# Functional Overview

- High performance and low power consumption 8-bit LGT8XM core
- Advanced RISC architecture
  - 131 instructions, more than 80% are executed in a single cycle
  - 32x8 general working register
  - Up to 32MIPS execution efficiency when working at 32MHz
  - Internal single cycle multiplier (8x8)
- Non-volatile program and data storage space
  - 32Kbytes on-chip online programmable FLASH program memory
  - 2Kbytes internal data SRAM
  - Programmable E2PROM Analogue interface, support byte access
  - Brand new program encryption algorithm to ensure user code security
- Peripheral Controller
  - Two 8-bit timers with independent pre-scaler, support compare output mode
  - Two 16-bit timers with independent pre-scalers, support input capture and compare output
  - Internal 32KHz calibratable RC oscillator to realize real-time counter function
  - Supports up to 9 PWM outputs, three groups of complementary programmable dead zone control
  - 12-channel 12-bit high-speed analogue-to-digital converter (ADC)
    - Optional internal and external reference voltage
    - Programmable gain (X1/8/16/32) differential amplifier input channel
    - Automatic threshold voltage monitoring mode
  - Two analogue comparators (AC), support expansion from ADC input channel
  - Internal 1.024V/2.048V/4.096V ±1% calibratable reference voltage source
  - An 8-bit programmable DAC that can be used to generate a reference voltage source
  - Programmable Watchdog Timer (WDT)
  - Programmable synchronous/asynchronous serial interface (USART/SPI)
  - Synchronous Peripheral Interface (SPI), programmable master/slave working mode
  - Two-wire serial interface (TWI), compatible with I2C master-slave mode
  - 16-bit digital operation acceleration unit (DSC), supports direct 16-bit data access
- Special processor function
  - SWD two-wire on-chip debugging/mass production interface
  - External interrupt source and I/O interrupt on change support
  - Built-in power-on reset circuit (POR) and programmable low voltage detection circuit (LVD)
  - Built-in 1% calibrated 32MHz RC oscillator, supports frequency multiplication output
  - Built-in 1% calibrated 32KHz RC oscillator
  - External support 32.768KHz and 400K~32MHz crystal input
  - 6x high current push-pull drive IO, support high-speed PWM application

LogicGreen

**8-bit LGT8XM**

RISC Microcontroller with In-System Programmable FLASH Memory

**LGT8F88P**

**LGT8F168P**

**LGT8F328P**

Data book

Version 1.0.5

Application field

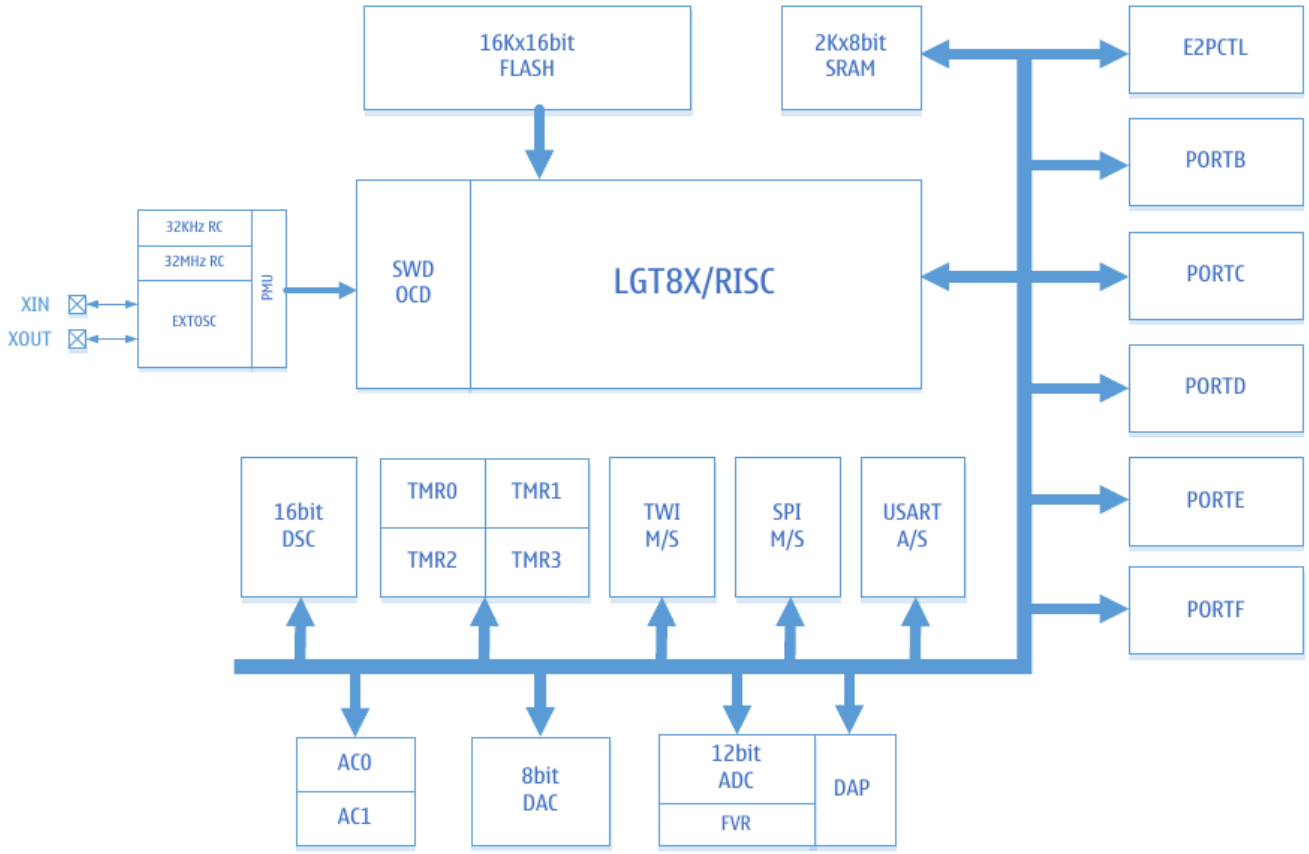
Home appliances

Motor drive

Automatic control

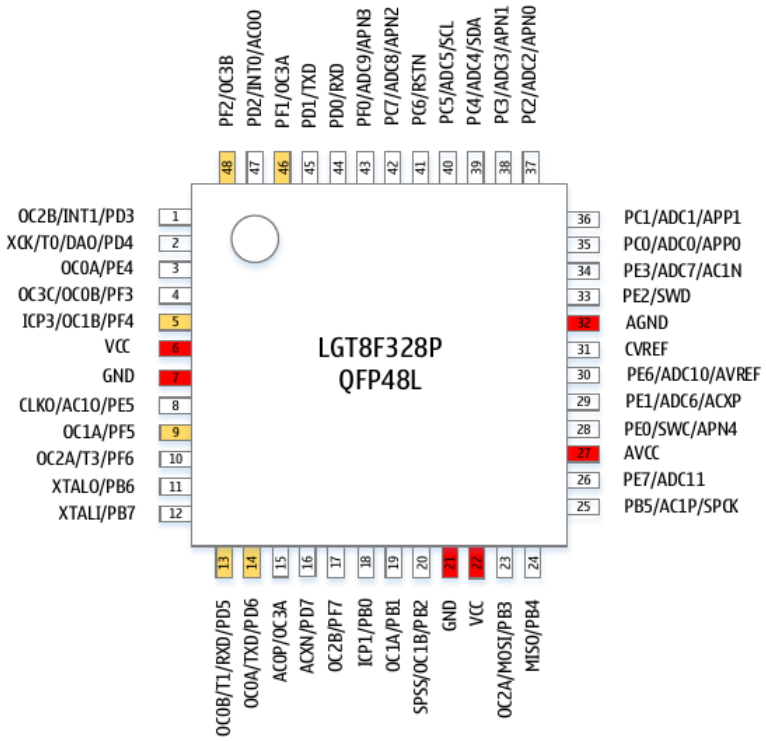
- I/O and package: QFP48/32L, SSOP20L
- Lowest power consumption: 1uA@3.3V
- Working environment
  - Voltage: 1.8V ~ 5.5V
  - Frequency: 0 ~ 32MHz
  - Temperature: -40C ~ +85C
  - HBM ESD: > 4KV

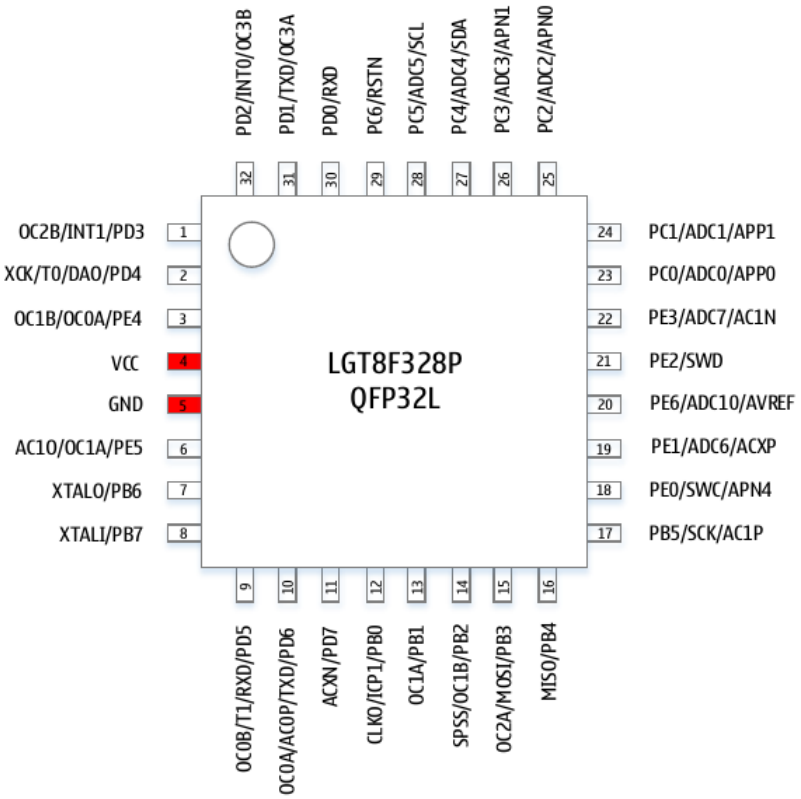
# System framework

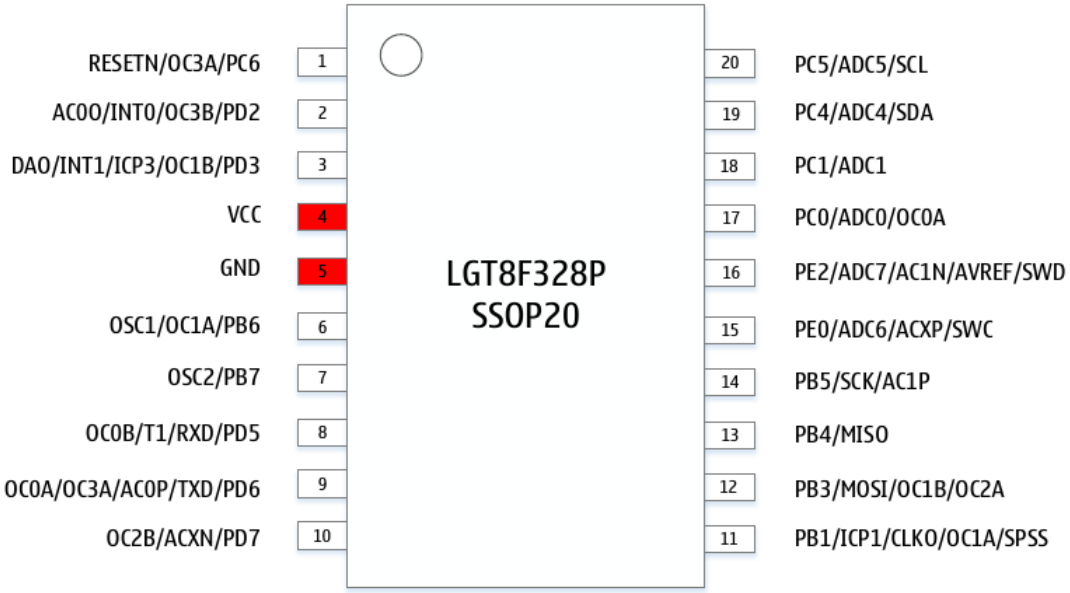



| Module name | Module function |
|---|---|
| SWD | Debug module, realize online debugging and ISP function at the same time |
| LGT8X | 8bit high-performance RISC core |
| E2PCTL | Data FLASH access interface controller |
| PMU | Power management module, responsible for managing the transition between the working states of the system |
| PORT B/C/D/E/F | Universal programmable input and output ports |
| DSC | 16-bit digital operation acceleration unit |
| ADC<br>DAP<br>IVREF | 8-channel 12-bit analogue-to-digital converter<br>Programmable gain differential amplifier<br>1.024V/2.048V/4.096V internal reference |
| AC0/1 | Analogue comparator |
| TMR0/1/2/3 | 8/16-bit timer/counter, PWM controller |
| WDT | Watchdog reset module |
| SPI M/S | Master-slave SPI controller |
| TWI M/S | Master-slave two-wire interface controller, compatible with I2C protocol |
| USART | Synchronous/asynchronous serial transceiver |
| DAC | 8-bit digital-to-analogue converter |

# Package definition




RESETN/OC3A/PC6
1
AC0O/INT0/OC3B/PD2
2
DAO/INT1/ICP3/OC1B/PD3
3
VCC
4
GND
5
OSC1/OC1A/PB6
6
OSC2/PB7
7
OC0B/T1/RXD/PD5
8
OC0A/OC3A/AC0P/TXD/PD6
9
OC2B/ACXN/PD7
10
LGT8F328P
SSOP20
20
PC5/ADC5/SCL
19
PC4/ADC4/SDA
18
PC1/ADC1
17
PC0/ADC0/OC0A
16
PE2/ADC7/AC1N/AVREF/SWD
15
PE0/ADC6/ACXP/SWC
14
PB5/SCK/AC1P
13
PB4/MISO
12
PB3/MOSI/OC1B/OC2A
11
PB1/ICP1/CLKO/OC1A/SPSS

# Pin description

In the LGT8FX8P series package, the QFP48L package leads to all pins. Other packages are produced by binding multiple internal I/Os to one pin on the basis of QFP48. Pay special attention when configuring the pin direction. The following table lists the binding of various package pins:

| QFP48 | QFP32 | SSOP20 | Function Description |
|---|---|---|---|
| 01 | 01 | 03 | PD3/INT1/OC2B* |
| | | | PD3: Programmable port D3<br>INT1: External interrupt input 1<br>OC2B: Timer 2 compare match output B |
| 02 | 02 | | PD4/DAO/T0/XCK |
| | | | PD4: Programmable port D4<br>DAO: Internal DAC output<br>T0: Timer0 external clock input<br>XCK: USART synchronous transmission clock |
| 03 | 03 | — | PE4/0C0A* |
| | | | PE4: Programmable port E4<br>OC0A: Timer 0 compare match output A |
| 04 | — | — | PF3/OC3C/OC0B* |
| | | | PF3: Programmable port F3<br>OC3C: Timer 3 compare match output C<br>OC0B: Timer 0 compare match output B |
| 05 | 03 | 03 | PF4/OC1B*/ICP3 |
| | | | PF4: Programmable port F4<br>OC1B: Timer 1 compare match output B<br>ICP3: Timer 3 capture input |
| 06 | 04 | 04 | VCC |
| 07 | 05 | 05 | GND |
| 08 | 06 | — | PE5/AC1O/CLKO* |
| | | | PE5: Programmable port E5<br>C1O: Analogue comparator AC1 output<br>CLKO: System clock output |
| 09 | | 06 | PF5/OC1A* |
| | | | PF5: Programmable port F5<br>OC1A: Timer 1 compare match output A |
| 10 | — | — | PF6/T3/OC2A* |
| | | | PF6: Programmable port F6<br>T3: Timer 3 external clock input<br>OC2A: Timer 2 compare match output A |
| 11 | 07 | 06 | PB6/XTALO |
| | | | PB6: Programmable port B6<br>XTALO: Crystal IO output port |

| | | | |
|---|---|---|---|
| **12** | **08** | **07** | PB7/XTALI |
| | | | PB7: Programmable port B7<br>XTALI: Crystal IO input port |
| **13** | **09** | **08** | PD5/RXD*/T1/OC0B |
| | | | PD5: Programmable port D5<br>RXD: USART data reception (optional)<br>T1: Timer 1 external clock input<br>OC0B: Timer 0 compare match output B |
| **14** | **10** | **09** | PD6/TXD*/OC0A |
| | | | PD6: Programmable port D6<br>TXD: USART data transmission (optional)<br>OC0A: Timer 0 compare match output A |
| **15** | | | AC0P/0C3A |
| | | | AC0P: Analogue comparator 0 positive input<br>OC3A: Timer 3 compare match output A |
| **16** | **11** | **10** | PD7/ACXN |
| | | | PD7: Programmable port D7<br>ACXN: Analogue comparator 0/1 common negative input |
| **17** | **—** | | PF7/OC2B |
| | | | PF7: Programmable port F7<br>OC2B: Timer 2 compare match output B |
| **18** | **12** | **11** | PB0/ICP1 |
| | | | PB0: Programmable port B0<br>ICP1: Timer 1 capture input |
| **19** | **13** | | PB1/OC1A |
| | | | PB1: Programmable port B1<br>OC1A: Timer 1 compare match output A |
| **20** | **14** | **12** | PB2/OC1B/SPSS |
| | | | PB2: Programmable port B2<br>OC1B: Timer 1 compare match output B<br>SPSS: SPI slave mode chip select |
| **21** | **—** | **—** | GND |
| **22** | **—** | **—** | VCC |
| **23** | **15** | **12** | PB3/MOSI/OC2A |
| | | | PB3: Programmable port B3<br>MOSI: SPI master output/slave input<br>OC2A: Timer 2 compare match output A |
| **24** | **16** | **13** | PB4/MISO |
| | | | PB4: Programmable port B4<br>MISO: SPI master input/slave output |
| **25** | **17** | **14** | PB5/SPCK/AC1P |
| | | | PB5: Programmable port B5<br>SPCK: SPI clock signal<br>AC1P: Analogue comparator 1 positive input |

| | | | |
|---|---|---|---|
| **26** | **—** | **—** | PE7/ADC11<br>PE7: Programmable port E7<br>ADC11: ADC analogue input channel 11 |
| **27** | **—** | **—** | AVCC: Internal analogue circuit power supply |
| **28** | **18** | **15** | PE0/SWC/APN4<br>PE0: Programmable port E0<br>SWC: SWD debug interface clock<br>APN4: Differential amplifier reverse input channel 4 |
| **29** | **19** | | PE1/ADC6/ACXP<br>PE1: Programmable port E1<br>ADC6: ADC analogue input channel 6<br>ACXP: Analogue comparator 0/1 common positive terminal input |
| **30** | **20** | **16** | PE6/ADC10/AVREF<br>PE6: Programmable port E6<br>ADC10: ADC analogue input channel 10<br>AVREF: ADC external reference input |
| **31** | **—** | **—** | CVREF: ADC reference voltage output<br>Only used for external 0.1uF filter capacitor |
| **32** | **—** | **—** | AGND: Internal analogue circuit ground |
| **33** | **21** | **16** | PE2/SWD<br>PE2: Programmable port E2<br>SWD: SWD debug interface data line |
| **34** | **22** | | PE3/ADC7/AC1N<br>PE3: Programmable port E3<br>ADC7: ADC analogue input channel 7<br>AC1N: Analogue comparator negative input |
| **35** | **23** | **17** | PC0/ADC0/APP0<br>PC0: Programmable port C0<br>ADC0: ADC analogue input channel 0<br>APP0: Differential amplifier forward input channel 0 |
| **36** | **24** | **18** | PC1/ADC1/APP1<br>PC1: Programmable port C1<br>ADC1: ADC analogue input channel 1<br>APP1: Differential amplifier forward input channel 1 |
| **37** | **25** | **—** | PC2/ADC2/APN0<br>PC2: Programmable port C2<br>ADC2: ADC analogue input channel 2<br>APN0: Differential amplifier reverse input channel 0 |
| **38** | **26** | **—** | PC3/ADC3/APN1<br>PC3: Programmable port C3<br>ADC3: ADC analogue input channel 3<br>APN1: Differential amplifier reverse input channel 1 |

| | | | |
|---|---|---|---|
| **39** | **27** | **19** | PC4/ADC4/SDA<br>PC4: Programmable port C4<br>ADC4: ADC analogue input channel 4<br>SDA: I2C controller data line |
| **40** | **28** | **20** | PC5/ADC5/SCL<br>PC5: Programmable port C5<br>ADC5: ADC analogue input channel 5<br>SCL: I2C controller clock line |
| **41** | **29** | **1** | PC6/RESETN<br>PC6: Programmable port C6<br>RESETN: External reset input |
| **42** | **—** | **—** | PC7/ADC8/APN2<br>PC7: Programmable port C7<br>ADC8: ADC analogue input channel 8<br>APN2: Differential amplifier reverse input channel 2 |
| **43** | **—** | **—** | PF0/ADC9/APN3<br>PF0: Programmable port F0<br>ADC9: ADC analogue input channel 9<br>APN3: Differential amplifier reverse input channel 3 |
| **44** | **30** | **—** | PD0/RXD<br>PD0: Programmable port D0<br>RXD: USART data receiving input |
| **45** | **31** | **—** | PD1/TXD<br>PD1: Programmable port D1<br>TXD: USART data transmission output |
| **46** | | **1** | PF1/OC3A<br>PF1: Programmable port F1<br>OC3A: Timer 3 compare match output A |
| **47** | **32** | **2** | PD2/INT0/AC0O<br>PD2: Programmable port D2<br>INT0: External interrupt input 0<br>AC0O: Analogue comparison 0 output |
| **48** | | | PF2/OC3B<br>PF2: Programmable port F2<br>OC3B: Timer 3 compare match output B |

# Programming Manual

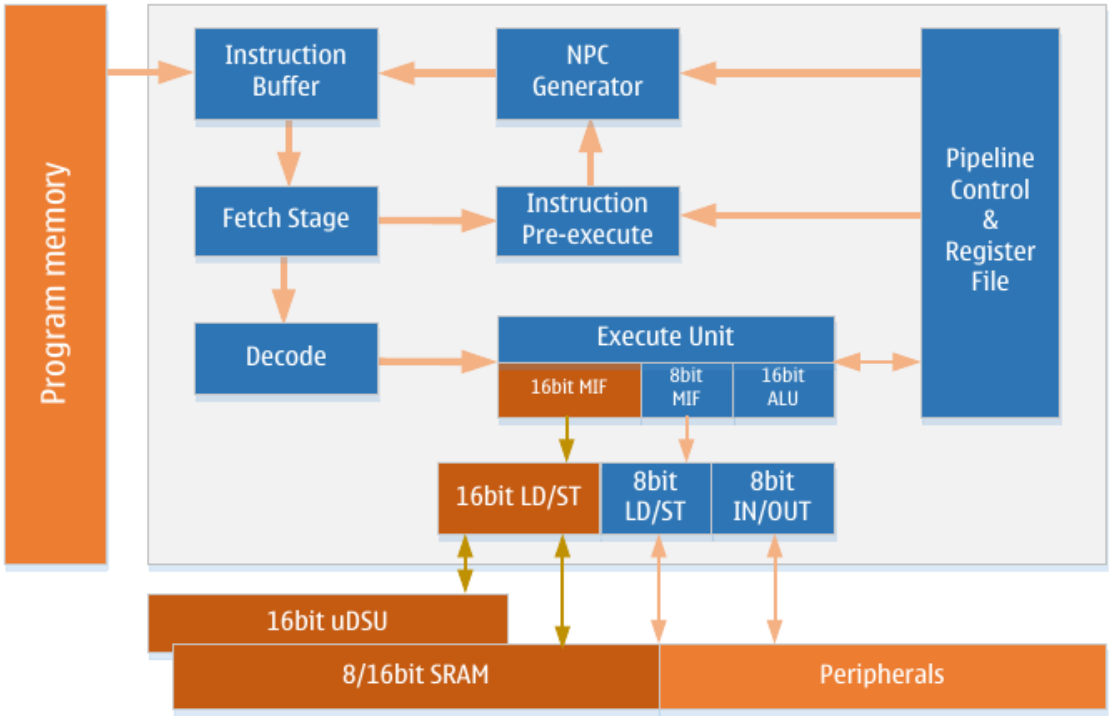
## LGT8XM core

- Low power design
- High-efficiency RISC architecture
- 16-bit LD/ST extension (uDSC dedicated)
- 130 instructions, of which more than 80% are single cycle
- Built-in online debugging (OCD) support

### Overview

This chapter mainly describes the LGT8XM core architecture and functions. The core is the brain of the MCU, responsible for ensuring the correctness of the program execution, so the core must be able to accurately perform calculations, control peripherals and handle various interrupts.

**The structure of the LGT8XM core**




In order to achieve greater efficiency and parallelism, the LGT8XM core adopts the Haval architecture – independent data and program buses. Instructions are executed through an optimized two-stage pipeline, which can reduce the number of inefficient instructions in the pipeline and reduce the amount of access to the FLASH program memory, thus reducing the power consumption of the core. At the same time, the LGT8XM core adds an instruction cache (where two instructions can be cached at the same time) in the front stage of the instruction fetch, and further reduces the access frequency to the FLASH program memory through the pre-execution module in the instruction fetch cycle; Thorough testing shows LGT8XM can reduce the access to FLASH by about 50% compared with other cores of similar architecture, which greatly reduces the operating power consumption of the system.

The LGT8XM core has 32 8-bit high-speed access general-purpose working registers (Register file), which helps to realize single-cycle arithmetic logic operations (ALU). Generally, the two operands of ALU operation come from the general working register, and the result of the ALU operation will also be written into the register file in one cycle.

Six of the 32 working registers are used for pairwise combination to form three 16-bit registers, which can be used for indirect addressing address pointers, used to access external storage space and FLASH program space. LGT8XM supports single-cycle 16-bit arithmetic operations, which greatly improves the efficiency of indirect addressing. The three special 16-bit registers in the LGT8XM core are named X, Y, and Z registers, which will be described in detail later.

ALU supports arithmetic and logic operations between registers and between constants and registers, and operations on a single register can also be performed in ALU. After the ALU operation is completed, the effect of the operation result on the state of the core is updated in the status register (SREG).

Program flow control is realized by conditional and unconditional jump/call, which can be addressed to all program areas. Most LGT8XM instructions are 16 bits. Each program address space corresponds to a 16-bit or 32-bit LGT8XM instruction.

After the core responds to an interrupt or subroutine call, the return address (PC) is stored in the stack. The stack is allocated in the general data SRAM of the system, so the size of the stack is only limited by the size and usage of the systems SRAM. All applications that support interrupts or subroutine calls must first initialize the stack pointer register (SP), which can be accessed through the IO space. The data SRAM can be accessed through 5 different addressing modes. The internal storage space of LGT8XM is linearly mapped to a unified address space. For details, please refer to the introduction in the storage chapter.

The LGT8XM core contains a flexible interrupt controller, the interrupt function can be controlled by a global interrupt enable flag in the status register. All interrupts have an independent interrupt vector. The priority of the interrupt corresponds to the interrupt vector address. The smaller the interrupt address, the higher the priority of the interrupt.

The I/O space contains 64 register spaces that can be directly addressed by IN/OUT instructions. These registers show the control functions of the core control and status registers, SPI and other I/O peripherals. This part of the space can be accessed directly through IN/OUT instructions, or through their addresses mapped to the data memory space (0x20-0x5F). In addition, the LGT8FX8P also contains extended I/O space, wich are mapped to data storage space 0x60-0xFF. This data storage space can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

To enhance the computing capability of the LGT8XM core, a 16-bit LD/ST extension has been added to the instruction line. This 16-bit LD/ST extension works with a 16-digit arithmetic acceleration unit (uDSC) for efficient 16-bit data arithmetic. The core also adds 16-bit access to RAM space. So the 16-bit LD/ST extension can transfer 16-bit data between uDSC, RAM, and working registers. For details, please refer to the "Computional Accelerator" chapter.

[OBS: In the original Chinese document there’s an erronous referrence: In the beginning of the document it’s referred (in text and diagram) to “Numerical Operation Accelerator (uDSU)”, whilst the correct referrence is "Digital Operation Accelerator Unit (DSC)" or “ Computional Accelerator (uDSC)”. Diagrams may be corrected in later revisions of this document.]

## Arithmetic and Logic Operation Unit (ALU)

LGT8XM contains a 16-bit arithmetic logic operation unit, which can complete 16-bit data arithmetic operations in one cycle. The efficient ALU is connected to 32 general-purpose working registers. It can complete arithmetic and logic operations between two registers or registers and immediate data in one cycle. There are three types of ALU operations: arithmetic, logic and bit operations. At the same time, the ALU part also contains a single-cycle hardware multiplier, which can realize the direct signed or unsigned operation of two 8-bit registers in one cycle. Please refer to the instruction set section for details.

*This area has been intentionally left blank*

## Status Register (SREG)

The status register mainly saves the result information generated by the latest ALU operation. This information is used to control the flow of program execution. The status register is updated after the ALU operation is completely completed, so that the use of a separate comparison instruction can be omitted, which can lead to a more compact and efficient code implementation. The value of the status register will not be automatically saved and restored when responding to and exiting from the interrupt, which requires software to implement.

### *SREG register definition*

| SREG system status register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x3F (0x5F) | | | | Default value: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | I | T | H | S | V | N | Z | C |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| *Bit definition* | | | | | | | | |

| | | |
|---|---|---|
| [0] | C | Carry flag, indicating that an arithmetic or logical operation caused a carry, please refer to the instruction description for details |
| [1] | Z | Zero flag, indicating that the result of arithmetic or logic operation is zero, please refer to the instruction description section |
| [2] | N | Negative flag, indicating that arithmetic or logical operation produced a negative number, please refer to the instruction description section |
| [3] | V | Overflow flag, indicating that the result of two's complement operation has overflowed, please refer to the instruction description section |
| [4] | S | Sign flag, equivalent to the XOR operation result of N and V, please refer to the instruction description section for details |
| [5] | H | The half-carry flag, useful in BCD operations, indicates the half-carry generated by byte operations |
| [6] | T | Temporary flag, used in bit copy (BLD) and bit store (BST) instructions, the T flag will be used as a temporary storage bit to temporarily store the value of a bit in a general-purpose register.<br>For details, please refer to the instruction description section |
| [7] | I | Global interrupt enable flag, this flag must be set to 1 to enable the core to respond to interrupt events. Different interrupt sources are controlled by independent control flags. The global interrupt enable flag is the last barrier to control the interrupt signal to enter the core. The I flag is automatically cleared by the hardware after the core responds to the interrupt vector, and is automatically set after the interrupt return instruction (RETI) is executed. The I flag can also be changed using SEI and CLI commands, please refer to the command description section |

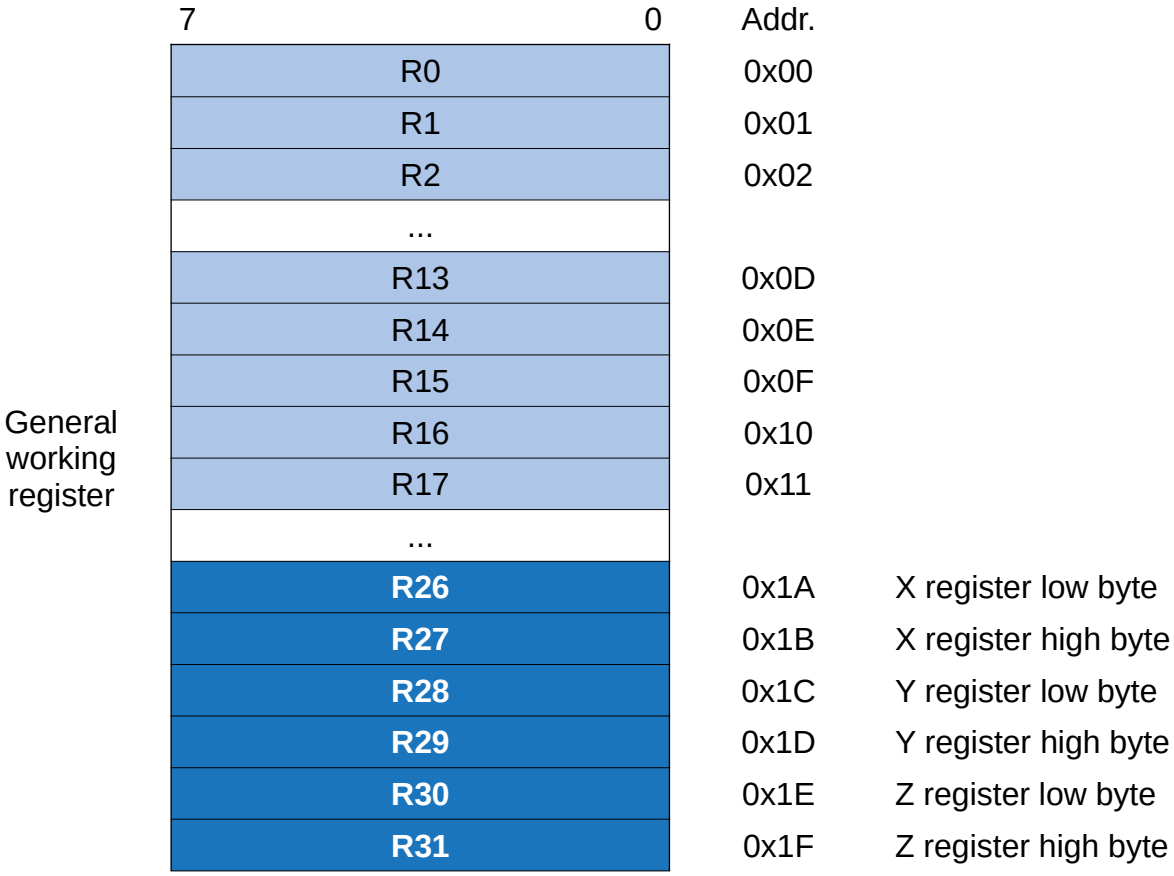
## General working register

The general working registers are optimized according to the LGT8XM instruction set architecture. In order to achieve the efficiency and flexibility required for core execution,

The general working register inside LGT8XM supports the following access modes:

- One 8-bit read at the same time as one 8-bit write operation
- Two 8-bit reads at the same time as one 8-bit write operation
- Two 8-bit reads at the same time as one 16-bit write operation
- One 16-bit read at the same time as one 16-bit write operation

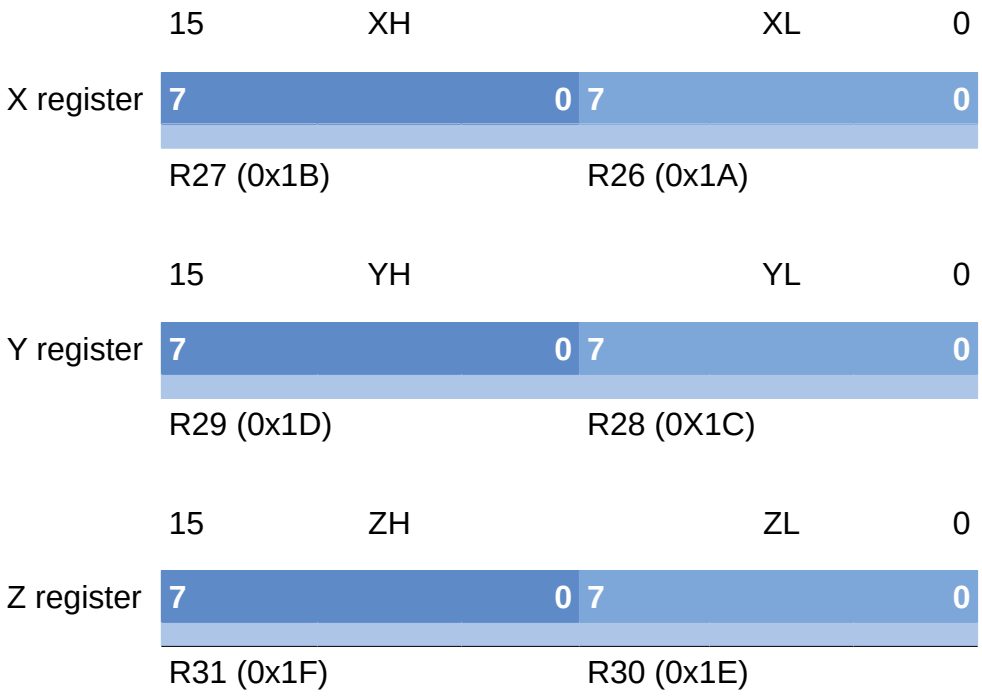
***LGT8XM** general working register*




Most instructions have direct access to all general-purpose working registers, and most of them are also single-cycle instructions. As shown in the figure above, each register corresponds to an address of a data storage space, and these general-purpose working registers are mapped to the data storage space. As soon as they don't really exist in SRAM, this uniformly mapped memory organization brings a lot of flexibility to accessing them. The X/Y/Z registers can be used as pointers to any general purpose register.

### *X/Y/Z register*

Registers R26...R31 can be combined in pairs to form three 16-bit registers. These three 16-bit registers are mainly used for the address pointer of indirect addressing access. The X/Y/Z register structure is as follows:




In different addressing modes, these registers are used as fixed offset, auto-increment and auto-decrement address pointers. For details, please refer to the instruction description section.

## Stack pointer

The stack is used to store temporary data, local variables, and the return address of interrupts and subroutine calls. It is important to note that the stack is not designed to grow from a high address to a low address. The stack pointer register (SP) always points to the top of the stack. The stack pointer points to the physical space where the data SRAM is located, where the stack space necessary for subroutine or interrupt calls is stored. The PUSH instruction will decrement the stack pointer.

The location of the stack in SRAM must be correctly set by software before subroutine execution or interrupt enable. Generally, the stack pointer is initialized to point to the highest address of SRAM. The stack pointer must be set to the upper SRAM start address. For the address mapped by SRAM in the system data storage, please refer to the system data storage section.

*Stack pointer related instructions*

| Instruction | Stack Pointer | Description |
|---|---|---|
| PUSH | Increase 1 | Data is pushed onto the stack |
| CALL<br>ICALL<br>RCALL | Increase 2 | The return address of the interrupt or subroutine call is pushed onto the stack |
| POP | Decrease 1 | Data is taken from the stack |
| RET<br>RETI | Decrease 2 | The return address of the interrupt or subroutine call is taken from the stack |

The stack pointer is composed of two 8-bit registers allocated in the I/O space. The actual length of the stack pointer depends on the system implementation. In some chip implementations of the LGT8XM architecture, the data space is so small that only SPL can meet the addressing needs. In this case, the SPH register will not appear.

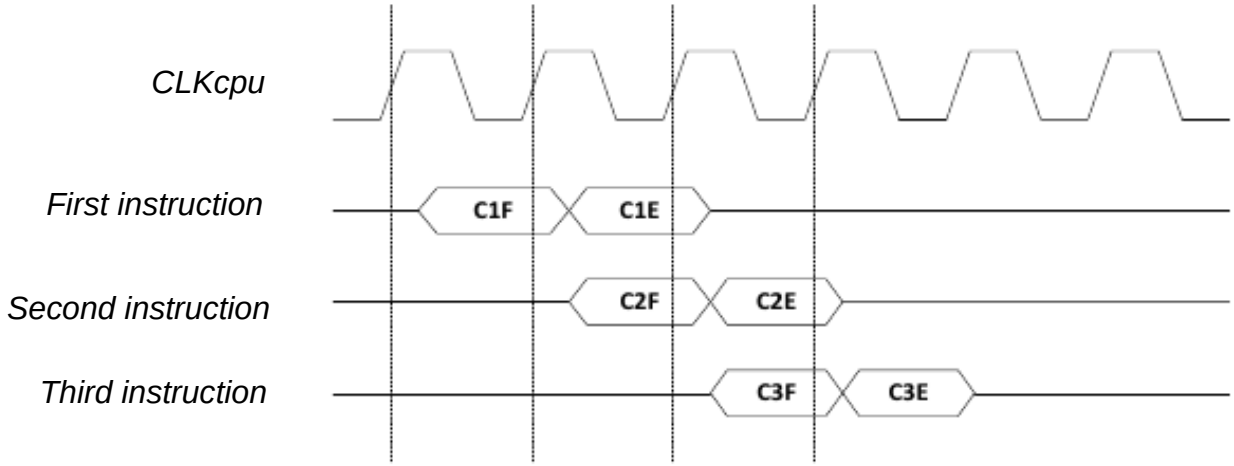
### *SPH/SPL stack pointer register definition*

| SPH/SPL stack pointer register | | |
|---|---|---|
| SPH: 0x3E (0x5E)<br>SPL: 0x3D (0x5D) | Default value: RAMEND | |
| SP | SP[15:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | SPL | The lower 8 bits of the stack pointer |
| [15:8] | SPH | The upper 8 bits of the stack pointer |

## Instruction execution timing

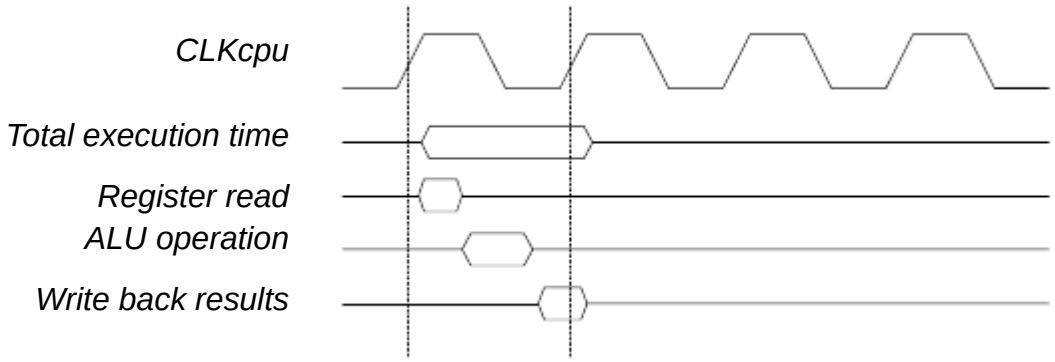
This chapter describes the general timing concepts of instruction execution. The LGT8XM core is driven by the core clock (CLKcpu), which comes directly from the system's clock source selection circuit.

The following figure shows the execution sequence of the instruction pipeline based on the Haval architecture and the concept of fast access to the register file. This is a physical guarantee that enables the core to obtain an execution efficiency of 1MIPS/MHz.




As can be seen from the above figure, during the execution of the first instruction, the second instruction will be read at the same time. When the second instruction enters the execution period, the third instruction will be read out at the same time. In this way, during the entire execution period, it does not need to spend extra cycles for reading instructions, and from the perspective of the pipeline, the efficiency of executing one instruction every clock cycle is realized.

The following figure shows the access timing of general working registers. In one cycle, ALU operations use two registers as operands, and the ALU execution result is written to the target register in this cycle.



## Reset and interrupt handling

LGT8XM supports multiple interrupt sources. These interrupts and reset vectors correspond to an independent program vector entry in the program space. Generally speaking, all interrupts are controlled by separate control flags. When this control flag is set and the global interrupt enable flag of the core is enabled, the core can respond to interrupt.

The lowest program space is reserved as the reset and interrupt vector area by default. For a complete list of interrupts supported by LGT8FX8P, please refer to the introduction in the interrupt chapter. This list also determines the priority of different interrupts. The lower the vector address of the interrupt, the higher the corresponding interrupt priority. Reset (RESET) has the highest priority, followed by INT0-external interrupt request 0. The start address of the interrupt vector table (except the reset vector) can be redefined to the beginning of any 256-byte alignment, which needs to be controlled by the MCU control register (MCUCR) in the IVSEL flag and IVBASE vector base address register implementation.

When the core responds to interrupt, the global interrupt enable flag I will be automatically cleared by the hardware. Users can realize interrupt nesting by enabling the I flag. So any subsequent interrupts will interrupt the current interrupt service routine. The I flag is automatically set after the interrupt return instruction (RETI) is executed, so that the subsequent interrupts can be responded to normally.

There are two basic types of interrupt. The first type is triggered by an event, and the interrupt flag is set after the interrupt event occurs. For this kind of interrupt, after the core responds to the interrupt request, the current PC value is directly replaced with the actual interrupt vector address, the corresponding interrupt service subroutine is executed, and the hardware automatically clears the interrupt flag. The interrupt flag can also be cleared by writing a 1 to the interrupt flag. If the interrupt enable flag is cleared when an interrupt occurs, the interrupt flag will still be set to record the interrupt event. After the interrupt is enabled, the recorded interrupt event will be responded to immediately. Similarly, if the global interrupt enable flag (SERG.I) is cleared when an interrupt occurs, the corresponding interrupt flag will also be set to record the interrupt event. After the global interrupt enable flag is set, these recorded interrupts will be executed in order of priority.

The second type of interrupt is when the interrupt condition persists, the interrupt is always serviced. This type of interrupt does not require interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be serviced.

When the LGT8XM core exits from the interrupt service routine, the execution flow will return to the main program. After executing one or several instructions in the main program, other pending interrupt requests can be responded to.

It should be noted that the system status register (SREG) will not be automatically saved after entering the interrupt service, nor will it be automatically restored after returning from the interrupt service. It must be handled by the software.

When interrupts are disabled using the CLI instruction, the interrupts will be disabled immediately. All interrupts that occur after a CLI instruction will not be serviced. Even an interrupt that occurs at the same time as the CLI instruction is executed will not be serviced. The following example shows how to use the CLI to avoid interrupts disrupting the EEPROM write sequence:

### Interrupt response time

The LGT8XM core is optimized for interrupt response, so that any interrupt must be serviced within 4 system clock cycles. After 4 system clock cycles, the interrupt service routine enters the execution cycle. In these 4 clocks, the PC value before the interrupt is pushed into the stack, and the system execution flow jumps to the interrupt vector corresponding to the interrupt service routine. If the interrupt occurs during the execution of a multi-cycle instruction, the core will ensure that the current instruction ends correctly. If the interrupt occurs while the system is in sleep state (SLEEP), an additional 4 clock cycles are required to respond to the interrupt. This added clock cycle is used for the synchronization cycle of the wake-up operation from the selected Sleep mode. For the detailed description of sleep mode, please refer to the relevant chapters on power management.

Returning from an interrupt service routine takes 2 clock cycles. During these 2 clock cycles, the PC is restored from the stack, the stack pointer is incremented by 2, and the global interrupt control flag is automatically enabled.

## Storage unit

### Overview

This chapter mainly describes the different storage units inside the LGT8FX8P series. The LGT8XM framework supports two main internal storage spaces, namely data storage space and program storage space. LGT8FX8P also contains data FLASH, and the data storage function of EEPROM interface can be realized through the internal controller. In addition, the LGT8FX8P system also contains a special storage unit for storing system configuration information and the global device number (GUID) of the chip.

The LGT8FX8P series of chips includes four different models of LGT8F88P/168P/328P; the peripherals and packages of the four models are fully compatible, and the difference is the FLASH program storage space and internal data SRAM. The following table describes the LGT8FX8P series of chips more clearly Different storage space configurations:

| DEVICE | FLASH | SRAM | E2PROM | Interrupt vector |
|---|---|---|---|---|
| LGT8F88P | 8KB | 1KB | 2KB | 1 instruction word |
| LGT8F168P | 16KB | 1KB | 4KB | 2 instruction word |
| LGT8F328P | 32KB | 2KB | Configurable as 0K/1K/2K/4K/8K (Shared with FLASH) | 2 instruction word |

**NOTE:** Four models? It states four models in the original Chinese text, but is it correct? (I only accounts for three.)

LGT8F328P does not have an independent FLASH space for simulating the E2PROM interface; the storage space for simulating E2PROM is shared with the program FLASH, and the user can choose the appropriate configuration according to the application requirements.

Due to the unique implementation of the emulated E2PROM interface, the system needs twice the program FLASH space to emulate the equivalent E2PROM storage space, for example, for LGT8F328P, if the user configures 1KB E2PROM space, there will be 2KB bytes of program space reserved, leaving 30KB FLASH space is used to store programs.

LGT8F328P program FLASH and E2PROM share configuration table:

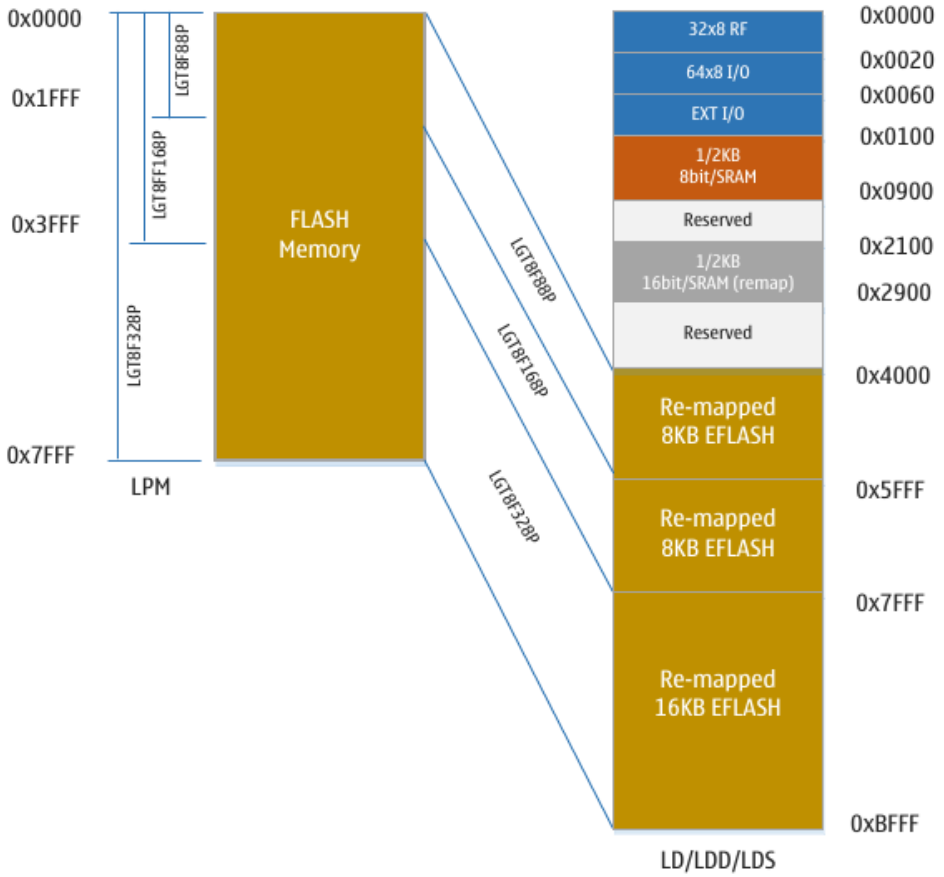
| DEVICE | FLASH | E2PROM |
|---|---|---|
| LGT8F328P | 32KB | 0KB |
| | 30KB | 1KB |
| | 28KB | 2KB |
| | 24KB | 4KB |
| | 16KB | 8KB |

## System programmable FLASH program storage unit

The LGT8FX8P series microcontrollers respectively include 8K/16K/32K bytes on-chip online programmable FLASH program storage units.

Program FLASH can guarantee at least 100,000 erase and write cycles. LGT8FX8P integrates FLASH interface controller, which can realize in-system programming (ISP) and program self-upgrade function. For the specific implementation details, please refer to the description in the section about the FLASH interface controller in this chapter.

The program space can also be directly accessed (read) through the LPM instruction. This feature can realize the application-related constant lookup table. At the same time, the FLASH program space is also mapped to the system data storage space, and users can also use LD/LDD/LDS to access the FLASH space. The program space is mapped to the address range starting from 0x4000 in the data storage space. As shown below:



## SRAM data storage unit

The LGT8FX8P series microcontrollers are relatively complex microcontrollers that support many different types of peripherals, whose controllers are allocated in 64 I/O register spaces and can be accessed directly through IN/OUT instructions. The control registers of other peripherals are allocated in the 0x60 ~ 0xFF area. Since this part of the space is mapped to the data storage space, it can only be accessed through ST/STS/STD and LD/LDS/LDD instructions.

The system data storage space of LGT8FX8P starts from address 0 and maps the general working register file, I/O space, extended I/O space and internal data SRAM space respectively. The first 32-byte address corresponds to the 32 general-purpose working registers of the LGT8XM core. The next 64 addresses are standard I/O spaces that can be directly accessed through IN/OUT instructions. Then the 160 addresses are the expansion I/O space, and the next is up to 2K bytes of data SRAM. The part of the space from 0x4000 to the end of 0xBFFF maps the FLASH program storage unit.

The 1K/2K byte SRAM in the system is mapped to two spaces respectively. The space from 0x0100 to the end of 0x0900 is read and written by the core in 8-bit bytes. Starting at 0x2100 and ending at 0x2900, this area is a 16-bit wide access space. The system RAM is mapped to the high-order address starting at 0x2100, which is mainly used to work with the uDSC module to achieve efficient 16-bit data storage. When programming, add the address of the ordinary 8-bit addressing variable to the offset of 0x2000 to switch to the 16-bit access mode.

The system supports 5 different addressing modes that can cover the entire data space: direct access, indirect access with offset, indirect access, indirect access with decrement address before access, and indirect access with increment address after access. The general working registers R26 to R31 are used as address pointers for indirect access. Indirect access can address the entire data storage space. The indirect access with offset address can address 63 address spaces around the Y/Z register as the base address.

When using the register indirect access mode that supports address auto-increment/decrement, the address register X/Y/Z will be automatically decremented/incremented by hardware before/after the access occurs. For details, please refer to the instruction set description section.

The 16-bit register X/Y/Z and the related automatic addressing modes (increment, decrement) also play a very important role in the 16-bit extended mode. The 16-bit extended mode can use the increment/decrement mode of LD/ST to realize automatic increment and decrement addressing with variables. This mode is very effective when performing operations on arrays. For specific implementation, please refer to the relevant chapters of "Digital Operation Accelerator (uDSC)".

## General I/O Register

The I/O space of LGT8FX8P has three general-purpose I/O registers GPIOR2/1/0. These three registers can be accessed using IN/OUT instructions to store user-defined data.

## Peripheral register space

For the detailed definition of I/O space, please refer to the "Register description" chapter in the LGT8FX8P data manual.

All peripherals of LGT8FX8P are allocated to I/O space. All I/O space addresses can be accessed by LD/LDS/LDDD and ST/STS/STD instructions. The data accessed is transmitted through 32 general-purpose working registers. The I/O registers between 0x00 and 0x1F can be accessed by flag addressing instructions SBI and CBI. In these registers, the value of a certain flag can be detected by the SBIS and SBIC instructions to control the execution flow of the program. For details, please refer to the instruction set description section.

When using the IN/OUT instruction to access the I/O register, the address between 0x00 ~ 0x3F must be addressed. When using the LD or ST instruction to access the I/O space, the I/O space must be accessed through the mapping address of the unified mapping space in the system data memory (plus an offset of 0x20). Some other peripheral registers (0x60 ~ 0xFF) allocated in the extended I/O space can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

In order to be compatible with future devices, reserved bits must be written as 0 during write operations. You cannot perform write operations on the reserved I/O space.

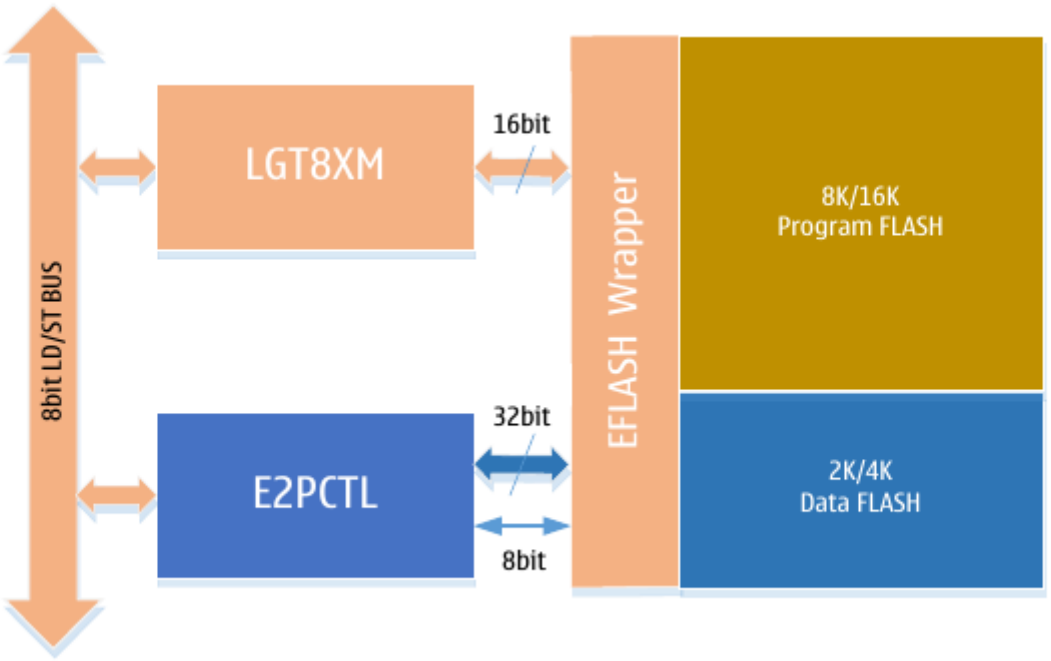
Some registers include status flags, which need to be written to 1 to clear them. It should be noted that the CBI and SBI instructions only support specific bits, so CBI/SBI can only work on the registers containing these status flags. In addition, CBI/SBI instructions can only work on registers within the address range of 0x00 to 0x1F.

## FLASH controller (E2PCTL)

LGT8FX8P internally integrates a flexible and reliable EFLASH read and write controller, which can use the existing data FLASH storage space in the system to realize the storage space for byte read and write access, and realize the storage application similar to E2PROM; E2PROM interface simulation adopts erasing and writing The balanced algorithm can increase the usage cycle of data FLASH by about 1 times, and can guarantee more than 100,000 erasing and writing cycles.

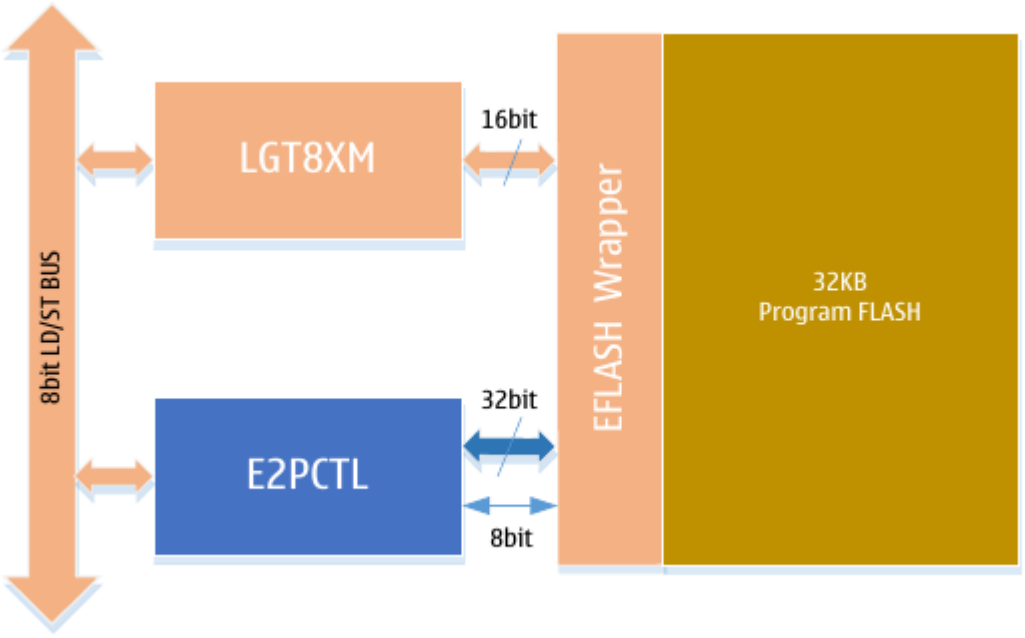
E2PCTL controller also realizes the online erasing and writing operation of FLASH program space, and can realize the function of online automatic firmware upgrade through software. Access to the programmable FLASH program space through the FLASH controller, only supports page erasure (1024 bytes) and 32-bit width read and write access.

*LGT8F88D/168D E2PCTL controller structure diagram*




When E2PCTL simulates the E2PROM function to access the data FLASH space, it can support 8-bit and 32-bit read and write widths. When accessing the program FLASH space, it supports page erasing and 32-bit data reading and writing. Since the smallest storage unit of the internal FLASH of LGT8FX8P is 32-bit, it is recommended to use 32-bit access, especially for write operations. The 32-bit access read and write operations are not only efficient, but also help protect the flash memory cell's erase and write life.

*LGT8F328P E2PCTL controller structure diagram*




There is no extra data FLASH in LGT8F328P. Therefore, LGT8XM core and E2PCTL share the internal 32K bytes FLASH storage space. Users can divide the 32K-byte FLASH space into program space and data space according to their needs. By configuring the E2PCTL controller, you can set the size of the simulated E2PROM space, the E2PCTL uses page swap to simulate E2PROM logic,

and the algorithm works with units of 1K bytes. Therefore, to simulate 1K bytes of E2PROM space it needs to occupy 2K bytes of FLASH space, to simulate 4K bytes of E2PROM space it needs to occupy 8K bytes of FLASH space, and so on. For the specific implementation, please refer to the description of the implementation of the E2PCTL algorithm.

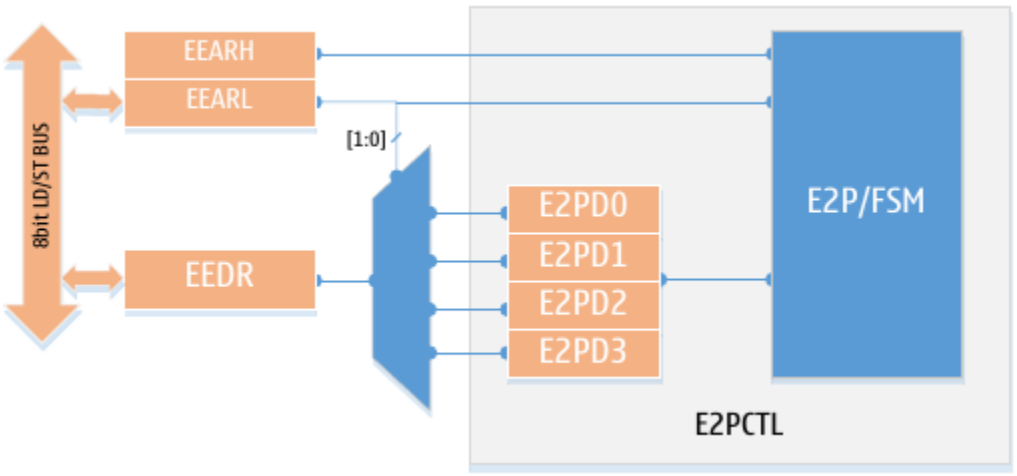
### *E2PCTL data register*

There are 4 bytes of data cache (E2PD0~3) inside the E2PCTL controller, and this 4 bytes of cache constitutes the 32-bit data interface that finally accesses the FLASH space.

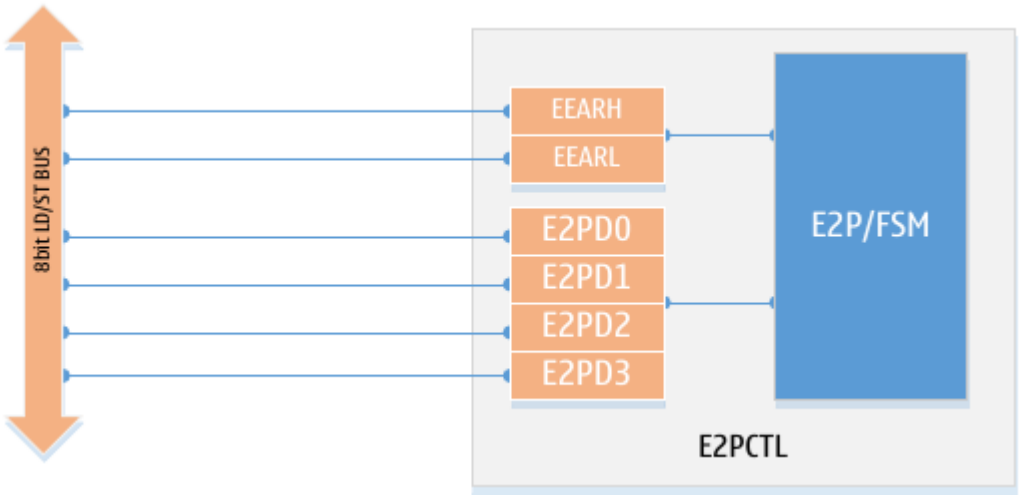
When the E2PCTL controller works in the byte read and write mode, EEDR is used as the interface for reading and writing byte data, E2PCTL loads the data into the correct data cache with the address information of EEARL[1:0], and according to the current FLASH target address The data complements another three bytes of data, and finally the combined complete 32-bit data is updated to FLASH.

When E2PCTL works in 32-bit read-write mode, the EEDR register can still be used as a public data interface, and the internal data cache can be addressed through EEARL[1:0] as an address to read and write a complete 32-bit data. In addition, it is also possible to directly access the registers (E0~3) that are mapped to the IO space using the data cache.

Schematic diagram of data access when E2PCTL works in 8-bit byte read and write mode:




Data access diagram when E2PCTL works in 32-bit word read/write mode:



The byte mode is used for backward compatibility with the byte read and write mode of the LGT8FX8D. The built-in FLASH of LGT8FX8P has a 32-bit interface width. Using 32-bit read-write mode will bring great benefits to read-write efficiency and FLASH erasing and writing life. Therefore, it is recommended to use 32-bit read-write mode.

### *E2PCTL simulates E2PROM interface algorithm*

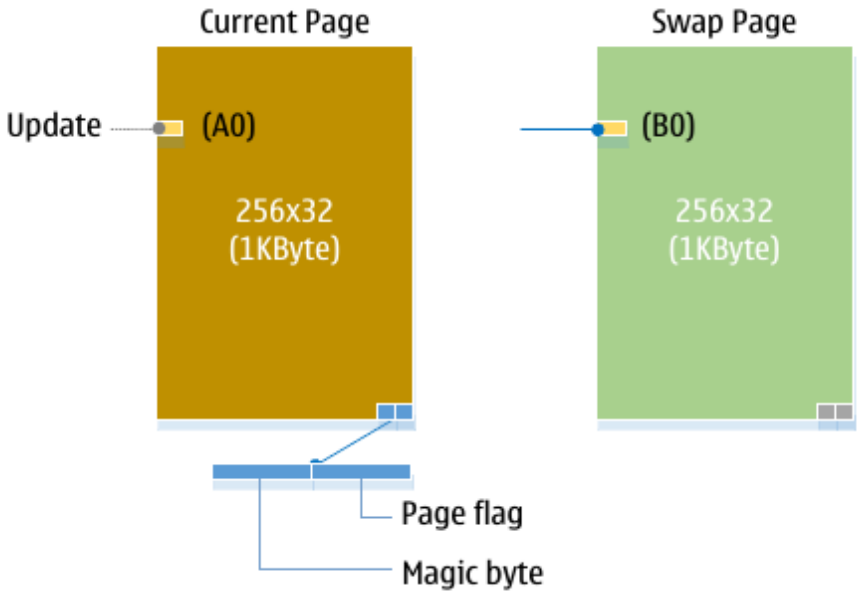
We know that the FLASH memory must be erased before writing, and the erase operation is based on pages. The size of one page of the built-in FLASH memory of LGT8FX8P is 1K bytes. Therefore, in order to update one byte of data in the page, it is also necessary to first erase the data of the entire page, then update the target address data, and restore the data of other bytes in the page at the same time. The whole operation is not only time-consuming, but also brings Risk of accidental loss of data due to power loss.

E2PCTL uses a page swap algorithm to simulate E2PROM. The page swap algorithm mode can ensure that the original data will not be lost due to unexpected conditions such as power failure when the page erase operation is performed. At the same time, the exchange algorithm uses two page spaces to exchange each other alternately, which also increases the service life of the simulated E2PROM space.

In terms of efficiency, the E2PCTL controller implements a continuous data update mode, reducing the repeated erasing and writing process caused by repeated data updates.

In terms of implementation, E2PCTL manages each page separately and occupies the last 2 bytes of a page as the page status information. Therefore, when users use E2PROM simulation space larger than 1K, they need to pay attention to the special handling of addresses crossing 1K space. Because the last 2 bytes of every 1K space are reserved for E2PCTL, users cannot read and write these 2 bytes normally.

The following figure shows the logic diagram of E2PCTL based on the page swap algorithm:

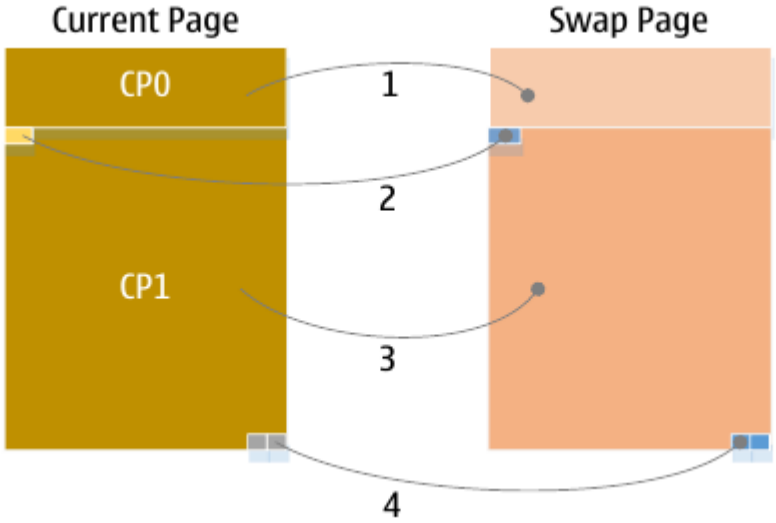



As shown in the figure, E2PCTL uses 2 pages internally to simulate a page-sized E2PROM space. One of these two pages is marked as the current page, and the other is the swap page. E2PCTL uses the last 2 bytes of the page to store page information. When we need to update a certain byte

in the page, such as the A0 byte in the figure above. First, we will not erase the current page, but erase the swap page. Then divide the current page into 3 part operations. The first is the data before A0, we call this part of the space CP0, and then the data after A0, this part of the space is CP1. E2PCTL will copy the data corresponding to CP0 to the corresponding address of the swap page according to the user configuration, then write the data that needs to be updated to the address corresponding to the swap page (B0), and finally copy the data of CP1 to the swap page.

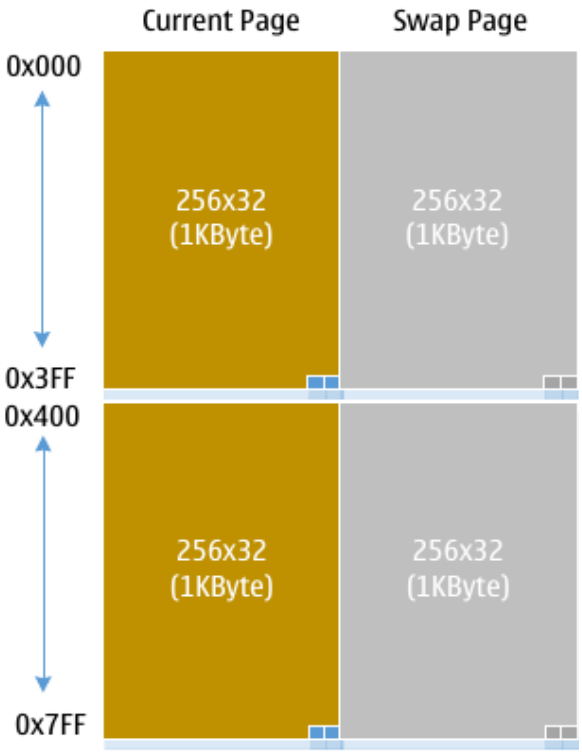
After completing the above operations, the data has been exchanged, but the page status has not been updated. Therefore, if a power failure or other abnormality occurs before this, because this update operation has not been completed, the previous data will not be destroyed, ensuring the integrity of the data. If all goes well, E2PCTL will write the updated page status to the page information of the previous exchange page at the end of the CP1 exchange data to realize the face-to-face page replacement. After that, the exchange page becomes the current page.

The E2PCTL page exchange process is shown in the figure below (1->2->3->4):




When the E2PROM simulation space of the system configuration is greater than 1K, E2PCTL still uses the page as the smallest unit to realize the simulation algorithm of the E2PROM space. For example, if the user configures a 2K E2PROM area, in fact, E2PCTL will occupy 4 pages (4K) of space. Among them, 2 pages are a group, which is used to simulate the E2PROM space of a page size.

It should be noted that the user-configured 2K bytes of E2PROM space is not continuous, because the last 2 bytes of each page will be used to store page status information.



### *E2PCTL continuous programming mode*

Since updating through E2PCTL will cause page swap, the swap page will be erased during the page swap process. Page erasing not only takes time, but also decreases the life of the FLASH. Therefore, E2PCTL adds a continuous write mode. In the continuous write mode, the user can continuously update the E2PROM area, and only at the end of the continuous address, will the page swap operation be performed. For applications that need to continuously update a whole block of data, the continuous mode is more effective.

Continuous programming mode The SWM flag of the E2PCTL control register ECCR is enabled. After continuous mode is enabled, subsequent write operations will directly write data to the address corresponding to the swap page. In SWM mode, the write operation will not execute the CP0/1 area data copy operation. Before writing the last byte, the software disables the continuous mode through SWM, and then executes the write. After that, E2PCTL will execute the complete CP0/1 copy operation and update the page status information.

### *E2PCTL read and write FLASH program space*

Through the E2PCTL controller, the read and write access to the program FLASH space can be realized. Unlike the emulated E2PROM, access to the program FLASH space through E2PCTL requires complete software control. Proceed as follows:

1. Erase the target page. Before updating the data, you need to erase the target page first. The page address is given by the EEAR register. For the erase command control of the FLASH page, please refer to the definition of the EECR register;
2. The FLASH space for program writing must be in 32-bit minimum unit. Set data through E2PD0~3;
3. The target address is given by the EEAR register, the address EEAR[1:0] will be ignored;

By reading and writing the program FLASH space through E2PCTL, the online *In Application Programming* (IAP) update can be realized, which is very useful in some applications that need to update application data on site and provide customized product updates.

### *E2PCTL interface operation process*

The E2PCTL controller is mainly realized through four registers, which are E2PCTL control status registers EECR, ECCR; data registers EEDR (E2PD0~E2PD3) and address registers EEAR (EEARL/EEARH).

The ECCR register is used to set the working state of E2PCTL. Most of the states need to be set before E2PCTL works, and this is generally implemented in the system initialization process. The SWM flag in the ECCR register is used to enable continuous write mode, this control flag needs to be set during the continuous write operation.

The EECR register is used to control mode of operation, like set, read and erase commands.

The EEDR register is used for read and write operations in 8-bit (byte) mode, and E2PD0~3 are used in 32-bit mode;

The EEAR register is used to set the target address for reading and writing, and it is also used to set the page address for page erase operations. The page address is aligned in page units, and the size of a page is 1K bytes. It should be noted that the address specified by EEAR is a byte address.

*Access the FLASH program space through the E2PCTL interface:*

The FLASH program space can be read, written and erased through the E2PCTL interface. Reading and writing to FLASH space only supports 32-bit access width. Erase operations are performed on an entire page, each page is 1K bytes in size (256x32).

Before writing [to] the FLASH program space, first erase the page where the target address is located. E2PCTL write FLASH program space does not support continuous mode, users need to complete the write operation in order. The following is the process of erasing and writing the FLASH program space:

1. Program FLASH page erase operation
    - Set EEAR[14:0] as the target page address to be erased, the size of the program FLASH page is 1K bytes, so EEAR[14:10] will be the page address, EEAR[9:0] is set to 0
    - Set EEPM[3:0] = 1X01, where EEPM[2] can be set to 0 or 1
    - Set EEMPE = 1, and EEPE = 0
    - In four cycles, set EEPE = 1, (to) start the program FLASH erase process
2. Program FLASH programming operation
    - Write E2PD0~3 to prepare 32-bit programming data
    - Set EEAR as the target address, where the address is 4 bytes aligned (32 bit)
    - Set EEPM[3:0] = 1X10, where EEPM[2] can be set to 0 or 1
    - Set EEMPE = 1, and EEPE = 0
    - In four cycles, set EEPE = 1, (to) start the FLASH programming process

*Access the E2PROM simulation space through the E2PCTL interface:*

The E2PCTL controls access to the data FLASH space by simulating the E2PROM interface logic. The simulated E2PROM supports 8-bit, 16-bit and 32-bit data width, for both read and write access. The 8-bit mode has better compatibility with E2PROM interface whilst the 32-bit mode is beneficial to improve the storage efficiency and the service life of FLASH, so 32-bit is the recommended read-write mode. The E2PROM simulated interface supports continuous read and write mode, and this has obvious advantages in data applications that need to update multiple consecutive addresses within a small time frame and is recommended.

For LGT8F88P/168P, data FLASH is an independent storage space and there is no need to configure and enable the FLASH data space through the ECCR register. LGT8F328P has no independent data FLASH space, data FLASH and program FLASH respectively share 32K bytes of FLASH space. Thus for the LGT8F328P it is necessary to enable the data FLASH partition function through the ECCR register, and configure the size of the data FLASH through the ECS[1:0] flags of the ECCR register. After the configuration takes effect, application data and program FLASH are the same as for LGT8F88P/168P.

When the FLASH controller implements the *E2PROM interface*, the logic of automatically erasing the data FLASH when necessary has been implemented internally, so the EPROM erasing command is optional, and this command is only used when the user needs to perform erasing alone. The EECR register controls the erase/write timing of FLASH, including program FLASH and E2PROM. The specific mode of operation needs to be set through EEPME and EEPM[3:0] of the EECR register.
The read operation of *E2PROM* is relatively simple. After setting the target address and mode, setting the EERE flag will read the 32-bit data corresponding to the target address into the FLASH controller, and the user can read the byte of interest through the EEDR register.
The FLASH controller does not implement a specific read operation of the *program FLASH space*, but the user can conveniently use the LPM or use the LD/LDD/LDS instructions to read the address of the unified data mapping space throughout the *program FLASH*.

1. 8 bit mode, programming E2PROM
   - Set the target address to the EEARH/L register
   - Set new data to EEDR register
   - Set EEPM[3:1] = 000, EEPM[0] can be set to 0 or 1
   - Set EEMPE = 1, and EEPE = 0
   - Within four cycles, set EEPE = 1
     When the setting is completed, the FLASH controller will start the programming operation. During the programming, the CPU will remain at the current instruction address and will not continue to run until the operation is completed. During the programming process, if the data FLASH needs to be erased, the FLASH controller will automatically start the erase process. (Use of the therm “CPU” seems odd, given this after all is a microcontroller, but this is the actual therm used in the original Chinese documentation.)
2. 32-bit mode, programming E2PROM
   - Prepare 32-bit data through E2PD0~3
   - Set the target address to the EEARH/L register. Note that this is a byte-aligned address, and the FLASH controller uses EEAR[15:2] as the address to access FLASH.
   - Set EEPM[3:1] = 010, EEPM[0] can be set to 0 or 1
   - Set EEMPE = 1, and EEPE = 0
   - Within four cycles, set EEPE = 1
3. 8 Bit mode, read E2PROM
   - Set the target address to the EEARH/L register
   - Set EEPM[3:1] = 000
   - Set EERE = 1 to start E2PROM read operation
   - Wait for 2 cycles (perform two NOP operations)

- The data corresponding to the target address is updated to the EEDR register

4. 32-bit mode, read E2PROM
   - Set EEARH/L as the target address, the address is 4 bytes aligned
   - Set EEPM[3:1] = 010, enable 32-bit interface mode
   - Set EERE = 1, start E2PROM read operation
   - Wait for 2 system clock cycles (execute two NOP instructions)

E2PCTL accesses the simulated E2PROM space and supports continuous programming mode. The continuous access mode is very efficient for applications that need to update one data block at a time, and it also helps to improve the life of FLASH. The continuous programming mode only supports 32-bit wide data programming operations.

The continuous access mode is enabled by the SWM flag of the ECCR register. After SWM is enabled, the following operations to simulate E2PROM space via E2PCTL are in continuous programming mode. In the continuous programming mode, the E2PCTL controller will automatically process the page change according to the data in the target address. However, if a page change occurs during the continuous programming mode, the controller will not automatically exchange the data in the CP0/1 area or update the page information during the continuous programming.

When the continuous programming reaches the last operation, end the continuous programming mode by clearing the SWM flag before starting the last programming operation, and complete in non-SWM mode. After programming completes, the E2PCTL will automatically copy the data in the CP0/1 area to the adjacent swap page and update the page information of the swap page to make it the current valid page, thus completing the entire continuous programming operation.

5. Operation process of continuous programming mode:
   - Configure the size of the data FLASH through ECCR and enable the SWM flag
   - Use 32-bit mode programming to simulate E2PROM area
   - If it is not the last operation, go back to step 2 and continue to program the next data
   - If the last programming is reached, first disable the continuous programming mode through SWM, and then use the operation flow of step 2 to complete the last programming

### *E2PCTL efficient FLASH data management*

In addition to realizing the continuous programming mode, the E2PCTL controller can also independently control the data exchange and replication of the page exchange process through the CP0/1 flags of the ECCR register. CP0/1 of the ECCR register are respectively used to control the exchange operation of the CP0/1 area data in the current page during the page exchange process. When the CP0/1 flag is cleared, the data in the corresponding area in the current page will not be swapped during the page swapping process. This section provides an efficient management approach that will take advantage of this feature.

In the FLASH data update process, the most time-consuming operation occurs in the swap page erasing process. Therefore, we can address a data management method that minimizes the number of page erases, which can improve programming efficiency and reduce lifetime loss.

Here we provide a reference algorithm that is suitable for data management applications based on data blocks:

1. Assuming that the user data is only a complete data block, the data block size is an integer multiple of 4 bytes;
2. Each data update will update a complete data block
3. In addition to storing user data, the data block meta-information also needs to be stored as a block of management information

Under the above three conditions, we can make full use of E2PCTL's continuous programming mode and automatic page switching mechanism to realize an efficient FLASH data management method.

Since the data updated each time is a data block of the same size, and the address information pointing to the next piece of data is stored in the data structure of each block, we can program the FLASH in the order of the address each time the data is updated, with no need to do CP0/1 Data replication. At the same time, since the data is stored to an erased area each time, page erasure will not occur.

When the last block of data is written, the next block of data pointed to by its structure information returns to the starting address of the page. After a data write operation occurs, E2PCTL will initiate a page erasing process and update the current active page.

### *Protective measures for FLASH operation*

If the VCC voltage is low, the FLASH erasing and writing operations may cause errors because the voltage below safe level.

There may be two reasons for FLASH/data erase and write operation errors under low voltage. First of all, normal FLASH erase and write operations require a minimum operating voltage, below this voltage, the operation will fail and cause data errors. The second reason is that the core runs at a certain frequency, and also needs a minimum voltage level. When the voltage is lower than this, it will cause an error in the execution of the instructions, thus causing an error in the operation of the FLASH.

You can avoid similar problems in the following simple ways:

When the power supply voltage is low, let the system enter the reset state. This can be achieved by configuring the internal low voltage detection circuit (VDT). If the VDT detects that the current working voltage is lower than the set threshold, the VDT will output a reset signal. If the VDT threshold cannot meet the needs of the application, consider adding an external reset circuit.

## Register description

### *FLASH address register-EEARH/EEARL*

| EEARH/EEARL | | |
|---|---|---|
| EEARH: 0x22 (0x42) | | Default value: 0x0000 |
| EEARL: 0x21 (0x41) | | |
| bits | EEAR[15:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | EEARL | EFLASH/E2PROM access address 8 lower bits. |
| [14:8] | EEARH | EFLASH/E2PROM access address 7 high bits |
| [15] | – | Unused |

When using the E2PCTL controller to access the program FLASH area, EEAR[14:2] is used to access the entire program space aligned with 4 bytes. EEAR[1:0] is only used when accessing the data register EEDR. For details, please refer to the description of the EEDR data register below. E2PCTL controller supports 8/16/32-bit mode, no matter which mode, EEAR here is byte-aligned addressing.

### *FLASH Data Register-EEDR/E2PD0*

| EEDR/E2PD0-FLASH/E2PROM data register 0 | | |
|---|---|---|
| EEDR/E2PD0: 0x20 (0x40) | | Default value: 0x00 |
| bits | EEDR[7:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | EEDR<br>E2PD0 | E2PCTL data register<br>In 16/32-bit mode, it is used to access the lowest byte |

### *FLASH Data Register-E2PD1*

| E2PD1 – E2PCTL data register 1 | | |
|---|---|---|
| E2PD1: 0x5A | | Default value: 0x00 |
| bits | E2PD1[7:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | E2PD1 | In 16-bit mode, it is used to store the upper 8 bits of 16-bit data<br>In 32-bit mode, it is used to store the upper 8 bits of the lower 16 bits of data |

### *FLASH Data Register-E2PD2*

| E2PD2-FLASH data register 2 | | |
|---|---|---|
| E2PD2: 0x57 | | Default value: 0x00 |
| bits | E2PD2[7:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | E2PD2 | In 32-bit mode, it is used to store the lower 8 bits of the upper 16 bits of data |

### *FLASH Data Register-E2PD3*

| E2PD3-FLASH data register 3 | | |
|---|---|---|
| | | Default value: 0x00 |
| bits | E2PD3[7:0] | |
| R/W | R/W | |
| *Bit definition* | | |
| [7:0] | E2PD3 | The upper 8 bits used to store the upper 16 bits of data in 32-bit mode |

### *FLASH Mode Control Register-ECCR*

| ECCR-FLASH/E2PROM configuration register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ECCR: 0x36 (0x56) | | | | | Default value: 0x00 | | | |
| bits | WEN | EEN | ERN | SWM | CP1 | CP0 | ECS1 | ECS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial val. | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| *Bit definition* | | | | | | | | |

| | | |
|---|---|---|
| [7] | WEN | ECCR write enable control<br>Before modifying ECCR, write WEN to 1, and then update the content of ECCR register within 6 system cycles |
| [6] | EEN | E2PROM enable, only valid for LGT8F328P.<br>1: Enable E2PROM emulation, will occupy space from 32K FLASH.<br>0: Disable E2PROM emulation, full 32K FLASH is used for program space. |
| [5] | ERN | Writing 1 will reset the E2PCTL controller |
| [4] | SWM | Continuous write mode, suitable for simulating E2PROM controller operation |
| [3] | CP1 | Page swap CP1 area enable control |
| [2] | CP0 | Page swap CP0 area enable control |
| [1:0] | ECS[1:0] | E2PROM space configuration.<br>00: 1KB E2PROM, 30KB program FLASH.<br>01: 2KB E2PROM, 28KB program FLASH.<br>10: 4KB E2PROM, 24KB program FLASH.<br>11: 8KB E2PROM, 16KB program FLASH. |

## *FLASH Access Control Register-EECR*

| EECR-FLASH/E2PROM control register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| EECR: 0x1F (0x3F) | | | | Default value: 0x00 | | | | |
| bits | EEPM3 | EEPM2 | EEPM1 | EEPM0 | EERIE | EEMPE | EEPE | EERE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial val. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *Bit definition* | | | | | | | | |

| | | EFLASH/EPROM access mode control flags | | | | |
|---|---|---|---|---|---|---|
| | | [3] | [2] | [1] | [0] | Mode description |
| [7:4] | EEPM[3:0] | 0 | 0 | 0 | x | 8-bit mode read/write E2PROM (default) |
| | | 0 | 0 | 1 | x | 16-bit mode read/write E2PROM |
| | | 0 | 1 | 0 | x | 32-bit mode read/write E2PROM |
| | | 1 | x | 0 | 0 | E2PROM erase (optional operation) |
| | | 1 | x | 0 | 1 | Program FLASH erase (page erase) |
| | | 1 | x | 1 | 0 | Program FLASH programming |
| | | 1 | x | 1 | 1 | Reset FLASH/E2PROM controller |
| [3] | EERIE | FLASH/E2PROM ready interrupt enable control. Write 1 to enable, write 0 to disable. When EEPE is automatically cleared by hardware, the E2PROM ready interrupt is valid. During EPROM operation, this interrupt will not be generated | | | | |
| [2] | EEMPE | The FLASH/E2PROM programming operation enable control flag EEMPE is used to control whether EEPE is valid. When EEMPE is set to 1, and EEPE is set to 0, setting EEPE to 1 within the nest four cycles will start the programming operation. Otherwise, programming operation is prohibited. After four cycles, EEMPE is automatically cleared. | | | | |
| [1] | EEPE | FLASH/E2PROM programming operation enable flag | | | | |
| [0] | EERE | E2PROM read enable flag, the data will be available after two system cycles | | | | |

## *General I/O Register-GPIOR2*

| GPIOR2-General purpose I/O register 2 | | |
|---|---|---|
| GPIOR2: 0x2B (0x4B) | | Default value: 0x00 |
| Bits | GPIOR2[7:0] | |
| R/W | R/W | |
| Initial val. | 0x00 | |
| Bit definition | | |
| [7:0] | GPIOR2 | General I/O register 2, used to store user-defined data |

### *General I/O Register-GPIOR1*

| GPIOR1-General I/O Register 1 | | |
|---|---|---|
| GPIOR1: 0x2A (0x4A) | | Default value: 0x00 |
| Bits | GPIOR1[7:0] | |
| R/W | R/W | |
| Initial val. | 0x00 | |
| Bit definition | | |
| [7:0] | GPIOR1 | General I/O register 1, used to store user-defined data |

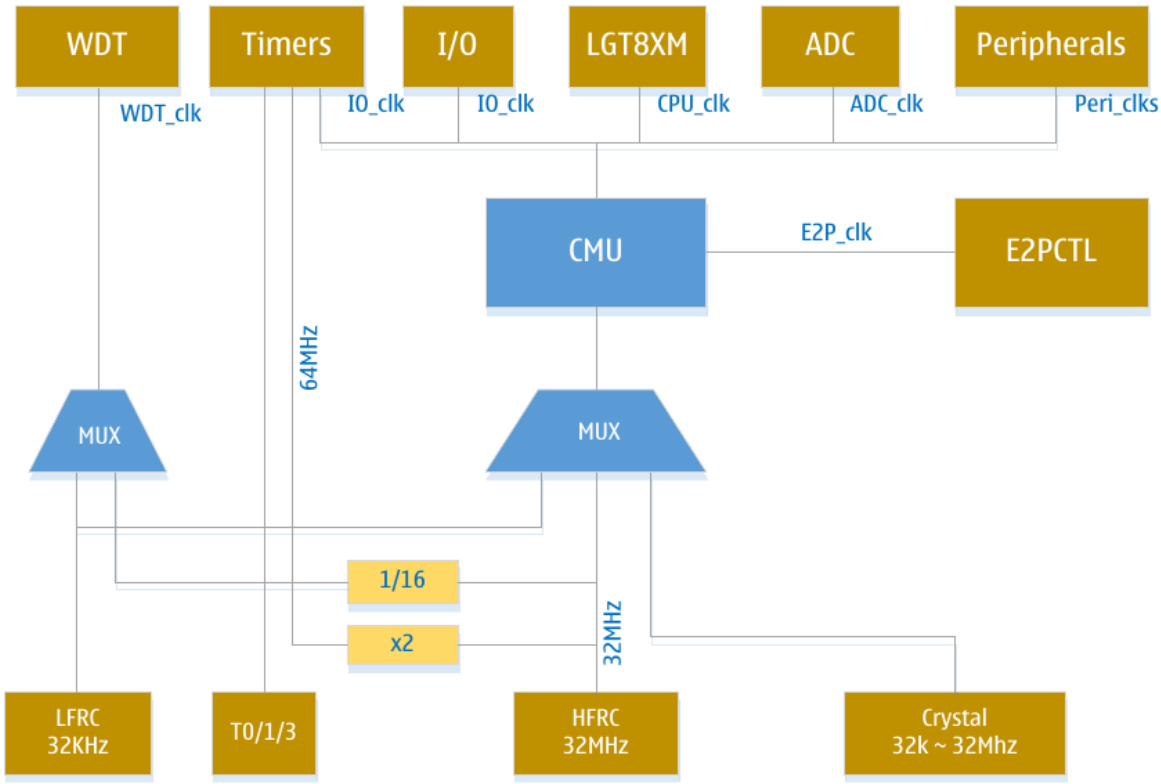
### *General I/O Register-GPIOR0*

| GPIOR1-General I/O Register 0 | | |
|---|---|---|
| GPIOR1: 0x1E (0x3A) | | Default value: 0x00 |
| Bits | GPIOR0[7:0] | |
| R/W | R/W | |
| Initial val. | 0x00 | |
| Bit definition | | |
| [7:0] | GPIOR0 | General I/O register 0, used to store user-defined data |

# System clock and configuration

## System clock distribution

LGT8FX8P supports multiple clock inputs. The system can work with three main clock sources, namely the internal 32KHz calibratable RC oscillator, the internal 32MHz calibratable RC oscillator and the external 400KHz ~ 32MHz crystal input. The following figure shows the LGT8FX8P clock system distribution. CMU is the centre of the entire clock management, responsible for the frequency division of the system clock, generating independent clocks for different modules, and controlling the clocks. In general applications, it is not necessary for all clocks to work at the same time. In order to reduce system power consumption, system power consumption management shuts off unused module clocks according to different sleep modes. For specific operation details, please refer to the relevant chapters on power consumption management.




***CPU_clk***

It is used to drive the operation of LGT8XM core and SRAM. Such as driving general working registers, status registers, etc. After the CPU clock is stopped, the core will not continue to execute instructions and perform calculations. After the system executes the SLEEP instruction to enter the sleep mode, the core clock will be turned off.

Used to drive the operation of the LGT8XM core and SRAM, such as driving general working registers, status registers and so on. After the system executes the SLEEP instruction to enter sleep mode, the core clock will be turned off, the CPU clock is stopped and the core will not continue to execute instructions and perform calculations.

***Peri_clk***

Used to drive most peripheral modules, such as timer/counter, SPI, USART, etc. The IO clock is also used to drive external interrupt modules. When the peripheral clock is stopped due to sleep, some peripheral parts that can be used to wake up the system work in independent clock or asynchronous mode. For example, the address recognition function of TWI can wake up most sleep modes, and the address recognition part works in asynchronous mode at this time.

***E2P_clk***

E2P_clk clock is used to generate FLASH interface access timing. E2P_clk generates the sequence of accessing E2PCTL to the FLASH interface. E2P_clk is fixed from the internal 32MHz HFRC oscillator divided by 32 (1MHz). If the user needs to use the E2PCTL module to read and write the internal program FLASH or data FLASH space, the internal 32MHz oscillator needs to be enabled in advance.

***Asy_clk***

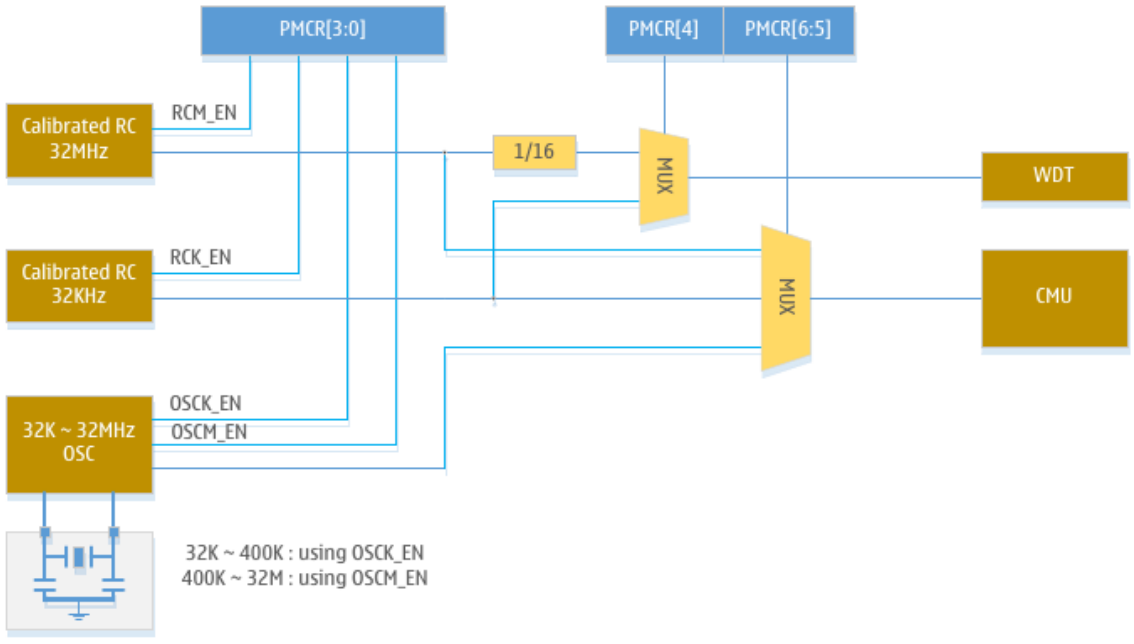
Asynchronous timer clock. The timer/counter can be directly driven by an external clock or crystal oscillator (32.768K). This independent clock mode can keep the timer running while the system is processing sleep mode.

***WDT_clk***

The internal watchdog timer clock source can be configured to select the internal 32KHz LFRC oscillator, or from the internal 32MHz HFRC divided by 16 (2MHz). After the system is powered on, the default clock source of the watchdog is 32KHz LFRC oscillator.

## Clock source selection

LGT8FX8P supports 4 kinds of clock source input, the user can enable and control the clock source and switch the desired main clock through via the PMCR register. The following is the control structure diagram of PMCR:




The internal OSC oscillator of LGT8FX8P can work in both high frequency and low frequency modes. The user needs to control the internal OSC oscillator to work in the correct mode according

to the actual size of the external crystal oscillator. The internal RC oscillator is also divided into high frequency and low frequency. The lowest 4 bits of the PMCR register are used to control these four clock sources. The control relationship is as follows:

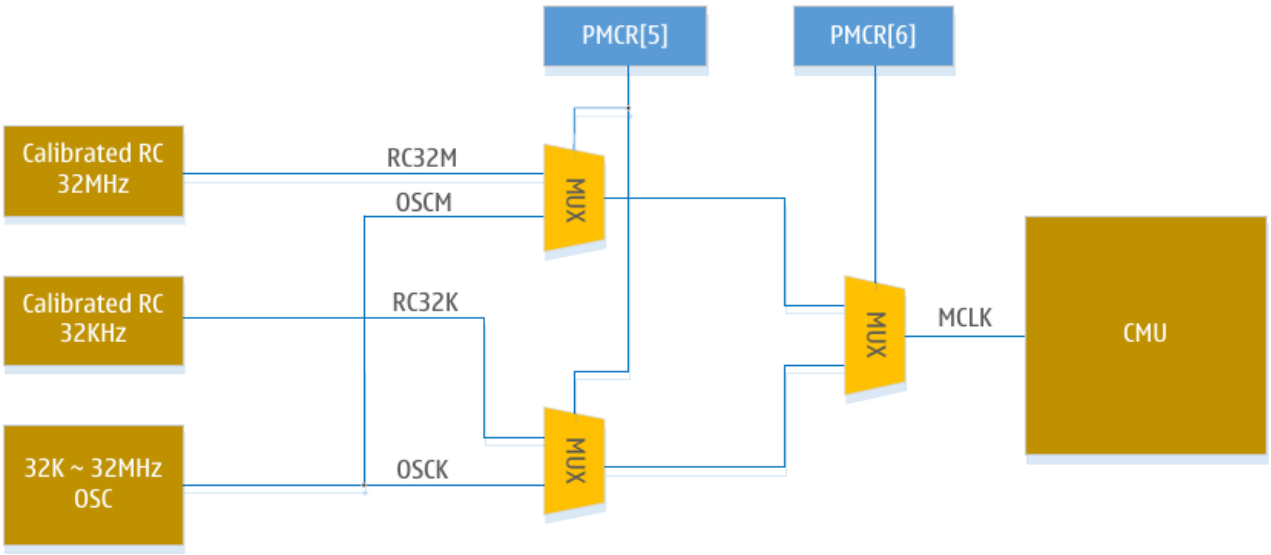
| PMCR | Corresponding clock source |
|---|---|
| PMCR[0] | 32MHz RC enable control, 1 enable, 0 disable |
| PMCR[1] | 32MHz RC enable control, 1 enable, 0 disable |
| PMCR[2] | 400K ~ 32MHz OSC mode enable, 1 enable, 0 disable |
| PMCR[3] | 32K ~ 400K OSC mode enable, 1 enable, 0 disable |

After the LGT8FX8P system is powered on, the 32MHz RC is used as the system clock source by default, and the core works at the clock source divided by 8 (4MHz). The user can change the default configuration by setting the PMCR register and the system prescaler register (CLKPR).

If the user needs to change the configuration of the main clock source, it is necessary to ensure that the switched clock source is in a stable working state before switching the clock. Therefore, it is necessary to enable the required clock source through PMCR[3:0] before switching the main clock source, and wait until the clock is stable before switching.

When the user switches the main clock to an external crystal oscillator, although the user has enabled the external crystal oscillator, it cannot be ruled out that the crystal oscillator cannot start oscillating due to configuration errors or failure of the crystal oscillator. If you switch to an external crystal oscillator at this time, the system will stop working after the switch. Therefore, from the perspective of system reliability, it is recommended to turn on the watchdog timer to avoid such problems from the perspective of software design.

After the clock source is enabled and waiting to stabilize, the main clock can be switched through PMCR[6:5]. PMCR[5] is used to select internal RC oscillator and external crystal oscillator, PMCR[6] is used to select high-speed clock source and low-speed clock source.



Main clock source selection:

| PMCR[6] | PMCR[5] | Main clock source |
|---|---|---|
| 0 | 0 | Internal 32MHz RC oscillator (system default) |
| 0 | 1 | External 400K ~ 32MHz high-speed crystal oscillator |
| 1 | 0 | Internal 32KHz RC oscillator |
| 1 | 1 | External 32K ~ 400KHz low-speed crystal oscillator |

### *Clock source control timing*

In order to protect the PMCR register from being accidentally modified, the modification of the PMCR register needs to strictly install the specified timing sequence. The highest bit of the PMCR register (PMCR[7]) is used to implement timing control. Before modifying the other bits of PMCR, the user must first set PMCR[7] to 1, and change the value of other PMCR registers within 6 cycles after the set operation. After 6 cycles, the direct modification of PMCR will be invalid.

The following is an example of switching to an external high-speed crystal oscillator to list the recommended operating steps:

1. Enable clock source
   - Set PMCR[7] = 1
   - In six cycles, set PMCR[2] = 1, enable external high-speed mode external crystal oscillator
   - Wait for the external crystal oscillator to stabilize (waiting time varies with crystal oscillators, generally μs level waits)
2. Switch the main clock source
   - Set PMCR[7] = 1
   - In six cycles, set PMCR[6:5] = 01, and the system will automatically switch the working clock to the external crystal oscillator
   - Perform several NOP operations to improve stability (optional operation)

***[Note]:*** *In the above operation of switching the main clock, ensure that the current system clock works normally. After switching to the external crystal oscillator, the previous internal RC oscillator can be turned off.*

## System clock prescaler control

There is a system clock prescaler inside LGT8FX8P, which can be controlled by the clock prescaler register (CLKPR). This function can be used to reduce system power consumption when the system does not require very high processing power. The prescaler setting is valid for all clock sources supported by the system. The clock prescaler can affect the core execution clock and all synchronized peripherals.

When switching between different clock prescaler settings, the system clock prescaler ensures that there will be no glitches during the switching process, but will also ensure that there will be no intermediate states of excessive high frequency. The frequency division switch is executed immediately. When the register change takes effect, the system clock is switched to the new frequency division clock after at most 2~3 current system clock cycles.

In order to avoid misoperation of the clock divider register, the modification of CLKPR must also follow a special timing sequence:

- Set the clock prescaler change enable flag (CLKPCE) to 1, CLKPR and other flags to 0
- In four cycles, write the required value to CLKPS, and write 0 to CLKPCE at the same time. Before changing the clock prescaler register, the interrupt function needs to be disabled to ensure that the write sequence can be completed.

For the specific definition of the master clock prescaler register CLKPR, please refer to the register description part of this chapter.

## Internal RC oscillator calibration

LGT8FX8P contains two calibratable RC oscillators, after calibration, they can achieve accuracy within ±1%. The 32MHz RC is used as the system working clock by default.

When the LGT8FX8P was manufactured, the internal 32MHz HFRC and 32KHz LFRC were calibrated and the calibration value was written into the system configuration information area. In the process of system power saving, these calibration values will be read into the internal register, and the RC frequency can be recalibrated through the register.

The calibration register is located in the IO address space, which can be read and written by the user program. For applications with special requirements for frequency, the frequency output of the internal oscillator can be adjusted by modifying the calibration register. Modifying the calibration register will not change the factory configuration information. When the system is re-powered or the configuration bit reload operation is initiated by the user, the calibration register will be restored to the factory settings.

## Register definition

### *32MHz HFRC Oscillator Calibration Register-RCMCAL*

| RCMCAL – 32MHz HFRC calibration register | | |
|---|---|---|
| RCMCAL: 0x66 | | Default value: factory configuration |
| Bits | RCCAL[7:0] | |
| R/W | R/W | |
| Bit definition | | |
| [7:0] | RCCAL | After the system is powered on, the value of the register will be replaced by the RC calibration value in the system configuration information. |

### *32KHz RC Oscillator Calibration Register-RCKCAL*

| RCKCAL – 32MHz RC calibration register | | |
|---|---|---|
| RCKCAL: 0x67 | | Default value: factory configuration |
| Bits | RCKCAL[7:0] | |
| R/W | R/W | |
| Bit definition | | |
| [7:0] | RCKCAL | Write the calibration value into the RCKCAL register to complete the calibration of the 32KHz RC oscillator |

### *Clock Source Management Register-PMCR*

| PMCR-clock source management register | | | | | | | |
|---|---|---|---|---|---|---|---|
| PMCR: 0xF2 | | | | Default value: 0x03 | | | |
| Bits | PMCE | CLKFS/CLKSS | WCLKS | OSCKEN | OSCMEN | RCKEN | RCMEN |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | |

| | | |
|---|---|---|
| [0] | RCMEN | Internal 32MHz RC oscillator enable control, 1 enable, 0 disable |
| [1] | RCKEN | Internal 32KHz RC oscillator enable control, 1 enable, 0 disable |
| [2] | OSCMEN | External high frequency crystal oscillator control, 1 enable, 0 disable |
| [3] | OSCKEN | External low frequency crystal oscillator control, 1 enable, 0 disable |
| [4] | WCLKS | WDT clock source selection,<br>0 – Select the internal 32MHz HFRC oscillator divided by 16<br>1-Internal 32KHz LFRC oscillator |
| [5] | CLKSS | Main clock source selection control, select the clock source type, please refer to the clock source selection section |
| [6] | CLKFS | Main clock source frequency control, select the clock frequency type, please refer to the clock source selection section |
| [7] | PMCE | The PMCR register changes the enable control flag.<br>Before changing other positions of PMCR, you must first set this flag, and then set the value of other bits within four cycles. |

### *Master clock prescaler register-CLKPR*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CLKPR: 0x61 | | | | | Default value: 0x03 | | | |
| Bits | WCE | CKOEN1 | CKOEN0 | – | PS3 | PS2 | PS1 | PS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| [3:0] | CLKPS | Clock prescaler selection bits | | | | |
| | | PS3 | PS2 | PS1 | PS0 | Freq. div. parameters |
| | | 0 | 0 | 0 | 0 | 1 |
| | | 0 | 0 | 0 | 1 | 2 |
| | | 0 | 0 | 1 | 0 | 4 |
| | | 0 | 0 | 1 | 1 | 8 (default) |
| | | 0 | 1 | 0 | 0 | 16 |
| | | 0 | 1 | 0 | 1 | 32 |
| | | 0 | 1 | 1 | 0 | 64 |
| | | 0 | 1 | 1 | 1 | 128 |
| | | 1 | 0 | 0 | 0 | 256 |
| | | Other value | | | | Reserved |
| [4] | – | Keep unused | | | | |
| [5] | CKOEN0 | Set whether the system clock is output on the PB0 pin | | | | |
| [6] | CKOEN1 | Set whether the system clock is output on the PE5 pin | | | | |
| [7] | WCE | Clock prescaler modification control flag.<br>Before changing the other bits of the CLKPR register, you must first set CKWEN to 1 separately, and then set the clock prescaler selection bits in the following four system cycles. After the four cycles are over, CKWEN is automatically cleared. | | | | |

# Power management

## Overview

Sleep mode reduces system power consumption by turning off the system clock and clock module. LGT8FX8P provides a very flexible and diverse sleep mode and module controller, users can implement the preferable low-power configuration according to their application.

When LGT8FX8P enters sleep mode, it will not automatically shut down the analogue function modules, such as ADC, DAC, comparator (AC), low voltage reset module (LVD) and so on. According to the application requirements, the software needs to turn off the unnecessary analogue functions before going to sleep, and restore the correct state after the system wakes up.

LGT8FX8P supports multiple sleep modes, including an ADC-specific noise cancellation mode which is used to eliminate the interference of the digital part on the ADC power supply during the ADC conversion process. In addition to this, it has five different power control modes:

| Sleep mode | Function Description |
|---|---|
| Idle mode (IDLE) | Just turn off the core clock, other peripheral modules work normally, all valid interrupt sources can wake up the core |
| Power saving mode (Save) | Same as DPS0 mode. Save mode is compatible with LGT8FX8D |
| Power down mode (DPS0) | Same as the Save mode, the supported wake-up sources include:<br>• All pin level changes<br>• Watchdog wakes up regularly<br>• TMR2 wake-up in asynchronous mode |
| Power down mode (DPS1) | Turn off all internal and external oscillators, the supported wake-up sources include:<br>• External level change of all pins<br>• External interrupt 0/1<br>• Watchdog timer working at 32K LFRC |
| Power down mode (DPS2) | Turn off the core power, the lowest power consumption mode, supported wake-up sources include:<br>• External reset<br>• PORTD pin level change<br>• LPRC timing wake-up (128ms/256ms/512ms/1s)<br>It should be noted that the process of waking up from DPS2 is the same as power-on reset |

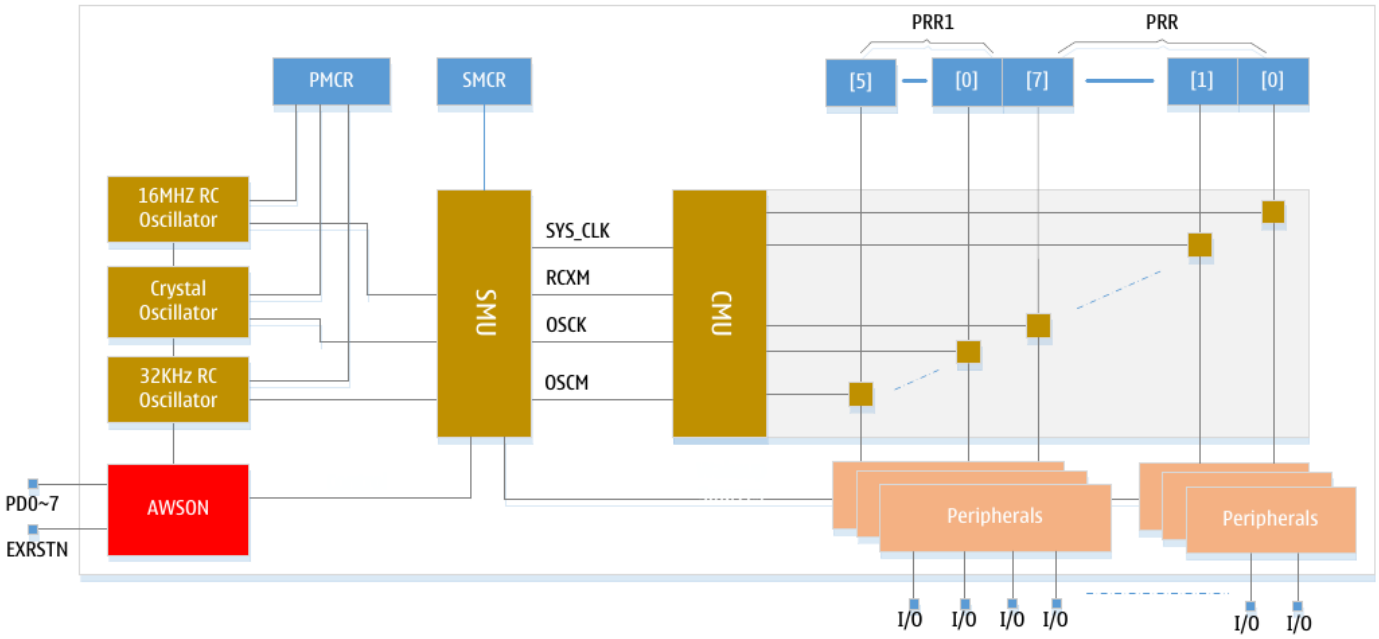
LGT8FX8P supports deep sleep DPS2. In this mode, the system’s internal LDO, the core’s registers, all peripheral controllers and SRAM are all in a power-down state, and the data in them will not be retained.

The FLASH storage unit will also be in a power-down state, so the DPS2 mode can reach the minimum power consumption of the system. Power-down mode can be awakened by port D (PORTD) pin level change, and can also choose 5-level timing wake-up. The DPS2 timer used for wake-up does not support calibration and has an accuracy of about 15%, so it is only suitable for low-precision timing wake-up applications.

When the system wakes up from DPS2 mode, the LDO will be turned on first. This process is the same as the power-on process. The chip will perform a complete power-on reset startup process, load the configuration information, and then run the program from the address pointed to by the reset vector.

In other modes except DPS2, the internal power supply will not be turned off, all register information and RAM data will be kept intact during hibernation. After waking up, the core continues execution from the last instruction before sleep.

Schematic diagram of system power management:




As shown in the figure above, LGT8FX8P mainly controls the power consumption of the entire system through the sleep mode controller (SMU) and the clock management unit (CMU). From the level of power saving, we can divide power consumption into 4 levels:

The first level is to control the working clock of the module through the PRR register, and to save the dynamic power consumption of the system's operation by turning off the clock of unused modules. Under normal circumstances, the power consumption that this level can save is not obvious.

The second stage is to switch the main clock source to the low-frequency clock, and turn off the unused clock source modules and other analogue modules. This mode can basically get a very considerable system power consumption, both while running and at sleep.

The third level is to let the system enter the power-down mode (DPS1). In DPS1 mode, LGT8FX8P can achieve excellent standby power consumption. After waking up, the software can read back the state the system had before the power-down through the MCUSR register.

The fourth level is the power-down mode (DPS2). This mode will shut down the core power and achieve the lowest system power consumption. Because the core power is turned off, all data information will be lost in this mode. Immediately after waking up, a power-on reset process is executed, and the system restarts from the reset vector.

## AWSON power management

Compared with the LGT8FX8D, the power-down mode DPS2 is a brand new power mode. DPS2 mode is used for applications with higher sleep power requirements. After entering the DPS2 mode, the system only maintains a static module (AWSON) in a working state, and other circuits are in a complete power-down state.

The AWSON module is dedicated to the sleep and wake-up control of the DPS2 mode. The AWSON module is mainly composed of IO wake-up control logic and a low-power LPRC. Software can control AWSON through IOCWK register and DPS2R register.

The IOCWK register is used to control the wake-up function of PD0~7 level changes. The DPS2R register is used to control the DPS2 mode as well as the functional mode of the LPRC. For details, please refer to the register description section at the end of this section.

Before using DPS2 mode, software sets IOCWK to enable the required wake-up IO, or enable LPRC and configure the timing wake-up cycle through the DPS2R register, and then enable DPS2 mode through the DPS2EN flag in the DPS2R register. After this setup is completed, the software needs to set the DPS2 sleep mode through the SMCR register, and then execute the SLEEP instruction to enter the sleep mode.

*This area has been intentionally left blank.*

## Sleep mode and wake-up source

LGT8FX8P supports 5 sleep modes, the user can choose the appropriate sleep mode according to the application requirements. The SMCR register contains the control settings for sleep mode. After executing the SLEEP instruction, the core enters sleep mode. In order to obtain a more ideal sleep power consumption, it is recommended to turn off all unused clocks and analog modules before the core enters sleep mode. However, it should be noted that some wake-up sources require a working clock. If you need to use this kind of wake-up source, please keep the relevant clock source running.

Sleep modes and wake-up methods:

| Sleep mode | Effective clock | | | Wake-up source | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Core clock | Peripheral clock | ADC clock | Asynchronous clock | Pin level change | External Interrupt 0/1 | TWI Address match | TMR2 Interrupt | ADC End of conversion | Watchdog overflow | Peripheral interrupt | PD Level change |
| Idle mode (IDLE) | | X | X | X | X | X | X | X | X | X | X | X |
| ADC noise rejection | | | X | X | X | X | X | X | X | X | | X |
| Power saving mode (SAVE) | | | | X | X | X | X | X | | X | | X |
| Power down mode (DPS0) (With RC32K) | | | | X | X | X | | X | | X | | X |
| Power down mode (DPS1) (Without RC32K) | | | | | | | | | | | | X |
| Power down mode (DPS2) (Without LDO) | | | | | | | | | | | | X |

If you need to enter the above five sleep modes, the SE flag in the SMCR must be set to 1 to enable sleep mode control. Then execute a SLEEP instruction. SM0/1/2 in SMCR are used to select different sleep modes. For specific information, please refer to the description below.

When the MCU is in sleep mode and the wake-up source is valid, the MCU will be awakened after 4 cycles and continue to execute instructions. If the interrupt remains active, the interrupt will also be responded immediately and the interrupt service routine will be entered. If a system reset occurs in SLEEP mode, the MCU will also wake up and start executing from the reset vector.

When the MCU is in Power down mode, the system can be awakened by the external interrupt INT0/1. After waking up, the MCU will continue to execute from the position before sleep.

## Idle mode (IDLE)

When SM2...0 is set to 000 and the SLEEP instruction is executed, the MCU enters the IDLE mode. The IDLE mode will turn off the core clock, and allows other peripherals to work normally.

IDLE mode can be awakened by external interrupts and internal interrupts. If you do not need to use the comparator and ADC as a wake-up source, it is recommended to turn them off.

The IDLE mode only shuts down the clock that runs the core, so it can't get a significant reduction in power consumption. In IDLE mode, the core will also stop executing and fetching instructions, so the operating power consumption of the internal program FLASH can be reduced.

However, IDLE mode has a more flexible wake-up mode, and users can obtain a more ideal operating power consumption by [instead] reducing the system main clock and turning off unnecessary modules.

## ADC noise suppression mode

{Write a short description of the purpose of this mode!!}

When SM2...0 is set to 001, after executing the SLEEP instruction, the MCU enters the ADC noise suppression mode. In this mode, the core and most peripherals will stop working, ADC, external interrupt, TWI address matching, WDT and timer/counter 2 working in asynchronous clock mode can all work normally.

The ADC noise constant mode is mainly used to provide a good working environment for ADC conversion. Reduces high frequency interference from digital blocks to analog conversion. After entering this mode, the ADC will automatically start the sampling conversion, and after the converted data is saved to the ADC data register, the ADC conversion end interrupt will wake up the MCU from the ADC noise mode.

## Power saving mode (Save)

When SM2...0 is set to 010, after executing the SLEEP instruction, the MCU enters the Save mode. In this mode, the system will turn off the working clocks of all modules. Because this mode turns off the working clocks of all modules, it can only be awakened through asynchronous mode. External interrupts, TWI address matching and WDT working in independent clock source mode can all generate the wake-up signal in this mode.

This mode can turn off all modules except the main clock source. In order to achieve a more ideal operating power consumption, it is recommended to switch the system main clock to the internal 32K RC or external 32KHz low-frequency crystal before entering this mode, and then turn off all unused clock sources and analogue modules.

## Power down mode DPS0

When SM[2:0] is set to 110 the MCU will enter the DPS0 mode after executing the SLEEP instruction. After entering DPS0, all clock sources are turned off except the internal 32KHz RC. This mode can be woken up by external interrupt INT0/1; if the interrupt function of WDT is enabled, it can also be woken up regularly by WDT.

## Power down mode DPS1

When SM[2:0] is set to 011, the MCU will enter the DPS1 mode after executing the SLEEP instruction. After entering DPS1, all clock sources of the system are turned off. This mode can use the level change of IO to wake up the watchdog.

## Power down mode DPS2

Set SM[2:0] to 111, and enable the AWSON module through DPS2EN of the DPSR2 register. After executing the SLEEP instruction, it will enter the DPS2 mode. After entering DPS2 mode, the system shuts down the core power, meaning all registers and RAM data will be lost. The wake-up process from DSP2 is the same as the power-on reset process.

In DPS2 mode, because the core voltage is turned off and the register information is lost, the control state of the port will also be restored to the input state, and all IO output drivers and pull-up controls will also be turned off.

## FLASH power control and fast wake-up

When the system is in SLEEP mode, the core will not continue to execute instructions. At this time, you can choose to turn off the power of FLASH to obtain lower standby power consumption. This function can be controlled by the FPDEN flag of the MCUCR register;

In power-down mode, the system can use external interrupt or WDT to wake up. In order to filter out possible interference from external signals, the internal wake-up circuit includes a configurable filter circuit. Users can select the appropriate filter width according to their needs. The configuration of the filter circuit can be realized by FWKPEN in MCUCR register.

MCUCR[FWKPEN] filter width control:

| FWKPEN | Filter width |
|---|---|
| 0 | 260μs (default) |
| 1 | 32μs |

## Register description

### *Sleep Mode Control Register-SMCR*

| SMCR-Sleep Mode Control Register | | | | | | |
|---|---|---|---|---|---|---|
| SMCR: 0x33(0x53) | | | Default value: 0x00 | | | |
| Bits | – | | SM2 | SM1 | SM0 | SE |
| R/W | – | | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | |
| [0] | SE | After the sleep mode enable control flag is set to 1, execute the SLEEP instruction and the core will enter the sleep mode. The SE flag protects the system from accidentally entering sleep mode. After waking up, it is recommended to clear the SE flag immediately. | | | | |
| [3:1] | SM | Sleep mode selection | | | | |
| | | SM2 | SM1 | SM0 | Mode description | |
| | | 0 | 0 | 0 | IDLE mode | |
| | | 0 | 0 | 1 | ADC noise suppression mode | |
| | | 0 | 1 | 0 | Save mode | |
| | | 0 | 1 | 1 | DPS1 mode | |
| | | 1 | 1 | 0 | DPS0 mode | |
| | | 1 | 1 | 1 | DPS2 mode | |
| | | Others | | | Keep unused | |
| [7:4] | – | Keep unused | | | | |

### *Power Saving Control Register-PRR*

| PRR-Power Saving Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PRR: 0x64 | | | | | Default value: 0x00 | | | |
| PRR | PRTWI | PRTIM2 | PRTIM0 | – | PRTIM1 | PRSPI | PRUART0 | PRADC |
| R/W | R/W | R/W | R/W | – | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | | |

| | | |
|---|---|---|
| [0] | PRADC | Set to 1, turn off the ADC controller clock |
| [1] | PRUART0 | Set to 1, turn off the clock of the USART0 module |
| [2] | PRSPI | Set to 1, turn off the clock of the SPI module |
| [3] | PRTIM1 | Set to 1, turn off the timer/counter 1 clock |
| – | – | Keep unused |
| [5] | PRTIM0 | Set to 1, turn off the timer/counter 0 clock |
| [6] | PRTIM2 | Set to 1, turn off the timer/counter 2 clock |
| [7] | PRTWI | Set to 1, turn off the clock of the TWI module |

### *Power Saving Control Register-PRR1*

| PRR1-Power Saving Control Register 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PRR1: 0x65 | | | Default value: 0x00 | | | | | |
| PRR1 | | | PRWDT | – | PRTIM3 | PREFL | PRPCI | – |
| R/W | | | R/W | – | R/W | R/W | R/W | – |
| Bit definition | | | | | | | | |
| [0] | – | Keep unused | | | | | | |
| [1] | PRPCI | Set to 1, turn off external interrupts and external interrupt module clock | | | | | | |
| [2] | PREFL | Set to 1, turn off the FLASH controller interface clock | | | | | | |
| [3] | PRTIM3 | Set to 1, turn off the clock of the TMR3 controller | | | | | | |
| [4] | – | Keep unused | | | | | | |
| [5] | PRWDT | Set to 1, turn off the WDT counter clock | | | | | | |
| [7:6] | – | Keep unused | | | | | | |

### *MCU control register-MCUCR*

| MCUCR – MCU control register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| MCUCR: 0x35(0x55) | | | | Default value: 0x00 | | | | |
| MCUCR | FWKEN | FPDEN | EXRFD | PUD | IRLD | IFAIL | IVSEL | WCE |
| R/W | R/W | R/W | R/W | R/W | W/O | R/O | R/W | R/W |
| Bit definition | | | | | | | | |
| [0] | WCE | MCUCR update enable flag. Before updating MCUCR you need to set this flag first, and then update the MCUCR register within 6 cycles | | | | | | |
| [1] | IVSEL | Interrupt vector selection flag. After this flag is set to 1, the interrupt vector address will be mapped to a new address according to the value of the IVBASE register | | | | | | |
| [2] | IFAIL | System configuration load failure flag,<br>0 = configuration information verification passed<br>1 = Failed to load configuration information | | | | | | |
| [3] | IRLD | Writing 1 will reload the system configuration information | | | | | | |
| [4] | PUD | Global pull-up prohibition flag<br>0 = Enable global pull-up control<br>1 = Disable all IO pull-up resistors | | | | | | |
| [5] | EXRFD | External reset filter disable flag<br>0 = enable external reset (190us) digital filter<br>1 = Disable the digital filter circuit for external reset | | | | | | |
| [6] | FPDEN | Flash Power/down-enable control flag<br>0: FLASH remains powered on after the system is SLEEP<br>1: FLASH power off after system SLEEP | | | | | | |
| [7] | FWKEN | Fast wake-up mode enable control, only valid for power down mode<br>0: 260μs filter delay<br>1: 32μs filter delay | | | | | | |

### *PD group wake-up on change control register-IOCWK*

| IOCWK-PD group wake-up on change control register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IOCWK: 0xAE | | | | Default value: 0x00 | | | | |
| Bits | IOCD7 | IOCD6 | IOCD5 | IOCD4 | IOCD3 | IOCD2 | IOCD1 | IOCD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | | |
| [7:0] | IOCWK | Set the corresponding flag to enable the IO pin change wake-up function of the PD group | | | | | | |

### *DPS2 Mode Control Register-DPS2R*

| DPS2R-DPS2 mode control register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DPS2R: 0xAF | | | | Default value: 0x00 | | | | |
| Bits | – | – | – | – | DPS2E | LPRCE | TOS1 | TOS0 |
| R/W | – | – | – | – | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | | |
| [1:0] | TOS | LPRC timing wake-up setting:<br>00 = 128ms<br>01 = 256ms<br>10 = 512ms<br>11 = 1s | | | | | | |
| [2] | LPRCE | LPRC enable control<br>0 = Disable LPRC timer<br>1 = enable LPRC timer | | | | | | |
| [3] | DPS2E | DPS2 mode enable control flag<br>0 = Disable DPS2 mode<br>1 = Enable DPS2 mode | | | | | | |
| [7:4] | – | Reserved | | | | | | |

# System control and reset

## Overview

After the system is reset, all I/O registers will be set to their initial values, and the program will be executed from the reset vector. On the interrupt vector address of LGT8FX8P, a RJMP-relative jump instruction must be used to jump to the reset handler. If the program does not use interrupts and the interrupt source is not enabled, the interrupt vector will not be used, and the interrupt vector area can be used to store the user's program code.

After the reset is effective, all I/O ports immediately enter their initial state. The initial state of most I/Os is input and the internal pull-up resistor is turned off. I/O with analogue input function is also initialized as digital I/O function.

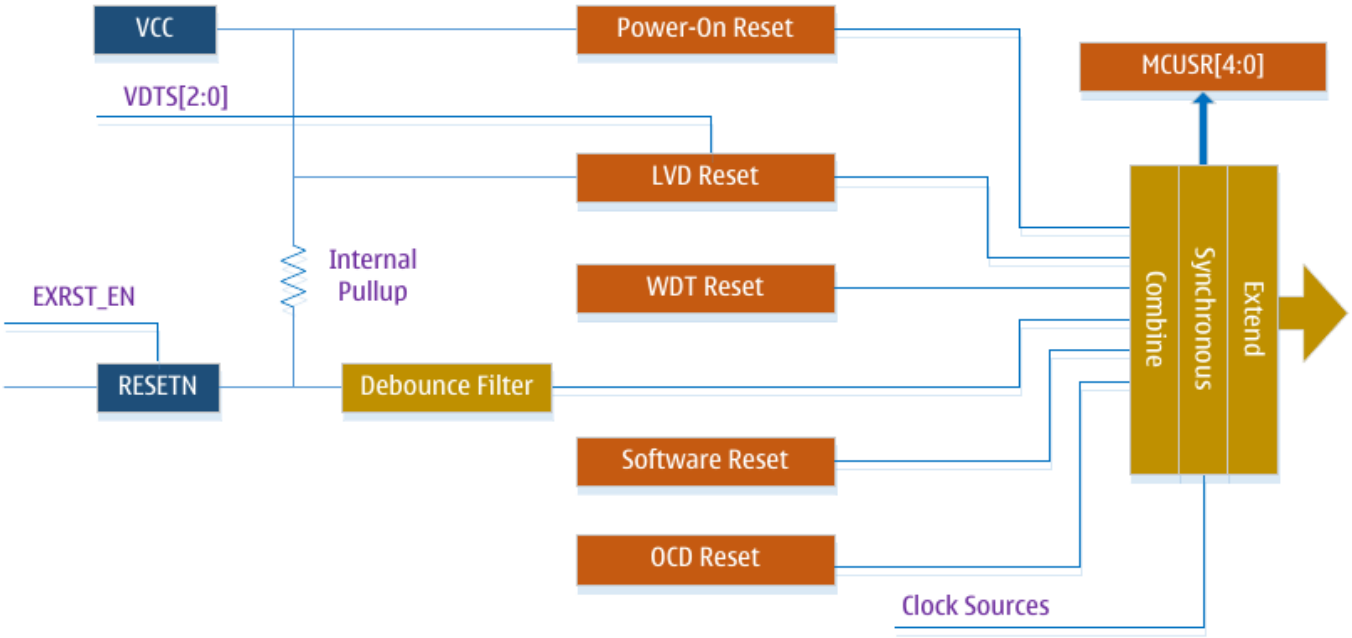
When the initial reset puls (or signal) ends, the LGT8FX8P has an internal time counter that extends the reset signal. Extending the width of the reset signal is used to ensure that the power and clock modules in the system enter a stable state.

## Reset source

LGT8FX8P supports a total of six reset sources:

- Power-on reset: Is activated when the working voltage of the system is lower than the reset threshold of the internal POR module.
- External reset: Is activated when a low-level pulse with a defined minimum width occurs on the external reset pin of the chip.
- Watchdog reset: The system will reset if a watchdog module is enabled and the watchdog timer runs out.
- Low voltage reset: There is a low voltage detection module (LVD) inside LGT8FX8P. When the system’s operating voltage is lower than the reset threshold, set by LVD, the MCU will be reset.
- Software reset: There is a dedicated software-triggered reset register inside LGT8FX8P. The user can reset the MCU at any time through this register.
- OCD reset: OCD reset is issued by the debugger module to directly reset the MCU core.

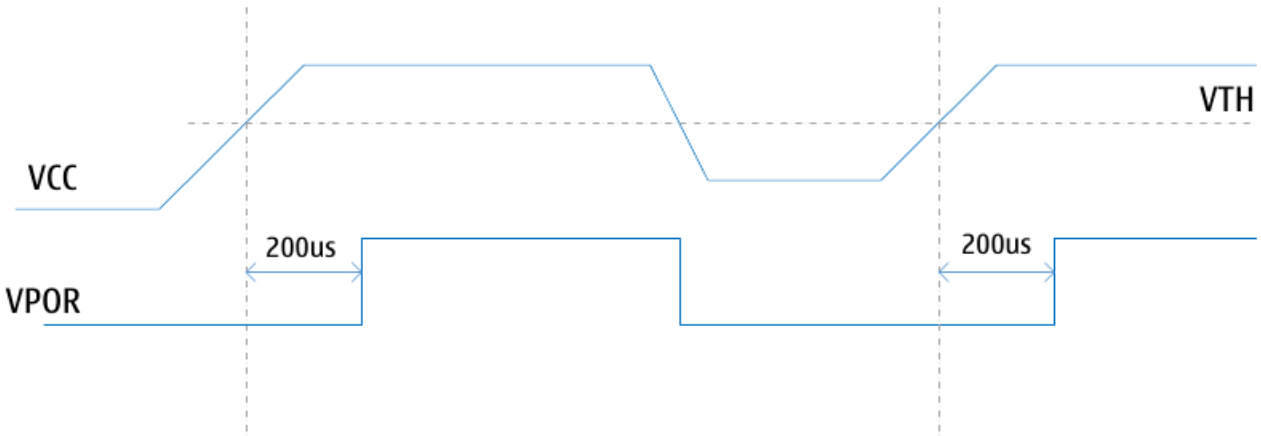
Reset system structure diagram:




### *Power-on reset*

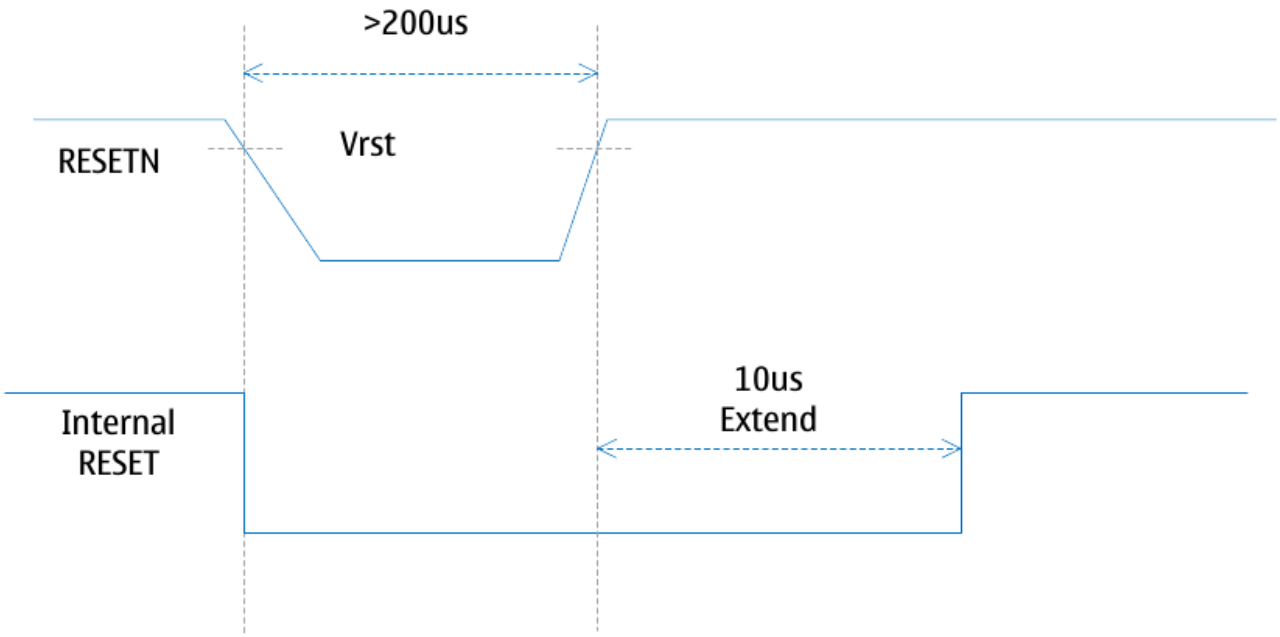
The power-on reset signal is generated by the internal voltage detection circuit. When the system power (VCC) is lower than the detection threshold, the power-on reset signal is triggered. For the detection threshold of power-on reset, please refer to the electrical parameters section.

The power-on reset circuit can ensure that the chip is in the reset state during the power-on process, and the chip can start to operate from a known (stable) state after power-on. The power-on reset signal will also be expanded by the counter inside the chip to ensure that various internal analogue modules, such as the RC oscillator, can enter a stable working state after power-on.



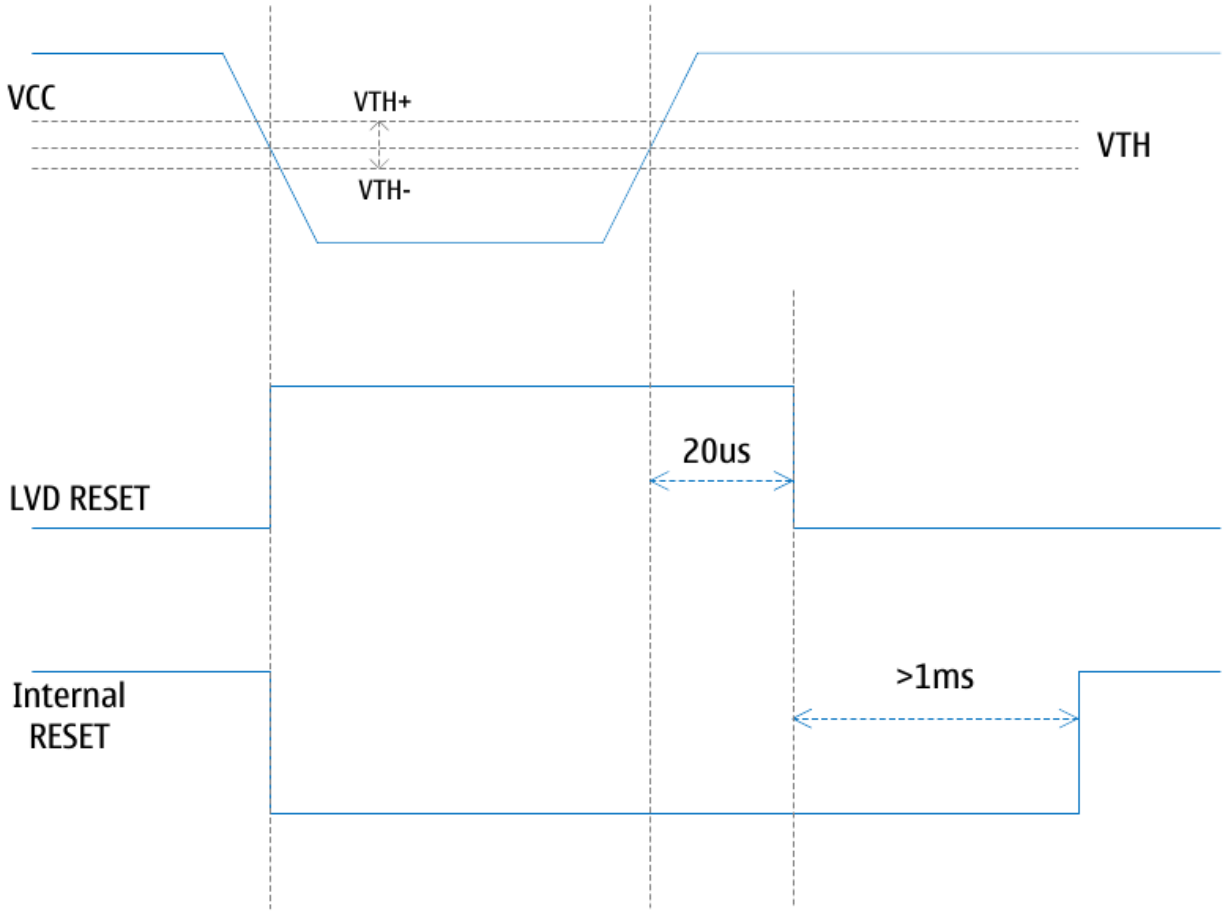
### *External reset*

When a low level pulse is applied to the external reset pin (RSTN) and the width of the low level puls is greater than the minimum reset pulse width requirement, the external reset is effective immediately. The external reset is an asynchronous reset. Even if the chip does not have a clock, the external reset can still reset the chip. The external reset pin of LGT8FX8P can also be used as a general-purpose I/O. After the chip is powered on, it defaults to an external reset function. The user can turn off the external reset function of this pin through register configuration, so it can be used as a normal I/O. For specific usage, please refer to the description of the IOCR register.



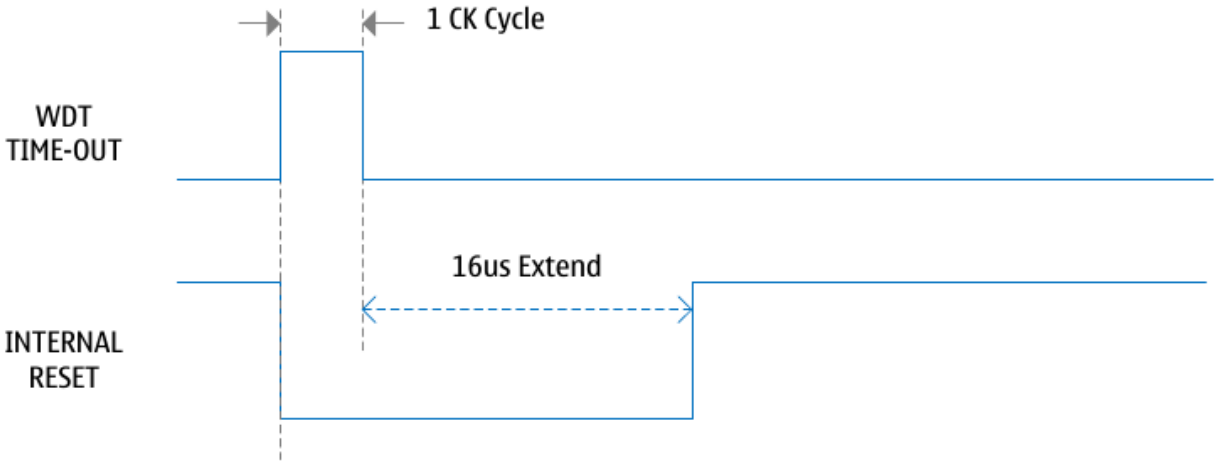

### *Low voltage detection (LVD) reset*

LGT8FX8P contains a programmable low voltage detection (LVD) circuit. LVD monitors voltage changes of VCC, but unlike the power-on reset user can modify the triggering threshold voltage of the LVD by programming. The user can select between different voltage thresholds by directly operating the VDTCR register. The LVD voltage detection circuit has a hysteresis characteristic of ±10mV~±50mV, which is used to filter out the jitter of the VCC voltage. When LVD is enabled, if the voltage of VCC drops to the set reset threshold, LVD reset will be effective immediately. When VCC increases above the reset threshold, the internal reset expansion circuit is activated, and the reset pulse will expand to at least 1 millisecond.




### *Watchdog reset*

When the watchdog timer overflows, if the watchdog system reset function is enabled, a single system reset pulse will be generated immediately. The common watchdog reset signal is also expanded by the internal delay counter. For detailed operation of the watchdog controller, please refer to the detailed introduction section below.




### *Software reset, OCD reset*

The software reset is triggered by the user by operating the sixth bit of the VDTCR register. The timing requirements for the software reset pulse is identical to the watchdog reset. The software reset pulse is internally extended by 16us.

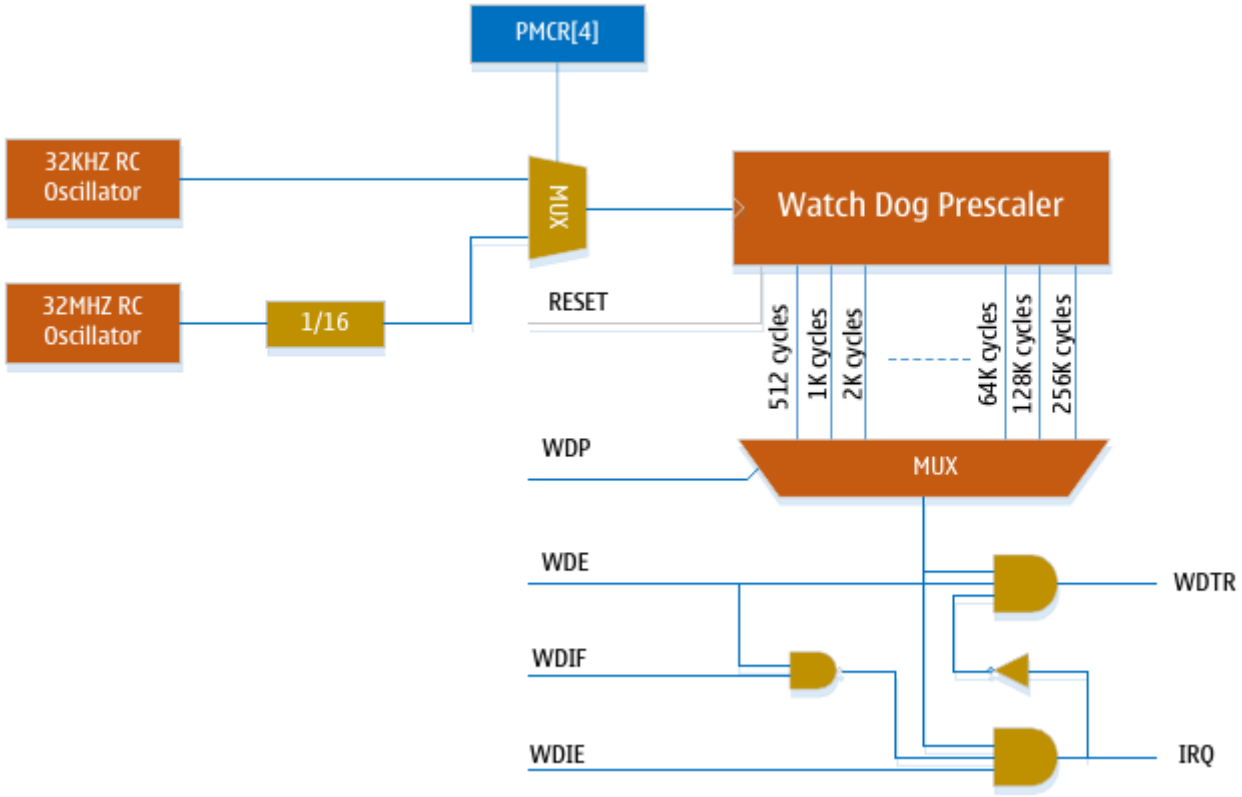
The OCD reset is generated by the debugger unit inside the chip. The OCD reset is generally controlled by the debugger, and the user software cannot trigger an OCD reset.

## Watchdog timer

- Clock can be selected as internal 32KHz RC or internal 32MHz RC divided by 16 (2MHz)
- Support for interrupt mode, reset mode and reset interrupt mode
- The timer timeout can be up to 8 seconds

The LGT8FX8P contains an enhanced Watchdog Timer (WDT) module inside. The working clock of the WDT timer can be the internal 32KHz RC oscillator, or the internal 32MHz RC oscillator can be divided by 16 (ie. 2MHz). After the WDT counter overflows, it can output an interrupt or a system reset signal. In normal use, software is required to execute a WDR - Watchdog Timer Reset instruction to restart the counter before overflowing. If the system does not immediately execute the WDR instruction, the WDT will generate an interrupt or system reset.

The structure diagram of the watchdog timer is shown in the following figure:




In interrupt mode, an interrupt request signal will be generated after the WDT overflows. This interrupt can be used as a wake-up signal from sleep mode or as a general system timer. For example, you can use this interrupt to limit the execution time of an operation, and terminate a current task in overflow. In system reset mode, the WDT generates a system reset signal immediately after the counter overflows. The most typical use is to prevent the system from crashing or running away. The third mode, the reset interrupt mode, combines the functions of interrupt and reset. First, the system will respond to the WDT interrupt function, and immediately switch to the

reset mode after exiting the WDT interrupt reset program. This function can support saving some critical parameter information before reset.

To prevent the WDT from being accidentally disabled, shutting down the WDT must follow a well-defined timing. The following code describes how to turn off the watchdog timer. The following example assumes that interrupts have been disabled so that the entire flow of operation is not interrupted.

Example code for watchdog enable and shutdown operations:

Assembly code

```
WDT_OFF:
    ; Turn off global interrupt
    CLI
    ; Reset watchdog timer
    WDR
    ; Clear WDRF in MCUSR
    IN r16, MCUSR
    ANDI r16, ~(1 << WDRF)
    OUT MCUSR, r16
    ; Write logical one to WDCE and WDE
    ; Keep old Prescaler setting to prevent unintentional time-out
    LDS r16, WDTCSR
    ORI r16, (1 << WDCE) | (1 << WDE)
    STS WDTCSR, r16
    ; Turn off WDT
    LDI r16, (0 << WDE)
    STS WDTCSR, r16
    ; Turn on global interrupt
    SEI
    RET
```

C code

```
void WDT_OFF(void)
{
    __disable_interrupt();
    __watchdog_reset();
    /* Clear WDRF in MCUSR */
    MCUSR &= ~(1 << WDRF);
    /* Write logical one to WDCE and WDE */
    /* Keep old Prescaler setting to prevent unintentional time-out */
    WDTCSR |= (1 << WDCE) | (1 << WDE);
    /* Turn off WDT */
    WDTCSR = 0x00;
    __enable_interrupt();
}
```

[Note]
If the WDT isn't disabled and the program gets lost in a loop condition, the WDT will overflow and reset the chip. Since a reset doesn't deactivate the WDT, this will cause a loop reset if the WDT isn't

explicitly handled in user code. To avoid this situation, it is recommended that user software clear the Watchdog Reset Flag (WDRF) and WDE control flags in the initialization routine.

The following code describes how to change the watchdog timer timeout value.

Assembly code

```
WDT_TOV_Change:
    ; Turn off global interrupt
    CLI
    ; Reset watchdog timer
    WDR
    ; Start timed sequence
    LDS r16, WDTCSR
    ORI r16, (1 << WDCE) | (1 << WDE)
    STS WDTCSR, r16
    ; -- Got for cycles to set the new value from here --
    ; Set new time-out value = 64k cycles
    LDI r16, (1 << WDE) | (1 << WDP2) | (1 << WDP0)
    STS WDTCSR, r16
    ; -- Finished setting new value, used 2 cycles -
    ; Turn on global interrupt
    SEI
    RET
```

C code

```
void WDT_TOV_Change(void)
{
    __disable_interrupt();
    __watchdog_reset();
    /* Start timed sequence */
    WDTCSR |= (1 << WDCE) | (1 << WDE);
    /* Set new time-out value = 64K cycles */
    WDTCSR |= (1 << WDE) | (1 << WDP2) | (1 << WDP0);
    __enable_interrupt();
}
```

【Note】

Before changing the WDP configuration bits, it is recommended to reset the watchdog timer: Changing the WDP bits to a smaller time-out period is likely to cause a watchdog time-out reset.

## Register definition

### *Low Voltage Detection (LVD) Control Register - VDTCR*

| VDTCR – LVD Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| VDTCR: 0x62 | | | | Default value: 0x00 | | | | |
| Bits | WCE | SWR | - | VDTS2 | VDTS1 | VDTS0 | VDREN | VDTEN |
| R/W | R/W | W/R | - | R/W | R/W | R/W | R/W | R/W |

| Bit definition | | |
|---|---|---|
| [0] | VDTEN | Low voltage detection module enable control, 1 enable, 0 disable |
| [1] | VDREN | Low voltage reset function enable control, 1 enable, 0 disable |
| [4:2] | VDTS | Low Voltage Detection Threshold Configuration flags<br>000 = 1.8V<br>001 = 2.2V<br>010 = 2.5V<br>011 = 2.9V<br>100 = 3.2V<br>101 = 3.6V<br>110 = 4.0V<br>111 = 4.4V |
| [5] | - | Reserved |
| [6] | SWR | Soft reset enable flag, clearing this flag will generate a software reset |
| [7] | WCE | VDTCR Value Change Enable Bit<br>Before changing the value of the VDTCR register, the user must first write this flag to 1, and in the following 6 clock cycles, change the value of other bits of VDTCR. After four cycles, WCE is automatically cleared, and the update operation of the VDTCR register is invalid. |

### *IO Function Multiplexing Register - PMX2*

| PMX2 – IO function multiplexing register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PMX2: 0xF0 | | | | Default value: 0x00 | | | | |
| Bits | WCE | STSC1 | STSC0 | - | - | XIEN | E6EN | C6EN |
| R/W | R/W | R/W | R/W | - | - | R/W | R/W | R/W |

| Bit definition | | |
|---|---|---|
| [0] | C6EN | The PC6 pin defaults to the reset function. Setting this flag to 1 will disable the external reset function. After the reset function is disabled, PC6 can be used as a common I/O. |
| [1] | E6EN | The PE6 pin is configured for analogue input by default. Setting this flag to 1 will turn off the analog input and enabling as a GPIO |
| [2] | XIEN | External clock input enable control |
| [4:3] | - | Reserved |
| [5] | STSC0 | Low-speed crystal oscillator startup control |

| | | |
|---|---|---|
| [6] | STSC1 | High-speed crystal oscillator startup control |
| [7] | WCE | IOCR Value Change Enable Bit<br>Before changing the value of the IOCR register, the user must first write 1 to this flag, and then change the value of other bits of the IOCR within the next 6 clock cycles. After four cycles, WCE is automatically cleared, and the update operation of the IOCR register is invalid. |

### *MCU Status Register - MCUSR*

| MCUSR – IO Special Function Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| MCUSR: 0x34(0x54) | | | | | Default value: 0x00 | | | |
| Bits | SWDD | - | PDRF | OCDRF | WDRF | BORF | EXTRF | PORF |
| R/W | R/W | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit definition | | | | | | | | |
| [0] | PORF | Power-on reset flag, write 0 to clear | | | | | | |
| [1] | EXTRF | External reset flag, automatically cleared by power-on reset, or by setting it to 0 | | | | | | |
| [2] | BORF | Low voltage detection reset, automatically cleared by power-on reset, or by setting it to 0 | | | | | | |
| [3] | WDRF | Watchdog reset flag, automatically cleared by power-on reset, or by setting it to 0 | | | | | | |
| [4] | OCDRF | OCD debugger reset flag, automatically cleared by power-on reset, or by setting it to 0 | | | | | | |
| [5] | PDRF | Wake-up flag from Power/off mode, please refer to Power Management chapter for details. | | | | | | |
| [6] | - | Reserved | | | | | | |
| [7] | SWDD | SWD interface disable flag. Setting this flag (‘1’) will turn off the SWD interface.<br>After the SWD interface is disabled, debugging and ISP operations will not be possible. If the SWD interface is being disabled in the user program, the operation of the internal program can be prohibited by pulling down RESET during the power-on process, and then debugging and ISP operations can again be performed. After the SWD interface is disabled, the two I/O interfaces occupied by the SWD can be used as general-purpose I/O. To avoid setting the SWDD flag by mistake, and thus unintentionally disabling the SWD interface, the user needs to set the SWDD flag once more within four cycles after first setting this flag. The SWD interface will not be disabled before the SWDD flag is set twice within four cycles. | | | | | | |

[Tips for use]:
In order to use the reset flag information more accurately and effectively, it is recommended that the user try to read the reset flag and then clear it in the early stage of program initialization.

## *Watchdog Control Status Register - WDTCSR*

| WDTCSR – WDT Control and Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x60 | | | | Default value: 0x00 | | | | |
| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | WDIF | WDIE | WDP3 | WDTOE | WDE | WDP2 | WDP1 | WDP0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| [7] | WDIF | WDT interrupt flag.<br>The WDIF flag is set when the WDT is operating in interrupt mode and overflows. When the WDT interrupt enable flag WDIE is "1" and the global interrupt is set, the WDT interrupt is generated on WDT overflow. The WDIF flag is cleared when the WDT interrupt is executed, and can also be cleared by clearing ("1") the WDIF flag. |
| [6] | WDIE | WDT interrupt enable control flag.<br>When the WDIE flag is set ("1") and the global interrupt is set, the WDT interrupt is enabled, and when the WDIE flag is cleared ("0"), the WDT interrupt is disabled.<br>The WDIE flag and the WDE flag together determine the watchdog operating mode, as shown in the table below. |

| WDE | WDIE | Mode | Action after overflow |
|---|---|---|---|
| 0 | 0 | Halt | None |
| 0 | 1 | Interrupt mode | Interrupt |
| 1 | 0 | Reset mode | Reset |
| 1 | 1 | Interrupt reset mode | Reset after interrupt |

| | | |
|---|---|---|
| [5] | WDP3 | The WDT prescaler selection controls flag 3.<br>WDP[3] and WDP[2:0] form the WDT prescale factor selection flags WDP[3:0], which are used to set the WDT timeout (overflow) period. |
| [4] | WDTOE | WDT shutdown enable control flag.<br>When clearing the WDE flag, the WDTOE flag must be set, otherwise the WDT will not be disabled. When the WDTOE flag is set, the hardware will clear the WDTOE flag after 4 clock cycles. |
| [3] | WDE | WDT enable control flag.<br>When the WDE flag is set ("1"), the WDT is enabled, and when the WDE flag is cleared ("0"), the WDT is disabled.<br>WDE can only be cleared when the WDTOE flag is set. To turn off an enabled WDT, the following sequence must be followed:<br>1. Set the WDTOE and WDE flags at the same time, even if WDE has been set, the WDE flag must be set ("1") before the shutdown operation starts;<br>2. During the next 4 clock cycles, clear ('0') the WDE flag. This will |

| | | |
|---|---|---|
| | | turn off WDT.<br>The WDT reset system flag WDRF (located in the MCUSR register) will be set when the WDE flag is "1" and the WDT overflow resets the system. The WDE flag is set when the WDRF flag is set. Therefore, to clear the WDE flag, the WDRF flag must be cleared first. |
| [2:0] | WDP | WDT prescale factor selection control.<br>Used to set the WDT time-out period. It is recommended to change the value of WDP when the WDT is not counting, and changing the value of WDP during the counting process will result in unexpected WDT overflow. |

Watchdog Prescaler Selection List:

| WDP3 | WDP2 | WDP1 | WDP0 | Watchdog Timer Overflow Cycles | 32KHz clock | 2MHz clock |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2K cycles | 64ms | 1ms |
| 0 | 0 | 0 | 1 | 4K cycles | 128ms | 2ms |
| 0 | 0 | 1 | 0 | 8K cycles | 256ms | 4ms |
| 0 | 0 | 1 | 1 | 16K cycles | 512ms | 8ms |
| 0 | 1 | 0 | 0 | 32K cycles | 1s | 16ms |
| 0 | 1 | 0 | 1 | 64K cycles | 2s | 32ms |
| 0 | 1 | 1 | 0 | 128K cycles | 4s | 64ms |
| 0 | 1 | 1 | 1 | 256K cycles | 8s | 128ms |
| 1 | 0 | 0 | 0 | 512K cycles | 16s | 256ms |
| 1 | 0 | 0 | 1 | 1024K cycles | 32s | 512ms |
| 1 | 0 | 1 | 0 | Reserved | | |
| 1 | 0 | 1 | 1 | | | |
| 1 | 1 | 0 | 0 | | | |
| 1 | 1 | 0 | 1 | | | |
| 1 | 1 | 1 | 0 | | | |
| 1 | 1 | 1 | 1 | | | |

## Interrupts and Interrupt Vectors

- 28 interrupt sources
- Programmable vector start address

The interrupt resources of LGT8F88P/168P/328P are basically the same, the main difference is: the interrupt vector of LGT8F88P is a 1 instruction word (16 bits), while the interrupt vector of LGT8F168P/328P are a 2 instruction word (32 bits).

## LGT8F88P Interrupt Vector List

| Numbering | Vector address | Interrupt source signal | Interrupt Source Description |
|---|---|---|---|
| 1 | 0x0000 | RESET | External reset, power-on reset, watchdog reset, SWD debug reset, low voltage reset |
| 2 | 0x0001 | INT0 | External interrupt request 0 |
| 3 | 0x0002 | INT1 | External interrupt request 1 |
| 4 | 0x0003 | PCI0 | Interrupt on pin level 0 |
| 5 | 0x0004 | PCI1 | Interrupt on pin level 1 |
| 6 | 0x0005 | PCI2 | Interrupt on pin level 2 |
| 7 | 0x0006 | WDT | Watchdog overflow interrupt |
| 8 | 0x0007 | TC2 COMPA | Timer 2 Compare Match A Interrupt |
| 9 | 0x0008 | TC2 COMPB | Timer 2 Compare Match B Interrupt |
| 10 | 0x0009 | TC2 OVF | Timer 2 overflow interrupt |
| 11 | 0x000A | TC1 CAPT | Timer 1 input capture interrupt |
| 12 | 0x000B | TC1 COMPA | Timer 1 Compare Match A Interrupt |
| 13 | 0x000C | TC1 COMPB | Timer 1 Compare Match B Interrupt |
| 14 | 0x000D | TC1 OVF | Timer 1 overflow interrupt |
| 15 | 0x000E | TC0 COMPA | Timer 0 Compare Match A Interrupt |
| 16 | 0x000F | TC0 COMPB | Timer 0 Compare Match B Interrupt |
| 17 | 0x0010 | TC0 OVF | Timer 0 overflow interrupt |
| 18 | 0x0011 | SPI STC | SPI serial transfer end interrupt |
| 19 | 0x0012 | USART RXC | USART receive end interrupt |
| 20 | 0x0013 | USART UDRE | USART data register empty interrupt |
| 21 | 0x0014 | USART TXC | USART send end interrupt |
| 22 | 0x0015 | ADC | ADC conversion end interrupt |
| 23 | 0x0016 | EE_RDY | EEPROM ready interrupt |
| 24 | 0x0017 | ANA_COMP | Analog Comparator 0 Interrupt |
| 25 | 0x0018 | TWI | Two-wire serial interface interrupt |
| 26 | 0x0019 | ANA_COMP1 | Analog Comparator 1 Interrupt |
| 27 | 0x001A | - | Reserve |
| 28 | 0x001B | PCI3 | Pin level interrupt 3 |
| 29 | 0x001C | PCI4 | Pin level interrupt 4 |
| 30 | 0x001D | TC3_INT | Timer 3 interrupt |

## LGT8F168P/328P Interrupt Vector List

| Numbering | Vector address | Interrupt source signal | Interrupt Source Description |
|---|---|---|---|
| 1 | 0x0000 | RESET | External reset, power-on reset, watchdog reset, SWD debug reset, low voltage reset |
| 2 | 0x0002 | INT0 | External interrupt request 0 |
| 3 | 0x0004 | INT1 | External interrupt request 1 |
| 4 | 0x0006 | PCI0 | Interrupt on pin level 0 |
| 5 | 0x0008 | PCI1 | Interrupt on pin level 1 |
| 6 | 0x000A | PCI2 | Interrupt on pin level 2 |
| 7 | 0x000C | WDT | Watchdog overflow interrupt |
| 8 | 0x000E | TC2 COMPA | Timer 2 Compare Match A Interrupt |
| 9 | 0x0010 | TC2 COMPB | Timer 2 Compare Match B Interrupt |
| 10 | 0x0012 | TC2 OVF | Timer 2 overflow interrupt |
| 11 | 0x0014 | TC1 CAPT | Timer 1 input capture interrupt |
| 12 | 0x0016 | TC1 COMPA | Timer 1 Compare Match A Interrupt |
| 13 | 0x0018 | TC1 COMPB | Timer 1 Compare Match B Interrupt |
| 14 | 0x001A | TC1 OVF | Timer 1 overflow interrupt |
| 15 | 0x001C | TC0 COMPA | Timer 0 Compare Match A Interrupt |
| 16 | 0x001E | TC0 COMPB | Timer 0 Compare Match B Interrupt |
| 17 | 0x0020 | TC0 OVF | Timer 0 overflow interrupt |
| 18 | 0x0022 | SPI STC | SPI serial transfer end interrupt |
| 19 | 0x0024 | USART RXC | USART receive end interrupt |
| 20 | 0x0026 | USART UDRE | USART data register empty interrupt |
| 21 | 0x0028 | USART TXC | USART send end interrupt |
| 22 | 0x002A | ADC | ADC conversion end interrupt |
| 23 | 0x002C | EE_RDY | EEPROM ready interrupt |
| 24 | 0x002E | ANA_COMP | Analog Comparator 0 Interrupt |
| 25 | 0x0030 | TWI | Two-wire serial interface interrupt |
| 26 | 0x0032 | ANA_COMP1 | Analog Comparator 1 Interrupt |
| 27 | 0x0034 | - | Reserve |
| 28 | 0x0036 | PCI3 | Pin level interrupt 3 |
| 29 | 0x0038 | PCI4 | Pin level interrupt 4 |
| 30 | 0x003A | TC3_INT | Timer 3 interrupt |

The reset vector of the LGT8FX8P starts at address 0x0000. Except for the reset vector, all other vector addresses can be redirected to the 512-byte aligned starting address through the IVSEL in the MCUCR register and the IVBASE register.

## Interrupt vector handling

The following code only takes LGT8F88P as an example to illustrate reset and interrupt vector programming, for reference only.

| Assembly Code Example – LGT8F88P | | |
|---|---|---|
| Address | Code | Instruction |
| 0x000 | RJMP RESET | RESET vector |
| 0x001 | RJMP EXT_INT0 | External interruption 0 |
| 0x002 | RJMP EXT_INT1 | External interruption 1 |
| 0x003 | RJMP PCINT0 | Interrupt on pin change 0 |
| 0x004 | RJMP PCINT1 | Interrupt on pin change 1 |
| 0x005 | RJMP PCINT2 | Interrupt on pin change 2 |
| 0x006 | RJMP WDT | Watchdog Timer Interrupt |
| 0x007 | RJMP TIM2_COMPA | Timer 2 Compare Match Group A Interrupt |
| 0x008 | RJMP TIM2_COMPB | Timer 2 Compare Match Group B Interrupt |
| 0x009 | RJMP TIM2_OVF | Timer 2 overflow interrupt |
| 0x00A | RJMP TIM1_CAPT | Timer 1 traps interrupt |
| 0x00B | RJMP TIM1_COMPA | Timer 1 Compare Match Group A Interrupt |
| 0x00C | RJMP TIM1_COMPB | Timer 1 Compare Match Group B Interrupt |
| 0x00D | RJMP TIM1_OVFR | Timer 1 overflow interrupt |
| 0x00E | RJMP TIM0_COMPA | Timer 0 Compare Match Group A Interrupt |
| 0x00F | RJMP TIM0_COMPB | Timer 0 Compare Match Group B Interrupt |
| 0x010 | RJMP TIM0_OVF | Timer 0 overflow interrupt |
| 0x011 | RJMP SPI_STC | SPI transfer complete interrupt |
| 0x012 | RJMP USART_RXC | USART receive complete interrupt |
| 0x013 | RJMP USART_UDRE | USART data register empty interrupt |
| 0x014 | RJMP USART_TXC | USART send complete interrupt |
| 0x015 | RJMP ADC | ADC conversion complete interrupt |
| 0x016 | RJMP EE_RDY | EEPROM controller ready interrupt |
| 0x017 | RJMP ANA_COMP | Comparator interrupt |
| 0x018 | RJMP TWI | TWI controller interrupt |
| 0x019 | NOP | Reserved address |
| 0x01A | NOP | Reserved address |
| 0x01B | RJMP PCI3 | Pin Change Interrupt 3 |
| ; | | |
| 0x01C (reset :) | LDI r16, high(RAMEND) | Main program starts |
| 0x01D | OUT SPH, r16 | Set the stack pointer to the top address of RAM |
| 0x01E | LDI r16, low(RAMEND) | |
| 0x01F | OUT SPL, r16 | |
| 0x020 | SEI | Enable global interrupt |
| 0x021 | ..... | |

## Register definition

### *MCU Control Register - MCUCR*

| MCUCR – MCU Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| MCUCR: 0x35(0x55) | | | | Default value: 0x00 | | | | |
| MCUCR | FWKEN | FPDEN | EXRFD | PUD | IRLD | IFAIL | IVSEL | WCE |
| R/W | R/W | R/W | R/W | R/W | W/O | R/O | R/W | R/W |

| Bit definition | | |
|---|---|---|
| [0] | WCE | MCUCR update enable flag. Before updating MCUCR you need to set this flag first, and then complete the update of the MCUCR register within 6 cycles |
| [1] | IVSEL | Interrupt vector selection flag. After this flag is set (‘1’), the interrupt vector address will be mapped to a new address according to the value of the IVBASE register |
| [2] | IFAIL | System configuration load failure flag,<br>0 = Configuration information verification passed<br>1 = Failed to load configuration information |
| [3] | IRLD | Set this flag (‘1’) to reload system configuration data |
| [4] | PUD | Global pull-up disable flag<br>0 = Enable global pull-up control<br>1 = turn off all IO pull-up resistors |
| [5] | EXRFD | External reset filter disable flag<br>0 = Enable external reset (190us) digital filter<br>1 = Digital filter circuit for external reset disabled |
| [6] | FPDEN | Flash Power/down enable control<br>0: FLASH remains powered on after system SLEEP<br>1: FLASH power off after system SLEEP |
| [7] | FWKEN | Fast wake-up mode enable control, only valid for Power/Off mode<br>0:260us filter delay<br>1:32us filter delay |

### *Interrupt Vector Base Address Register - IVBASE*

| IVBASE – Interrupt Vector Base Address Register | | |
|---|---|---|
| IVBASE: 0x75 | | Default value: 0x00 |
| IVBASE | IVBASE[7:0] | |
| R/W | R/W | |

| Bit definition | | |
|---|---|---|
| [7:0] | IVBASE | If IVSEL is 1, interrupt vectors (except reset vectors) will be remapped on 512-byte pages with IVBASE as the base address.<br>The mapped interrupt vector base address is: (IVBASE << 8) + the corresponding vector address in Table 1 |

# External interruption

- 2 external interrupt sources
- Configurable level- or edge-triggered interrupts
- Can be used as a wake-up source in sleep mode

## Overview

External interrupts are triggered by the INT0 and INT1 pins. Interrupts can be triggered even if these 2 pins are configured as outputs as long as external interrupts are enabled, and this can be used to generate software interrupts. External interrupt can be triggered by rising edge, falling edge or low level, configured by the external interrupt control register EICRA. When external interrupts are enabled and configured as level-sensitive (only the INT0 and INT1 pins), the interrupt will continue to be generated as long as the pin is low. The rising or falling edge of the INT0 and INT1 pins requires the IO clock to work normally, while the low-level interrupts of the INT0 and INT1 pins are detected asynchronously. In addition to idle mode, the IO clocks are stopped in other sleep modes. Therefore, both external interrupts can be used as wake-up sources in sleep modes other than idle mode.

If the level-triggered interrupt is used as the wake-up source in power-saving mode, the changed level must be maintained for a certain period of time to wake up the MCU, so as to reduce the sensitivity of the MCU to noise. The required level must be held long enough for the MCU to end the wake-up process and then trigger a level interrupt.

## Register definition

Register list

| Register | Address | Defaults | Description |
|---|---|---|---|
| EICRA | 0x69 | 0x00 | External Interrupt Control Register A |
| EIMSK | 0x3D | 0x00 | External Interrupt Mask Register |
| EIFR | 0x3C | 0x00 | External Interrupt Flag Register |

### *External Interrupt Control Register A- EICRA*

| EICRA – External Interrupt Control Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x69 | | | | Default value: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | - | ISC11 | ISC10 | ISC01 | ISC00 |
| R/W | - | - | - | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:4 | - | Reserve. |
| 3 | ISC11 | INT1, interrupt trigger mode control flag MSB. |
| 2 | ISC10 | INT1, interrupt trigger mode control flag LSB.<br>External Interrupt 1 is triggered by the INT1 pin when the global interrupt is set and the corresponding interrupt mask control flag in the GICR register is set. The interrupt triggering method is described in the table. The MCU first samples the level on the INT1 pin before edge detection. If the edge trigger mode or the level change trigger mode is selected, the pulse whose duration is longer than 1 system clock cycle will trigger the interrupt. A pulse that is shorter cannot guarantee trigging the interrupt. If the low level trigger mode is selected, the low level must be maintained until the current instruction execution is completed before the interrupt is triggered. |
| 1 | ISC01 | INT0, interrupt trigger mode control flag MSB. |
| 0 | ISC00 | INT0, interrupt trigger mode control flag LSB.<br>External Interrupt 0 is triggered by the INT0 pin when the global interrupt is set and the corresponding interrupt mask control flag in the GICR register is set. The interrupt triggering method is described in the table. The MCU first samples the level on the INT0 pin before edge detection. If the edge trigger mode or the level change trigger mode is selected, the pulse whose duration is longer than 1 system clock cycle will trigger the interrupt. A pulse that is too short cannot guarantee trigging the interrupt. If the low level trigger mode is selected, the low level must be maintained until the execution of the current instruction is completed before the interrupt is triggered. |

The trigger mode of external interrupt 1 is shown in the following table.

External interrupt 1 trigger mode control

| ISC1[1:0] | Description |
|---|---|
| 0 | External pin INT1 low level trigger |
| 1 | External pin INT1 rising or falling edge trigger |
| 2 | External pin INT1 falling edge trigger |
| 3 | External pin INT1 rising edge trigger |

The trigger mode of external interrupt 0 is shown in the table below.

External interrupt 0 trigger mode control

| ISC0[1:0] | Description |
|---|---|
| 0 | External pin INT0 low level trigger |
| 1 | External pin INT0 rising edge or falling edge trigger |
| 2 | External pin INT0 falling edge trigger |
| 3 | External pin INT0 rising edge trigger |

### *External Interrupt Mask Register - EIMSK*

| EIMSK – External Interrupt Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x3D | | | | Default value: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | - | - | - | INT1 | INT0 |
| R/W | - | - | - | - | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:2 | - | Reserve |
| 1 | INT1 | External pin INT1 interrupt enable flag.<br>When the INT1 flag is set (‘1’) and the global interrupt is set, the external pin INT1 interrupt and the wake-up function are enabled. Even if the INT1 pin is configured as an output, an interrupt will be generated whenever the pin level changes accordingly. When the INT1 flag is cleared (‘0’), the external pin 1 interrupt and the wake-up function are disabled. |
| 0 | INT0 | External pin INT0 interrupt enable flag.<br>When the INT0 flag is set (‘1’), and the global interrupt is set, the external pin INT0 interrupt and the wake-up function are enabled. Even if the INT0 pin is configured as an output, an interrupt will be generated whenever the pin level changes accordingly. When the INT0 flag is cleared (‘0’), the external pin 0 interrupt and the wake-up function are disabled. |

### *External Interrupt Flag Register - EIFR*

| EIFR – External Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x3C | | | | Default value: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | - | - | - | INTF1 | INTF0 |
| R/W | - | - | - | - | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:2 | - | Reserve |
| 1 | INTF1 | External pin 1 interrupt flag.<br>INTF1 is set when an edge-triggered external pin 1 interrupt occurs.<br>The INTF1 flag is not set when the external pin 1 interrupt is triggered low.<br>If the external pin 1 interrupt enable INT1EN flag is "1" and the global interrupt flag is set at this time, an external pin 1 interrupt will be generated.<br>INTF1 will be automatically cleared when this interrupt service routine is executed, or this flag can be cleared by writing "1" to the INTF1 flag. |
| 0 | INTF0 | External pin 0 interrupt flag.<br>INTF0 is set when an edge triggered external pin 0 interrupt occurs.<br>The INTF0 flag is not set when the external pin 0 interrupt is triggered low.<br>If the external pin 0 interrupt enable INT0EN flag is "1" and the global interrupt flag is set at this time, an external pin 0 interrupt will be generated.<br>INTF0 will be cleared automatically when this interrupt service routine is executed, or this flag can be cleared by writing "1" to the INTF0 flag. |

**[FIXME: This is confusing. INTF[0,1] defaults to ‘0’, implying that to ‘set’ it means writing ‘1’ into the flag. BUT the Chinese original text does say that writing ‘1’ to the flag constitutes clearing it. ??? This must be proof-read properly and seen in context with all the other passages explaining the functionality of this register.**]

# Computational Accelerator (uDSC)

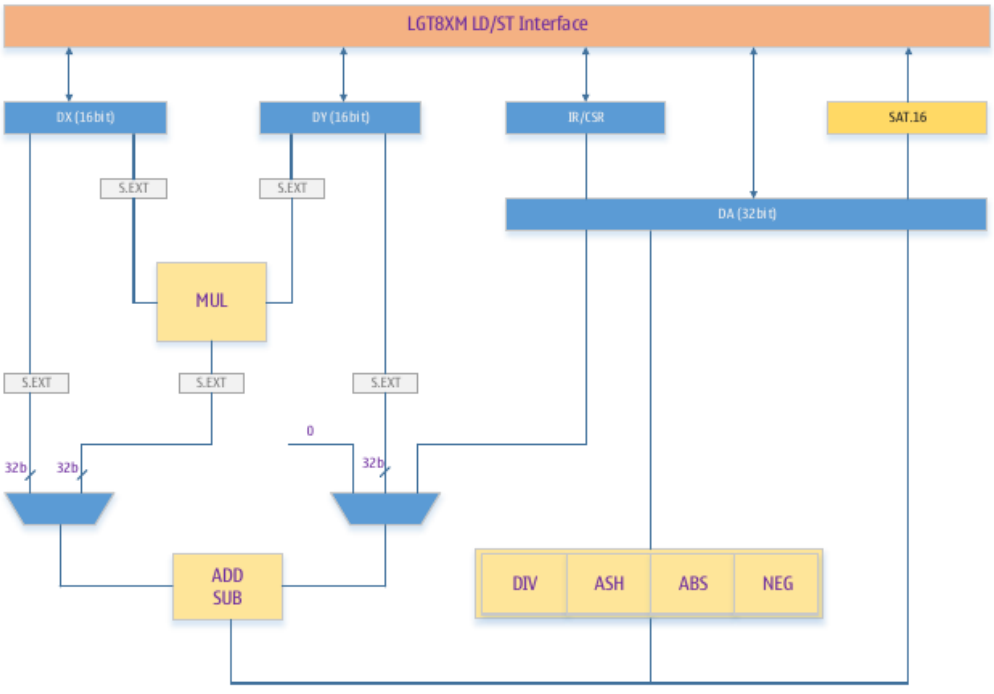
- 16-bit storage mode (LD/ST)
- 32-bit accumulator (DX)
- Single-cycle 16-bit multiplier (MUL)
- 32-bit Arithmetic Logic Unit (ALU)
- 16-bit Saturation (SD)
- 8-cycle 32/16 divider
- Single-cycle multiply-add/multiply-subtract operations (MAC/MSC)

## Overview

The Digital Operation Accelerator (uDSC), as an operation co-processing module of the LGT8XM core, cooperates with the 16-bit LD/ST mode of the LGT8XM core to realize a 16-bit digital signal processing unit. It can meet the processing of most control digital signals.

uDSC internals and functionality:

1. 16-bit operand register DX/DY
2. 32-bit accumulation register DA
3. Single-cycle 17-bit multiplier (can implement 16-bit signed/unsigned multiplication)
4. 32-bit ALU (can implement 16/32-bit addition, subtraction and shift operations)
5. 16-bit saturation operation (for storing the operation result to RAM space)
6. 32/16 divider, complete operation in 8 cycles



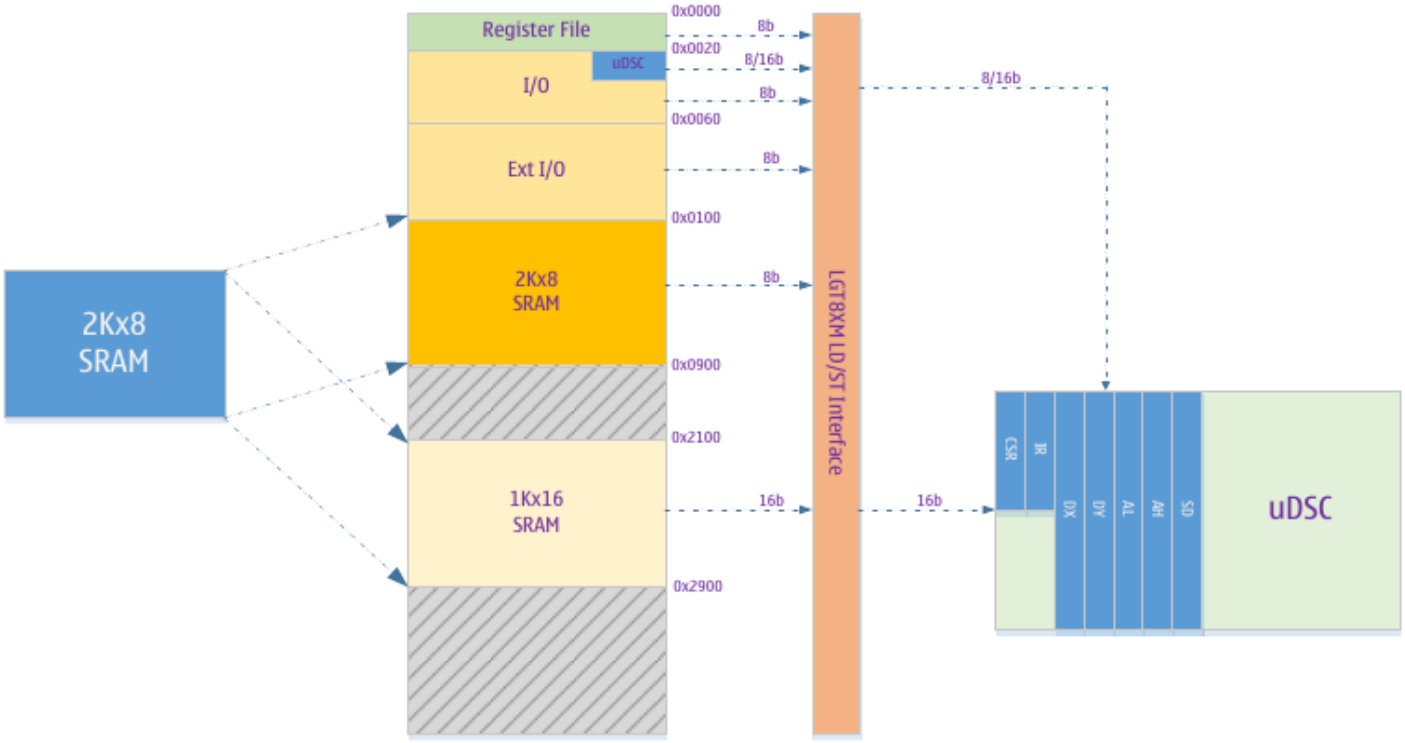

uDSC structure diagram

## 16-bit LD/ST working mode

In order to improve the efficiency of uDSC processing a large number of data operations, LGT8XM core implements a dedicated 16-bit LD/ST memory channel, which can use LDD/STD instructions to efficiently perform 16-bit data exchange between uDSC and SRAM and general-purpose register files.

In order not to destroy the normal LD/ST instruction system, the LGT8XM core remaps the SRAM space to 0x2100~0x28FF. When using the LD/ST instruction to access the SRAM from the space of 0x2100~0x28FF, the core automatically enables the 16-bit LD/ST function and opens the direct access channel between the SRAM and the uDSC.

The following figure shows the data space address distribution of the LGT8XM core:




As shown in the figure above, the LGT8XM core can directly perform 16-bit data memory access between the DX/DY/DA registers of the uDSC and the SRAM by using the LD/ST instructions. At the same time, the internal registers of the uDSC are also mapped to the I/O space, and the access to the uDSC registers is divided into 8/16 modes.

In addition to the DX/DY/DA registers used for operation, the uDSC also contains two other 8-bit registers:
uDSC controls the status register CSR and the operation instruction register IR. CSR/IR can only be accessed in bytes through I/O space; 16-bit mode when accessing DX/DY/AL/AH. It can be accessed using IN/OUT and LD/ST/LDD/STD/LDS/STD instructions.

The related control status and data registers of uDSC are mapped to the IO space, which can be directly addressed by the IN/OU instruction, which can complete 8/16-bit data access in one instruction cycle.

The CSR is used to control the working mode of the uDSC and record the status flags of the current operation performed by the uDSC. The IR controls the specific operations implemented by the

uDSC. Most of the operations supported by uDSC will be completed in one cycle, and the division operation requires 7 waiting cycles. You can also judge whether the current division operation is completed through the flag in the CSR register.

The standard LD/ST instruction uses the general working register inside the LGT8XM as the data of LD/ST, and uses X/Y/Z as the target address. When the target address falls in the 16-bit SRAM mapping space, the meaning of the LD/ST instruction operand changes, in which X/Y/Z are still used as the target address, and the meaning of the general-purpose working register addressing will be based on the uDSC mapping mode. This gives two processing methods. The mapping mode of uDSC is only used for access to addresses 0x2100~0x28FF. The mapping mode is set by bit 6 (MM) of the CSR register.

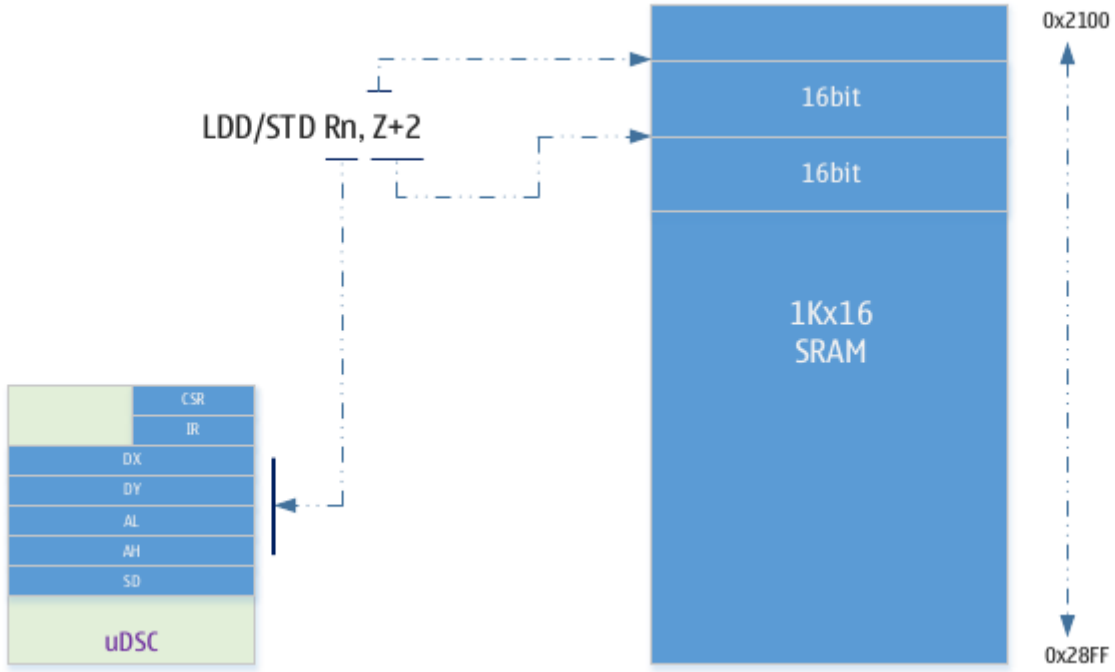
In 16-bit LD/ST mode, the instruction "LDD Rn, Z+q" means to load the 16-bit data of address [Z] into the data register of uDSC, and then increase the value of Z by an offset "q" . The relationship between the meaning of Rn and the mapping mode CSR[MM] is as follows:

| LDD Rn, Z/Y+q | | | |
|---|---|---|---|
| CSR[MM] | [Z+q] | Opcode | Operations |
| 0 | 0x2100~0x28FF | LDD R0, Z+q | DX = [Z]; Z = Z + q; R0 kept unchanged |
| | | LDD R1, Z+q | DY = [Z]; Z= Z + q; R1 kept unchanged |
| | | LDD R2, Z+q | AL = [Z]; Z= Z + q; R2 kept unchanged |
| | | LDD R3, Z+q | AH = [Z]; Z= Z + q; R3 kept unchanged |
| 1 | 0x2100~0x28FF | LDD Rn, Z+q | {Rn} address for DX/DY/AL/AH in I/O region<br>[DX/DY/AL/AY] = [Z]; Z = Z + q<br>Rn keep unchanged |
| | | | |
| STD Rn, Z/Y+q | | | |
| 0 | 0x2100~0x28FF | STD Z+q, R0 | [Z] = DX; Z = Z + q; R0 kept unchanged |
| | | STD Z+q, R1 | [Z] = DY; Z = Z + q; R1 kept unchanged |
| | | STD Z+q, R2 | [Z] = AL; Z = Z + q; R2 kept unchanged |
| | | STD Z+q, R3 | [Z] = AH; Z = Z + q; R3 kept unchanged |
| | | STD Z+q, R4 | [Z] = SD; Z = Z + q; R4 kept unchanged |
| 1 | 0x2100~0x28FF | STD Z+q, Rn | {Rn} address for DX/DY/AL/AH/SD in I/O region<br>[Z] = [DX/DY/AL/AH/SD] addressed by {Rn}<br>Rn keep unchanged |

LD/ST, LDS/STS in the LGT8XM instruction set can access the 0x2100~0x28FF area, but the Y/Z+q addressing mode of LDD/STD is more efficient. The addressing in LDD/STD mode is based on a base address, we can set Y/Z as the base address of the data in the RAM, by using the Y/Z+q addressing mode of the LDD/STD instruction, the instruction can be executed in one cycle and access data, and automatically move the address pointer to the next target address.

The Y/Z+q offset addressing mode of the standard LDD/STD instruction of the LGT8XM core uses [Y/Z+q] as the address of 8-bit data when the instruction is executed, and the value of Y/Z does not

increase after the execution is completed. When LDD/STD is used to address the address in the range of 0x2100~0x28FF, the instruction behavior of LDD/STD changes: when the instruction is executed, use [Y/Z] as the 16-bit data addressing address, after execution, the value of Y/Z Increments the offset specified by "q". This feature can improve the efficiency of our continuous addressing. By setting "q=2", the addressing of continuous 16-bit data can be achieved.




*Mapping between variable addresses and 16-bit mode addresses*

The LGT8XM is an 8-bit processor, and data access is in bytes. The LGT8F328P has a built-in 2K byte data space. This part of the space is mapped to the address of 0x0100~0x08FF. C/C++ compilation automatically assigns variables between 0x0100~0x08FF. If we define a 16-bit array in C/C++ and need to use uDSC for operation, we need to first map the address of the variable to the address area (0x2100~0x28FF) accessed by 16-bit LD/ST. The method is very simple, only The address of the variable needs to be increased by an offset of 0x2000.

## uDSC operation instruction definition

The software specifies the operation to be implemented through the IR register of the uDSC. All operations of uDSC are performed between DX/DY/DA. Users can use the 16-bit LD/ST channel to exchange data directly and quickly in DX/DY/DA and SRAM.

| Classification | IR[7:0] | | | | | | | | Function |
|---|---|---|---|---|---|---|---|---|---|
| ADD/SUB | 0 | 0 | $S^1$ | 0 | 0 | 1 | 0 | 1 | DA = DX + DY |
| | 0 | 0 | $S^1$ | 0 | 0 | 0 | 0 | 1 | DA = DX – DY |
| | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | DA = DY |
| | 0 | 0 | $S^1$ | 1 | 1 | 0 | 0 | 1 | DA = -DY |
| | 0 | 0 | $S^1$ | 1 | 0 | 1 | 1 | 1 | DA = DA + DY |
| | 0 | 0 | $S^1$ | 1 | 0 | 0 | 1 | 1 | DA = DA – DY |
| MAC/MSC | 0 | 1 | $S1^2$ | $S0^2$ | 0 | 1 | 0 | 0 | DA = DX * DY |
| | 0 | 1 | $S1^2$ | $S0^2$ | 0 | 0 | 0 | 0 | DA = -DX * DY |
| | 0 | 1 | $S1^2$ | $S0^2$ | 1 | 1 | 0 | 0 | DA = (DX * DY) >> 1 |
| | 0 | 1 | $S1^2$ | $S0^2$ | 1 | 0 | 0 | 0 | DA = (-DX * DY) >> 1 |
| | 0 | 1 | $S1^2$ | $S0^2$ | 0 | 1 | 1 | S | DA = DA + DX * DY |
| | 0 | 1 | $S1^2$ | $S0^2$ | 1 | 1 | 1 | S | DA = (DA + DX * DY) >> 1 |
| | 0 | 1 | $S1^2$ | $S0^2$ | 0 | 0 | 1 | S | DA = DA – DX * DY |
| | 0 | 1 | $S1^2$ | $S0^2$ | 1 | 0 | 1 | S | DA = (DA – DX * DY) >> 1 |
| MISC | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DA = 0 |
| | 1 | 0 | 0 | 0 | 0 | 1 | 0 | S | DA = NEG(DA) |
| | 1 | 0 | 0 | 0 | 1 | 0 | 0 | S | DA = DX^2 |
| | 1 | 0 | 0 | 0 | 1 | 0 | 1 | S | DA = DY^2 |
| | 1 | 0 | 1 | 0 | 0 | 0 | 0 | S | DA = ABS(DA) |
| | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | DA = DA/DY |
| | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | DA = DA/DY, DY = DA%DY |
| SHIFT | 1 | 1 | 0 | 0 | N3 | N2 | N1 | N0 | DA = DA << N |
| | 1 | 1 | S | 1 | N3 | N2 | N1 | N0 | DA = DA >> N |

Description:

1. S indicates whether the operation is a signed or unsigned operation of course
2. S1 indicates whether DX is a signed number, S2 indicates whether DY is a signed number
3. N3–N0 defines what bit-shift operation to implement, out of 15 alternatives
4. – Indicates that the value of this bit is not without significance. Although it can be set to 0 or 1, it is recommended to set it to 0

## Register definition

| Name | IO address | Function |
|---|---|---|
| DCSR | 0x20(0x00) | uDSC Control Status Register |
| DSIR | 0x21(0x01) | Operation instruction register |
| DSSD | 0x22(0x02) | 16-bit saturation result of accumulator DSA |
| DSDX | 0x10(0x30) | Operand DSDX, 16-bit read and write access |
| DSDY | 0x11(0x31) | Operand DSDY, 16-bit read and write access |
| DSAL | 0x38(0x58) | 32-bit accumulator DSA[15:0], 16-bit read and write access |
| DSAH | 0x39(0x59) | 32-bit accumulator DSA[31:16], 16-bit read and write access |

### *DSCR - Control Status Register*

| DSCR – uDSC Control Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x20 (0x00) | | | | Default: 0010_xxxx | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSUEN | MM | D1 | D0 | - | N | Z | C |
| R/W | R/W | R/W | R/W | R/W | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | DSUEN | uDSC module enable control; 1 = enable, 0 = disable |
| 6 | MM | uDSC register mapping mode; please refer to the introduction of 16-bit working mode for detailed definition.<br>0 = fast access mode, 1 = IO mapped mode |
| 5 | D1 | Divide operation complete flag, 1 = operation complete |
| 4 | D0 | Divide by 0 flag |
| 3 | - | Unimplemented |
| 2 | N | The result of the operation is a negative number flag |
| 1 | Z | The result of the operation is zero flag |
| 0 | C | 32 Adder carry/borrow flag |

### *DSIR - Operation Instruction Register*

| DSIR – uDSC arithmetic instruction register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x21 (0x01) | | | | Default: 0000_0000 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSIR[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 7:0 | IR | uDSC operation command. Please refer to the description in the "Operation Instruction Definition" chapter |

### *DSDX - Operand Register DSDX*

| DSDX – uDSC Operand Register DX | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address: 0x30 (0x11) | | | | | | | | Default: 0000_0000 | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSDX[15:0] | | | | | | | | | | | | | | | |
| R/W | R/W | | | | | | | | | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 15:0 | DSDX | 16-bit operand register DSDX |

### *DSDY - Operand Register DSDY*

| DSDY – uDSC Operand Register DY | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address: 0x31 (0x11) | | | | | | | | Default: 0000_0000 | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSDY[15:0] | | | | | | | | | | | | | | | |
| R/W | R/W | | | | | | | | | | | | | | | |
| Bit | Name | | | | | | | Description | | | | | | | | |
| 15:0 | DSDY | | | | | | | 16-bit operand register DSDY | | | | | | | | |

### *DSAL - Lower 16 bits of 32-bit accumulator DA*

| DSAL – Lower 16 bits of uDSC Operand Register DSA | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address: 0x58 (0x38) | | | | | | | | Default: 0000_0000 | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSA[15:0] | | | | | | | | | | | | | | | |
| R/W | R/W | | | | | | | | | | | | | | | |
| Bit | Name | | | | | | | Description | | | | | | | | |
| 15:0 | DSAL | | | | | | | Lower 16 bits of 32-bit accumulator DSA | | | | | | | | |

### *DSAH - upper 16 bits of 32-bit accumulator DA*

| DSAH – upper 16 bits of uDSC operand register DSA | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address: 0x59 (0x39) | | | | | | | | Default: 0000_0000 | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSA[31:16] | | | | | | | | | | | | | | | |
| R/W | R/W | | | | | | | | | | | | | | | |
| Bit | Name | | | | | | | Description | | | | | | | | |
| 15:0 | DSAH | | | | | | | The upper 16 bits of the 32-bit accumulator DSA | | | | | | | | |

### *DSSD - DA Saturation Operation Register*

| DSSD – 16-bit DA saturation operation result | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address: 0x22 (0x02) | | | | | | | | Default: 0000_0000 | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DSSD[15:0] | | | | | | | | | | | | | | | |
| R/W | R/W | | | | | | | | | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 15:0 | DSSD | 16-bit saturation result of 32-bit accumulator DSA |

## uDSC application example

Example 1. Basic configuration and operation

The following is a simple subroutine (AVRGCC) that implements a 16-bit multiplication and returns a 32-bit result.:

unsigned long dsu_xmuluu (unsigned short dy, unsigned short dx);

The following is the assembly implementation code of this C function:

```
#include  “udsc_def.inc”           ; opcode definitions
          .global   dsu_xmuluu     ; declare for called from C/C++ code

dsu_xmuluu:
          out       DSDX, r24      ; load DX
          out       DSDY, r22      ; load DY
          ldi       r20, XMULUU    ; load opcode
          out       DSIR, r20      ; do multiply
          in        r22, DSAL      ; {r23, r22} = AL
          in        r24, DSAH      ; {r25, r24} = AH
          ret
```

# General-Purpose Programmable Port (GPIO)

## Overview

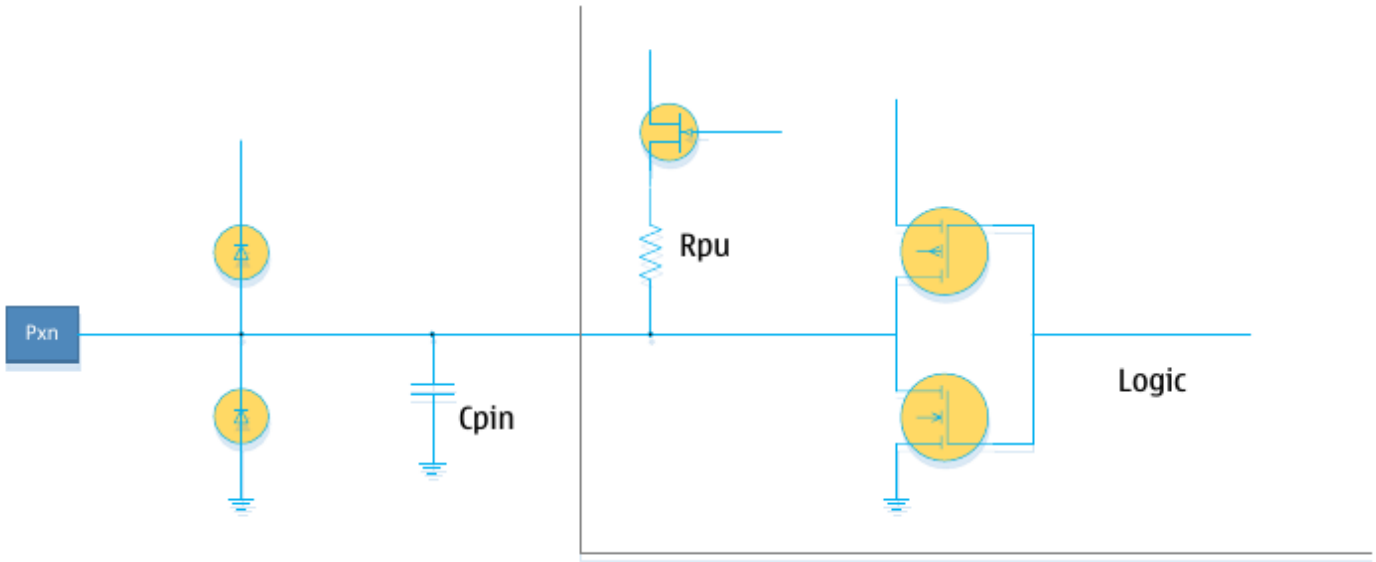
All MCUs implemented based on the LGT8XM core family have I/O port read-modify-write capabilities. This means that the state of a port can be changed individually using SBI and CBI instructions without affecting any other I/O. Likewise, changing the direction of a port or controlling its pull-up resistors can do the same.

Most of the I/Os of the LGT8FX8P have symmetrical drive characteristics and can drive and sink large currents. The I/O has two levels of drive capability, and the user can control the drive capability of each group of I/Os. The drive capability of the I/O can directly drive some LEDs.

Most of the I/Os of the LGT8FX8P can drive up to 30mA and can be directly used to drive segment code LEDs.

All I/Os have independent ESD protection diodes directly on VCC and GND, and are designed to withstand ESD pulses up to 5000V at least.

I/O Equivalent Circuit Diagram:




All the registers below in this chapter are described in a unified way. The lowercase "x" represents the alphabetic serial number name of the port, and the lowercase "n" represents the bit number in the port. But when using port registers in a program, the exact register names must be used. For example, PORTB3, which represents the third digit of PORTB, is represented by PORTxn here. For the detailed definition of I/O related registers, please refer to the register description section.

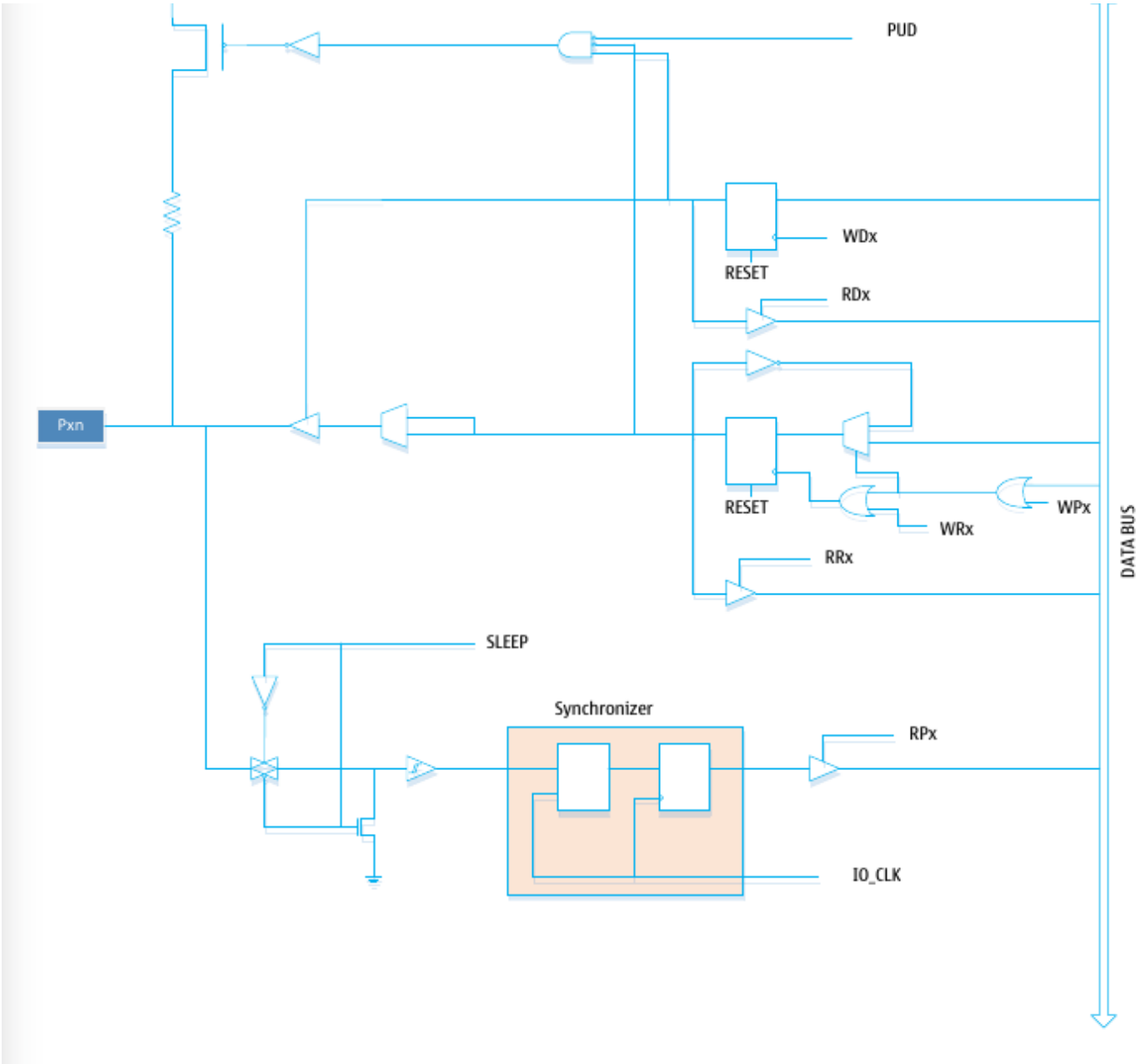
Each port has allocated three I/O register spaces, which are: port data output register (PORTx), port direction register (DDRx), port data input register (PINx). The port data input registers are read-only registers. The data output register and port direction register can be read and written. The PUD flag in the MCUCR register is used to control the pull-up resistors of all I/Os. When the PUD flag is 1, the pull-up resistors of all I/Os will be disabled.

In addition to general-purpose input/output functions, most of the I/Os are also reused for other peripheral functions. For specific multiplexing functions, please refer to the chapter on port function multiplexing.

It should be noted that enabling the multiplexing function of some ports will not affect the use of these ports as digital I/O. And some alternate functions may also need to control the input/output direction of the port through the I/O register. The specific settings will be introduced in the documentation of each multiplexing module.

## General purpose input/output ports

When used as a general-purpose I/O, the port is a bidirectional drive I/O port with an internal programmable pull-up. The following figure shows the equivalent circuit diagram of a general-purpose I/O port:




PUD: PULLUP DISABLE

SLEEP: SLEEP CONTROL

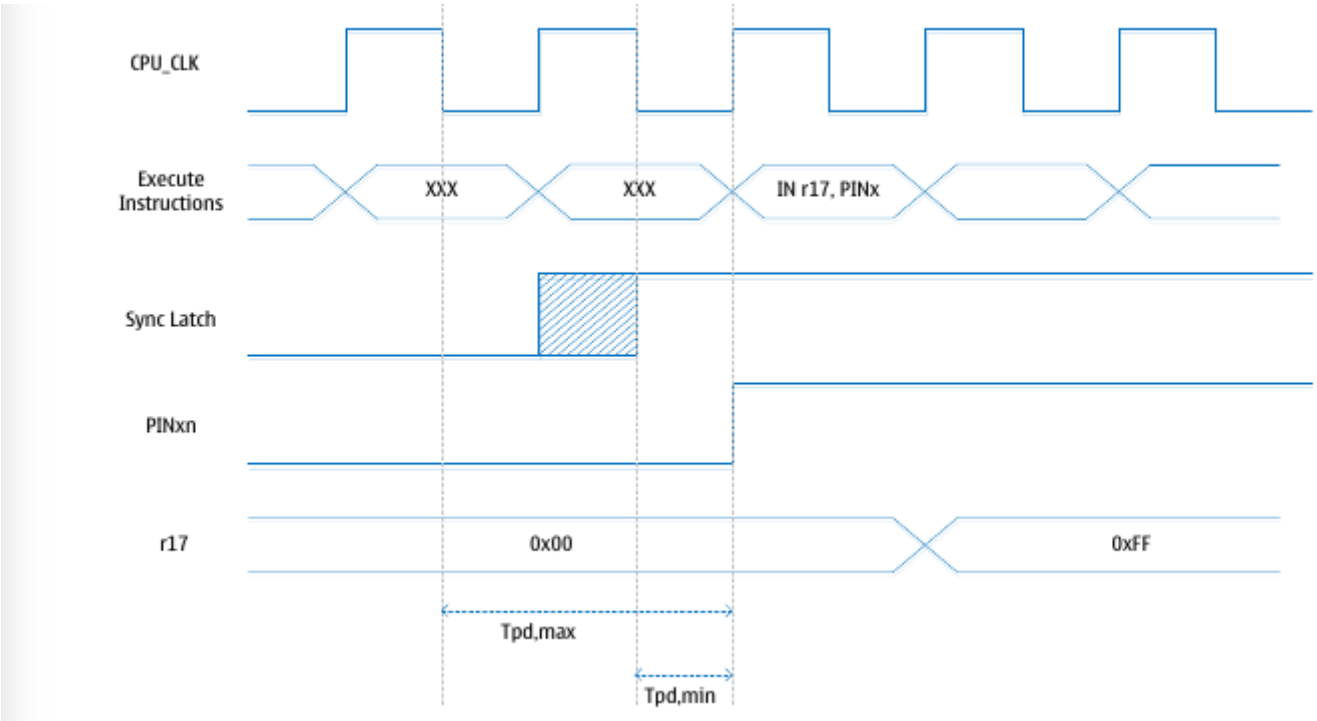
IO_CLK: I/O CLOCK

| | |
|---|---|
| WDx: | WRITE DDRx |
| RDx: | READ DDRx |
| WRx: | WRITE PORTx |
| RRx: | READ PORTx REGISTER |
| RPx: | READ PORTx PIN |
| WPx: | WRITE PINx REGISTER |

### *Port usage configuration*

Each port is controlled by three register flags: DDxn, PORTxn and PINxn. Where DDxn can be accessed through the DDRx register, PORTxn can be accessed through the PORTx register, and PINxn can be accessed through the PINx register.

The DDRxn register flags are used to set the input/output direction of the port. If DDxn is set to 1, the Pxn port is configured as an output port. If DDxn is set to 0, Pxn is configured as an input port.




If the PORTxn flag is set to 1 and the port is configured as an input port, the pull-up resistor for this port is active. If you want to disable the port's pull-up resistor, PORTxn must be set to 0 or the port must be configured as an output port.

The reset initialization mode of the port is input, and the associated pull-up resistor is disabled.

Say PORTxn is set to 1. If this port is configured as an output port, the external port will be driven high. If PORTxn is set to 0, the port will be driven low.

### *Input/output switching*

When the I/O state toggles between tri-state ([DDxn, PORTxn]) = 0b00) and output high ([DDxn, PORTxn] = 0b11), there will be an intermediate port pull-up or output low state. Usually, pull-up resistors are acceptable because in a high-impedance environment, the difference between driving high and pulling up is usually not important. If this is not the case, the pull-up function of all ports can be disabled by the PUD flag in the MCUCR register.

Likewise, the same problem occurs when switching between a pull-up-enabled input and an output low. User must use tri-state ([DDxn, PORTxn] = 0b00) or output high ([DDxn, PORTxn] = 0b11) as intermediate state.

Port driver configuration table:

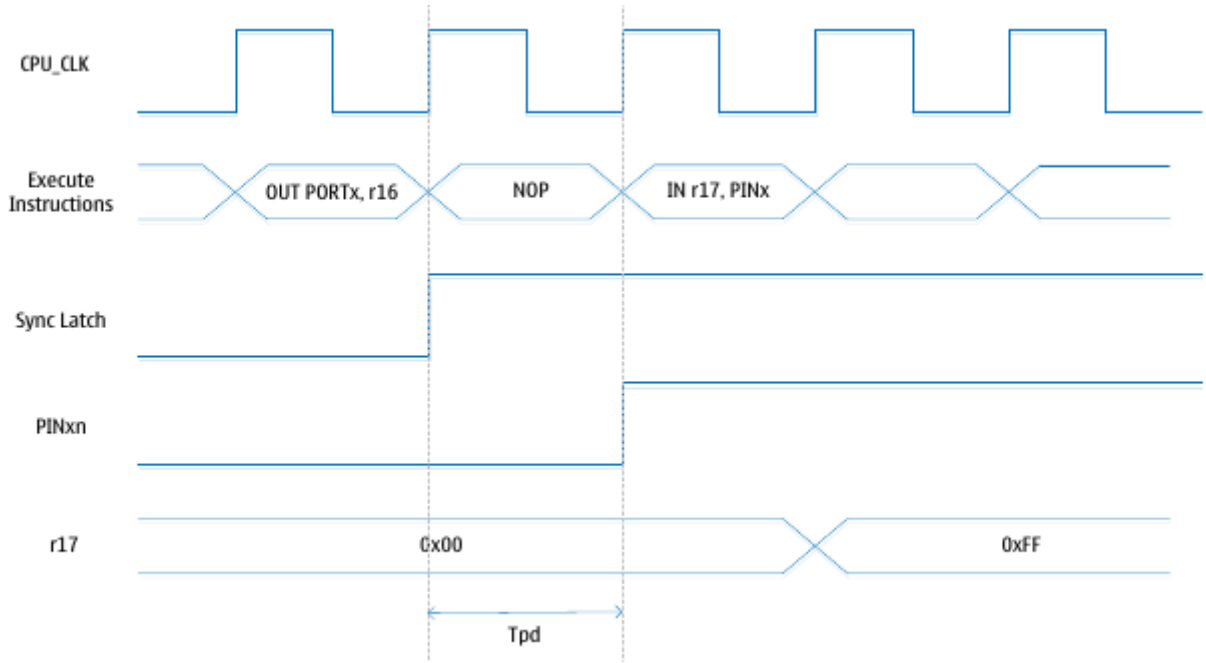
| DDxn | PORTxn | PUD | Port status | Pull up | Function |
|---|---|---|---|---|---|
| 0 | 0 | X | Input | Disable | Tri-state (High-Z) |
| 0 | 1 | 0 | Input | Enable | Input + internal pull-up mode |
| 0 | 1 | 1 | Input | Disable | Tri-state (High-Z) |
| 1 | 0 | X | Output | Disable | Output low (fan-in) |
| 1 | 1 | X | Output | Disable | Output high (fan out) |

### *Read port value*

Regardless of how the port direction flag DDxn is set, the current state of the port can be read through the PINxn register. To avoid the metastability caused by direct reading of the port, the PINxn register are going through a synchronizer. The synchronizer is composed of a latch and a register, so there is a small delay between the value of PINxn and the current port. This delay is a result of the presence of the synchronizer, and the delay time is at most 1 and a half system cycles.

We assume that the system cycle starts from the first falling edge of the system clock, the latches latch data when the clock is low, and the data goes straight through the latches when the clock is high, as shown in the shaded area in the figure above. Port data is latched while the clock is low and is registered to the PINxn register on the next rising edge of the clock. Tpd,max and Tpd,min in the above figure are the maximum and minimum delays of port data, which are divided into 1.5 cycles and 0.5 cycles.

If you want to read the port value set by software, you need to insert a no operation instruction (NOP) in the I/O write and read byte support. The timing is shown in the following figure:



The following code shows how to set pins 0/1 of port B as high and 2/3 as low, define pins 4~7 as inputs and enable the pull-up resistors of pins 6 and 7. The value of the pin is then read back into the general-purpose working register, and a NOP instruction is inserted directly at the output and input of the pin as described earlier.

Assembly code

```
; Define Pull-ups and set outputs high
; Define directions for port pins
LDI r16, (1<<PB7)|(1<<PB6)|(1<<PB1)|1<<PB0)
LDI r17, (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
OUT PORTB, r16
OUT DDRB, r17
; Insert nop for synchronization
NOP
; Read port pins
IN r16, PINB
```

C code

```
unsigned char I;
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization */
__no_operation();
/* Read port pins */
I = PINB;
```

### *Input Enable and Sleep Control*

From the equivalent circuit diagram of the I/O, we can see that the digital input can be clamped to ground level under the control of the SLEEP signal. The SLEEP signal is controlled by the MCU's sleep controller and various sleep modes. In this way, it can be ensured that the system will not cause power leakage due to the floating of the port input after entering the sleep mode.

The SLEEP control role of the port is replaced by the external interrupt function. If the external interrupt request is invalid, the SLEEP control can still function. The SLEEP control function will also be replaced by some other secondary functions. For details, please refer to the introduction of the secondary function of the port below.

### *Quickly flip the port state*

The port state is set to output IO, and the port state can be changed through the PORTn register. If the output state of the current port needs to be flipped, it is usually necessary to first read the current port state PINx, and then write it back to the PORTn register to complete the flip. However LGT8FX8P provides another more efficient way to flip the port state. By directly writing 1 to

the PINx register, the specified port state is flipped. For example, if we write PINB[3] as 1, the port state of PB3 flips. This method is very practical for applications that need to generate for example an output clock.

### *Digital/Analog Multiplex Port*

Some ports of LGT8FX8P are mixed multiplexing ports of digital and analog functions. Except for the output PD4 of the internal DAC, all other mixed ports are useable as analog inputs. When the port is used as an analog input, the software needs to set the port to input mode and close the internal pull-up as needed to avoid affecting the analog measurements. The DIDR0~2 registers are used to close the digital input channel of the mixed function port, so as to avoid unnecessary power drain caused by the analog input to the digital circuit. DIDRx does not turn off the digital output function of the port.

### *High current push-pull drive port*

LGT8FX8P supports up to 6 high-current push-pull drive ports, and supports push-pull drive up to 80mA. Considering the limitation of the maximum over-current capability of the chip VCC, it is not recommended to turn on 6 high-current drivers at the same time. Especially for the QFP32 package with only one set of power ports, it is recommended not to turn on and drive more than 4 high-current loads at the same time.

The drive of the common port is 12mA, and the software needs to enable the high-current drive function of the port through the HDR register. The ports with high current drive capability are as follows:

| HDR port | QFP48 | QFP32 | HDR | Function |
|---|---|---|---|---|
| PD5 | PD5 | PD5 | HDR[0] | N/A |
| PD6 | PD6 | PD6 | HDR[1] | N/A |
| PF1 | PF1 | PD1<br>PF1 | HDR[2] | The PD1 of the QFP32 package is internally equivalent to the PD1 of the QFP48 and the PF1 in parallel |
| PF2 | PF2 | PD2<br>PF2 | HDR[3] | The PD2 of the QFP32 package is equivalent to the PD2 of the QFP48 in parallel with the PF2 |
| PF4 | PF4 | PE4<br>PF4 | HDR[4] | The internal PE4 of the QFP32 package is equivalent to the parallel connection between the PF4 and PE4 of the QFP48 |
| PF5 | PF5 | PE5<br>PF5 | HDR[5] | The internal PE5 of the QFP32 package is equivalent to the parallel connection between the PF5 and PE5 of the QFP48 |

### *Handling of free ports*

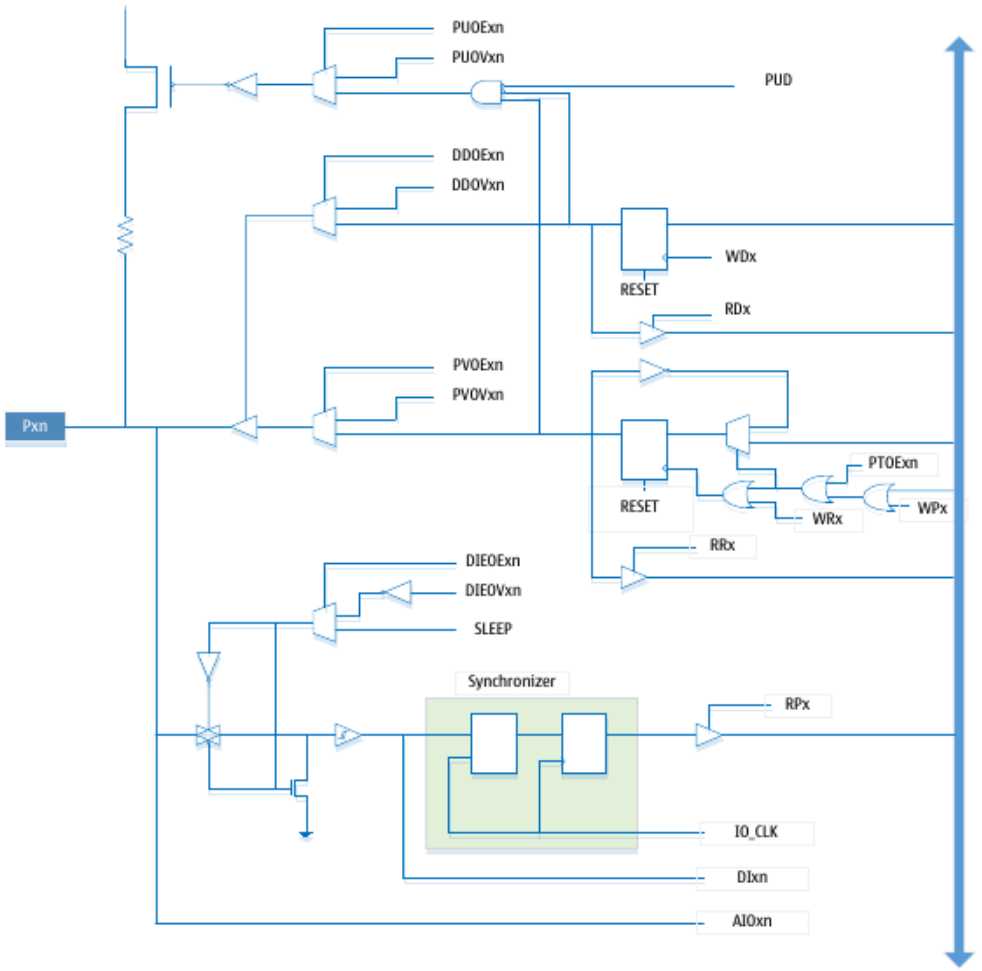
If some ports are not used, it is recommended to drive them to a fixed level. Floating pins will bring more power consumption, and will cause the system to become unstable under strong interference.

The easiest way to give a port a fixed level is to turn on the port's pull-up resistor. Note that the pull-up resistors are disabled during power-on reset. The internal pull-up resistors will inevitable also

bring increased load on the chip's internal power rail, and therefore it is recommended to use an external pull-up or pull-down resistor connection. Connecting the port directly to power or ground is not recommended, because if these pins are configured as outputs there is the potential to cause very large currents to flow through the port, which could be damaging to the chip.

## Port multiplexing function

Most ports have multiplexing function, the following equivalent circuit illustrates the port multiplexing function to control the port. These alternate functions do not necessarily exist on all port pins.




| | | | |
|---|---|---|---|
| PUOExn: | Pxn PULL-UP OVERRIDE ENABLE | PUD: | PULLUP DISABLE |
| PUOVxn: | Pxn PULL-UP OVERRIDE VALUE | WDx: | WRITE DDRx |
| DDOExn: | Pxn DATA DIRECTION OVERRRIDE ENABLE | RDx: | READ DDRx |
| DDOVxn: | Pxn DATA DIRECTION OVERRIDE VALUE | RRx: | READ PORTx REGISTER |
| PVOExn: | Pxn PORT VALUE OVERRIDE ENABLE | WRx: | WRITE PORTx |
| PVOVxn: | Pxn PORT VALUE OVERRIDE VALUE | RPx: | READ PORTx PIN |
| DIEOExn: | Pxn INPUT-ENABLE OVERRIDE ENABLE | WPx: | WRITE PINx |
| DIEOVxn: | Pxn INPUT-ENABLE OVERRIDE VALUE | IO_CLK: | I/O CLOCK |
| SLEEP: | SLEEP CONTROL | DIxn: | INPUT PIN n ON PORTx |
| PTOExn: | Pxn PORT TOGGLE OVERRIDE ENABLE | AIOxn: | ANALOG I/O PIN n ON PORTx |

General description of multiplexing function control signals:

| Signal | Designation | Functional description |
|---|---|---|
| PUOE | Pull-up multiplexing enable | When this flag is 1, the pull-up enable is controlled by PVOV; if this flag is 0, the pull-up enable is jointly controlled by DDxn, PORTxn and PUD |
| PUOV | Pull-up multiplexed value | If PUOE is 1, this flag will enable the pin's pull-up resistor. Otherwise, if PUOE is 0, it will disable the pin's pull-up resistor |
| DDOE | Port Direction Multiplexing Enable | If PUOV is 1, the pin output enable is controlled by DDOE, otherwise it is controlled by DDxn |
| DDOV | Port Direction Multiplexing Value | If DDOE is 1 and PUOV is 1, the output function of the pin will be enabled, otherwise the output of the pin will be disabled |
| PVOE | Port data multiplexing enable | If PUOV is 1 and the pin output is enabled, the output value of the pin will be controlled by PVOV, otherwise it will be controlled by PORTxn |
| PVOV | Port data multiplexing value | Refer to PVOE function description |
| PTOE | Port flip multiplexing enable | When PUOV is 1, the PORTxn flag will toggle |
| DIEOE | Digital input enable multiplexing enable | If PUOV is 1, the port digital input enable will be controlled by DIEOV; otherwise it will be controlled by the running state of the MCU |
| DIEOV | Digital input enable multiplexed value | If DIEOE is 1, the digital input function of the port will be controlled by PUOV, regardless of the MCU running state |
| DI | digital input | This is the digital input signal input to the substitute function module.<br>As you can see from the I/O circuit diagram below, this value is after the Schmitt trigger, but before the I/O input synchronizer. This signal is connected to the peripheral module, and the peripheral module will perform synchronization processing as needed |
| AIO | analog input | Analog input/output signal, this signal is directly connected to the PAD of the I/O and can be used as an analog bidirectional signal. This signal is directly connected to the port of the internal ADC, comparator and other analog modules |

The following subsections will briefly describe the alternate function of each pin and the associated control signals.

### *Port B alternate function*

| Pin | Alternate Function |
|---|---|
| PB7 | XTALI/TOSC2 (external main crystal pin XI)<br>PCINT7 (pin change interrupt 7) |
| PB6 | XTALO/TOSC1 (external main crystal oscillator pin XO)<br>PCINT6 (pin change interrupt 6) |
| PB5 | SCK (SPI bus master clock input)<br>PCINT5 (pin change interrupt 5) |
| PB4 | MISO (SPI bus master input/slave output)<br>PCINT4 (pin change interrupt 4) |
| PB3 | MOSI (SPI bus master output/slave input)<br>OC2A (Timer/Event Counter 2 compare match output A)<br>PCINT3 (pin change interrupt 3) |
| PB2 | SSN (SPI bus slave select input)<br>OC1B (Timer/Event Counter 1 compare match output B)<br>PCINT2 (pin change interrupt 2) |
| PB1 | OC1A (Timer/Event Counter 1 compare match output A)<br>PCINT1 (pin change interrupt 1) |
| PB0 | ICP1 (Timer/Counter 1 Capture Input)<br>CLKO (system clock output)<br>PCINT0 (pin change interrupt 0) |

**XTALI/TOSC2/PCINT7 – Port B pin 7**

*XTALI*: External crystal oscillator pin XI. When used to drive the external crystal oscillator, this pin will not be used as I/O.

*TOSC2*: Timer external crystal pin 2. When the internal RC is configured as the main working clock of the chip, and the asynchronous timer function is enabled (ASSR register configuration), this pin will be used as the external crystal oscillator pin of the timer. When AS2 of the ASSR register is set to 1 and EXCLK is set to 0, the asynchronous clock function of timer/counter 2 using external crystal oscillator is enabled, PB7 will be disconnected from the internal I/O port and become the inverse of the internal oscillator amplifier output pin. In this mode, the external crystal oscillator is connected to the pin.

*PCINT7*: Pin Change Interrupt 7. PB7 is an external interrupt source.

If PB7 is used as the crystal pin, the values of DDB7, PORTB7 and PINB7 will have no meaning.

**XTALO/TOSC1/PCINT6 - Port B pin 6**

*XTALO*: External crystal oscillator pin XO.

*TOSC1*: Timer external crystal oscillator pin 1. When the internal RC is configured as the main working clock of the chip, and the asynchronous timer function is enabled (ASSR register configuration), this pin will be used as the external crystal oscillator pin of the timer. When AS2 of the ASSR register is set to 1 and EXCLK is set to 0, the asynchronous clock function of

timer/counter 2 using external crystal oscillator is enabled, PB6 will be connected with the internal I/O port port and become the input pin of the internal oscillator amplifier . In this mode, the external crystal oscillator is connected to the pin.

*PCINT6*: Pin Change Interrupt 6. PB6 is an external interrupt source.

If PB6 is used as a crystal pin, the values of DDB6, PORTB6 and PINB6 will have no meaning.

**SCK/PCINT5- Port B pin 5**

*SCK*: SPI controller master device clock output, slave device clock input. When the SPI controller is configured as a slave device, this pin will be configured as an input pin, not controlled by DDB5. When the SPI controller is configured as a master device, the direction of this pin is controlled by DDB5. When this pin is forced as an input by the SPI, the pull-up resistor can still be controlled via the PORTB5 flag.

*PCINT5*: Interrupt on pin change. PB5 is an external interrupt source.

**MISO/PCINT4- Port B pin 4**

*MISO*: SPI controls the data input of the master device and the data output of the slave device. When the SPI is configured as a master device, this pin will be forced as an input and will not be controlled by DDB4. When the SPI is a slave device, the data direction of this pin is controlled by DDB4. When this pin is forced as an input by the SPI controller, its pull-up resistor can still be controlled by PROTB4.

*PCINT4*: Interrupt on pin change. PB4 is an external interrupt source.

**MOSI/OC2A/PCINT3- Port B pin 3**

*MOSI*: SPI controller master device data output, slave device data input. When the SPI is configured as a slave device, this pin will be forced as an input and not controlled by DDB3. When the SPI controller is configured as a master device, the data direction of this pin is controlled by DDB3. When this pin is forced to be an input by SPI control, its pull-up resistor can still be controlled via PORTB3.

*OC2A*: Timer/Event Counter 2 Group A compare match output. PB3 can be used as an external output for Timer/Event Counter 2 compare match. At this point the pin must be set as an output via DDB3. At the same time, OC2A is also the PWM mode output pin of Timer 2.

*PCINT3*: Interrupt on pin change. PB3 is an external interrupt source.

**SSN/OC1B/PCINT2 - Port B pin 2**

*SSN*: SPI slave chip select input. When the SPI controller is configured as a slave device, this pin will be forced as an input and will not be controlled by DDB2. As a slave device, the SPI controller is active when SSN is driven low. When the SPI controller is configured as a master device, the direction of this pin is controlled by DDB2. When this pin is forced as an input by the SPI controller, the pull-up resistor can still be controlled via PORTB2.

*OC1B*: Group B compare match output of Timer/Event Counter 1. PB2 can be used as an external output for Timer/Event Counter 1 compare match. At this point the pin must be set as an output via DDB2. At the same time, OC1B is also the PWM mode output pin of Timer 1.

*PCINT2*: Interrupt on pin change. PB2 is an external interrupt source.

**OC1A/PCINT1- Port B pin 1**

*OC1A*: Timer/Event Counter 1 Group A compare match output. PB1 can be used as an external output for Timer/Event Counter 1 compare match. The pin must be set as an output via DDB1 at this time. At the same time, OC1A is also the PWM mode output pin of Timer 1.

*PCINT1*: Interrupt on pin change. PB1 is an external interrupt source.

**ICP1/CLKO/PCINT0- Port B pin 0**

*ICP1*: Capture input pin for Timer/Event Counter 1

*CLKO*: System operating clock output, when the CLKOE flag in the CLKPR register is 1, this pin will be forced to be an output and not controlled by DDB0. The output frequency is the working clock frequency of the current system.

*PCINT0*: Interrupt on pin change. PB0 is the external interrupt source.

### *Port C alternate function*

| Pin | Alternate Function |
|---|---|
| PC7 | ADC8 (ADC input channel 8)<br>APN2 (DAP reverse input 2)<br>PCINT15 (pin level change input 15) |
| PC6 | RESETN (external reset input)<br>PCINT14 (pin change input 14) |
| PC5 | ADC5 (ADC input channel 5)<br>SCL (TWI clock line)<br>PCINT13 (pin change input 13) |
| PC4 | ADC4 (ADC input channel 4)<br>SDA (TWI data cable)<br>PCINT12 (pin change input 12) |
| PC3 | ADC3 (ADC input channel 3)<br>PCINT11 (pin change input 11) |
| PC2 | ADC2 (ADC input channel 2)<br>PCINT10 (pin change input 10) |
| PC1 | ADC1 (ADC input channel 1)<br>PCINT9 (pin change input 9) |
| PC0 | ADC0 (ADC input channel 0)<br>PCINT8 (pin change input 8) |

**ADC8/APN2/PCINT15 - Port C pin 6**

*ADC8*: ADC external input channel 8

*APN2*: Inverting input port 2 of differential amplifier

*PCINT15*: Interrupt on pin change. After disabling the external reset input function of this pin, PC7 can be used as an external interrupt source.

**RESETN/PCINT14 - Port C pin 6**

*RESETN*: External reset input pin. After power-on reset, this pin defaults to the external reset function. The external reset function can be disabled through the IOCR register. After disabling the external reset function, this pin can be used as a general purpose I/O. However, it should be noted that during power-on and other reset processes, this pin is the reset input by default, so if the user needs to use the general-purpose I/O function of this pin, the external circuit cannot affect the power-on and reset of the chip. During the process, it is recommended to configure this pin as an I/O with output function, and add an appropriate pull-up resistor externally.

*PCINT14*: Interrupt on pin change. After disabling the external reset input function of this pin, PC6 can be used as an external interrupt source.

**SCL/ADC5/PCINT13 - Port C pin 5**

*SCL*: TWI interface clock signal. After the TWEN flag in the TWCR register is set to 1, the TWI interface is enabled, and PC5 will be controlled by the TWI and become the clock signal of the TWI interface.

*ADC5*: ADC input channel 5. The DIDR register is used to close the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*PCINT13*: Pin Change Interrupt 13

**SDA/ADC4/PCINT12 - Port C pin 4**

*SDA*: TWI interface data signal. After the TWEN flag in the TWCR register is set to 1, the TWI interface is enabled, and PC4 will be controlled by the TWI and become the data signal of the TWI interface.

*ADC4*: ADC input channel 4. The DIDR register is used to close the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*PCINT12*: Interrupt on pin change 12

**ADC3/APN1/PCINT11 - Port C pin 3**

*ADC3*: ADC input channel 3. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*APN1*: Differential Amplifier Inverting Input 1

*PCINT11*: Interrupt on pin change 11

**ADC2/APN0/PCINT10- Port C pin 2**

*ADC2*: ADC input channel 2. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*APN0*: Differential Amplifier Inverting Input 0

*PCINT10*: Interrupt on pin change 10

**ADC1/APP1/PCINT9- Port C pin 1**

*ADC1*: ADC input channel 1. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*APP1*: Differential amplifier positive input 1

*PCINT9*: Interrupt on pin change 9

**ADC0/APP0/PCINT8- Port C pin 0**

*ADC0*: ADC input channel 0. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters.

*APP0*: Differential amplifier positive input 0

*PCINT8*: Interrupt on pin change 8

### *Port D alternate function*

| Pin | Alternate Function |
|---|---|
| PD7 | ACXN (Analog Comparator 0/1 Common Negative Input)<br>PCINT23 (pin change interrupt 23) |
| PD6 | AC0P (QFP32: Analog Comparator 0 Positive Input)<br>OC0A (Timer/Event Counter 0 compare match output A)<br>OC3A (QFP32: Timer/Event Counter 3 Compare Match Output A)<br>PCINT22 (pin change interrupt 22) |
| PD5 | T1 (Timer/Event Counter 1 external count clock input)<br>OC0B (Timer/Event Counter 0 compare match output B)<br>PCINT21 (pin change interrupt 21) |
| PD4 | XCK (USART external clock input/output)<br>DAO (internal 8bit DAC analog output)<br>T0 (Timer/Event Counter 0 external count clock input)<br>PCINT20 (pin change interrupt 20) |
| PD3 | INT1 (external interrupt input 1)<br>OC2B (Timer/Event Counter 2 compare match output B)<br>PCINT19 (pin change interrupt 19) |
| PD2 | INT0 (external interrupt input 0)<br>AC0O (Comparator 0 output)<br>OC3B (QFP32: Timer/Event Counter 3 Compare Match Output B)<br>PCINT18 (pin change interrupt 18) |
| PD1 | TXD (USART data output)<br>OC3A (QFP32: Timer/Event Counter 3 Compare Match Output A)<br>PCINT17 (pin change interrupt 17) |
| PD0 | RXD (USART data input)<br>PCINT16 (pin change interrupt 16) |

**ACXN/OC2B/PCINT23- Port D pin 7**

*ACXN*: Analog comparator 0/1 common negative input

*OC2B*: Group B compare match output of Timer/Event Counter 2. PD7 can be used as an external output for Timer/Event Counter 2 compare match. At this point the pin must be set as an output via DDD7. At the same time, OC2B is also the PWM mode output pin of timer 2;

*PCINT23*: Interrupt on pin change 23

**AC0P/OC0A/PCINT22- Port D pin 6**

*AC0P*: Analog Comparator 0 positive input.

*OC0A*: Timer/Event Counter 0 Group A compare match output. PD6 can be used as an external output for Timer/Event Counter 0 compare match. The pin must be set as an output via DDD6 at this time. At the same time, OC0A is also the PWM mode output pin of timer 0

*PCINT22*: Interrupt on pin change 22

**T1/OC0B/PCINT21 - Port D pin 5**

*T1*: External count clock input for Timer/Event Counter 1

*OC0B*: Group B compare match output of Timer/Event Counter 0. PD5 can be used as an external output for Timer/Event Counter 0 compare match. The pin must be set as an output via DDD5 at this time. At the same time, OC0B is also the PWM mode output pin of timer 0

*PCINT21*: Interrupt on pin change 21

**XCK/T0/DAO/PCINT20- Port D pin 4**

*XCK*: External clock signal for USART in synchronous mode

*T0*: External count clock input for Timer/Event Counter 0

*DAO*: Internal 8-bit DAC analog output

*PCINT20*: Interrupt on pin change 20

**INT1/OC2B/PCINT19- Port D pin 3**

*INT1*: External interrupt input 1

*OC2B*: Group B compare match output of Timer/Event Counter 2. PD3 can be used as an external output for Timer/Event Counter 2 compare match. The pin must be set as an output via DDD3 at this time. At the same time, OC2B is also the PWM mode output pin of timer 2

*PCINT19*: Interrupt on pin change 19

**INT0/OC3B/AC0O/PCINT18- Port D pin 2**

*INT0*: External interrupt input 0

*OC3B*: Timer counter 3 compare match output B. Only in QFP32 package, PD2 and QFP48/PF2 are combined into one IO, so the OC3B function on PF2 will also be output from PD2

*AC0O*: The comparison result of analog comparator 0 is output directly. Controlled by AC0FR register

*PCINT18*: Interrupt on pin change 18

**TXD/OC3A/PCINT17- Port D pin 1**

*TXD*: transmit data (USART data output). After the USART transmitter is enabled, PD1 will be forced to output and not controlled by DDD1

*OC3A*: Timer counter 3 compare match output A. Only in QFP32 package, PD1 and QFP48/PF1 are combined into one IO, so the OC3A function on PF1 will also be output from PD1

*PCINT17*: Interrupt on pin change 17

**RXD/PCINT16- Port D pin 0**

*RXD*: transmit data (USART data input). When the USART receiver is enabled, PD0 will be forced as an input and not controlled by DDD0. When the pin is forced as an input by the USART, the pull-up resistor can still be controlled by the PORTD0 flag

*PCINT16*: Interrupt on pin change 16

### *Port E alternate function*

| Pin | Alternate Function |
|---|---|
| PE7 | ADC11 (ADC input channel 11)<br>PCINT31 (pin change interrupt 31) |
| PE6 | AVREF (QFP32: ADC external reference voltage)<br>ADC10 (ADC input channel 10)<br>PCINT30 (pin change interrupt 30) |
| PE5 | CLKO (system clock output)<br>AC1O (Analog Comparator 1 Output)<br>PCINT29 (pin change interrupt 29) |
| PE4 | OC0A (Timer/Event Counter 0 Compare Configuration Output A)<br>PCINT28 (pin change interrupt 28) |
| PE3 | ADC7 (ADC input channel 7)<br>AC1N (Analog Comparator 1 Negative Input)<br>PCINT27 (pin change interrupt 27) |
| PE2 | SWD (SWD Debugger Data Line)<br>PCINT26 (pin change interrupt 26) |
| PE1 | ADC6 (ADC input channel 6)<br>ACXP (analog ratio machine 0/1 common positive input)<br>PCINT25 (pin change interrupt 25) |
| PE0 | SWC (SWD Debugger Clock Input)<br>APN4 (Differential Amplifier Inverting Input 4)<br>PCINT24 (pin change interrupt 24) |

**ADC11/PCINT31- Port E pin 7**

*ADC11*: ADC external input channel 11

*PCINT31*: Interrupt on pin change 30

**AVREF/ADC10/PCINT30 - Port E pin 6**

*AVREF*: ADC external reference power input, when used as analog input, you need to set the corresponding digital I/O as input, and turn off the pull-up resistor to avoid the digital circuit interfering with the analog circuit

*ADC10*: ADC analog input channel 10

*PCINT30*: Interrupt on pin change 30

**CLKO/AC1O/PCINT29 - Port E pin 5**

*CLKO*: This function is the same as the CLKO function of PB0. Can be used as alternate pin for PB0/CLKO

*AC1O*: Analog Comparator 1 Output

*PCINT29*: Pin Change Interrupt 29

**OC0A/PCINT28- Port E pin 4**

*OC0A*: Timer/Event Counter 0 Group A compare match output. PE4 can be used as an external output for Timer/Event Counter 0 compare match. At this point the pin must be set as an output via DDE4. At the same time, OC0A is also the PWM mode output pin of Timer 0.

*PCINT28*: Interrupt on pin change 28

**ADC7/ AC1N/PCINT27 - Port E pin 3**

*ADC7*: ADC input channel 7. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters

*AC1N*: Analog Comparator 1 Negative Input

*PCINT27*: Interrupt on pin change 27

**SWD/PCINT26 - Port E pin 2**

*SWD*: SWD debugger data line. PE2 defaults to SWD functionality. The user can disable the SWD debugger function by setting the SWDD flag of the MCUSR register to 1. After SWD is turned off, the debugging function will not be available.

*PCINT26*: Interrupt on pin change 26

**ADC6/ACXP/PCINT25 - Port E pin 1**

*ADC6*: ADC input channel 6. The DIDR register is used to terminate the digital function of the digital-analog multiplexing I/O to avoid the influence of the digital part on the analog circuit. For details, please refer to ADC related chapters

*ACXP*: Analog comparator 0/1 common positive input

*PCINT25*: Interrupt on pin change 25

**SWC/APN4/PCINT24 - Port E pin 0**

*SWC*: SWD debugger clock line. PE0 defaults to the SWC function. The user can disable the SWD debugger function by setting the SWDD flag of the MCUSR register to 1. After SWD is turned off, the debugging function will not work

*APN4*: Differential Amplifier Inverting Input 4

*PCINT24*: Interrupt on pin change 24

### *Port F alternate function*

| Pin | Alternate Function |
|---|---|
| PF7 | OC2B (Timer/Event Counter 2 compare match output B)<br>PCINT39 (pin change interrupt 39) |
| PF6 | T3 (Timer/Event Counter 3 external clock input)<br>OC2A (Timer/Event Counter 2 compare match output A)<br>PCINT38 (pin change interrupt 38) |
| PF5 | OC1A (Timer/Event Counter 1 compare match output A)<br>PCINT37 (pin change interrupt 37) |
| PF4 | OC1B (Timer/Event Counter 1 Compare Configuration Output B)<br>ICP3 (Timer/Counter 3 external capture input)<br>PCINT36 (pin change interrupt 36) |
| PF3 | OC0B (Timer/Event Counter 0 Compare Configuration Output B)<br>PCINT35 (pin change interrupt 35) |
| PF2 | OC3B (Timer/Event Counter 3 compare match output B)<br>PCINT34 (pin change interrupt 34) |
| PF1 | OC3A (Timer/Event Counter 3 compare match output A)<br>PCINT33 (pin change interrupt 33) |
| PF0 | ADC9 (ADC external input channel 9)<br>APN3 (Differential Amplifier Inverting Input 3)<br>PCINT32 (pin change interrupt 32) |

**OC2B/PCINT39 - Port F pin 7**

OC2B: Timer/Event Counter 2 compare match output B. Output selection is controlled by the PMX1 register

PCINT39: Interrupt on pin change 39

**OC2A/T3/PCINT38 - Port F pin 6**

OC2A: Timer/Event Counter 2 compare match output A. Output selection is controlled by the PMX1 register

T3: Timer/Event Counter 3 external clock input

PCINT38: Interrupt on pin change 38

**0C1A/PCINT37 - Port F pin 5**

OC1A: Timer/Event Counter 1 compare match output A. Output selection is controlled by the PMX0 register

PCINT37: Interrupt on pin change 37

**ICP3/OC1B/PCINT36 - Port F pin 4**

OC1B: Group B compare match output of Timer/Event Counter 1. Output selection is controlled by the PMX0 register

ICP3: Timer/Counter 3 External Capture Input

PCINT36: Interrupt on pin change 36

**OC3C/OC0B/PCINT35- Port F pin 3**

OC0B: Group B compare match output of Timer/Event Counter 0. Output selection is controlled by the PMX0 register

OC3C: Timer/Event Counter 3 Group C Compare Match Output

PCINT35: Interrupt on pin change 35

**OC3B/PCINT34 - Port F pin 2**

OC3B: Group B compare match output of Timer/Event Counter 3

PCINT34: Interrupt on pin change 34

**OC3A/PCINT33- Port F pin 1**

OC3A: Group B compare match output of Timer/Event Counter 3. Output selection is controlled by the PMX1 register

PCINT33: Interrupt on pin change 33

**ADC9/APN3/PCINT32- Port F pin 0**

ADC9: ADC external mode input channel 9

APN3: Differential Amplifier Inverting Input 3

PCINT32: Interrupt on pin change 32

## Register definition

### *Port B Output Data Register - PORTB*

| PORTB – Port B Output Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PORTB: 0x05(0x25) | | | | Default: 0x00 | | | | |
| Bits | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PORTB | Group B Port Output Registers | | | | | | |

### *Port B Direction Register - DDRB*

| DDRB – Port B Direction Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DDRB: 0x04(0x24) | | | | Default: 0x00 | | | | |
| DDRB | DDB7 | DDB6 | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | DDB | Port B group direction control bits; 1 = output, 0 = input | | | | | | |

### *Port B Input Data Register - PINB*

| PINB – Port B Input Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PINB: 0x03(0x23) | | | | Default: 0x00 | | | | |
| PINB | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PINB | Group B port status register. Read PINB to get the current state of the port directly;<br>Setting PINBn flag to 1 will toggle the output state of PORTBn | | | | | | |

### *Port C Output Data Register - PORTC*

| PORTC – Port C Output Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PORTC: 0x08(0x28) | | | | Default: 0x00 | | | | |
| PORTC | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PORTC | Group C Port Output Registers | | | | | | |

### *Port C Direction Register - DDRC*

| DDRC – Port C Direction Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DDRC: 0x07(0x27) | | | | Default: 0x00 | | | | |
| DDRC | DDC7 | DDC6 | DDC5 | DDC4 | DDC3 | DDC2 | DDC1 | DDC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | DDC | Group C port direction control bits; 1 = output, 0 = input | | | | | | |

### *Port C Input Data Register - PINC*

| PINC – Port C Input Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PINB: 0x06(0x26) | | | | Default: 0x00 | | | | |
| PINC | PINC7 | PINC6 | PINC5 | PINC4 | PINC3 | PINC2 | PINC1 | PINC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PINC | Group C port status register; read PINC to get the current port status<br>Writing to PINC will toggle the current port output | | | | | | |

### *Port D Output Data Register - PORTD*

| PORTD – Port D Output Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PORTD: 0x0B(0x2B) | | | | Default: 0x00 | | | | |
| Bits | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PORTD | Group D Port Output Registers | | | | | | |

### *Port D Direction Register - DDRD*

| DDRD – Port D Direction Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DDRD: 0x0A(0x2A) | | | | Default: 0x00 | | | | |
| DDRD | DDD7 | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | DDD | D group port output direction control register | | | | | | |

### *Port D Input Data Register - PIND*

| PIND – Port D Input Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PIND: 0x09(0x29) | | | | | Default: 0x00 | | | |
| PIND | PIND7 | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PIND | Group D Port Status Register<br>Read PIND to get the current port level status<br>Write PINDn to 1, flip the state of the corresponding flag in PORTDn | | | | | | |

### *Port E Output Data Register - PORTE*

| PORTE – Port E output data register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PORTE: 0x0E(0x2E) | | | | | Default: 0x00 | | | |
| Bits | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PORTE | Group E Port Output Registers | | | | | | |

### *Port E Direction Register - DDRE*

| DDRE – Port E Direction Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DDRE: 0x0D(0x2D) | | | | | Default: 0x00 | | | |
| DDRE | DDE7 | DDE6 | DDE5 | DDE4 | DDE3 | DDE2 | DDE1 | DDE0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | DDE | Group E Port Direction Control Register | | | | | | |

### *Port E Input Data Register - PINE*

| PINE – Port E Input Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PINE: 0x0C(0x2C) | | | | | Default: 0x00 | | | |
| PINE | PINE7 | PINE6 | PINE5 | PINE4 | PINE3 | PINE2 | PINE1 | PINE0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PINE | Group E Port Status Register<br>Read PINE to get the current port level status<br>Writing PINEn to 1 flips the state of the PORTEn flag | | | | | | |

### *Port F Output Register – PORTF*

| PINF – Port F Input Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PORTF: 0x14(0x34) | | | | | Default: 0x00 | | | |
| Bits | PF7 | PF6 | PF5 | PF4 | PF3 | PF2 | PF1 | PF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PORTF | Group F Port Status Register<br>Port in input mode, writing 1 to the corresponding flag will enable the internal pull-up<br>Port in output mode, writing 1 to the corresponding flag will drive the output to a high level | | | | | | |

### *Port F Direction Control Register - DDRF*

| DDRF – Port F Direction Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DDRF: 0x13(0x33) | | | | | Default: 0x00 | | | |
| Bits | DDF7 | DDF6 | DDF5 | DDF4 | DDF3 | DDF2 | DDF1 | DDF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | DDRF | Group F Port Direction Control Register | | | | | | |

### *Port F Status Register - PINF*

| PINF – Port F Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PINF: 0x12(0x32) | | | | | Default: 0x00 | | | |
| Bits | PINF7 | PINF6 | PINF5 | PINF4 | PINF3 | PINF2 | PINF1 | PINF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:0] | PINF | Group F Port Status Register<br>Read PINF to get the current level state of port F<br>Write 1 to PINFn, flip the state of the corresponding flag of PORTFn | | | | | | |

### *Port Driver Control Register - HDR*

| HDR0 – Port Driver Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| HDR: 0xE0 | | | | | Default: 0x00 | | | |
| Bits | - | - | HDR5 | HDR4 | HDR3 | HDR2 | HDR1 | HDR0 |
| R/W | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:6] | - | Reserved | | | | | | |
| 5 | HDR5 | PF5 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |
| 4 | HDR4 | PF4 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |
| 3 | HDR3 | PF2 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |
| 2 | HDR2 | PF1 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |
| 1 | HDR1 | PD6 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |
| 0 | HDR0 | PD5 output drive control; 1 = 80mA drive, 0 = 12mA drive | | | | | | |

### *Port Multiplexing Control Register 0- PMX0*

| PMX0 – Port Multiplexing Control Register 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PMX0: 0xEE | | | | | Default: 0x00 | | | |
| Bit | WCE | C1BF4 | C1AF5 | C0BF3 | C0AC0 | SSB1 | TXD6 | RXD5 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | | | | | | | | |
| 7 | WCE | PMX0/1 update enable control; before updating the PMX0/1 register, the WCE flag needs to be written to 1, and the update to PMX0/1 is completed in the next 6 system cycles. | | | | | | |
| 6 | C1BF4 | OC1B Auxiliary Output Control<br>1 = OC1B output to PF4<br>0 = OC1B output to PB2 | | | | | | |
| 5 | C1AF5 | OC1A Auxiliary Output Control<br>1 = OC1A output to PF5<br>0 = OC1A output to PB1 | | | | | | |
| 4 | C0BF3 | OC0B Auxiliary output control<br>1 = OC0B output to PF3<br>0 = OC0B output to PD5 | | | | | | |
| 3 | C0AC0 | OC0A Auxiliary output control<br>The 0C0A output is controlled by the C0AC0 flag in conjunction with the C0AS flag of the TCCR0B register:<br>{ C0AC0, C0AS} =<br>00 = OC0A output to PD6<br>01 = 0C0A output to PE4<br>10 = 0C0A output to PC0<br>11 = OC0A is output to both PE4 and PC0 | | | | | | |

| | | |
|---|---|---|
| 2 | SSB1 | SPSS Auxiliary Output Control<br>1 = SPSS output to PB1<br>0 = SPSS output to PB2 |
| 1 | TXD6 | Serial port TXD auxiliary output control<br>1 = TXD output to PD6, 0 = TXD output to PD1 |
| 0 | RXD5 | Serial RXD Auxiliary Input Control<br>1 = RXD input is from PD5, 0 = RXD input is from PD0 |

### *Port Multiplexing Control Register 1- PMX1*

| PMX1 – Port Multiplexing Control Register 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PMX1: 0xED | | | | | Default: 0x00 | | | |
| Bit | - | - | - | - | - | C3AC | C2BF7 | C2AF6 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |
| Bit | | | | | | | | |
| [7:3] | - | Reserved | | | | | | |
| 2 | C3AC | OC3A Auxiliary Output Control<br>1 = OC3A output to QFP48/AC0P<br>0 = OC3A output to PF1 | | | | | | |
| 1 | C2BF7 | OC2B Auxiliary Output Control<br>1 = OC2B output to PF7<br>0 = OC2B output to PD3 | | | | | | |
| 0 | C2AF6 | OC2A Auxiliary Output Control<br>1 = OC2A output to PF6<br>0 = OC2A output to PB3 | | | | | | |
| Instructions for use | | | | | | | | |
| PMX0/1 shared register update protection control flag PMX0[7], when updating PMX1, please refer to the control description of PMX0 register to PMX0[7]. | | | | | | | | |

### *Port Multiplexing Control Register 2 – PMX2*

| PMX2 – Port Multiplexing Control Register 2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PMX2: 0xF0 | | | | Default: 0x00 | | | | |
| Bit | WCE | STSC1 | STSC0 | - | - | XIEN | E6EN | C6EN |
| R/W | R/W | R/W | R/W | - | - | R/W | R/W | R/W |

| Bit | | |
|---|---|---|
| 7 | WCE | PMX2 update enable control; before updating the PMX2 register, the WCE flag needs to be written to 1, and the update to PMX2 is completed in the next 6 system cycles. |
| 6 | STSC1 | High-speed crystal oscillator IO startup circuit control<br>After enabling the high-speed crystal oscillator through PMCR, STSC1 is automatically enabled. After switching the system clock to an external high-speed crystal oscillator, STSC1 is automatically cleared. The software can also manually clear STSC1 after the crystal oscillator is stable, and the crystal oscillator startup circuit has been turned off to save power consumption. |
| 5 | STSC0 | Low-speed crystal oscillator IO startup circuit control<br>After enabling the low-speed crystal oscillator through PMCR, STSC0 is automatically enabled. After switching the system clock to an external low-speed crystal oscillator, STSC0 is automatically cleared. The software can also manually clear STSC0 after the crystal oscillator is stable, and the crystal oscillator startup circuit has been closed to save power consumption. |
| 4:3 | - | Reserved |
| 2 | XIEN | To enable external clock input, external crystal oscillator needs to be enabled at the same time |
| 1 | E6EN | Enable the general-purpose IO function of PE6; the default PE6 is the AVREF function |
| 0 | C6EN | Enable general-purpose IO function of PC6; default PC6 is external reset input |

# Interrupt on pin change

- 40 pin-change interrupt-on-change sources
- 5 interrupt entries

## Overview

Pin-on-change interrupts are triggered by the PBn, PCn, PDn, PEn and PFn pins. Interrupts can be triggered even if these pins are configured as outputs, as long as interrupt-on-change pins are enabled. This can be used to generate software interrupts.

Any enabled PBn pin flip will trigger pin level interrupt PCI0, enabled PCn pin flip will trigger PCI1, enabled PDn pin flip will trigger PCI2, and enabled PEn pin flip will trigger PCI3 . The enable of each pin change interrupt is controlled by the PCMSK0~4 registers respectively. All pin-change interrupts are detected asynchronously and can be used as a wake-up source in some sleep modes.

## Register definition

Pin Change Interrupt Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| PCICR | 0x68 | 0x00 | Pin Change Interrupt Control Register |
| PCIFR | 0x3B | 0x00 | Pin Change Interrupt Flag Register |
| PCMSK0 | 0x6B | 0x00 | Pin Change Interrupt Mask Register 0 |
| PCMSK1 | 0x6C | 0x00 | Pin Change Interrupt Mask Register 1 |
| PCMSK2 | 0x6D | 0x00 | Pin Change Interrupt Mask Register 2 |
| PCMSK3 | 0x73 | 0x00 | Pin Change Interrupt Mask Register 3 |
| PCMSK4 | 0x74 | 0x00 | Pin Change Interrupt Mask Register 4 |

## *PCICR - Pin Change Interrupt Control Register*

| PCICR - Pin Change Interrupt Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x68 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | PCIE4 | PCIE3 | PCIE2 | PCIE1 | PCIE0 |
| R/W | - | - | - | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:5 | - | Reserve |
| 4 | PCIE4 | Pin Change Interrupt Enable Control flag 4.<br>When the PCIE4 flag is set to "1" and the global interrupt is enabled, the pin change interrupt 4 is enabled.<br>A change in any of the enabled PFn pins will generate a PCI4 interrupt. The enable of the PFn pin interrupt can be controlled separately by the PCMSK4 register.<br>When the PCIE3 flag is set to "0", the pin change interrupt 3 is disabled. |
| 3 | PCIE3 | Pin Change Interrupt Enable Control flag 3.<br>When the PCIE3 flag is set to "1" and the global interrupt is enabled, the pin change interrupt 3 is enabled.<br>A change in any of the enabled PEn pins will generate a PCI3 interrupt. The enable of the PEn pin interrupt can be controlled separately by the PCMSK3 register.<br>When the PCIE3 flag is set to "0", the pin change interrupt 3 is disabled. |
| 2 | PCIE2 | Pin Change Interrupt Enable Control flag 2.<br>When the PCIE2 flag is set to "1" and the global interrupt is enabled, the pin change interrupt 2 is enabled.<br>A PCI2 interrupt is generated by a change on any of the enabled PDn pins. The enable of the PDn pin interrupt can be controlled separately by the PCMSK2 register.<br>When the PCIE2 flag is set to "0", the pin change interrupt 2 is disabled. |
| 1 | PCIE1 | Pin change interrupt enable control flag 1.<br>When the PCIE1 flag is set to "1" and the global interrupt is enabled, the pin change interrupt 1 is enabled.<br>A PCI1 interrupt is generated by a change on any of the enabled PCn pins. The enable of the PCn pin interrupt can be controlled separately by the PCMSK1 register.<br>When the PCIE1 flag is set to "0", the pin change interrupt 1 is disabled. |
| 0 | PCIE0 | Pin change interrupt enable control flag 0.<br>When the PCIE0 flag is set to "1" and the global interrupt is enabled, the pin change interrupt 0 is enabled.<br>A PCI0 interrupt is generated by a change on any of the enabled PBn pins. The enable of the PBn pin interrupt can be controlled separately by the PCMSK0 register.<br>When setting the PCIE0 flag to "0", pin change interrupt 0 is disabled. |

## *PCIFR - Pin Change Interrupt Flag Register*

| PCIFR - Pin Change Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x3B | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | PCIF4 | PCIF3 | PCIF2 | PCIF1 | PCIF0 |
| R/W | - | - | - | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:5 | - | Reserve |
| 4 | PCIF4 | Pin change interrupt flag flag 4.<br>A change on any of the enabled PFn pins will set PCIF4. When both PCIE4 and global interrupt are set, the MCU will jump to the PCI4 interrupt entry address. The enable of the PFn pin interrupt can be controlled separately by the PCMSK4 register.<br>Executing an Interrupt Service Routine or writing a '1' to the PCIF4 flag will clear the PCIF4 flag. |
| 3 | PCIF3 | Pin change interrupt flag 3.<br>A change on any of the enabled PEn pins will set PCIF3. When both PCIE3 and global interrupt are set, the MCU will jump to the PCI3 interrupt entry address. The enabling of the PEn pin interrupts can be controlled separately by the PCMSK3 register.<br>Executing an Interrupt Service Routine or writing a '1' to the PCIF3 flag will clear the PCIF3 flag. |
| 2 | PCIF2 | Pin change interrupt flag 2.<br>A change on any of the enabled PDn pins will set<br>PCIF2. When both PCIE2 and global interrupt are set, the MCU will jump to the PCI2 interrupt entry address. The enabling of interrupts on the PDn pins can be controlled separately by the PCMSK2 register.<br>Executing an Interrupt Service Routine or writing a '1' to the PCIF2 flag will clear the PCIF2 flag. |
| 1 | PCIF1 | Pin change interrupt flag 1.<br>A change on any of the enabled PCn pins will set PCIF1. When both PCIE1 and global interrupt are set, the MCU will jump to the PCI1 interrupt entry address. The enabling of the PCn pin interrupts can be controlled separately by the PCMSK1 register.<br>Executing an Interrupt Service Routine or writing a '1' to the PCIF1 flag will clear the PCIF1 flag. |
| 0 | PCIF0 | Pin change interrupt flag 0.<br>A change on any of the enabled PBn pins will set PCIF0. When both PCIE0 and global interrupt are set, the MCU will jump to the PCI0 interrupt entry address. The enable of the PBn pin interrupt can be controlled separately by the PCMSK0 register.<br>Executing an Interrupt Service Routine or writing a '1' to the PCIF0 flag will clear the PCIF0 flag. |

## *PCMSK0 – Pin Change Interrupt Mask Register 0*

| PCMSK0 – Pin Change Mask Register 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x6B | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PCINT7 | PCINT6 | PCINT5 | PCINT4 | PCINT3 | PCINT2 | PCINT1 | PCINT0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PCINT7 | Pin change enable mask flag 7.<br>When the PCINT7 flag is set to "1", the PB7 pin change interrupt is enabled. A level change on the PB7 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT7 flag is set to "0", the PB7 pin change interrupt is disabled. |
| 6 | PCINT6 | Pin change enable mask flag 6.<br>When the PCINT6 flag is set to "1", the PB6 pin change interrupt is enabled. A level change on the PB6 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT6 flag is set to "0", the PB6 pin change interrupt is disabled. |
| 5 | PCINT5 | Pin change enable mask flag 5.<br>When the PCINT5 flag is set to "1", the PB5 pin change interrupt is enabled. A level change on the PB5 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT5 flag is set to "0", the PB5 pin change interrupt is disabled. |
| 4 | PCINT4 | Pin change enable mask flag 4.<br>When the PCINT4 flag is set to "1", the PB4 pin change interrupt is enabled. A level change on the PB4 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT4 flag is set to "0", the PB4 pin change interrupt is disabled. |
| 3 | PCINT3 | Pin change enable mask flag 3.<br>When the PCINT3 flag is set to "1", the PB3 pin change interrupt is enabled. A level change on the PB3 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT3 flag is set to "0", the PB3 pin change interrupt is disabled. |
| 2 | PCINT2 | Pin change enable mask flag 2.<br>When the PCINT2 flag is set to "1", the PB2 pin change interrupt is enabled. A level change on the PB2 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated.<br>When the PCINT2 flag is set to "0", the PB2 pin change interrupt is disabled. |
| 1 | PCINT1 | Pin change enable mask flag 1.<br>When the PCINT1 flag is set to "1", the PB1 pin change interrupt is enabled. A level change on the PB1 pin will set PCIF0, and if the PCIE0 |

| | | |
|---|---|---|
| | | flag and the global interrupt are set, a PCI0 interrupt will be generated. When the PCINT1 flag is set to "0", the PB1 pin change interrupt is disabled. |
| 0 | PCINT0 | Pin change enable mask flag 0. When the PCINT0 flag is set to "1", the PB0 pin change interrupt is enabled. A level change on the PB0 pin will set PCIF0, and if the PCIE0 flag and the global interrupt are set, a PCI0 interrupt will be generated. When the PCINT0 flag is set to "0", the PB0 pin change interrupt is disabled. |

### *PCMSK1 – Pin Change Interrupt Mask Register 1*

| PCMSK1 – Pin Change Mask Register 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x6C | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PCINT15 | PCINT14 | PCINT13 | PCINT12 | PCINT11 | PCINT10 | PCINT9 | PCINT8 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PCINT15 | Pin change enable mask flag 15. When the PCINT15 flag is set to "1", the PC7 pin change interrupt is enabled. A level change on the PC7 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated. When the PCINT15 flag is set to "0", the PC7 pin change interrupt is disabled. |
| 6 | PCINT14 | Pin change enable mask flag 14. When the PCINT14 flag is set to "1", the PC6 pin change interrupt is enabled. A level change on the PC6 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated. When the PCINT14 flag is set to "0", the PC6 pin change interrupt is disabled. |
| 5 | PCINT13 | Pin change enable mask flag 13. When the PCINT13 flag is set to "1", the PC5 pin change interrupt is enabled. A level change on the PC5 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated. When the PCINT13 flag is set to "0", the PC5 pin change interrupt is disabled. |
| 4 | PCINT12 | Pin change enable mask flag 12. When the PCINT12 flag is set to "1", the PC4 pin change interrupt is enabled. A level change on the PC4 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated. When the PCINT12 flag is set to "0", the PC4 pin change interrupt is disabled. |
| 3 | PCINT11 | Pin change enable mask flag 11. When the PCINT11 flag is set to "1", the PC3 pin change interrupt is enabled. A level change on the PC3 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated. |

| | | |
|---|---|---|
| | | When the PCINT11 flag is set to "0", the PC3 pin change interrupt is disabled. |
| 2 | PCINT10 | Pin change enable mask flag 2.<br>When the PCINT10 flag is set to "1", the PC2 pin change interrupt is enabled. A level change on the PC2 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated.<br>When the PCINT10 flag is set to "0", the PC2 pin change interrupt is disabled. |
| 1 | PCINT9 | Pin change enable mask flag 1.<br>When the PCINT9 flag is set to "1", the PC1 pin change interrupt is enabled. A level change on the PC1 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated.<br>When the PCINT9 flag is set to "0", the PC1 pin change interrupt is disabled. |
| 0 | PCINT8 | Pin change enable mask flag 0.<br>When the PCINT8 flag is set to "1", the PC0 pin change interrupt is enabled. A level change on the PC0 pin will set PCIF1, and if the PCIE1 flag and the global interrupt are set, a PCI1 interrupt will be generated.<br>When the PCINT8 flag is set to "0", the PC0 pin change interrupt is disabled. |

### *PCMSK2 – Pin Change Interrupt Mask Register 2*

| PCMSK2 – Pin Change Mask Register 2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x6D | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PCINT23 | PCINT22 | PCINT21 | PCINT20 | PCINT19 | PCINT18 | PCINT17 | PCINT16 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PCINT23 | Pin change enable mask flag 23.<br>When the PCINT23 flag is set to "1", the PD7 pin level change interrupt is enabled. A level change on the PD7 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT23 flag is set to "0", the PD7 pin level change interrupt is disabled. |
| 6 | PCINT22 | Pin change enable mask flag 6.<br>When the PCINT22 flag is set to "1", the PD6 pin level change interrupt is enabled. A level change on the PD6 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT22 flag is set to "0", the PD6 pin level change interrupt is disabled. |
| 5 | PCINT21 | Pin change enable mask flag 21.<br>When the PCINT21 flag is set to "1", the PD5 pin level change interrupt is enabled. A level change on the PD5 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be |

| | | |
|---|---|---|
| | | generated.<br>When the PCINT21 flag is set to "0", the PD5 pin level change interrupt is disabled. |
| 4 | PCINT20 | Pin change enable mask flag 20.<br>When the PCINT20 flag is set to "1", the PD4 pin change interrupt is enabled. A level change on the PD4 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT20 flag is set to "0", the PD4 pin level change interrupt is disabled. |
| 3 | PCINT19 | Pin change enable mask flag 19.<br>When the PCINT19 flag is set to "1", the PD3 pin level change interrupt is enabled. A level change on the PD3 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT19 flag is set to "0", the PD3 pin level change interrupt is disabled. |
| 2 | PCINT18 | Pin change enable mask flag 18.<br>When the PCINT18 flag is set to "1", the PD2 pin level change interrupt is enabled. A level change on the PD2 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT18 flag is set to "0", the PD2 pin level change interrupt is disabled. |
| 1 | PCINT17 | Pin change enable mask flag 17.<br>When the PCINT17 flag is set to "1", the PD1 pin level change interrupt is enabled. A level change on the PD1 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT17 flag is set to "0", the PD1 pin level change interrupt is disabled. |
| 0 | PCINT16 | Pin change enable mask flag 16.<br>When the PCINT16 flag is set to "1", the PD0 pin level change interrupt is enabled. A level change on the PD0 pin will set PCIF2, and if the PCIE2 flag and the global interrupt are set, a PCI2 interrupt will be generated.<br>When the PCINT16 flag is set to "0", the PD0 pin level change interrupt is disabled. |

## *PCMSK3 – Pin Change Interrupt Mask Register 3*

| PCMSK3 – Pin Change Mask Register 3 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x73 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PCINT31 | PCINT30 | PCINT29 | PCINT28 | PCINT27 | PCINT26 | PCINT25 | PCINT24 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PCINT31 | Pin change enable mask flag 31.<br>When the PCINT31 flag is set to "1", the PE7 pin level change interrupt is enabled. A level change on the PE7 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT31 flag is set to "0", the PE7 pin level change interrupt is disabled. |
| 6 | PCINT30 | Pin change enable mask flag 30.<br>When the PCINT30 flag is set to "1", the PE6 pin level change interrupt is enabled. A level change on the PE6 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT30 flag is set to "0", the PE6 pin level change interrupt is disabled. |
| 5 | PCINT29 | Pin change enable mask flag 39.<br>When the PCINT29 flag is set to "1", the PE5 pin level change interrupt is enabled. A level change on the PE5 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT29 flag is set to "0", the PE5 pin level change interrupt is disabled. |
| 4 | PCINT28 | Pin change enable mask flag 28.<br>When the PCINT28 flag is set to "1", the PE4 pin level change interrupt is enabled. A level change on the PE4 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT28 flag is set to "0", the PE4 pin level change interrupt is disabled. |
| 3 | PCINT27 | Pin change enable mask flag 27.<br>When the PCINT27 flag is set to "1", the PE3 pin change interrupt is enabled. A level change on the PE3 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT27 flag is set to "0", the PE3 pin level change interrupt is disabled. |
| 2 | PCINT26 | Pin change enable mask flag 26.<br>When the PCINT26 flag is set to "1", the PE2 pin level change interrupt is enabled. A level change on the PE2 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated. |

| | | |
|---|---|---|
| | | When the PCINT26 flag is set to "0", the PE2 pin level change interrupt is disabled. |
| 1 | PCINT25 | Pin change enable mask flag 25.<br>When the PCINT25 flag is set to "1", the PE1 pin level change interrupt is enabled. A level change on the PE1 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT25 flag is set to "0", the PE1 pin level change interrupt is disabled. |
| 0 | PCINT24 | Pin change enable mask flag 24.<br>When the PCINT24 flag is set to "1", the PE0 pin level change interrupt is enabled. A level change on the PE0 pin will set PCIF3, and if the PCIE3 flag and the global interrupt are set, a PCI3 interrupt will be generated.<br>When the PCINT24 flag is set to "0", the PE0 pin level change interrupt is disabled. |

### *PCMSK4 – Pin Change Interrupt Mask Register 4*

| PCMSK4 – Pin Change Mask Register 4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x74 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PCINT39 | PCINT38 | PCINT37 | PCINT36 | PCINT35 | PCINT34 | PCINT33 | PCINT32 |
| R/W | R/W | R/W | R/W | R/W | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PCINT39 | Pin change enable mask flag 39.<br>When the PCINT39 flag is set to "1", the PF7 pin level change interrupt is enabled. A level change on the PF7 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT39 flag is set to "0", the PF7 pin level change interrupt is disabled. |
| 6 | PCINT38 | Pin change enable mask flag 38.<br>When the PCINT38 flag is set to "1", the PF6 pin level change interrupt is enabled. A level change on the PF6 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT38 flag is set to "0", the PF6 pin level change interrupt is disabled. |
| 5 | PCINT37 | Pin change enable mask flag 37.<br>When the PCINT37 flag is set to "1", the PF5 pin level change interrupt is enabled. A level change on the PF5 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT37 flag is set to "0", the PF5 pin level change interrupt is disabled. |

| | | |
|---|---|---|
| 4 | PCINT36 | Pin change enable mask flag 36.<br>When the PCINT36 flag is set to "1", the PF4 pin level change interrupt is enabled. A level change on the PF4 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT36 flag is set to "0", the PF4 pin level change interrupt is disabled. |
| 3 | PCINT35 | Pin change enable mask flag 35.<br>When the PCINT35 flag is set to "1", the PF3 pin level change interrupt is enabled. A level change on the PF3 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT35 flag is set to "0", the PF3 pin level change interrupt is disabled. |
| 2 | PCINT34 | Pin change enable mask flag 34.<br>When the PCINT34 flag is set to "1", the PF2 pin level change interrupt is enabled. A level change on the PF2 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT34 flag is set to "0", the PF2 pin level change interrupt is disabled. |
| 1 | PCINT33 | Pin change enable mask flag 33.<br>When the PCINT33 flag is set to "1", the PF1 pin level change interrupt is enabled. A level change on the PF1 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT33 flag is set to "0", the PF1 pin level change interrupt is disabled. |
| 0 | PCINT32 | Pin change enable mask flag 32.<br>When the PCINT31 flag is set to "1", the PF0 pin level change interrupt is enabled. A level change on the PF0 pin will set PCIF4, and if the PCIE4 flag and the global interrupt are set, a PCI4 interrupt will be generated.<br>When the PCINT32 flag is set to "0", the PF0 pin level change interrupt is disabled. |

# Timer/Counter 0 (TMR0)

- 8-bit counter
- Two independent comparison units
- Auto clear counter and auto reload on compare match
- Phase-corrected PWM output with no glitches
- frequency generator
- External event counter
- 10-bit clock prescaler
- Overflow and Compare Match Interrupts
- With dead time control
- 6 selectable trigger sources automatically shut down PWM output
- Generate high-speed high-resolution (500KHz@7Bit) PWM in high-speed clock mode

## Overview

TC0 is a general-purpose 8-bit timer counter module that supports PWM output and can accurately generate waveforms. TC0 includes a count clock generation unit, an 8-bit counter, a waveform generation mode control unit and 2 output comparison units. At the same time, TC0 can share a 10-bit prescaler with TC1, or can use a 10-bit prescaler independently. The prescaler divides the system clock clkio or the high-speed clock rcm2x (2 times the output clock of the internal 32M RC oscillator rc32m) to generate the count clock Clkt0. The waveform generation mode control unit controls the working mode of the counter and the generation of the comparison output waveform. According to different working modes, the counter implements clearing, incrementing or decrementing for each count clock Clkt0. Clkt0 can be generated from an internal clock source or an external clock source. When the count value TCNT0 of the counter reaches the maximum value (equal to the maximum value 0xFF or the output compare register OCR0A, which is defined as TOP, and the maximum value is defined as MAX to show the difference), the counter will be cleared or decremented by one. When the count value TCNT0 of the counter reaches the minimum value (equal to 0x00, defined as BOTTOM), the counter will increment by one. When the count value TCNT0 of the counter reaches OCR0A/OCR0B, which is also called a compare match, the output compare signal OC0A/OC0B will be cleared or set to generate a PWM waveform. When the dead time insertion is enabled, the set dead time (counting clocks corresponding to the DTR0 register) will be inserted into the generated PWM waveform. Software can close the waveform output of OC0A/OC0B by clearing the COM0A/COM0B flag to zero, or set the corresponding trigger source. When a trigger event occurs, the hardware will automatically clear the COM0A/COM0B flag to close the waveform output of OC0A/OC0B.

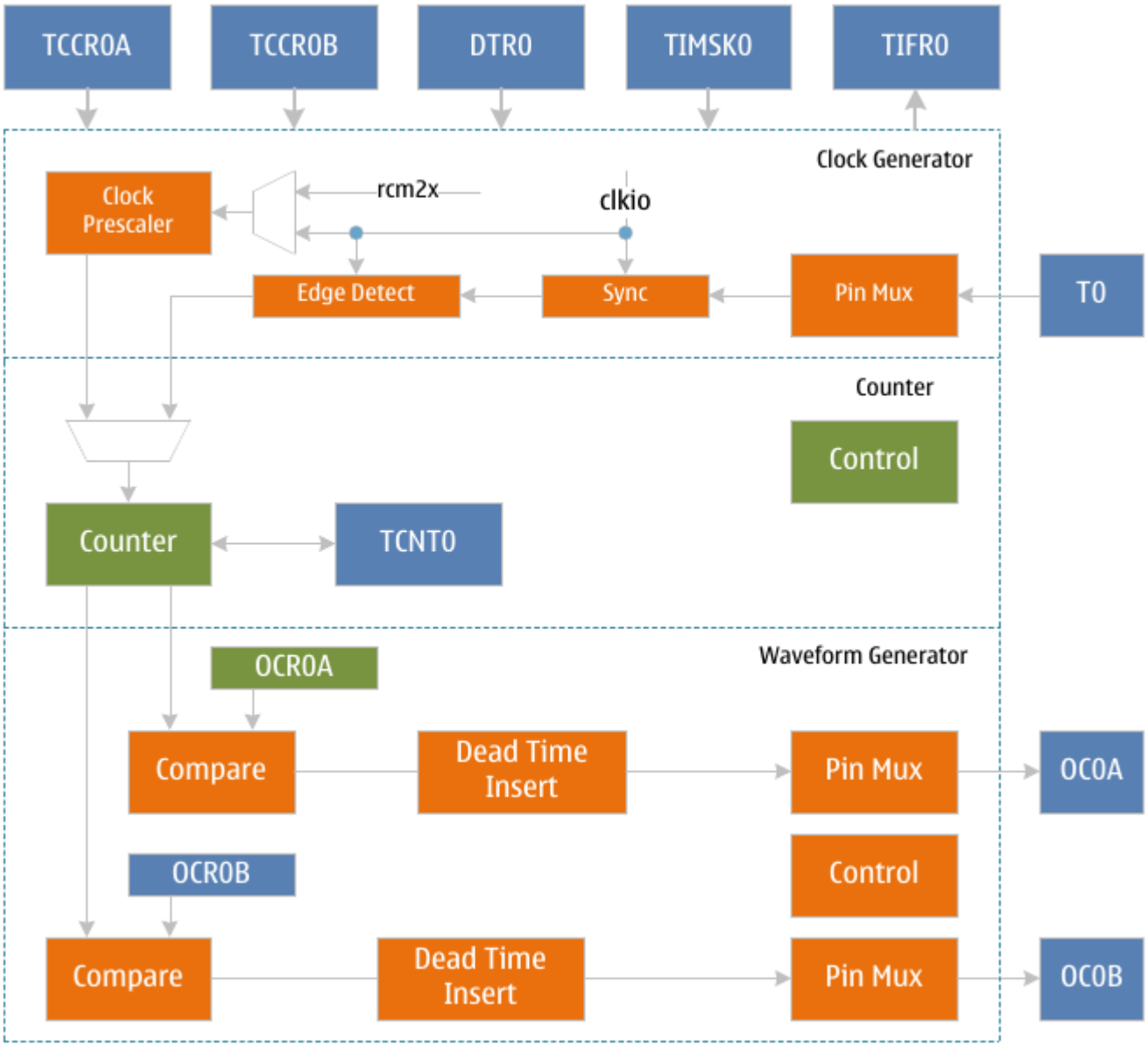
The count clock can be generated by an internal or external clock source. The clock source selection and frequency division selection are controlled by the CS0 flag located in the TCCR0B register. See the TC0 and TC1 prescaler chapters for detailed description.

The length of the counter is 8 bits and supports bidirectional counting. The waveform generation mode, that is, the working mode of the counter, is controlled by the WGM0 flag located in the TCCR0A and TCCR0B registers. According to different working modes, the counter realizes the operation of clearing, adding one or subtracting one for each count clock Clkt0. When the count overflows, the count overflow flag TOV0 in the TIFR0 register will be set. The TC0 count overflow interrupt can be generated when the interrupt is enabled.

The output compare unit pairs the count value TCNT0 and the output compare register. The values of OCR0A and OCR0B are compared, when TCNT0 is equal to OCR0A or OCR0B, it is called a compare match, and the output compare flag OCF0A or OCF0B in the TIFR0 register will be set. The TC0 output compare match interrupt can be generated when the interrupt is enabled. It should be noted that in the PWM operating mode, the OCR0A and OCR0B registers are double-buffered registers. In normal mode and CTC mode, double buffering is disabled. When the count reaches the maximum or minimum value, the value in the buffer register is synchronously updated to the compare registers OCR0A and OCR0B. See the description in the working mode chapter for details.

The waveform generator uses compare match and count overflow to generate output compare waveform signals OC0A and OC0B according to waveform generation mode control and comparison output mode settings. For the specific generation method, please refer to the description of the working mode and register chapter. To output the output comparison waveform signals OC0A and OC0B to the corresponding pins, the data direction register of this pin must also be set as output.

The following figure shows the internal structure of TC0. TC0 includes a counting clock generation unit, an 8-bit counter, 2 output comparison units and 2 waveform generation control units.




TC0 block diagram

## Operating mode

Timer 0 has four different working modes, including normal mode (Normal), clear-to-zero on compare match (CTC) mode, fast pulse width modulation (FPWM) mode and phase correction pulse width modulation (PCPWM) mode. These modes are controlled by waveform Mode control bits WGM0[2:0], and they are described in detail below. Since there are two independent output comparison units, they are represented by "A" and "B" respectively, and a lowercase "x" is used to represent these two output comparison unit channels.

### *Normal mode*

The normal mode is the simplest working mode of the timer counter. At this time, the waveform generation mode control bits WGM0[2:0]=0, and the maximum value TOP of the count is MAX (0xFF). In this mode, the counting method adds one increment to each count clock, and when the counter reaches TOP and overflows, it returns to BOTTOM and starts accumulating again. The timer

counter overflow flag TOV0 is set in the same count clock when the count value TCNT0 becomes zero. In this mode, the TOV0 flag is like the 9th count bit, but it will only be set and not cleared by the counter. The overflow interrupt service routine will automatically clear the TOV0 flag, which can be used by software to improve the resolution of the timer counter. There are no special cases to be considered in normal mode, and a new count value can be written at any time. The waveform of the output compare signal OC0x can be obtained only when the data direction register of the OC0x pin is set to output. When COM0x=1, the OC0x signal will be toggled when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{oc0xnormal} = f_{sys}/(2*N*256)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).
The output compare unit can be used to generate interrupts, but it is not recommended to use interrupts in normal mode, which will take up too much CPU time.

### *CTC mode*

When setting WGM0[2:0]=2, timer counter 0 enters CTC mode, and the maximum count TOP is OCR0A. In this mode, the counting method is incremented by one for each count clock, and the counter is cleared when the value of the counter TCNT0 is equal to TOP. OCR0A defines the maximum value of the count, that is, the resolution of the counter. This mode allows the user to easily control the frequency of the compare match output and also simplifies the operation of external event counting. When the counter reaches the maximum count, the output compare match flag OCF0 is set, and the corresponding interrupt enable is set to generate an interrupt. The OCR0A register, which is the maximum value of the count, can be updated in an interrupt service routine.

In this mode OCR0A does not use double buffering, be careful when updating the maximum value close to the minimum value when the counter is operating with no prescaler or very low prescaler. If the value written to OCR0A is less than the current TCNT0 value, the counter will lose a compare match. Before the next compare match occurs, the counter has to count to TOP, and then from BOTTOM to the OCR0A value. As in the normal mode, the count value returns to the BOTTOM count clock and sets the TOV0 flag. The waveform of the output compare signal OC0x can be obtained only when the data direction register of the OC0x pin is set to output. When COM0x=1, the OC0x signal will be toggled when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{oc0xctc} = f_{sys}/(2*N*(1+OCR0x))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).
It can be seen from the formula that when OCR0A is set to 0x0 and there is no prescaler, the output waveform with a maximum frequency of fsys/2 can be obtained.

### *Fast PWM mode*

When setting WGM0[2:0]=3 or 7, timer counter 0 enters fast PWM mode, which can be used to generate high-frequency PWM waveform. The maximum count value TOP is MAX(0xFF) or OCR0x respectively. Fast PWM mode differs from other PWM modes in that it operates in one direction. The

counter accumulates from the minimum value of 0x00 to TOP and then returns to BOTTOM to count again. When the count value TCNT0 reaches OCR0x or BOTTOM, the output compare signal OC0x will be set or cleared, depending on the setting of the compare output mode COM0x, see the register description for details. Because of the unidirectional operation, the fast PWM mode operates at twice the frequency of the phase-corrected PWM mode with bidirectional operation. High frequency characteristics make fast PWM mode suitable for power regulation, rectification and DAC applications. High-frequency signals can reduce the size of external components (inductors, capacitors, etc.), thereby reducing system cost.

When the count value reaches the maximum value, the timer counter overflow flag TOV0 will be set, and the value of the compare buffer will be updated to the compare value. If the interrupt is enabled, the compare buffer OCR0x register can be updated in the interrupt service routine. The waveform of the output compare signal OC0x can be obtained only when the data direction register of the OC0x pin is set to output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc0xfpwm} = f_{sys}/(N*(1+TOP))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).
When TCNT0 and OCR0x are compared and matched, the waveform generator will set (clear) the OC0x signal. When TCNT0 is cleared, the waveform generator will clear (set) the OC0x signal to generate PWM waves. This extreme value of OCR0x will generate a special PWM waveform. When OCR0x is set to 0x00, the output PWM is a narrow spike pulse per (1+TOP) count clock. When OCR0x is set to the maximum value, the output waveform is continuous high level or low level.

### *Phase Corrected PWM Mode*

When setting WGM0[2:0]=1 or 5, timer counter 0 enters the phase correction PWM mode, and the maximum count TOP is MAX(0xFF) or OCR0A respectively. The counter operates in two directions, incrementing from BOTTOM to TOP and then decrementing to BOTTOM, and repeating this operation until configuration changes. When the count reaches TOP and BOTTOM, the counting direction changes, and the count value will only stay on TOP or BOTTOM for one count clock cycle. During the increment or decrement process the output compare signal OC0x, depending on the setting of the compare output mode COM0x, will be cleared or set when the count value TCNT0 matches OCR0x. Compared to unidirectional operation, the maximum frequency achievable in bidirectional operation is lower, but its excellent symmetry is more suitable for motor control.

In phase correction PWM mode, the TOV0 flag is set when the count reaches BOTTOM, and the value of the compare buffer is updated to the compare value when the count reaches TOP. If the interrupt is enabled, the compare buffer OCR0x register can be updated in the interrupt service routine.

The waveform of the output compare signal OC0x can be obtained only when the data direction register of the OC0x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc0xpcpwm} = f_{sys} /(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the up-counting process, when TCNT0 matches OCR0x, the waveform generator clears (sets) the OC0x signal. During down counting, when TCNT0 matches OCR0x, the waveform generator sets (clears) the OC0x signal. Therefore, the extreme value of OCR0x will generate a special PWM wave. When OCR0x is set to the maximum or minimum value, the OC0x signal output will always remain low or high.

In order to ensure the symmetry of the output PWM wave on both sides of the minimum value, when there is no comparison match, the OC0x signal will also be flipped in two cases. The first case is when the value of OCR0x is changed from the maximum value of 0xFF to other data. When OCR0x is the maximum value and the count value reaches the maximum value, the output of OC0x is the same as the result of the comparison and match in the previous descending count, that is, keep OC0x unchanged. At this time, the comparison value will be updated to the new value of OCR0x (not 0xFF), and the value of OC0x will be kept until a comparison match occurs during the ascending count and flips. At this time, the OC0x signal is not symmetrical around the minimum value, so it is necessary to flip the OC0x signal when TCNT0 reaches the maximum value, which is the first case of flipping the OC0x signal when there is no comparison match. In the second case, when TCNT0 starts counting from a value higher than OCR0x, a compare match is lost, causing an asymmetric situation. It is also necessary to flip the OC0x signal to achieve symmetry on both sides of the minimum.

### *Automatic shutdown and restart of PWM outputs*

When the DOC0x flag of the TCCR0A register is set high, the automatic shutdown function of the PWM output will be enabled. When the trigger condition is met, the hardware will clear the corresponding COM0x flag, disconnect the PWM output signal OC0x from its output pin, and switch to General-purpose IO output to realize automatic shutdown of PWM output. At this time, the state of the output pin can be controlled by the output of the general IO port.

After the automatic shutdown of the PWM output is enabled, the trigger condition needs to be set, and the trigger source is selected by the DSX0n flag of the TCCR0C register. Trigger sources include analog comparator interrupt, external interrupt, pin level change interrupt and timer overflow interrupt. For details, please refer to the description of the TCCR0C register. When one or some trigger sources are selected as trigger conditions, at the same time when these interrupt flag bits are set, the hardware will clear the COM0x flag to close the PWM output.

When a trigger event occurs and the PWM output is turned off, the timer module has no corresponding interrupt flag to detect this change. The software needs to read the interrupt flag of the trigger source to know the trigger condition and trigger event.

When the PWM output is automatically turned off and the output needs to be restarted again, the software only needs to reset the COM0x flag to switch the OC0x signal output to the corresponding pin. It should be noted that after the automatic shutdown occurs, the timer does not stop working, and the state of the OC0x signal has been updated. The software can set the COM0x flag to output the OC0x signal after the timer overflow or compare match, so that a clear PWM output state can be obtained.

## Dead time control

When setting the DTEN0 flag to "1", the function of inserting dead time is enabled, and the output waveforms of OC0A and OC0B will insert the set dead time based on the waveform generated by the B channel comparison output. The length of this dead time is the time value corresponding to the count clock number of the DTR0 register. As shown in the figure below, the dead time insertion of OC0A and OC0B is based on the comparison output waveform of channel B. When COM0A and COM0B are both "2" or "3", the waveform polarity of OC0A is the same as that of OC0B. When COM0A and COM0B are "2" or "3" respectively, the waveform of OC0A is the same as that of OC0B. opposite polarity.

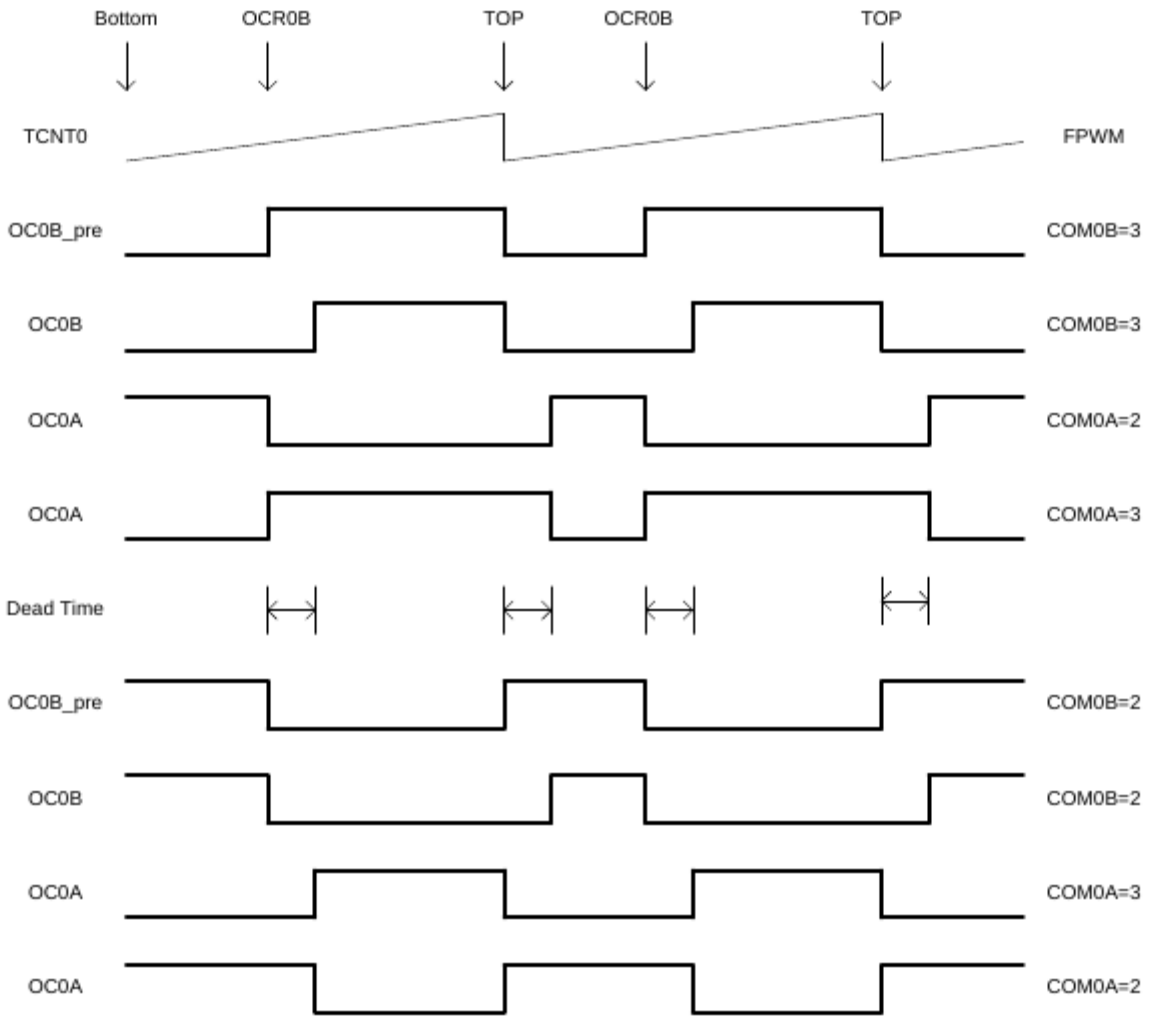



Figure 1 TC0 Dead Time Control in FPWM Mode

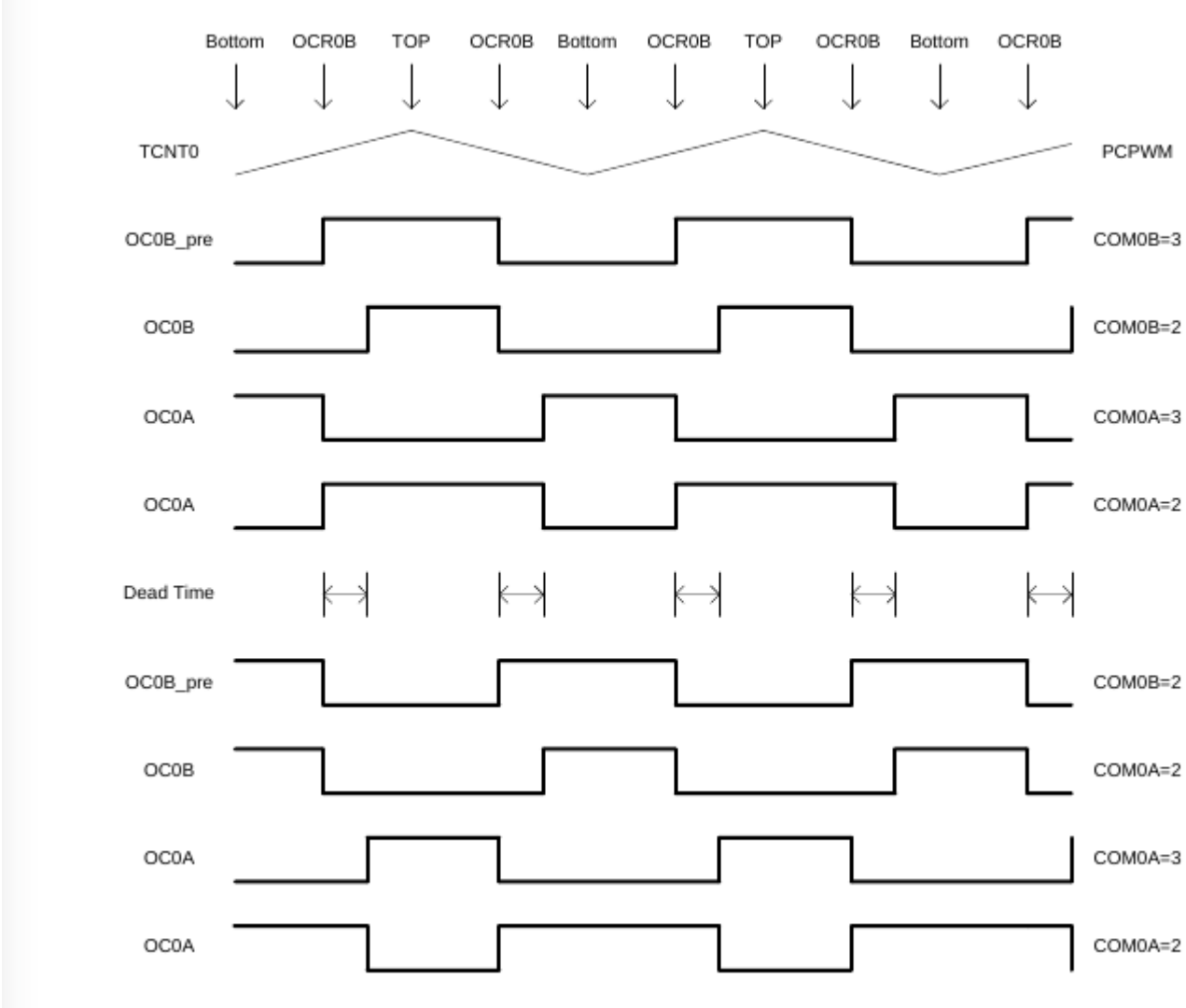



Figure 2 TC0 Dead Time Control in PCPWM Mode

When the DTEN0 flag is set to "0", the function of inserting dead time is disabled, and the output waveforms of OC0A and OC0B are the waveforms generated by their respective comparison outputs.

## High-speed clock mode

In high-speed clock mode, a higher-frequency clock is used as the counting clock source to generate PWM waveforms with higher speed and higher resolution. This high frequency clock is generated by multiplying the output clock rc32m of the internal 32M RC oscillator by 2. Therefore, before entering the high frequency mode, the frequency multiplication function of the internal 32M RC oscillator needs to be enabled.

That is, set the F2XEN flag of the TCKCSR register, and wait for a certain time until the output of the multiplied clock signal is stable. Then, the TC2XS0 flag in TCKCSR can be set to put the timer counter into high-speed clock mode.

In this mode, the system clock and the high-speed clock are asynchronous, and some registers (see the TC0 register list) work in the high-speed clock domain. Therefore, the configuration and reading of such registers are also asynchronous, and attention should be paid to the operation.

There are no special requirements for non-sequential read and write operations on registers in the high-speed clock domain, and when continuous read and write operations are performed, it is necessary to wait for a system clock. Follow the steps below:

1) Write register A;

2) wait for a system clock (NOP or register under the operating system clock);

3) Read or write register A or B.

4) Wait for a system clock (NOP or register under operating system clock).

When reading the registers in the high-speed clock domain, all registers except TCNT0 can be read directly while the counter is still counting, but the value of TCNT0 will change with the high-speed clock. To mitigate this the counter can be paused (set CS0 to zero) and then read the value of TCNT0.

## Register definition

TC0 Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| TCCR0A* | 0x44 | 0x00 | TC0 Control Register A |
| TCCR0B* | 0x45 | 0x00 | TC0 Control Register B |
| TCNT0* | 0x46 | 0x00 | TC0 count value register |
| OCR0A* | 0x47 | 0x00 | TC0 output compare register A |
| OCR0B* | 0x48 | 0x00 | TC0 output compare register B |
| DSX0* | 0x49 | 0x00 | TC0 trigger source control register |
| DTR0* | 0x4F | 0x00 | TC0 Dead Time Register |
| TIMSK0 | 0x6E | 0x00 | Timer Counter 0 Interrupt Mask Register |
| TIFR0 | 0x35 | 0x00 | Timer Counter 0 Interrupt Flag Register |
| TCKCSR | 0xEC | 0x00 | TC Clock Control and Status Register |

【Notice】

Registers with "*" work in the system clock and high-speed clock domains, and registers without "*" only work in the system clock domain.

### *TC0 Control Register A- TCCR0A*

| TCCR0A – TC0 Control Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x44 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | COM0A1 | COM0A0 | COM0B1 | COM0B0 | DOC0B | DOC0A | WGM01 | WGM00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | COM0A1 | TC0 compare match A, output mode control upper flag.<br>COM0A1 and COM0A0 together form the comparison output mode control COM0A[1:0], which is used to control the output waveform of OC0A. If both flags 1 or 2 of COM0A are set, the output compare waveform occupies the OC0A pin, but the data direction register for this pin must be set high to output this waveform. In different working modes, COM0A controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 6 | COM0A0 | TC0 compare match A, output mode control lower flag.<br>COM0A0 and COM0A1 together form the comparison output mode control COM0A[1:0], which is used to control the output waveform of OC0A. If both flags 1 or 2 of COM0A are set, the output compare waveform occupies the OC0A pin, but the data direction register for this pin must be set high to output this waveform. In different working |

| | | |
|---|---|---|
| | | modes, COM0A controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 5 | COM0B1 | TC0 compare match B, output mode control upper flag.<br>COM0B1 and COM0B0 together form the comparison output mode control COM0B[1:0], which is used to control the output waveform of OC0B. If both flags 1 or 2 of COM0B are set, the output compare waveform occupies the OC0B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM0B controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 4 | COM0B0 | TC0 compare match B, output mode control lower flag.<br>COM0B0 and COM0B1 together form the comparison output mode control COM0B[1:0], which is used to control the output waveform of OC0B. If both flags 1 or 2 of COM0B are set, the output compare waveform occupies the OC0B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM0B controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 3 | DOC0B | Enable TC0 to turn off the OCM0B enable flag.<br>When the DOC0B flag is "1", the trigger source is off and the output compare signal OC0B is enabled. When a trigger event occurs, the hardware automatically clears the COM0B flag and turns off the OC0B waveform output. Software can turn PWM output back on by setting COMB.<br>When the DOC0B flag is "0", the trigger source is off and the output compare signal OC0B is disabled. |
| 2 | DOC0A | Enable TC0 to turn off the OCM0A enable flag.<br>When the DOC0A flag is set to "1", the trigger source is off and the output compare signal OC0A is enabled. When a trigger event occurs, the hardware automatically turns off the waveform output of the OC0A.<br>When the DOC0A flag is set to "0", the trigger source is off and the output compare signal OC0A is disabled.<br>When a trigger event occurs, the waveform output of the OC0A will not be turned off. |
| 1 | WGM01 | TC0 waveform generation mode control middle flag.<br>WGM01, WGM00, WGM02 together form a waveform generation mode to control WGM0[2:0], control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |
| 0 | WGM00 | TC0 waveform generation mode control lower flag.<br>WGM00, WGM01, WGM02 together form a waveform generation mode to control WGM0[2:0], to control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |

### *TC0 Control Register B- TCCR0B*

| TCCR0B – TC0 Control Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x45 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FOC0A | FOC0B | OC0AS | DTEN0 | WGM02 | CS02 | CS01 | CS00 |
| R/W | W | W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | FOC0A | TC0 Force Output Compare A control flag.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC0A. A forced compare match will not set the OCF0A flag, nor will it reload or clear the timer, but the output pin OC0A will be updated according to the COM0A setting, as if a compare match had actually occurred.<br>The return value of reading FOC0A is always zero. |
| 6 | FOC0B | TC0 Force Output Compare B control flag.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC0B. A forced compare match will not set the OCF0B flag, nor will it reload or clear the timer, but the output pin OC0B will be updated according to the COM0B setting, as if a compare match actually occurred.<br>The return value of reading FOC0B is always zero. |
| 5 | OC0AS | OC0A Output port selection control flag. When the OC0AS flag is set to "0", the waveform of OC0A is output from pin PD6; when the OC0AS flag is set to "1", the waveform of OC0A is output from pin PE4 (valid in QFP32 package). |
| 4 | DTEN0 | TC0 dead time enable control flag.<br>Dead time insertion is enabled when the DTEN0 flag is set to '1'. Both OC0A and OC0B insert dead time on the basis of the waveform generated by the B channel comparison output, and the inserted dead time interval is determined by the count time corresponding to the DTR0 register. The polarity of the OC0A output waveform is determined by the corresponding relationship between COM0 and COM0B. For details, see the waveform polarity table after OC0A inserts dead time.<br>When the DTEN0 flag is set to "0", the dead time insertion is disabled, and the waveforms of OC0A and OC0B are the waveforms generated by their respective comparison outputs. |
| 3 | WGM02 | TC0 waveform generation mode control upper flag.<br>WGM02 and WGM00 , WGM01 together form a waveform to generate The mode controls WGM0[2:0], which controls the counting mode of the counter and the waveform generation mode. For details, see the description of the waveform generation mode table. |
| 2 | CS02 | TC0 clock select control upper flag.<br>Used to select the clock source of timer counter 0. |
| 1 | CS01 | TC0 clock select control medium flag.<br>Used to select the clock source of timer counter 0. |
| 0 | CS00 | TC0 clock select control low flag. |

| | | | |
|---|---|---|---|
| | | Used to select the clock source of timer counter 0. | |
| | | CS0[2:0] | Description |
| | | 0 | No clock source, stop counting |
| | | 1 | $clk_{sys}$ |
| | | 2 | $clk_{sys}$/8, from the prescaler |
| | | 3 | $clk_{sys}$/64, from the prescaler |
| | | 4 | $clk_{sys}$/256, from the prescaler |
| | | 5 | $clk_{sys}$/1024, from the prescaler |
| | | 6 | External clock T0 pin, falling edge trigger |
| | | 7 | External clock T0 pin, rising edge trigger |

The following table shows the control of output comparison waveform in comparison output mode in non-PWM mode (ie normal mode and CTC mode).

| COM0x[1:0] | Description |
|---|---|
| 0 | OC0x disconnected, general IO port operation |
| 1 | Toggle OC0x signal on compare match |
| 2 | Clear OC0x signal on compare match |
| 3 | Assert OC0x signal on compare match |

The following table shows the control of output compare waveform in compare output mode in fast PWM mode.

| COM0x[1:0] | Description |
|---|---|
| 0 | OC0x disconnected, general IO port operation |
| 1 | Reserve |
| 2 | Clear OC0x signal when compare match, set OC0x signal when maximum value match |
| 3 | Set OC0x signal when compare match, clear OC0x signal when maximum value match |

The following table shows the control of output comparison waveform in comparison output mode in phase correction mode.

| COM0x[1:0] | Description |
|---|---|
| 0 | OC0x disconnected, general IO port operation |
| 1 | Reserve |
| 2 | The OC0x signal is cleared when the comparison matches in the ascending order count, and the OC0x signal is set when the comparison match in the descending order count |
| 3 | The OC0x signal is set when the comparison matches in the ascending order count, and the OC0x signal is cleared when the comparison match in the descending order count |

The following table is the waveform generation mode control.

| WGM0[2:0] | Operating mode | TOP value | Update OCR0X moment | When TOV0 is set |
|---|---|---|---|---|
| 0 | Normal | 0xFF | Immediately | MAX |
| 1 | PCPWM | 0xFF | TOP | BOTTOM |
| 2 | CTC | OCR0A | Immediately | MAX |
| 3 | FPWM | 0xFF | TOP | MAX |
| 4 | Reserve | - | - | - |
| 5 | PCPWM | OCR0A | TOP | BOTTOM |
| 6 | Reserve | - | - | - |
| 7 | FPWM | OCR0A | TOP | TOP |

The following table shows the polarity control of the OC0A signal output waveform when the dead time is enabled.

Polarity Control of OC0A Signal Output Waveform in Dead Time Enable Mode

| DTEN0 | COM0A[1:0] | COM0B[1:0] | Description |
|---|---|---|---|
| 0 | - | - | OC0A signal polarity is controlled by OC0A compare output mode |
| 1 | 0 | - | OC0A disconnected, general IO port operation |
| 1 | 1 | - | Reserve |
| 1 | 2 | 2 | The OC0A signal has the same polarity as the OC0B signal |
| | | 3 | OC0A signal is opposite to OC0B signal |
| 1 | 3 | 2 | OC0A signal is opposite to OC0B signal |
| | | 3 | The OC0A signal has the same polarity as the OC0B signal |

【Note】:
The polarity of the output waveform of the OC0B signal is controlled by the OC0B comparison output mode, which is the same as the dead time mode not enabled.

### *TC0 Control Register C – TCCR0C*

| TCCR0C - TC0 Control Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x49 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DSX07 | DSX06 | DSX05 | DSX04 | - | - | DSX01 | DSX00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | DSX07 | TC0 trigger source selection control enable bit 7.<br>When the DSX07 bit is set to "1", the TC1 overflow is enabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX07 bit is set to "0", the TC1 overflow is disabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. |
| 6 | DSX06 | TC0 trigger source selection control enable bit 6.<br>When the DSX06 bit is set to "1", the TC2 overflow is enabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX06 bit is set to "0", the TC2 overflow is disabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. |
| 5 | DSX05 | TC0 trigger source selection control enable bit 5.<br>When the DSX05 bit is set to "1", the pin level changes to 0 as the trigger source of the output compare signal waveform OC0A/OC0B is enabled. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX05 bit is set to "0", the pin level changes to 0 as the trigger source of the output compare signal waveform OC0A/OC0B is disabled. |
| 4 | DSX04 | TC0 trigger source selection control enable bit 4.<br>When the DSX04 bit is set to "1", the external interrupt 0 is enabled as the trigger source for the output compare signal waveform OC0A/OC0B. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX04 bit is set to "0", the external interrupt 0 is disabled as the trigger source of the output compare signal waveform OC0A/OC0B. |
| 3:2 | - | Reserve |
| 1 | DSX01 | TC0 trigger source selection control enable bit 1.<br>When the DSX01 bit is set to "1", the analog comparator 1 is enabled as the trigger source to turn off the output compare signal waveform |

| | | |
|---|---|---|
| | | OC0A/OC0B. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX01 bit is set to "0", the analog comparator 1 is disabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. |
| 0 | DSX00 | TC0 trigger source selection control enable bit 0.<br>When the DSX00 bit is set to "1", the analog comparator 0 is enabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. When the DOC0A/DOC0B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC0A/OC0B waveform output.<br>When the DSX00 bit is set to "0", the analog comparator 0 is disabled as the trigger source to turn off the output compare signal waveform OC0A/OC0B. |

[Content of this table must be properly proofread and understood. As it stands it's bewildering to say the least, and hard to judge what is a reasonable English representation of the original Chinese text.]

The following table shows the selection control of the trigger source of the waveform output.

Disable the trigger source selection control of OC0A/OC0B waveform output

| DOC0x | DSX0n=1 | Trigger source | Description |
|---|---|---|---|
| 0 | - | - | When the DOC0x flag is "0", the trigger source is turned off and the waveform output function is disabled |
| 1 | 0 | Analog Comparator 0 | The rising edge of ACIF0 will turn off the OC0x waveform output |
| 1 | 1 | Analog Comparator 1 | The rising edge of ACIF1 will turn off the OC0x waveform output |
| 1 | 4 | External interrupt 0 | The rising edge of INTF0 will turn off the OC0x waveform output |
| 1 | 5 | Pin level change 0 | The rising edge of PCIF0 will turn off the OC0x waveform output |
| 1 | 6 | TC2 overflow | The rising edge of TOV2 will turn off the OC0x waveform output |
| 1 | 7 | TC1 overflow | The rising edge of TOV1 will turn off the OC0x waveform output |

[Note:]

DSX0n=1 means that when the $n_{th}$ bit of the DSX0 register is 1, all register bits can be set simultaneously.

### *TC0 count value register - TCNT0*

| TCNT0 –TC0 count value register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x46 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT07 | TCNT06 | TCNT05 | TCNT04 | TCNT03 | TCNT02 | TCNT01 | TCNT00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT0 | TC0 count value register.<br>The 8 bit value of the counter can be read and written directly through the TCNT0 register. A write to the TCNT0 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer has stopped. This allows initializing the TCNT0 register to match the value of OCR0 without causing an interrupt.<br>If the value written to TCNT0 equals or bypasses the OCR0 value, the compare match will be lost, resulting in incorrect waveform generation results.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT0. CPU write counters have higher priority than clearing or incrementing and decrementing operations. |

### *TC0 Output Compare Register A- OCR0A*

| OCR0A – TC0 Output Compare Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x47 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR0A7 | OCR0A6 | OCR0A5 | OCR0A4 | OCR0A3 | OCR0A2 | OCR0A1 | OCR0A0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR0A | TC0 output compare register.<br>OCR0A contains an 8-bit value that is continuously compared with the counter value TCNT0. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC0A pin.<br>When using PWM mode, the OCR0A register uses a double-buffered register. While in normal working mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR0A register with the count maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR0A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR0A itself. |

### *TC0 Output Compare Register B- OCR0B*

| OCR0B – TC0 Output Compare Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x48 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR0B7 | OCR0B6 | OCR0B5 | OCR0B4 | OCR0B3 | OCR0B2 | OCR0B1 | OCR0B0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR0B | TC0 output compare B register.<br>OCR0B contains an 8-bit value that is continuously compared with the counter value TCNT0. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC0B pin. When using PWM mode, the OCR0B register uses a double-buffered register. While in normal operating mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR0B register with the count maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR0B buffer register, and when the double buffer function is disabled, the CPU accesses the OCR0B itself. |

### *TC0 Interrupt Mask Register - TIMSK0*

| TIMSK0 – TC0 Interrupt Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x6E | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | - | OCIE0B | OCIE0A | TOIE0 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:3 | | Reserve. |
| 2 | OCIE0B | TC0 output compare B match interrupt enable flag.<br>When the OCIE0B flag is "1" and the global interrupt is set, the TC0 output compare B match interrupt is enabled. When a compare match occurs, that is, when the OCF0B flag in TIFR0 is set, an interrupt is generated.<br>When the OCIE0B flag is "0", the TC0 output compare B match interrupt is disabled. |
| 1 | OCIE0A | TC0 output compare A match interrupt enable flag.<br>When the OCIE0A flag is "1" and the global interrupt is set, the TC0 output compare A match interrupt is enabled. An interrupt is generated when a compare match occurs, that is, when the OCF0A flag in TIFR0 is set. |

| | | |
|---|---|---|
| | | When the OCIE0A flag is "0", the TC0 output compare A match interrupt is disabled. |
| 0 | TOIE0 | TC0 overflow interrupt enable flag.<br>When the TOIE0 flag is "1" and the global interrupt is set, the TC0 overflow interrupt is enabled. An interrupt is generated when TC0 overflows, i.e. the TOV0 flag in TIFR is set.<br>When the TOIE0 flag is "0", the TC0 overflow interrupt is disabled. |

### *TC0 Interrupt Flag Register - TIFR0*

| TIFR0 – TC0 Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x35 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OC0A | OC0B | - | - | - | OCF0B | OCF0A | TOV0 |
| R/W | R/O | R/O | - | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | OC0A | Output the comparison waveform signal OC0A.<br>Output comparison waveform signal OC0A, software readable but not writable. The software can read the value of the OC0A flag to obtain the polarity of the compared waveform signal to be output before the OC0A signal is not enabled to output to its corresponding IO pin, and can be changed by configuring the COM0A flag and setting the FOC0A flag. its polarity to avoid unnecessary interference pulses after enabling the OC0A signal output to its corresponding IO pin. |
| 6 | OC0B | Output the comparison waveform signal OC0B.<br>Output comparison waveform signal OC0B, readable but not writeable by software. The software can first read the value of the OC0B flag to obtain the polarity of the compared waveform signal to be output before the OC0B signal is not enabled to output to its corresponding IO pin, and can be changed by configuring the COM0B flag and setting the FOC0B flag. its polarity to avoid unnecessary interference pulses after enabling the OC0B signal output to its corresponding IO pin. |
| 5:3 | | |
| 2 | OCF0B | TC0 output compare B match flag.<br>When TCNT0 is equal to OCR0B, the comparison unit gives a match signal and sets the comparison flag OCF0B. If the output compare B interrupt enable OCIE0B is "1" and the global interrupt flag is set at this time, the output compare B interrupt will be generated. OCF0B will be cleared automatically when this interrupt service routine is executed, or it can be cleared by writing "1" to the OCF0B flag. |
| 1 | OCF0A | TC0 output compare A match flag.<br>When TCNT0 is equal to OCR0A, the comparison unit gives a match signal and sets the comparison flag OCF0A. If the output compare A interrupt enable OCIE0A is "1" and the global interrupt flag is set, the output compare A interrupt will be generated. OCF0A will be cleared automatically when this interrupt service routine is executed, or this flag |

| | | |
|---|---|---|
| | | can be cleared by writing "1" to the OCF0A flag. |
| 0 | TOV0 | TC0 overflow flag.<br>When the counter overflows, the overflow flag TOV0 is set. If the overflow interrupt enable TOIE0 is "1" and the global interrupt flag is set, an overflow interrupt will be generated. TOV0 is automatically cleared when this interrupt service routine is executed, or can be cleared by writing a '1' to the TOV0 flag. |

[Content of this table must be properly proofread and understood. As it stands it's bewildering to say the least, and hard to judge what is a reasonable English representation of the original Chinese text.]

### *DTR0 - TC0 Dead Time Control Register*

| DTR0 – TC0 Dead Time Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x4F | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTR07 | DTR06 | DTR05 | DTR04 | DTR03 | DTR02 | DTR01 | DTR00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:4 | DTR0H | TC0 Dead time register upper bits.<br>When the DTEN0 flag of the TCCR0B register is "1", OC0A and OC0B form a complementary output, the insertion dead time control is enabled, the dead time inserted on the OC0B channel is determined by DTR0H, and the length of the time is determined by DTR0H count clock's corresponding time. |
| 3:0 | DTR0L | TC0 Dead time register lower bits.<br>When the DTEN0 flag of the TCCR0B register is "1", OC0A and OC0B form a complementary output, the insertion dead time control is enabled, the dead time inserted on the OC0A channel is determined by DTR0L, and the length of the time is determined by DTR0H count clock's corresponding time. |

## *TCKCSR - TC Clock Control and Status Register*

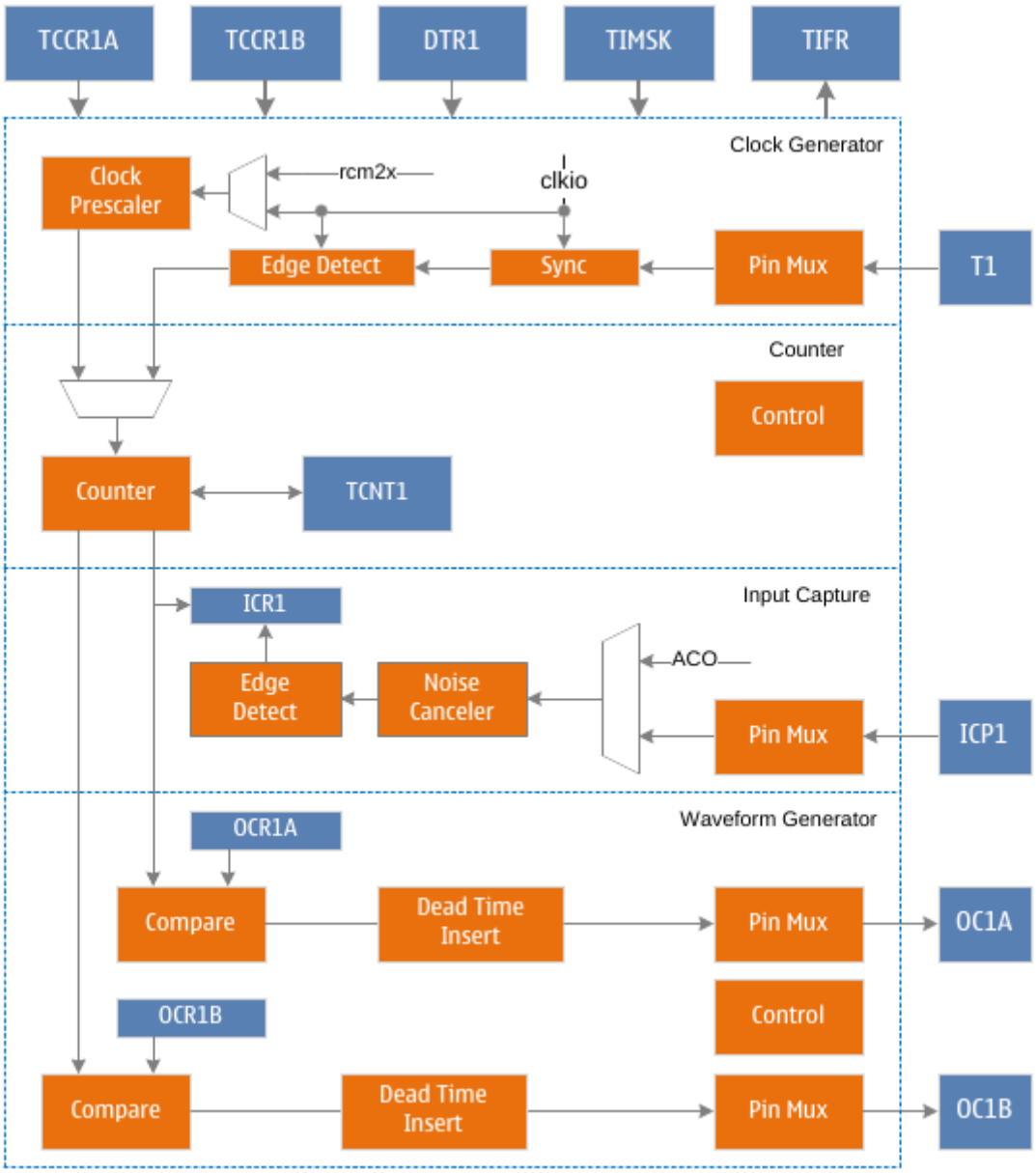
| TCKSCR – TC Clock Control and Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xEC | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | F2XEN | TC2XF1 | TC2XF0 | - | AFCKS | TC2XS1 | TC2XS0 |
| R/W | - | R/W | R | R | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | - | Reserve |
| 6 | F2XEN | RC 32M Multiplier output enable control flag.<br>When the F2XEN flag is set to "1", the multiplier output of the 32M RC oscillator is enabled to output a 64M high-speed clock.<br>When the F2XEN flag is set to "1", the multiplier output of the 32M RC oscillator is disabled, and the 64M high-speed clock cannot be output. |
| 5 | TC2XF1 | TC high-speed clock mode flag 1.<br>See timer counter 1 register description. |
| 4 | TC2XF0 | TC high-speed clock mode flag 0.<br>When the TC2XF0 flag is read as "1", it indicates that timer counter 0 works in high-speed clock mode, and when it is "0", it indicates that timer counter 0 works in system clock mode. |
| 3:2 | - | Reserve |
| 1 | TC2XS1 | TC High-speed clock mode selection control flag 1.<br>See timer counter 1 register description. |
| 0 | TC2XS0 | TC High-speed clock mode selection control flag 0.<br>When setting the TC2XS0 flag to "1", select timer counter 0 to work in high-speed clock mode.<br>When the TC2XS0 flag is set to "0", the timer counter 0 is selected to work in the system clock mode. |

# Timer/Counter 1 (TMR1)

- True 16-bit design, allowing 16-bit PWM
- 2 independent output compare units
- Double-buffered output compare register
- 1 input capture unit
- Input Capture Noise Suppressor
- Auto clear counter and auto reload on compare match
- Phase-corrected PWM without glitches
- Variable PWM period
- frequency generator
- External event counter
- 4 independent interrupt sources
- PWM with Dead Time Control
- 6 selectable trigger sources automatically shut down PWM output
- Generate high-speed high-resolution (500KHZ@7BIT) PWM in high-speed clock mode

## Overview




TC1 block diagram

TC1 is a general-purpose 16-bit timer counter module that supports PWM output and can accurately generate waveforms. TC1 includes a 16-bit counter, a waveform generation mode control unit, 2 independent output comparison units and an input capture unit. At the same time, TC1 can share a 10-bit prescaler with TC0, or can use a 10-bit prescaler independently. The prescaler divides the system clock clkio or the high-speed clock rcm2x (2 times the output clock of the internal 32M RC oscillator rc32m) to generate the count clock Clkt1. The waveform generation mode control unit controls the working mode of the counter and the generation of the comparison output waveform. According to different working modes, the counter realizes the operation of clearing, adding one or subtracting one for each count clock Clkt1. Clkt1 can be generated from an internal clock source or an external clock source. When the count value TCNT1 of the counter reaches the maximum value (equal to the maximum value 0xFFFF or fixed value or output compare register OCR1A or input capture register ICR1, defined as TOP, define the maximum value as MAX for distinction), the

counter will be cleared or decremented by one. When the count value TCNT1 of the counter reaches the minimum value (equal to 0x0000, defined as BOTTOM), the counter will increment by one. When the count value TCNT1 of the counter reaches OCR1A or OCR1B, also known as a compare match, the output compare signal OC1A or OC1B will be cleared or set to generate a PWM waveform. When the dead time insertion is enabled, the set dead time (counting clock number corresponding to the DTR1 register) will be inserted into the generated PWM waveform. When the input capture function is enabled, the counter starts or stops counting when triggered, and the ICR1 register will record the count value within the trigger period of the capture signal. Software can close the waveform output of OC1A/OC1B by clearing the COM1A/COM1B flag (set it to zero), or set the corresponding trigger source. When the trigger event occurs, the hardware will automatically clear the COM1A/COM1B flag to close the waveform output of OC1A/OC1B.

The count clock can be generated by internal or external clock source. The clock source selection and frequency division selection are controlled by the CS1 flag in the TCCR1B register. See the TC0 and TC1 prescaler chapters for detailed description.

The length of the counter is 16 bits and supports bidirectional counting. The waveform generation mode, that is, the working mode of the counter, is controlled by the WGM1 flag located in the TCCR1A and TCCR1B registers. According to different working modes, the counter implements clearing, adding one or subtracting one for each count clock Clkt1. When the count overflows, the TOV1 flag, the count overflow flag in the TIFR1 register, will be set. The TC1 count overflow interrupt can be generated when the interrupt is enabled.

The output compare unit compares the count value TCNT1 and the output compare register

The values of OCR1A and OCR1B are compared. When TCNT1 is equal to OCR1A or OCR1B, a compare match occurs, and the output compare flag OCF1A or OCF1B in the TIFR1 register will be set. A TC1 output compare match interrupt can be generated when the interrupt is enabled. It should be noted that in PWM operation mode the OCR1A and OCR1B registers are double-buffered registers. In normal mode and CTC mode, double buffering is disabled. When the count reaches the maximum or minimum value, the value in the buffer register is synchronously updated to the compare registers OCR1A and OCR1B. See the description in the working mode chapter for details.

The waveform generator uses compare match and count overflow, etc. to generate the output compare waveform signals OC1A and OC1B according to the waveform generation mode control and the comparison output mode control. For the specific generation method, please refer to the description of the working mode and register chapter. To send the output comparison waveform signals OC1A and OC1B to the corresponding pins, the data direction register of this pin must also be set as output.

## Operating mode

Timer 1 has six different working modes, including normal mode (Normal), clear on compare match (CTC) mode, fast pulse width modulation (FPWM) mode, phase correction pulse width modulation (PCPWM) mode, phase frequency correction mode Pulse Width Modulation (PFCPWM) mode, and Input Capture (ICP) mode. Required mode is selected by the waveform generation mode control bits WGM1[3:0]. The six modes are described in detail below. Since there are two independent output comparison units, they are represented by "A" and "B" respectively, and the lowercase "x" is used to represent these two output comparison unit channels.

### *Normal mode*

The normal mode is the simplest working mode of the timer counter where the waveform generation mode control flags WGM1[3:0]=0, and the maximum value TOP of the count is MAX (0xFFFF). In this mode, the counting method is one increment for each count clock. When the counter reaches TOP and overflows, it returns to BOTTOM and starts accumulating again. The timer counter overflow flag TOV1 is set in the same count clock when the count value TCNT1 becomes zero. In this mode, the TOV1 flag is like the 17th count bit, but it will only be set and not cleared. The overflow interrupt service routine will automatically clear the TOV1 flag, which can be used by software to improve the resolution of the timer counter. There are no special cases to be considered in normal mode, and a new count value can be written at any time.

The waveform of the output compare signal OC1x can be obtained only when the data direction register of the OC1x pin is set as output. When COM1x=1, the OC1x signal will be toggled when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{oc1xnormal} = f_{sys}/(2*N*65536)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

The output compare unit can be used to generate interrupts, but it is not recommended to use interrupts in normal mode, which will take too much time.

### *CTC mode*

When setting WGM1[3:0]=4 or 12, timer counter 1 enters CTC mode. When WGM1[3]=0, the maximum count value TOP is OCR1A, and when WGM1[3]=1, the maximum count value TOP is ICR1. The following takes WGM1[3:0]=4 as an example to describe the CTC mode. In this mode, the counting method is incremented by each count clock. When the counter value TCNT1 is equal to TOP, the counter is cleared. This mode allows the user to easily control the frequency of the compare match output and also simplifies the operation of external event counting.

When the counter reaches TOP, the output compare match flag OCF1 is set, and an interrupt will be generated when the corresponding interrupt enable is set. The OCR1A register can be updated in the interrupt service routine. In this mode the OCR1A does not use double buffering, be careful when updating the maximum value close to the minimum value when the counter is operating with no prescaler or very low prescaler. If the value written to OCR1A is less than the current value of TCNT1, the counter will lose a compare match. Before the next compare match occurs, the counter has to count up to MAX and then from BOTTOM to OCR1A. As in normal mode, the TOV1 flag is set in the count clock when the count value returns to 0x0.

The waveform of the output compare signal OC1x can be obtained only when the data direction register of the OC1x pin is set to output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc1xctc} = f_{sys}/(2*N*(1+OCR1A))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

It can be seen from the formula that when OCR1A is set to 0x0 and there is no prescaler, the output waveform with a maximum frequency of fsys/2 can be obtained.

When WGM1[3:0]=12, it is similar to WGM1[3:0]=4, just replace OCR1A with ICR1.

### *Fast PWM mode*

When setting WGM1[3:0]=5, 6, 7, 14 or 15, timer counter 1 enters fast PWM mode, and the maximum count TOP is 0xFF, 0x1FF, 0x3FF, ICR1 or OCR1A, which can be used to generate high-frequency PWM waveform. Fast PWM mode differs from other PWM modes in that it operates in one direction. The counter accumulates from BOTTOM to TOP and then returns to BOTTOM to count again. When the count value TCNT1 reaches TOP or BOTTOM, the output compare signal OC1x will be set or cleared, depending on the setting of the compare output mode COM1, see the register description for details. Because of the unidirectional operation, the fast PWM mode operates at twice the frequency of the phase-corrected PWM mode with bidirectional operation. The high frequency characteristics make fast PWM mode suitable for power regulation, rectification and DAC applications. High frequency signals can reduce the size of external components (inductors, capacitors, etc.), thereby reducing system cost.

When the count value reaches TOP, the timer counter overflow flag TOV1 will be set, and the value of the compare buffer will be updated to the compare value. If the interrupt is enabled, the OCR1A register can be updated in the interrupt service routine.

The waveform of the output compare signal OC1x can be obtained only when the data direction register of the OC1x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc1xfpwm} = f_{sys}/(N*(1+TOP))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

When TCNT1 and OCR1x are compared and matched, the waveform generator will set (clear) the OC1x signal. When TCNT1 is cleared, the waveform generator will clear (set) the OC1x signal, so as to generate the PWM wave. Therefore, the extreme value of OCR1x will generate a special PWM waveform. When OCR1x is set to 0x00, the output PWM is a narrow spike pulse per (1+TOP) count clock. When OCR1x is set to TOP, the output waveform is a continuous high or low level. If use OCR1A as TOP and set COM1A=1, the output compare signal OC1A will generate PWM wave with 50% duty cycle.

### *Phase Corrected PWM Mode*

When setting WGM0[3:0]=1,2,3,10 or 11, timer counter 1 enters the phase correction PWM mode, and the maximum value TOP of the count is 0xFF, 0x1FF, 0x3FF, ICR1 or OCR1A respectively. The counter operates in both directions, incrementing from BOTTOM to TOP, then decrementing to BOTTOM, and repeating this operation. When the count reaches TOP and BOTTOM, the counting direction is changed, and the count value only stays for one count clock on TOP or BOTTOM. During incrementing or decrementing, the count value

When TCNT1 matches OCR1x, the output compare signal OC1x will be cleared or set, depending on the setting of compare output mode COM1. Compared to unidirectional operation, the maximum

frequency achievable in bidirectional operation is smaller, but its excellent symmetry is more suitable for motor control.

In phase correction PWM mode, the TOV1 flag is set when the count reaches BOTTOM, and the value of the compare buffer is updated to the compare value when the count reaches TOP. If interrupts are enabled, the compare buffer OCR1x registers can be updated in the interrupt service routine.

The output compare signal OC1x waveform can be obtained only when the data direction register of the OC1x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc1xcpcpwm} = f_{sys}/(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the up-counting process, when TCNT1 matches OCR1x, the waveform generator clears (sets) the OC1x signal. During down counting, when TCNT1 matches OCR1x, the waveform generator sets (clears) the OC1x signal. Therefore, the extreme value of OCR1x will generate a special PWM wave. When OCR1x is set to TOP or BOTTOM, OC1x signal output will keep low level or high level all the time. If use OCR1A as TOP and set COM1A=1, the output compare signal OC1A will generate PWM wave with 50% duty cycle.

In order to ensure the symmetry of the output PWM wave on both sides of the BOTTOM, the OC1x signal will also be flipped in two cases when there is no comparison match. The first case is when the value of OCR1x is changed from TOP to other data. When OCR1x is TOP and the count value reaches TOP, the output of OC1x is the same as the result of the comparison match in the previous descending count, that is, keep OC1x unchanged. At this time, the comparison value will be updated to the new value of OCR1x (not TOP), and the value of OC1x will be maintained until a comparison match occurs during ascending counting and flips. At this time, the OC1x signal is not symmetrical around the minimum value, so it is necessary to flip the OC1x signal when TCNT1 reaches the maximum value, which is the first case of flipping the OC1x signal when there is no comparison match. The second case is that when TCNT1 starts counting from a value higher than OCR1x, a compare match is lost, causing an asymmetric situation. It is also necessary to flip the OC1x signal to achieve symmetry on both sides of the minimum.

### *Phase Frequency Correction PWM Mode*

When setting WGM0[3:0]=8 or 9, timer counter 1 enters the phase frequency correction PWM mode, and the maximum count value TOP is ICR1 or OCR1A respectively. The counter operates in two directions, incrementing from BOTTOM to TOP, then decrementing to BOTTOM, and repeating this operation. When the count reaches TOP and BOTTOM, the counting direction is changed, and the count value only stays for one count clock on TOP or BOTTOM. During the increment or decrement process, when the count value TCNT1 matches OCR1x, the output compare signal OC1x will be cleared or set, depending on the setting of the compare output mode COM1. Compared to unidirectional operation, the maximum achievable frequency is smaller in bidirectional operation, but its excellent symmetry is more suitable for motor control.

In the phase frequency correction PWM mode, when the count reaches BOTTOM, the TOV1 flag is set, and the value of the comparison buffer is updated to the comparison value. The time to update the comparison value is the biggest difference between the phase frequency correction PWM mode and the phase correction PWM mode. If the interrupt is enabled, the compare buffer OCR1x register can be updated in the interrupt service routine. When the CPU changes the TOP value, that is, the value of ORC1A or ICR1, it must ensure that the new TOP value is not less than the TOP value already in use, otherwise the comparison match will not happen again.

The output compare signal OC1x waveform can be obtained only when the data direction register of the OC1x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc1xcpfcpwm} = f_{sys}/(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the up-counting process, when TCNT1 matches OCR1x, the waveform generator clears (sets) the OC1x signal. During down counting, when TCNT1 matches OCR1x, the waveform generator sets (clears) the OC1x signal. Therefore, the extreme value of OCR1x will generate a special PWM wave. When OCR1x is set to TOP or BOTTOM, OC1x signal output will keep low level or high level all the time. If use OCR1A as TOP and set COM1A=1, the output compare signal OC1A will generate PWM wave with 50% duty cycle.

Because the OCR1x registers are updated at BOTTOM time, the count lengths in ascending and descending order on both sides of the TOP value are the same, resulting in a symmetrical waveform with correct frequency and phase.

When using a fixed TOP value, it is better to use the ICR1 register as the TOP value, that is, set WGM1[3:0]=8, at this time, the OCR1A register is only used to generate PWM output. If you want to generate a PWM wave with varying frequency, you must change the TOP value, and the double-buffering feature of the OCR1A will be more suitable for this application.

### *Input capture mode*

Input capture is used to capture an external event and give it a time stamp to indicate the moment when the event occurs. It can be performed in the previous count mode, but except the waveform generation mode that uses the ICR1 value as the count TOP value.

The trigger signal of external event is input by pin ICP1, which can also be realized by analog comparator unit. When the logic level on pin ICP1 changes, or the output ACO level of the analog comparator changes, and this level change is captured by the input capture unit, the input capture is triggered, and the 16-bit count value TCNT1 The data is copied to the input capture register ICR1, and the input capture flag ICF1 is set at the same time. If the ICIE1 flag is "1", the input capture flag will generate an input capture interrupt.

The input capture trigger source ICP1 or ACO is selected by setting the analog compare input capture control flag ACIC of the analog compare control and status register ACSR. It should be noted that changing the trigger source may cause an input capture, so after changing the trigger source, ICF1 must be cleared to avoid erroneous results.

The input capture signal is sent to the edge detector after passing through an optional noise suppressor. According to the configuration of the input capture selection control flag ICES1, it is checked whether the detected edge meets the trigger condition. The noise suppressor is a simple digital filter that samples the input signal 4 times, and its output is fed to the edge detector only when all 4 samples are equal. The noise suppressor is enabled or disabled by the ICNC1 flag of the TCCR1B register.

When using the input capture function, the value of the ICR1 register should be read as early as possible after ICF1 is set, because the value of ICR1 will be updated after the next capture event. It is recommended to enable the input capture interrupt, and in any input capture mode of operation, it is not recommended to change the count TOP value during operation.

The time stamps captured by the input can be used to calculate frequency, duty cycle, and other characteristics of the signal, as well as to create a log of trigger events. When measuring the duty cycle of an external signal, it is required to change the trigger edge after each capture, so the trigger signal edge must be changed as soon as possible after reading the ICR1 value.

### *Automatic shutdown and restart of PWM outputs*

When the DOC1x flag of the TCCR1C register is set to high, the automatic shutdown function of the PWM output will be enabled. When the trigger condition is met, the hardware will clear the corresponding COM1x flag, disconnect the PWM output signal OC1x from its output pin, and switch to General-purpose IO output to realize automatic shutdown of PWM output. At this time, the state of the output pin can be controlled by the output of the general IO port.

After the automatic shutdown of the PWM output is enabled, the trigger condition needs to be set, and the trigger source is selected by the DSX1n flag of the TCCR1D register. Trigger sources include analog comparator interrupt, external interrupt, pin level change interrupt and timer overflow interrupt. For details, please refer to the description of the TCCR1D register. When one or some trigger sources are selected as trigger conditions, at the same time when these interrupt flags are set, the hardware will clear the COM1x flag to turn off the PWM output.

When a trigger event occurs and the PWM output is turned off, the timer module has no corresponding interrupt flag, and the software needs to know the trigger condition and trigger event by reading the interrupt flag of the trigger source.

When the PWM output is automatically turned off and the output needs to be restarted again, the software only needs to reset the COM1x flag to switch the OC1x signal output to the corresponding pin. It should be noted that after the automatic shutdown occurs, the timer does not stop working, and the state of the OC1x signal is also constantly updated. The software can set the COM1x flag to output the OC1x signal after the timer overflow or compare match, so that a clear PWM output state can be obtained.

## Dead time control

When the DTEN1 flag is set to "1", the function of inserting dead time is enabled, and the output waveforms of OC1A and OC1B will insert the set dead time based on the waveform generated by the B channel comparison output. The length of the time is DTR1 The time value corresponding to the count clock number of the register. As shown in the figure below, the dead-time insertion of both OC1A and OC1B is based on the comparison output waveform of channel B. When COM1A and COM1B are both "2" or "3", the waveform polarity of OC1A is the same as that of OC1B. When COM1A and COM1B are "2" or "3" respectively, the waveform of OC1A is the same as that of OC1B. opposite polarity.

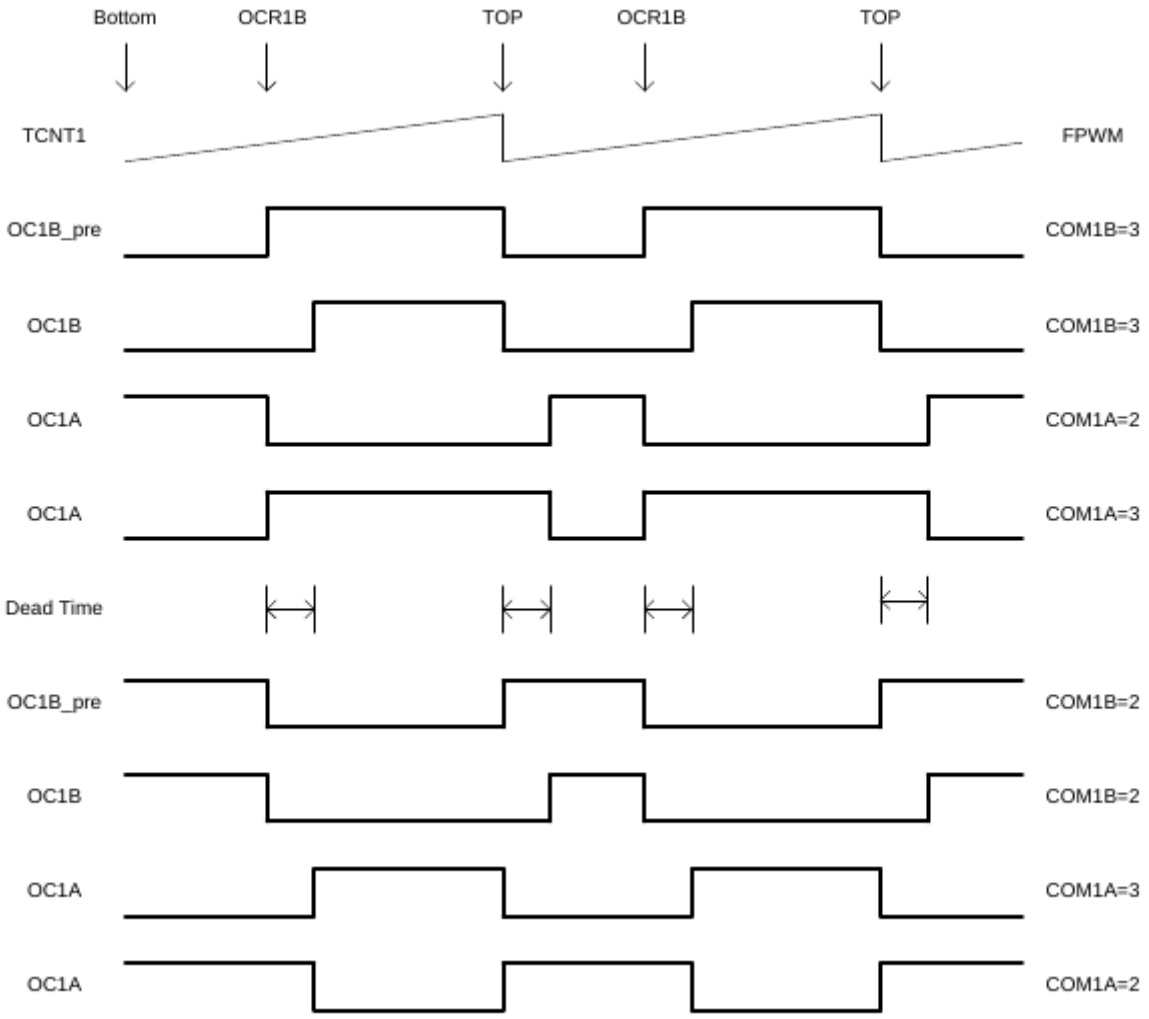



Figure 3 TC1 Dead Time Control in FPWM Mode

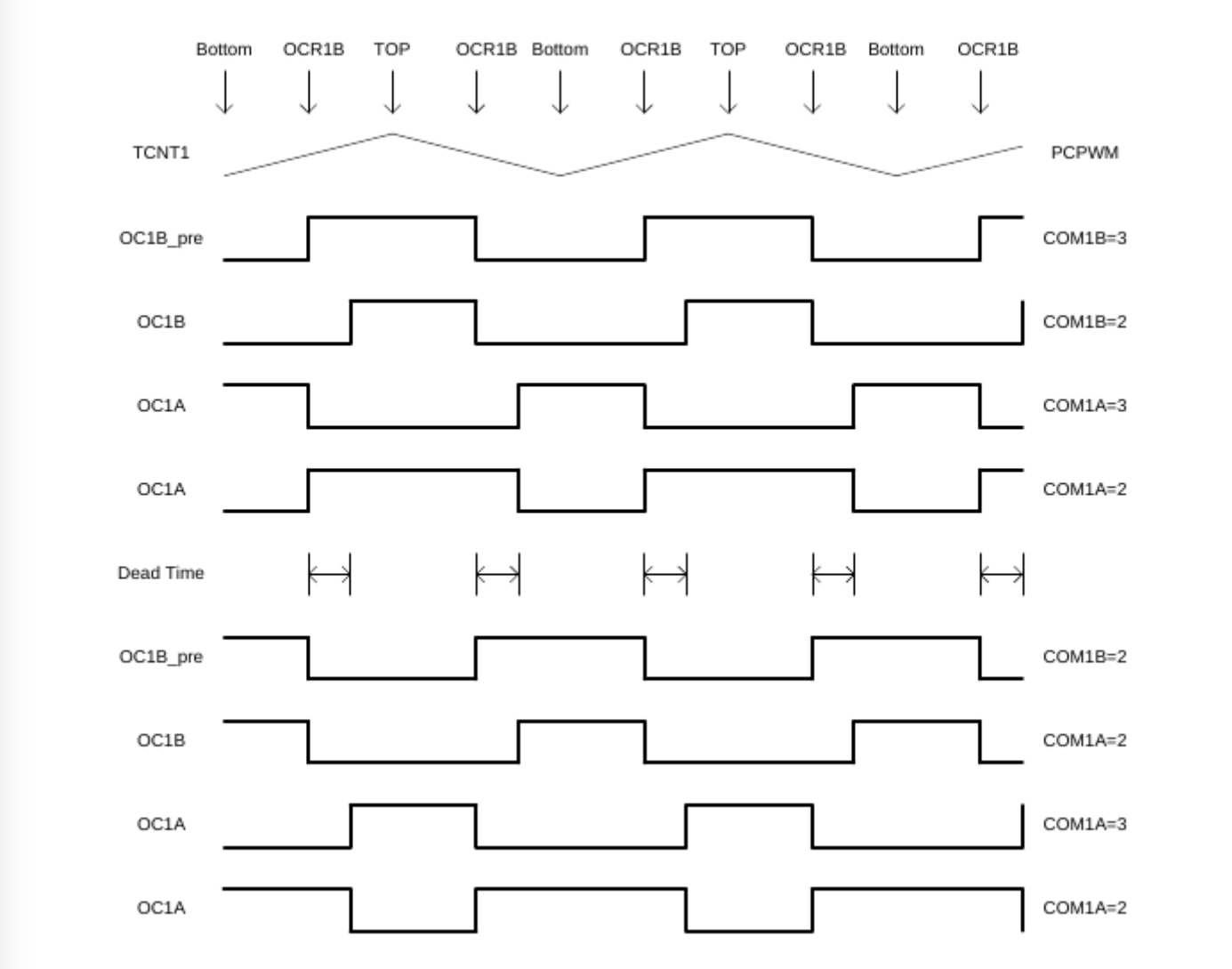



Figure 4 TC1 Dead Time Control in PCPWM Mode

When the DTEN1 flag is set to "0", the function of inserting dead time is disabled, and the output waveforms of OC1A and OC1B are the waveforms generated by their respective comparison outputs.

## High-speed counting mode

In high-speed clock mode, a higher-frequency clock is used as the counting clock source to generate PWM waveforms with higher speed and higher resolution. This high frequency clock is generated by multiplying the output clock rc32m of the internal 32M RC oscillator by 2. Therefore, before entering the high frequency mode, it is necessary to enable the frequency multiplication function of the internal 32M RC oscillator, that is, set the F2XEN flag of the TCKCSR register, and wait for a certain time until the output of the multiplied clock signal is stable. Then, TCKCSR can be set the TC2XS1 flag to enable the timer counter to enter high-speed clock mode.

In this mode, the system clock is asynchronous with the high-speed clock, and some registers (see the TC1 register list) work in the high-speed clock domain. Therefore, the configuration and reading of such registers are also asynchronous, and attention should be paid to the operation.

There are no special requirements for non-sequential read and write operations on registers in the high-speed clock domain, and when continuous read and write operations are performed, a system clock needs to be waited, and the following steps can be followed:

5) Write register A;

6) Wait for a system clock (NOP or register under the operating system clock);

7) Read or write register A or B.

8) Wait for a system clock (NOP or register under operating system clock).

When reading the registers in the high-speed clock domain, the registers with a width of 8 bits can be directly read, and when reading the value of the 16-bit register (OCR1A, OCR1B, ICR1, TCNT1), read the value of the lower register first. , After waiting for a system clock, read the value of the high-order register, and when reading the value of TCNT1, when the counter is still counting, the value of TCNT1 will change with the high-speed clock, and the counter can be suspended (set CS1 to zero) Then read the value of TCNT1.

When reading OCR1A, OCR1B and ICR1, follow these steps:

1) Read OCR1AL/OCR1BL/ICR1L;

2) Wait for a system clock (NOP);

3) Read OCR1AH/OCR1BH/ICR1H.

When reading TCNT1, follow the steps below:

1) Set CS1 to zero;

2) Wait for a system clock (NOP);

3) Read the value of TCNT1L;

4) Wait for a system clock (NOP);

Read the value of TCNT1H.

## Register definition

TC1 Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| TCCR1A* | 0x80 | 0x00 | TC1 Control Register A |
| TCCR1B* | 0x81 | 0x00 | TC1 Control Register B |
| TCCR1C* | 0x82 | 0x00 | TC1 Control Register C |
| DSX1 | 0x83 | 0x00 | TC1 trigger source control register |
| TCNT1L* | 0x84 | 0x00 | TC1 count value register low byte |
| TCNT1H* | 0x85 | 0x00 | TC1 count value register high byte |
| ICR1L* | 0x86 | 0x00 | TC1 input capture register low byte |
| ICR1H* | 0x87 | 0x00 | TC1 input capture register high byte |
| OCR1AL* | 0x88 | 0x00 | TC1 output compare register A low byte |
| OCR1AH* | 0x89 | 0x00 | TC1 output compare register A high byte |
| OCR1BL* | 0x8A | 0x00 | TC1 output compare register B low byte |
| OCR1BH* | 0x8B | 0x00 | TC1 output compare register B high byte |
| DTR1* | 0x8C | 0x00 | TC1 Dead Time Control Register |
| TIMSK1 | 0x6F | 0x00 | Timer Counter Interrupt Mask Register |
| TIFR1 | 0x36 | 0x00 | Timer Counter Interrupt Flag Register |
| TCKCSR1 | 0xEC | 0x00 | TC1 Clock Control Status Register |

【Notice】

Registers with "*" work in the system clock and high-speed clock domains, and registers without "*" only work in the system clock domain.

### *TCCR1A – TC1 Control Register A*

| TCCR1A – TC1 Control Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x80 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | COM1A1 | COM1A0 | COM1B1 | COM1B0 | - | - | WGM11 | WGM10 |
| R/W | R/W | R/W | R/W | R/W | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | COM1A1 | Compare match output A mode control, upper flag.<br>COM1A1 and COM1A0 form COM1A[1:0] to control the output compare waveform OC1A. If both bits 1 or 2 of COM1A are set, the output compare waveform occupies the OC1A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM1A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |

| | | |
|---|---|---|
| 6 | COM1A0 | Compare match output A mode control lower flag.<br>COM1A1 and COM1A0 form COM1A[1:0] to control the output compare waveform OC1A. If both bits 1 or 2 of COM1A are set, the output compare waveform occupies the OC1A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM1A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 5 | COM1B1 | Compare match output B mode control upper flag.<br>COM1B1 and COM1B0 form COM1B[1:0] to control the output compare waveform OC1B. If both bits 1 or 2 of COM1B are set, the output compare waveform occupies the OC1B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM1B controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 4 | COM1B0 | Compare match output B mode control lower flag.<br>COM1B1 and COM1B0 form COM1B[1:0] to control the output compare waveform OC1B. If both bits 1 or 2 of COM1B are set, the output compare waveform occupies the OC1B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM1B controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 3:2 | - | Leave unchanged |
| 1 | WGM11 | The waveform generation mode control, flag 2.<br>WGM11 and WGM13, WGM12, WGM10 together form the waveform generation mode control WGM1[3:0], which controls the counting mode of the counter and the waveform generation mode. For details, see the description of the waveform generation mode table. |
| 0 | WGM10 | The waveform generation mode control, flag 1 (LSB).<br>WGM10 and WGM13, WGM12, WGM11 together form the waveform generation mode control WGM1[3:0], control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |

The following table shows the control of output comparison waveform in comparison output mode in non-PWM mode (ie normal mode and CTC mode).

| COM1x[1:0] | Description |
|---|---|
| 0 | OC1x disconnected, general IO port operation |
| 1 | Toggle OC1x signal on compare match |
| 2 | Clear OC1x signal on compare match |
| 3 | Assert OC1x signal on compare match |

The following table shows the control of output compare waveform in compare output mode in fast PWM mode.

| COM1x[1:0] | Description |
|---|---|
| 0 | OC1x disconnected, general IO port operation |
| 1 | When WGM1 is 15: OC1A signal is toggled when compare match, OC1B is disconnected<br>When WGM1 is other value: OC1x is disconnected, general IO port operation |
| 2 | Clear OC1x signal when compare match, set OC1x signal when maximum value match |
| 3 | Set OC1x signal when compare match, clear OC1x signal when maximum value match |

The following table shows the control of output comparison waveform in comparison output mode in phase correction mode.

| COM1x[1:0] | Description |
|---|---|
| 0 | OC1x disconnected, general IO port operation |
| 1 | When WGM1 is 9 or 11: OC1A signal is toggled when compare match, OC1B is disconnected<br>When WGM1 is other value: OC1x is disconnected, general IO port operation |
| 2 | The OC1x signal is cleared by the comparison match in the ascending sequence count, and the OC1x signal is set by the compare match in the descending sequence count |
| 3 | The OC1x signal is set by the comparison match in the ascending order count, and the OC1x signal is cleared by the comparison match in the descending order count. |

### *TCCR1B – TC1 Control Register B*

| TCCR1B – TC1 Control Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x81 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICNC1 | ICES1 | - | WGM13 | WGM12 | CS12 | CS11 | CS10 |
| R/W | R/W | R/W | - | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | ICNC1 | Input capture noise suppressor enable control flag.<br>When the ICNC1 flag is set to "1", the input capture noise suppressor is enabled. At this time, the input of the external pin ICP1 is filtered, and the input signal is valid only when the consecutive 4 sample values are equal. This function makes the input capture delayed by 4 clock cycles.<br>When the ICNC1 flag is set to "0", the input capture noise suppressor is disabled, and the input of the external pin ICP1 is directly valid. |
| 6 | ICES1 | Input capture trigger edge select control flag.<br>When the ICES1 flag is set to "1", the rising edge of the selected level |

<table>
<tr><td></td><td></td><td colspan="2">triggers the input capture; when the ICES1 flag is set to "0", the falling edge of the selected level triggers the input capture.<br>When an event is captured, the value of the counter is copied to the ICR1 register and the input capture flag ICF1 is set at the same time. If the interrupt is enabled, an input capture interrupt is generated.</td></tr>
<tr><td>5</td><td>-</td><td colspan="2">Reserve.</td></tr>
<tr><td>4</td><td>WGM13</td><td colspan="2">Waveform generation mode control, flag 4 (MSB).<br>WGM13 and WGM12, WGM11, WGM10 together form the waveform generation mode control WGM1[3:0], which controls the counting mode of the counter and the waveform generation mode. For details, see the waveform generation mode table description.</td></tr>
<tr><td>3</td><td>WGM12</td><td colspan="2">The waveform generation mode control, flag 3.<br>WGM12 and WGM13, WGM11, WGM10 together form the waveform generation mode control WGM1[3:0], which controls the counting mode of the counter and the waveform generation mode. For details, see the waveform generation mode table description.</td></tr>
<tr><td>2</td><td>CS12</td><td colspan="2">Clock select control upper flag. Used to select the clock source of timer counter 1.</td></tr>
<tr><td>1</td><td>CS11</td><td colspan="2">Clock select control middle flag. Used to select the clock source of timer counter 1.</td></tr>
<tr><td rowspan="10">0</td><td rowspan="10">CS10</td><td colspan="2">Clock Select Control lower flag.<br>Used to select the clock source of timer counter 1.</td></tr>
<tr><td>CS1[2:0]</td><td>Description</td></tr>
<tr><td>0</td><td>No clock source, stop counting</td></tr>
<tr><td>1</td><td>$clk_{sys}$</td></tr>
<tr><td>2</td><td>$clk_{sys}$/8, from the prescaler</td></tr>
<tr><td>3</td><td>$clk_{sys}$/64, from the prescaler</td></tr>
<tr><td>4</td><td>$clk_{sys}$/512, from the prescaler</td></tr>
<tr><td>5</td><td>$clk_{sys}$/1024, from the prescaler</td></tr>
<tr><td>6</td><td>External clock T1 pin, falling edge trigger</td></tr>
<tr><td>7</td><td>External clock T1 pin, rising edge trigger</td></tr>
</table>

The following table is the waveform generation mode control.

| WGM1[3:0] | Operating mode | TOP value | Update OCR0 moment | When TOV0 is set |
|---|---|---|---|---|
| 0 | Normal | 0xFFFF | Immediately | MAX |
| 1 | 8-bit PCPWM | 0x00FF | TOP | BOTTOM |
| 2 | 9-bit PCPWM | 0x01FF | TOP | BOTTOM |
| 3 | 10-bit PCPWM | 0x03FF | TOP | BOTTOM |
| 4 | CTC | OCR1A | Immediately | MAX |
| 5 | 8-bit PCPWM | 0x00FF | BOTTOM | TOP |
| 6 | 9-bit PCPWM | 0x01FF | BOTTOM | TOP |
| 7 | 10-bit PCPWM | 0x03FF | BOTTOM | TOP |
| 8 | PFCPWM | ICR1 | BOTTOM | BOTTOM |
| 9 | PFCPWM | OCR1A | BOTTOM | BOTTOM |
| 10 | PCPWM | ICR1 | TOP | BOTTOM |
| 11 | PCPWM | OCR1A | TOP | BOTTOM |
| 12 | CTC | ICR1 | Immediately | MAX |
| 13 | Reserve | - | - | - |
| 14 | FPWM | ICR1 | TOP | TOP |
| 15 | FPWM | OCR1A | TOP | TOP |

### *TCCR1C – TC1 Control Register C*

| TCCR1C – TC1 Control Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x82 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FOC1A | FOC1B | DOC1B | DOC1A | DTEN1 | - | - | - |
| R/W | W | W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | FOC1A | Force output compare A.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC1A. A forced compare match will not set the OCF1A flag, nor will it reload or clear the timer, but the output pin OC1A will be updated according to the COM1A setting, as if a real compare match had occurred.<br>When operating in PWM mode, it must be cleared when writing to the TCCR1A register.<br>The return value of reading FOC1A is always zero. |
| 6 | FOC1B | Force output compare B.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC1B. A forced compare match will not set the OCF1B flag, nor will it reload or clear |

| | | |
|---|---|---|
| | | the timer, but the output pin OC1B will be updated according to the COM1B setting, as if a real compare match had occurred. When working in PWM mode, write to TCCR1A register to clear it. The return value of reading FOC1B is always zero. |
| 5 | DOC1B | TC1 turns off the output compare enable control high flag.<br>When the DOC1B flag is set to "1", the trigger source is off and the output compare signal OC1B is enabled. When a trigger event occurs, the hardware automatically clears the COM1B flag and turns off the OC1B waveform output. Software can turn the PWM output back on by setting COM1B.<br>When the DOC1B flag is set to "0", the trigger source is off and the output compare signal OC1B is disabled. |
| 4 | DOC1A | TC1 turns off the output compare enable control low flag.<br>When the DOC1A flag is set to "1", the trigger source is off and the output compare signal OC1A is enabled. When a trigger event occurs, the hardware automatically clears the COM1A flag and turns off the OC1A waveform output. Software can turn the PWM output back on by setting COM1A.<br>When the DOC1A flag is set to "0", the trigger source is off and the output compare signal OC1A is disabled. |
| 3 | DTEN1 | TC1 Dead time enable control flag.<br>Dead time insertion is enabled when the DTEN1 flag is set to '1'. Both OC1A and OC1B insert dead time on the basis of the waveform generated by the B channel comparison output, and the inserted dead time interval is determined by the count time corresponding to the DTR1 register. The polarity of the OC1A output waveform is determined by the corresponding relationship between COM1A and COM1B. For details, see the waveform polarity table after OC1A inserts dead time.<br>When the DTEN1 flag is set to "0", the dead time insertion is disabled, and the waveforms of OC1A and OC1B are the waveforms generated by their respective comparison outputs. |
| 2:0 | - | Reserve |

The following table shows the polarity control of the OC1A signal output waveform when dead time is enabled.

Polarity Control of OC1A Signal Output Waveform in Dead Time Enable Mode

| DTEN1 | COM1A[1:0] | COM1B[1:0] | Description |
|---|---|---|---|
| 0 | - | - | OC1A signal polarity is controlled by OC1A compare output mode |
| 1 | 0 | - | OC1A disconnected, general IO port operation |
| 1 | 1 | - | Reserve |
| 1 | 2 | 2 | The OC1A signal has the same polarity as the OC1B signal |
| | | 3 | The OC1A signal has the opposite polarity to the OC1B signal |
| 1 | 3 | 2 | The OC1A signal has the opposite polarity to the OC1B signal |
| | | 3 | The OC1A signal has the same polarity as the OC1B signal |

【Notice】:

The polarity of the OC1B signal output waveform is controlled by the OC1B compare output mode, which is the same as when the dead time mode is not enabled.

### *TCCR1D – TC1 Control Register D*

| TCCR1D – TC Control Register D | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x83 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DSX17 | DSX16 | DSX15 | DSX14 | - | - | DSX11 | DSX10 |
| R/W | R/W | R/W | R/W | R/W | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | DSX17 | TC1 trigger source selection control enable flag 7.<br>When the DSX17 flag is set to "1", the TC0 overflow is enabled as the trigger source to turn off the output compare signal waveform OC1A/OC1B. When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX17 flag is set to "0", the TC0 overflow is used as the trigger source for the output compare signal waveform OC1A/OC1B to be disabled. |
| 6 | DSX16 | TC1 trigger source selection control enable flag 6.<br>When the DSX16 flag is set to "1", the TC2 overflow is enabled as the trigger source to turn off the output compare signal waveform |

| | | |
|---|---|---|
| | | OC1A/OC1B. When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX16 flag is set to "0", the TC2 overflow is disabled as the trigger source for the output compare signal waveform OC1A/OC1B. |
| 5 | DSX15 | TC1 trigger source selection control enable flag 5.<br>When the DSX15 flag is set to "1", the pin level change of 1 is enabled as the trigger source of the output compare signal waveform OC1A/OC1B.<br>When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX15 flag is set to "0", the pin level change of 1 is disabled as the trigger source of the output compare signal waveform OC1A/OC1B. |
| 4 | DSX14 | TC1 trigger source selection control enable flag 4.<br>When the DSX14 flag is set to "1", the external interrupt 1 is enabled as the trigger source to turn off the output compare signal waveform OC1A/OC1B. When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX14 flag is set to "0", the external interrupt 1 is disabled as the trigger source of the output compare signal waveform OC1A/OC1B. |
| 3:2 | - | Reserve |
| 1 | DSX11 | TC1 trigger source selection control enable flag 1.<br>When the DSX11 flag is set to "1", the analog comparator 1 is enabled as the trigger source to turn off the output compare waveform OC1A/OC1B. When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX11 flag is set to "0", the analog comparator 1 is disabled as the trigger source for the output compare signal waveform OC1A/OC1B. |
| 0 | DSX10 | TC1 trigger source selection control enable flag 0.<br>When the DSX10 flag is set to "1", the analog comparator 0 is enabled as the trigger source to turn off the output compare waveform OC1A/OC1B. When the DOC1A/DOC1B flag is "1", the rising edge of the interrupt flag of the selected trigger source will automatically turn off the OC1A/OC1B waveform output.<br>When the DSX10 flag is set to "0", analog comparator 0 is disabled as the trigger source for the output compare waveform OC1A/OC1B. |

The following table shows the selection control of the trigger source of the waveform output.

Disable the trigger source selection control of OC1A/OC1B waveform output

| DOC1x | DSX1n=1 | Trigger source | Description |
|---|---|---|---|
| 0 | - | - | When the DOC1x flag is "0", the trigger source is turned off and the waveform output function is disabled |
| 1 | 0 | Analog Comparator 0 | A rising edge of ACIF0 will turn off the OC1x waveform output |
| 1 | 1 | Analog Comparator 1 | A rising edge on ACIF1 will turn off the OC1x waveform output |
| 1 | 4 | External Interrupt 1 | A rising edge on INTF1 will turn off the OC1x waveform output |
| 1 | 5 | Pin level change 1 | A rising edge on PCIF1 will turn off the OC1x waveform output |
| 1 | 6 | TC2 overflow | A rising edge on TOV2 will turn off the OC1x waveform output |
| 1 | 7 | TC0 overflow | A rising edge on TOV0 will turn off the OC1x waveform output |

【Notice】:

DSX1n=1 means that when the nth bit of the DSX1 register is 1, all register bits can be set at the same time.

### *TCNT1L – TC1 count value register low byte*

| TCNT1L – TC1 count value register low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x84 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT1L7 | TCNT1L6 | TCNT1L5 | TCNT1L4 | TCNT1L3 | TCNT1L2 | TCNT1L1 | TCNT1L0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT1 | Low byte of TC1 count value.<br>TCNT1H and TCNT1L combine to form TCNT1 through<br>The TCNT1 register can directly read and write the 16-bit count value of the counter. Reading and writing a 16-bit register requires two operations. When writing 16-bit TCNT1, TCNT1H should be written first. When reading 16-bit TCNT1, TCNT1L should be read first.<br>A write to the TCNT1 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer has stopped. This allows initializing the TCNT1 register to match the value of OCR1x without causing an interrupt. If the value written to TCNT1 equals or bypasses the OCR1x value, the compare match will be lost, resulting in incorrect waveform generation results.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT1.<br>CPU write counters have higher priority than clearing or incrementing and decrementing operations. |

### *TCNT1H – TC1 count value register high byte*

| TCNT1H – TC1 count value register high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x85 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT1H7 | TCNT1H6 | TCNT1H5 | TCNT1H4 | TCNT1H3 | TCNT1H2 | TCNT1H1 | TCNT1H0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT1H | High byte of TC1 count value.<br>TCNT1H and TCNT1L are combined together to form TCNT1, and the 16-bit count value of the counter can be read and written directly through the --TCNT1 register. Reading and writing a 16-bit register requires two operations. When writing 16-bit TCNT1, TCNT1H should be written first. When reading 16-bit TCNT1, TCNT1L should be read first.<br>A write to the TCNT1 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer has stopped. This allows initializing the TCNT1 register to match the value of OCR1x without causing an interrupt.<br>If the value written to TCNT1 equals or bypasses the OCR1x value, the compare match will be lost, resulting in incorrect waveform generation |

| | | |
|---|---|---|
| | | results.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT1.<br>CPU write counters have higher priority than clearing or incrementing and decrementing operations. |

### *ICR1L – TC1 Input Capture Register Low Byte*

| ICR1L – TC1 Input Capture Register Low Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x86 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICR1L7 | ICR1L6 | ICR1L5 | ICR1L4 | ICR1L3 | ICR1L2 | ICR1L1 | ICR1L0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit. | Name | Description | | | | | | |
| 7:0 | ICR1L | TC1 Input low byte of captured value.<br>ICR1H and ICR1L combine to form the 16-bit ICR1. Reading and writing a 16-bit register requires two operations. When writing 16-bit ICR1, ICR1H should be written first. When reading 16-bit ICR1, ICR1L should be read first. When the input capture is triggered, the count value TCNT1 will be updated and copied to the ICR1 register. The ICR1 register can also be used to define the TOP value of the count. | | | | | | |

### *ICR1H – TC1 Input Capture Register High Byte*

| ICR1H – TC1 Input Capture Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x87 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICR1H7 | ICR1H6 | ICR1H5 | ICR1H4 | ICR1H3 | ICR1H2 | ICR1H1 | ICR1H0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | Name | Description | | | | | | |
| 7:0 | ICR1H | TC1 Input high byte of captured value.<br>ICR1H and ICR1L combine to form the 16-bit ICR1. Reading and writing a 16-bit register requires two operations. When writing 16-bit ICR1, ICR1H should be written first. When reading 16-bit ICR1, ICR1L should be read first. When the input capture is triggered, the count value TCNT1 will be updated and copied to the ICR1 register. The ICR1 register can also be used to define the TOP value of the count. | | | | | | |

## *OCR1AL – TC1 output compare register A low byte*

| OCR1AL – TC1 output compare register A low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x88 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR1AL7 | OCR1AL6 | OCR1AL5 | OCR1AL4 | OCR1AL3 | OCR1AL2 | OCR1AL1 | OCR1AL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR1AL | Output compare register A low byte.<br>OCR1AL and OCR1AH combine to form the 16-bit OCR1A. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR1A, OCR1AH should be written first. When reading 16-bit OCR1A, OCR1AL should be read first.<br>OCR1A is continuously compared with the counter value TCNT1. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC1A pin.<br>When using PWM mode, the OCR1A register is a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR1A register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating the interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR1A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR1A itself. |

## *OCR1AH – TC1 output compare register A high byte*

| OCR1AH – TC1 output compare register A high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x89 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR1AH7 | OCR1AH6 | OCR1AH5 | OCR1AH4 | OCR1AH3 | OCR1AH2 | OCR1AH1 | OCR1AH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR1AH | Output compare register A high byte.<br>OCR1AL and OCR1AH combine to form the 16-bit OCR1A. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR1A, OCR1AH should be written first. When reading 16-bit OCR1A, OCR1AL should be read first.<br>OCR1A is continuously compared with the counter value TCNT1. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC1A pin.<br>When using PWM mode, the OCR1A register is a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR1A register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating the interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR1A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR1A itself. |

## *OCR1BL – TC1 output compare register B low byte*

| OCR1BL – TC1 output compare register B low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x8A | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR1BL7 | OCR1BL6 | OCR1BL5 | OCR1BL4 | OCR1BL3 | OCR1BL2 | OCR1BL1 | OCR1BL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR1BL | Output compare register B low byte.<br>OCR1BL and OCR1BH combine to form the 16-bit OCR1B. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR1B, OCR1BH should be written first. When reading 16-bit OCR1B, OCR1BL should be read first.<br>OCR1B is continuously compared with the counter value TCNT1. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC1B pin.<br>When using PWM mode, the OCR1B register is a double-buffered register. While in normal working mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR1B register with the count maximum or minimum time, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffering function is used, the CPU accesses the OCR1B buffer register, and when the double buffering function is disabled, the CPU accesses the OCR1B itself. |

## *OCR1BH – TC1 output compare register B high byte*

| OCR1BH – TC1 output compare register B high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x8B | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR1BH7 | OCR1BH6 | OCR1BH5 | OCR1BH4 | OCR1BH3 | OCR1BH2 | OCR1BH1 | OCR1BH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR1BH | Output compare register B high byte.<br>OCR1BL and OCR1BH combine to form the 16-bit OCR1B. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR1B, OCR1BH should be written first. When reading 16-bit OCR1B, OCR1BL should be read first.<br>OCR1B is continuously compared with the counter value TCNT1. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC1B pin.<br>When using PWM mode, the OCR1B register is a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR1B register with the count maximum or minimum time, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR1B buffer register, and when the double buffer function is disabled, the CPU accesses the OCR1B itself. |

## *TIMSK1 – TC1 Interrupt Mask Register*

| TIMSK1 – TC1 Interrupt Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x6F | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | TICIE1 | - | - | OCIE1A | OCIE1B | TOIE1 |
| R/W | - | - | R/W | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | - | Reserve |
| 5 | TICIE1 | TC1 Input capture interrupt enable control flag.<br>When the ICIE1 flag is "1" and the global interrupt is set, the TC1 input capture interrupt is enabled. When the input capture is triggered, that is, the ICF1 flag of TIFR1 is set, the interrupt occurs.<br>When the ICIE1 flag is "0", the TC1 input capture interrupt is disabled. |
| 4:3 | - | Reserve |
| 2 | OCIE1A | TC1 output compare B match interrupt enable flag.<br>When the OCIE1B flag is "1" and the global interrupt is set, the TC1 output compare B match interrupt is enabled. When a compare match occurs, that is, when the OCF1B flag in TIFR is set, an interrupt is generated.<br>When the OCIE1B flag is "0", the TC1 output compare B match interrupt is disabled. |
| 1 | OCIE1B | TC1 output compare A match interrupt enable flag.<br>When the OCIE1A flag is "1" and the global interrupt is set, the TC1 output compare A match interrupt is enabled. When a compare match occurs, that is, when the OCF1A flag in TIFR is set, an interrupt is generated.<br>When the OCIE1A flag is "0", the TC1 output compare A match interrupt is disabled. |
| 0 | TOIE1 | TC1 overflow interrupt enable flag.<br>When the TOIE1 flag is "1" and the global interrupt is set, the TC1 overflow interrupt is enabled. An interrupt is generated when TC1 overflows, i.e. the TOV1 flag in TIFR is set.<br>When the TOIE1 flag is "0", the TC1 overflow interrupt is disabled. |

### *TIFR1 – TC1 Interrupt Flag Register*

| TIFR1 – TC1 Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x36 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | ICF1 | - | - | OCF1B | OCF1A | TOV1 |
| R/W | - | - | R/W | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | - | Reserve |
| 5 | ICF1 | Input capture flag flag.<br>When an input capture event occurs, the ICF1 flag is set. When ICR1 is used as the TOP value of the count, and the count value reaches the TOP value, the ICF1 flag is set. If ICIE1 is "1" and the global interrupt flag is set, an input capture interrupt will be generated. ICF1 will be cleared automatically when this interrupt service routine is executed, or it can be cleared by writing “1” to the ICF1 flag. |
| 4:3 | - | Reserve |
| 2 | OCF1B | Output compare B match flag.<br>When TCNT1 is equal to OCR1B, the comparison unit gives a match signal and sets the comparison flag OCF1B. If the output compare interrupt enable OCIE1B is "1" and the global interrupt flag is set, an output compare interrupt will be generated. OCF1B will be cleared automatically when this interrupt service routine is executed, or it can be cleared by writing "1" to the OCF1B flag. |
| 1 | OCF1A | Output compare A match flag.<br>When TCNT1 is equal to OCR1A, the comparison unit gives a match signal and sets the comparison flag OCF1A. If the output compare interrupt enable OCIE1A is "1" and the global interrupt flag is set, an output compare interrupt will be generated. OCF1A will be cleared automatically when this interrupt service routine is executed, or it can be cleared by writing "1" to the OCF1A flag. |
| 0 | TOV1 | Overflow flag.<br>When the counter overflows, the overflow flag TOV1 is set. If the overflow interrupt enable TOIE1 is "1" and the global interrupt flag is set, an overflow interrupt will be generated. TOV1 will be cleared automatically when this interrupt service routine is executed, or can be cleared by writing a '1' to the TOV1 flag. |

### *DTR1L – TC1 Dead Time Register Low Byte*

| DTR1–TC1 Dead Time Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x8C | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTR1L | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | DTR1L | Dead time register high byte.<br>When the DTEN1 flag is high, OC1A and OC1B are complementary outputs, and the dead time inserted on the OC1A output is determined by DTR1L count clocks. |

### *DTR1H – TC1 Dead Time Register High Byte*

| DTR1H–TC1 Dead Time Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x8D | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTR1H | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | DTR1H | Dead time register high byte.<br>When the DTEN1 flag is high, OC1A and OC1B are complementary outputs, and the dead time inserted on the OC1B output is determined by DTR1H count clocks. |

### *TCKCSR – TC Clock Control Status Register*

| TCKCSR–TC Clock Control Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xEC | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | F2XEN | TC2XF1 | TC2XF0 | - | AFCKS | TC2XS1 | TC2XS0 |
| R/W | - | R/W | R/O | R/O | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | - | Reserve |
| 6 | F2XEN | RC 32M multiplier output enable control flag<br>When the F2XEN flag is set to "1", the multiplier output of the 32M RC oscillator is enabled, and the high-speed clock of 64M is output. When the F2XEN flag is set to "1", the multiplier output of the 32M RC oscillator is disabled and cannot be output. 64M high-speed clock |
| 5 | TC2XF1 | TC high-speed clock mode flag 1<br>When the TC2XF1 flag is read as "1", it indicates that timer counter 1 |

| | | |
|---|---|---|
| | | works in high-speed clock mode, and when it is "0", it indicates that timer counter 1 works in system clock mode |
| 4 | TC2XF0 | TC high-speed clock mode flag 0, refer to timer counter 0 register description |
| 3:2 | - | Reserve |
| 1 | TC2XS1 | TC high-speed clock mode selection control flag 1<br>When the TC2XS1 flag is set to "1", the timer counter 1 is selected to work in the high-speed clock mode. When the TC2XS1 flag is set to "0", the timer counter 1 is selected to work in the system clock mode. |
| 0 | TC2XS0 | TC high-speed clock mode selection control flag 0, refer to timer counter 0 register description |

# TMR0/1/3 Prescaler

- Three 10-bit prescalers
- TC0, TC1 and TC3 multiplexed prescaler CPS310 in multiplexed mode
- In standalone mode, TC0 uses exclusive prescaler CPS310, TC1 uses exclusive prescaler CPS1, TC3 uses exclusive prescaler CPS3
- Support software reset

## Overview

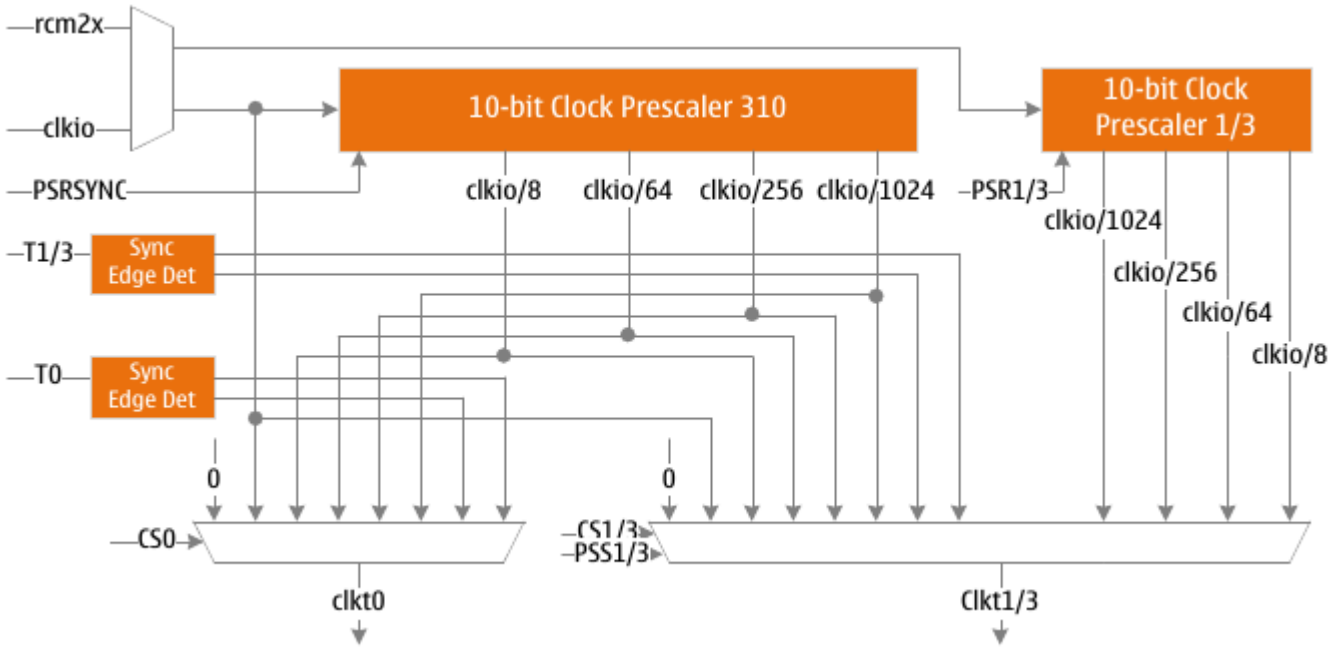
In multiplexed mode (PSS1=0 and PSS3=0), TC0, TC1 and TC3 share a 10-bit prescaler CPS310, but they have different divider settings.

In single-use mode (PSS1=1 and PSS3=0), TC1 independently uses a 10-bit prescaler CPS1, TC0 and TC3 share a 10-bit prescaler CPS310, but they have different frequency division settings.

In single-use mode (PSS1=0 and PSS3=1), TC3 independently uses a 10-bit prescaler CPS3, TC0 and TC1 share a 10-bit prescaler CPS310, but they have different frequency division settings.

In independent mode (PSS1=1 and PSS3=1), TC0 independently uses a 10-bit prescaler CPS310, TC1 independently uses a 10-bit prescaler CPS1, and TC3 independently uses a prescaler CPS3.

The following descriptions are used for TC0, TC1 and TC3, where n represents 0, 1 or 3.




TC0/TC1 /TC3 Prescaler Block Diagram

## Internal clock source

When setting CSn[2:0]=1, timer 3 can only be driven by the system clock clkio, and timer counter 0 or 1 can be directly driven by the system clock clkio or high-speed clock rcm2x (2 times the output clock of the internal 32M RC oscillator) drive. The prescaler can output 4 different clock frequencies, namely clkio/8, clkio/64, clkio/256 and clkio/1024.

## Divider reset

### *Multiplexing mode*

When the PSS1 flag is set to "0" and the PSS3 flag is set to "0", TC0, TC1 and TC3 share a prescaler CPS310.

The prescaler operates independently, its operation is independent of TC's clock selection logic, and it is shared by TC0, TC1, and TC3. Since it is not affected by clock selection control, the state of the prescaler has an impact on the application of the divided clock. The effect occurs when the timer is enabled and the output of the prescaler is selected as the count clock source (6>CSn[2:0]>1). It may take 1 to N+1 system clocks from timer enable to first count, where N is the prescale factor (8, 64, 256 or 1024).

It is possible to synchronize timer and program operation by resetting the prescaler. However, it must be noted that if another timer is using this prescaler, resetting the prescaler will affect all timers connected to it.

### *Single-use mode*

When the PSS1 flag is set to "1", TC1 uses the prescaler CPS1 independently, and the reset of the prescaler is controlled by the PSR1 flag. The respective resets work independently and do not affect other prescalers.

When the PSS3 flag is set to "1", the TC3 uses the prescaler CPS3 independently, and the reset of the prescaler is controlled by the PSR3 flag. The respective resets work independently and do not affect other prescalers.

When the PSS1 flag is set to "1" and the PSS3 flag is set to "1"; TC0 independently uses the prescaler CPS310, the reset of the prescaler is controlled by the PSRSYNC flag, TC1 independently uses the prescaler CPS1 and TC3 independently uses prescaler CPS3. The respective resets work independently and will not affect other prescalers.

## External clock source

An external clock source provided by the T0/T1/T3 pins can be used as the count clock source. The signal on the T0/T1/T3 pin is used as the clock source for the counter after the synchronization logic and edge detector. Each rising edge (Sn[2:0]=7) or falling edge (CSn[2:0]=6) generates a count pulse. External clock sources are not fed into the prescaler.

Due to the presence of synchronization and edge detection circuits on the pins, the level change on T0/T1/T3 requires a delay of 2.5 to 3.5 system clocks for the counter to update.

Disabling or enabling clock input must be done after T0/T1/T3 remains stable for at least one system clock cycle, otherwise it may generate wrong count clock pulses.

To ensure correct sampling, the external clock pulse width must be greater than one system clock period, and the external clock frequency must be less than half the system clock frequency when the duty cycle is 50%. Due to the difference in system clock frequency and duty cycle caused by the error of the oscillator itself, it is recommended that the maximum frequency of the external clock should not be greater than fsys/2.5.

## Register definition

### *GTCCR – General Purpose Timer Counter Control Register*

| GTCCR – General Purpose Timer Counter Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x43 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TSM | - | - | - | - | - | PSRASY | PSRSYNC |
| R/W | R/W | - | - | - | - | - | W | W |

| Bit | Name | Description |
|---|---|---|
| 7 | TSM | Timer counter synchronization mode control flag.<br>When the TSM flag is set to "1", the timer counter is in synchronous mode. In Synchronous mode, the values written to the PSRASY and PSRSYNC flags are held, keeping the corresponding prescalers always reset.<br>This ensures that the corresponding timer counters are terminated and initiated with the same value.<br>When the TSM flag is set to "0", the value of the PSRASY flag and the PSRSYNC flag will be cleared by hardware, and the timer counter will start to work at the same time. |
| 6:2 | - | Reserve |
| 1 | PSRASY | See Timer TC2 register description. |
| 0 | PSRSYNC | Prescaler CPS310 reset control flag.<br>When the PSRSYNC flag is set to "1", the prescaler CPS310 will be reset. When the TSM flag is not set, the hardware will clear the PSRSYNC flag after a reset.<br>When the PSRSYNC flag is set to "0", the setting has no effect.<br>In multiplexed mode, TC0/TC1/TC3 share the prescaler, and reset will affect these three timers.<br>In standalone mode, reset only affects TC0.<br>Reading this flag will always return a "0". |

## *PSSR – Prescaler Select Register*

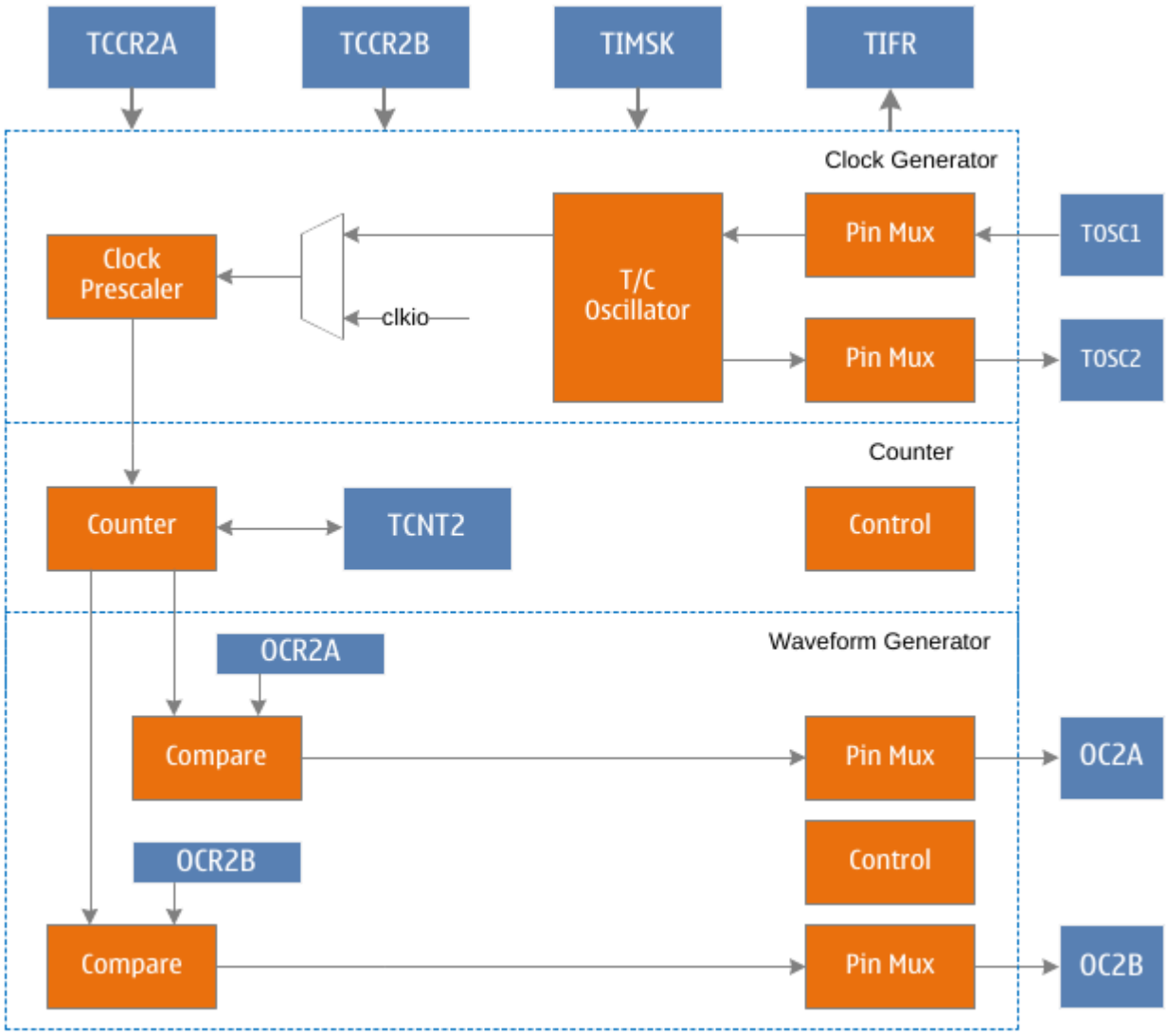
| PSSR – Prescaler Select Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xE2 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PSS1 | PSS3 | - | - | - | - | PSR3 | PSR1 |
| R/W | R/W | R/W | - | - | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PSS1 | Prescaler selection control flag.<br>When the PSS1 flag is set to "1", TC1 uses the prescaler CPS1 alone.<br>When the PSS1 flag is set to "0", it is the prescaler multiplexing mode. TC0 and TC1 share the prescaler CPS310. The prescaler CPS1 is inactive and will always be reset. If the PSS3 flag is "0" at the same time, TC3, TC0, TC1 share the prescaler CPS310. Prescalers CPS1 and CPS3 are inactive and will always be reset. |
| 6 | PSS3 | Prescaler selection control flag.<br>When the PSS3 flag is set to "1", TC3 uses the prescaler CPS3 alone.<br>When the PSS3 flag is set to "0", it is the prescaler multiplexing mode. TC0 and TC3 share the prescaler CPS310. The prescaler CPS3 is inactive and will always be reset. If the PSS1 flag is "0" at the same time, TC1, TC0, TC3 share the prescaler CPS310. Prescalers CPS1 and CPS3 are inactive and will always be reset. |
| 5:2 | - | Reserve |
| 1 | PSR3 | Prescaler CPS3 reset control flag. The PSR3 flag is only valid in TC3 single-use mode.<br>When the PSR3 flag is set to "1", the prescaler CPS3 will be reset. Hardware will clear the PSR3 flag after a reset. When the PSR3 flag is set to "0", the setting is invalid.<br>Reading this flag will always return a "0". |
| 0 | PSR1 | Prescaler CPS1 reset control flag. The PSR1 flag is only valid in TC1 single-use mode.<br>When the PSR1 flag is set to "1", the prescaler CPS1 will be reset. Hardware will clear the PSR1 flag after reset. When the PSR1 flag is set to "0", the setting is invalid.<br>Reading this flag will always return a "0". |

# Timer/Counter 2 (TMR2)

- 8-bit counter
- Two independent comparison units
- Auto clear counter and auto reload on compare match
- Phase-corrected PWM output without glitches
- frequency generator
- External event counter
- 10-bit clock prescaler
- Overflow and Compare Match Interrupts
- Allows the use of an external 32.768KHz RTC crystal to count

## Overview




TC2 block diagram

C2 is a general-purpose 8-bit timer counter module that supports PWM output and can accurately generate waveforms. TC2 contains an 8-bit counter, waveform generation mode control unit and 2 output comparison units. Waveform generation Mode control unit controls the working mode of the counter and the generation of the comparison output waveform. According to different working modes, the counter implements the operation of clearing, adding one or subtracting one for each count clock Clkt2. Clkt2 can be generated from an internal clock source or an external clock source. When counting with an external 32.768KHz crystal oscillator, TC2 can be used as an RTC counter. When the count value TCNT2 of the counter reaches the maximum value (equal to the maximum value 0xFF or the output compare register OCR2A, which is defined as TOP, and the maximum value is defined as MAX for difference), the counter will be cleared or decremented by one operation. When the count value TCNT2 of the counter reaches the minimum value (equal to 0x00, defined as BOTTOM), the counter will increment by one. When the count value TCNT2 of the counter reaches OCR2A/OCR2B, which is also called a compare match, the output compare signal OC2A/OCR2B will be cleared or set to generate a PWM waveform.

## Operating mode

Timer 2 has four different working modes, including normal mode (Normal), clear on compare match (CTC) mode, fast pulse width modulation (FPWM) mode and phase correction pulse width modulation (PCPWM) mode. Mode control bits WGM2[2:0] to select. These four modes are described in detail below. Since there are two independent output comparison units, they are represented by "A" and "B" respectively, and a lowercase "x" is used to represent these two output comparison unit channels.

## Normal mode

The normal mode is the simplest working mode of the timer counter. At this time, the waveform generation mode control flags WGM2[2:0]=0, and the maximum value TOP of the count is MAX (0xFF). In this mode, the counting method adds one increment to each count clock, and when the counter reaches TOP and overflows, it returns to BOTTOM and starts accumulating again. The timer counter overflow flag TOV2 is set in the same count clock when the count value TCNT2 becomes zero. In this mode, the TOV2 flag is like the 9th count bit, but it will only be set and not cleared. The overflow interrupt service routine will automatically clear the TOV2 flag, which can be used by software to improve the resolution of the timer counter. There are no special cases to be considered in normal mode, and a new count value can be written at any time.

The waveform of the output compare signal OC2x can be obtained only when the data direction register of the OC2x pin is set to output. When COM2x=1, the OC2x signal will be toggled when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{oc2xnormal} = f_{sys}/(2*N*256)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

The output compare unit can be used to generate interrupts, but it is not recommended to use interrupts in normal mode, which will take up too much CPU time.

## CTC mode

When setting WGM2[2:0]=2, timer counter 2 enters CTC mode, and the maximum count TOP is OCR2A. In this mode, the counting method is incremented by one for each count clock, and the counter is cleared when the value of the counter TCNT2 is equal to TOP. OCR2A defines the maximum count, which is the resolution of the counter. This mode allows the user to easily control the frequency of the compare match output and also simplifies the operation of external event counting.

When the counter reaches the maximum count value, the output compare match flag OCF2 is set, and an interrupt will be generated when the corresponding interrupt enable is set. In the interrupt service routine, the OCR2A register, which is the maximum value of the count, can be updated. In this mode the OCR2A does not use double buffering, be careful when updating the maximum value close to the minimum value when the counter is operating with no prescaler or very low prescaler. If the value written to OCR2A is less than the current TCNT2 value, the counter will lose a compare match. Before the next compare match occurs, the counter has to count to TOP and then from BOTTOM to the OCR2A value. As in normal mode, the count value returns to the BOTTOM count clock and sets the TOV2 flag.

The waveform of the output compare signal OC2x can be obtained only when the data direction register of the OC2x pin is set to output. When COM2x=1, the OC2x signal will be toggled when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{oc2xctc} = f_{sys}/(2*N*(1+OCR2A))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024). It can be seen from the formula that when OCR2x is set to 0x0 and there is no prescaler, the output waveform with a maximum frequency of fsys/2 can be obtained.

## Fast PWM mode

When setting WGM2[2:0]=3 or 7, timer counter 2 enters fast PWM mode, which can be used to generate high-frequency PWM waveform. The maximum count TOP is MAX(0xFF) or OCR2A respectively. Fast PWM mode differs from other PWM modes in that it operates in one direction. The counter accumulates from the minimum value of 0x00 to TOP and then returns to BOTTOM to count again. When the count value TCNT2 reaches OCR2x or BOTTOM, the output compare signal OC2x will be set or cleared, depending on the setting of the compare output mode COM2x, see the register description for details. Because of the unidirectional operation, the fast PWM mode operates at twice the frequency of the phase-corrected PWM mode with bidirectional operation. High frequency characteristics make fast PWM mode suitable for power regulation, rectification and DAC applications. High-frequency signals can reduce the size of external components (inductors, capacitors, etc.), thereby reducing system cost.

When the count value reaches the maximum value, the timer counter overflow flag TOV2 will be set, and the value of the compare buffer will be updated to the compare value. If the interrupt is enabled, the compare buffer OCR2x registers can be updated in the interrupt service routine.

The waveform of the output compare signal OC2x can be obtained only when the data direction register of the OC2x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc2xfpwm} = f_{sys}/(N*(1+TOP))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

When TCNT2 and OCR2x are compared and matched, the waveform generator will set (clear) the OC2x signal, and when TCNT2 is cleared, the waveform generator will clear (set) the OC2x signal to generate PWM waves. Therefore, the extreme value of OCR2x will generate a special PWM waveform. When OCR2x is set to 0x00, the output PWM is a narrow spike in every (1+TOP) count clock. When OCR2x is set to the maximum value, the output waveform is a continuous high or low level.

## Phase Correction PWM Mode

When setting WGM2[2:0]=1 or 5, timer counter 2 enters the phase correction PWM mode, and the maximum count TOP is MAX(0xFF) or OCR2A respectively. The counter operates in two directions, incrementing from BOTTOM to TOP, then decrementing to BOTTOM, and repeating this operation. When the count reaches TOP and BOTTOM, the counting direction will be changed, and the count value will only stay on TOP or BOTTOM for one count clock. During the increment or decrement process, when the count value TCNT2 matches OCR2x, the output compare signal OC2x will be cleared or set, depending on the setting of the compare output mode COM2x. Compared to unidirectional operation, the maximum frequency achievable in bidirectional operation is lower, but its excellent symmetry is more suitable for motor control.

In phase correction PWM mode, the TOV2 flag is set when the count reaches BOTTOM, and the value of the compare buffer is updated to the compare value when the count reaches TOP. If the interrupt is enabled, the compare buffer OCR2x registers can be updated in the interrupt service routine.

The waveform of the output compare signal OC2x can be obtained only when the data direction register of the OC2x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{oc2xpcpwm} = f_{sys} /(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the counting up process, when TCNT2 matches OCR2x, the waveform generator clears (sets) the OC2x signal. During down counting, when TCNT2 matches OCR2x, the waveform generator sets (clears) the OC2x signal. This extreme value of OCR2x will generate a special PWM wave. When the OCR2x is set to the maximum or minimum value, the OC2x signal output will always remain low or high.

In order to ensure the symmetry of the output PWM wave on both sides of the minimum value, the OC2x signal will also be flipped in two cases when no compare match occurs. The first case is when the value of OCR2x is changed from the maximum value of 0xFF to other data. When OCR2x is the maximum value and the count value reaches the maximum, the output of OC2x is the same as the

result of the comparison and matching in the previous descending count, that is, keep OC2x unchanged. At this time, the comparison value will be updated to the new value of OCR2x (not 0xFF), and the value of OC2x will be kept until a comparison match occurs during the ascending count and flips. At this time, the OC2x signal is not symmetrical around the minimum value, so it is necessary to flip the OC2x signal when TCNT2 reaches the maximum value, which is the first case of flipping the OC2x signal when there is no comparison match. In the second case, when TCNT2 starts counting from a value higher than OCR2x, a compare match is lost, causing an asymmetric situation. It is also necessary to flip the OC2x signal to achieve symmetry on both sides of the minimum.

## Asynchronous operation of TC2

- When the AS2 flag in the ASSR register is "1", TC2 works in asynchronous mode, and the clock source of the counter comes from the oscillator of the external timer counter. The following points should be considered for the operation of TC2 in asynchronous mode. Transitions between synchronous and asynchronous modes may cause corruption of TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B data. The safe operation steps are as follows:
  - 1. Clear the OCIE2A, TOIE2 and OCIE2B registers to disable the TC2 interrupt;
  - 2. Set the AS2 flag to select the appropriate clock source;
  - 3. Write new data to TCNT2, OCR2A, TCCR2A, OCR2B and TCCR2B registers;
  - 4. When switching to asynchronous mode, wait for the TCN2UB, OCR2AUB, TCR2AUB, OCR2BUB and TCR2BUB flags to be cleared;
  - 5. Clear the interrupt flag of TC2;
  - 6. Enable the desired interrupt.
- The oscillator is best to use a 32.768KHz watch crystal. The system clock frequency must be more than 4 times higher than the crystal frequency.
- When the CPU writes TCNT2, OCR2A, TCCR2A, OCR2B and TCCR2B, the hardware will put the data into the scratchpad first, and then latch into the corresponding register after the rising edge of the two TOSC1 clocks. A new data write cannot be performed until the data is latched from the scratchpad to the destination register. Each register has its own independent scratchpad, so writing to TCNT2 will not interfere with writing to OCR2. The asynchronous status register ASSR is used to check whether data has been written to the destination register.
- If TC2 is used as the wake-up condition of the MCU sleep mode, it cannot enter the sleep mode until the end of each register update, otherwise the MCU may enter the sleep mode before the TC2 setting takes effect, so that the TC2 cannot wake up the system.
- If TC2 is used as the wake-up condition for MCU sleep mode, attention must be paid to the process of re-entering sleep mode. The interrupt logic requires one TOSC1 clock cycle to reset. If the time from wake-up to re-entering sleep is less than one TOSC1 clock cycle, the interrupt will no longer occur and the device will not wake up. The following method of operation is recommended:

    - 1. Write appropriate data to each register;
    - 2. Wait for the corresponding update busy flag of ASSR to clear;
    - 3. Enter sleep mode.
- If asynchronous operation mode is selected, the oscillator of TC2 will continue to run until power-down mode is entered. The user must be aware that the stabilization time of this oscillator may be as long as 1 second, therefore, it is recommended that the user wait at least 1 second after enabling the oscillator of the TC2 before using the asynchronous operation mode of the TC2.
- The wake-up process in sleep mode in asynchronous working mode: after the interrupt condition is met, the wake-up process is started at the next timer clock. That is, the counter accumulates at least one more clock before the processor can read the value of the counter. After waking up, the MCU executes the interrupt service routine, and then starts to execute the program after the SLEEP statement.
- Reading the value of TCNT2 shortly after waking up from Sleep may return incorrect data. Because TCNT2 is driven by the asynchronous TOSC1 clock, reading TCNT2 must be done through a register that is synchronized to the internal system clock, which occurs on every rising edge of TOSC1. After waking up from sleep mode, the system clock is reactivated, and the value of TCNT2 read is the value before entering sleep mode, and it will not be updated until the next TOSC1 rising edge. The phase of TOSC1 when waking up from sleep mode is completely unpredictable and depends on the wake-up time. Therefore, the recommended sequence for reading TCNT2 values is:
    - 1. Write an arbitrary value to OCR2A or TCCR2A;
    - 2. Wait for the corresponding update busy flag to be cleared;
    - 3. Read TCNT2.
- In asynchronous mode, the synchronization of interrupt flags requires 3 system clock cycles plus 1 timer cycle. The counter has accumulated at least one more clock before the MCU can read the value of the counter that caused the interrupt flag to be set. Changes in the output compare signal are synchronized with the timer clock, not the system clock.

## Prescaler for TC2

The input clock of the TC2 prescaler is called clkt2s, and the AS2 flag in the ASSR register selects the internal system clock clkio or the external TOSC1 clock source, which is connected to the system clock clkio by default. If AS2 is set, TC2 will be driven asynchronously by TOSC1. When TOSC1 pin and TOSC2 pin are connected with a 32.768KHz watch crystal oscillator, TC2 can be used as RTC counter. Applying an external clock signal directly on the TOSC1 pin is not recommended.

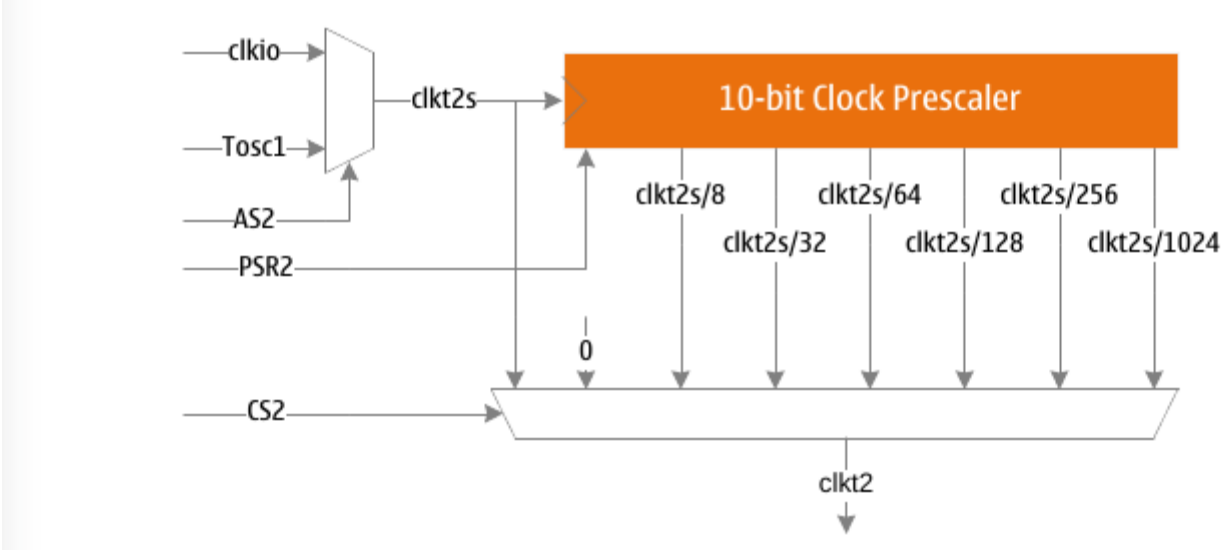



Figure 5 TC2 Prescaler Block Diagram

The above picture shows the TC2 prescaler, as shown, the possible prescale options are: clkt2s/8, clkt2s/32, clkt2s/64, clkt2s/128, clkt2s/256, and clkt2s/1024. In addition, clkt2s and 0 (stop counting) can be selected. Setting the PSR2 flag in the SFIOR register will reset the prescaler, allowing the user to start from a predictable prescaler.

## Register definition

TC2 register list

| Register | Address | Defaults | Description |
|---|---|---|---|
| CCR2A | 0xB0 | 0x00 | TC2 Control Register A |
| TCCR2B | 0xB1 | 0x00 | TC2 Control Register B |
| TCNT2 | 0xB2 | 0x00 | TC2 count value register |
| OCR2A | 0xB3 | 0x00 | TC2 output compare register A |
| OCR2B | 0xB4 | 0x00 | TC2 output compare register B |
| ASSR | 0xB6 | 0x00 | TC2 Asynchronous Status Register |
| TIMSK2 | 0x70 | 0x00 | Timer Counter Interrupt Mask Register |
| TIFR2 | 0x37 | 0x00 | Timer Counter Interrupt Flag Register |

## *TCCR2A–TC2 Control Register A*

| TCCR2 A–TC2 Control Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB0 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | COM2A1 | COM2A0 | COM2B1 | COM2B0 | - | - | WGM21 | WGM20 |
| R/W | W | R/W | R/W | R/W | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | COM2A1 | TC2 compare match output A mode control upper flag.<br>COM2A1 and COM2A0 together form the output compare mode control COM2A[1:0], which controls the output waveform of OC2A. If flag 1 or 2 of COM2A are set, the output compare waveform occupies the OC2A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM2A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 6 | COM2A0 | TC2 compare match output A mode control lower flag.<br>COM2A0 and COM2A1 together form the output compare mode control COM2A[1:0], which controls the output waveform of OC2A. If flag 1 or 2 of COM2A are set, the output compare waveform occupies the OC2A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM2A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 5 | COM2B1 | TC2 compare match output B mode control upper flag.<br>COM2B1 and COM2B0 together form the output compare mode control COM2B[1:0], which controls the output waveform of OC2B. If both flags 1 or 2 of COM2B are set, the output compare waveform occupies the OC2B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM2B controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 4 | COM2B0 | TC2 compare match output B mode control lower flag.<br>COM2B0 and COM2B1 together form the output compare mode control COM2B[1:0], which controls the output waveform of OC2B. If both flags 1 or 2 of COM2B are set, the output compare waveform occupies the OC2B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM2B controls the output comparison waveform differently, see the description of the comparison output mode control table for details. |
| 3:2 | - | Reserve |
| 1 | WGM21 | TC2 waveform generation mode control middle flag.<br>WGM20, WGM21, WGM22 together form a waveform generation mode to control WGM2[2:0], to control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |

| | | |
|---|---|---|
| 0 | WGM20 | TC2 waveform generation mode control lower flag.<br>WGM21, WGM20, WGM22 together form a waveform generation mode to control WGM2[2:0], control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |

### *TCCR2B – TC2 Control Register B*

| TCCR2B – TC2 Control Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB1 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FOC2A | FOC2B | - | - | WGM22 | CS22 | CS21 | CS20 |
| R/W | W | W | - | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | FOC2A | TC2 forces the output compare A control flag.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC2A. A forced compare match will not set the OCF2A flag, nor will it reload or clear the timer, but the output pin OC2A will be updated according to the setting of COM2A, as if a compare match had actually occurred.<br>The return value of reading FOC2A is always zero. |
| 6 | FOC2B | TC2 forces the output compare B control flag.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC2B. A forced compare match will not set the OCF2B flag, nor will it reload or clear the timer, but the output pin OC2B will be updated according to the COM2B setting, just as if a compare match actually occurred.<br>The return value of reading FOC2B is always zero. |
| 5:4 | - | Reserve |
| 3 | WGM22 | TC2 waveform generation mode control upper flag.<br>WGM22 and WGM20, WGM21 together form the waveform generation mode. Control WGM2[2:0], control the counting mode of the counter and the waveform generation mode. For details, see the description of the waveform generation mode table. |
| 2 | CS22 | TC2 clock select control upper flag.<br>Used to select the clock source of timer counter 2. |
| 1 | CS21 | TC2 clock select control middle flag.<br>Used to select the clock source of timer counter 2. |
| 0 | CS20 | TC2 clock select control lower flag.<br>Used to select the clock source of timer counter 2. |

| CS2[2:0] | Description |
|---|---|
| 0 | No clock source, stop counting |
| 1 | $clk_{t2s}$ |
| 2 | $clk_{t2s}$/8, from the prescaler |
| 3 | $clk_{t2s}$/32, from the prescaler |

| | | | |
|---|---|---|---|
| | | 4 | $clk_{t2s}$/64, from the prescaler |
| | | 5 | $clk_{t2s}$/128, from the prescaler |
| | | 6 | $clk_{t2s}$/256, from the prescaler |
| | | 7 | $clk_{t2s}$/1024, from the prescaler |

The following table shows the control of output comparison waveform in comparison output mode in non-PWM mode (ie normal mode and CTC mode).

Table 1 OC2x Compare Output Mode Control in Non-PWM Mode

| COM2x[1:0] | Description |
|---|---|
| 0 | OC2x disconnected, general IO port operation |
| 1 | Toggle OC2x signal on compare match |
| 2 | Clear OC2x signal on compare match |
| 3 | Assert OC2x signal on compare match |

The following table shows the control of output compare waveform in compare output mode in fast PWM mode.

Table 2 OC2x Compare Output Mode Control in Fast PWM Mode

| COM2x[1:0] | Description |
|---|---|
| 0 | OC2x disconnected, general IO port operation |
| 1 | Reserve |
| 2 | Clear OC2x signal when compare match, set OC2x signal when maximum value match |
| 3 | Set OC2x signal when compare match, clear OC2x signal when maximum value match |

The following table shows the control of output comparison waveform in comparison output mode in phase correction mode.

Table 3 OC2x Compare Output Mode Control in Phase Correction PWM Mode

| COM2x[1:0] | Description |
|---|---|
| 0 | OC2x disconnected, general IO port operation |
| 1 | Reserve |
| 2 | The OC2x signal is cleared when the comparison matches in the ascending order count, and the OC2x signal is set when the compare match in the descending order count. |
| 3 | The OC2x signal is set when the comparison matches in the ascending order count, and the OC2x signal is cleared when the comparison match in the descending order count. |

The following table is the waveform generation mode control.

Table 4 Waveform Generation Mode Control

| WGM2[2:0] | Operating mode | TOP value | Update OCR2x moment | Set TOV2 moment |
|---|---|---|---|---|
| 0 | Normal | 0xFF | Immediately | MAX |
| 1 | PCPWM | 0xFF | TOP | BOTTOM |
| 2 | CTC | OCR2A | Immediately | MAX |
| 3 | FPWM | 0xFF | TOP | MAX |
| 4 | Reserve | - | - | - |
| 5 | PCPWM | OCR2A | TOP | BOTTOM |
| 6 | Reserve | - | - | - |
| 7 | FPWM | OCR2A | TOP | TOP |

### *TCNT2 –TC2 count value register*

| TCNT2 –TC2 count value register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB2 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT27 | TCNT26 | TCNT25 | TCNT24 | TCNT23 | TCNT22 | TCNT21 | TCNT20 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT2 | TC2 count value register.<br>The 8 count value of the counter can be directly read and written through the TCNT2 register.<br>A write to the TCNT2 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer has stopped. This allows initializing the TCNT2 register to match the value of OCR2 without causing an interrupt.<br>If the value written to TCNT2 equals or bypasses the OCR2 value, the compare match will be lost, resulting in incorrect waveform generation results.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT2. CPU write counters have higher priority than clearing or incrementing and decrementing operations. |

## *OCR2A – TC2 Output Compare Register A*

| OCR2A – TC2 Output Compare Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB3 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR2A7 | OCR2A6 | OCR2A5 | OCR2A4 | OCR2A3 | OCR2A2 | OCR2A1 | OCR2A0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR2A | TC2 output compare register A.<br>OCR2A contains an 8-bit data that is continuously compared with the counter value TCNT2. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC2A pin. When using PWM mode, the OCR2A register uses a double-buffered register. While in normal working mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR2A register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR2A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR2A itself. |

## *OCR2B – TC2 Output Compare Register B*

| OCR2B – TC2 Output Compare Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB4 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR2B7 | OCR2B6 | OCR2B5 | OCR2B4 | OCR2B3 | OCR2B2 | OCR2B1 | OCR2B0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR2B | TC2 output compare B register.<br>OCR2B contains an 8-bit data that is continuously compared with the counter value TCNT2. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC2B pin. When using PWM mode, the OCR2B register is a double-buffered register. While in normal operating mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR2B register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses. When the double buffer function is used, the CPU accesses the OCR2B buffer register, and when the double buffer function is disabled, the CPU accesses the OCR2B itself. |

### *TIMSK2 – TC2 Interrupt Mask Register*

| TIMSK2 – TC2 Interrupt Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x70 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | - | OCIE2B | OCIE2A | TOIE2 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:3 | - | Reserve |
| 2 | OCIE2B | TC2 output compare B match interrupt enable flag.<br>When the OCIE2B flag is "1" and the global interrupt is set, the TC2 output compare B match interrupt is enabled.<br>When a compare match occurs, that is, when the OCF2B flag in TIFR2 is set, an interrupt is generated.<br>When the OCIE2B flag is "0", the TC2 output compare B match interrupt is disabled. |
| 1 | OCIE2A | TC2 output compare A match interrupt enable flag.<br>When the OCIE2A flag is "1" and the global interrupt is set, the TC2 output compare A match interrupt is enabled.<br>When a compare match occurs, that is, when the OCF2A flag in TIFR2 is set, an interrupt is generated.<br>When the OCIE2A flag is "0", the TC2 output compare A match interrupt is disabled. |
| 0 | TOIE2 | TC2 overflow interrupt enable flag.<br>When the TOIE2 flag is "1" and the global interrupt is set, the TC2 overflow interrupt is enabled. An interrupt is generated when TC2 overflows, i.e. the TOV2 flag in TIFR2 is set. When the TOIE2 flag is "0", the TC2 overflow interrupt is disabled. |

### *TIFR2 – TC2 Interrupt Flag Register*

| TIFR2 – TC2 Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x37 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | - | OCF2B | OCF2A | TOV2 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:3 | - | Reserve |
| 2 | OCF2B | TC2 output compare B match flag.<br>When TCNT2 is equal to OCR2B, the comparison unit gives a match signal and sets the comparison flag OCF2B. If the output compare B interrupt enable OCIE2B is "1" and the global interrupt flag is set at this time, the output compare B interrupt will be generated. OCF2B will be cleared automatically when this interrupt service routine is executed, or |

| | | |
|---|---|---|
| | | it can be cleared by writing "1" to the OCF2B flag. |
| 1 | OCF2A | TC2 output compare A match flag.<br>When TCNT2 is equal to OCR2A, the comparison unit gives a match signal and sets the comparison flag OCF2A. If the output compare A interrupt enable OCIE2A is "1" and the global interrupt flag is set at this time, the output compare A interrupt will be generated. OCF2A will be cleared automatically when this interrupt service routine is executed, or it can be cleared by writing "1" to the OCF2A flag. |
| 0 | TOV2 | TC2 overflow flag.<br>When the counter overflows, the overflow flag TOV2 is set. If the overflow interrupt enable TOIE2 is "1" and the global interrupt flag is set, an overflow interrupt will be generated. TOV2 will be cleared automatically when this interrupt service routine is executed, or can be cleared by writing a '1' to the TOV2 flag. |

### *ASSR – Asynchronous Interface Status Register*

| ASSR – TC2 Asynchronous Interface Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB6 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | INTCK | - | AS2 | TCN2UB | OCR2AUB | OCR2BUB | TCR2AUB | TCR2BUB |
| R/W | R/W | - | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | INTCK | Asynchronous clock select control flag.<br>When the INTCK flag is set to 1, the internal RC32K is selected as the asynchronous clock source.<br>When the INTCK flag is set to 0, the external crystal clock is selected as the asynchronous clock source. |
| 6 | - | Reserve |
| 5 | AS2 | Timer 2 asynchronous mode selection control flag.<br>When the AS2 flag is set to 1, Timer 2 works in asynchronous mode, and its clock source is selected by the INTCK flag.<br>When the AS2 flag is set to 0, Timer 2 works in synchronous mode, and its clock source is Clkio.<br>When the value of AS2 is changed, the value of the TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B registers may be incorrect and need to be reset. |
| 4 | TCN2UB | TCNT2 register update flag flag.<br>When Timer 2 is operating in asynchronous mode, the TCN2UB flag will be set when TCNT2 is written. When the value of TCNT2 is updated, the hardware will clear the TCN2UB flag. TCNT2 can only be updated when the TCN2UB flag is 0. |
| 3 | OCR2AUB | OCR2A register update flag.<br>When Timer 2 is operating in asynchronous mode, the OCR2AUB flag will be set when OCR2A is written. When the value of OCR2A is updated, the hardware will clear the OCR2AUB flag. OCR2A can only |

| | | |
|---|---|---|
| | | be updated when the OCR2AUB flag is 0. |
| 2 | OCR2BUB | OCR2B register update flag.<br>When Timer 2 operates in asynchronous mode, the OCR2BUB flag will be set when OCR2B is written. When the value of OCR2B is updated, the hardware will clear the OCR2BUB flag. OCR2B can only be updated when the OCR2BUB flag is 0. |
| 1 | TCR2AUB | TCCR2A register update flag.<br>When Timer 2 is operating in asynchronous mode, the TCR2AUB flag will be set when a write to TCCR2A is performed. When the value of TCCR2A is updated, the hardware will clear the TCR2AUB flag. TCCR2A can only be updated when the TCR2AUB flag is 0. |
| 0 | TCR2BUB | TCCR2B register update flag.<br>When Timer 2 is operating in asynchronous mode, the TCR2BUB flag will be set when a write to TCCR2B is performed. When the value of TCCR2B is updated, the hardware will clear the TCR2BUB flag. TCCR2B can only be updated when the TCR2BUB flag is 0. |

# Timer/Counter 3 (TMR3)

- True 16-bit design, allowing 16-bit PWM
- 3 independent output compare units
- Double-buffered output compare register
- 1 input capture unit
- Input Capture Noise Suppressor
- Auto clear counter and auto reload on compare match
- Phase-corrected PWM without glitches
- Variable PWM period
- frequency generator
- External event counter
- 5 independent interrupt sources
- With dead time control
- 6 selectable trigger sources automatically shut down PWM output

## Overview

TC3 is a general-purpose 16-bit timer counter module that supports PWM output and can accurately generate waveforms. TC3 includes a 16-bit counter, a waveform generation mode control unit, 2 independent output comparison units and an input capture unit. The waveform generation mode control unit controls the working mode of the counter and the generation of the comparison output waveform. According to different working modes, the counter realizes the operation of clearing, adding one or subtracting one for each count clock Clkt3. Clkt3 can be generated from an internal clock source or an external clock source. When the count value TCNT3 of the counter reaches the maximum value (equal to the maximum value 0xFFFF or fixed value or output compare register OCR3A or input capture register ICR3, defined as TOP, the maximum value is defined as MAX to show the difference), the counter will be cleared. or subtract one operation. When the count value TCNT3 of the counter reaches the minimum value (equal to 0x0000, defined as BOTTOM), the counter will increment by one. When the count value TCNT3 of the counter reaches OCR3A or OCR3B or OCR3C, which is also called a compare match, it will clear or set the output compare signal OC3A or OC3B or OC3C to generate a PWM waveform. When the input capture function is enabled, the counter starts or stops counting when triggered, and the ICR3 register will record the count value within the trigger period of the capture signal.

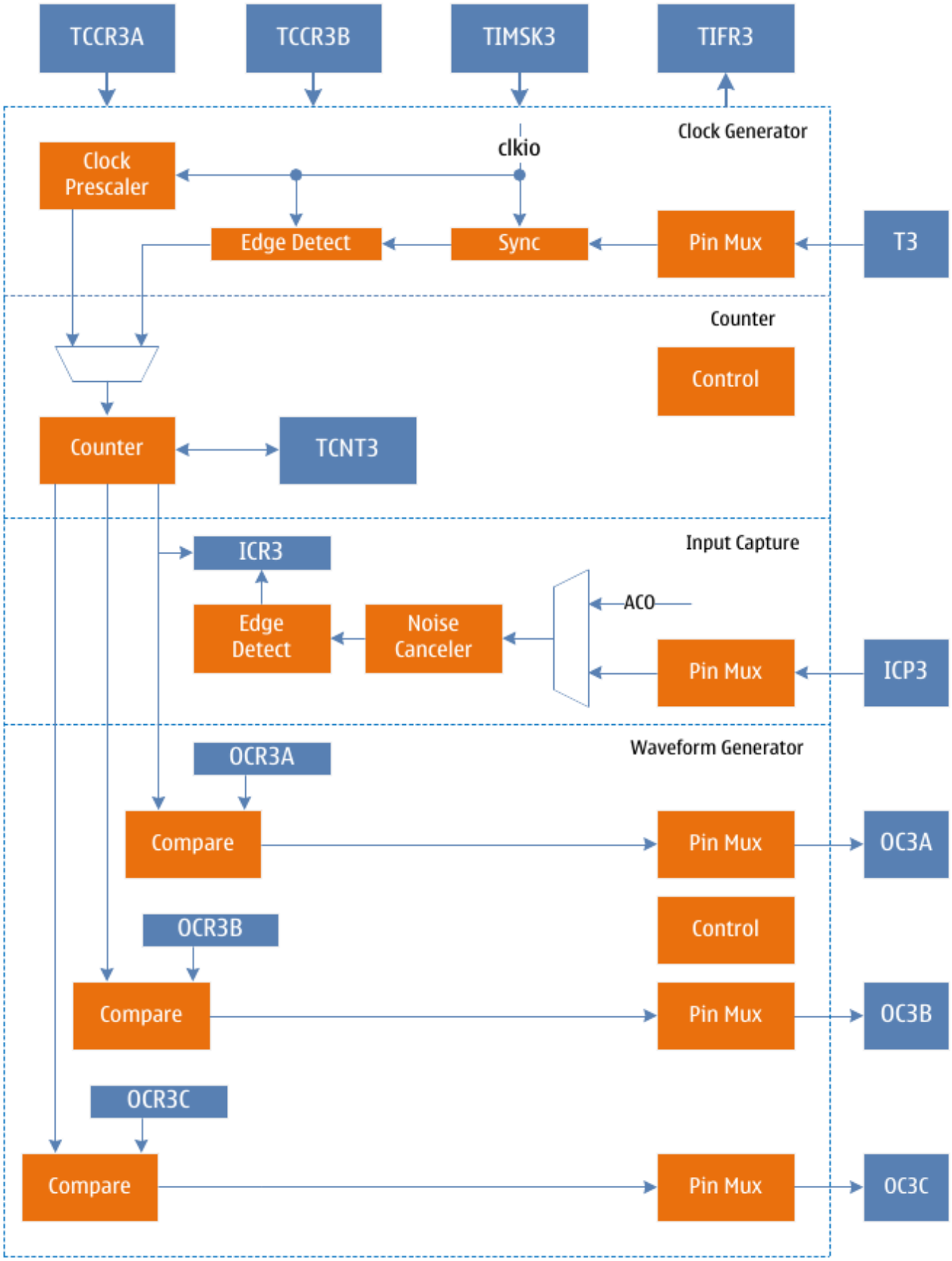



Figure 6 TC3 block diagram

## Operating mode

Timer 1 has six different working modes, including normal mode (Normal), clear on compare match (CTC) mode, fast pulse width modulation (FPWM) mode, phase correction pulse width modulation (PCPWM) mode, phase frequency correction mode Pulse Width Modulation (PFCPWM) mode, and Input Capture (ICP) mode. It is selected by the waveform generation mode control bits WGM3[3:0]. The six modes are described in detail below. Since there are three independent output compare units, they are denoted by "A", "B" and "C" respectively, and a lowercase "x" is used to denote these two output compare unit channels.

### *Normal mode*

The normal mode is the simplest working mode of the timer counter. At this time, the waveform generation mode control flags WGM3[3:0]=0, and the maximum count TOP is MAX(0xFFFF). In this mode, the counting method is incremented for each count clock, and when the counter reaches TOP and overflows, it returns to BOTTOM and starts accumulating again. The timer counter overflow flag TOV3 is set in the same count clock when the count value TCNT3 becomes zero. In this mode, the TOV3 flag is like the 17th count bit, but it will only be set and not cleared. The overflow interrupt service routine will automatically clear the TOV3 flag, which can be used by software to improve the resolution of the timer counter. There are no special cases to be considered in normal mode, and a new count value can be written at any time.

The waveform of the output compare signal OC3x can only be obtained when the data direction register of the OC3x pin is set to output. When COM3x=1, the OC3x signal will be inverted when a compare match occurs. In this case, the frequency of the waveform can be calculated by the following formula:

$$f_{OC3xnormal} = f_{sys} /(2*N*65536)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

The output compare unit can be used to generate interrupts, but it is not recommended to use interrupts in normal mode, which will take too much time.

### *CTC mode*

When setting WGM3[3:0]=4 or 12, timer counter 1 enters CTC mode. When WGM3[3]=0, the maximum count value TOP is OCR3A, and when WGM3[3]=1, the maximum count value TOP is ICR3. The following takes WGM3[3:0]=4 as an example to describe the CTC mode. In this mode, the counting method is incremented by each count clock. When the counter value TCNT3 is equal to TOP, the counter is cleared. This mode allows the user to easily control the frequency of the compare match output and simplifies the operation of external event counting.

When the counter reaches TOP=OCR3A, the output compare match flag OCF3A is set. When the counter reaches TOP=ICR3, the output compare match flag ICF3 is set, and the corresponding interrupt enable is set to generate an interrupt. The OCR3A register can be updated in the interrupt service routine. In this mode the OCR3A does not use double buffering, be careful when updating the maximum value close to the minimum value when the counter is operating with no prescaler or very low prescaler. If the value written to OCR3A is less than the current TCNT3 value, the counter will lose a compare match. Before the next compare match occurs, the counter has to count up to

MAX and then from BOTTOM to OCR3A. As in normal mode, the TOV3 flag is set in the count clock when the count value returns to 0x0.

The waveform of the output compare signal OC3x can only be obtained when the data direction register of the OC3x pin is set to output. The frequency of the waveform can be calculated using the following formula:

$$f_{OC3xctc} = f_{sys}/(2*N*(1+OCR3A))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

It can be seen from the formula that when OCR3A is set to 0x0 and there is no prescaler, the output waveform with a maximum frequency of fsys/2 can be obtained.

When WGM3[3:0]=12, it is similar to WGM3[3:0]=4, just replace OCR3A with ICR3.

### *Fast PWM mode*

When setting WGM3[3:0]=5, 6, 7, 14 or 15, timer counter 1 enters fast PWM mode, and the maximum count TOP is 0xFF, 0x1FF, 0x3FF, ICR3 or OCR3A, which can be used to generate high-frequency PWM waveform. Fast PWM mode differs from other PWM modes in that it operates in one direction. The counter accumulates from BOTTOM to TOP and then returns to BOTTOM to count again. When the count value TCNT3 reaches TOP or BOTTOM, the output compare signal OC3x will be set or cleared, depending on the setting of the compare output mode COM3, see the register description for details. Because of the unidirectional operation, the fast PWM mode operates at twice the frequency of the phase-corrected PWM mode with bidirectional operation. The high frequency characteristics make fast PWM mode suitable for power regulation, rectification and DAC applications. High frequency signals can reduce the size of external components (inductors, capacitors, etc.), thereby reducing system cost.

When the count value reaches TOP, the timer counter overflow flag TOV3 will be set, and the value of the compare buffer will be updated to the compare value. If the interrupt is enabled, the OCR3A register can be updated in the interrupt service routine.

The waveform of the output compare signal OC3x can only be obtained when the data direction register of the OC3x pin is set to output. The frequency of the waveform can be calculated using the following formula:

$$f_{OC3xfpwm} = f_{sys}/(N*(1+TOP))$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

When TCNT3 and OCR3x are compared and matched, the waveform generator will set (clear) the OC3x signal, when TCNT3 is cleared, the waveform generator will clear (set) the OC3x signal, so as to generate the PWM wave. Therefore, the extreme value of OCR3x will generate a special PWM waveform. When OCR3x is set to 0x00, the output PWM is a narrow spike pulse per (1+TOP) count clock. When OCR3x is set to TOP, the output waveform is a continuous high or low level.

If use OCR3A as TOP and set COM3A=1, the output compare signal OC3A will generate PWM wave with 50% duty cycle.

### *Phase Correction PWM Mode*

When setting WGM3[3:0]=1, 2, 3, 10 or 11, timer counter 1 enters the phase correction PWM mode, and the maximum count TOP is 0xFF, 0x1FF, 0x3FF, ICR3 or OCR3A respectively. The counter operates in two directions, incrementing from BOTTOM to TOP, then decrementing to BOTTOM, and repeating this operation. When the count reaches TOP and BOTTOM, the counting direction is changed, and the count value only stays for one count clock on TOP or BOTTOM. During the increment or decrement process, when the count value TCNT3 matches OCR3x, the output compare signal OC3x will be cleared or set, depending on the setting of the compare output mode COM3. Compared to unidirectional operation, the maximum frequency achievable in bidirectional operation is smaller, but its excellent symmetry is more suitable for motor control.

In phase correction PWM mode, the TOV3 flag is set when the count reaches BOTTOM, and the value of the compare buffer is updated to the compare value when the count reaches TOP. If interrupts are enabled, the compare buffer OCR3x registers can be updated in the interrupt service routine.

The output compare signal OC3x waveform can be obtained only when the data direction register of the OC3x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{OC3xcpcpwm} = f_{sys}/(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the up-counting process, when TCNT3 matches OCR3x, the waveform generator clears (sets) the OC3x signal.

During down counting, when TCNT3 matches OCR3x, the waveform generator sets (clears) the OC3x signal. Therefore, the extreme value of OCR3x will generate a special PWM wave. When OCR3x is set to TOP or BOTTOM, the OC3x signal output will always keep low level or high level. If OCR3A is used as TOP and COM3A=1 is set, the output compare signal OC3A will generate a PWM wave with a duty cycle of 50%.

In order to ensure the symmetry of the output PWM wave on both sides of the BOTTOM, when there is no comparison match, the OC3x signal will also be flipped in two cases. The first case is when the value of OCR3x is changed from TOP to other data. When OCR3x is TOP and the count value reaches TOP, the output of OC3x is the same as the result of the comparison and match in the previous descending count, that is, keep OC3x unchanged. At this time, the comparison value will be updated to the new value of OCR3x (not TOP), and the value of OC3x will be maintained until a comparison match occurs during the ascending count and flips. At this time, the OC3x signal is not symmetrical around the minimum value, so it is necessary to flip the OC3x signal when TCNT3 reaches the maximum value, which is the first case of flipping the OC3x signal when there is no comparison match. In the second case, when TCNT3 starts counting from a value higher than OCR3x, a compare match is lost, causing an asymmetric situation. It is also necessary to flip the OC3x signal to achieve symmetry on both sides of the minimum.

### *Phase Frequency Correction PWM Mode*

When setting WGM3[3:0]=8 or 9, timer counter 1 enters the phase frequency correction PWM mode, and the maximum count TOP is ICR3 or OCR3A respectively. The counter operates in two directions, incrementing from BOTTOM to TOP, then decrementing to BOTTOM, and repeating this operation. When the count reaches TOP and BOTTOM, the counting direction is changed, and the count value only stays for one count clock on TOP or BOTTOM. During the increment or decrement process, when the count value TCNT3 matches OCR3x, the output compare signal OC3x will be cleared or set, depending on the setting of the compare output mode COM3. Compared to unidirectional operation, the maximum achievable frequency is smaller in bidirectional operation, but its excellent symmetry is more suitable for motor control.

In the phase frequency correction PWM mode, when the count reaches BOTTOM, the TOV3 flag is set, and the value of the comparison buffer is updated to the comparison value. The time to update the comparison value is the biggest difference between the phase frequency correction PWM mode and the phase correction PWM mode. If the interrupt is enabled, the compare buffer OCR3x registers can be updated in the interrupt service routine. When the CPU changes the TOP value, that is, the value of OCR3A or ICR3, it must ensure that the new TOP value is not less than the already used TOP value, otherwise the comparison and matching will not occur again.

The output compare signal OC3x waveform can be obtained only when the data direction register of the OC3x pin is set as output. The frequency of the waveform can be calculated using the following formula:

$$f_{OC3xcpfcpwm} = f_{sys}/(N*TOP*2)$$

Among them, N represents the prescale factor (1, 8, 64, 256 or 1024).

During the up-counting process, when TCNT3 matches OCR3x, the waveform generator clears (sets) the OC3x signal.

During down counting, when TCNT3 matches OCR3x, the waveform generator sets (clears) the OC3x signal. Therefore, the extreme value of OCR3x will generate a special PWM wave. When OCR3x is set to TOP or BOTTOM, the OC3x signal output will always keep low level or high level. If OCR3A is used as TOP and COM3A=1 is set, the output compare signal OC3A will generate a PWM wave with a duty cycle of 50%.

Because the OCR3x registers are updated at BOTTOM time, the count length in ascending and descending order on both sides of the TOP value is the same, resulting in a symmetrical waveform with correct frequency and phase.

When using a fixed TOP value, it is better to use the ICR3 register as the TOP value, that is, set WGM3[3:0]=8, at this time, the OCR3A register is only used to generate PWM output. If you want to generate a PWM wave with changing frequency, you must change the TOP value, and the double-buffering feature of the OCR3A will be more suitable for this application.

### *Input capture mode*

Input capture is used to capture an external event and assign a time stamp to it to indicate the moment when the event occurs. It can be performed in the previous count mode, except for the waveform generation mode that uses the ICR3 value as the count TOP value.

The trigger signal of external event is input by pin ICP3, which can also be realized by analog comparator unit. When the logic level on pin ICP3 changes, or the output ACO level of the analog comparator changes, and this level change is captured by the input capture unit, the input capture is triggered, and the 16-bit count value TCNT3 The data is copied to the input capture register ICR3, and the input capture flag ICF3 is set at the same time. If the ICIE1 flag is "1", the input capture flag will generate an input capture interrupt.

The input capture trigger source, ICP3 or ACO, is selected by setting the analog compare input capture control flag ACIC in the analog compare control and status register ACSR. It should be noted that changing the trigger source may cause an input capture, so after changing the trigger source, ICF3 must be cleared once to avoid erroneous results.

The input capture signal is sent to the edge detector after passing through an optional noise suppressor. According to the configuration of the input capture selection control flag ICES1, it is checked whether the detected edge meets the trigger condition. The noise suppressor is a simple digital filter that samples the input signal 4 times, and its output is fed to the edge detector only when all 4 samples are equal. The noise suppressor is enabled or disabled by the ICNC1 flag of the TCCR3B register.

When using the input capture function, the value of the ICR3 register should be read as early as possible after ICF3 is set, because the value of ICR3 will be updated after the next capture event. It is recommended to enable the input capture interrupt. In any input capture working mode, it is not recommended to change the count TOP value during operation.

The time stamps captured by the input can be used to calculate frequency, duty cycle, and other characteristics of the signal, as well as to create a log of trigger events. When measuring the duty cycle of an external signal, it is required to change the trigger edge after each capture, so the trigger signal edge must be changed as soon as possible after reading the ICR3 value.

### *Automatic shutdown and restart of PWM outputs*

When the DOC3x flag of the TCCR3C register is set to high, the automatic shutdown function of the PWM output will be enabled. When the trigger condition is met, the hardware will clear the corresponding COM3x flag, disconnect the PWM output signal OC3x from its output pin, and switch to General-purpose IO output to realize automatic shutdown of PWM output. At this time, the state of the output pin can be controlled by the output of the general IO port.

After the automatic shutdown of the PWM output is enabled, the trigger condition needs to be set, and the trigger source is selected by the DSX3n flag of the TCCR3D register. Trigger sources include analog comparator interrupt, external interrupt, pin level change interrupt and timer overflow interrupt. For details, please refer to the description of the TCCR3D register. When one or some trigger sources are selected as trigger conditions, the hardware will clear the COM3x flag to close the PWM output at the same time when these interrupt flag bits are set.

When a trigger event occurs and the PWM output is turned off, the timer module has no corresponding interrupt flag. The software needs to read the interrupt flag of the trigger source to know the trigger condition and trigger event.

When the PWM output is automatically turned off and the output needs to be restarted again, the software only needs to reset the COM3x flag to switch the OC3x signal output to the corresponding

pin. It should be noted that after the automatic shutdown occurs, the timer does not stop working, and the state of the OC3x signal is also constantly updated. The software can set the COM3x flag to output the OC3x signal after the timer overflow or compare match, so that a clear PWM output state can be obtained.

## Dead time control

When the DTEN3 flag is set to "1", the function of inserting dead time is enabled, and the output waveforms of OC3A and OC3B will insert the set dead time based on the waveform generated by the B channel comparison output. The length of the time is DTR3 The time value corresponding to the count clock number of the register. As shown in the figure below, the dead-time insertion of OC3A and OC3B is based on the comparison output waveform of channel B. When COM3A and COM3B are both "2" or "3", the waveform polarity of OC3A is the same as that of OC3B. When COM3A and COM3B are "2" or "3" respectively, the waveform of OC3A is the same as that of OC3B. opposite polarity.

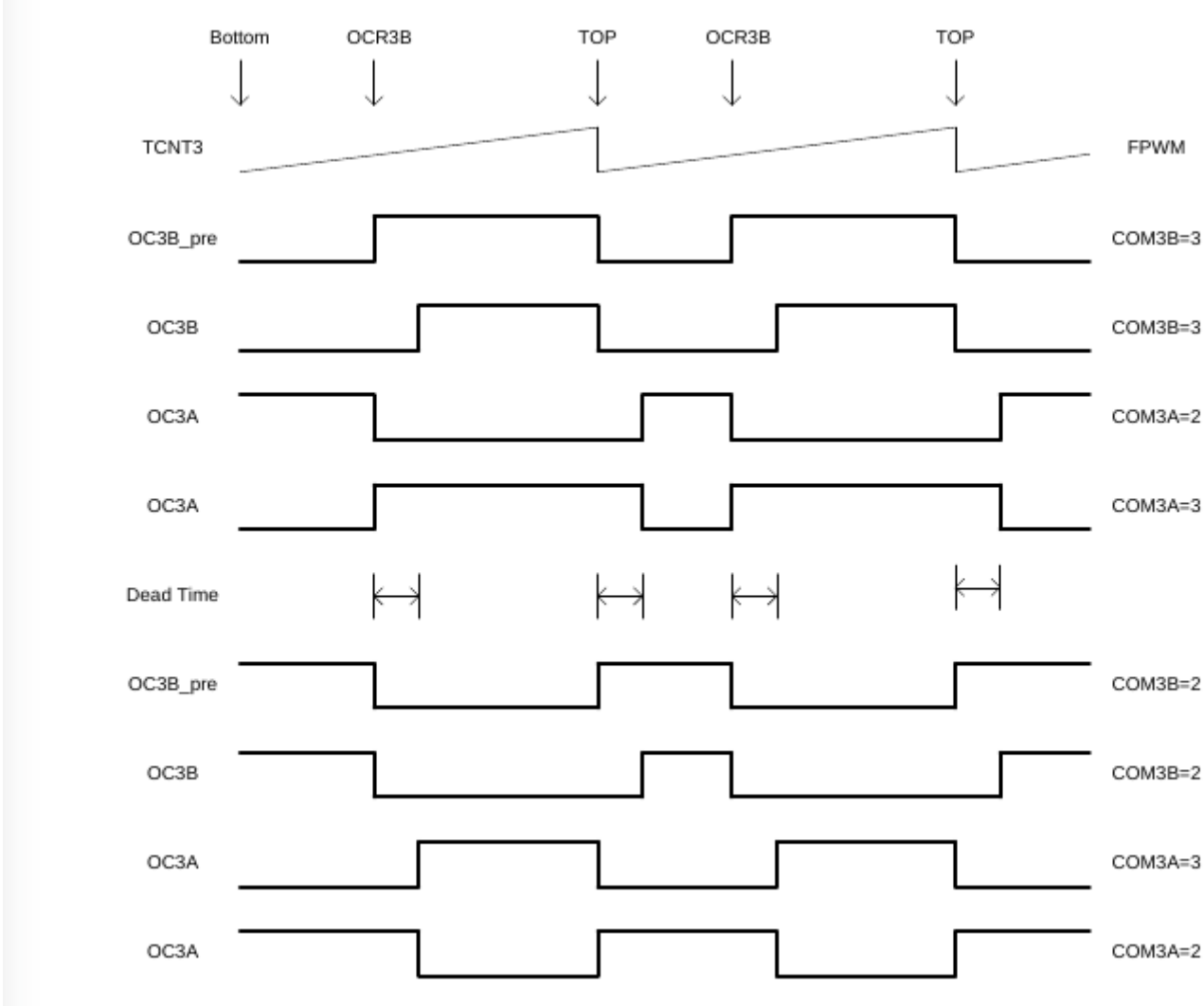



Figure 7 TC3 Dead Time Control in FPWM Mode

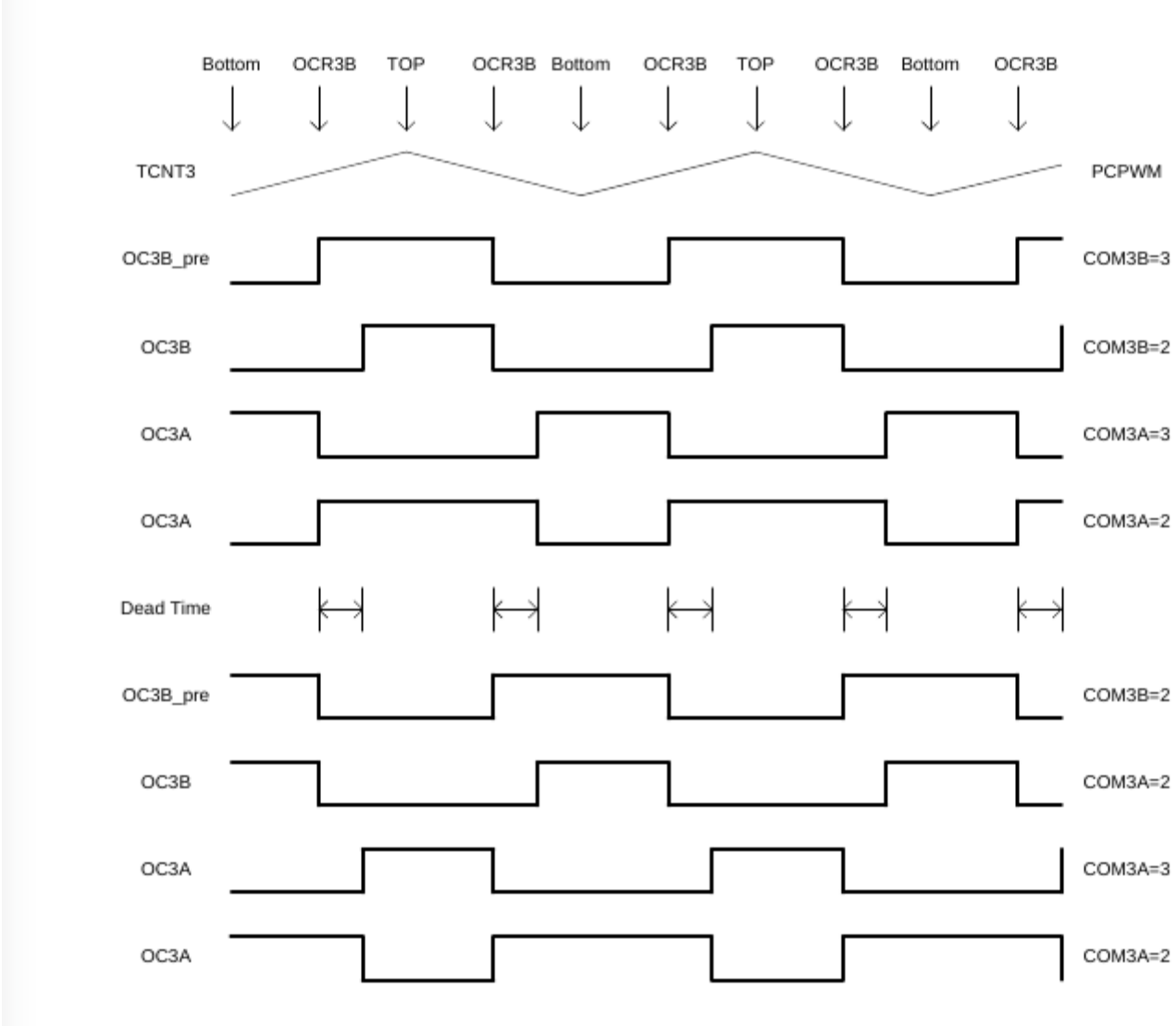



Figure 8 TC3 Dead Time Control in PCPWM Mode

When the DTEN3 flag is set to "0", the function of inserting dead time is disabled, and the output waveforms of OC3A and OC3B are the waveforms generated by their respective comparison outputs.

## Register definition

TC3 register list

| Register | Address | Defaults | Description |
|---|---|---|---|
| TCCR3A | 0x90 | 0x00 | TC3 Control Register A |
| TCCR3B | 0x91 | 0x00 | TC3 Control Register B |
| TCCR3C | 0x92 | 0x00 | TC3 Control Register C |
| TCCR3D | 0x93 | 0x00 | TC3 Control Register D |
| TCNT3L | 0x94 | 0x00 | TC3 count value register low byte |
| TCNT3H | 0x95 | 0x00 | TC3 count value register high byte |
| ICR3L | 0x96 | 0x00 | TC3 input capture register low byte |
| ICR3H | 0x97 | 0x00 | TC3 input capture register high byte |
| OCR3AL | 0x98 | 0x00 | TC3 output compare register A low byte |
| OCR3AH | 0x99 | 0x00 | TC3 output compare register A high byte |
| OCR3BL | 0x9A | 0x00 | TC3 output compare register B low byte |
| OCR3BH | 0x9B | 0x00 | TC3 output compare register B high byte |
| DTR3L | 0x9C | 0x00 | TC3 Dead Time Register Low Byte |
| DTR3H | 0x9D | 0x00 | TC3 Dead Time Register High Byte |
| OCR3CL | 0x9E | 0x00 | TC3 output compare register C low byte |
| OCR3CH | 0x9F | 0x00 | TC3 output compare register C high byte |
| TIMSK3 | 0x71 | 0x00 | Timer Counter Interrupt Mask Register |
| TIFR3 | 0x38 | 0x00 | Timer Counter Interrupt Flag Register |

### *TCCR3A–TC3 Control Register A*

| TCCR3A – TC3 Control Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x90 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | COM3A1 | COM3A0 | COM3B1 | COM3B0 | COM3C1 | COM3C0 | WGM31 | WGM30 |
| R/W | R/W | R/W | R/W | R/W | W | W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | COM3A1 | Compare match output A mode control high.<br>COM3A1 and COM3A0 form COM3A[1:0] to control the output comparison waveform OC3A. If both bits 1 or 2 of COM3A are set, the output compare waveform occupies the OC3A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 6 | COM3A0 | Compare match output A mode control low flag.<br>COM3A1 and COM3A0 form COM3A[1:0] to control the output |

| | | |
|---|---|---|
| | | comparison waveform OC3A. If both bits 1 or 2 of COM3A are set, the output compare waveform occupies the OC3A pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3A controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 5 | COM3B1 | Compare match output B mode control high.<br>COM3B1 and COM3B0 form COM3B[1:0] to control the output compare waveform OC3B. If both bits 1 or 2 of COM3B are set, the output compare waveform occupies the OC3B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3B controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 4 | COM3B0 | Compare match output B mode control low flag.<br>COM3B1 and COM3B0 form COM3B[1:0] to control the output compare waveform OC3B. If both bits 1 or 2 of COM3B are set, the output compare waveform occupies the OC3B pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3B controls the output comparison waveform differently. For details, see the description of the comparison output mode control table. |
| 3 | COM3C1 | Compare match output C mode control high flag.<br>COM3C1 and COM3C0 form COM3C[1:0] to control the output compare waveform OC3C. If both bits 1 or 2 of COM3C are set, the output compare waveform occupies the OC3C pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3C controls the output comparison waveforms differently. For details, see the description of the comparison output mode control table. |
| 2 | COM3C0 | Compare match output C mode control low flag.<br>COM3C1 and COM3C0 form COM3C[1:0] to control the output compare waveform OC3C. If both bits 1 or 2 of COM3C are set, the output compare waveform occupies the OC3C pin, but the data direction register of this pin must be set high to output this waveform. In different working modes, COM3C controls the output comparison waveforms differently. For details, see the description of the comparison output mode control table. |
| 1 | WGM31 | The waveform generation mode controls the second-lowest flag.<br>WGM31 and WGM33, WGM32, WGM30 together form a waveform generation mode to control WGM3[3:0], control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |
| 0 | WGM30 | The waveform generation mode controls the lower flag (LSB).<br>WGM30 and WGM33, WGM32, WGM31 together form a waveform generation mode to control WGM3[3:0], control the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |

The following table shows the control of output comparison waveform in comparison output mode in non-PWM mode (ie normal mode and CTC mode).

Compare Output Mode Control in Non-PWM Mode

| COM3x[1:0] | Description |
|---|---|
| 0 | OC3x disconnected, general IO port operation |
| 1 | Toggle OC3x signal on compare match |
| 2 | Clear OC3x signal on compare match |
| 3 | Assert OC3x signal on compare match |

The following table shows the control of output compare waveform in compare output mode in fast PWM mode.

Compare Output Mode Control in Fast PWM Mode

| COM3x[1:0] | Description |
|---|---|
| 0 | OC3x disconnected, general IO port operation |
| 1 | When WGM3 is 15: OC3A signal is flipped when compare match, OC3B is disconnected<br>When WGM3 is other value: OC3x is disconnected, general IO port operation |
| 2 | Clear OC3x signal when compare match, set OC3x signal when maximum value match |
| 3 | Set OC3x signal when compare match, clear OC3x signal when maximum value match |

The following table shows the control of output comparison waveform in comparison output mode in phase correction mode.

Phase Correction and Phase Frequency Correction
Comparison Output Mode Control in PWM Mode

| COM3x[1:0] | Description |
|---|---|
| 0 | OC3x disconnected, general IO port operation |
| 1 | When WGM3 is 9 or 11: OC3A signal is toggled when compare match, OC3B is disconnected<br>When WGM3 is other value: OC3x is disconnected, general IO port operation |
| 2 | The OC3x signal is cleared by the comparison match in the ascending sequence count, and the OC3x signal is set by the compare match in the descending sequence count |
| 3 | The OC3x signal is set by the comparison match in the ascending sequence count, and the OC3x signal is cleared by the compare match in the descending sequence count |

### *TCCR3B–TC3 Control Register B*

| TCCR3B – TC3 Control Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x91 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICNC3 | ICES3 | - | WGM33 | WGM32 | CS32 | CS31 | CS30 |
| R/W | R/W | R/W | - | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | ICNC3 | Input capture noise suppressor enable control flag.<br>When the ICNC3 flag is set to "1", the input capture noise suppressor is enabled. At this time, the input of the external pin ICP3 is filtered, and the input signal is valid only when the consecutive 4 sample values are equal. This function makes the input capture delayed by 4 clock cycles.<br>When the ICNC3 flag is set to "0", the input capture noise suppressor is disabled, and the input of the external pin ICP3 is directly valid. |
| 6 | ICES3 | Input capture trigger edge select control flag.<br>When the ICES3 flag is set to "1", the rising edge of the selected level triggers the input capture; when the ICES3 flag is set to "0", the falling edge of the selected level triggers the input capture.<br>When an event is captured, the value of the counter is copied to the ICR3 register and the input capture flag ICF3 is set at the same time. If the interrupt is enabled, an input capture interrupt is generated. |
| 5 | - | Reserve |
| 4 | WGM33 | Waveform generation mode control high.<br>WGM33 and WGM32, WGM31, WGM30 together form the waveform generation mode control.<br>WGM3[3:0], controls the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |
| 3 | WGM32 | The waveform generation mode controls the second highest flag.<br>WGM32 and WGM33, WGM31, WGM30 together form the waveform generation mode control.<br>WGM3[3:0], controls the counting mode of the counter and the waveform generation mode, see the waveform generation mode table description for details. |
| 2 | CS32 | Clock select control high.<br>Used to select the clock source of timer counter 3. |
| 1 | CS31 | The clock select controls middle flag.<br>Used to select the clock source of timer counter 3. |
| 0 | CS30 | Clock Select Control Low.<br>Used to select the clock source of timer counter 3. |

| CS3[2:0] | Description |
|---|---|
| 0 | No clock source, stop counting |
| 1 | $clk_{sys}$ |
| 2 | $clk_{sys}$/8, from the prescaler |

| | | | |
|---|---|---|---|
| | | 3 | $clk_{sys}$/64, from the prescaler |
| | | 4 | $clk_{sys}$/256, from the prescaler |
| | | 5 | $clk_{sys}$/1024, from the prescaler |
| | | 6 | External clock T3 pin, falling edge trigger |
| | | 7 | External clock T3 pin, rising edge trigger |

The following table is the waveform generation mode control.

Table 5 Waveform Generation Mode Control

| WGM3[3:0] | Operating mode | TOP value | Update OCR1A moment | Set TOV3 moment |
|---|---|---|---|---|
| 0 | Normal | 0xFFFF | Immediately | MAX |
| 1 | 8-bit PCPWM | 0x00FF | TOP | BOTTOM |
| 2 | 9-bit PCPWM | 0x01FF | TOP | BOTTOM |
| 3 | 10-bit PCPWM | 0x03FF | TOP | BOTTOM |
| 4 | CTC | OCR3A | Immediately | MAX |
| 5 | 8-bit FPWM | 0x00FF | BOTTOM | TOP |
| 6 | 9-bit FPWM | 0x01FF | BOTTOM | TOP |
| 7 | 10-bit FPWM | 0x03FF | BOTTOM | TOP |
| 8 | PFCPWM | ICR3 | BOTTOM | BOTTOM |
| 9 | PFCPWM | OCR3A | BOTTOM | BOTTOM |
| 10 | PCPWM | ICR3 | TOP | BOTTOM |
| 11 | PCPWM | OCR3A | TOP | BOTTOM |
| 12 | CTC | ICR3 | Immediately | MAX |
| 13 | Reserve | - | - | - |
| 14 | FPWM | ICR3 | TOP | TOP |
| 15 | FPWM | OCR3A | TOP | TOP |

### *TCCR3C–TC3 Control Register C*

| TCCR3C – TC3 Control Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x92 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FOC3A | FOC3B | DOC3B | DOC3A | DTEN3 | - | DOC3C | FOC3C |
| R/W | W | W | - | - | - | - | - | - |

| Bit | Name | Description |
|---|---|---|
| 7 | FOC3A | Force output compare A.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC3A. A forced compare match will not set the OCF3A flag, nor will it reload or clear the timer, but the output pin OC3A will be updated according to the setting of COM3A, as if a compare match had actually occurred.<br>When operating in PWM mode, it must be cleared when writing to the TCCR3A register.<br>The return value of reading FOC3A is always zero. |
| 6 | FOC3B | Force output compare B.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC3B. A forced compare match will not set the OCF3B flag, nor will it reload or clear the timer, but the output pin OC3B will be updated according to the COM3B setting, as if a compare match had actually occurred.<br>When operating in PWM mode, it must be cleared when writing to the TCCR3A register.<br>The return value of reading FOC3B is always zero. |
| 5 | DOC3B | Disable output compare B enable control flag.<br>When the DOC3B flag is high, the hardware disables the output compare B and is enabled. After the output disable condition is met, the COM3B flag will be cleared, the output pin OC3B will be disconnected, and the pin will become a general-purpose IO operation.<br>When the DOC3B flag is low, the hardware disable output compare B function is invalid. |
| 4 | DOC3A | Disable output compare A enable control flag.<br>When the DOC3A flag is high, the hardware disables the output comparison A and is enabled. After the output disable condition is met, the COM3A flag will be cleared, the output pin OC3A will be disconnected, and the pin will become a general-purpose IO operation.<br>When the DOC3A flag is low, the hardware disable output compare A function is invalid. |
| 3 | DTEN3 | Dead time enable control flag.<br>When the DTEN3 flag is high, dead time is enabled, OC3A and OC3B become complementary outputs, and the dead time is inserted as set by DTR3L and DTR3H.<br>Dead time is disabled when the DTEN3 flag is low. Both OC3A and OC3B are single output. |
| 2 | - | |

| | | |
|---|---|---|
| 1 | DOC3C | Disable output compare C enable control flag.<br>When the DOC3C flag is high, the hardware prohibits the output compare C and is enabled. After the output prohibition conditions are met, the COM3C flag will be cleared, the output pin OC3C will be disconnected, and the pin will become a general-purpose IO operation.<br>When the DOC3C flag is low, the hardware disable output compare C function is invalid. |
| 0 | FOC3C | Force output compare C.<br>When working in non-PWM mode, a compare match can be generated by writing "1" to the forced output compare flag FOC3C. A forced compare match will not set the OCF3C flag, nor will it reload or clear the timer, but the output pin OC3C will be updated according to the COM3C setting, as if a compare match had actually occurred.<br>When operating in PWM mode, it must be cleared when writing to the TCCR3A register.<br>The return value of reading FOC3C is always zero. |

### *TCCR3D–TC3 Control Register D*

| TCCR3D – TC3 Control Register D | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x93 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DSX37 | DSX36 | DSX35 | DSX34 | - | - | DSX31 | DSX30 |
| R/W | R/W | R/W | R/W | R/W | - | - | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | DSX37 | TC3 trigger source selection control enable flag 7.<br>When the DSX37 flag is set to "1", the TC0 overflow is enabled as the trigger source to turn off the output compare signal waveform OC3x.<br>When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output.<br>When the DSX37 flag is set to "0", the TC0 overflow is disabled as the trigger source to turn off the output compare signal waveform OC3x. |
| 6 | DSX36 | TC3 trigger source selection control enable flag 6.<br>When the DSX36 flag is set to "1", the TC2 overflow is enabled as the trigger source to turn off the output compare signal waveform OC3x.<br>When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output.<br>When the DSX36 flag is set to "0", the TC2 overflow is disabled as the trigger source to turn off the output compare signal waveform OC3x. |
| 5 | DSX35 | TC3 trigger source selection control enable flag 5.<br>When the DSX35 flag is set to "1", the pin level changes 1 as the trigger source of the output compare signal waveform OC3x is enabled.<br>When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output. |

| | | |
|---|---|---|
| | | When the DSX35 flag is set to "0", the pin level changes 1 as the trigger source of the output compare signal waveform OC3x is disabled. |
| 4 | DSX34 | TC3 trigger source selection control enable flag 4.<br>When the DSX34 flag is set to "1", the external interrupt 1 is enabled as the trigger source to turn off the output compare signal waveform OC3x. When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output.<br>When the DSX34 flag is set to "0", the external interrupt 1 is disabled as the trigger source to turn off the output compare signal waveform OC3x. |
| 3:2 | - | Reserve |
| 1 | DSX31 | TC3 trigger source selection control enable flag 1.<br>When the DSX31 flag is set to "1", the analog comparator 1 is enabled as the trigger source to turn off the output compare signal waveform OC3x. When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output.<br>When the DSX31 flag is set to "0", the analog comparator 1 is disabled as the trigger source to turn off the output compare signal waveform OC3x. |
| 0 | DSX30 | TC3 trigger source selection control enable flag 0.<br>When the DSX30 flag is set to "1", the analog comparator 0 is enabled as the trigger source to turn off the output compare signal waveform OC3x. When the DOC3x flag is "1", the rising edge of the interrupt flag register flag of the selected trigger source will automatically turn off the OC3x waveform output.<br>When the DSX30 flag is set to "0", the analog comparator 0 is disabled as the trigger source to turn off the output compare signal waveform OC3x. |

The following table shows the selection control of the trigger source of the waveform output. Disable the trigger source selection control of the OC3x waveform output

| DOC3x | DSX3n=1 | Trigger source | Description |
|---|---|---|---|
| 0 | - | - | When the DOC3x flag is "0", the trigger source is turned off and the waveform output function is disabled |
| 1 | 0 | Analog Comparator 0 | A rising edge of ACIF0 will turn off the OC3x waveform output |
| 1 | 1 | Analog Comparator 1 | A rising edge on ACIF1 will turn off the OC3x waveform output |
| 1 | 4 | External Interrupt 1 | A rising edge on INTF1 will turn off the OC3x waveform output |
| 1 | 5 | Pin level change 1 | A rising edge on PCIF1 will turn off the OC3x waveform output |
| 1 | 6 | TC2 overflow | A rising edge on TOV2 will turn off the OC3x waveform output |
| 1 | 7 | TC0 overflow | A rising edge on TOV0 will turn off the OC3x waveform output |

Notice:

2) DSX3n=1 means that when the $n^{th}$ bit of the TCCR1D register is 1, each register bit can be set at the same time.

### *TCNT3L–TC3 Counter Register Low Byte*

| TCNT3L – TC3 count value register low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x94 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT3L7 | TCNT3L6 | TCNT3L5 | TCNT3L4 | TCNT3L3 | TCNT3L2 | TCNT3L1 | TCNT3L0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT3L | Low byte of TC3 count value.<br>TCNT3H and TCNT3L are combined together to form TCNT3, and the 16-bit count value of the counter can be read and written directly through the TCNT3 register. Reading and writing a 16-bit register requires two operations. When writing 16-bit TCNT3, TCNT3H should be written first.<br>When reading 16-bit TCNT3, TCNT3L should be read first.<br>A write to the TCNT3 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer is stopped. This allows initializing the TCNT3 register to match the OCR3x value without causing an interrupt.<br>If the value written to TCNT3 equals or bypasses the OCR3x value, the compare match will be lost, resulting in an incorrect waveform result.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT3. The CPU write counter has higher priority than a clear or increment or decrement operation. |

### *TCNT3H–TC3 Counter Register High Byte*

| TCNT3H – TC3 count value register high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x95 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TCNT3H7 | TCNT3H6 | TCNT3H5 | TCNT3H4 | TCNT3H3 | TCNT3H2 | TCNT3H1 | TCNT3H0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TCNT3H | High byte of TC3 count value.<br>TCNT3H and TCNT3L are combined together to form TCNT3, and the 16-bit count value of the counter can be read and written directly through the TCNT3 register. Reading and writing a 16-bit register requires two operations. When writing 16-bit TCNT3, TCNT3H should be written first.<br>When reading 16-bit TCNT3, TCNT3L should be read first.<br>A write to the TCNT3 register by the CPU will prevent a compare match from occurring on the next timer clock cycle, even if the timer is stopped. This allows initializing the TCNT3 register to match the OCR3x value without causing an interrupt.<br>If the value written to TCNT3 equals or bypasses the OCR3x value, the compare match will be lost, resulting in an incorrect waveform result.<br>The timer stops counting when no clock source is selected, but the CPU can still access TCNT3. The CPU write counter has higher priority than a clear or increment or decrement operation. |

### *ICR3L–TC3 capture register low byte*

| ICR3L –TC3 Input Capture Register Low Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x96 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICR3L7 | ICR3L6 | ICR3L5 | ICR3L4 | ICR3L3 | ICR3L2 | ICR3L1 | ICR3L0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | ICR3L | TC3 Input the low byte of the captured value.<br>ICR3H and ICR3L combine to form the 16-bit ICR3. Reading and writing a 16-bit register requires two operations. When writing 16-bit ICR3, ICR3H should be written first. When reading 16-bit ICR3, ICR3L should be read first.<br>When the input capture is triggered, the count value TCNT3 will be updated and copied to the ICR3 register. The ICR3 register can also be used to define the TOP value of the count. |

### *ICR3H–TC3 capture register high byte*

| ICR3H –TC3 Input Capture Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x97 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ICR3H7 | ICR3H6 | ICR3H5 | ICR3H4 | ICR3H3 | ICR3H2 | ICR3H1 | ICR3H0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | ICR3H | TC3 Input high byte of captured value.<br>ICR3H and ICR3L combine to form the 16-bit ICR3. Reading and writing a 16-bit register requires two operations. When writing 16-bit ICR3, ICR3H should be written first. When reading 16-bit ICR3, ICR3L should be read first.<br>When the input capture is triggered, the count value TCNT3 will be updated and copied to the ICR3 register. The ICR3 register can also be used to define the TOP value of the count. |

### *OCR3AL–TC3 Output Compare Register A Low Byte*

| OCR3AL – TC3 output compare register A low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x98 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3AL7 | OCR3AL6 | OCR3AL5 | OCR3AL4 | OCR3AL3 | OCR3AL2 | OCR3AL1 | OCR3AL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3AL | Output compare register A low byte.<br>OCR3AL and OCR3AH combine to form the 16-bit OCR3A. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3A, OCR3AH should be written first. When reading 16-bit OCR3A, OCR3AL should be read first.<br>OCR3A continuously compares with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3A pin.<br>When using PWM mode, the OCR3A register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3A register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR3A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR3A itself. |

## *OCR3AH–TC3 Output Compare Register A High Byte*

| OCR3AH – TC3 output compare register A high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x99 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3AH7 | OCR3AH6 | OCR3AH5 | OCR3AH4 | OCR3AH3 | OCR3AH2 | OCR3AH1 | OCR3AH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3AH | Output compare register A high byte.<br>OCR3AL and OCR3AH combine to form the 16-bit OCR3A. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3A, OCR3AH should be written first. When reading 16-bit OCR3A, OCR3AL should be read first.<br>OCR3A continuously compares with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3A pin.<br>When using PWM mode, the OCR3A register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3A register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR3A buffer register, and when the double buffer function is disabled, the CPU accesses the OCR3A itself. |

### *OCR3BL–TC3 Output Compare Register B Low Byte*

| OCR3BL – TC3 output compare register B low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9A | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3BL7 | OCR3BL6 | OCR3BL5 | OCR3BL4 | OCR3BL3 | OCR3BL2 | OCR3BL1 | OCR3BL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3BL | Output compare register B low byte.<br>OCR3BL and OCR3BH combine to form the 16-bit OCR3B. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3B, OCR3BH should be written first. When reading 16-bit OCR3B, OCR3BL should be read first.<br>OCR3B is continuously compared with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3B pin.<br>When using PWM mode, the OCR3B register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3B register with the count maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR3B buffer register, and when the double buffer function is disabled, the CPU accesses the OCR3B itself. |

### *OCR3BH–TC3 Output Compare Register B High Byte*

| OCR3BH – TC3 output compare register B high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9B | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3BH7 | OCR3BH6 | OCR3BH5 | OCR3BH4 | OCR3BH3 | OCR3BH2 | OCR3BH1 | OCR3BH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3BH | Output compare register B high byte.<br>OCR3BL and OCR3BH combine to form the 16-bit OCR3B. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3B, OCR3BH should be written first. When reading 16-bit OCR3B, OCR3BL should be read first.<br>OCR3B is continuously compared with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3B pin.<br>When using PWM mode, the OCR3B register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3B register with the time of counting the maximum or minimum |

| | | |
|---|---|---|
| | | value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When using the double buffer function, the CPU accesses the OCR3B buffer register. When the double buffer function is disabled, the CPU accesses the OCR3B buffer register.<br>OCR3B itself. |

### *OCR3CL–TC3 Output Compare Register C Low Byte*

| OCR3CL – TC3 output compare register C low byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9E | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3CL7 | OCR3CL6 | OCR3CL5 | OCR3CL4 | OCR3CL3 | OCR3CL2 | OCR3CL1 | OCR3CL0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3CL | Output compare register C low byte.<br>OCR3CL and OCR3CH are combined to form the 16-bit OCR3C. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3C, OCR3CH should be written first. When reading 16-bit OCR3C, OCR3CL should be read first.<br>OCR3C continuously compares with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3C pin.<br>When using PWM mode, the OCR3C register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3C register with the time of counting the maximum or minimum value, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffering function is used, the CPU accesses the OCR3C buffer register, and when the double buffering function is disabled, the CPU accesses the OCR3C itself. |

### *OCR3CH–TC3 Output Compare Register C High Byte*

| OCR3CH –TC3 output compare register C high byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9F | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OCR3CH7 | OCR3CH6 | OCR3CH5 | OCR3CH4 | OCR3CH3 | OCR3CH2 | OCR3CH1 | OCR3CH0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | OCR3CH | Output compare register C high byte.<br>OCR3CL and OCR3CH are combined to form the 16-bit OCR3C. Reading and writing a 16-bit register requires two operations. When writing 16-bit OCR3C, OCR3CH should be written first. When reading 16-bit OCR3C, OCR3CL should be read first.<br>OCR3C continuously compares with the counter value TCNT3. A compare match can be used to generate an output compare interrupt, or to generate a waveform on the OC3C pin.<br>When using PWM mode, the OCR3C register uses a double-buffered register. In normal operation mode and match clear mode, double buffering is disabled. Double buffering can synchronize the update of the OCR3C register with the count maximum or minimum time, thereby preventing the generation of asymmetric PWM pulses and eliminating interference pulses.<br>When the double buffer function is used, the CPU accesses the OCR3C buffer register, and when the double buffer function is disabled, the CPU accesses the OCR3C itself. |

### *DTR3L–TC3 Dead Time Register Low Byte*

| DTR3L – TC3 Dead Time Register Low Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9C | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTR3L7 | DTR3L6 | DTR3L5 | DTR3L4 | DTR3L3 | DTR3L2 | DTR3L1 | DTR3L0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | DTR3L | Dead time register low byte.<br>When the DTEN3 flag is high, OC3A and OC3B are complementary outputs, and the dead time inserted on the OC3A output is determined by DTR3L count clocks. |

### *DTR3H–TC3 Dead Time Register High Byte*

| DTR3H – TC3 Dead Time Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x9D | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTR3H7 | DTR3H6 | DTR3H5 | DTR3H4 | DTR3H3 | DTR3H2 | DTR3H1 | DTR3H0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | DTR3H | Dead time register high byte.<br>When the DTEN3 flag is high, OC3A and OC3B are complementary outputs, and the dead time inserted on the OC3B output is determined by DTR3H count clocks. |

### *TIMSK3–TC3 Interrupt Mask Register*

| TIMSK3 – TC3 Interrupt Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x71 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | ICIE3 | - | OCIE3C | OCIE3B | OCIE3A | TOIE3 |
| R/W | - | - | R/W | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | - | Reserve |
| 5 | ICIE3 | TC3 Input capture interrupt enable control flag.<br>When the ICIE3 flag is "1" and the global interrupt is set, the TC3 input capture interrupt is enabled. When the input capture is triggered, i.e. the ICF3 flag of TIFR3 is set, an interrupt occurs.<br>When the ICIE3 flag is "0", the TC3 input capture interrupt is disabled. |
| 4 | - | Reserve |
| 3 | OCIE3C | TC3 output compare C match interrupt enable flag.<br>When the OCIE3C flag is "1" and the global interrupt is set, the TC3 output compare C match interrupt is enabled. When a compare match occurs, that is, when the OCF3C flag in TIFR3 is set, an interrupt is generated.<br>When the OCIE3C flag is "0", the TC3 output compare C match interrupt is disabled. |
| 2 | OCIE3B | TC3 output compare B match interrupt enable flag.<br>When the OCIE3B flag is "1" and the global interrupt is set, the TC3 output compare B match interrupt is enabled. When a compare match occurs, that is, when the OCF3B flag in TIFR3 is set, an interrupt is generated.<br>When the OCIE3B flag is "0", the TC3 output compare B match interrupt is disabled. |
| 1 | OCIE3A | TC3 output compare A match interrupt enable flag.<br>When the OCIE3A flag is "1" and the global interrupt is set, the TC3 |

| | | |
|---|---|---|
| | | output compare A match interrupt is enabled. When a compare match occurs, that is, when the OCF3A flag in TIFR3 is set, an interrupt is generated. When the OCIE3A flag is "0", the TC3 output compare A match interrupt is disabled. |
| 0 | TOIE3 | TC3 overflow interrupt enable flag.<br>When the TOIE3 flag is "1" and the global interrupt is set, the TC3 overflow interrupt is enabled. An interrupt is generated when TC3 overflows, i.e. the TOV3 flag in TIFR3 is set. When the TOIE3 flag is "0", the TC3 overflow interrupt is disabled. |

### *TIFR3–TC3 Interrupt Flag Register*

| TIFR3 – TC3 Interrupt Flag Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x38 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | ICF3 | - | OCF3C | OCF3B | OCF3A | TOV3 |
| R/W | - | - | R/W | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | - | Reserve |
| 5 | ICF3 | Input capture flag flag.<br>When an input capture event occurs, the ICF3 flag is set. When ICR3 is used as the TOP value of the count, and the count value reaches the TOP value, the ICF3 flag is set. If ICIE1 is "1" and the global interrupt flag is set, an input capture interrupt will be generated. The ICF3 flag will not be cleared automatically, and software needs to write "1" to the ICF3 flag to clear it. |
| 4 | - | Reserve |
| 3 | OCF3C | Output compare C match flag.<br>When TCNT3 is equal to OCR3C, the comparison unit gives a match signal and sets the comparison flag OCF3C.<br>If the output compare interrupt enable OCIE3C is "1" and the global interrupt flag is set, an output compare interrupt will be generated. The OCF3 flag will not be cleared automatically, it is required to write "1" to the OCF3C flag by software to clear this flag. |
| 2 | OCF3B | Output compare B match flag.<br>When TCNT3 is equal to OCR3B, the comparison unit gives a match signal and sets the comparison flag OCF3B.<br>If the output compare interrupt enable OCIE3B is "1" and the global interrupt flag is set, an output compare interrupt will be generated.<br>The OCF3B flag will not be cleared automatically, and software needs to write "1" to the OCF3B flag to clear it. |
| 1 | OCF3A | Output compare A match flag.<br>When TCNT3 is equal to OCR3A, the comparison unit gives a match signal and sets the comparison flag OCF3A.<br>If the output compare interrupt enable OCIE3A is "1" and the global |

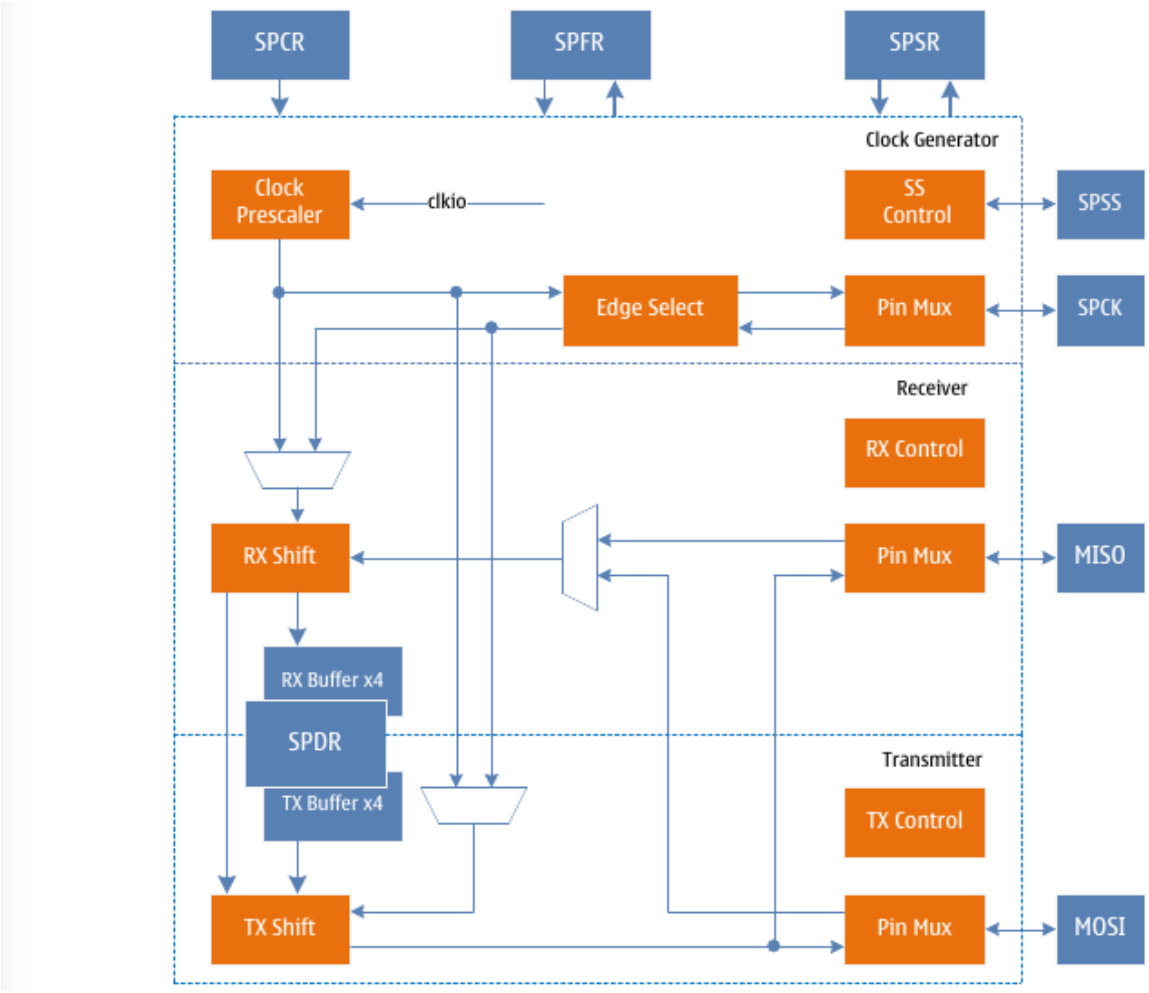
| | | |
|---|---|---|
| | | interrupt flag is set, an output compare interrupt will be generated. The OCF3A flag will not be cleared automatically, and software needs to write "1" to the OCF3A flag to clear it. |
| 0 | TOV3 | Overflow flag.<br>When the counter overflows, the overflow flag TOV3 is set. If the overflow interrupt enable TOIE3 is "1" and the global interrupt flag is set, an overflow interrupt will be generated.<br>The TOV3 flag will not be automatically cleared, and software needs to write "1" to the TOV3 flag to clear it. |

# Synchronous Serial Peripheral Interface (SPI)

- Full-duplex, three-wire synchronous data transmission
- Master or Slave Operation
- LSB or MSB first transmission
- 7 programmable bit rates
- send end interrupt flag
- Write conflict flag protection mechanism
- Wake-up from idle mode
- Host operation with double speed mode
- Support host two-line input mode
- 4 buffer registers for input/output

## Overview

SPI mainly includes three parts: clock prescaler, clock detector, slave select detector, transmitter and receiver.




SPI block diagram

Control and status registers are shared by these three parts. The clock prescaler only works in the host operation mode, and the frequency division factor is selected by the bit rate control flag, thereby generating the corresponding frequency division clock and outputting it to the SPCK pin. The clock detector only works in the slave operation mode, detects the clock edge input from the SPCK pin, and performs shift operations on the transmit and receive shift registers according to the SPI data transfer mode. The slave selection detector detects the slave selection signal SPSS, and obtains the transmission state to control the operation of the transmitter and receiver. The transmitter consists of a shift register and transmit control logic. The receiver consists of a shift register, four receive buffers and receive control logic.

## Clock generation

The clock generation logic is divided into master clock prescaler and slave clock detector, which work in master operation and slave operation mode respectively. The clock prescaler selects the frequency division coefficient by the bit rate control flag and the multiplier control flag, and generates the corresponding frequency division clock (there are 7 optional frequency division coefficients in total, see the register description for details), and the output to the SPCK pin is The communication provides the clock, as well as the shift clock for the internal transmit and receive shift registers. The clock detector performs edge detection on the input clock SPCK, and performs shift operations on the transmitter and receiver according to the SPI data transfer mode. In order to ensure the correct sampling of the clock signal, the width of the high level and the low level of the SPCK clock must be greater than 2 system clock cycles.

## Send and receive

The SPI module supports simultaneous transmission and reception in single-wire mode, and only supports dual-wire reception by the host in dual-wire mode.

## One-wire transmit and receive

The SPI host pulls down the slave selection signal SPSS that needs to be communicated to start a transmission process. The master and the slave prepare the data to be transmitted, the master generates a clock pulse on the clock signal SPCK to exchange data, the data of the master is moved out from MOSI, and moved in from MISO, and the data of the slave is moved out from MISO, and moved in from MOSI, and the exchange is completed. After the data, the host can pull up the SPSS signal to complete the communication.

When configured as a host, the SPI module does not control the SPSS pins and must be handled by the user software. The software pulls down the SPSS pin, selects the slave to communicate, and starts the transmission. When the software writes the data to be transmitted into the SPDR register, the clock generator is started, the hardware generates the communication clock, and shifts the 8-bit data out to the slave, and shifts the data in the slave at the same time. After shifting one byte of data, stop the clock generator and set the transfer completion flag SPIF. The software can write data to the SPDR register again to continue the transmission of the next byte, or it can pull the SPSS signal high to end the current transmission. The last incoming data will be held in the receive buffer.

When configured as a slave, the SPI module will stay asleep and keep the MISO pin tri-stated as long as the SPSS signal is high. The software can then update the contents of the SPDR register.

Even if there is an input clock pulse on the SPCK pin at this time, the data of SPDR will not be shifted out until the SPSS signal is pulled low. When the data transfer of one byte is completed, the hardware sets the transfer complete flag SPIF. At this time, the software can continue to write data to the SPDR register before reading the shifted data, and the last incoming data will be stored in the receive buffer.

The SPI module has only four buffers in the transmit direction and four buffers in the receive direction. When sending data, when the sending buffer is not full (that is, the sending buffer full flag WRFULL is low), the SPDR register can be written. When receiving data, when the receive buffer is in a non-empty state (that is, the receive buffer empty flag RDEMPT is low), the received character can be read by accessing the SPDR register.

## Host two-wire reception

The two-wire mode of the SPI module is only valid in the host operation mode. The difference from the single-wire mode is that both MOSI and MISO are used for the host to receive data, and each SPCK clock pulse simultaneously receives 2-bit data (the data on the MISO line is first. , the data on the MOSI line is behind), after receiving two bytes of data, the hardware sets the transmission completion flag SPIF, and the data is stored in the receive buffer and shift register. At this point software must read the SPDR register twice to get the two bytes of data received. It should be noted that although the master does not send data to the slave in dual-wire mode, the software still needs to write data to the SPDR register to start the clock generator to generate the communication clock, and write the SPDR register once to receive two bytes of data.

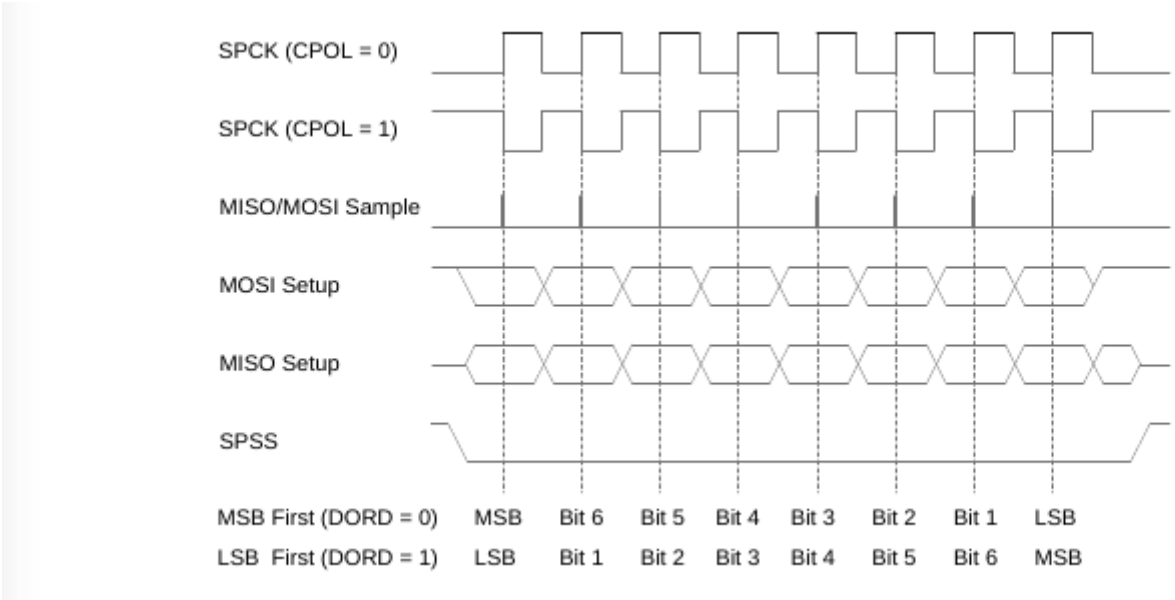
## Data schema

In single-wire mode, SPI has 4 combinations of SPCK phase and polarity relative to serial data, which are controlled by CPHA and CPOL, as shown in the table below.

*CPHA and CPOL select data transfer mode*

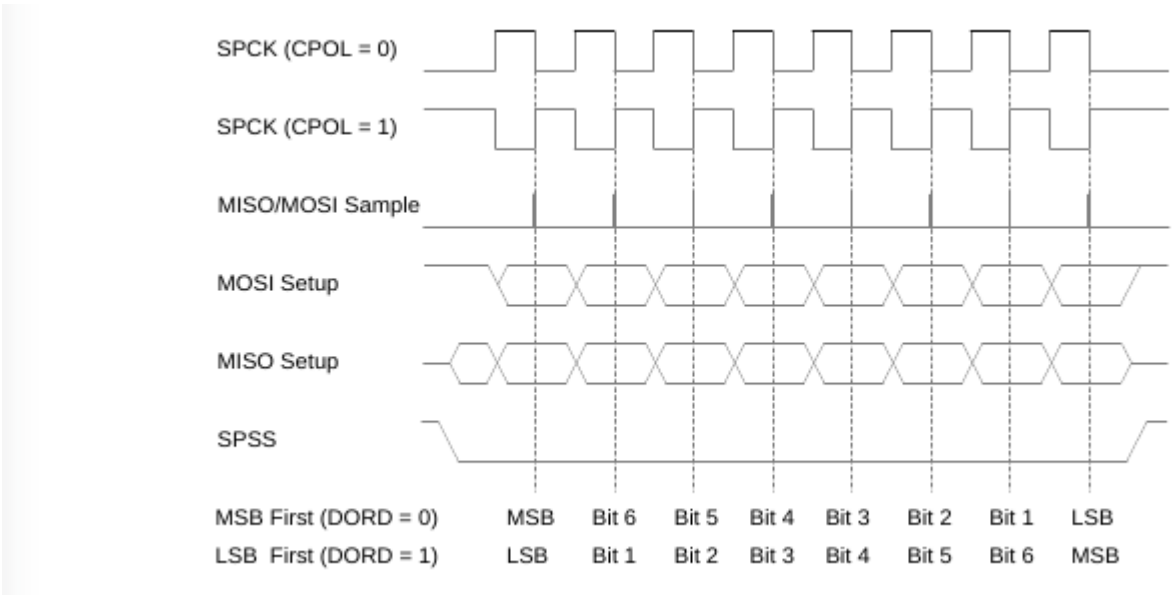
| CPOL | CPHA | Start edge | End edge | SPI mode |
|---|---|---|---|---|
| 0 | 0 | Sampling (rising edge) | Set (falling edge) | 0 |
| 0 | 1 | Set (rising edge) | Sample (falling edge) | 1 |
| 1 | 0 | Sample (falling edge) | Set (rising edge) | 2 |
| 1 | 1 | Set (falling edge) | Sampling (rising edge) | 3 |

When CPHA = 0, the clock edges for data sampling and setting are as shown in the following figure:



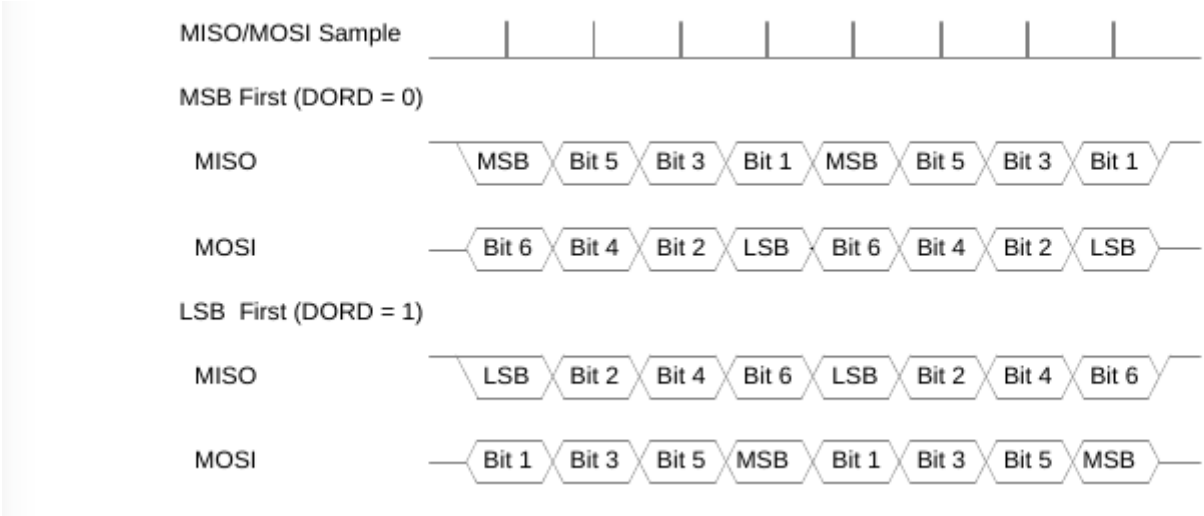

SPI data transfer mode when CPHA is "0"

When CPHA = 1, the clock edges for data sampling and setting are as shown below:




SPI data transfer mode when CPHA is "1"

In the two-wire mode, MISO and MISO are both used as the input of the host, and the data sampling time is still determined by the data transmission mode. The sampling method is shown in the following figure:




SPI data sampling mode when DUAL is "1" in master mode

## SPSS pin function

When configured as a slave, the slave select signal SPSS pin is always used as an input. When the SPSS pin is kept low, the SPI interface is activated, the MISO pin becomes an output pin (the software performs the corresponding port configuration), and other pins are all inputs. When the SPSS pin is held high, the SPI module is reset and no longer receives data. The SPSS pins are useful for packet/byte synchronization, synchronizing the slave's bit counter with the master's clock generator. When SPSS is pulled high, the SPI slave immediately resets the receive and transmit logic and discards the incomplete data in the shift register.

When configured as a host, user software can determine the orientation of the SPSS pins.

If SPSS is configured as an output, it can be used to drive the slave's SPSS pins. If SPSS is configured as an input, it must be kept high to ensure normal operation of the host. When it is configured as a master and the SPSS pin is an input, and the external circuit pulls down the SPSS pin, the SPI module will consider it as another master and select itself as a slave and begin to transmit data. In order to prevent bus conflict, the SPI module will perform the following actions:

1. Clear the MSTR flag in the SPCR register, switch to slave, so that MOSI and SPCK become inputs;
2. Set the SPIF flag in the SPSR register to generate an SPI interrupt if the interrupt is enabled.

Therefore, the interrupt service routine should check whether the MSTR flag is "1" when using the interrupt method to handle the data transfer of the SPI master, and there is a possibility that SPSS is pulled low. If cleared, software must set it to re-enable SPI master mode.

## SPI initialization

The SPI must be initialized before communication. The initialization process usually includes the selection of the master-slave operation, the setting of the data transmission mode, the selection of

the bit rate, and the direction control of each pin. Among them, the control of the pin direction under the master and slave operation is different, as shown in the following table:

Pin Direction Control

| Pin | Orientation in host mode | Orientation in slave mode |
|---|---|---|
| MOSI | User software defined | IN/enter |
| MISO | IN/enter | User software defined |
| SPCK | User software defined | IN/enter |
| SPSS | User software defined | IN/enter |

## SPI master initialization

The initialization process of SPI master mode is as follows:

1. Set the MSTR flag, set the bit rate selection control flag, data transfer mode, data transfer order, interrupt enable or not, and dual line enable or not;
2. Set the MOSI and SPCK pins as outputs;
3. Set the SPE flag.

In master mode, when you do not want the SPI module to be selected as a slave by another master, you can set the SPSS pin as output.

## SPI slave initialization

The SPI slave mode initialization process is as follows:

1. Clear the MSTR flag, set the data transfer mode, data transfer order, and interrupt enable or not;
2. Set MISO pin as output;
3. Set the SPE flag.

## SPI interrupt

The SPI interrupt flag SPIF will be set when one or more of the following events occur:

1. When configured as a host and the SPSS pin is an input, the external circuit pulls down the SPSS pin;
2. When the sending buffer status is full, the software continues to write data to the SPDR register;
3. When the receive buffer status is full;
4. When the data written in the transmit buffer has been sent out, the transmit buffer status is empty.

An SPI interrupt is generated when the SPIF flag is set and both the SPI interrupt enable flag SPIE and the global interrupt enable flag are high. After entering the interrupt service routine, the

hardware will clear SPIF. If the SPIF flag is set by 1 and 2 in the above event, SPIF will be cleared; if the SPIF flag is set by 3 and 4 in the above event, SPIF will not be cleared because the receive Or when the state of the transmit buffer has not changed, the SPIF flag will still be set, and it needs to be cleared by software operation at this time.

In the SPI interrupt service routine, the operation sequence of software clearing the SPIF flag is as follows:

1. Read the state of the SPIF flag, if it is low, it means that the SPIF flag has been cleared by hardware, no need to clear it again by software; if it is high, continue the operation;
2. Read the SPFR register, if the RDFULL flag is high, it indicates that the current receive buffer status is full, read the SPDR register to obtain the received data, the RDFULL flag will become low, the software can continue to read the SPDR register to obtain the received data until RDEMPT flag is high;
3. Read the SPFR register, if the RDFULL flag is low and the WREMPT flag is high, it indicates that the current receive buffer state is not full, but the transmit buffer state is empty, the software can read the SPDR register to get the received data, until the RDEMPT flag is high;
4. After the software obtains the received data, it executes clearing the SPIF flag. Since the SPIF flag is a read-only bit, the SPIF flag cannot be cleared directly. Instead, it is necessary to read the SPSR register first, and then access the SPDR (read or write the SPDR register) to clear the SPIF flag.

## Register definition

SPI Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| SPCR | 0x4C | 0x00 | SPI Control Register |
| SPSR | 0x4D | 0x00 | SPI Status Register |
| SPDR | 0x4E | 0x00 | SPI data register |
| SDFR | 0x39 | 0x00 | SPI buffer register |

### *SPCR – SPI Control Register*

| SPCR – SPI Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x4C | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | SPIE | SPI interrupt enable flag.<br>When the SPIE flag is set to "1", the SPI interrupt is enabled. An SPI interrupt is generated when the SPIF flag in the SPSR register is set and the global interrupt is enabled.<br>When the SPIE flag is set to "0", the SPI interrupt is disabled. |
| 6 | SPE | SPI enable flag.<br>When the SPE flag is set to "1", the SPI module is enabled. SPE must be set before any SPI operation.<br>When setting the SPE flag to "0", the SPI module is disabled. |
| 5 | DORD | Data order control bits.<br>When the DORD flag is set to "1", the LSB of the data is transmitted first.<br>When the DORD flag is set to '0', the MSB of the data is sent first. |
| 4 | MSTR | Master slave select control bits.<br>When the MSTR flag is set to "1", the master mode is selected.<br>When the MSTR flag is set to "0", the slave mode is selected.<br>In the host mode, when the SPSS pin is configured as an input and pulled low, the MSTR flag will be cleared, the SPIF in the SPSR register will be set, and the user must reset the MSTR to enter the host mode. |
| 3 | CPOL | Clock polarity control flag.<br>When the CPOL flag is set to "1", SPCK is high in idle state.<br>When the CPOL flag is set to "0", SPCK is low in idle state. |

| CPOL | Start edge | End edge |
|---|---|---|
| 0 | Rising edge | Falling edge |
| 1 | Falling edge | Rising edge |

<table>
<tr><td>2</td><td>CPHA</td><td>Clock phase control bits.<br>When the CPHA flag is set to "1", the starting edge sets data, and the ending edge samples data.<br>When the CPHA flag is set to "0", the data is sampled at the start edge, and the data is set at the end edge.
<table>
<tr><th>CPHA</th><th>Start edge</th><th>End edge</th></tr>
<tr><td>0</td><td>Sampling</td><td>Set up</td></tr>
<tr><td>1</td><td>Set up</td><td>Sampling</td></tr>
</table></td></tr>
<tr><td>1</td><td>SPR1</td><td>Clock rate select flag 1.<br>SPR1 and SPR0 are used to select the clock rate for SPI transfers. For the specific control method, see the relationship table between SPCK and system clock.</td></tr>
<tr><td>0</td><td>SPR0</td><td>Clock rate select flag 0.<br>SPR1 and SPR0 are used to select the clock rate for SPI transfers. For the specific control method, see the relationship table between SPCK and system clock.</td></tr>
</table>

### *SPSR – SPI Status Register*

| SPSR – SPI Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x4D | | | | | Default: 0x00 | | | |
| | | | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | SPIF | WCOL | - | - | - | DUAL | - | SPI2X |
| R/W | R | R | R | R | R | R/W | R | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Description |
|---|---|---|
| 7 | SPIF | SPI interrupt flag flag.<br>The SPIF flag is set after the serial transmission is completed. In the host mode, when the SPSS pin is configured as an input and pulled low, the SPIF will also be set. If the SPIE flag in the SPCR register and the global interrupt enable flag are both set, an SPI interrupt will be generated. The SPIF flag is automatically cleared after entering the interrupt service routine, or by reading the SPSR register first and then accessing the SPDR register to clear the SPIF flag. |
| 6 | WCOL | Write conflict flag.<br>Writing to the SPDR register during a data transfer will set the WCOL flag. The WCOL flag can be cleared by reading the SPSR register before accessing the SPDR register. |
| 5 | - | Reserve |
| 4 | - | Reserve |
| 3 | - | Reserve |

| | | |
|---|---|---|
| 2 | DUAL | Two-wire mode control bits.<br>When the DUAL flag is set to "1", the SPI two-wire transfer mode is enabled.<br>When the DUAL flag is set to "0", the SPI two-wire transfer mode is disabled.<br>The two-wire transmission mode is only valid in the SPI host mode. Both MISO and MOSI are used as the host data input. For the data transmission method, see the description in the host two-wire reception and data mode chapters. |
| 1 | - | Reserve |
| 0 | SPI2X | SPI speed control flag.<br>When the SPI2X flag is set to "1", the transfer speed of SPI is doubled.<br>When the SPI2X flag is set to "0", the transfer speed of SPI is not doubled.<br>For the specific control method, see the relationship table between SPCK and system clock. |

The table below shows the relationship between SPCK and system clock.

The relationship between SPCK and system clock

| SPI2X | SPR1 | SPR0 | Frequency of SPCK |
|---|---|---|---|
| 0 | 0 | 0 | fsys/4 |
| 0 | 0 | 1 | fsys/16 |
| 0 | 1 | 0 | fsys/64 |
| 0 | 1 | 1 | fsys/128 |
| 1 | 0 | 0 | fsys/2 |
| 1 | 0 | 1 | fsys/8 |
| 1 | 1 | 0 | fsys/32 |
| 1 | 1 | 1 | fsys/64 |

### *SPDR – SPI Data Register*

| SPDR – SPI Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x4E | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | SPDR7 | SPDR6 | SPDR5 | SPDR4 | SPDR3 | SPDR2 | SPDR1 | SPDR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | SPDR | Data sent and received by SPI.<br>SPI transmit data and receive data share the SPI data register SPDR. Writing data to SPDR means writing to the transmit data shift register, and reading data from SPDR means reading the receive data buffer. |

### *SPFR – SPI Buffer Register*

| SPFR – SPI Buffer Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x39 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | RDFULl | RDEMPT | RDPTR1 | RDPTR0 | WRFULL | WREMPT | WRPTR1 | WRPTR0 |
| R/W | R | R/W | R | R | R | R/W | R | R |

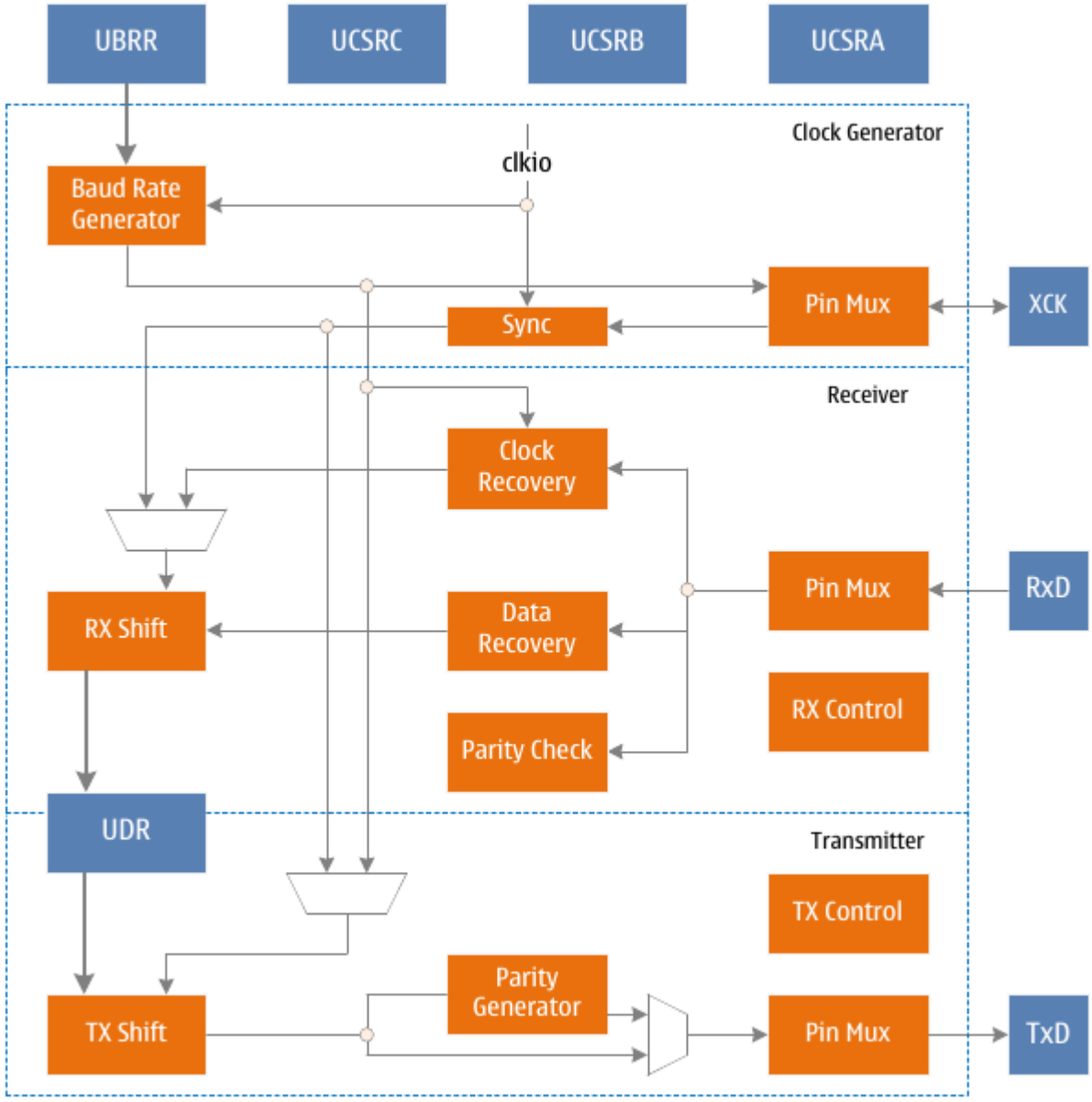
| Bit | Name | Description |
|---|---|---|
| 7 | RDFULL | Receive buffer full flag.<br>When the data in the receive buffer reaches four bytes, the RDFULL flag is high, indicating that the receive buffer is full, and the interrupt flag is set. If the software does not read the data in the receive buffer in time, when the data is received again, the receive buffer overflows, and the previous data will be overwritten by the new data.<br>When the data in the receive buffer is less than four bytes, the RDFULL flag is low, indicating that the receive buffer is not full and data can be received.<br>When the RDEMPT flag and the WREMPT flag are set at the same time, the receive and transmit buffer addresses and the SPI shift register pointer are reset to zero and the RDFULL flag is low. |
| 6 | RDEMPT | Receive buffer empty flag.<br>When no data has been received, the RDEMPT flag is high, indicating that the receive buffer is empty. When there is received data, it will be stored in the receive buffer. The RDEMPT flag is low, indicating that the receive buffer is not empty. At this time, the MCU can read the data in the receive buffer by accessing the SPDR register. To ensure that the received data is not lost, software can read the data in the receive buffer when the receive buffer is not empty, that is, when the RDEMPT flag is low.<br>When the RDEMPT flag is set (write 1), the receive buffer address will return to zero.<br>When the RDEMPT flag and the WREMPT flag are set at the same time, the receive and transmit buffer addresses and the SPI shift register pointer are reset to zero and the RDEMPT flag is high. |
| 5 | RDPTR1 | Receive buffer address high order. |
| 4 | RDPTR0 | Receive buffer address low order.<br>When the SPDR register is read, the MCU will read the received data from the receive buffer, and the receive buffer address will be accumulated at the same time.<br>When the RDEMPT flag is set (write 1), the receive buffer address will return to zero. |
| 3 | WRFULL | Transmit buffer full flag.<br>When the data in the transmit buffer reaches four bytes, the WRFULL flag is high, indicating that the transmit buffer is full.<br>When the transmit buffer contains less than four bytes of data, the WRFULL flag is low, indicating that the transmit buffer is not full. If you |

| | | |
|---|---|---|
| | | want to increase the transmission speed, the software can write data when the transmit buffer is not full, that is, when the WRFULL flag is low, the SPI controller will send the data in turn. |
| 2 | WREMPT | Transmit buffer empty flag.<br>When the data written to the transmit buffer has been sent, the WREMPT flag is high, indicating that the transmit buffer is empty, and the interrupt flag SPIF will be set at the same time.<br>After the SPDR register is written, the address of the transmit buffer will be accumulated, and when the data written in the transmit buffer is not all sent, there is at least one byte of data in the receive buffer, and the WREMPT flag is low, indicating that the transmit buffer is non empty.<br>When the WREMPT flag is set (write 1), the transmit buffer address will be reset to zero.<br>When the RDEMPT flag and the WREMPT flag are set at the same time, the receive and transmit buffer addresses and the SPI shift register pointer are reset to zero and the WREMPT flag is high. |
| 1 | WRPTR1 | Transmit buffer address high order. |
| 0 | WRPTR0 | Transmit buffer address low order.<br>When the SPDR register is written, the data in the SPDR will be written to the transmit buffer, and the transmit buffer address will be accumulated at the same time.<br>When the WREMPT flag is set (write 1), the transmit buffer address will be reset to zero. |

# USART0 - Universal Synchronous/Asynchronous Serial Transceiver

- Full-duplex operation (separate serial receive and transmit registers)
- Asynchronous or synchronous operation
- Master or Slave Operation
- High precision baud rate generator
- Supports 5, 6, 7, 8, or 9 data bits and 1, or 2 stop bits
- Hardware-supported parity generation and checking mechanism
- Data overrun detection
- Frame Error Detection
- Noise filtering, including false start bit detection and digital low-pass filtering
- Three separate interrupts: Transmit End Interrupt, Transmit Data Register Empty Interrupt and Receive End Interrupt
- Multiprocessor Communication Mode
- Double-speed asynchronous communication mode

## Overview




USART Structure Diagram

USART mainly includes three parts: clock generator, transmitter and receiver. Control and status registers are shared by these three sections. The clock generator consists of a baud rate generator and synchronization logic for an externally input clock in synchronous slave mode of operation. The XCK pin is only used in synchronous transfer mode. The transmitter includes a write data buffer, serial shift register, parity generator, and control logic required to handle different frame formats. The write data buffer allows data to be sent continuously without introducing delays between data frames. The receiver has a clock and data recovery unit for asynchronous data reception. In addition to the recovery unit, the receiver also includes parity, control logic, serial shift registers and a two-stage receive buffer UDR. The receiver supports the same frame format as the transmitter and can detect framing errors, data overruns and parity errors.

## Clock generation

The clock generation logic generates the base clock for the transmitter and receiver. The USART supports 4 clock modes: normal asynchronous mode, double-speed asynchronous mode, master synchronous mode, and slave synchronous mode. The UMSEL bits of the USCRC are used to select synchronous or asynchronous mode. The U2X flag of USCRA controls the double-speed enable in asynchronous mode. The XCK pin's data direction register (multiplexed with IO), which is valid only in synchronous mode, determines whether the clock source is generated internally (master mode) or externally (slave mode).

## Baud rate generator

The baud rate register UBRR and the descending counter are connected together as a programmable prescaler or baud rate generator for the USART. The descending counter works under the system clock (fsys) and will automatically load the value of the UBRR register when it counts to zero or when the UBRRL register is written. When the count reaches zero, a clock is generated, which is used as the output clock of the baud rate generator with a frequency of fsys/(UBRR+1).

The table below gives the formulas for calculating the baud rate (bits per second) and the UBRR value for various operating modes.

| Operating mode | Baud rate calc. formula (1) | UBRR value calc. formula |
|---|---|---|
| Asynchronous normal mode | BAUD = fsys/(16*(UBRR+1)) | UBRR = fsys/(16*BAUD) - 1 |
| Asynchronous double-speed mode | BAUD = fsys/(8*(UBRR+1)) | UBRR = fsys/(8*BAUD) - 1 |
| Synchronous host mode | BAUD = fsys/(2*(UBRR+1)) | UBRR = fsys/(2*BAUD) - 1 |

Note:

1. Baud rate is defined as bits per second (bps);

2. BUAD is the baud rate, fsys is the system clock, and UBRR is the combined value of the baud rate registers UBRRH and UBRRL.

## Double-speed working mode

The transmission rate can be doubled by setting the U2X flag of the UCSRA register. This flag is only valid in the asynchronous working mode, and it is set to "0" in the synchronous working mode.

Setting this flag will halve the division value of the baud rate divider, effectively doubling the transfer rate for asynchronous communications. In this case, the receiver uses only half the number of samples for sampling and clock recovery of the data, so a more precise baud rate setting and system clock are required. The transmitter is unchanged.

## External clock

The synchronous slave mode of operation is driven by an external clock. The external clock is used by the transmitter and the receiver after passing through the synchronization register and edge detector. This process will introduce a delay of two system clocks, so the maximum clock frequency of the external XCK is limited by the following formula:

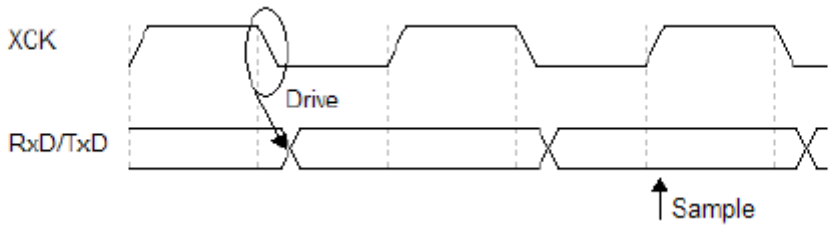
$$f_{XCK} < fsys/4$$

It should be noted that fsys is determined by the stability of the system clock. In order to prevent data loss due to frequency drift, it is recommended to reserve sufficient margin.

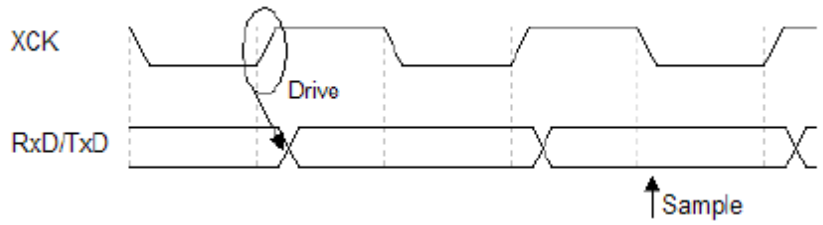
## Synchronized Clock Operation

In synchronous mode, the XCK pin is used for clock input (slave mode) or clock output (master mode). The basic law of the relationship between the edge of the clock and the data sampling and data change is: the clock edge used for sampling the data input terminal (RxD) is opposite to the clock edge used for the data output terminal change.

UCPOL = 1




UCPOL = 0




XCK Timing in Synchronous Mode

As shown in the figure above, when the UCPOL value is "1", the data output is changed on the falling edge of XCK, and data sampling is performed on the rising edge of XCK; when the UCPOL value is "0", the data output is changed on the rising edge of XCK, Data is sampled on the falling edge of XCK.

## Frame format

A serial data frame consists of data words plus synchronization bits (start and stop bits) and parity bits for error correction.

USART accepts the following 30 combinations of data frame formats:

- 1 start bit
- 5, 6, 7, 8 or 9 data bits
- No parity bit, odd parity bit or even parity bit
- 1 or 2 stop bits

A data frame begins with a start bit, followed by the least significant bit of the data word, followed by other data bits, and ends with the most significant bit of the data word. Up to 9 bits of data are successfully transmitted. If parity is enabled, the parity bit will follow the data word, followed by the stop bit. When a complete data frame is transmitted, the next new data frame can be transmitted immediately, or the transmission line can be placed in an idle (high level) state. The figure below shows a possible data frame structure, the bits in square brackets are optional.


USART frame structure

Notes:

1. **IDLE** No data transmission on the communication line (RxD or TxD), line must be high when it’s idle.
2. **St** start bit, always low.
3. **0-8** data bits.
4. **P** parity bit, odd parity or even parity.
5. **Sp** stop bit, always high.

The structure of the data frame is set by UCSZ[2:0], UPM[1:0] and USBS in the UCSRB and UCSRC registers. Receive and send use the same settings. Any changes to settings may disrupt ongoing data transfers. Among them, UCSZ[2:0] determines the number of data bits of the data frame, UPM[1:0] is used to enable and determine the type of checksum, and USBS sets the frame to have one or two end bits. The receiver ignores the second stop bit, so framing errors are only detected when the first stop bit is '0'.

## Parity Bit Calculation

The calculation of the parity bit is to perform an exclusive OR operation on each bit of data. If odd parity is selected, the result needs to be negated after XOR operation.

The relationship between parity bits and data bits is as follows:

$$P_{even} = dn\text{-}1 \oplus ... \oplus d3 \oplus d2 \oplus d1 \oplus d0 \oplus 0$$

$$P_{odd} = dn\text{-}1 \oplus ... \oplus d3 \oplus d2 \oplus d1 \oplus d0 \oplus 1$$

Note:

1) $P_{even}$ even parity result

2) $P_{odd}$ odd check result

3) $d_n$ $n^{th}$ data bit

## USART initialization

The USART must be initialized before communication. The initialization process usually includes setting the baud rate, setting the frame structure, and enabling the receiver or transmitter as needed. For interrupt-driven USART operation, the global interrupt flag should be cleared and all interrupts of the USART should be disabled during initialization.

When performing re-initialization such as changing the baud rate or frame structure, it must be ensured that no data is transmitted. The TXC flag can be used to detect whether the transmitter has completed all transmissions, and the RXC flag can be used to detect whether there is still data in the receive buffer that has not been read. If the TXC flag is used for this purpose, the TXC flag must be cleared before each data transmission (before writing to the UDR register).

## Transmitter / Sender

Setting the TXEN flag in the UCSRB register enables data transmission from the USART. After enabling, the general IO function of the TxD pin is replaced by the USART function and becomes the serial output of the transmitter. Before sending data, set the baud rate, working mode and frame format. If the synchronous transmission mode is used, the clock signal applied to the XCK pin is the clock for data transmission.

## Send frames 5 to 8 as data

Load the data to be sent into the transmit buffer to start data transmission. The CPU loads data by writing to the UDR register. When the transmit shift register can send a new frame of data, the data in the buffer will be transferred to the shift register.

When the shift register is idle (no data transfer in progress), or the last stop bit of the previous frame of data has been sent, it will be loaded with new data. Once the shift register is loaded with new data, it will transmit a complete frame according to the established settings.

## Send a frame with 9 bits of data

If a frame of 9-bit data is sent, the 9th bit of the data should be written into the TXB8 flag of the UCSRB register first, and then the lower 8-bit data should be written into the transmit data register UDR. The ninth bit of data is used to represent the address frame in multi-computer communication, and can be used for protocol processing in synchronous communication.

## Send parity

The parity generation circuit generates corresponding parity bits for the serial data frame. When the parity bit is enabled (UPM1 = 1), the transmit control logic inserts a parity bit between the last bit and the first stop bit of the data word.

## Sending Flags and Interrupt Handling

The USART transmitter has two flag bits: the USART data register empty flag UDRE and the transmission end flag TXC, both of which can generate interrupts.

The data register empty flag UDRE is used to indicate whether the transmit buffer can write a new data. This flag is set to "1" when the transmit buffer is empty and set to "0" when it is full. When the UDRE flag is "1", the CPU can write new data to the data register UDR, and vice versa.

When the data register empty interrupt enable flag UDRIE in the UCSRB register is "1", as long as UDRE is set (and the global interrupt is enabled), a USART data register empty interrupt request will be generated. Writing to register UDR will clear UDRE.

When using the interrupt mode to transmit data, a new data must be written to the UDR in the data register empty interrupt service routine to clear UDRE, or the data register empty interrupt must be disabled. Otherwise, once the interrupt service routine ends, a new interrupt will be generated again.

When the entire data frame is shifted out of the transmit shift register and there is no new data in the transmit register, the transmit end flag TXC will be set. When the transmission end interrupt enable flag TXCIE on UCSRB (and the global interrupt enable) is set to "1", the USART transmission end interrupt will be executed as the TXC flag is set. Once the interrupt service routine is entered, the TXC flag will be automatically cleared, and the CPU can also write "1" to this flag to clear it.

## Disable transmitter / Sender

When TXEN is cleared, the transmitter can be disabled only after all the data are sent, that is, there is no data to be transmitted in the transmit shift register and transmit buffer register. After the transmitter is disabled, the TxD pin resumes its general purpose IO function.

## Receiver

The USART receiver is enabled by setting the receive enable flag (RXEN) of the UCSRB register. After enabling, the general IO function of the RxD pin is replaced by the USART function and becomes the serial input port of the receiver. Before receiving data, first set the baud rate, operation mode and frame format. If the synchronous receive mode is used, the clock on the XCK pin is used as the transmit clock.

## Receive frames with 5 to 8 bits of data

Once the receiver detects a valid start bit, it begins to accept data. Each bit of data after the start bit will be received at the set baud rate or XCK clock until the first stop bit of a frame of data is received, and the second stop bit will be ignored by the receiver. Each bit of data received is sent into the receive shift register. After receiving the first stop bit, the receiver sets the received data completion flag RXC flag in the UCSRA register, and transfers the complete data frame in the shift register. To the receive buffer, the CPU can obtain the received data by reading the UDR register.

## Receive a frame of 9-bit data

If the data frame of 9-bit data is set, before reading the lower 8-bit data from the UDR, the RXB8 flag of the register UCSRB must be read first to obtain the 9th-bit data. This rule also applies to the status flags FE, DOR and PE. Reading a UDR location changes the state of the receive buffer, which in turn changes the TXB8, FE, DOR, and PE bits that are also stored in the buffer.

## Receive end flag and interrupt handling

The USART receiver has a flag: the reception end flag RXC, which is used to indicate whether there is unread data in the receive buffer. When there is unread data in the receive buffer, this flag is "1", otherwise it is "0". If the receiver is disabled, the receive buffer will be flushed and RXC will be cleared.

After setting the receive end interrupt enable flag RXCIE of UCSRB, as long as the RXC flag is set (and the global interrupt is enabled), the USART receive end interrupt will be generated. When using the interrupt mode to receive data, the interrupt service routine must read the data from the UDR to clear the RXC flag when the data is received, otherwise a new interrupt will be generated as soon as the interrupt handler ends.

## Receive error flag

The USART receiver has three error flags: framing error FE, data overflow DOR and parity error PE. They are both located in the UCSRA register. The error flag is stored in the receive buffer along with the data frame. All error flags cannot generate interrupts.

The frame error flag FE indicates the status of the first stop bit of the next readable frame stored in the receive buffer. If the stop bit is correct (the value is "1"), the FE flag is "0", otherwise the FE flag is "1". This flag can be used to detect loss of synchronization, transmission interruption, and also for protocol processing.

The data overflow flag DOR indicates that data was lost due to a full receive buffer. When the receive buffer is full and there is data in the receive shift register, if a new start bit is detected at this time, a data overflow occurs. The DOR flag is set to indicate that one or more data frames were lost between the last UDR read and the next UDR read. When the data frame is successfully transferred from the shift register to the receive buffer, the DOR flag is cleared.

The parity error flag PE indicates that the next frame of data in the receive buffer has a parity error during reception. If parity is not enabled, PE is cleared.

## Parity checker

Setting the parity mode flag UPM1 will start the parity checker. The parity mode (even or odd) is determined by UPM0. When the parity check is enabled, the checker will calculate the parity of the incoming data and compare the result with the parity bits of the data frame.

The check result will be stored in the receive buffer along with the data and stop bits. The CPU checks the received frame for parity errors by reading the PE flag. If the next data read from the receive buffer has a parity error and parity is enabled, UPE is set and remains valid until the receive buffer UDR is read.

## Disable the receiver

In contrast to the transmitter, disabling the receiver works instantly. Data being received will be lost. After the receiver is disabled (RXEN is cleared), the receiver will no longer occupy the RxD pin and the receive buffer will be refreshed.

## Asynchronous data reception

The USART has a clock recovery unit and data recovery unit to handle asynchronous data reception. The clock recovery logic is used to synchronize the asynchronous serial data input from the RxD pin with the internal baud rate clock. Data recovery logic is used to collect the data and filter each bit of incoming data through a low-pass filter, thereby improving the receiver's immunity to interference. The working range of asynchronous reception depends on the precision of the internal baud rate clock, the rate of frame input and the number of bits of data contained in a frame.

## Asynchronous work scope

The receiver operating range depends on the mismatch between the received data rate and the internal baud rate. If the transmitter transmits data at a bit rate that is too fast or too slow, or if the baud rate generated internally by the receiver does not have the same frequency, then the receiver cannot synchronize to the start bit. In order to ensure that the receiver does not miss the sampling of the start bit of the next frame, the data input rate and the internal receiver baud rate cannot be too different, and the ratio between them is used to describe the error range of the baud rate. The following two tables give the maximum allowable baud rate error range in normal mode and double-speed mode, respectively.

Maximum receiver baud rate error range in normal mode

| Frame length (data + parity bits) | Maximum error range (%) | Recommended error range (%) |
|---|---|---|
| 5 | +6.7/-6.8 | ±3.0 |
| 6 | +5.8/-5.9 | ±2.5 |
| 7 | +5.1/-5.2 | ±2.0 |
| 8 | +4.6/-4.5 | ±3.0 |
| 9 | +4.1/-4.2 | ±1.5 |
| 10 | +3.8/-3.8 | ±1.5 |

Maximum receiver baud rate error range in double speed mode

| Frame length (data + parity bits) | Maximum error range (%) | Recommended error range (%) |
|---|---|---|
| 5 | +5.7/-5.9 | ±2.5 |
| 6 | +4.9/-5.1 | ±2.0 |
| 7 | +4.4/-4.5 | ±1.5 |
| 8 | +3.9/-4.0 | ±1.5 |
| 9 | +3.5/-3.6 | ±1.0 |
| 10 | +3.2/-3.3 | ±1.0 |

It can be seen from the table that the baud rate is allowed to have a wider range of variation in normal mode. The above recommended baud rate error ranges assume that the receiver and transmitter contribute equally to the maximum total error. There are two possible causes of receiver baud rate errors. First, the stability of the receiver system clock is related to operating voltage and temperature. This problem generally does not occur when using a crystal oscillator to generate the system clock, but when using the internal oscillator, the system clock may be skewed. The second reason is that the baud rate generator cannot necessarily divide the system clock to get the exact desired baud rate. At this point, the value of UBRR can be adjusted so that the error is acceptable.

## Baud rate setting and introducing errors

For standard crystal oscillator and resonator frequencies, the actual communication baud rate in asynchronous mode can be obtained through the baud rate calculation formula, and the error between it and the commonly used communication baud rate can be calculated by the following formula:

$$Error[\%] = (Baud_{real}/Baud - 1)*100\%$$

Among them, Baud is the commonly used communication baud rate, Baudreal is the baud rate calculated by the calculation formula, and the baud rate error can be obtained by bringing the baud rate calculation formula into the value of the system clock fsys and the baud rate register BRR. The relationship is as follows:

Normal mode:

$$Error[\%] = (f_{sys}/(16*(UBRR+1))/Baud - 1)*100\%$$

Double speed mode:

$$Error[\%] = (f_{sys}/(8*(UBRR+1))/Baud - 1)*100\%$$

When the clock error on both sides of the communication is not considered, that is, when the system clock fsys is the standard clock, the relationship between the baud rate error UBRR value can be obtained. The following table shows the baud rate error under different UBRR value settings under 16MHz system clock.

Error caused by setting UBRR value at 16MHz system clock

| baud rate (bps) | fsys = 16.000MHz | | | |
|---|---|---|---|---|
| | Normal mode (U2X = 0) | | Double Speed Mode (U2X = 1) | |
| | UBRR | Error | UBRR | Error |
| 2400 | 416 | -0.1% | 832 | 0.0% |
| 4800 | 207 | 0.2% | 416 | -0.1% |
| 9600 | 103 | 0.2% | 207 | 0.2% |
| 14.4K | 68 | 0.6% | 138 | -0.1% |
| 19.2K | 51 | 0.2% | 103 | 0.2% |
| 28.8K | 34 | -0.8% | 68 | 0.6% |
| 38.4K | 25 | 2.1% | 34 | -0.8% |
| 57.6K | 16 | 0.2% | 51 | 0.2% |
| 76.8K | 12 | 0.2% | 25 | 0.2% |
| 115.2K | 8 | -3.5% | 16 | 2.1% |
| 230.4K | 3 | 8.5% | 8 | -3.5% |
| 250K | 3 | 0% | 7 | 0% |
| 0.5M | 1 | 0% | 3 | 0% |
| 1M | 0 | 0% | 1 | 0% |

## Multiprocessor Communication Mode

Setting the Multiprocessor Communication Mode (MPCM) flag of UCSRA enables filtering of data frames received by the USART receiver. Frames without address information will be ignored and will not be stored in the receive buffer. In a multiprocessor system, each processor communicates through the same serial bus, and this filtering effectively reduces the number of data frames that need to be processed by the CPU. The setting of the MPCM flag does not affect the operation of the transmitter, but it is used differently in systems with multiprocessor communication.

If the length of the data frame received by the receiver is 5 to 8 bits, the first stop bit will be used to indicate whether the current frame contains data or address information. If the length of the data frame received by the receiver is 9 bits, then the 9th bit determines whether it is data or address information. If the frame type flag is "1", then this is an address frame, otherwise it is a data frame.

In multiprocessor communication mode, multiple slave processors are allowed to receive data from one master processor. The first thing to do is to determine which slave processor is being addressed by decoding the address frame. The addressed slave will receive subsequent data normally, while other slaves will ignore these data frames until the next address frame is received.

For a host processor, it can use the 9-bit data frame format, and use the 9th bit of data to identify the frame format. In this communication mode, the slave processor must also work in the 9-bit data frame format.

The following are the steps for data exchange in multiprocessor communication mode:

1. All slave processors work in multiprocessor communication mode (set MPCM);
2. The master processor sends an address frame, and all slave processors receive this frame. The RXC flag of the slave processor's UCSRA register is normally set;

3. Each slave reads the contents of the UDR register and decodes the address frame to determine if it is selected. If it is selected, it will clear the MPCM flag of the UCSRA register. If it is not selected, it will keep MPCM as "1" and wait for the arrival of the next address frame;
4. The addressed slave receives all data frames until a new address frame is received. Unaddressed slaves ignore these data frames;
5. After the addressed slave processor receives the last data frame, it sets the MPCM flag and waits for the arrival of the next address frame. Then repeat from the second step.

Frame formats using 5 to 8 bits of data are possible, but impractical because the receiver must switch between using n and n+1 frame formats. This setup makes full-duplex operation difficult because the receiver and transmitter use the same character length setting. If a frame format of 5 to 8 bits of data is used, the transmitter should set two stop bits, the first of which is used to determine the frame type.

## Register definition

### *UCSRA – USART Control and Status Register A*

| UCSRA – USART Control and Status Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC0 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | RXC | TXC | UDRE | FE | DOR | PE | U2X | MPME |
| R/W | R | R/W | R | R | R | R | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | RXC | Receive end flag.<br>When the value of RXC is "1", it indicates that there is unread data in the receive buffer. When the value of RXC is "0", it indicates that there is no unread data in the receive buffer. When the receiver is disabled, the receive buffer is flushed, causing RXC to be cleared. When the receive end interrupt enable flag RXCIE is "1", RXC can be used to generate a receive end interrupt. |
| 6 | TXC | Send end flag.<br>TXC is set when the data in the transmit shift register is sent out and the transmit buffer is empty. TXC is automatically cleared when the transmission end interrupt is executed, and can also be cleared by writing "1" to TXC. When the transmission end interrupt enable flag TXCIE is "1", TXC can be used to generate a transmission end interrupt. |
| 5 | UDRE | Data register empty flag.<br>When UDRE is "1", it indicates that the USART transmit data buffer is empty and data can be written.<br>When UDRE is "0", it indicates that the USART transmit data buffer is full and cannot write data.<br>When the data register empty interrupt enable flag UDRIE is "1", UDRE can be used to generate a data register empty interrupt. |
| 4 | FE | Framing error flag.<br>When FE is "1", it indicates that the data received by the receive data buffer has a frame error, that is, the first stop bit is "0". When FE is "0", it indicates that the data received by the receive data buffer has no frame error, that is, the first stop bit is "1". Once FE is set, it will remain active until the UDR is read. When writing to UCSRA, write "0" to the FE flag. |
| 3 | DOR | Data overflow flag.<br>When the receive buffer is full (contains two data), there is data in the receive shift register, if a new start bit is detected at this time, a data overflow occurs, DOR is set, and it is valid until UDR is read. . When writing to UCSRA, write "0" to the DOR flag. |
| 2 | PE | Parity error flag.<br>When the parity check is enabled (UPM1 is "1"), and the data frame received in the receive buffer has a parity check error, PE is set and is valid until the UDR is read. When writing to UCSRA, the PE flag should |

| | | |
|---|---|---|
| | | be written with "0". |
| 1 | U2X | Double-speed transmission enable flag.<br>When U2X is "1", the transmission rate of asynchronous communication mode is doubled. When U2X is "0", the transmission rate of the asynchronous communication mode is the normal rate.<br>This flag is valid only in asynchronous operation mode, clear this flag when using synchronous operation mode. |
| 0 | MPME | Multiprocessor communication mode enable flag.<br>Setting the MPCM flag will enable multiprocessor communication mode. When MPCM is set, incoming frames received by the USART receiver that do not contain address information will be ignored. Transmitters are not affected by MPCM settings. |

### *UCSRB – USART Control and Status Register B*

| UCSRB – USART Control and Status Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC1 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | RXCIE | TXCIE | UDRIE | RXEN | TXEN | UCSZ2 | RXB8 | TXB8 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | RXCIE | Receive end interrupt enable flag.<br>Set to enable RXC interrupt, clear to disable RXC interrupt. When RXCIE is "1", the global interrupt is enabled, and the USART reception end interrupt can be generated when the RXC of the UCSRA register is "1". |
| 6 | TXCIE | End of transmission interrupt enable flag.<br>Enables the TXC interrupt when set, and disables the TXC interrupt when cleared. When TXCIE is "1", the global interrupt is enabled, and the USART transmission end interrupt can be generated when the TXC of the UCSRA register is "1". |
| 5 | UDRIE | Data register empty interrupt enable flag.<br>Set to enable UDRE interrupt, clear to disable UDRE interrupt. When UDRIE is "1", the global interrupt is enabled, and the USART data register empty interrupt can be generated when the UDRE of the UCSRA register is "1". |
| 4 | RXEN | Receive enable flag.<br>When asserted, the USART receiver is enabled. The general purpose IO function of the RxD pin is replaced by USART receive. Disabling the receiver will flush the receive buffer and deassert the FE, DOR and PE flags. |
| 3 | TXEN | Send enable flag.<br>When asserted, starts the USART transmitter. The general purpose IO function of the TxD pin is replaced by a USART transmit. After TXEN is cleared, only after all data transmission is completed, the USART transmission can be really prohibited. |

| | | |
|---|---|---|
| 2 | UCSZ2 | Character length control flag 2.<br>UCSZ2 is combined with UCSZ1:0 of the UCSRC register to set the number of data bits contained in the data frame. |
| 1 | RXB8 | Receive the 8th bit of data.<br>When the data frame length is 9 bits, RXB8 is the highest bit of the received data. RXB8 must be read before reading the lower 8 bits of data contained in the UDR. |
| 0 | TXB8 | Send the 8th bit of data.<br>When the data frame length is 9 bits, TXB8 is the highest bit of the transmitted data. TXB8 must be written before writing the lower 8 bits of data contained in the UDR. |

### *UCSRC – USART Control and Status Register C*

| UCSRC – USART Control and Status Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC2 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UMSEL1 | UMSEL0 | UPM1 | UPM0 | USBS | UCSZ1 | UCSZ0 | UCPOL |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | UMSEL1:0 | USART mode selection flags.<br>UMSEL selects synchronous or asynchronous operation mode. |

| UMSEL | Method |
|---|---|
| 0 | USART asynchronous mode of operation |
| 1 | USART Synchronous Operation Mode |
| 2 | SPI Slave Operation Mode |
| 3 | SPI master mode of operation |

| Bit | Name | Description |
|---|---|---|
| 5:4 | UPM1:0 | Parity mode selection bits.<br>The upper UPM1 selects to enable or disable parity, and the lower UPM0 selects odd or even. |

| UPM1:0 | Method |
|---|---|
| 0 | Disable parity |
| 1 | Reserve |
| 2 | Enable even parity |
| 3 | enable odd parity |

| Bit | Name | Description |
|---|---|---|
| 3 | USBS | Stop bit selection flag. Select the number of stop bits. |

| USBS | Number of stop bits |
|---|---|
| 0 | 1 |
| 1 | 2 |

| Bit | Name | Description |
|---|---|---|
| 2:1 | UCSZ1:0 | Data frame character length selection bits.<br>UCSZ1:0 combine with UCSZ2 of UCSRB register to set the number of data bits contained in the data frame. |

| UCSZ2:0 | Data frame length |
|---|---|

| | | | |
|---|---|---|---|
| | | 0 | 5 bits |
| | | 1 | 6 bits |
| | | 2 | 7 bits |
| | | 3 | 8 bits |
| | | 4 | Reserve |
| | | 5 | Reserve |
| | | 6 | Reserve |
| | | 7 | 9 bits |
| 0 | UCPOL | Clock polarity select bits.<br>In USART synchronous operation mode, UCPOL sets the relationship between the change of output data and the sampling of input data and the synchronization clock XCK. It has nothing to do with UCPOL in the asynchronous working mode, clear this flag | |
| | | UCPOL | Send data / Receive data |
| | | 0 | Rising edge of XCK / Falling edge of XCK |
| | | 1 | Falling edge of XCK / Rising edge of XCK |

### *UBRRL – USART Baud Rate Register Low Byte*

| UBRRL – USART Baud Rate Register Low Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC4 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UBRR7 | UBRR6 | UBRR5 | UBRR4 | UBRR3 | UBRR2 | UBRR1 | UBRR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | Name | Description | | | | | | |
| 7:0 | UBRR[7:0] | The low byte portion of the USART baud rate register.<br>The USART baud rate register contains two parts, UBRRL and UBRRH, which together are used to set the baud rate of communication. | | | | | | |

### *UBRRH – USART Baud Rate Register High Byte*

| UBRRH – USART Baud Rate Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC5 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | - | UBRR11 | UBRR10 | UBRR9 | UBRR8 |
| R/W | - | - | - | - | R/W | R/W | R/W | R/W |
| Bit | Name | Description | | | | | | |
| 7:4 | - | Reserve | | | | | | |
| 3:0 | UBRR[11:8] | The high byte portion of the USART baud rate register. | | | | | | |

| | | |
|---|---|---|
| | | The USART baud rate register contains two parts, UBRRL and UBRRH, which together are used to set the baud rate of communication.<br>**UBRR = {UBRR[11:8], UBRRL}** |

| Operating mode | Baud rate calculation formula |
|---|---|
| Asynchronous normal mode | BAUD = fsys/(16*(UBRR+1)) |
| Asynchronous double-speed mode | BAUD = fsys/(8*(UBRR+1)) |
| Synchronous host mode | BAUD = fsys/(2*(UBRR+1)) |

### *UDR – USART Data Register*

| UDR – USART Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC6 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UDR7 | UDR6 | UDR5 | UDR4 | UDR3 | UDR2 | UDR1 | UDR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

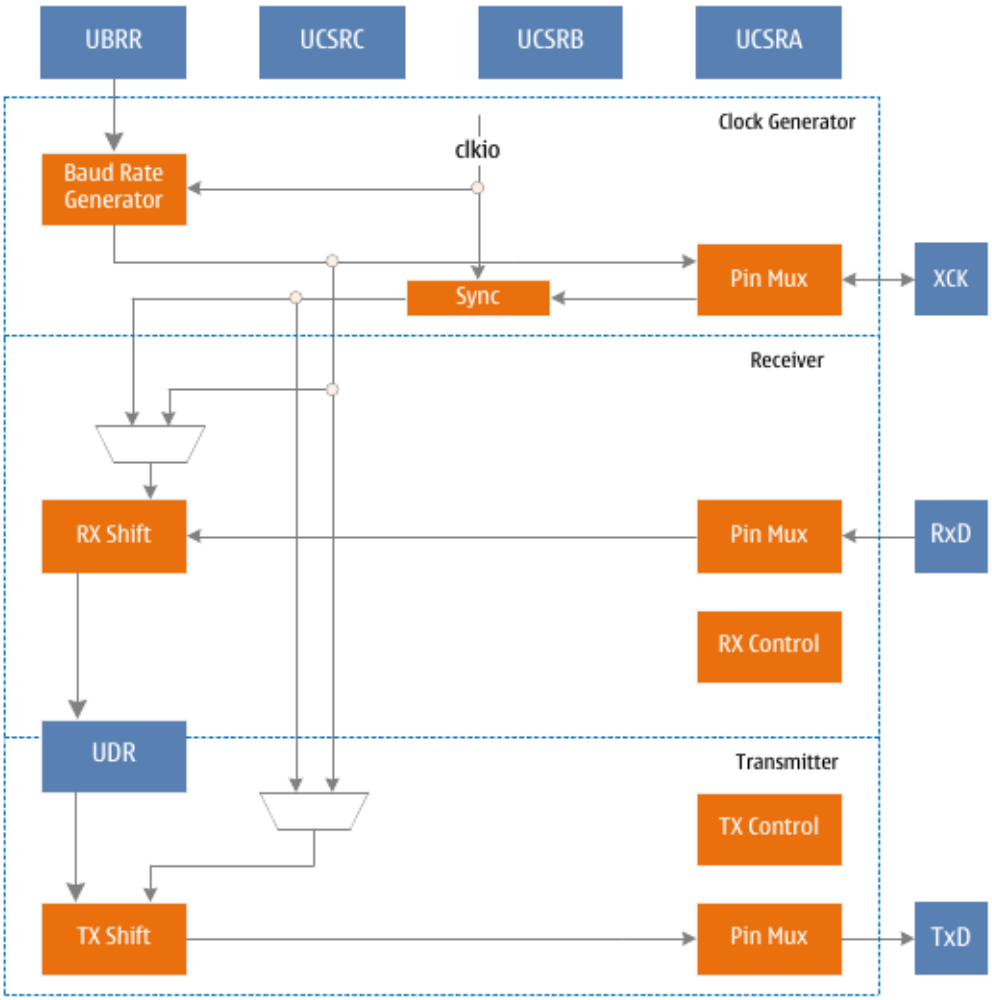
| Bit | Name | Description |
|---|---|---|
| 7:0 | UDR | Data sent and received by the USART.<br>The USART transmit data buffer and receive data buffer share the USART data register UDR. Writing data to the UDR means writing to the transmit data buffer, and reading data from the UDR means reading the receive data buffer.<br>In 5- to 8-bit data frame mode, the unused 9th bit is ignored by the transmitter and set to 0 by the receiver.<br>Only when the UDRE flag of the UCSRA register is "1" can the transmit buffer be written, otherwise the operation of the transmitter will be wrong. When the transmit shift register is empty, the transmitter will load the data in the transmit buffer into the transmit shift register, and then the data will be serially output from the TxD pin.<br>The receive buffer contains a two-level FIFO that changes its state once the receive buffer is read. |

# USART0 - SPI working mode

- Full-duplex operation, three-wire synchronous data transfer
- Master or Slave Operation
- Supports all four operating modes (modes 0, 1, 2 and 3)
- Low-order or high-order transfer first (configurable data transfer order)
- Queue operation (double buffer)
- High Resolution Baud Rate Generator

## Overview

When the UMSEL1 flag of USCRC is set to "1", the SPI working mode is enabled, which is represented by USPI. This SPI module is a three-wire SPI working mode. Compared with the four-wire SPI mode, it lacks the slave selection line, and the other three lines are consistent. The USPI occupies the resources of the USART, including the transmit and receive shift registers and buffers, and the baud rate generator. Parity generation and check logic, data and clock recovery logic are disabled. The addresses of the control and status registers are the same, but the function of the register bits will change as required by the SPI operating mode.




USART in SPI block diagram

## Clock generation

When the SPI works in the master mode, it needs to provide a clock for communication, and the baud rate generator of the USART is multiplexed to generate this clock. This clock is output from the XCK pin, so the data direction register (DDR_XCK) of the XCK pin must be set to "1".

The clock frequency is determined by the following formula:

$$BAUD = f_{sys}/(2*(UBRR+1))$$

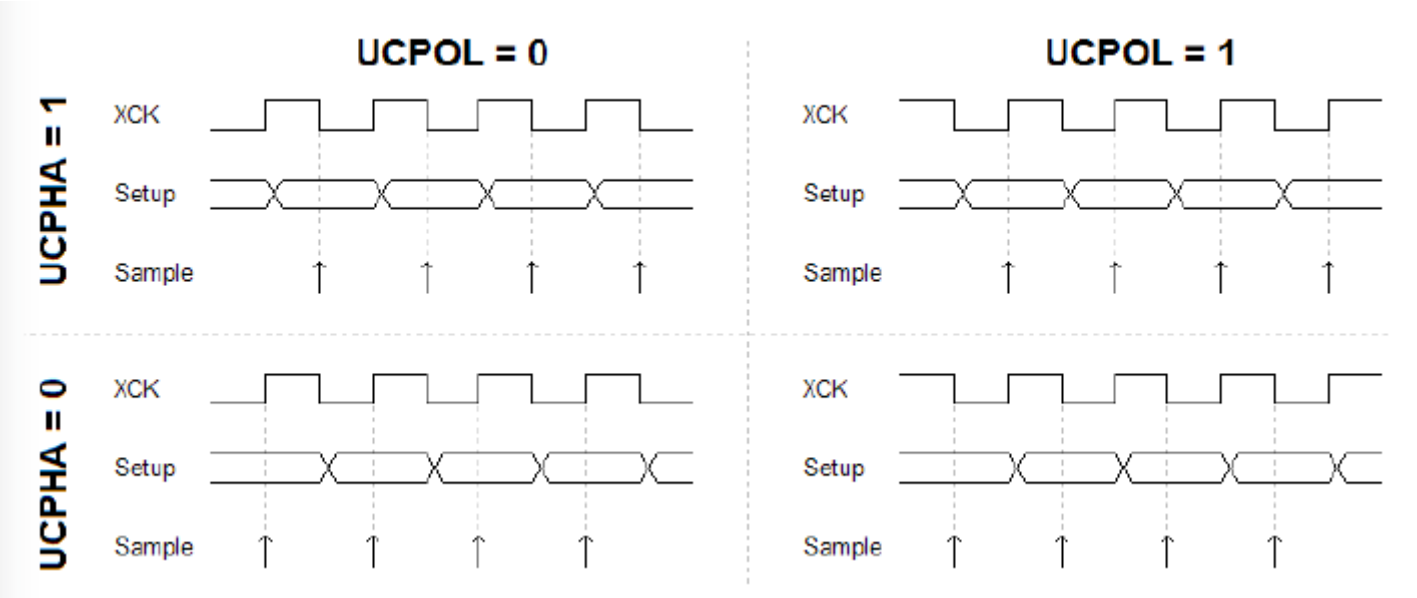
When the SPI works in the slave mode, the communication clock is provided by the external host and input from the XCK pin, so the data direction register (DDR_XCK) of the XCK pin must be set to "0".

## SPI Data Modes and Timing

SPI has four combinations of clock phase and polarity, which are determined by control bits UCPHA and UCPOL. The specific control is shown in the following table and the following figure:

SPI working mode

| SPI mode | UCPOL | UCPHA | Start edge | Stop edge |
|---|---|---|---|---|
| 0 | 0 | 0 | Rising edge sampling | Falling edge setting |
| 1 | 0 | 1 | Rising edge setting | Falling edge sampling |
| 2 | 1 | 0 | Falling edge sampling | Rising edge setting |
| 3 | 1 | 1 | Falling edge setting | Rising edge sampling |




SPI working mode diagram

## Frame format

A serial frame of SPI can start with the lowest or highest bit and end with the highest or lowest bit, with a total of 8 bits of data. After a frame ends, a new frame can be transmitted immediately, and the data line can be pulled up to idle state after the transmission is over.

## Data transmission

The SPI sets the TXEN flag in the UCSRB register to "1" to enable the transmitter, and the TxD pin is occupied by the transmitter to transmit serial output data. The receiver can be disabled at this time.

The SPI sets the RXEN flag in the UCSRB register to "1" to enable the receiver, and the RxD pin is occupied by the receiver to receive serial input data. The transmitter must be enabled at this time.

Both SPI transmit and receive use XCK as the transmit clock.

The SPI must be initialized before communication. The initialization process usually includes setting the baud rate, setting the frame data bit transmission sequence, and enabling the receiver or transmitter as needed. For interrupt-driven SPI operation, clear the global interrupt flag and disable all SPI interrupts during initialization.

When performing re-initialization such as changing the baud rate or frame structure, it must be ensured that no data is transmitted. The TXC flag can be used to detect whether the transmitter has completed all transmissions, and the RXC flag can be used to detect whether there is still data in the receive buffer that has not been read out. If the TXC flag is used for this purpose, the TXC flag must be cleared before each data transmission (before writing to the UDR register).

After initializing the SPI, write data to the UDR register to start data transfer. Since the transmitter controls the transmit clock, both sending and receiving data operate in this way. When the transmit shift register is ready to send a new frame of data, the transmitter will move the data written into the UDR register from the transmit buffer to the transmit shift register and send it out. To keep the input buffer synchronized with the transmitted data, the UDR register must be read after each byte of data is transmitted. When a data overflow occurs, the most recently received data will be lost, not the oldest received data.

## Send flags and interrupts

The SPI transmitter has two flag bits: the SPI data register empty flag UDRE and the transfer end flag TXC, both of which can generate interrupts.

The data register empty flag UDRE is used to indicate whether the transmit buffer can write a new data. This flag is set to "1" when the transmit buffer is empty and set to "0" when it is full. When the UDRE flag is "1", the CPU can write new data to the data register UDR, and vice versa.

When the data register empty interrupt enable flag UDRIE in the UCSRB register is "1", as long as UDRE is set (and the global interrupt is enabled), an SPI data register empty interrupt request will be generated. Writing to register UDR will clear UDRE. When using the interrupt mode to transfer data, the data register empty interrupt service routine must write a new data to the UDR to clear UDRE, or disable the data register empty interrupt. Otherwise, once the interrupt service routine ends, a new interrupt will be generated again.

When the entire data frame is shifted out of the transmit shift register and there is no new data in the transmit register, the transmit end flag TXC will be set. When the transmission end interrupt enable flag TXCIE on UCSRB (and the global interrupt enable) is set to "1", the SPI transmission end interrupt will be executed as the TXC flag is set. Once the interrupt service routine is entered, the TXC flag is automatically cleared, and the CPU can also write "1" to this flag to clear it.

## Disable transmitter

When TXEN is cleared, the transmitter can be disabled only after all data are sent, that is, there is no data to be transmitted in the transmit shift register and transmit buffer register. After the transmitter is disabled, the TxD pin resumes its general purpose IO function.

## Receive end flag and interrupt

The SPI receiver has a flag: the receive end flag RXC, which is used to indicate whether there is unread data in the receive buffer. When there is unread data in the receive buffer, this flag is "1", otherwise it is "0". If the receiver is disabled, the receive buffer will be flushed and RXC will be cleared. After setting the receive end interrupt enable flag RXCIE of UCSRB, as long as the RXC flag is set (and the global interrupt is enabled), the SPI receive end interrupt will be generated. When using the interrupt mode to receive data, the interrupt service routine must read the data from the UDR to clear the RXC flag when the data is received. Otherwise, as soon as the interrupt handler ends, a new interrupt will be generated.

## Disable the receiver

In contrast to the transmitter, disabling the receiver works instantly. Data being received will be lost. After the receiver is disabled (RXEN is cleared), the receiver will no longer occupy the RxD pin and the receive buffer will be flushed.

## Register definition

USART Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| UCSRA | 0xC0 | 0x20 | USPI Control and Status Register A |
| UCSRB | 0xC1 | 0x00 | USPI Control and Status Register B |
| UCSRC | 0xC2 | 0x06 | USPI Control and Status Register C |
| UBRRL | 0xC4 | 0x00 | USPI Baud Rate Register Low Byte |
| UBRRH | 0xC5 | 0x00 | USPI Baud Rate Register High Byte |
| UDR | 0xC6 | 0x00 | USPI data register |

### *UCSRA – USPI Control and Status Register A*

| UCSRA – USPI Control and Status Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC0 | | | | Default: 0x20 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | RXC | TXC | UDRE | - | - | - | - | - |
| R/W | R | R/W | R | - | - | - | - | - |

| Bit | Name | Description |
|---|---|---|
| 7 | RXC | Receive end flag.<br>When the value of RXC is "1", it indicates that there is unread data in the receive buffer. When the value of RXC is "0", it indicates that there is no unread data in the receive buffer. When the receiver is disabled, the receive buffer is flushed, causing RXC to be cleared. When the receive end interrupt enable flag RXCIE is "1", RXC can be used to generate a receive end interrupt. |
| 6 | TXC | Send end flag.<br>TXC is set when the data in the transmit shift register is sent out and the transmit buffer is empty. Execution TXC is automatically cleared when the transmission completes interrupt, and can also be cleared by writing "1" to TXC.<br>When the transmission end interrupt enable flag TXCIE is "1", TXC can be used to generate a transmission end interrupt. |
| 5 | UDRE | Data register empty flag.<br>When UDRE is "1", it indicates that the USPI transmit data buffer is empty and data can be written. When UDRE is "0", it indicates that the USPI send data buffer is full and cannot write data. When the data register empty interrupt enable flag UDRIE is "1", UDRE can be used to generate a data register empty interrupt. |
| 4:0 | - | Reserved under USPI. |

## *UCSRB – USPI Control and Status Register B*

| UCSRB – USPI Control and Status Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC1 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | RXCIE | TXCIE | UDRIE | RXEN | TXEN | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | - | - | - |

| Bit | Name | Description |
|---|---|---|
| 7 | RXCIE | Receive end interrupt enable flag.<br>Set to enable RXC interrupt, clear to disable RXC interrupt. When RXCIE is "1", the global interrupt is enabled, and the USPI receive end interrupt can be generated when the RXC of the UCSRA register is "1". |
| 6 | TXCIE | End of transmission interrupt enable flag.<br>Enables the TXC interrupt when set, and disables the TXC interrupt when cleared. When TXCIE is "1", the global interrupt is enabled, and the USPI transmission end interrupt can be generated when the TXC of the UCSRA register is "1". |
| 5 | UDRIE | Data register empty interrupt enable flag.<br>Set to enable UDRE interrupt, clear to disable UDRE interrupt. When UDRIE is "1", the global interrupt is enabled, and the USPI data register empty interrupt can be generated when the UDRE of the UCSRA register is "1". |
| 4 | RXEN | Receive enable flag.<br>Starts the USPI receiver when asserted. The general purpose IO function of the RxD pin is replaced by USPI receive.<br>Disabling the receiver will flush the receive buffer. |
| 3 | TXEN | Send enable flag.<br>When asserted, starts the USPI transmitter. The general purpose IO function of the TxD pin is replaced by USPI transmit.<br>After TXEN is cleared, the USART can be really disabled only after all data transmission is completed. |
| 2:0 | - | Reserved under USPI. |

## *UCSRC – USART Control and Status Register C*

| UCSRC – USART Control and Status Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC2 | | | | Default: 0x86 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UMSEL1 | UMSEL0 | - | - | - | DORD | UCPHA | UCPOL |
| R/W | R/W | R/W | - | - | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:6 | UMSEL1:0 | USART mode selection flags.<br>UMSEL selects synchronous or asynchronous operation mode. |

| UMSEL | Method |
|---|---|
| 0 | USART asynchronous mode of operation |
| 1 | USART Synchronous Operation Mode |
| 2 | SPI Slave Operation Mode |
| 3 | SPI master mode of operation |

| Bit | Name | Description |
|---|---|---|
| 5:3 | - | Reserved under USPI. |
| 2 | DORD | Data transfer order selection bits. |

| DORD | Data order |
|---|---|
| 0 | Start edge |
| 1 | End edge |

| Bit | Name | Description |
|---|---|---|
| 1 | UCPHA | Clock phase selection.<br>UCPHA selects that data sampling occurs on the start or end edge. |

| UCPHA | Sampling time |
|---|---|
| 0 | Start edge |
| 1 | End edge |

| Bit | Name | Description |
|---|---|---|
| 0 | UCPOL | Clock polarity selection.<br>UCPOL selects that data changes and sampling occur on rising or falling edges. |

| UCPOL | Changes in sent data | Sampling of received data |
|---|---|---|
| 0 | Rising edge of XCK | Falling edge of XCK |
| 1 | Falling edge of XCK | Rising edge of XCK |

### *UBRRL – USPI Baud Rate Register Low Byte*

| UBRRL – USPI Baud Rate Register Low Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC4 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UBRR7 | UBRR6 | UBRR5 | UBRR4 | UBRR3 | UBRR2 | UBRR1 | UBRR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | UBRR[7:0] | The low byte portion of the USPI baud rate register. The USPI baud rate register contains two parts, UBRRL and UBRRH, which are combined to set the baud rate of communication. |

### *UBRRH – USPI Baud Rate Register High Byte*

| UBRRH – USPI Baud Rate Register High Byte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC5 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | - | - | - | UBRR11 | UBRR10 | UBRR9 | UBRR8 |
| R/W | - | - | - | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:4 | - | Reserved under USPI. |
| 3:0 | UBRR[11:8] | The high byte portion of the USPI baud rate register.<br>The USPI baud rate register contains two parts, UBRRL and UBRRH, which are combined to set the baud rate of communication.<br>UBRR = {UBRR[11:8], UBRRL} |

| Operating mode | Baud rate calculation formula |
|---|---|
| Slave mode | Baud rate determined by external host |
| Host mode | BAUD = fsys/(2*(UBRR+1)) |

### *UDR – USPI Data Register*

| UDR – USPI Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC6 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | UDR7 | UDR6 | UDR5 | UDR4 | UDR3 | UDR2 | UDR1 | UDR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | UDR | Data sent and received by USPI.<br>The USPI transmit data buffer and receive data buffer share the USPI data register UDR.<br>Writing data to the UDR means writing to the transmit data buffer, and reading data from the UDR means reading the receive data buffer.<br>In 5- to 8-bit data frame mode, the unused 9th bit is ignored by the transmitter and set to 0 by the receiver.<br>Only when the UDRE flag of the UCSRA register is "1" can the transmit buffer be written, otherwise the operation of the transmitter will be wrong. When the transmit shift register is empty, the transmitter will load the data in the transmit buffer into the transmit shift register, and then the data will be serially output from the TxD pin.<br>The receive buffer contains a two-level FIFO that changes its state once the receive buffer is read. |

# TWI – Two-Wire Serial Bus (I2C)

- Simple yet powerful and flexible communication interface requiring only 2 wires
- Supports master and slave operation
- Device can operate in transmitter mode or receiver mode
- 7-bit address space allows 128 slaves
- Support for multi-master arbitration
- Data transfer rates up to 400Kbps
- Fully programmable slave address as well as public address
- Can wake up on address match in sleep mode

## TWI bus introduction

The two-wire serial interface TWI is well suited for typical processor applications. The TWI protocol allows system designers to interconnect 128 different devices together using only two bidirectional transmission lines. These two lines are clock SCL and data SDA. External hardware only requires two pull-up resistors on each line. All devices connected to the bus have their own addresses. The TWI protocol solves the problem of bus arbitration.

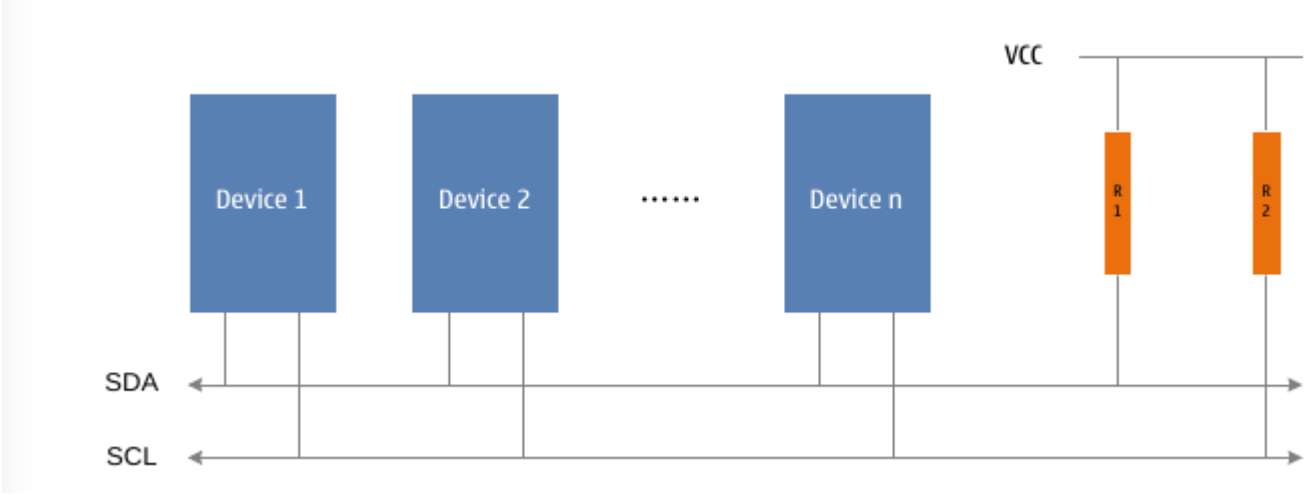
## TWI terminology

The terms defined below will appear frequently in this section.

| Termonology | Description |
|---|---|
| Host | Device to start and stop transfers. The host is also responsible for generating the SCL clock. |
| Slave | Device addressed by the host |
| Transmitter | Device that puts data on the bus |
| Reciever | A device that receives data from the bus |

## Electrical connections

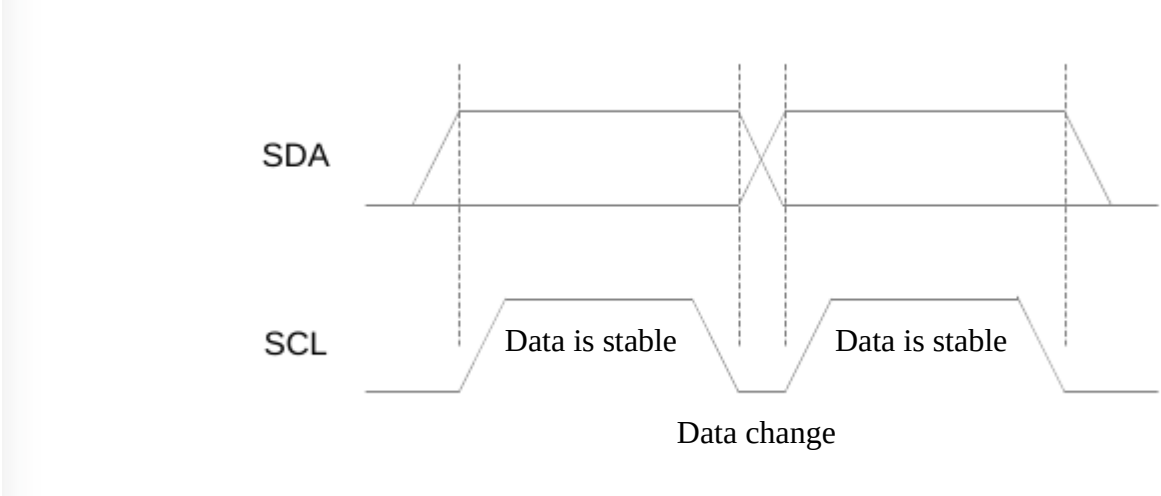
As shown in the figure below, both lines of the TWI interface are connected to the positive power supply through pull-up resistors. The bus drivers for all TWI compliant devices are open-drain or open-collector, thus implementing a wired-AND function for interface operations. When the TWI device output is "0", the TWI bus will generate a low level. When all TWI device outputs are tri-stated, the bus allows pull-up resistors to pull the voltage high. To ensure all bus operations, all devices connected to the TWI bus must be powered on.




TWI bus interconnection diagram

## Data transfer and frame structure

Every data transfer on the TWI bus is synchronized with the clock. When the clock line is high, the level on the data line must remain stable, except to generate a start or stop condition.

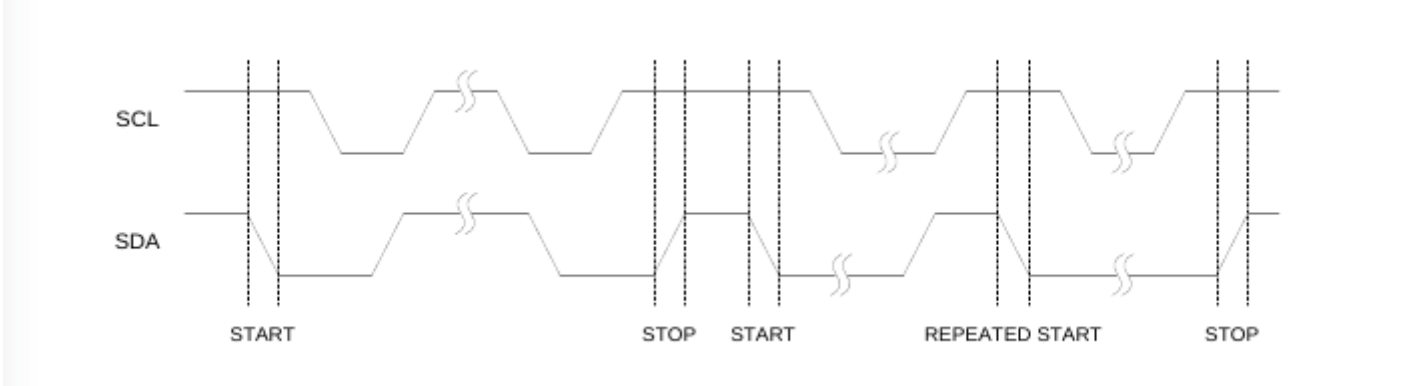



TWI data validity graph

## Start and stop states

The transfer of TWI is started and stopped by the host. The host sends a START state on the bus to send data transfer, and sends a STOP state to stop data transfer. Between the START and STOP states, the bus is considered busy and no other master is allowed to attempt to take control of the bus. There is a special case that only allows a new START state to occur between the START and

STOP states, which is called the REPEATED START state, and is suitable for the current host to start a new transfer without giving up bus control. The bus is still considered busy after a REPEATED START until the next STOP. This is consistent with START, so in this document, unless otherwise specified, START is used to express START and REPEATED START. As shown in the figure below, the START and STOP conditions change the state of the SDA line while the SCL line is high.




START, REPEATED START and STOP state diagram

## Address packet format

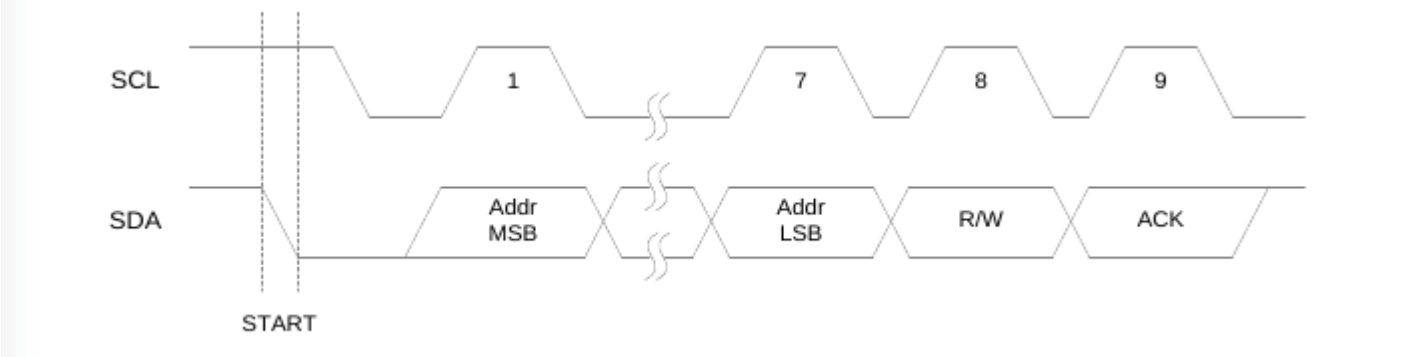
All address packets transmitted on the TWI bus are 9-bit data length, composed of 7-bit address, 1-bit READ/WRITE control flag and 1-bit acknowledge flag. When the READ/WRITE flag is "1", a read operation is performed; when the READ/WRITE flag is "0", a write operation is performed. After the slave is addressed, it must acknowledge in the 9th SCL(ACK) cycle by pulling the SDA line low. If the slave is busy or cannot respond to the master for other reasons, it should keep the SDA line high during the ACK cycle. The master can then issue a STOP condition or a REPEATED START condition to resume transmission.

The address packet includes a slave address and a read or write control flag, represented by SLA+R or SLA+W, respectively.

The MSB bit of the address byte occurs first. Except that the reserved address "00000000" is reserved for general calls and all addresses in the format "1111xxxx" need to be reserved for future use, other slave addresses can be freely assigned by the designer.

When a general call occurs, all slaves should respond by pulling the SDA line low during the ACK cycle. The broadcast function can be used when the master needs to send the same information to multiple slaves. After the general call address plus the WRITE flag is sent to the bus, all slaves that need to respond to the general call will pull the SDA line low during the ACK cycle. All of these slaves that respond to the general call will receive the following packets. It should be noted that it is meaningless to send the general call address plus the READ flag, because if several slaves send different data at the same time, it will cause a bus conflict.

The format of the address packet is shown in the following figure:



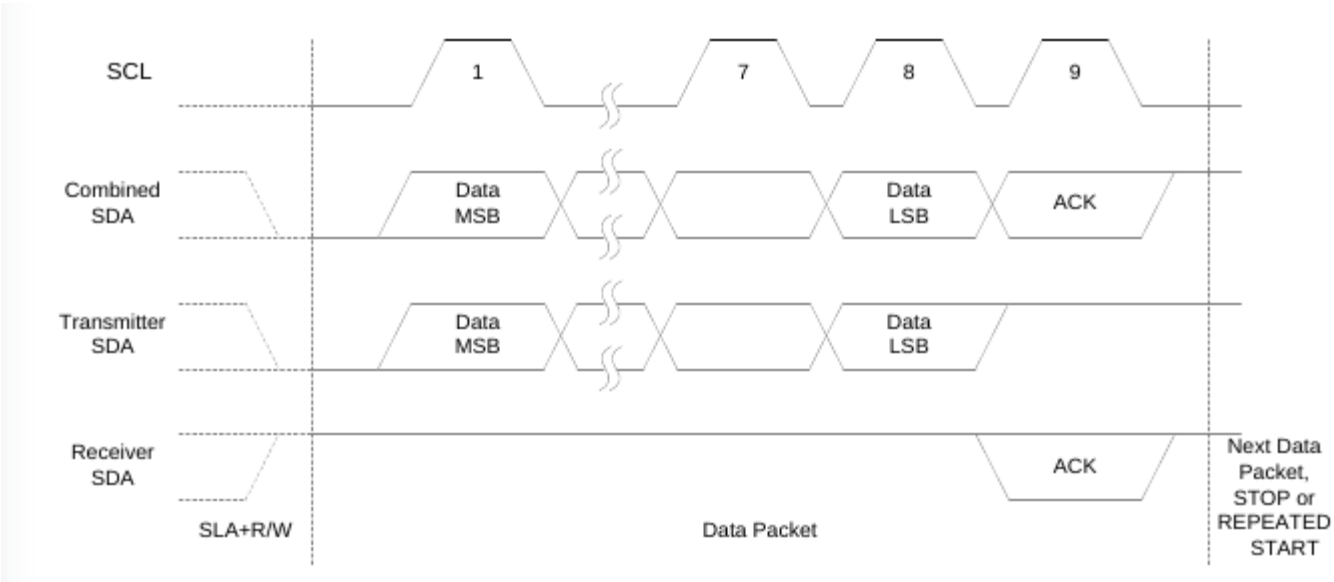

TWI address packet format diagram

## Packet format

All data packets transmitted on the TWI bus are 9-bit data length, consisting of 1 data byte and 1 acknowledge bit. During data transmission, the host is responsible for generating the transmission clock SCL and START and STOP states, the transmitter sends the byte data to be transmitted, and the receiver generates a reception response. The acknowledgment signal ACK is generated by the receiver in the ninth SCL(ACK) cycle by pulling the SDA line low. If the receiver keeps the SDA line high during the ACK cycle, the unacknowledged signal NACK is sent.

When the receiver has received the last byte, or cannot receive any more data for some reason, it should inform the sender by sending a NACK after the last byte is received. The MSB bit of the data byte is transmitted first.

The packet format is shown in the following figure:

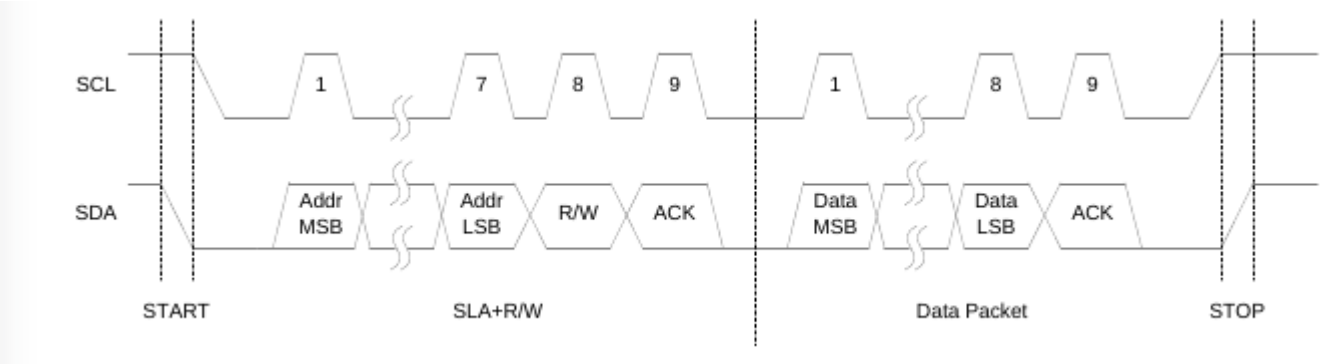



TWI packet format diagram

The transmission of combined address and data packet, a transmission basically consists of 1 START, 1 SLA+R/W, 1 or more data packets and 1 STOP. Only empty messages for START and STOP are illegal. The master-slave handshake can be implemented using the wired-AND function of the SCL line. The slave can extend the ground period of SCL by pulling the SCL line low. This feature is useful when the master is programmed with a much faster clock speed than the slave, or when the slave needs extra time to process data.

Extending the low period of SCL by the slave does not affect the high period of SCL, it is still determined by the master. It can be seen that the slave can reduce the data transmission speed of TWI by changing the duty cycle of SCL.

The figure below shows a typical data transfer. Note that multiple bytes can be transferred between SLA+R/W and STOP, depending on the application software implementation protocol.




Typical TWI transmission

## Multi-Master System and Its Arbitration and Synchronization

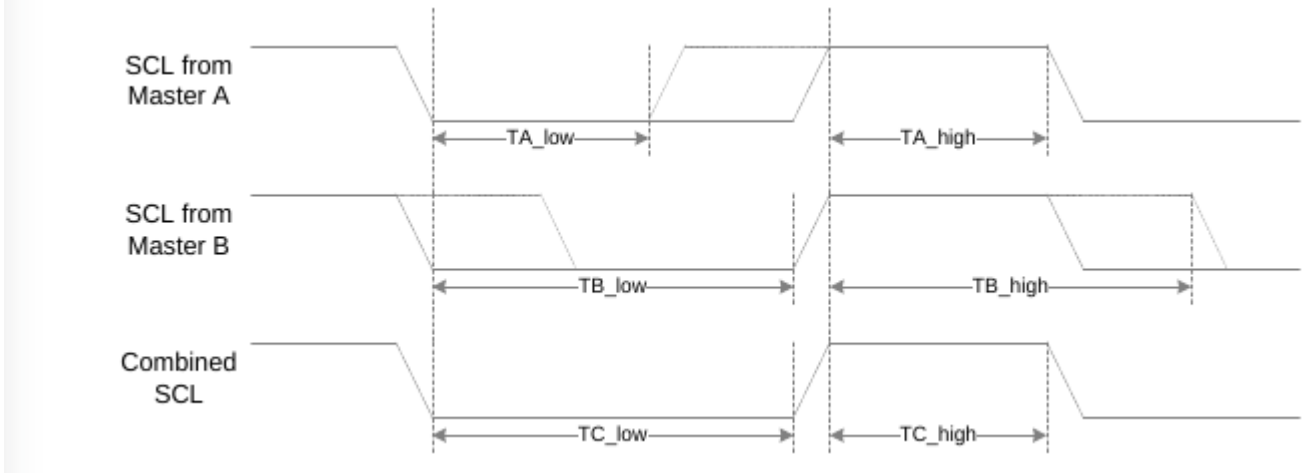
The TWI protocol allows multiple masters on the bus, and takes special measures to ensure that even if two or more masters initiate a transfer at the same time, it will be treated like a normal transfer. There are two problems with multi-host systems:

1. The implemented algorithm allows only one host in the multi-host to complete the transfer. Other hosts must stop their transmissions when they find that they have lost the option. This selection process is called arbitration. When the competing master finds that it has lost arbitration, it should switch to slave mode immediately to detect whether it is addressed by the master that has gained control of the bus. The fact that multiple masters start transmitting at the same time should not be detected by the slaves, that is, it is not allowed to corrupt the data being transmitted on the bus.

2. Different hosts may use different SCL frequencies. In order to ensure the consistency of transmission, a scheme of synchronizing the serial clock of the host must be designed. This simplifies the arbitration process.

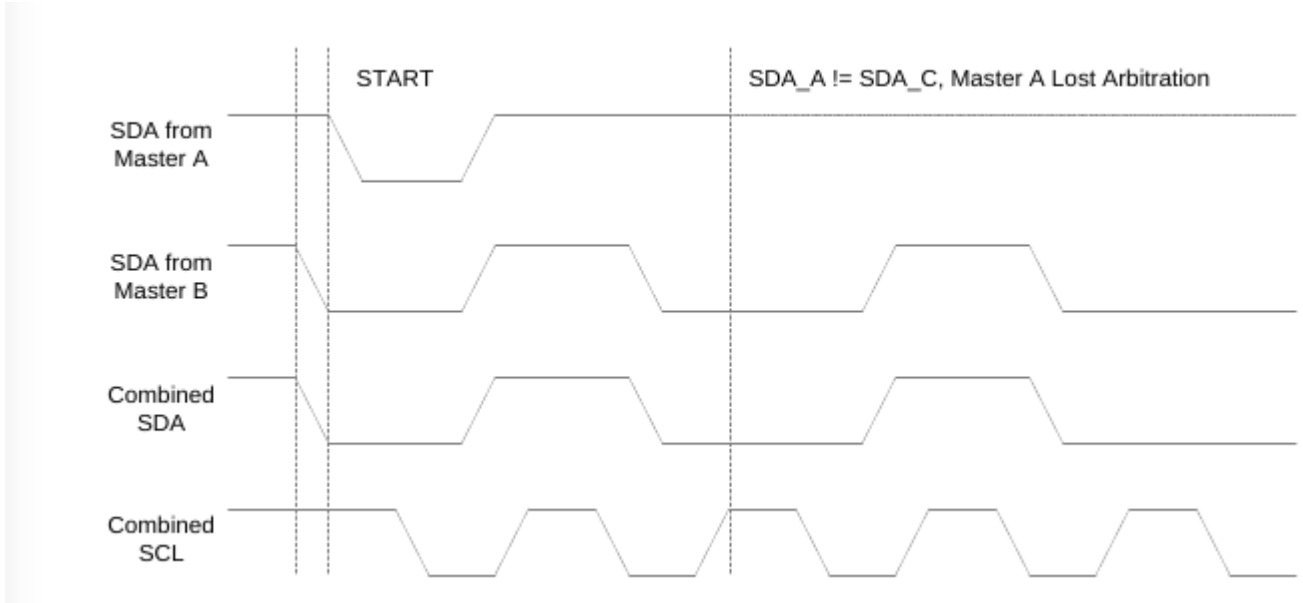
The line AND function of the bus is used to solve the above problems. The serial clocks of all the masters are wired together to generate a combined clock whose high time is equal to the shortest of all master clocks, and its low time is equal to the longest of all master clocks. All masters monitor SCL, and when the combined SCL clock goes high or low, they can effectively start counting their respective SCL high and low overflow periods, respectively.

The SCL clock synchronization mechanism of multiple masters is shown in the following figure:




Multi-Master SCL Clock Synchronization Timing Diagram

After outputting data, all masters continue to monitor the SDA line to achieve arbitration. If the value read back from the SDA does not match the value output by the master, the master loses arbitration. It should be noted that arbitration is lost only when one master outputs SDA high and the other master outputs SDA low. A master that loses arbitration should immediately transition to slave mode and detect if it is addressed. A master that loses arbitration must take the SDA line high, but can still generate a clock signal before the end of the current data or address packet. Arbitration will continue until only one master remains in the system, which may consume multiple bits. If multiple masters address the same slave, arbitration will continue until the packet.




Arbitration between two hosts

Note that arbitration is not allowed in the following situations:

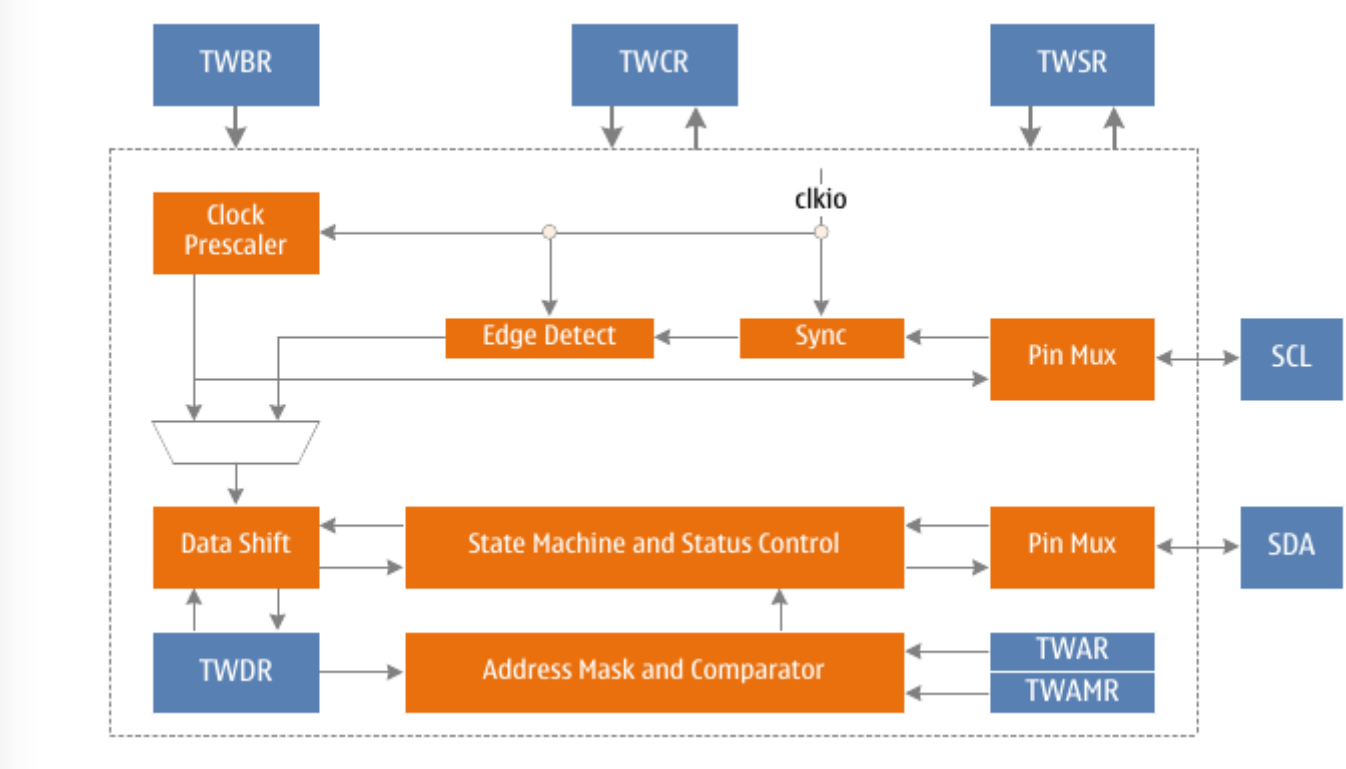
- between a REPEATED START state and a data bit;
- between a STOP state and a data bit;
- between a REPEATED START state and a STOP state;

The application software must take the above into account and ensure that these illegal arbitration situations will not occur. This means that in a multi-master system, all data transfers must consist of

the same SLA+R/W and packets. In other words, all transfers must contain the same number of packets, otherwise the outcome of the arbitration cannot be defined.

## Overview of TWI Modules

The structure diagram of the TWI module is shown in the following figure.




TWI Block Structure Diagram

The TWI module mainly includes a bit rate generator, a bus interface unit, an address comparator and a control unit. See the detailed description below for details.

## Bit rate generator unit

The bit rate generator unit mainly controls the SCL clock period in master mode. The SCL clock period is determined by the prescaler control bits in the TWI flag rate register TWBR and the TWI status register TWSR. Slave operation is not affected by bit rate or prescaler settings, but make sure that the slave's operating clock is at least 16 times the SCL frequency. Note that the slave may extend the low period of SCL, thereby reducing the average clock frequency of the TWI bus. The SCL clock frequency is generated by the following formula:

$$f_{scl} = f_{sys}/(16 + 2*TWBR*4^{TWPS})$$

Among them, TWBR is the value of the TWI bit rate register, and TWPS is the prescaler control flag in the TWI status register.

## Bus interface unit

The bus interface unit includes data and address shift register TWDR, START/STOP controller and arbitration decision hardware circuit.

TWDR contains the address or data bytes to be sent, or the address or data bytes received. In addition to containing an 8-bit TWDR, the bus interface unit also includes a transmit or receive ACK/NACK register. This ACK/NACK register cannot be directly accessed by application software. When data is received, it can be set or cleared through the TWI control register TWCR. When sending data, the received ACK/NACK value is reflected by the TWS value in the TWI status register TWSR.

The START/STOP controller is responsible for generating and detecting START, REPEATED START and STOP states. When the MCU is in some sleep modes, the START/STOP controller can still detect START and STOP states and wake the MCU from sleep mode when addressed by the host on the TWI bus.

If the TWI initiates a data transfer in master mode, the arbitration detection circuit will continuously monitor the bus to determine if it still has control of the bus. When the TWI module loses control of the bus, the control unit will perform correct actions and generate appropriate status codes to notify the MCU.

## Address matching unit

The address matching unit is used to check whether the received address byte matches the 7-bit address in the TWI address register. When the TWI General Call Identification Enable flag (TWGCE) in the TWAR register is set, the address received from the bus is also compared to the broadcast address. Once the address match is successful, the control unit will perform the correct action. The TWI module may or may not respond to addressing from the host, depending on the setting of the TWCR register. Even in sleep mode, the address matching unit can compare addresses and wake up the MCU from sleep mode if addressed by the host on the bus.

## Control unit

The control unit is responsible for monitoring the bus and responding accordingly based on the TWCR settings. When an event occurs on the TWI bus that requires application software to participate, the TWI interrupt flag TWINT will be set. In the next clock cycle, the TWI status register TWSR will be updated with the status code indicating the event. When TWINT is set, TWSR contains the exact status information. At other times, TWSR is a special status code indicating that no exact status information is available. Once the TWINT flag is set, the SCL line remains low, suspending TWI transfers on the bus and allowing the application software to process the event.

The TWINT flag will be set when:

- After TWI transmits START/REPEATED START status
- After TWI transmits SLA+R/W
- After TWI transmits an address byte
- After TWI Bus Arbitration Lost
- After TWI is addressed by the host (slave address matching or broadcast mode)

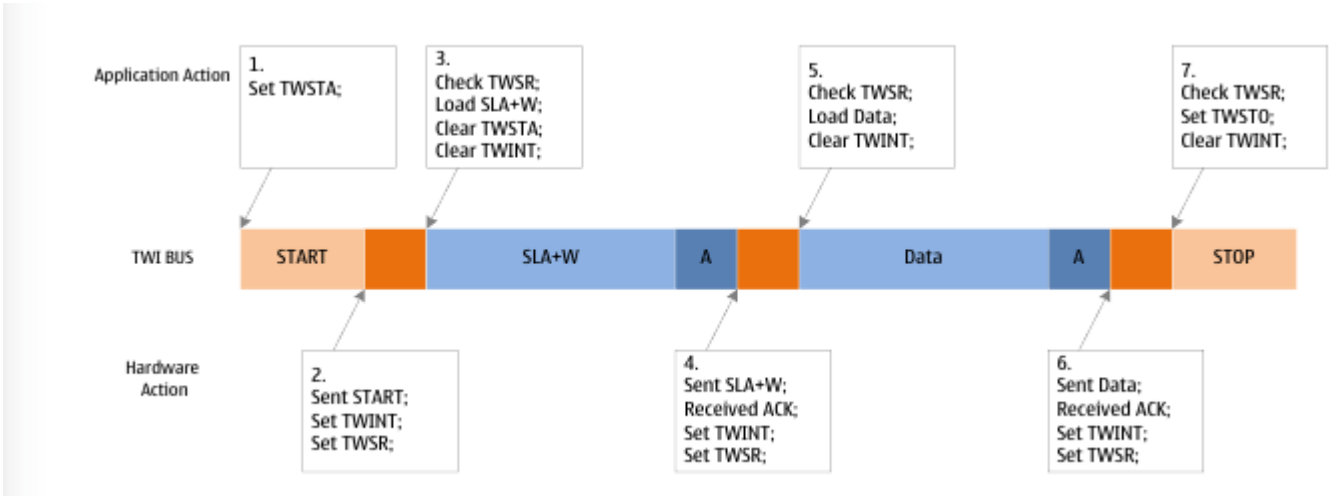
- When addressed as a slave, after receiving a STOP or REPEATED START
- When a bus error is caused by an illegal START or STOP state

## Use of TWI

The TWI interface is byte-oriented and interrupt-based. All bus events, such as receiving a byte or sending a START signal, will generate a TWI interrupt. Since TWI is based on interrupts, the application software can freely perform other operations during the transfer of TWI bytes. The TWI Interrupt Enable flag, TWIE, in the TWCR register, together with the Global Interrupt Enable flag, controls whether a TWI interrupt is generated when the TWINT flag is set. If the TWIE flag is cleared, the application software must detect the action on the TWI bus by polling the TWINT flag.

When the TWINT flag is set, it means that the TWI interface has completed the current operation and is waiting for the response from the application software. In this case, the TWI status register TWSR contains a status code that reflects the current bus status. The application software can decide how the TWI interface should work in the next TWI bus cycle by setting the TWCR and TWDR registers.

The figure below shows an example of an application connecting to the TWI interface. In this example, the master expects to send one byte of data to the slave. The description here is very simple, and the following chapters will show it in more detail.




Typical transmission process diagram of TWI

The typical transmission process diagram of 1.2.3.4.5.6.7.TWI shown in the figure The TWI transmission process is:

1. The first step in a TWI transfer is to send a START. The TWI hardware is instructed to send a START signal by writing a specific value to the TWCR register. The written value will be described in detail later. It is very important that TWINT is set in the written value, writing a '1' to the TWINT flag will clear the flag. The TWI will not initiate any operation while the TWINT in the TWCR register is set. Once the software clears the TWINT flag, the TWI module immediately initiates the transmission of the START signal.

2. When the START status is sent, the TWINT flag of TWCR will be set, and TWSR will be updated to the new status code, indicating that the START signal has been successfully sent.
3. The application looks at the value of TWSR to determine that the START state has been sent successfully. If TWSR is displayed with other values, the application can perform some special action, such as calling an error handler. After determining that the status code is as expected, the program loads the value of SLA+W into the TWDR register. The TWDR register can be used for both address and data.
4. The software then writes a specific value to the TWCR register, instructing the TWI hardware to send the value of SLA+W in TWDR. The written value will be described in detail later. Set TWINT in the written value to clear the TWINT flag. The TWI will not initiate any operation while the TWINT in the TWCR register is set. Once the software clears the TWINT flag, the TWI module immediately starts the transfer of the address packet.
5. When the address packet is sent, the TWINT flag of TWCR will be set, and TWSR will be updated to a new status code, indicating that the address packet has been sent successfully. The status code will also reflect whether the slave responds to the address packet.
6. The application checks the value of TWSR to determine that the address packet has been successfully sent, and the ACK received is the expected value. If TWSR is displayed with other values, the application can perform some special action, such as calling an error handler. After confirming that the status code is as expected, the program loads the value of Data into the TWDR register. Then the software writes a specific value to the TWCR register, instructing the TWI hardware to send the value of Data in TWDR. The written value will be described in detail later. Set TWINT in the written value to clear the TWINT flag. The TWI will not initiate any operation while the TWINT in the TWCR register is set. Once the software clears the TWINT flag, the TWI module immediately starts the transmission of the data packet.
7. When the data packet is sent, the TWINT flag of TWCR will be set, and the TWSR will be updated to a new status code, indicating that the data packet is successfully sent. The status code will also reflect whether the slave responds to the packet.
8. The application looks at the value of TWSR to determine that the packet has been successfully sent and the ACK received is the expected value. If TWSR is displayed with other values, the application can perform some special action, such as calling an error handler. After confirming that the status code is as expected, the software writes a specific value to the TWCR register, instructing the TWI hardware to send a STOP signal. The value to be written will be described in detail later. Set TWINT in the written value to clear the TWINT flag. The TWI will not initiate any operation while the TWINT in the TWCR register is set. Once the software clears the TWINT flag, the TWI module immediately initiates the transmission of the STOP signal. It should be noted that TWINT will not be set after the STOP signal is sent.

Although the example is relatively simple, it contains all the rules of the TWI data transfer process. Summarized as follows:

- The TWINT flag is set when the TWI completes an operation and waits for feedback from the application. The SCL clock line will be pulled low until TWINT is cleared;

- When the TWINT flag is set, the user must update all TWI registers to the values associated with the next TWI bus cycle. For example, the TWDR register must be loaded with the value to be sent on the next bus cycle.
- After updating all registers and completing other necessary operations, the application program writes to the TWCR register. When writing to TWCR, the TWINT flag must be set to clear the TWINT flag. After TWINT is cleared, TWI starts to perform the operation set by TWCR.

## Transfer mode

TWI can work in the following 4 main modes: Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Multiple modes can be used in the same application. For example, the TWI can use the MT mode to write data to the TWI EEPROM and use the MR mode to read data from the EEPROM. If there are other hosts on the system, some of which may also be sending data to the TWI, SR mode will be used. It is up to the application software to decide which mode to use.

These modes are described in detail below. In each mode of data transfer, a picture is used to describe the possible status codes. These pictures contain the following abbreviations:

S: Start state

Rs: REPEATED START status

R: Read operation flag (SDA is high)

W: Write operation flag (SDA is low)

A: Acknowledge flag (SDA is low)

NA: No acknowledge flag (SDA is high)

Data: 8-bit data byte

P: STOP state

SLA: Slave address

The circle in the picture is used to indicate that the TWINT flag is set, and the number in the circle indicates the status code in the TWSR register, in which the prescaler control flag is masked to "0". In these places, the application must take the appropriate action to continue or complete the TWI transfer. TWI transfers are suspended until the TWINT flag is cleared.

When the TWINT flag is set, the status code in TWSR is used to determine the appropriate software action. Details of the software action required under each status code and the subsequent serial transfer are given in the tables. Note that the prescaler control bits in TWSR are masked to "0" in the table.

## Host send mode

In master transmit mode, the TWI will transmit a certain number of data bytes to the slave receiver. To enter master mode, a START signal must be sent. The following address packet format determines whether the TWI enters host transmitter mode or host receiver mode. If sending SLA+W, enter the host sending mode. If sending SLA+R, it will enter the host receiving mode. The status codes mentioned in this section assume that the prescaler control bits are '0'.

A START signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 1 | 0 | x | 1 | 0 | x |

The TWEN flag must be set to "1" to enable the TWI interface, TWSTA to be set to "1" to send a START signal, and TWINT to be set to "1" to clear the TWINT flag. The TWI module detects the bus state and sends a START signal immediately when the bus is idle. After sending the START, the hardware sets the TWINT flag and updates the status code of TWSR to 0x08.

In order to enter master transmit mode, SLA+W must be sent. This can be done by doing the following. First write SLA+W to the TWDR register, then write “1” to the TWINT flag to clear the TWINT flag to continue the transmission, that is, write the following values to the TWCR register to send SLA+W:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 0 | x | 1 | 0 | x |

When the SLA+W transmission is completed and the response signal is received, TWINT is set again, and the status code of TWSR is updated at the same time. Possible status codes are 0x18, 0x20 or 0x38. Appropriate responses for each status code are described in detail in the status code table.

When the SLA+W is sent successfully, you can start sending data packets. This can be done by writing data to the TWDR register. TWDR can only be written when the TWINT flag is high. Otherwise, the access is ignored and the write conflict flag TWWC is set. After updating the TWDR, write "1" to the TWINT flag to clear the TWINT flag to continue the transmission. That is, write the following values to the TWCR register to send data:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 0 | x | 1 | 0 | x |

When the data packet is sent and the response signal is received, TWINT is set again, and the status code of TWSR is updated at the same time. Possible status codes are 0x28 or 0x30. Appropriate responses for each status code are described in detail in the status code table.

When the data is sent successfully, you can continue to send data packets. This process repeats until the last byte is sent. The entire transfer ends when the master generates a STOP signal or a REPEATED START signal.

A STOP signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 1 | x | 1 | 0 | x |

A REPEATED START signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 1 | 0 | x | 1 | 0 | x |

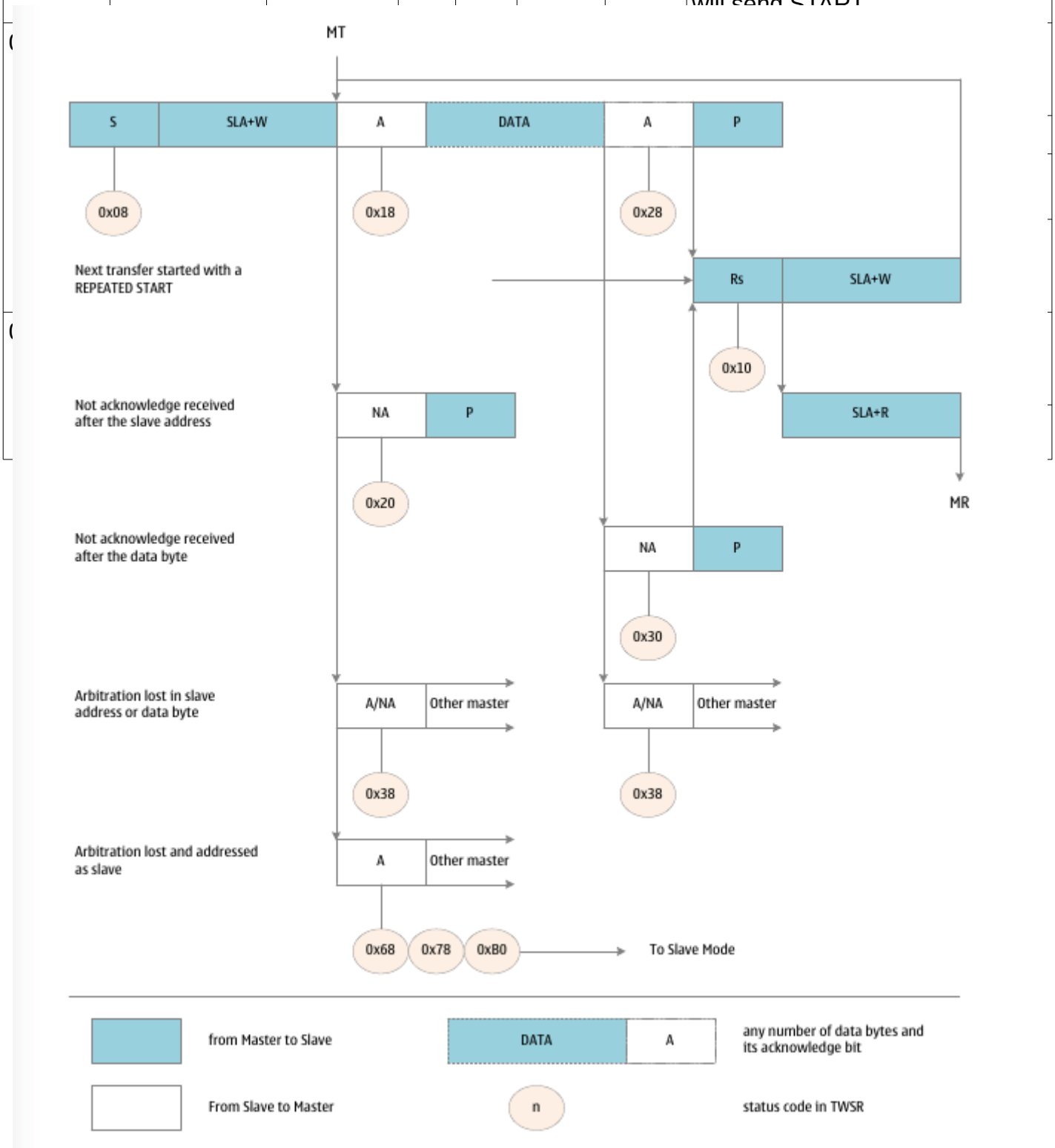
After sending REPEATED START (status code 0x10), the TWI interface can access the same slave again, or access a new slave without sending a STOP signal. REPEATED START allows the master to switch between slave, master transmitter and master receiver modes without losing control of the bus.

The status codes and corresponding operations in the host sending mode are shown in the following table:

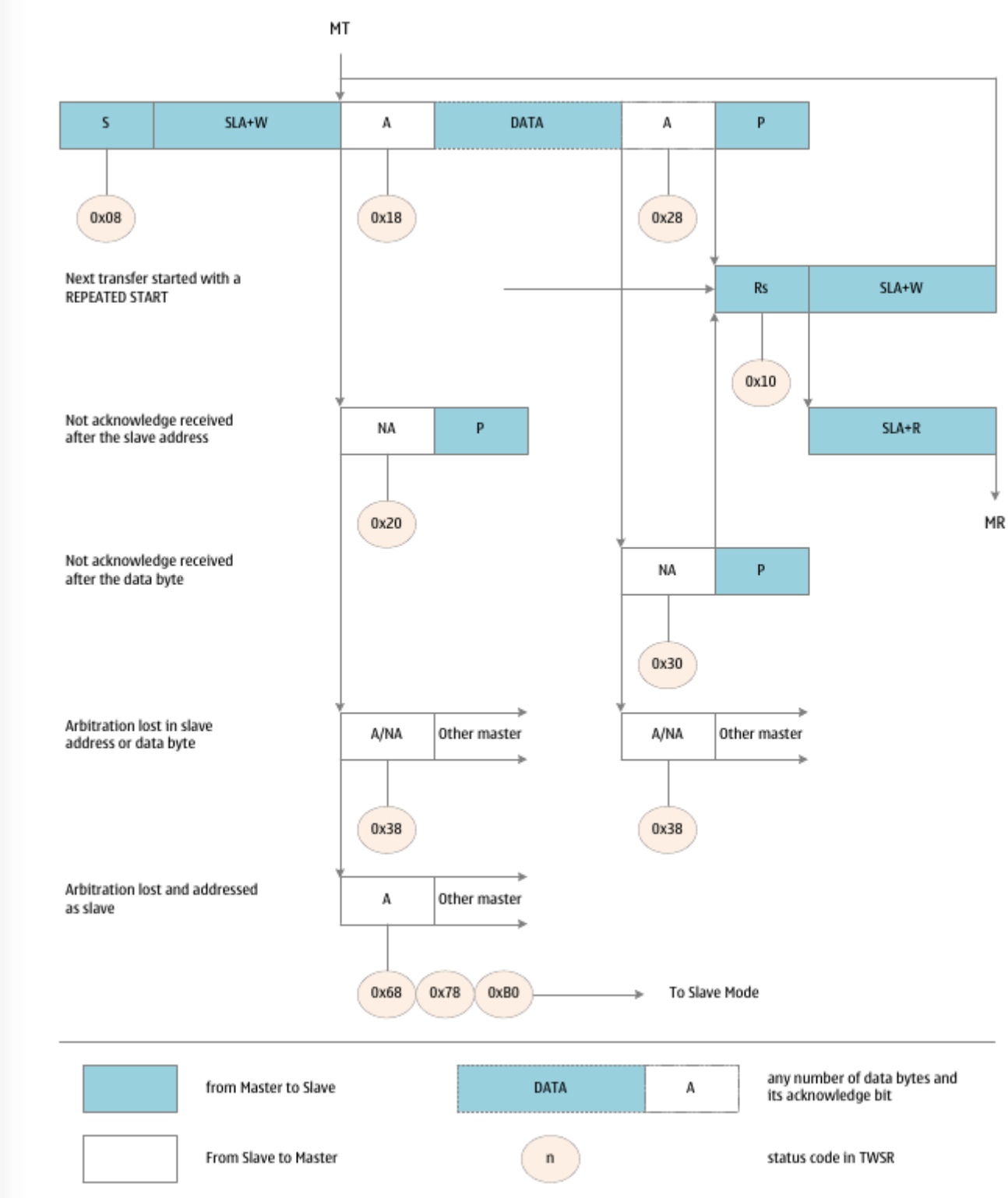
Status code table for host transmit mode

| status code | Bus and Hardware Status | Application software response | | | | | The next move for the hardware |
|---|---|---|---|---|---|---|---|
| | | read/write TWDR | Action on TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0x08 | START sent | load/ loading SLA+W | 0 | 0 | 1 | x | will send SLA+W; ACK or NACK will be received |
| 0x10 | REPEATED START sent | load/ loading SLA+W | 0 | 0 | 1 | x | will send SLA+W; ACK or NACK will be received |
| | | load/ loading SLA+R | 0 | 0 | 1 | x | will send SLA+R; will receive ACK or NACK; will switch to MR mode |
| 0x18 | SLA+W sent; ACK received | load/ loading data | 0 | 0 | 1 | x | will send data; ACK or NACK will be received |
| | | no action | 1 | 0 | 1 | x | will send REPEATED START |
| | | no action | 0 | 1 | 1 | x | will send STOP; will reset the TWSTO flag |
| | | no action | 1 | 1 | 1 | x | will send STOP; will reset the TWSTO flag; will send START |
| 0x20 | SLA+W sent; NACK received | load/ loading data | 0 | 0 | 1 | x | will send data; ACK or NACK will be received |
| | | no action | 1 | 0 | 1 | x | will send REPEATED START |
| | | no action | 0 | 1 | 1 | x | will send STOP; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | will reset the TWSTO flag |
| | | no action | 1 | 1 | 1 | x | will send STOP;<br>will reset the TWSTO flag;<br>will send START |
| 0x28 | Data byte sent;<br>ACK received | load/ loading data | 0 | 0 | 1 | x | will send data;<br>ACK or NACK will be received |
| | | no action | 1 | 0 | 1 | x | will send REPEATED START |
| | | no action | 0 | 1 | 1 | x | will send STOP;<br>will reset the TWSTO flag |
| | | no action | 1 | 1 | 1 | x | will send STOP;<br>will reset the TWSTO flag;<br>will send START |



The format and status of the host sending mode are shown in the following figure:




Format and State Diagram for Host Transmit Mode

## Host receive mode

In master receive mode, the TWI will receive a certain number of data bytes from the slave transmitter. To enter master mode, a START signal must be sent. The following address packet

format determines whether the TWI enters host transmitter mode or host receiver mode. If sending SLA+W, enter the host sending mode. If sending SLA+R, it will enter the host receiving mode. The status codes mentioned in this section assume that the prescaler control bit is '0'.

A START signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 1 | 0 | x | 1 | 0 | x |

The TWEN flag must be set to "1" to enable the TWI interface, TWSTA to be set to "1" to send a START signal, and TWINT to be set to "1" to clear the TWINT flag. The TWI module detects the bus state and sends a START signal immediately when the bus is idle. After sending the START, the hardware sets the TWINT flag and updates the status code of TWSR to 0x08.

In order to enter master receive mode, SLA+R must be sent. This can be done by doing the following. First write SLA+R to the TWDR register, then write “1” to the TWINT flag to clear the TWINT flag to continue the transmission, that is, write the following values to the TWCR register to send SLA+R:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 0 | x | 1 | 0 | x |

When the SLA+R transmission is completed and the response signal is received, TWINT is set again, and the status code of TWSR is updated at the same time. Possible status codes are 0x38, 0x40 or 0x48. Appropriate responses for each status code are described in detail in the status code table.

When SLA+R is sent successfully, it can start to receive data packets. Reception continues by clearing the TWINT flag by writing a '1' to the TWINT flag. That is, write the following value to the TWCR register to start receiving:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 0 | x | 1 | 0 | x |

When the data packet is received and the acknowledgment signal is sent, TWINT is set again, and the status code of TWSR is updated at the same time. Possible status codes are 0x50 or 0x58. Appropriate responses for each status code are described in detail in the status code table.

When the data is received successfully, you can continue to receive data packets. This process repeats until the last byte is received. After the master receives the last byte, it must send a NACK response signal to the slave transmitter. The whole reception ends when the host generates a STOP signal or a REPEATED START signal.

A STOP signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 0 | 1 | x | 1 | 0 | x |

A REPEATED START signal is issued by writing the following values to the TWCR register:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 1 | x | 1 | 0 | x | 1 | 0 | x |

After sending REPEATED START (status code 0x10), the TWI interface can access the same host again, or access a new host without sending a STOP signal. REPEATED START allows the master to switch between slave, master transmitter and master receiver modes without losing control of the bus.

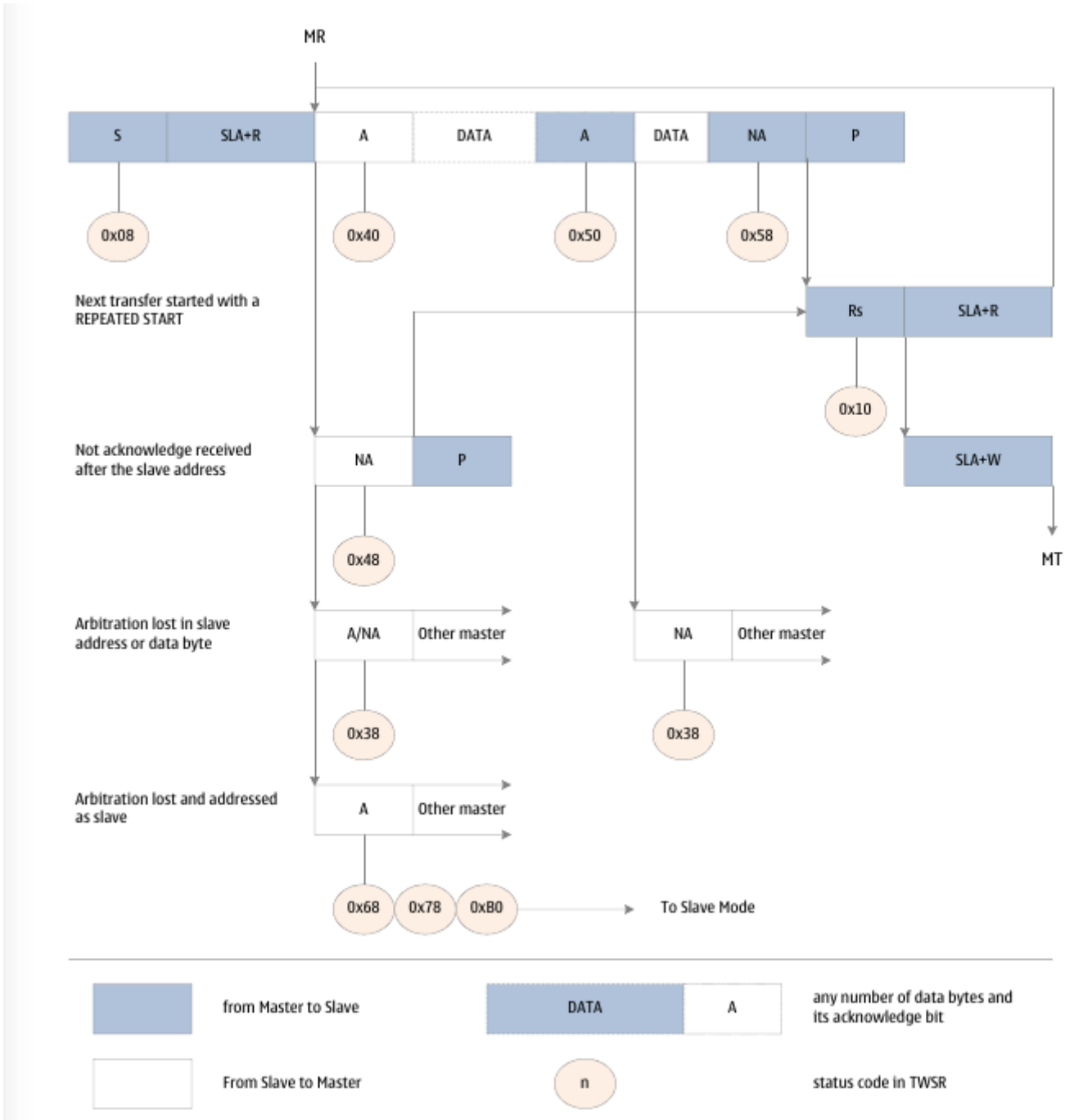
The status codes and corresponding operations in the host receiving mode are shown in the following table:

Status code table for host receiving mode

| status code | Bus and Hardware Status | Application software response | | | | | The next move for the hardware |
|---|---|---|---|---|---|---|---|
| | | read/write TWDR | Action on TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0x08 | START sent | load/ loading SLA+R | 0 | 0 | 1 | x | will send SLA+R; ACK or NACK will be received |
| 0x10 | REPEATED START sent | load/ loading SLA+R | 0 | 0 | 1 | x | will send SLA+R; ACK or NACK will be received |
| | | load/ loading SLA+W | 0 | 0 | 1 | x | will send SLA+W; will receive ACK or NACK; will switch to MT mode |
| 0x38 | SLA+R or data negotiation (arbitration) failed | no action | 0 | 0 | 1 | x | will release the bus; will enter unaddressed slave mode |
| | | no action | 0 | 0 | 1 | x | will send START when idle |
| 0x40 | SLA+R sent; ACK received | no action | 0 | 0 | 1 | 0 | will receive data; will send NACK |
| | | no action | 0 | 0 | 1 | 1 | will receive data; ACK will be sent |
| 0x48 | SLA+R sent; received NACK | no action | 1 | 0 | 1 | x | will send REPEATED START |
| | | no action | 0 | 1 | 1 | x | will send STOP; will reset the TWSTO flag |
| | | no action | 1 | 1 | 1 | x | will send STOP; will reset the TWSTO flag; will send START |
| 0x50 | data bytes received; ACK sent | read data | 0 | 0 | 1 | 0 | will receive data; will send NACK |
| | | read data | 0 | 0 | 1 | 1 | will receive data; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | ACK will be sent |
| 0x58 | data bytes received; NACK sent | read data | 1 | 0 | 1 | x | will send REPEATED START |
| | | read data | 0 | 1 | 1 | x | will send STOP; will reset the TWSTO flag |
| | | read data | 1 | 1 | 1 | x | will send STOP; will reset the TWSTO flag; will send START |

The format and status of the host receiving mode are shown in the following figure:




Format and State Diagram for Host Receive Mode

## Slave Receive Mode

In slave receive mode, a certain number of data bytes can be received from the master transmitter. The status codes mentioned in this section assume that the prescaler control flag is '0'.

To enable slave receive mode, set the TWAR and TWCR registers.

TWAR needs to be set as follows:

| TWA6 | TWA5 | TWA4 | TWA3 | TWA2 | TWA1 | TWA0 | TWGCE |
|---|---|---|---|---|---|---|---|
| Device Slave Address | | | | | | | |

The upper 7 bits of TWAR are the slave address that the TWI interface will respond to when the master addresses it. If LSB is set, TWI will respond to the general call address (0x00), otherwise ignore the general call address.

TWCR needs to be set as follows:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | x |

TWEN must be set to enable the TWI interface, and TWEA must be set to return an acknowledgment message ACK when the master addresses (slave address or general call) to itself. TWSTA and TWSTO must be cleared.

After initializing TWAR and TWCR, the TWI interface starts to wait until its own slave address (or broadcast address) is addressed. When the data direction flag following the slave address is "0" (indicating a write operation), the TWI enters the slave receive mode. When the data direction flag is "1" (indicating a read operation), the TWI enters slave transmit mode. After receiving its own slave address and write operation flag, the TWINT flag is set, and the valid status code is also updated to TWSR. Appropriate responses for each status code are described in detail in the status code table. It should be noted that the slave receiver mode can also be entered after the TWI arbitration in the master mode fails (see status codes 0x68 and 0x78).

If the TWEA flag is reset during a transfer, TWI will return NACK (high) to the SDA line after receiving a byte. This can be used to indicate that the slave cannot receive more data. When the TWEA flag is "0", the TWI will not respond to its own slave address. However, the TWI will still listen to the bus, and once TWEA is set, address recognition and response can be resumed. That is, TWEA can be used to temporarily isolate the TWI interface from the bus.

In other sleep modes except idle mode, the clock of the TWI interface can be turned off. If the slave receiver mode is enabled, the interface will continue to respond to the slave address or broadcast address using the bus clock. An address match will wake up the MCU. During wake-up, the TWI interface will keep SCL low until the TWINT flag is cleared. When the clock of the TWI interface returns to normal, more data can be received.

The status codes of the slave receiver mode are shown in the following table:

Status code table of slave receiver mode

| status code | Bus and Hardware Status | Application software response | | | | | The next move for the hardware |
|---|---|---|---|---|---|---|---|
| | | read/write TWDR | Action on TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0x60 | SLA+W received; ACK sent | no action | x | 0 | 1 | 0 | will receive data; will send NACK |
| | | no action | x | 0 | 1 | 1 | will receive data; will send ACK |
| 0x68 | negotiation (arbitration) failed when sending SLA+R/W; SLA+W received; ACK sent | no action | x | 0 | 1 | 0 | will receive data; will send NACK |
| | | no action | x | 0 | 1 | 1 | will receive data; will send ACK |
| 0x70 | broadcast address received; ACK sent | no action | x | 0 | 1 | 0 | will receive data; will send NACK |
| | | no action | x | 0 | 1 | 1 | will receive data; will send ACK |
| 0x78 | negotiation (arbitration) failed when sending SLA+R/W; SLA+W received; ACK sent | no action | x | 0 | 1 | 0 | will receive data; will send NACK |
| | | no action | x | 0 | 1 | 1 | will receive data; will send ACK |
| 0x80 | addressed (own) data received; ACK sent | read data | x | 0 | 1 | 0 | will receive data; will send NACK |
| | | read data | x | 0 | 1 | 1 | will receive data; will send ACK |
| 0x88 | addressed (own) data received; NACK sent | read data | 0 | 0 | 1 | 0 | will switch to unaddressed slave mode; will not respond to slave addresses and broadcasts |
| | | read data | 0 | 0 | 1 | 1 | will switch to unaddressed slave mode; will respond to the slave address; TWGCE=1 will respond to broadcast |
| | | read data | 1 | 0 | 1 | 0 | will switch to unaddressed slave mode; will not respond to slave |

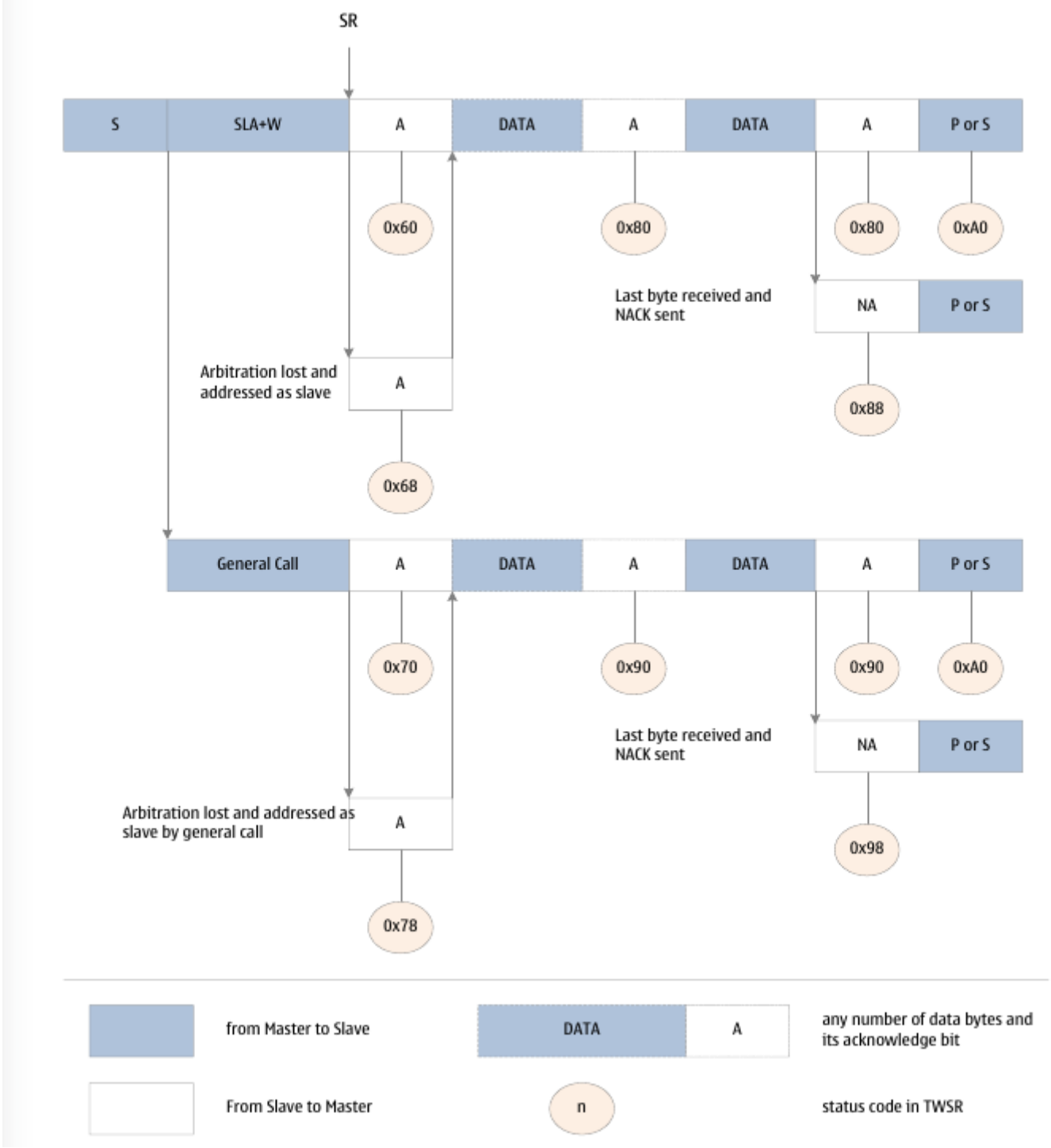
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | addresses and broadcasts;<br>START will be sent when the bus is idle |
| | | read data | 1 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast;<br>START will be sent when the bus is idle |
| 0x90 | broadcast data received; ACK sent | read data | x | 0 | 1 | 0 | will receive data;<br>will send NACK |
| | | read data | x | 0 | 1 | 1 | will receive data;<br>will send ACK |
| 0x98 | broadcast data received; NACK sent | read data | 0 | 0 | 1 | 0 | will switch to unaddressed slave mode;<br>will not respond to slave addresses and broadcasts |
| | | read data | 0 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast |
| | | read data | 1 | 0 | 1 | 0 | will switch to unaddressed slave mode;<br>will not respond to slave addresses and broadcasts;<br>START will be sent when the bus is idle |
| | | read data | 1 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast;<br>START will be sent when the bus is idle |
| 0xA0 | STOP or REPEATED START received when slave is working | no action | 0 | 0 | 1 | 0 | will switch to unaddressed slave mode;<br>will not respond to slave addresses and broadcasts |
| | | no action | 0 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast |
| | | no action | 1 | 0 | 1 | 0 | will switch to unaddressed slave mode; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | will not respond to slave addresses and broadcasts;<br>START will be sent when the bus is idle |
| | | no action | 1 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast;<br>START will be sent when the bus is idle |

*Intentionally left blank*

The format and state diagram of the slave receiver mode are as follows:

Format and State Diagram for Slave Receive Mode



## Slave Transmit Mode

In slave transmit mode, a certain number of data bytes can be sent to the master receiver. The status codes mentioned in this section assume that the prescaler control flag is '0'.

To enable slave receive mode, set the TWAR and TWCR registers.

TWAR needs to be set as follows:

| TWA6 | TWA5 | TWA4 | TWA3 | TWA2 | TWA1 | TWA0 | TWGCE |
|---|---|---|---|---|---|---|---|
| Device Slave Address | | | | | | | |

The upper 7 bits of TWAR are the slave address that the TWI interface will respond to when the master addresses it. If LSB is set, TWI will respond to the general call address (0x00), otherwise ignore the general call address.

TWCR needs to be set as follows:

| TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | x |

TWEN must be set to enable the TWI interface, and TWEA must be set to return an acknowledgment message ACK when the master addresses (slave address or general call) to itself. TWSTA and TWSTO must be cleared.

After initializing TWAR and TWCR, the TWI interface starts to wait until its own slave address (or broadcast address) is addressed. When the data direction flag following the slave address is "0" (indicating a write operation), the TWI enters the slave receive mode. When the data direction flag is "1" (indicating a read operation), the TWI enters slave transmit mode. After receiving its own slave address and read operation flag, the TWINT flag is set, and the valid status code is also updated to TWSR. Appropriate responses for each status code are described in detail in the status code table. It should be noted that when the TWI arbitration in the master mode fails, the slave transmission mode can also be entered (see status code 0xB0).

If the TWEA flag is reset during a transfer, the TWI will switch to Unaddressed Slave mode after the last byte has been sent. The status code in the TWSR register will be updated to 0xC0 or 0xC8 after the host receiver gives a NACK or ACK for the last byte of transmission. If the master receiver continues the transfer operation, the slave transmitter will not respond and the master will receive all "1" data (ie 0xFF). When the slave sends the last byte of data (TWEA is cleared) and expects a NACK response, and the master wants to receive more data and sends an ACK as a response, the TWSR will be updated to 0xC8.

When the TWEA flag is "0", the TWI will not respond to its own slave address. However, the TWI will still listen to the bus, and once TWEA is set, address recognition and response can be resumed. That is, TWEA can be used to temporarily isolate the TWI interface from the bus.

In other sleep modes except idle mode, the clock of the TWI interface can be turned off. If the slave receiver mode is enabled, the interface will continue to respond to the slave address or broadcast

address using the bus clock. An address match will wake up the MCU. During wake-up, the TWI interface will keep SCL low until the TWINT flag is cleared. When the clock of the TWI interface returns to normal, more data can be received.

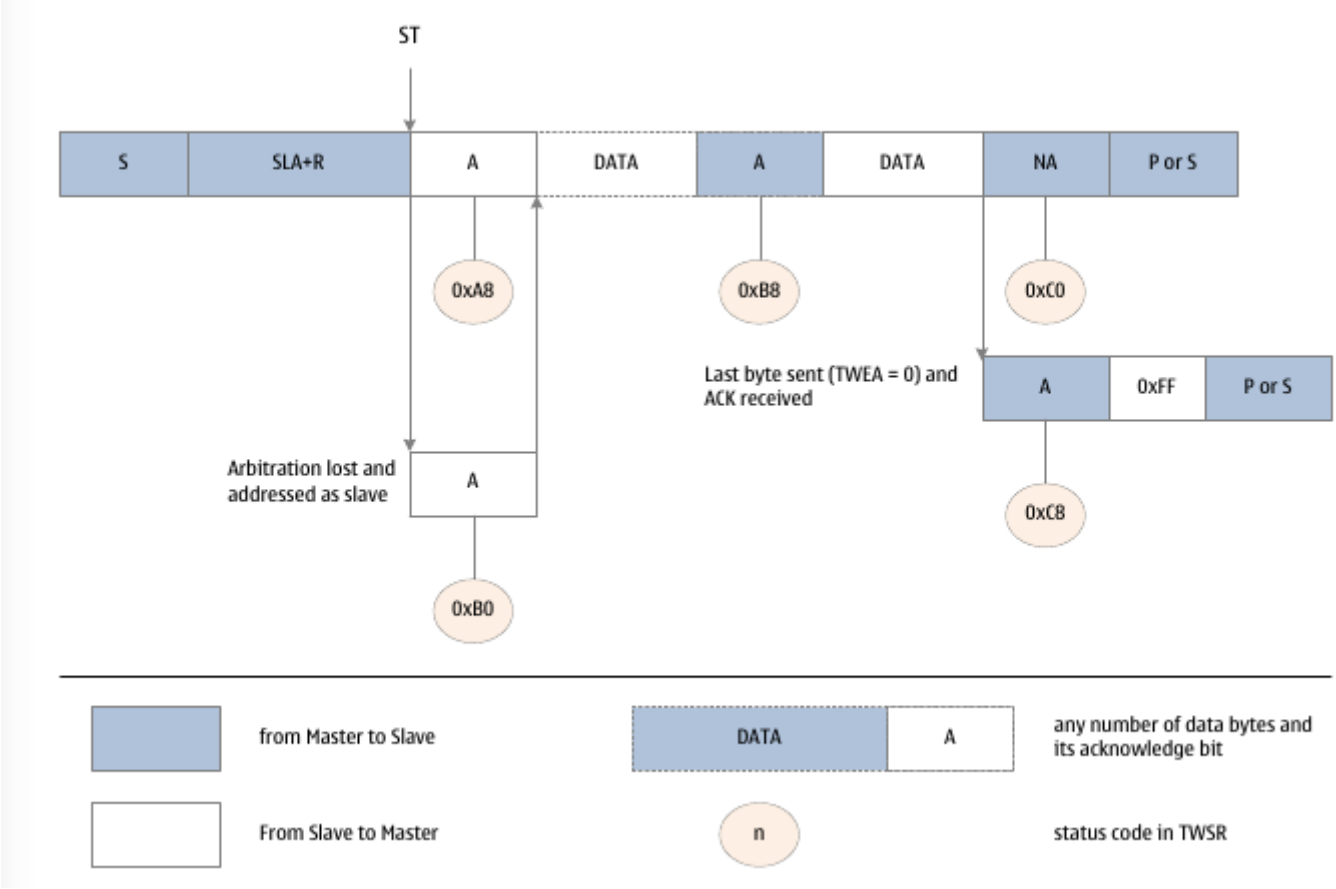
The status codes of slave sending mode are shown in the following table:

Status code table of slave transmit mode

| status code | Bus and Hardware Status | Application software response | | | | | The next move for the hardware |
|---|---|---|---|---|---|---|---|
| | | read/write TWDR | Action on TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0xA8 | SLA+R received; ACK has been sent | load data | x | 0 | 1 | 0 | will send the last data; expect to receive NACK |
| | | load data | x | 0 | 1 | 1 | will send data; will receive ACK |
| 0xB0 | negotiation (arbitration) failed when sending SLA+R/W; SLA+R received; ACK has been sent | load data | x | 0 | 1 | 0 | will send the last data; expect to receive NACK |
| | | load data | x | 0 | 1 | 1 | will send data; will receive ACK |
| 0xB8 | data has been sent; ACK received | load data | x | 0 | 1 | 0 | will send the last data; expect to receive NACK |
| | | load data | x | 0 | 1 | 1 | will send data; will receive ACK |
| 0xC0 | data has been sent; NACK received | no action | 0 | 0 | 1 | 0 | will switch to unaddressed slave mode; will not respond to slave addresses and broadcasts |
| | | no action | 0 | 0 | 1 | 1 | will switch to unaddressed slave mode; will respond to the slave address; TWGCE=1 will respond to broadcast |
| | | no action | 1 | 0 | 1 | 0 | will switch to unaddressed slave mode; will not respond to slave addresses and broadcasts; START will be sent when the bus is idle |
| | | no action | 1 | 0 | 1 | 1 | will switch to unaddressed slave mode; will respond to the slave address; TWGCE=1 will respond to broadcast; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | START will be sent when the bus is idle |
| 0xC8 | The last data has been sent;<br>ACK received | no action | 0 | 0 | 1 | 0 | will switch to unaddressed slave mode;<br>will not respond to slave addresses and broadcasts |
| | | no action | 0 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast |
| | | no action | 1 | 0 | 1 | 0 | will switch to unaddressed slave mode;<br>will not respond to slave addresses and broadcasts;<br>START will be sent when the bus is idle |
| | | no action | 1 | 0 | 1 | 1 | will switch to unaddressed slave mode;<br>will respond to the slave address;<br>TWGCE=1 will respond to broadcast;<br>START will be sent when the bus is idle |

The format and status of the slave sending mode are shown in the following figure:




Format and State Diagram of Slave Transmit Mode

## Other status

There are two status codes that do not have corresponding TWI status definitions, as shown in the following table:

Other Status Code Table

| status code | Bus and Hardware Status | Application software response | | | | | The next move for the hardware |
|---|---|---|---|---|---|---|---|
| | | read/write TWDR | Action on TWCR | | | | |
| | | | STA | STO | TWINT | TWEA | |
| 0xF8 | no state information; TWINT=0 | no action | Do not operate TWCR | | | | wait or perform current operation |
| 0x00 | bus error illegal START or STOP | no action | 0 | 1 | 1 | x | Only affects internal hardware; will not send STOP to the bus; Bus release and clear TWSTO flag |

The status code 0xF8 indicates that there is currently no relevant information, because the TWINT flag is "0". This state may occur when the TWI interface is not involved in a serial transfer or the current transfer has not completed.

Status 0x00 indicates a bus error during serial transfer. A bus error occurs when an illegal START or STOP occurs. For example, there is a START or STOP between address and data, address and ACK. A bus error will set TWINT. To recover from an error, TWSTO must be set and TWINT must be cleared by writing a '1'. This will put the TWI interface into Unaddressed Slave mode without generating a STOP, release SCL and SDA, and clear the TWSTO flag.

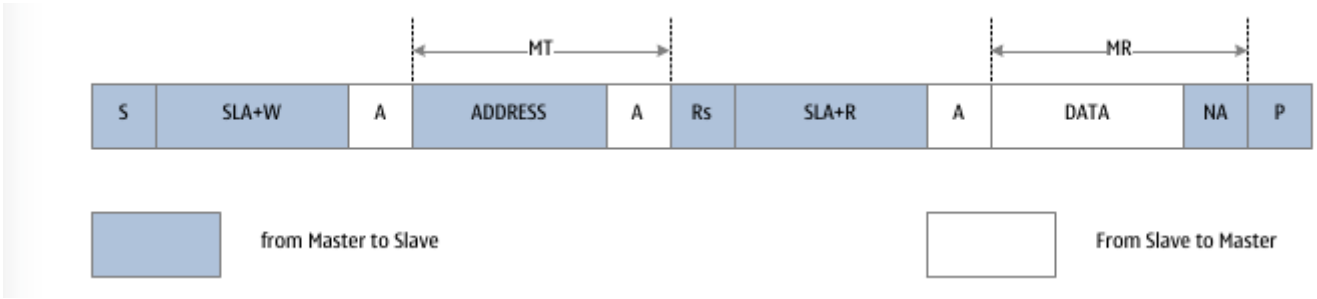
Combination mode

In some cases, several TWI modalities must be combined in order to accomplish the desired work. For example, to read data from a serial EEPROM, a typical transfer consists of the following steps:

1. The transfer must be started;
2. The EEPROM must be told where the data should be read;
3. The read operation must be completed;
4. The transfer must end.

Note that data can be transferred from master to slave and vice versa. The master tells the slave where to read the data, using the master sending mode. Next, read data from the slave, using the master receive mode. The direction of transmission will change. The host must maintain control of the bus at all stages, and all steps are uninterrupted operations. If in a multi-host system, another host changes the position of reading data between steps 2 and 3, this principle is broken, and the position of the host reading data will be wrong. Changing the direction of a data transfer is accomplished by sending a REPEATED START between sending the address byte and receiving the data. After sending REPEATED START, the master still has control of the bus.

The following diagram describes this transfer process:




Combining Multiple TWI Modes to Access Serial EEPROM Map

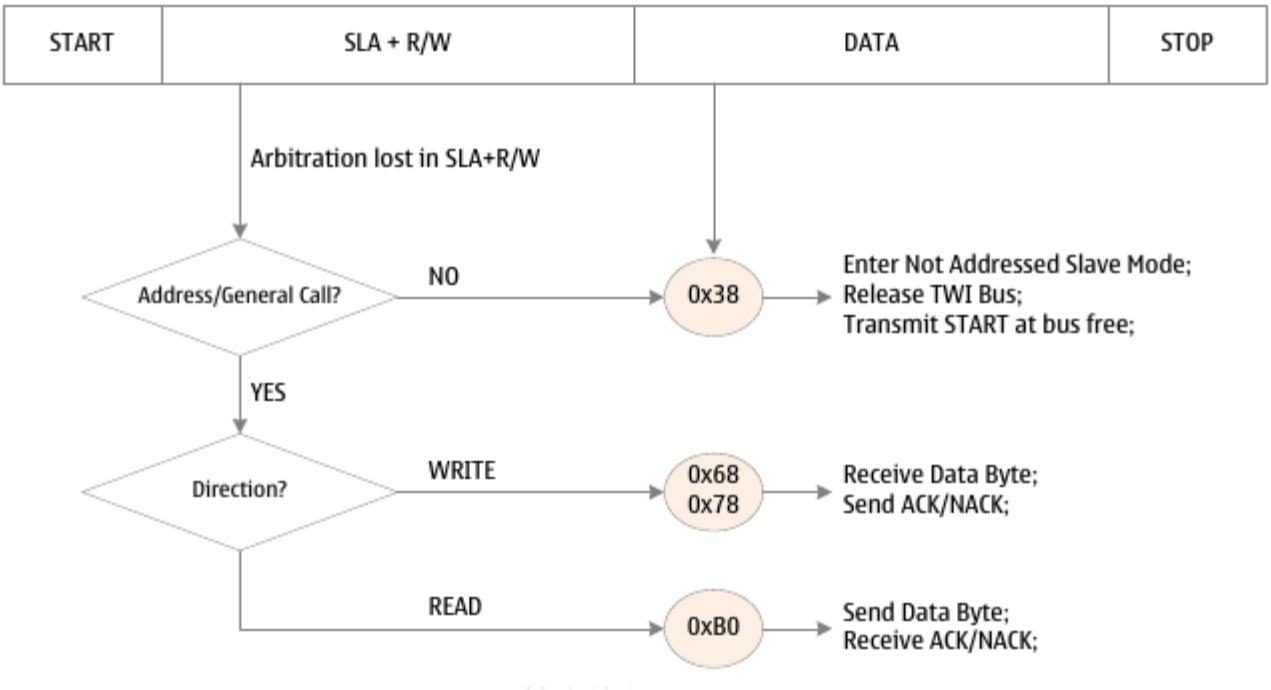
## Multi-Master Systems and negotiations (Arbitrations)

If there are multiple hosts connected to the same TWI bus, one or more of them may start data transfer at the same time. The TWI protocol ensures that in this case, through an arbitration process, one of the hosts is allowed to transmit without losing data. The process of bus arbitration is described below by taking two masters trying to send data to the slave as an example.

There are several different situations that generate the bus arbitration process:

- Two or more masters communicate with a slave simultaneously. In this case, neither the master nor the slave is aware of contention on the bus;
- Two or more masters simultaneously access the same slave in different data or operation directions. In this case arbitration will occur, either on the READ/WRITE flag or the data bit. When another master sends a '0' on the SDA line, the master that sends a '1' on the SDA line loses arbitration. The failed master will switch to unaddressed slave mode, or wait for the bus to be free and send a new START signal, depending on the application software operation.
- Two or more masters access different slaves. In this case, bus arbitration occurs during the SLA phase. When another master sends a "0" to the SDA line, the master that sends a "1" to the SDA line will lose arbitration. A master that fails in SLA bus arbitration will switch to slave mode and check to see if it is addressed by the master that has gained control of the bus. If addressed, it will enter SR or ST mode, depending on the READ/WRITE flag following the SLA. If not addressed, it will switch to unaddressed slave mode, or wait for the bus to be free and send a new START signal, depending on the application software operation.

The following diagram describes the process of bus arbitration:




Bus Arbitration Process Diagram

## Register definition

TWI register list

| Register | Address | Defaults | Description |
|---|---|---|---|
| TWBR | 0xB8 | 0x00 | TWI bit rate register |
| TWSR | 0xB9 | 0x00 | TWI Status Register |
| TWAR | 0xBA | 0x00 | TWI address register |
| TWDR | 0xBB | 0x00 | TWI data register |
| TWCR | 0xBC | 0x00 | TWI Control Register |
| TWAMR | 0xBD | 0x00 | TWI address mask register |

### *TWBR – TWI Bit Rate Register*

| TWBR – TWI Bit Rate Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB8 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TWBR7 | TWBR6 | TWBR5 | TWBR4 | TWBR3 | TWBR2 | TWBR1 | TWBR0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TWBR[7:0] | TWI Bit rate selection control bits.<br>TWBR is the bit rate generator divide factor. The bit rate generator is a divider used to generate the SCL clock in master mode. The formula for calculating bit rate is as follows:<br>fscl = fsys/(16 + 2*TWBR*4TWPS) |

### *TWSR – TWI Status Register*

| TWSR – TWI Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xB9 | | | | | Default: 0xF8 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | TWS7 | TWS6 | TWS5 | TWS4 | TWS3 | - | TWPS1 | TWPS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:3 | TWS[7:3] | TWI status flag.<br>The 5-bit TWS reflects the state of the TWI logic and bus. Different status values have different meanings, see the description of TWI working mode for details. The value read from TWSR includes a 5-bit status value and a 2-bit prescaler control flag. When detecting the status, the prescaler flag should be masked to "0". This is state detection independent of the prescaler setting. |
| 2 | - | Reserve |
| 1 | TWPS1 | TWI prescaler control high flag.<br>TWPS1 and TWPS0 together form TWPS[1:0], which is used to control the bit rate prescale factor, and together with TWBR to control the bit rate. |
| 0 | TWPS0 | TWI prescaler control low flag.<br>TWPS0 and TWPS1 together form TWPS[1:0], which is used to control the bit rate prescale factor, and together with TWBR to control the bit rate. |

| TWPS[1:0] | Prescaler |
|---|---|
| 0 | 1 |
| 1 | 4 |
| 2 | 16 |
| 3 | 64 |

### *TWAR – TWI address register*

| TWAR – TWI address register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xBA | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | TWAR6 | TWAR5 | TWAR4 | TWAR3 | TWAR2 | TWAR1 | TWAR0 | TWGCE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:1 | TWA[6:0] | TWI slave address bits. |

| | | |
|---|---|---|
| | | TWA is the TWI slave address. When TWI works in slave mode, TWI will respond according to this address. Host mode does not require this address. But in a multi-host system, the slave address also needs to be set for other hosts to access. |
| 0 | TWGCE | TWI Broadcast Identification Enable control flag.<br>When the TWGCE flag is set to "1", the TWI bus broadcast identification is enabled.<br>When the TWGCE flag is set to "0", the TWI bus broadcast identification is disabled.<br>The TWI module will respond to this bus broadcast when TWGCE is set and the received address frame is 0x00. |

### *TWDR – TWI Data Register*

| TWDR – TWI Data Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xBB | | | | | Default: 0xFF | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | TWD7 | TWD6 | TWD5 | TWD4 | TWD3 | TWD2 | TWD1 | TWD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:0 | TWD[7:0] | TWI data register.<br>TWD is the next byte to be transmitted on the bus, or the previous byte just received from the bus. |

### *TWCR – TWI Control Register*

| TWCR – TWI Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xBC | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | TWINT | TWI interrupt flag.<br>The hardware will set the TWINT flag when the TWI has finished its current work and wants the application software to intervene.<br>If the global interrupt is set and the TWIE flag is set, a TWI interrupt will be generated, and the MCU will execute the TWI interrupt service routine. When the TWINT flag is set, the low level of the SCL signal will be stretched.<br>The TWINT flag can only be cleared by writing "1" to this flag. Even if the interrupt service routine is executed, the hardware will not automatically clear this flag. Also note that clearing this flag will immediately enable TWI operation. Therefore, before clearing the TWINT flag, accesses to the TWAR, TWAMR, TWSR and TWDR registers must be completed first. |
| 6 | TWEA | TWI enable acknowledge control flag.<br>The TWEA flag controls the generation of the acknowledge pulse.<br>When the TWEA flag is set to "1" and one of the following conditions is met, an acknowledge pulse will be generated on the TWI bus:<br>1) Receive the slave address of the device;<br>2) A general call is received when TWGCE is set;<br>3) One byte of data is received in master receive or slave receive mode.<br>When the TWEA flag is set to '0', the device is temporarily disconnected from the TWI bus. When asserted, the device resumes address recognition. |
| 5 | TWSTA | TWI start state control flag.<br>The TWSTA flag is set when the CPU wants to be the master on the TWI bus. The hardware will detect if the bus is available, and when the bus is free, it will generate a start condition on the bus. When the bus is not idle, the TWI will wait until it detects a stop condition and then generate a start condition to declare that it wishes to become the master. Software must clear the TWSTA flag after sending the start state. |
| 4 | TWSTO | TWI stop status control flag.<br>In master mode when the TWSTO flag is "1", the TWI will generate a stop state on the bus and then automatically clear the TWSTO flag. In slave mode, setting the TWSTO flag allows the TWI to recover from an error state. There is no stop state at this time, only the TWI returns to a defined unaddressed slave mode, and the SCL and SDA signal lines are released to the high-impedance state. |

| | | |
|---|---|---|
| 3 | TWWC | TWI write conflict flag.<br>When the TWINT flag is low, writing to the TWDR register will set the TWWC flag.<br>When the TWINT flag is high, writing to the TWDR register will clear the TWWC flag. |
| 2 | TWEN | TWI enable control flag.<br>The TWEN flag enables TWI operation and activates the TWI interface. When the TWEN flag is set to "1", the TWI control IO pins are connected to the SCL and SDA pins. When the TWEN flag is set to "0", the TWI interface module is turned off and all transfers are terminated, including ongoing operations. |
| 1 | - | Reserve |
| 0 | TWIE | TWI interrupt enable control flag.<br>When the TWIE flag is set to "1" and the global interrupt is set, the TWI interrupt request is activated as long as the TWINT flag is high. |

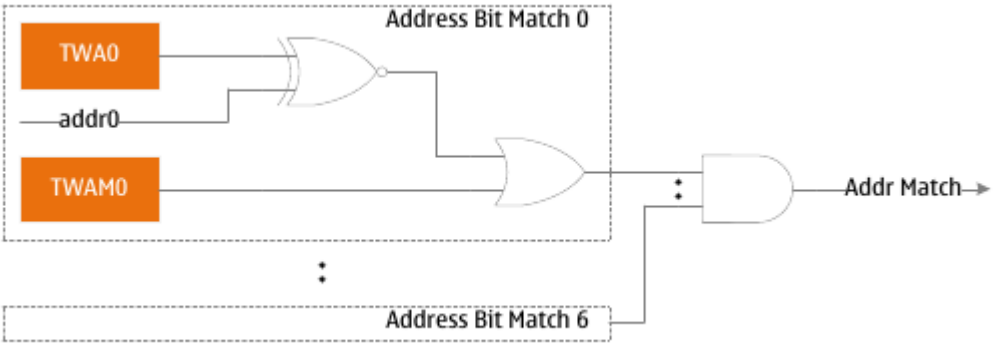
### *TWAMR – TWI Address Mask Register*

| TWAMR – TWI Address Mask Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xBD | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | TWAR6 | TWAR5 | TWAR4 | TWAR3 | TWAR2 | TWAR1 | TWAR0 | TWGCE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:1 | TWAM[6:0] | TWI address mask control bits.<br>TWAM is the 7-bit TWI slave address mask control. Each bit of TWAM is used to mask (disable) the corresponding address bit in TWAR. When the mask bit is set, the address match logic will ignore the result of comparing the received address bit with the corresponding bit in TWA. The following figure gives the details of the address matching logic. |
| 0 | - | Reserve |

### *TWI address matching logic*

The following figure shows the logic block diagram of TWI address matching:




# Analog Comparator 0 (AC0)

- 10mV Comparison Accuracy
- Factory offset calibration
- Supports 3 off-chip analog inputs
- Multiplexed input (ADMUX) with ADC support
- Supports Internal Differential Amplifier Input (DFFO)
- Supports internal 8-bit DAC input (DAO)
- Programmable output digital filter control

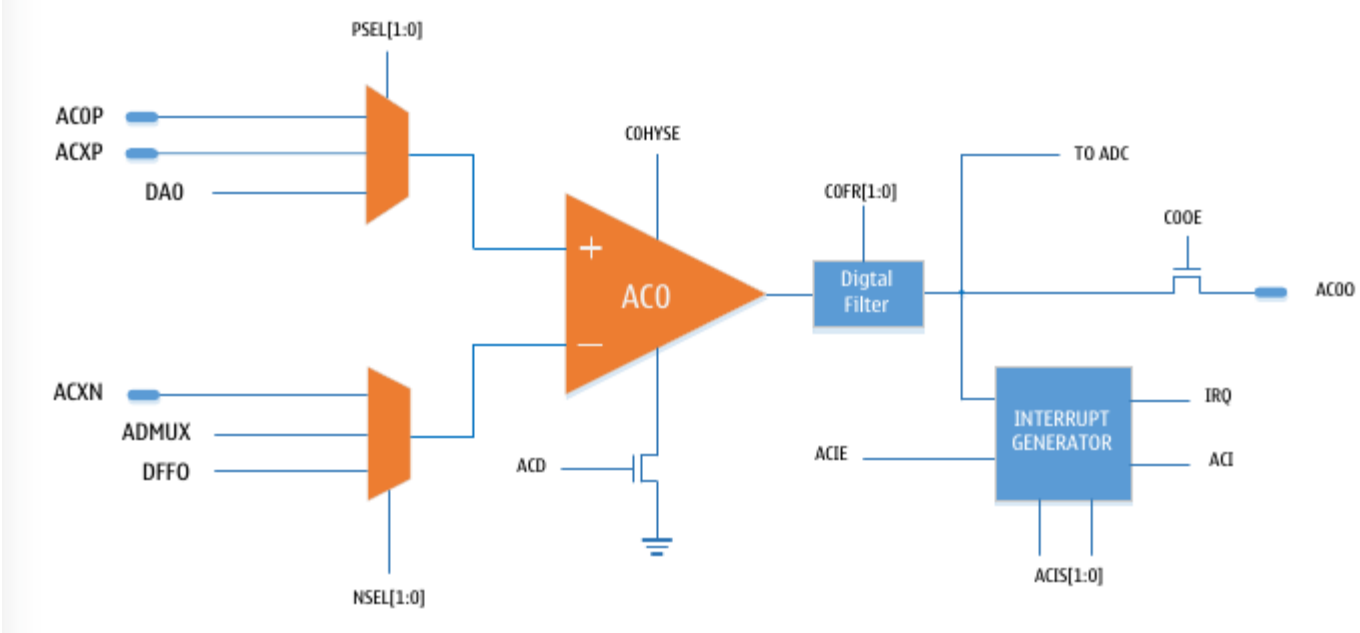
## Overview

The analog comparator compares the level of the positive terminal and the negative terminal of the input comparator. When the positive terminal voltage is higher than the negative terminal voltage, the output ACO of the analog comparator is set. When the level of ACO changes, the edge of the signal can be used to trigger an interrupt. The output signal ACO can also be used to trigger the timer counter 1 input capture and to control the PWM output generated by the timer.

The LGT8FX8P integrates the analog comparator AC0, including a multi-channel analog input selector. The positive and negative input sources of the comparator can be selected from an external port or from a variety of internally generated reference sources. The analog comparator itself supports offset calibration, which can ensure the consistency of the comparator operation. The comparator supports an optional hardware hysteresis function to improve the stability of the comparator output. At the same time, a hardware programmable digital filter is integrated at the output end of the comparator, and an appropriate filter setting can be selected according to the application requirements to obtain a more stable comparison output.

The output state of the comparator can be read directly through the register, and an interrupt request can also be generated to achieve a more efficient real-time event capture function. The output of the comparator can also be directly output to the external IO port.

The block diagram of op amp/analog comparator 0 is shown in the following figure.




Analog Comparator 0 Function Diagram

## Analog Comparator Input

Both inputs of the analog comparator support a variety of selectable input sources. Positive input three-way optional:

1. External independent analog input AC0P
2. Analog comparator 0/1 common analog input ACXP
3. Internal 8-bit DAC output DAO

The selection of the input source is controlled by the C0BG flag in the control status register C0SR and the C0PS0 flag in the C0XR register. For details, please refer to the register description part of this chapter.

AC0P is a dedicated positive mode input channel for AC0. Note that the pins of AC0P are slightly different in different packages. The AC0P in the QFP48 package is an independent port. QFP32 encapsulates this AC0P port in parallel with PD6 on one port.

ACXP is the common positive input of comparator 0/1. There are two analog comparators inside the LGT8FX8P, and ACXP is connected to the positive-side multiplexing selectors of the two comparators at the same time, which is convenient to realize the cooperative work of the two comparators.

DAO comes from the output of the internal 8-bit DAC. The reference source for the DAC can be selected from the system power supply, the internal reference, or the input from an external reference. For the configuration of the DAC, please refer to the relevant chapters of the DAC.

| C0BG | C0PS0 | AC0 positive input source |
|---|---|---|
| 0 | 0 | AC0P |
| 0 | 1 | ACXP |
| 1 | 0 | DAO |
| 1 | 1 | Turn off the comparator positive input channel |

The negative input can also choose from three different analog inputs:

1. Comparator 0/1 common analog input ACXN
2. The output of the ADC multiplexer, ADMUX
3. Internal differential amplifier output DFFO

Comparator negative input channel selection is controlled by the CME00/01 bits in the ADCSRB register from the ADC module.

When the negative input of the comparator is selected as ADMUX, the analog input channel needs to be selected through the CHMUX flag of the ADMUX register of the ADC module. In this mode, the input of the comparator can be expanded more flexibly.

ACXN is the common negative input of comparator 0/1, which is convenient to realize the cooperative work of comparator 0/1;

DFFO comes from the internal differential amplifier output. The differential amplifier has optional x1/ x8/x16/x32 gain control, which can realize the detection and measurement of small signals.

| CME01 | CME00 | AC0 negative input source |
|---|---|---|
| 0 | 0 | ACXN |
| 0 | 1 | ADMUX |
| 1 | 0 | DFFO |
| 1 | 1 | Turn off the negative input channel of the comparator |

## Comparator output filtering

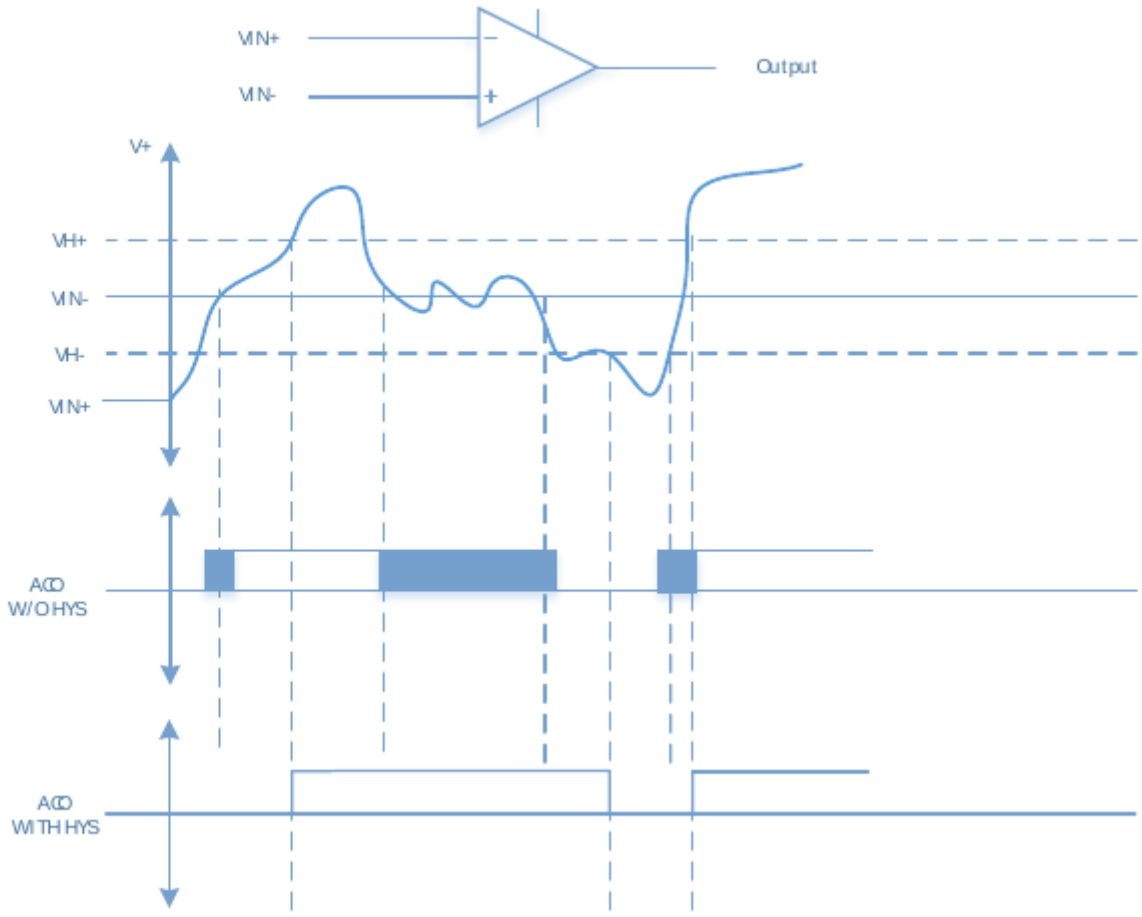
A controllable hysteresis is supported internally at the comparator output. The user can enable the hysteresis circuit via the C0HYSE flag in the C0XR register. The hysteresis circuit can eliminate the unstable state of the comparator state change process and achieve the output filtering function.

It is recommended that users turn on the hysteresis circuit to obtain a stable comparator output when using the comparator.

As shown in the figure below, the hysteresis circuit is located between the analog output of the comparator and the digital output. When the input voltage VIN+ of the positive terminal of the comparator is greater than (VIN- + VH+), the output of the comparator COUT is high; when the voltage of VIN+ is less than (VIN- - VH-), the output of the comparator is low.

The hysteresis circuit avoids the jitter caused by the circuit itself when the positive terminal voltage of the comparator is close to the negative terminal voltage.

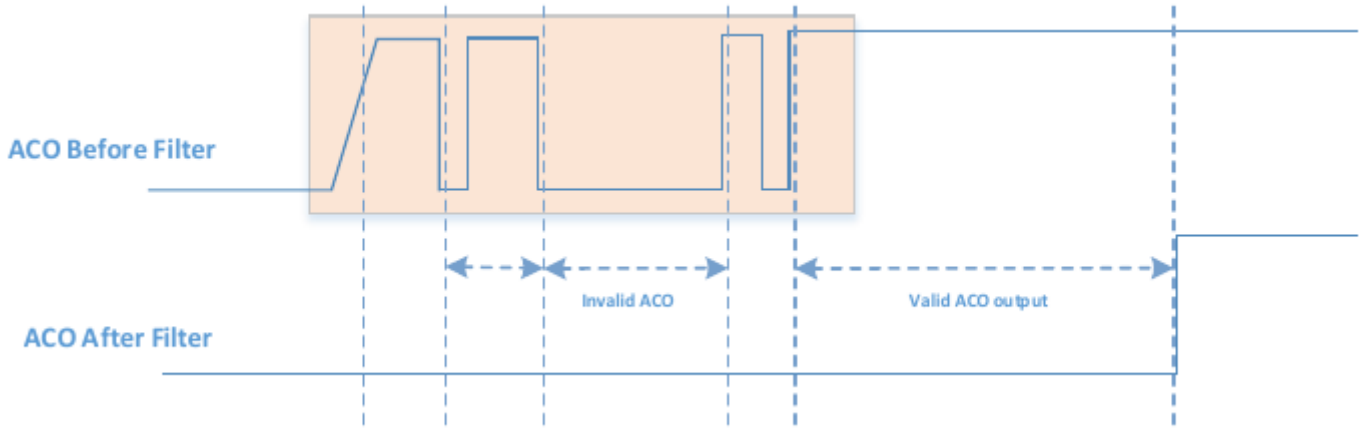
Comparator hysteresis voltage vs. comparator output graph:




Although the hysteresis circuit is very effective in suppressing the voltage ripple close to the threshold of the comparator, in the actual application environment, the input signal will be disturbed by different strengths. Strong interference may cause the input level to rise instantaneously, which exceeds the threshold range of the hysteresis circuit and cannot be effectively suppressed. The LGT8FX8P integrates a programmable digital filter at the comparator output, which can filter out the

effects of transient disturbances on the comparator output. The digital filter can choose an appropriate filter time width according to the application requirements. Only when the output of the comparator is stable and continuously meets the filter time limit, the filter circuit will update the output of the comparator.

So as to achieve a more stable output result.




The digital filtering of AC0 is controlled by the C0FEN and C0FS bits of the C0XR register. For the specific setting method, please refer to the register definition part of this chapter.

## Comparator output and PWM control

LGT8FX8P supports multi-channel PWM output, the PWM signal can be used with the comparator module. The output of the comparator can be used to directly turn off the PWM signal, thus realizing a more flexible PWM protection scheme.

For the control related to PWM output, please refer to the relevant part of the Timer chapter.

## Register definition

### *C0SR – AC0 Control and Status Register*

| C0SR – AC0 Control and Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x50 | | | | | Default: 0x80 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | C0D | C0BG | C0O | C0I | C0IE | C0IC | C0IS1 | C0IS0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | C0D | Analog comparator disable flag.<br>When the C0D flag is set to "1", the analog comparator is turned off.<br>When the C0D flag is set to "0", the analog comparator is turned on. |
| 6 | C0BG | Analog Comparator 0 Positive Input Source Select Bit. The C0BG and C0PS0 bits of the C0XR register together set the AC0 positive input source, {C0BG, C0PS0} =<br>00 = AC0P as positive input<br>01 = ACXP as positive input<br>10 = output of internal DAC as positive input<br>11 = Turn off the positive input source of AC0 |
| 5 | C0O | Analog comparator output status bits.<br>The output of the analog comparator is synchronized directly to the C0O flag. Software can read the value of the C0O flag to obtain the output value of the analog comparator. |

<table>
<tr><td>4</td><td>C0I</td><td>The interrupt flag for the analog comparator.<br>The C0I flag is set when an output event from the analog comparator triggers the interrupt mode defined by the C0IS flag. An interrupt is generated when the interrupt enable flag C0IE is "1" and the global interrupt is set.<br>When the analog comparator interrupt service routine is executed, C0I will be cleared automatically, or it can be cleared by writing "1" to the C0I flag.</td></tr>
<tr><td>3</td><td>C0IE</td><td>Interrupt enable flag for the analog comparator.<br>When the C0IE flag is set to 1 and the global interrupt is enabled, the AC0 interrupt is enabled.<br>When the C0IE flag is set to 0, the AC0 interrupt is disabled.</td></tr>
<tr><td>2</td><td>C0IC</td><td>Analog Comparator Input Capture Enable Bit<br>C0IC = 1, the input capture source of timer counter 1 comes from the output of the analog comparator.<br>C0IC = 0, the input capture source of timer counter 1 comes from external pin ICP1.</td></tr>
<tr><td>1</td><td>C0IS1</td><td>Analog Comparator Interrupt Mode Control High.</td></tr>
<tr><td rowspan="6">0</td><td rowspan="6">C0IS0</td><td>Analog Comparator Interrupt Mode Control Low. C0IS0 and C0IS1 together form C0IS[1:0], which is used to control the interrupt trigger mode of the analog comparator.</td></tr>
<tr><td><table><tr><th>C0IS[1:0]</th><th>Interrupt mode</th></tr><tr><td>00</td><td>Trigger on rising or falling edge of ACO</td></tr><tr><td>01</td><td>Reserve</td></tr><tr><td>10</td><td>ACO falling edge trigger</td></tr><tr><td>11</td><td>ACO rising edge trigger</td></tr></table></td></tr>
</table>

### *ADCSRB – ADC Control and Status Register B*

| ADCSRB – ADC Control and Status Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7B | Default: 0x00 | | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | CME01 | CME00 | CME11 | CME10 | ACTS | ADTS2 | ADTS1 | ADTS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | CME01 | AC0 negative input selection, CME0 = {CME01, CME00}<br>00: External port ACXN is used as AC0 negative input<br>01: ADC multiplexed output as AC0 negative input<br>10: Differential amplifier output as AC0 negative input<br>11: Turn off the negative input source of AC0 |
| 6 | CME00 | |
| 5 | CME11 | AC1 negative input selection, CME1 = {CME11, CME10}<br>00: External port ACXN is used as AC1 negative input<br>01: External port AC1N as AC1 negative input<br>10: ADC internal 1/5 voltage divider as AC1 negative input |
| 4 | CME10 | |

| | | |
|---|---|---|
| | | 11: The output of the differential op amp is used as the negative input of AC1 |
| 3 | ACHS | AC trigger source channel selection<br>0 – AC0 output as ADC auto-conversion trigger source<br>1 – AC1 output as ADC auto-conversion trigger source |
| 2:0 | ADTS | See ADC register description. |

### *C0XR – AC0 Auxiliary Control Register*

| C0XR – AC0 Auxiliary Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x51 | | Default: 0x00 | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | C0OE | C0HYSE | C0PS0 | C0WKE | C0FEN | C0FS1 | C0FS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | - | Reserve |
| 6 | C0OE | AC0 Comparator output to external port enable control<br>C0OE = 1, the comparator output of AC0 is output to external port PD2<br>C0OE = 0, disable comparator output to external port |
| 5 | C0HYSE | AC0 output hysteresis function enable control.<br>1 = Enable output hysteresis<br>0 = disable output hysteresis |
| 4 | C0PS0 | AC0 Positive input source selection low flag.<br>C0PS0 and C0BG jointly control the positive input source of AC0, please refer to the C0SR register definition |
| 3 | C0WKE | AC0 is used to enable control of wake-up from sleep.<br>1 = Enables the wake-up function of the comparator output<br>0 = disables the wake-up function of the comparator output |
| 2 | C0FEN | Comparator digital filter enable control.<br>1 = digital filter enabled<br>0 = digital filter disabled |
| 1:0 | C0FS[1:0] | Comparator digital filter width setting<br>00 = off<br>01 = 32us<br>10 = 64us<br>11 = 96us |

## Analog Comparator 1 (AC1)

- 10mV Comparison Accuracy
- Factory offset calibration
- Supports 4 off-chip analog inputs

- Supports Internal 1/5 Voltage Divider Input (VDO)
- Supports Internal Differential Amplifier Input (DFFO)
- Supports internal 8-bit DAC input (DAO)
- Programmable output filter control

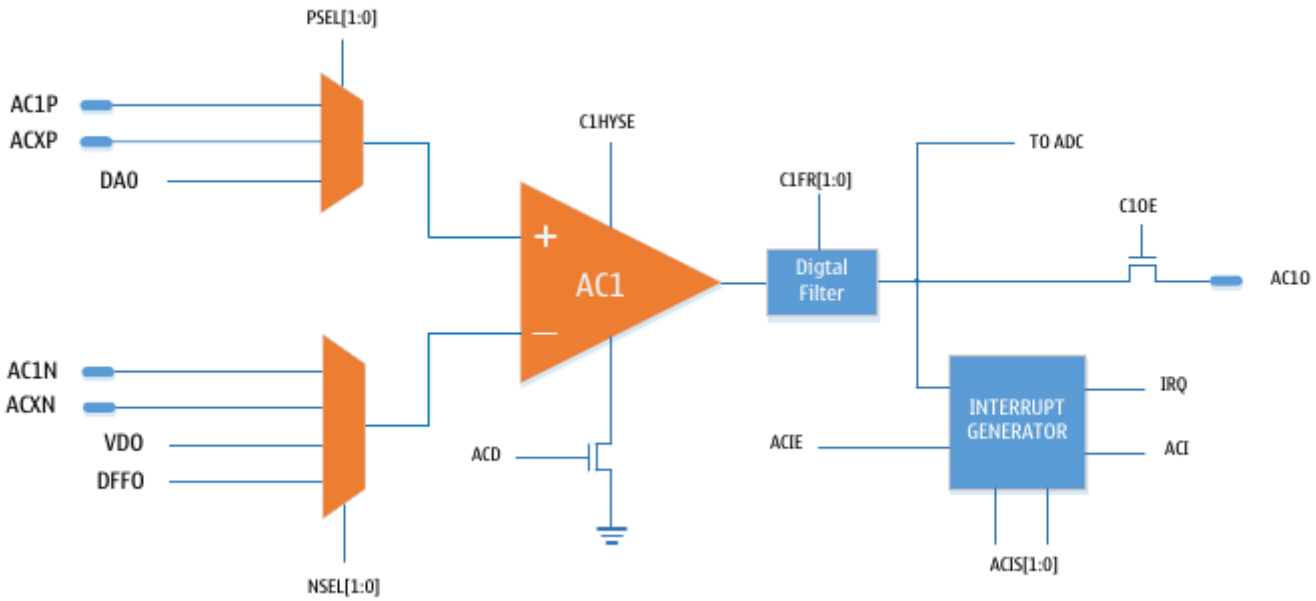
## Overview

The analog comparator compares the level of the positive terminal and the negative terminal of the input comparator. When the positive terminal voltage is higher than the negative terminal voltage, the output ACO of the analog comparator is set. When the level of ACO changes, the edge of the signal can be used to trigger an interrupt. The output signal ACO can also be used to trigger the timer counter 1 input capture and to control the PWM output generated by the timer.

The LGT8FX8P integrates the analog comparator AC1, including a multi-channel analog input selector. The positive and negative input sources of the comparator can be selected from an external port or from a variety of internally generated reference sources. The analog comparator itself supports offset calibration, which can ensure the consistency of the comparator operation. The comparator supports an optional hardware hysteresis function to improve the stability of the comparator output. At the same time, a hardware programmable digital filter is integrated at the output end of the comparator, and an appropriate filter setting can be selected according to the application requirements to obtain a more stable comparison output.

The output state of the comparator can be read directly through the register, and an interrupt request can also be generated to achieve a more efficient real-time event capture function. The output of the comparator can also be directly output to the external IO port.

The block diagram of analog comparator 1 is shown in the figure below.




Block Diagram of Analog Comparator 1

## Analog Comparator Input

Both inputs of the analog comparator support a variety of selectable input sources. Positive input three-way optional:

1. External independent analog input AC1P
2. Analog comparator 0/1 common analog input ACXP
3. Internal 8-bit DAC output DAO

The selection of the input source is controlled by the C1BG flag in the control status register C1SR and the C1PS0 flag in the C1XR register. For details, please refer to the register description part of this chapter.

AC1P is a dedicated positive mode input channel for AC1.

ACXP is the common positive input of comparator 0/1. There are two analog comparators inside the LGT8FX8P, and ACXP is connected to the positive-side multiplexing selectors of the two comparators at the same time, which is convenient to realize the cooperative work of the two comparators.

DAO comes from the output of the internal 8-bit DAC. The reference source for the DAC can be selected from the system power supply, the internal reference, or the input from an external reference. For the configuration of the DAC, please refer to the relevant chapters of the DAC.

| C1BG | C1PS0 | AC1 positive input |
|---|---|---|
| 0 | 0 | ACXP |
| 0 | 1 | AC1P |
| 1 | 0 | DAO |
| 1 | 1 | Turn off the comparator positive input channel |

The negative input can also select 4 different analog inputs:

1. External analog input AC1N as AC1 negative input
2. Comparator 0/1 common negative input ACXN
3. ADC internal 1/5 voltage divider output as negative input of AC1
4. The internal differential amplifier output DFFO is used as the negative input of AC1

Comparator negative input channel selection is controlled by the CME11/10 bits in the ADCSRB register from the ADC module.

When the negative input of the comparator is selected as the output of the internal multiplexer of the ADC, it is necessary to select the input reference source of the multiplexer through the VDS flag of the ADCSRC register of the ADC module.

ACXN is the common negative input of comparator 0/1, which is convenient to realize the cooperative work of comparator 0/1;

DFFO comes from the internal differential amplifier output. The differential amplifier has optional x1/ x8/x16/x32 gain control, which can realize the detection and measurement of small signals.

| CME11 | CME10 | AC1 negative input |
|---|---|---|
| 0 | 0 | ACXN |
| 0 | 1 | AC1N |
| 1 | 0 | VDO |
| 1 | 1 | DFFO |

## Comparator output filtering

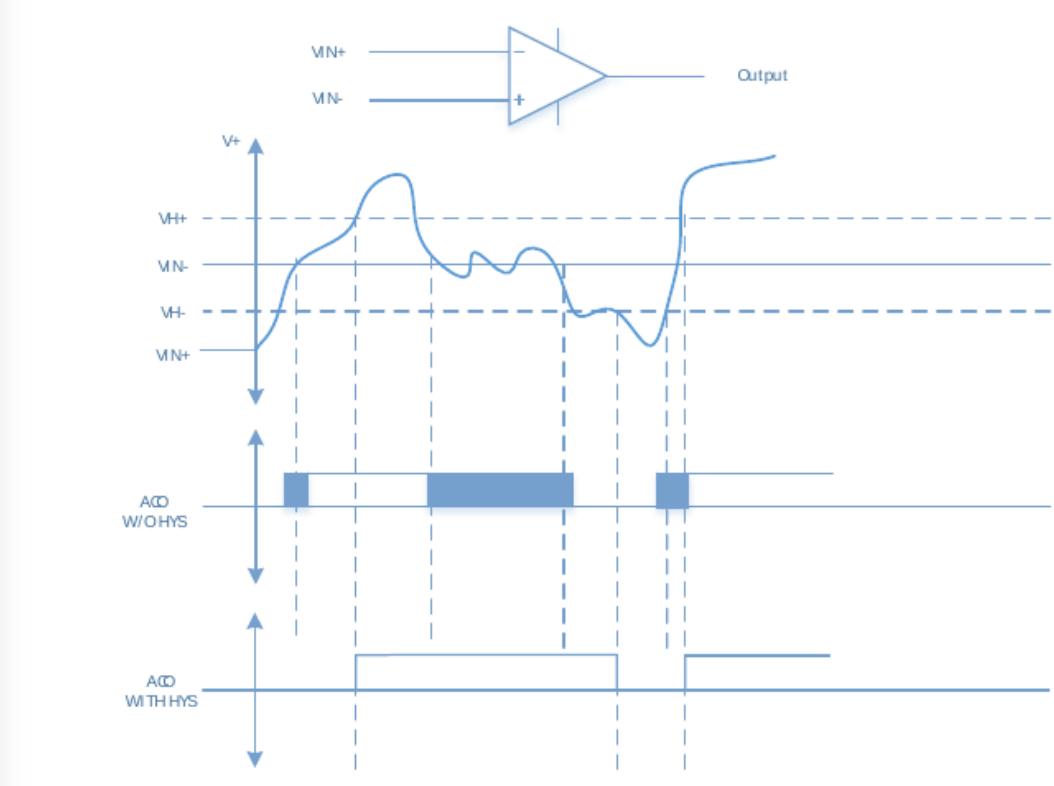
A controllable hysteresis is supported internally at the comparator output. The user can enable the hysteresis circuit via the C1HYSE flag in the C1XR register. The hysteresis circuit can eliminate the unstable state of the comparator state change process and achieve the output filtering function.

It is recommended that users turn on the hysteresis circuit to obtain a stable comparator output when using the comparator.

As shown in the figure below, the hysteresis circuit is located between the analog output of the comparator and the digital output. When the input voltage VIN+ of the positive terminal of the comparator is greater than (VIN- + VH+), the output of the comparator COUT is high; when the voltage of VIN+ is less than (VIN- - VH-), the output of the comparator is low.

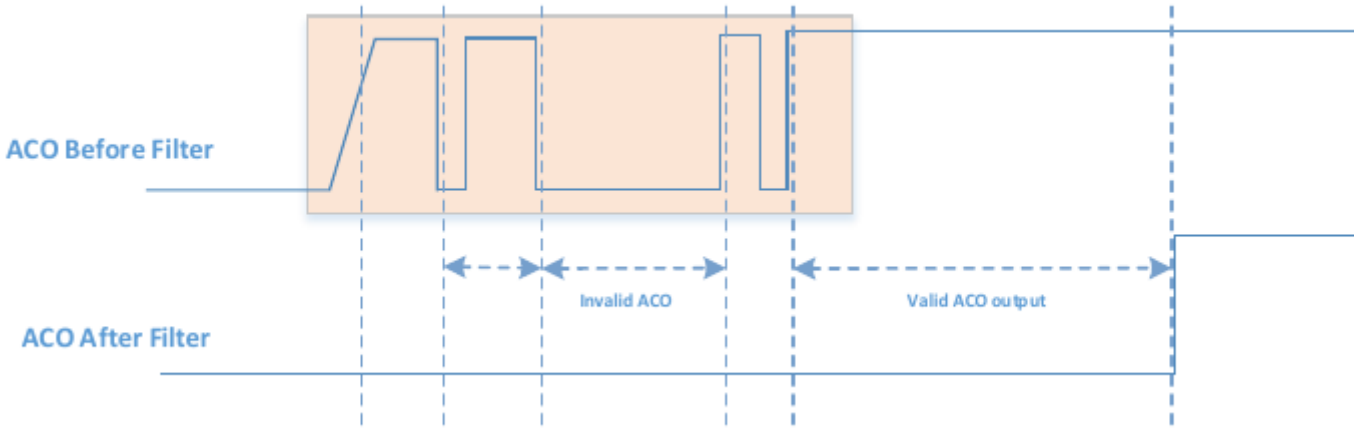
The hysteresis circuit avoids the jitter caused by the circuit itself when the positive terminal voltage of the comparator is close to the negative terminal voltage.

Comparator hysteresis voltage vs. comparator output graph:



Although the hysteresis circuit is very effective in suppressing the voltage ripple close to the threshold of the comparator, in the actual application environment, the input signal will be disturbed by different strengths. Strong interference may cause the input level to rise instantaneously, which exceeds the threshold range of the hysteresis circuit and cannot be effectively suppressed. The LGT8FX8P integrates a programmable digital filter at the comparator output, which can filter out the effects of transient disturbances on the comparator output. The digital filter can choose an appropriate filter time width according to the application requirements. Only when the output of the comparator is stable and continuously meets the filter time limit, the filter circuit will update the output of the comparator.

So as to achieve a more stable output result.




Comparator Output Filter Timing

The digital filtering of AC1 is controlled by the C0FEN and C1FS bits of the C1XR register. For the specific setting method, please refer to the register definition part of this chapter.

## Comparator output and PWM control

LGT8FX8P supports multi-channel PWM output, the PWM signal can be used with the comparator module. The output of the comparator can be used to directly turn off the PWM signal, thus realizing a more flexible PWM protection scheme.

For the control related to PWM output, please refer to the relevant part of the Timer chapter.

## Register definition

### *C1SR – AC1 Control and Status Register*

| C1SR – AC1 Control and Status Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x2F | | | | | Default: 0x80 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | C1D | C1BG | C1O | C1I | C1IE | C1IC | C1IS1 | C1IS0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | C1D | Analog comparator disable flag.<br>When the C1D flag is set to "1", the analog comparator is turned off.<br>When the C1D flag is set to "0", the analog comparator is turned on. |
| 6 | C1BG | Analog comparator 1 positive input source selection flag. The C1BG and C1PS0 bits of the C1XR register together set the AC1 positive input source, {C1BG, C1PS0} =<br>00 = ACXP as positive input<br>01 = AC1P as positive input<br>10 = output of internal DAC as positive input<br>11 = Turn off the positive input source of AC1 |
| 5 | C1O | Analog comparator output status bits.<br>The output of the analog comparator is synchronized directly to the C1O flag. Software can read the value of the C1O flag to obtain the output value of the analog comparator. |
| 4 | C1I | The interrupt flag for the analog comparator.<br>The C1I flag is set when an output event from the analog comparator triggers the interrupt mode defined by the C1IS flag. An interrupt is generated when the interrupt enable flag C1IE is "1" and the global interrupt is set. When the analog comparator interrupt service routine is executed, C1I will be automatically cleared, or write "1" to the C1I flag This flag can also be cleared. |
| 3 | C1IE | Interrupt enable flag for the analog comparator.<br>When the C1IE flag is set to 1 and the global interrupt is enabled, the AC1 interrupt is enabled.<br>When the C1IE flag is set to 0, the AC1 interrupt is disabled. |
| 2 | C1IC | Analog Comparator Input Capture Enable Bit<br>C1IC = 1, the input capture source of timer counter 1 comes from the output of the analog comparator.<br>C1IC = 0, the input capture source of timer counter 1 comes from external pin ICP1. |
| 1 | C1IS1 | Analog Comparator Interrupt Mode Control High. |
| 0 | C1IS0 | Analog Comparator Interrupt Mode Control Low. C1IS0 and C1IS1 together form C1PS[1:0], which is used to control the interrupt trigger mode of the analog comparator. |

| C1IS[1:0] | Interrupt mode |
|---|---|

| | | | |
|---|---|---|---|
| | | 00 | Trigger on rising or falling edge of AC1 |
| | | 01 | Reserve. |
| | | 10 | Falling edge trigger of AC1 |
| | | 11 | Triggered on the rising edge of AC1 |

### *ADCSRB – ADC Control and Status Register B*

| ADCSRB – ADC Control and Status Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7B | | Default: 0x00 | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | CME01 | CME00 | CME11 | CME10 | ACTS | ADTS2 | ADTS1 | ADTS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | CME01 | AC0 negative input selection, CME0 = {CME01, CME00}<br>00: External port ACXN is used as AC0 negative input<br>01: ADC multiplexed output as AC0 negative input<br>10: Differential amplifier output as AC0 negative input<br>11: Turn off the negative input source of AC0 |
| 6 | CME00 | |
| 5 | CME11 | AC1 negative input selection, CME1 = {CME11, CME10}<br>00: External port ACXN is used as AC1 negative input<br>01: External port AC1N as AC1 negative input<br>10: ADC internal 1/5 voltage divider as AC1 negative input<br>11: The output of the differential op amp is used as the negative input of AC1 |
| 4 | CME10 | |
| 3 | ACHS | AC trigger source channel selection<br>0 – AC0 output as ADC auto-conversion trigger source<br>1 – AC1 output as ADC auto-conversion trigger source |
| 2:0 | ADTS | See ADC register description. |

### *C1XR – AC1 Auxiliary Control Register*

| C1XR – AC1 Auxiliary Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x3A | | Default: 0x00 | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | - | C1OE | C1HYSE | C1PS0 | C1WKE | C1FEN | C1FS1 | C1FS0 |
| R/W | - | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | - | Reserve |
| 6 | C1OE | AC1 comparator output to external port enable control<br>C1OE = 1, the comparator output of AC1 is output to external port PE5<br>C1OE = 0, disable comparator output to external port |

| | | |
|---|---|---|
| 5 | C1HYSE | AC1 output hysteresis function enable control.<br>1 = Enable output hysteresis<br>0 = disable output hysteresis |
| 4 | C1PS0 | AC1 positive terminal input source selection low flag.<br>C1PS0 and C1BG jointly control the positive input source of AC1, please refer to the C1SR register definition |
| 3 | C1WKE | AC1 is used to enable control of wake-up from sleep.<br>1 = Enables the wake-up function of the comparator output<br>0 = disables the wake-up function of the comparator output |
| 2 | C1FEN | Comparator digital filter enable control.<br>1 = digital filter enabled<br>0 = digital filter disabled |
| 1:0 | C1FS[1:0] | Comparator digital filter width setting<br>00 = off<br>01 = 32us<br>10 = 64us<br>11 = 96us |

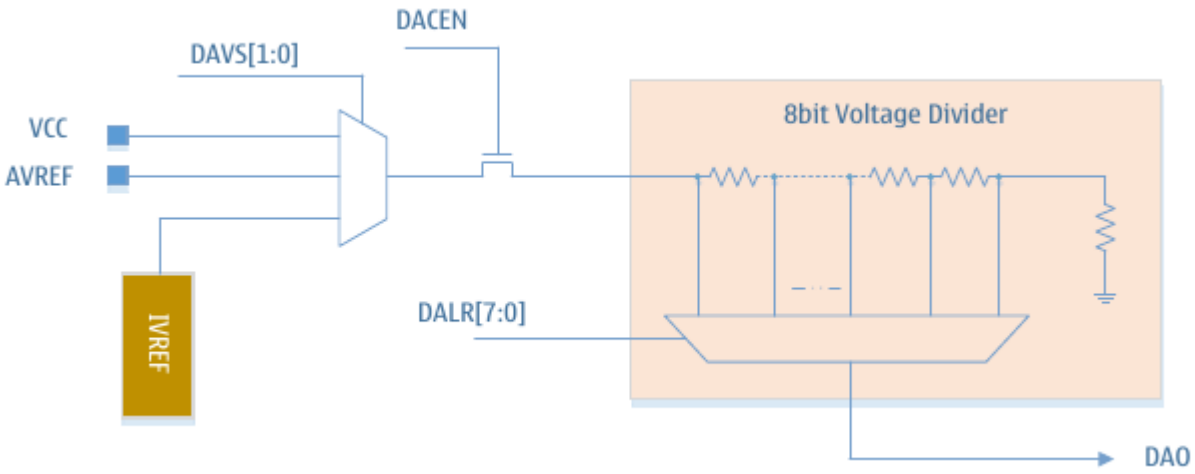
# Digital-to-Analog Converter (DAC)

- 8 bit-to-analog output
- DAC output can be used as analog comparator reference input
- Support DAC output to external port (DAO)
- Optional VCC/AVREF/IVREF divider supply

## Overview

The LGT8FX8P integrates an 8-bit programmable digital-to-analog converter (DAC). The reference power input of the DAC can be selected from the system operating power supply, the internal reference voltage source or the AVREF input from the external port of the chip. The output of the DAC can be selected as the input source of the internal comparator AC0/1, or it can be directly output to the external pin of the chip as an external reference.

When the DAC is output to an external pin, it cannot be used to drive the load directly, it needs to go through a voltage follower or other similar driving circuit. The internal structure of the DAC is shown in the following figure:



## Register definition

### *DACON – DAC Control Register*

| DACON – DAC Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA0 | | | | | Default: 0000_0000 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | DACEN | DAOE | DAVS1 | DAVS0 |
| R/W | - | - | - | - | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7:4 | - | Reserve |
| 3 | DACEN | DAC enable control flag<br>1 = DAC module is enabled<br>0 = DAC module disabled |
| 2 | DAOE | DAC output to external port enable control<br>1 = Enable DAC output to external terminal PD4<br>0 = Disable DAC output to external port |
| 1 | DAVS1 | DAC reference voltage source select flag 1 |
| 0 | DAVS0 | DAC reference voltage source selection flag 0. [DVS1, DVS0] =<br>00 : Voltage source selection system operating voltage VCC<br>01 : The voltage source is selected as external input AVREF<br>10 : The voltage source is selected as the internal reference voltage<br>11 : Turn off the DAC reference source, and also turn off the DAC module |

### *DALR – DAC Data Register*

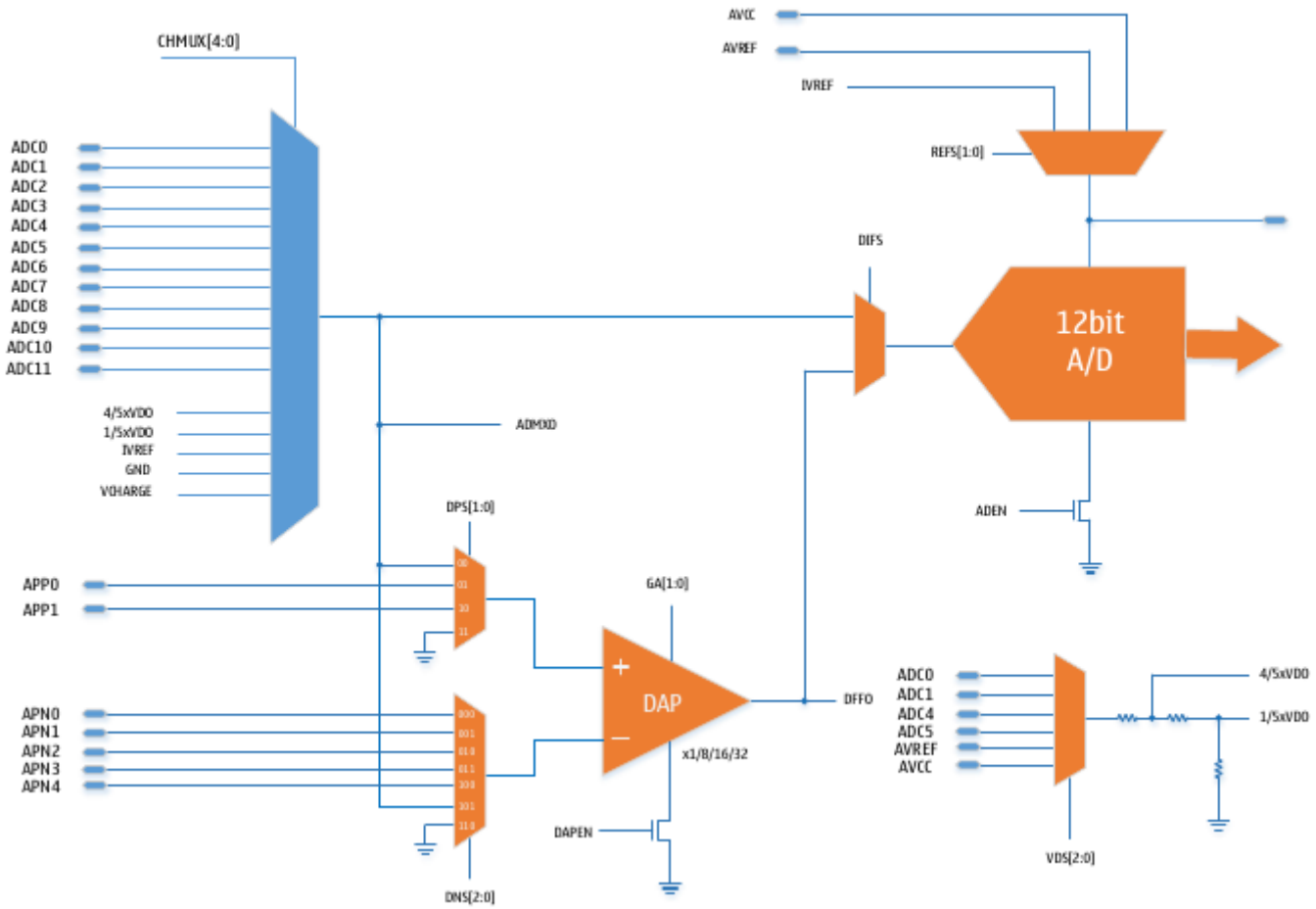
| VRCON1– DAC1 Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA1 | | | | | Default: 0000_0000 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DALR[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 7:0 | DALR | DAC data register, set the output voltage of DAC mode The relationship between DAC output voltage and DALR:<br>$V_{DAO} = V_{REF}*(DALR + 1)/256$<br>in:<br>$V_{DAO}$ output analog voltage for DAC<br>$V_{REF}$ is the DAC reference voltage source, selected by the DAVS flag in the DACON register |

# 12-bit analog-to-digital converter (ADC)

- 12-bit resolution, DNL is ±1LSB, INL is ±1.5LSB
- Sampling rate up to 500KSPS at highest resolution
- 12 multiplexed single-ended input channels
- Multiple Input Programmable Gain Differential Amplifier Channels
- Input voltage range is 0-VCC
- Internal 1.024V/2.048V/4.096V reference voltage
- Supports AVCC and external reference voltage input
- Internal multi-input 1/5, 4/5 voltage divider circuit
- Supports offset calibration in both positive and negative directions
- Automatic start conversion trigger mode based on interrupt source
- Automatic channel monitoring with overflow/underflow support
- Conversion result supports optional alignment mode
- Conversion end interrupt request

## Overview




ADC Block Diagram

The analog-to-digital converter is a 12-bit successive approximation ADC. The ADC is connected with a 17-channel analog multiplexer, which can sample and convert 12 analog inputs from the chip's external port and 5-channel internal voltage source. A differential operational amplifier with programmable gain of x1/x8/x16/x32 is integrated inside the ADC, and the amplifier input can come from an external port or the output of the ADC multiplexer. The result of the differential op amp can be used as the analog input to the ADC.

The internal analog input source of ADC includes multiple input voltage divider from ADC; internal reference voltage source; internal analog reference ground and analog output from touch key module. The internal multi-input voltage divider outputs 4/5 and 1/5 voltages at the same time; the input of the voltage divider can be selected from the level of the external port or from the system power supply.

The ADC supports offset calibration. The process of offset calibration is software controlled. Offset calibration includes calibration quantities in both forward and reverse directions. When offset calibration is enabled, the ADC controller will automatically use the positive and negative calibration values to calibrate the ADC sampling results.

For the method of offset calibration, please refer to the relevant part of this chapter.

## Operation of the ADC

The ADC converts the input analog voltage into a 12-bit digital quantity by successive approximation method. The minimum value represents GND, and the maximum value represents the reference voltage minus 1LSB. The reference voltage source can be the ADC power supply voltage AVCC, the external reference voltage AVREF or the internal 1.024V/2.048V reference voltage, which can be selected by writing the REFS flag of the ADMUX register.

The analog input channel can be selected by writing the CHMUX bits of the ADMUX register. Any ADC input pin, external reference voltage pin, and internal reference voltage source can be used as ADC single-ended input. The ADC input channel can be switched to the internal differential amplifier by setting DIFS in the ADTMR register. The relative input source and gain of the differential amplifier can be set by the DAPCR register.

The ADC can be started by setting the ADEN flag in the ADCSRA register. The ADC does not consume power when ADEN is cleared, so it is recommended to turn off the ADC before entering sleep mode.

The ADC conversion result is 12 bits, which are stored in the ADC data registers ADCH and ADCL. The conversion result is right justified by default, but can be made left justified by setting the ADLAR flag in the ADMUX register.

If the conversion result is set to be left-justified, and only 8-bit conversion precision is required, then just reading ADCH is enough. Otherwise, read ADCL first, and then read ADCH to ensure that the contents of the data register are the result of the same conversion. Once ADCL is read, the data registers ADCL and ADCH are latched, and the conversion result can be updated to the data registers ADCL and ADCH after reading ADCH.

The end of ADC conversion can trigger an interrupt. Even if the end of conversion occurs between reading ADCL and ADCH, the interrupt will still trigger.

## Start a conversion

A single conversion is initiated by writing a '1' to the ADC Start Conversion flag ADSC. This flag remains high during the conversion process until cleared by hardware after the conversion is complete. If the channel is changed during the conversion, the ADC will complete the conversion before changing the channel.

ADC conversions have different trigger sources. Setting the ADC auto-trigger enable flag ADATE in the ADCSRA register enables auto-triggering. The trigger source can be selected by setting the ADC trigger selection bits ADTS in the ADCSRB register. When the selected trigger signal has a rising edge, the ADC prescaler is reset and conversion begins. This provides a way to initiate conversions at fixed time intervals. Even if the trigger signal is still present after the conversion, a new conversion will not be started. If the trigger signal generates a rising edge during the conversion process, this rising edge will also be ignored. Even if the specific interrupt is disabled or the global interrupt enable flag is "0", its interrupt flag will still be set. This triggers a conversion without generating an interrupt. However, in order to trigger a new conversion on the next interrupt event, the interrupt flag must be cleared.

Using the ADC interrupt flag as a trigger source, the next ADC conversion can be started after the current conversion is completed. The ADC then works in continuous conversion mode, continuously sampling and updating the ADC data registers. The first conversion is initiated by writing a '1' to the ADSC flag in the ADCSRA register. In this mode, subsequent ADC conversions do not depend on whether the ADC interrupt flag ADIF is set.

If automatic triggering is enabled, setting the ADSC in the ADCSRA register will initiate a single conversion. The ADSC flag can also be used to detect if a conversion is in progress. Regardless of how the conversion is initiated, ADSC is always "1" during the conversion process.

## Prescaler and ADC Conversion Timing

Under default conditions, the successive approximation circuit requires an input clock from 300KHz to 3MHz for maximum accuracy. If the required conversion accuracy is lower than 12 bits, the input clock frequency can be higher than 3MHz to achieve higher sampling rates.

The ADC module includes a prescaler that generates an acceptable ADC input clock from the system clock. The prescaler is set by the ADPS bits of the ADCSRA register. Setting ADEN in the ADCSRA register enables the ADC and the prescaler starts counting. As long as the ADEN flag is '1', the prescaler continues to count until ADEN is cleared.

After the ADSC flag in the ADCSRA register is set, a single-ended conversion starts on the rising edge of the next ADC clock cycle. A normal conversion requires 15 ADC clock cycles. After the ADC is enabled (ADEN in the ADCSRA register is set), it takes 50 ADC input clock cycles to initialize the analog circuitry before the first conversion can be valid.

During an ADC conversion, the sample-and-hold starts 1.5 ADC input clocks after the conversion starts, and the result output of the first ADC conversion occurs 14.5 ADC input clocks after the start. After the conversion is completed, the ADC result is transferred to the ADC data register and the ADIF flag is set. ADSC is cleared at the same time. The software can then set the ADSC flag again or automatically trigger to start a new conversion.

## Sampling channel and reference voltage

The MUX and REFS in the ADMUX register implement single buffering through scratch registers. The CPU has random access to scratch registers. The selection of channels and reference sources can be configured by the CPU at any time before a conversion is initiated. In order to ensure that the ADC has sufficient sampling time, the configuration of channel and reference selection is not allowed once a conversion has started. The channel and reference selections are not updated until the conversion is complete (ADIF in the ADCSRA register is set). The conversion starts at the rising edge of the next ADC input clock after ADSC is set. Therefore, it is recommended that users do not operate ADMUX to select a new channel and reference source within one ADC input clock cycle after setting ADSC.

When using automatic triggering, the time when the trigger event occurs is indeterminate. To control the effect of new settings on conversions, special care must be taken when updating the ADMUX registers. If both ADATE and ADEN are set, the interrupt time can occur at any time, thereby automatically triggering and starting the ADC conversion. If the contents of the ADMUX register are changed during this time, the user cannot tell whether the next conversion will be based on the old configuration or the new configuration. Users are advised to update ADMUX at the following safe times:

1) ADATE or ADEN flag is "0";

2) During conversion, but at least one ADC input clock cycle after the trigger event;

3) After the conversion is completed, but before the interrupt flag of the trigger source is cleared.

If ADMUX is updated in any of the cases mentioned above, the new configuration will take effect before the next switchover. When selecting the ADC input channel, it should be noted that the channel is selected before the conversion is started, and the new analog input channel can be selected one ADC clock cycle after the ADSC is set, but the easiest way is to wait until the conversion is completed before changing the channel. . The ADC's reference voltage source, Vref, reflects the ADC's conversion range. If the single-ended channel level exceeds Vref, the conversion result will be close to the maximum value of 0xFFF. Vref can be AVCC, the voltage of the external AREF pin, or the internal reference voltage source.

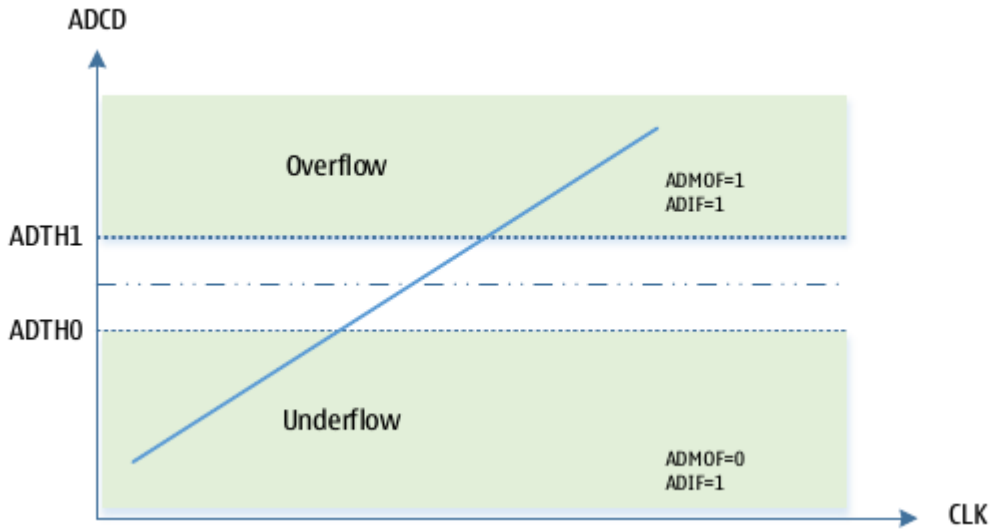
Notes on using the internal reference (1.024V/2.048V/4.096V):

After the chip is powered on, the internal reference is calibrated to 1.024V by default. If the user uses the 1.024V internal reference, it can be used directly without other operations. But if you need to use the internal reference voltage of 2.048V or 4.096V, you need to update the calibration value of the internal reference yourself. The calibration value of 2.048V/4.096V is loaded into the register VCAL2/3 (0xCE/0xCC) after power-on. When the program is initialized, the value of VCAL2/3 is read and written to the VCAL (0XC8) register to complete the calibration .

## Automatic channel monitoring

The automatic channel monitoring mode is used to monitor the voltage change of the selected ADC input channel in real time. The software enables the automatic channel monitoring function by setting the AMEN flag in the ADCCSRC register. The ADC automatically converts the voltage of the selected channel. When the conversion result is outside the given overflow range, the ADC interrupt

flag (ADIF) will be set and the ADC interrupt flag (ADIF) will be set. At the same time stop automatic monitoring. Software can respond to overflow events by interrupting or polling. The AMOF flag of the ADMSC register is used to indicate the type of overflow event. The ADIF flag is automatically cleared by hardware after a reset in response to an interrupt; in polling mode, it can be cleared by software by writing a 1. Auto-monitoring mode can only be re-enabled when ADIF is cleared and the AMEN flag in the ADCSRC register is set.




To overcome the instability of single ADC conversion results, auto-detection supports a configurable digital filtering function. Digital filtering detects continuous conversion results, and only triggers an overflow event when a consistent result is obtained within a limited number of continuous conversions. The number of consecutive conversions can be set by the AMFC[3:0] bits of the ADMSC register.

The automatic channel monitoring function is controlled by the AMEN flag of the ADCSRC register. Register ADT0 is used to set the underflow threshold; ADT1 is used to set the overflow threshold. ADT0/1 are 16-bit registers. After the software sets the AMEN flag, it will immediately stop the current conversion of the ADC, and reset the ADC control state, and then enter the automatic conversion mode.

Before starting the automatic channel detection mode, you need to set the detected channel and other related configurations. Software can disable auto-detect mode at any time by clearing the AMEN register.

## Multiple Input Voltage Divider (VDS)

The ADC contains a multi-input voltage divider module. Voltage divider input Voltage source can be selected from external ADC input channel (ADC0/1/4/5), external reference AVREF or analog operating power supply. The voltage divider module simultaneously outputs 4/5 and 1/5 voltages to the internal 12 and 13 input channels of the ADC. Among them, 4/5 is mostly used for ADC offset calibration; 1/5 is mostly used for power supply voltage detection and other similar applications except for internal offset calibration. The functions related to the voltage divider circuit are mainly controlled by the ADCSRD register.

## ADC offset calibration

Due to the deviation of the manufacturing process and the inherent characteristics of the circuit structure, the internal comparator circuit of the ADC will produce different degrees of offset error. Therefore, compensating for the offset voltage is very critical for producing a high-precision ADC conversion structure. The ADC inside the LGT8FX8P chip supports the interface related to the offset voltage test, and the offset measurement and calibration can be completed with the cooperation of the software.

The principle of offset calibration:

Offset calibration is mainly by changing the input polarity of the internal comparator to test the ADC conversion results in the forward and reverse directions. Since the offset voltage in the forward and reverse directions also has two polarities, an intermediate offset error value can be obtained by subtracting the two conversion results. In normal application, the conversion result can be adjusted accordingly according to this offset voltage.

Offset Calibration Process:

1. Configure the VDS module and select the VDS input source as the analog power supply (AVCC)
2. The reference voltage of ADC is selected as analog power supply (AVCC)
3. ADCSRC[SPN] = 0, ADC reads 4/5VDO channel, the converted value is recorded as PVAL
4. ADCSRC[SPN] = 1, ADC reads 4/5VDO channel, converted value record flag NVAL
5. Store the value (NVAL – PVAL) >>1 into the OFR0 register
6. ADCSRC[SPN] = 1, ADC reads 1/5VDO channel, and the conversion result is recorded as NVAL
7. ADCSRC[SPN] = 0, ADC reads 1/5VDO channel, conversion result record flag PVAL
8. Store the value (NVAL – PVAL) >> 1 into the OFR1 register
9. Set ADCSRC[OFEN]=1 to enable offset compensation function

Note: Since the offset error has positive and negative directions, the above data and operations are all signed operations.

The ADC related configuration needs to be changed during the offset calibration, so it is recommended that the offset calibration be done before the normal configuration. In order to improve the calibration accuracy, it is recommended to sample multiple times of filtering when the ADC reads the channel conversion.

Offset Calibration After OFR0/1 configuration is complete, automatic offset compensation is enabled through the OFEN flag. After the normal conversion in the future, the ADC control will automatically use OFR0/1 for compensation according to the ADC conversion result.

## ADC dynamic calibration

The offset calibration method described above is based on the offset under a test environment and test input. When the system environment changes, the offset of the ADC will also change.

Therefore, if real-time calibration compensation can be realized, it is very important to overcome the performance difference caused by the change of the device with the working environment and improve the ADC measurement accuracy.

A proposed algorithm is provided here. Based on the principle of the offset calibration algorithm, it can dynamically compensate the offset error caused by the working environment and obtain consistent and accurate test results.

This method eliminates the need to calculate the offset voltage and to enable offset compensation (OFEN). The algorithm only needs to control the polarity of ADC conversion through SPN, and sample two measurement results under different SPNs. The errors introduced by the offset in the two results are in both positive and negative directions, so we can simply add and average the results. method to cancel out the error caused by the offset.

We assume that when the ADC is converted, the test error introduced by the offset is VOFS, so the SPN is controlled to perform two consecutive ADC conversions, and the obtained ADC conversion result can be expressed as:

When SPN = 1, VADC1 = VREL + VOFS1

When SPN = 0, VADC0 = VREL – VOFS0

By adding the two measurements, we can remove the effect of VOFS on the actual sampled input VREL. Due to the matching characteristics of the circuit, VOFS1 and VOFS0 may not be exactly the same, but the offset error compensation effect can still be achieved overall.

Dynamic offset compensation algorithm flow:

1. Initialize ADC conversion parameters according to application needs
2. Set SPN=1, start ADC sampling, record ADC sampling result as VADC1
3. Set SPN=0, start ADC sampling, record ADC sampling result as VADC2
4. (VADC1 + VADC2) >> 1 is the conversion result of this ADC

In practical applications, this algorithm can be combined with the sampling average algorithm to obtain more ideal results.

## Register definition

ADC Register List

| Register | Address | Defaults | Description |
|---|---|---|---|
| ADCL | 0x78 | 0x00 | ADC Data Low Byte Register |
| ADCH | 0x79 | 0x00 | ADC Data High Byte Register |
| ADCSRA | 0x7A | 0x00 | ADC Control and Status Register A |
| ADCSRB | 0x7B | 0x00 | ADC Control and Status Register B |
| ADMUX | 0x7C | 0x00 | ADC Multiplexing Control Register |
| ADCSRC | 0x7D | 0x01 | ADC Control and Status Register C |
| DIDR0 | 0x7E | 0x00 | Digital Input Disable Control Register 0 |

| | | | |
|---|---|---|---|
| DIDR1 | 0x7F | 0x00 | Digital Input Disable Control Register 0 |
| DAPCR | 0xDC | 0x00 | Differential Amplifier Control Register |
| OFR0 | 0xA3 | 0x00 | Offset Compensation Register 0 |
| OFR1 | 0xA4 | 0x00 | Offset Compensation Register 1 |
| ADT0L | 0xA5 | 0x00 | Automatically monitor the lower 8 bits of the underflow threshold |
| ADT0H | 0xA6 | 0x00 | Automatic monitoring underflow threshold high 8 bits |
| ADT1L | 0xAA | 0x00 | Automatically monitor the lower 8 bits of the overflow threshold |
| ADT1H | 0xAB | 0x00 | Automatic monitoring overflow threshold high 8 bits |
| ADMSC | 0xAC | 0x01 | Automatic monitoring of status and control registers |
| ADCSRD | 0xAD | 0x00 | ADC Control and Status Register D |

### *ADCL – ADC Data Low Byte Register*

| ADCL – ADC Data Low Byte Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x78 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name0 | ADC7 | ADC6 | ADC5 | ADC4 | ADC3 | ADC2 | ADC1 | ADC0 |
| Name1 | ADC3 | ADC2 | ADC1 | ADC0 | - | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Description |
|---|---|---|
| 7:0 | ADC[7:0]/ ADC[3:0] | ADC data low byte register.<br>When the ADLAR flag is "0", the ADC output data is stored in the register according to the low-order alignment, that is, ADCL is ADC[7:0], as shown in Name0; when the ADLAR flag is "1", the ADC output data is stored in the register. The storage in is high-order aligned, that is, the high-order 4 bits of ADCL are ADC[3:0], and the low-order 4 bits are meaningless, as shown in Name1. |

### *ADCH – ADC Data High Byte Register*

| ADCH – ADC Data High Byte Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x79 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name0 | - | - | - | - | ADC11 | ADC10 | ADC9 | ADC8 |
| Name1 | ADC11 | ADC10 | ADC9 | ADC8 | ADC7 | ADC6 | ADC5 | ADC4 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Description |
|---|---|---|

| | | |
|---|---|---|
| 7:0 | ADC[11:8]/ ADC[11:4] | ADC data low byte register.<br>When the ADLAR flag is "0", the ADC output data is stored in the register according to the low-order alignment, that is, the low 4 bits of ADCH are ADC[11:8], and the high 4 bits are meaningless, as shown in Name0; when the ADLAR flag is When "1", the ADC output data is stored in the register according to the high-order alignment, that is, ADCH is ADC[11:4], as shown in Name1. |

### *ADCSRA – ADC Control and Status Register A*

| ADCSRA – ADC Control and Status Register A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7A | | | | Default: 0x05 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit | Name | Description |
|---|---|---|
| 7 | ADEN | ADC enable control flag.<br>When the ADEN flag is set to "1", the ADC is enabled.<br>When the ADEN flag is set to "0", the ADC is disabled. |
| 6 | ADSC | ADC starts converting.<br>In single conversion mode, setting ADSC will initiate a conversion. In continuous conversion mode, setting ADSC will initiate the first conversion. |
| 5 | ADATE | ADC auto trigger enable control flag.<br>When the ADATE flag is set to "1", the automatic trigger function is enabled. A rising edge of the selected trigger signal starts a conversion. The selection of the trigger source is controlled by ADTS in the ADCSRB register.<br>When the ADATE flag is set to "0", the automatic trigger function is disabled. |
| 4 | ADIF | ADC interrupt flag.<br>ADIF is set when the ADC completes a conversion and updates the data register. If the ADC interrupt enable flag ADIE is "1" and the global interrupt is set, the ADC interrupt will be generated. Executing an ADC interrupt will clear the ADIF flag, which can also be cleared by writing a '1' to this flag. |
| 3 | ADIE | ADC interrupt enable control flag.<br>The ADC interrupt is enabled when the ADIE flag is set to '1' and the global interrupt is set.<br>When the ADIE flag is set to "0", the ADC interrupt is disabled. |
| 2:0 | ADPS[2:0] | ADC prescaler selection control bits.<br>ADPS selects the system clock to generate the prescale factor of the ADC clock. |

| ADPS[2:0] | Prescaler |
|---|---|

| | | | |
|---|---|---|---|
| | | 0 | 2 |
| | | 1 | 2 |
| | | 2 | 4 |
| | | 3 | 8 |
| | | 4 | 16 |
| | | 5 | 32 (default) |
| | | 6 | 64 |
| | | 7 | 128 |

### *ADCSRB – ADC Control and Status Register B*

| ADCSRB – ADC Control and Status Register B | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7B | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ACME01 | ACME00 | ACME11 | ACME10 | ACTS | ADTS2 | ADTS1 | ADTS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Description |
|---|---|---|
| 7 | ACME01 | Comparator 0 negative input selection<br>00: Negative terminal selects external input ACIN0<br>01: Negative terminal selects ADC multiplexed output<br>1X: The negative terminal selects the output of op amp 0 |
| 6 | ACME00 | |
| 5 | ACME11 | Comparator 1 negative input selection<br>00: Negative terminal selects external input ACIN2<br>01: Negative terminal selects ADC multiplexed output<br>1X: The negative terminal selects the output of op amp 1 |
| 4 | ACME10 | |
| 3 | ACTS | AC trigger source channel selection<br>0 – AC0 output as ADC auto-conversion trigger source<br>1 – AC1 output as ADC auto-conversion trigger source |
| 2:0 | ADTS[2:0] | ADC auto trigger source selection control flag.<br>When the ADATE flag is set to "1", the automatic trigger function is enabled, and the selection of the trigger source is controlled by ADTS. When the ADATE flag is set to "0", the setting of ADTS is invalid. The rising edge of the interrupt flag of the selected trigger signal initiates a conversion. When switching from a trigger source with an interrupt flag cleared to a trigger source with an interrupt flag set, the trigger signal will generate a rising edge, and if ADEN is set at this time, the ADC will also start a conversion. When switching to continuous conversion mode (ADTS=0), the automatic trigger function is disabled. |

| ADTS[2:0] | Trigger source |
|---|---|
| 0 | Continuous conversion mode |
| 1 | Comparator 0/1 |

| | | | |
|---|---|---|---|
| | | 2 | External interrupt 0 |
| | | 3 | Timer counter 0 compare match |
| | | 4 | Timer counter 0 overflow |
| | | 5 | Timer Counter 1 Compare Match B |
| | | 6 | Timer counter 1 overflow |
| | | 7 | Timer Counter 1 Input Capture Event |

### *ADMUX – ADC Multiplexing Control Register*

| ADMUX – ADC Multiplexing Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7C | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | REFS1 | REFS0 | ADLAR | CHMUX4 | CHMUX3 | CHMUX2 | CHMUX1 | CHMUX0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Description | |
|---|---|---|---|
| 7:6 | REFS[1:0] | Used in conjunction with REFS2 of the ADCSRD register to select the reference voltage source for the ADC<br>The reference voltage is selected by setting the REFS control flag, if REFS is changed during conversion<br>settings, changes will take effect only after the current conversion is complete. | |
| | | REFS2, REFS[1:0] | Reference voltage selection |
| | | 0_00 | AREF |
| | | 0_01 | AVCC |
| | | 0_10 | On-chip 2.048V reference |
| | | 0_11 | On-chip 1.024V reference |
| | | 1_00 | On-chip 4.096V reference |
| 5 | ADLAR | The conversion result is left-justified enable control flag.<br>When the ADLAR flag is set to '1', the conversion result is left justified in the ADC data register.<br>When the ADLAR flag is set to '0', the conversion result is right-justified in the ADC data register. | |

| | | | | |
|---|---|---|---|---|
| 4:0 | CHMUX[4:0] | ADC input source selection control bits. | | |
| | | CHMUX[4:0] | Single-ended input source | Description |
| | | 0_0000 | PC0 | External port input |
| | | 0_0001 | PC1 | |
| | | 0_0010 | PC2 | |
| | | 0_0011 | PC3 | |
| | | 0_0100 | PC4 | |
| | | 0_0101 | PC5 | |
| | | 0_0110 | PE1 | |
| | | 0_0111 | PE3 | |
| | | 0_1001 | PC7 | |
| | | 0_1010 | PF0 | |
| | | 0_1011 | PE6 | |
| | | 0_1100 | PE7 | |
| | | 0_1110 | 4/5VDO | Internal voltage divider circuit |
| | | 0_1000 | 1/5VDO | |
| | | 0_1101 | IVREF | Internal reference |
| | | 0_1111 | AGND | Analogously |
| | | 1_XXXX | DACO | Internal DAC output |

### *ADCSRC – ADC Control Status Register C*

| ADCSRC – ADC Control Status Register C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7D | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | OFEN | - | SPN | AMEN | - | SPD | DIFS | ADTM |
| R/W | R/W | - | R/W | R/W | - | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | OFEN | 1=Enable offset compensation; 0=Disable offset compensation |
| 6 | - | Unimplemented |
| 5 | SPN | ADC conversion input polarity control, only used for offset calibration process. Must be cleared when normal |
| 4 | AMEN | Channel automatic monitoring enable;<br>1: Enable channel automatic monitoring function<br>0: Disable channel automatic monitoring function |
| 3 | - | Unimplemented |
| 2 | SPD | 0=ADC low speed conversion mode<br>1=ADC high-speed conversion mode, only for low-impedance analog input |

| | | |
|---|---|---|
| 1 | DIFS | 0 = ADC conversions are from the ADC multiplexer<br>1 = ADC conversion is from internal differential amplifier |
| 0 | ADTM | Test mode, output internal reference voltage from AVREF port |

### *DIDR0 – Digital Input Disable Control Register 0*

| DIDR0 – Digital Input Disable Control Register 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7E | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PE3D | PE1D | PC5D | PC4D | PC3D | PC2D | PC1D | PC0D |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | PE3D | 1=Disable PE3 digital input function |
| 6 | PE1D | 1=Disable PE1 digital input function |
| 5 | PC5D | 1=Disable PC5 digital input function |
| 4 | PC4D | 1=Disable PC4 digital input function |
| 3 | PC3D | 1=Disable PC3 digital input function |
| 2 | PC2D | 1=Disable PC2 digital input function |
| 1 | PC1D | 1=Disable PC1 digital input function |
| 0 | PC0D | 1=Disable PC0 digital input function |

### *DIDR1 – Digital Input Disable Control Register 1*

| DIDR1 – Digital Input Disable Control Register 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0x7F | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | PE7D | PE6D | PE0D | C0PD | PF0D | PC7D | PD7D | PD6D |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 0 | PD6D | 1=Disable PD6 digital input function |
| 1 | PD7D | 1=Disable PD7 digital input function |
| 2 | PC7D | 1=Disable PC7 digital input function |
| 3 | PF0D | 1=Disable PF0 digital input function |
| 4 | C0PD | 1=Disable AC0P digital input function (LQFP48) |
| 5 | PE0D | 1=Disable PE0 digital input function |
| 6 | PE6D | 1=Disable PE6 digital input function |
| 7 | PE7D | 1=Disable PE7 digital input function |

### *ADCSRD – ADC Control Register D*

| ADCSRD – ADC Control Register D | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xAD | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | BGEN | REFS2 | IVSEL1 | IVSEL0 | - | VDS2 | VDS1 | VDS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | BGEN | Internal reference global enable control, 1=enable |
| 6 | REFS2 | Used in combination with REFS of ADMUX register to select reference voltage for ADC conversion<br>Please refer to the definition of REFS in the ADMUX register |
| 5:4 | IVSEL | When the reference voltage of ADC is selected as VCC or AVREF, IVSEL is used to control the output voltage of the internal reference:<br>00 = 1.024V<br>01 = 2.048V<br>1x = 4.096V |
| 3 | - | Reserve |
| 2:0 | VDS[2:0] | Voltage divider circuit input source selection<br>000/111 = turn off the voltage divider block<br>001 = ADC0<br>010 = ADC1<br>011 = ADC4<br>100 = ADC5<br>101 = External reference input (AVREF)<br>110 = System Power |

### *DAPCR – Differential Op Amp Control Register*

| DAPCR – Differential Op Amp Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xDC | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | DAPEN | GA1 | GA0 | DNS2 | DNS1 | DNS0 | DPS1 | DPS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | DAPEN | 1 = differential amplifier enabled; 0 = differential amplifier disabled |
| 6:5 | GA[1:0] | Differential Amplifier Gain Control<br>00 = x1<br>01 = x8<br>10 = x16<br>11 = x32 |
| 4:2 | DNS[2:0] | Differential amplifier inverting input input source selection flag<br>000 = ADC2/APN0 |

| | | |
|---|---|---|
| | | 001 = ADC3/APN1<br>010 = ADC8/APN2<br>011 = ADC9/APN3<br>100 = PE0/APN4<br>101 = ADC multiplexing<br>110 = AGND<br>111 = Turn off differential amplifier inverting input |
| 1:0 | DPS[1:0] | Differential amplifier positive input input source selection flag<br>00 = ADC multiplexed<br>01 = ADC0/APP0<br>10 = ADC1/APP1<br>11 = AGND |

### *OFR0 – Offset Compensation Register 0*

| OFR0 – Offset Compensation Register 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA3 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | OFR0[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | OFR0 | Offset Compensation Register 0; OFR0 is a signed number. Store in two's complement format | | | | | | |

### *OFR1 – Offset Compensation Register 1*

| OFR1 – Offset Compensation Register 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA4 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | OFR1[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | OFR1 | Offset Compensation Register 1; OFR1 is a signed number. Store in two's complement format | | | | | | |

### *ADMSC – ADC Channel Monitor Status Control Register*

| ADMSC – ADC Channel Monitor Status Control Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xAC | | | | Default: 0x01 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | AMOF | - | - | - | AMFC3 | AMFC2 | AMFC1 | AMFC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Name | Description |
|---|---|---|
| 7 | AMOF | Automatically monitor overflow event type flag; 1=overflow, 0=underflow |
| 6:4 | - | Unimplemented |
| 3:0 | AMFC | Automatic monitoring of digital filter control bits:<br>0000 = disable configuration<br>0001 = one conversion, no filtering<br>0010 = two consecutive matches<br>0011 = Three consecutive matches<br>.....<br>1110 = 14 consecutive matches<br>1111 = 15 consecutive matches |

### *ADT0L – Automatically monitor the lower 8 bits of the underflow threshold*

| ADT0L – Automatically monitor the lower 8 bits of the underflow threshold | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA5 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ADT0L[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 7:0 | ADT0L | Automatically monitor the lower 8 bits of the underflow threshold register |

### *ADT0H – Automatic monitoring underflow threshold high 8 bits*

| ADT0H – Automatic monitoring underflow threshold high 8 bits | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xA6 | | | | Default: 0x00 | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ADT0H[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 7:0 | ADT0H | Automatic monitoring underflow threshold register upper 8 bits |

### *ADT1L – Automatically monitor the lower 8 bits of the overflow threshold*

| ADT0L – Automatically monitor the lower 8 bits of the overflow threshold | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xAA | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ADT1L[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | ADT1L | Automatically monitor the lower 8 bits of the overflow threshold register | | | | | | |

### *ADT1H – Automatic monitoring overflow threshold upper 8 bits*

| ADT1H – Automatic monitoring overflow threshold upper 8 bits | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xAB | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | ADT1H[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | ADT1H | Automatic monitoring overflow threshold register upper 8 bits | | | | | | |

### *VCAL – Internal Reference Calibration Register*

| VCAL – Internal Reference Calibration Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xC8 | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | VCAL[7:0] | | | | | | | |
| R/W | R/W | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | VCAL | Internal reference calibration register. The calibration value of 1.024V is loaded by default after power-on.<br>The calibration of the relevant reference can be achieved by writing the calibration value of other reference voltages into this register.<br>For example, after the reference configuration is 2.048V, write VCAL2 to the new register to complete the calibration of the 2.048V internal reference. | | | | | | |

### *VCAL1 – 1.024V Reference Calibration Register*

| VCAL1 – 1.024V Internal Reference Calibration Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xCD | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | VCAL1[7:0] | | | | | | | |
| R/W | R/O | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | VCAL1 | 1.024V Internal Reference Calibration Factor | | | | | | |

### *VCAL2 – 2.048V Reference Calibration Register*

| VCAL2 – 2.048V Internal Reference Calibration Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xCE | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | VCAL2[7:0] | | | | | | | |
| R/W | R/O | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | VCAL2 | 2.048V Internal Reference Calibration Factor | | | | | | |

### *VCAL3 – 4.096V Reference Calibration Register*

| VCAL1 – 4.096V Internal Reference Calibration Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: 0xCC | | | | | Default: 0x00 | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | VCAL3[7:0] | | | | | | | |
| R/W | R/O | | | | | | | |
| Bit | Name | Description | | | | | | |
| 7:0 | VCAL3 | 4.096V Internal Reference Calibration Coefficient | | | | | | |

# Register Cheat Sheet

| Addr | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | Extended IO Register | | | | |
| $F6 | GUID3 | GUID Byte 3 | | | | | | | |
| $F5 | GUID2 | GUID Byte 2 | | | | | | | |
| $F4 | GUID1 | GUID Byte 1 | | | | | | | |
| $F3 | GUID0 | GUID Byte 0 | | | | | | | |
| $F2 | PMCR | PMCE | CLKFS | CLKSS | WCLKS | OSCKEN | OSCMEN | RCKEN | RCMEN |
| $F0 | PMX2 | WCE | STOSC1 | STOSC0 | - | - | XIEN | E6EN | C6EN |
| $EE | PMX0 | PMXCE | C1BF4 | C1AF5 | C0BF3 | C0AC0 | SSB1 | TXD6 | RXD5 |
| $ED | PMX1 | - | - | - | - | - | C3AC | C2BF7 | C2AF6 |
| $EC | TCKSR | - | F2XEN | TC2XF1 | TC2XF0 | - | AFCKS | TC2XS1 | TC2XS0 |
| $E2 | PSSR | PSS1 | PSS3 | - | - | - | - | PSR3 | PSR1 |
| $E1 | OCPUE | PUE7 | PUE6 | PUE5 | PUE4 | PUE3 | PUE2 | PUE1 | PUE0 |
| $E0 | HDR | - | - | HDR5 | HDR4 | HDR3 | HDR2 | HDR1 | HDR0 |
| $DE | DAPTE | DAPTE | - | - | - | - | - | - | - |
| $DD | DAPTR | DAPTP | | | | DAP Trimming | | | |
| $DC | DAPCR | DAPEN | GA1 | GA0 | DNS2 | DNS1 | DNS0 | DPS1 | DPS0 |
| $D8 | | | | | | | | | |
| $D7 | | | | | | | | | |
| $D6 | | | | | | | | | |
| $D5 | | | | | | | | | |
| $D4 | | | | | | | | | |
| $D2 | | | | | | | | | |
| $D1 | | | | | | | | | |
| $D0 | | | | | | | | | |
| $CF | LDOCR | WCE | | | | PDEN | VSEL2 | VSEL1 | VSEL0 |
| $CE | VCAL2 | Calibration value for 2.048V internal reference | | | | | | | |
| $CD | VCAL1 | Calibration value for 1.024V internal reference | | | | | | | |
| $CC | VCAL3 | Calibration value for 4.096V internal reference | | | | | | | |
| $C8 | VCAL | Internal Voltage Reference calibration register | | | | | | | |
| $C6 | UDR | USART Data Register | | | | | | | |
| $C5 | UBRRH | - | - | - | - | USART Baud Rate Register High | | | |
| $C4 | UBRRL | USART Baud Rate Register Low | | | | | | | |
| $C2 | UCSRC | UMSEL1 | UMSEL0 | UPM1 | UPM0 | USBS0 | UCSZ01 | UCSZ00 | UCPOL0 |
| $C1 | UCSRB | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | UCSZ02 | RXB80 | TXB80 |
| $C0 | UCSRA | RXC0 | TXC0 | UDRE0 | FE0 | DOR0 | UPE0 | U2X0 | MPCM0 |
| $BD | TWAMR | | | | TWI Address Mask | | | | - |
| $BC | TWCR | TWINT | TWEA | TWSTA | TWSTO | TWWC | TWEN | - | TWIE |
| $BB | TWDR | TWI Data | | | | | | | |
| $BA | TWAR | | | | TWI Address | | | | TWGCE |

| Addr | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| $B9 | TWSR | TWI Status bits | | | | | - | TWPS | |
| $B8 | TWBR | TWI Bit Rate register | | | | | | | |
| $B6 | ASSR | INTCK | - | AS2 | TCN2UB | OCR2AUB | OCR2BUB | TCR2AUB | TCR2BUB |
| $B4 | OCR2B | Timer 2 Output Compare Register B | | | | | | | |
| $B3 | OCR2A | Timer 2 Output Compare Register A | | | | | | | |
| $B2 | TCNT2 | Timer 2 Counter Register | | | | | | | |
| $B1 | TCCR2B | FOC2A | FOC2B | - | - | WGM22 | | CS2 | |
| $B0 | TCCR2A | COM2A1 | COM2A0 | COM2B1 | COM2B0 | - | - | WGM21 | WGM20 |
| $AF | DPS2R | - | - | - | - | DPS2E | LPRCE | TOS1 | TOS0 |
| $AE | IOCWK | IOCD7 | IOCD6 | IOCD5 | IOCD4 | IOCD3 | IOCD2 | IOCD1 | IOCD0 |
| $AD | ADCSRD | BGEN | REFS2 | IVSEL1 | IVSEL0 | - | VDS2 | VDS1 | VDS0 |
| $AC | ADMSC | AMOF | - | - | - | AMFC3 | AMFC2 | AMFC1 | AMFC0 |
| $AB | ADT1H | ADC Auto-monitor Overflow threshold high byte | | | | | | | |
| $AA | ADT1L | ADC Auto-monitor Overflow threshold low byte | | | | | | | |
| $A9 | PORTE | Port Output E (for compatible with LGT8FX8D) | | | | | | | |
| $A8 | DDRE | Data Direction E (for compatible with LGT8FX8D) | | | | | | | |
| $A7 | PINE | Port Input E (for compatible with LGT8FX8D) | | | | | | | |
| $A6 | ADT0H | ADC Auto-monitor Underflow threshold high byte | | | | | | | |
| $A5 | ADT0L | ADC Auto-monitor Underflow threshold low byte | | | | | | | |
| $A4 | OFR1 | ADC positive offset trimming | | | | | | | |
| $A3 | OFR0 | ADC negative offset trimming | | | | | | | |
| $A1 | DALR | DAC data register | | | | | | | |
| $A0 | DACON | - | - | - | - | DACEN | DAOE | DAVS1 | DAVS0 |
| $9F | OCR3CH | Compare output register high byte of Timer3 C channel | | | | | | | |
| $9E | OCR3CL | Compare output register low byte of Timer3 C channel | | | | | | | |
| $9D | DTR3H | Dead-band register high byte of Timer3 | | | | | | | |
| $9C | DTR3L | Dead-band register low byte of Timer3 | | | | | | | |
| $9B | OCR3BH | Compare output register high byte of Timer3 B channel | | | | | | | |
| $9A | OCR3BL | Compare output register low byte of Timer3 B channel | | | | | | | |
| $99 | OCR3AH | Compare output register high byte of Timer3 A channel | | | | | | | |
| $98 | OCR3AL | Compare output register low byte of Timer3 A channel | | | | | | | |
| $97 | ICR3H | Input capture register high byte of Timer3 | | | | | | | |
| $96 | ICR3L | Input capture register low byte of Timer3 | | | | | | | |
| $95 | TCNT3H | Counter register high byte of Timer3 | | | | | | | |
| $94 | TCNT3L | Counter register low byte of Timer3 | | | | | | | |
| $93 | TCCR3D | Control register D of Timer3 | | | | | | | |
| $92 | TCCR3C | Control register C of Timer3 | | | | | | | |
| $91 | TCCR3B | Control register B of Timer3 | | | | | | | |
| $90 | TCCR3A | Control register A of Timer3 | | | | | | | |
| $8D | DTR1H | Dead-band register high byte of Timer1 | | | | | | | |
| $8C | DTR1L | Dead-band register low byte of Timer1 | | | | | | | |
| $8B | OCR1BH | Timer 1 Output Compare B High | | | | | | | |

| Addr | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| $8A | OCR1BL | Timer 1 Output Compare B Low | | | | | | | |
| $89 | OCR1AH | Timer 1 Output Compare A High | | | | | | | |
| $88 | OCR1AL | Timer 1 Output Compare A Low | | | | | | | |
| $87 | ICR1H | Timer 1 Input Capture High | | | | | | | |
| $86 | ICR1L | Timer 1 Input Capture Low | | | | | | | |
| $85 | TCNT1H | Timer 1 Counter High | | | | | | | |
| $84 | TCNT1L | Timer 1 Counter Low | | | | | | | |
| $83 | TCCR1D | DSX17 | DSX16 | DSX15 | DAX14 | - | - | DSX11 | DSX10 |
| $82 | TCCR1C | FOC1A | FOC1B | DOC1B | DOC1A | DTEN1 | - | - | - |
| $81 | TCCR1B | ICNC1 | ICES1 | - | WGM13 | WGM12 | | CS1 | |
| $80 | TCCR1A | COM1A1 | COM1A0 | COM1B1 | COM1B0 | - | - | WGM11 | WGM10 |
| $7F | DIDR1 | PE7D | PE6D | PE0D | C0PD | PF0D | PC7D | PD7D | PD6D |
| $7E | DIDR0 | PE3D | PE1D | PC5D | PC4D | PC3D | PC2D | PC1D | PC0D |
| $7D | ADCSRC | OFEN | - | SPN | AMEN | - | SPD | DIFS | ADTM |
| $7C | ADMUX | REFS1 | REFS0 | ADLAR | | | CHMUX | | |
| $7B | ADCSRB | CME01 | CME00 | CME11 | CME10 | - | | ADTS | |
| $7A | ADCSRA | ADEN | ADSC | ADATE | ADIF | ADIE | | ADPS | |
| $79 | ADCH | ADC Data High | | | | | | | |
| $78 | ADCL | ADC Data Low | | | | | | | |
| $76 | DIDR2 | - | PB5D | - | - | - | - | - | - |
| $75 | IVBASE | Interrupt Vector Base Address | | | | | | | |
| $74 | PCMSK4 | | | | | | | | |
| $73 | PCMSK3 | PCINT[39:32] | | | | | | | |
| $71 | TIMSK3 | | | ICIE3 | - | OCIE3C | OCIE3B | OCIE3A | TOIE3 |
| $70 | TIMSK2 | - | - | - | - | - | OCIE2B | OCIE2A | TOIE2 |
| $6F | TIMSK1 | - | - | ICIE1 | - | - | OCIE1B | OCIE1A | TOIE1 |
| $6E | TIMSK0 | - | - | - | - | - | OCIE0B | OCIE0A | TOIE0 |
| $6D | PCMSK2 | PCINT[23:16] | | | | | | | |
| $6C | PCMSK1 | PCINT[15:8] | | | | | | | |
| $6B | PCMSK0 | PCINT[7:0] | | | | | | | |
| $69 | EICRA | - | - | - | - | ISC11 | ISC10 | ISC01 | ISC00 |
| $68 | PCICR | - | - | - | PCIE4 | PCIE3 | PCIE2 | PCIE1 | PCIE0 |
| $67 | RCKCAL | RC32K Calibration | | | | | | | |
| $66 | RCMCAL | RC32M Calibration | | | | | | | |
| $65 | PRR1 | - | - | PRWDT | - | PRTIM3 | PREFL | PRPCI | - |
| $64 | PRR/0 | PRTWI | PRTIM2 | PRTIM0 | - | PRTIM1 | PRSPI | PRUART0 | PRADC |
| $62 | VDTCR | WCE | SWR | - | | VDTS | | VDREN | VDTEN |
| $61 | CLKPR | WCE | CKOE1 | CKOE0 | - | | CLKPS | | |
| $60 | WDTCSR | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 |
| | | | | | Direct IO Register | | | | |
| $5F | SREG | I | T | H | S | V | N | Z | C |
| $5E | SPH | Stack Point High | | | | | | | |

| Addr | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| $5D | SPL | Stack Point Low | | | | | | | |
| $5C | E2PD3 | E2PCTL Data register byte 3 | | | | | | | |
| $5B | C1TR | AC1 trimming data | | | | | | | |
| $5A | E2PD1 | E2PCTL Data register byte1 | | | | | | | |
| $59 | DSAH | DSA[31:16] access port of uDSC | | | | | | | |
| $58 | DSAL | DSA[15:0] access port of uDSC | | | | | | | |
| $57 | E2PD2 | E2PCTL Data register byte 2 | | | | | | | |
| $56 | ECCR | WEN | EEN | ERN | SWM | CP1 | CP0 | ECS1 | ECS0 |
| $55 | MCUCR | FWKEN | FPDEN | SWR | PUD | IRLD | IFAIL | IVSEL | WCE |
| $54 | MCUSR | SWDD | - | - | OCDRF | WDRF | BORF | EXTRF | PORF |
| $53 | SMCR | - | - | - | - | | SM | | SE |
| $52 | C0TR | AC0 Trimming register | | | | | | | |
| $51 | C0XR | - | C0OE | C0HYSE | C0PS0 | C0WKE | C0FEN | C0FS1 | C0FS0 |
| $50 | C0SR | C0D | C0BG | C0O | C0I | C0IE | C0IC | C0IS | |
| $4F | DTR0 | TC0 Dead-band timing control register | | | | | | | |
| $4E | SPDR | SPI Data register | | | | | | | |
| $4D | SPSR | SPIF | WCOL | - | - | - | DUAL | - | SPI2X |
| $4C | SPCR | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR | |
| $4B | GPIOR2 | General Purpose Register 2 | | | | | | | |
| $4A | GPIOR1 | General Purpose Register 1 | | | | | | | |
| $49 | TCCR0C | DSX07 | DSX06 | DSX05 | DSX04 | - | - | DSX01 | DSX00 |
| $48 | OCR0B | Timer 0 Output Compare Register B | | | | | | | |
| $47 | OCR0A | Timer 0 Output Compare Register A | | | | | | | |
| $46 | TCNT0 | Timer 0 Counter | | | | | | | |
| $45 | TCCR0B | FOC0A | FOC0B | OC0AS | DTEN0 | WGM02 | CS02 | CS01 | CS00 |
| $44 | TCCR0A | COM0A1 | COM0A0 | COM0B1 | COM0B0 | DOC0B | DOC0A | WGM01 | WGM00 |
| $43 | GTCCR | TSM | - | - | - | - | - | PSRASY | PSRSYNC |
| $42 | EEARH | E2PCTL Address High | | | | | | | |
| $41 | EEARL | E2PCTL Address Low | | | | | | | |
| $40 | E2PD0 | E2PCTL Data byte 0 | | | | | | | |
| $3F | EECR | EEPM2 | EEPM2 | EEPM1 | EEPM0 | EERIE | EEMWE | EEWE | EERE |
| $3E | GPIOR0 | General Purpose Register 0 | | | | | | | |
| $3D | EIMSK | - | - | - | - | - | - | INT1 | INT0 |
| $3C | EIFR | - | - | - | - | - | - | INTF1 | INTF0 |
| $3B | PCIFR | - | - | - | - | PCIF3 | PCIF2 | PCIF1 | PCIF0 |
| $3A | C1XR | - | C1OE | C1HYSE | C1PS0 | C1WKE | C1FEN | C1FS1 | C1FS0 |
| $39 | SPFR | RDFULL | RDEMPT | RDPTR1 | RDPTR0 | WRFULL | WREMPT | WRPTR1 | WRPTR0 |
| $38 | TIFR3 | - | - | ICF3 | - | - | OCF3B | OCF3A | TOV3 |
| $37 | TIFR2 | - | - | - | - | - | OCF2B | OCF2A | TOV2 |
| $36 | TIFR1 | - | - | ICF1 | - | - | OCF1B | OCF1A | TOV1 |
| $35 | TIFR0 | - | - | - | - | - | OCF0B | OCF0A | TOV0 |
| $34 | PORTF | Port Output of Group F | | | | | | | |

| Addr | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| $33 | DDRF | Data Direction of Group F | | | | | | | |
| $32 | PINF | Port Input of Group F | | | | | | | |
| $31 | DSDY | DSDY access port of uDSC | | | | | | | |
| $30 | DSDX | DSDX access port of uDSC | | | | | | | |
| $2F | C1SR | C1D | C1BG | C1O | C1I | C1IE | C1IC | C1IS | |
| $2E | PORTE | Port Output of Group E | | | | | | | |
| $2D | DDRE | Data Direction of Group E | | | | | | | |
| $2C | PINE | Port Input of Group E | | | | | | | |
| $2B | PORTD | Port Output of Group D | | | | | | | |
| $2A | DDRD | Data Direction of Group D | | | | | | | |
| $29 | PIND | Port Input of Group D | | | | | | | |
| $28 | PORTC | Port Output of Group C | | | | | | | |
| $27 | DDRC | Data Direction of Group C | | | | | | | |
| $26 | PINC | Port Input of Group C | | | | | | | |
| $25 | PORTB | Port Output of Group B | | | | | | | |
| $24 | DDRB | Data Direction of Group B | | | | | | | |
| $23 | PINB | Port Input of Group B | | | | | | | |
| $22 | DSSD | DSSD access port of uDSC | | | | | | | |
| $21 | DSIR | Instruction regiter of uDSC | | | | | | | |
| $20 | DSCR | DSUEN | MM | D1 | D0 | - | DSN | DSZ | DSC |

# Instruction Set Cheat Sheet

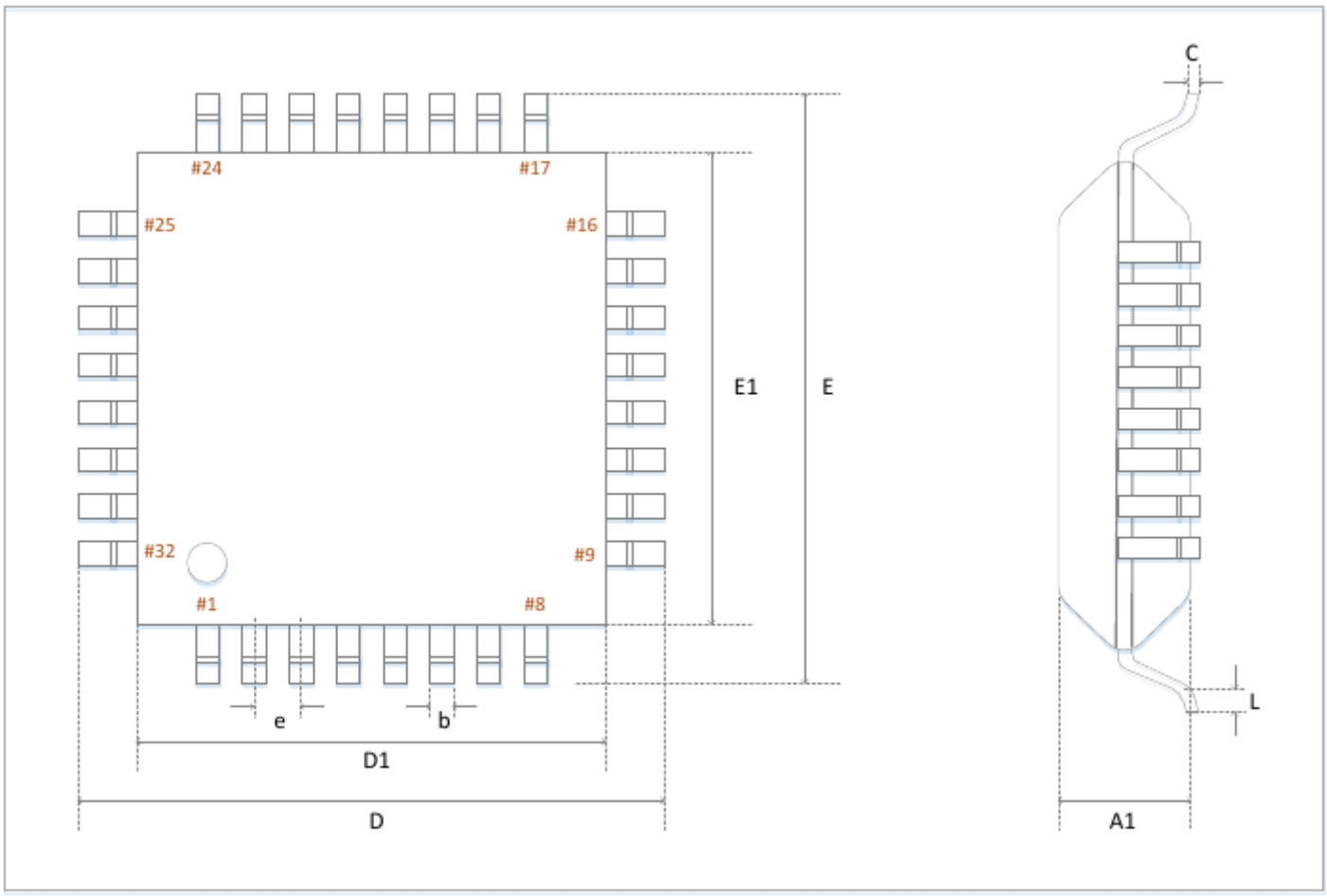
| Instruction | Operand | Description | Operation | Flag | Cycle |
|---|---|---|---|---|---|
| Arith. logic operations | arithmetic instruction | | | | |
| ADD | Rd, R r | Register addition | Rd ← Rd + Rr | Z,C,N,V,H | 1 |
| ADC | Rd, R r | Register Addition with Carry | Rd ← Rd + Rr + C | Z,C,N,V,H | 1 |
| ADIW | Rdl, K | Add literal and word | Rdh:Rdl ← Rdh:Rdl + K | Z,C,N,V,S | 1 |
| SUB | Rd, R r | Register addition and subtraction | Rd ← Rd - Rr | Z,C,N,V,H | 1 |
| SUBI | Rd, K | Register decrement constant | Rd ← Rd - K | Z,C,N,V,H | 1 |
| SBC | Rd, R r | Register addition and subtraction with borrow | Rd ← Rd - Rr - C | Z,C,N,V,H | 1 |
| SBCI | Rd, K | Register decrement constant with borrow | Rd ← Rd – K - C | Z,C,N,V,H | 1 |
| SBIW | Rdl, K | Subtract an immediate number from a word | Rdh:Rdl ← Rdh:Rdl - K | Z,C,N,V,S | 1 |
| AND | Rd, R r | Logical AND | Rd ← Rd & Rr | Z,N,V | 1 |
| ANDI | Rd, K | Register logic and constants | Rd ← Rd & K | Z,N,V | 1 |
| OR | Rd, R r | Logical OR | Rd ← Rd \| Rr | Z,N,V | 1 |
| ORI | Rd, K | Register logic OR constant | Rd ← Rd \| K | Z,N,V | 1 |
| EOR | Rd, R r | Register XOR | Rd ← Rd⊕Rr | Z,N,V | 1 |
| COM | Rd | One's complement | Rd ← $FF - Rd | Z,C,N,V | 1 |
| NEG | Rd | 2 forbids complement (2 禁制补码) | Rd ← $00 - Rd | Z,C,N,V,H | 1 |
| SBR | Rd, K | Set bit in register | Rd ← Rd v K | Z,N,V | 1 |
| CBR | Rd, K | Clear register bits | Rd ← Rd v ($FF – K) | Z,N,V | 1 |
| INC | Rd | Increment | Rd ← Rd + 1 | Z,N,V | 1 |
| DEC | Rd | Decrement | Rd ← Rd - 1 | Z,N,V | 1 |
| TST | Rd | Test for 0 or negative | Rd ← Rd & Rd | Z,N,V | 1 |
| CLR | Rd | Clear register | Rd ← Rd⊕Rd | Z,N,V | 1 |
| SER | Rd | All registers are set to 1 | Rd ← $FF | None | 1 |
| MUL | Rd, R r | Unsigned multiplication | R1: R0 ← Rd x Rr | Z,C | 1 |
| MULS | Rd, R r | Signed multiplication | R1: R0 ← Rd x Rr | Z,C | 1 |
| MULSU | Rd, R r | Multiply signed number by unsigned number | R1: R0 ← Rd x Rr | Z,C | 1 |
| FMUL | Rd, R r | Unsigned multiply, shift | R1: R0 ← (Rd x Rr) << 1 | Z,C | 1 |
| FMULS | Rd, R r | Signed multiplication, shift | R1: R0 ← (Rd x Rr) << 1 | Z,C | 1 |
| FMULSU | Rd, R r | Multiply signed number by unsigned number, shift | R1: R0 ← (Rd x Rr) << 1 | Z,C | 1 |
| Jump instructions | | | | | |
| RJMP | K | Relative jump | PC ← PC + K + 1 | None | 1 |
| IJMP | | Indirect jump (to address pointed to by Z) | PC ← Z | None | 2 |
| JMP | K | Jump directly | PC ← K | None | 2 |
| RCALL | K | Relative address subroutine call | PC ← PC + K + 1 | None | 1 |
| ICALL | | Indirect subroutine call (Z points to address) | PC ← Z | None | 2 |
| CALL | K | Direct subroutine call | PC ← K | None | 2 |
| RET | | Subroutine return | PC ← Stack | None | 2 |
| RETI | | Return from interrupt | PC ← Stack | I | 2 |
| CPSE | Rd, Rr | Equal to jump (相等即跳转 ) | If( Rd=Rr) PC ← PC + 2 or 3 | None | 1/2 |
| CP | Rd, Rr | Compare | Rd - Rr | Z,N,V,C,H | 1 |
| CPC | Rd, Rr | Compare with carry | Rd - Rr – C | Z,N,V,C,H | 1 |
| CPI | Rd, K | Compare with immediate value | Rd - K | Z,N,V,C,H | 1 |
| SBRC | Rr, b | A bit of 0 skips the next instruction | If(Rr(b)=0) PC ← PC + 2 or 3 | None | 1/2 |
| SBRS | Rr, b | A bit of 1 skips the next instruction | If(Rr(b)=1) PC ← PC + 2 or 3 | None | 1/2 |
| SBIC | P, b | I/O bit 0 skips next instruction | If(P(b)=0) PC ← PC + 2 or 3 | None | 1/2 |
| SBIS | P, b | I/O bit 1 skips next instruction | If(P(b)=1) PC ← PC + 2 or 3 | None | 1/2 |
| BRBS | s, k | Jump when the status is marked as 1 | If(SREG(S)=1) PC ← PC + K + 1 | None | 1/2 |
| BRBC | s, k | Jump when the status is marked as 0 | If(SREG(S)=0) PC ← PC + K + 1 | None | 1/2 |
| BREQ | k | Branch On Equal (Equal to jump) | if (Z = 1) then PC ← PC + k + 1 | None | 1/2 |

| Instruction | Operand | Description | Operation | Flag | Cycle |
|---|---|---|---|---|---|
| Jump instructions | (Cont.) | | | | |
| BRNE | k | Jump without waiting/jump before waiting | if (Z = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRCS | k | Jump on carry | if (C = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRCC | k | Jump without carry | if (C = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRSH | k | Jump if not less than | if (C = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRLO | k | Jump if less than | if (C = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRMI | k | Jump if negative | if (N = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRPL | k | For regular jump | if (N = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRGE | k | Signed no less than jump | if (N⊕V= 0) then PC ← PC + k + 1 | None | 1/2 |
| BRLT | k | Signed less than 0 is a jump | if (N⊕V= 1) then PC ← PC + k + 1 | None | 1/2 |
| BRHS | k | Jump if the half-carry is 1 | if (H = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRHC | k | Jump if the half-carry is 0 | if (H = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRTS | k | Jump if T is set | if (T = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRTC | k | Jump when T is cleared | if (T = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRVS | k | Jump on overflow | f (V = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRVC | k | Jump if not overflow | f (V = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRIE | k | Jump if global interrupt is enabled | f ( I = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRID | k | Jump if global interrupt is disabled | f ( I = 0) then PC ← PC + k + 1 | None | 1/2 |
| Data transfer | arithmetic instruction | | | | |
| MOV | Rd, Rr | Move data between registers | Rd ← Rr | None | 1 |
| MOVW | Rd, Rr | Move a word of data | Rd+1:Rd ← Rr+1:Rr | None | 1 |
| LDI | Rd, K | Load immediate | Rd ← K | None | 1 |
| LD | Rd, X | Indirect loading | Rd ← (X) | None | 1/2 |
| LD | Rd, X+ | Indirect load, address incrementing | Rd ← (X), X ← X + 1 | None | 1/2 |
| LD | Rd, -X | Address decrement, indirect load | X ← X - 1, Rd ← (X) | None | 1/2 |
| LD | Rd, Y | Indirect loading | Rd ← (Y) | None | 1/2 |
| LD | Rd, Y+ | Indirect load, address incrementing | Rd ← (Y), Y ← Y + 1 | None | 1/2 |
| LD | Rd, -Y | Address decrement, indirect load | Y ← Y - 1, Rd ← (Y) | None | 1/2 |
| LDD | Rd, Y+q | Indirect loading with offset | Rd ← (Y + q) | None | 1/2 |
| LD | Rd, Z | Indirect loading | Rd ← (Z) | None | 1/2 |
| LD | Rd, Z+ | Indirect load, address incrementing | Rd ← (Z), Z ← Z+1 | None | 1/2 |
| LD | Rd, -Z | Address decrement, indirect load | Z ← Z - 1, Rd ← (Z) | None | 1/2 |
| LDD | Rd, Z+q | Indirect loading with offset | Rd ← (Z + q) | None | 1/2 |
| LDS | Rd, k | Load directly from SRAM | Rd ← (k) | None | 2 |
| ST | X, Rr | Indirect storage | (X) ← Rr | None | 1 |
| ST | X+, Rr | Indirect storage, address incrementing | (X) ← Rr, X ← X + 1 | None | 1 |
| ST | -X, Rr | Address decrement, indirect storage | X ← X - 1, (X) ← Rr | None | 1 |
| ST | Y, Rr | Indirect storage | (Y) ← Rr | None | 1 |
| ST | Y+, Rr | Indirect storage, address incrementing | (Y) ← Rr, Y ← Y + 1 | None | 1 |
| ST | -Y, Rr | Address decrement, indirect storage | Y ← Y - 1, (Y) ← Rr | None | 1 |
| STD | Y+q, Rr | Indirect storage with offset | (Y + q) ← Rr | None | 1 |
| ST | Z, Rr | Indirect storage | (Z) ← Rr | None | 1 |
| ST | Z+, Rr | Indirect storage, address incrementing | (Z) ← Rr, Z ← Z + 1 | None | 1 |
| ST | -Z, Rr | Address decrement, indirect storage | Z ← Z - 1, (Z) ← Rr | None | 1 |
| STD | Z+q, Rr | Indirect storage with offset | (Z + q) ← Rr | None | 1 |
| STS | k, Rr | Store directly into SRAM | (k) ← Rr | None | 2 |
| LPM | | Loader space data | R0 ← (Z) | None | 2 |
| LPM | Rd, Z | Loader space data | Rd ← (Z) | None | 2 |
| LPM | Rd, Z+ | Load program data, address incremented | Rd ← (Z), Z ← Z+1 | None | 2 |
| LD | Rd, Z+ | Indirect load, address incrementing | Rd ← (Z), Z ← Z+1 | None | 1 |
| LD | Rd, -Z | Address decrement, indirect load | Z ← Z - 1, Rd ← (Z) | None | 1 |
| LDD | Rd, Z+q | Indirect loading with offset | Rd ← (Z + q) | None | 1 |

| Instruction | Operand | Description | Operation | Flag | Cycle |
|---|---|---|---|---|---|
| Data transfer | (Cont.) | | | | |
| LDS | Rd, k | Load directly from SRAM | Rd ← (k) | None | 2 |
| | | | | | |
| IN | Rd, P | Read port | Rd ← P | None | 1 |
| OUT | P, Rr | Write port | P ← Rr | None | 1 |
| PUSH | Rr | Push (onto) the stack | STACK ← Rr | None | 1 |
| POP | Rd | Pop out of stack | Rd ← STACK | None | 1/2 |
| | | | | | |
| SBI | P, b | Set IO register | I/O(P, b) ← 1 | None | 1 |
| CBI | P, b | Clear IO register | I/O(P, b) ← 0 | None | 1 |
| LSL | Rd | logical left shift | Rd(n+1) ← Rd(n), Rd(0) ← 0 | Z,C,N,V | 1 |
| LSR | Rd | logical shift right | Rd(n) ← Rd(n+1), Rd(7) ← 0 | Z | 1 |
| ROL | Rd | Rotate left with carry | Rd(0)←C, Rd(n+1) ← Rd(n), C←Rd(7) | Z | 1 |
| ROR | Rd | Rotate right with carry | Rd(7)←C, Rd(n) ← Rd(n+1), C←Rd(0) | Z | 1 |
| ASR | Rd | Arithmetic shift right | Rd(n) ← Rd(n+1), n=0:6 | Z | 1 |
| SWAP | Rd | Bit swap | Rd(3:0) ← Rd(7:4), Rd(7:4) ← Rd(3:0) | None | 1 |
| BSET | s | Set status bit | SREG(s) ← 1 | SREG(s) | 1 |
| BCLR | s | Clear status bit | SREG(s) ← 0 | SREG(s) | 1 |
| BST | Rr, b | Store to T bit | T ← Rr(b) | T | 1 |
| BLD | Rd, b | Read T bit into register | Rd(b) ← T | None | 1 |
| SEC | | Set the carry flag | C ← 1 | C | 1 |
| CLC | | Clear carry flag | C ← 0 | C | 1 |
| SEN | | Set negative flag | N ← 1 | N | 1 |
| CLN | | Clear negative flag | N ← 0 | N | 1 |
| SEZ | | Set zero flag | Z ← 1 | Z | 1 |
| CLZ | | Clear zero flag | Z ← 0 | Z | 1 |
| SEI | | Enable global interrupt | I ← 1 | I | 1 |
| CLI | | Disable global interrupts | I ← 0 | I | 1 |
| SES | | Set Symbol Test Flags | S ← 1 | S | 1 |
| CLS | | Clear Symbol Test Flag | S ← 0 | S | 1 |
| SEV | | Set the two's complement overflow flag | V ← 1 | V | 1 |
| CLV | | Clear the two's complement overflow flag | V ← 0 | V | 1 |
| SET | | Set T bit (SREG) | T ← 1 | T | 1 |
| CLT | | Clear T bit (SREG) | T ← 0 | T | 1 |
| MCU control | arithmetic instruction | | | | |
| NOP | | Empty instruction | | None | 1 |
| SLEEP | | Enter sleep mode | | None | 1 |
| WDR | | Watchdog reset | | None | 1 |
| BREAK | | Soft breakpoint | For debugging purposes only | None | N/A |
| NOP | | Empty instruction | | None | 1 |
| SLEEP | | Enter sleep mode | | None | 1 |

# Package’s pfysical parameters

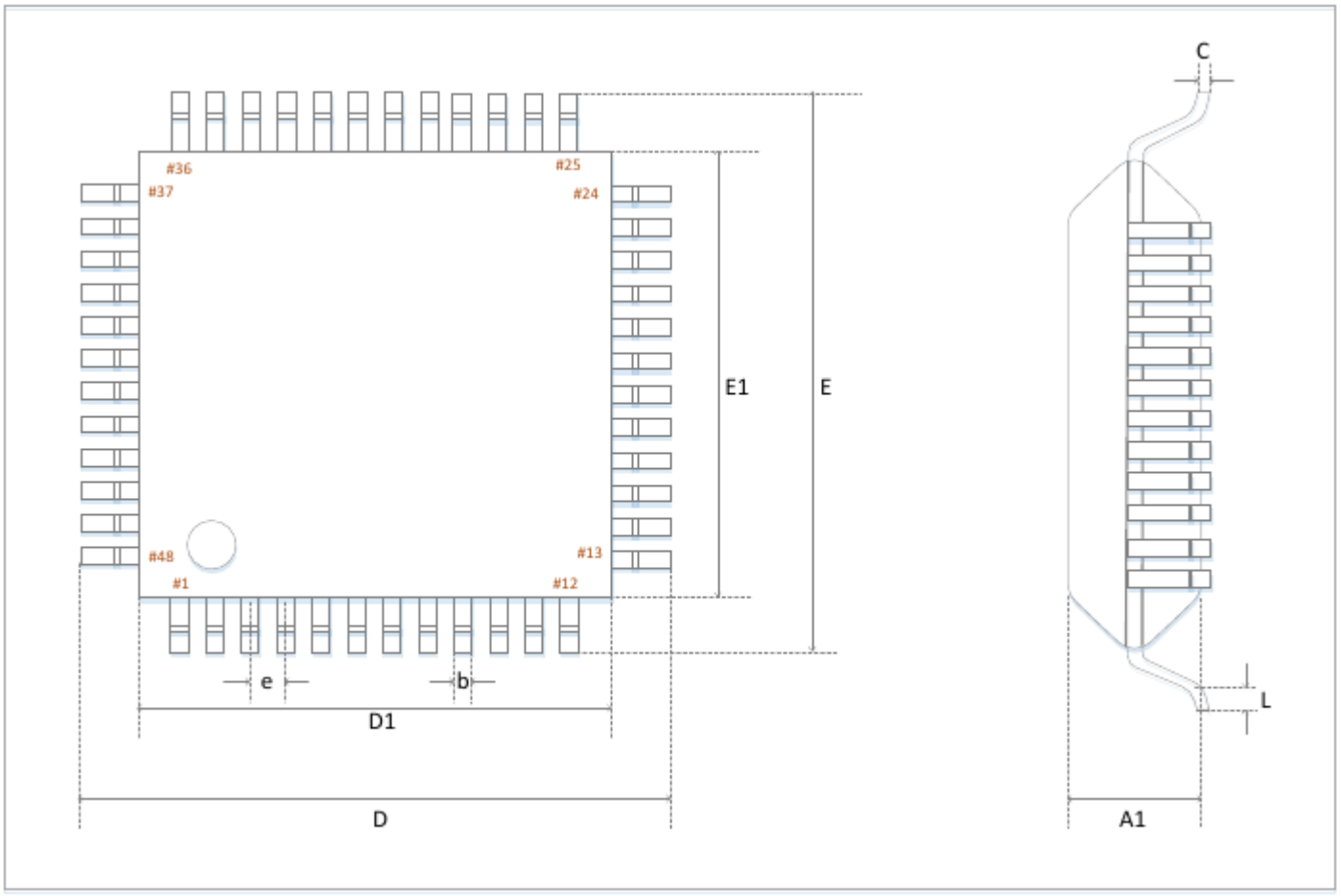
## LQFP32




## LQFP32 general dimension definition

| Designation | Minimum value | Typical value | Maximum value | Unit |
|---|---|---|---|---|
| **D** | 8.90 | 9.00 | 9.10 | mm |
| **D1** | 6.90 | 7.00 | 7.10 | mm |
| **b** | 0.2 | 0.30 | 0.4 | mm |
| **e** | 0.75 | 0.80 | 0.85 | mm |
| **E** | 8.90 | 9.00 | 9.10 | mm |
| **E1** | 6.90 | 7.00 | 7.10 | mm |
| **C** | - | 0.10 | - | mm |
| **L** | 0.55 | 0.60 | 0.65 | mm |
| **A1** | - | 1.40 | - | mm |

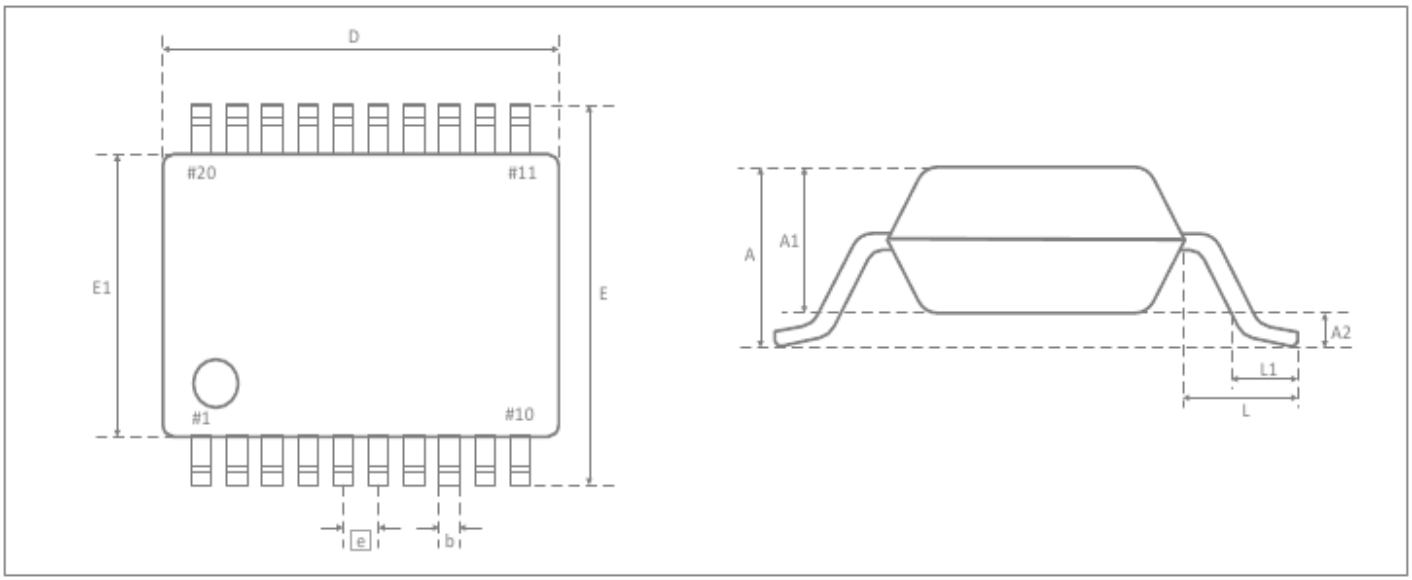
## LQFP48




## LQFP48 General Dimension Definition

| Designation | Minimum value | Typical value | Maximum value | Unit |
|---|---|---|---|---|
| **D** | 8.80 | 9.00 | 9.20 | mm |
| **D1** | 6.80 | 7.00 | 7.20 | mm |
| **b** | 0.17 | 0.22 | 0.27 | mm |
| **e** | - | 0.50BSC | - | mm |
| **E** | 8.80 | 9.00 | 9.20 | mm |
| **E1** | 6.80 | 7.00 | 7.20 | mm |
| **C** | 0.09 | - | 0.2 | mm |
| **L** | 0.45 | 0.60 | 0.75 | mm |
| **A1** | 1.35 | 1.40 | 1.45 | mm |

## SSOP20L




## SSOP20L General Dimension Definition

| Designation | Minimum value | Typical value | Maximum value | Unit |
|---|---|---|---|---|
| **D** | 6.90 | 7.20 | 7.50 | mm |
| **A2** | 0.03 | 0.05 | 0.07 | mm |
| **b** | 0.22 | 0.30 | 0.38 | mm |
| **e** | - | 0.65 | - | mm |
| **E** | 7.40 | 7.80 | 8.20 | mm |
| **E1** | 5.00 | 5.30 | 5.60 | mm |
| **L1** | 0.55 | - | 0.95 | mm |
| **L** | - | - | - | mm |
| **A1** | - | 2.0 | - | mm |

# About this document

Any copyright for this manual belongs to LogicGreen Technologies Co., LTD. I certainly does not claim any IPR for this manual, this translation is done in good faith and in order to be able to use this rather interesting microcontroller. I can't read Chinese, so the entire translation is done using machine-translators as the primary tools.

This translation is done as close to the original Chinese document as I am, with my tools and limitations, able to get it. If the translation diverges semantically from the original Chinese document, it's non-intentional and probably caused by lacking linguistical skills on by behalf. If anyone spots any misconceptions or faulty translation, please inform me so I can correct my mistake. Any help will be credited unless the contributor explicitly refuses to be mentioned.

- Gjermund Skogstad -

# Version history

| | |
|---|---|
| V1.0.5-GEN1<br>2022/2/02 | Initial semi-automatic translation from Chinese to English.<br>Conceptual and linguistic quality control pending.<br>The document ought to be usable, although maybe not an easy read. |
| | |
| V1.0.5<br>2018/9/26 | Remove ADC11 function on QFP32/PB5 pin<br>Corrected configuration for positive side selection in AC1 |
| V1.0.4<br>2017/11/15 | Correct the definition of SSOP20 PIN8/11 |
| V1.0.3<br>2017/6/23 | Added SSOP20 package definition<br>Instructions for updating the TMR3 interrupt flag bits |
| V1.0.2<br>2017/5/15 | Updated description of automatic PWM shutdown and restart in TMR0/TRM1/TMR3<br>Updated the description of SPI interrupt handling in the SPI chapter and the description of updating the SPFR register |
| V1.0.1<br>2017/2/13 | Remove I2C1 part, this function is not available<br>Improve the definition of some registers |
| V1.0.0<br>2016/12/29 | Initial version |