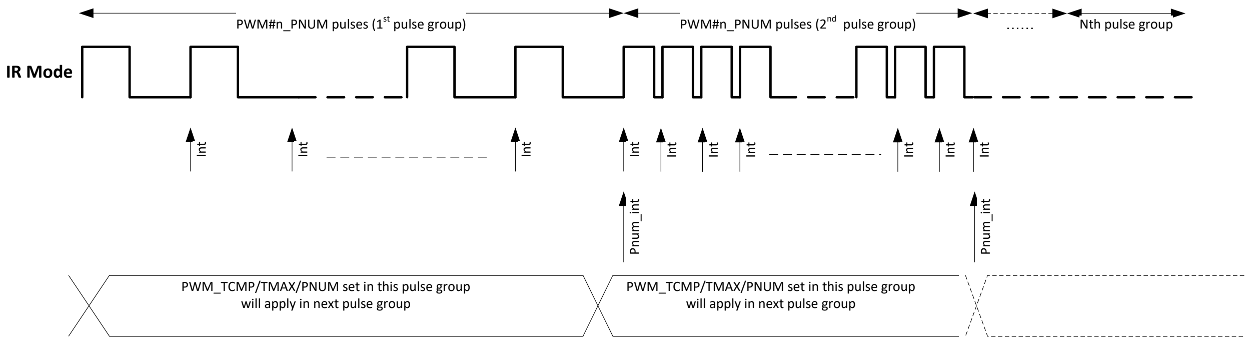

PWM#n_PNUM pulses (1st pulse group)
PWM#n_PNUM pulses (2nd pulse group)
......
Nth pulse group
IR Mode
Int
Pnum_int
PWM_TCMP/TMAX/PNUM set in this pulse group will apply in next pulse group
PWM_TCMP/TMAX/PNUM set in this pulse group will apply in next pulse group

Figure 8- 5 IR mode (n=0)

8.5.5 IR FIFO mode

IR FIFO mode is designed to allow IR transmission of long code patterns without the continued intervention of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36kHz, 38kHz, 40kHz, or 56kHz.

Only PWM0 supports IR FIFO mode. Address 0x783[3:0] should be set as 4b’0111 to select PWM0 IR FIFO mode.

An element (“FIFO CFG Data”) is defined as basic unit of IR waveform, and written into FIFO. This element consists of 16 bits, including:

- ✧ bit[13:0] defines PWM pulse number of current group.
- ✧ bit[14] determines duty cycle and period for current PWM pulse group.

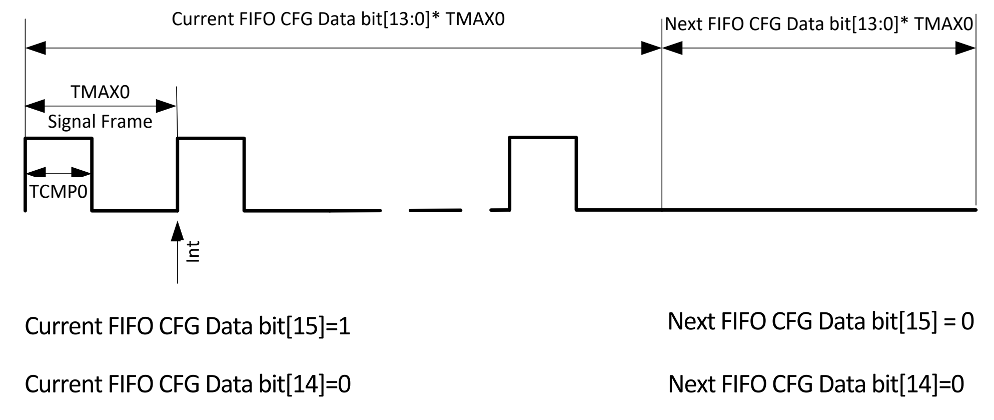
  0: use configuration of TCMP0 and TMAX0 in 0x794~0x797;

  1: use configuration of TCMP0_SHADOW and TMAX0_SHADOW in 0x7c4~0x7c7.
- ✧ bit[15] determines whether current PWM pulse group is used as carrier, i.e. whether PWM will output pulse (1) or low level (0).

User should use FIFO_DATA_ENTRY in 0x7c8~0x7cb to write the 16-bit “FIFO CFG Data” into FIFO by byte or half word or word.

- ✧ To write by byte, user should successively write 0x7c8, 0x7c9, 0x7ca and 0x7cb.
- ✧ To write by half word, user should successively write 0x7c8 and 0x7ca.
- ✧ To write by word, user should write 0x7c8.

FIFO depth is 8 bytes. User can read the register FIFO_SR in 0x7cd to view FIFO empty/full status and check FIFO data number.


Current FIFO CFG Data bit[13:0]* TMAX0
Next FIFO CFG Data bit[13:0]* TMAX0
TMAX0
Signal Frame
TCMP0
Int
Current FIFO CFG Data bit[15]=1
Current FIFO CFG Data bit[14]=0
Next FIFO CFG Data bit[15] = 0
Next FIFO CFG Data bit[14]=0

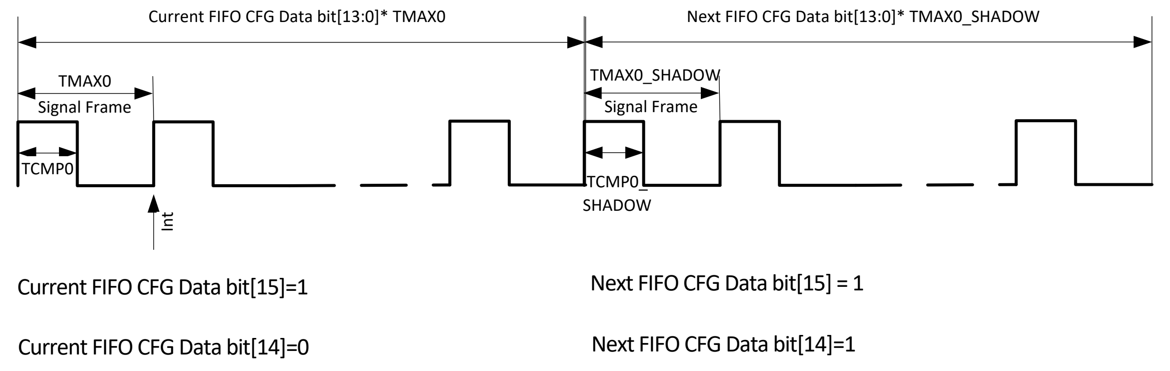

Current FIFO CFG Data bit[13:0]* TMAX0
Next FIFO CFG Data bit[13:0]* TMAX0_SHADOW
TMAX0
Signal Frame
TCMP0
Int
TMAX0_SHADOW
Signal Frame
TCMP0_
SHADOW
Current FIFO CFG Data bit[15]=1
Current FIFO CFG Data bit[14]=0
Next FIFO CFG Data bit[15] = 1
Next FIFO CFG Data bit[14]=1

Figure 8- 6 IR format examples

When “FIFO CFG Data” is configured in FIFO and PWM0 is enabled via PWM_EN0 (address 0x781[0]), the configured waveforms will be output from PWM0 in sequence. As long as FIFO doesn’t overflow, user can continue to add waveforms during IR waveforms sending process, and long IR code that exceeds the FIFO depth can be implemented this way. After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically.

The FIFO_CLR register (address 0x7ce[0]) serves to clear data in FIFO. Writing 1b’1 to this register will clear all data in the FIFO. Note that the FIFO can only be cleared when not in active transmission.

8.5.6 IR DMA FIFO mode

IR DMA FIFO mode is designed to allow IR transmission of long code patterns without occupation of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36kHz, 38kHz, 40kHz, or 56kHz.

Only PWM0 supports IR DMA FIFO mode. Address 0x783[3:0] should be set as 4b’1111 to select

PWM0 IR DMA FIFO mode.

This mode is similar to IR FIFO mode, except that “FIFO CFG Data” is written into FIFO by DMA instead of MCU. User should write the configuration of “FIFO CFG Data” into RAM, and then enable DMA channel 5. DMA will automatically write the configuration into FIFO.

***Note:** In this mode, when DMA channel 5 is enabled, PWM will automatically output configured waveform, without the need to manually enable PWM0 via 0x781[0] (i.e. 0x781[0] will be set as 1b’1 automatically).

Example 1:

Suppose Mark carrier (pulse) frequency1(F1) = 40kHz, duty cycle 1/3

Mark carrier (pulse) frequency2(F2) = 50kHz, duty cycle 1/2

Space carrier (low level) frequency(F3) = 40kHz

If user wants to make PWM send waveforms in following format (PWM CLK =24MHz):

Burst(20[F1]), i.e. 20 F1 pulses

Burst(30[F2]),

Burst(50[F1]) ,

Burst(50[F2]),

Burst(20[F1],10[F3]),

Burst(30[F2],10[F3])

Step1: Set carrier F1 frequency as 40kHz, set duty cycle as 1/3.

Set **PWM_TMAX0** as 0x258 (i.e. 24MHz/40kHz=600=0x258).

Since duty cycle is 1/3, set **PWM_TCMP0** as 0xc8 (i.e. 600/3=200=0xc8).

Set carrier F2 frequency as 50kHz, set duty cycle as 1/2.

Set **PWM_TMAX0_SHADOW** as 0x1e0 (i.e. 24MHz/50kHz=480=0x1e0).

Since duty cycle is 1/2, set **PWM_TCMP0_SHADOW** as 0xf0 (i.e. 480/2=240=0xf0).

Step2: Generate “FIFO CFG Data” sequence.

Burst(20[F1]): {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d20}=0x8014.

Burst(30[F2]): {[15]: 1’b1, [14]: 1’b1, [13:0]: ’d30}=0xc01e.

Burst(50[F1]) : {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d50}=0x8032.

Burst(50[F2]): {[15]: 1’b1, [14]: 1’b1, [13:0]:’d50}=0xc032.

Burst(20[F1],10[F3]): {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d20}=0x8014,

{[15]: 1’b0, [14]: 1’b0, [13:0]: ’d10}=0x000a.

Burst(30[F2],10[F3]): {[15]: 1’b1, [14]: 1’b1, [13:0]: ’d30}=0xc01e, {[15]:1’b0, [14]: 1’b0, [13:0]: ’d10}=0x000a.

Step3: Write “FIFO CFG Data” into SRAM in DMA format.

DMA SOURCE ADDRESS+0x00: 0x0000_0010 (dma transfer-length: 16byte)

DMA SOURCE ADDRESS+0x04: 0xc01e_8014 (LITTLE ENDIAN)

DMA SOURCE ADDRESS+0x08: 0xc032_8032

DMA SOURCE ADDRESS+0x0c: 0x000a_8014

DMA SOURCE ADDRESS+0x10: 0x000a_c01e

Step4: Enable DMA channel 5 to send PWM waveforms.

Write 1’b1 to address 0x524[5] to enable DMA channel 5.

After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically (address 0x781[0] is automatically cleared). The FIFO mode stop interrupt flag bit (address 0x7b3[0]) will be automatically set as 1b’1. If the interrupt is enabled by setting PWM_MASK1 (address 0x7b2[0]) as 1b’1, a FIFO mode stop interrupt will be generated. User needs to write 1b’1 to the flag bit to manually clear it.

Example 2:

Suppose carrier frequency is 38kHz, system clock frequency is 24MHz, duty cycle is 1/3, and the format of IR code to be sent is shown as below:

1) Preamble waveform: 9ms carrier + 4.5ms low level.
2) Data 1 waveform: 0.56ms carrier + 0.56ms low level.
3) Data 0 waveform: 0.56ms carrier + 1.69ms low level.
4) Repeat waveform: 9ms carrier + 2.25ms low level + 0.56ms carrier. Repeat waveform duration is 11.81ms, interval between two adjacent repeat waveforms is 108ms.
5) End waveform: 0.56ms carrier.

User can follow the steps below to configure related registers:

Step1: Set carrier frequency as 38kHz, set duty cycle as 1/3.

Set **PWM_TMAX0** as 0x277 (i.e. 24MHz/38kHz=631=0x277).

Since duty cycle is 1/3, set **PWM_TCMP0** as 0xd2 (i.e. 631/3=210=0xd2).

Step2: Generate “FIFO CFG Data” sequence.

Preamble waveform:

9ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 9*38=’d 342=14’h 156}=0x8156

4.5ms low level: {[15]:1’b0, [14]:1’b0, [13:0]: 4.5*38=’d 171=14’h ab}=0x00ab

Data 1 waveform:

0.56ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 0.56*38=’d 21=14’h 15}=0x8015

0.56ms low level: {[15]:1’b0, [14]:1’b0, [13:0]: 0.56*38=’d 21=14’h 15}=0x0015

Data 0 waveform:

0.56ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 0.56*38=’d 21=14’h 15}=0x8015

1.69ms low level: {[15]:1’b0, [14]:1’b0, [13:0]: 1.69*38=’d 64=14’h 40}=0x0040

Repeat waveform:

9ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 9*38=’d 342=14’h 156}=0x8156

2.25ms low level: {[15]:1’b0, [14]:1’b0, [13:0]: 2.25*38=’d 86=14’h 56}=0x0056

0.56ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 0.56*38=’d 21=14’h 15}=0x8015

108ms -11.81ms =96.19ms low level:

{[15]:1’b0, [14]:1’b0, [13:0]: 96.19*38=’d 3655=14’h e47}=0x0e47

End waveform:

0.56ms carrier: {[15]:1’b1, [14]:1’b0, [13:0]: 0.56*38=’d 21=14’h 15}=0x8015

Step3: Write “IR CFG Data” into SRAM in DMA format.

If user want PWM0 to send IR waveform in following format:

Preamble+0x5a+Repeat+End

Preamble: 0x8156, 0x00ab

0x5a=8’b01011010

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Repeat: 0x8156, 0x0056, 0x8015, 0x0e47

End: 0x8015.

User needs to write the configuration information above into source address of DMA channel 5, as shown below:

DMA SOURCE ADDRESS+0x00: 0x0000_002e (dma transfer-length: 46byte)

DMA SOURCE ADDRESS+0x04: 0x00ab_8156 (Preamble) (LITTLE ENDIAN)

DMA SOURCE ADDRESS+0x08: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x0c: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x10: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x14: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x18: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x1c: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x20: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x24: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x28: 0x0056_8156 (Repeat)

DMA SOURCE ADDRESS+0x2c: 0x0e47_8015 (Repeat)

DMA SOURCE ADDRESS+0x30: 0x8015 (End)

Step4: Enable DMA channel 5 to send PWM waveforms.

Write 1'b1 to address 0x524[5] to enable DMA channel 5.

After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically (address 0x781[0] is automatically cleared). The FIFO mode stop interrupt flag bit (address 0x7b3[0]) will be automatically set as 1b'1. If the interrupt is enabled by setting PWM_MASK1 (address 0x7b2[0]) as 1b'1, a FIFO mode stop interrupt will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

8.6 PWM interrupt

There are 9 interrupt sources from PWM function.

After each signal frame, PWM#n (n=0~5) will generate a frame-done IRQ (Interrupt Request) signal.

In Counting mode and IR mode, PWM0 will generate a Pnum IRQ signal after completing a pulse group.

In IR FIFO mode, PWM0 will generate a FIFO mode count IRQ signal when the FIFO_NUM value is less than the FIFO_NUM_LVL, and will generate a FIFO mode stop IRQ signal after FIFO becomes empty.

In IR DMA FIFO mode, PWM0 will generate an IR waveform send done IRQ signal, after DMA has sent all configuration data, FIFO becomes empty and final waveform is sent.

To enable PWM interrupt, the total enabling bit "irq_pwm" (address 0x641[6], see **section 6 Interrupt**) should be set as 1b'1. To enable various PWM interrupt sources, PWM_MASK0 (address 0x7b0[7:0]) and PWM_MASK1 (address 0x7b2[0]) should be set as 1b'1 correspondingly.

Interrupt status can be cleared via register PWM_INT0 (address 0x7b1[7:0]) and PWM_INT1 (address 0x7b3[0]).

9 Quadrature Decoder

The TLSR8355 embeds one quadrature decoder (QDEC) which is designed mainly for applications such as wheel. The QDEC implements debounce function to filter out jitter on the two phase inputs, and generates smooth square waves for the two phase.

9.1 Input pin selection

The QDEC supports two phase input; each input is selectable from the 2 pins of PortA via setting address 0xd2[2:0] (for channel a)/0xd3[2:0] (for channel b).

Table 9- 1 Input pin selection

Address 0xd2[2:0]/0xd3[2:0]	Pin
0	PA[2]
1	PA[3]
2	RSVD (PB[6])
3	RSVD (PB[7])
4	RSVD (PC[2])
5	RSVD (PC[3])
6	RSVD (PD[6])
7	RSVD (PD[7])

Note: To use corresponding IO as QDEC input pin, it's needed first to enable GPIO function, enable "IE" (1) and disable "OEN" (1) for this IO.

9.2 Common mode and double accuracy mode

The QDEC embeds an internal hardware counter, which is not connected with bus.

Address 0xd7[0] serves to select common mode or double accuracy mode.

For each wheel rolling step, two pulse edges (rising edge or falling edge) are generated.

If address 0xd7[0] is cleared to select common mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 only when the same rising/falling edges are detected from the two phase signals.

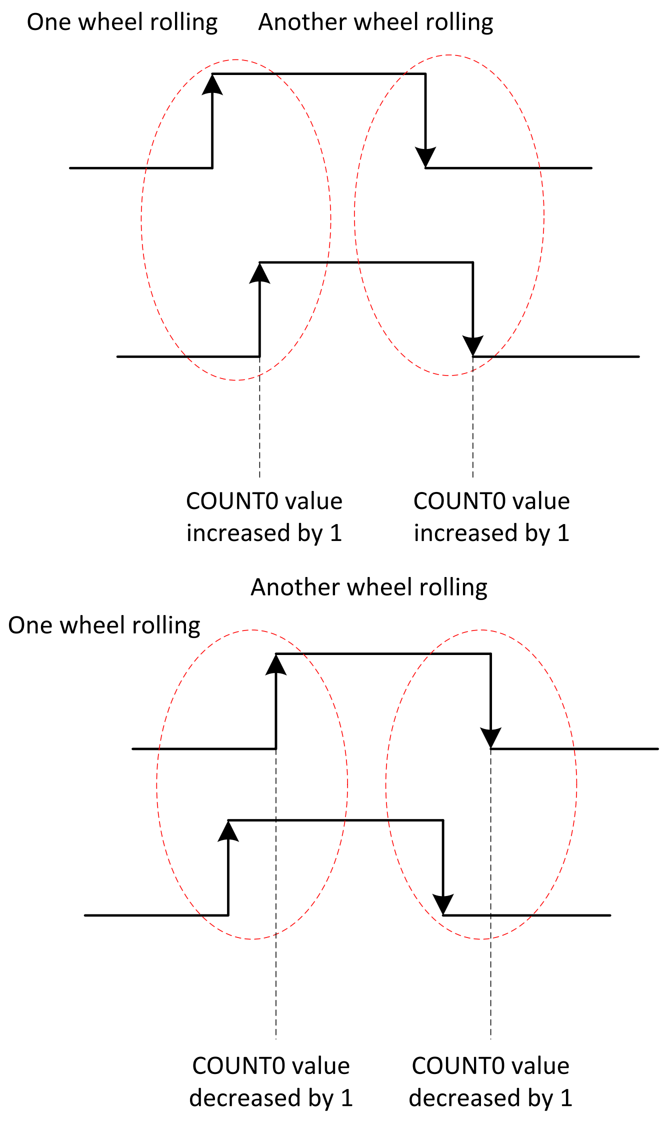

One wheel rolling
Another wheel rolling
COUNT0 value increased by 1
COUNT0 value increased by 1
Another wheel rolling
One wheel rolling
COUNT0 value decreased by 1
COUNT0 value decreased by 1

Figure 9- 1 Common mode

If address 0xd7[0] is set to 1b'1 to select double accuracy mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 on each rising/falling edge of the two phase signals; the COUNT0 will be increased/decreased by 2 for one wheel rolling.

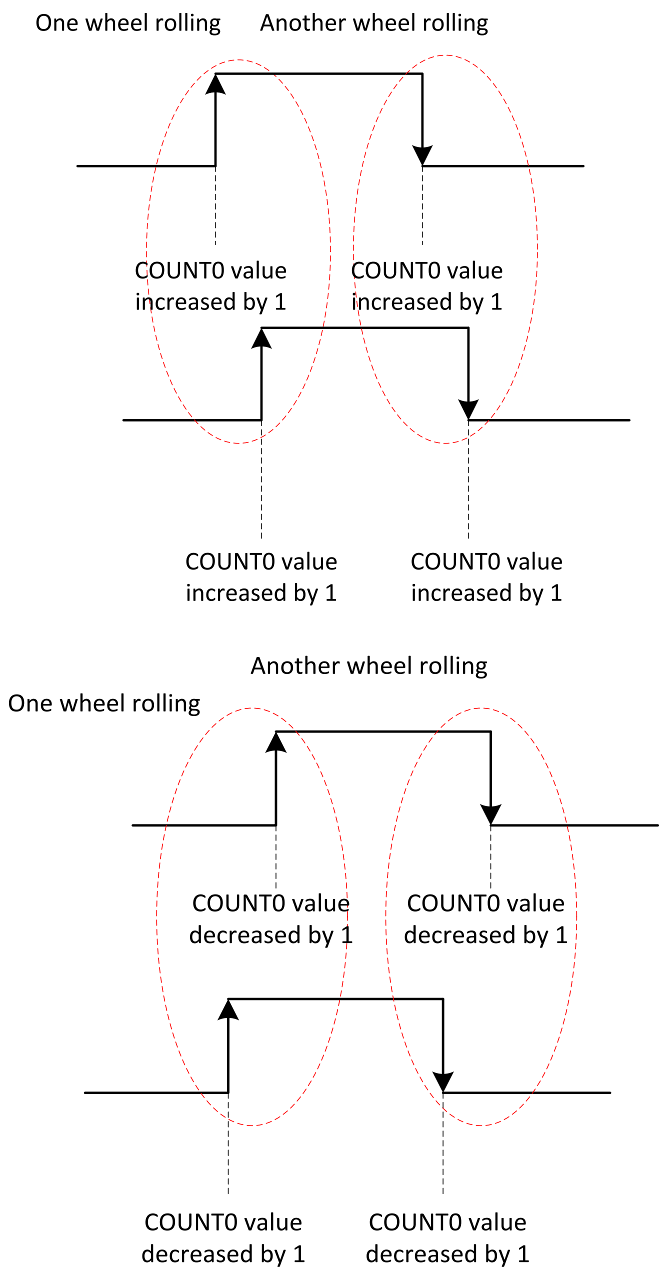

One wheel rolling
Another wheel rolling
COUNT0 value increased by 1
COUNT0 value increased by 1
COUNT0 value increased by 1
COUNT0 value increased by 1
Another wheel rolling
One wheel rolling
COUNT0 value decreased by 1
COUNT0 value decreased by 1
COUNT0 value decreased by 1
COUNT0 value decreased by 1

Figure 9- 2 Double accuracy mode

9.3 Read real time counting value

Neither can Hardware Counter value be read directly via software, nor can the counting value in address 0xd0 be updated automatically.

To read real time counting value, first write address 0xd8[0] with 1b'1 to load Hardware Counter data into the QDEC_COUNT register, then read address 0xd0.

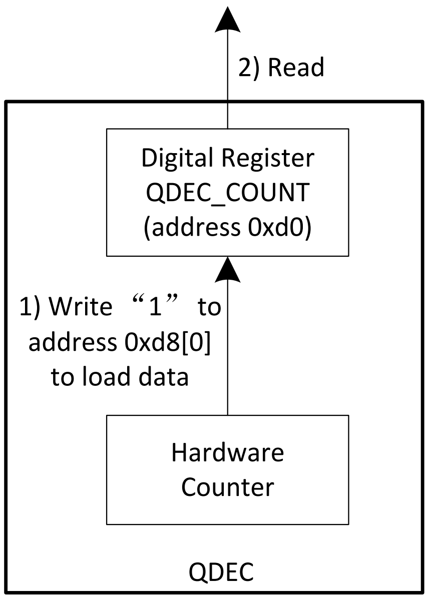

2) Read
Digital Register QDEC_COUNT (address 0xd0)
1) Write “1” to address 0xd8[0] to load data
Hardware Counter
QDEC

Figure 9- 3 Read real time counting value

9.4 QDEC reset

Address 0x60[5] serves to reset the QDEC. The QDEC Counter value is cleared to zero.

9.5 Other configuration

The QDEC supports hardware debouncing. Address 0xd1[2:0] serves to set filtering window duration. All jitter with period less than the value will be filtered out and thus does not trigger count change.

Address 0xd1[4] serves to set input signal initial polarity.

Address 0xd1[5] serves to enable shuttle mode. Shuttle mode allows non-overlapping two phase signals as shown in the following figure.

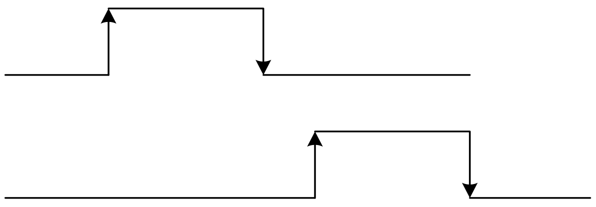


Figure 9- 4 Shuttle mode

9.6 Timing sequence

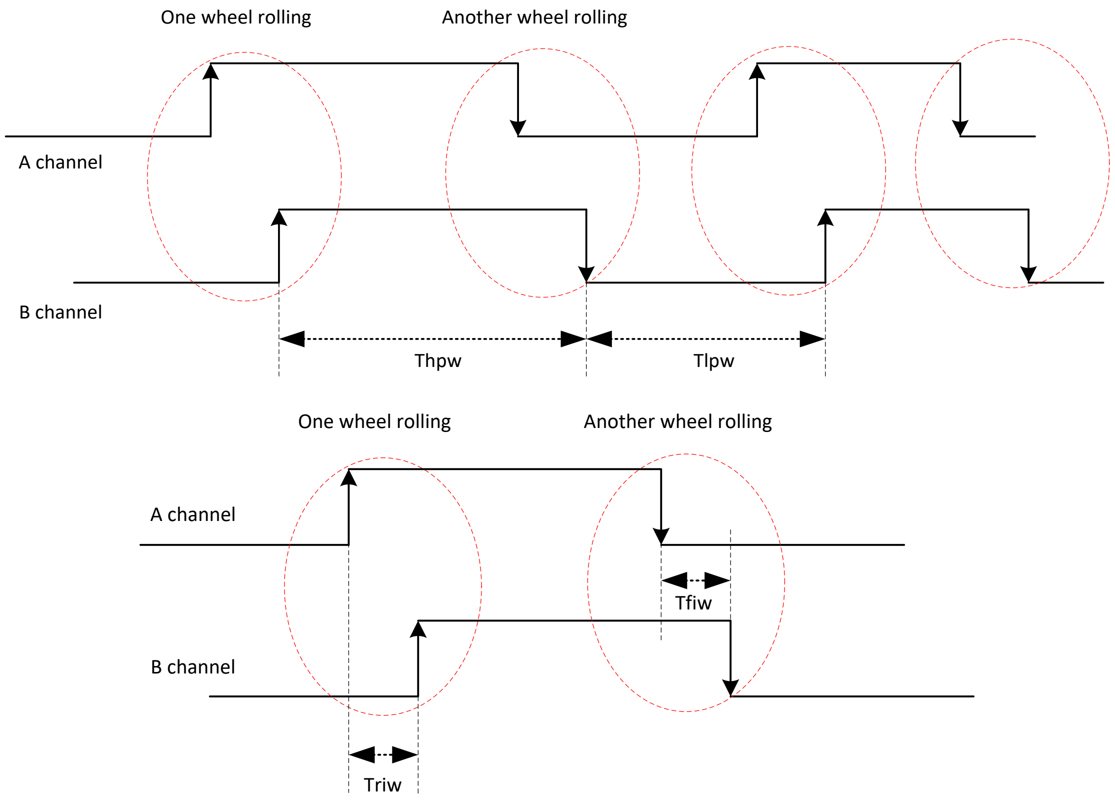


Figure 9- 5 Timing sequence chart

Table 9- 2 Timing

Time interval	Min Value
Thpw (High-level pulse width)	2^(n+1) *clk_32kHz *3 (n=0xd1[2:0])
Tlpw (Low-level pulse width)	2^(n+1) *clk_32kHz *3 (n=0xd1[2:0])
Triw (Interval width between two rising edges)	2^(n+1) *clk_32kHz (n=0xd1[2:0])
Tfiw (Interval width between two falling edges)	2^(n+1) *clk_32kHz (n=0xd1[2:0])

QDEC module works based on 32kHz clock to ensure it can work in suspend mode. QDEC module supports debouncing function, and any signal with width lower than the threshold (i.e. "2^(n+1) *clk_32kHz *3 (n=0xd1[2:0])) will be regarded as jitter. Therefore, effective signals input from Channel A and B should contain high/low level with width Thpw/Tlpw more than the threshold. The 2^n *clk_32kHz clock is used to synchronize input signal of QDEC module, so the interval between two adjacent rising/falling edges from Channel A and B, which are marked as Triw and Tfiw, should exceed "2^(n+1) *clk_32kHz".

Only when the timing requirements above are met, can QDEC module recognize wheel rolling times correctly.

9.7 Register table

Table 9- 3 Register table for QDEC

Address	Mnemonic	Type	Description	Reset value
0xd0	QDEC_COUNT	R	QDEC Counting value (read to clear): Pulse edge number	0x00
0xd1	QDEC_CC	R/W	[2:0] : filter time (can filter 2^n *clk_32k*2 width deglitch) [4]: pola, input signal pola 0: no signal is low, 1: no signal is high [5]:shuttle mode 1 to enable shuttle mode	0x00
0xd2	QDEC_CHNA	R/W	[2:0] QDEC input pin select for channel a choose 1 of 8 pins for input channel a 7~0: {rsvd(pd[7:6]), rsvd(pc[3:2]), rsvd(pb[7:6]), pa[3:2]}	0x00
0xd3	QDEC_CHNB	R/W	[2:0] QDEC input pin select for channel b choose 1 of 8 pins for input channel b 7~0: {rsvd(pd[7:6]), rsvd(pc[3:2]), rsvd(pb[7:6]), pa[3:2]}	0x01
0xd6	QDEC_RST	R/W	[0]RSVD	0x00
0xd7	QDEC_DOUBLE	R/W	[0]Enable double accuracy mode	0x01
0xd8	DATA_LOAD	R/W	[0]write 1 to load data when load completes it will be 0	0x00

10 Manchester Decoder

The TLSR8355 integrates one Manchester Decoder (MDEC). The MDEC is designed to decode the input Manchester code, data after Manchester coding, into binary data.

10.1 Frame format

The MDEC's input sequence includes a Carrier signal, a Start flag, a 39-bit mdec_data filed (mdec_data[38:0]), and an End flag.

- ✧ Carrier signal duration should be no less than 3ms.
- ✧ Support duty cycle of 50%~90%.
- ✧ Period for each bit is 408us.
- ✧ The Start flag is Manchester code 1, a positive edge from low level to high level.
- ✧ The End flag is Manchester code 0, a negative edge from high level to low level.

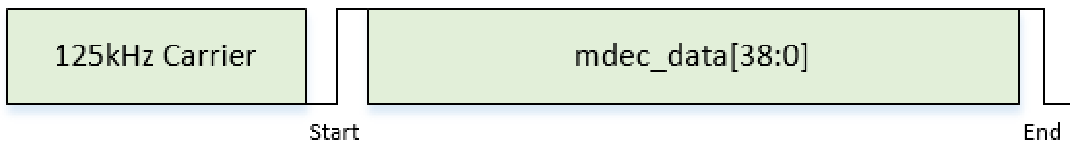


Figure 10- 1 Frame format

10.2 Function description

10.2.1 Block diagram

The MDEC uses 32kHz clock, and it mainly embeds a finite State machine, three counters, and a Shift Register to implement its function, including:

- ✧ Finite State Machine: It includes Idle state, Carrier state, Start state, Data state, and End state.
- ✧ count_carrier: This counter serves to detect carrier signal in Idle state. When a carrier signal is detected, the MDEC's state machine enters Start state.
- ✧ count_32k: After entering Start state, this counter serves to calculate the interval between two adjacent positive edges, so as to judge the input data.
- ✧ count_bit: This counter serves to record the number of bits that have been decoded, so as to judge whether data decoding of a frame is finished. When the bit number reaches 39, it indicates decoding is finished.
- ✧ Shift Register: This register serves to store binary data after decoding.

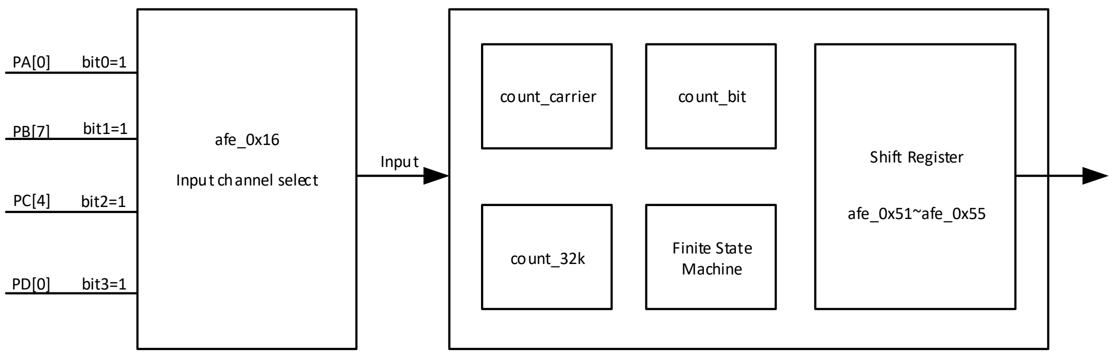


Figure 10- 2 Function block diagram

10.2.2 Reset MDEC

The analog register afe_0x16 bit[4] serves to reset the MDEC module. To use the MDEC, it’s needed to set this bit as 1b’0.

10.2.3 Select input channel

User can input the Manchester code from specific GPIO pin into the MDEC.

The analog register afe_0x16 bits[3:0] serves to select PA[0], PB[7], PC[4] and PD[0] as input channel, respectively.

10.2.4 Read result data

Data after decoding, mdec_data[38:0], are available in the Shift Register, i.e. the analog registers afe_0x51~afe_0x55.

After data decoding of a frame is finished, if the 4-bit mdec_data[38:35] in the analog register afe_0x51[7:4]) is consistent with the mdec_match_value written in the analog register afe_0x17[3:0], a MCU wakeup signal will be generated.

10.3 Register table

Table 10- 1 Analog registers for MDEC

Address	Bit range	Type	Description	Default value
afe_0x16	[4]	RW	reset mdec 1: reset MDEC and clear MDEC wakeup status (afe_0x44[4]); To use MDEC, please set as 0.	1
	[3]	RW	select PD[0] as data input	0
	[2]	RW	Rsvd (select PC[4] as data input)	0
	[1]	RW	Rsvd (select PB[7] as data input)	0
	[0]	RW	Rsvd (select PA[0] as data input)	0
afe_0x17	[3:0]	RW	mdec_match_value	2
afe_0x44	[4]	RO	MDEC wakeup status	
afe_0x51	[7:4]	RO	mdec_data[38:35]	
	[3]	RO	RSVD	
	[2:0]	RO	mdec_data[34:32]	
afe_0x52	[7:0]	RO	mdec_data[31:24]	
afe_0x53	[7:0]	RO	mdec_data[23:16]	
afe_0x54	[7:0]	RO	mdec_data[15:8]	
afe_0x55	[7:0]	RO	mdec_data[7:0]	

11 SAR ADC

The TLSR8355 integrates one SAR ADC module, which can be used to sample analog input signals such as battery voltage and temperature sensor.

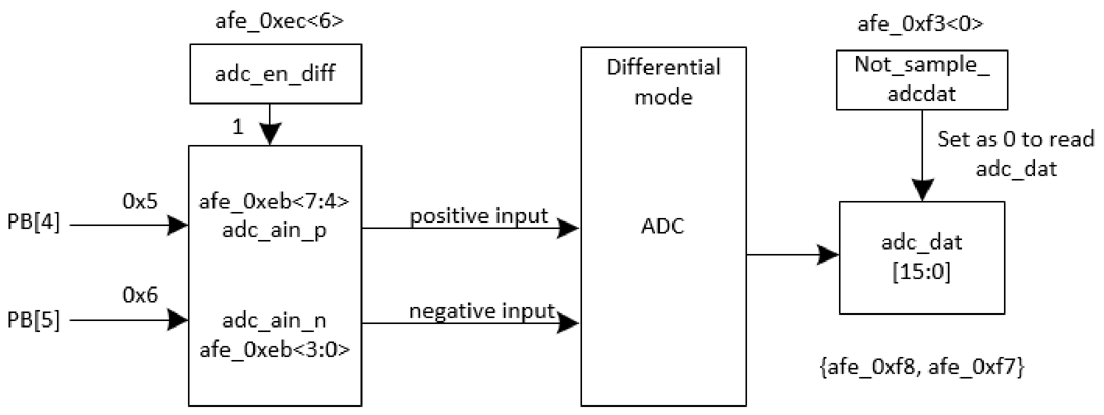


Figure 11- 1 Block diagram of ADC

11.1 Power on/down

The SAR ADC is disabled by default. To power on the ADC, the analog register adc_pd (afe_0xfc<5>) should be set as 1b’0.

11.2 ADC clock

ADC clock is derived from external 24MHz crystal source, with frequency dividing factor configurable via the analog register adc_clk_div (afe_0xf4<2:0>).

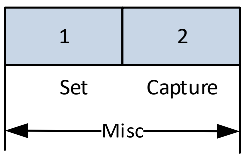
ADC clock frequency (marked as F_{ADC_clk}) = 24MHz/(adc_clk_div+1)

11.3 ADC control in auto mode

11.3.1 Set max state and enable channel

The SAR ADC supports Misc channel which consists of one “Set” state and one “Capture” state.

✧ The analog register r_max_scnt (afe_0xf2<5:4>) serves to set the max state index. As shown below, the r_max_scnt should be set as 0x02.



✧ The Misc channel can be enabled via r_en_misc (afe_0xf2<2>).

11.3.2 “Set” state

The length of “Set” state for the Misc channel is configurable via the analog register r_max_s (afe_0xf1<3:0>).

“Set” state duration (marked as T_{sd}) = r_max_s / 24MHz.

Each “Set” state serves to set ADC control signals for the Misc channel via corresponding analog registers, including:

✧ adc_en_diff: afe_0xec<6>. MUST set as 1b’1 to select differential input mode.

✧ adc_ain_p: afe_0xeb<7:4>. Select positive input in differential mode.

✧ adc_ain_n: afe_0xeb<3:0>. Select negative input in differential mode.

✧ adc_vref: afe_0xea<1:0>. Set reference voltage V_{REF}. ADC maximum input range is determined by the ADC reference voltage.

✧ adc_sel_ai_scale: afe_0xfa<7:6>. Set scaling factor for ADC analog input as 1 (default), or 1/8.

By setting this scaling factor, ADC maximum input range can be extended based on the V_{REF}.

For example, suppose the V_{REF} is set as 1.2V:

Since the scaling factor is 1 by default, the ADC maximum input range should be 0~1.2V (negative input is GND) / -1.2V~+1.2V (negative input is ADC GPIO pin).

If the scaling factor is set as 1/8, in theory ADC maximum input range should change to 0~9.6V (negative input is GND) / -9.6V~+9.6V (negative input is ADC GPIO pin). But limited by input voltage of the chip’s PAD, the actual range is narrower.

✧ adc_res: afe_0xec<1:0>. Set resolution as 8/10/12/14 bits.

ADC data is always 16-bit format no matter what the resolution is set. For example, 14 bits resolution indicates ADC data consists of 14-bit valid data and 2-bit sign extension bit.

✧ adc_tsamp: afe_0xee<3:0>. Set sampling time which determines the speed to stabilize input signals.

Sampling time (marked as T_{samp}) = adc_tsamp / F_{ADC_clk}.

The lower sampling cycle, the shorter ADC convert time.

11.3.3 "Capture" state

For the Misc channel, at the beginning of its "Capture" state, a "run" signal is issued automatically to start an ADC sampling and conversion process; at the end of "Capture" state, ADC output data is captured.

✧ The length of "Capture" state is configurable via the analog register r_max_mc[9:0] (afe_0xf1<7:6>, afe_0xef<7:0>).

"Capture" state duration for Misc channel (marked as T_{cd}) = r_max_mc / 24MHz.

✧ The "VLD" bit (afe_0xf6<0>) will be set as 1b'1 at the end of "Capture" state to indicate the ADC data is valid, and this flag bit will be cleared automatically.

✧ The 16-bit ADC output data can be read from the analog register adc_dat[15:0] (afe_0xf8<7:0>, afe_0xf7<7:0>) while the afe_0xf3<0> is set as 1b'0 (default). If the afe_0xf3<0> is set as 1b'1, the data in the afe_0xf8 and afe_0xf7 won't be updated.

Note: The total duration "T_{td}", which is the sum of the length of "Set" state and "Capture" state, determines the sampling rate.

Sampling frequency (marked as F_s) = 1 / T_{td}

11.3.4 Usage case with detailed register setting

This case introduces the register setting details for Misc channel sampling.

In this case, afe_0xf2<2> should be set as 1b'1, so as to enable the Misc channel, while the max state index should be set as "2" by setting afe_0xf2<5:4> as 0x2.

The total duration (marked as T_{td}) = (1*r_max_s+1*r_max_mc) / 24MHz.

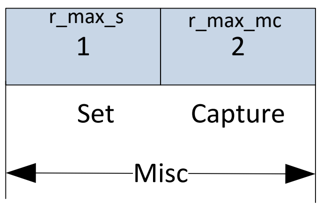


Table 11- 3 Overall register setting

Function	Register setting
Power on the ADC	afe_0xfc<5> = 1b'0
Set F_{ADC_clk} (ADC clock frequency) as 4MHz	afe_0xf4<2:0> = 5 F_{ADC_clk} = 24MHz/(5+1)=4MHz
Enable the Misc channel	afe_0xf2<2> = 1b'1
Set the max state index as "2"	afe_0xf2<5:4> = 2b'10
Set T_{sd} ("Set" state duration)	afe_0xf1<3:0> = 10 T_{sd} = r_max_s/24MHz = 10/24MHz = 0.417us
Set T_{cd} ("Capture" state duration)	afe_0xf1<7:6>=1, afe_0xef<7:0>=0xea T_{cd} = r_max_mc[9:0]/24MHz = 490/24MHz = 20.417us
T_{td} (total duration)	T_{td} = (1*r_max_s+1*r_max_mc) / 24MHz = 500/24MHz =20.83us
F_s (Sampling frequency)	F_s = 1 / T_{td} = 24MHz/500 = 48kHz
Set differential input	afe_0xec<6>=1
Set input channel	afe_0xeb=0x56 Select PB[4] as positive input and PB[5] as negative input
Set reference voltage V_{REF}	afe_0xea<1:0>=2 V_{REF} =1.2V
Set scaling factor for ADC analog input	afe_0xfa<7:6>=0 scaling factor: 1
	ADC maximum input range: -1.2V ~ +1.2V
Set resolution	afe_0xec<1:0>=3 resolution: 14bits
Set T_{samp} (determines the speed to stabilize input before sampling)	afe_0xee<3:0>=3 T_{samp} = adc_tsamp / F_{ADC_clk} = 12/4MHz=3us

11.4 Register table

Table 11- 4 Register table related to SAR ADC

Address	Mnemonic	Default value	Description
afe_0xea<1:0>	adc_vrefm	00	Select V_{REF} for Misc channel 0x0: 0.6V 0x1: 0.9V 0x2: 1.2V 0x3: rsvd
afe_0xea<7:2>	RSVD		
afe_0xeb<3:0>	adc_ain_m_n	0000	Select negative input for Misc channel: 0x0: No input 0x1: rsvd (B[0]) 0x2: rsvd (B[1]) 0x3: rsvd (B[2]) 0x4: rsvd (B[3]) 0x5: B[4] 0x6: B[5] 0x7: rsvd (B[6]) 0x8: rsvd (B[7]) 0x9: rsvd (C[4]) 0xa: rsvd (C[5]) 0xb: rsvd 0xc: rsvd 0xd: tempsensor_n (Temperature sensor negative output) 0xe: Ground 0xf: Ground
afe_0xeb<7:4>	adc_ain_m_p	0000	Select positive input for Misc channel: 0x0: No input 0x1: rsvd (B[0]) 0x2: rsvd (B[1]) 0x3: rsvd (B[2]) 0x4: rsvd (B[3]) 0x5: B[4] 0x6: B[5] 0x7: rsvd (B[6]) 0x8: rsvd (B[7]) 0x9: rsvd (C[4]) 0xa: rsvd (C[5]) 0xb: rsvd 0xc: rsvd 0xd: tempsensor_p (Temperature sensor positive output) 0xe: rsvd 0xf: rsvd

Address	Mnemonic	Default value	Description
afe_0xec<1:0>	adc_resm	11	Set resolution for Misc channel 0x0: 8bits 0x1: 10bits 0x2: 12bits 0x3: 14bits
afe_0xec<5:2>	RSVD		
afe_0xec<6>	adc_en_diffm	0	Select input mode for Misc channel. 0: rsvd 1: differential mode
afe_0xec<7>	RSVD		
afe_0xee<3:0>	adc_tsampm	0000	Number of ADC clock cycles in sampling phase for Misc channel to stabilize the input before sampling: 0x0: 3 cycles 0x1: 6 cycles 0x2: 9 cycles 0x3: 12 cycles … 0xf: 48 cycles
afe_0xef<7:0>	r_max_mc[7:0]		r_max_mc[9:0]serves to set length of "capture" state for Misc channel. r_max_s serves to set length of "set" state for Misc channel. Note: State length indicates number of 24M clock cycles occupied by the state.
afe_0xf0<7:0>	rsvd		
afe_0xf1<3:0>	r_max_s		
afe_0xf1<5:4>	rsvd		
afe_0xf1<7:6>	r_max_mc[9:8]		
afe_0xf2<0>	rsvd		
afe_0xf2<1>	rsvd		
afe_0xf2<2>	r_en_misc		Enable Misc channel sampling. 1: enable
afe_0xf2<3>	r_writetocore_disable	0	0: enable write to core 1: disable write to core
afe_0xf2<5:4>	r_max_scnt	00	Set total length for sampling state machine (i.e. max state index)
afe_0xf2<7>	rsvd		
afe_0xf3<0>	Not_sample_adcdat	0	0: sample ADC data to afe_0xf8 and afe_0xf7 1: not sample ADC data to afe_0xf8 and afe_0xf7
afe_0xf3<7:2>	rsvd		
afe_0xf4<2:0>	adc_clk_div	011	ADC clock (derive from external 24M crystal) ADC clock frequency = 24M/(adc_clk_div+1)
afe_0xf4<7:3>	rsvd		
afe_0xf5<7:0>	rsvd		rsvd
afe_0xf6<0>	vld		[0]: vld, ADC data valid status bit (This bit will be set as 1 at the end of capture state to indicate the ADC data is valid, and will be cleared when set state starts.)
afe_0xf6<7:1>	rsvd		rsvd
afe_0xf7<7:0>	adc_dat[7:0]		Read only

Address	Mnemonic	Default value	Description
			[7:0]: Misc adc dat[7:0]
afe_0xf8<7:0>	adc_dat[15:8]		Read only [7:0]: Misc adc_dat[15:8]
afe_0xf9<3:2>	rsvd	00	rsvd
afe_0xfa<7:6>	adc_sel_ai_scale	0	Analog input pre-scaling select sel_ai_scale[1:0]: scaling factor 0x0: 1 0x1: rsvd 0x2: rsvd 0x3: 1/8
afe_0xfc<4>	rsvd	0	rsvd
afe_0xfc<5>	adc_pd	1	Power down ADC 1: Power down 0: Power up

12 Temperature Sensor

The TLSR8355 integrates a temperature sensor and it's used in combination with the SAR ADC to detect real-time temperature.

The temperature sensor is disabled by default. The analog register afe_0x06<2> should be set as 1b'0 to enable the temperature sensor.

Table 12- 1 Analog register for temperature sensor

Address	Name	Description	Default Value
afe3V_reg06<2>	pd_temp_sensor_3V	Power down of temperature sensor: 0: Power up 1: Power down	1

The temperature sensor embeds a pnp transistor. It takes the real-time temperature (T) as input, and outputs voltage drop (V_{EB}) signals of pnp transistor as positive and negative output respectively.

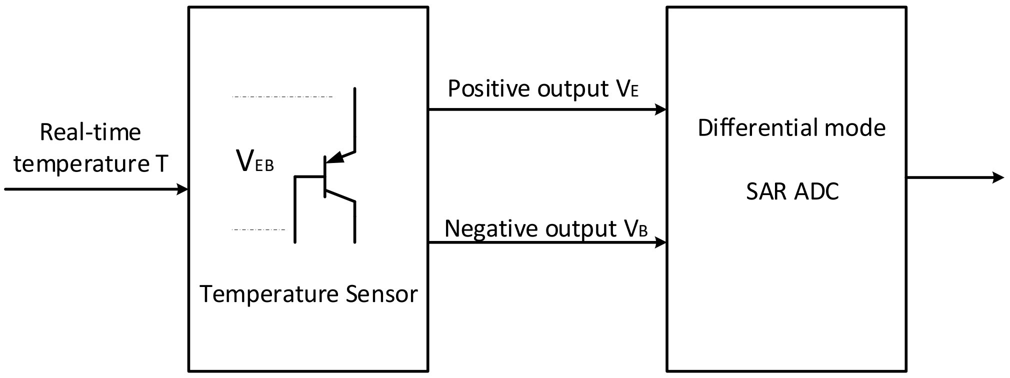


Figure 12- 1 Block diagram of temperature sensor

The voltage drop V_{EB} signals is determined by the real-time temperature T, as shown below:

$$\begin{aligned} V_{EB} &= 884mV - 1.4286mV/℃ * \left(T - (-40℃)\right) \\ &= 884mV - 1.4286mV/℃ * (T + 40℃) \end{aligned}$$

In this formula, "884mV" indicates the value of V_{EB} at the temperature of "−40℃".

To detect the temperature, the positive and negative output of the temperature sensor should be enabled as the input channels of the SAR ADC. The ADC will convert the V_{EB} signals into digital signal.

✧ The ADC should be configured as differential mode, and the positive and negative output of the temperature sensor should be configured as differential input of the ADC. The ADC should initiate one operation and obtain one output signal (ADCOUT); therefore,

$$V_{EB} = \frac{ADCOUT}{2^N - 1} * V_{REF}.$$

In the formula, “N” and “V_{REF}” indicate the selected resolution and reference voltage of the SAR ADC.

Then the real-time temperature T can be calculated according to the V_{EB}.

13 Low Power Comparator

The TLSR8355 embeds a low power comparator. This comparator takes two inputs: input derived from external PortB (PB<1>~PB<7>), and reference input derived from internal reference, PB<0>, PB<3>, AVDD3 or float.

By comparing the input voltage multiplied by selected scaling coefficient with reference input voltage, the low power comparator will output high or low level accordingly.

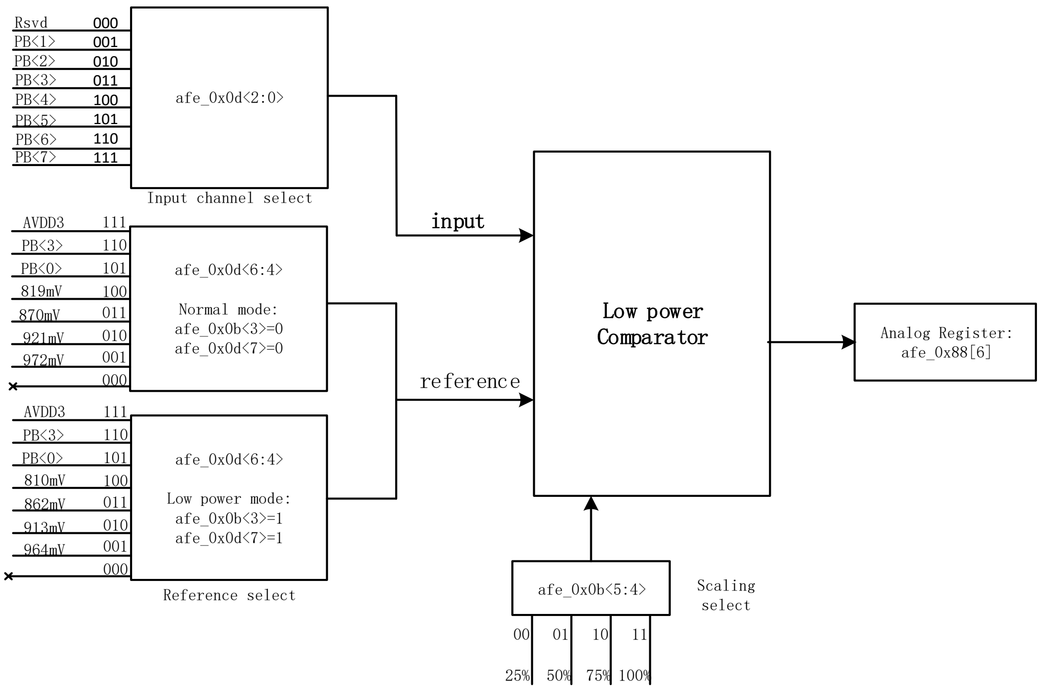


Figure13- 1 Block diagram of low power comparator

13.1 Power on/down

The low power comparator is powered down by default.

The analog register afe_0x07<3> serves to control power state of the low power comparator: By clearing this bit, this comparator will be powered on; by setting this bit to 1b'1, this comparator will be powered down.

To use the low power comparator, first set afe_0x07<3> as 1b'0, then the 32K RC clock source is enabled as the comparator clock.

13.2 Select input channel

Input channel is selectable from the PortB (PB<1>~PB<7>) via the analog register afe_0x0d<2:0>.

13.3 Select mode and input channel for reference

Generally, it's needed to clear both the afe_0x0b<3> and afe_0x0d<7> to select the normal mode. In normal mode, the internal reference is derived from UVLO and has higher accuracy, but current bias is larger (10uA); reference voltage input channel is selectable from internal reference of 972mV, 921mV, 870mV and 819mV, as well as PB<0>, PB<3>, AVDD3 and float.

To select the low power mode, both the afe_0x0b<3> and afe_0x0d<7> should be set as 1b'1. In low power mode, the internal reference is derived from Bandgap and has lower accuracy, but current bias is decreased to 50nA; reference voltage input channel is selectable from internal reference of 964mV, 913mV, 862mV and 810mV, as well as PB<0>, PB<3>, AVDD3 and float.

13.4 Select scaling coefficient

Equivalent reference voltage equals the selected reference input voltage divided by scaling coefficient.

The analog register afe_0x0b<5:4> serves to select one of the four scaling options: 25%, 50%, 75% and 100%.

13.5 Low power comparator output

The low power comparator output is determined by the comparison result of the value of [input voltage *scaling] and reference voltage input. The comparison principle is shown as below:

- ✧ If the value of [input voltage *scaling] is larger than reference voltage input, the output will be low ("0").
- ✧ If the value of [input voltage *scaling] is lower than reference voltage input, the output will be high ("1").
- ✧ If the value of [input voltage *scaling] equals reference voltage input, or input channel is selected as float, the output will be uncertain.

User can read the output of the low power comparator via the analog register afe_0x88[6].

The output of the low power comparator can be used as signal to wakeup system from low power modes.

13.6 Register table

Table 13- 1 Analog register table related to low power comparator

Address	Name	Description	Default Value
afe_0x07<3>	pd_lc_comp_3V	Power on/down low power comparator: 0: Power up 1: Power down	1
afe_0x0b<3>	ref_mode_sel	Reference mode select: 0: normal mode (current bias 10uA) 1: low power mode (current bias 50nA) See afe_0x0d<7>.	1
afe_0x0b<5:4>	comp_refscale<1:0>	Reference voltage scaling: 00: 25% 01: 50%	01

Address	Name	Description	Default Value
		10: 75% 11: 100%	
afe_0x0d<2:0>	comp_chsel<2:0>	Input Channel select: 000: rsvd 001: B<1> 010: B<2> 011: B<3> 100: B<4> 101: B<5> 110: B<6> 111: B<7>	000
afe_0x0d<3>	vbus_inen	rsvd	0
afe_0x0d<6:4>	comp_refesel<2:0>	Reference select: normal mode low power mode 000: Float 000: Float 001: 972mV 001: 964mV 010: 921mV 010: 913mV 011: 870mV 011: 862mV 100: 819mV 100: 810mV 101: B<0> 101: B<0> 110: B<3> 110: B<3> 111: AVDD3 111: AVDD3	000
afe_0x0d<7>	pd_I10u	Enable or disable 10uA current bias: 0: Enable 10uA current bias 1: Disable 10uA current bias	1

14 AES

The TLSR8355 embeds AES module with encryption and decryption function. The input 128bit plaintext in combination of key is converted into the final output ciphertext via encryption; the 128bit ciphertext in combination of key can also be converted into 128bit plaintext via decryption.

The AES hardware accelerator provides automatic encryption and decryption. It only takes (1000*system clock cycles) to implement AES encryption/decryption. Suppose system clock is 20MHz, the time needed for AES encryption/decryption is 50us.

Both RISC mode and DMA mode are supported for AES operation.

14.1 RISC mode

For RISC mode, configuration of related registers is as follows:

1) Set the value of key via writing registers AES_KEY0~ AES_KEY15 (address 0x550~0x55f).
2) Set operation method of AES module via register AES_CTRL: set address 0x540[0] as 1b'1 for decryption method, while clear this bit for encryption method.
3) For encryption method, write registers AES-DAT0~ AES-DAT3 (address 0x548~0x54b) for four times to set the 128bit plaintext. After encryption, the 128bit ciphertext can be obtained by reading address 0x548~0x54b for four times.
4) For decryption method, write registers AES-DAT0~ AES-DAT3 (address 0x548~0x54b) for four times to set the 128bit ciphertext. After decryption, the 128bit plaintext can be obtained by reading address 0x548~0x54b for four times.
5) Address 0x540 bit[1] and bit[2] are read only bits: bit[1] will be cleared automatically after quartic writing of address 0x548~0x54b; bit[2] will be set as 1 automatically after encryption/decryption, and then cleared automatically after quartic reading of address 0x548~0x54b.

14.2 DMA mode

As for DMA mode, it is only needed to configure the value of key and encryption/decryption method for AES module. Please refer to point 1) ~ 2) in section **14.1**.

14.3 AES-CCM

The AES-CCM (Counter with the CBC-MAC) mode is disabled by default. AES output is directly determined by current encryption and decryption, irrespective of previous encryption and decryption result.

If 0x540[7] is set as 1b'1 to enable AES-CCM mode, AES output will also take previous encryption and decryption result into consideration.

14.4 Register table

Table 14- 1 Register table related to AES

Address	Mnemonic	Type	Description	Reset Value
0x540	AES_CTRL	R/W	[0] Select decrypt/encrypt. 1: decrypt, 0: encrypt [1] Read-only. 1: input data needed, 0: input data ready. [2] Read-only. 0: output data not ready, 1: output data ready. [7] 1: enable AES-CCM mode.	0x02
0x548	AES-DAT0		Input/Output Data byte 0	0x00
0x549	AES-DAT1		Input/Output Data byte 1	0x00
0x54a	AES-DAT2		Input/Output Data byte 2	0x00
0x54b	AES-DAT3		Input/Output Data byte 3	0x00
0x550	AES_KEY0	R/W	[7:0] KEY0	0x00
0x551	AES_KEY1	R/W	[7:0] KEY1	0x00
0x552	AES_KEY2	R/W	[7:0] KEY2	0x00
0x553	AES_KEY3	R/W	[7:0] KEY3	0x00
0x554	AES_KEY4	R/W	[7:0] KEY4	0x00
0x555	AES_KEY5	R/W	[7:0] KEY5	0x00
0x556	AES_KEY6	R/W	[7:0] KEY6	0x00
0x557	AES_KEY7	R/W	[7:0] KEY7	0x00
0x558	AES_KEY8	R/W	[7:0] KEY8	0x00
0x559	AES_KEY9	R/W	[7:0] KEY9	0x00
0x55a	AES_KEY10	R/W	[7:0] KEY10	0x00
0x55b	AES_KEY11	R/W	[7:0] KEY11	0x00
0x55c	AES_KEY12	R/W	[7:0] KEY12	0x00
0x55d	AES_KEY13	R/W	[7:0] KEY13	0x00
0x55e	AES_KEY14	R/W	[7:0] KEY14	0x00
0x55f	AES_KEY15	R/W	[7:0] KEY15	0x00

15 Key Electrical Specifications

Note: The electrical characteristics currently listed in this section are target specifications and only supplied for reference. Some data may be updated according to actual test results.

15.1 Absolute maximum ratings

Table 15- 1 Absolute Maximum Ratings

Characteristics	Sym.	Min.	Max	Unit	Test Condition
Supply Voltage	VDD	-0.3	3.6	V	All AVDD, DVDD and VDD_IO pin must have the same voltage
Voltage on Input Pin	V_{In}	-0.3	VDD+ 0.3	V	
Output Voltage	V_{Out}	0	VDD	V	
Storage temperature Range	T_{Str}	-65	150	^{o}C	
Soldering Temperature	T_{Sld}		260	^{o}C	

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

15.2 Recommended operating condition

Table 15- 2 Recommended operation condition

Item	Sym.	Min	Typ.	Max	Unit	Condition
Power-supply voltage	VDD	1.8	3.3	3.6	V	All AVDD, DVDD and VDD_IO pin must have the same voltage
Supply rise time (from 1.6V to 1.8V)	t_R			10	ms	
Operating Temperature Range	T_{Opr}	-40		85	^{o}C	ET version
		-40		125	^{o}C	AT version

15.3 DC characteristics

Table 15- 3 DC characteristics (VDD=3.3V, T=25℃)

Item	Sym.	Min	Typ.	Max	Unit	Condition
RX current	I_{Rx}		10.2		mA	Whole Chip
TX current	I_{Tx}		10.7		mA	whole chip @ 0dBm with DCDC
Deep sleep with 16kB SRAM retention	I_{Deep1}		1.3		uA	
Deep sleep with 32kB SRAM retention			1.5		uA	
Deep sleep without SRAM retention	I_{Deep2}		0.4		uA	

15.4 AC characteristics

Table 15- 4 AC Characteristics (VDD=3.3V, T=25℃)

Item	Sym.	Min	Typ.	Max	Unit	Condition
Digital inputs/outputs						
Input high voltage	VIH	0.7VDD		VDD	V	
Input low voltage	VIL	VSS		0.3VDD	V	
Output high voltage	VOH	0.9VDD		VDD	V	
Output low voltage	VOL	VSS		0.1VDD	V	
USB characteristics						
USB Output Signal Cross-over Voltage	V_{Crs}	1.3	-	2.0	V	
RF performance						
Item		**Min**	**Typ**	**Max**	**Unit**	
RF frequency range		2380		2500	MHz	Programmable in 1MHz step

Item	Sym.	Min	Typ.	Max	Unit	Condition
Data rate	BLE/2.4G Proprietary 1Mbps, ±250kHz deviation BLE/2.4G Proprietary 2Mbps, ±500kHz deviation BLE 125kbps, ±250kHz deviation BLE 500kbps, ±250kHz deviation 802.15.4 250kbps, ±500kHz deviation 2.4G Proprietary 500kbps, ±125kHz deviation 2.4G Proprietary 250kbps, ±62.5kHz deviation					
Proprietary 2Mbps RF_Rx performance (±500kHz deviation)*[3]						
Sensitivity	2Mbps		-94.5		dBm	
Frequency Offset Tolerance		-250		+200	kHz	
Co-channel rejection(I/C)			-7		dB	Wanted signal at -67dBm
In-band blocking rejection	+2/-2 MHz offset		9/9		dB	Wanted signal at -67dBm
	+4/-4 MHz offset		40/34		dB	
	>4MHz offset		45		dB	
Image rejection			34		dB	Wanted signal at -67dBm
Proprietary 2Mbps RF_Tx performance						
Output power, maximum setting			10		dBm	
Output power, minimum setting			-45		dBm	
Programmable output power range		55			dB	
Modulation 20dB bandwidth			2.5		MHz	
RSSI						
RSSI range		-100		10	dBm	
Resolution			1		dB	
24MHz crystal						
Nominal frequency (parallel resonant)	f_{NOM}		24		MHz	

[3] For actual sensitivity level of BLE 2Mbps mode, please refer to Bluetooth 5 specification.

Item	Sym.	Min	Typ.	Max	Unit	Condition
Frequency tolerance	f_{TOL}	-20		+20	ppm	
Load capacitance	C_L	5	12	18	pF	Programmable on chip load cap
Equivalent series resistance	ESR		50	100	ohm	
32.768kHz crystal						
Nominal frequency (parallel resonant)	f_{NOM}		32.768		kHz	
Frequency tolerance	f_{TOL}	-100		+100	ppm	
Load capacitance	C_L	6	9	12.5	pF	Programmable on chip load cap
Equivalent series resistance	ESR		50	80	kohm	
24MHz RC oscillator						
Nominal frequency	f_{NOM}		24		MHz	
Frequency tolerance	f_{TOL}		1		%	On chip calibration
32kHz RC oscillator						
Nominal frequency	f_{NOM}		32		kHz	
Frequency tolerance	f_{TOL}		0.03		%	On chip calibration
Calibration time			3		ms	
ADC						
Differential nonlinearity	DNL			1	LSB	10bit resolution mode
Integral nonlinearity	INL			2	LSB	10bit resolution mode
Signal-to-noise and distortion ratio	SINAD		70		dB	fin=1kHz, fS=16kHz
Effective Number of Bits	ENOB		10.5		bits	
Sampling frequency	Fs			200	ksps	

15.5 SPI characteristics

Table 15- 5 SPI characteristics

(over process, voltage 1.9~3.6V, and T=-40~+85℃)

Item	Sym.	Min	Typ.	Max	Unit	Condition
CK frequency	F_{CK}			4	MHz	Slave
CK duty cycle clock			50		%	Master
DI setup time		30			ns	Slave
		90			ns	Master
DI hold time		10			ns	Slave
		90			ns	Master
CK low to DO valid time				30	ns	Slave
				120	ns	Master
CN setup time		60			ns	Master/Slave
CN high to DI tri-state*[4]					ns	Master

[4] Note: Master actively stops reading during transmission, and Slave releases its driver DO and turns to tri-state.

15.6 I2C characteristics

Table 15- 6 I2C characteristics

(over process, voltage 1.9~3.6V, and T=-40~+85℃)

Item	Sym.	Standard mode		Fast mode		Unit	Condition
		Min	Max	Min	Max		
SCL frequency	F_{SCL}		100		400	kHz	
Rise time of SDA and SCL signals	T_R		1000		300	ns	
Fall time of SDA and SCL signals	T_F		300		300	ns	
START condition hold time	$T_{HD;STA}$	4		0.6		us	
Data hold time	$T_{HD;DAT}$	0	3.45		0.9	us	
Data setup time	$T_{SU;DAT}$	250		100		ns	
STOP condition setup time	$T_{SU;STO}$	4		0.6		us	

15.7 Flash characteristics

Table 15- 7 Flash memory characteristics

(T= -40℃~85℃)

Item	Sym.	Min	Typ.	Max	Unit	Condition
Retention period		20			year	
Number of erase cycles		100k			cycle	
VDD for programming		1.65		2.0	V	Note this refers to the SoC supply
Sector size			4		kB	
Page programming time	TPP		1.6	6	ms	
Sector erase time	TSE		150	500	ms	
Block erase time (32kB/64kB)	TBE		0.5/0.8	2.0/3.0	s	
Program current	I_P			10	mA	
Erase current	I_E			10	mA	

16 Public Key Engine

The TLSR8355 embeds Public Key Engine Standard Performance acceleration module and this section describes its function and use.

16.1 Calculation Model Overview

PKE (Public Key Engine) is specifically designed to accelerate large digital-to-analog operations in public key cryptographic operations. PKE SP-ECC is a version optimized for the elliptic curve algorithm. In this version, the following features are available.

- ✧ Support different bit width ECC (prime field): 192, 256 bits
- ✧ Support curve parameters: NIST p192, NIST p256, X25519, EdDSA

16.2 Function description

16.2.1 Module description

There are a large number of large digital-to-analog operations in public key cryptographic operations. PKE is designed to accelerate large digital-to-analog operations involved in RSA and Elliptic Curve Cryptography (ECC) operations in public key cryptography. Recently PKE can directly complete modular exponentiation in RSA and point multiplication in ECC. The CPU can query the operation of the PKE by polling or interrupting. The PKE includes one program memory unit (ROM), one instruction arithmetic unit (IEU), one 32-bit arithmetic unit (ALU), two pseudo-double-ended data RAMs, one register combination with interface module. According to different register configurations, PKE can perform the following operations with different precisions:

- ✧ ECC (Prime field): 192 and 256 bits

  In addition, the calculation of the PKE is finished in the form of Microcode and the Microcode is stored in the program storage unit. Therefore, different kind of public key cryptographic calculations can be implemented by pouring different microcode into the program storage unit. For instance, a high security public key algorithm instruction can be injected into a program storage unit in the PKE module in a SoC with high security requirements. Certainly these arithmetic instructions can be written to the ROM with a large program memory unit capacity. The CPU makes real-time calls according to different usage scenarios. The full microcode size is approximately 2KB.

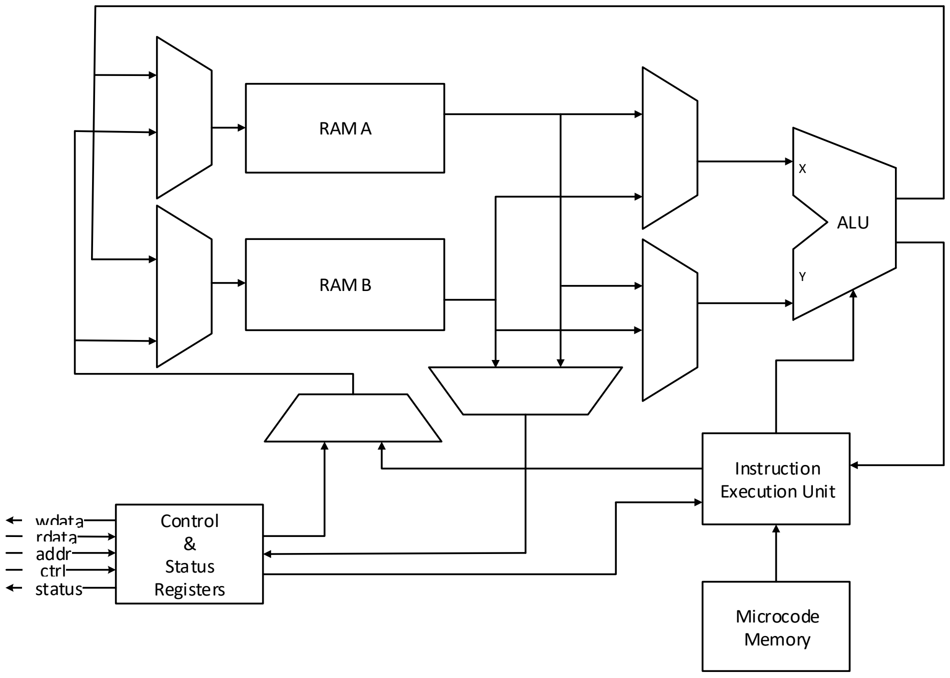


Figure 16- 1 Block diagram of PKE SP module

16.2.2 Software interface (Programming model)

The interfaces of the PKE SP are all mapped into the 7KB address space. The block of address mapping space mainly contains all the operands that the CPU can access. These operands contain modulus, power exponents, partial intermediate variables, and so on. In addition to this, the address map also contains control and status registers. The CPU can configure and monitor the PKE module through these control and status registers.

In the operations supported by PKE, the operands are also 192 bits at minimum. Therefore, it will encounter the problem of big-endian and little-endian when putting data into data RAM in the CPU or DMA. In the PKE module, words are arranged following an order of little-endian.

In PKE, the smallest operand is 32 bits (1 word), because the current ALU bit width input is 32 bits. If the operand is not word aligned, the high bit needs to be filled as 0.

After the PKE receives the start command, it starts the operation. During the operation, the host computer can query the current running state through the status register, or interrupt the current operation through the control register. In addition, the result of partial intermediate operations can be obtained by accessing the data RAM address.

The host computer can obtain the result of target operation finish by PKE through polling or interrupting. Data RAM supports word aligned and does not support byte alignment.

Table 16- 1 Dual port ram address map

	ECC		
Operand	256 bits	512 bits	1024 bits
First Address			
A0	0x0400	0x0400	0x0400
A1	0x0424	0x0444	0x0484
A2	0x0448	0x0488	0x0508
A3	0x046C	0x04CC	0x058C
A4	0x0490	0x0510	0x0610
A5	0x04B4	0x0554	0x0694
A6	0x04D8	0x0598	0x0718
A7	0x04FC	0x05DC	0x079C
A8	0x0520	0x0620	0x0820
A9	0x0544	0x0664	0x08A4
B0	0x1000	0x1000	0x1000
B1	0x1024	0x1044	0x1084
B2	0x1048	0x1088	0x1108
B3	0x106C	0x10CC	0x118C
B4	0x1090	0x1110	0x1210
B5	0x10B4	0x1154	0x1294
B6	0x10D8	0x1198	0x1318
B7	0x10FC	0x11DC	0x139C
B8	0x1120	0x1220	0x1420
B9	0x1144	0x1264	0x14A4

The above table shows the address assignment of two RAMs in ECC mode. The operand registers are distributed in two blocks of data RAM, using the prefixes A and B to distinguish the two blocks of RAM. The addresses listed in the table are all CPU addressable addresses, RAM A has an address offset of 0x400, and RAM B has an address offset of 0x1000. The actual space used by RAM will be larger than the space listed in the table and some intermediate variable storage is not open to the CPU. Data will be stored in the mode of little-endian in RAM.

16.3 Register description

16.3.1 Register map

Table 16- 2 Register map

Address	Mnemonic	Type	Description	Reset Value
0x2000	CTRL	W1S	[0] Go Start signal. When write 1 to the byte, the PKE will start running in the next clock cycle. The operation of the PKE is based on the configuration of the control registers and data registers for that clock cycle written as 1. [7:1] Rsvd	0x00
0x2001	CTRL	RO	[15:8] Rsvd	0x00
0x2002	CTRL	W1S	[16] Stop Stop signal. When write 1 to the byte, PKE will stop in the next clock cycle. [23:17] Rsvd	0x00
0x2003	CTRL	RO	[31:24] Rsvd	0x00
0x2004	CONF	RO	[7:0] Rsvd	0x00
0x2005	CONF	RW	[8] IRQEN Interrupt enable. When the bit is set as 1, the o_irq interface is valid. Regardless of whether the bit is set as 1, the STAT register is not affected by it. [15:9] Rsvd	0x00
0x2006	CONF	RW	[23:16] Partial_Radix Select part of BASE_RADIX to determine the bit width that the operation really needs to use during the operation. The value of this field indicates the number of words, and the bit width of the operand is PARTICAL_RADIX*32 bits. For example, if BASE_RADIX=2, PARTIAL_RADIX=6, then the bit width of the operand is (6/ (256/32))*256=192. If the operations of ECC-192 need to be processed, BASE_RADIX and PARTIAL_RADIX should be configured as shown in this example. When using operands of other bit widths, configure BASE_RADIX and PARTIAL_RADIX according to the above formula.	0x00
0x2007	CONF	RW	[31:27] Rsvd [26:24] Base_Radix This field indicates the bit width	0x02

Address	Mnemonic	Type	Description	Reset Value
			cardinality at which the operation is performed. At the same time, the cardinality also represents the space required for the operand to be stored in the data RAM. For ECC point operations, the value of this field should be 2. 2: 256 bits Others: Reserved	
0x2010	MC_PTR	RW	[7:0] ADDR This field indicates the address of the next instruction to be executed by the PKE. This register can only be rewritten when the PKE is not working. Any write operation while the PKE is operating will be ignored. This field is also updated in real time when running the PKE and always pointing to the address of the instruction that will be executed next. Therefore, this register can also be combined with CTRL.STOP for debugging. It should be noted that the instructions are all word aligned. Therefore, the lowest 2 bits of the field are 0. When writing an instruction address to this field, it is limited to the address range of 0x00~0x2F. The written address will proceed “And” Operation with a mask, therefore ignoring the upper 6 bits.	0x00
0x2011	MC_PTR	RW	[11:8] ADDR See above description for [7:0] [15:12] Rsvd	0x00
0x2012	MC_PTR	RO	[23:16] Rsvd	0x00
0x2013	MC_PTR	RO	[31:24] Rsvd	0x00
0x2020	STAT	W1C	[0] Done When the bit is set to 1, it indicates that the operation ends. When this bit is set as 1 from external, the bit is cleared. In addition, this bit also acts as a clear bit for the external interrupt. When the bit is high as CTRL.IRQEN is active, the external interrupt signal is also pulled high. To write 1 from external, the external interrupt is also cleared. [7:1] Rsvd	0x00

Address	Mnemonic	Type	Description	Reset Value
0x2021	STAT	RO	[15:8] Rsvd	0x00
0x2022	STAT	RO	[23:16] Rsvd	0x00
0x2023	STAT	RO	[31:24] Rsvd	0x00
0x2024	RT_CODE	RO	[3:0] STOP_LOG This field is used to indicate the reason when the PKE stopped. If the PKE stopped because the operation is complete, the value of this field is 0. If the value of this field is non-zero, then it proves that PKE operation is not completed, maybe encountering some exceptions and then external processing is required, the result is not available. 0: Normal stop 1: Received a termination request (CTRL.STOP is high) 2: No valid modulo inverse 3: Point is not on the curve (CTRL.CMD:PVER) 4: Invalid Microcode Others: Reserved [7:4] Rsvd	0x00
0x2025	RT_CODE	RO	[15:8] Rsvd	0x00
0x2026	RT_CODE	RO	[23:16] Rsvd	0x00
0x2027	RT_CODE	RO	[31:24] Rsvd	0x00
0x2050	EXE_CONF	RW	[0] IAFF_R0 The input form of R0 is affine coordinate system enabled, this bit is only valid for ECC operations. When the bit is high, the input point is a point on the affine coordinate system. When the bit is low, the input point is a point on the Jacobian coordinate system. When it comes to the modular multiplication, if the bit is low, it will first convert the number on its scope to the Jacobian coordinate system before computing. [1] IMON_R0 The input form of R0 is Montgomery enabled. When the bit is high, data will be input in the form of Montgomery. When the bit is low, data will be input in the normal form.	0x2a

Address	Mnemonic	Type	Description	Reset Value
			When it comes to the modular multiplication, if the bit is low, it will first convert the number on its scope to the Montgomery form before computing. [2] IAFF_R1 The input form of R1 is affine coordinate system enabled, this bit is only valid for ECC operations. When the bit is high, the input point is a point on the affine coordinate system. When the bit is low, the input point is a point on the Jacobian coordinate system. When it comes to the modular multiplication, if the bit is low, it will first convert the number on its scope to the Jacobian coordinate system before computing. [3] IMON_R1 The input form of R1 is Montgomery enabled. When the bit is high, data will be input in the form of Montgomery. When the bit is low, data will be input in the normal form. When it comes to the modular multiplication, if the bit is low, it will first convert the number on its scope to the Montgomery form before computing. [4] OAFF The output form is affine coordinate system enabled, this bit is only valid for ECC operations. When the bit is high, the output point is a point on the affine coordinate system. When the bit is low, the output point is a point on the Jacobian coordinate system. [5] OMON The output form is Montgomery enabled. When the bit is high, the output is in the form of Montgomery. When the bit is low, the output is in the normal form.	

Address	Mnemonic	Type	Description	Reset Value
			[7:6] Rsvd	
0x2051	EXE_CONF	RW	[9:8] ME_SCA_EN The secure modular exponentiation algorithm selects a signal that is valid only for modular exponentiation in RSA operations. 00: The secure modular exponentiation algorithm requires a public key and a private key. The exponentiation index is register B1 under the algorithm. 01: The secure modular exponentiation algorithm requires a private key. The exponentiation index is register B1 under the algorithm. 10: Montgomery stepwise modular exponentiation algorithm. 11: Non-secure modular exponentiation requires a public key. Under this algorithm, the exponentiation index is register A1. Among them, the decryption and signature of the RSA can only use the secure modular exponentiation algorithm. There are two different algorithms of selecting 01 or 10 according to whether using a public key. For RSA encryption and verification, non-secure modular exponentiation algorithm can be used. [15:10] Rsvd	0x00
0x2052	EXE_CONF	RO	[23:16] Rsvd	0x00
0x2053	EXE_CONF	RO	[31:24] Rsvd	0x00
0x2080	VERSION	RO	[3:0] MIR Secondary version number [7:4] MAR Main version number	0x00
0x2081	VERSION	RO	[15:8] Rsvd	0x00
0x2082	VERSION	RO	[23:16] PROJECT Project number	0x00
0x2083	VERSION	RO	[31:24] PROJECT Project number	0x00
0x2400~0x2E10	MEM_REGION_A	RW	[31:0] DATA_A This field is used to store operational data.	
0x3000~0x4A10	MEM_REGION_B	RW	[31:0] DATA_B This field is used to store operational data.	

17 True Random Number Generator (TRNG)

17.1 Model Overview

This section describes the function and its use of the True Random Number Generator Module.

The random number generator module contains entropy source and post processing (DRBG). The entropy source is designed using RO. The top block diagram of the random number generator is shown in 错误!未找到引用源。.

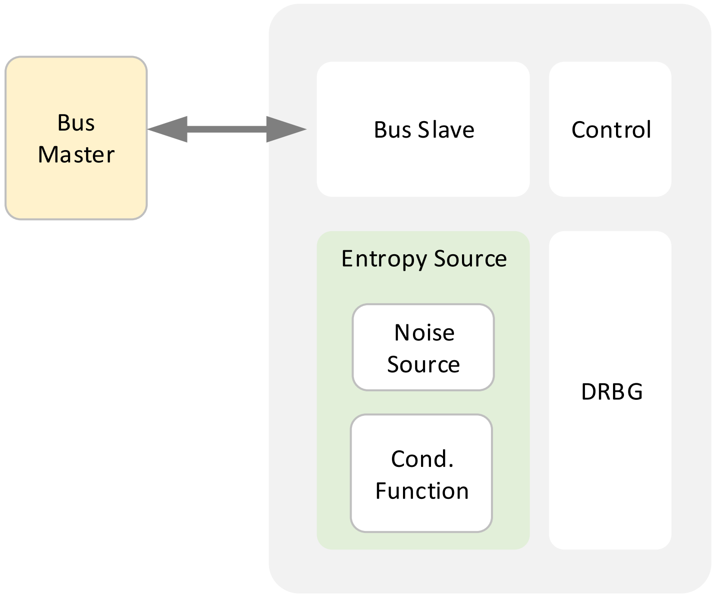


Figure 17- 1 Module Boundary

17.2 Register description

Table 17- 1 Register Map

Address	Mnemonic	Type	Description	Reset Value
0x4000	TRNG_CR	RW	[0] RBGEN Random bit generator enable. [4:1] ROSEN Each bit states enable for one RO SOURCE. From RO SOURCE1 to RO SOURCE4. [7:5] Rsvd	0x1f
0x4001	TRNG_CR	RO	[15:8] Rsvd	0x00
0x4002	TRNG_CR	RW	[16] Rsvd [17] DIEN Data interrupt enable. 0: Data interrupt is disabled. 1: Data interrupt is enabled. [18] ERIEN Empty read interrupt enable.	0x02

Address	Mnemonic	Type	Description	Reset Value
			0: Empty read interrupt is disabled. 1: Empty read interrupt is enabled. [23:19] Rsvd	
0x4003	TRNG_CR	RW	[24] IRQEN Global interrupt enable. Bit 16~19 active only when this bit is 1. 0: Disable interrupt function 1: Enable interrupt function [31:25] Rsvd	0x01
0x4004	RBG_RTCR	RW	[0] MSEL Mode select. 0: TRBG without post-processing. 1: TRBG with post-processing. [7:1] Rsvd	0x00
0x4005	RBG_RTCR	RO	[15:8] Rsvd	0x00
0x4006	RBG_RTCR	RO	[23:16] Rsvd	0x00
0x4007	RBG_RTCR	RO	[31:24] Rsvd	0x00
0x4008	RBG_SR	W1C	[0] Rsvd [1] DRDY Data ready. [2] ERERR Empty read error. [7:3] Rsvd	0x00
0x4009	RBG_SR	RO	[15:8] Rsvd	0x00
0x400a	RBG_SR	RO	[23:16] Rsvd	0x00
0x400b	RBG_SR	RO	[31:24] Rsvd	0x00
0x400c	RBG_DR	RO	[7:0] Rsvd	0x00
0x400d	RBG_DR	RO	[15:8] Rsvd	0x00
0x400e	RBG_DR	RO	[23:16] Rsvd	0x00
0x400f	RBG_DR	RO	[31:24] Rsvd	0x00
0x4010	RBG_VERSION	RO	[3:0] MIR Sub version number. [7:4] MAR Main version number.	0x00
0x4011	RBG_VERSION	RO	[15:8] Rsvd	0x00
0x4012	RBG_VERSION	RO	[23:16] PROJECT PROJECT number.	0x00
0x4013	RBG_VERSION	RO	[31:24] PROJECT PROJECT number.	0x00
0x4020	RBG_FIFO_CR	RW	[2:0] DFTV DRNG FIFO count threshold value. DRDY interrupt will be generated when actual TRBG FIFO count exceeds this threshold. E.g if set to 5, an interrupt will be generated when the actual FIFO count transits from 4 to 5.	0x07

Address	Mnemonic	Type	Description	Reset Value
			[7:3] Rsvd	
0x4021	RBG_FIFO_CR	RO	[15:8] Rsvd	0x00
0x4022	RBG_FIFO_CR	RW	[18:16] TFTV TRNG FIFO count threshold value. DRDY interrupt will be generated when actual TRBG FIFO count exceeds this threshold. E.g. if set to 5, an interrupt will be generated when the actual FIFO count transits from 4 to 5. [23:19] Rsvd	0x07
0x4023	RBG_FIFO_CR	RO	[31:24] Rsvd	0x00
0x4024	RBG_FIFO_SR	RO	[7:0] DFCNT DRBG FIFO count. Current number of random number in TRBG FIFO.	0x00
0x4025	RBG_FIFO_SR	RO	[8] DFE DRBG FIFO empty. [15:9] Rsvd [18:16] TFCNT TRBG FIFO count. Current number of random number in TRBG FIFO.	0x01
0x4026	RBG_FIFO_SR	RO	[23:19] TFCNT TRBG FIFO count. Current number of random number in TRBG FIFO.	0x00
0x4027	RBG_FIFO_SR	RO	[24] TFE TRBG FIFO empty. [31:25] Rsvd	0x01
0x4080	RO_CR1	RW	[7:0] ROEN2 RO enable of RO SOURCE2. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 2.	0xff
0x4081	RO_CR1	RW	[15:8] ROEN2 RO enable of RO SOURCE2. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 2.	0xff
0x4082	RO_CR1	RW	[23:16] ROEN1 RO enable of RO SOURCE1. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 1.	0xff
0x4083	RO_CR1	RW	[31:24] ROEN1 RO enable of RO SOURCE1. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 1.	0xff
0x4084	RO_CR2	RW	[7:0] ROEN4	0xff

Address	Mnemonic	Type	Description	Reset Value
			RO enable of RO SOURCE4. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 4.	
0x4085	RO_CR2	RW	[15:8] ROEN4 RO enable of RO SOURCE4. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 4.	0xff
0x4086	RO_CR2	RW	[23:16] ROEN3 RO enable of RO SOURCE3. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 3.	0xff
0x4087	RO_CR2	RW	[31:24] ROEN3 RO enable of RO SOURCE3. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 3.	0xff

17.3 Interrupt description

The RBG module has the following interrupt sources:

- ✧ CPU reads RBG_DR without data
- ✧ Data valid

The above interrupts can be set by RBG_CR. By default, the data valid interrupt is enabled.

When the RBGEN of RBG_CR is low, the interrupt signal will not be cleared. Therefore, before enabling RBGEN, it is necessary to ensure that there is no previous interrupt signal, otherwise it will affect the next interrupt.

17.3.1 CPU reads RBG_DR without data

In order to prevent the CPU from reading the invalid data, the RBG can remind the CPU to read in such a situation when there is no valid random number. In order to avoid the CPU reading the empty data, it is recommended to read the RBG_FIFO_SR first every time to get the random number before the CPU gets data in the current FIFO to avoid invalid data.

The CPU can clear the interrupt by writing 1 to ERERR in RBG_SR. If the write is successful, the interrupt will be cleared. When the above situation occurs again, the interrupt will be valid again.

17.3.2 Data valid

RBG provides two ways to output data. When the interrupt is enabled, the random number can be read by the way of interrupting. In this design, the data in the corresponding FIFO will only be pulled up after the threshold is reached, thus the CPU can obtain multiple data at once. The threshold can be set by RBG_FIFO_CR. The CPU can clear the interrupt by writing 1 to DRDY of RBG_SR. If the write is successful, the interrupt will be pulled down. The interrupt is pulled high again when the data in the FIFO reaches the threshold again.

It is important to note that the interrupt will only be pulled up when the amount of data in the FIFO reaches the threshold. Therefore, the data in the FIFO exceeds the threshold firstly and then RBG module pulls up the interrupt. When the CPU doesn’t obtain data or have obtained data but the amount of data remaining in the FIFO is still larger than the threshold, then clear the interrupt. Although the data in the FIFO is still larger than the threshold, it will not be interrupted.

In addition, the CPU can use the RBG_FIFO_SR register to view the remaining data in the FIFO. It can also use this method to obtain a random number. Check the RBG_FIFO_SR register when the random number is needed and the number of random numbers indicated by the register can be fetched at one time. If the rate at which the CPU handles random numbers is slower than the rate at which RBG random numbers are generated, it is generally not recommended to use interrupt to obtain random numbers.

17.4 Usage Procedure

17.4.1 Normal Operation

Turn off the RBG module first after the CPU works normally, that is to set RBGEN of the RBG_CR to 0. Then it can be configured and write 1 to RBGEN after the configuration is complete to make it work normally.

The CPU can configure RBG module by configuring RBG_CR, RBG_FIFO_CR and other optional configuration registers. For detailed configuration instructions, please refer to the description in Section 17.2.

When writing 1 to RBGEN in RBG_CR, the modification of the value of the above register will not affect the RBG. Therefore, when configuring, set the RBGEN in the RBG_CR register after configuring other registers to enable the OSR_RBG module.

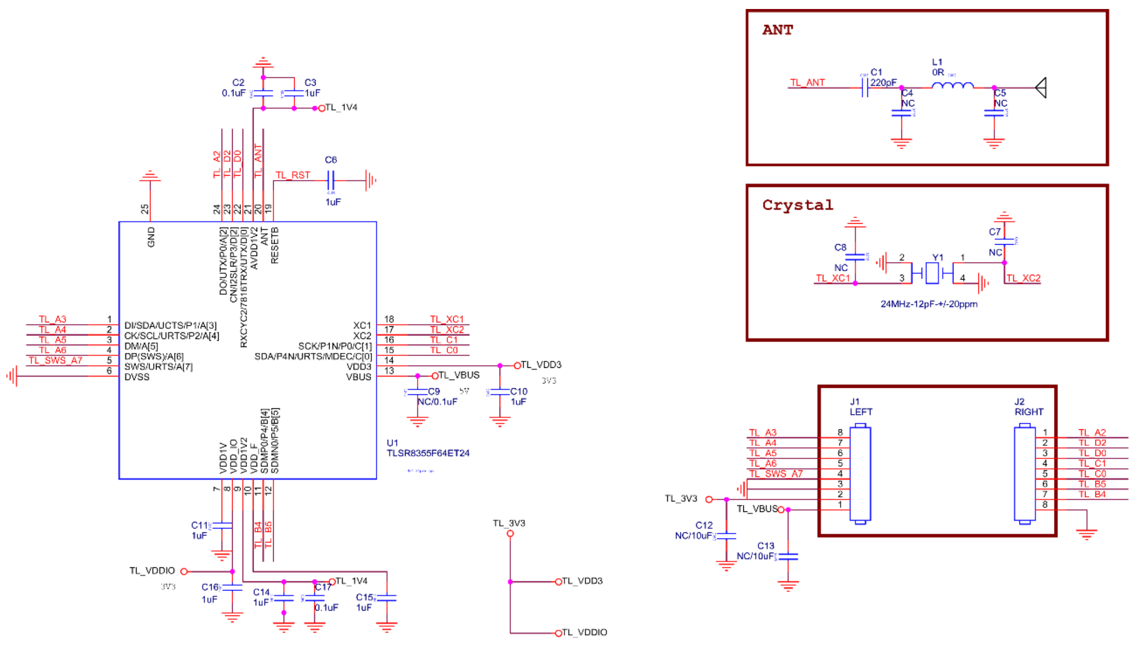
TRBG and DRBG can be switched by modifying RBG_RTCR during the operation to meet different usage environments.

17.4.2 Entropy source

In this design, the random number generator module uses RO RNG as the entropy source. RO RNG contains modules such as random source and post-processing. RO RNG has four independent RO entropy sources. Each entropy source can choose to use its own RO CLK as the sampling clock or select the system clock as the sampling clock. The selection is determined by the input of I_rbg_sclk_sel, which is high for the system clock and low for the internal RO CLK. All RO enable signals are open at the same time and some of the ROs can be turned on or off by controlling the register.

18 Reference Design

18.1 Schematic for TLSR8355F64ET24



Power supply	
TL_3V3	C9=C13=NC
TL_VBUS	C9=0.1uF;C13=10uF

Figure 18- 1 Schematic

18.2 BOM (Bill of Material)

Table 18- 1 BOM Table

Quantity	Designator	Value	Description	PCB Footprint
1	C1	220pF	Capacitance,X7R,±10%	0402
2	C2,C17	0.1uF	Capacitance,X5R,±10%	0402
7	C3,C6,C10,C11 ,C14,C15,C16	1uF	Capacitance,X5R,±10%	0402
4	C4,C5,C7,C8	N.C.	Not mounted	0402
1	C9	N.C./0.1uF	Capacitance,X5R,±10%	0402
2	C12,C13	N.C./10uF	Capacitance,X5R,±10%	0402
1	J1	LEFT	Pin headers	hdr254f-1x8x850
1	J2	RIGHT	Pin headers	hdr254f-1x8x850
1	L1	0R	Resistance,5%	0402

Quantity	Designator	Value	Description	PCB Footprint
1	U1	TLSR8355F 64ET24	2.4G	QFN-24
1	Y1	24MHz	XTAL SMD 3225,24 MHz,Cl=12pF,total tol.±20ppm	XTAL_3225