FEATURES

- **High performance, low power 8bit RISC core**
  131 Instructions, 80% execute in one cycle
  32x8 general purpose registers
  Up to 32MIPS when running at 32MHz
  Integrated one-cycle 8x8 Multiplier
- **Data and Programming Memory**
  8Kbytes In-system-programmable FLASH memory
  1Kbytes Internal SRAM
  504Bytes Data FLASH, Support Byte-wise access (E2PROM like)
  Creative flash encryption based on state changing.
- **Peripherals**
  Two 8bit Timer/Counter, support compare-match output
  One 16bit Timer/Counter with separated clock prescalar, Support Input Capture and compare-match output
  Internal 32 KHz RC oscillator, support calibrated to ±1%
  Up to 6-channel PWM
  8-channel 10bit Analog/Digital Converter
    - 3-channel Difference input, x7.5, x15, x30 gain control
    - Integrated thermal sensor
  2-channel Analog Comparator, channel can be extended from ADC
  Programmable Watch dog timer
  Programmable serial USART
  Master/slave SPI serial Interface
  Byte-oriented 2-wire serial interface (Philips I2C compatible)
- **Special Microcontroller features**
  Serial Wire on-chip Debug (SWD)
  External and internal interrupt sources
  Power on reset and 3-level Brown-out Reset (Low voltage reset)
  Internal 32 MHz RC oscillator, ±1% after calibration
  Internal 32 KHz RC oscillator, ±1% after calibration
  External crystal support 32.768 KHz or 400K~32MHz
  Up to 12-channel capacitive touch keys
  8-channel NMOS I/O, sink up to 80mA current.
- **I/O and Package**
  QFP32L (provide up to 30 GPIO)
  S/SOP28L (provide up to 26 GPIO)
- **Operating Environment**

Power supply:	1.8V ~ 5.5V
Frequency:	0 ~ 32MHz
Temperature:	-40C ~ +85C
HBM ESD:	4000V

LogicGreen

8-bit LGT8XM

RISC Microcontroller with 8192 Bytes In-System Programmable FLASH Memory

LGT8F88A

Data book

Version 1.1.1

Application

Kitchen

- Microwave oven
- Induction cooker
- Electric cooker

Smart home appliance

- Milk machine
- Coffee maker
- Water heater

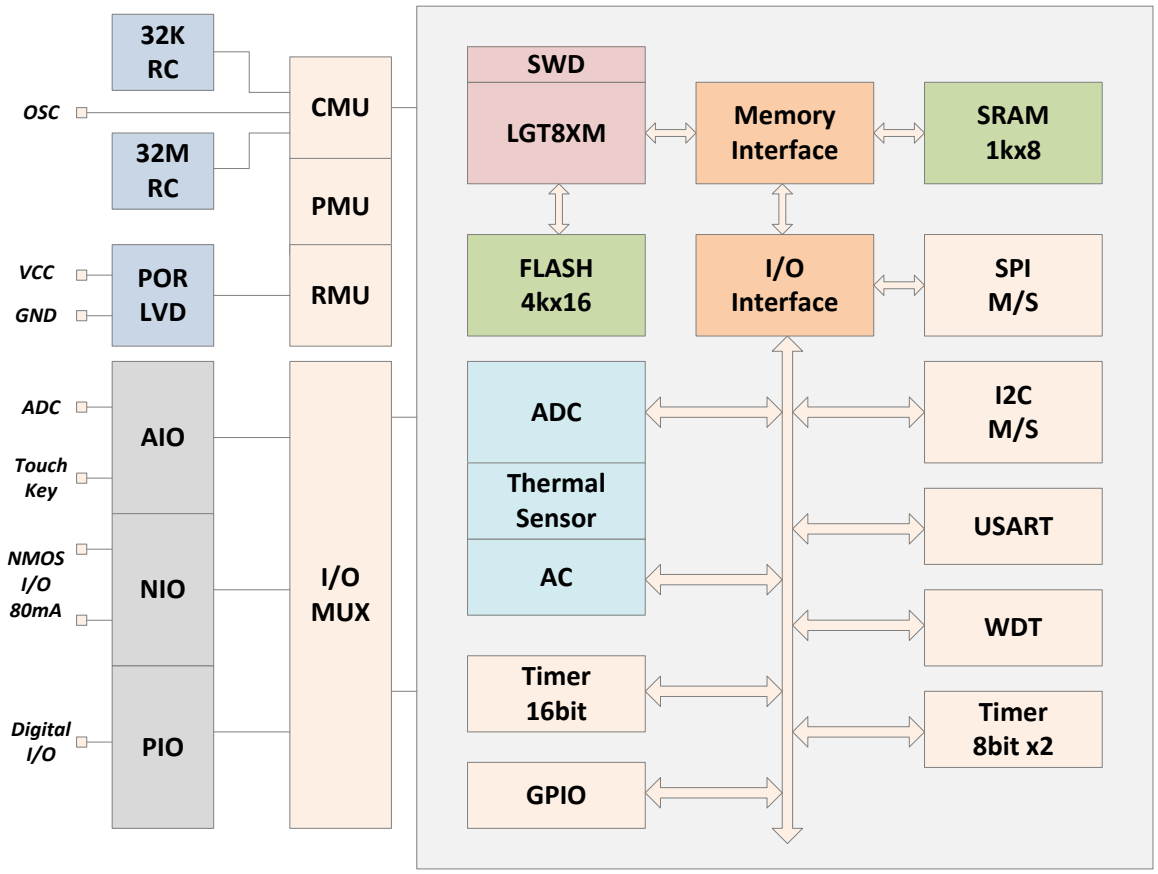
Smart control devices

- Li-on charger
- Motor control
- Smart toys

Hand-held device

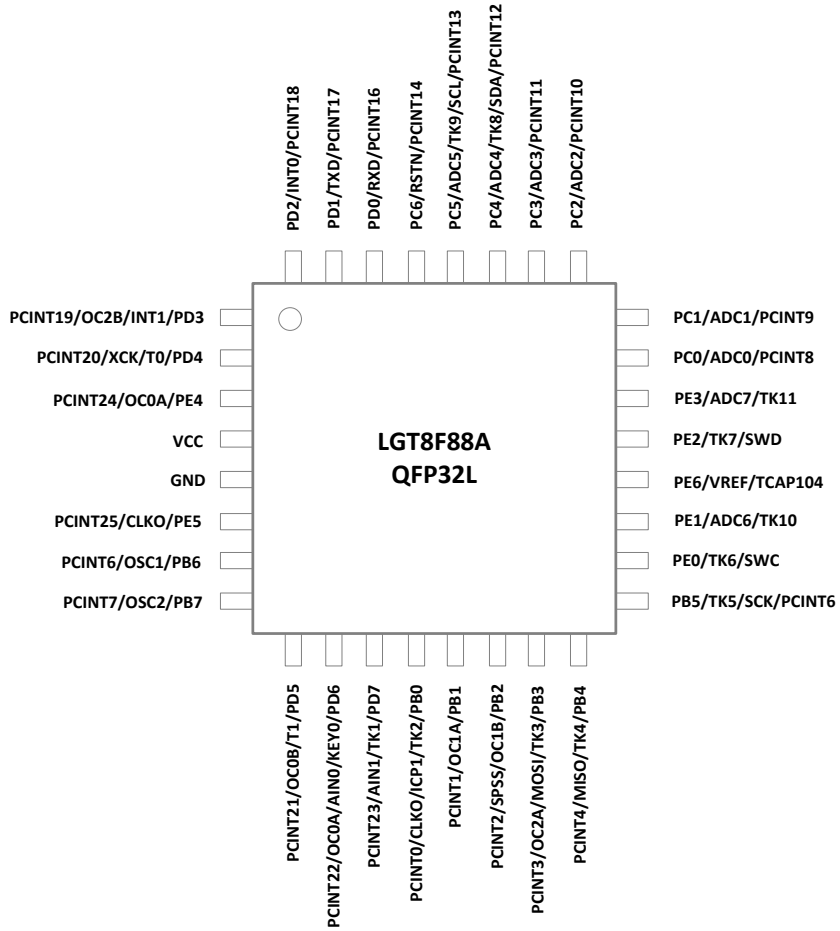
System Architecture

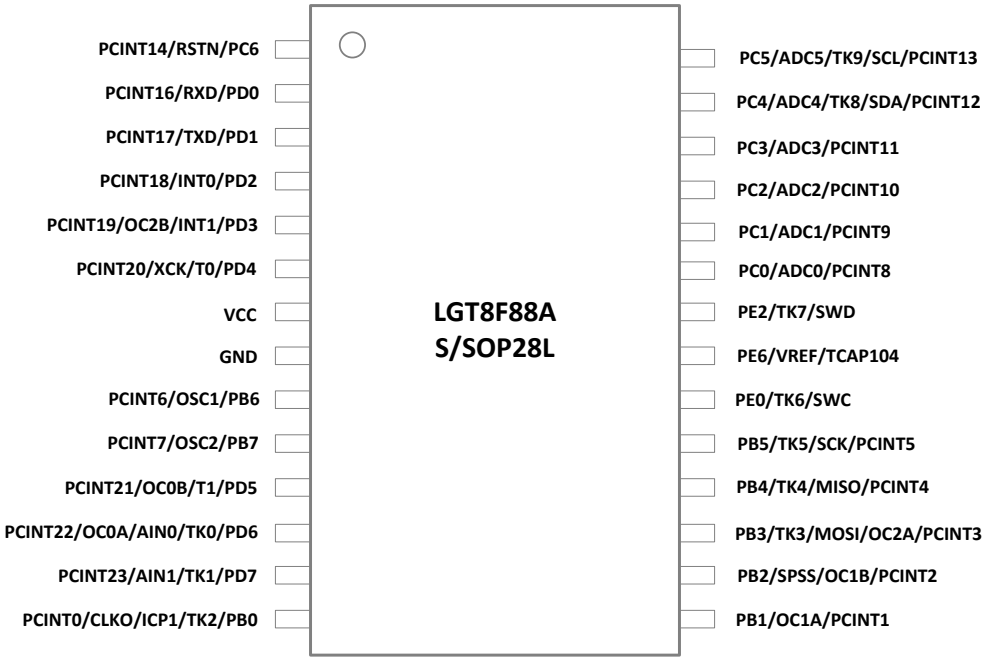
LGT8F88A Diagram



Module Name	Function Description
SWD	On-chip debugger
LGT8XM	8bit High performance RISC core
CMU	Clock management Unit
PMU	Power Management Unit
RMU	Reset Management Unit
POR/LVD	Power on Reset and Low voltage detector
ADC Thermal Sensor AC	8-channel 10bit ADC Thermal Sensor Analog Comparator
AIO NIO PIO	ADC and Touch Key inputs 80mA high sink NMOS I/O Programmable Digital I/O
WDT	Watch Dog Timer

Pin-out Assignment





Pin-out Definition

PIN Name	Function Description
VCC	Power supply (1.8V ~ 5.5V)
GND	System Ground
OSC1 **OSC2**	External Crystal or clock input
RSTN	External Reset input, low active
RXD **TXD** **XCK**	USART interface
INT0/1	External Interrupts or external wake-up sources
OC0A/B	Timer/Counter 0 compare-match output (PWM0A/B)
OC1A/B	Timer/Counter 1 compare-match output (PWM1A/B)
OC2A/B	Timer/Counter 2 compare-match output (PWM2A/B)
SCL **SDA**	Byte-oriented Two wire interface (I2C compatible)
SCK **SPSS** **MISO** **MOSI**	Master/Slave SPI interface
T0	External clock input of Timer0
T1	External clock input of Timer1
ICP1	Capture input of Timer1
SWD **SWC**	SWD on-chip debugger or ISP interface
PCINTX	Pin status change interrupts
ADC7…0	Analog input channels of ADC
TK11…0	Capacitive touch key inputs
VREF/TCAP104	External VREF of ADC External filter-capacitance (0.1uF) of Touch Key circuit
AIN0 **AIN1**	Input channel of Analog Comparator
CLKO	System clock output
PB7…0	Programmable General Purpose I/O
PD7…0	Programmable General Purpose I/O
PC6…0	Programmable General Purpose I/O
PE6…0	Programmable General Purpose I/O
PD5…0 **PE5…4**	NMOS I/O, Can be sink up to 80mA

REGISTERS INDEX

Address	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	LFEN	EXTEN	WCES	OSCKEN	OSCMEN	RCKEN	RCMEN
$F1	DSCR	DSCE	-	-	DSC4	DSC3	DSC2	DSC1	DSC0
$F0	IOCR	IOCE	-	-	-	-	-	REFIOEN	RSTIOEN
$E2	PSSR	PSS1	-	-	-	-	-	-	PSR1
$CF	DIDR3	-	-	-	-	TIN11D	TIN10D	TIN9D	TIN8D
$CE	DIDR2	TIN7D	TIN6D	TIN5D	TIN4D	TIN3D	TIN2D	TIN1D	TIN0D
$CD	TKCSR	TKPD	TKPSEL			TKMUX			
$C6	UDR0	USART Data							
$C5	UBRR0H	-	-	-	-	USART Baud Rate Register High			
$C4	UBRR0L	USART Baud Rate Register Low							
$C2	UCSR0C	UMSEL0		UPM0		USBS0	UCSZ01/ UDORD0	UCSZ00/ UCPHA0	UCPOL0
$C1	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
$C0	UCSR0A	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
$B9	TWSR	TWI Status					-	TWPS	
$B8	TWBR	TWI Bit Rate							
$B6	ASSR	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
$B4	OCR2B	Timer/Counter 2 Output Compare Register B							
$B3	OCR2A	Timer/Counter 2 Output Compare Register A							
$B2	TCNT2	Timer/Counter 2 Counter Register							
$B1	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS2		
$B0	TCCR2A	COM2A		COM2B		-	-	WGM21	WGM20

$A9	PORTE	Port Output E							
$A8	DDRE	Data Direction E							
$A7	PINE	Port Input E							
$8B	OCR1BH	Timer/Counter 1 Output Compare B High							
$8A	OCR1BL	Timer/Counter 1 Output Compare B Low							
$89	OCR1AH	Timer/Counter 1 Output Compare A High							
$88	OCR1AL	Timer/Counter 1 Output Compare A Low							
$87	ICR1H	Timer/Counter 1 Input Capture High							
$86	ICR1L	Timer/Counter 1 Input Capture Low							
$85	TCNT1H	Timer/Counter 1 Counter High							
$84	TCNT1L	Timer/Counter 1 Counter Low							
$82	TCCR1C	FOC1A	FOC1B	-	-	-	-	-	-
$81	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS1		
$80	TCCR1A	COM1A		COM1B		-	-	WGM11	WGM10
$7F	DIDR1	-	-	-	-	-	-	AIN1D	AIN0D
$7E	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
$7D	ADTMR	GAIN		-	-	-	ADTM		
$7C	ADMUX	REFS		ADLAR	-	MUX			
$7B	ADCSRB	-	ACME	-	ICTL	-	ADTS		
$7A	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS		
$79	ADCH	ADC Data High							
$78	ADCL	ADC Data Low							
$77	EEDRH	EEPROM Data High							
$75	IVBASE	Interrupt Vector Base Address							
$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	-	-	-	-	ISC1		ISC0	
$68	PCICR	-	-	-	-	-	PCIE2	PCIE1	PCIE0
$66	OSCCAL	-	-	OSC Calibration					
$65	PRR1	-	-	PRWDT	-	-	PREFL	PRPCI	-
$64	PRR	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	PRUSART0	PRADC

$62	VDTCR	VDTCE	SWRSTN	-	-	-	VDTSEL		VDTEN
$61	CLKPR	CLKPCE	CLKOEN 0	CLKOEN 1	-	CLKPS			
$60	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
$5F($3F)	SREG	I	T	H	S	V	N	Z	C
$5E($3E)	SPH	Stack point high byte							
$5D($3D)	SPL	Stack point low byte							
$55($35)	MCUCR	-	BODS	BODSE	PUD	-	-	IVSEL	IVCE
$54($34)	MCUSR	SWDD	-	-	OCDRF	WDRF	BORF	EXTRF	PORF
$53($33)	SMCR	-	-	-	-	SM			SE
$50($30)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS	
$4E(0x2E)	SPDR	SPI Data Register							
$4D($2D)	SPSR	SPIF	WCOL	-	-	-	DUAL	-	SPI2X
$4C($2C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR	
$4B($2B)	GPIOR2	General purpose I/O register 2							
$4A($2A)	GPIOR1	General purpose I/O register 1							
$48($28)	OCR0B	Timer/counter 0 output compare register B							
$47($27)	OCR0A	Timer/counter 0 output compare register A							
$46($26)	TCNT0	Timer/Counter 0 counter							
$45($25)	TCCR0B	FOC0A	FOC0B	OC0AS	-	WGM02	CS0		
$44($24)	TCCR0A	COM0A		COM0B		-	-	WGM01	WGM00
$43($23)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC
$42($22)	EEARH	EEPROM Address high byte							
$41($21)	EEARL	EEPROM Address low byte							
$40($20)	EEDR	EEPROM Data							
$3F($1F)	EECR	EEPM2	-	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE
$3E($1E)	GPIOR0	General purpose IO register 0							
$3D($1D)	EIMSK	-	-	-	-	-	-	INT1	INT0
$3C($1C)	EIFR	-	-	-	-	-	-	INTF1	INTF0
$3B($1B)	PCIFR	-	-	-	-	-	PCIF2	PCIF1	PCIF0
$37($17)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2
$36($16)	TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1
$35($15)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0
$2B($0B)	PORTD	Port output D							
$2A($0A)	DDRD	Data direction D							
$29($09)	PIND	Port input D							
$28($08)	PORTC	Port output C							

$27($07)	DDRC	Port direction C
$26($06)	PINC	Port input C
$25($05)	PORTB	Port output B
$24($04)	DDRB	Port direction B
$23($03)	PINB	Port input B

INSTRUCTION INDEX

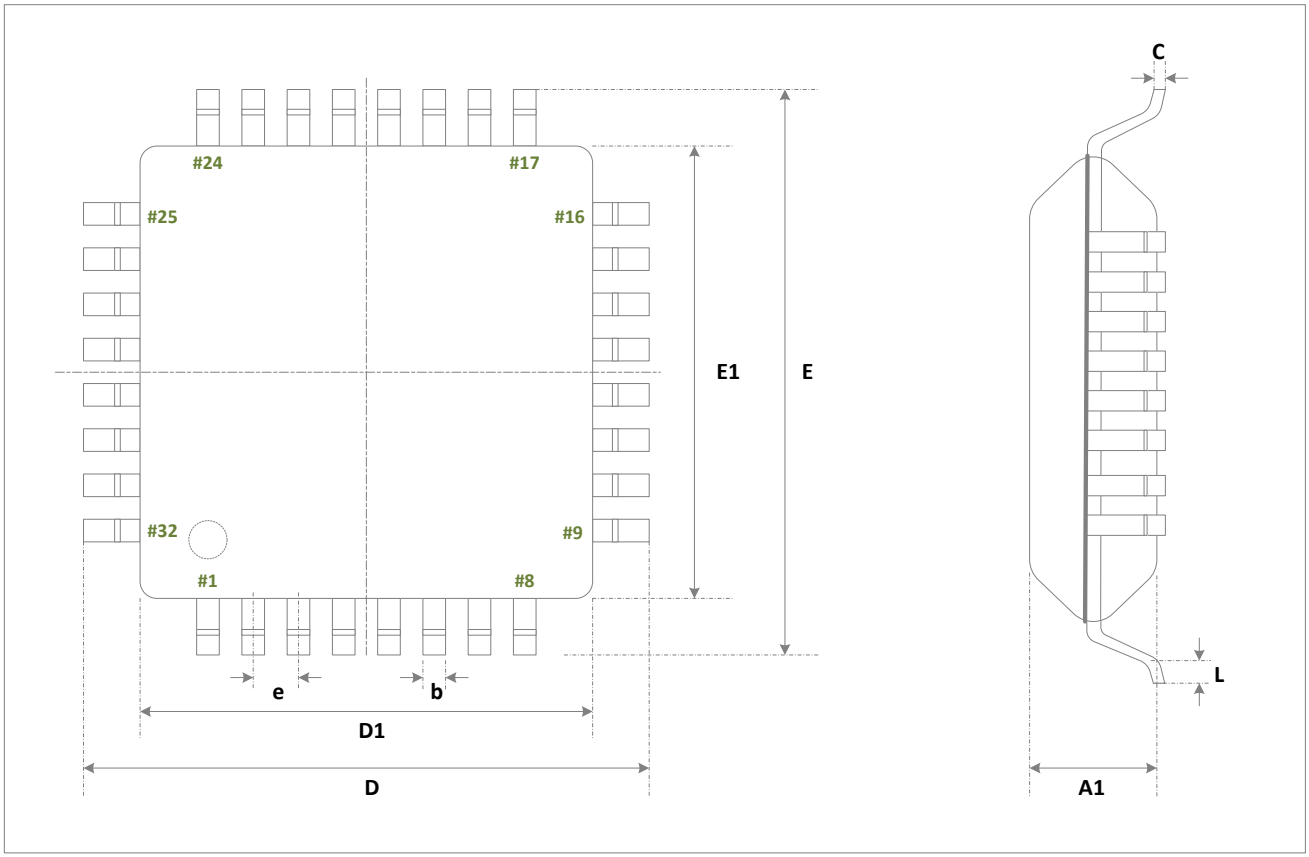
INST.	OPC.	FUNCITONS	OPERATION	FLAG	CYCLE
Arithmetic and Logic operation					
ADD	R_d, R_r	Add two registers	$R_d \leftarrow R_d + R_r$	Z,C,N,V,H	1
ADC	R_d, R_r	Add with carry two regiters	$R_d \leftarrow R_d + R_r + C$	Z,C,N,V,H	1
ADIW	R_{dl}, K	Add immediate to word	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} + K$	Z,C,N,V,S	1
SUB	R_d, R_r	Subtract two registers	$R_d \leftarrow R_d - R_r$	Z,C,N,V,H	1
SUBI	R_d, K	Subtract constant from registers	$R_d \leftarrow R_d - K$	Z,C,N,V,H	1
SBC	R_d, R_r	Subtract with carry	$R_d \leftarrow R_d - R_r - C$	Z,C,N,V,H	1
SBCI	R_d, K	Subtract with carry constant	$R_d \leftarrow R_d – K - C$	Z,C,N,V,H	1
SBIW	R_{dl}, K	Subtract immediate from word	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} - K$	Z,C,N,V,S	1
AND	R_d, R_r	Logical AND	$R_d \leftarrow R_d \& R_r$	Z,N,V	1
ANDI	R_d, K	Logical AND register and constant	$R_d \leftarrow R_d \& K$	Z,N,V	1
OR	R_d, R_r	Logical OR	$R_d \leftarrow R_d \mid R_r$	Z,N,V	1
ORI	R_d, K	Logical OR register and constant	$R_d \leftarrow R_d \mid K$	Z,N,V	1
EOR	R_d, R_r	Exclusive OR	$R_d \leftarrow R_d \oplus R_r$	Z,N,V	1
COM	R_d	One’s complement	$R_d \leftarrow \$FF - R_d$	Z,C,N,V	1
NEG	R_d	Two’s complement	$R_d \leftarrow \$00 - R_d$	Z,C,N,V,H	1
SBR	R_d, K	Set bit(s) in Register	$R_d \leftarrow R_d$ v K	Z,N,V	1
CBR	R_d, K	Clear bit(s) in Rigister	$R_d \leftarrow R_d$ v ($FF – K)	Z,N,V	1
INC	R_d	Increment	$R_d \leftarrow R_d + 1$	Z,N,V	1
DEC	R_d	Decrement	$R_d \leftarrow R_d - 1$	Z,N,V	1
TST	R_d	Test for zero or minus	$R_d \leftarrow R_d \& R_d$	Z,N,V	1
CLR	R_d	Clear register	$R_d \leftarrow R_d \oplus R_d$	Z,N,V	1
SER	R_d	Set register	$R_d \leftarrow \$FF$	None	1
MUL	R_d, R_r	Multiply unsigned	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULS	R_d, R_r	Multiply signed	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULSU	R_d, R_r	Multiply signed with unsigned	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
FMUL	R_d, R_r	Fractional MUL	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULS	R_d, R_r	Fractional MULS	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULSU	R_d, R_r	Fractional MULSU	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
Branch Instructions					
RJMP	K	Relative jump	PC ← PC + K + 1	None	1
IJMP		Indirect jump to (Z)	PC ← Z	None	2
JMP	K	Direct jump	PC ← K	None	2
RCALL	K	Relative subroutine call	PC ← PC + K + 1	None	1
ICALL		Indirect call to (Z)	PC ← Z	None	2
CALL	K	Direct subroutine call	PC ← K	None	2
RET		Subroutine return	PC ← Stack	None	2
RETI		Interrupt return	PC ← Stack	I	2

INST.	OPC.	FUNCITONS	OPERATION	FLAG	CYCLE
Branch Instructions (Cont'd)					
CPSE	R_d, R_r	Compare, skip if equal	If(R_d=R_r) PC ← PC + 2 or 3	None	1/2
CP	R_d, R_r	Compare	R_d - R_r	Z,N,V,C,H	1
CPC	R_d, R_r	Compare with carry	R_d - R_r – C	Z,N,V,C,H	1
CPI	R_d, K	Compare with immediate	R_d - K	Z,N,V,C,H	1
SBRC	R_r, b	Skip if bit in register cleared	If(R_r(b)=0) PC ← PC + 2 or 3	None	1/2
SBRS	R_r, b	Skip if bit in register set	If(R_r(b)=1) PC ← PC + 2 or 3	None	1/2
SBIC	P, b	Skip if bit in I/O cleared	If(P(b)=0) PC ← PC + 2 or 3	None	1/2
SBIS	P, b	Skip if bit in I/O set	If(P(b)=1) PC ← PC + 2 or 3	None	1/2
BRBS	s, k	Branch if status flag set	If(SREG(S)=1) PC ← PC + K + 1	None	1/2
BRBC	s, k	Branch if status flag cleared	If(SREG(S)=0) PC ← PC + K + 1	None	1/2
BREQ	k	Branch if equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zerio, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	f (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag cleared	f (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	f (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	f (I = 0) then PC ← PC + k + 1	None	1/2
DATA TRANSFER Instructions					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy register word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load immediate	Rd ← K	None	1
LD	Rd, X	Load indirect	Rd ← (X)	None	1
LD	Rd, X+	Load indirect and post-inc.	Rd ← (X), X ← X + 1	None	1
LD	Rd, -X	Load indirect and pre-dec	X ← X - 1, Rd ← (X)	None	1
LD	Rd, Y	Load indirect	Rd ← (Y)	None	1
LD	Rd, Y+	Load indirect and post-inc	Rd ← (Y), Y ← Y + 1	None	1
LD	Rd, -Y	Load indirect and pre-dec	Y ← Y - 1, Rd ← (Y)	None	1
LDD	Rd, Y+q	Load indirect with displacement	Rd ← (Y + q)	None	1

LD	Rd, Z	Load indirect	Rd ← (Z)	None	1
LD	Rd, Z+	Load indirect and post-inc	Rd ← (Z), Z ← Z+1	None	1
LD	Rd, -Z	Load indirect and pre-dec	Z ← Z - 1, Rd ← (Z)	None	1
LDD	Rd, Z+q	Load indirect with displacement	Rd ← (Z + q)	None	1
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store indirect	(X) ← Rr	None	1
ST	X+, Rr	Store indirect and post-inc	(X) ← Rr, X ← X + 1	None	1
ST	-X, Rr	Store indirect and pre-dec	X ← X - 1, (X) ← Rr	None	1
ST	Y, Rr	Store indirect	(Y) ← Rr	None	1
ST	Y+, Rr	Store indirect and post-inc	(Y) ← Rr, Y ← Y + 1	None	1
ST	-Y, Rr	Store indirect and pre-dec	Y ← Y - 1, (Y) ← Rr	None	1
STD	Y+q, Rr	Store indirect with displacement	(Y + q) ← Rr	None	1
ST	Z, Rr	Store indirect	(Z) ← Rr	None	1
ST	Z+, Rr	Store indirect and post-inc	(Z) ← Rr, Z ← Z + 1	None	1
ST	-Z, Rr	Store indirect and pre-dec	Z ← Z - 1, (Z) ← Rr	None	1
STD	Z+q, Rr	Store indirect with displacement	(Z + q) ← Rr	None	1
STS	k, Rr	Store direct	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	2
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	2
LPM	Rd, Z+	Load program and post-inc	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, Z+	Load	Rd ← (Z), Z ← Z+1	None	1
LD	Rd, -Z	Load indirect and pre-dec	Z ← Z - 1, Rd ← (Z)	None	1
LDD	Rd, Z+q	Load indirect with displacement	Rd ← (Z + q)	None	1
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	1
POP	Rd	Pop register from stack	Rd ← STACK	None	1
BIT and BIT-TEST Instructions					
SBI	P, b	Set bit in I/O register	I/O(P, b) ← 1	None	1
CBI	P, b	Clear bit in I/O register	I/O(P, b) ← 0	None	1
LSL	Rd	Logical shift left	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 through carry	Rd(0)←C, Rd(n+1) ← Rd(n), C←Rd(7)	Z	1
ROR	Rd	Rotate right through 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	Swap nibbles	Rd(3:0) ← Rd(7:4), Rd(7:4) ← Rd(3:0)	None	1
BSET	s	Flag set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit store from register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1

CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1
CLN		Clar negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1
CLI		Global interrupt disable	I ← 0	I	1
SES		Set signed test flag	S ← 1	S	1
CLS		Clear signed test flag	S ← 0	S	1
SEV		Set 2’s complement overflow	V ← 1	V	1
CLV		Clear 2’s complement overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep		None	1
WDR		Watchdog reset		None	1
BREAK		Software break	Only for debug purpose	None	N/A

Package Definitions



LQFP32L Dimension

Simboly	Min.	Typical.	Max.	Unit
D	8.90	9.00	9.10	mm
D1	6.90	7.00	7.10	mm
b	0.15	0.20	0.25	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

功能概述

- **高性能低功耗 8 位 LGT8XM 内核**
- **高级 RISC 构架**
  131 条指令，80%以上为单周期执行
  32x8 个通用工作寄存器
  32MHz 工作时最高可达 32MIPS 的执行效率
  内部单周期乘法器(8x8)
- **非易失程序与数据存储空间**
  8Kbytes 片上可在线编程 **FLASH** 程序存储器
  1Kbytes 内部 **SRAM**
  504 字节数据 **FLASH**，支持字节访问(**E2PROM**)
  全新的程序加密算法，保证用户代码安全
- **外设控制器**
  两个具有独立预分频器的 8 位定时器，支持比较输出模式
  一个具有独立预分频器的 16 为定时器，支持输入俘获和比较输出
  内部 32KHz 可校准 **RC** 振荡器实现实时计数器功能
  最多可支持 6 路 **PWM** 输出
  8 通道 10 位高速模数转换器(**ADC**)
  - 3 通道差分输入，支持增益控制
  - 内置温度传感器(**Thermal Sensor**)

  2 通道模拟比较器，支持来自 ADC 输入通道的扩展
  可编程看门狗定时器（**WDT**）
  可编程同步/异步串行接口（**USART**）
  同步外设接口(**SPI**)，可编程主/从工作模式
  可编程双线串行接口(**TWI**)，兼容 **I2C** 主从模式
- **特殊处理器功能**
  SWD 双线调试/量产接口
  外部中断源与 I/O 电平变化中断支持
  内置上电复位电路（**POR**）与 3 级低电压检测电路（**LVD**）
  内置 1%可校准 32MHz RC 振荡器
  内置 1%可校准 32KHz RC 振荡器
  外部支持 32.768KHz 以及 400K~32MHz 晶振输入
  最多 12 通道触摸按键 I/O
  8 通道电流 **NMOS I/O**，最高可扇入 80mA 电流
- **I/O 与封装**
  QFP32L (最多可提供 30 个 I/O)
  S/SOP28L（最多可提供 26 个 I/O）
- **工作环境**
  工作电压： 1.8V ~ 5.5V
  工作频率： 0 ~ 32MHz
  工作温度： -40C ~ +85C
  HBM ESD： 5000V+

LogicGreen

8-bit LGT8XM

RISC Microcontroller with
8192 Bytes In-System
Programmable
FLASH Memory

LGT8F88A

Datasheet summary
Version 1.1.0

应用领域

厨电
- 电磁炉
- 微波炉
- 电饭锅等

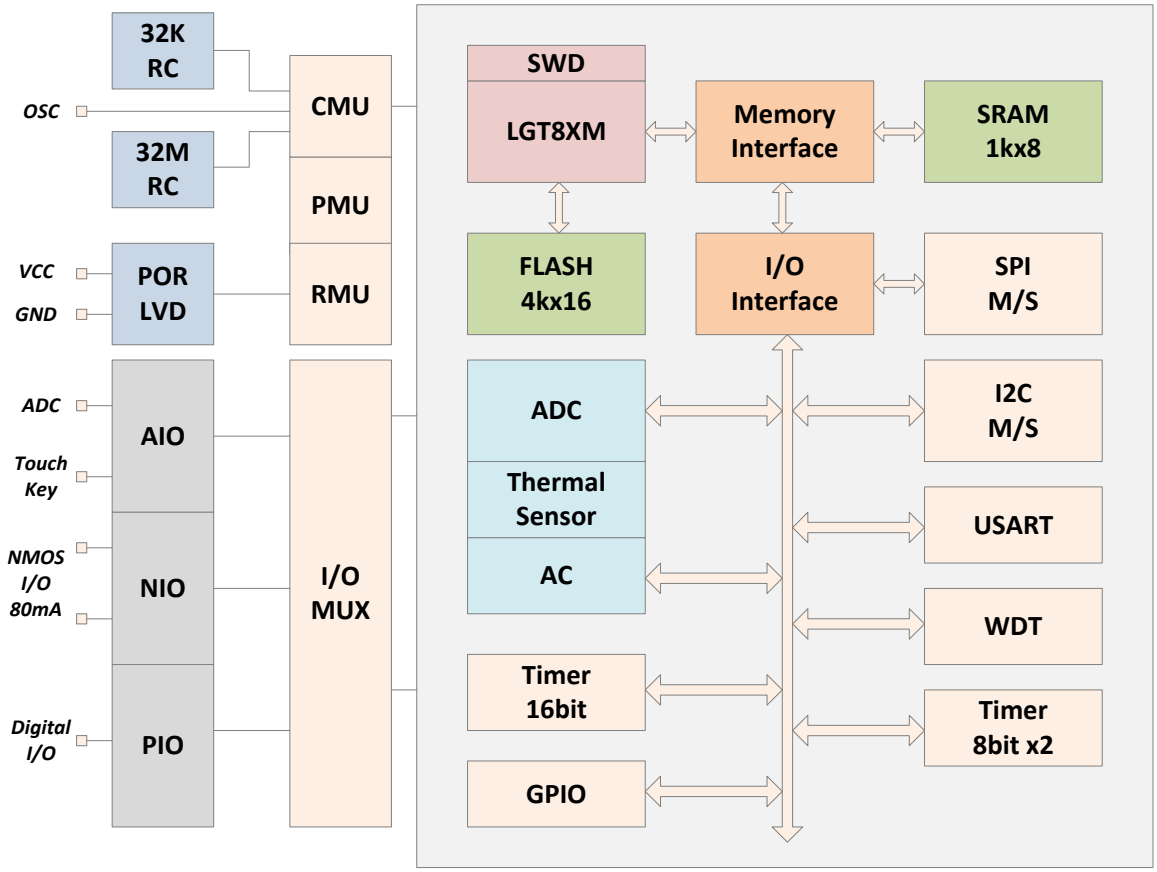
小家电
- 豆浆机
- 咖啡壶
- 热水器等

智能控制电路
- 蓄电池管理
- 电动产品
- 智能玩具

手持仪器

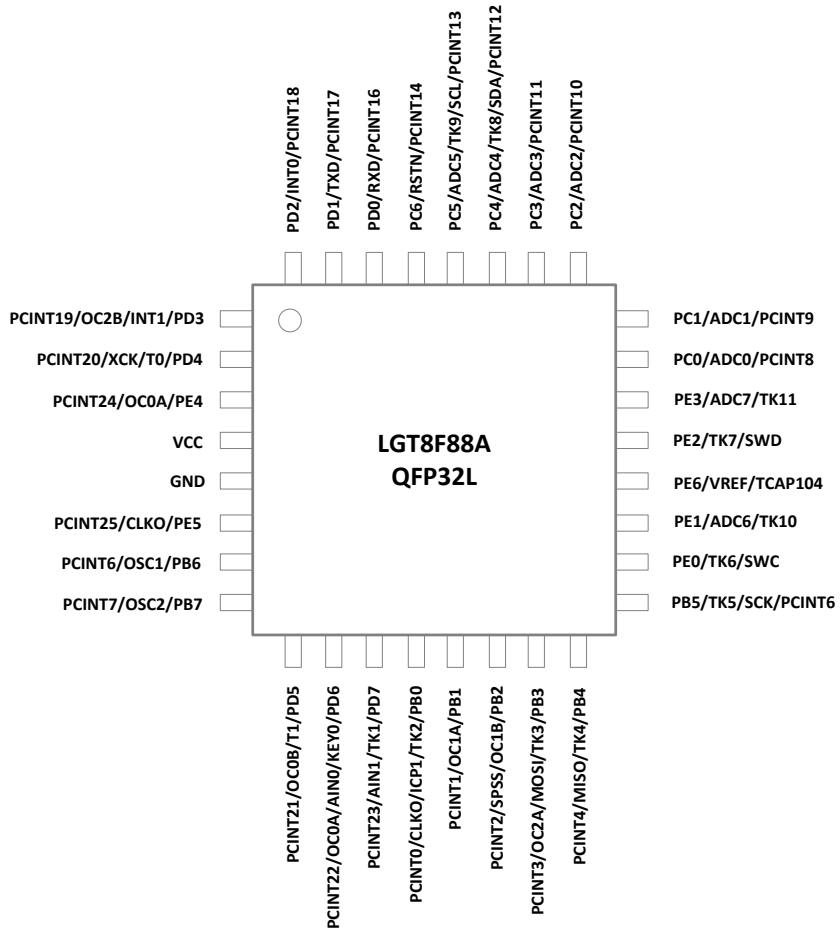
系统框架

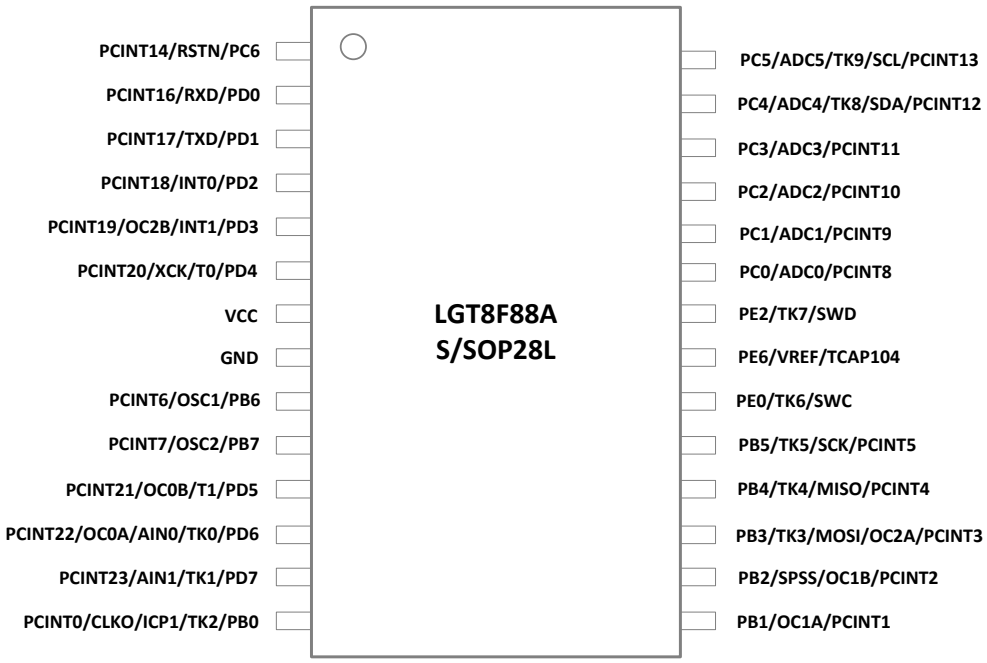
LGT8F88A Diagram



模块名称	模块功能
SWD	调试模块，同时实现在线调试与 ISP 功能
LGT8XM	8bit 高性能 RISC 内核
CMU	时钟管理模块，产生系统需要的各种工作时钟
PMU	功耗管理模块， 负责管理系统工作状态之间的转换
RMU	复位产生模块
POR/LVD	上电复位模块与低电压检测电路
ADC Thermal Sensor AC	8 通道 10 位模数转换器 温度传感器 模拟比较器
AIO NIO PIO	ADC/触摸按键输入通道 80mA 驱动能力的 NMOS I/O， 可直接驱动共阴极 LED 可编程数字 I/O
WDT	看门狗复位模块

封装定义


PD2/INT0/PCINT18
PD1/TXD/PCINT17
PD0/RXD/PCINT16
PC6/RSTN/PCINT14
PC5/ADC5/TK9/SCL/PCINT13
PC4/ADC4/TK8/SDA/PCINT12
PC3/ADC3/PCINT11
PC2/ADC2/PCINT10
PCINT19/OC2B/INT1/PD3
PCINT20/XCK/T0/PD4
PCINT24/OC0A/PE4
VCC
GND
PCINT25/CLKO/PE5
PCINT6/OSC1/PB6
PCINT7/OSC2/PB7
LGT8F88A
QFP32L
PC1/ADC1/PCINT9
PC0/ADC0/PCINT8
PE3/ADC7/TK11
PE2/TK7/SWD
PE6/VREF/TCAP104
PE1/ADC6/TK10
PE0/TK6/SWC
PB5/TK5/SCK/PCINT6
PCINT21/OC0B/T1/PD5
PCINT22/OC0A/AIN0/KEY0/PD6
PCINT23/AIN1/TK1/PD7
PCINT0/CLKO/ICP1/TK2/PB0
PCINT1/OC1A/PB1
PCINT2/SPSS/OC1B/PB2
PCINT3/OC2A/MOSI/TK3/PB3
PCINT4/MISO/TK4/PB4


PCINT14/RSTN/PC6
PCINT16/RXD/PD0
PCINT17/TXD/PD1
PCINT18/INT0/PD2
PCINT19/OC2B/INT1/PD3
PCINT20/XCK/T0/PD4
VCC
GND
PCINT6/OSC1/PB6
PCINT7/OSC2/PB7
PCINT21/OC0B/T1/PD5
PCINT22/OC0A/AIN0/TK0/PD6
PCINT23/AIN1/TK1/PD7
PCINT0/CLKO/ICP1/TK2/PB0
LGT8F88A
S/SOP28L
PC5/ADC5/TK9/SCL/PCINT13
PC4/ADC4/TK8/SDA/PCINT12
PC3/ADC3/PCINT11
PC2/ADC2/PCINT10
PC1/ADC1/PCINT9
PC0/ADC0/PCINT8
PE2/TK7/SWD
PE6/VREF/TCAP104
PE0/TK6/SWC
PB5/TK5/SCK/PCINT5
PB4/TK4/MISO/PCINT4
PB3/TK3/MOSI/OC2A/PCINT3
PB2/SPSS/OC1B/PCINT2
PB1/OC1A/PCINT1

引脚说明

引脚名称	功能描述
VCC	系统电源(1.8V ~ 5.5V)
GND	系统地
OSC1 OSC2	外部晶振输入输出
RSTN	外部异步复位输入
RXD TXD XCK	同步/异步 UART 接口
INT0/1	外部中断输入、异步唤醒源
OC0A/B	定时器 0 比较输出(PWM0A/B)
OC1A/B	定时器 1 比较输出(PWM1A/B)
OC2A/B	定时器 2 比较输出(PWM2A/B)
SCL SDA	TWI 双线数据接口(I2C)
SCK SPSS MISO MOSI	SPI 接口
T0	定时器 0 外部时钟输入
T1	定时器 1 外部时钟输入
ICP1	定时器 1 外部俘获输入
SWD SWC	SWD 调试接口
PCINTX	引脚电平改变中断功能
ADC7…0	ADC 输入通道
TK11…0	触摸按键输入通道
VREF/TCAP104	ADC 外部参考电压输入 触摸按键外部 0.1uF 滤波电容
AIN0 AIN1	模拟比较器外部输入
CLKO	系统时钟输出
PB7…0	可编程 I/O
PD7…0	可编程 I/O
PC6…0	可编程 I/O
PE6…0	可编程 I/O
PD5…0 PE5…4	大电路驱动 I/O，最大可直接驱动到 80mA

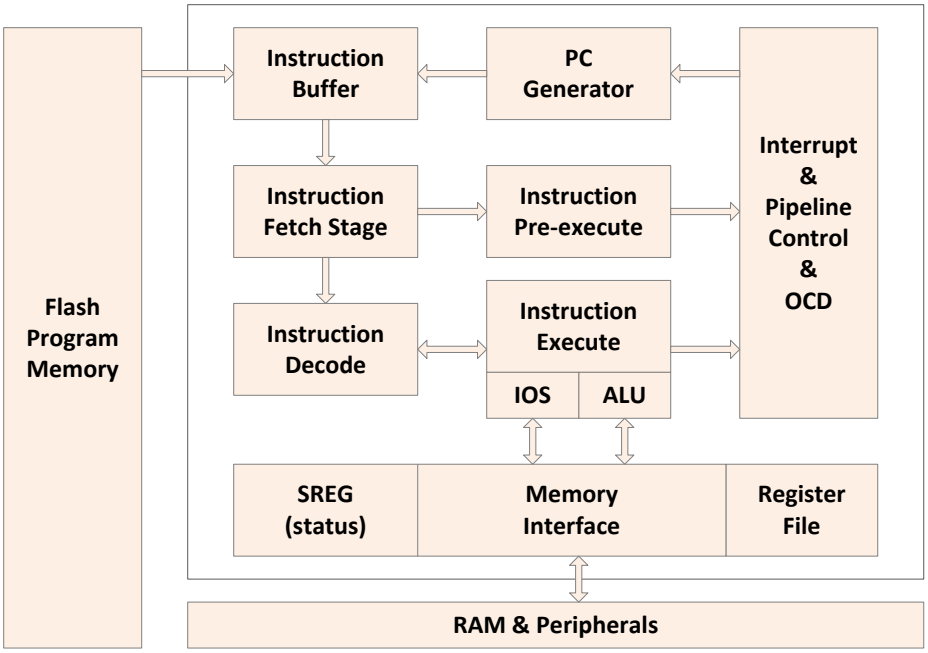
LGT8XM 内核

- 低功耗设计
- 高效率 RISC 构架
- 130 条指令，其中 80%以上为单周期
- 内嵌在线调试(OCD)支持

概述

本章节主要描述 LGT8XM 内核构架和功能。 内核是 MCU 的大脑,，负责保证程序的正确执行，因此内核必须能够准确的执行计算，控制外设以及处理各种中断。

下图为 LGT8XM 内核的结构：



为了实现更大的效率和并行性，LGT8XM 内核采用哈弗构架 – 独立的数据和程序总线。指令通过一个优化的两级流水线执行，两级流水线能够减少流水线中无效指令的个数，减少了对 FLASH 程序存储器的访问量，因此可以降低内核运行的功耗。同时 LGT8XM 内核在取指令的前级中增加了指令缓存（可以同时缓存 2 条指令），通过在取指令周期的预执行模块，进一步减少了对 FLASH 程序存储器的访问频率；经我们大量测试，LGT8XM 可以比其他同类构架的内核减少约 50%对 FLASH 的访问，大大降低了整个系统的运行功耗。

LGT8XM 内核具有 32 个 8 位高速访问的通用工作寄存器(Register file)，有助于实现单周期的算术逻辑运算(ALU)。一般情况下，ALU 运算的两个操作数均来自与通用工作寄存器，ALU 运算的结果也会在一个周期内写入到寄存器文件中。

32 个通过工作寄存器中的 6 个用于两两结合构成三个 16 位寄存器，可用于间接寻址地址指针，用于访问外部存储空间以及 FLASH 程序空间。LGT8XM 支持单周期的 16 位算术运算，极大的提高了间接寻址的效率。LGT8XM 内核中这三个特殊的 16 位寄存器被命名为 X, Y, Z 寄存器，将在后面详细介绍。

ALU 支持寄存器之间以及常数与寄存器之间的算术逻辑运算，单个寄存器的运算也可以在 ALU 中执行。ALU 运算完成后，运算结果对内核状态的影响更新到状态寄存器中(SREG)。

程序流程控制通过条件和无条件跳转/调用实现，可以寻址到所以的程序区域。大部分 LGT8XM 指令为 16 位。每个程序地址空间对应一个 16 位或者 32 位的 LGT8XM 指令。

内核响应中断或子程序调用后，返回地址(PC)被存储在堆栈中。堆栈被分配在系统的一般数据 SRAM 中，因此堆栈的大小仅受限于系统中 SRAM 的大小和用法。所有的支持中断或子程序调用的应用，必须首先初始化堆栈指针寄存器(SP)，SP 可以通过 IO 空间访问。数据 SRAM 可以通过 5 种不同的寻址模式访问。LGT8XM 的内部存储空间都被线性的映射到一个统一的地址空间。具体请参考存储章节的介绍。

LGT8XM 内核包含了一个灵活的中断控制器，中断功能可以通过状态寄存器中的一个全局中断使能位控制。所有的中断都有一个独立的中断向量。中断的优先级与中断向量地址有对应关系，中断地址越小，中断的优先级就越高。

I/O 空间包含了 64 个可以通过 IN/OUT 指令直接寻址的寄存器空间。这些寄存器现实对内核控制以及状态寄存器，SPI 以及其他 I/O 外设的控制功能。这部分空间可以通过 IN/OUT 指令直接访问，也可以通过他们映射到数据存储器空间的地址访问(0x20 – 0x5F)。另外，LGT8F88A 也包含扩展的 I/O 空间，他们被映射到数据存储空间 0x60 – 0xFF，这里只能使用 ST/STS/STD 以及 LD/LDS/LDD 指令访问。

算术逻辑运算单元（ALU）

LGT8XM 内部包含了一个 16 位的算术逻辑运算单元，能够在一个周期内完成 16 为数据的算术运算。高效的 ALU 与 32 个通用工作寄存器相连。能够在一个周期内完成两个寄存器或者寄存器与立即数之间的算术逻辑运算。ALU 的运算分为三种：算术，逻辑以及位运算。同时 ALU 部分也包含了一个单周期的硬件乘法器，能够在一个周期内实现两个 8 位寄存器直接的有符号或者无符号运算。请参考指令集部分的详细介绍。

状态寄存器（SREG）

状态寄存器中主要保存了因执行最近一次 ALU 运算而产生的结果信息。这些信息用于控制程序执行流程。状态寄存器是在 ALU 操作完全结束后更新，这样就可以省去了使用单独的比较指令，可以带来更加紧凑高效的代码实现。

状态寄存器的值在响应中断和从中断中退出时并不会自动保存和恢复，这需要软件去实现。

SREG 寄存器定义

SREG 系统状态寄存器								
地址: 0x3F (0x5F)				默认值: 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
Initial	0	0	0	0	0	0	0	0
位定义								
[0]	C	进位标志，表示算术或逻辑操作导致了进位，具体请参考指令描述						
[1]	Z	零标志，表示算术或逻辑运算的结果为零，请参考指令描述部分						

[2]	N	负标志，表示算术或逻辑运算产生了一个负数，请参考指令描述部分
[3]	V	溢出标志，表示二进制补码运算结果产生溢出，请参考指令描述部分
[4]	S	符号位，等效于 N 与 V 的异或运算结果，具体请参考指令描述部分
[5]	H	半进位标志，在 BCD 运算中有用，表示字节运算产生了的半进位
[6]	T	临时位，位复制(BLD)和位存储(BST)指令中使用，T 位将作为一个临时的存储位，用于临时存放通用寄存器中的某一位的值。具体请参考指令描述部分
[7]	I	全局中断使能位，必须设置此位为 1 才能使能内核响应中断事件。不同的中断源是由独立的控制位控制。全局中断使能位是控制中断信号进入内核的最后一道屏障。I 位在内核响应中断向量后由硬件自动清除，在执行中断返回指令(RETI)后自动置位。I 位也可以使用 SEI 和 CLI 指令改变，请参考指令描述部分

通用工作寄存器

通用工作寄存器根据 LGT8XM 指令集构架优化。为了达到内核执行需要的效率和灵活性，LGT8XM 内部的通用工作寄存器支持一下几种访问模式：

- 一个 8 位的读同时一个 8 位的写操作
- 两个 8 位的读同时一个 8 位的写操作
- 两个 8 位的读同时一个 16 位的写操作
- 一个 16 位的读同时一个 16 位的写操作

LGT8XM 通用工作寄存器

	7	*0*	*Addr.*	
通用工作寄存器	**R0**		***0x00***	
	R1		***0x01***	
	R2		***0x02***	
	…			
	R13		***0x0D***	
	R14		***0x0E***	
	R15		***0x0F***	
	R16		***0x10***	
	R17		***0x11***	
	…			
	R26		***0x1A***	***X寄存器低字节***
	R27		***0x1B***	***X寄存器高字节***
	R28		***0x1C***	***Y寄存器低字节***
	R29		***0x1D***	***Y寄存器高字节***
	R30		***0x1E***	***Z寄存器低字节***
	R31		***0x1F***	***Z寄存器高字节***

大部分指令能够直接访问到全部的通用工作寄存器，他们大部分也都是单周期指令。

如上图所示，每一个寄存器都对应一个数据存储空间的地址，这些通用工作寄存器被映射到数据储存空间。尽快他们不没有真正的存在于 SRAM 中，但这种统一映射的存储组织给访问他们带来了很大的灵活性。X/Y/Z 寄存器可以作为指针索引到任何通用寄存器。

X/Y/Z 寄存器

寄存器 R26…R31 可以两两组合，构成三个 16 位寄存器。这三个 16 位寄存器主要用于间接寻址访问的地址指针，X/Y/Z 寄存器结构如下：

	15	XH		XL	0
X寄存器	7	0	7		0
	R27 (0x1B)		*R26 (0x1A)*		
	15	YH		YL	0
Y寄存器	7	0	7		0
	R29 (0x1D)		*R28 (0x1C)*		
	15	ZH		ZL	0
Z寄存器	7	0	7		0
	R31 (0x1F)		*R30 (0x1E)*		

在不同的寻址模式下， 这些寄存器被用作固定偏移，自动递增以及自动递减的地址指针，具体细节请参考指令描述部分。

堆栈指针

堆栈用于存储临时数据，局部变量以及中断和子程序调用的返回地址。需要特别注意的是，堆栈别设计为从高地址向低地址生长。堆栈指针寄存器(SP)总是指向堆栈的顶部。堆栈指针指向数据 SRAM 所在的物理空间，这里存放子程序或中断调用必须的堆栈空间。PUSH 指令将会使得堆栈指针递减。

堆栈在 SRAM 中的位置必须在子程序执行或者中断使能之前由软件正确的设置。一般情况下是将堆栈指针初始化指向 SRAM 的最高地址处。堆栈指针必须设置为高位 SRAM 开始地址。SRAM 在系统数据存储映射的地址请参考系统数据存储部分。

堆栈指针相关的指令

指令	堆栈指针	描述
PUSH	增加 1	数据压入堆栈
CALL ICALL RCALL	增加 2	中断或者子程序调用的返回地址压入堆栈
POP	减少 1	数据从堆栈取出
RET RETI	减少 2	中断或者子程序调用的返回地址从堆栈中取出

堆栈指针由分配在 I/O 空间的两个 8 位的寄存器构成。堆栈指针的实际长度与系统实现相关。在 LGT8XM 构架的有些芯片实现中，数据空间非常小，以至于仅仅 SPL 就能满足寻址需要，这种情况下，SPH 寄存器将不会出现。

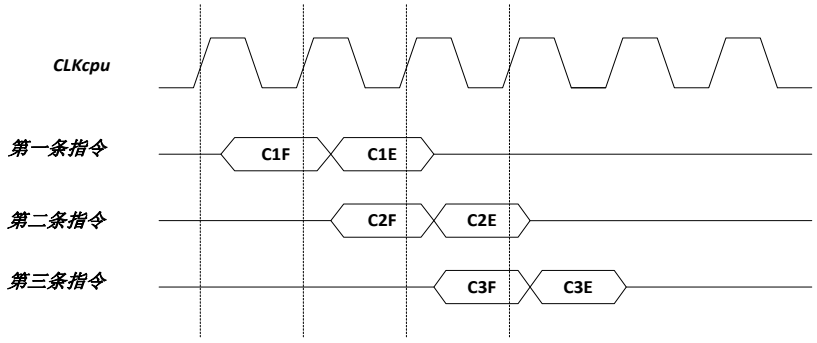
SPH/SPL 堆栈指针寄存器定义

SPH/SPL 堆栈指针寄存器		
SPH: 0x3E (0x5E)		默认值: RAMEND
SPL: 0x3D (0x5D)		
SP	SP[15:0]	
R/W	R/W	
Initial	RAMEND	
位定义		
[7:0]	SPL	堆栈指针低 8 位
[15:8]	SPH	堆栈指针高 8 位

指令执行时序

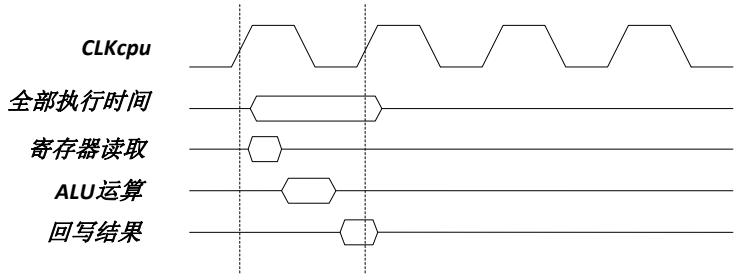
这一章节描述指令执行的一般时序概念。LGT8XM 内核由内核时钟(CLKcpu)驱动，这个时钟直接来自与系统的时钟源选择电路。

下图展示了哈弗构架与快速访问寄存器文件概念基础上的指令流水线执行时序。这是使得内核能够获得 1MIPS/MHz 的执行效率的物理保证。



从上图可以看出，第一条指令的执行期间同时会读出第二条指令。当第二条指令进入执行期间，同时又会读出第三条指令。这样在整个执行期间，并不需要为读取指令花费额外的周期，从流水线上看，实现了每个周一执行一条指令的效率。

下图展示通用工作寄存器的访问时序，在一个周期内，ALU 操作使用到两个寄存器作为操作数，并在这个周期内将 ALU 执行结果写入到目标寄存器中。



复位与中断处理

LGT8XM 支持多个中断源。这些中断以及复位向量在程序空间都对应一个独立的程序向量入口。一般而言，所有的中断都有单独的控制位控制。当设置了该控制位，并且使能了内核的全局中断使能位后，内核才能响应这个中断。

最低的程序空间默认保留为复位以及中断向量区域。LGT8F88A 支持的完整的中断列表请参考中断章节的介绍。这个列表同时也决定了不同中断的优先级。向量地址越低的中断，对应的中断优先级就越高。复位(RESET)具有最高的优先级，然后是 INT0 – 外部中断请求 0. 中断向量表的起始地址（复位向量除外）可以被重新定义到任何 256 字节对齐的开始处，需要通过 MCU 控制寄存器(MCUCR)中的 IVSEL 位以及 IVBASE 向量基地址寄存器实现。

当内核响应中断后，全局中断使能标志为 I 会被硬件自动清除。用户可以通过将 I 位使能实现中断嵌套。这样任何随后发生的中断都会中断当前的中断服务程序。I 位在执行中断返回指令(RETI)后自动置位，从而可以正常响应随后发生的中断。

有种基本的中断类型。第一种类型由事件触发，中断事件发生后置位中断标志位。对于这种中断来说，内核响应中断请求后，当前的 PC 值被直接替换为实际的中断向量地址，执行对应的中断服务子程序，同时硬件自动清除掉中断标志位。中断标志位也可以通过向中断标志位的位置写 1 清除。如果在发生中断时，中断使能位被清除，中断标志位仍然会被设置以记录中断事件。等到中断使能后，这个记录的中断事件会被立即响应。同样，如果在中断发生时，全局中断使能位(SERG.I)被清除，对应的中断标志位也会被设置以记录中断事件，等到全局中断使能位被设置后，这些被记录的中断将会依照优先级依次执行。

第二种中断类型是当中断条件一直存在时，中断就一直响应。这种中断不需要中断标志位。如果中断条件在中断使能之前消失，这个中断将不会得到响应。

当 LGT8XM 内核从中断服务子程序中退出后，执行流程会返回到主程序中。在主程序中执行一条或几条指令后，才能响应其他等待的中断请求。

需要注意的是，系统状态寄存器(SREG)在进入中断服务后并不会自动保存，也不会在从中断服务返回后自动恢复。它必须由软件负责处理。

当使用 CLI 指令禁止中断后，中断将会被立即禁止。在 CLI 指令之后发生的所以中断都不会得到响应。即使是和 CLI 指令执行时同时发生的中断，也不会被响应。下面的例子中说明如何利用 CLI 避免中断打乱 EEPROM 的写时序：

汇编代码实例

```
IN R16, SREG       ; store SREG value
CLI                ; disable interrupts during timed sequence
SBI EECR, EEMPE    ; start EEPROM write
SBI EECR, EEPE
OUT SREG, R16      ; restore SREG value ( including I bit)
```

C 语言代码实例

```
char cSREG;
cSREG = SREG;   /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1 << EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (including I-bit) */
```

当使用 SEI 指令使能中断时，在 SEI 指令之后的一条指令将会首先在中断得到响应之前被执行，如下面的代码实例：

汇编代码实例

```
SEI      ; set Global Interrupt Enable
SLEEP    ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending interrupt(s)
```

C 语言代码实例

```
__enable_interrupt();   /* set Global Interrupt Enable */
__sleep();   /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

中断响应时间

LGT8XM 内核针对中断响应进行了优化，使得任何中断在 4 个系统时钟周期内一定得到响应。4 个系统时钟周期后，中断服务子程序进入执行周期。在这 4 个时钟内，中断之前的 PC 值被压入堆栈，系统执行流程流程跳转到中断向量对应中断服务程序。如果中断发生在一个多周期指令执行期间，内核将保证当前指令正确的执行结束。如果中断发生在系统处于休眠状态下(SLEEP)，中断响应需要额外增加 4 个时钟周期。这增加的时钟周期用于从选择的休眠模式下唤醒操作的同步周期。休眠模式的具体描述，请参考功耗管理的相关章节。

从中断服务子程序中返回需要 2 个时钟周期。在这 2 个时钟周期内，PC 从堆栈中恢复，堆栈指针加 2，SREG(I)位设置为 1。

存储子系统

概述

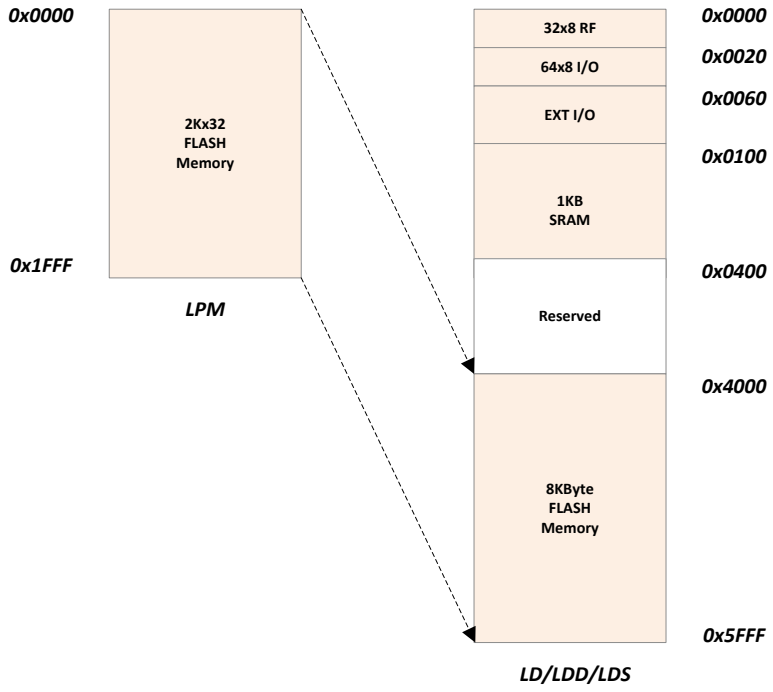
本章节主要描述 LGT8F88A 内部不同的存储单元。LGT8XM 构架支持两种主要的内部存储空间，分别是数据存储空间和程序存储空间。另外，LGT8F88A 内部也包含了数据 FLASH，通过内部的控制器可以实现 EEPROM 接口的数据存储功能。另外，LGT8F88A 系统中还包含了特殊的存储单元，用于存放系统配置信息以及芯片的全局设备号(GUID)。

系统可编程 FLASH 程序存储单元

LGT8F88A 内部包含 8K 字节的片上在线可编程 FLASH 程序存储单元。因为所有 LGT8XM 的指令都是 16 或者 32 位宽，这个 FLASH 就选择了 2K x 32 的结构。32 位的接口宽度，可以实现一次读取 2 条 LGT8XM 指令，这样可以最多降低一倍对 FLASH 的访问频率，不但提高的访问效率， 也减少了因访问 FLASH 产生的功耗。

程序 FLASH 能保证至少 20,000 次以上的擦写周期。LGT8F88A 内部集成 FLASH 接口控制器，能够实现在系统编程(ISP)以及程序的自升级功能。具体实现细节请参考本章在关于 FLASH 接口控制器部分的描述。

程序空间也可以通过 LPM 指令直接访问(读取)，这个特点可以实现应用相关的常数查找表。同时 FLASH 程序空间也被映射到系统数据存储空间内，这样也用户也可以使用 LD/LDD/LDS 实现对 FLASH 空间的访问。程序空间被映射到数据存储空间 0x4000 开始的地址范围内。如下图所示：



SRAM 数据存储单元

LGT8F88A 是一种相对复杂的微控制器，它支持多种不同类型的外设，这些外设的控制器被分配在 64 个 I/O 寄存器空间内。可以直接通过 IN/OUT 指令访问。另一些外设的控制寄存器分配在 0x60 ~ 0xFF 区域内，由于这部分空间是映射到数据存储空间内，只能通过 ST/STS/STD 以及 LD/LDS/LDD 等指令访问。

LGT8F88A 的系统数据存储空间从 0 地址开始，分别映射了通用工作寄存器文件，I/O 空间，扩展 I/O 空间以及内部数据 SRAM 空间。最开始的 32 个字节地址对应 LGT8XM 内核 32 个通用工作寄存器。接下来的 64 个地址是可以通过 IN/OUT 指令直接访问的标准 I/O 空间。然后的 160 个地址是扩展 I/O 空间，在接下来就是 1024 字节的数据 SRAM。从 0x4000 开始到 0x5FFF 结束的这部分空间，映射了 8192 字节的 FLASH 程序存储单元。

系统数据存储统一映射空间

系统支持 5 种不同的寻址模式可以覆盖到整个数据空间：直接访问，带偏移的间接访问，间接访问，访问前递减地址的间接访问，访问后递增地址的间接访问。通用工作寄存器 R26 到 R31 用于间接访问的地址指针。间接访问可以寻址整个数据存储空间。带偏移地址的间接访问能够寻址到以 Y/Z 寄存器为基地址的附近 63 个地址空间。

当使用支持地址自动递增/递减的寄存器间接访问模式，地址寄存器 X/Y/Z 会在访问发生前/后自动由硬件递减/递增。具体请参考指令集描述部分。

通用 I/O 寄存器

LGT8F88A 的 I/O 空间有三个通用 I/O 寄存器 **GPIOR2/1/0**，这三个寄存器可以使用 IN/OUT 指令访问，用于存放用户自定义数据。

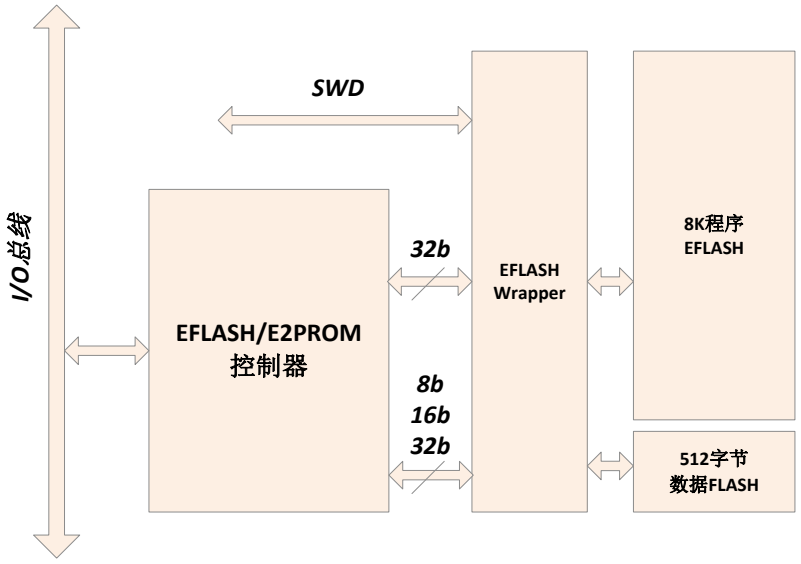
EFLASH/E2PROM 接口控制器

LGT8F88A 内部包含 512 字节的数据 FLASH，其中最后 8 个字节保留给系统配置信息(熔丝)使用。剩下的 504 字节被 EFLASH 控制器管理，实现 E2PROM 接口访问逻辑。可以支持 8/16/32 位宽度的读写访问。E2PROM 接口模拟采用擦写均衡的算法，可以将数据 FLASH 的使用周期提高 1 倍左右，能够保证 50,000 次以上的擦写周期。

FLASH 控制器也可以访问到 FLASH 程序空间，可以通过软件实现在线自动升级固件的功能。通过 FLASH 控制器访问程序 FLASH 空间，只支持 32 位宽度的读写访问。

E2PROM 以及程序 FLASH 空间的访问细节，请参考下面的详细描述。

FLASH/E2PROM 控制器结构图



FLASH/E2PROM 接口读写访问

FLASH 控制器所有的控制器寄存器都可以通过 I/O 空间访问。对 FLASH 以及 E2PROM 空间的操作，也都是通过配置和控制这些寄存器实现。详细的使用方法，将会在寄存器描述部分单独说明。

FLASH/E2PROM 的写访问时间，请参考后面给出的表格。FLASH 控制器能够自动更新当前操作的状态，用户软件可以通过检测这些状态确定当前操作是否完成，从而开始下一个字节的操作。如果用户代码中包含了对 FLASH/E2PROM 的操作，那么就需要遵循一些原则。首先是在上电或掉电期间，V_{CC} 上升和下降的比较慢，导致设备会有段时间在低电压下运行，这会影响到当前系统运行的最大频率的最小电压要求，也会对 FLASH 的编程操作产生影响。此时就需要采取必要的保护措施。这将会在下面一小节中详细描述。

为避免对 E2PROM 的误操作，对 E2PROM 的操作必须遵循一个特殊的流程。请参考本章节最后对 EFLASH/E2PROM 控制寄存器的描述。

当操作 EFLASH/E2PROM 时，LGT8XM 内核的执行将会被保持住，直到操作完成后，内核才能恢复运行。

FLASH/E2PROM 操作的保护措施

在 VCC 低压期间，FLASH/E2PROM 的数据操作可能会因为电压太低而发生错误。这和使用板级 E2PROM 芯片一样，可以使用相同的设计方案。

FLASH/E2PROM 数据在低压下的操作错误可能由两种原因。首先，一个正常的 FLASH/E2PROM 操作需要一个最小工作电压，低于这个电压，操作将会失败而导致数据发生错误。第二个原因，是内核运行在某一频率下，也同样需要一个最小电压要求，当低于这个电压，而 CPU 又保持在这个频率下运行，将会导致指令执行出错，从而使得 FLASH/E2PROM 的操作发生错误。

可以通过下面简单的方法避免类似问题：

在供电电压较低时，让 CPU 保持复位状态。这可以通过配置内部的低压检测电路(VDT)实现。如果 VDT 检测到当前的工作电压低于设置的阀值，VDT 将会输出一个复位信号。如果 VDT 的阀值不能满足应用的需要，可以考虑在外部增加一个复位电路。

I/O 寄存器空间

I/O 空间的详细定义，请参考 LGT8F88A 数据手册中“寄存器概述”章节。

LGT8F88A 所以的外设都被分配到 I/O 空间。所有的 I/O 空间地址都可以被 LD/LDS/LDDD 以及 ST/STS/STD 指令访问。访问的数据都是通过 32 个通用工作寄存器传递。在 0x00 ~ 0x1F 之间的 I/O 寄存器可以通过位寻址指令 SBI 和 CBI 访问。在这些寄存器中，某一个位的值可以使用 SBIS 和 SBIC 指令检测，用以控制程序的执行流程。具体请参考指令集描述部分。

当使用 IN/OUT 指令访问 I/O 寄存器时，必须寻址 0x00 ~ 0x3F 之间的地址。当使用 LD 或 ST 指令访问 I/O 空间时，必须通过 I/O 空间在系统数据存储器统一映射空间的映射地址访问(加上 0x20 的偏移)。其他一些分配在扩展 I/O 空间的外设寄存器(0x60 ~ 0xFF)， 只能够使用 ST/STS/STD 和 LD/LDS/LDD 指令访问。

为了与未来的设备兼容，保留位在写操作时必须写 0。不能在保留的 I/O 空间上执行写操作。

一些寄存器中包括了状态标志，需要被写 1 才能清零。需要注意的是，CBI 和 SBI 指令仅仅支持特定的位，因此 CBI/SBI 也只能工作在包含这些状态标志的寄存器上。除此之外，CBI/SBI 指令只能工作在 0x00 到 0x1F 这个地址范围内的寄存器。

寄存器描述

FLASH/E2PROM 地址寄存器- EEARH/EEARL

EEARH/EEARL		
EEARH: 0x22 (0x42)		默认值: 0x0000
EEARL: 0x21 (0x41)		
EEAR	EEAR[15:0]	
R/W	R/W	

初始值	0x0000	
位定义		
[7:0]	EEARL	EFLASH/E2PROM 访问地址低 8 位。
[12:8]	EEARH	EFLASH/E2PROM 访问地址高 5 位
[15:13]	-	保留不用

当使用 EFLASH/E2PROM 控制器访问程序 FLASH 区域时，EEAR[12:2]用作访问以 4 字节对齐的整个 8192 字节程序空间。EEAR[1:0]只在访问数据寄存器 EEDR 时使用。具体请参考下面关于 EEDR 数据寄存器的描述。

当使用 EFLASH/E2PROM 控制器访问数据 FLASH 区域(E2PROM)时，EEAR[8:0]用于访问 512 字节的数据 FLASH 空间。此时的访问支持 8/16/32 位模式，无论是哪一种模式，EEAR 都是以字节对齐寻址。

另外，512 字节数据 FLASH 最后 8 个字节保留给系统配置信息使用，用户只能使用前 504 字节的区域存放数据。

FLASH/E2PROM 数据寄存器- EEDR

EEDR – FLASH/E2PROM 数据寄存器		
EEDR: 0x20 (0x40)		默认值: 0x00
EEDR	EEDR[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[7:0]	EEDR	EFLASH/E2PROM 数据寄存器

重要说明:

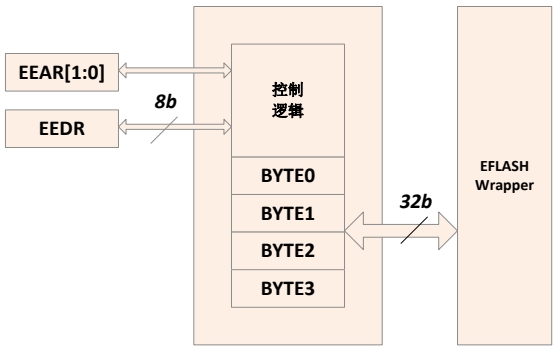
LGT8F88A 内部的 FLASH 为 32 位接口，读/写数据的最小单位为 32 位。因此 FLASH 控制器内部的数据寄存器为 32 位。EEAR[1:0]就用于寻址内部的 32 位数据。

EEDR 是一个 8 位宽的数据寄存器，它的含义根据访问模式不同而不同。当使用 FLASH 控制器访问内部程序 FLASH 时，FLASH 控制器工作在 32 位模式下，此时的 EEDR 作为访问内部 32 位数据寄存器的接口，将和 EEAR[1:0]配合工作。当使用 FLASH 控制器访问数据 FLASH 时，FLASH 控制器访问接口可以工作在 8/16/32 位三种模式。

当工作在 8 位模式时，EEDR 就是对应的就是需要读/写的实际数据，EEAR[8:0]用于寻址 512 字节的数据 FLASH 空间。 硬件将会自动完成 8 位数据到 32 位数据访问的接口转化，用户无需任何额外的操作。

当工作与 16/32 位模式时，EEDR 也将是作为访问内部数据的一个接口，将配合 EEAR[1:0]一起工作。在这 2 种模式下，用户需要通过 EEAR[1:0]与 EEDR，设置好要写入 FLASH 的数据，或者通过 EEAR[1:0]与 EEDR，读出需要的字节数据。

下图说明 I/O 寄存器 EEAR/EEDR 与 FLASH 控制器内部接口之间的关系：



当使用 8 位模式时，EEAR[8:0]配合 EEDR 一起更新指定字节位置的数据，其余位置的数据由 FLASH 控制器内部的控制逻辑自动拼凑。用户不必关心具体实现。
当使用 16 位模式时，用户需要更新 16 位数据，也就是 2 个字节的数据。硬件根据 EEAR[1]来决定更新高 16 位或低 16 位。
当使用 32 位模式时，用户需要更新 32 位数据，也就是全部的 4 个字节。更新数据的方法如下：

```
OUT     EEARL, $0
OUT     EEDR, BYTE0
OUT     EEARL, $1
OUT     EEDR, BYTE1
OUT     EEARL, $2
OUT     EEDR, BYTE2
OUT     EEARL, $3
OUT     EEDR, BYTE3
#设置编程的目标地址
OUT     EEARL, ADDRL
OUT     EEARH, ADDRH
… …
```

关于 FLASH 控制接口的访问模式

FLASH 控制的访问接口分为 8/16/32 位三种模式，其中 8 位模式工作最为简单直接。但为何要还要支持 16/32 两种模式呢？ 这主要是从 FLASH 的特性考虑：
LGT8F88A 内部采用了一个数据接口为 32 位的 FLASH。读写操作以 32 位为最小单位，擦除以页为单位，一页的大小为 128x32(512 字节)。 FLASH 的特性是在改写数据之前必须先擦除。每个最小的编程单位(32 位)可以被重复编程 2 次。
因此，在 8 位模式下，一个最小的编程单位将会被编程 4 次。这样就超出了重复编程的限制。因此需要编程 2 次后必须擦除一次，然后才能执行另外两次编程操作。这样做就增大了 FLASH 的擦写周期，减少了使用寿命。
如果是 16 位模式，可以把最小编程单位分为高 16 位和低 16 分别操作，这样一个最小编程单位可以重复编程 2 次，编程效率和寿命都能有明显提高。32 位工作模式也同样能达到这样的效果。

FLASH/E2PROM 控制器控制寄存器- EECR

EECR – FLASH/E2PROM 控制寄存器								
EECR: 0x1F (0x3F)				默认值: 0x00				
EECR	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
初始值	0	0	0	0	0	0	0	0

位定义

[7:4]	EEPM[3:0]	EFLASH/EPROM 访问模式控制位				
		[3]	[2]	[1]	[0]	模式说明
		0	0	0	x	8 位模式读/写 E2PROM（默认）
		0	0	1	x	16 位模式读/写 E2PROM
		0	1	0	x	32 位模式读/写 E2PROM
		0	x	x	0	在非 32 位模式下，设置 M16 模式 M16 模式的具体定义将在后面给出
		1	x	0	0	E2PROM 擦除(可选操作)
		1	x	0	1	程序 FLASH 擦除(页擦除)
		1	x	1	0	程序 FLASH 编程
[3]	EERIE	FLASH/E2PROM 就绪中断使能控制。写 1 使能，写 0 禁止。当 EEPE 被硬件自动清零后，E2PROM 就绪中断有效。在 EPROM 操作过程中，将不会产生这个中断				
[2]	EEMPE	FLASH/E2PROM 编程操作使能控制位 EEMPE 用于控制 EEPE 是否有效，当同时设置 EEMPE 为 1，EEPE 为 0 后，在之后的四个周期内，设置 EEPE 为 1 将启动编程操作。否则编程操作无效。四个周期后，EEMPE 被自动清零				
[1]	EEPE	FLASH/E2PROM 编程操作使能位				
[0]	EERE	E2PROM 读使能位				

FLASH/E2PROM 读写控制时序

EECR 寄存器控制了所有 FLASH 操作有关的实现。其中 EEPM 主要控制操作模式和选择操作类型。EEPM[3]主要选择是要操作数据 FLASH(E2PROM)还是程序 FLASH。当操作对象是程序 FLASH 时，数据接口固定为 32 位模式。当操作对象为数据 FLASH(E2PROM)时，可以选择不同的数据宽度。默认为 8 位模式，这种模式操作最为简单直观。

FLASH 控制器在实现 E2PROM 接口时，内部已经实现了在必要时自动擦除数据 FLASH 的逻辑，所以 EPROM 擦除命令是可选的，这个命令只在用户需要单独执行擦除时使用。EEMPE 控制 FLASH 的擦/写时序，包括程序 FLASH 和 E2PROM。都必须在 EEMPE 的控制下完成响应的操作。EEPE 在 EEMPE 有效时序期间内，能够启动所有擦除和编程操作。具体的操作类型由 EEPM[3:0]决定。

对 E2PROM 的读操作比较简单，在设置好目标地址和模式后，写 EERE 位即将目标地址对应的 32 位数据读入 FLASH 控制器内部，用户可以通过 EEDR 寄存器读取感兴趣的字节。FLASH 控制器并没有实现对程序 FLASH 空间的读操作，用户可以方便的使用 LPM 或

者通过程序 FLASH 在数据统一映射空间的地址处使用 LD/LDD/LDS 指令读取。

M16 模式

M16 模式仅在 FLASH 控制器工作在 8 位/16 位模式下有效。M16 模式是将 FLASH 访问的最小单位 32 位，分为高低各 16 位。使能 M16 模式后，FLASH 控制器将对高低 16 位编程操作看作独立的两个单元。当检测其中一个 16 位是否可以不用擦除时（检测数据是不是 0xFFFF），不会考虑另外 16 位的数据值。这样可以充分利用 FLASH 本身可以重复 2 次编程的特性，减少不必要的擦写周期，同时也能提高编程效率。
而在 32 位模式下， 因为每次的操作都为 FLASH 编程的最小单位，所以没必要分 M16 模式。

数据 FLASH/E2PROM 编程流程实例

1. 8 位模式，编程 E2PROM

- 检测 EEPE 位，等待 FLASH 控制器空闲
- 设置目标地址到 EEAR[8:0]
- 设置新的数据到 EEDR
- 设置 EEPM[3:1] = 000，EEPM[0]可设置为 0 或 1
- 设置 EEMPE = 1， 同时 EEPE = 0
- 在四个周期内，设置 EEPE = 1

当设置完成后，FLASH 控制器将启动编程操作，编程期间 CPU 将保持在当前的指令地址上，直到操作完成后才会继续运行。在编程过程中，如果需要擦除数据 FLASH，FLASH 控制器将会自动启动擦除流程。

2. 32 位模式，编程 E2PROM

- 检测 EEPE 位，等待 FLASH 控制器空闲
- 通过 EEAR[1:0]与 EEDR，设置 32 位数据，请参考 EEDR 寄存器定义部分
- 设置目标地址到 EEAR[8:0]。注意这里是字节对齐的地址，FLASH 控制器用 EEAR[8:2]作为访问 FLASH 的地址。
- 设置 EEPM[3:1] = 010，EEPM[0]可设置为 0 或 1
- 设置 EEMPE = 1， 同时 EEPE = 0
- 在四个周期内，设置 EEPE = 1

3. 8 位模式，读 E2PROM

- 检测 EEPE 位，等待 FLASH 控制器空闲
- 设置目标地址到 EEAR[8:0]
- 设置 EEPM[3:1] = 000
- 设置 EERE = 1 启动 E2PROM 读操作
- 等待 2 个周期(执行两个 NOP 操作)
- 目标地址对应的数据被更新到 EEDR 寄存器内

4. 32 位模式，读 E2PRM

- 检测 EEPE 位，等待 FLASH 控制器处于空闲状态
- 设置 EEAR[8:0]为目标地址，地址为 4 字节对齐
- 设置 EEPM[3:1] = 010，开启 32 位接口模式

- 设置 EERE = 1，启动 E2PROM 读操作
- 等待 2 个系统时钟周期(执行两个 NOP 指令)
- 目标地址对应的数据更新到控制器内部 32 位寄存器中，用户可以使用 EEAR[1:0]和 EEDR 读取指定字节的数据或全部数据。

5. 程序 FLASH 擦除操作

- 检测 EEPE 位，等待 FLASH 控制器空闲
- 设置 EEAR[12:0]为需要擦除的目标页地址，程序 FLASH 一页大小为 512 字节，因此 EEAR[12:9]将作为页地址，EEAR[8:0]设置为 0
- 设置 EEPM[3:0] = 1X01，其中 EEPM[2]可设置为 0 或 1
- 设置 EEMPE = 1， 同时 EEPE = 0
- 在四个周期内，设置 EEPE = 1，启动程序 FLASH 擦除流程

6. 程序 FLASH 编程操作

- 检测 EEPE 位，等待 FLASH 控制器空闲
- 通过 EEAR[1:0]与 EEDR，设置 32 位编程数据
- 设置 EEAR[12:2]为目标地址，此处地址为 4 字节对齐
- 设置 EEPM[3:0] = 1X10， 其中 EEPM[2]可设置为 0 或 1
- 设置 EEMPE = 1，同时 EEPE = 0
- 在四个周期内，设置 EEPE = 1，启动 FLASH 编程流程

系统配置信息(熔丝)读/写

数据 FLASH 的最高地址部分的 8 个字节，用于存储系统配置信息以及 FLASH 控制器实现 E2PROM 接口所需的标记位。其中只有配置信息这部分可以被用户访问，

下面是配置信息定义：

位置	功能描述
Fuse[5:0]	内部 32MHz RC 振荡器校准位
Fuse[7:6]	保留不用
Fuse[10:8]	配置信息使能位，000 使能配置信息
Fuse[13:11]	低压复位电路配置信息 Fuse[11]：为 1 使能低压复位功能 Fuse[13:12] = 00 复位阀值为 1.8V Fuse[13:12] = 01 复位阀值为 2.7V Fuse[13:12] = 10 复位阀值为 4.3V Fuse[13:12] = 11 保留不用
Fuse[15:14]	保留不用
Fuse[31:16]	保留不用

对这部分数据的操作和操作其他 E2PROM 区域一样。设置 EEAR[8:2] = 0x7F 寻址系统配置信息。配置信息中，保留不用的位都需要设置为 1。具体操作时序请参考 FLASH/E2PROM 控制器 32 位模式编程流程实例。

通用 I/O 寄存器- GPIOR2

GPIOR2 – 通用 I/O 寄存器 2		
GPIOR2: 0x2B (0x4B)		默认值: 0x00
GPIOR2	GPIOR2[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[7:0]	GPIOR2	通用 I/O 寄存器 2，用于存储用户自定义数据

通用 I/O 寄存器- GPIOR1

GPIOR1 – 通用 I/O 寄存器 1		
GPIOR1: 0x2A (0x4A)		默认值: 0x00
GPIOR1	GPIOR1[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[7:0]	GPIOR1	通用 I/O 寄存器 1，用于存储用户自定义数据

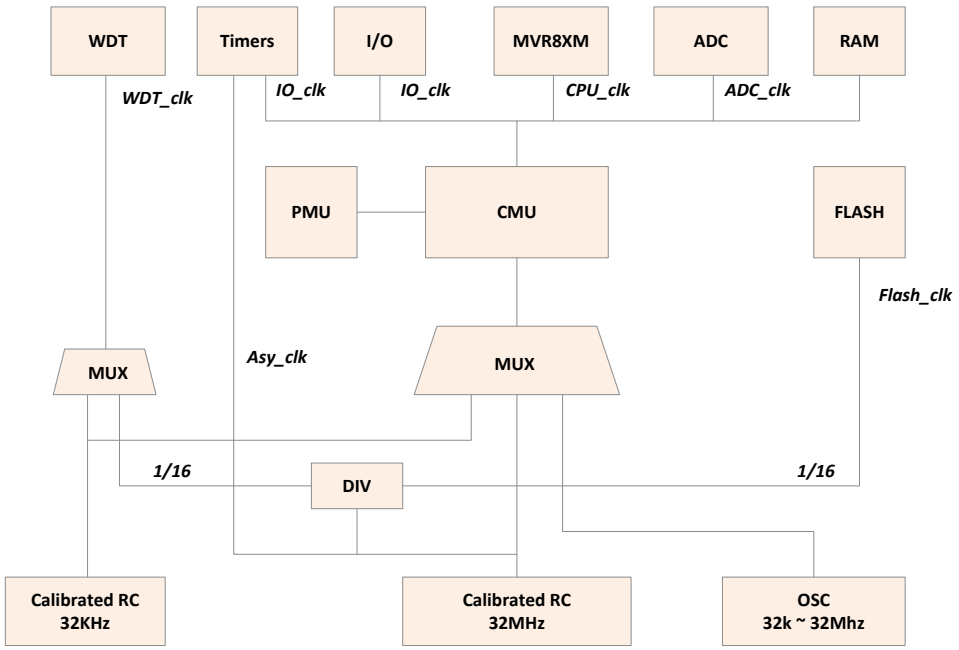
通用 I/O 寄存器- GPIOR0

GPIOR0 – 通用 I/O 寄存器 0		
GPIOR0: 0x1E (0x3E)		默认值: 0x00
GPIOR0	GPIOR0[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[7:0]	GPIOR0	通用 I/O 寄存器 0，用于存储用户自定义数据

系统时钟与配置

系统时钟分布

LGT8F88A 支持多种时钟输入。系统可以工作在三种主要的时钟源，分别是内部 32KHz 可校准 RC 振荡器，内部 32MHz 可校准 RC 振荡器以及外部 32KHz ~ 32MHz 晶振输入。
下图为 LGT8F88A 时钟系统分布，CMU 是整个时钟管理的中心，负责系统时钟的分频，为不同的模块产生独立的时钟以及对时钟进行控制等等。一般的应用中，并不不要全部的时钟同时工作，为了减小系统功耗，系统功耗管理根据不同的休眠模式，关闭没有使用的模块时钟。具体操作细节，请参考功耗管理相关章节。



CPU_clk

CPU 时钟用于驱动 LGT8XM 内核以及 SRAM 的运行。比如驱动通用工作寄存器，状态寄存器等。CPU 时钟停止后，内核将不会继续执行指令和进行计算。

IO_clk

IO 时钟用于驱动大部分外设模块，比如定时/计数器，SPI，USART 等。IO 时钟也用于驱动外部中断模块。当 IO 时钟因休眠而停止后，某些可以用了唤醒系统的外设部分工作在独立的时钟或异步模式。比如 TWI 的地址识别功能可以唤醒大部分休眠模式，此时的地址识别部分工作在异步模式。

Flash_clk

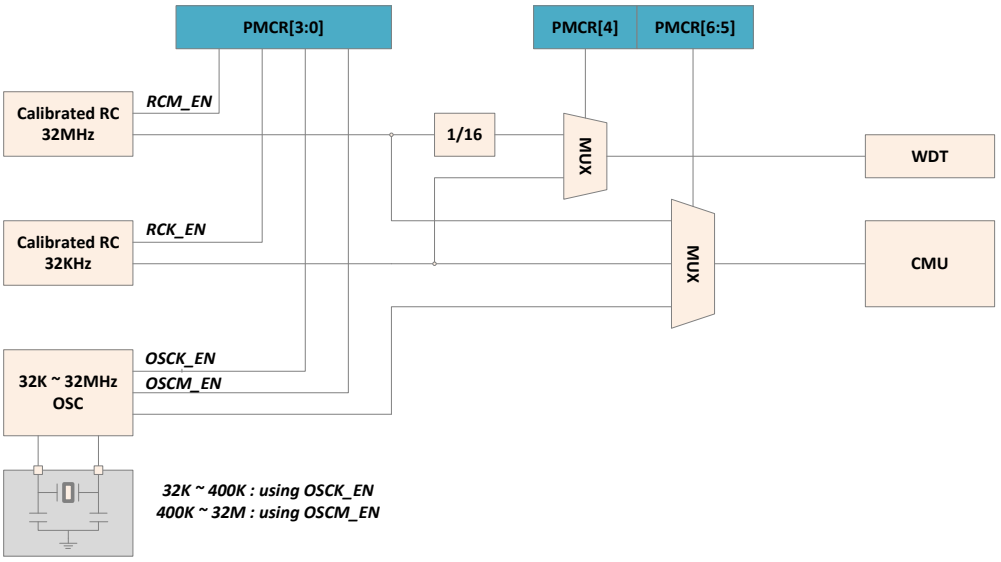
FLASH 时钟用于产生 FLASH 接口访问时序。为避免时钟不确定造成时序误差，FLASH 时钟使用 32MHz 可校准 RC 的 16 分频作为源。FLASH 时钟主要用于程序通过 FLASH 控制器对程序 FLASH 以及数据 FLASH 的访问。

Asy_clk

异步定时器时钟。定时/计数器可以直接使用外部时钟或晶振(32.768K)驱动。这种独立的时钟模式，可以在系统处理休眠模式时，定时器仍然保持运行。

时钟源选择

LGT8F88A 支持 4 种时钟源输入，用户可以通过 PMCR 寄存器实现对时钟源的使能控制以及完成主时钟的切换。下面是 PMCR 的控制结构图：



LGT8F88A 内部 OSC 振荡器可以工作在高频和低频两种模式下，用户需要根据外接晶振的实际大小控制内部 OSC 振荡器工作在正确的模式下。同样内部的 RC 振荡器也分为高频和低频两种。PMCR 寄存器的最低 4 位用于控制这四种时钟源。控制关系如下：

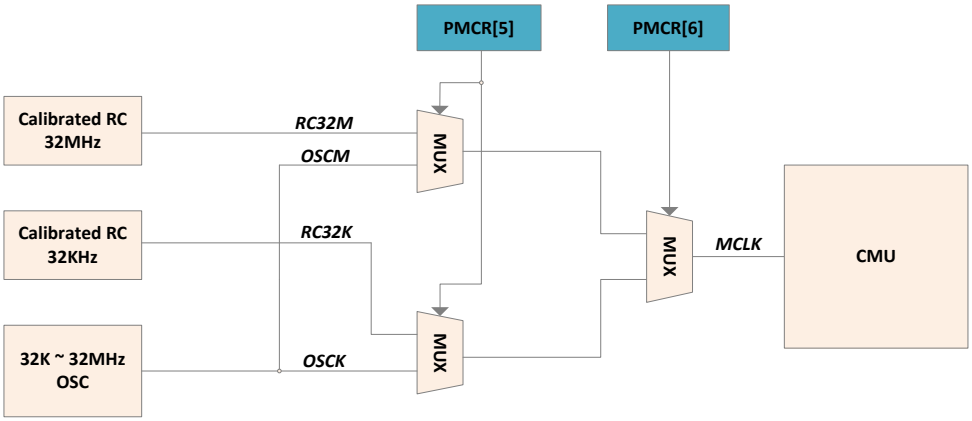
PMCR	对应时钟源
PMCR[0]	32MHz RC 使能控制，1 使能，0 关闭
PMCR[1]	32KHz RC 使能控制，1 使能，0 关闭
PMCR[2]	400K ~ 32MHz OSC 模式使能，1 使能，0 关闭
PMCR[3]	32K ~ 400K OSC 模式使能， 1 使能，0 关闭

LGT8F88A 系统上电后，默认使用 32MHz RC 作为系统时钟源，内核工作在时钟源的 8 分频(4MHz)。用户可以通过设置 PMCR 寄存器以及系统预分频寄存器(CLKPR)改变默认配置。

如果用户需要更改主时钟源配置，需要在切换时钟前保证切换后的时钟源处于稳定的工作状态。因此需要在切换主时钟源之前，通过 PMCR[3:0]使能所需时钟源，并等待到时钟稳定后才能进行切换。

当用户切换主时钟到外部晶振时，虽然用户使能了外部晶振，但也不排除因配置错误或晶振失效导致晶振无法起振。如果在此时切换到外部晶振，切换后系统将立即停止工作。因此，从系统可靠性考虑，建议通过打开看门狗定时器的方式，从软件设计的角度避免此类问题。

时钟源使能并等待稳定后，可以通过 PMCR[6:5]切换主时钟。其中 PMCR[5]用于选择是内部 RC 振荡器和外部晶振，PMCR[6]用于选择高速时钟源和低速时钟源。



主时钟源选择：

PMCR[6]	PMCR[5]	主时钟源
0	0	内部 32MHz RC 振荡器(系统默认)
0	1	外部 400K ~ 32MHz 高速晶振
1	0	内部 32KHz RC 振荡器
1	1	外部 32K ~ 400KHz 低速晶振

时钟源控制时序

为保护 PMCR 寄存器被意外修改，对 PMCR 寄存器的修改需要严格安装指定的时序进行。PMCR 寄存器的最高位(PMCR[7])用于实现时序控制。用户在修改 PMCR 其他位之前，必须首先要将 PMCR[7]置 1，在置 1 操作后的 6 个周期内，更改 PMCR 其他寄存器的值。6 个周期之后，对 PMCR 的直接修改将失效。

下面以切换到外部高速晶振为例，列出建议的操作步骤：

(1) 使能时钟源

- 设置 PMCR[7] = 1
- 在六个周期内，设置 PMCR[3] = 1，使能外部高速模式外部晶振
- 等待外部晶振稳定(等待时间因晶振不同而不同，一般 us 级等待即可)

(2) 切换主时钟源

- 设置 PMCR[7] = 1
- 在六个周期内，设置 PMCR[6:5] = 01，系统将工作时钟自动切换至外部晶振
- 执行几个 NOP 操作，提高稳定性(可选操作)

[注意]：在以上切换主时钟的操作中，要保证当前系统时钟正常工作，在切换到外部晶振以后，才可以关闭之前的内部 RC 振荡器。

系统时钟预分频控制

LGT8F88A 内部有一个系统时钟预分频器，可以通过时钟预分频寄存器(CLKPR)进行控制。这种功能可以用于当系统不需要非常高的处理能力时，减小系统功耗。预分频设置对系统支持的时钟源都有效。时钟预分频能够影响到内核执行时钟以及所以同步外设。

当在不同的时钟预分频设置之间切换时，系统时钟预分频确保在切换过程中不会产生毛刺，而已会保证不会有过高频的中间状态。分频切换是立即执行的，当寄存器改变生效后，最多在 2~3 个当前系统时钟周期后，系统时钟就切换到了新的分频时钟。

为了避免对时钟分频寄存器的误操作，对 CLKPR 的修改也必须遵循一个特殊的时序流程：

- 设置时钟预分频更改使能位(CLKPCE)为 1，CLKPR 其他所以位为 0
- 在四个周期内，把需要的值写入 CLKPS，同时 CLKPCE 写 0

在更改时钟预分频寄存器前， 需要禁止中断功能，以保证写时序能够完整的进行。
关于主时钟预分频寄存器 CLKPR 的具体定义，请参考本章节寄存器描述部分。

内部 RC 振荡器校准

LGT8F88A 内部包含两个可校准 RC 振荡器，经过校准后，均可达到±1%以内的精度。其中 32MHz RC 默认用于系统工作时钟。

每一颗 LGT8F88A 出产前，都对内部 32MHz 的 RC 进行了校准，并把校准值写入系统配置信息区域。用户客户通过 I/O 寄存器空间的 RCCAL 寄存器访问这个校准值。

内部 32KHz 的 RC 在出产前并没有经过校准，如果用户需要一个非常精确的低频时钟，可以自行校准这个 RC 振荡器，并把校准值写入到 LGT8F88A 自带的数据 FLASH 中。在每次芯片启动后，由软件将这个校准值读出，写入到 RCKCAL 寄存器中，完成对 32KHz RC 振荡器的校准。具体校准方法请参考下面的介绍。

内部 32KHz RC 校准方法：

在校准内部 32KHz RC 前，需要能够测量到当前内部 32KHz 的时钟频率。比较简单的方法是把系统工作时钟切换到内部 32KHz RC 振荡器。然后通过 I/O 输出一个比较易于测量的方波。外部通过测量方波频率的方式，得到此时内部 RC 的频率。为了减小测量误差，建议采用多次测量求均值的方式。

具体校准流程如下：

- 首先设置内部 **RCKCAL** = 000000，这次测量初始化状态
- 通过输出的方波测量内部 RC 的频率，如果频率满足要求，即完成校准
- 否则，通过如下方式计算调整值：

  假设 **Fm** 为测量值，**Ft** 为目标值

  如果 **Fm > Ft**:

  F_{LSB} = 1% X Fm

  Y = (Fm – Ft) / F_{LSB}

  Z = 64 – ROUND(Y)

  RCKCAL = BINARY(Z), 即 Z 的二进制

如果 **Fm < Ft**:

F_{LSB} = 1% X Fm

Y = (Ft – Fm) / F_{LSB}

Z = 64 – ROUND(Y)

RCKCAL = BINARY(Z), 即 Z 的二进制

以上即为 RC 的校准方法，32MHz RC 振荡器的校准方法与此相同。得到 RCKCAL 后，可以将其值写入到数据 FLASH(E2PROM)一个保留的区域，以供后续软件校准使用。

寄存器定义

32MHz RC 振荡器校准寄存器- RCCAL

RCCAL – 32MHz RC 校准寄存器		
RCCAL: 0x66	默认值: 0x00	
RCCAL	RCCAL[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[5:0]	RCCAL	6bit RC 校准值，系统上电后，寄存器的值将被系统配置信息中的 RC 校准值替换。
[7:6]	-	保留不用

时钟源管理寄存器- PMCR

PMCR – 时钟源管理寄存器							
PMCR: 0xF2			默认值: 0x01				
PMCR	PMCE	-	WCLKS	OSCK_EN	OSCM_EN	RC32K_EN	RC32M_EN
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
初始值	0	-	0	0	0	0	1
位定义							

[0]	RC32M_EN	内部 32MHz RC 振荡器使能控制，1 使能，0 禁止
[1]	RC32K_EN	内部 32KHz RC 振荡器使能控制，1 使能，0 禁止
[2]	OSCM_EN	外部高频晶振使能控制，1 使能，0 禁止
[3]	OSCK_EN	外部低频晶振使能控制，1 使能，0 禁止
[4]	WCLKS	WDT 时钟源选择，请参考 WDT 相关章节
[5]	CLKSS	主时钟源选择控制，选择时钟源类型，请参考时钟源选择部分
[6]	CLKFS	主时钟源频率控制，选择时钟频率类型，请参考时钟源选择部分
[7]	PMCE	PMCR 寄存器更改使能控制位。 在更改 PMCR 其他位置之前，必须首先设置此位，然后在四个周期内设置其他位的值。

主时钟预分频寄存器- CLKPR

CLKPR – 主时钟预分频寄存器							
CLKPR: 0x61				默认值: 0x03			
CLKPR	CLKPCE	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0
R/W	R/W	-	-	R/W	R/W	R/W	R/W
初始值	0	-	-	0	0	1	1
位定义							
[3:0]	CLKPS	时钟预分频选择位					
		CLKPS0	CLKPS1	CLKPS2	CLKPS3	分频参数	
		0	0	0	0	1	
		0	0	0	1	2	
		0	0	1	0	4	
		0	***0***	***1***	***1***	***8(默认配置)***	
		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	
		其他值				保留	
[6:4]	-	保留不用					
[7]	CLKPCE	时钟预分频更改时钟控制 在改变 CLKPR 寄存器的其他位之前，必须首先单独设置 CLKPCE 为 1，然后在之后的四个系统周期内，对其他位进行设置。四个周期结束后，CLKPCE 自动清零。					

32KHz RC 振荡器校准寄存器- RCKCAL

RCKCAL – 32MHz RC 校准寄存器		
RCKCAL: 0x67	默认值: 0x00	
RCKCAL	RCKCAL[7:0]	
R/W	R/W	
初始值	0x00	
位定义		
[5:0]	RCKCAL	6bit RC 校准值，用于将校准值写入 RCKCAL 寄存器完成对 32KHz RC 振荡器的校准。具体校准方法请参考本章中 RC 校准部分。
[7:6]	-	保留不用

功耗管理

概述

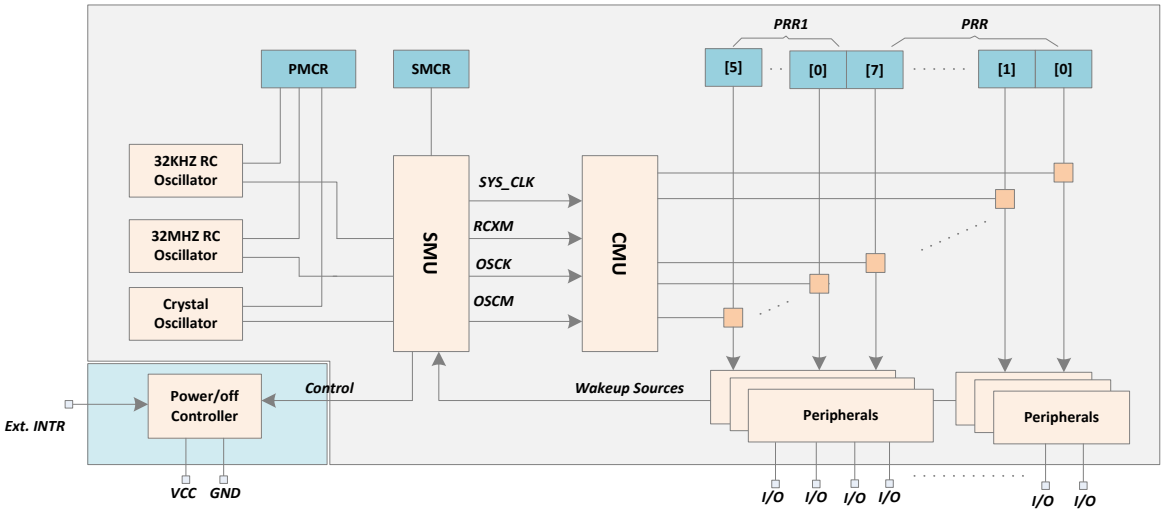
休眠模式用于关闭 MCU 中没有被使用的模块，从而减小系统功耗。LGT8F88A 提供了非常灵活多样的休眠模式和模块控制器，用户可以根据应用，实现最理想的低功耗配置。

当使能低电压检测(LVD)模块后，LVD 在休眠模式下仍然会继续工作。为了进一步降低系统功耗，可以在进入休眠模式前将其关闭。

LGT8F88A 支持掉电模式，掉电模式下，系统的数字部分，大部分 I/O 以及模拟模块都将处于掉电状态。掉电模式能够最小系统功耗。掉电模式只能通过指定的外部引脚唤醒，唤醒过程与系统上电过程一致。软件可以通过读取 MCU 状态寄存器得到系统之前的状态。

LGT8F88A 内部包含一个精度可校准的 32KHz RC 振荡器，用户可以在不需要处理复杂的任务时，将系统时钟切换至 32KHz RC，使用合适的休眠模式。这样也可以在非掉电模式下获得理想的系统功耗。如果用户外接了 32768Hz 的晶振，也可以将主时钟切换到外部晶振，然后关闭掉没有使用的时钟源以及其他模拟模块，然后进入休眠模式，这样可以进一步节省功耗。

系统功耗管理示意图：



如上图所示，LGT8F88A 主要通过休眠模式控制器(SMU)以及时钟管理单元(CMU)控制整个系统的功耗。从节省功耗的级别上，我们可以把功耗分为 3 个等级，第一级是通过 PRR 寄存器控制模块工作时钟，通过关闭没有使用模块的时钟，节省系统运行的动态功耗。一般情况下，这种级别能够节省的功耗并不明显。第二级是通过切换主时钟源到低频时钟上，并关闭没有使用的时钟源模块以及其他模拟模块，这种模式基本上可以得到非常可观的系统运行功耗和休眠功耗。第三级别是通过让系统进入到断电(Power/off)模式，断电模式下 LGT8F88A 运行功耗最小，此模式只能通过外部中断引脚唤醒。从断电模式唤醒后，系统重新执行一个上电复位过程，软件可以通过 MCUSR 寄存器读取复位前的状态。

休眠模式

LGT8F88A 支持 4 种休眠模式，用户可以根据应用需求选择合适的休眠模式。SMCR 寄存器包含了休眠模式的控制设置，执行 SLEEP 指令后，内核进入休眠模式。为获得更加理想的休眠功耗，建议在内核进入休眠模式前，关闭所有没有使用的时钟以及模拟模块。但需要注意的是，某些唤醒源的产生需要工作时钟，如果需要使用这类唤醒源，请保持相关时钟源的工作状态。

休眠模式与唤醒方式：

	有效时钟				唤醒源							
Sleep Mode	CPU_CLK FLASH_CLK	IO_CLK	ADC_CLK	ASY_CLK	PIN Change	INT0 INT1	TWI Address Match	Timer2	ADC	WDT	Other INT	Other I/O
Idle		X	X	X	X	X	X	X	X	X	X	X
ADC Noise Reduction			X	X	X	X	X	X	X	X		
Power/Down					X	X	X			X		
Power/Off						X						

如果需要进入以上 4 种休眠模式，SMCR 中的 SE 位必须置 1，使能休眠模式控制。然后执行一条 SLEEP 指令即可。SMCR 中的 SM0/1/2 用于选择不同的休眠模式。具体的信息请参考下面的描述。

在 MCU 处于休眠模式下，如果唤醒源有效，MCU 将会在 4 个周期后被唤醒，继续执行指令。如果中断保持有效，中断也将立即响应，进入中断服务子程序。如果在 SLEEP 模式下发生了系统复位，MCU 也将会被唤醒，并从复位向量开始执行。

当 MCU 处于 Power/Off 模式下，系统只能够通过外部中断 INT0/1 唤醒，系统将重新执行一个上电复位过程，MCU 将从复位向量地址开始执行。

IDLE 模式

当 SM2…0 设置为 000，执行 SLEEP 指令后，MCU 进入到 IDLE 模式，IDLE 模式将会关闭掉内核工作时钟，除此之外的其他外设都能正常工作。

IDLE 模式可以通过外部中断以及内部中断等唤醒。如果不需要使用比较器以及 ADC 作为唤醒源，建议将其关闭。

IDLE 模式因为仅仅关闭了内核运行的时钟，所以并不能得到明显的功耗降低。IDLE 模式下，内核也将停止执行和取指令，因此可以降低内部程序 FLASH 的运行功耗。

但 IDLE 模式拥有比较灵活的唤醒方式，用户可以通过降低系统主时钟以及关闭不需要的模块获取更加理想的运行功耗。

ADC 噪声抑制模式

当 SM2…0 设置为 001，执行 SLEEP 指令后，MCU 进入 ADC 噪声抑制模式。此模式下，内核以及大部分外设都将停止工作，ADC，外部中断，TWI 地址匹配，WDT 以及工作在异步时钟模式下的定时/计数器 2 都可以正常工作。

ADC 噪声一直模式主要用于为 ADC 转化提供一个良好的工作环境。降低数字模块对模拟转换的高频干扰。进入这个模式后，ADC 将自动启动采样转换，转换的数据保存到 ADC 数据寄存器后，ADC 转换结束中断将 MCU 从 ADC 噪声模式下唤醒。

Power/Down 模式

当 SM2…0 设置为 010，执行 SLEEP 指令后，MCU 进入到 Power/down 模式。这种模式下，系统将关闭掉所有模块的工作时钟。此模式因为关闭了所有模块的工作时钟，因此只能通过异步模式唤醒，外部中断，TWI 地址匹配以及工作在独立时钟源模式下的 WDT 都可以产生此模式下的唤醒信号。

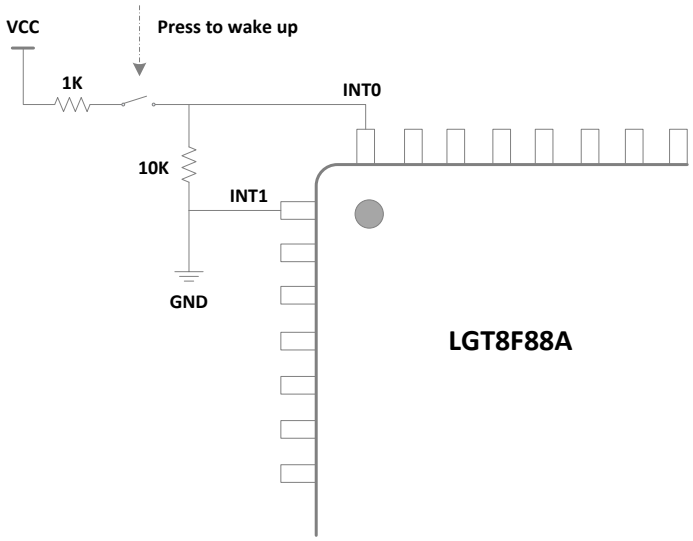
此种模式可以关闭除主时钟源以为的所有模块。为实现更加理想的运行功耗，建议在进入此中模式前，将系统主时钟切换到内部 32K RC 或者外部 32KHz 低频晶振，然后关闭掉所以没有被使用的时钟源以及模拟模块。

Power/Off 模式

当 SM…0 设置为 011，执行 SLEEP 指令后，MCU 将进入到 Power/Off 模式。进入 Power/Off 后，系统中大部分电路将进入掉电状态。只有负责唤醒的一部分逻辑和 I/O 可以正常工作。此种模式只能通过外部中断 INT0/1 唤醒，外部复位信号也将没有作用。

Power/Off 模式只支持 INT0/1 的电平唤醒。进入 Power/Off 模式后，在 INT0/1 上给一个持续长度的高电平，可以唤醒系统中处于掉电的模块，系统将重新开始一个上电复位过程。软件需要在初始化阶段通过设置 SM2…0 为 100 锁定 Power/Off 控制器后，外部的唤醒信号才可以释放，否则，将无法完成唤醒工作。INT0/1 上的唤醒信号最小需要保持 2 毫秒。

下面是 Power/Off 唤醒电路的一般设计：



上图中，我们使用 INT0 作为唤醒源，INT1 不用，但需要接地保持为无效状态。

LGT8F88A 的 INT0/1 引脚内部有专用的去抖电路(de-bounce)，可以滤除掉按键过程中引入的干扰，保存内部唤醒电路正常无误的工作。

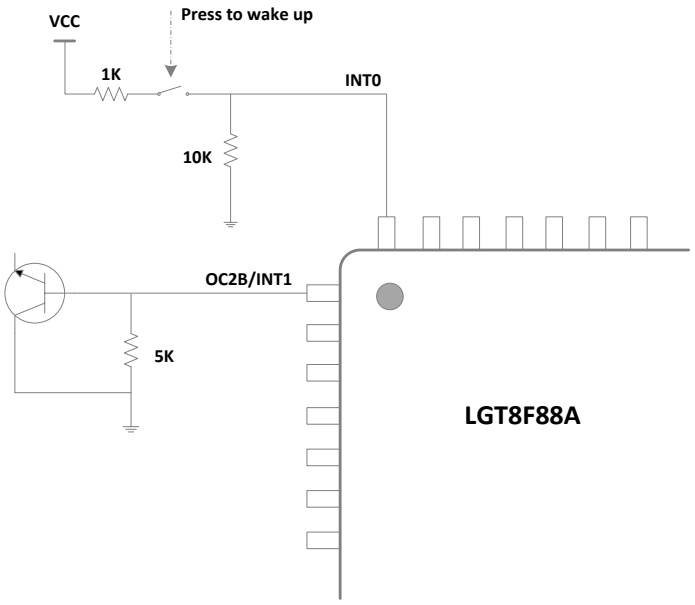
MCU 唤醒后，需要立即通过 SMCR 寄存器，设置 SM2…0 为 100，锁住此时的唤醒状态。之后可以释放掉外部的按键状态。MCU 进入工作状态，直到一下次执行 SLEEP。

INT0/1 被设计为可用于将 MCU 从 Power/off 状态下唤醒。如果应用中不需要使用

Power/off 模式，INT0/1 所在的引脚可以作为一般功能使用，不会影响到 MCU 的运行状态。如果系统需要 Power/Off 模式以降低功耗，但只用到 INT0/1 其中一个引脚作为唤醒源，另外一个仍然可以作为正常的 I/O 使用。但需要注意以下使用方法，下面我们假设使用 INT0 作为唤醒源，INT1 可以通过一下几种方式使用：

1. 如果没有使用 INT1 引脚的任何功能，需要将其直接接地。这样既不会影响到 Power/Off 模式的工作，也可以增加系统的抗干扰能力。
2. 如果使用到 INT1 引脚的外部中断功能，建议使用高电平中断模式或上升沿触发模式。
3. 如果使用 INT1 引脚作为 GPIO 或 OC2B 功能使用，需要加一个下拉，这样就不会影响系统的 Power/Off 控制。

下图为一个典型的使用实例：



上图中，INT0 用于 Power/Off 模式唤醒。INT1 用于驱动一个三极管。系统上电后，在正常工作下，INT0/1 可作为一般的功能使用。但在进入 Power/Off 模式之前， 需要保证 INT1 引脚设置为输入模式或输出驱动为低电平。进入 Power/Off 之后，用户按下与 INT0 相连的唤醒开关，即可唤醒系统。

寄存器描述

休眠模式控制寄存器- SMCR

SMCR – 休眠模式控制寄存器								
SMCR: 0x33(0x53)					默认值: 0x00			
SMCR				SM2	SM1	SM0	SE	
R/W	-			R/W	R/W	R/W	R/W	
初始值	-			0	0	0	0	
位定义								
[0]	SE	休眠模式使能控制位，设置为 1 后，执行 SLEEP 指令，内核将进入休眠啥模式。SE 位可以保护系统意外进入休眠模式。建议用户在设置此位为 1 后的，紧接着执行 SLEEP 指令。唤醒后，建议立刻清除 SE 位。						
[3:1]	SM	休眠模式选择						
		SM2	SM1	SM0	模式说明			
		0	0	0	IDLE 模式			
		0	0	1	ADC 噪声抑制模式			
		0	1	0	Power/Down 模式			
		0	1	1	Power/Off 模式			
		1	0	0	Power/Off Lock			
		Others			保留不用			
[7:4]	-	保留不用						

省电控制寄存器- PRR

PRR – 省电控制寄存器								
PRR: 0x64					默认值: 0x00			
PRR	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	PRUSART0	PRADC
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
初始值	0	0	0	-	0	0	0	0
位定义								
[0]	PRADC	设置为 1，关闭 ADC 控制器时钟						
[1]	PRUSART0	设置为 1，关闭 USART0 模块的时钟						
[2]	PRSPI	设置为 1，关闭 SPI 模块的时钟						
[3]	PRTIM1	设置为 1，关闭定时/计数器 1 的时钟						
-	-	保留不用						
[5]	PRTIM0	设置为 1，关闭定时/计数器 0 的时钟						
[6]	PRTIM2	设置为 1，关闭定时/计数器 2 的时钟						
[7]	PRTWI	设置为 1，关闭 TWI 模块的时钟						

省电控制寄存器- PRR1

PRR1 – 省电控制寄存器 1								
PRR1: 0x65					默认值: 0x00			
PRR1			PRWDT	-	-	PREFL	PRPCI	-
R/W			R/W	-	-	R/W	R/W	-
初始值			0	-	-	0	0	-

位定义		
[0]	-	保留不用
[1]	PRPCI	设置为 1，关闭外部引脚变化以及外部中断模块工作时钟
[2]	PREFL	设置为 1，关闭 FLASH 控制器接口时序时钟
[4:3]	-	保留不用
[5]	PRWDT	设置为 1，关闭 WDT 计数器时钟
[7:6]	-	保留不用

系统控制与复位

概述

系统复位以后，所有的 I/O 寄存器都会被设置为它们的初始值，程序从复位向量处开始执行。LGT8F88A 的中断向量地址上， 必须用一个 RJMP – 相对跳转指令跳转到复位处理程序。如果程序没用使用到中断，没有使能中断源，中断向量也就不会被使用，中断向量区域就可以用来存放用户的程序代码。

复位有效后，所有 I/O 端口立即进入它们的初始状态。大部分 I/O 的初始化状态为输入并关闭掉内部上拉电阻。有模拟输入功能的 I/O，也初始化为数字 I/O 功能。

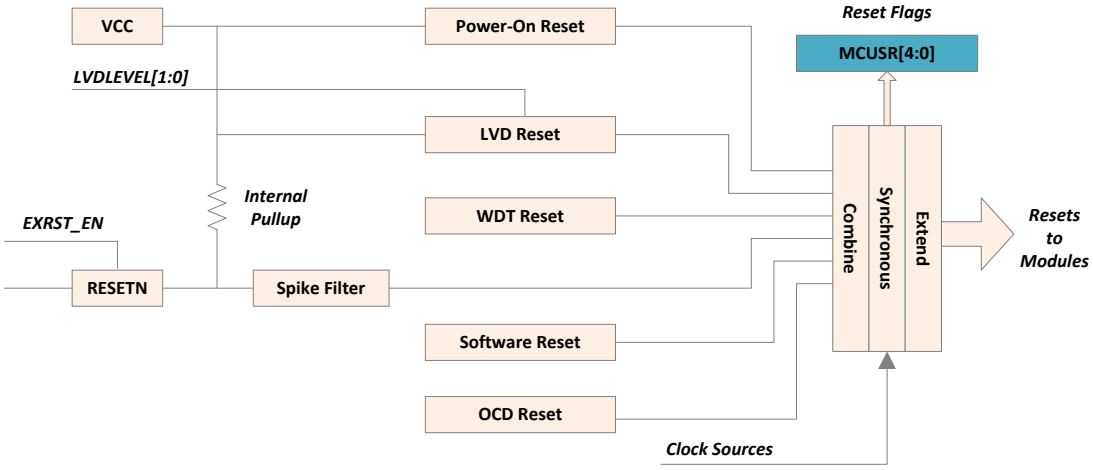
当复位变为无效后，LGT8F88A 内部的定时计数器开始启动，用于展宽复位。展宽复位信号的宽度用于保证系统中的电源以及时钟等模块进入到稳定的状态。

复位源

LGT8F88A 共支持六种复位源：

- 上电复位：当系统的工作电压低压内部 POR 模块的复位阀值时，上电复位有效。
- 外部复位：当在芯片的 RESETN 引脚上出现一个大于最小复位宽度的脉冲后，外部复位有效。
- 看门狗复位：使能看门狗模块后，如果看门狗定时器超时，系统将会复位。
- 低电压复位：LGT8F88A 内部有一个低电压检测模块(LVD)，当系统工作电源低于 LVD 设定的复位阀值时，MCU 也将会被复位。
- 软件复位: LGT8F88A 内部有一个专用的软件触发的复位寄存器，用户可以通过这个寄存器随时复位 MCU。
- OCD 复位：OCD 复位是有调试器模块发出的，用于直接复位 MCU 内核。

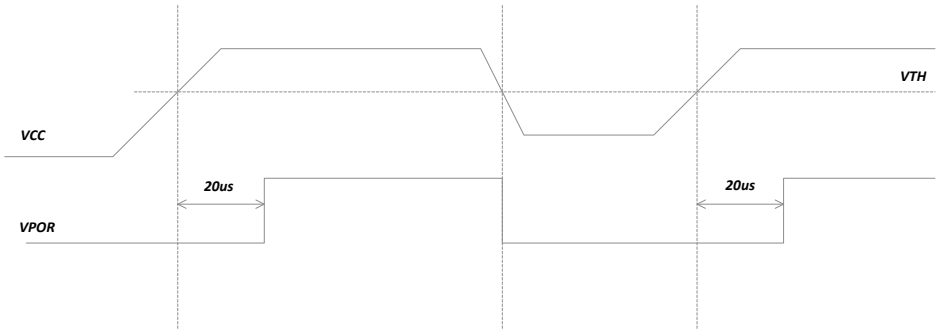
复位系统结构图：



上电复位

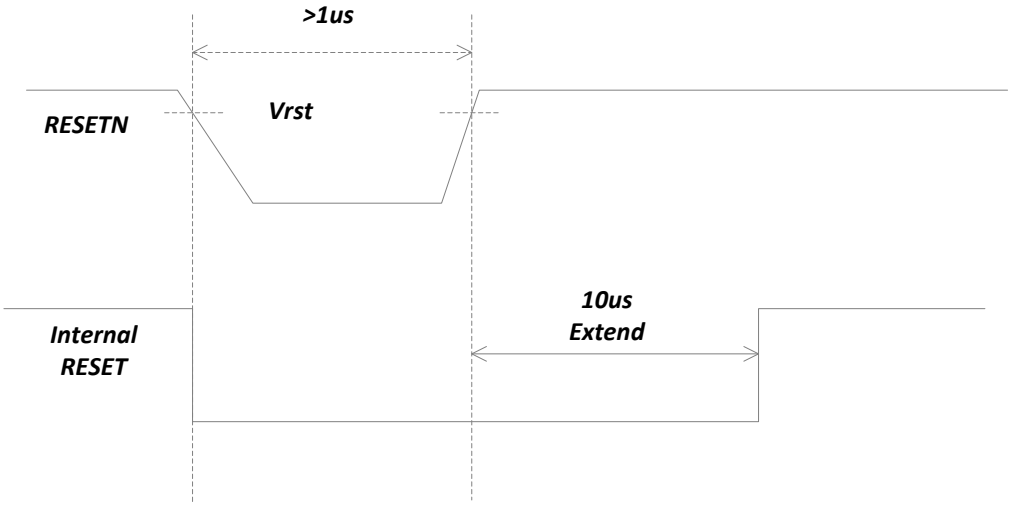
上电复位信号由内部的电压检测电路产生。当系统电源(VCC)低于检测阀值时，上电复位信号有效。上电复位的检测阀值，请参考电气参数部分。

上电复位电路能够保证芯片在上电过程中处于复位状态，芯片上电后能够从一个已知的稳定的状态开始运行。上电复位信号也会被芯片内部的计数器展宽，以保证上电后内部的各种模拟模块，比如 RC 振荡器等能够进入稳定的工作状态。



外部复位

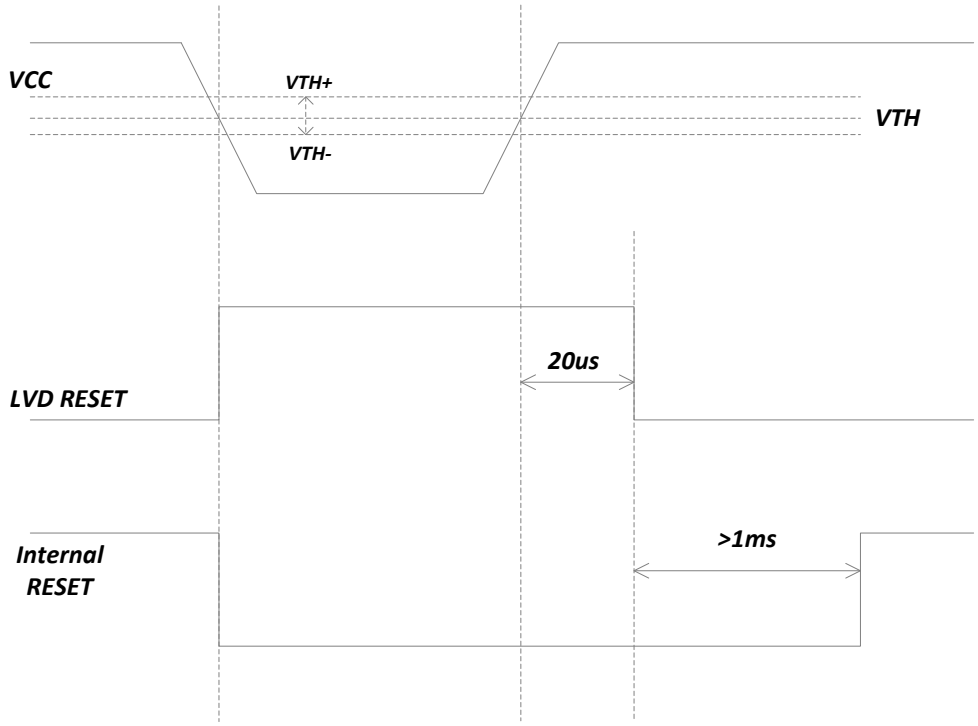
在外部复位引脚(RSTN)上施加一个低电平，外部复位立即有效。低电平的宽度要大于一个最小复位脉冲宽度要求。外部复位为异步复位，即使芯片没有时钟工作，外部复位仍然能够对芯片进行复位。LGT8F88A 的外部复位引脚同时也可以作为通用 I/O 使用。在芯片上电以后， 默认作为外部复位功能。用户可以通过寄存器配置，关闭该引脚的外部复位功能，从而可以当作普通的 I/O 使用。具体使用请参考 IOCR 寄存器的描述部分。



低电压检测(LVD)复位

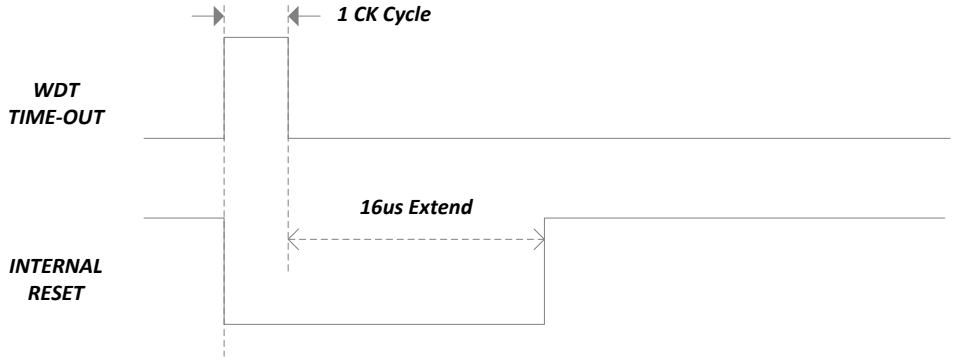
LGT8F88A 内部包含一个可编程低电压检测(LVD)电路。LVD 同样是检测 VCC 的电压变化，但与上电复位不同的是，LVD 可以选择检测电压的阀值。用户可以通过系统配置位或者直接通过操作 VDTCR 寄存器在三种不同的电压阀值之间选择。LVD 的电压检测电路具有±10mV~±50mV 的迟滞特性， 用于滤除 VCC 电压的抖动。当 LVD 使能后，如果 VCC 的电压下降到设定的复位阀值，LVD 复位将立刻有效。当 VCC 增加到复位阀值以上后，内部的复位展开电

路启动，将复位继续展宽至少 1 毫秒。



看门狗复位

当看门狗定时器溢出时，如果使能了看门狗系统复位功能，将立刻产生一个周期的系统复位信号。看门狗复位信号通用也会被内部的延时计数器展宽。看门狗控制器的详细操作，请参考下面的详细介绍部分。



软件复位、OCD 复位

软件复位是用户通过操作 VDTCR 寄存器的第六位触发，软件复位的时序与看门狗复位完全相似。内部将复位信号展宽 16us。

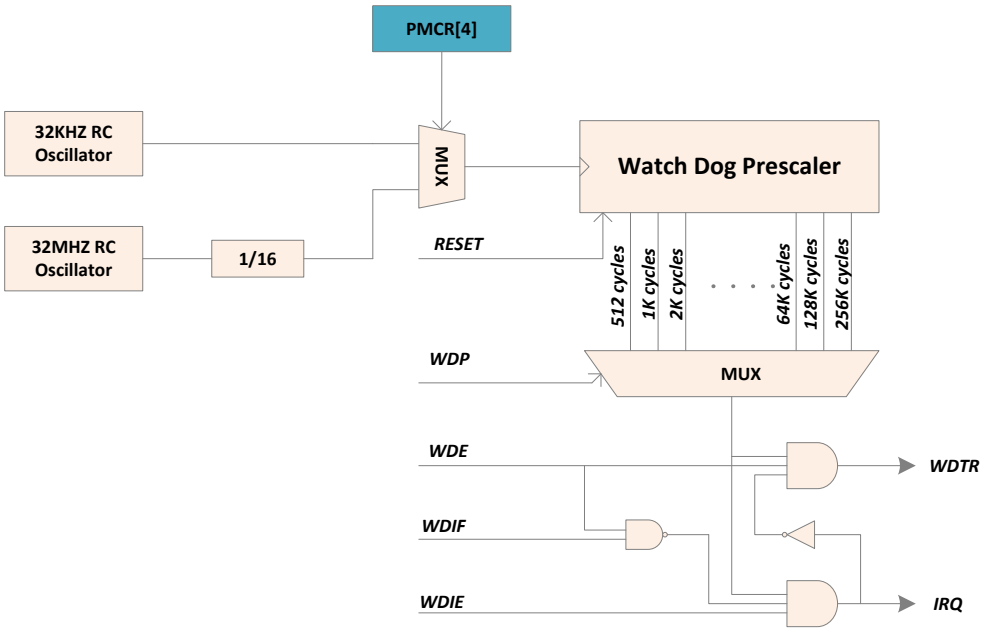
OCD 复位由芯片内部的调试器单元产生，OCD 复位一般是由调试器控制，用户软件无法触发 OCD 复位。

看门狗定时器

- 时钟可选内部 32KHz RC 或内部 32MHz RC 的 16 分频(2MHz)
- 支持中断模式，复位模式以及复位中断模式
- 定时器超时最大可到 8 秒

LGT8F88A 内部包含一个增强的看门狗定时器(WDT)模块。WDT 定时器的工作时钟可以是内部的 32KHz RC 振荡器，也可以是内部 32MHz RC 振荡器的 16 分频。WDT 计数器溢出后，可以输出一个中断或者一个系统复位信号。在正常使用时，需要软件执行一个 WDR –看门狗定时器复位指令在溢出之前重启计数器。如果系统没有即使的执行 WDR 指令，WDT 将会产生中断或系统复位。

看门狗定时器的结构图如下图所示：



在中断模式下，WDT 溢出后会产生一个中断请求信号。可以使用这个中断作为休眠模式的唤醒信号，也可以作为一个一般的系统定时器使用。比如可以使用这个中断限制某个操作的执行时间，在溢出中终止当前某一个任务。在系统复位模式下，WDT 在计数器溢出后立刻产生一个系统复位信号。最典型的用途就是用于防止系统死机或跑飞。第三种模式，就是复位中断模式，结合了中断和复位两种功能。首先系统将响应 WDT 中断功能，退出 WDT 中断复位程序后，立刻切换到复位模式。这个功能可以支持在复位之前保存一些比较关键的参数信息。

为了防止 WDT 被意外禁止，关闭 WDT 的操作必须按照一个严格定义的时序进行。下面的代码描述如何关闭看门狗定时器。下面的例子假设中断已经被禁止，这样整个操作流程就不会被中断。

汇编代码

```
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 语言代码

```
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();
}
```

[使用提示]

如果 WDT 被意外使能，比如程序跑飞，芯片会被复位，但是 WDT 仍然还是在使能状态。如果用户代码里没有处理 WDT，这将会导致循环复位。为避免这种情况，建议用户软件在初始化程序中清除看门狗复位标记位(WDRF)和 WDE 控制位。

下面的代码描述如何改变看门狗定时器的超时值。

汇编代码

```
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 语言代码

```
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();
}
```

[使用提示]

在改变 WDP 配置位之前，建议复位看门狗定时器。因为更改 WDP 位到比较小的超时周期很可能会导致看门狗超时复位。

寄存器定义

低压检测(LVD)控制寄存器- VDTCR

VDTCR – LVD 控制寄存器						
VDTCR: 0x62			默认值: 0x00 或加载自系统配置信息			
GPIOR2	CE	SWR	-	VDTS1	VDTS0	VDTEN
R/W	R/W	W	-	R/W	R/W	R/W
初始值	0x00 或加载自系统配置信息					
位定义						
[0]	VDTEN	低压检测模块使能控制，1 使能，0 禁止				
[2:1]	VDTS	低压检测阀值配置位 VDTS = 00 : 1.8V VDTS = 01 : 2.7V VDTS = 10 : 4.3V				
[5:3]	-	保留不用				
[6]	SWR	软复位使能位，写 1 产生软件复位				
[7]	CE	VDTCR 值改变使能位 用户在改变 VDTCR 寄存器的值之前，必须首先将此位写 1，在之后的 4 个时钟周期内，更改 VDTCR 其他位的值。四个周期后 CE 自动清零，对 VDTCR 寄存器的更新操作无效。				

IO 特殊功控制寄存器- IOCR

IOCR – IO 特殊功能控制寄存器				
IOCR: 0xF0		默认值: 0x00		
IOCR	CE	-	RVIO_EN	EXIO_EN
R/W	R/W	-	R/W	R/W
初始值	0x00 或加载自系统配置信息			
位定义				
[0]	EXIO_EN	PC6 引脚默认为复位功能，设置此位为 1 将禁止外部复位功能，复位功能禁止后，PC6 可作为一个普通的 I/O 使用		
[1]	RVIO_EN	VREF 引脚默认为模拟输入功能，设置此位为 1，将关闭模拟输入功能，这个引脚可以作为 PE6 使用		
[6:2]	-	保留不用		
[7]	CE	IOCR 值改变使能位 用户在改变 IOCR 寄存器的值之前，必须首先将此位写 1，在之后的 4 个时钟周期内，更改 IOCR 其他位的值。四个周期后 CE 自动清零，对 IOCR 寄存器的更新操作无效。		

MCU 状态寄存器- MCUSR

MCUSR – IO 特殊功能控制寄存器								
MCUSR: 0x34(0x54)				默认值: 0x00				
MCUSR	SWDD	PDRF	-	OCDRF	WDRF	BORF	EXTRF	PORF
R/W	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
初始值	0x00							

位定义		
[0]	PORF	上电复位标志，写 1 清零
[1]	EXTRF	外部复位标志，上电复位自动清零，或写 0 清零
[2]	BORF	低电压检测复位，上电复位自动清零，或写 0 清零
[3]	WDRF	看门狗复位标志，上电复位自动清零，或写 0 清零
[4]	OCDRF	OCD 调试器复位标志，上电复位自动清零，或写 0 清零
[5]	-	保留不用
[6]	PDRF	从 Power/off 模式唤醒标志，具体描述请参考功耗管理章节。
[7]	SWDD	SWD 接口禁止位。写 1 将关闭 SWD 接口。 SWD 接口关闭后，将无法进行调试和 ISP 操作。如果用户程序中关闭了 SWD 接口，可以通过上电过程中拉低 RESET 的方式禁止内部程序的运行，然后进行调试和 ISP 操作。SWD 接口关闭后，SWD 占用的两个 I/O 接口可以作为通用 I/O 使用。 为避免对 SWDD 的误操作，用户需要在第一次更新 SWDD 位之后的四个周期内再写一次 SWDD 才能生效。

[使用提示]：

为了更加准确有效的使用复位标志信息，建议用户尽量在程序的初始化前期读取复位标志然后将其清零。

看门狗控制状态寄存器- WDTCSR

WDTCSR – WDT 控制和状态寄存器								
地址: 0x60					默认值: 0x00			
Bit	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
[7]	WDIF	WDT 中断标志位。 当 WDT 工作在中断模式并发生溢出时会置位 WDIF 位。当 WDT 中断使能位 WDIE 为“1”且全局中断置位时，WDT 中断产生。执行 WDT 中断时会清零 WDIF 位，对 WDIF 位写“1”也可清零该位。

<table>
<tr><td>[6]</td><td>WDIE</td><td>WDT 中断使能控制位。
当设置 WDIE 位为“1”，且全局中断置位时，WDT 中断被使能。
当设置 WDIE 位为“0”时，WDT 中断被禁止。
WDIE 位和 WDE 位一起决定看门狗的工作模式，如下表所示。
<table>
<tr><th>WDE</th><th>WDIE</th><th>模式</th><th>溢出后动作</th></tr>
<tr><td>0</td><td>0</td><td>停止</td><td>无</td></tr>
<tr><td>0</td><td>1</td><td>中断模式</td><td>中断</td></tr>
<tr><td>1</td><td>0</td><td>复位模式</td><td>复位</td></tr>
<tr><td>1</td><td>1</td><td>中断及复位模式</td><td>中断后复位</td></tr>
</table></td></tr>
<tr><td>[5]</td><td>WDP3</td><td>WDT 预分频因子选择控制第 3 位。
WDP[3]和 WDP[2:0]组成 WDT 预分频因子选择位 WDP[3:0]，用来设置 WDT 的溢出周期。</td></tr>
<tr><td>[4]</td><td>WDTOE</td><td>WDT 关闭使能控制位。
当要把 WDE 位清零时，WDTOE 位须置位，否则 WDT 不会被关闭。当 WDTOE 位被置位后，硬件会在 4 个时钟周期后清零 WDTOE 位。</td></tr>
<tr><td>[3]</td><td>WDE</td><td>WDT 使能控制位。
当设置 WDE 位为“1”时，WDT 被使能。当设置 WDE 位为“0”时，WDT 被禁止。
只有在 WDTOE 位置位时 WDE 才能被清零。要关闭已经使能了的 WDT，必须按照下列时序操作：
1. 同时置位 WDTOE 和 WDE 位，即使 WDE 已经被置位，在关闭操作开始之前也必须对 WDE 位写入“1”;
2. 在接下来的 4 个时钟周期内，对 WDE 位写入“0”。这将关闭 WDT。
当 WDE 位为“1”且 WDT 溢出复位系统后会置位 WDT 复位系统标志 WDRF（位于 MCUSR 寄存器）。当 WDRF 位处于置位状态时会置位 WDE 位。因此要清零 WDE 位，必须先清零 WDRF 位。</td></tr>
<tr><td>[2:0]</td><td>WDP</td><td>WDT 预分频因子选择控制。
用来设置 WDT 的溢出周期。建议在 WDT 未计数时改变 WDP 的值，在计数过程中改变 WDP 的值就会产生不可预期的 WDT 溢出。</td></tr>
</table>

看门狗预分频选择列表：

WDP3	WDP2	WDP1	WDP0	看门狗定时器溢出周期数	32KHz时钟	2MHz时钟
0	0	0	0	512 cycles	16ms	256us
0	0	0	1	1K cycles	32ms	512us
0	0	1	0	2K cycles	64ms	1us
0	0	1	1	4K cycles	128ms	2ms
0	1	0	0	8K cycles	256ms	4ms
0	1	0	1	16K cycles	512ms	8ms
0	1	1	0	32K cycles	1.0s	16ms
0	1	1	1	64K cycles	2.0s	32ms
1	0	0	0	128K cycles	4.0s	64ms
1	0	0	1	256K cycles	8.0s	128ms

1	0	1	0	保留不用
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

中断与中断向量

- 28 个中断源
- 可编程向量起始地址

中断向量列表

表 1：LGT8F88A 中断向量列表

编号	向量地址	中断源信号	中断源说明
1	0x0000	RESET	外部复位，上电复位，看门狗复位，SWD 调试复位，低电压复位
2	0x0001	INT0	外部中断请求 0
3	0x0002	INT1	外部中断请求 1
4	0x0003	PCI0	引脚电平中断 0
5	0x0004	PCI1	引脚电平中断 1
6	0x0005	PCI2	引脚电平中断 2
7	0x0006	WDT	看门狗溢出中断
8	0x0007	TC2 COMPA	定时器 2 比较匹配 A 中断
9	0x0008	TC2 COMPB	定时器 2 比较匹配 B 中断
10	0x0009	TC2 OVF	定时器 2 溢出中断
11	0x000A	TC1 CAPT	定时器 1 输入捕捉中断
12	0x000B	TC1 COMPA	定时器 1 比较匹配 A 中断
13	0x000C	TC1 COMPB	定时器 1 比较匹配 B 中断
14	0x000D	TC1 OVF	定时器 1 溢出中断
15	0x000E	TC0 COMPA	定时器 0 比较匹配 A 中断
16	0x000F	TC0 COMPB	定时器 0 比较匹配 B 中断
17	0x0010	TC0 OVF	定时器 0 溢出中断
18	0x0011	SPI STC	SPI 串行传输结束中断
19	0x0012	USART RXC	USART 接收结束中断
20	0x0013	USART UDRE	USART 数据寄存器空中断
21	0x0014	USART TXC	USART 发送结束中断
22	0x0015	ADC	ADC 转换结束中断
23	0x0016	EE_RDY	EEPROM 就绪中断
24	0x0017	ANA_COMP	模拟比较器中断

25	0x0018	TWI	两线串行接口中断
26	0x0019	-	保留
27	0x001A	-	保留
28	0x001B	PCI3	引脚电平中断 3

LGT8F88A 的复位向量从地址 0x0000 开始执行。除复位向量外，其他向量地址都可以通过 MCUCR 寄存器中的 IVSEL 以及 IVBASE 寄存器重新定向到 512 字节对齐的起始地址。

中断向量处理

下面的代码实例是常用复位以及中断向量编程，仅供参考：

汇编代码实例		
地址	代码	说明
0x000	RJMP RESET	复位向量
0x001	RJMP EXT_INT0	外部中断 0
0x002	RJMP EXT_INT1	外部中断 1
0x003	RJMP PCINT0	引脚电平变化中断 0
0x004	RJMP PCINT1	引脚电平变化中断 1
0x005	RJMP PCINT2	引脚电平变化中断 2
0x006	RJMP WDT	看门狗定时器中断
0x007	RJMP TIM2_COMPA	定时器 2 比较匹配 A 组中断
0x008	RJMP TIM2_COMPB	定时器 2 比较匹配 B 组中断
0x009	RJMP TIM2_OVF	定时器 2 溢出中断
0x00A	RJMP TIM1_CAPT	定时器 1 俘获中断
0x00B	RJMP TIM1_COMPA	定时器 1 比较匹配 A 组中断
0x00C	RJMP TIM1_COMPB	定时器 1 比较匹配 B 组中断
0x00D	RJMP TIM1_OVFR	定时器 1 溢出中断
0x00E	RJMP TIM0_COMPA	定时器 0 比较匹配 A 组中断
0x00F	RJMP TIM0_COMPB	定时器 0 比较匹配 B 组中断
0x010	RJMP TIM0_OVF	定时器 0 溢出中断
0x011	RJMP SPI_STC	SPI 传输完成中断
0x012	RJMP USART_RXC	USART 接收完成中断
0x013	RJMP USART_UDRE	USART 数据寄存器空中断
0x014	RJMP USART_TXC	USART 发送完成中断
0x015	RJMP ADC	ADC 转换完成中断
0x016	RJMP EE_RDY	EEPROM 控制器准备好中断
0x017	RJMP ANA_COMP	比较器中断
0x018	RJMP TWI	TWI 控制器中断
0x019	NOP	保留地址
0x01A	NOP	保留地址
0x01B	RJMP PCI3	引脚电平变化中断 3

;		
0x01C (**RESET :**)	LDI r16, high(RAMEND)	主程序开始
0x01D	OUT SPH, r16	设置堆栈指针为 RAM 顶端地址
0x01E	LDI r16, low(RAMEND)	
0x01F	OUT SPL, r16	
0x020	SEI	使能全局中断
0x021	……	

寄存器定义

MCU 控制寄存器- MCUCR

MCUCR – MCU 控制寄存器								
MCUCR: 0x35(0x55)				默认值: 0x00				
MCUCR	-	-	PUD	-	-	-	IVSEL	IVCE
R/W	-	-	R/W	-	-	-	R/W	R/W
初始值	-	-	0	-	-	-	0	0

位定义		
[0]	IVCE	中断向量选择更改使能位，在更改 IVSEL 之前，需要首先设置此位，然后在 6 个周期内，设置 IVSEL。
[1]	IVSEL	中断向量选择位，此位置 1 后，中断向量地址将根据 IVBASE 寄存器的值映射到新的地址。详细映射地址，请参考 IVBASE 寄存器说明
[2]	-	保留不用
[3]	-	保留不用
[4]	PUD	全局上拉禁止位
[5]	-	保留不用
[6]	-	保留不用
[7]	-	保留不用

中断向量基地址寄存器 - IVBASE

IVBASE – 中断向量基地址寄存器		
IVBASE: 0x75		默认值: 0x00
IVBASE	IVBASE[7:0]	
R/W	R/W	
初始值	0x00	

位定义		
[7:0]	IVBASE	如果 IVSEL 为 1，中断向量(复位向量除外)将以 IVBASE 为基地址在 512 字节的页面上重新映射。 映射后的中断向量基地址为: (IVBASE << 8) + 表 1 中对应的向量地址

外部中断

- 2 个外部中断源
- 可配置的电平或边沿触发中断
- 可用作睡眠模式下的唤醒源

概述

外部中断由 INT0 和 INT1 引脚触发。只要外部中断被使能，即使这 2 个引脚配置为输出也能触发中断。这可以用来产生软件中断。外部中断可以由上升沿，下降沿或低电平触发，由外部中断控制寄存器 EICRA 来配置。当外部中断使能并且配置为电平触发（只有 INT0 和 INT1 引脚）时，只要引脚电平为低，中断就会一直产生。INT0 和 INT1 引脚的上升沿或下降沿中断触发需要 IO 时钟正常工作，而 INT0 和 INT1 引脚的低电平触发中断都是异步检测的。除了空闲模式，其它睡眠模式下 IO 时钟都是停止工作的。因此，这 2 个外部中断都可用作除空闲模式外的其它睡眠模式下的唤醒源。

若电平触发中断用作省电模式下的唤醒源，改变的电平必须保持一定的时间来唤醒 MCU，以降低 MCU 对噪声的敏感程度。要求的电平必须保持足够长的时间使 MCU 结束唤醒过程，然后触发电平中断。

寄存器定义

寄存器列表

寄存器	地址	默认值	描述
EICRA	0x69	0x00	外部中断控制寄存器 A
EIMSK	0x3D	0x00	外部中断屏蔽寄存器
EIFR	0x3C	0x00	外部中断标志寄存器

外部中断控制寄存器 A- EICRA

EICRA – 外部中断控制寄存器 A								
地址: 0x69				默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:4	-	保留。
3	ISC11	INT1 引脚中断触发方式控制位高位。
2	ISC10	INT1 引脚中断触发方式控制位低位。 当全局中断置位且 GICR 寄存器的相应中断屏蔽控制位被置位时，外部中断 1 由 INT1 引脚激发。中断的触发方式见表格描述。在边沿检测之前 MCU 首先采

		样 INT1 引脚上的电平。如果选用了边沿触发方式或电平变化触发方式，那么持续时间大于 1 个系统时钟周期的脉冲将触发中断，过短的脉冲则不能保证触发中断。如果选择低电平触发方式，那么低电平必须保持到当前指令执行完成才会触发中断。
1	ISC01	INT0 引脚中断触发方式控制位高位。
0	ISC00	INT0 引脚中断触发方式控制位低位。 当全局中断置位且 GICR 寄存器的相应中断屏蔽控制位被置位时，外部中断 0 由 INT0 引脚激发。中断的触发方式见表格描述。在边沿检测之前 MCU 首先采样 INT0 引脚上的电平。如果选用了边沿触发方式或电平变化触发方式，那么持续时间大于 1 个系统时钟周期的脉冲将触发中断，过短的脉冲则不能保证触发中断。如果选择低电平触发方式，那么低电平必须保持到当前指令执行完成才会触发中断。

外部中断 1 触发方式见下表。

外部中断 1 触发方式控制

ISC1[1:0]	描述
0	外部引脚 INT1 低电平触发
1	外部引脚 INT1 上升沿或下降沿触发
2	外部引脚 INT1 下降沿触发
3	外部引脚 INT1 上升沿触发

外部中断 0 触发方式见下表。

外部中断 0 触发方式控制

ISC0[1:0]	描述
0	外部引脚 INT0 低电平触发
1	外部引脚 INT0 上升沿或下降沿触发
2	外部引脚 INT0 下降沿触发
3	外部引脚 INT0 上升沿触发

外部中断屏蔽寄存器- EIMSK

EIMSK – 外部中断屏蔽寄存器								
地址: 0x3D							默认值: 0x00	
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	INT1	INT0
R/W	-	-	-	-	-	-	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:2	-	保留。
1	INT1	外部引脚 1 中断使能控制位。 当设置 INT1 位为“1”时，且全局中断置位，外部引脚 1 中断被使能，唤醒

		功能被使能。即使 INT1 引脚被配置为输出，只要引脚电平发生了相应的变化，中断将产生。 当设置 INT1 位为“0”时，外部引脚 1 中断被禁止，唤醒功能也被禁止。
0	INT0	外部引脚 0 中断使能控制位。 当设置 INT0 位为“1”时，且全局中断置位，外部引脚 0 中断被使能，唤醒功能被使能。即使 INT0 引脚被配置为输出，只要引脚电平发生了相应的变化，中断将产生。 当设置 INT0 位为“0”时，外部引脚 0 中断被禁止，唤醒功能也被禁止。

外部中断标志寄存器- EIFR

EIFR – 外部中断标志寄存器								
地址: 0x3C					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	INTF1	INTF0
R/W	-	-	-	-	-	-	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:2	-	保留。
1	INTF1	外部引脚 1 中断标志位。 当边沿触发外部引脚 1 中断时，INTF1 被置位。当低电平触发外部引脚 1 中断时，不会置位 INTF1 位。若此时外部引脚 1 中断使能 INT1EN 位为“1”且全局中断标志置位，则会产生外部引脚 1 中断。执行此中断服务程序时 INTF1 将自动清零，或对 INTF1 位写“1”也可清零该位。
0	INTF0	外部引脚 0 中断标志位。 当边沿触发外部引脚 0 中断时，INTF0 被置位。当低电平触发外部引脚 0 中断时，不会置位 INTF0 位。若此时外部引脚 0 中断使能 INT0EN 位为“1”且全局中断标志置位，则会产生外部引脚 0 中断。执行此中断服务程序时 INTF0 将自动清零，或对 INTF0 位写“1”也可清零该位。

输入/输出子系统

概述

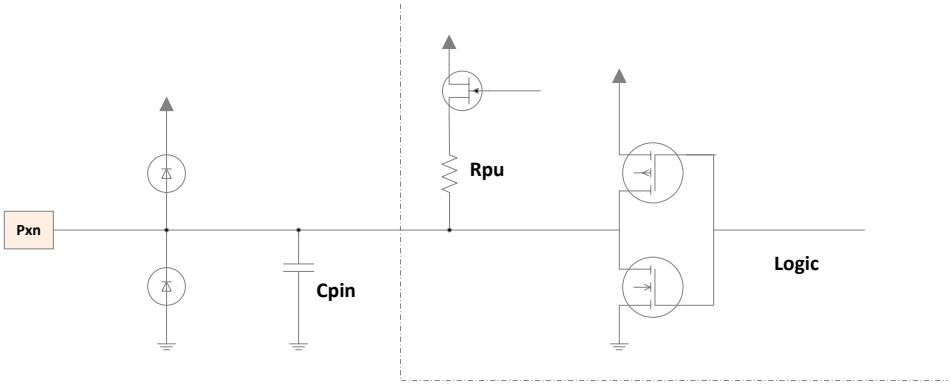
所有基于 MVR8 内核系列实现的 MCU 都具有 I/O 端口读-改-写功能。这意味着，某一个端口的状态可以使用 SBI 和 CBI 指令单独的改变，而不会影响到其他任何 I/O。同样，改变一个端口的方向或者控制它的上拉电阻也可以如此。

LGT8F88A 的大部分 I/O 拥有对称的驱动特性，能够驱动和吸收较大的电流。I/O 具有两级驱动能力，用户可以控制每组 I/O 的驱动能力。I/O 的驱动能力可以直接驱动一些 LED。

LGT8F88A 也包含了多达 6 个大电流扇入能力的 I/O，可以吸收高达 80mA 的电流，可以直接驱动共阴极的七段码 LED。关于端口驱动能力的配置，请参考 DSCR 寄存器的说明部分。

所有的 I/O 的 VCC 和 GND 直接都有独立的 ESD 保护二极管，设计至少可以承受高达 5000V 的 ESD 脉冲。

I/O 等效电路图：



本章下面所有寄存器采用统一描述方式，小写的”x”表示端口的字母序号名，小写的”n”表示端口中的位号。但当在程序中使用端口寄存器时，必须使用准确的寄存器名字。比如 PORTB3，它表示 PORTB 的第三位，这里则统一用 PORTxn 表示。I/O 相关寄存器的详细定义，请参考寄存器描述部分。

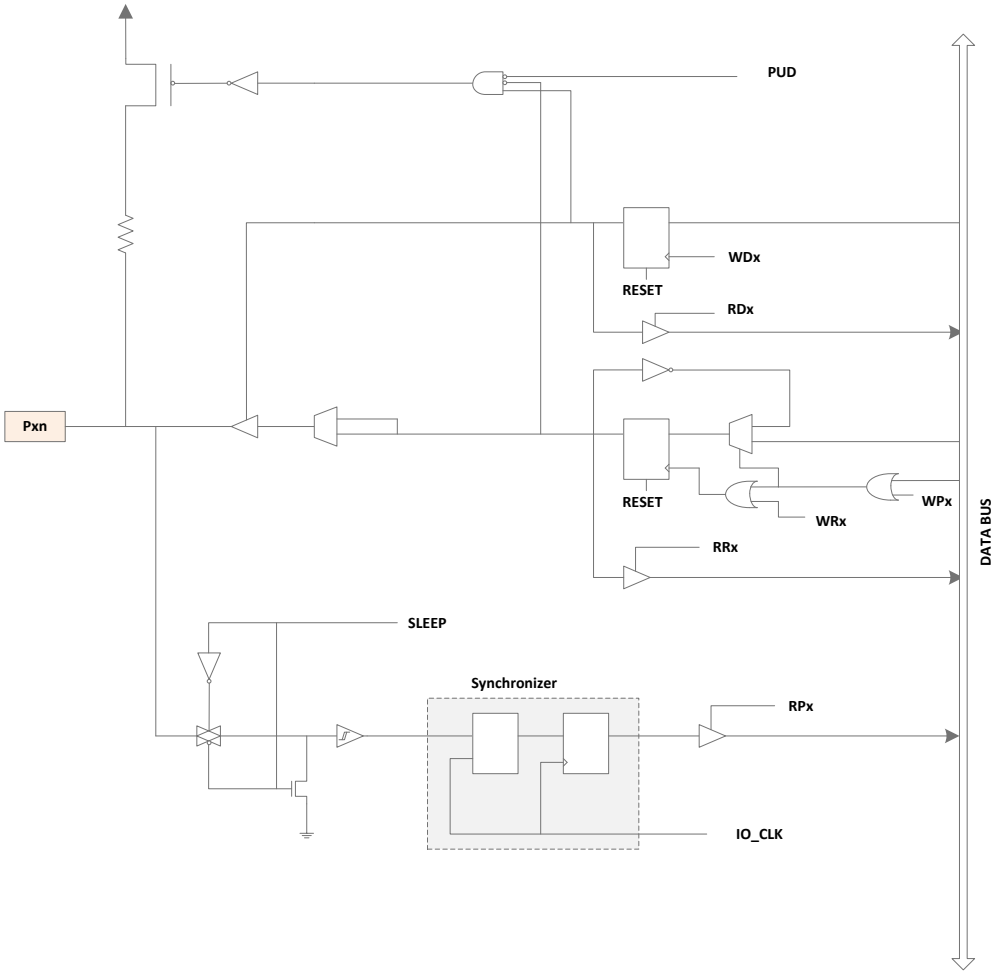
每个端口分配有三个 I/O 寄存器空间，它们为：端口数据输出寄存器(PORTx)，端口方向寄存器(DDRx)，端口数据输入寄存器(PINx)。端口数据输入寄存器为只读寄存器。数据输出寄存器与端口方向寄存器可读也可以改写。MCUCR 寄存器中的 PUD 位，用于控制所有 I/O 的上拉电阻，当 PUD 位为 1 时，将禁止所以 I/O 的上拉电阻。

大部分 I/O 除了具有通用输入/输出功能，也会被复用为其他外设功能。具体的复用功能请参考关于端口功能复用的章节。

需要注意的是，使能某些端口的复用功能并不会影响这些端口作为数字 I/O 使用。而且某些复用功能也可能需要通过 I/O 寄存器控制端口的输入/输出方向。具体的设置将会在各个复用模块的文档的介绍。

通用输入/输出端口

作为通用 I/O 时，端口为双向驱动 I/O 端口，内部可编程上拉。



下图为通用 I/O 端口的等效电路图：

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

端口使用配置

每个端口由三个寄存器位控制：DDxn，PORTxn 和 PINxn。其中 DDxn 用于可以通过 DDRx 寄存器访问，PORTxn 可以通过 PORTx 寄存器访问，PINxn 可以通过 PINx 寄存器访问。

DDRxn 寄存器位用于设置端口的输入/输出方向。如果 DDxn 设置为 1，Pxn 端口就被配置为一个输出端口。如果 DDxn 设置为 0，Pxn 就被配置为一个输入端口。

如果 PORTxn 位被写 1，同时这个端口被配置为输入端口，这个端口的上拉电阻有效。如果想要禁止端口的上拉电阻，PORTxn 必须写为 0 或者将这个端口配置为输出端口。

端口的复位初始化状态为输入状态，上拉电阻无效。

PORTxn 设置为 1，同时这个端口被配置为输出端口，外部端口将会被驱动为高电平。如果 PORTxn 设置为 0，端口将会被驱动为低。

输入/输出切换

当 I/O 状态在三态([DDxn, PORTxn]) = 0b00)和输出高电平([DDxn, PORTxn] = 0b11)之间切换时，将会出现一个端口上拉或者输出为低的中间状态。通常，上拉电阻是可以被接受的，因为在一个高阻环境下，驱动为高和上拉之间的区别并不重要。如果不是这种情况，可以通过 MCUCR 寄存器中的 PUD 位关闭所以端口的上拉功能。

同样，在上拉使能的输入与输出低电平之间切换时，也会出现同样的问题。用户必须使用三态([DDxn, PORTxn] = 0b00)或者输出高([DDxn, PORTxn] = 0b11)作为中间状态。

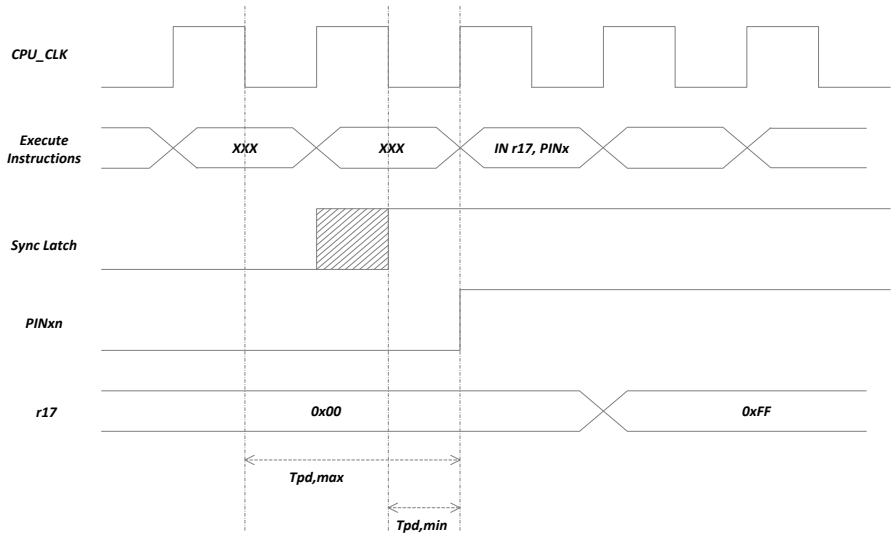
端口驱动配置表：

DDxn	PORTxn	PUD	端口状态	上拉	功能说明
0	0	X	输入	禁止	三态(High-Z)
0	1	0	输入	使能	如果外部下拉，引脚将扇出电流
0	1	1	输入	禁止	三态(High-Z)
1	0	X	输出	禁止	输出低(扇入)
1	1	X	输出	禁止	输出高(扇出)

读端口值

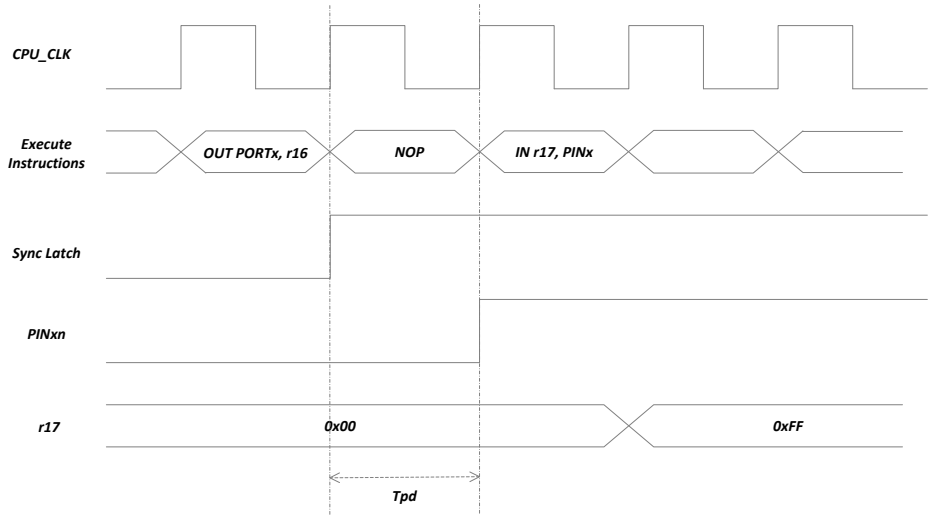
无论端口方向位 DDxn 如何设置，都可以通过 PINxn 寄存器位读取到端口的当前状态。为避免直接读取端口产生的亚稳态，PINxn 寄存器位是端口经过一个同步器的结果。同步器为一个锁存器和一个寄存器共同组成，因此 PINxn 的值与当前端口之间有一个很小的延迟。这个延迟是因为同步器存在的结果，延迟时间最多为 1 个半系统周期。

延迟时间如下图中的 Tpd,max 与 Tpd,min:



我们假设系统周期从系统时钟的第一个下降沿开始，锁存器在时钟为低的时候锁存数据，时钟为高时数据直通过锁存器，如上图中阴影部分所示。在时钟为低电平时，端口数据被锁存，并且在下一个时钟的上升沿被寄存器到 PINxn 寄存器。上图中的 Tpd,max 以及 Tpd,min 为端口数据的最大和最小延迟，分为为 1.5 周期和 0.5 周期。

如果要读取到软件设置的端口值，需要在 I/O 的写和读字节支持插入一个空操作指令(NOP)。时序如下图所示：



下面的代码说明如何设置端口 B 的引脚 0/1 为高，2/3 为低，定义引脚 4~7 为输入并且使能了引脚 6、7 的上拉电阻。然后引脚的值回读到通用工作寄存器中，按照之前的描述，在引脚的输出和输入直接插入了一个 NOP 指令。

汇编代码

```
; 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 语言代码

```
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;
```

输入使能与休眠控制

从 I/O 的等效电路图中我们可以看到，数字输入可以在 SLEEP 信号的控制下被钳位到地电平。SLEEP 信号由 MCU 的休眠控制器以及各种休眠模式控制。这样可以保证在进入休眠后，系统不会因为端口输入浮空而造成漏电。

端口的 SLEEP 控制作用会被外部中断功能取代。如果外部中断请求无效，SLEEP 控制仍然可以起作用。SLEEP 控制功能也会被其他一些第二功能取代，具体请参考下面关于端口第二功能的介绍。

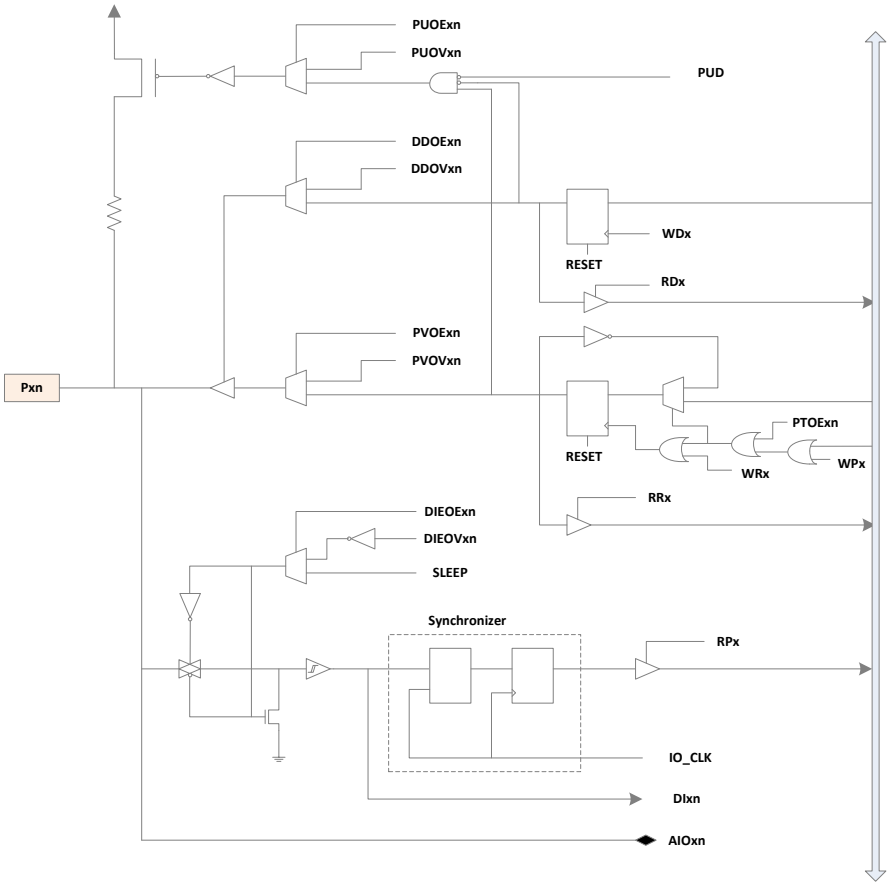
空闲端口的处理

如果一些端口没有被使用，建议将他们驱动到一个固定的电平。在任何情况下，浮空的引脚都会带来更多的功耗，并且会导致系统在强干扰下变的不稳定。

给端口一个固定电平最简单的方法就是打开端口的上拉电阻。需要注意的是，上拉电阻在上电复位过程中是禁止的。上拉电阻的方式也会带来多余的漏电。因此建议使用外部的上拉或者下拉电阻连接。直接将端口与电源或地连接是不建议的，因为如果这些引脚被配置为输出，会有可能导致非常大的电流由端口经过，对芯片造成破坏性的影响。

端口复用功能

大部分端口都有复用功能，下面的等效电路说明了端口复用功能对端口的控制。这些复用功能并不一定存在与所以的端口引脚。



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

复用功能控制信号一般描述：

信号	全称	功能描述
PUOE	上拉复用使能	此位为 1，上拉使能由 PVOV 控制；如果此位为 0，上拉使能受 DDxn, PORTxn 以及 PUD 共同控制
PUOV	上拉复用值	如果 PUOE 为 1，此位为 1 将使能引脚的上拉电阻，否则将禁止引脚上拉电阻
DDOE	端口方向复用使能	次位为 1，引脚输出使能由 DDOE 控制，否则由 DDxn 控制
DDOV	端口方向复用值	如果 DDOE 为 1，次位为 1，将使能引脚的输出功能，否则关闭引脚的输出
PVOE	端口数据复用使能	如果次位为 1，并且引脚输出使能，引脚的输出值将由 PVOV 控制，否则是由 PORTxn 控制
PVOV	端口数据复用值	参考 PVOE 功能描述
PTOE	端口翻转复用使能	次位为 1，PORTxn 位将翻转
DIEOE	数字输入使能复用使能	如果次位为 1，端口数字输入使能将由 DIEOV 控制；否则将有 MCU 的运行状态控制
DIEOV	数字输入使能复用值	如果 DIEOE 为 1，端口的数字输入功能将由次位控制，与 MCU 运行状态无关
DI	数字输入	这个是输入给替代功能模块的数字输入信号。从 I/O 等下电路图中可以看到，这个值在施密特触发器之后，但在 I/O 输入同步器之前。这个信号连接到外设模块中，外设模块将会根据需要进行同步处理
AIO	模拟输入	模拟输入/输出信号，这个信号直接与 I/O 的 PAD 相连，可作为模拟的双向信号使用。这个信号直接与内部的 ADC、电容触摸控制器等模拟模块的端口相连接

下面一小节将会简短的描述每个引脚的复用功能和相关的控制信号。

端口 B 的复用功能

引脚	复用功能描述
PB7	XTAL2/TOSC2 (外部主晶振引脚 2) PCINT7 (引脚电平变化中断 7)
PB6	XTAL1/TOSC1 (外部主晶振引脚 1) PCINT6 (引脚电平变化中断 6)
PB5	TK5 (触摸按键输入通道 5) SCK (SPI 总线主时钟输入) PCINT5 (引脚电平变化中断 5)
PB4	TK4 (触摸按键输入通道 4) MISO (SPI 总线主输入/从输出) PCINT4 (引脚电平变化中断 4)
PB3	TK3 (触摸按键输入通道 3) MOSI (SPI 总线主输出/从输入) OC2A (定时/计数器 2 比较匹配输出 A) PCINT3 (引脚电平变化中断 3)
PB2	SSN (SPI 总线从设备选择输入) OC1B (定时/计数器 1 比较匹配输出 B) PCINT2 (引脚电平变化中断 2)
PB1	OC1A (定时/计数器 1 比较匹配输出 A) PCINT1 (引脚电平变化中断 1)
PB0	TK2 (触摸按键输入通道 2) ICP1 (定时/计数器 1 俘获输入) CLKO (系统时钟输出) PCINT0 (引脚电平变化中断 0)

XTAL2/TOSC2/PCINT7 – 端口 B 引脚 7

XTAL2: 外部晶振引脚 2。当用作晶振的时钟信号时，这个引脚将不能作为 I/O 使用。
TOSC2: 定时器外部晶振引脚 2。当内部 RC 被配置为芯片的主工作时钟，并且使能了异步定时器功能(ASSR 寄存器配置)，此引脚将作为定时器的外部晶振引脚。当 ASSR 寄存器的 AS2 被设置为 1，EXCLK 为设置为 0，便使能了定时/计数器 2 使用外部晶振的异步时钟功能，PB7 将与内部 I/O 端口断开，成为内部振荡放大器的反向输出引脚。这种模式下，外部晶振与引脚相连接。
PCINT7: 引脚电平变化中断 7。PB7 为外部中断源。
如果 PB7 被用于晶振引脚，DDB7,PORTB7 和 PINB7 的值将没有任何意义。

XTAL1/TOSC1/PCINT6- 端口 B 引脚 6

XTAL1: 外部晶振引脚 1。
TOSC1: 定时器外部晶振引脚 1。当内部 RC 被配置为芯片的主工作时钟，并且使能了异步定时器功能(ASSR 寄存器配置)，此引脚将作为定时器的外部晶振引脚。当 ASSR 寄存器的 AS2 被设置为 1，EXCLK 为设置为 0，便使能了定时/计数器 2 使用外部晶振的异步

时钟功能，PB6 将与内部 I/O 端口端口，成为内部振荡放大器的输入引脚。这种模式下，外部晶振与引脚相连接。

PCINT6: 引脚电平变化中断 6。PB6 为外部中断源。

如果 PB6 被用于晶振引脚，DDB6,PORTB6 和 PINB6 的值将没有任何意义。

TK5/SCK/PCINT5- 端口 B 引脚 5

TK5: 触摸按键输入通道 5

SCK: SPI 控制器主设备时钟输出，从设备时钟输入。当 SPI 控制器被配置为一个从设备，这个引脚将被配置为一个输入引脚，不受 DDB5 的控制。当 SPI 控制器被配置为主设备，这个引脚的方向由 DDB5 控制。当这个引脚被 SPI 强制为输入后，仍然可以通过 PORTB5 位控制上拉电阻。

PCINT5: 引脚电平变化中断。PB5 为外部中断源。

TK4/MISO/PCINT4- 端口 B 引脚 4

TK4: 触摸按键输入通道 4

MISO: SPI 控制主设备数据输入，从设备数据输出。当 SPI 被配置为主设备，这个引脚将会被强制为输入，并不受 DDB4 的控制。当 SPI 作为一个从设备时，这个引脚的数据方向由 DDB4 控制。当这个引脚被 SPI 控制器强制为输入后，它的上拉电阻仍然可以通过 PROTB4 控制。

PCINT4: 引脚电平变化中断。PB4 为外部中断源。

TK3/MOSI/OC2A/PCINT3- 端口 B 引脚 3

TK3: 触摸按键输入通道 3

MOSI: SPI 控制器主设备数据输出，从设备数据输入。当 SPI 被配置为从设备，这个引脚将会被强制为输入，并不受 DDB3 的控制。当 SPI 控制器被配置为主设备，这个引脚的方法由 DDB3 控制。当这个引脚被 SPI 控制强制为输入，仍然可以通过 PORTB3 控制它的上拉电阻。

OC2A: 定时/计数器 2 的 A 组比较匹配输出。PB3 可以作为定时/计数器 2 比较匹配的外部输出。此时必须通过 DDB3 将引脚设置为输出。同时，OC2A 也是定时器 2 的 PWM 模式输出引脚。

PCINT3: 引脚电平变化中断。PB3 为外部中断源。

SSN/OC1B/PCINT2- 端口 B 引脚 2

SSN: SPI 从设备片选输入。当 SPI 控制器配置为从设备，这个引脚将会被强制为输入，并不受 DDB2 的控制。作为一个从设备，SPI 控制器在 SSN 被驱动为低是有效。当 SPI 控制器配置为主设备，这个引脚的方向由 DDB2 控制。当这个引脚被 SPI 控制器强制为输入后，仍然可以通过 PORTB2 控制上拉电阻。

OC1B: 定时/计数器 1 的 B 组比较匹配输出。PB2 可以作为定时/计数器 1 比较匹配的外部输出。此时必须通过 DDB2 将引脚设置为输出。同时，OC1B 也是定时器 1 的 PWM 模式输出引脚。

PCINT2: 引脚电平变化中断。PB2 为外部中断源。

OC1A/PCINT1- 端口 B 引脚 1

OC1A: 定时/计数器 1 的 A 组比较匹配输出。PB1 可以作为定时/计数器 1 比较匹配的外部输出。此时必须通过 DDB1 将引脚设置为输出。同时，OC1A 也是定时器 1 的 PWM 模式输出引脚。

PCINT1: 引脚电平变化中断。PB1 为外部中断源。

TK2/ICP1/CLKO/PCINT0- 端口 B 引脚 0

TK2: 触摸按键输入通道 2

ICP1: 定时/计数器 1 的俘获输入引脚

CLKO: 系统工作时钟输出，当 CLKPR 寄存器中的 CLKOE 位为 1，这个引脚将会被强制为输出，不受 DDB0 的控制。输出频率为当前系统的工作时钟频率。

PCINT0: 引脚电平变化中断。PB0 为外部中断源。

PB7…PB4 复用能控制逻辑表

信号名称	PB7/XTAL2/ TOSC2/PCINT7	PB6/XTAL1/ TOSC1/PCINT6	PB5/TK5/SCK PCINT5	PB4/TK4/MISO PCINT4
PUOE	OSCEN | AS2	OSCEN | AS2	SPE&$\overline{\text{MSTR}}$	SPE&MSTR
PUOV	0	0	PORTB5&$\overline{\text{PUD}}$	PORTB4&$\overline{\text{PUD}}$
DDOE	OSCEN | AS2	OSCEN | AS2	SPE&$\overline{\text{MSTR}}$	SPE&MSTR
DDOV	0	0	0	0
PVOE	0	0	SPE&MSTR	SPE&$\overline{\text{MSTR}}$
PVOV	0	0	SCK Output	SPI Slave Output
DIEOE	PCINT7 Enable	PCINT6 Enable	PCINT5 Enable	PCINT4 Enable
DIEOV	1	1	1	1
DI	PCINT7 Input	PCINT6 Input	PCINT5 Input SCK Input	PCINT4 Input SPI Master Input
AIO	XTAL2 TOSC2	XTAL1 TOSC1	TK5	TK4

[说明]: OSCEN 包括 OSCK_EN 与 OSCM_EN，请参考 PMCR 寄存器描述

PB3…PB0 复用功能控制逻辑表

信号名称	PB3/TK3/MOSI/ OC2A/PCINT3	PB2/SSN/ OC1B/PCINT2	PB1/OC1A/ PCINT1	PB0/TK2/ICP1/ CLKO/PCINT0
PUOE	SPE&$\overline{\text{MSTR}}$	SPE&$\overline{\text{MSTR}}$	0	0
PUOV	PORTB3&$\overline{\text{PUD}}$	PORTB2&$\overline{\text{PUD}}$	0	0
DDOE	SPE&$\overline{\text{MSTR}}$	SPE&$\overline{\text{MSTR}}$	0	CLKO ENABLE 0
DDOV	0	0	0	1
PVOE	SPE&MSTR + OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	CLKO ENABLE 0
PVOV	SPI Master Output OC2A	OC1B	OC1A	CLKO
DIEOE	PCINT3 Enable	PCINT2 Enable	PCINT1 Enable	PCINT0 Enable

DIEOV	1	1	1	1
DI	PCINT3 Input SPI Slave Input	PCINT2 Input SPI Slave Select	PCINT1 Input	PCINT0 Input ICP1 Input
AIO	TK3	-	-	TK2

端口 C 复用功能

引脚	复用功能描述
PC6	RESETN (外部复位输入) PCINT14 (引脚电平变化中断 14)
PC5	TK9 (触摸按键输入通道 9) ADC5 (ADC 输入通道 5) SCL (TWI 时钟线) PCINT13 (引脚电平变化中断 13)
PC4	TK8 (触摸按键输入通道 8) ADC4 (ADC 输入通道 4) SDA (TWI 数据线) PCINT12 (引脚电平变化中断 12)
PC3	ADC3 (ADC 输入通道 3) PCINT11 (引脚电平变化中断 11)
PC2	ADC2 (ADC 输入通道 2) PCINT10 (引脚电平变化中断 10)
PC1	ADC1 (ADC 输入通道 1) PCINT9 (引脚电平变化中断 9)
PC0	ADC0 (ADC 输入通道 0) PCINT8 (引脚电平变化中断 8)

RESETN/PCINT4- 端口 C 引脚 6

RESETN: 外部复位输入引脚。上电复位后，这个引脚默认为外部复位功能。可以通过 IOCR 寄存器关闭外部复位功能。关闭外部复位功能后，这个引脚可作为通用 I/O 使用。但需要注意的是，在上电和其他复位过程中，这个引脚默认为复位输入，所以如果用户需要用到这个引脚的通用 I/O 功能，外部电路不能影响到芯片的上电和复位过程，建议将这个引脚配置为输出功能的 I/O，并在外部加一个适当的上拉电阻。

PCINT14: 引脚电平变化中断。关闭这个引脚的外部复位输入功能后， PC6 可以做为外部中断源。

TK9/SCL/ADC5/PCINT13- 端口 C 引脚 5

TK9: 触摸按键输入通道 9

SCL: TWI 接口时钟信号。TWCR 寄存器中的 TWEN 位置 1 后，使能 TWI 接口，PC5 将被 TWI 控制，成为 TWI 接口的时钟信号。

ADC5: ADC 输入通道 5。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。

PCINT13: 引脚电平变化中断 13。

TK8/SDA/ADC4/PCINT12- 端口 C 引脚 4

TK8: 触摸按键输入通道 8
SDA: TWI 接口数据信号。TWCR 寄存器中的 TWEN 位置 1 后，使能 TWI 接口，PC4 将被 TWI 控制，成为 TWI 接口的数据信号。
ADC4: ADC 输入通道 4。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。
PCINT12: 引脚电平变化中断 12。

ADC3/PCINT11- 端口 C 引脚 3

ADC3: ADC 输入通道 3。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。
PCINT11: 引脚电平变化中断 11。

ADC2/PCINT1- 端口 C 引脚 2

ADC2: ADC 输入通道 2。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。
PCINT10: 引脚电平变化中断 10。

ADC1/PCINT9- 端口 C 引脚 1

ADC1: ADC 输入通道 1。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。
PCINT9: 引脚电平变化中断 9。

ADC0/PCINT8- 端口 C 引脚 0

ADC0: ADC 输入通道 0。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。
PCINT8: 引脚电平变化中断 8。

PC6…PC4 复用控制逻辑表

信号名称	PC6/RESETN/ PCINT14	PC5 /ADC5/TK9/SCL/ PCINT13	PC4/ADC4/TK8/SDA/ PCINT12
PUOE	$\overline{\text{RSTIOEN}}$	TWI Enable	TWI Enable
PUOV	1	PORTC4&$\overline{\text{PUD}}$	PORTC4&$\overline{\text{PUD}}$
DDOE	$\overline{\text{RSTIOEN}}$	TWI Enable	TWI Enable
DDOV	0	$\overline{\text{SCL Output}}$	$\overline{\text{SDA Output}}$
PVOE	$\overline{\text{RSTIOEN}}$	TWI Enable	TWI Enable
PVOV	1	0	0
DIEOE	PCINT14 Enable + $\overline{\text{RSTIOEN}}$	PCINT13 Enable + TWI Enable	PCINT12 Enable + TWI Enable

DIEOV	1	1	1
DI	PCINT14 Input External Reset Input	PCINT13 Input SCL Input	PCINT12 Input SDA Input
AIO	-	ADC5 / TK9	ADC4 / TK8

PC3…PC0 复用控制逻辑表

信号名称	PC3/ADC3/ PCINT11	PC2/ADC2/ PCINT10	PC1/ADC1/ PCINT9	PC0/ADC0/ PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT11 Enable	PCINT10 Enable	PCINT9 Enable	PCINT8 Enable
DIEOV	1	1	1	1
DI	PCINT11 Input	PCINT10 Input	PCINT9 Input	PCINT8 Input
AIO	ADC3	ADC2	ADC1	ADC0

端口 D 复用功能

引脚	复用功能描述
PD7	AIN1 (模拟比较器负端输入) TK1 (触摸按键输入通道 1) PCINT23 (引脚电平变化中断 23)
PD6	AIN0 (模拟比较器正端输入) TK0 (触摸按键输入通道 0) OC0A (定时/计数器 0 比较匹配输出 A) PCINT22 (引脚电平变化中断 22)
PD5	T1 (定时/计数器 1 外部计数时钟输入) OC0B (定时/计数器 0 比较匹配输出 B) PCINT21 (引脚电平变化中断 21)
PD4	XCK (USART 外部时钟输入/输出) T0 (定时/计数器 0 外部计数时钟输入) PCINT20 (引脚电平变化中断 20)
PD3	INT1 (外部中断输入 1) OC2B (定时/计数器 2 比较匹配输出 B) PCINT19 (引脚电平变化中断 19)
PD2	INT0 (外部中断输入 0) PCINT18 (引脚电平变化中断 18)
PD1	TXD (USART 数据输出) PCINT17 (引脚电平变化中断 17)
PD0	RXD (USART 数据输入) PCINT16 (引脚电平变化中断 16)

AIN1/TK1/OC2B/PCINT23- 端口 D 引脚 7

AN1: 模拟比较器负端输入。通过 DIDR1 寄存器关闭 PD7 引脚的数字输入功能，并关闭端口的上拉电阻，以避免数字端口对模拟电路的干扰。
TK1: 触摸按键输入通道 1。使用方法与 AN1 相同。
OC2B: 定时/计数器 2 的 B 组比较匹配输出。PD7 可以作为定时/计数器 2 比较匹配的外部输出。此时必须通过 DDD7 将引脚设置为输出。同时，OC2B 也是定时器 2 的 PWM 模式输出引脚。
PCINT23: 引脚电平变化中断 23。

AIN0/OC0A/PCINT22- 端口 D 引脚 6

AN0: 模拟比较器正端输入。通过 DIDR1 寄存器关闭 PD6 引脚的数字输入功能，并关闭端口的上拉电阻，以避免数字端口对模拟电路的干扰。
TK0: 触摸按键输入通道 0。使用方法与 AN0 相同。
OC0A: 定时/计数器 0 的 A 组比较匹配输出。PD6 可以作为定时/计数器 0 比较匹配的外部输出。此时必须通过 DDD6 将引脚设置为输出。同时，OC0A 也是定时器 0 的 PWM 模式输出引脚。
PCINT22: 引脚电平变化中断 22。

T1/OC0B/PCINT21- 端口 D 引脚 5

T1: 定时/计数器 1 的外部计数时钟输入
OC0B: 定时/计数器 0 的 B 组比较匹配输出。PD5 可以作为定时/计数器 0 比较匹配的外部输出。此时必须通过 DDD5 将引脚设置为输出。同时，OC0B 也是定时器 0 的 PWM 模式输出引脚。
PCINT21: 引脚电平变化中断 21。

XCK/T0/PCINT20- 端口 D 引脚 4

XCK: 同步模式 USART 的外部时钟信号
T0: 定时/计数器 0 的外部计数时钟输入
PCINT20: 引脚电平变化中断 20。

INT1/OC2B/PCINT19- 端口 D 引脚 3

INT1: 外部中断输入 1
OC2B: 定时/计数器 2 的 B 组比较匹配输出。PD3 可以作为定时/计数器 2 比较匹配的外部输出。此时必须通过 DDD3 将引脚设置为输出。同时，OC2B 也是定时器 2 的 PWM 模式输出引脚。
PCINT19: 引脚电平变化中断 19。

INT0/PCINT18- 端口 D 引脚 2

INT0: 外部中断输入 0

PCINT18: 引脚电平变化中断 18。

TXD/PCINT17- 端口 D 引脚 1

TXD: 传输数据(USART 数据输出)。USART 发送器使能后，PD1 将被强制为输出，不受 DDD1 的控制。

PCINT17: 引脚电平变化中断 17。

RXD/PCINT16- 端口 D 引脚 0

RXD: 传输数据(USART 数据输入)。USART 接收器使能后，PD0 将被强制为输入，不受 DDD0 的控制。当引脚被 USART 强制为输入后，上拉电阻仍然可以通过 PORTD0 位控制。

PCINT16: 引脚电平变化中断 16。

PD7…PD4 复用控制逻辑表：

信号名称	PD7/AIN1/ PCINT23	PD6/AIN0/ OC0A/PCINT22	PD5/OC0B/ PCINT21	PD4/XCK/ T0/PCINT20
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	OC0AEN&$\overline{\text{OC0AS}}$	OC0B Enable	XCKOEN
PVOV	0	OC0A	OC0B	XCK Output
DIEOE	PCINT23 Enable	PCINT22 Enable	PCINT21 Enable + T1EN	PCINT20 Enable + XCKIEN + T0EN
DIEOV	1	1	1	1
DI	PCINT23 Input	PCINT22 Input	PCINT21 Input T1 Input	PCINT20 Input XCK Input T0 Input
AIO	TK1	TK0	-	-

PD3…PD0 复用控制逻辑表：

信号名称	PD3/OC2B/ INT1/PCINT19	PD2/INT0/ PCINT18	PD1/TXD/ PCINT17	PD0/RXD/ PCINT16
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0&$\overline{\text{PUD}}$
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	OC2B Enable	0	TXEN	0
PVOV	OC2B	0	TXD	0
DIEOE	PCINT19 Enable + INT1 Enable	PCINT18 Enable + INT0 Enable	PCINT17 Enable	PCINT16 Enable + RXEN

DIEOV	1	1	1	1
DI	PCINT19 Input INT1 Input	PCINT18 Input INT0 Input	PCINT17 Input	PCINT16 Input RXD
AIO	-	-	-	-

端口 E 复用功能

引脚	复用功能描述
PE6	VREF (ADC 外部参考电压) TCAP (外接触摸按键控制器滤波电容) PCINT30 (引脚电平变化中断 30)
PE5	CLKO (系统时钟输出) PCINT29 (引脚电平变化中断 29)
PE4	OC0A (定时/计数器 0 比较配置输出 A) PCINT28 (引脚电平变化中断 28)
PE3	ADC7 (ADC 输入通道 7) TK11 (触摸按键输入通道 11) PCINT27 (引脚电平变化中断 27)
PE2	TK7 (触摸按键输入通道 7) SWD (SWD 调试器数据线) PCINT26 (引脚电平变化中断 26)
PE1	ADC6 (ADC 输入通道 6) TK10 (触摸按键输入通道 10) PCINT25 (引脚电平变化中断 25)
PE0	TK6 (触摸按键输入通道 6) SWC (SWD 调试器时钟输入) PCINT24 (引脚电平变化中断 24)

VREF/TCAP/PCINT30- 端口 E 引脚 6

VREF: ADC 外部参考电源输入，用作模拟功能时，需要将对应的数字 I/O 设置为输入，并关闭上拉电阻，以避免数字电路对模拟电路产生干扰。

TCAP: 内部电容触摸电路充放电的外接滤波电容。用户需要根据应用环境，调整电容的实际大小。通常情况下可以接一个 0.1uF 的滤波电容。

PCINT30: 引脚电平变化中断 30

CLKO/PCINT29- 端口 E 引脚 5

CLKO: 此功能与 PB0 的 CLKO 功能相同。可作为 PB0/CLKO 的备用引脚

PCINT29: 引脚电平变化中断 29

OC0A/PCINT28- 端口 E 引脚 4

OC0A: 定时/计数器 0 的 A 组比较匹配输出。PE4 可以作为定时/计数器 0 比较匹配的外部输出。此时必须通过 DDE4 将引脚设置为输出。同时，OC0A 也是定时器 0 的 PWM 模

式输出引脚。

PCINT28: 引脚电平变化中断 28

ADC7/TK11/PCINT27- 端口 E 引脚 3

ADC7: ADC 输入通道 7。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。

TK11: 触摸按键输入通道 11。

PCINT27: 引脚电平变化中断 27

TK7/SWD/PCINT26- 端口 E 引脚 2

TK7: 触摸按键输入通道 7。

SWD: SWD 调试器数据线。系统上电复位后，PE2 默认为 SWD 功能。用户可以通过将 MCUSR 寄存器 SWDD 位置 1 关闭 SWD 调试器功能。SWD 被关闭后，调试功能将不能使用。

PCINT26: 引脚电平变化中断 26

ADC6/TK10/PCINT25- 端口 E 引脚 1

ADC6: ADC 输入通道 6。DIDR 寄存器用于关闭数模复用 I/O 的数字功能，以避免数字部分对模拟电路的影响。具体请参考 ADC 相关章节。

TK10: 触摸按键输入通道 10。

PCINT25: 引脚电平变化中断 25

TK6/SWC/PCINT24- 端口 E 引脚 0

TK6: 触摸按键输入通道 6。

SWC: SWD 调试器时钟线。系统上电复位后，PE0 默认为 SWC 功能。用户可以通过将 MCUSR 寄存器 SWDD 位置 1 关闭 SWD 调试器功能。SWD 被关闭后，调试功能将不能使用。

PCINT24: 引脚电平变化中断 24

PE6...PE4 复用控制逻辑表

信号名称	PE6/VREF/ PCINT30	PE5/CLKO/ PCINT29	PE4/OC0A/ PCINT28
PUOE	$\overline{\text{REFIOEN}}$	0	0
PUOV	0	0	0
DDOE	$\overline{\text{REFIOEN}}$	CLKO Enable 1	0
DDOV	0	1	0
PVOE	$\overline{\text{REFIOEN}}$	CLKO Enable 1	OC0AEN&OC0AS
PVOV	1	CLKO	OC0A
DIEOE	PCINT30 Enable + $\overline{\text{REFIOEN}}$	PCINT29 Enable	PCINT28 Enable
DIEOV	1	1	1

DI	PCINT30 Input	PCINT29 Input	PCINT28 Input
AIO	VREF	-	-

PE3…PE0 复用控制逻辑表：

信号名称	PE3/ADC7/TK11/ PCINT27	PE2/TK7/SWD/ PCINT26	PE1/ADC6/TK10/ PCINT25	PE0/TK6/SWC/ PCINT24
PUOE	0	$\overline{SWDD}$	0	$\overline{SWDD}$
PUOV	0	1	0	1
DDOE	0	$\overline{SWDD}$	0	$\overline{SWDD}$
DDOV	0	$\overline{SWD\ Output}$	0	0
PVOE	0	$\overline{SWDD}$	0	0
PVOV	0	0	0	0
DIEOE	PCINT27 Enable	PCINT26 Enable + $\overline{SWDD}$	PCINT25 Enable	PCINT24 Enable + $\overline{SWDD}$
DIEOV	1	1	1	1
DI	PCINT27 Input	PCINT24 Input SWD Input	PCINT25 Input	PCINT24 Input SWC Input
AIO	ADC7 TK11	TK7	ADC6 TK10	TK6

端口驱动能力

LGT8F88A 的数字 I/O 采用了可编程驱动能力设计，用户可以通过 DSCR 寄存器控制每组端口的驱动能力。除此之外，LGT8F88A 还包含了多达 8 个高扇入能力的 I/O，可扇入高达 80mA 的电流，可直接驱动共阴极七段 LED。下面的表格给出了每组端口不同的引脚驱动能力与 DSCR 寄存器配置直接的关系：

引脚	DSCR		I_{OL} (扇入)			I_{OH} (扇出)			单位
			Min	Typ.	Max	Min	Typ.	Max	
PB[7:0]	DSC1	0	5.1	8.6	12.3	15.3	25.7	36.8	mA
		1	5.6	15.0	28.0	12.7	33.7	63.1	
PC[6:0]	DSC2	0	5.1	8.6	12.3	15.3	25.7	36.8	
		1	5.6	15.0	28.0	12.7	33.7	63.1	
PD[5:0]	DSC3	0	5.1	8.6	12.3	15.3	25.7	36.8	
		1	**50**	**-**	**80**	**12.7**	**33.7**	**63.1**	
PD[7:6]	DSC3	0	5.1	8.6	12.3	15.3	25.7	36.8	
		1	5.6	15.0	28.0	12.7	33.7	63.1	
PE[6] PE[3:0]	DSC4	0	5.1	8.6	12.3	15.3	25.7	36.8	
		1	5.6	15.0	28.0	12.7	33.7	63.1	
PE[5:4]	DSC4	0	5.1	8.6	12.3	15.3	25.7	36.8	
		1	**50**	**-**	**80**	**12.7**	**33.7**	**63.1**	

寄存器定义

MCU 控制寄存器- MCUCR

MCUCR – MCU 控制寄存器								
MCUCR: 0x35(0x55)				默认值: 0x00				
MCUCR	-	-	PUD	-	-	-	IVSEL	IVCE
R/W	-	-	R/W	-	-	-	R/W	R/W
初始值	-	-	0	-	-	-	0	0

位定义		
[0]	IVCE	中断向量选择更改使能位
[1]	IVSEL	中断向量选择位
[2]	-	保留不用
[3]	-	保留不用
[4]	PUD	全局上拉禁止位
[5]	-	保留不用
[6]	-	保留不用
[7]	-	保留不用

端口驱动能力寄存器- DSCR

DSCR – 端口驱动能力寄存器								
DSCR: 0xF1				默认值: 0x00				
DSCR	DSCE	-	-	DSC4	DSC3	DSC2	DSC1	-
R/W	R/W	-	-	R/W	R/W	R/W	R/W	-
初始值	0	-	-	0	0	0	0	-

位定义		
[0]	-	保留不用
[1]	DSC1	端口 B 驱动能力控制位 0 – 8mA 1 – 24mA
[2]	DSC2	端口 C 驱动能力控制位 0 – 8mA 1 – 24mA
[3]	DSC3	端口 D 驱动能力控制位 0 – 8mA 1 – PD5 ~ PD0 为 80mA 扇入，24mA 扇出 1 – PD7 ~ PD6 为 24mA
[4]	DSC4	端口 E 驱动能力控制位 0 – 8mA 1 – PE5 ~ PE4 为 80mA 扇入，24mA 扇出 1 – PE6，PE3 ~ PE0 为 24mA
[5]	-	保留不用

[6]	-	保留不用
[7]	DSCE	端口驱动寄存器更改使能位。如需更改端口驱动能力配置，必须先将此位写，然后在 6 个周期内更改 DSCR 其他位的值。6 个周期后 DSCE 自动清零。

端口 B 输出数据寄存器- PORTB

PORTB – 端口 B 输出数据寄存器								
PORTB: 0x05(0x25)				默认值: 0x00				
PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

[0]	PORTB0	端口 B 输出第 0 位
[1]	PORTB1	端口 B 输出第 1 位
[2]	PORTB2	端口 B 输出第 2 位
[3]	PORTB3	端口 B 输出第 3 位
[4]	PORTB4	端口 B 输出第 4 位
[5]	PORTB5	端口 B 输出第 5 位
[6]	PORTB6	端口 B 输出第 6 位
[7]	PORTB7	端口 B 输出第 7 位

端口 B 方向寄存器- DDRB

DDRB – 端口 B 方向寄存器								
DDRB: 0x04(0x24)				默认值: 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
初始值	0	0	0	0	0	0	0	0

[0]	DDB0	PB0 方向控制位
[1]	DDB1	PB1 方向控制位
[2]	DDB2	PB2 方向控制位
[3]	DDB3	PB3 方向控制位
[4]	DDB4	PB4 方向控制位
[5]	DDB5	PB5 方向控制位
[6]	DDB6	PB6 方向控制位
[7]	DDB7	PB7 方向控制位

端口 B 输入数据寄存器- PINB

PINB – 端口 B 输入数据存器								
PINB: 0x03(0x23)				默认值: 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
初始值	0	0	0	0	0	0	0	0
[0]	PINB0	PB0 端口数据						
[1]	PINB1	PB1 端口数据						
[2]	PINB2	PB2 端口数据						
[3]	PINB3	PB3 端口数据						
[4]	PINB4	PB4 端口数据						
[5]	PINB5	PB5 端口数据						
[6]	PINB6	PB6 端口数据						
[7]	PINB7	PB7 端口数据						

端口 C 输出数据寄存器- PORTC

PORTC – 端口 C 输出数据寄存器								
PORTC: 0x08(0x28)				默认值: 0x00				
PORTC	-	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0
[0]	PORTC0	端口 C 输出第 0 位						
[1]	PORTC1	端口 C 输出第 1 位						
[2]	PORTC2	端口 C 输出第 2 位						
[3]	PORTC3	端口 C 输出第 3 位						
[4]	PORTC4	端口 C 输出第 4 位						
[5]	PORTC5	端口 C 输出第 5 位						
[6]	PORTC6	端口 C 输出第 6 位						
[7]	-	保留不用						

端口 C 方向寄存器- DDRC

DDRC – 端口 C 方向寄存器								
DDRC: 0x07(0x27)				默认值: 0x00				
DDRC	-	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

初始值	-	0	0	0	0	0	0	0
[0]	DDC0	PC0 方向控制位						
[1]	DDC1	PC1 方向控制位						
[2]	DDC2	PC2 方向控制位						
[3]	DDC3	PC3 方向控制位						
[4]	DDC4	PC4 方向控制位						
[5]	DDC5	PC5 方向控制位						
[6]	DDC6	PC6 方向控制位						
[7]	-	保留不用						

端口 C 输入数据寄存器- PINC

PINC – 端口 C 输入数据存器								
PINB: 0x06(0x26)					默认值: 0x00			
PINC	-	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0
[0]	PINC0	PC0 端口数据						
[1]	PINC1	PC1 端口数据						
[2]	PINC2	PC2 端口数据						
[3]	PINC3	PC3 端口数据						
[4]	PINC4	PC4 端口数据						
[5]	PINC5	PC5 端口数据						
[6]	PINC6	PC6 端口数据						
[7]	-	保留不用						

端口 D 输出数据寄存器- PORTD

PORTD – 端口 D 输出数据寄存器								
PORTD: 0x0B(0x2B)					默认值: 0x00			
PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0
[0]	PORTD0	端口 D 输出第 0 位						
[1]	PORTD1	端口 D 输出第 1 位						
[2]	PORTD2	端口 D 输出第 2 位						
[3]	PORTD3	端口 D 输出第 3 位						

[4]	PORTD4	端口 D 输出第 4 位
[5]	PORTD5	端口 D 输出第 5 位
[6]	PORTD6	端口 D 输出第 6 位
[7]	PORTD7	端口 D 输出第 7 位

端口 D 方向寄存器- DDRD

DDRD – 端口 D 方向寄存器								
DDRD: 0x0A(0x2A)				默认值: 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
初始值	0	0	0	0	0	0	0	0

[0]	DDD0	PD0 方向控制位
[1]	DDD1	PD1 方向控制位
[2]	DDD2	PD2 方向控制位
[3]	DDD3	PD3 方向控制位
[4]	DDD4	PD4 方向控制位
[5]	DDD5	PD5 方向控制位
[6]	DDD6	PD6 方向控制位
[7]	DDD7	PD7 方向控制位

端口 D 输入数据寄存器- PIND

PIND – 端口 D 输入数据存器								
PIND: 0x09(0x29)				默认值: 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
初始值	0	0	0	0	0	0	0	0

[0]	PIND0	PD0 端口数据
[1]	PIND1	PD1 端口数据
[2]	PIND2	PD2 端口数据
[3]	PIND3	PD3 端口数据
[4]	PIND4	PD4 端口数据
[5]	PIND5	PD5 端口数据
[6]	PIND6	PD6 端口数据
[7]	PIND7	PD7 端口数据

端口 E 输出数据寄存器- PORTE

PORTE – 端口 E 输出数据寄存器								
PORTE: 0xA9					默认值: 0x00			
PORTE	-	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0

[0]	PORTE0	端口 E 输出第 0 位
[1]	PORTE1	端口 E 输出第 1 位
[2]	PORTE2	端口 E 输出第 2 位
[3]	PORTE3	端口 E 输出第 3 位
[4]	PORTE4	端口 E 输出第 4 位
[5]	PORTE5	端口 E 输出第 5 位
[6]	PORTE6	端口 E 输出第 6 位
[7]	-	保留不用

端口 E 方向寄存器- DDRE

DDRE – 端口 E 方向寄存器								
DDRE: 0xA8					默认值: 0x00			
DDRE	-	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0

[0]	DDE0	PE0 方向控制位
[1]	DDE1	PE1 方向控制位
[2]	DDE2	PE2 方向控制位
[3]	DDE3	PE3 方向控制位
[4]	DDE4	PE4 方向控制位
[5]	DDE5	PE5 方向控制位
[6]	DDE6	PE6 方向控制位
[7]	-	保留不用

端口 E 输入数据寄存器- PINE

PINE – 端口 E 输入数据存器								
PINE: 0xA7					默认值: 0x00			
PINE	-	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	-	0	0	0	0	0	0	0

[0]	PINE0	PE0 端口数据
[1]	PINE1	PE1 端口数据
[2]	PINE2	PE2 端口数据
[3]	PINE3	PE3 端口数据
[4]	PINE4	PE4 端口数据
[5]	PINE5	PE5 端口数据
[6]	PINE6	PE6 端口数据
[7]	-	保留不用

引脚电平变化中断

- 30 个引脚电平变化中断源
- 4 个中断入口

综述

引脚电平变化中断由 PBn，PCn，PDn 和 PEn 引脚触发。只要引脚电平变化中断被使能，即使这些引脚配置为输出也能触发中断。这可以用来产生软件中断。

任何一个使能的 PBn 引脚翻转都会触发引脚电平中断 PCI0，使能的 PCn 引脚翻转将触发 PCI1，使能的 PDn 引脚翻转将触发 PCI2，使能的 PEn 引脚翻转将触发 PCI3。各个引脚变化中断的使能分别由 PCMSK0，PCMSK1，PCMSK2 和 PCMSK3 寄存器来控制。所有的引脚电平变化中断都是异步检测的，可用作某些睡眠模式下的唤醒源。

寄存器定义

Pin Change Interrupt 寄存器列表

寄存器	地址	默认值	描述
PCICR	0x68	0x00	引脚改变中断控制寄存器
PCIFR	0x3B	0x00	引脚改变中断标志寄存器
PCMSK0	0x6B	0x00	引脚改变中断屏蔽寄存器 0
PCMSK1	0x6C	0x00	引脚改变中断屏蔽寄存器 1
PCMSK2	0x6D	0x00	引脚改变中断屏蔽寄存器 2
PCMSK3	0x73	0x00	引脚改变中断屏蔽寄存器 3

PCICR – 引脚改变中断控制寄存器

PCICR – 引脚改变中断控制寄存器								
地址: 0x68					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	PCIE3	PCIE2	PCIE1	PCIE0
R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:4	-	保留。
3	PCIE3	引脚改变中断使能控制位 3。 当设置 PCIE3 位为“1”且全局中断使能时，引脚改变中断 3 被使能。任何一个使能的 PEn 引脚的电平变化都会产生 PCI3 中断。PEn 引脚中断的使能可分别由 PCMSK3 寄存器来控制。 当设置 PCIE3 位为“0”时，引脚改变中断 3 被禁止。
2	PCIE2	引脚改变中断使能控制位 2。 当设置 PCIE2 位为“1”且全局中断使能时，引脚改变中断 2 被使能。任何一个使能的 PDn 引脚的电平变化都会产生 PCI2 中断。PDn 引脚中断的使能可分别由 PCMSK2 寄存器来控制。 当设置 PCIE2 位为“0”时，引脚改变中断 2 被禁止。
1	PCIE1	引脚改变中断使能控制位 1。 当设置 PCIE1 位为“1”且全局中断使能时，引脚改变中断 1 被使能。任何一个使能的 PCn 引脚的电平变化都会产生 PCI1 中断。PCn 引脚中断的使能可分别由 PCMSK1 寄存器来控制。 当设置 PCIE1 位为“0”时，引脚改变中断 1 被禁止。
0	PCIE0	引脚改变中断使能控制位 0。 当设置 PCIE0 位为“1”且全局中断使能时，引脚改变中断 0 被使能。任何一个使能的 PBn 引脚的电平变化都会产生 PCI0 中断。PBn 引脚中断的使能可分别由 PCMSK0 寄存器来控制。 当设置 PCIE0 位为“0”时，引脚改变中断 0 被禁止。

PCIFR – 引脚改变中断标志寄存器

PCIFR – 引脚改变中断标志寄存器								
地址: 0x3B					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	PCIF3	PCIF2	PCIF1	PCIF0
R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述

7:4	-	保留。
3	PCIF3	引脚改变中断标志位 3。 任何一个使能的 PEn 引脚的电平变化都会置位 PCIF3。当 PCIE3 和全局中断均置位时，MCU 将会跳转至 PCI3 中断入口地址。PEn 引脚中断的使能可分别由 PCMSK3 寄存器来控制。 执行中断服务程序或往 PCIF3 位写“1”都会清零 PCIF3 位。
2	PCIF2	引脚改变中断标志位 2。 任何一个使能的 PDn 引脚的电平变化都会置位 PCIF2。当 PCIE2 和全局中断均置位时，MCU 将会跳转至 PCI2 中断入口地址。PDn 引脚中断的使能可分别由 PCMSK2 寄存器来控制。 执行中断服务程序或往 PCIF2 位写“1”都会清零 PCIF2 位。
1	PCIF1	引脚改变中断标志位 1。 任何一个使能的 PCn 引脚的电平变化都会置位 PCIF1。当 PCIE1 和全局中断均置位时，MCU 将会跳转至 PCI1 中断入口地址。PCn 引脚中断的使能可分别由 PCMSK1 寄存器来控制。 执行中断服务程序或往 PCIF1 位写“1”都会清零 PCIF1 位。
0	PCIF0	引脚改变中断标志位 0。 任何一个使能的 PBn 引脚的电平变化都会置位 PCIF0。当 PCIE0 和全局中断均置位时，MCU 将会跳转至 PCI0 中断入口地址。PBn 引脚中断的使能可分别由 PCMSK0 寄存器来控制。 执行中断服务程序或往 PCIF0 位写“1”都会清零 PCIF0 位。

PCMSK0 – 引脚改变中断屏蔽寄存器 0

PCMSK0 – 引脚改变屏蔽寄存器 0								
地址: 0x6B						默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	PCINT7	引脚改变使能屏蔽位 7。 当设置 PCINT7 位为“1”时，PB7 引脚电平改变中断被使能。PB7 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT7 位为“0”时，PB7 引脚电平改变中断被禁止。
6	PCINT6	引脚改变使能屏蔽位 6。 当设置 PCINT6 位为“1”时，PB6 引脚电平改变中断被使能。PB6 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT6 位为“0”时，PB6 引脚电平改变中断被禁止。
5	PCINT5	引脚改变使能屏蔽位 5。 当设置 PCINT5 位为“1”时，PB5 引脚电平改变中断被使能。PB5 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。

		当设置 PCINT5 位为“0”时，PB5 引脚电平改变中断被禁止。
4	PCINT4	引脚改变使能屏蔽位 4。 当设置 PCINT4 位为“1”时，PB4 引脚电平改变中断被使能。PB4 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT4 位为“0”时，PB4 引脚电平改变中断被禁止。
3	PCINT3	引脚改变使能屏蔽位 3。 当设置 PCINT3 位为“1”时，PB3 引脚电平改变中断被使能。PB3 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT3 位为“0”时，PB3 引脚电平改变中断被禁止。
2	PCINT2	引脚改变使能屏蔽位 2。 当设置 PCINT2 位为“1”时，PB2 引脚电平改变中断被使能。PB2 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT2 位为“0”时，PB2 引脚电平改变中断被禁止。
1	PCINT1	引脚改变使能屏蔽位 1。 当设置 PCINT1 位为“1”时，PB1 引脚电平改变中断被使能。PB1 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT1 位为“0”时，PB1 引脚电平改变中断被禁止。
0	PCINT0	引脚改变使能屏蔽位 0。 当设置 PCINT0 位为“1”时，PB0 引脚电平改变中断被使能。PB0 引脚上的电平改变将置位 PCIF0，若 PCIE0 位和全局中断置位，将会产生 PCI0 中断。 当设置 PCINT0 位为“0”时，PB0 引脚电平改变中断被禁止。

PCMSK1 – 引脚改变中断屏蔽寄存器 1

PCMSK1 – 引脚改变屏蔽寄存器 1								
地址: 0x6C						默认值: 0x00		
Bit	7	6	5	4	3	2	1	0
Name	-	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
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	描述
7	-	保留。
6	PCINT14	引脚改变使能屏蔽位 14。 当设置 PCINT14 位为“1”时，PC6 引脚电平改变中断被使能。PC6 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT14 位为“0”时，PC6 引脚电平改变中断被禁止。
5	PCINT13	引脚改变使能屏蔽位 13。 当设置 PCINT13 位为“1”时，PC5 引脚电平改变中断被使能。PC5 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT13 位为“0”时，PC5 引脚电平改变中断被禁止。
4	PCINT12	引脚改变使能屏蔽位 12。

		当设置 PCINT12 位为“1”时，PC4 引脚电平改变中断被使能。PC4 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT12 位为“0”时，PC4 引脚电平改变中断被禁止。
3	PCINT11	引脚改变使能屏蔽位 11。 当设置 PCINT11 位为“1”时，PC3 引脚电平改变中断被使能。PC3 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT11 位为“0”时，PC3 引脚电平改变中断被禁止。
2	PCINT10	引脚改变使能屏蔽位 2。 当设置 PCINT10 位为“1”时，PC2 引脚电平改变中断被使能。PC2 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT10 位为“0”时，PC2 引脚电平改变中断被禁止。
1	PCINT9	引脚改变使能屏蔽位 1。 当设置 PCINT9 位为“1”时，PC1 引脚电平改变中断被使能。PC1 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT9 位为“0”时，PC1 引脚电平改变中断被禁止。
0	PCINT8	引脚改变使能屏蔽位 0。 当设置 PCINT8 位为“1”时，PC0 引脚电平改变中断被使能。PC0 引脚上的电平改变将置位 PCIF1，若 PCIE1 位和全局中断置位，将会产生 PCI1 中断。 当设置 PCINT8 位为“0”时，PC0 引脚电平改变中断被禁止。

PCMSK2 – 引脚改变中断屏蔽寄存器 2

PCMSK2 – 引脚改变屏蔽寄存器 2								
地址: 0x6D					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	PCINT23	引脚改变使能屏蔽位 23。 当设置 PCINT23 位为“1”时，PD7 引脚电平改变中断被使能。PD7 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT23 位为“0”时，PD7 引脚电平改变中断被禁止。
6	PCINT22	引脚改变使能屏蔽位 6。 当设置 PCINT22 位为“1”时，PD6 引脚电平改变中断被使能。PD6 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT22 位为“0”时，PD6 引脚电平改变中断被禁止。
5	PCINT21	引脚改变使能屏蔽位 21。 当设置 PCINT21 位为“1”时，PD5 引脚电平改变中断被使能。PD5 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT21 位为“0”时，PD5 引脚电平改变中断被禁止。
4	PCINT20	引脚改变使能屏蔽位 20。

		当设置 PCINT20 位为“1”时，PD4 引脚电平改变中断被使能。PD4 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT20 位为“0”时，PD4 引脚电平改变中断被禁止。
3	PCINT19	引脚改变使能屏蔽位 19。 当设置 PCINT19 位为“1”时，PD3 引脚电平改变中断被使能。PD3 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT19 位为“0”时，PD3 引脚电平改变中断被禁止。
2	PCINT18	引脚改变使能屏蔽位 18。 当设置 PCINT18 位为“1”时，PD2 引脚电平改变中断被使能。PD2 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT18 位为“0”时，PD2 引脚电平改变中断被禁止。
1	PCINT17	引脚改变使能屏蔽位 17。 当设置 PCINT17 位为“1”时，PD1 引脚电平改变中断被使能。PD1 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT17 位为“0”时，PD1 引脚电平改变中断被禁止。
0	PCINT16	引脚改变使能屏蔽位 16。 当设置 PCINT16 位为“1”时，PD0 引脚电平改变中断被使能。PD0 引脚上的电平改变将置位 PCIF2，若 PCIE2 位和全局中断置位，将会产生 PCI2 中断。 当设置 PCINT16 位为“0”时，PD0 引脚电平改变中断被禁止。

PCMSK3 – 引脚改变中断屏蔽寄存器 3

PCMSK3 – 引脚改变屏蔽寄存器 3								
地址: 0x73					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
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	描述
7	-	保留。
6	PCINT30	引脚改变使能屏蔽位 30。 当设置 PCINT30 位为“1”时，PE6 引脚电平改变中断被使能。PE6 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT30 位为“0”时，PE6 引脚电平改变中断被禁止。
5	PCINT29	引脚改变使能屏蔽位 39。 当设置 PCINT29 位为“1”时，PE5 引脚电平改变中断被使能。PE5 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT29 位为“0”时，PE5 引脚电平改变中断被禁止。
4	PCINT28	引脚改变使能屏蔽位 28。 当设置 PCINT28 位为“1”时，PE4 引脚电平改变中断被使能。PE4 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT28 位为“0”时，PE4 引脚电平改变中断被禁止。

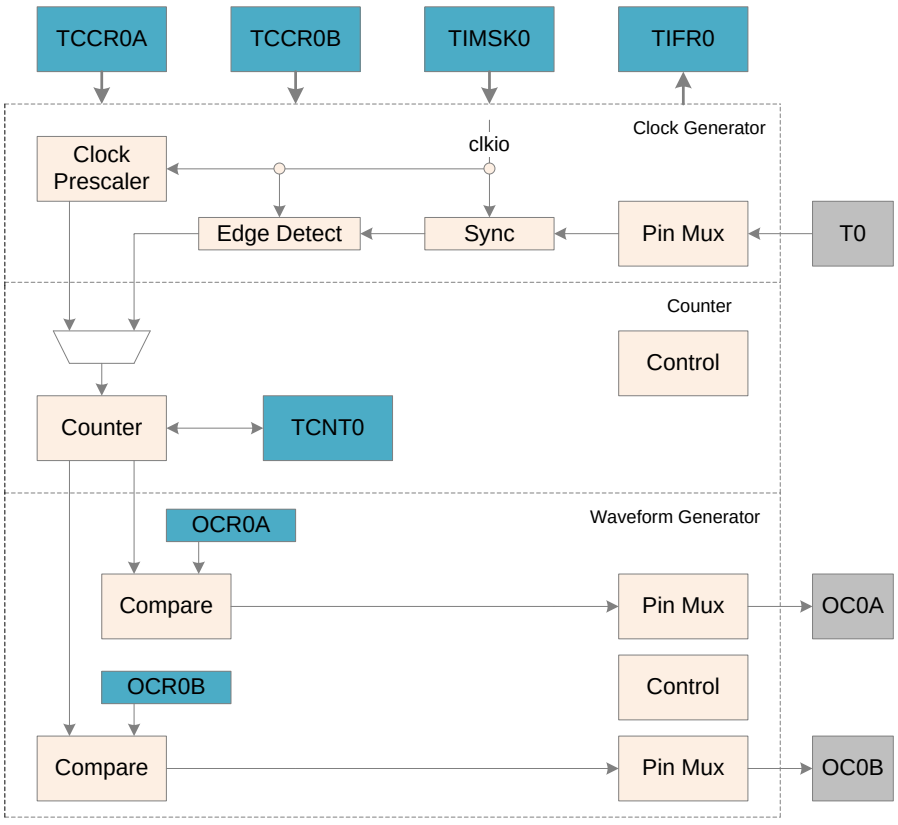
3	PCINT27	引脚改变使能屏蔽位 27。 当设置 PCINT27 位为“1”时，PE3 引脚电平改变中断被使能。PE3 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT27 位为“0”时，PE3 引脚电平改变中断被禁止。
2	PCINT26	引脚改变使能屏蔽位 26。 当设置 PCINT26 位为“1”时，PE2 引脚电平改变中断被使能。PE2 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT26 位为“0”时，PE2 引脚电平改变中断被禁止。
1	PCINT25	引脚改变使能屏蔽位 25。 当设置 PCINT25 位为“1”时，PE1 引脚电平改变中断被使能。PE1 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT25 位为“0”时，PE1 引脚电平改变中断被禁止。
0	PCINT24	引脚改变使能屏蔽位 24。 当设置 PCINT24 位为“1”时，PE0 引脚电平改变中断被使能。PE0 引脚上的电平改变将置位 PCIF3，若 PCIE3 位和全局中断置位，将会产生 PCI3 中断。 当设置 PCINT24 位为“0”时，PE0 引脚电平改变中断被禁止。

8 位定时/计数器 0

- 8 位计数器
- 两个独立的比较单元
- 比较匹配发生时自动清零计数器并自动加载
- 无干扰脉冲的相位修正的 PWM 输出
- 频率发生器
- 外部事件计数器
- 10 位的时钟预分频器
- 溢出和比较匹配中断

概述

TC0 是一个通用 8 位定时计数器模块，支持 PWM 输出，可以精确地产生波形。TC0 包含 1 个 8 位计数器，波形产生模式控制单元和 2 个输出比较单元。波形产生模式控制单元控制着计数器的工作模式和比较输出波形的产生。根据不同的工作模式，计数器对每一个计数时钟 Clkt0 实现清零、加一或减一操作。Clkt0 可以由内部时钟源或外部时钟源产生。当计数器的计数值 TCNT0 到达最大值（等于极大值 0xFF 或输出比较寄存器 OCR0A，定义为 TOP，定义极大值为 MAX 以示区别）时，计数器会进行清零或减一操作。当计数器的计数值 TCNT0 到达最小值（等于 0x00，定义为 BOTTOM）时，计数器会进行加一操作。当计数器的计数值 TCNT0 到达 OCR0A/OCR0B 时，也被称为发生比较匹配时，会清零或置位输出比较信号 OC0A/OC0B，来产生 PWM 波形。



TC0 结构图

工作模式

定时计数器 0 有四种不同的工作模式，包括普通模式（Normal），比较匹配时清零（CTC）模式，快速脉冲宽度调制（FPWM）模式和相位修正脉冲宽度调制（PCPWM）模式，由波形产生模式控制位 WGM0[2:0]来选择。下面具体来描述这四种模式。由于有两个独立的输出比较单元，分别用“A”和“B”来表示，用小写的“x”来表示这两个输出比较单元通道。

普通模式

普通模式是定时计数器最简单的工作模式，此时波形产生模式控制位 WGM0[2:0]=0，计数的最大值 TOP 为 MAX（0xFF）。在这种模式下，计数方式为每一个计数时钟加一递增，当计数器到达 TOP 溢出后就回到 BOTTOM 重新开始累加。在计数值 TCNT0 变成零的同一个计数时钟里置位定时计数器溢出标志 TOV0。这种模式下 TOV0 标志就像是第 9 计数位，只是只会被置位不会被清零。溢出中断服务程序会自动清除 TOV0 标志，软件可以用它来提高定时计数器的分辨率。普通模式下没有特殊情形需要考虑，可以随时写入新的计数值。

设置 OC0x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0x 的波形。当 COM0x=1 时，发生比较匹配时会翻转 OC0x 信号，这种情况下波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

输出比较单元可以用来产生中断，但是在普通模式下不推荐使用中断，这样会占用太多 CPU 的时间。

CTC 模式

设置 WGM0[2:0]=2 时，定时计数器 0 进入 CTC 模式，计数的最大值 TOP 为 OCR0A。在这个模式下，计数方式为每一个计数时钟加一递增，当计数器的数值 TCNT0 等于 TOP 时计数器清零。OCR0A 定义了计数的最大值，亦即计数器的分辨率。这个模式使得用户可以很容易的控制比较匹配输出的频率，也简化了外部事件计数的操作。

当计数器到达计数的最大值时，输出比较匹配标志 OCF0 被置位，相应的中断使能置位时将会产生中断。在中断服务程序里可以更新 OCR0A 寄存器即计数的最大值。在这个模式下 OCR0A 没有使用双缓冲，在计数器以无预分频器或很低的预分频器工作下将最大值更新为接近最小值的时候要小心。如果写入 OCR0A 的数值小于当时的 TCNT0 值时，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到 TOP，然后再从 BOTTOM 开始计数到 OCR0A 值。和普通模式一样，计数值回到 BOTTOM 的计数时钟里置位 TOV0 标志。

设置 OC0x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0x 的波形。当 COM0x=1 时，发生比较匹配时会翻转 OC0x 信号，这种情况下波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

从公式可以看出，当设置 OCR0A 为 0x0 且无预分频器时，可以获得最大频率为 $f_{sys}/2$ 的输出波形。

快速 PWM 模式

设置 WGM0[2:0]=3 或 7 时，定时计数器 0 进入快速 PWM 模式，可以用来产生高频的 PWM 波形，计数最大值 TOP 分别为 MAX（0xFF）或 OCR0x。快速 PWM 模式和其他 PWM 模

式不同在于它是单向操作。计数器从最小值 0x00 累加到 TOP 后又回到 BOTTOM 重新计数。当计数值 TCNT0 到达 OCR0x 或 BOTTOM 时，输出比较信号 OC0x 会被置位或清零，取决于比较输出模式 COM0x 的设置，详情见寄存器描述。由于采用单向操作，快速 PWM 模式的操作频率是采用双向操作的相位修正 PWM 模式的两倍。高频特性使得快速 PWM 模式适用于功率调节，整流以及 DAC 应用。高频信号可以减小外部元器件（电感电容等）的尺寸，从而降低系统成本。

当计数值到达最大值时，定时计数器溢出标志 TOV0 将会被置位，并把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR0x 寄存器。

设置 OC0x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0x 的波形。波形的频率可用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

当 TCNT0 和 OCR0x 发生比较匹配时，波形产生器就置位（清零）OC0x 信号，当 TCNT0 被清零时，波形产生器就清零（置位）OC0x 信号，以此来产生 PWM 波。由此 OCR0x 的极值将会产生特殊的 PWM 波形。当 OCR0x 设置为 0x00 时，输出的 PWM 为每(1+TOP)个计数时钟里有一个窄的尖峰脉冲。当 OCR0x 设置为最大值时，输出的波形为持续的高电平或低电平。

相位修正 PWM 模式

当设置 WGM0[2:0]=1 或 5 时，定时计数器 0 进入相位修正 PWM 模式，计数的最大值 TOP 分别为 MAX（0xFF）或 OCR0A。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT0 与 OCR0x 匹配时，输出比较信号 OC0x 将会被清零或置位，取决于比较输出模式 COM0x 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV0 标志，当计数到达 TOP 时把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR0x 寄存器。

设置 OC0x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0x 的波形。波形的频率可用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT0 与 OCR0x 匹配时，波形产生器就清零（置位）OC0x 信号。在递减计数过程中，当 TCNT0 与 OCR0x 匹配时，波形产生器就置位（清零）OC0x 信号。由此 OCR0x 的极值会产生特殊的 PWM 波。当 OCR0x 设置为最大值或最小值时，OC0x 信号输出会一直保持低电平或高电平。

为了保证输出 PWM 波在最小值两侧的对称性，在没有发生比较匹配时，有两种情况下也会翻转 OC0x 信号。第一种情况是，当 OCR0x 的值由最大值 0xFF 改变为其他数据时。当 OCR0x 为最大值，计数值达到最大时，OC0x 的输出与前面降序计数时比较匹配的结果相同，即保持 OC0x 不变。此时会更新比较值为新的 OCR0x 的值（非 0xFF），OC0x 的值会一直保持，直到升序计数时发生比较匹配而翻转。此时 OC0x 信号并不以最小值为中心对称，因此需要在 TCNT0 到达最大值时翻转 OC0x 信号，此即没有发生比较匹配时翻转 OC0x 信号的第一种情况。第二种情况是，当 TCNT0 从比 OCR0x 高的值开始计数时，因而会丢失一次比较匹配，从而引起不对称情形的产生。同样需要翻转 OC0x 信号去实现最小值两侧的对称性。

寄存器定义

TC0 寄存器列表

寄存器	地址	默认值	描述
TCCR0A	0x44	0x00	TC0 控制寄存器 A
TCCR0B	0x45	0x00	TC0 控制寄存器 B
TCNT0	0x46	0x00	TC0 计数值寄存器
OCR0A	0x47	0x00	TC0 输出比较寄存器 A
OCR0B	0x48	0x00	TC0 输出比较寄存器 B
TIMSK0	0x6E	0x00	定时计数器 0 中断屏蔽寄存器
TIFR0	0x35	0x00	定时计数器 0 中断标志寄存器

TC0 控制寄存器 A- TCCR0A

TCCR0A –TC0 控制寄存器 A								
地址: 0x44					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
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	描述
7	COM0A1	TC0 比较匹配 A 输出模式控制高位。 COM0A1 和 COM0A0 一起组成比较输出模式控制 COM0A[1:0]，用来控制 OC0A 的输出波形。如果 COM0A 的 1 位或者 2 位都置位，输出比较波形占据着 OC0A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM0A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
6	COM0A0	TC0 比较匹配 A 输出模式控制低位。 COM0A0 和 COM0A1 一起组成比较输出模式控制 COM0A[1:0] ，用来控制 OC0A 的输出波形。如果 COM0A 的 1 位或者 2 位都置位，输出比较波形占据着 OC0A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM0A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
5	COM0B1	TC0 比较匹配 B 输出模式控制高位。 COM0B1 和 COM0B0 一起组成比较输出模式控制 COM0B[1:0]，用来控制 OC0B 的输出波形。如果 COM0B 的 1 位或者 2 位都置位，输出比较波形占据着 OC0B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM0B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
4	COM0B0	TC0 比较匹配 B 输出模式控制低位。

		COM0B0 和 COM0B1 一起组成比较输出模式控制 COM0B[1:0] ，用来控制 OC0B 的输出波形。如果 COM0B 的 1 位或者 2 位都置位，输出比较波形占据着 OC0B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM0B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
3:2	-	保留。
1	WGM01	TC0 波形产生模式控制中位。 WGM01 和 WGM00，WGM02 一起组成波形产生模式控制 WGM0[2:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
0	WGM00	TC0 波形产生模式控制低位。 WGM00 和 WGM01，WGM02 一起组成波形产生模式控制 WGM0[2:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

TC0 控制寄存器 B- TCCR0B

TCCR0B –TC0 控制寄存器 B								
地址: 0x45					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	FOC0A	FOC0B	OC0AS	-	WGM02	CS02	CS01	CS00
R/W	W	W	W/R	-	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	FOC0A	TC0 强制输出比较 A 控制位。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC0A 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF0A 标志，也不会重载或清零定时器，但是输出引脚 OC0A 将被按照 COM0A 的设置相应的更新，就跟真的发生了比较匹配一样。 读取 FOC0A 的返回值一直为零。
6	FOC0B	TC0 强制输出比较 B 控制位。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC0B 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF0B 标志，也不会重载或清零定时器，但是输出引脚 OC0B 将被按照 COM0B 的设置相应的更新，就跟真的发生了比较匹配一样。 读取 FOC0B 的返回值一直为零。
5	OC0AS	OC0A 输出端口选择控制位。当设置 OC0AS 位为“0”时，OC0A 的波形从引脚 PD6 输出；当设置 OC0AS 位为“1”时，OC0A 的波形从引脚 PE4 输出（QFP32 封装下有效）。
4	-	保留。
3	WGM02	TC0 波形产生模式控制高位。 WGM02 和 WGM00，WGM01 一起组成波形产生模式控制 WGM0[2:0]，控

<table>
<tr><td></td><td></td><td colspan="2">制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。</td></tr>
<tr><td>2</td><td>CS02</td><td colspan="2">TC0 时钟选择控制高位。
用于选择定时计数器 0 的时钟源。</td></tr>
<tr><td>1</td><td>CS01</td><td colspan="2">TC0 时钟选择控制中位。
用于选择定时计数器 0 的时钟源。</td></tr>
<tr><td rowspan="10">0</td><td rowspan="10">CS00</td><td colspan="2">TC0 时钟选择控制低位。
用于选择定时计数器 0 的时钟源。</td></tr>
<tr><td>CS0[2:0]</td><td>描述</td></tr>
<tr><td>0</td><td>无时钟源，停止计数</td></tr>
<tr><td>1</td><td>clk_{sys}</td></tr>
<tr><td>2</td><td>clk_{sys}/8，来自预分频器</td></tr>
<tr><td>3</td><td>clk_{sys}/64，来自预分频器</td></tr>
<tr><td>4</td><td>clk_{sys}/256，来自预分频器</td></tr>
<tr><td>5</td><td>clk_{sys}/1024，来自预分频器</td></tr>
<tr><td>6</td><td>外部时钟 T0 引脚，下降沿触发</td></tr>
<tr><td>7</td><td>外部时钟 T0 引脚，上升沿触发</td></tr>
</table>

下表为非 PWM 模式（即普通模式和 CTC 模式）下，比较输出模式对输出比较波形的控制。

COM0x[1:0]	描述
0	OC0x 断开，通用 IO 口操作
1	比较匹配时翻转 OC0x 信号
2	比较匹配时清零 OC0x 信号
3	比较匹配时置位 OC0x 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

COM0x[1:0]	描述
0	OC0x 断开，通用 IO 口操作
1	保留
2	比较匹配时清零 OC0x 信号，最大值匹配时置位 OC0x 信号
3	比较匹配时置位 OC0x 信号，最大值匹配时清零 OC0x 信号

下表为相位修正模式下比较输出模式对输出比较波形的控制。

COM0x[1:0]	描述
0	OC0x 断开，通用 IO 口操作
1	保留
2	升序计数下比较匹配时清零 OC0x 信号，降序计数下比较匹配时置位 OC0x 信号
3	升序计数下比较匹配时置位 OC0x 信号，降序计数下比较匹配时清零 OC0x 信号

下表为波形产生模式控制。

WGM0[2:0]	工作模式	TOP 值	更新 OCR0X 时刻	置位 TOV0 时刻
0	Normal	0xFF	立即	MAX
1	PCPWM	0xFF	TOP	BOTTOM
2	CTC	OCR0A	立即	MAX
3	FPWM	0xFF	TOP	MAX
4	保留	-	-	-
5	PCPWM	OCR0A	TOP	BOTTOM
6	保留	-	-	-
7	FPWM	OCR0A	TOP	TOP

TC0 计数值寄存器- TCNT0

TCNT0 –TC0 计数值寄存器								
地址: 0x46					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	TCNT0	TC0 计数值寄存器。 通过 TCNT0 寄存器可以直接对计数器的 8 为计数值进行读写访问。 CPU 对 TCNT0 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT0 寄存器的值与 OCR0 的值一致而不会引发中断。 如果写入 TCNT0 的数值等于或绕过 OCR0 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT0。CPU 写计数器比清零或加减操作的优先级高。

TC0 输出比较寄存器 A- OCR0A

OCR0A – TC0 输出比较寄存器 A								
地址: 0x47					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述

7:0	OCR0A	TC0 输出比较寄存器。 OCR0A 包含一个 8 位的数据，不间断地与计数器数值 TCNT0 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC0A 引脚上产生波形。 当使用 PWM 模式时，OCR0A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR0A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR0A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR0A 本身。

TC0 输出比较寄存器 B- OCR0B

OCR0B – TC0 输出比较寄存器 B								
地址: 0x20E1					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	OCR0B	TC0 输出比较 B 寄存器。 OCR0B 包含一个 8 位的数据，不间断地与计数器数值 TCNT0 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC0B 引脚上产生波形。 当使用 PWM 模式时，OCR0B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR0B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR0B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR0B 本身。

TC0 中断屏蔽寄存器- TIMSK0

TIMSK0 – TC0 中断屏蔽寄存器								
地址: 0x6E					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:3		保留。
2	OCIE0B	TC0 输出比较 B 匹配中断使能位。

		当 OCIE0B 位为“1”，且全局中断置位，TC0 输出比较 B 匹配中断使能。当比较匹配发生时，即 TIFR0 中 OCF0B 位被置位时，中断产生。 当 OCIE0B 位为“0”时，TC0 输出比较 B 匹配中断被禁止。
1	OCIE0A	TC0 输出比较 A 匹配中断使能位。 当 OCIE0A 位为“1”，且全局中断置位，TC0 输出比较 A 匹配中断使能。当比较匹配发生时，即 TIFR0 中 OCF0A 位被置位时，中断产生。 当 OCIE0A 位为“0”时，TC0 输出比较 A 匹配中断被禁止。
0	TOIE0	TC0 溢出中断使能位。 当 TOIE0 位为“1”，且全局中断置位，TC0 溢出中断使能。当 TC0 发生溢出，即 TIFR 中的 TOV0 位被置位时，中断产生。 当 TOIE0 位为“0”时，TC0 溢出中断被禁止。

TC0 中断标志寄存器- TIFR0

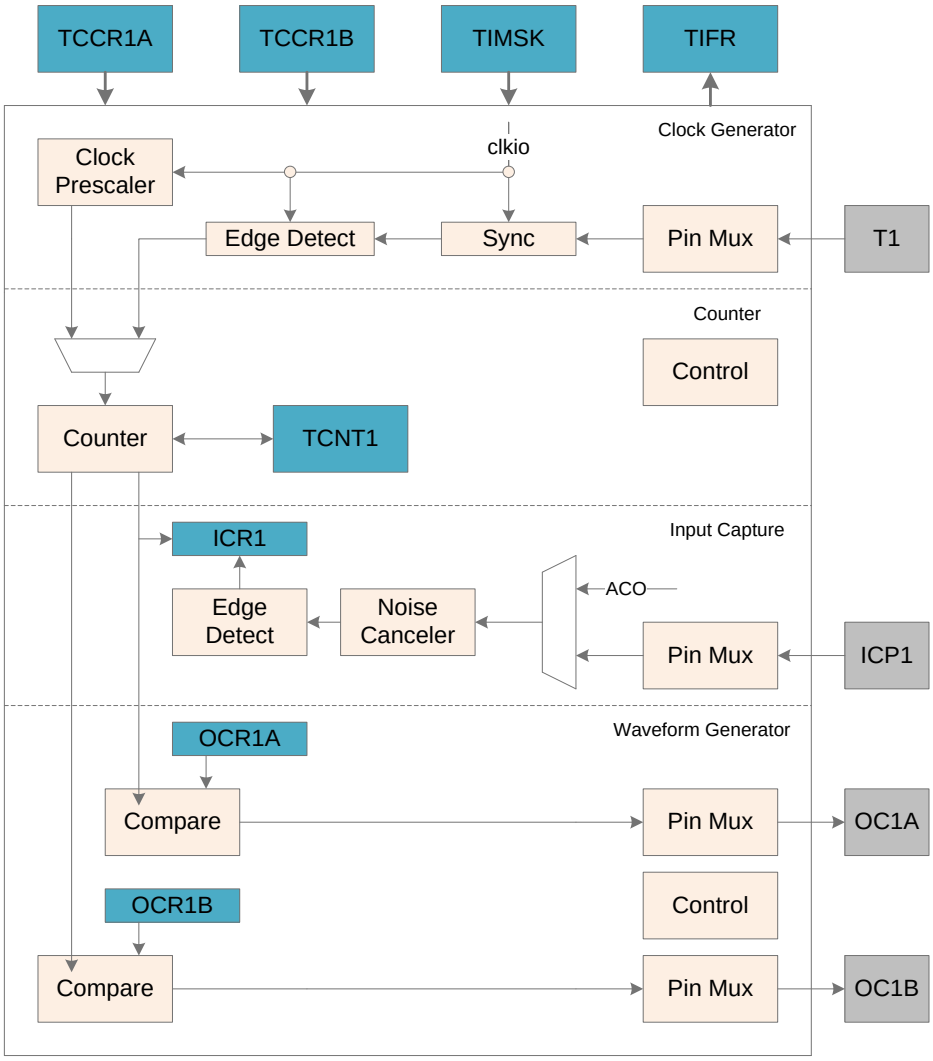
TIFR0 – TC0 中断标志寄存器								
地址: 0x35					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	OCF0B	OCF0A	TOV0
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:3		保留。
2	OCF0B	TC0 输出比较 B 匹配标志位。 当 TCNT0 等于 OCR0B 时，比较单元就给出匹配信号，并置位比较标志 OCF0B。若此时输出比较 B 中断使能 OCIE0B 为“1”且全局中断标志置位，则会产生输出比较 B 中断。执行此中断服务程序时 OCF0B 将自动清零，或对 OCF0B 位写“1”也可清零该位。
1	OCF0A	TC0 输出比较 A 匹配标志位。 当 TCNT0 等于 OCR0A 时，比较单元就给出匹配信号，并置位比较标志 OCF0A。若此时输出比较 A 中断使能 OCIE0A 为“1”且全局中断标志置位，则会产生输出比较 A 中断。执行此中断服务程序时 OCF0A 将自动清零，或对 OCF0A 位写“1”也可清零该位。
0	TOV0	TC0 溢出标志位。 当计数器发生溢出时，置位溢出标志 TOV0。若此时溢出中断使能 TOIE0 为“1”且全局中断标志置位，则会产生溢出中断。执行此中断服务程序时 TOV0 将自动清零，或对 TOV0 位写“1”也可清零该位。

16 位定时/计数器 1

- 真正的 16 位设计，允许 16 位的 PWM
- 2 个独立的输出比较单元
- 双缓冲的输出比较寄存器
- 1 个输入捕捉单元
- 输入捕捉噪声抑制器
- 比较匹配时自动清零计数器并自动加载
- 无干扰脉冲的相位修正的 PWM
- 可变的 PWM 周期
- 频率发生器
- 外部事件计数器
- 4 个独立的中断源

概述



TC1 结构图

TC1 是一个通用 16 位定时计数器模块，支持 PWM 输出，可以精确地产生波形。TC1

包含 1 个 16 位计数器，波形产生模式控制单元，2 个独立的输出比较单元和 1 个输入捕捉单元。波形产生模式控制单元控制着计数器的工作模式和比较输出波形的产生。根据不同的工作模式，计数器对每一个计数时钟 Clkt1 实现清零、加一或减一操作。Clkt1 可以由内部时钟源或外部时钟源产生。当计数器的计数值 TCNT1 到达最大值（等于极大值 0xFFFF 或固定值或输出比较寄存器 OCR1A 或输入捕捉寄存器 ICR1，定义为 TOP，定义极大值为 MAX 以示区别）时，计数器会进行清零或减一操作。当计数器的计数值 TCNT1 到达最小值（等于 0x0000，定义为 BOTTOM）时，计数器会进行加一操作。当计数器的计数值 TCNT1 到达 OCR1A 或 OCR1B 时，也被称为发生比较匹配时，会清零或置位输出比较信号 OC1A 或 OC1B，来产生 PWM 波形。当开启输入捕捉功能时，计数器被触发即开始或停止计数，ICR1 寄存器会记录捕捉信号触发周期内的计数值。

工作模式

定时计数器 1 有六种不同的工作模式，包括普通模式（Normal），比较匹配时清零（CTC）模式，快速脉冲宽度调制（FPWM）模式，相位修正脉冲宽度调制（PCPWM）模式，相位频率修正脉冲宽度调制（PFCPWM）模式，和输入捕捉（ICP）模式。由波形产生模式控制位 WGM1[3:0]来选择。下面具体来描述这六种模式。由于有两个独立的输出比较单元，分别用“A”和“B”来表示，用小写的“x”来表示这两个输出比较单元通道。

普通模式

普通模式是定时计数器最简单的工作模式，此时波形产生模式控制位 WGM1[3:0]=0，计数的最大值 TOP 为 MAX（0xFFFF）。在这种模式下，计数方式为每一个计数时钟加一递增，当计数器到达 TOP 溢出后就回到 BOTTOM 重新开始累加。在计数值 TCNT1 变成零的同一个计数时钟里置位定时计数器溢出标志 TOV1。这种模式下 TOV1 标志就像是第 17 计数位，只是只会被置位不会被清零。溢出中断服务程序会自动清除 TOV1 标志，软件可以用它来提高定时计数器的分辨率。普通模式下没有特殊情形需要考虑，可以随时写入新的计数值。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。当 COM1x=1 时，发生比较匹配时会翻转 OC1x 信号，这种情况下波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

输出比较单元可以用来产生中断，但是在普通模式下不推荐使用中断，这样会占用太多 CPU 的时间。

CTC 模式

设置 WGM1[3:0]=4 或 12 时，定时计数器 1 进入 CTC 模式。当 WGM1[3]=0 时，计数最大值 TOP 为 OCR1A，当 WGM1[3]=1 时，计数最大值 TOP 为 ICR1。下面以 WGM1[3:0]=4 为例来描述 CTC 模式在这个模式下，计数方式为每一个计数时钟加一递增，当计数器的数值 TCNT1 等于 TOP 时计数器清零。这个模式使得用户可以很容易的控制比较匹配输出的频率，也简化了外部事件计数的操作。

当计数器到达 TOP 时，输出比较匹配标志 OCF1 被置位，相应的中断使能置位时将会产生中

断。在中断服务程序里可以更新 OCR1A 寄存器。在这个模式下 OCR1A 没有使用双缓冲，在计数器以无预分频器或很低的预分频器工作下将最大值更新为接近最小值的时候要小心。如果写入 OCR1A 的数值小于当时的 TCNT1 值时，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到 MAX，然后再从 BOTTOM 开始计数到 OCR1A。和普通模式一样，计数值回到 0x0 的计数时钟里置位 TOV1 标志。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

从公式可以看出，当设置 OCR1A 为 0x0 且无预分频器时，可以获得最大频率为 $f_{sys}/2$ 的输出波形。

当 WGM1[3:0]=12 时与 WGM1[3:0]=4 类似，只是把与 OCR1A 相关的换成 ICR1 即可。

快速 PWM 模式

设置 WGM1[3:0]=5，6，7，14 或 15 时，定时计数器 1 进入快速 PWM 模式，计数最大值 TOP 分别为 0xFF，0x1FF，0x3FF，ICR1 或 OCR1A，可以用来产生高频的 PWM 波形。快速 PWM 模式和其他 PWM 模式不同在于它是单向操作。计数器从 BOTTOM 累加到 TOP 后又回到 BOTTOM 重新计数。当计数值 TCNT1 到达 TOP 或 BOTTOM 时，输出比较信号 OC1x 会被置位或清零，取决于比较输出模式 COM1 的设置，详情见寄存器描述。由于采用单向操作，快速 PWM 模式的操作频率是采用双向操作的相位修正 PWM 模式的两倍。高频特性使得快速 PWM 模式适用于功率调节，整流以及 DAC 应用。高频信号可以减小外部元器件（电感电容等）的尺寸，从而降低系统成本。

当计数值到达 TOP 时，定时计数器溢出标志 TOV1 将会被置位，并把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新 OCR1A 寄存器。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。波形的频率可用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

当 TCNT1 和 OCR1x 发生比较匹配时，波形产生器就置位（清零）OC1x 信号，当 TCNT1 被清零时，波形产生器就清零（置位）OC1x 信号，以此来产生 PWM 波。由此 OCR1x 的极值将会产生特殊的 PWM 波形。当 OCR1x 设置为 0x00 时，输出的 PWM 为每(1+TOP)个计数时钟里有一个窄的尖峰脉冲。当 OCR1x 设置为 TOP 时，输出的波形为持续的高电平或低电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

相位修正 PWM 模式

当设置 WGM0[3:0]=1，2，3，10 或 11 时，定时计数器 1 进入相位修正 PWM 模式，计数的最大值 TOP 分别为 0xFF，0x1FF，0x3FF，ICR1 或 OCR1A。计数器采用双向操作，由 BOTTOM

递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT1 与 OCR1x 匹配时，输出比较信号 OC1x 将会被清零或置位，取决于比较输出模式 COM1 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV1 标志，当计数到达 TOP 时把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR1x 存器。

设置 OC1x 脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 波形。波形的频率可用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就清零（置位）OC1x 信号。在递减计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就置位（清零）OC1x 信号。由此 OCR1x 的极值会产生特殊的 PWM 波。当 OCR1x 设置为 TOP 或 BOTTOM 时，OC1x 信号输出会一直保持低电平或高电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

为了保证输出 PWM 波在 BOTTOM 两侧的对称性，在没有发生比较匹配时，有两种情况下也会翻转 OC1x 信号。第一种情况是，当 OCR1x 的值由 TOP 改变为其他数据时。当 OCR1x 为 TOP，计数值达到 TOP 时，OC1x 的输出与前面降序计数时比较匹配的结果相同，即保持 OC1x 不变。此时会更新比较值为新的 OCR1x 的值（非 TOP），OC1x 的值会一直保持，直到升序计数时发生比较匹配而翻转。此时 OC1x 信号并不以最小值为中心对称，因此需要在 TCNT1 到达最大值时翻转 OC1x 信号，此即没有发生比较匹配时翻转 OC1x 信号的第一种情况。第二种情况是，当 TCNT1 从比 OCR1x 高的值开始计数时，因而会丢失一次比较匹配，从而引起不对称情形的产生。同样需要翻转 OC1x 信号去实现最小值两侧的对称性。

相位频率修正 PWM 模式

当设置 WGM0[3:0]=8 或 9 时，定时计数器 1 进入相位频率修正 PWM 模式，计数的最大值 TOP 分别为 ICR1 或 OCR1A。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT1 与 OCR1x 匹配时，输出比较信号 OC1x 将会被清零或置位，取决于比较输出模式 COM1 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位频率修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV1 标志，并且把比较缓冲器的值更新到比较值，更新比较值的时间是相位频率修正 PWM 模式和相位修正 PWM 模式的最大不同点。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR1x 存器。当 CPU 改变 TOP 值即 ORC1A 或 ICR1 的值时，必须保证新的 TOP 值不小于已经在使用的 TOP 值，否则比较匹配将不会再发生。

设置 OC1x 脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 波形。波形的频率可用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就清零（置位）OC1x 信号。在递减计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就置位（清零）OC1x 信号。由此 OCR1x 的极值会产生特殊的 PWM 波。当 OCR1x 设置为 TOP 或 BOTTOM 时，OC1x 信号输出会一直保持低电平或高电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

因为 OCR1x 寄存器是在 BOTTOM 时刻更新的，所以 TOP 值两边升序和降序的计数长度是一样的，也就产生了频率和相位都正确的对称波形。

当使用固定 TOP 值时，最好采用 ICR1 寄存器作为 TOP 值，即设置 WGM1[3:0]=8，此时 OCR1A 寄存器只需用来产生 PWM 输出。如果要产生频率变化的 PWM 波，必须通过改变 TOP 值，OCR1A 的双缓冲特性会更适合于这个应用。

输入捕捉模式

输入捕捉用来捕获外部事件，并为其赋予时间标记以说明此事件发生的时刻，可以在前面的计数模式下进行，不过要除去使用 ICR1 值作为计数 TOP 值的波形产生模式。

外部事件发生的触发信号由引脚 ICP1 输入，也可以通过模拟比较器单元来实现。当引脚 ICP1 上的逻辑电平发生变化，或模拟比较器的输出 ACO 电平发生变化，并且这个电平变化被输入捕捉单元所捕获，输入捕捉即被触发，此时 16 位的计数值 TCNT1 数据被复制到输入捕捉寄存器 ICR1，同时输入捕捉标志 ICF1 置位，若 ICIE1 位为“1”，输入捕捉标志将产生输入捕捉中断。

通过设置模拟比较控制与状态寄存器 ACSR 的模拟比较输入捕捉控制位 ACIC 来选择输入捕捉触发源 ICP1 或 ACO。需注意的是，改变触发源有可能造成一次输入捕捉，因此在改变触发源后必须对 ICF1 进行一次清零操作来避免出现错误的结果。

输入捕捉信号经过一个可选的噪声抑制器之后送入边沿检测器，根据输入捕捉选择控制位 ICES1 的配置，看检测到的边沿是否满足触发条件。噪声抑制器是一个简单的数字滤波，对输入信号进行 4 次采样，只有当 4 次采样值都相等时其输出才会送入边沿检测器。噪声抑制器由 TCCR1B 寄存器的 ICNC1 位控制其使能或禁止。

使用输入捕捉功能时，当 ICF1 被置位后，应尽可能早的读取 ICR1 寄存器的值，因为下一次捕捉事件发生后 ICR1 的值将会被更新。推荐使能输入捕捉中断，在任何输入捕捉工作模式下，都不推荐在操作过程中改变计数 TOP 值。

输入捕捉到的时间标记可用来计算频率、占空比及信号的其它特征，以及为触发事件创建日志。测量外部信号的占空比时要求每次捕捉后都要改变触发沿，因此读取 ICR1 值以后须尽快改变触发的信号边沿。

寄存器定义

TC1 寄存器列表

寄存器	地址	默认值	描述
TCCR1A	0x80	0x00	TC1 控制寄存器 A
TCCR1B	0x81	0x00	TC1 控制寄存器 B
TCCR1C	0x82	0x00	TC1 控制寄存器 C
TCNT1L	0x84	0x00	TC1 计数值寄存器低字节
TCNT1H	0x85	0x00	TC1 计数值寄存器高字节
ICR1L	0x86	0x00	TC1 输入捕捉寄存器低字节
ICR1H	0x87	0x00	TC1 输入捕捉寄存器高字节
OCR1AL	0x88	0x00	TC1 输出比较寄存器 A 低字节
OCR1AH	0x89	0x00	TC1 输出比较寄存器 A 高字节
OCR1BL	0x8A	0x00	TC1 输出比较寄存器 B 低字节
OCR1BH	0x8B	0x00	TC1 输出比较寄存器 B 高字节
TIMSK1	0x6F	0x00	定时计数器中断屏蔽寄存器
TIFR1	0x36	0x00	定时计数器中断标志寄存器

TCCR1A –TC1 控制寄存器 A

TCCR1A –TC1 控制寄存器 A								
地址: 0x80					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
R/W	R/W	R/W	R/W	R/W	W	W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	COM1A1	比较匹配输出 A 模式控制高位。 COM1A1 和 COM1A0 组成 COM1A[1:0]来控制输出比较波形 OC1A。如果 COM1A 的 1 位或者 2 位都置位，输出比较波形占据着 OC1A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
6	COM1A0	比较匹配输出 A 模式控制低位。 COM1A1 和 COM1A0 组成 COM1A[1:0]来控制输出比较波形 OC1A。如果 COM1A 的 1 位或者 2 位都置位，输出比较波形占据着 OC1A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
5	COM1B1	比较匹配输出 B 模式控制高位。 COM1B1 和 COM1B0 组成 COM1B[1:0]来控制输出比较波形 OC1B。如果

		COM1B 的 1 位或者 2 位都置位，输出比较波形占据着 OC1B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
4	COM1B0	比较匹配输出 B 模式控制低位。 COM1B1 和 COM1B0 组成 COM1B[1:0]来控制输出比较波形 OC1B。如果 COM1B 的 1 位或者 2 位都置位，输出比较波形占据着 OC1B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
3:2	-	保留。
1	WGM11	波形产生模式控制次低位。 WGM11 和 WGM13,WGM12,WGM10 一起组成波形产生模式控制 WGM1[3:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
0	WGM10	波形产生模式控制最低位。 WGM10 和 WGM13,WGM12,WGM11 一起组成波形产生模式控制 WGM1[3:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

下表为非 PWM 模式（即普通模式和 CTC 模式）下，比较输出模式对输出比较波形的控制。

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	比较匹配时翻转 OC1x 信号
2	比较匹配时清零 OC1x 信号
3	比较匹配时置位 OC1x 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	WGM1 为 15 时：比较匹配时翻转 OC1A 信号, OC1B 断开 WGM1 为其它值时：OC1x 断开，通用 IO 口操作
2	比较匹配时清零 OC1x 信号，最大值匹配时置位 OC1x 信号
3	比较匹配时置位 OC1x 信号，最大值匹配时清零 OC1x 信号

下表为相位修正模式下比较输出模式对输出比较波形的控制。

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	WGM1 为 9 或 11 时：比较匹配时翻转 OC1A 信号, OC1B 断开 WGM1 为其它值时：OC1x 断开，通用 IO 口操作
2	升序计数下比较匹配清零 OC1x 信号，降序计数下比较匹配置位 OC1x 信号
3	升序计数下比较匹配置位 OC1x 信号，降序计数下比较匹配清零 OC1x 信号

TCCR1B –TC1 控制寄存器 B

TCCR1B –TC1 控制寄存器 B								
地址: 0x81					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
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	描述
7	ICNC1	输入捕捉噪声抑制器使能控制位。 当设置 ICNC1 位为“1”时，使能输入捕捉噪声抑制器，此时外部引脚 ICP1 的输入被滤波，连续 4 个采样值相等时输入信号才有效，该功能使得输入捕捉被延迟了 4 个时钟周期。 当设置 ICNC1 位为“0”时，禁止输入捕捉噪声抑制器，此时外部引脚 ICP1 的输入直接有效。
6	ICES1	输入捕捉触发沿选择控制位。 当设置 ICES1 位为“1”时，选择电平的上升沿触发输入捕捉；当设置 ICES1 位为“0”时，选择电平的下降沿触发输入捕捉。 当捕获到一个事件后，计数器的数值被复制到 ICR1 寄存器，同时置位输入捕捉标志 ICF1。如果中断使能，产生输入捕捉中断。
5	-	保留。
4	WGM13	波形产生模式控制高位。 WGM13 和 WGM12,WGM11,WGM10 一起组成波形产生模式控制 WGM1[3:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
3	WGM12	波形产生模式控制次高位。 WGM12 和 WGM13,WGM11,WGM10 一起组成波形产生模式控制 WGM1[3:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
2	CS12	时钟选择控制高位。用于选择定时计数器 1 的时钟源。
1	CS11	时钟选择控制中位。用于选择定时计数器 1 的时钟源。
0	CS10	时钟选择控制低位。 用于选择定时计数器 1 的时钟源。

CS1[2:0]	描述
0	无时钟源，停止计数
1	clk_{sys}
2	clk_{sys}/8，来自预分频器
3	clk_{sys}/64，来自预分频器
4	clk_{sys}/256，来自预分频器
5	clk_{sys}/1024，来自预分频器
6	外部时钟 T1 引脚，下降沿触发
7	外部时钟 T1 引脚，上升沿触发

下表为波形产生模式控制。

WGM1[3:0]	工作模式	TOP 值	更新 OCR0 时刻	置位 TOV0 时刻
0	Normal	0xFFFF	立即	MAX
1	8 位 PCPWM	0x00FF	TOP	BOTTOM
2	9 位 PCPWM	0x01FF	TOP	BOTTOM
3	10 位 PCPWM	0x03FF	TOP	BOTTOM
4	CTC	OCR1A	立即	MAX
5	8 位 FPWM	0x00FF	BOTTOM	TOP
6	9 位 FPWM	0x01FF	BOTTOM	TOP
7	10 位 FPWM	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	立即	MAX
13	保留	-	-	-
14	FPWM	ICR1	TOP	TOP
15	FPWM	OCR1A	TOP	TOP

TCCR1C –TC1 控制寄存器 C

TCCR1C –TC1 控制寄存器 C								
地址: 0x82					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	FOC1A	FOC1B	-	-	-	-	-	-
R/W	W	W	-	-	-	-	-	-
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	FOC1A	强制输出比较 A。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC1A 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF1A 标志，也不会重载或清零定时器，但是输出引脚 OC1A 将被按照 COM1A 的设置相应的更新，就跟真的发生了比较匹配一样。 工作于 PWM 模式时，写 TCCR1A 寄存器时要对其清零。 读取 FOC1A 的返回值一直为零。
6	FOC1B	强制输出比较 B。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC1B 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF1B 标志，也不会重载或清零定时器，但是输出引脚 OC1B 将被按照 COM1B 的设置相应的更新，就跟真的发生了比较匹配一样。工作于 PWM 模式时，写 TCCR1A 寄存器时要对其清零。读取 FOC1B 的返回值一直为零。

5:0	-	保留。

TCNT1L –TC1 计数值寄存器低字节

TCNT1L **–TC1 计数值寄存器低字节**								
地址: 0x84					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	TCNT1L 7	TCNT1L 6	TCNT1L 5	TCNT1L 4	TCNT1L 3	TCNT1L 2	TCNT1L 1	TCNT1L 0
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	描述
7:0	TCNT1[7:0]	TC1 计数值的低字节。 TCNT1H 和 TCNT1L 结合到一起组成 TCNT1，通过 TCNT1 寄存器可以直接对计数器的 16 位计数值进行读写访问。读写 16 位寄存器需要两次操作。写 16 位 TCNT1 时，应先写入 TCNT1H。读 16 位 TCNT1 时，应先读取 TCNT1L。 CPU 对 TCNT1 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT1 寄存器的值与 OCR1x 的值一致而不会引发中断。 如果写入 TCNT1 的数值等于或绕过 OCR1x 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT1。CPU 写计数器比清零或加减操作的优先级高。

TCNT1H –TC1 计数值寄存器高字节

TCNT1H **–TC1 计数值寄存器高字节**								
地址: 0x85					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	TCNT1H 7	TCNT1H 6	TCNT1H 5	TCNT1H 4	TCNT1H 3	TCNT1H 2	TCNT1H 1	TCNT1H 0
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	描述
7:0	TCNT1[15:8]	TC1 计数值的高字节。 TCNT1H 和 TCNT1L 结合到一起组成 TCNT1，通过 TCNT1 寄存器可以直接对计数器的 16 位计数值进行读写访问。读写 16 位寄存器需要两次操作。写 16 位 TCNT1 时，应先写入 TCNT1H。读 16 位 TCNT1 时，应先读取 TCNT1L。

		CPU 对 TCNT1 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT1 寄存器的值与 OCR1x 的值一致而不会引发中断。 如果写入 TCNT1 的数值等于或绕过 OCR1x 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT1。CPU 写计数器比清零或加减操作的优先级高。

ICR1L –TC1 输入捕捉寄存器低字节

ICR1L –TC1 输入捕捉寄存器低字节								
地址: 0x86					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	ICR1[7:0]	TC1 输入捕捉值的低字节。 ICR1H 和 ICR1L 结合到一起组成 16 位的 ICR1。读写 16 位寄存器需要两次操作。写 16 位 ICR1 时，应先写入 ICR1H。读 16 位 ICR1 时，应先读取 ICR1L。当输入捕捉被触发时，计数值 TCNT1 就会更新复制到 ICR1 寄存器里。ICR1 寄存器也可用来定义计数的 TOP 值。

ICR1H –TC1 输入捕捉寄存器高字节

ICR1H –TC1 输入捕捉寄存器高字节								
地址: 0x87					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	ICR1[15:8]	TC1 输入捕捉值的高字节。 ICR1H 和 ICR1L 结合到一起组成 16 位的 ICR1。读写 16 位寄存器需要两次操作。写 16 位 ICR1 时，应先写入 ICR1H。读 16 位 ICR1 时，应先读取 ICR1L。当输入捕捉被触发时，计数值 TCNT1 就会更新复制到 ICR1 寄存器里。ICR1 寄存器也可用来定义计数的 TOP 值。

OCR1AL –TC1 输出比较寄存器 A 低字节

OCR1AL –TC1 输出比较寄存器 A 低字节								
地址: 0x88					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	OCR1AL 7	OCR1AL 6	OCR1AL 5	OCR1AL 4	OCR1AL 3	OCR1AL 2	OCR1AL 1	OCR1AL 0
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	描述
7:0	OCR1A[7:0]	输出比较寄存器 A 的低字节。 OCR1AL 和 OCR1AH 结合到一起组成 16 位的 OCR1A。读写 16 位寄存器需要两次操作。写 16 位 OCR1A 时，应先写入 OCR1AH。读 16 位 OCR1A 时，应先读取 OCR1AL。 OCR1A 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1A 引脚上产生波形。 当使用 PWM 模式时，OCR1A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1A 本身。

OCR1AH –TC1 输出比较寄存器 A 高字节

OCR1AH –TC1 输出比较寄存器 A 高字节								
地址: 0x89					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	OCR1AH 7	OCR1AH 6	OCR1AH 5	OCR1AH 4	OCR1AH 3	OCR1AH 2	OCR1AH 1	OCR1AH 0
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	描述
7:0	OCR1A[15:8]	输出比较寄存器 A 的高字节。 OCR1AL 和 OCR1AH 结合到一起组成 16 位的 OCR1A。读写 16 位寄存器需要两次操作。写 16 位 OCR1A 时，应先写入 OCR1AH。读 16 位 OCR1A 时，应先读取 OCR1AL。 OCR1A 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1A 引脚上产生波形。 当使用 PWM 模式时，OCR1A 寄存器使用双缓冲寄存器。而普通

		工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1A 本身。

OCR1BL –TC1 输出比较寄存器 B 低字节

OCR1BL –TC1 输出比较寄存器 B 低字节								
地址: 0x8A					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	OCR1B[7:0]	输出比较寄存器 B 的低字节。 OCR1BL 和 OCR1BH 结合到一起组成 16 位的 OCR1B。读写 16 位寄存器需要两次操作。写 16 位 OCR1B 时，应先写入 OCR1BH。读 16 位 OCR1B 时，应先读取 OCR1BL。 OCR1B 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1B 引脚上产生波形。 当使用 PWM 模式时，OCR1B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1B 本身。

OCR1BH –TC1 输出比较寄存器 B 高字节

OCR1BH –TC1 输出比较寄存器 B 高字节								
地址: 0x8B					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	OCR1B[15:8]	输出比较寄存器 B 的高字节。

		OCR1BL 和 OCR1BH 结合到一起组成 16 位的 OCR1B。读写 16 位寄存器需要两次操作。写 16 位 OCR1B 时，应先写入 OCR1BH。读 16 位 OCR1B 时，应先读取 OCR1BL。 OCR1B 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1B 引脚上产生波形。 当使用 PWM 模式时，OCR1B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1B 本身。

TIMSK1 – TC1 中断屏蔽寄存器

TIMSK1 – TC1 中断屏蔽寄存器								
地址: 0x6F					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	TICIE1	-	-	OCIE1A	OCIE1B	TOIE1
R/W	-	-	R/W	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:6	-	保留。
5	TICIE1	TC1 输入捕捉中断使能控制位。 当 ICIE1 位为“1”时，且全局中断置位，TC1 输入捕捉中断被使能。当输入捕捉触发时，即 TIFR1 的 ICF1 标志被置位，中断发生。 当 ICIE1 位为“0”时，TC1 输入捕捉中断被禁止。
4:3	-	保留。
2	OCIE1B	TC1 输出比较 B 匹配中断使能位。 当 OCIE1B 位为“1”，且全局中断置位，TC1 输出比较 B 匹配中断使能。当比较匹配发生时，即 TIFR 中 OCF1B 位被置位时，中断产生。 当 OCIE1B 位为“0”时，TC1 输出比较 B 匹配中断被禁止。
1	OCIE1A	TC1 输出比较 A 匹配中断使能位。 当 OCIE1A 位为“1”，且全局中断置位，TC1 输出比较 A 匹配中断使能。当比较匹配发生时，即 TIFR 中 OCF1A 位被置位时，中断产生。 当 OCIE1A 位为“0”时，TC1 输出比较 A 匹配中断被禁止。
0	TOIE1	TC1 溢出中断使能位。 当 TOIE1 位为“1”，且全局中断置位，TC1 溢出中断使能。当 TC1 发生溢出，即 TIFR 中的 TOV1 位被置位时，中断产生。 当 TOIE1 位为“0”时，TC1 溢出中断被禁止。

TIFR1 – TC1 中断标志寄存器

TIFR1 – TC1 中断标志寄存器								
地址: 0x36					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1
R/W	-	-	R/W	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:6	-	保留。
5	ICF1	输入捕捉标志位。 当输入捕捉事件发生时，ICF1 标志被置位。当 ICR1 被用作计数的 TOP 值，且计数值到达 TOP 值时，ICF1 标志被置位。若 ICIE1 为“1”且全局中断标志置位，则会产生输入捕捉中断。执行此中断服务程序时 ICF1 将自动清零，或对 ICF1 位写“1”也可清零该位。
4:3	-	保留。
2	OCF1B	输出比较 B 匹配标志位。 当 TCNT1 等于 OCR1B 时，比较单元就给出匹配信号，并置位比较标志 OCF1B。若此时输出比较中断使能 OCIE1B 为“1”且全局中断标志置位，则会产生输出比较中断。执行此中断服务程序时 OCF1B 将自动清零，或对 OCF1B 位写“1”也可清零该位。
1	OCF1A	输出比较 A 匹配标志位。 当 TCNT1 等于 OCR1A 时，比较单元就给出匹配信号，并置位比较标志 OCF1A。若此时输出比较中断使能 OCIE1A 为“1”且全局中断标志置位，则会产生输出比较中断。执行此中断服务程序时 OCF1A 将自动清零，或对 OCF1A 位写“1”也可清零该位。
0	TOV1	溢出标志位。 当计数器发生溢出时，置位溢出标志 TOV1。若此时溢出中断使能 TOIE1 为“1”且全局中断标志置位，则会产生溢出中断。执行此中断服务程序时 TOV1 将自动清零，或对 TOV1 位写“1”也可清零该位。

定时/计数器 0/1 预分频器

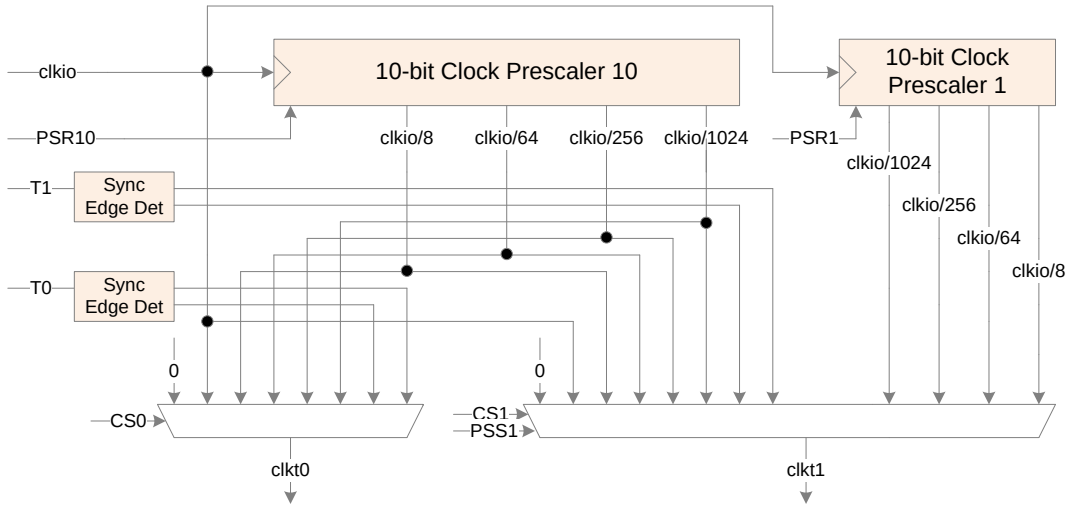
- 2 个 10 位预分频器
- 复用模式下 TC0 和 TC1 复用预分频器 CPS10
- 单用模式下 TC0 独用预分频器 CPS10，TC1 独用预分频器 CPS1
- 支持软件复位

概述

复用模式下（PSS1=0），TC0 和 TC1 共用一个 10 位的预分频器 CPS10，但它们有不同的分频设置。

单用模式下（PSS1=1），TC0 独立使用预分频器 CPS10，TC1 独立使用预分频器 CPS1，它们有不同的分频设置。

以下的描述使用于 TC0 和 TC1，其中 n 代表 0 或 1。



TC0/TC1 Prescaler **结构图**

内部时钟源

当设置 CSn[2:0]=1 时，定时计数器可直接由系统时钟 clkio 驱动，这也是 TCn 最高频率的时钟源。预分频器可以输出 4 个不同的时钟频率，分别是 clkio/8，clkio/64，clkio/256 和 clkio/1024。

分频器复位

复用模式

当设置 PSS1 位为“0”时，TC0 和 TC1 共用一个预分频器 CPS10。

预分频器是独立运行的，其操作独立于 TC 的时钟选择逻辑，且它有 TC0 和 TC1 共享。由于不受时钟选择控制的影响，预分频器的状态对分频时钟的应用会有影响。当定时器使能并且选用预分频器的输出作为计数时钟源（6>CSn[2:0]>1）时，影响就会产生。从定时器使能到第一次计数可能要花费 1 到 N+1 个系统时钟，其中 N 为预分频因子（8，64，256 或 1024）。

通过复位预分频器来同步定时器和程序运行是可能的。但是必须注意，另一个定时器是否正在使用这个预分频器，复位预分频器会影响到所有与其连接的定时器。

单用模式

当设置 PSS1 位为“1”时，TC0 独立使用预分频器 CPS10，预分频器的复位由 PSR10 位来控制。TC1 独立使用预分频器 CPS1，预分频器的复位由 PSR1 位来控制。各自的复位单独起作用，不会影响其它预分频器。

外部时钟源

由 T0/T1 引脚提供的外部时钟源可以用作计数时钟源。T0/T1 引脚的信号经过同步逻辑和边沿检测器之后作为计数器的时钟源。每个上升沿（CSn[2:0]=7）或下降沿（CSn[2:0]=6）都会产生一个计数脉冲。外部时钟源不会送入预分频器。

由于引脚上同步与边沿检测电路的存在，T0/T1 上电平的变化需要延迟 2.5 到 3.5 个系统时钟才能使计数器更新。

禁止或使能时钟输入必须在 T0/T1 保持稳定至少需要一个系统时钟周期后才能进行，否则有产生错误计数时钟脉冲的可能。

为了保证正确的采样，外部时钟脉冲宽度必须大于一个系统时钟周期，在占空比为 50% 时外部时钟频率必须小于系统时钟频率的一半。由于振荡器本身的误差带来的系统时钟频率及占空比的差异，建议外部时钟的最高频率不要大于 $f_{sys}/2.5$。

寄存器定义

GTCCR – 通用定时计数器控制寄存器

GTCCR – 通用定时计数器控制寄存器								
地址: 0x43					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	TSM	-	-	-	-	-	PSRASY	PSRSYNC
R/W	R/W	R/W	R/W	-	R/W	R/W	W	W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	TSM	定时计数器同步模式控制位。 当设置 TSM 位为“1”时，为定时计数器同步模式。同步模式下，写入 PSRASY 位和 PSRSYNC 位的值会保持，让相应的预分频器一直被复位。这能确保相应的定时计数器中止并配置成相同的值。 当设置 TSM 位为“0”时，PSRASY 位和 PSRSYNC 位的值会被硬件清零，且定

		时计数器同时开始工作。
6:2	-	保留。
1	PSRASY	见定时器 TC2 寄存器描述。
0	PSRSYNC	预分频器 CPS10 复位控制位。 当设置 PSRSYNC 位为“1”时，预分频器 CPS10 将被复位。当 TSM 位未置位时，复位之后硬件将清零 PSRSYNC 位。 当设置 PSRSYNC 位为“0”时，设置无效。 复用模式下，TC0/TC1 共用预分频器，复位将会影响这两个定时器。 单用模式下，复位只会影响 TC0。 读取这一位的值将始终为“0”。

PSSR – 预分频器选择寄存器

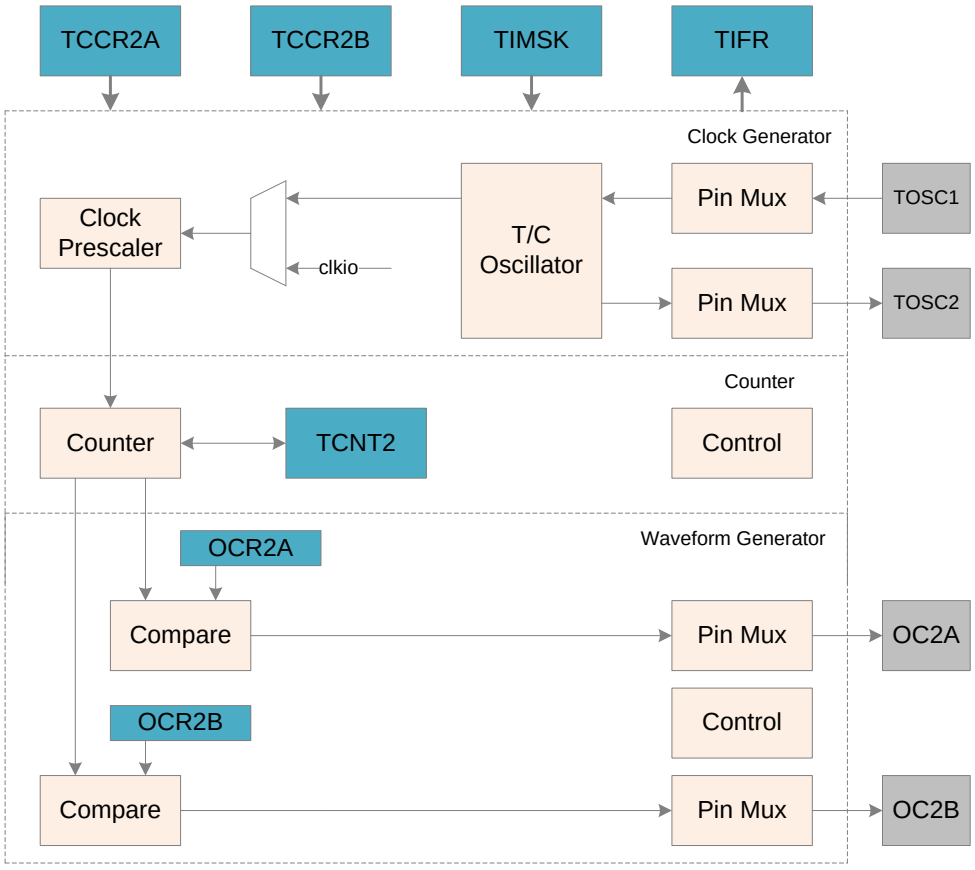
PSSR – 预分频器选择寄存器								
地址: 0xE2					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	PSS1	-	-	-	-	-	-	PSR1
R/W	R/W	-	-	-	-	-	-	W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	PSS1	预分频器选择控制位。 当设置 PSS1 位为“1”时，为预分频器单用模式。TC0 单独使用预分频器 CPS10，TC1 单独使用预分频器 CPS1。 当设置 PSS1 位为“0”时，为预分频器复用模式。TC0 和 TC1 共用预分频器 CPS10。预分频器 CPS1 无效，将会一直被复位。
6:1	-	保留。
0	PSR1	预分频器 CPS1 复位控制位。 PSR1 位只在单用模式下有效。 当设置 PSR1 位为“1”时，预分频器 CPS1 将被复位。复位之后硬件将清零 PSR1 位。当设置 PSR1 位为“0”时，设置无效。 读取这一位的值将始终为“0”。

8 位定时/计数器 2

- 8 位计数器
- 两个独立的比较单元
- 比较匹配发生时自动清零计数器并自动加载
- 无干扰脉冲的相位修正的 PWM 输出
- 频率发生器
- 外部事件计数器
- 10 位的时钟预分频器
- 溢出和比较匹配中断
- 允许使用外部 32.768KHz 的 RTC 晶振计数

概述



TC2 结构图

TC2 是一个通用 8 位定时计数器模块，支持 PWM 输出，可以精确地产生波形。TC2 包含 1 个 8 位计数器，波形产生模式控制单元和 2 个输出比较单元。波形产生模式控制单元控制着计数器的工作模式和比较输出波形的产生。根据不同的工作模式，计数器对每一个计数时钟 Clkt2 实现清零、加一或减一操作。Clkt2 可以由内部时钟源或外部时钟源产生。当使用外部 32.768KHz 的晶振计数时，TC2 可用作 RTC 计数器。当计数器的计数值 TCNT2 到达最大值（等于极大值 0xFF 或输出比较寄存器 OCR2A，定义为 TOP，定义极大值为 MAX 以示区别）时，计数器会进行清零或减一操作。当计数器的计数值 TCNT2 到达最小值（等

于 0x00，定义为 BOTTOM）时，计数器会进行加一操作。当计数器的计数值 TCNT2 到达 OCR2A/OCR2B 时，也被称为发生比较匹配时，会清零或置位输出比较信号 OC2A/OCR2B，来产生 PWM 波形。

工作模式

定时计数器 2 有四种不同的工作模式，包括普通模式（Normal），比较匹配时清零（CTC）模式，快速脉冲宽度调制（FPWM）模式和相位修正脉冲宽度调制（PCPWM）模式，由波形产生模式控制位 WGM2[2:0]来选择。下面具体来描述这四种模式。由于有两个独立的输出比较单元，分别用“A”和“B”来表示，用小写的“x”来表示这两个输出比较单元通道。

普通模式

普通模式是定时计数器最简单的工作模式，此时波形产生模式控制位 WGM2[2:0]=0，计数的最大值 TOP 为 MAX（0xFF）。在这种模式下，计数方式为每一个计数时钟加一递增，当计数器到达 TOP 溢出后就回到 BOTTOM 重新开始累加。在计数值 TCNT2 变成零的同一个计数时钟里置位定时计数器溢出标志 TOV2。这种模式下 TOV2 标志就像是第 9 计数位，只是只会被置位不会被清零。溢出中断服务程序会自动清除 TOV2 标志，软件可以用它来提高定时计数器的分辨率。普通模式下没有特殊情形需要考虑，可以随时写入新的计数值。

设置 OC2x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC2x 的波形。当 COM2x=1 时，发生比较匹配时会翻转 OC2x 信号，这种情况下波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

输出比较单元可以用来产生中断，但是在普通模式下不推荐使用中断，这样会占用太多 CPU 的时间。

CTC 模式

设置 WGM2[2:0]=2 时，定时计数器 2 进入 CTC 模式，计数的最大值 TOP 为 OCR2A。在这个模式下，计数方式为每一个计数时钟加一递增，当计数器的数值 TCNT2 等于 TOP 时计数器清零。OCR2A 定义了计数的最大值，亦即计数器的分辨率。这个模式使得用户可以很容易的控制比较匹配输出的频率，也简化了外部事件计数的操作。

当计数器到达计数的最大值时，输出比较匹配标志 OCF2 被置位，相应的中断使能置位时将会产生中断。在中断服务程序里可以更新 OCR2A 寄存器即计数的最大值。在这个模式下 OCR2A 没有使用双缓冲，在计数器以无预分频器或很低的预分频器工作下将最大值更新为接近最小值的时候要小心。如果写入 OCR2A 的数值小于当时的 TCNT2 值时，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到 TOP，然后再从 BOTTOM 开始计数到 OCR2A 值。和普通模式一样，计数值回到 BOTTOM 的计数时钟里置位 TOV2 标志。设置 OC2x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC2x 的波形。当 COM2x=1 时，发生比较匹配时会翻转 OC2x 信号，这种情况下波形的频率可以用下面的公式来计算：

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

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。从公式可以看出，当设置 OCR2x 为 0x0 且无预分频器时，可以获得最大频率为 $f_{sys}/2$ 的输出波形。

快速 PWM 模式

设置 WGM2[2:0]=3 或 7 时，定时计数器 2 进入快速 PWM 模式，可以用来产生高频的 PWM 波形，计数最大值 TOP 分别为 MAX（0xFF）或 OCR2A。快速 PWM 模式和其他 PWM 模式不同在于它是单向操作。计数器从最小值 0x00 累加到 TOP 后又回到 BOTTOM 重新计数。当计数值 TCNT2 到达 OCR2x 或 BOTTOM 时，输出比较信号 OC2x 会被置位或清零，取决于比较输出模式 COM2x 的设置，详情见寄存器描述。由于采用单向操作，快速 PWM 模式的操作频率是采用双向操作的相位修正 PWM 模式的两倍。高频特性使得快速 PWM 模式适用于功率调节，整流以及 DAC 应用。高频信号可以减小外部元器件（电感电容等）的尺寸，从而降低系统成本。

当计数值到达最大值时，定时计数器溢出标志 TOV2 将会被置位，并把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR2x 寄存器。

设置 OC2x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC2x 的波形。波形的频率可用下面的公式来计算：

$f_{oc2xfpwm} = f_{sys}/(N*(1+TOP))$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

当 TCNT2 和 OCR2x 发生比较匹配时，波形产生器就置位（清零）OC2x 信号，当 TCNT2 被清零时，波形产生器就清零（置位）OC2x 信号，以此来产生 PWM 波。由此 OCR2x 的极值将会产生特殊的 PWM 波形。当 OCR2x 设置为 0x00 时，输出的 PWM 为每(1+TOP)个计数时钟里有一个窄的尖峰脉冲。当 OCR2x 设置为最大值时，输出的波形为持续的高电平或低电平。

相位修正 PWM 模式

当设置 WGM2[2:0]=1 或 5 时，定时计数器 2 进入相位修正 PWM 模式，计数的最大值 TOP 分别为 MAX（0xFF）或 OCR2A。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT2 与 OCR2x 匹配时，输出比较信号 OC2x 将会被清零或置位，取决于比较输出模式 COM2x 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV2 标志，当计数到达 TOP 时把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR2x 寄存器。

设置 OC2x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC2x 的波形。波形的频率可用下面的公式来计算：

$f_{oc2xpcpwm} = f_{sys}/(N*TOP*2)$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT2 与 OCR2x 匹配时，波形产生器就清零（置位）OC2x 信号。在递减计数过程中，当 TCNT2 与 OCR2x 匹配时，波形产生器就置位（清零）OC2x 信号。由此 OCR2x 的极值会产生特殊的 PWM 波。当 OCR2x 设置为最大值或最小值时，OC2x 信号输出会一直保持低电平或高电平。

为了保证输出 PWM 波在最小值两侧的对称性，在没有发生比较匹配时，有两种情况下也会翻转 OC2x 信号。第一种情况是，当 OCR2x 的值由最大值 0xFF 改变为其他数据时。当 OCR2x 为最大值，计数值达到最大时，OC2x 的输出与前面降序计数时比较匹配的结果相同，即保持 OC2x 不变。此时会更新比较值为新的 OCR2x 的值（非 0xFF），OC2x 的值会一直保持，直

到升序计数时发生比较匹配而翻转。此时 OC2x 信号并不以最小值为中心对称，因此需要在 TCNT2 到达最大值时翻转 OC2x 信号，此即没有发生比较匹配时翻转 OC2x 信号的第一种情况。第二种情况是，当 TCNT2 从比 OCR2x 高的值开始计数时，因而会丢失一次比较匹配，从而引起不对称情形的产生。同样需要翻转 OC2x 信号去实现最小值两侧的对称性。

TC2 的异步操作方式

当位于 ASSR 寄存器的 AS2 位为“1”时，TC2 工作在异步模式，计数器的时钟源来自于外部定时计数器的振荡器。异步模式下 TC2 的操作要考虑如下几点。

- 在同步和异步模式之间的转换有可能造成 TCNT2、OCR2A、OCR2B、TCCR2A 和 TCCR2B 数据的损坏。安全的操作步骤如下所示：
  1. 清零 OCIE2A，TOIE2 和 OCIE2B 寄存器位来关闭 TC2 的中断；
  2. 置位 AS2 位选择合适的时钟源；
  3. 对 TCNT2、OCR2A、TCCR2A、OCR2B 和 TCCR2B 寄存器写入新的数据；
  4. 切换到异步模式时，需等待 TCN2UB、OCR2AUB、TCR2AUB、OCR2BUB 和 TCR2BUB 位清零；
  5. 清零 TC2 的中断标志位；
  6. 使能需要使用的中断。
- 振荡器最好使用 32.768KHz 的手表晶振。系统时钟频率必须比晶振频率高 4 倍以上。
- CPU 写 TCNT2、OCR2A、TCCR2A、OCR2B 和 TCCR2B 时，硬件会将数据先放入暂存器，两个 TOSC1 时钟上升沿后才锁存到对应的寄存器中。在数据从暂存器锁存到目的寄存器之前不能执行新的数据写入操作。各个寄存器都有各自独立的暂存器，因此写 TCNT2 并不会干扰写 OCR2。异步状态寄存器 ASSR 用来检查数据是否已经写入到目的寄存器。
- 如果使用 TC2 作为 MCU 休眠模式的唤醒条件，则在各个寄存器更新结束之前不能进入休眠模式，否则 MCU 可能会在 TC2 设置生效之前进入休眠模式，从而 TC2 无法唤醒系统。
- 如果使用 TC2 作为 MCU 休眠模式的唤醒条件，必须注意重新进入休眠模式的过程。中断逻辑需要一个 TOSC1 时钟周期进行复位，如果从唤醒到重新进入休眠的时间小于一个 TOSC1 时钟周期，中断将不再发生，器件也无法唤醒。推荐采用如下的操作方法：
  1. 对各个寄存器写入合适的数据；
  2. 等待 ASSR 相应的更新忙标志位清零；
  3. 进入休眠模式。
- 若选择了异步工作模式，TC2 的振荡器将一直工作，除非进入掉电模式。用户必须注意，此振荡器的稳定时间可能长达 1 秒钟，因此，建议用户在使能 TC2 的振荡器后至少等待 1 秒钟后再使用 TC2 的异步工作模式。
- 异步工作模式时休眠模式下唤醒的过程：中断条件满足后，在下一个定时器时钟启动唤醒过程。也就是说，在处理器可以读取计数器的数值之前计数器至少又累加了一个时钟。唤醒后 MCU 执行中断服务程序，之后开始执行 SLEEP 语句之后的程序。
- 从休眠模式唤醒之后短时间内读取 TCNT2 的值可能返回不正确的数据。因为 TCNT2 是由异步的 TOSC1 时钟驱动的，而读取 TCNT2 必须通过一个内部系统时钟同步的寄存器来完成，同步发生于每个 TOSC1 的上升沿。从休眠模式唤醒后系统时钟重新激活，读取的 TCNT2 数值为进入休眠模式之前的值，直到下一个 TOSC1 上升沿的到来才会更新。从休眠模式唤醒时 TOSC1 的相位完全不可预测，而与唤醒时间有关。因此，读取 TCNT2 值的推荐序列为：
  1. 写一个任意数值到 OCR2A 或 TCCR2A；

2. 等待相应的更新忙标志位被清零；
3. 读取 TCNT2。

- 异步模式下，中断标志位的同步需要 3 个系统时钟周期加 1 个定时器周期。在 MCU 可以读取引起中断标志置位的计数器数值之前计数器至少又累加了一个时钟。输出比较信号的变化与定时器时钟同步，而不是系统时钟。

TC2 的预分频器

TC2 预分频器的输入时钟称为 clkt2s，由位于 ASSR 寄存器的 AS2 位来选择内部系统时钟 clkio 或者外部 TOSC1 时钟源，缺省为与系统时钟 clkio 相连接。若 AS2 置位，TC2 将由 TOSC1 异步驱动。当 TOSC1 引脚和 TOSC2 引脚外接一个 32.768KHz 的钟表晶振，TC2 可用作 RTC 计数器。不推荐在 TOSC1 引脚上直接施加外部时钟信号。

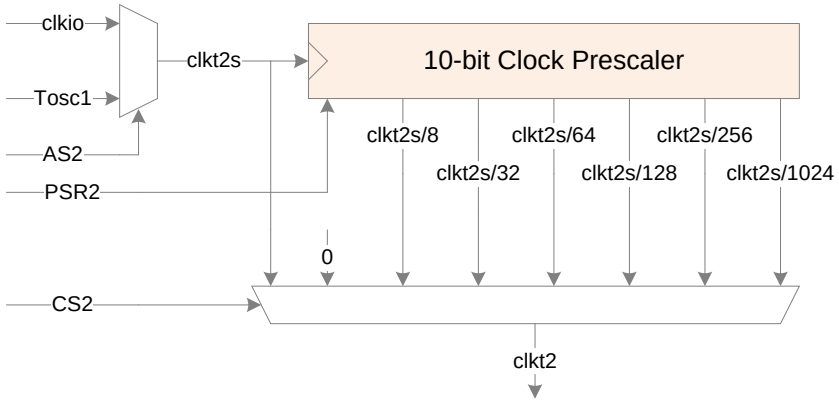


Figure 1 TC2 预分频器结构图

上图为 TC2 预分频器，如图所示，可能的预分频选项有：clkt2s/8, clkt2s/32, clkt2s/64, clkt2s/128, clkt2s/256,和 clkt2s/1024。此外还可以选择 clkt2s 和 0（停止计数）。置位 SFIOR 寄存器的 PSR2 位将复位预分频器，从而允许用户从可预测的预分频器开始工作。

寄存器定义

TC2 寄存器列表

寄存器	地址	默认值	描述
TCCR2A	0xB0	0x00	TC2 控制寄存器 A
TCCR2B	0xB1	0x00	TC2 控制寄存器 B
TCNT2	0xB2	0x00	TC2 计数值寄存器
OCR2A	0xB3	0x00	TC2 输出比较寄存器 A
OCR2B	0xB4	0x00	TC2 输出比较寄存器 B
ASSR	0xB6	0x00	TC2 异步状态寄存器
TIMSK2	0x70	0x00	定时计数器中断屏蔽寄存器
TIFR2	0x37	0x00	定时计数器中断标志寄存器

TCCR2 A–TC2 控制寄存器 A

TCCR2 A–TC2 控制寄存器 A								
地址: 0xB0					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
R/W	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	描述
7	COM2A1	TC2 比较匹配输出 A 模式控制高位。 COM2A1 和 COM2A0 一起组成输出比较模式控制 COM2A[1:0]，控制 OC2A 的输出波形。如果 COM2A 的 1 位或者 2 位都置位，输出比较波形占据着 OC2A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM2A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
6	COM2A0	TC2 比较匹配输出 A 模式控制低位。 COM2A0 和 COM2A1 一起组成输出比较模式控制 COM2A[1:0]，控制 OC2A 的输出波形。如果 COM2A 的 1 位或者 2 位都置位，输出比较波形占据着 OC2A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM2A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
5	COM2B1	TC2 比较匹配输出 B 模式控制高位。 COM2B1 和 COM2B0 一起组成输出比较模式控制 COM2B[1:0]，控制 OC2B 的输出波形。如果 COM2B 的 1 位或者 2 位都置位，输出比较波形占据着 OC2B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM2B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
4	COM2B0	TC2 比较匹配输出 B 模式控制低位。 COM2B0 和 COM2B1 一起组成输出比较模式控制 COM2B[1:0]，控制 OC2B 的输出波形。如果 COM2B 的 1 位或者 2 位都置位，输出比较波形占据着 OC2B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM2B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
3:2	-	保留。
1	WGM21	TC2 波形产生模式控制高位。 WGM20 和 WGM21，WGM22 一起组成波形产生模式控制 WGM2[2:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
0	WGM20	TC2 波形产生模式控制低位。 WGM21 和 WGM20，WGM22 一起组成波形产生模式控制 WGM2[2:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

TCCR2B –TC2 控制寄存器 B

TCCR2B –TC2 控制寄存器 B								
地址: 0xB1					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20
R/W	W	W	-	-	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	FOC2A	TC2 强制输出比较 A 控制位。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC2A 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF2A 标志，也不会重载或清零定时器，但是输出引脚 OC2A 将被按照 COM2A 的设置相应的更新，就跟真的发生了比较匹配一样。 读取 FOC2A 的返回值一直为零。
6	FOC2B	TC2 强制输出比较 B 控制位。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC2B 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF2B 标志，也不会重载或清零定时器，但是输出引脚 OC2B 将被按照 COM2B 的设置相应的更新，就跟真的发生了比较匹配一样。 读取 FOC2B 的返回值一直为零。
5:4	-	保留。
3	WGM22	TC2 波形产生模式控制高位。 WGM22 和 WGM20，WGM21 一起组成波形产生模式控制 WGM2[2:0]，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。
2	CS22	TC2 时钟选择控制高位。 用于选择定时计数器 2 的时钟源。
1	CS21	TC2 时钟选择控制中位。 用于选择定时计数器 2 的时钟源。
0	CS20	TC2 时钟选择控制低位。 用于选择定时计数器 2 的时钟源。

CS2[2:0]	描述
0	无时钟源，停止计数
1	clk_{t2s}
2	clk_{t2s}/8，来自预分频器
3	clk_{t2s}/32，来自预分频器
4	clk_{t2s}/64，来自预分频器
5	clk_{t2s}/128，来自预分频器
6	clk_{t2s}/256，来自预分频器
7	clk_{t2s}/1024，来自预分频器

下表为非 PWM 模式（即普通模式和 CTC 模式）下，比较输出模式对输出比较波形的控制。

Table 1 非 PWM 模式下 OC2x 比较输出模式控制

COM2x[1:0]	描述
0	OC2x 断开，通用 IO 口操作
1	比较匹配时翻转 OC2x 信号
2	比较匹配时清零 OC2x 信号
3	比较匹配时置位 OC2x 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

Table 2 快速 PWM 模式下 OC2x 比较输出模式控制

COM2x[1:0]	描述
0	OC2x 断开，通用 IO 口操作
1	保留
2	比较匹配时清零 OC2x 信号，最大值匹配时置位 OC2x 信号
3	比较匹配时置位 OC2x 信号，最大值匹配时清零 OC2x 信号

下表为相位修正模式下比较输出模式对输出比较波形的控制。

Table 3 相位修正 PWM 模式下 OC2x 比较输出模式控制

COM2x[1:0]	描述
0	OC2x 断开，通用 IO 口操作
1	保留
2	升序计数下比较匹配时清零 OC2x 信号，降序计数下比较匹配时置位 OC2x 信号
3	升序计数下比较匹配时置位 OC2x 信号，降序计数下比较匹配时清零 OC2x 信号

下表为波形产生模式控制。

Table 4 波形产生模式控制

WGM2[2:0]	工作模式	TOP 值	更新 OCR2x 时刻	置位 TOV2 时刻
0	Normal	0xFF	立即	MAX
1	PCPWM	0xFF	TOP	BOTTOM
2	CTC	OCR2A	立即	MAX
3	FPWM	0xFF	TOP	MAX
4	保留	-	-	-
5	PCPWM	OCR2A	TOP	BOTTOM
6	保留	-	-	-
7	FPWM	OCR2A	TOP	TOP

TCNT2 –TC2 计数值寄存器

TCNT2 –TC2 计数值寄存器								
地址: 0xB2					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	TCNT2	TC2 计数值寄存器。 通过 TCNT2 寄存器可以直接对计数器的 8 为计数值进行读写访问。 CPU 对 TCNT2 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT2 寄存器的值与 OCR2 的值一致而不会引发中断。 如果写入 TCNT2 的数值等于或绕过 OCR2 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT2。CPU 写计数器比清零或加减操作的优先级高。

OCR2A – TC2 输出比较寄存器 A

OCR2A – TC2 输出比较寄存器 A								
地址: 0xB3					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	OCR2A	TC2 输出比较寄存器 A。 OCR2A 包含一个 8 位的数据，不间断地与计数器数值 TCNT2 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC2A 引脚上产生波形。 当使用 PWM 模式时，OCR2A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR2A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR2A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR2A 本身。

OCR2B – TC2 输出比较寄存器 B

OCR2B – TC2 输出比较寄存器 B								
地址: 0xB4					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	OCR2B	TC2 输出比较 B 寄存器。 OCR2B 包含一个 8 位的数据，不间断地与计数器数值 TCNT2 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC2B 引脚上产生波形。 当使用 PWM 模式时，OCR2B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR2B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。使用双缓冲功能时，CPU 访问的是 OCR2B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR2B 本身。

TIMSK2 – TC2 中断屏蔽寄存器

TIMSK2 – TC2 中断屏蔽寄存器								
地址: 0x70					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:3		保留。
2	OCIE2B	TC2 输出比较 B 匹配中断使能位。 当 OCIE2B 位为“1”，且全局中断置位，TC2 输出比较 B 匹配中断使能。当比较匹配发生时，即 TIFR2 中 OCF2B 位被置位时，中断产生。 当 OCIE2B 位为“0”时，TC2 输出比较 B 匹配中断被禁止。
1	OCIE2A	TC2 输出比较 A 匹配中断使能位。 当 OCIE2A 位为“1”，且全局中断置位，TC2 输出比较 A 匹配中断使能。当比较匹配发生时，即 TIFR2 中 OCF2A 位被置位时，中断产生。 当 OCIE2A 位为“0”时，TC2 输出比较 A 匹配中断被禁止。
0	TOIE2	TC2 溢出中断使能位。 当 TOIE2 位为“1”，且全局中断置位，TC2 溢出中断使能。当 TC2 发生溢出，即 TIFR2 中的 TOV2 位被置位时，中断产生。当 TOIE2 位为“0”时，TC2 溢出中断被禁止。

TIFR2 – TC2 中断标志寄存器

TIFR2 – TC2 中断标志寄存器								
地址: 0x37					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	OCF2B	OCF2A	TOV2
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

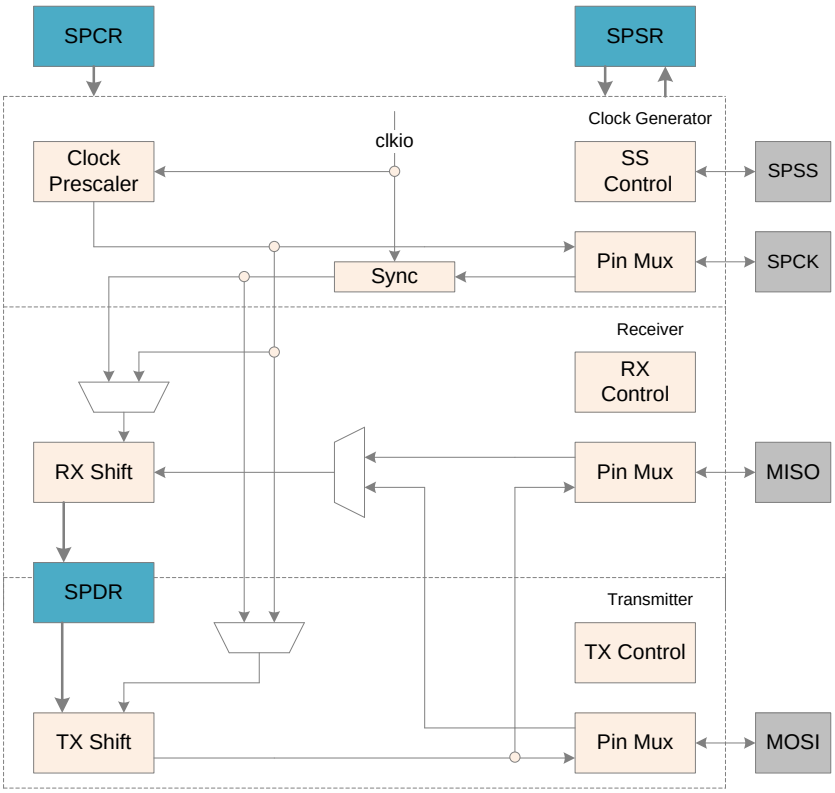
Bit	Name	描述
7:3	-	保留。
2	OCF2B	TC2 输出比较 B 匹配标志位。 当 TCNT2 等于 OCR2B 时，比较单元就给出匹配信号，并置位比较标志 OCF2B。若此时输出比较 B 中断使能 OCIE2B 为“1”且全局中断标志置位，则会产生输出比较 B 中断。执行此中断服务程序时 OCF2B 将自动清零，或对 OCF2B 位写“1”也可清零该位。
1	OCF2A	TC2 输出比较 A 匹配标志位。 当 TCNT2 等于 OCR2A 时，比较单元就给出匹配信号，并置位比较标志 OCF2A。若此时输出比较 A 中断使能 OCIE2A 为“1”且全局中断标志置位，则会产生输出比较 A 中断。执行此中断服务程序时 OCF2A 将自动清零，或对 OCF2A 位写“1”也可清零该位。
0	TOV2	TC2 溢出标志位。 当计数器发生溢出时，置位溢出标志 TOV2。若此时溢出中断使能 TOIE2 为“1”且全局中断标志置位，则会产生溢出中断。执行此中断服务程序时 TOV2 将自动清零，或对 TOV2 位写“1”也可清零该位。

SPI 串行外设接口

- 全双工，三线同步数据传输
- 主机或从机操作
- 最低位或最高位优先传输
- 7 种可编程的比特率
- 发送结束中断标志
- 写入冲突标志保护机制
- 可从闲置模式唤醒
- 主机操作时具有倍速模式
- 支持主机双线输入模式

综述

SPI 主要包括三个部分：时钟预分频器，时钟检测器，从机选择检测器，发送器和接收器。控制和状态寄存器由这三个部分共享。时钟预分频器只工作在主机操作模式下，由比特率控制位来选择分频系数，从而产生相应的分频时钟，输出到 SPCK 引脚上。时钟检测器只工作在从机操作模式下，检测从 SPCK 引脚上输入的时钟沿，根据 SPI 的数据传输模式对发送和接收移位寄存器进行移位操作。从机选择检测器对从机选择信号 SPSS 进行检测，得到传输的状态来控制发送器和接收器的操作。发送器由一个移位寄存器和发送控制逻辑组成。接收器由一个移位寄存器，一个接收缓冲器和接收控制逻辑组成。



SPI 结构图

时钟产生

时钟产生逻辑分为主机时钟预分频器和从机时钟检测器，分别工作在主机操作和从机操作模式下。时钟预分频器由比特率控制位和倍速控制位来选择分频系数，产生相应的分频时钟（共有 7 种可选的分频系数，详细信息见寄存器描述），输出到 SPCK 引脚为通信提供时钟，同时为内部发送和接收移位寄存器提供移位时钟。时钟检测器对输入时钟 SPCK 进行边沿检测，根据 SPI 的数据传输模式对发送器和接收器进行移位操作。为保证对时钟信号的正确采样，SPCK 时钟的高电平和低电平的宽度均须大于 2 个系统时钟周期。

发送和接收

SPI 模块在单线模式下支持同时发送和接收，在双线模式下只支持主机双线接收。

单线发送和接收

SPI 的主机将需要通信的从机选择信号 SPSS 拉低，即可启动一次传输过程。主机和从机将需要传输的数据准备好，主机在时钟信号 SPCK 上产生时钟脉冲以交换数据，主机的数据从 MOSI 移出，从 MISO 移入，从机的数据从 MISO 移出，从 MOSI 移入，交换完数据后主机拉高 SPSS 信号即可完成通信。

当配置为主机时，SPI 模块并不控制 SPSS 引脚，必须由用户软件来处理。软件拉低 SPSS 引脚，选择要通信的从机，启动传输。软件将需要传输的数据写入 SPDR 寄存器即会启动时钟发生器，硬件产生通信的时钟，并把 8 位数据移出给从机，同时把从机的数据移入。移位一个字节的数据后，停止时钟发生器，并置位传输完成标志 SPIF。软件可再次写入数据到 SPDR 寄存器来继续传输下一个字节，也可以拉高 SPSS 信号来结束当前传输。最后进来的数据将保存在接收缓冲器中。

当配置为从机时，只要 SPSS 信号一直为高，SPI 模块将保持睡眠状态，并保持 MISO 引脚为三态。这时软件可更新 SPDR 寄存器的内容。即使此时 SPCK 引脚上有输入时钟脉冲，SPDR 的数据也不会被移出，直至 SPSS 信号被拉低。当一个字节的数据传输完成之后，硬件置位传输完成标志 SPIF。此时软件在读取移入的数据之前可继续往 SPDR 寄存器写入数据，最后进来的数据将保存在接收缓冲器中。

SPI 模块在发送方向只有一个缓冲器，而在接收方向有两个缓冲器。在发送数据时，一定要等到移位过程全部结束之后才能对 SPDR 寄存器进行写操作。而在接收数据时，需要在下一个字节移位过程结束之前通过访问 SPDR 寄存器读取已经接收到的字符，否则前一个字节将丢失。

主机双线接收

SPI 模块的双线模式只在主机操作模式下有效，与单线模式的不同在于 MOSI 和 MISO 都用于主机接收数据，每一个 SPCK 时钟脉冲同时接收 2 个比特的数据（MISO 线上的数据在前，MOSI 线上的数据在后），接收完两个字节的数据之后硬件置位传输完成标志 SPIF，数据保存到接收缓冲器和移位寄存器中。此时软件须读取 SPDR 寄存器两次来得到所接收的两个字节的数据。需要注意的是，虽然双线模式下主机不向从机发送数据，软件仍需要往 SPDR 寄存器写入数据来启动时钟发生器产生通信时钟，写入一次 SPDR 寄存器即可接收两个字节

的数据。

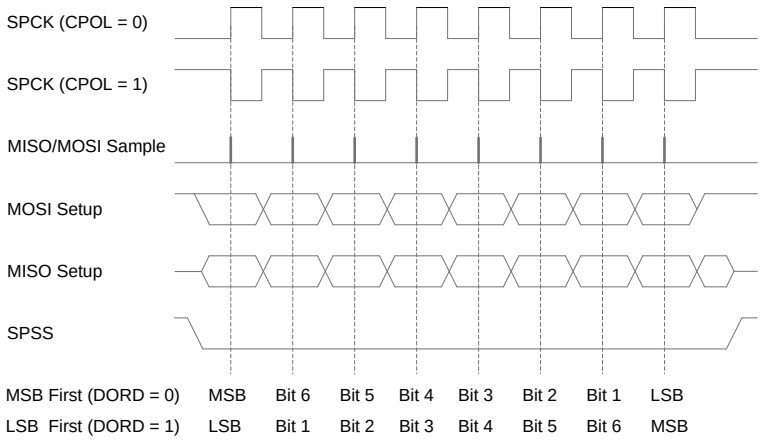
数据模式

单线模式下，相对于串行数据，SPI 有 4 种 SPCK 相位和极性的组合方式，由 CPHA 和 CPOL 来控制，如下表所示。

CPHA 和 CPOL 选择数据传输模式

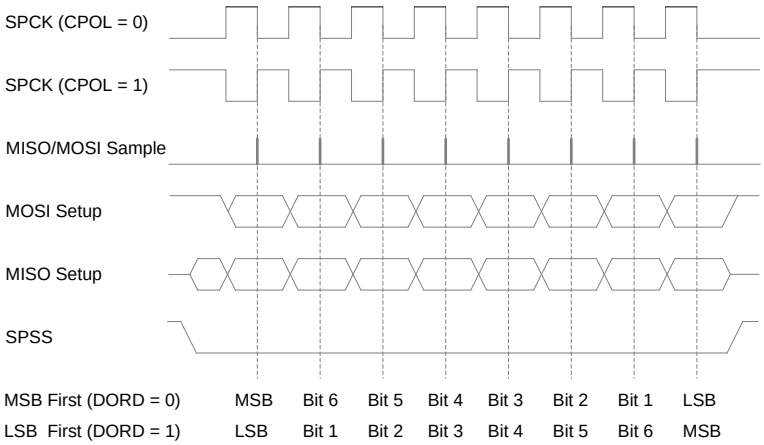
CPOL	CPHA	起始沿	结束沿	SPI 模式
0	0	采样（上升沿）	设置（下降沿）	0
0	1	设置（上升沿）	采样（下降沿）	1
1	0	采样（下降沿）	设置（上升沿）	2
1	1	设置（下降沿）	采样（上升沿）	3

当 CPHA = 0 时，数据采样和设置的时钟沿如下图所示：



CPHA 为“0”时 SPI 数据传输模式

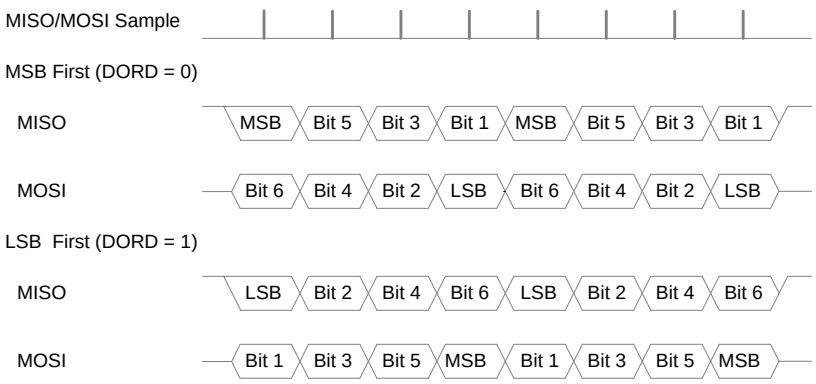
当 CPHA = 1 时，数据采样和设置的时钟沿如下图所示：



CPHA 为“1”时 SPI 数据传输模式

双线模式下，MISO 和 MISO 均用做主机的输入，数据采样的时刻仍由数据传输模式决定，

采样的方式如下图所示：



主机模式下 DUAL 为“1”时 SPI 数据采样模式

SPSS 引脚功能

当配置为从机时，从机选择信号 SPSS 引脚总是作为输入。当 SPSS 引脚保持为低时，SPI 接口被激活，MISO 引脚成为输出引脚（软件进行相应的端口配置），其它引脚均为输入。当 SPSS 引脚保持为高时，SPI 模块被复位，且不再接收数据。SPSS 引脚对于数据包/字节的同步非常有用，可以使从机的位计数器和主机的时钟发生器同步。当 SPSS 拉高时，SPI 从机立即复位接收和发送逻辑，并丢弃移位寄存器里不完整的数据。

当配置为主机时，用户软件可以决定 SPSS 引脚的方向。
若 SPSS 配置为输出，则它可以用来驱动从机的 SPSS 引脚。若 SPSS 配置为输入，必须保持为高以保证主机的正常工作。当配置为主机且 SPSS 引脚为输入，外部电路拉低 SPSS 引脚时，SPI 模块会认为是另外一个主机选择自己作为从机并开始传输数据。为了防止总线冲突，SPI 模块将进行如下动作：

1. 清零位于 SPCR 寄存器的 MSTR 位，转换为从机，从而 MOSI 和 SPCK 变为输入；
2. 置位位于 SPSR 寄存器的 SPIF 位，若中断使能则产生 SPI 中断。

因此，使用中断方式处理 SPI 主机的数据传输，并且存在 SPSS 被拉低的可能性时，中断服务程序应该检查 MSTR 位是否为“1”。若被清零，软件须将其置位，以重新使能 SPI 主机模式。

SPI 初始化

进行通信之前首先要对 SPI 进行初始化。初始化过程通常包括主机从机操作的选择，数据传输模式的设定，比特率的选择，以及各个引脚的方向控制等。其中主机和从机操作下引脚方向的控制各不相同，如下表所示：

引脚方向控制

引脚	主机模式下的方向	从机模式下的方向
MOSI	用户软件定义	输入
MISO	输入	用户软件定义
SPCK	用户软件定义	输入
SPSS	用户软件定义	输入

SPI 主机初始化

SPI 主机模式的初始化过程如下：

1. 置位 MSTR 位，设置比特率选择控制位，数据传输模式，数据传输次序，中断使能与否，以及双线使能与否；
2. 设置 MOSI 和 SPCK 引脚为输出；
3. 置位 SPE 位。

主机模式下，当不希望 SPI 模块被别的主机选择作为从机使用时，可设置 SPSS 引脚为输出。

SPI 从机初始化

SPI 从机模式初始化过程如下：

1. 清零 MSTR 位，设置数据传输模式，数据传输次序，中断使能与否；
2. 设置 MISO 引脚为输出；
3. 置位 SPE 位。

寄存器定义

SPI 寄存器列表

寄存器	地址	默认值	描述
SPCR	0x4C	0x00	SPI 控制寄存器
SPSR	0x4D	0x00	SPI 状态寄存器
SPDR	0x4E	0x00	SPI 数据寄存器

SPCR – SPI 控制寄存器

SPCR – SPI 控制寄存器								
地址: 0x4C					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	SPIE	SPI 中断使能位。 当设置 SPIE 位为“1”时，SPI 中断被使能。当位于 SPSR 寄存器中的 SPIF 位被置位且全局中断使能时，产生 SPI 中断。 当设置 SPIE 位为“0”时，SPI 中断被禁止。
6	SPE	SPI 使能位。 当设置 SPE 位为“1”时，SPI 模块被使能。进行任何 SPI 操作之前必须置位 SPE。 当设置 SPE 位为“0”时，SPI 模块被禁止。
5	DORD	数据次序控制位。 当设置 DORD 位为“1”时，数据的 LSB 首先发送。

		当设置 DORD 位为“0”时，数据的 MSB 首先发送。
4	MSTR	主机从机选择控制位。 当设置 MSTR 位为“1”时，选择为主机模式。 当设置 MSTR 位为“0”时，选择为从机模式。 主机模式下，SPSS 引脚配置为输入且被拉低时，MSTR 位将被清零，位于 SPSR 寄存器的 SPIF 被置位，用户必须重新设置 MSTR 进入主机模式。
3	CPOL	时钟极性控制位。 当设置 CPOL 位为“1”时，空闲状态下 SPCK 为高电平。 当设置 CPOL 位为“0”时，空闲状态下 SPCK 为低电平。

CPOL	起始沿	结束沿
0	上升沿	下降沿
1	下降沿	上升沿

2	CPHA	时钟相位控制位。 当设置 CPHA 位为“1”时，起始沿设置数据，结束沿采样数据。 当设置 CPHA 位为“0”时，起始沿采样数据，结束沿设置数据。

CPHA	起始沿	结束沿
0	采样	设置
1	设置	采样

1	SPR1	时钟速率选择位 1。 SPR1 和 SPR0 用来选择 SPI 传输的时钟速率。具体控制方式见 SPCK 和系统时钟的关系表格。
0	SPR0	时钟速率选择位 0。 SPR1 和 SPR0 用来选择 SPI 传输的时钟速率。具体控制方式见 SPCK 和系统时钟的关系表格。

SPSR – SPI 状态寄存器

SPSR – SPI 状态寄存器								
地址: 0x4D					默认值: 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	描述
7	SPIF	SPI 中断标志位。 串行传输结束后置位 SPIF 标志，主机模式下，配置 SPSS 引脚为输入且被拉低时，SPIF 也将被置位。若此时 SPCR 寄存器的 SPIE 位和全局中断使能位都被置位，SPI 中断产生。进入中断服务程序后 SPIF 位自动清零，或者通过先读取 SPSR 寄存器再访问 SPDR 寄存器来清零 SPIF 位。
6	WCOL	写冲突标志位。 在数据传输的过程中写 SPDR 寄存器将置位 WCOL 位。WCOL 位可以通过先读取

		SPSR 寄存器再访问 SPDR 寄存器来清零。
5	-	保留。
4	-	保留。
3	-	保留。
2	DUAL	双线模式控制位。 当设置 DUAL 位为“1”时，使能 SPI 双线传输模式。 当设置 DUAL 位为“0”时，禁止 SPI 双线传输模式。 双线传输模式只在 SPI 主机模式下有效，MISO 和 MOSI 均用作主机数据输入，数据的传输方式见主机双线接收和数据模式章节描述。
1	-	保留。
0	SPI2X	SPI 倍速控制位。 当设置 SPI2X 位为“1”时，SPI 的传输速度加倍。 当设置 SPI2X 位为“0”时，SPI 的传输速度不加倍。 具体控制方式见 SPCK 和系统时钟的关系表格。

SPDR – SPI 数据寄存器

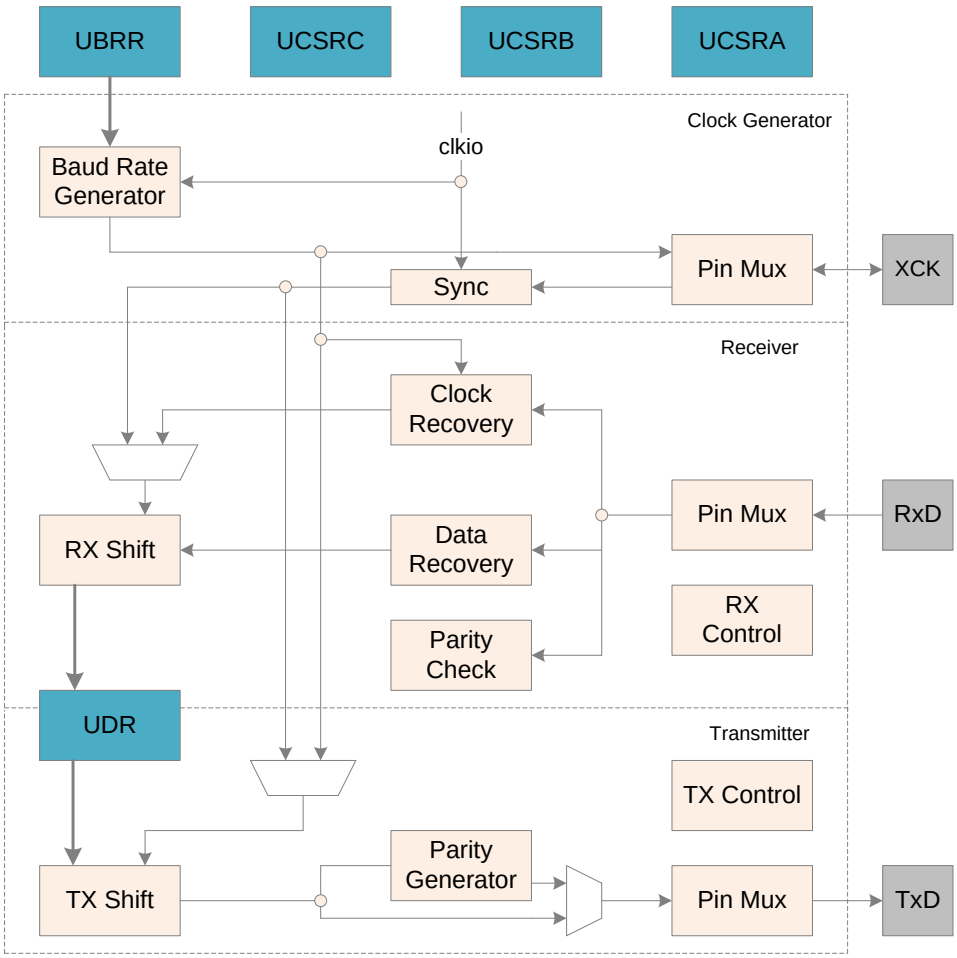
SPDR – SPI 数据寄存器								
地址: 0x4E					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	SPDR	SPI 发送和接收的数据。 SPI 发送数据和接收数据共享 SPI 数据寄存器 SPDR。将数据写入 SPDR 即写入发送数据移位寄存器器，从 SPDR 读取数据即读取接收数据缓冲器。

USART0 - 通用同步/异步串行收发器

- 全双工操作 (独立的串行接收和发送寄存器)
- 异步或同步操作
- 主机或从机操作
- 高精度的波特率发生器
- 支持 5，6，7，8，或 9 个数据位和 1，或 2 个停止位
- 硬件支持的奇偶产生和校验机制
- 数据过速检测
- 帧错误检测
- 噪声滤波，包括错误的起始位检测以及数字低通滤波器
- 三个独立的中断：发送结束中断，发送数据寄存器空中断以及接收结束中断
- 多处理器通信模式
- 倍速异步通信模式

综述



USART 结构图

USART 主要包括三个部分：时钟发生器，发送器和接收器。控制和状态寄存器由这三个部分共享。时钟发生器由波特率发生器和同步从机操作模式下外部输入时钟的同步逻辑组成。

XCK 引脚只用于同步传输模式。发送器包括一个写数据缓冲器，串行移位寄存器，奇偶发生器以及处理不同帧格式所需的控制逻辑。写数据缓冲器允许连续发送数据而不会在数据帧之间引入延迟。接收器具有时钟和数据恢复单元，用于异步数据的接收。除了恢复单元，接收器还包括奇偶校验，控制逻辑，串行移位寄存器和一个两级接收缓冲器 UDR。接收器支持与发送器相同的帧格式，而且可以检测帧错误，数据过速和奇偶校验错误。

时钟产生

时钟产生逻辑为发送器和接收器产生基础时钟。USART 支持 4 种模式的时钟：正常的异步模式，倍速的异步模式，主机同步模式，以及从机同步模式。USCRC 的 UMSEL 位用于选择同步或异步模式。USCRA 的 U2X 位控制异步模式下的倍速使能。仅在同步模式下有效的 XCK 引脚的数据方向寄存器(与 IO 复用)决定了时钟源是由内部产生(主机模式)还是外部产生(从机模式)。

波特率发生器

波特率寄存器 UBRR 和降序计数器连接在一起作为 USART 的可编程的预分频器或波特率发生器。降序计数器工作在系统时钟(f_{sys})下，当其计数到零或 UBRRL 寄存器被写时，会自动加载 UBRR 寄存器的值。当计数到零时产生一个时钟，该时钟作为波特率发生器的输出时钟，频率为 f_{sys}/(UBRR+1)。

下表给出了各种工作模式下计算波特率（位/秒）以及 UBRR 值的公式。

工作模式	波特率计算公式(1)	UBRR 值计算公式
异步正常模式	BAUD = f_{sys}/(16*(UBRR+1))	UBRR = f_{sys}/(16*BAUD) - 1
异步倍速模式	BAUD = f_{sys}/(8*(UBRR+1))	UBRR = f_{sys}/(8*BAUD) - 1
同步主机模式	BAUD = f_{sys}/(2*(UBRR+1))	UBRR = f_{sys}/(2*BAUD) - 1

说明：

1. 波特率定义为每秒的位传输速度（bps）；
2. BUAD 为波特率，f_{sys} 为系统时钟，UBRR 为波特率寄存器 UBRRH 和 UBRRL 的组合值。

倍速工作模式

通过设定 UCSRA 寄存器的 U2X 位可以是传输速率加倍，该位只在异步工作模式下有效，同步工作模式下置该位为“0”。

设置该位将会把波特率分频器的分频值减半，有效地加倍异步通信的传输速率。在这种情况下，接收器只使用一半的采样数来对数据进行采样及时钟恢复，因此需要更精准的波特率设置和系统时钟。发送器则没有变化。

外部时钟

同步从机操作模式由外部时钟驱动。外部时钟经过同步寄存器和边沿检测器之后才被发送器和接收器使用，这一过程会引入两个系统时钟的延时，因此外部 XCK 的最大时钟频率由以下公式限制：

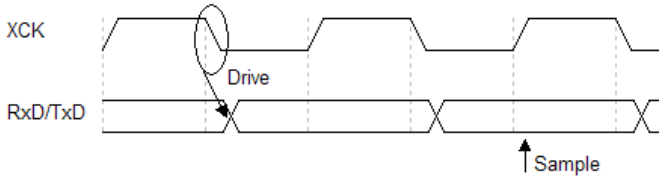
$$f_{XCK} < f_{sys}/4$$

要注意 fsys 有系统时钟的稳定性决定，为了防止因频率漂移而丢失数据，建议保留足够的裕量。

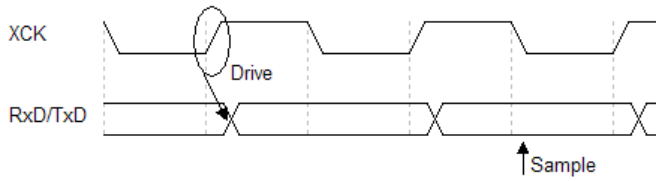
同步时钟操作

同步模式下，XCK 引脚被用于时钟输入（从机模式）或时钟输出（主机模式）。时钟的边沿与数据采样和数据变化关系的基本规律是：对数据输入端（RxD）采样所使用的时钟沿与数据输出端变化所使用的时钟沿是相反的。

UCPOL = 1



UCPOL = 0



同步模式下的 XCK 时序

如上图所示，当 UCPOL 值为“1”时，在 XCK 的下降沿改变数据输出，在 XCK 的上升沿进行数据采样；当 UCPOL 值为“0”时，在 XCK 的上升沿改变数据输出，在 XCK 的下降沿进行数据采样。

帧格式

一个串行数据帧由数据字加上同步位（起始位和停止位）以及用于纠错的奇偶校验位构成。USART 接受以下 30 种组合的数据帧格式：

- 1 个起始位
- 5、6、7、8 或 9 个数据位
- 无校验位、奇校验位或偶校验位
- 1 或 2 个停止位

数据帧以起始位开始，紧接着是数据字的最低位，接着是其它数据位，以数据字的最高位结束，最多成功传输 9 位数据。如果使能了校验，校验位将紧接着数据字，最后是停止位。当一个完整的数据帧传输后，可以立即传输下一个新的数据帧，或者使传输线处于空闲（高电平）状态。下图为可能的数据帧结构，方括号中的位是可选的。

USART 帧结构图

说明：

1）IDLE 通信线（RxD 或 TxD）上没有数据传输，线路空闲时必须为高电平
2）St 起始位，总是为低电平
3）0-8 数据位
4）P 校验位，奇校验或偶校验
5）Sp 停止位，总是为高电平

数据帧的结构由 UCSRB 和 UCSRC 寄存器中的 UCSZ[2:0]、UPM[1:0]和 USBS 设定。接收与发送使用相同的设置。设置的任何改变都可能破坏正在进行的数据传输。其中，UCSZ[2:0]确定了数据帧的数据位数，UPM[1:0]用于使能和确定校验的类型，USBS 设置帧有一位或两位结束位。接收器会忽略第二个停止位，因此帧错误只在第一个结束位为“0”时被检测到。

校验位计算

校验位的计算是对数据的各个位进行异或运算。如果选择了奇校验，则异或结果还需要取反。校验位与数据位的关系如下：

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

说明：
1）P_{even} 偶校验结果
2）P_{odd} 奇校验结果
3）d_n 第 n 个数据位

USART 初始化

进行通信之前首先要对 USART 进行初始化。初始化过程通常包括波特率的设定，帧结构的设定，以及根据需要使能接收器或发送器。对于中断驱动的 USART 操作，在初始化时要清零全局中断标志并禁止 USART 的所有中断。

在进行重新初始化比如改变波特率或帧结构时，必须确保没有数据传输。TXC 标志位可以用来检测发送器是否完成了所有传输，RXC 标志位可以用来检测接收缓冲器中是否还有数据未被读出。如果 TXC 标志位用作此用途，在每次发送数据之前（写 UDR 寄存器之前）必须清零 TXC 标志位。

发送器

置位 UCSRB 寄存器的 TXEN 位将使能 USART 的数据发送。使能后 TxD 引脚的通用 IO 功能即被 USART 功能所取代，成为发送器的串行输出。发送数据之前要设置好波特率、工作模式与帧格式。如果使用同步发送模式，施加于 XCK 引脚上的时钟信号即为数据发送的时钟。

发送 5 到 8 为数据的帧

将需要发送的数据加载到发送缓冲器中来启动数据发送。CPU 通过写 UDR 寄存器来加载数据。当发送移位寄存器可以发送新一帧数据的时候，缓冲器中的数据将转移到移位寄存器中。当移位寄存器处于空闲状态（没有正在进行的数据传输），或者前一帧数据的最后一个停止位发送完毕，它将加载新的数据。一旦移位寄存器加载了新的数据，它将按照既定的设置传输一个完整的帧。

发送 9 位数据的帧

如果发送 9 位数据的帧，应先将数据的第 9 位写入寄存器 UCSRB 的 TXB8 位，然后再将低 8 位数据写入发送数据寄存器 UDR。第 9 位数据在多机通信中用于表示地址帧，在同步通信中可以用于协议处理。

发送奇偶校验位

奇偶校验产生电路为串行数据帧生成相应的校验位。当校验位使能时（UPM1 = 1），发送控制逻辑电路会在数据字的最后一位与第一个停止位之间插入奇偶校验位。

发送标志位与中断处理

USART 发送器有两个标志位：USART 数据寄存器空标志 UDRE 和传输结束标志 TXC，两个标志位都可以产生中断。

数据寄存器空标志 UDRE 用来表示发送缓冲器是否可以写入一个新的数据。该位在发送缓冲器空时被置“1”，满时被置“0”。当 UDRE 位为“1”时，CPU 可以往数据寄存器 UDR 写入新的数据，反之则不能。

当 UCSRB 寄存器中的数据寄存器空中断使能位 UDRIE 为“1”时，只要 UDRE 被置位（且全局中断使能），就将产生 USART 数据寄存器空中断请求。对寄存器 UDR 执行写操作将清零 UDRE。当采用中断方式传输数据时，在数据寄存器空中断服务程序中必须写入一个新的数据到 UDR 以清零 UDRE，或者是禁止数据寄存器空中断。否则一旦该中断服务程序结束，一个新的中断将再次产生。

当整个数据帧被移出发送移位寄存器，同时发送寄存器中又没有新的数据时，发送结束标志 TXC 将被置位。当 UCSRB 上的发送结束中断使能位 TXCIE（且全局中断使能）置“1”时，随着 TXC 标志位被置位，USART 发送结束中断将被执行。一旦进入中断服务程序，TXC 标志位即被自动清零，CPU 也可以对该位写“1”来清零。

禁止发送器

当 TXEN 清零后，只有等所有的数据都发送完成以后发送器才能够真正禁止，即发送移位寄存器与发送缓冲寄存器中都没有要传送的数据。发送器禁止以后，TxD 引脚恢复其通用 IO 功能。

接收器

置位 UCSRB 寄存器的接收允许位（RXEN）即可启动 USART 接收器。使能后 RxD 引脚的通用 IO 功能被 USART 功能所取代，成为接收器的串行输入口。进行数据接收之前首先要设置好波特率、操作模式及帧格式。如果使用同步接收模式，XCK 引脚上的时钟被用为传输时钟。

接收 5 到 8 位数据的帧

一旦接收器检测到一个有效的起始位，便开始接受数据。起始位后的每一位数据都将以所设定的波特率或 XCK 时钟来进行接收，直到收到一帧数据的第一个停止位，第二个停止位会被接收器忽略。接收到的每一位数据被送入接收移位寄存器，收到第一个停止位以后，接收器置位位于 UCSRA 寄存器的接收数据完成标志 RXC 位，并把移位寄存器中完整的数据帧转移

到接收缓冲器中，CPU 通过读取 UDR 寄存器就可以获得接收到的数据。

接收 9 位数据的帧

如果设定了 9 位数据的数据帧，在从 UDR 读取低 8 位数据之前必须首先读取寄存器 UCSRB 的 RXB8 位来获得第 9 位数据。这个规则同样适用于状态标志位 FE、DOR 以及 PE。读取 UDR 存储单元会改变接收缓冲器的状态，进而改变同样存储于缓冲器中的 TXB8、FE、DOR 及 PE 位。

接收结束标志及中断处理

USART 接收器有一个标志位：接收结束标志 RXC，用来表明接收缓冲器中是否有未被读出的数据。当接收缓冲器中有未被读出的数据时，此位为“1”，反之为“0”。如果接收器被禁止，接收缓冲器会被刷新，RXC 也会被清零。

置位 UCSRB 的接收结束中断使能位 RXCIE 后，只要 RXC 标志被置位（且全局中断被使能），就会产生 USART 接收结束中断。使用中断方式进行数据接收时，数据接收结束中断服务程序必须从 UDR 读取数据来清零 RXC 标志，否则只要中断处理程序一结束，一个新的中断就会产生。

接收错误标志

USART 接收器有三个错误标志：帧错误 FE、数据溢出 DOR 及奇偶校验错误 PE。它们都位于 UCSRA 寄存器。错误标志与数据帧一起保存在接收缓冲器当中。所有的错误标志都不能产生中断。

帧错误标志 FE 表明存储在接收缓冲器中的下一个可读帧的第一个停止位的状态。停止位正确（值为“1”）则 FE 标志为“0”，否则 FE 标志为“1”。这个标志可用来检测同步丢失、传输中断，也可用于协议处理。

数据溢出标志 DOR 表明由于接收缓冲器满造成了数据丢失。当接收缓冲器为满，接收移位寄存器中已有数据，若此时检测到一个新的起始位，数据溢出就产生了。DOR 标志被置位即表明在最近一次读取 UDR 和下一次读取 UDR 之间丢失了一个或多个数据帧。当数据帧成功地从移位寄存器转入接收缓冲器后，DOR 标志被清零。

奇偶校验错标志 PE 表明接收缓冲器中的下一帧数据在接收时有奇偶错误。如果不使能奇偶校验，PE 被清零。

奇偶校验器

置位奇偶校验模式位 UPM1 将启动奇偶校验器。校验的模式（偶校验或奇校验）由 UPM0 决定。奇偶校验使能后，校验器将计算输入数据的奇偶并把结果与数据帧的奇偶位进行比较。校验结果将与数据和停止位一起存储在接收缓冲器中。CPU 通过读取 PE 位来检查接收的帧当中是否有奇偶错误。如果下一个从接收缓冲器中读出的数据有奇偶错误，并且奇偶校验使能，则 UPE 被置位，一直有效到接收缓冲器 UDR 被读取。

禁止接收器

与发送器相比，禁止接收器即刻起作用。正在接收的数据将丢失。禁止接收器（RXEN 清零）

后，接收器将不再占用 RxD 引脚，接收缓冲器也会被刷新。

异步数据接收

USART 有一个时钟恢复单元和数据恢复单元来处理异步数据接收。时钟恢复逻辑用于同步从 RxD 引脚输入的异步串行数据和内部的波特率时钟。数据恢复逻辑用于采集数据，并通过低通滤波器过滤所输入的每一位数据，从而提高接收器的抗干扰性能。异步接收的工作范围依赖于内部波特率时钟的精度、帧输入的速率及一帧所包含的数据位数。

异步工作范围

接收器的工作范围依赖于接收到的数据速率与内部波特率之间的不匹配程度。如果发送器以过快或过慢的比特率传输数据，或者接收器内部产生的波特率没有相同的频率，那么接收器就无法与起始位同步。为了确保接收器不会错过下一帧起始位的采样，数据输入速率和内部接收器波特率不能相差太大，用它们之间的比值来描述波特率的误差范围。下面两个表格分别给出了普通模式下和倍速模式下容许的最大波特率误差范围。

普通模式下最大接收器波特率误差范围

数据位+奇偶位长度和	最大误差范围（%）	推荐误差范围（%）
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

倍速模式下最大接收器波特率误差范围

数据位+奇偶位长度和	最大误差范围（%）	推荐误差范围（%）
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

从表中可以看出，普通模式下波特率允许有更大的变化范围。上述推荐的波特率误差范围是假定接收器和发送器对最大总误差具有同等贡献的前提下得出的。产生接收器波特率误差的可能原因有两个。首先，接收器系统时钟的稳定性与工作电压和温度有关。使用晶振来产生系统时钟时一般不会有此问题，但使用内部振荡器时，系统时钟可能会有偏差。第二个原因是波特率发生器不一定能通过对系统时钟的分频来得到恰好想要的波特率。此时可以调整 UBRR 的值，使得误差低至可以接受。

波特率设置及引入误差

对于标准晶振及谐振器频率来说，异步模式下的实际通信的波特率可通过波特率计算公式来获得，它与常用通信波特率之间的误差可用如下公式来计算：

$$Error[\%] = (Baud_{real}/Baud - 1)*100\%$$

其中，Baud 为常用的通信波特率，$Baud_{real}$ 为通过计算公式算出来的波特率，带入波特率计算公式即可得到波特率误差与系统时钟 f_{sys} 和波特率寄存器 UBRR 值之间的关系如下：

普通模式：

$$Error[\%] = (f_{sys}/(16*(UBRR+1))/Baud - 1)*100\%$$

倍速模式：

$$Error[\%] = (f_{sys}/(8*(UBRR+1))/Baud - 1)*100\%$$

当不考虑通信两边的时钟误差，即系统时钟 f_{sys} 为标准时钟时，即可得到波特率误差 UBRR 值之间的关系。下表即为 16MHz 系统时钟下不同 UBRR 值设置下的波特率误差。

16MHz 系统时钟下设置 UBRR 值所产生的误差

波特率 (bps)	f_{sys} = 16.000MHz			
	普通模式(U2X = 0)		倍速模式(U2X = 1)	
	UBRR	误差	UBRR	误差
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%

多处理器通信模式

置位 UCSRA 的多处理器通信模式（MPCM）位可以对 USART 接收器接收到的数据帧进行过滤。那些没有地址信息的帧将被忽略，也不会存入接收缓冲器。在一个多处理器系统中，各处理器通过相同的串行总线进行通信，这种过滤有效的减少了需要 CPU 处理的数据帧的数量。MPCM 位的设置不影响发送器的工作，但在多处理器通信的系统中，它的使用方法会有所不同。

如果接收器所接收的数据帧长度为 5 到 8 位，那么第一个停止位会用来表示当前帧包含的是数据还是地址信息。如果接收器所接收的数据帧长度是 9 位，那么由第 9 位来确定是数据还是地址信息。如果帧类型标志位为“1”，那么这是地址帧，否则为数据帧。

在多处理器通信模式下，允许多个从处理器从一个主处理器接收数据。首先要通过解码地址帧来确定所寻址的是哪一个从处理器。被寻址的从处理器将正常接收后续的数据，而其他的从处理器则会忽略这些数据帧直到接收到下一个地址帧。

对于一个作为主机的处理器来说，它可以使用 9 位数据帧格式，并用第 9 位数据来标识帧格式。在这种通信模式下，从处理器也必须工作于 9 位数据帧格式。
下面即为多处理器通信模式下进行数据交换的步骤：

1. 所有从处理器都工作在多处理器通信模式（置位 MPCM）；
2. 主处理器发送地址帧，所有从处理器都接收此帧。从处理器 UCSRA 寄存器的 RXC 位正常置位；
3. 每个从处理器都读取 UDR 寄存器的内容，解码地址帧来确定是否被选中。如果选中，就清零 UCSRA 寄存器的 MPCM 位，未被选中，则保持 MPCM 为“1”并等待下一个地址帧的到来；
4. 被寻址的从处理器接收所有的数据帧，直到收到一个新的地址帧。未被寻址的从处理器忽略这些数据帧；
5. 被寻址的从处理器收到最后一个数据帧后，置位 MPCM 位，并等待下一个地址帧的到来。然后从第二步骤重复进行。

使用 5 到 8 位数据的帧格式是可以的，但是不切实际，因为接收器必须在使用 n 和 n+1 帧格式之间进行切换。由于接收器和发送器使用相同的字符长度设置，这种设置使得全双工操作变得很困难。如果使用 5 到 8 位数据的帧格式，发送器应该设置两个停止位，其中第一个停止位被用于判断帧类型。

寄存器定义

UCSRA – USART 控制和状态寄存器 A

UCSRA – USART 控制和状态寄存器 A								
地址: 0xC0					默认值: 0x20			
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
Initial	0	0	1	0	0	0	0	0

Bit	Name	描述
7	RXC	接收结束标志位。 当 RXC 的值为“1”时，表明接收缓冲器中有未读出的数据。当 RXC 的值为“0”时，表明接收缓冲器中没有未读出的数据。接收器禁止时，接收缓冲器被刷新，导致 RXC 被清零。当接收结束中断使能位 RXCIE 为“1”时，RXC 可用来产生接收结束中断。
6	TXC	发送结束标志位。

		发送移位寄存器中的数据被送出，且发送缓冲器为空时 TXC 置位。执行发送结束中断时 TXC 自动清零，也可以通过对 TXC 写“1”来进行清零。当发送结束中断使能位 TXCIE 为“1”时，TXC 可用来产生发送结束中断。
5	UDRE	数据寄存器空标志位。 当 UDRE 为“1”时，表明 USART 发送数据缓冲器为空，可以写入数据。当 UDRE 为“0”时，表明 USART 发送数据缓冲器为满，不能写入数据。当数据寄存器空中断使能位 UDRIE 为“1”时，UDRE 可用来产生数据寄存器空中断。
4	FE	帧错误标志位。 当 FE 为“1”时，表明接收数据缓冲器接收到的数据有帧错误，即第一个停止位为“0”。当 FE 为“0”时，表明接收数据缓冲器接收到的数据没有帧错误，即第一个停止位为“1”。FE 被置位后会一直有效到 UDR 被读取。对 UCSRA 进行写入时，FE 这一位要写“0”。
3	DOR	数据溢出标志位。 当接收缓冲器为满（包含了两个数据），接收移位寄存器中有数据，若此时检测到一个新的起始位，数据溢出产生，DOR 被置位，一直有效到 UDR 被读取。对 UCSRA 进行写入时，DOR 这一位要写“0”。
2	PE	奇偶校验错误标志位。 当奇偶校验使能（UPM1 为“1”）时，且接收缓冲器中所接收到的数据帧有奇偶校验错误，PE 被置位，一直有效到 UDR 被读取。对 UCSRA 进行写入时，PE 这一位要写“0”。
1	U2X	倍速发送使能位。 当 U2X 为“1”时，异步通信模式的传输速率加倍。当 U2X 为“0”时，异步通信模式的传输速率为普通速率。 这一位仅在异步操作模式下有效，使用同步操作模式时将此位清零。
0	MPCM	多处理器通信模式使能位。 设置 MPCM 位将启动多处理器通信模式。MPCM 置位后，USART 接收器接收到的那些不包含地址信息的输入帧都将被忽略。发送器不受 MPCM 设置的影响。

UCSRB – USART 控制和状态寄存器 B

UCSRB – USART 控制和状态寄存器 B								
地址: 0xC1					默认值: 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	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	RXCIE	接收结束中断使能位。 置位后使能 RXC 中断，清零后禁止 RXC 中断。当 RXCIE 为“1”，全局中断使能，UCSRA 寄存器的 RXC 为“1”时可以产生 USART 接收结束中断。
6	TXCIE	发送结束中断使能位。 置位后使能 TXC 中断，清零后禁止 TXC 中断。当 TXCIE 为“1”，全局中断使能，UCSRA

		寄存器的 TXC 为“1”时可以产生 USART 发送结束中断。
5	UDRIE	数据寄存器空中断使能位。 置位后使能 UDRE 中断，清零后禁止 UDRE 中断。当 UDRIE 为“1”，全局中断使能，UCSRA 寄存器的 UDRE 为“1”时可以产生 USART 数据寄存器空中断。
4	RXEN	接收使能位。 置位后启动 USART 接收器。RxD 引脚的通用 IO 功能被 USART 接收所取代。禁止接收器将刷新接收缓冲器，并使 FE、DOR 及 PE 标志无效。
3	TXEN	发送使能位。 置位后启动 USART 发送器。TxD 引脚的通用 IO 功能被 USART 发送所取代。TXEN 清零后，只有等到所有的数据发送完成后才能够真正禁止 USART 发送。
2	UCSZ2	字符长度控制第 2 位。 UCSZ2 与 UCSRC 寄存器的 UCSZ1:0 结合在一起设置数据帧所包含的数据位数。
1	RXB8	接收数据第 8 位。 当数据帧长度为 9 位时，RXB8 是接收数据的最高位。读取 UDR 所包含的低 8 位数据之前要先读取 RXB8。
0	TXB8	发送数据第 8 位。 当数据帧长度为 9 位时，TXB8 是发送数据的最高位。写入 UDR 所包含的低 8 位数据之前要先写入 TXB8。

UCSRC– USART 控制和状态寄存器 C

UCSRC– USART 控制和状态寄存器 C								
地址: 0xC2					默认值: 0x86			
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
Initial	0	0	0	0	0	1	1	0

Bit	Name	描述	
7:6	UMSEL1:0	USART 模式选择位。 UMSEL 选择同步或异步操作模式。	
		UMSEL	模式
		0	USART 异步操作模式
		1	USART 同步操作模式
		2	SPI 从机操作模式
		3	SPI 主机操作模式
5:4	UPM1:0	奇偶校验模式选择位。 高位 UPM1 选择使能或禁止奇偶校验，低位 UPM0 选择奇校验或偶校验。	
		UPM1:0	模式
		0	禁止奇偶校验
		1	保留
		2	使能偶校验
		3	使能奇校验

3	USBS	停止位选择位。选择停止位的位数。		
		USBS	停止位位数	
		0	1	
		1	2	
2:1	UCSZ1:0	数据帧字符长度选择位。 UCSZ1:0 与 UCSRB 寄存器的 UCSZ2 结合起来设置数据帧包含的数据位数。		
		UCSZ2:0	数据帧长度	
		0	5 位	
		1	6 位	
		2	7 位	
		3	8 位	
		4	保留	
		5	保留	
		6	保留	
		7	9 位	
0	UCPOL	时钟极性选择位。 在 USART 同步工作模式下，UCPOL 设置了输出数据的改变和输入数据的采样与同步时钟 XCK 之间的关系。使用异步工作模式下与 UCPOL 无关，将这一位清零		
		UCPOL	发送数据改变	接收数据采样
		0	XCK 的上升沿	XCK 的下降沿
		1	XCK 的下降沿	XCK 的上升沿

UBRRL – USART 波特率寄存器低字节

UBRRL – USART 波特率寄存器低字节								
地址: 0xC4					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	UBRR[7:0]	USART 波特率寄存器的低字节部分。 USART 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。

UBRRH – USART 波特率寄存器高字节

UBRRH – USART 波特率寄存器高字节	
地址: 0xC5	默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述	
7:4	-	保留。	
3:0	UBRR[11:8]	USART 波特率寄存器的高字节部分。 USART 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。 UBRR = {UBRR[11:8], UBRRL}	
		工作模式	波特率计算公式
		异步正常模式	BAUD = f_{sys}/(16*(UBRR+1))
		异步倍速模式	BAUD = f_{sys}/(8*(UBRR+1))
		同步主机模式	BAUD = f_{sys}/(2*(UBRR+1))

UDR – USART 数据寄存器

UDR – USART 数据寄存器								
地址: 0xC6					默认值: 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
Initial	0	0	0	0	0	0	0	0

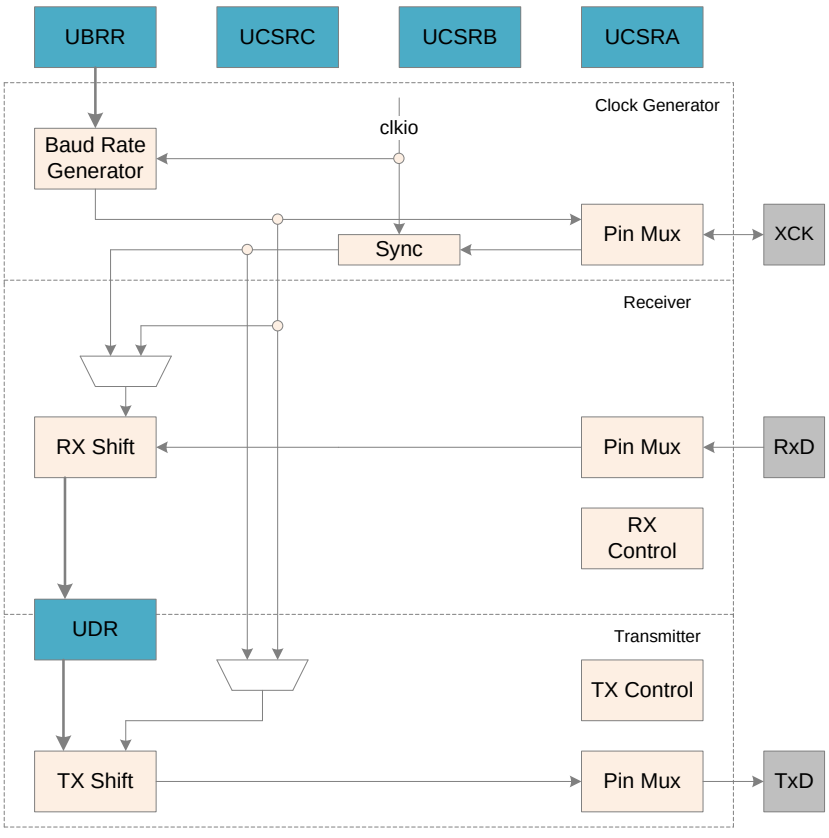
Bit	Name	描述
7:0	UDR	USART 发送和接收的数据。 USART 发送数据缓冲器和接收数据缓冲器共享 USART 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 在 5 到 8 位数据帧模式下，未使用的第 9 位被发送器忽略，而接收器则将它们设置为 0。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 TxD 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

USART0 - SPI 工作模式

- 全双工操作，三线同步数据传输
- 主机或从机操作
- 支持全部四种工作模式（模式 0、1、2 和 3）
- 低位或高位首先传输（可配置的数据传输顺序）
- 队列操作（双缓冲器）
- 高分辨率波特率产生器

综述

当设置 USCRC 的 UMSEL1 位为“1”时，使能 SPI 工作模式，用 USPI 来表示。此 SPI 模块为三线 SPI 工作模式，与四线 SPI 模式相比，缺少从机选择线，其它三根线均一致。USPI 占用 USART 的资源，包括发送和接收移位寄存器和缓冲器，以及波特率发生器。奇偶校验产生和检查逻辑，数据和时钟恢复逻辑均无效。控制和状态寄存器的地址是一样的，不过寄存器位的功能会随着 SPI 工作模式的需要而发生改变。



USART in SPI 结构图

时钟产生

当 SPI 工作在主机模式时，需要提供通信用的时钟，复用 USART 的波特率发生器来产生这个时钟。该时钟从 XCK 引脚输出，因此 XCK 引脚的数据方向寄存器（DDR_XCK）必须设置为“1”。

时钟频率有以下计算公式决定：

BAUD = f_{sys}/(2*(UBRR+1))

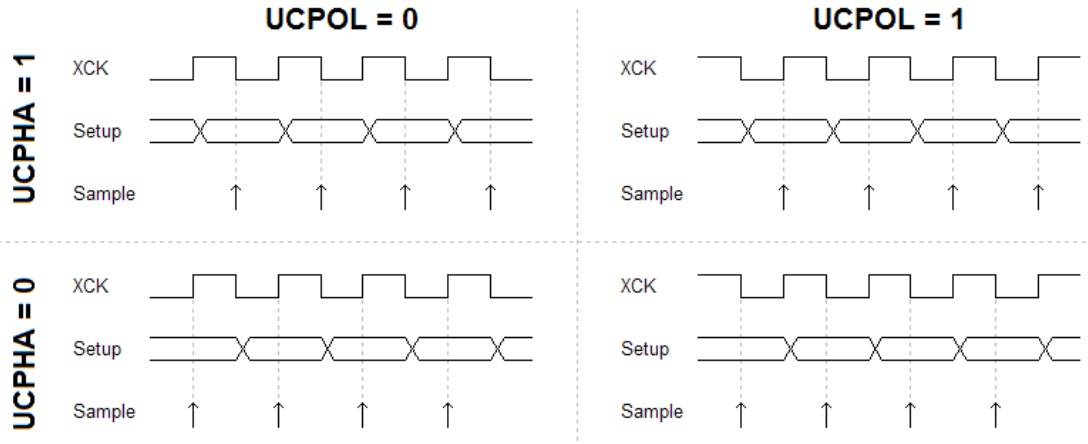
当 SPI 工作在从机模式时，通信时钟由外部主机提供，从 XCK 引脚输入，因此 XCK 引脚的数据方向寄存器（DDR_XCK）必须设置为“0”。

SPI 数据模式和时序

SPI 有四种时钟相位和极性的组合方式，有控制位 UCPHA 和 UCPOL 来决定，具体的控制如下表和下图所示：

SPI 工作模式

SPI 模式	UCPOL	UCPHA	起始沿	结束沿
0	0	0	上升沿采样	下降沿设置
1	0	1	上升沿设置	下降沿采样
2	1	0	下降沿采样	上升沿设置
3	1	1	下降沿设置	上升沿采样



SPI 工作模式图示

帧格式

SPI 的一个串行帧可以由最低位或最高位开始，到最高位或最低位结束，总共 8 位数据。一帧结束以后，可以紧接着传输新的一帧，传输结束即可拉高数据线为空闲状态。

数据传输

SPI 置 UCSRB 寄存器的 TXEN 位为“1”来使能发送器，TxD 引脚被发送器占用来发送串行输出数据。此时接收器可以不使能。

SPI 置 UCSRB 寄存器的 RXEN 位为“1”来使能接收器，RxD 引脚被接收器占用来接收串行输入数据。此时发送器须使能。

SPI 发送和接收都使用 XCK 来当作传输时钟。

进行通信之前首先要对 SPI 进行初始化。初始化过程通常包括波特率的设定，帧数据位传输顺序的设定，以及根据需要使能接收器或发送器。对于中断驱动的 SPI 操作，在初始化时要清零全局中断标志并禁止 SPI 的所有中断。
在进行重新初始化比如改变波特率或帧结构时，必须确保没有数据传输。TXC 标志位可以用来检测发送器是否完成了所有传输，RXC 标志位可以用来检测接收缓冲器中是否还有数据未被读出。如果 TXC 标志位用作此用途，在每次发送数据之前（写 UDR 寄存器之前）必须清零 TXC 标志位。

初始化 SPI 以后，往 UDR 寄存器写入数据即可开始数据传输。由于发送器控制着传输时钟，发送和接收数据均是如此操作。当发送移位寄存器准备好发送新一帧数据的时候，发送器就会把写入到 UDR 寄存器的数据从发送缓冲器移到发送移位寄存器里并发送出去。为了保证输入缓冲器和发送数据同步，每发送一个字节的数据后都必须读取一次 UDR 寄存器。当发生数据溢出时，最近收到的数据将会丢失，而不是最早收到的数据。

发送标志位与中断

SPI 发送器有两个标志位：SPI 数据寄存器空标志 UDRE 和传输结束标志 TXC，两个标志位都可以产生中断。

数据寄存器空标志 UDRE 用来表示发送缓冲器是否可以写入一个新的数据。该位在发送缓冲器空时被置“1”，满时被置“0”。当 UDRE 位为“1”时，CPU 可以往数据寄存器 UDR 写入新的数据，反之则不能。

当 UCSRB 寄存器中的数据寄存器空中断使能位 UDRIE 为“1”时，只要 UDRE 被置位（且全局中断使能），就将产生 SPI 数据寄存器空中断请求。对寄存器 UDR 执行写操作将清零 UDRE。当采用中断方式传输数据时，在数据寄存器空中断服务程序中必须写入一个新的数据到 UDR 以清零 UDRE，或者是禁止数据寄存器空中断。否则一旦该中断服务程序结束，一个新的中断将再次产生。

当整个数据帧被移出发送移位寄存器，同时发送寄存器中又没有新的数据时，发送结束标志 TXC 将被置位。当 UCSRB 上的发送结束中断使能位 TXCIE（且全局中断使能）置“1”时，随着 TXC 标志位被置位，SPI 发送结束中断将被执行。一旦进入中断服务程序，TXC 标志位即被自动清零，CPU 也可以对该位写“1”来清零。

禁止发送器

当 TXEN 清零后，只有等所有的数据都发送完成以后发送器才能够真正禁止，即发送移位寄存器与发送缓冲寄存器中都没有要传送的数据。发送器禁止以后，TxD 引脚恢复其通用 IO 功能。

接收结束标志及中断

SPI 接收器有一个标志位：接收结束标志 RXC，用来表明接收缓冲器中是否有未被读出的数

据。当接收缓冲器中有未被读出的数据时，此位为“1”，反之为“0”。如果接收器被禁止，接收缓冲器会被刷新，RXC 也会被清零。

置位 UCSRB 的接收结束中断使能位 RXCIE 后，只要 RXC 标志被置位（且全局中断被使能），就会产生 SPI 接收结束中断。使用中断方式进行数据接收时，数据接收结束中断服务程序必须从 UDR 读取数据来清零 RXC 标志，否则只要中断处理程序一结束，一个新的中断就会产生。

禁止接收器

与发送器相比，禁止接收器即刻起作用。正在接收的数据将丢失。禁止接收器（RXEN 清零）后，接收器将不再占用 RxD 引脚，接收缓冲器也会被刷新。

寄存器定义

USART 寄存器列表

寄存器	地址	默认值	描述
UCSRA	0xC0	0x20	USPI 控制和状态寄存器 A
UCSRB	0xC1	0x00	USPI 控制和状态寄存器 B
UCSRC	0xC2	0x06	USPI 控制和状态寄存器 C
UBRRL	0xC4	0x0	USPI 波特率寄存器低字节
UBRRH	0xC5	0x0	USPI 波特率寄存器高字节
UDR	0xC6	0x0	USPI 数据寄存器

UCSRA – USPI 控制和状态寄存器 A

UCSRA – USPI 控制和状态寄存器 A								
地址: 0xC0					默认值: 0x20			
Bit	7	6	5	4	3	2	1	0
Name	RXC	TXC	UDRE	-	-	-	-	-
R/W	R	R/W	R	-	-	-	-	-
Initial	0	0	1	0	0	0	0	0

Bit	Name	描述
7	RXC	接收结束标志位。 当 RXC 的值为“1”时，表明接收缓冲器中有未读出的数据。当 RXC 的值为“0”时，表明接收缓冲器中没有未读出的数据。接收器禁止时，接收缓冲器被刷新，导致 RXC 被清零。当接收结束中断使能位 RXCIE 为“1”时，RXC 可用来产生接收结束中断。
6	TXC	发送结束标志位。 发送移位寄存器中的数据被送出，且发送缓冲器为空时 TXC 置位。执行发送结束中断时 TXC 自动清零，也可以通过对 TXC 写“1”来进行清零。当发送结束中断使能位 TXCIE 为“1”时，TXC 可用来产生发送结束中断。
5	UDRE	数据寄存器空标志位。 当 UDRE 为“1”时，表明 USPI 发送数据缓冲器为空，可以写入数据。当 UDRE 为

		“0”时，表明 USPI 发送数据缓冲器为满，不能写入数据。当数据寄存器空中断使能位 UDRIE 为“1”时，UDRE 可用来产生数据寄存器空中断。
4:0	-	USPI 下保留。

UCSRB – USPI 控制和状态寄存器 B

UCSRB – USPI 控制和状态寄存器 B								
地址: 0xC1					默认值: 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	-	-	-
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	RXCIE	接收结束中断使能位。 置位后使能 RXC 中断，清零后禁止 RXC 中断。当 RXCIE 为“1”，全局中断使能，UCSRA 寄存器的 RXC 为“1”时可以产生 USPI 接收结束中断。
6	TXCIE	发送结束中断使能位。 置位后使能 TXC 中断，清零后禁止 TXC 中断。当 TXCIE 为“1”，全局中断使能，UCSRA 寄存器的 TXC 为“1”时可以产生 USPI 发送结束中断。
5	UDRIE	数据寄存器空中断使能位。 置位后使能 UDRE 中断，清零后禁止 UDRE 中断。当 UDRIE 为“1”，全局中断使能，UCSRA 寄存器的 UDRE 为“1”时可以产生 USPI 数据寄存器空中断。
4	RXEN	接收使能位。 置位后启动 USPI 接收器。RxD 引脚的通用 IO 功能被 USPI 接收所取代。禁止接收器将刷新接收缓冲器。
3	TXEN	发送使能位。 置位后启动 USPI 发送器。TxD 引脚的通用 IO 功能被 USPI 发送所取代。TXEN 清零后，只有等到所有的数据发送完成后才能够真正禁止 USART 发送。
2:0	-	USPI 下保留。

UCSRC– USART 控制和状态寄存器 C

UCSRC– USART 控制和状态寄存器 C								
地址: 0xC2					默认值: 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
Initial	0	0	0	0	0	1	1	0

Bit	Name	描述
7:6	UMSEL1:0	USART 模式选择位。

		UMSEL 选择同步或异步操作模式。
		UMSEL: 0 → USART 异步操作模式 1 → USART 同步操作模式 2 → SPI 从机操作模式 3 → SPI 主机操作模式
5:3	-	USPI 下保留。
2	DORD	数据传输顺序选择位。 DORD: 0 → 高位先传输 1 → 低位先传输
1	UCPHA	时钟相位选择。 UCPHA 选择数据采样发生在起始沿或结束沿。 UCPHA: 0 → 起始沿 1 → 结束沿
0	UCPOL	时钟极性选择。 UCPOL 选择数据改变和采样发生在上升沿或下降沿。 UCPOL: 0 → 发送数据的改变: XCK 的上升沿, 接收数据的采样: XCK 的下降沿 1 → 发送数据的改变: XCK 的下降沿, 接收数据的采样: XCK 的上升沿

UBRRL – USPI 波特率寄存器低字节

UBRRL – USPI 波特率寄存器低字节								
地址: 0xC4					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	UBRR[7:0]	USPI 波特率寄存器的低字节部分。 USPI 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。

UBRRH – USPI 波特率寄存器高字节

UBRRH – USPI 波特率寄存器高字节								
地址: 0xC5					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述	
7:4	-	USPI 下保留。	
3:0	UBRR[11:8]	USPI 波特率寄存器的高字节部分。 USPI 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。 UBRR = {UBRR[11:8], UBRRL}	
		工作模式	波特率计算公式
		从机模式	波特率由外部主机决定
		主机模式	BAUD = f_{sys}/(2*(UBRR+1))

UDR – USPI 数据寄存器

UDR – USPI 数据寄存器								
地址: 0xC6					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	UDR	USPI 发送和接收的数据。 USPI 发送数据缓冲器和接收数据缓冲器共享 USPI 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 在 5 到 8 位数据帧模式下，未使用的第 9 位被发送器忽略，而接收器则将它们设置为 0。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 TxD 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

TWI – 双线串行总线(I2C)

- 简单且强大而灵活的通讯接口，只需要 2 线
- 支持主机和从机操作
- 器件可以工作于发送器模式或接收器模式
- 7 位地址空间允许有 128 个从机
- 支持多主机仲裁
- 高达 400Kbps 的数据传输率
- 完全可编程的从机地址以及公共地址
- 睡眠模式下地址匹配时可以唤醒

TWI 总线介绍

两线串行接口 TWI 很适合于典型的处理器应用。TWI 协议允许系统设计者只用两根双向的传输线就可以将 128 个不同的设备互连到一起。这两根线是时钟 SCL 和数据 SDA。外部硬件只需要在每根线上接两个上拉电阻。所有连接到总线上的设备都有自己的地址。TWI 协议解决了总线仲裁的问题。

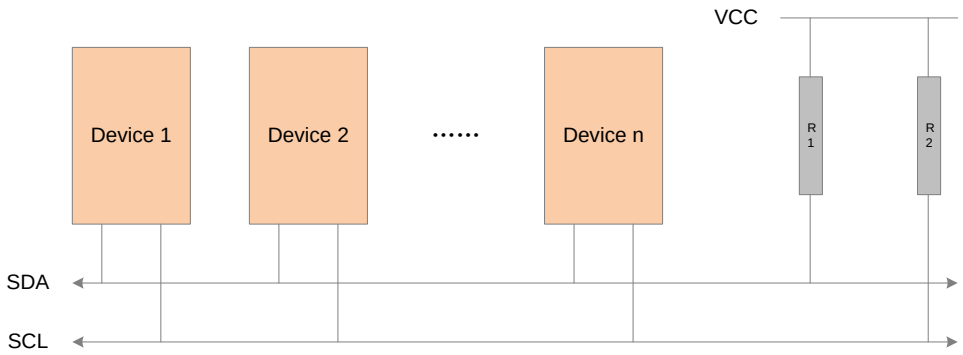
TWI 术语

以下定义的术语将在本节频繁出现。

术语	描述
主机	启动和停止传输的设备。主机还要负责产生 SCL 时钟。
从机	被主机寻址的设备
发送器	将数据放到总线上的设备
接收器	从总线上接收数据的设备

电气连接

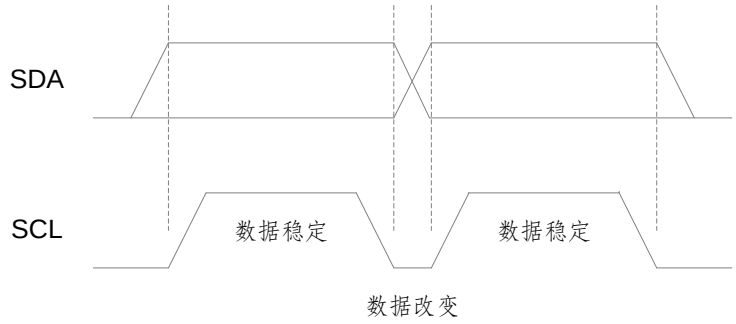
如下图所示，TWI 接口的两根线都通过上拉电阻与正电源连接。所有 TWI 兼容器件的总线驱动都是漏极开路或集电极开路的，这样就实现了对接口操作的线与功能。当 TWI 器件输出为“0”时，TWI 总线会产生低电平。当所有的 TWI 器件输出为三态时，总线允许上拉电阻将电压拉高。为保证所有的总线操作，凡是与 TWI 总线连接的器件都必须上电。



TWI 总线互连图

数据传输和帧结构

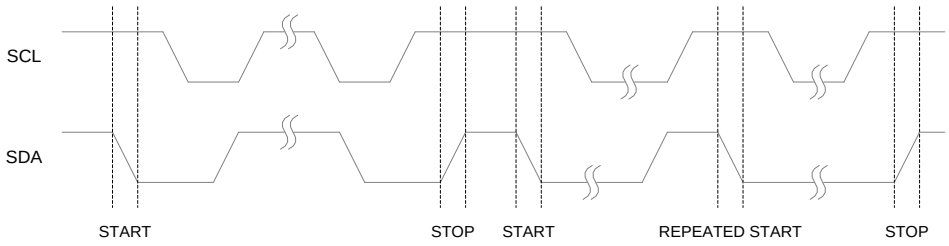
TWI 总线上的每一位数据传输都是和时钟同步的。当时钟线为高时，数据线上的电平必须保持稳定，除非是为了产生开始或停止状态。



TWI **数据有效性图**

开始和停止状态

TWI 的传输由主机来启动和停止。主机在总线上发出 START 状态以其送数据传输，发出 STOP 状态以停止数据传输。在 START 和 STOP 状态之间，总线被认为是忙碌的，不允许其它主机试图占用总线的控制权。有一种特殊情况只允许发生在 START 和 STOP 状态之间产生一个新的 START 状态，这被称为 REPEATED START 状态，适用于当前主机在不放弃总线控制的情况下启动新的传输。REPEATED START 之后直到下一个 STOP 之前，总线仍然被认为是忙碌的。这与 START 是一致的，因此在本文档中，如果没有特殊说明，均采用 START 来表述 START 和 REPEATED START。如下图所示，START 和 STOP 条件是在 SCL 线为高时，改变 SDA 线的电平状态。



START、REPEATED START **和** STOP **状态图**

地址包格式

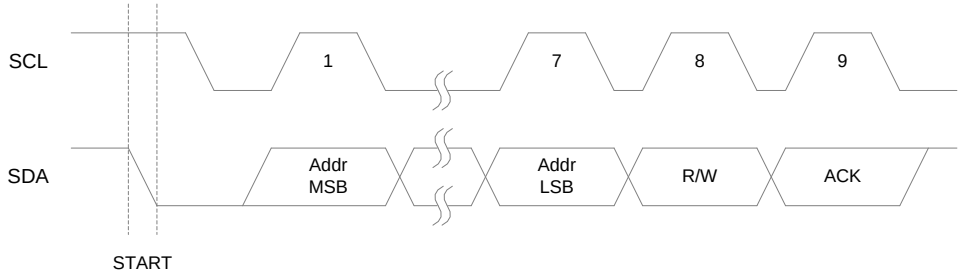
所有 TWI 总线上传输的地址包都是 9 位数据长度，由 7 位地址，1 位 READ/WRITE 控制位和 1 位应答位组成。当 READ/WRITE 位为“1”，则执行读操作；当 READ/WRITE 位为“0”时，执行写操作。从机被寻址后，必须在第 9 个 SCL（ACK）周期通过拉低 SDA 线做出应答。若该从机忙或有其它原因无法响应主机，则应在 ACK 周期保持 SDA 线为高。然后主机可以发出 STOP 状态或 REPEATED START 状态重新开始发送。

地址包包括一个从机地址和一个读或写控制位，分别用 SLA+R 或 SLA+W 来表示。

地址字节的 MSB 位首先发生。除了保留地址“00000000”被留用作广播呼叫以及所有形如“1111xxxx”格式的地址需要保留作将来使用外，其它从机地址可由设计者自由分配。

当发生广播呼叫时，所有的从机应在 ACK 周期通过拉低 SDA 线来做出应答。当主机需要发送相同的信息给多个从机时可以使用广播功能。当广播呼叫地址加上 WRITE 位被发送到总线上以后，所有需要响应该广播呼叫的从机将在 ACK 周期拉低 SDA 线。所有这些响应了广播呼叫的从机将会接收紧跟的数据包。需要注意的是，发送广播呼叫地址加上 READ 位是没有意义的，因为如果几个从机同时发送不同的数据会带来总线冲突。

地址包格式如下图所示：

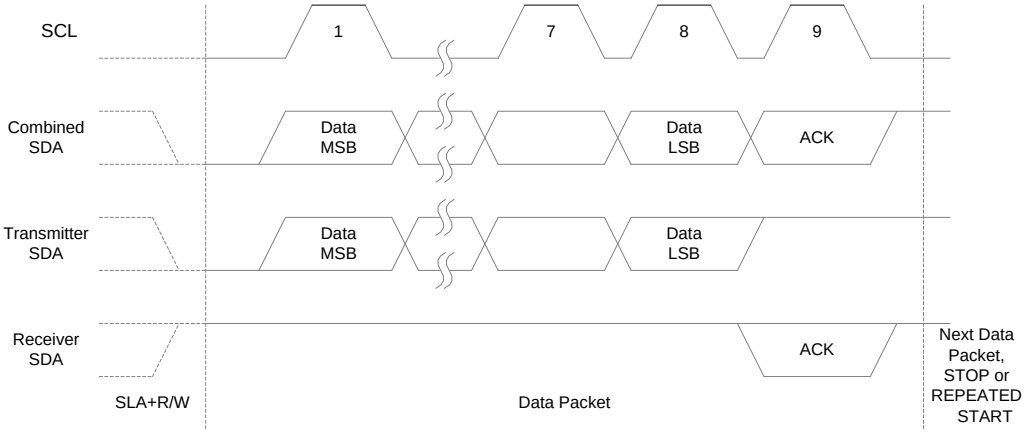


TWI 地址包格式图

数据包格式

所有 TWI 总线上传输的数据包都是 9 位数据长度，由 1 个数据字节和 1 位应答位组成。在数据传输期间，主机负责产生传输时钟 SCL 和 START 及 STOP 状态，发送器发送要传输的字节数据，接收器产生接收响应。确认信号 ACK 是接收器在第 9 个 SCL（ACK）周期通过拉低 SDA 线来产生的。如果接收器在 ACK 周期保持 SDA 线为高，则发出的是未确认信号 NACK。当接收器已经接收到了最后一个字节，或者由于某些原因不能再接收任何数据，则应该在收到最后字节后通过发送 NACK 来告知发送器。数据字节的 MSB 位先传输。

数据包格式如下图所示：

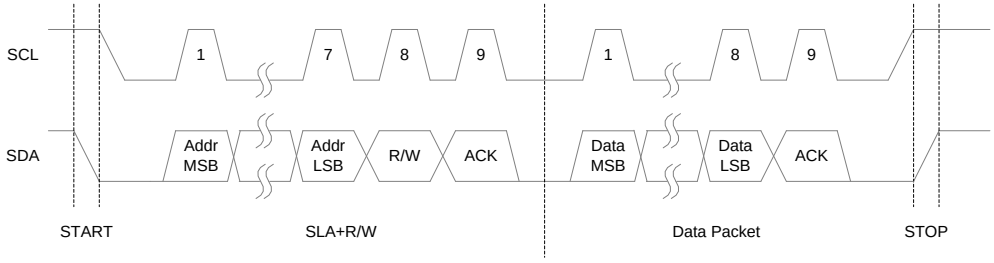


TWI 数据包格式图

组合地址和数据包的传输

一次传输基本上由 1 个 START，1 个 SLA+R/W，1 个或多个数据包以及 1 个 STOP 组成。只有 START 和 STOP 的空信息是非法的。可以使用 SCL 线的线与功能来实现主机与从机的握手。从机可以通过拉低 SCL 线来延长 SCL 的地电平周期。当主机设定的时钟速度远远快于从机，或从机需要额外的时间来处理数据时，这个特性就非常有用。从机延长 SCL 的低电平周期并不会影响 SCL 的高电平周期，它仍然是由主机决定的。由此可知，从机可以通过改变 SCL 的占空比来降低 TWI 的数据传输速度。

下图所示的是一个典型的数据传输。注意 SLA+R/W 与 STOP 之间可以传送多个字节，取决于应用软件的实现协议。



典型的 TWI 传输

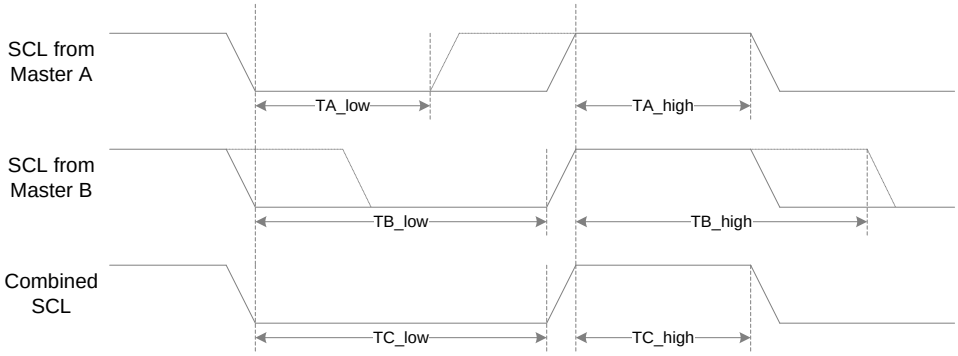
多主机系统及其仲裁和同步

TWI 协议允许总线上有多个主机，并采用了特殊的措施来保证即使两个或多个主机同时启动传输也能够像普通传输一样处理。多主机系统会出现两个问题：

1. 实现的算法只允许多主机中的一个主机完成传输。当其它主机发现它们失去选择权后必须停止它们的传输。这个选择的过程就叫做仲裁。当竞争中的主机发现其仲裁失败后，应立即切换到从机模式来检测自己是否被获得总线控制权的主机寻址。事实上多主机同时开始传输时不应该被从机检测到，即不允许破坏正在总线上传送的数据。
2. 不同的主机可能使用不同的 SCL 频率。为保证传送的一致性，必须设计一种同步主机串行时钟的方案。这会简化仲裁过程。

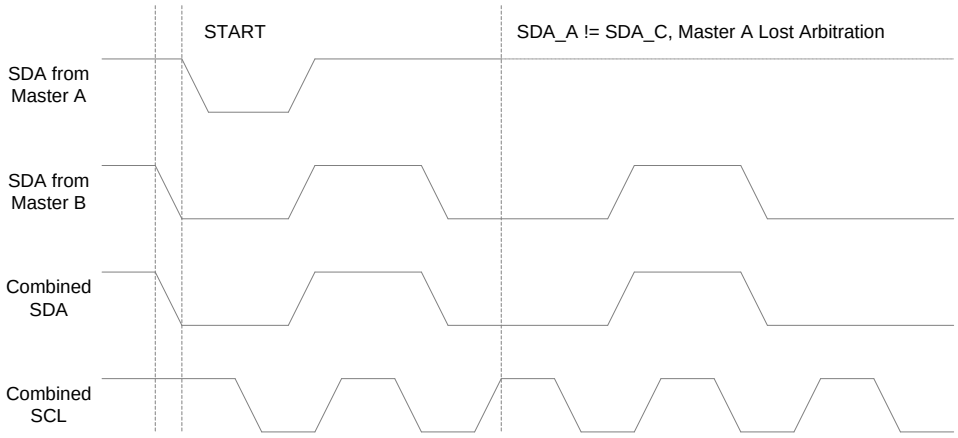
总线的线与功能就是用来解决上述问题的。所有主机的串行时钟都会线与到一起产生一个组合时钟，其高电平时间等于所有主机时钟中最短的一个，其低电平则等于所有主机时钟中最长的一个。所有主机都监听 SCL，当组合 SCL 时钟变高或变低时，它们可以有效地分别开始计算各自 SCL 高电平和低电平溢出周期。

多主机的 SCL 时钟同步机制如下图所示：



多主机 SCL 时钟同步时序图

输出数据之后所有的主机都持续监听 SDA 线来实现仲裁。如果从 SDA 读回的数值与主机输出的数值不匹配，该主机即失去仲裁。要注意的是，主机输出高电平的 SDA，而另一个主机输出低电平的 SDA 时才会失去仲裁。失去仲裁的主机应立即转换为从机模式，并检测是否被寻址。失去仲裁的主机必须将 SDA 线置高，但在当前的数据或地址包结束之前还可以产生时钟信号。仲裁将会持续到系统只剩下一个主机，这可能会占用多个比特。如果多个主机对相同的从机寻址，仲裁将会持续到数据包。



两个主机之间的仲裁

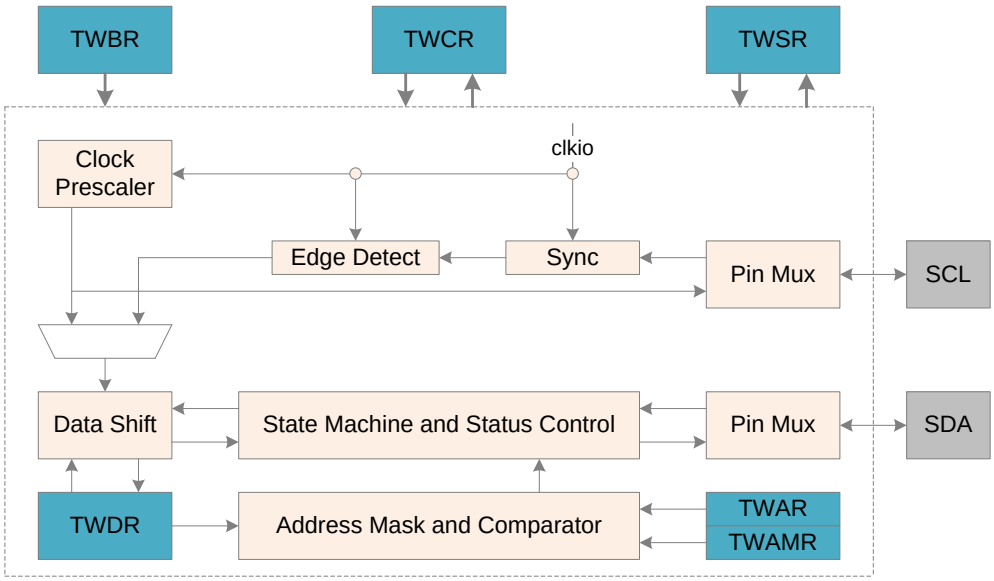
注意不允许在以下情形进行仲裁：

- 一个 REPEATED START 状态与一个数据位之间；
- 一个 STOP 状态与一个数据位之间；
- 一个 REPEATED START 状态与 STOP 状态之间；

应用软件必须考虑上述情况，保证不会出现这些非法仲裁情形。这意味着在多主机系统中，所有的数据传输必须由相同的 SLA+R/W 与数据包组成。换句话说，所有的传送必须包含相同数目的数据包，否则仲裁结果无法定义。

TWI 模块综述

TWI 模块的结构图如下图所示。



TWI Block **结构图**

TWI 模块主要包括比特率发生器，总线接口单元，地址比较器和控制单元等。具体见下列详细描述。

比特率发生器单元

比特率发生器单元主要控制主机模式下的 SCL 时钟周期。SCL 时钟周期由 TWI 比特率寄存器 TWBR 和 TWI 状态寄存器 TWSR 中的预分频控制位共同决定。从机操作不受比特率或预分频设置的影响，但要保证从机的工作时钟至少是 SCL 频率的 16 倍。注意，从机可能会延长 SCL 的低电平周期，从而降低 TWI 总线的平均时钟频率。SCL 时钟频率有以下的计算公式产生：

$$f_{scl} = f_{sys}/(16 + 2*TWBR*4^{TWPS})$$

其中，TWBR 为 TWI 比特率寄存器的数值，TWPS 为 TWI 状态寄存器中的预分频控制位。

总线接口单元

总线接口单元包括数据和地址移位寄存器 TWDR，START/STOP 控制器和仲裁判定硬件电路。

TWDR 包含要发送的地址或数据字节，或者已接收的地址或数据字节。除了包含 8 位的 TWDR，总线接口单元还包括发送或接收的 ACK/NACK 寄存器。这个 ACK/NACK 寄存器不能直接被应用软件访问。当接收数据时，它可以通过 TWI 控制寄存器 TWCR 来置位或清零。当发送数据时，接收到的 ACK/NACK 值由 TWI 状态寄存器 TWSR 中的 TWS 值来反映。

START/STOP 控制器负责产生和检测 START，REPEATED START 和 STOP 状态。当 MCU 处于某些休眠模式时，START/STOP 控制器仍可以检测 START 和 STOP 状态，当被 TWI 总线上的主机寻址时将 MCU 从休眠模式唤醒。

如果 TWI 以主机模式启动了数据传输，仲裁检测电路将持续监听总线，以确定是否仍拥有总线控制权。当 TWI 模块丢失总线控制权后，控制单元将会执行正确的动作并产生合适的

状态码来通知 MCU。

地址匹配单元

地址匹配单元用来检查接收到的地址字节是否与 TWI 地址寄存器中的 7 位地址相匹配。当 TWAR 寄存器中的 TWI 广播呼叫识别使能位（TWGCE）置位，从总线接收到的地址也会与广播地址比较。一旦地址匹配成功，控制单元将执行正确的动作。TWI 模块可以响应或不响应主机的寻址，这取决于 TWCR 寄存器的设置。即使在休眠模式下，地址匹配单元也可以比较地址，若被总线上的主机寻址，则将 MCU 从休眠模式唤醒。

控制单元

控制单元负责监听总线并根据 TWCR 的设置产生相应的响应。当 TWI 总线上发生需要应用软件参与的事件时，TWI 中断标志位 TWINT 将会被置位。在接下来的一个时钟周期，TWI 状态寄存器 TWSR 将会被更新为表明该事件的状态码。在 TWINT 被置位时，TWSR 包含确切的状态信息。在其它时间里，TWSR 为一个特殊的状态码，表示没有确切的状态信息。一旦 TWINT 标志位被置位，SCL 线就一直保持低电平，暂停总线上的 TWI 传输，让应用软件处理事件。

下列情形下，TWINT 标志位将置位：

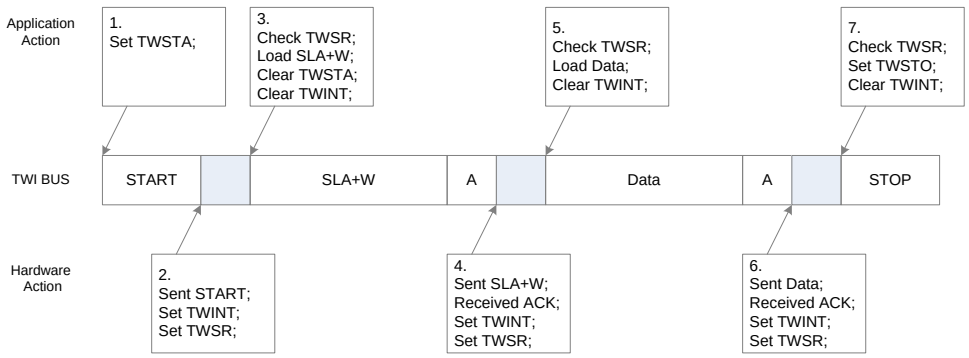
- TWI 传送完 START/REPEATED START 状态后
- TWI 传送完 SLA+R/W 后
- TWI 传送完一个地址字节后
- TWI 总线仲裁失败后
- TWI 被主机寻址后（从机地址匹配或广播方式）
- 被寻址作为从机工作时，收到 STOP 或 REPEATED START 后
- 由非法的 START 或 STOP 状态所引起的总线错误时

TWI 的使用

TWI 接口是面向字节和基于中断的。所有的总线事件，如接收到一个字节或发送了一个 START 信号等，都会产生一个 TWI 中断。由于 TWI 是基于中断的，因此在 TWI 字节传送的过程中，应用软件可以自如的进行其它操作。TWCR 寄存器中的 TWI 中断使能位 TWIE 和全局中断使能位一起来控制在 TWINT 标志位置位时是否产生 TWI 中断。如果 TWIE 位被清零，应用软件必须采用查询 TWINT 标志位的方式来检测 TWI 总线上的动作。

当 TWINT 标志位被置位时，表示 TWI 接口完成了当前的操作，等待应用软件的响应。在这种情况下，TWI 状态寄存器 TWSR 中包含了反映当前总线状态的状态码。应用软件可以通过设置 TWCR 和 TWDR 寄存器，来决定在接下来的 TWI 总线周期 TWI 接口该如何工作。

下图给出的是应用程序与 TWI 接口连接的例子。该例中，主机期望发送一个字节的数据给从机。这里的描述很简单，接下来的章节会有更详细的展示。



TWI 典型的传输过程图

图中所示的 TWI 传输过程为：

1. TWI 传输的第一步是发送 START。通过往 TWCR 寄存器写入特定值，指示 TWI 硬件发送 START 信号。写入的值将在随后详细说明。在写入的值中要置位 TWINT，这非常重要，往 TWINT 位写“1”会清零该位。TWCR 寄存器的 TWINT 置位期间 TWI 不会启动任何操作。一旦软件清零 TWINT 位，TWI 模块立即启动 START 信号的传送。
2. 当 START 状态发送完毕，TWCR 的 TWINT 标志位会被置位，TWSR 更新为的新的状态码，表示 START 信号成功发送。
3. 应用程序查看 TWSR 的值，确定 START 状态已经成功发送。如果 TWSR 显示为其它值，应用程序可以执行一些特殊操作，比如调用错误处理程序。当确定状态码与预期一致后，程序将 SLA+W 的值载入到 TWDR 寄存器中。TWDR 寄存器可同时在地址和数据中使用。随后软件往 TWCR 寄存器写入特定值，指示 TWI 硬件发送 TWDR 中的 SLA+W 的值。写入的值将在随后详细说明。在写入的值中要置位 TWINT，来清零 TWINT 标志位。TWCR 寄存器的 TWINT 置位期间 TWI 不会启动任何操作。一旦软件清零 TWINT 位，TWI 模块立即启动地址包的传送。
4. 当地址包发送完毕后，TWCR 的 TWINT 标志位会被置位，TWSR 更新为新的状态码，表示地址包成功发送。状态码同样会反映从机是否响应该地址包。
5. 应用程序查看 TWSR 的值，确定地址包已成功发送，收到的 ACK 为期望值。如果 TWSR 显示为其它值，应用程序可以执行一些特殊操作，比如调用错误处理程序。当确定状态码与预期一致后，程序将 Data 的值载入到 TWDR 寄存器中。随后软件往 TWCR 寄存器写入特定值，指示 TWI 硬件发送 TWDR 中的 Data 的值。写入的值将在随后详细说明。在写入的值中要置位 TWINT，来清零 TWINT 标志位。TWCR 寄存器的 TWINT 置位期间 TWI 不会启动任何操作。一旦软件清零 TWINT 位，TWI 模块立即启动数据包的传送。
6. 当数据包发送完毕后，TWCR 的 TWINT 标志位会被置位，TWSR 更新为新的状态码，表示数据包成功发送。状态码同样会反映从机是否响应该数据包。
7. 应用程序查看 TWSR 的值，确定数据包已成功发送，收到的 ACK 为期望值。如果 TWSR 显示为其它值，应用程序可以执行一些特殊操作，比如调用错误处理程序。当确定状态码与预期一致后，软件往 TWCR 寄存器写入特定值，指示 TWI 硬件发送 STOP 信号。写入的值将在随后详细说明。在写入的值中要置位 TWINT，来清零 TWINT 标志位。TWCR 寄存器的 TWINT 置位期间 TWI 不会启动任何操作。一旦软件清零 TWINT 位，TWI 模块立即启动 STOP 信号的传送。需要注意的是，在 STOP 信号发送完毕之后 TWINT 不会被置位。

尽管示例比较简单，但它包含了 TWI 数据传输过程中的所有规则。总结如下：

- 当 TWI 完成一次操作并等待应用程序的反馈时，TWINT 标志置位。SCL 时钟线会被一直

拉低直到 TWINT 被清零；

- 当 TWINT 标志置位，用户必须更新所有 TWI 寄存器的值为与下一个 TWI 总线周期相关的值。例如，TWDR 寄存器必须载入下一个总线周期要发送的值。
- 当更新完所有的寄存器，同时完成其它必要的操作之后，应用程序写 TWCR 寄存器。在写 TWCR 时，TWINT 位必须被置位，用来清零 TWINT 标志。TWINT 被清零之后，TWI 开始执行由 TWCR 设定的操作。

传输模式

TWI 可以工作在下面 4 种主要的模式：主机发送器（MT），主机接收器（MR），从机发送器（ST）和从机接收器（SR）。同一应用下可以使用多种模式。例如，TWI 可以使用 MT 模式往 TWI EEPROM 写入数据，用 MR 模式从 EEPROM 读取数据。如果该系统上还有其它主机，有些也可能往 TWI 发送数据，则会使用 SR 模式。这是由应用软件来决定采用何种模式。

下面会对这些模式进行详细说明。在每种模式下的数据传输中，会结合图片来描述可能的状态码。这些图片包含了如下的缩写：

S： Start 状态
Rs： REPEATED START 状态
R： 读操作标志位（SDA 为高电平）
W： 写操作标志位（SDA 为低电平）
A: 应答位（SDA 为低电平）
NA： 无应答位（SDA 为高电平）
Data：8 位数据字节
P： STOP 状态
SLA： 从机地址

图片中的圆圈用来表示 TWINT 标志置位，圆圈中的数字表示 TWSR 寄存器中的状态码，其中预分频控制位被屏蔽为“0”。在这些地方，应用程序必须执行相应的操作来继续或完成 TWI 传输。TWI 传输会被挂起，直到 TWINT 标志位被清零。

当 TWINT 标志被置位，TWSR 中的状态码用来决定适当的软件操作。各表格中给出了每个状态码下所需的软件操作和后续串行传输的细节。注意表格里 TWSR 中的预分频控制位被屏蔽为“0”。

主机发送模式

在主机发送模式中，TWI 会发送一定数量的数据字节到从机接收器。为了进入主机模式，必须发送 START 信号。接下来的地址包格式决定 TWI 是进入主机发送器模式还是主机接收器模式。如果发送 SLA+W，则进入主机发送模式。如果发送 SLA+R，则进入主机接收模式。这一章节所提到的状态码均假设预分频控制位为“0”。

通过往 TWCR 寄存器写入下列数值来发出 START 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x

TWEN 位必须置“1”来使能 TWI 接口，TWSTA 置“1”来发送 START 信号，TWINT 置“1”

来清零 TWINT 标志位。TWI 模块检测总线状态，在总线空闲时立即发送 START 信号。当发送完 START 后，硬件置位 TWINT 标志位，同时更新 TWSR 的状态码为 0x08。

为了进入主机发送模式，必须发送 SLA+W。这可通过下面操作来完成。先往 TWDR 寄存器写入 SLA+W，然后往 TWINT 位写“1”清零 TWINT 标志位来继续传输，即往 TWCR 寄存器写入下列数值来发送 SLA+W：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x

当 SLA+W 发送完成且收到应答信号后，TWINT 又被置位，同时 TWSR 的状态码更新。可能的状态码为 0x18、0x20 或 0x38。各个状态码下合适的响应会在状态码表格中详细描述。

当 SLA+W 发送成功后，可以开始发送数据包。这可通过往 TWDR 寄存器写入数据来完成。TWDR 只有在 TWINT 标志位为高时才可以写入。否则，访问被忽略，同时写冲突标志位 TWWC 会被置位。更新完 TWDR 后，往 TWINT 位写“1”清零 TWINT 标志位来继续传输。即往 TWCR 寄存器写入下列数值来发送数据：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x

当数据包发送完成且收到应答信号后，TWINT 又被置位，同时 TWSR 的状态码更新。可能的状态码为 0x28 或 0x30。各个状态码下合适的响应会在状态码表格中详细描述。

当数据发送成功后，可以继续发送数据包。这个过程一直重复，直到最后一个字节发送完毕。主机产生 STOP 信号或 REPEATED START 信号整个传输才结束。

通过往 TWCR 寄存器写入下列数值来发出 STOP 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x

通过往 TWCR 寄存器写入下列数值来发出 REPEATED START 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x

在发送 REPEATED START（状态码为 0x10）之后，TWI 接口可以再次访问相同的从机，或访问新的从机而不用发送 STOP 信号。REPEATED START 使得主机可以在不丢失总线控制权的情况下在不同从机之间，主机发送器和主机接收器模式之间进行切换。

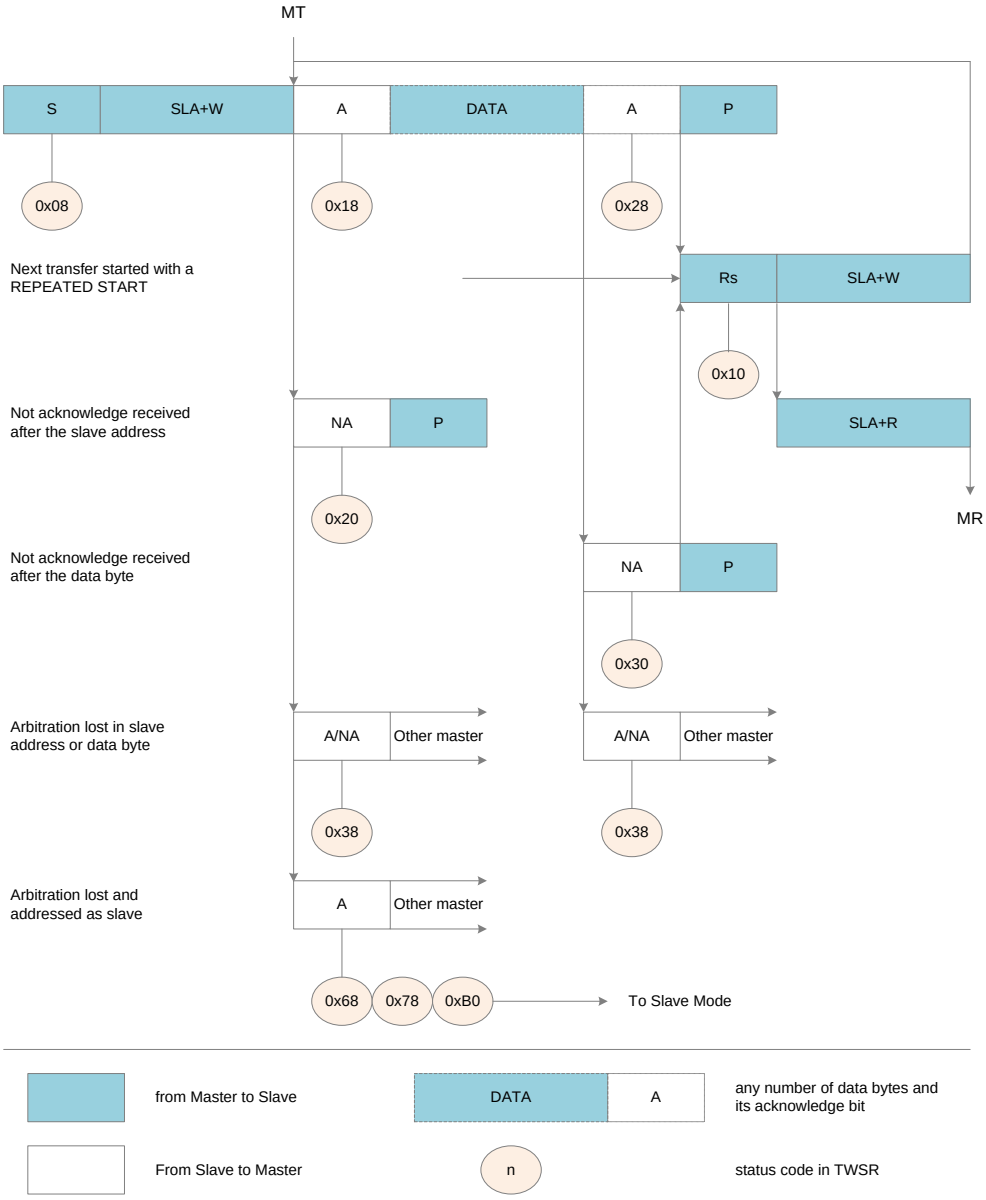
主机发送模式下的状态码及相应的操作如下表所示：

主机发送模式的状态码表

状态码	总线和硬件状态	应用软件的响应					硬件的下一步动作
		读/写 TWDR	对 TWCR 的操作				
			STA	STO	TWINT	TWEA	
0x08	START 已发送	加载 SLA+W	0	0	1	x	将发送 SLA+W； 将接收 ACK 或 NACK
0x10	REPEATED START 已发送	加载 SLA+W	0	0	1	x	将发送 SLA+W； 将接收 ACK 或 NACK

		加载 SLA+R	0	0	1	x	将发送 SLA+R； 将接收 ACK 或 NACK; 将切换到 MR 模式
0x18	SLA+W 已发送； 接收到 ACK	加载数据	0	0	1	x	将发送数据； 将接收 ACK 或 NACK
		无操作	1	0	1	x	将发送 REPEATED START
		无操作	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		无操作	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START
0x20	SLA+W 已发送； 接收到 NACK	加载数据	0	0	1	x	将发送数据； 将接收 ACK 或 NACK
		无操作	1	0	1	x	将发送 REPEATED START
		无操作	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		无操作	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START
0x28	数据字节已发送；接收到 ACK	加载数据	0	0	1	x	将发送数据； 将接收 ACK 或 NACK
		无操作	1	0	1	x	将发送 REPEATED START
		无操作	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		无操作	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START
0x30	数据字节已发送；接收到 NACK	加载数据	0	0	1	x	将发送数据； 将接收 ACK 或 NACK
		无操作	1	0	1	x	将发送 REPEATED START
		无操作	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		无操作	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START
0x38	SLA+W 或数据仲裁失败	无操作	0	0	1	x	将释放总线； 将进入未寻址从机模式
		无操作	1	0	1	x	将在空闲时发送 START

主机发送模式的格式和状态如下图所示：



主机发送模式的格式和状态图

主机接收模式

在主机接收模式中，TWI 会从从机发送器接收一定数量的数据字节。为了进入主机模式，必须发送 START 信号。接下来的地址包格式决定 TWI 是进入主机发送器模式还是主机接收器模式。如果发送 SLA+W，则进入主机发送模式。如果发送 SLA+R，则进入主机接收模式。这一章节所提到的状态码均假设预分频控制位为“0”。

通过往 TWCR 寄存器写入下列数值来发出 START 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x

TWEN 位必须置“1”来使能 TWI 接口，TWSTA 置“1”来发送 START 信号，TWINT 置“1”来清零 TWINT 标志位。TWI 模块检测总线状态，在总线空闲时立即发送 START 信号。当发送

完 START 后，硬件置位 TWINT 标志位，同时更新 TWSR 的状态码为 0x08。

为了进入主机接收模式，必须发送 SLA+R。这可通过下面操作来完成。先往 TWDR 寄存器写入 SLA+R，然后往 TWINT 位写“1”清零 TWINT 标志位来继续传输，即往 TWCR 寄存器写入下列数值来发送 SLA+R：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x

当 SLA+R 发送完成且收到应答信号后，TWINT 又被置位，同时 TWSR 的状态码更新。可能的状态码为 0x38、0x40 或 0x48。各个状态码下合适的响应会在状态码表格中详细描述。

当 SLA+R 发送成功后，可以开始接收数据包。通过往 TWINT 位写“1”清零 TWINT 标志位来继续接收。即往 TWCR 寄存器写入下列数值来启动接收：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	0	x	1	0	x

当数据包接收完成且发送应答信号后，TWINT 又被置位，同时 TWSR 的状态码更新。可能的状态码为 0x50 或 0x58。各个状态码下合适的响应会在状态码表格中详细描述。

当数据接收成功后，可以继续接收数据包。这个过程一直重复，直到最后一个字节接收完毕。主机接收到最后一个字节后，必须发送 NACK 应答信号给从机发送器。主机产生 STOP 信号或 REPEATED START 信号整个接收才结束。

通过往 TWCR 寄存器写入下列数值来发出 STOP 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x

通过往 TWCR 寄存器写入下列数值来发出 REPEATED START 信号：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	1	0	x	1	0	x

在发送 REPEATED START（状态码为 0x10）之后，TWI 接口可以再次访问相同的主机，或访问新的主机而不用发送 STOP 信号。REPEATED START 使得主机可以在不丢失总线控制权的情况下在不同从机之间，主机发送器和主机接收器模式之间进行切换。

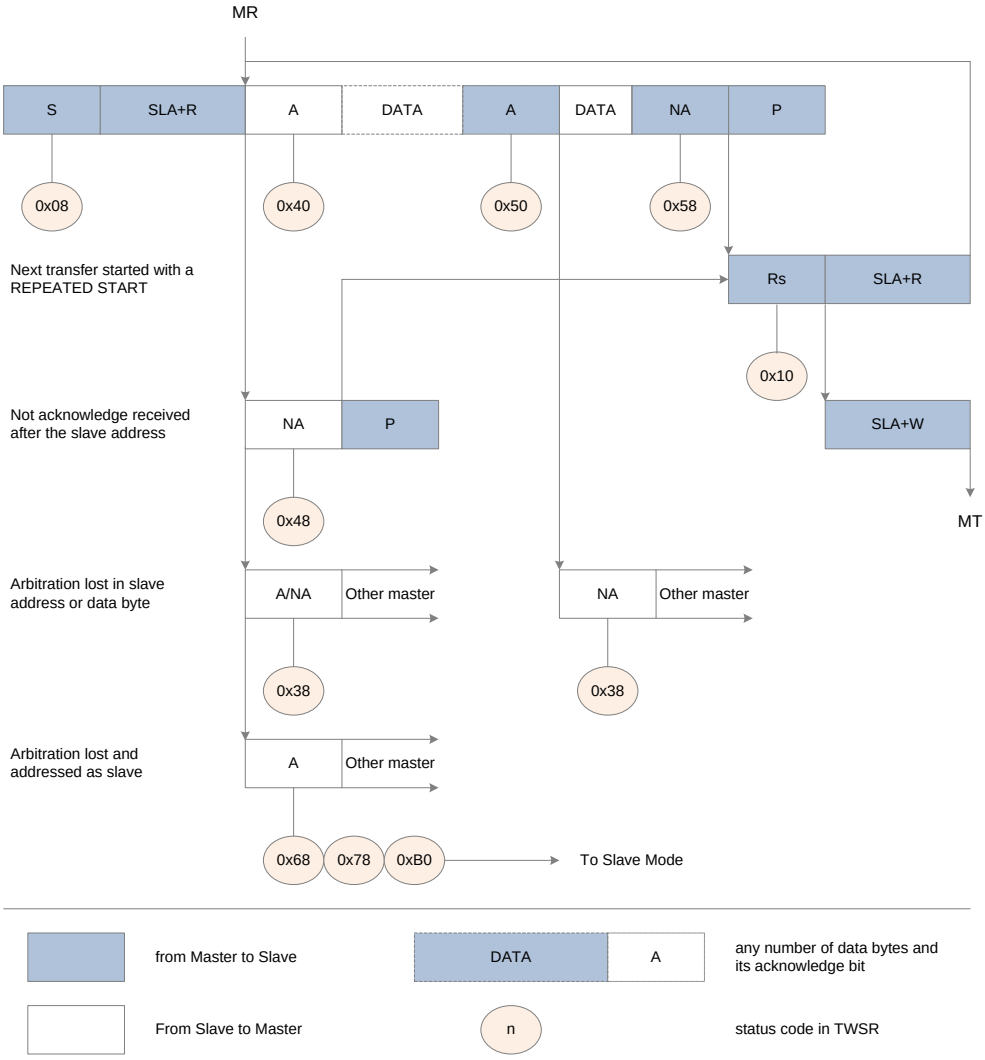
主机接收模式下的状态码及相应的操作如下表所示：

主机接收模式的状态码表

状态码	总线和硬件状态	应用软件的响应					硬件的下一步动作
		读/写 TWDR	对 TWCR 的操作				
			STA	STO	TWINT	TWEA	
0x08	START 已发送	加载 SLA+R	0	0	1	x	将发送 SLA+R； 将接收 ACK 或 NACK
0x10	REPEATED START 已发送	加载 SLA+R	0	0	1	x	将发送 SLA+R； 将接收 ACK 或 NACK
		加载 SLA+W	0	0	1	x	将发送 SLA+W； 将接收 ACK 或 NACK;

							将切换到 MT 模式
0x38	SLA+R 或数据仲裁失败	无操作	0	0	1	x	将释放总线； 将进入未寻址从机模式
		无操作	1	0	1	x	将在空闲时发送 START
0x40	SLA+R 已发送； 接收到 ACK	无操作	0	0	1	0	将接收数据； 将发送 NACK
		无操作	0	0	1	1	将接收数据； 将发送 ACK
0x48	SLA+R 已发送； 接收到 NACK	无操作	1	0	1	x	将发送 REPEATED START
		无操作	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		无操作	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START
0x50	数据字节已接收； ACK 已发送	读取数据	0	0	1	0	将接收数据； 将发送 NACK
		读取数据	0	0	1	1	将接收数据； 将发送 ACK
0x58	数据字节已接收； NACK 已发送	读取数据	1	0	1	x	将发送 REPEATED START
		读取数据	0	1	1	x	将发送 STOP； 将复位 TWSTO 标志
		读取数据	1	1	1	x	将发送 STOP； 将复位 TWSTO 标志； 将发送 START

主机接收模式的格式和状态如下图所示：



主机接收模式的格式和状态图

从机接收模式

在从机接收模式中，可以从主机发送器接收一定数量的数据字节。这一章节所提到的状态码均假设预分频控制位为“0”。

为启动从机接收模式，要设置 TWAR 和 TWCR 寄存器。

TWAR 需设置如下：

TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
器件从机地址							

TWAR 的高 7 位是主机寻址时 TWI 接口会响应的从机地址。若 LSB 置位，TWI 会响应广播呼叫地址（0x00），否则忽略广播呼叫地址。

TWCR 需设置如下：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
0	1	0	0	0	1	0	x

TWEN 必须置位以使能 TWI 接口，TWEA 必须置位以使主机寻址（从机地址或广播呼叫）到自己时返回确认信息 ACK。TWSTA 和 TWSTO 必须清零。

初始化 TWAR 和 TWCR 之后，TWI 接口开始等待，直到自己的从机地址（或广播地址）被寻址。当紧跟着从机地址的数据方向位为“0”（表示写操作）时，TWI 进入从机接收模式。当数据方向位为“1”（表示读操作）时，TWI 进入从机发送模式。接收到自己的从机地址和写操作标志位后，TWINT 标志位被置位，有效的状态码也更新到 TWSR 中。各个状态码下合适的响应会在状态码表格中详细描述。需要注意的是，当主机模式下的 TWI 仲裁失败后也可以进入从机接收模式（见状态码 0x68 和 0x78）。

如果在传输过程中 TWEA 位被复位，TWI 将在接收到一个字节后返回 NACK（高电平）到 SDA 线上。这可用来表示从机不能接收更多的数据。当 TWEA 位为“0”时，TWI 也不会响应自己的从机地址。不过 TWI 仍会监听总线，一旦 TWEA 被置位，就可以恢复地址识别并响应。也就是说，可以利用 TWEA 暂时将 TWI 接口从总线中隔离出来。

在除空闲模式外的其它休眠模式时，TWI 接口的时钟可以被关闭。若是能了从机接收模式，接口将利用总线时钟继续响应从机地址或广播地址。地址匹配将唤醒 MCU。在唤醒期间，TWI 接口将保持 SCL 为低电平，直到 TWINT 标志被清零。当 TWI 接口时钟恢复正常后可以接收更多的数据。

从机接收模式的状态码如下表所示：

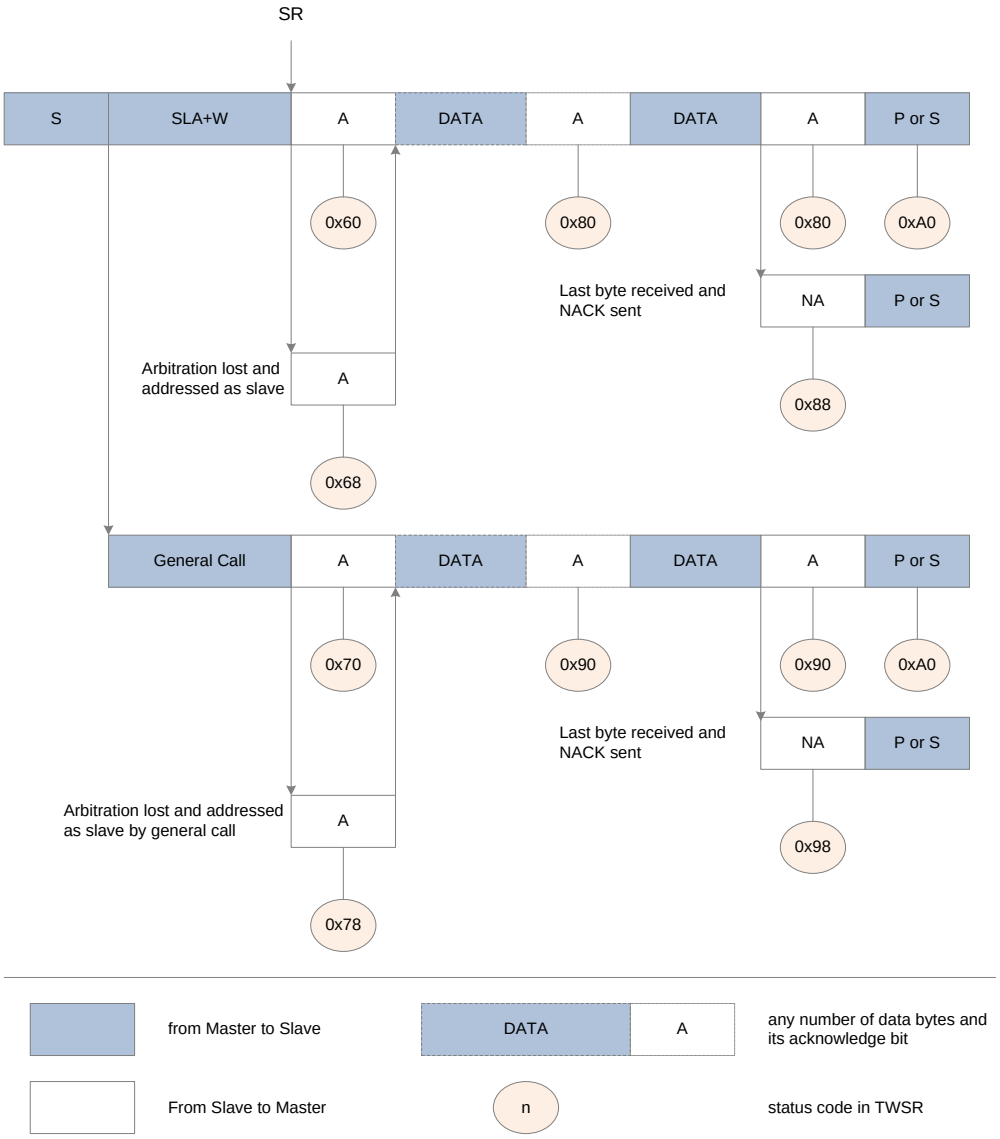
从机接收模式的状态码表

状态码	总线和硬件状态	应用软件的响应					硬件的下一步动作
		读/写	对 TWCR 的操作				
		TWDR	STA	STO	TWINT	TWEA	
0x60	SLA+W 已接收；ACK 已发送	无操作	x	0	1	0	将接收数据；将发送 NACK
		无操作	x	0	1	1	将接收数据；将发送 ACK
0x68	发送 SLA+R/W 时仲裁失败；SLA+W 已接收；ACK 已发送	无操作	x	0	1	0	将接收数据；将发送 NACK
		无操作	x	0	1	1	将接收数据；将发送 ACK
0x70	广播地址已接收；ACK 已发送	无操作	x	0	1	0	将接收数据；将发送 NACK
		无操作	x	0	1	1	将接收数据；将发送 ACK
0x78	发送 SLA+R/W 时仲裁失败；SLA+W 已接收；ACK 已发送	无操作	x	0	1	0	将接收数据；将发送 NACK
		无操作	x	0	1	1	将接收数据；将发送 ACK
0x80	自身数据已接收；ACK 已发送	读取数据	x	0	1	0	将接收数据；将发送 NACK
		读取数据	x	0	1	1	将接收数据；将发送 ACK
0x88	自身数据已接收；	读取数	0	0	1	0	将切换到未寻址从

	NACK 已发送	据					机模式； 将不响应从机地址和广播
		读取数据	0	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播
		读取数据	1	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播； 总线空闲时将发送 START
		读取数据	1	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播； 总线空闲时将发送 START
0x90	广播数据已接收； ACK 已发送	读取数据	x	0	1	0	将接收数据； 将发送 NACK
		读取数据	x	0	1	1	将接收数据； 将发送 ACK
0x98	广播数据已接收； NACK 已发送	读取数据	0	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播
		读取数据	0	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播
		读取数据	1	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播； 总线空闲时将发送 START
		读取数据	1	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应

							广播； 总线空闲时将发送 START
0xA0	从机工作时接收到 STOP 或 REPEATED START	无操作	0	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播
		无操作	0	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播
		无操作	1	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播； 总线空闲时将发送 START
		无操作	1	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播； 总线空闲时将发送 START

从机接收模式的格式和状态图如下所示：



从机接收模式的格式和状态图

从机发送模式

在从机发送模式中，可以往主机接收器发送一定数量的数据字节。这一章节所提到的状态码均假设预分频控制位为“0”。

为启动从机接收模式，要设置 TWAR 和 TWCR 寄存器。

TWAR 需设置如下：

TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
器件从机地址							

TWAR 的高 7 位是主机寻址时 TWI 接口会响应的从机地址。若 LSB 置位，TWI 会响应广播呼叫地址（0x00），否则忽略广播呼叫地址。

TWCR 需设置如下：

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
0	1	0	0	0	1	0	x

TWEN 必须置位以使能 TWI 接口，TWEA 必须置位以使主机寻址（从机地址或广播呼叫）到自己时返回确认信息 ACK。TWSTA 和 TWSTO 必须清零。

初始化 TWAR 和 TWCR 之后，TWI 接口开始等待，直到自己的从机地址（或广播地址）被寻址。当紧跟着从机地址的数据方向位为“0”（表示写操作）时，TWI 进入从机接收模式。当数据方向位为“1”（表示读操作）时，TWI 进入从机发送模式。接收到自己的从机地址和读操作标志位后，TWINT 标志位被置位，有效的状态码也更新到 TWSR 中。各个状态码下合适的响应会在状态码表格中详细描述。需要注意的是，当主机模式下的 TWI 仲裁失败后也可以进入从机发送模式（见状态码 0xB0）。

如果在传输过程中 TWEA 位被复位，TWI 将在发送最后一个字节后切换到未寻址从机模式。主机接收器为最后一个字节的传输给出 NACK 或 ACK 后，TWSR 寄存器中的状态码将会更新为 0xC0 或 0xC8。如果主机接收器继续传输操作，从机发送器不会响应，主机将会接收到全“1”的数据（即 0xFF）。当从机发送完最后一个字节的数据（TWEA 被清零）并期望得到 NACK 响应，而主机想要接收更多的数据而发送 ACK 作为响应时，TWSR 会更新为 0xC8。

当 TWEA 位为“0”时，TWI 也不会响应自己的从机地址。不过 TWI 仍会监听总线，一旦 TWEA 被置位，就可以恢复地址识别并响应。也就是说，可以利用 TWEA 暂时将 TWI 接口从总线中隔离出来。

在除空闲模式外的其它休眠模式时，TWI 接口的时钟可以被关闭。若是能了从机接收模式，接口将利用总线时钟继续响应从机地址或广播地址。地址匹配将唤醒 MCU。在唤醒期间，TWI 接口将保持 SCL 为低电平，直到 TWINT 标志被清零。当 TWI 接口时钟恢复正常后可以接收更多的数据。

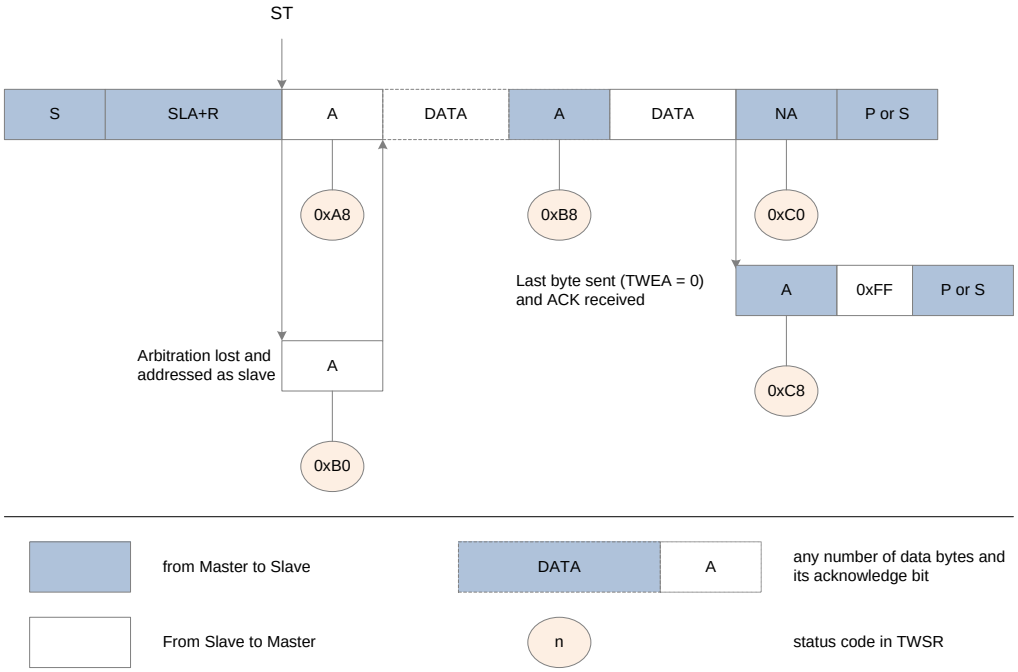
从机发送模式的状态码如下表所示：

从机发送模式的状态码表

状态码	总线和硬件状态	应用软件的响应					硬件的下一步动作
		读/写 TWDR	对 TWCR 的操作				
			STA	STO	TWINT	TWEA	
0xA8	SLA+R 已接收；ACK 已发送	加载数据	x	0	1	0	将发送最后一个数据；期望接收 NACK
		加载数据	x	0	1	1	将发送数据；将接收 ACK
0xB0	发送 SLA+R/W 时仲裁失败；SLA+R 已接收；ACK 已发送	加载数据	x	0	1	0	将发送最后一个数据；期望接收 NACK
		加载数据	x	0	1	1	将发送数据；将接收 ACK
0xB8	数据已发送；ACK 已接收	加载数据	x	0	1	0	将发送最后一个数据；期望接收 NACK
		加载数据	x	0	1	1	将发送数据；

							将接收 ACK
0xC0	数据已发送； NACK 已接收	无操作	0	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播
		无操作	0	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播
		无操作	1	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播； 总线空闲时将发送 START
		无操作	1	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播； 总线空闲时将发送 START
0xC8	最后一个数据已发送； ACK 已接收	无操作	0	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播
		无操作	0	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播
		无操作	1	0	1	0	将切换到未寻址从机模式； 将不响应从机地址和广播； 总线空闲时将发送 START
		无操作	1	0	1	1	将切换到未寻址从机模式； 将响应从机地址； TWGCE=1 时将响应广播； 总线空闲时将发送 START

从机发送模式的格式和状态如下图所示：



从机发送模式的格式和状态图

其他状态

有两个状态码没有相应的 TWI 状态定义，如下表所示：

其他状态码表

状态码	总线和硬件状态	应用软件的响应					硬件的下一步动作
		读/写 TWDR	对 TWCR 的操作				
			STA	STO	TWINT	TWEA	
0xF8	无状态信息；TWINT= 0	无操作	不操作 TWCR				等待或进行当前操作
0x00	非法的 START 或 STOP 引起的总线错误	无操作	0	1	1	x	只影响内部硬件；不会发送 STOP 到总线上；总线释放并清零 TWSTO 位

状态码 0xF8 表示当前没有相关信息，因为 TWINT 标志为“0”。这种状态可能发生在 TWI 接口没有参与串行传输或当前传输还没有完成。

状态 0x00 表示串行传输过程中发生了总线错误。当非法的 START 或 STOP 出现时总线错误就会发生。比如说在地址和数据、地址和 ACK 之间出现了 START 或 STOP。总线错误将置位 TWINT。为了从错误中恢复，必须置位 TWSTO，并通过写“1”以清零 TWINT。这将使 TWI 接口进入未寻址从机模式而不会产生 STOP，以及释放 SCL 和 SDA，并清零 TWSTO 位。

组合模式

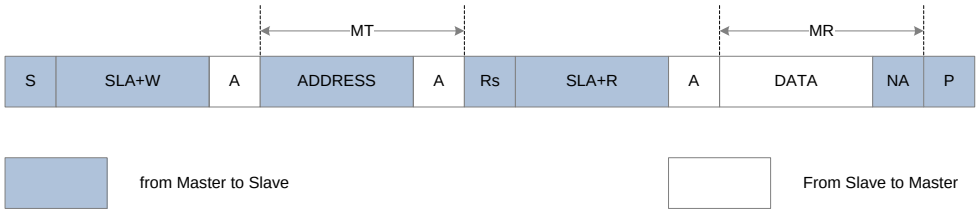
在某些情况下，为了完成期望的工作，必须将几种 TWI 模式组合起来。例如，从串行 EEPROM

读取数据，典型的传输包括以下步骤：

1. 传输必须启动；
2. 必须告诉 EEPROM 应该读取数据的位置；
3. 必须完成读操作；
4. 传输必须结束。

注意数据可以从主机传送到从机，反之亦然。主机告诉从机要读取数据的位置，采用的是主机发送模式。接下来，从从机读取数据，采用的是主机接收模式。传输的方向会改变。主机必须保持各个阶段的总线控制权，所有的步骤是不间断的操作。如果在多主机系统中，在步骤 2 和 3 之间另有主机改变了读取数据的位置，则打破了这一原则，主机读取数据的位置会是错误的。改变数据传输的方向是通过在传送地址字节和接收数据之间发送 REPEATED START 来实现的。发送 REPEATED START 之后，主机仍拥有总线控制权。

下图描述了这个传输过程：



组合多种 TWI 模式来访问串行 EEPROM 图

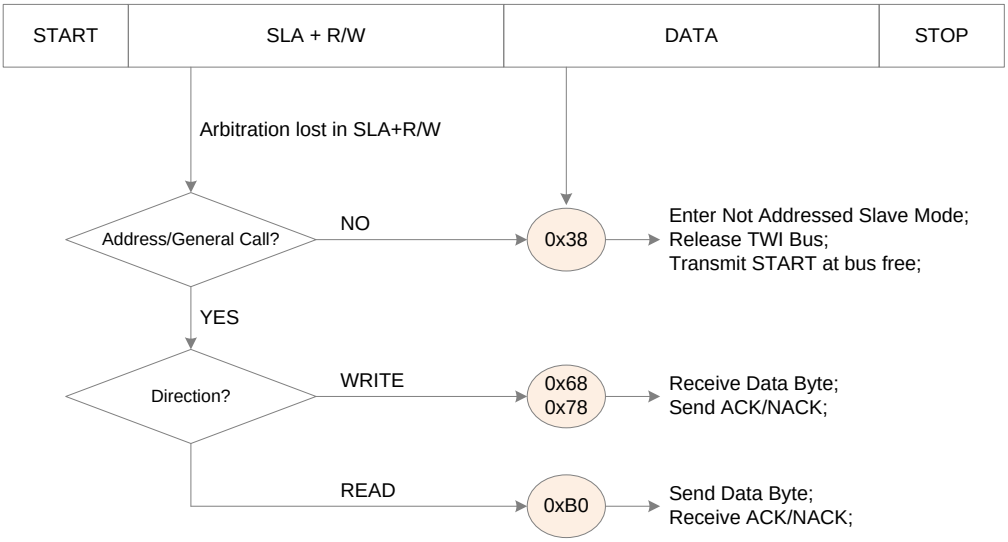
多主机系统及仲裁

如果有多个主机连接在同一 TWI 总线上，它们中的一个或多个也许会同时开始数据传输。TWI 协议确保在这种情况下，通过一个仲裁过程，允许其中的一个主机进行传送也不会丢失数据。下面以两个主机试图向从机发送数据为例来描述总线仲裁的过程。

有几种不同的情况会产生总线仲裁过程：

- 两个或更多的主机同时与一个从机进行通信。在这种情况下，无论主机还是从机都不知道总线上有竞争；
- 两个或更多的主机同时对同一个从机进行不同的数据或操作方向访问。这种情况下就会发生仲裁，在 READ/WRITE 位或数据位。当有其它主机往 SDA 线上发送“0”时，往 SDA 线上发送“1”的主机就会仲裁失败。失败的主机将会切换到未被寻址的从机模式，或者等待总线空闲时发送一个新的 START 信号，这都取决于应用软件的操作。
- 两个或更多的主机访问不同的从机。在这种情况下，总线仲裁发生在 SLA 阶段。当有其它主机往 SDA 线上发送“0”时，往 SDA 线上发送“1”的主机就会仲裁失败。在 SLA 总线仲裁时失败的主机将切换到从机模式，并检查自己是否被获得总线控制权的主机寻址。如果被寻址，它将进入 SR 或 ST 模式，这取决于 SLA 后面的 READ/WRITE 位。如果未被寻址，它将切换到未被寻址的从机模式，或者等待总线空闲时发送一个新的 START 信号，这取决于应用软件的操作。

下图描述了总线仲裁的过程：



总线仲裁过程图

寄存器定义

TWI 寄存器列表

寄存器	地址	默认值	描述
TWBR	0x B8	0x00	TWI 比特率寄存器
TWSR	0xB9	0x00	TWI 状态寄存器
TWAR	0xBA	0x00	TWI 地址寄存器
TWDR	0xBB	0x00	TWI 数据寄存器
TWCR	0xBC	0x00	TWI 控制寄存器
TWAMR	0xBD	0x00	TWI 地址屏蔽寄存器

TWBR – TWI 比特率寄存器

TWBR – TWI 比特率寄存器								
地址: 0xB8					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	TWBR[7:0]	TWI 比特率选择控制位。 TWBR 是比特率发生器分频因子。比特率发生器是一个分频器，用来在主机模式下产生 SCL 时钟。比特率的计算公式如下所示： $f_{scl} = f_{sys}/(16 + 2*TWBR*4^{TWPS})$。

TWSR – TWI 状态寄存器

TWSR – TWI 状态寄存器								
地址: 0xB9					默认值: 0x00			
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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:3	TWS[7:3]	TWI 状态标志位。 5 位的 TWS 反应 TWI 逻辑和总线的状态。不同的状态值有不同的含义，具体见 TWI 工作模式的描述。从 TWSR 读到的值包括 5 位的状态值和 2 位的预分频控制位，在检测状态时应屏蔽预分频位为“0”。这是状态检测独立于预分频器的设置。
2	-	保留。
1	TWPS1	TWI 预分频控制高位。 TWPS1 和 TWPS0 一起组成 TWPS[1:0]，用来控制比特率预分频因子，和 TWBR 一起控制比特率。
0	TWPS0	TWI 预分频控制低位。 TWPS0 和 TWPS1 一起组成 TWPS[1:0]，用来控制比特率预分频因子，和 TWBR 一起控制比特率。

TWPS[1:0]	预分频因子
0	1
1	4
2	16
3	64

TWAR – TWI 地址寄存器

TWAR – TWI 地址寄存器								
地址: 0xBA					默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:1	TWA[6:0]	TWI 从机地址位。 TWA 为 TWI 从机地址。当 TWI 工作在从机模式下时，TWI 将根据这个地址进行响应。主机模式不需要此地址。但在多主机系统中，也需要设置从机地址以便其它主机访问。

0	TWGCE	TWI 广播识别使能控制位。 当设置 TWGCE 位为“1”时，使能 TWI 总线广播识别。 当设置 TWGCE 位为“0”时，禁止 TWI 总线广播识别。 当 TWGCE 置位且接收到的地址帧为 0x00 时，TWI 模块会响应此总线广播。

TWDR – TWI 数据寄存器

TWDR – TWI 数据寄存器								
地址: 0xBB						默认值: 0x00		
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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:0	TWD[7:0]	TWI 数据寄存器。 TWD 是将要传送总线上的下一个字节，或者是刚从总线上接收到的上一个字节。

TWCR – TWI 控制寄存器

TWCR – TWI 控制寄存器								
地址: 0xBC						默认值: 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	R/W	-	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	TWINT	TWI 中断标志位。 当 TWI 完成当前工作，希望应用软件介入时，硬件将置位 TWINT 位。若全局中断置位且 TWIE 位置位时，将产生 TWI 中断，MCU 将执行 TWI 中断服务程序。 当 TWINT 标志被置位时，SCL 信号的低电平将被延长。 TWINT 标志位只能通过往该位写“1”的方式来清零。即使执行中断服务程序，硬件也不会自动清零该位。同时要注意，清零该位将立即开启 TWI 的操作。因此，在清零 TWINT 位之前，要首先完成对 TWAR，TWAMR，TWSR 和 TWDR 寄存器的访问。
6	TWEA	TWI 使能应答控制位。 TWEA 位控制应答脉冲的产生。当设置 TWEA 位为“1”，且满足如下条件之一时，将会在 TWI 总线上产生应答脉冲： 1） 收到器件的从机地址； 2） TWGCE 置位时收到广播呼叫；

		3） 在主机接收或从机接收模式下收到一个字节的数据。 当设置 TWEA 位为“0”时，器件暂时和 TWI 总线脱离连接。置位后器件重新恢复地址识别。
5	TWSTA	TWI 起始状态控制位。 当 CPU 希望自己成为 TWI 总线上的主机时需要置位 TWSTA 位。硬件将检测总线是否可用，当总线是空闲时，就在总线上产生起始状态。当总线非空闲时，TWI 将一直等到检测到停止状态出现，然后产生起始状态来声明自己希望成为主机。发送完起始状态之后软件必须清零 TWSTA 位。
4	TWSTO	TWI 停止状态控制位。 在主机模式下当 TWSTO 位为“1”时，TWI 将在总线上产生停止状态，然后自动清零 TWSTO 位。在从机模式下，置位 TWSTO 位可以使 TWI 从错误状态恢复过来。这时不会产生停止状态，只会让 TWI 返回到一个定义好的未被寻址的从机模式，同时释放 SCL 和 SDA 信号线至高阻状态。
3	TWWC	TWI 写冲突标志位。 当 TWINT 标志位为低时，写 TWDR 寄存器将会置位 TWWC 标志位。当 TWINT 标志位为高时，写 TWDR 寄存器将会清零 TWWC 标志位。
2	TWEN	TWI 使能控制位。 TWEN 位使能 TWI 操作并激活 TWI 接口。当设置 TWEN 位为“1”时，TWI 控制 IO 引脚连接到 SCL 和 SDA 引脚。当设置 TWEN 位为“0”时，TWI 接口模块被关闭，所有的传输被终止，包括正在进行的操作。
1	-	保留。
0	TWIE	TWI 中断使能控制位。 当设置 TWIE 位为“1”，且全局中断置位时，只要 TWINT 标志位为高，就会激活 TWI 中断请求。

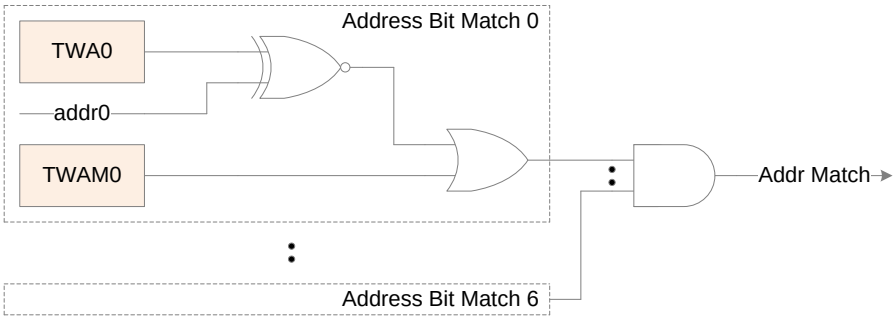
TWAMR – TWI 地址屏蔽寄存器

TWAMR – TWI 地址屏蔽寄存器								
地址: 0xBD						默认值: 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
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:1	TWAM[6:0]	TWI 地址屏蔽控制位。 TWAM 为 7 位 TWI 从机地址屏蔽控制。 TWAM 的每一位用来屏蔽（禁止）TWAR 中相应地址位。当屏蔽位置位时，地址匹配逻辑将忽略接收到的地址位与 TWA 相应位的比较结果。下图给出了地址匹配逻辑的详细信息。
0	-	保留。

TWI 地址匹配逻辑

下图为 TWI 地址匹配逻辑框图：



TWI 地址匹配逻辑结构图

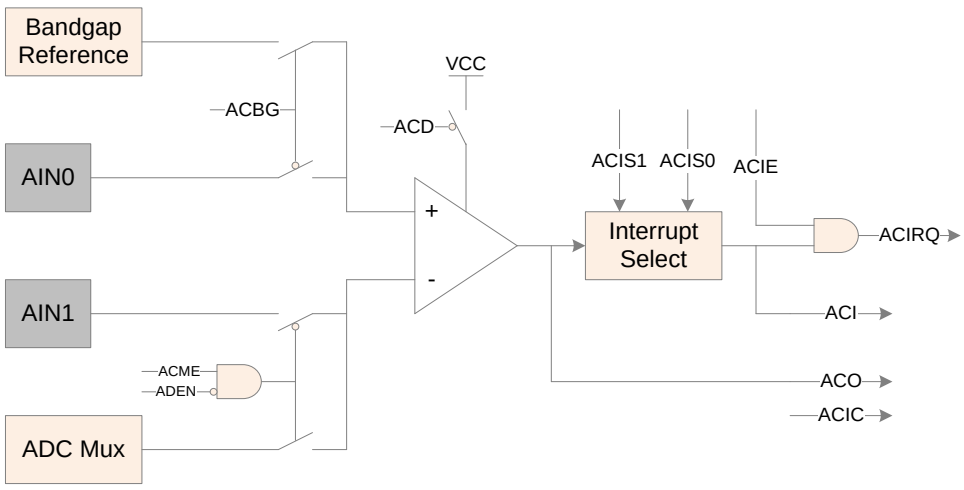
模拟比较器

- 6mV 的比较精度
- 支持 2 路外部独立模拟输入
- 支持 8 路 ADC 通道输入
- 支持内部 1.25V/2.56V 参考电压

综述

模拟比较器对正极的值与负极的值进行比较，当正极上的电压比负极上的电压高时，模拟比较器的输出 ACO 被置位。当 ACO 的电平发生变化时，信号的边沿可用来触发中断。输出信号 ACO 还可用来触发定时计数器 1 的输入捕捉功能。

模拟比较器的结构图如下图所示。



Analog Comparator 结构图

模拟比较器的输入

模拟比较器的两个输入端都有不同的输入源。正极的输入有外部引脚 AIN0 和内部的基准电压源，输入源的选择由位于 AC 控制和状态寄存器的 ACSR 中的 ACBG 位来控制，具体见寄存器描述。负极的输入有外部引脚 AIN1 和 ADC 多路器的输出，输入源的选择由位于 ADC 寄存器中的 ACME 和 ADEN 位来控制，当 ADC 的功能未使能即 ADEN 位为“0”，且模拟比较器多路器使能 ACME 位为“1”时，负极的输入源为 ADC 多路器的输出，否则为外部引脚 AIN1 输入。

下表为模拟比较器的输入控制表格。

AC 负极输入端控制

ACME	ADEN	MUX[2:0]	AC Negative Input
0	x	xxx	AIN1
1	1	000	ADC0
1	1	001	ADC1
1	1	010	ADC2
1	1	011	ADC3
1	1	100	ADC4
1	1	101	ADC5
1	1	110	ADC6
1	1	111	ADC7

寄存器定义

AC 寄存器列表

寄存器	地址	默认值	描述
ACSR	0x50	0x00	AC 控制和状态寄存器
ADCSRB	0x7B	0x00	ADC 控制和状态寄存器 B
DIDR1	0x7F	0x00	数字输入禁止控制寄存器 1

ACSR – AC 控制和状态寄存器

ACSR – AC 控制和状态寄存器								
地址: 0x50					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	ACD	模拟比较器禁止位。

		当设置 ACD 位为“1”时，模拟比较器被关闭。 当设置 ACD 位为“0”时，模拟比较器被开启。
6	ACBG	模拟比较器基准电压选择控制位。 当设置 ACBG 位为“1”时，正极选择内部基准电压源作为输入。 当设置 ACBG 位为“0”时，正极选择外部引脚 AIN0 作为输入。
5	ACO	模拟比较器的输出状态位。 模拟比较器的输出经过同步之后直接连到 ACO 位。软件可读取 ACO 位的值来获取模拟比较器的输出值。
4	ACI	模拟比较器的中断标志位。 当模拟比较器的输出事件触发了由 ACIS 位定义的中断模式时，ACI 位被置位。 当中断使能位 ACIE 为“1”且全局中断置位时，中断产生。执行模拟比较器中断服务程序时，ACI 将自动清零，或对 ACI 位写“1”也可清零该位。
3	ACIE	模拟比较器的中断使能位。 当设置 ACIE 位为“1”，且全局中断置位时，模拟比较器的中断被使能。 当设置 ACIE 位为“0”时，模拟比较器的中断被禁止。
2	ACIC	模拟比较器输入捕捉使能位。 当设置 ACIC 位为“1”时，定时计数器 1 的输入捕捉源来自模拟比较器的输出 ACO。 当设置 ACIC 位为“0”时，定时计数器 1 的输入捕捉源来自外部引脚 ICP1。
1	ACIS1	模拟比较器中断模式控制高位。 ACIS1 和 ACIS0 一起组成 ACIS[1:0]，用来控制模拟比较器的中断触发方式。
0	ACIS0	模拟比较器中断模式控制低位。 ACIS0 和 ACIS1 一起组成 ACIS[1:0]，用来控制模拟比较器的中断触发方式。

ACIS[1:0]	中断模式
0	ACO 的上升沿或下降沿触发
1	保留。
2	ACO 的下降沿触发
3	ACO 的上升沿触发

ADCSRB – ADC 控制和状态寄存器 B

ADCSRB – ADC 控制和状态寄存器 B								
地址: 0x7B	默认值: 0x00							
Bit	7	6	5	4	3	2	1	0
Name	-	ACME	-	-	-	ADTS2	ADTS1	ADTS0
R/W	-	R/W	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	-	保留。
6	ACME	模拟比较器多路复用器使能控制位。 当设置 ACME 位为“1”且 ADC 被关掉（ADCSRA 寄存器的 ADEN 位为“0”）时，

		模拟比较器的负极输入由 ADC 的多路复用器来选择。 当设置 ACME 位为“0”时，模拟比较器的负极输入来自于 AIN1 引脚。
5:3	-	保留。
2:0	ADTS	见 ADC 寄存器描述。

DIDR1 – 数字输入禁止控制寄存器 1

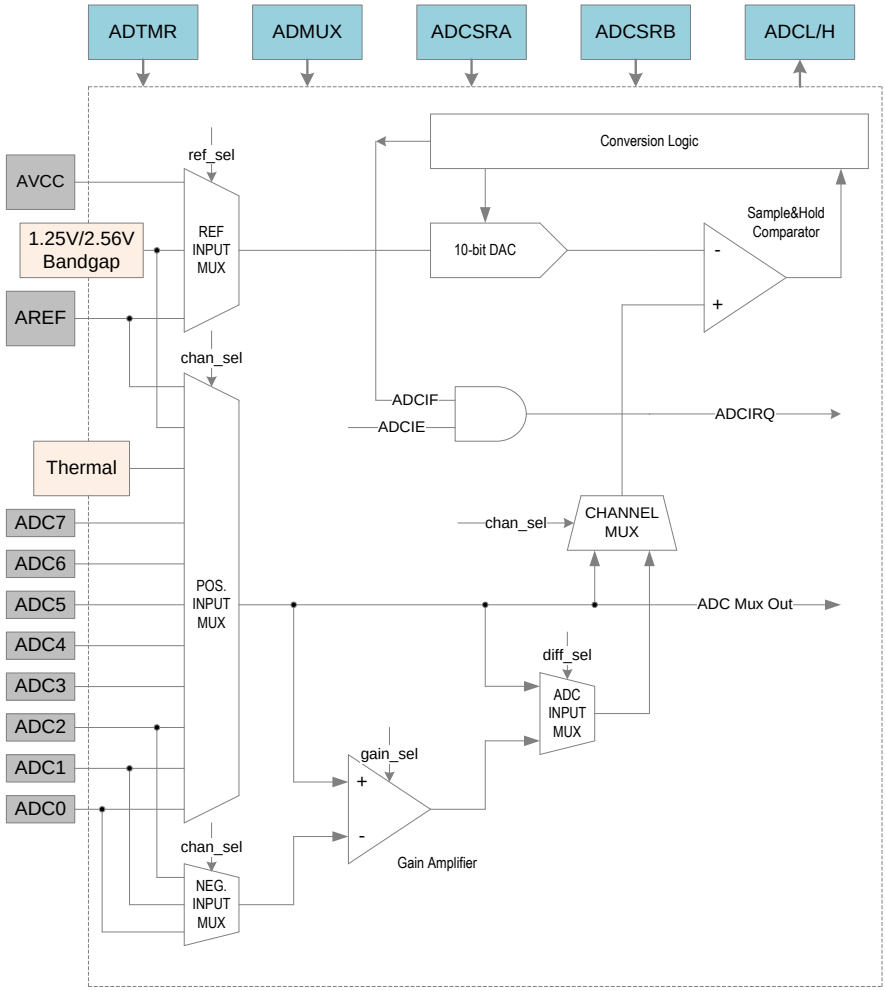
DIDR1 – 数字输入禁止控制寄存器 1								
地址: 0x7F					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	AIND1	AIND0
R/W	-	-	-	-	-	-	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7:2	-	保留。
1	AIND1	AIN1 引脚数字输入禁止控制位。 当设置 AIND1 位为“1”时，引脚 AIN1 的数字输入端被禁止，并一直为零。当使能模拟比较器时，AIN1 的数字输入端功能不需要，因此须置位 AIND1。 当设置 AIND1 位为“0”时，引脚 AIN1 的数字输入端被使能，引脚上的信号可输入到内部数字逻辑，此时须置位 ACD 位，即关闭模拟比较器。
0	AIND0	AIN0 引脚数字输入禁止控制位。 当设置 AIND0 位为“1”时，引脚 AIN0 的数字输入端被禁止，并一直为零。当使能模拟比较器时，AIN0 的数字输入端功能不需要，因此须置位 AIND0。 当设置 AIND0 位为“0”时，引脚 AIN0 的数字输入端被使能，引脚上的信号可输入到内部数字逻辑，此时须置位 ACD 位，即关闭模拟比较器。

ADC 模数转换器

- 10 位分辨率，DNL 为±1LSB，INL 为±1.5LSB
- 最高分辨率时采样率高达 250KSPS
- 8 路复用的单端输入通道
- 3 组带增益放大的差分输入通道，增益放大倍数为 7.5，15，22.5 和 30
- 自带温度传感器
- ADC 输入电压范围为 0-VCC
- 连续转换或单次转换模式
- 可选内部 1.25V/2.56V 参考电压或外部 VCC 为参考电压
- 基于中断源的自动开始转换触发模式
- 转换结果支持可选左对齐模式
- ADC 转换结束中断

综述



ADC **结构图**

模数转换器为一个 10 位的逐次逼近型 ADC。ADC 与一个 8 通道的模拟多路器连接，能对来自端口 A 的 8 路单端输入电压进行采样，单端输入电压以 0V（GND）为基准。

ADC 还支持 3 路差分电压输入组合，分别为 ADC0 与 ADC3，ADC1 与 ADC4 以及 ADC2 与 ADC5 组合。每组均有可编程增益放大器，在 AD 转换前给差分输入电压提供 7.5x，15x，22.5x 和 30x 的可选放大级。ADC 包含一个采样保持电路，以确保在转换过程中输入到 ADC 的电压保持恒定。

ADC 的操作

ADC 通过逐次逼近的方法将输入的模拟电压转换成一个 10 位的数字量。最小值代表 GND，最大值代表基准电压减去 1LSB。基准电压源可以为 ADC 的电源电压 AVCC，外部基准电压 AREF 或内部 1.25V/2.56V 的参考电压，通过写 ADMUX 寄存器的 REFS 位来选择。

模拟输入通道可以通过写 ADMUX 寄存器的 MUX 位来选择。任何 ADC 的输入引脚，外部基准电压引脚，以及内部参考电压源均可作为 ADC 的单端输入。ADC 的输入引脚 0-5 可作为 ADC 的差分输入。差分增益可通过写 ADTMR 寄存器的 GAIN 位来选择。

通过设置 ADCSRA 寄存器的 ADEN 位即可启动 ADC，ADEN 清零时 ADC 并不耗电，因此建议在进入睡眠模式之前关闭 ADC。

ADC 转换结果为 10 位，存放与 ADC 数据寄存器 ADCH 及 ADCL 中。默认情况下转换结果为右对齐，但可通过设置 ADMUX 寄存器的 ADLAR 位变为左对齐。

如果设置为转换结果左对齐，且最高只需要 8 位的转换精度，那么只要读取 ADCH 就足够了。否则要先读取 ADCL，再读取 ADCH，以保证数据寄存器中的内容是同一次转换的结果。一旦读取 ADCL 后，数据寄存器 ADCL 和 ADCH 被锁存，读取 ADCH 后转换结果即可再更新到数据寄存器 ADCL 和 ADCH。

ADC 转换结束可以触发中断。即使转换结束发生在读取 ADCL 与 ADCH 之间，中断仍将触发。

启动一次转换

向 ADC 启动转换位 ADSC 位写“1”可以启动单次转换。在转换过程中此位保持为高，直到转换结束后被硬件清零。如果在转换过程中改变了通道，那么 ADC 会在改变通道前完成这一次转换。

ADC 转换有不同的触发源。设置 ADCSRA 寄存器的 ADC 自动触发允许位 ADATE 可以使能自动触发。设置 ADCSRB 寄存器的 ADC 触发选择位 ADTS 可以选择触发源。当所选的触发信号产生上升沿时，ADC 预分频器复位并开始转换。这提供了一个在固定时间间隔下启动转换的方法。转换结束后即使触发信号仍然存在，也不会启动一次新的转换。如果在转换过程中触发信号又产生了一个上升沿，这个上升沿也将被忽略。即使特定的中断被禁止或全局中断使能位为“0”，其中断标志仍将置位。这样可以在不产生中断的情况下触发一次转换。但是为了在下次中断事件发生时触发新的转换，必须将中断标志清零。

使用 ADC 中断标志作为触发源，可以在当前进行的转换结束后即开始下一次 ADC 转换。之后 ADC 便工作于连续转换模式，持续地进行采样并对 ADC 数据寄存器进行更新。第一次转

换是通过往 ADCSRA 寄存器的 ADSC 位写“1”来启动。在此模式下，后续的 ADC 转换不依赖于 ADC 中断标志 ADIF 是否置位。

如果使能了自动触发，置位 ADCSRA 寄存器的 ADSC 将启动单次转换。ADSC 标志还可用来检测转换是否在进行之中。不论转换是如何启动，在转换过程中 ADSC 一直为“1”。

预分频及 ADC 转换时序

在默认条件下，逐次逼近电路需要一个从 300KHz 到 3MHz 的输入时钟以获得最大精度。如果所需的转换精度低于 10 位，那么输入时钟的频率可以高于 3MHz，以达到更高的采样率。

ADC 模块包括一个预分频器，它可以由系统时钟来产生可接受的 ADC 输入时钟。预分频器通过 ADCSRA 寄存器的 ADPS 位进行设置。置位 ADCSRA 寄存器的 ADEN 将使能 ADC，预分频器开始计数。只要 ADEN 位为“1”，预分频器就持续计数，直到 ADEN 被清零。

ADCSRA 寄存器的 ADSC 被置位后，单端转换在下一个 ADC 时钟周期的上升沿开始启动。正常转换需要 15 个 ADC 时钟周期。ADC 使能（ADCSRA 寄存器的 ADEN 置位）后需要 50 个 ADC 输入时钟周期初始化模拟电路，之后才能有效进行第一次转换。

在 ADC 转换过程中，采样保持在转换启动之后的 1.5 个 ADC 输入时钟开始，而第一次 ADC 转换的结果输出则发生在启动之后的 14.5 个 ADC 输入时钟。转换结束后，ADC 结果被送入 ADC 数据寄存器，且 ADIF 标志位被置位。ADSC 同时被清零。之后软件可以再次置位 ADSC 标志或自动触发，从而启动一次新的转换。

改变通道或基准源

ADMUX 寄存器中的 MUX 及 REFS 通过临时寄存器实现了单缓冲。CPU 可对临时寄存器进行随机访问。在转换启动之前，CPU 可随时对通道及基准源的选择进行配置。为了保证 ADC 有充足的采样时间，一旦转换开始后，就不允许通道及基准源选择的配置。在转换完成（ADCSRA 寄存器的 ADIF 置位）之后，通道及基准源的选择才会被更新。转换的开始时刻为 ADSC 置位后的下一个 ADC 输入时钟的上升沿。因此，建议用户在置位 ADSC 之后的一个 ADC 输入时钟周期内，不要操作 ADMUX 以选择新的通道及基准源。

使用自动触发时，触发事件发生的时间是不确定的。为了控制新设置对转换的影响，在更新 ADMUX 寄存器时要特别小心。若 ADATE 及 ADEN 都置位，则中断时间可以在任意时刻发生，从而自动触发，启动 ADC 的转换。如果在此期间改变 ADMUX 寄存器的内容，那么用户就无法辨别下一次转换是基于旧的配置还是新的配置。建议用户在以下安全时刻对 ADMUX 进行更新：

1） ADATE 或 ADEN 位为“0”；
2） 在转换过程中，但是在触发事件发生后至少一个 ADC 输入时钟周期；
3） 转换结束之后，但是在触发源的中断标志清零之前。

如果在上面所提到的任一种情况下更新 ADMUX，那么新配置将在下一次转换前生效。

选择 ADC 输入通道时须注意，在启动转换之前先选定通道，在 ADSC 置位后的一个 ADC 输入时钟周期之后就可以选择新的模拟输入通道，但最简单的办法是等到转换结束之后再改变

通道。

ADC 的参考电压源 V_{ref} 反映了 ADC 的转换范围。若单端通道电平超过了 V_{ref}，其转换结果将接近最大值 0x3FF。V_{ref} 可以是 AVCC，外接 AREF 引脚的电压，内部 1.25V 或 2.56V 基准电压源。如果使用了差分通道，V_{ref} 需大于 1.5V，即可以是 AVCC，外接 AREF 引脚的电压（大于 1.5V），内部 2.56V 基准电压源。

温度测量

ADC 内部有温度传感器，可用来测量温度范围为-40℃到 125℃。测量的过程如下：

1） 设置 CHMUX = 0x18 选择 ADC 输入通道为温度传感器；
2） 设置 ICTL = 0，开启 ADC 转换，等待 ADC 转换结束后读取 ADC 的值，记为 Data0；
3） 设置 ICTL = 1，开启 ADC 转换，等待 ADC 转换结束后读取 ADC 的值，记为 Data1；
4） 计算 Data1 – Data0，记为 Delta，通过温度查找表找到对应的温度值 Temp。

须注意一下几点：

1） 测量温度时建议采用较低的 ADC 输入时钟如 50KHz 左右；
2） 为保证温度传感器已工作稳定，每次转换结束后多等待一段时间（至少为 4 次转换所需的时间）再读取 ADC 的数据；
3） 建议多次测量 Data0 和 Data1 的值，取其平均值用于计算 delta 值。

温度查找表简表如下所示，详表请见附录。

温度查找表简表

Data0	Data1	Delta	Temperature
688	889	201	0℃
649	857	208	10℃
609	824	215	20℃
570	792	222	30℃
530	759	229	40℃
490	726	236	50℃
450	694	244	60℃
410	661	251	70℃
370	628	258	80℃
329	595	266	90℃
289	562	273	100℃

寄存器定义

ADC 寄存器列表

寄存器	地址	默认值	描述
ADCL	0x78	0x00	ADC 数据低字节寄存器
ADCH	0x79	0x00	ADC 数据高字节寄存器
ADCSRA	0x7A	0x00	ADC 控制和状态寄存器 A
ADCSRB	0x7B	0x00	ADC 控制和状态寄存器 B
ADMUX	0x7C	0x00	ADC 多路选择控制寄存器
ADTMR	0x7D	0x01	ADC 模式控制寄存器
DIDR0	0x7E	0x00	数字输入禁止控制寄存器 0

ADCL – ADC 数据低字节寄存器

ADCL – ADC 数据低字节寄存器								
地址: 0x78					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Name1	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	描述
7:0	ADC[7:0]/ ADC[1:0]	ADC 数据低字节寄存器。 当 ADLAR 位为“0”时，ADC 输出数据在寄存器中的存放按低位对齐，即 ADCL 为 ADC[7:0]，如 Name0 所示；当 ADLAR 位为“1”时，ADC 输出数据在寄存器中的存放按高位对齐，即 ADCL 的高两位为 ADC[1:0]，低六位无意义，如 Name1 所示。

ADCH – ADC 数据高字节寄存器

ADCH – ADC 数据高字节寄存器								
地址: 0x79					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name0	-	-	-	-	-	-	ADC9	ADC8
Name1	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
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	描述
7:0	ADC[9:8]/ ADC[9:2]	ADC 数据低字节寄存器。 当 ADLAR 位为“0”时，ADC 输出数据在寄存器中的存放按低位对齐，即 ADCH

		的低两位为 ADC[9:8]，高六位无意义，如 Name0 所示；当 ADLAR 位为“1”时，ADC 输出数据在寄存器中的存放按高位对齐，即 ADCH 为 ADC[9:2]，如 Name1 所示。

ADCSRA – ADC 控制和状态寄存器 A

ADCSRA – ADC 控制和状态寄存器 A								
地址: 0x7A					默认值: 0x00			
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	0	0

Bit	Name	描述
7	ADEN	ADC 使能控制位。 当设置 ADEN 位为“1”时，ADC 被使能。 当设置 ADEN 位为“0”时，ADC 被禁止。
6	ADSC	ADC 开始转换。 在单次转换模式下，ADSC 置位将启动一次转换。在连续转换模式下，ADSC 置位将启动首次转换。
5	ADATE	ADC 自动触发使能控制位。 当设置 ADATE 位为“1”时，自动触发功能被使能。所选中触发信号的上升沿开启一次转换。触发源的选择由 ADCSRB 寄存器的 ADTS 来控制。 当设置 ADATE 位为“0”时，自动触发功能被禁止。
4	ADIF	ADC 中断标志位。 当 ADC 完成一次转换并更新数据寄存器后置位 ADIF。若 ADC 中断使能位 ADIE 为“1”且全局中断置位，ADC 中断产生。执行 ADC 中断会清零 ADIF 位，也可对该位写“1”来清零。
3	ADIE	ADC 中断使能控制位。 当设置 ADIE 位为“1”且全局中断置位时，ADC 中断被使能。 当设置 ADIE 位为“0”时，ADC 中断被禁止。
2:0	ADPS[2:0]	ADC 预分频器选择控制位。 ADPS 选择系统时钟产生 ADC 时钟的预分频因子。

ADPS[2:0]	预分频因子
0	2
1	2
2	4
3	8
4	16
5	32
6	64
7	128

ADCSRB – ADC 控制和状态寄存器 B

ADCSRB – ADC 控制和状态寄存器 B								
地址: 0x7B					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	-	ACME	-	ICTL	-	ADTS2	ADTS1	ADTS0
R/W	-	R/W	-	R/W	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

Bit	Name	描述
7	-	保留。
6	ACME	模拟比较器多路复用器使能控制位。 当设置 ACME 位为“1”且 ADC 被关掉（ADCSRA 寄存器的 ADEN 位为“0”）时，模拟比较器的负极输入由 ADC 的多路复用器来选择。 当设置 ACME 位为“0”时，模拟比较器的负极输入来自于 AIN1 引脚。
5	-	保留。
4	ICTL	温度传感器模式选择控制位。 设置 ICTL 位为“0”或“1”来选择温度传感器的不同工作模式，用不同模式下所测得的值来计算温度，详见温度测量章节。
3	-	保留。
2:0	ADTS[2:0]	ADC 自动触发源选择控制位。 当设置 ADATE 位为“1”时，自动触发功能被使能，触发源的选择由 ADTS 来控制。 当设置 ADATE 位为“0”时，ADTS 的设置无效。所选中触发信号中断标志的上升沿开启一次转换。当从一个中断标志清零的触发源切换到中断标志置位的触发源会使触发信号产生一个上升沿，如果此时 ADEN 置位，ADC 也会开启一次转换。 当切换到连续转换模式（ADTS=0）时，自动触发功能被禁止。

ADTS[2:0]	触发源
0	连续转换模式
1	模拟比较器
2	外部中断 0
3	定时计数器 0 比较匹配
4	定时计数器 0 溢出
5	定时计数器 1 比较匹配 B
6	定时计数器 1 溢出
7	定时计数器 1 输入捕捉事件

ADMUX – ADC 多路选择控制寄存器

ADMUX – ADC 多路选择控制寄存器								
地址: 0x7C					默认值: 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	描述
7:6	REFS[1:0]	参考电压选择控制位。 通过设置 REFS 控制位来选择参考电压，若在转换过程中改变 REFS 的设置，只有等到当前的转换结束之后改变才会起作用。

REFS[1:0]	参考电压选择
0	AREF
1	AVCC
2	片内 1.25V 基准电压源
3	片内 2.56V 基准电压源

Bit	Name	描述
5	ADLAR	转换结果左对齐使能控制位。 当设置 ADLAR 位为“1”时，转换结果在 ADC 数据寄存器中为左对齐。 当设置 ADLAR 位为“0”时，转换结果在 ADC 数据寄存器中为右对齐。
4:0	CHMUX[4:0]	ADC 输入源选择控制位。

CHMUX[4:0]	单端输入源	差分正端	差分负端
0	PC0	N/A	
1	PC1		
2	PC2		
3	PC3		
4	PC4		
5	PC5		
6	PE1		
7	PE3		
8	N/A	ADC3	ADC0
9		ADC4	ADC1
10		ADC5	ADC2
11-23	保留。	N/A	
24	片内温度传感器		
25	片内基准电压源		
26	AREF/触摸按键		
27-31	保留。		

ADTMR – ADC 模式控制寄存器

ADTMR – ADC 模式控制寄存器								
地址: 0x7D				默认值: 0x01				
Bit	7	6	5	4	3	2	1	0

Name	GAIN1	GAIN0	-	-	-	ADTM2	ADTM1	ADTM0
R/W	R/W	R/W	-	-	-	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	1

Bit	Name	描述
7:6	GAIN[1:0]	ADC 差分放大增益选择控制位。 当 ADC 工作于差分输入通道时，放大器的增益系数由 GAIN 位来选择。

GAIN[1:0]	增益系数
0	7.5
1	15
2	22.5
3	30

5:3	-	保留。
2:0	ADTM[2:0]	ADC 测试模式选择控制位。 用户设置 ADTM 位为“001”即可，其它设置将使 ADC 工作不正常。

DIDR0 – 数字输入禁止控制寄存器 0

DIDR0 – 数字输入禁止控制寄存器 0	
地址: 0x7E	默认值: 0x00

Bit	7	6	5	4	3	2	1	0
Name	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
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	描述
7:0	ADCD[7:0]	数字输入禁止控制位。 当设置 ADCxD 位为“1”时，引脚 ADCx 的数字输入端被禁止，并一直为零。当使能模拟比较器时，ADCx 的数字输入端功能不需要，因此须置位 ADCxD。 当设置 ADCxD 位为“0”时，引脚 ADCx 的数字输入端被使能，引脚上的信号可输入到内部数字逻辑，此时须清零 ADEN 位，及关闭模拟比较器。

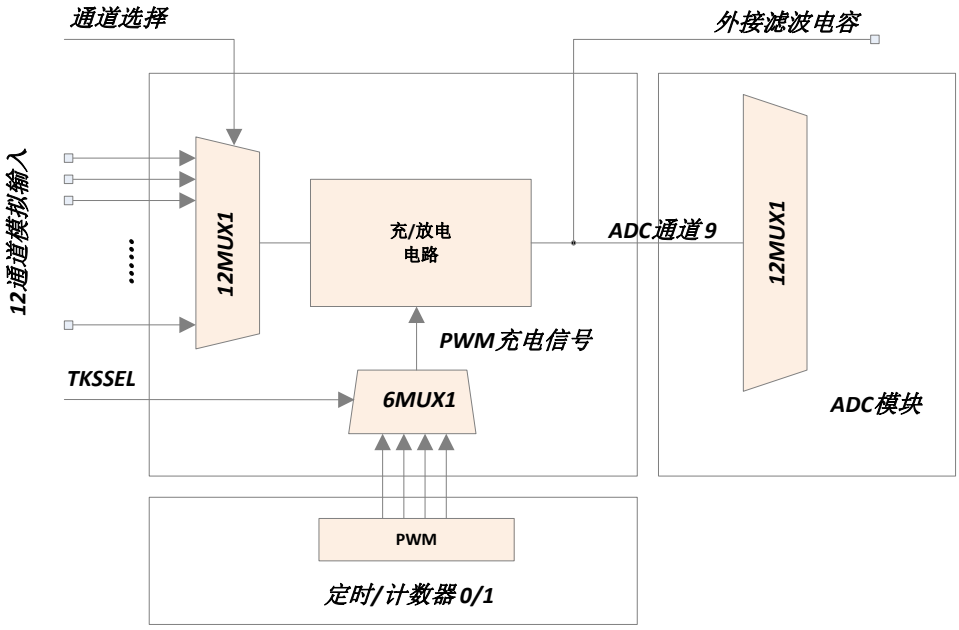
电容触摸按键控制器

- 支持掉电模式
- 最多支持 12 通道触摸按键输入
- 内部独立 ADC 通道
- 内外部滤波设计，抗干扰能力强

概述

本章节主要描述 LGT8F88A 内部集成电容触摸按键控制器模块。电容触摸按键以成本低廉，耐用度高等优点，在消费类产品特别是家电类成品中被广泛使用。电容按键方案中最重要的就是其抗干扰能力和在不同环境下的适应能力。LGT8F88A 内部集成的触摸电容模块是在大量实际应用中总结的最优设计。在不增加系统设计难度的前提下，提高系统的抗干扰能力。能够在电磁炉类似高干扰的产品中正常工作。

LGT8F88A 内部电容触摸按键控制器结构图：



触摸按键输入 I/O

LGT8F88A 最多支持 12 路触摸按键输入。其中 4 路与 ADC 的模拟输入通道复用。触摸按键控制电路内部包含一个 12 选 1 的模拟开关，可以通过内部寄存器选择当前工作通道。

LGT8F88A 的模拟 I/O，都同时具备数字 I/O 功能。系统上电后，这些 I/O 都默认为数字 I/O 功能，需要通过 DIDR 寄存器寄存器，禁止数字功能，模拟通道才可以正常工作。LGT8F88A 内部共有三个 DIDR 寄存器，分别为 DIDR0/1/2/3。其中 DIDR0/1 用于控制 ADC 相关的模拟输入通道，另外的 DIDR2/3 用于控制 12 路触摸按键输入(包括与 ADC 复用的最高 4 通道)

下面表格给出了 LGT8F88A 所有 12 通道输入的 I/O 以及对应 DIDR 控制位，DIDR2/3 的定义请参考本章节后面的寄存器描述部分。

DIDR	I/O	功能描述
DIDR2[0]	PD6	触摸按键通道 0
DIDR2[1]	PD7	触摸按键通道 1
DIDR2[2]	PB0	触摸按键通道 2
DIDR2[3]	PB3	触摸按键通道 3
DIDR2[4]	PB4	触摸按键通道 4
DIDR2[5]	PB5	触摸按键通道 5
DIDR2[6]	PE0	触摸按键通道 6
DIDR2[7]	PE2	触摸按键通道 7
DIDR3[0]	PC4	触摸按键通道 8
DIDR3[1]	PC5	触摸按键通道 9
DIDR3[2]	PE1	触摸按键通道 10
DIDR3[3]	PE3	触摸按键通道 11

PWM 充放电控制

电容触摸按键的测量原理，一般都是根据电容的充放电特性测量电容的大小和变化。为了减小设计的复杂性，我们采用一种比较简单实用的充放电方法 – 直接使用内部的 PWM 脉冲对电容进行充电。使用 PWM 脉冲充电同时也可以增强测量的抗干扰能力。

对电容充电的 PWM 脉冲采用 500KHz 频率的方波(50%占空比)，充电的稳定时间因外部按键电容的大小不同而稍有区别。由于实际电路中，因不同的电路设计和 PCB 材料，以及电容按键的面积等因素，需要用户根据实际应用确定电路的固有电容大小，通过判断充电稳定后的电压变化，确定是否为按键动作。此处我们给出一组典型值，供设计参考：

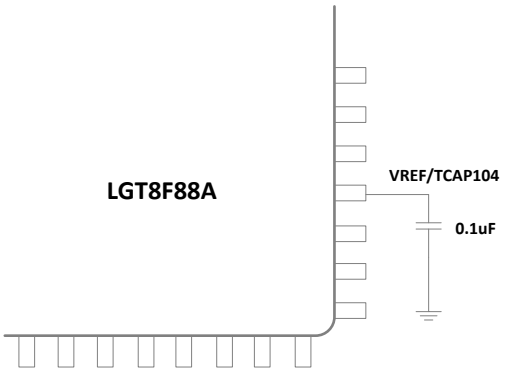
电压 3.3V/温度 25C		
按键电容	充电稳定电压	稳定时间
2pF	1.48V	20ms
7pF	1.11V	25ms
10pF	0.94V	25ms
14pF	0.75V	25ms
20pF	0.56V	35ms
50pF	0.33V	40ms

LGT8F88A 内部的定时/计数器可以输出最多 6 路 PWM。可以任选一路，下表给出 PWM 的配置：

TKSSEL[2:0]	PWM 通道
010	OC0A
011	0C0B
100	0C1A
101	OC1B
110	OC2A
111	OC2B

电容触摸按键滤波设计

在电容测试过程中，很容易会受到周围电磁环境的干扰而导致电容本身发生变化，这样的干扰可以通过软件级别的滤波算法滤除。另外也有一些干扰来自电源，电源上的杂波很容易影响到模拟信号的电压，导致测量结果不准确。为此，LGT8F88A 专门为电容测量电路预留了一个可以接外部滤波电容的引脚。用户可以根据产品的使用环境，选择合适的滤波电容，能够非常有效的滤除来自系统电源上的干扰。



在触摸按键的设计过程中，同样也要采用适当的抗干扰措施。减少环境干扰对电路电容的影响。PCB 触摸按键设计的内容，请参考相关资料。

除了以上硬件上的措施，软件滤波也是必不可少。优秀的软件滤波算法可以决定最终触摸按键方案的品质。我们也准对 LGT8F88A 触摸控制器模块的特性，设计了一套专用的软件库共用户使用，具体请参考相关设计文档或直接与 LGT 技术支持联系。

寄存器描述

触摸按键控制器寄存器- TKCR

TKCR 触摸按键控制寄存器		
TKCR: 0xCD		默认值: 0x80
TKCR	TKCR[7:0]	
R/W	R/W	
Initial	0x80	
位定义		
[3:0]	TKMUX	触摸按键模拟输入选择 0000 : TKEYIN0 -触摸按键通道 0 0001 : TKEYIN1 –触摸按键通道 1 ……. 1010 : TKEYIN10 –触摸按键通道 10 1011 : TKEYIN11 –触摸按键通道 11
[6:4]	TKSSEL	PWM 充电脉冲源选择
[7]	PD	掉电控制，设置 1 进入掉电模式

数字 I/O 输入禁用寄存器- DIDR2

DIDR2 数字输入禁用寄存器 2		
DIDR2: 0xCE		默认值: 0x00
DIDR2	DIDR2[7:0]	
R/W	R/W	
Initial	0x00	
位定义		
[0]	DIDR20	PD6 数字输入禁用，对应触摸输入通道 0
[1]	DIDR21	PD7 数字输入禁用，对应触摸输入通道 1
[2]	DIDR22	PB0 数字输入禁用，对应触摸输入通道 2
[3]	DIDR23	PB3 数字输入禁用，对应触摸输入通道 3
[4]	DIDR24	PB4 数字输入禁用，对应触摸输入通道 4
[5]	DIDR25	PB5 数字输入禁用，对应触摸输入通道 5
[6]	DIDR26	PE0 数字输入禁用，对应触摸输入通道 6
[7]	DIDR27	PE2 数字输入禁用，对应触摸输入通道 7

数字 I/O 输入禁用寄存器- DIDR3

DIDR3 数字输入禁用寄存器 3		
DIDR3: 0xCF		默认值: 0x00
DIDR3	DIDR3[7:0]	
R/W	R/W	
Initial	0x00	
位定义		
[0]	DIDR30	PC4 数字输入禁用，对应触摸输入通道 8
[1]	DIDR31	PC5 数字输入禁用，对应触摸输入通道 9
[2]	DIDR32	PE1 数字输入禁用，对应触摸输入通道 10
[3]	DIDR33	PE3 数字输入禁用，对应触摸输入通道 11

寄存器速查表

Address	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	LFEN	EXTEN	WCES	OSCKEN	OSCMEN	RCKEN	RCMEN
$F1	DSCR	DSCE	-	-	DSC4	DSC3	DSC2	DSC1	DSC0
$F0	IOCR	IOCE	-	-	-	-	-	REFIOEN	RSTIOEN
$E2	PSSR	PSS1	-	-	-	-	-	-	PSR1
$CF	DIDR3	-	-	-	-	TIN11D	TIN10D	TIN9D	TIN8D
$CE	DIDR2	TIN7D	TIN6D	TIN5D	TIN4D	TIN3D	TIN2D	TIN1D	TIN0D
$CD	TKCSR	TKPD	TKPSEL			TKMUX			
$C6	UDR0	USART Data							
$C5	UBRR0H	-	-	-	-	USART Baud Rate Register High			
$C4	UBRR0L	USART Baud Rate Register Low							
$C2	UCSR0C	UMSEL0		UPM0		USBS0	UCSZ01/ UDORD0	UCSZ00/ UCPHA0	UCPOL0
$C1	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
$C0	UCSR0A	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
$B9	TWSR	TWI Status					-	TWPS	
$B8	TWBR	TWI Bit Rate							
$B6	ASSR	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
$B4	OCR2B	Timer/Counter 2 Output Compare Register B							
$B3	OCR2A	Timer/Counter 2 Output Compare Register A							
$B2	TCNT2	Timer/Counter 2 Counter Register							

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
$B1	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS2		
$B0	TCCR2A	COM2A		COM2B		-	-	WGM21	WGM20
$A9	PORTE	Port Output E							
$A8	DDRE	Data Direction E							
$A7	PINE	Port Input E							
$8B	OCR1BH	Timer/Counter 1 Output Compare B High							
$8A	OCR1BL	Timer/Counter 1 Output Compare B Low							
$89	OCR1AH	Timer/Counter 1 Output Compare A High							
$88	OCR1AL	Timer/Counter 1 Output Compare A Low							
$87	ICR1H	Timer/Counter 1 Input Capture High							
$86	ICR1L	Timer/Counter 1 Input Capture Low							
$85	TCNT1H	Timer/Counter 1 Counter High							
$84	TCNT1L	Timer/Counter 1 Counter Low							
$82	TCCR1C	FOC1A	FOC1B	-	-	-	-	-	-
$81	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS1		
$80	TCCR1A	COM1A		COM1B		-	-	WGM11	WGM10
$7F	DIDR1	-	-	-	-	-	-	AIN1D	AIN0D
$7E	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
$7D	ADTMR	GAIN		-	-	-	ADTM		
$7C	ADMUX	REFS		ADLAR	-	MUX			
$7B	ADCSRB	-	ACME	-	ICTL	-	ADTS		
$7A	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS		
$79	ADCH	ADC Data High							
$78	ADCL	ADC Data Low							
$77	EEDRH	EEPROM Data High							
$75	IVBASE	Interrupt Vector Base Address							
$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	-	-	-	-	ISC1		ISC0	

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
$68	PCICR	-	-	-	-	-	PCIE2	PCIE1	PCIE0
$66	OSCCAL	-	-	OSC Calibration					
$65	PRR1	-	-	PRWDT	-	-	PREFL	PRPCI	-
$64	PRR	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	PRUSART0	PRADC
$62	VDTCR	VDTCE	SWRSTN	-	-	-	VDTSEL		VDTEN
$61	CLKPR	CLKPCE	CLKOEN0	CLKOEN1	-	CLKPS			
$60	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
IO Register									
$5F($3F)	SREG	I	T	H	S	V	N	Z	C
$5E($3E)	SPH	Stack Point High							
$5D($3D)	SPL	Stack Point Low							
$55($35)	MCUCR	-	BODS	BODSE	PUD	-	-	IVSEL	IVCE
$54($34)	MCUSR	SWDD	-	-	OCDRF	WDRF	BORF	EXTRF	PORF
$53($33)	SMCR	-	-	-	-	SM			SE
$50($30)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS	
$4E($2E)	SPDR	SPI Data							
$4D($2D)	SPSR	SPIF	WCOL	-	-	-	DUAL	-	SPI2X
$4C($2C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR	
$4B($2B)	GPIOR2	General Purpose IO Register 2							
$4A($2A)	GPIOR1	General Purpose IO Register 1							
$48($28)	OCR0B	Timer/Counter 0 Output Compare Register B							
$47($27)	OCR0A	Timer/Counter 0 Output Compare Register A							
$46($26)	TCNT0	Timer/Counter 0 Counter							
$45($25)	TCCR0B	FOC0A	FOC0B	OC0AS	-	WGM02	CS0		
$44($24)	TCCR0A	COM0A		COM0B		-	-	WGM01	WGM00
$43($23)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC
$42($22)	EEARH	EEPROM Address High							
$41($21)	EEARL	EEPROM Address Low							
$40($20)	EEDR	EEPROM Data Low							
$3F($1F)	EECR	EEPM2	-	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE
$3E($1E)	GPIOR0	General Purpose IO Register 0							
$3D($1D)	EIMSK	-	-	-	-	-	-	INT1	INT0
$3C($1C)	EIFR	-	-	-	-	-	-	INTF1	INTF0
$3B($1B)	PCIFR	-	-	-	-	-	PCIF2	PCIF1	PCIF0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
$37($17)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2
$36($16)	TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1
$35($15)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0
$2B($0B)	PORTD	Port Output D							
$2A($0A)	DDRD	Data Direction D							
$29($09)	PIND	Port Input D							
$28($08)	PORTC	Port Output C							
$27($07)	DDRC	Data Direction C							
$26($06)	PINC	Port Input C							
$25($05)	PORTB	Port Output B							
$24($04)	DDRB	Data Direction B							
$23($03)	PINB	Port Input B							

指令集速查表

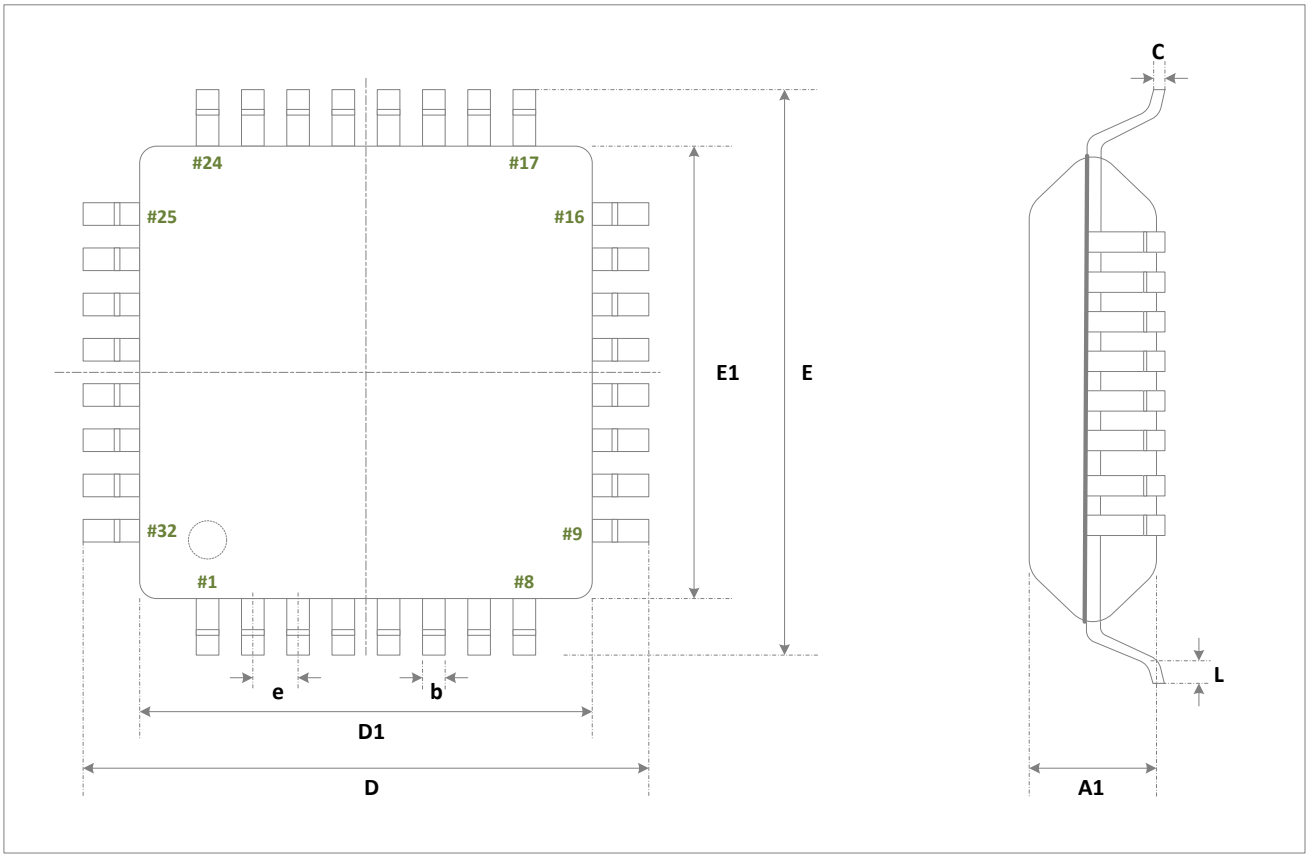
指令	操作数	描述	操作	标记位	周期
算术逻辑运算指令					
ADD	R_d, R_r	寄存器相加	$R_d \leftarrow R_d + R_r$	Z,C,N,V,H	1
ADC	R_d, R_r	带进位的寄存器相加	$R_d \leftarrow R_d + R_r + C$	Z,C,N,V,H	1
ADIW	R_{dl}, K	立即数与字相加	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} + K$	Z,C,N,V,S	1
SUB	R_d, R_r	寄存器相加减	$R_d \leftarrow R_d - R_r$	Z,C,N,V,H	1
SUBI	R_d, K	寄存器减常数	$R_d \leftarrow R_d - K$	Z,C,N,V,H	1
SBC	R_d, R_r	带借位的寄存器相加减	$R_d \leftarrow R_d - R_r - C$	Z,C,N,V,H	1
SBCI	R_d, K	带借位的寄存器减常数	$R_d \leftarrow R_d – K - C$	Z,C,N,V,H	1
SBIW	R_{dl}, K	立即数与字相减	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} - K$	Z,C,N,V,S	1
AND	R_d, R_r	逻辑与	$R_d \leftarrow R_d \& R_r$	Z,N,V	1
ANDI	R_d, K	寄存器逻辑与常数	$R_d \leftarrow R_d \& K$	Z,N,V	1
OR	R_d, R_r	逻辑或	$R_d \leftarrow R_d \mid R_r$	Z,N,V	1
ORI	R_d, K	寄存器逻辑或常数	$R_d \leftarrow R_d \mid K$	Z,N,V	1
EOR	R_d, R_r	寄存器异或	$R_d \leftarrow R_d \oplus R_r$	Z,N,V	1
COM	R_d	反码	$R_d \leftarrow \$FF - R_d$	Z,C,N,V	1
NEG	R_d	2 禁制补码	$R_d \leftarrow \$00 - R_d$	Z,C,N,V,H	1
SBR	R_d, K	设置寄存器中的位	$R_d \leftarrow R_d$ v K	Z,N,V	1
CBR	R_d, K	清寄存器中的位	$R_d \leftarrow R_d$ v ($FF – K)	Z,N,V	1
INC	R_d	递增	$R_d \leftarrow R_d + 1$	Z,N,V	1
DEC	R_d	递减	$R_d \leftarrow R_d - 1$	Z,N,V	1
TST	R_d	测试为 0 或负数	$R_d \leftarrow R_d \& R_d$	Z,N,V	1
CLR	R_d	清寄存器	$R_d \leftarrow R_d \oplus R_d$	Z,N,V	1
SER	R_d	寄存器全设置为 1	$R_d \leftarrow \$FF$	None	1
MUL	R_d, R_r	无符号乘法	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULS	R_d, R_r	有符号乘法	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULSU	R_d, R_r	有符号数乘无符号数	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
FMUL	R_d, R_r	无符号乘法，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULS	R_d, R_r	有符号乘法，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULSU	R_d, R_r	有符号数乘无符号数，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
跳转指令					
RJMP	K	相对跳转	PC ← PC + K + 1	None	1
IJMP		间接跳转（到 Z 指向地址）	PC ← Z	None	2
JMP	K	直接跳转	PC ← K	None	2
RCALL	K	相对地址子程序调用	PC ← PC + K + 1	None	1
ICALL		间接子程序调用（Z 指向地址）	PC ← Z	None	2
CALL	K	直接子程序调用	PC ← K	None	2
RET		子程序返回	PC ← Stack	None	2
RETI		中断返回	PC ← Stack	I	2

指令	操作数	描述	操作	标记位	周期
跳转指令（续）					
CPSE	R_d, R_r	相等即跳转	If(R_d=R_r) PC ← PC + 2 or 3	None	1/2
CP	R_d, R_r	比较	R_d - R_r	Z,N,V,C,H	1
CPC	R_d, R_r	带进位比较	R_d - R_r – C	Z,N,V,C,H	1
CPI	R_d, K	与立即数比较	R_d - K	Z,N,V,C,H	1
SBRC	R_r, b	位为 0 即跳过下一条指令	If(R_r(b)=0) PC ← PC + 2 or 3	None	1/2
SBRS	R_r, b	位为 1 即跳过下一条指令	If(R_r(b)=1) PC ← PC + 2 or 3	None	1/2
SBIC	P, b	I/O 位为 0 即跳过下一条指令	If(P(b)=0) PC ← PC + 2 or 3	None	1/2
SBIS	P, b	I/O 位为 1 即跳过下一条指令	If(P(b)=1) PC ← PC + 2 or 3	None	1/2
BRBS	s, k	状态标记为 1 即跳转	If(SREG(S)=1) PC ← PC + K + 1	None	1/2
BRBC	s, k	状态标记为 0 即跳转	If(SREG(S)=0) PC ← PC + K + 1	None	1/2
BREQ	k	相等即跳转	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	不等即跳转	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	进位则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	无进位则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	不小于则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	小于则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	为负则跳转	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	为正则跳转	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	有符号的不小于即跳转	if (N⊕V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	有符号的小于 0 即跳转	if (N⊕V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	半进位为 1 则跳转	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	半进位为 0 则跳转	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	T 置位则跳转	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	T 清零则跳转	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	溢出则跳转	f (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	不溢出则跳转	f (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	全局中断使能则跳转	f (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	全局中断禁止则跳转	f (I = 0) then PC ← PC + k + 1	None	1/2
数据传输指令					
MOV	Rd, Rr	寄存器之间移动数据	Rd ← Rr	None	1
MOVW	Rd, Rr	移动一个字的数据	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	加载立即数	Rd ← K	None	1
LD	Rd, X	间接加载	Rd ← (X)	None	1
LD	Rd, X+	间接加载，地址递增	Rd ← (X), X ← X + 1	None	1
LD	Rd, -X	地址递减，间接加载	X ← X - 1, Rd ← (X)	None	1
LD	Rd, Y	间接加载	Rd ← (Y)	None	1
LD	Rd, Y+	间接加载，地址递增	Rd ← (Y), Y ← Y + 1	None	1
LD	Rd, -Y	地址递减，间接加载	Y ← Y - 1, Rd ← (Y)	None	1
LDD	Rd, Y+q	带偏移量的间接加载	Rd ← (Y + q)	None	1

LD	Rd, Z	间接加载	Rd ← (Z)	None	1
LD	Rd, Z+	间接加载，地址递增	Rd ← (Z), Z ← Z+1	None	1
LD	Rd, -Z	地址递减，间接加载	Z ← Z - 1, Rd ← (Z)	None	1
LDD	Rd, Z+q	带偏移量的间接加载	Rd ← (Z + q)	None	1
LDS	Rd, k	直接从 SRAM 中加载	Rd ← (k)	None	2
ST	X, Rr	间接存储	(X) ← Rr	None	1
ST	X+, Rr	间接存储，地址递增	(X) ← Rr, X ← X + 1	None	1
ST	-X, Rr	地址递减，间接存储	X ← X - 1, (X) ← Rr	None	1
ST	Y, Rr	间接存储	(Y) ← Rr	None	1
ST	Y+, Rr	间接存储，地址递增	(Y) ← Rr, Y ← Y + 1	None	1
ST	-Y, Rr	地址递减，间接存储	Y ← Y - 1, (Y) ← Rr	None	1
STD	Y+q, Rr	带偏移量的间接存储	(Y + q) ← Rr	None	1
ST	Z, Rr	间接存储	(Z) ← Rr	None	1
ST	Z+, Rr	间接存储，地址递增	(Z) ← Rr, Z ← Z + 1	None	1
ST	-Z, Rr	地址递减，间接存储	Z ← Z - 1, (Z) ← Rr	None	1
STD	Z+q, Rr	带偏移量的间接存储	(Z + q) ← Rr	None	1
STS	k, Rr	直接存储到 SRAM 中	(k) ← Rr	None	2
LPM		加载程序空间数据	R0 ← (Z)	None	2
LPM	Rd, Z	加载程序空间数据	Rd ← (Z)	None	2
LPM	Rd, Z+	加载程序数据，地址递增	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, Z+	间接加载，地址递增	Rd ← (Z), Z ← Z+1	None	1
LD	Rd, -Z	地址递减，间接加载	Z ← Z - 1, Rd ← (Z)	None	1
LDD	Rd, Z+q	带偏移量的间接加载	Rd ← (Z + q)	None	1
LDS	Rd, k	直接从 SRAM 中加载	Rd ← (k)	None	2
IN	Rd, P	读端口	Rd ← P	None	1
OUT	P, Rr	写端口	P ← Rr	None	1
PUSH	Rr	压栈	STACK ← Rr	None	1
POP	Rd	出栈	Rd ← STACK	None	1
SBI	P, b	设置 IO 寄存器	I/O(P, b) ← 1	None	1
CBI	P, b	清零 IO 寄存器	I/O(P, b) ← 0	None	1
LSL	Rd	逻辑左移	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	逻辑右移	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z	1
ROL	Rd	包含进位的循环左移	Rd(0)←C, Rd(n+1) ← Rd(n), C←Rd(7)	Z	1
ROR	Rd	包含进位的循环右移	Rd(7)←C, Rd(n) ← Rd(n+1), C←Rd(0)	Z	1
ASR	Rd	算术右移	Rd(n) ← Rd(n+1), n=0:6	Z	1
SWAP	Rd	位交换	Rd(3:0) ← Rd(7:4), Rd(7:4) ← Rd(3:0)	None	1
BSET	s	设置状态位	SREG(s) ← 1	SREG(s)	1
BCLR	s	清零状态位	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	存储到 T 位	T ← Rr(b)	T	1
BLD	Rd, b	读出 T 位到寄存器	Rd(b) ← T	None	1
SEC		设置进位标志	C ← 1	C	1

CLC		清楚进位标志	C ← 0	C	1
SEN		设置负数标志	N ← 1	N	1
CLN		清除负数标志	N ← 0	N	1
SEZ		设置零标志	Z ← 1	Z	1
CLZ		清除零标志	Z ← 0	Z	1
SEI		使能全局中断	I ← 1	I	1
CLI		禁制全局中断	I ← 0	I	1
SES		设置符号测试标志	S ← 1	S	1
CLS		清除符号测试标志	S ← 0	S	1
SEV		设置二进制补码溢出标志	V ← 1	V	1
CLV		清除二进制补码溢出标志	V ← 0	V	1
SET		设置 T 位（SREG）	T ← 1	T	1
CLT		清除 T 位（SREG）	T ← 0	T	1
MCU 控制指令					
NOP		空指令		None	1
SLEEP		进入休眠模式		None	1
WDR		看门狗复位		None	1
BREAK		软断点	仅用于调试目的	None	N/A
NOP		空指令		None	1
SLEEP		进入休眠模式		None	1

封装参数



LQFP32L 通用尺寸定义

字符代号	最小值	典型值	最大值	单位
D	8.90	9.00	9.10	毫米
D1	6.90	7.00	7.10	毫米
b	0.15	0.20	0.25	毫米
e	0.75	0.80	0.85	毫米
E	8.90	9.00	9.10	毫米
E1	6.90	7.00	7.10	毫米
C	-	0.10	-	毫米
L	0.55	0.60	0.65	毫米
A1	-	1.40	-	毫米