

**IRC12x**  
**1KW OTP type**  
**IR Remote Control MCU**  
***Datasheet***

***Version 0.02***

***Oct. 15, 2019***

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### Table of content

<b>1. Features</b>	<b>6</b>
1.1. Special Features	6
1.2. System Features	6
1.3. CPU Features	6
1.4. Package Information	6
<b>2. General Description and Block Diagram</b>	<b>7</b>
<b>3. Pin Assignment and Description</b>	<b>8</b>
<b>4. Device Characteristics</b>	<b>11</b>
4.1. AC/DC Device Characteristics	11
4.2. Absolute Maximum Ratings	12
4.3. Typical IHRC frequency vs. VDD (calibrated to 16MHz)	12
4.4. Typical ILRC frequency vs. VDD	13
4.5. Typical IHRC Frequency vs. Temperature (calibrated to 16MHz)	13
4.6. Typical ILRC Frequency vs. Temperature	14
4.7. Typical operating current vs. VDD @ system clock = IHRC/n	14
4.8. Typical operating current vs. VDD @ system clock = ILRC/n	15
4.9. Typical IO pull high resistance	15
4.10. Typical IO driving current ( $I_{OH}$ ) and sink current ( $I_{OL}$ )	16
4.11. Typical IO input high/low threshold voltage ( $V_{IH}/V_{IL}$ )	17
4.12. Typical power down current ( $I_{PD}$ ) and power save current ( $I_{PS}$ )	18
<b>5. Functional Description</b>	<b>19</b>
5.1. Program Memory – OTP	19
5.2. Boot Procedure	19
5.3. Data Memory – SRAM	20
5.4. Oscillator and clock	20
5.4.1. Internal High RC oscillator and Internal Low RC oscillator	20
5.4.2. IHRC calibration	21
5.4.3. IHRC Frequency Calibration and System Clock	21
5.4.4. System Clock	22
5.4.5. System Clock Switching	23
5.5. 16-bit Timer (Timer16)	24

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

5.6.	WatchDog Timer .....	25
5.7.	Interrupt.....	26
5.8.	Power-Save and Power-Down .....	28
5.8.1.	Power-Save mode (“stopexe”) .....	28
5.8.2.	Power-Down mode (“stopsys”).....	29
5.8.3.	Wake-up .....	29
5.9.	T-type Key Scan Wake-up.....	30
5.10.	IO Pins .....	31
5.11.	Reset .....	32
5.12.	8-bits IR Carrier Frequency Generator (IRTMC / IRTMB) .....	33
<b>6.</b>	<b>IO Registers.....</b>	<b>35</b>
6.1.	ACC Status Flag Register ( <i>flag</i> ), IO address =0’h00 .....	35
6.2.	Stack Pointer Register ( <i>sp</i> ), IO address =0x02 .....	35
6.3.	Clock Mode Register ( <i>clkmd</i> ), IO address =0x03.....	35
6.4.	Interrupt Enable Register ( <i>inten</i> ), IO address =0x04 .....	35
6.5.	Interrupt Request Register ( <i>intrq</i> ), IO address =0x05.....	36
6.6.	Timer16 mode Register ( <i>t16m</i> ), IO address =0x06 .....	36
6.7.	MISC Register ( <i>misc</i> ), IO address = 0x1b.....	37
6.8.	Interrupt Edge Select Register ( <i>integs</i> ), IO address =0x0c .....	37
6.9.	Port A Digital Input Enable Register ( <i>padier</i> ), IO address =0x0d.....	37
6.10.	Port B Digital Input Enable Register ( <i>pbdier</i> ), IO address =0x0e.....	38
6.11.	Port A Data Register ( <i>pa</i> ), IO address =0x10.....	38
6.12.	Port A Control Register ( <i>pac</i> ), IO address =0x11 .....	38
6.13.	Port A Pull-High Register ( <i>paph</i> ), IO address =0x12 .....	38
6.14.	Port B Data Register ( <i>pb</i> ), IO address =0x14.....	38
6.15.	Port B Control Register ( <i>pbc</i> ), IO address =0x15.....	39
6.16.	Port B Pull-High Register ( <i>pbph</i> ), IO address =0x16.....	39
6.17.	Port A T-scan Select Register ( <i>pats</i> ), IO address =0x13.....	39
6.18.	Port B T-scan Select Register ( <i>pbts</i> ), IO address =0x17 .....	39
6.19.	T-scan Control Register ( <i>tscnc</i> ), IO address =0x1E .....	39
6.20.	IR Timer Control Register ( <i>irtmc</i> ), IO address =0x1C.....	40
6.21.	IR Timer Bound Register ( <i>irtmb</i> ), IO address =0x1D.....	40
<b>7.</b>	<b>Instructions .....</b>	<b>41</b>
7.1.	Data Transfer Instructions .....	42
7.2.	Arithmetic Operation Instructions .....	44

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

7.3.	Shift Operation Instructions .....	47
7.4.	Logic Operation Instructions.....	48
7.5.	Bit Operation Instructions .....	50
7.6.	Conditional Operation Instructions.....	50
7.7.	System control Instructions .....	51
7.8.	Summary of Instructions Execution Cycle .....	53
7.9.	Summary of affected flags by Instructions .....	53
7.10.	BIT definition .....	53
<b>8.</b>	<b>Code Options .....</b>	<b>54</b>
<b>9.</b>	<b>Special Notes .....</b>	<b>54</b>
9.1.	Using IC .....	54
9.1.1.	IO pin usage and setting .....	54
9.1.2.	Interrupt .....	55
9.1.3.	System clock switching .....	55
9.1.4.	Power-Down mode, Wake-up and Watchdog .....	55
9.1.5.	TIMER time out.....	56
9.1.6.	IHRC.....	56
9.1.7.	Program writing of IRC12x.....	56
9.1.8.	Application Schematic for IRC12x.....	57
9.2.	Using ICE.....	59

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### Revision History:

Revision	Date	Description
0.00	2019/06/11	Preliminary version
0.01	2019/07/01	1. Delete some sensitive words and pictures 2. Delete the Important Notice 3. Delete the original section 9.1 4. Amend section 9.1.6 5. Amend section 9.1.7 6. Amend section 9.1.8
0.02	2019/10/15	1. Amend Section 4.1 2. Amend Section 6.14, 6.15 3. Amend Section 9.1.8 4. Amend Section 9.2

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 1. Features

#### 1.1. Special Features

- ◆ General purpose series
- ◆ Please don't apply to AC RC step-down powered, high power ripple or high EFT requirement application
- ◆ Operating temperature range: -20°C ~ 70°C

#### 1.2. System Features

- ◆ 1KW OTP program memory
- ◆ 32 Bytes data SRAM
- ◆ One hardware 16-bit timer
- ◆ One hardware Key Scan wake-up machine
- ◆ One IR Carrier frequency generator
- ◆ One dedicated REM pin supports IRTX large sink current output
- ◆ 13 IO pins with optional pull-high resistor
- ◆ Clock sources: Internal high RC oscillator and internal low RC oscillator

#### 1.3. CPU Features

- ◆ One processing unit operating mode
- ◆ 79 Powerful instructions
- ◆ Most instructions are 1T execution cycle
- ◆ Programmable stack pointer and stack level (Use 2 bytes SRAM for one level stack)
- ◆ Direct and indirect addressing modes for data access. Data memories are available for use as an index pointer of Indirect addressing mode
- ◆ IO space and memory space are independent

#### 1.4. Package Information

- ◆ IRC12x-S08B: SOP8 (150mil)
- ◆ IRC12x-S16D: SOP16D (150mil)

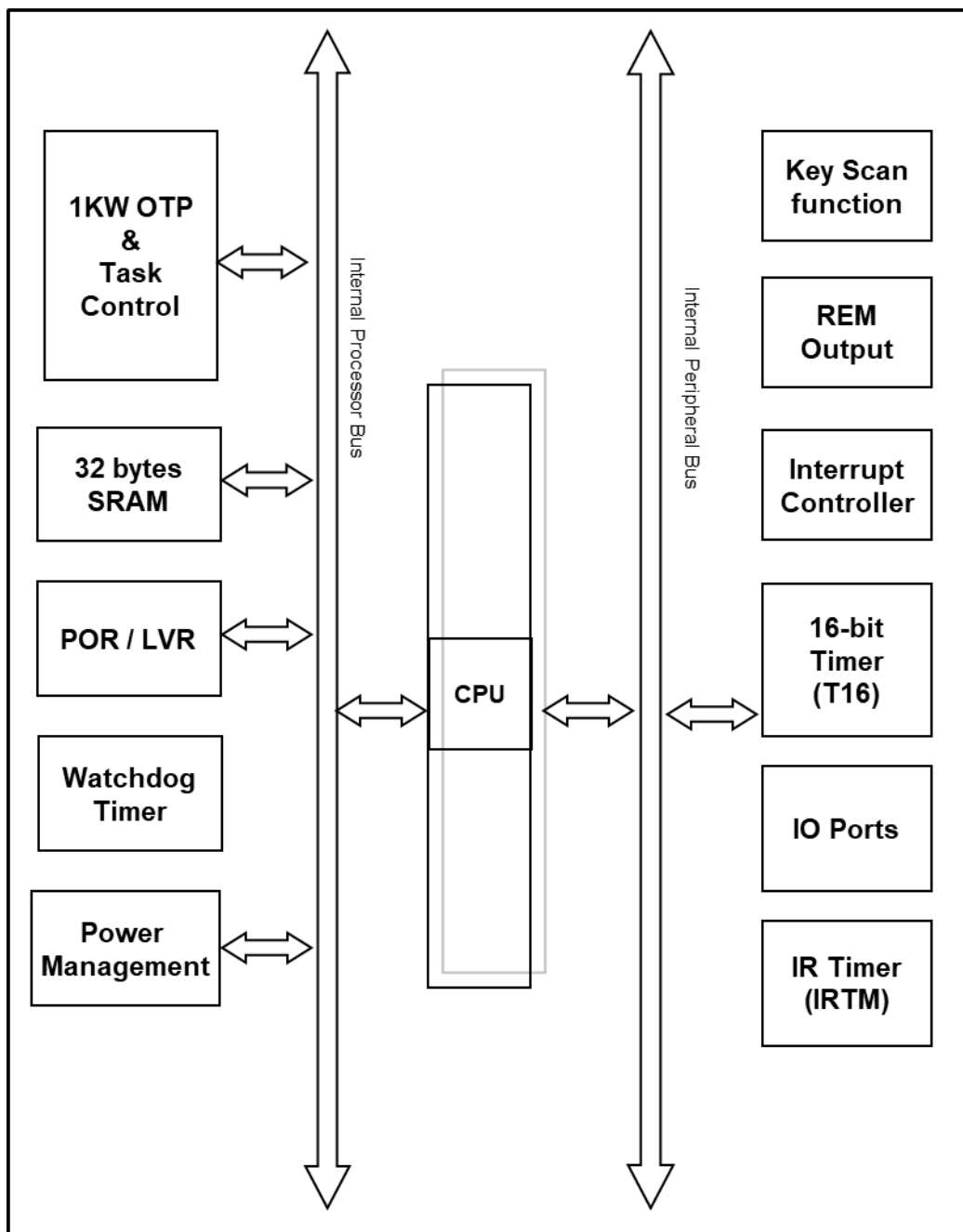
# IRC12x

## 1KW OTP type IR Remote Control MCU

### 2. General Description and Block Diagram

The IRC12x is a fully static, OTP-based CMOS 8-bit microcontroller; it employ RISC architecture and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access. 1KW bits OTP program memory, 32 bytes data SRAM and one hardware 16-bit timer are inside.

IRC12x is best suit for remote controller applications; it has a dedicated REM output for IRTX output. An 8-bit IR timer is also supported for IR Carrier frequency generator. In addition, it also supports both T-type and Matrix key scan wake-up.

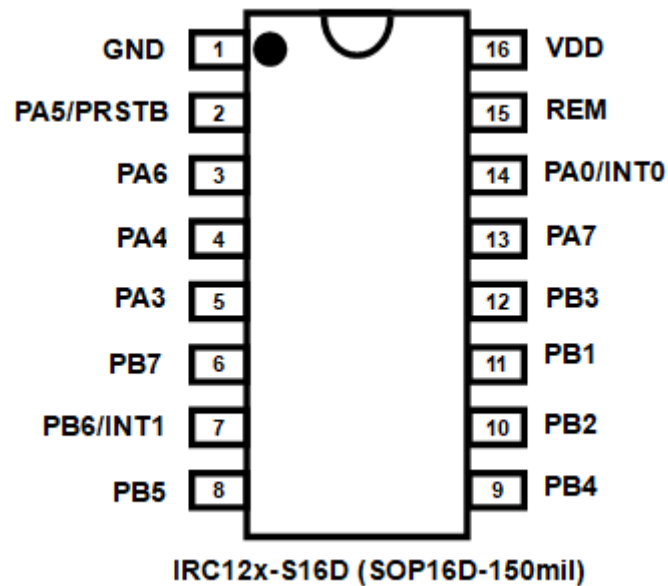
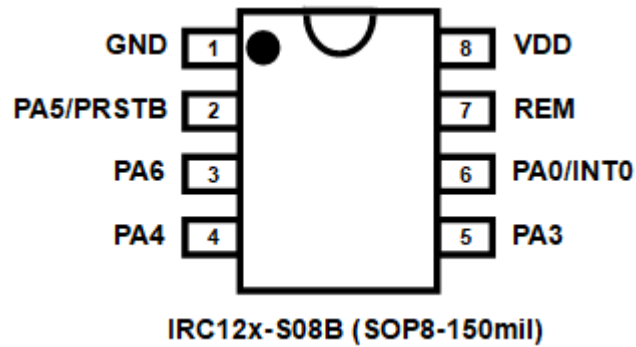


# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 3. Pin Assignment and Description





# IRC12x

## 1KW OTP type IR Remote Control MCU

---

Pin Name	Pin Type & Buffer Type	Description
PA7	IO ST / CMOS	The functions of this pin can be: Bit 7 of port A. It can be configured as input or output with pull-high resistor.
PA6	IO ST / CMOS	The functions of this pin can be: Bit 6 of port A. It can be configured as input or output with pull-high resistor.
PA5 / PRSTB	IO ST / CMOS	The functions of this pin can be: (1) Bit 5 of port A. It can be configured as input or open-drain output, with pull-up resistor. (2) Hardware reset
PA4	IO ST / CMOS	The functions of this pin can be: Bit 4 of port A. It can be configured as input or output with pull-high resistor.
PA3	IO ST / CMOS	The functions of this pin can be: Bit 3 of port A. It can be configured as input or output with pull-high resistor.
PA0 / INT0	IO ST / CMOS	The functions of this pin can be: (1) Bit 0 of port A. It can be configured as input or output with pull-high resistor. (2) External interrupt line 0. <u>Both rising edge and falling edge are accepted to request interrupt service.</u>
REM	Output	IRTX output to provide sink current for IR LED.
PB1	IO ST / CMOS	The functions of this pin can be: Bit 1 of port B. It can be configured as input or output with pull-high resistor.
PB2	IO ST / CMOS	The functions of this pin can be: Bit 2 of port B. It can be configured as input or output with pull-high resistor.
PB3	IO ST / CMOS	The functions of this pin can be: Bit 3 of port B. It can be configured as input or output with pull-high resistor.
PB4	IO ST / CMOS	The functions of this pin can be: Bit 4 of port B. It can be configured as input or output with pull-high resistor.
PB5	IO ST / CMOS	The functions of this pin can be: Bit 5 of port B. It can be configured as input or output with pull-high resistor.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

Pin Name	Pin Type & Buffer Type	Description
PB6 / INT1	IO ST / CMOS	The functions of this pin can be: (1) Bit 6 of port B. It can be configured as input or output with pull-high resistor. (2) External interrupt line 1. <u>Both rising edge and falling edge are accepted to request interrupt service.</u>
PB7	IO ST / CMOS	The functions of this pin can be: Bit 7 of port B. It can be configured as input or output with pull-high resistor.
PA7	IO ST / CMOS	The functions of this pin can be: Bit 7 of port A. It can be configured as input or output with pull-high resistor.
VDD		Positive power
GND		Ground
<b>Notes:</b> <b>IO:</b> input/output; <b>ST:</b> Schmitt Trigger input; <b>CMOS:</b> CMOS voltage level		

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 4. Device Characteristics

#### 4.1. AC/DC Device Characteristics

Symbol	Description	Min	Typ	Max	Unit	Conditions
V <sub>DD</sub>	Operating Voltage	1.8*		3.6	V	* Subject to LVR tolerance
V <sub>LVR</sub>	LVR Voltage		1.6 1.6 1.6	1.8 1.9 2.0	V	Ta = 25°C 0°C < Ta < 70°C -20°C < Ta < 70°C
I <sub>OP</sub>	Operating Current		0.5 5		mA uA	f <sub>SYS</sub> =IHRC/16=1MIPS@3V f <sub>SYS</sub> =ILRC=36KHz@3V
I <sub>PD</sub>	Power Down Current (by <b>stopsys</b> command)		1		uA	V <sub>DD</sub> =3V
I <sub>PS</sub>	Power Save Current (by <b>stopexe</b> command) *Disable IHRC		1		uA	V <sub>DD</sub> =3V
V <sub>IL</sub>	Input low voltage for IO lines	0		0.1 V <sub>DD</sub>	V	
V <sub>IH</sub>	Input high voltage for IO lines	0.8 V <sub>DD</sub> 0.7 V <sub>DD</sub>		V <sub>DD</sub> V <sub>DD</sub>	V	PA5 Other IO
I <sub>OL</sub>	IO lines sink current REM Others		220 20		mA	V <sub>DD</sub> =3V, V <sub>OL</sub> =0.3V
I <sub>OH</sub>	IO lines drive current PA5/REM Others		0 10		mA	V <sub>DD</sub> =3V, V <sub>OH</sub> =2.7V
V <sub>IN</sub>	Input voltage	-0.3		V <sub>DD</sub> +0.3	V	
I <sub>INJ (PIN)</sub>	Injected current on pin		1		uA	V <sub>DD</sub> +0.3 ≥ V <sub>IN</sub> ≥ -0.3
R <sub>PH</sub>	Pull-high Resistance		120		KΩ	V <sub>DD</sub> =3V
f <sub>IHRC</sub>	Frequency of IHRC after calibration *	15.76*	16*	16.24*	MHz	V <sub>DD</sub> =1.8V~3.6V -20°C < Ta < 70°C
f <sub>ILRC</sub>	Frequency of ILRC *		36		KHz	V <sub>DD</sub> = 3V
t <sub>INT</sub>	Interrupt pulse width	30			ns	V <sub>DD</sub> = 3V
t <sub>WDT</sub>	Watchdog timeout period		4k		ILRC clock period	misc[1:0]=00 (default)
			8k			misc[1:0]=01
			32k			misc[1:0]=10
			128k			misc[1:0]=11
t <sub>SBP</sub>	System boot-up period from power-on		40		ms	@ V <sub>DD</sub> =3V
t <sub>RST</sub>	External reset pulse width	120			us	@ V <sub>DD</sub> =3V

\* These parameters are for design reference, not tested for each chip.

# IRC12x

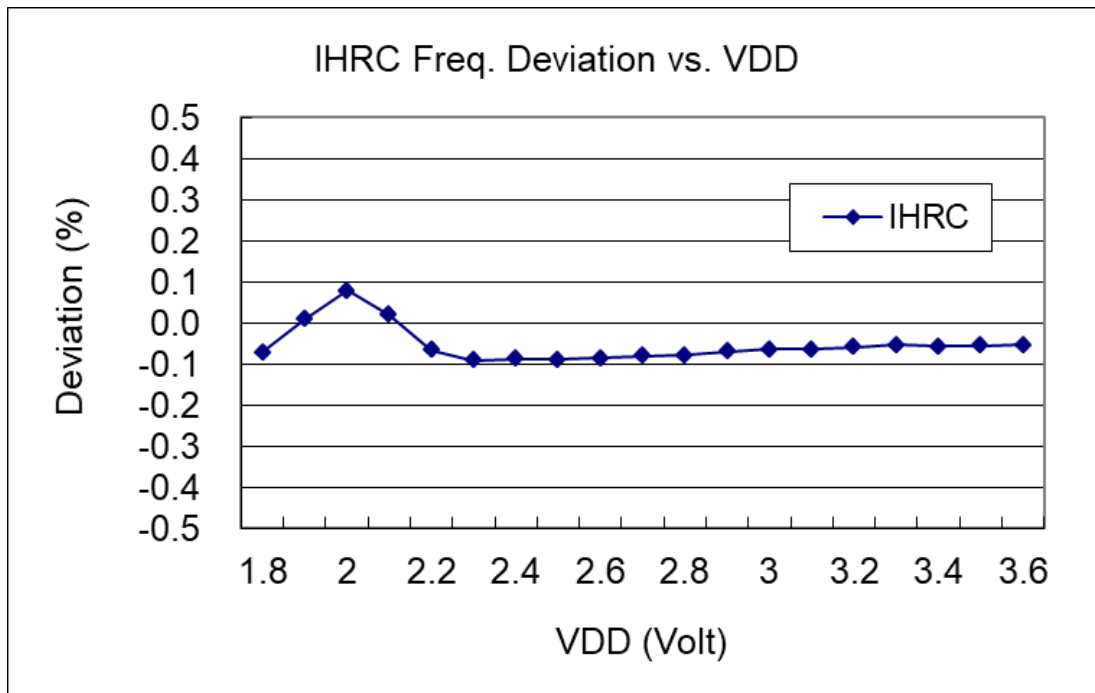
## 1KW OTP type IR Remote Control MCU

---

### 4.2. Absolute Maximum Ratings

- Supply Voltage..... 1.8V ~ 3.6V
- Input Voltage.....  $-0.3V \sim V_{DD} + 0.3V$
- Operating Temperature.....  $-20^{\circ}\text{C} \sim 70^{\circ}\text{C}$
- Junction Temperature.....  $-50^{\circ}\text{C} \sim 125^{\circ}\text{C}$
- Storage Temperature.....  $150^{\circ}\text{C}$

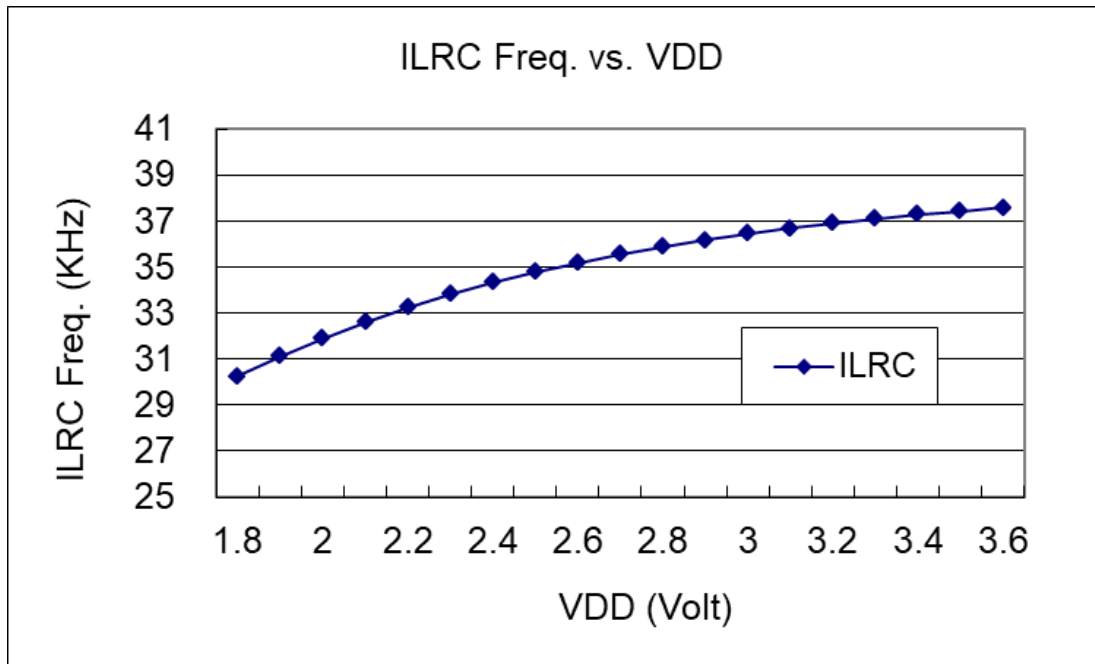
### 4.3. Typical IHRC frequency vs. VDD (calibrated to 16MHz)



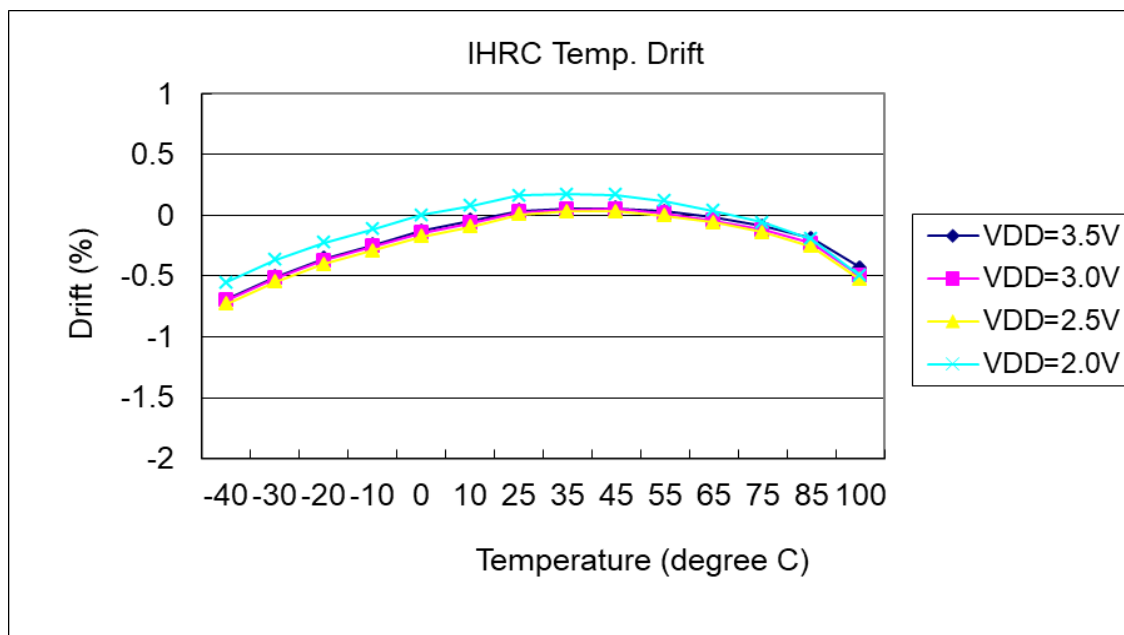
# IRC12x

## 1KW OTP type IR Remote Control MCU

### 4.4. Typical ILRC frequency vs. VDD



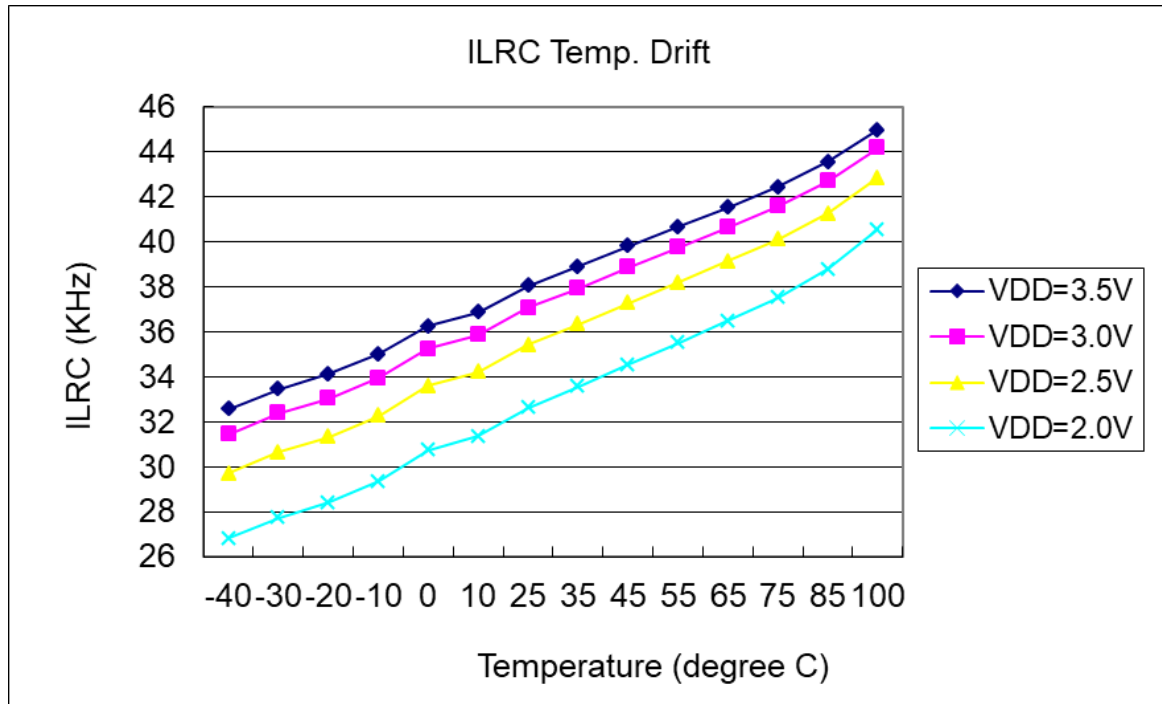
### 4.5. Typical IHRC Frequency vs. Temperature (calibrated to 16MHz)



# IRC12x

## 1KW OTP type IR Remote Control MCU

### 4.6. Typical ILRC Frequency vs. Temperature

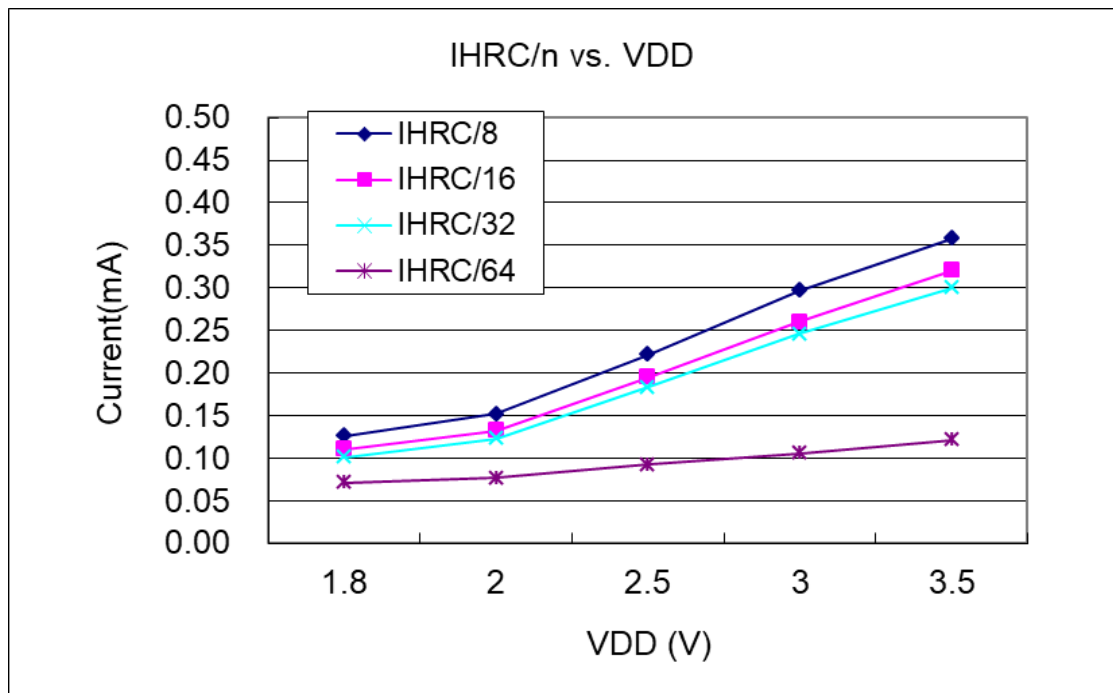


### 4.7. Typical operating current vs. VDD @ system clock = IHRC/n

➤ Conditions:

Pa0 interval (1s) high and low level flip.

**ON:** Band-gap, LVR, IHRC; **OFF:** T16, Interrupt, ILRC; other: input and no floating



# IRC12x

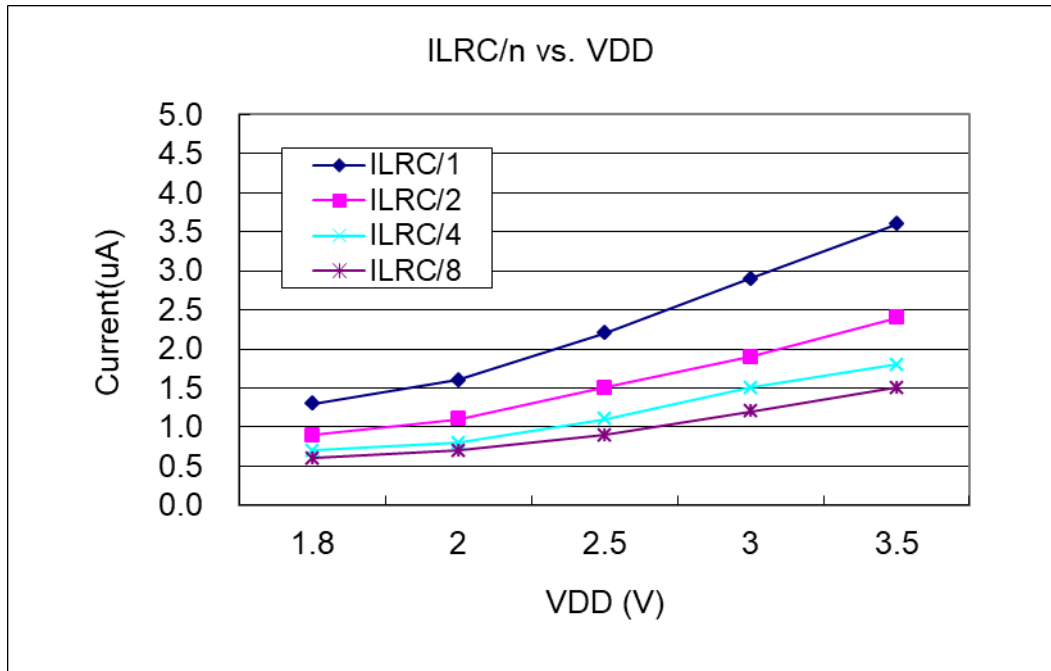
## 1KW OTP type IR Remote Control MCU

### 4.8. Typical operating current vs. VDD @ system clock = ILRC/n

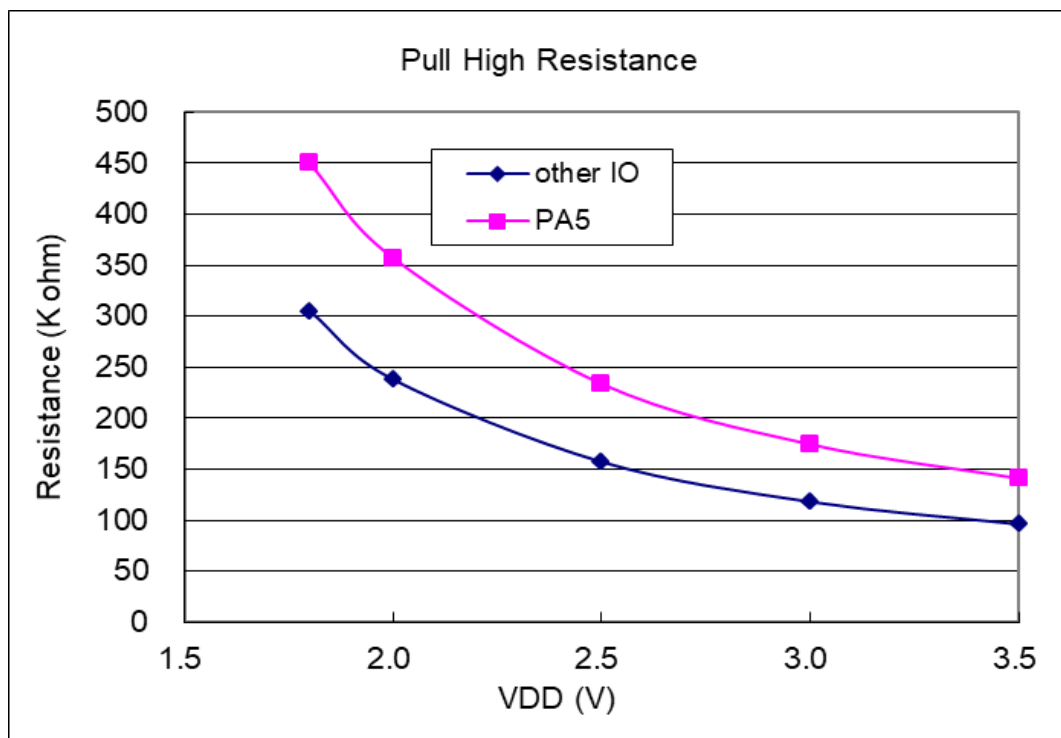
➤ Conditions:

Pa0 interval (1s) high and low level flip.

**ON:** Band-gap, LVR, ILRC; **OFF:** T16, Interrupt, IHRC; other: input and no floating



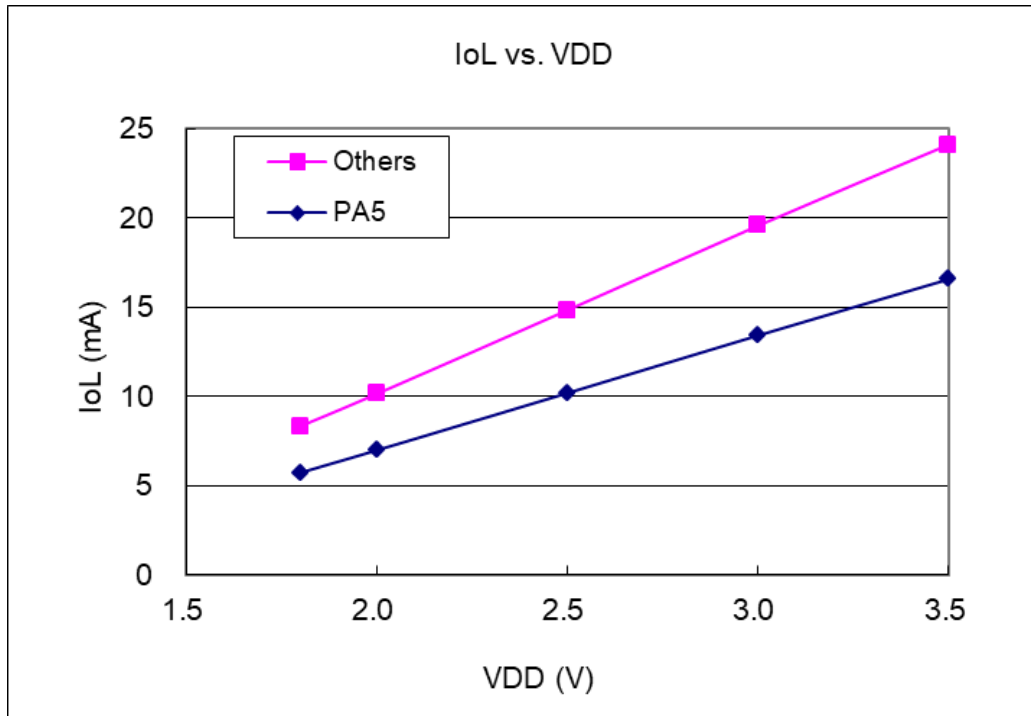
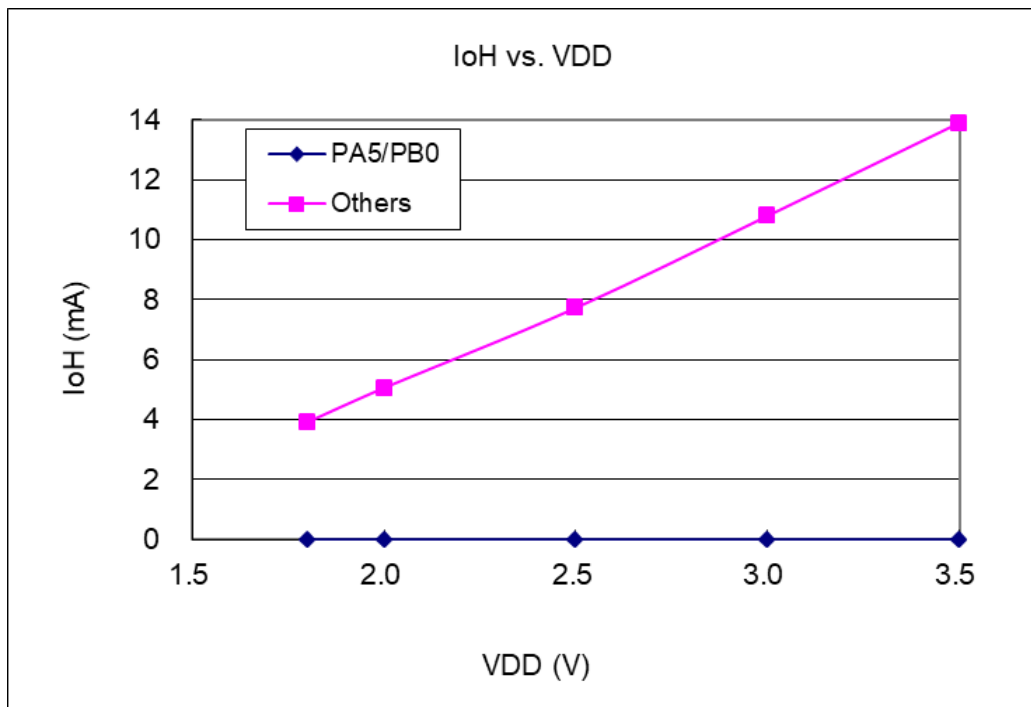
### 4.9. Typical IO pull high resistance



# IRC12x

## 1KW OTP type IR Remote Control MCU

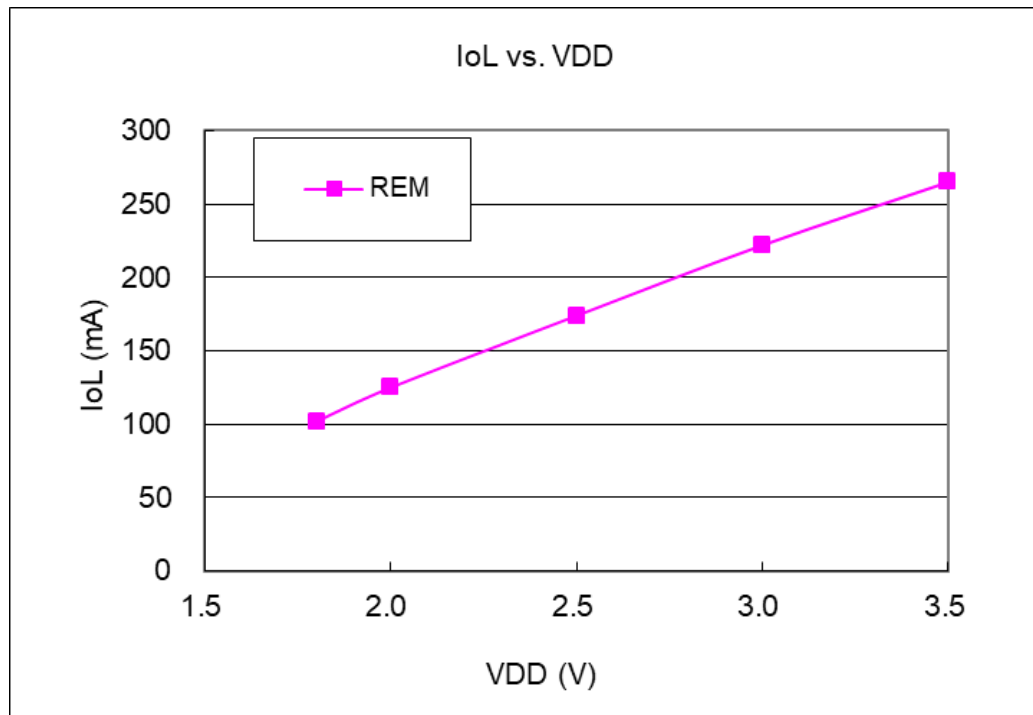
### 4.10. Typical IO driving current ( $I_{OH}$ ) and sink current ( $I_{OL}$ )



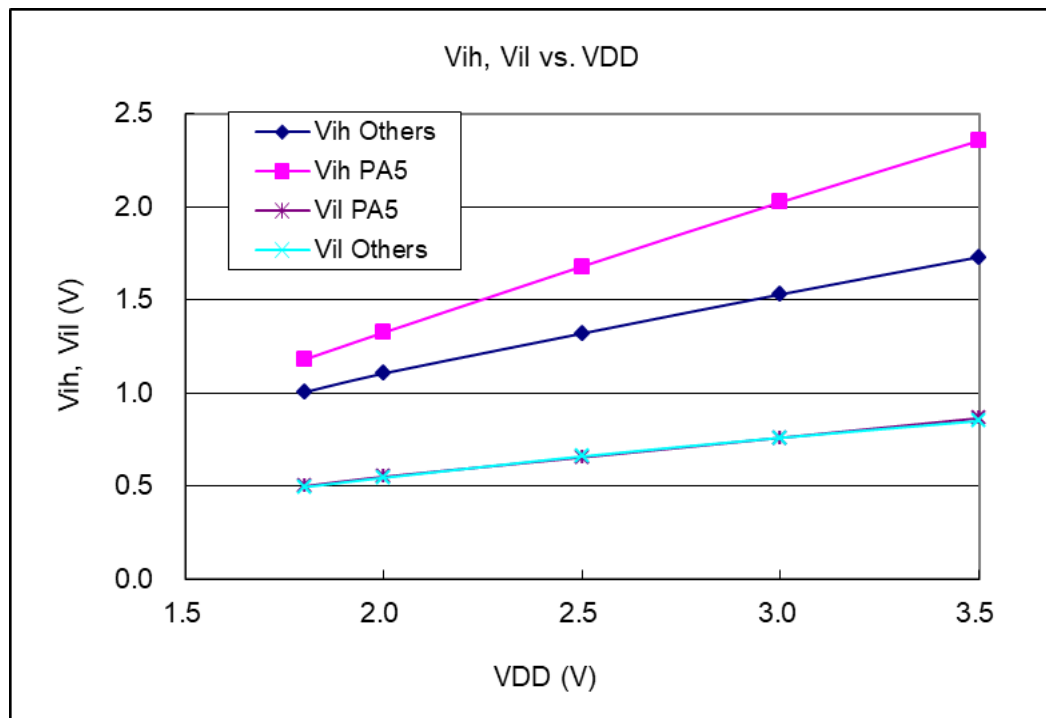


# IRC12x

## 1KW OTP type IR Remote Control MCU



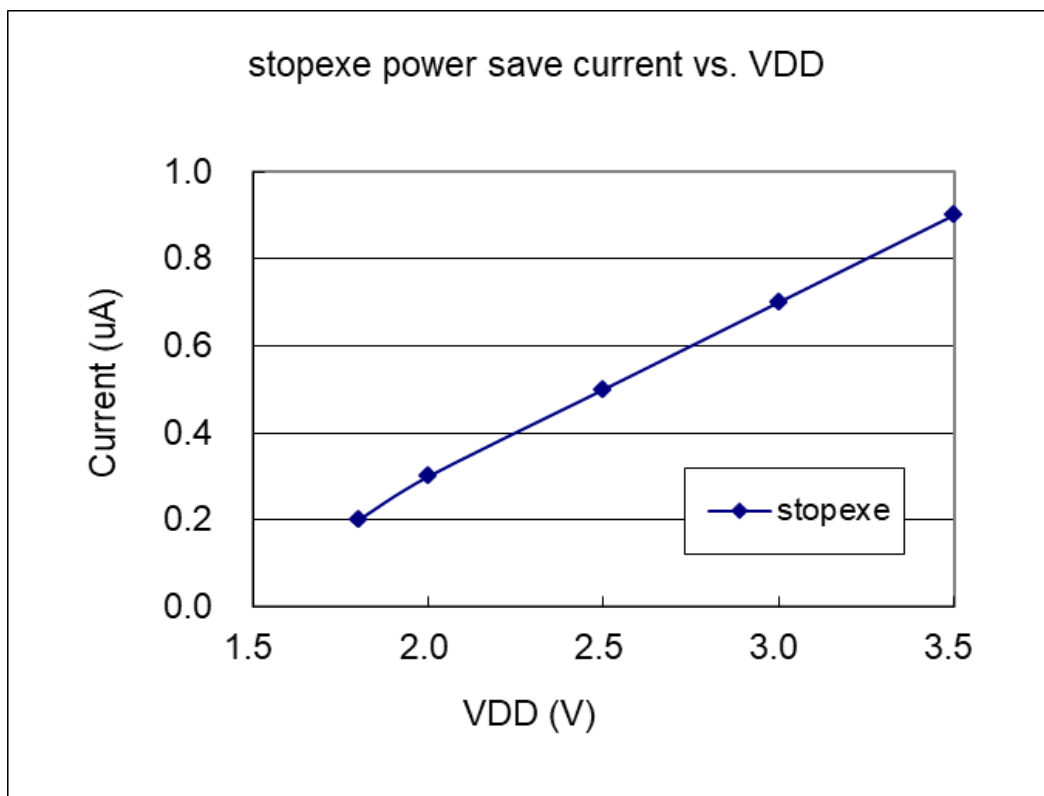
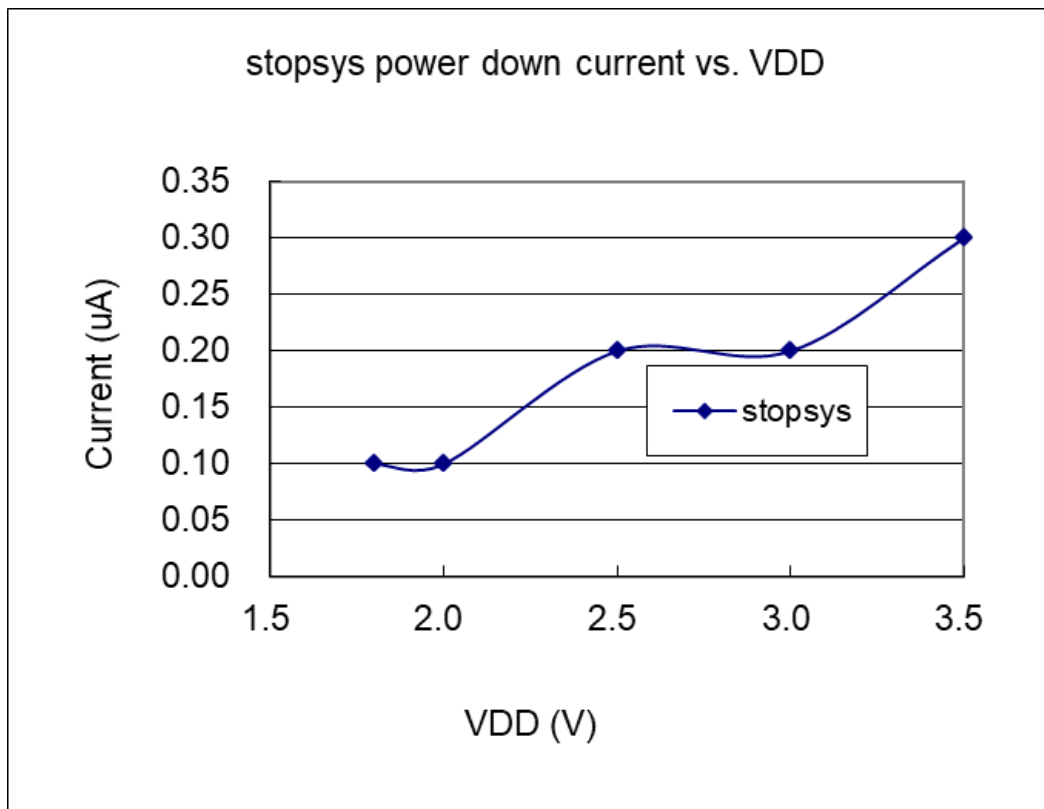
### 4.11. Typical IO input high/low threshold voltage ( $V_{IH}/V_{IL}$ )



# IRC12x

## 1KW OTP type IR Remote Control MCU

### 4.12. Typical power down current ( $I_{PD}$ ) and power save current ( $I_{PS}$ )



# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5. Functional Description

#### 5.1. Program Memory – OTP

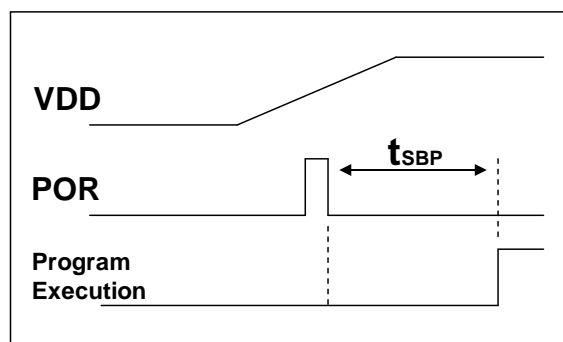
The OTP (One Time Programmable) program memory is used to store the program to be executed. The OTP program memory may contains the data, tables and interrupt entry. After reset, the initial address for FPP0 is 0x000. The interrupt entry is 0x010. The OTP program memory for IRC12x is 1KW that is partitioned as Table1. The OTP memory from address 0x3F0 to 0x3FF is for system using, address space from 0x001 to 0x00F and from 0x011 to 0x3EF is user program space.

Address	Function
0x000	FPP0 reset – goto instruction
0x001	User program
•	•
•	•
0x00F	User program
0x010	Interrupt entry address
0x011	User program
•	•
0x3EF	User program
0x3F0	System Using
•	•
0x3FF	System Using

Table 1: Program Memory Organization

#### 5.2. Boot Procedure

POR (Power-On-Reset) is used to reset IRC12x when power up, the boot-up time is about 1000 ILRC clock cycles. Customer must ensure the stability of supply voltage after power up no matter which option is chosen, the power up sequence is shown in the Fig. 1 and  $t_{SBP}$  is the boot-up time.



Boot up from Power-On Reset

Fig 1: Power UP Sequence

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 5.3. Data Memory – SRAM

The access of data memory can be byte or bit operation. Besides data storage, the SRAM data memory is also served as data pointer of indirect access method and the stack memory.

The stack memory is defined in the data memory. The stack pointer is defined in the stack pointer register; the depth of stack memory is defined by the user. The arrangement of stack memory fully flexible and can be dynamically adjusted by the user.

For indirect memory access mechanism, the data memory is used as the data pointer to address the data byte. All the data memory could be the data pointer; it's quite flexible and useful to do the indirect memory access. Since the data width is 8-bit, all the 32 bytes data memory of IRC12x can be accessed by indirect access mechanism.

### 5.4. Oscillator and clock

There are two oscillator circuits provided by IRC12x: internal high RC oscillator (IHRC) and internal low RC oscillator (ILRC), and these two oscillators are enabled or disabled by registers `clkmd.4` and `clkmd.2` independently. User can choose one of these two oscillators as system clock source and use ***clkmd*** register to target the desired frequency as system clock to meet different applications.

In the fast boot up mode, IHRC is defaulted by on. In the slow boot up mode, IHRC is defaulted by off. The instruction of `.ADJUST_IC` will adjust the switch of IHRC by user chose, when the program executes.

Oscillator Module	Enable/Disable
IHRC	<code>clkmd.4</code>
ILRC	<code>clkmd.2</code>

Table 2: Two oscillation circuits

#### 5.4.1. Internal High RC oscillator and Internal Low RC oscillator

After boot-up, the IHRC and ILRC oscillators are enabled. The frequency of IHRC can be calibrated to eliminate process variation by ***ihrcr*** register, normally it is calibrated to 16MHz. Please refer to the measurement chart for IHRC frequency verse  $V_{DD}$  and IHRC frequency verse temperature.

The frequency of ILRC will vary by process, supply voltage and temperature, please refer to DC specification and do not use for accurate timing application.

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5.4.2. IHRC calibration

The IHRC frequency and band-gap reference voltage may be different chip by chip due to manufacturing variation, IRC12x provide the IHRC frequency calibration to eliminate this variation, and this function can be selected when compiling user's program and the command will be inserted into user's program automatically.

The calibration command is shown as below:

**.ADJUST\_IC** SYSCLK=IHRC/(p1), IHRC=(p2)MHz, V<sub>DD</sub>=(p3)V;

**p1**= 8, 16, 32; In order to provide different system clock.

**p2**=14 ~ 18; In order to calibrate the chip to different frequency, 16MHz is the usually one.

**p3**=2.3 ~ 5.5; In order to calibrate the chip under different supply voltage.

### 5.4.3. IHRC Frequency Calibration and System Clock

During compiling the user program, the options for IHRC calibration and system clock are shown as Table 3:

<b>SYSClk</b>	<b>CLKMD</b>	<b>IHRCR</b>	<b>Description</b>
○ Set IHRC / 8	= 34h (IHRC / 8)	Calibrated	IHRC calibrated to 16MHz, CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 14h (IHRC / 16)	Calibrated	IHRC calibrated to 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 74h (IHRC / 32)	Calibrated	IHRC calibrated to 16MHz, CLK=0.5MHz (IHRC/32)
○ Set ILRC	= E4h (ILRC / 1)	Calibrated	IHRC calibrated to 16MHz, CLK=ILRC
○ Disable	No change	No change	IHRC not calibrated, CLK not changed

Table 3: Options for IHRC Frequency Calibration

Usually, **.ADJUST\_IC** will be the first command after boot up, in order to set the target operating frequency whenever starting the system. The program code for IHRC frequency calibration is executed only one time that occurs in writing the codes into OTP memory; after then, it will not be executed again. If the different option for IHRC calibration is chosen, the system status is also different after boot. The following shows the status of IRC12x for different option:

(1) **.ADJUST\_IC** SYSCLK=IHRC/8, IHRC=16MHz, V<sub>DD</sub>=2.5V

After boot up, CLKMD = 0x34:

- ◆ IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=2.5V and IHRC module is enabled
- ◆ System CLK = IHRC/8 = 2MHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(2) **.ADJUST\_IC** SYSCLK=IHRC/16, IHRC=16MHz, V<sub>DD</sub>=2.3V

After boot up, CLKMD = 0x14:

- ◆ IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=2.3V and IHRC module is enabled
- ◆ System CLK = IHRC/16 = 1MHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

# IRC12x

## 1KW OTP type IR Remote Control MCU

(3) .ADJUST\_IC SYSClk=IHRC/32, IHRC=16MHz, V<sub>DD</sub>=5V

After boot up, CLKMD = 0x74:

- ◆ IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=5V and IHRC module is enabled
- ◆ System CLK = IHRC/32 = 500kHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(4) .ADJUST\_IC SYSClk=ILRC, IHRC=16MHz, V<sub>DD</sub>=5V

After boot up, CLKMD = 0xE4:

- ◆ IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=5V and IHRC module is disabled
- ◆ System CLK = ILRC
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(5) .ADJUST\_IC DISABLE

After boot up, CLKMD is not changed (Do nothing):

- ◆ IHRC is not calibrated and IHRC module is disabled (only enabled in fast boot-up mode)
- ◆ System CLK = ILRC or IHRC/64
- ◆ Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode

### 5.4.4. System Clock

The clock source of system clock comes from IHRC and ILRC, the hardware diagram of system clock in the IRC12x is shown as Fig.2.

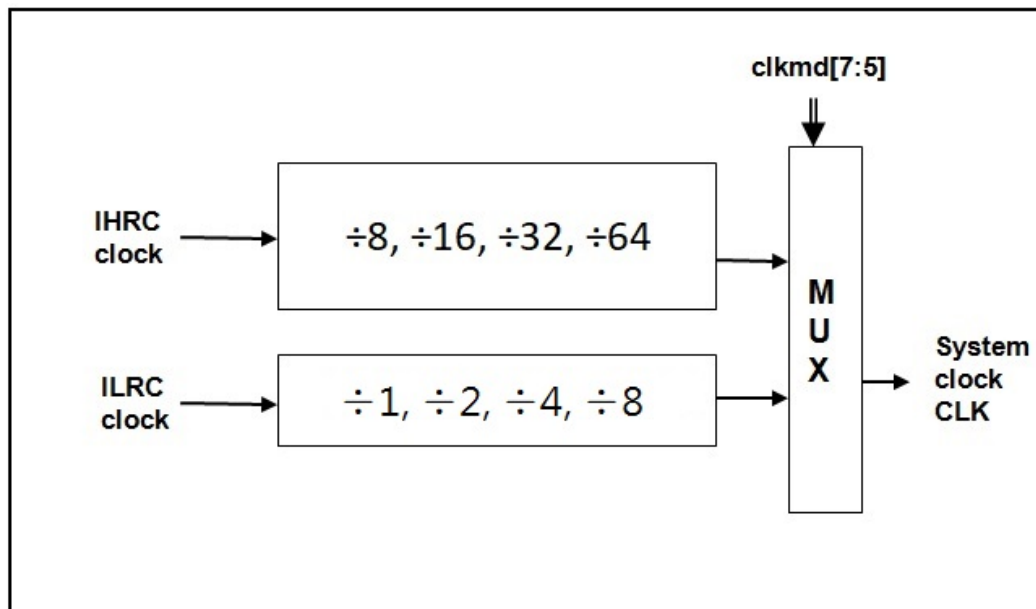


Fig.2: Options of System Clock

User can choose different operating system clock depends on its requirement.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 5.4.5. System Clock Switching

After IHRC calibration, user may want to switch system clock to a new frequency or may switch system clock at any time to optimize the system performance and power consumption. Basically, the system clock of IRC12x can be switched among IHRC and ILRC by setting the **clkmd** register at any time; system clock will be the new one after writing to **clkmd** register immediately. Please notice that the original clock module can NOT be turned off at the same time as writing command to **clkmd** register. The examples are shown as below and more information about clock switching, please refer to the “Help” -> “Application Note” -> “IC Introduction” -> “Register Introduction” -> CLKMD”.

#### Case 1: Switching system clock from ILRC to IHRC/8

```
...                               // system clock is ILRC
CLKMD      =  0x34;           // switch to IHRC/8, ILRC CAN NOT be disabled here
CLKMD.2    =  0;              // ILRC CAN be disabled at this time
...
```

#### Case 2: Switching system clock from IHRC/8 to ILRC

```
...                               // system clock is IHRC/8
CLKMD      =  0xF4;           // switch to ILRC, IHRC CAN NOT be disabled here
CLKMD.4    =  0;              // IHRC CAN be disabled at this time
...
```

#### Case 3: Switching system clock from IHRC/8 to IHRC16

```
...                               // system clock is IHRC/8, ILRC is enabled here
CLKMD      =  0X14;          // switch to IHRC/16
...
```

#### Case 4: System may hang if it is to switch clock and turn off original oscillator at the same time

```
...                               // system clock is ILRC
CLKMD      =  0x30;          // CAN NOT switch clock from ILRC to IHRC/8 and
                                turn off ILRC oscillator at the same time
```

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5.5. 16-bit Timer (Timer16)

A 16-bit hardware timer (Timer16) is implemented in the IRC12x, the clock sources of Timer16 may come from system clock (CLK), internal high RC oscillator (IHRC), internal low RC oscillator (ILRC), PA4 and PA0, a multiplex is used to select clock output for the clock source. Before sending clock to the counter16, a pre-scaling logic with divided-by-1, 4, 16, and 64 is used for wide range counting. The 16-bit counter performs up-counting operation only, the counter initial values can be stored from memory by **stt16** instruction and the counting values can be loaded to memory by **ldt16** instruction.

The interrupt source of Timer16 comes from one of bit 8 to 15 of 16-bit counter, and the interrupt type can be rising edge trigger or falling edge trigger which is specified in the **intgs.4** register. The hardware diagram of Timer16 is shown as Fig.3.

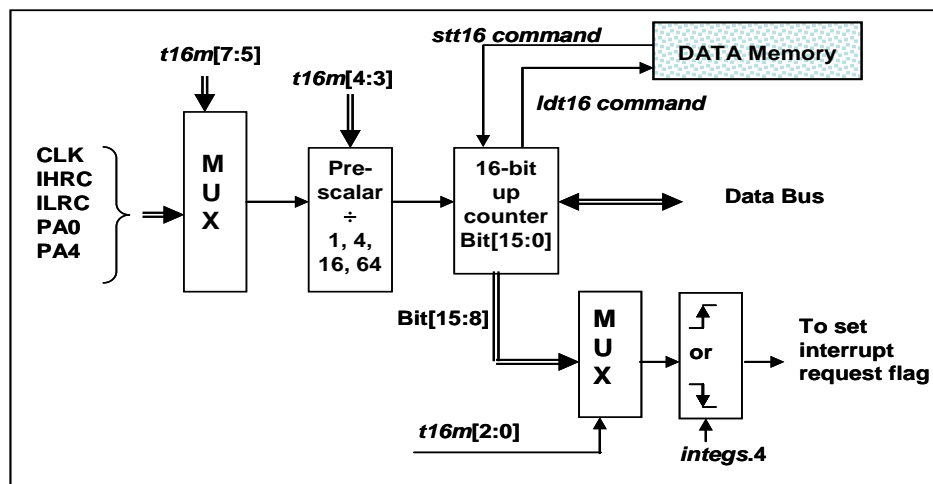


Fig.3: Hardware diagram of Timer16

When using the Timer16, the syntax for Timer16 has been defined in the .INC file. There are three parameters to define the Timer16; 1<sup>st</sup> parameter is used to define the clock source of Timer16, 2<sup>nd</sup> parameter is used to define the pre-scaler and the last one is to define the interrupt source. The detail description is shown as below:

```

T16M  IO_RW  0x06
$ 7~5:  STOP, SYSCLK, X, PA4_F, IHRC, X, ILRC, PA0_F           // 1st par.
$ 4~3:  /1, /4, /16, /64                                       // 2nd par.
$ 2~0:  BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15   // 3rd par.

```

User can define the parameters of T16M based on system requirement, some examples are shown below and more examples please refer to “Help → Application Note → IC Introduction → Register Introduction → T16M” in IDE utility.

```

$ T16M    SYSCLK, /64, BIT15;
// choose (SYSCLK/64) as clock source, every 2^16 clock to set INTRQ.2=1
// System Clock = IHRC / 8 = 2 MHz
// SYSCLK/64 = 2 MHz/64 = 31 kHz(32us), about every 2 S to generate INTRQ.2=1

```



# IRC12x

## 1KW OTP type IR Remote Control MCU

---

```
$ T16M PA0, /1, BIT8;  
// choose PA0 as clock source, every 2^9 to generate INTRQ.2=1  
// receiving every 512 times PA0 to generate INTRQ.2=1  
  
$ T16M STOP;  
// stop Timer16 counting
```

### 5.6. WatchDog Timer

The watchdog timer (WDT) is a counter with clock coming from ILRC. There are four different timeout periods of watchdog timer to be chosen by setting the *misc* register, it is:

- ◆ 4k ILRC clocks period if register misc[1:0]=00 (default)
- ◆ 8k ILRC clocks period if register misc[1:0]=01
- ◆ 32k ILRC clocks period if register misc[1:0]=10
- ◆ 128k ILRC clocks period if register misc[1:0]=11

The frequency of ILRC may drift a lot due to the variation of manufacture, supply voltage and temperature; user should reserve guard band for save operation. Besides, the watchdog period will also be shorter than expected after Reset or Wakeup events. It is suggested to clear WDT by **wdreset** command after these events to ensure enough clock periods before WDT timeout.

When WDT is timeout, IRC12x will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig.4.

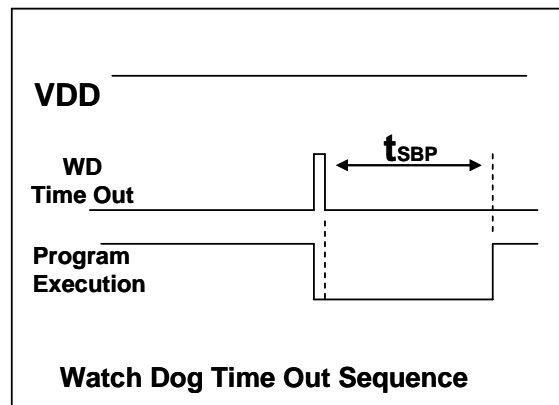


Fig. 4: Sequence of Watch Dog Time Out

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5.7. Interrupt

There are four interrupt lines for IRC12x:

- ◆ External interrupt PA0
- ◆ External interrupt PB6 (ICE does NOT support.)
- ◆ Timer16 interrupt
- ◆ ILRC/8

Every interrupt request line has its own corresponding interrupt control bit to enable or disable it; the hardware diagram of interrupt function is shown as Fig. 5. All the interrupt request flags are set by hardware and cleared by writing **intrq** register. When the request flags are set, it can be rising edge, falling edge or both, depending on the setting of register **intregs**. All the interrupt request lines are also controlled by **engint** instruction (enable global interrupt) to enable interrupt operation and **disgint** instruction (disable global interrupt) to disable it.

The stack memory for interrupt is shared with data memory and its address is specified by stack register **sp**. Since the program counter is 16 bits width, the bit 0 of stack register **sp** should be kept 0. Moreover, user can use pushaf / popaf instructions to store or restore the values of ACC and flag register to / from stack memory. Since the stack memory is shared with data memory, the stack position and level are arranged by the compiler in Mini-C project. When defining the stack level in ASM project, users should arrange their locations carefully to prevent address conflicts.

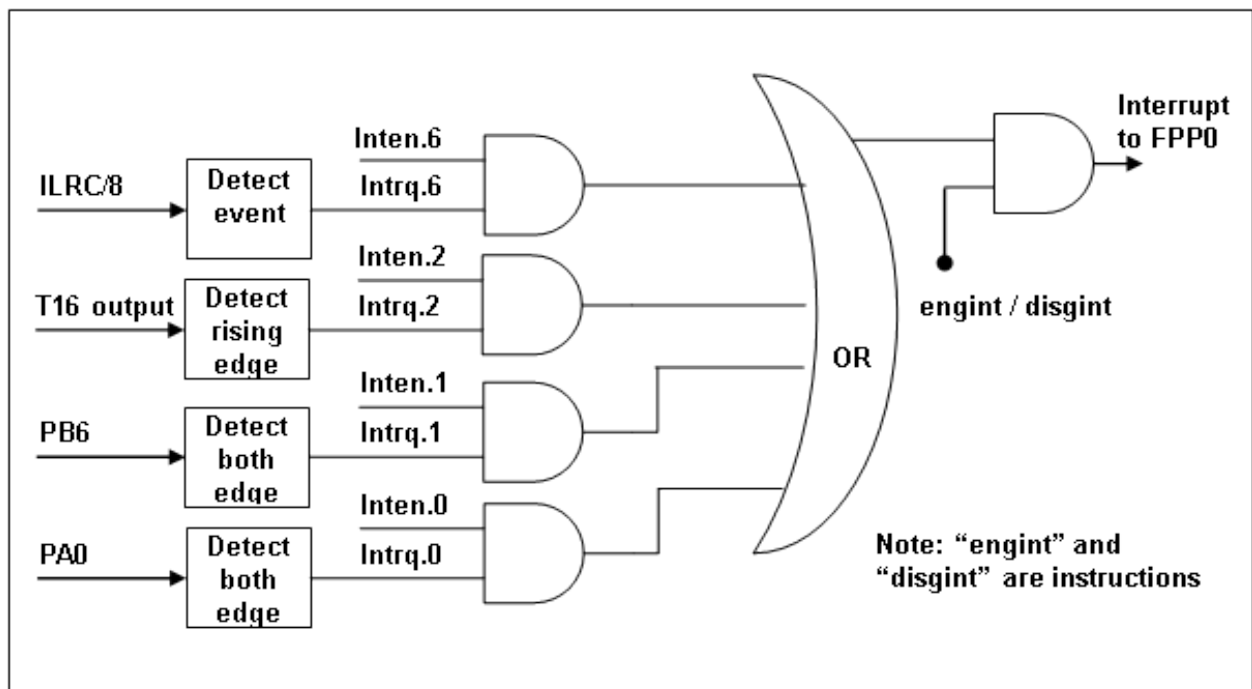


Fig.5: Hardware diagram of interrupt controller

Once the interrupt occurs, its operation will be:

- ◆ The program counter will be stored automatically to the stack memory specified by register **sp**.
- ◆ New **sp** will be updated to **sp+2**.
- ◆ Global interrupt will be disabled automatically.
- ◆ The next instruction will be fetched from address 0x010.

During the interrupt service routine, the interrupt source can be determined by reading the **intrq** register.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

Note: Even if INTEN=0, INTRQ will be still triggered by the interrupt source.

After finishing the interrupt service routine and issuing the **reti** instruction to return back, its operation will be:

- ◆ The program counter will be restored automatically from the stack memory specified by register sp.
- ◆ New sp will be updated to sp-2.
- ◆ Global interrupt will be enabled automatically.
- ◆ The next instruction will be the original one before interrupt.

User must reserve enough stack memory for interrupt, two bytes stack memory for one level interrupt and four bytes for two levels interrupt. For interrupt operation, the following sample program shows how to handle the interrupt, noticing that it needs four bytes stack memory to handle interrupt and **pushaf**.

```
void    FPPA0    (void)
{
    ...
    $ INTEN PA0;           // INTEN =1; interrupt request when PA0 level changed
    INTRQ = 0;             // clear INTRQ
    ENGINT                // global interrupt enable
    ...
    DISGINT               // global interrupt disable
    ...
}

void Interrupt (void)      // interrupt service routine
{
    PUSHAF                // store ALU and FLAG register

    // If INTEN.PA0 will be opened and closed dynamically,
    // user can judge whether INTEN.PA0 =1 or not.
    // Example: If (INTEN.PA0 && INTRQ.PA0) {...}

    // If INTEN.PA0 is always enable,
    // user can omit the INTEN.PA0 judgement to speed up interrupt service routine.

    If (INTRQ.PA0)
    {
        // Here for PA0 interrupt service routine
        INTRQ.PA0 = 0;    // Delete corresponding bit (take PA0 for example)
        ...
    }

    ...
    // X : INTRQ = 0;      // It is not recommended to use INTRQ = 0 to clear all at the end of the
                          // interrupt service routine.
                          // It may accidentally clear out the interrupts that have just occurred
                          // and are not yet processed.
    POPAF                // restore ALU and FLAG register
}
```

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5.8. Power-Save and Power-Down

There are three operational modes defined by hardware: ON mode, Power-Save mode and Power-Down modes. ON mode is the state of normal operation with all functions ON, Power-Save mode (“**stopexe**”) is the state to reduce operating current and CPU keeps ready to continue, Power-Down mode (“**stopsys**”) is used to save power deeply. Therefore, Power-Save mode is used in the system which needs low operating power with wake-up periodically and Power-Down mode is used in the system which needs power down deeply with seldom wake-up. Table 4 shows the differences in oscillator modules between Power-Save mode (“**stopexe**”) and Power-Down mode (“**stopsys**”).

Differences in oscillator modules between STOPSYS and STOPEXE		
	IHRC	ILRC
STOPSYS	Stop	Stop
STOPEXE	No Change	No Change

Table 4: Differences in oscillator modules between STOPSYS and STOPEXE

#### 5.8.1. Power-Save mode (“**stopexe**”)

Using “**stopexe**” instruction to enter the Power-Save mode, only system clock is disabled, remaining all the oscillator modules active. For CPU, it stops executing; however, for Timer16, counter keep counting if its clock source is not the system clock. The wake-up sources for “**stopexe**” can be IO-toggle or Timer16 counts to set values when the clock source of Timer16 is IHRC or ILRC modules. Wake-up from input pins can be considered as a continuation of normal execution. The detail information for Power-Save mode shows below:

- IHRC oscillator modules: No change, keep active if it was enabled.
- ILRC oscillator modules: must remain enabled, need to start with ILRC when be wakening up
- System clock: Disable, therefore, CPU stops execution.
- OTP memory is turned off.
- Timer16: Stop counting if system clock is selected or the corresponding oscillator module is disabled; otherwise, it keeps counting.
- Wake-up sources: IO toggle in digital mode (PxDIER bit is 1) or Timer16

An example shows how to use Timer16 to wake-up from “**stopexe**”:

```
$ T16M  ILRC, /1, BIT8           // Timer16 setting
...
WORD    count    =    0;
STT16   count;
stopexe;
nop;
...
```

The initial counting value of Timer16 is zero and the system will be woken up after the Timer16 counts 256 ILRC clocks.

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 5.8.2. Power-Down mode (“stopsys”)

Power-Down mode is the state of deeply power-saving with turning off all the oscillator modules. By using the “**stopsys**” instruction, this chip will be put on Power-Down mode directly. The following shows the internal status of IRC12x detail when “**stopsys**” command is issued:

- All the oscillator modules are turned off.
- OTP memory is turned off.
- The contents of SRAM and registers remain unchanged.
- Wake-up sources: IO toggle in digital mode (PxDIER bit is 1).

Wake-up from input pins can be considered as a continuation of normal execution. To minimize power consumption, all the I/O pins should be carefully manipulated before entering power-down mode. The reference sample program for power down is shown as below:

```
CLKMD = 0xF4;          // Change clock from IHRC to ILRC, disable watchdog timer
CLKMD.4 = 0;           // disable IHRC
...
while (1)
{
    STOPSYS;           // enter power-down
    if (...) break;    // if wakeup happen and check OK, then return to high speed,
                        // else stay in power-down mode again.
}
CLKMD = 0x34;          // Change clock from ILRC to IHRC/8
```

### 5.8.3. Wake-up

After entering the Power-Down or Power-Save modes, the IRC12x can be resumed to normal operation by toggling IO pins, Wake-up from T16 are available for Power-Save mode ONLY. Table 5 shows the differences in wake-up sources between STOPSYS and STOPEXE.

Differences in wake-up sources between Power-Save mode and Power-Down mode		
	IO Toggle	T16 wake-up
STOPSYS	Yes	No
STOPEXE	Yes	Yes

Table 5: Differences in wake-up sources between Power-Save mode and Power-Down mode

When using the IO pins to wake-up the IRC12x, registers **padier** should be properly set to enable the wake-up function for every corresponding pin. The time for normal wake-up is about 1K ILRC clocks counting from wake-up event; fast wake-up can be selected to reduce the wake-up time by **misc** register, and the time for fast wake-up is about 16 ILRC clocks from IO toggling.

# IRC12x

## 1KW OTP type IR Remote Control MCU

Suspend mode	Wake-up mode	Wake-up time ( $t_{WUP}$ ) from IO toggle
STOPEXE suspend or STOPSYS suspend	Fast wake-up	$16 * T_{ILRC}$ , Where $T_{ILRC}$ is the time period of ILRC
STOPEXE suspend or STOPSYS suspend	Normal wake-up	$1K * T_{ILRC}$ , Where $T_{ILRC}$ is the clock period of ILRC

Table 6: Suspend mode/Wake-up mode/IO Wake-up time

### 5.9. T-type Key Scan Wake-up

IRC12x supports a low power hardware T-type scan to wake-up from STOPEXE for T-type matrix key configuration. Every IO can be chosen to be a T-scan port by setting **PATS** & **PBTS** registers. However, a T-scan port must be an input setting and pull-high enabled port. Once a pad is enabled as a T-scan port, only a LOW level is accept to wake-up the system.

The scan rate is selected by `tscnc[1:0]` register. Lower the scan rate has lower power consumption, but less sensitive to user input. Before entering STOPEXE, turn on the hardware T-scan by `tscnc.7` register, and turn off it after wake-up from STOPEXE.

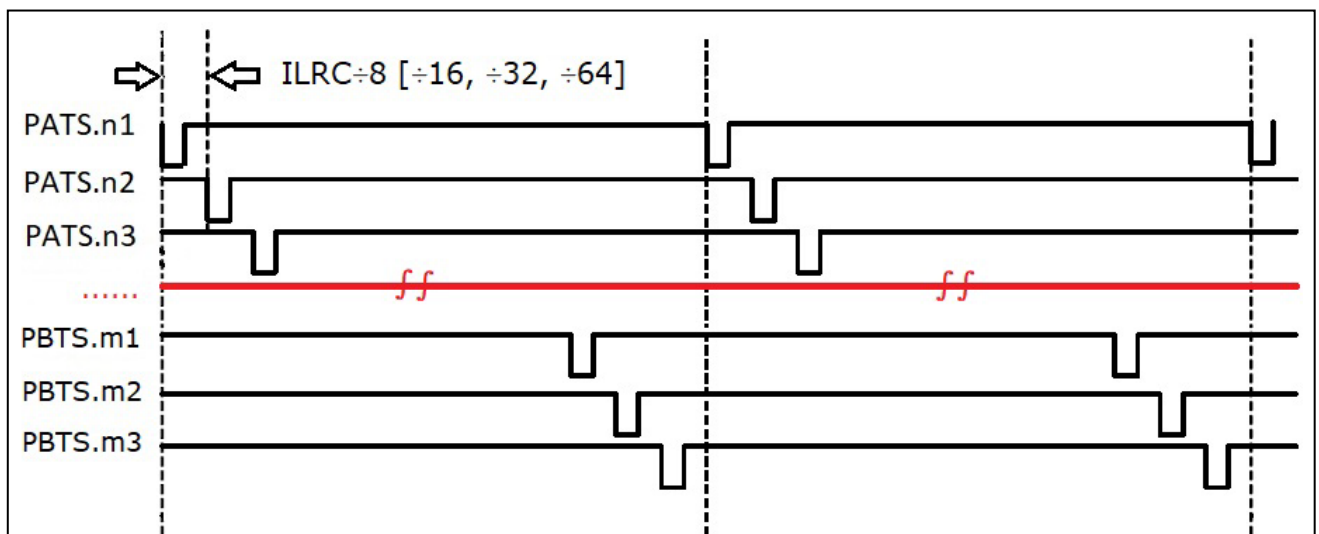


Fig. 6: T-type Key Scan waveform

Program modules can be divided into: initial stage, the software judging Key Scan, and entering STOP EXE as the hardware Key Scan.

Initial stage:

```

PATS = 0x11;           // PA0, PA4 be IO pins for Key Scan.
PBTS = 0x42;           // PB1, PB5 be IO pins for Key Scan.
PAPH = xxx | 0x11;     // Set the corresponding Pull High
PBPH = xxx | 0x42;
PAC  = xxx & ~0x11;    // Set the corresponding input mode
PBC  = xxx & ~0x42;
$TSCNC ILRC/8;         // Set the frequency of Scan is ILRC/8
  
```

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

The software judging Key Scan:

```
// During the use of program, PA0 / PA4 / PB1 / PB5 allow In/Out toggle to do Key Scan.  
// Need to be the input mode, before enter STOPEXE.
```

Enter STOPEXE as the hardware Key Scan.

```
$ CLKMD      En_ILRC, En_IHRC, ILRC/1;    //   Enter ILRC  
CLKMD.En_IHRC    =    0;                  //   Disable IHRC  
TSCNC.Enable =    1;                      //   Start Key Scan  
stopexe;         //   Sleep  
TSCNC.Enable =    0;                      //   ICwake up, doesn't do hardware Key Scan  
$ CLKMD      En_ILRC, En_IHRC, IHRC/8;    //   Enter to the fast IHRC/8
```

### 5.10. IO Pins

Other than PA5, all the pins can be independently set into two state output or input by configuring the data register (**pa**, **pb**), control registers (**pac**, **pbc**) and pull-high registers (**paph**, **pbph**). All these pins have Schmitt-trigger input buffer and output driver with CMOS level. When it is set to low, the pull-high resistor is turned off automatically. If user wants to read the pin state, please notice that it should be set to input mode before reading the data port; if user reads the data port when it is set to output mode, the reading data comes from data register, NOT from IO pad. As an example, Table 7 shows the configuration table of bit 0 of port A. The hardware diagram of IO buffer is also shown Fig. 7.

<b>pa.0</b>	<b>pac.0</b>	<b>paph.0</b>	<b>Description</b>
X	0	0	Input without pull-high resistor
X	0	1	Input with pull-high resistor
0	1	X	Output low without pull-high resistor
1	1	X	Output high without pull-high resistor

Table 7: PA0 Configuration Table

# IRC12x

## 1KW OTP type IR Remote Control MCU

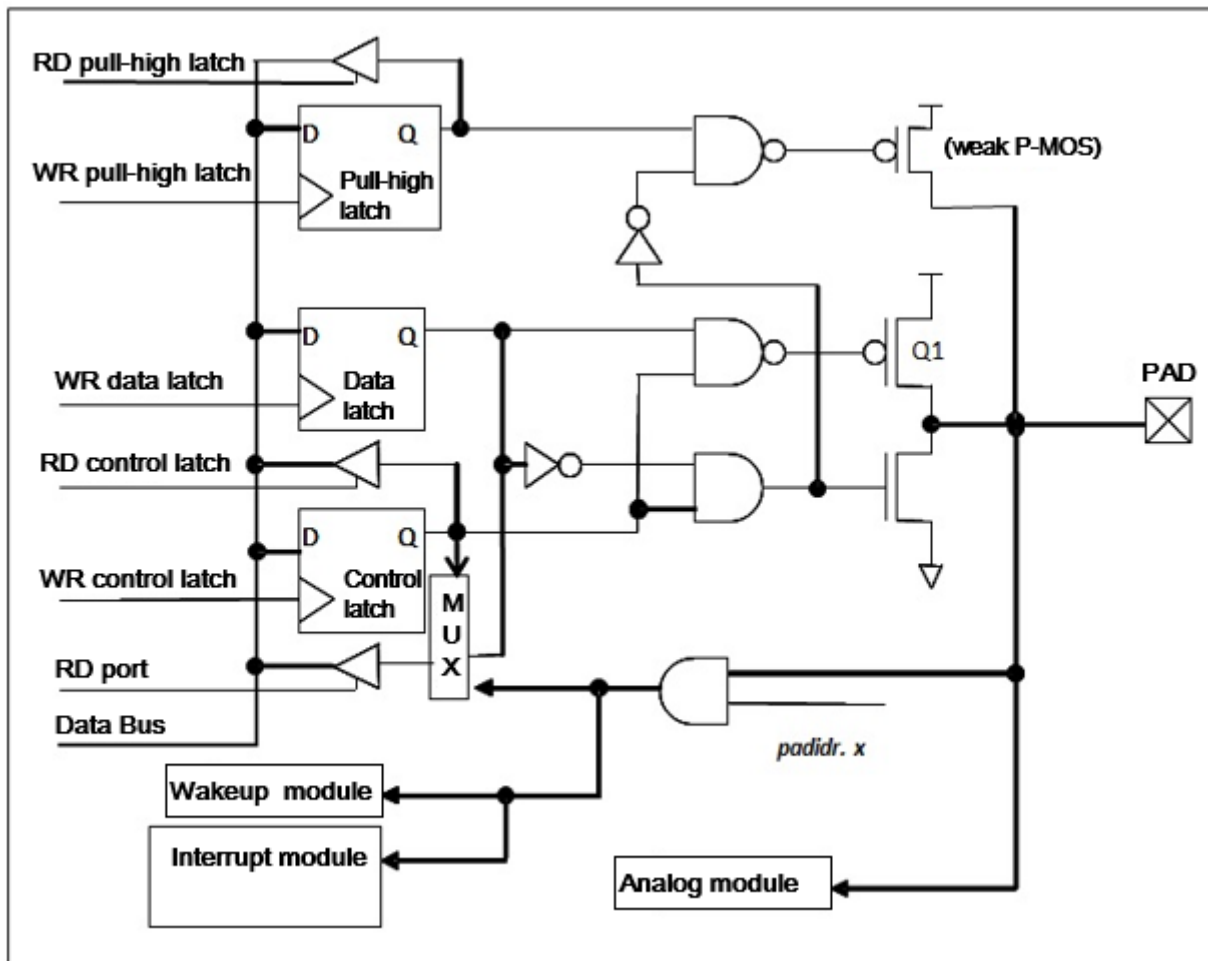


Fig. 7: Hardware diagram of IO buffer

Other than PA5, all the IO pins have the same structure; PA5 can be open-drain ONLY when setting to output mode (without Q1). The corresponding bits in registers ***padier*** & ***pbdier*** should be set to low to prevent leakage current for those pins are selected to be analog function. When IRC12x is put in power-down or power-save mode, every pin can be used to wake-up system by toggling its state. Therefore, those pins needed to wake-up system must be set to input mode and set the corresponding bits of registers ***padier*** & ***pbdier*** to high. The same reason, ***padier***.0 should be set high when PA0 is used as external interrupt pin.

## 5.11. Reset

There are many causes to reset the IRC12x, once reset is asserted, most of all the registers in IRC12x will be set to default values, system should be restarted once abnormal cases happen, or by jumping program counter to address 0x0. The data memory is in uncertain state when reset comes from power-up and LVR; however, the content will be kept when reset comes from PRSTB pin or WDT timeout.



# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 5.12.8-bits IR Carrier Frequency Generator (IRTMC / IRTMB)

One 8-bits IR Carrier frequency generator (IRTMB) is built inside the IRC12x; Fig.8 is the Hardware diagram of IRTMC. The clock source of counter may come from internal high RC oscillator. The register IRTMC is provided with divided-by-1, 2, 16, and 64 options, and they are controlled by bit [5:4].

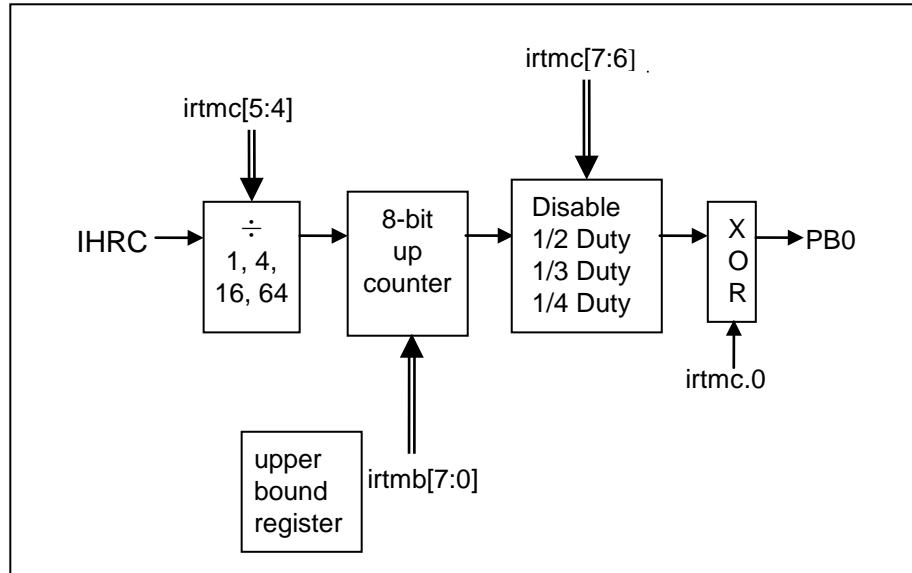


Fig.8: the Hardware diagram of IRTMC

User can select one of the three Duty Cycles, 1/2, 1/3 or 1/4 Duty, for the IR carrier by ***irtmc[7:6]***. In addition, together with ***irtmc.0***, there are two more Duty Cycles, 2/3 and 3/4 Duty, available by inverting the IR carrier. The IR carrier frequency is generated by IRTM, the period for 1/2, 1/3 and 1/4 Duty corresponds to ***irtmb*** \*2, ***irtmb*** \*3 and ***irtmb*** \*4.

IRTMB is 8-bits counter. If IR output is 40KHz, we need 16MHz divided by 400 to reach. The counting 200 times is equal to half cycle of IR.

```
IRTMB    =    200 -1;           //    Set 199 equal to counting 200 times
$ IRTMC   1/2, IHRC/1;          //    1/2 duty, 16MHz
                                     //    IR carrier = 40KHz
```

If IR output is 38KHz, we need 16MHz divided by 421 to reach. The counting 210 times is equal to half cycle of IR.

```
IRTMB    =    210 -1;           //    Set 209 equal to counting 210 times
$ IRTMC   1/2, IHRC/1;          //    1/2 duty, 16MHz
                                     //    IR carrier = 38.095KHz
```

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

To modulate the IR carrier to the REM output, setting the ***pbc.0*** first to select output mode for REM pad. Please reference the table listed below, after selecting the IR carrier frequency, the rest user has to do is to control ***pb.0*** for modulation.

<b><i>pbc.0</i></b>	<b><i>pb.0</i></b>	<b>IRTMC[7:6]</b>	<b>Description</b>
0	X	X	Disable REM sink current output
1	1	X	Disable REM sink current output
1	0	00	REM output LOW
1	0	others	REM output IR carrier

Table 8: REM Setting Configuration Table

During simulation, all R/W to IRTMC / IRTMB will be delayed, and cannot be simulated in time. Please use IC to do the actual verification.

IRTMC	IO_RW	0x1C (0x3F)	
\$ 7 ~ 6 :	Disable, 1/2, 1/3, 1/4		// <b>Select duty</b>
\$ 5 ~ 4 :	IHRC/1, IHRC/2, IHRC/16, IHRC/64		// <b>Clock Source</b>
\$ 0 :	X, Inverse		// <b>Output Inverse</b>
IRTMB	IO_WO	0x1D (0x3F)	

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 6. IO Registers

#### 6.1. ACC Status Flag Register (*flag*), IO address =0'h00

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. These four bits are "1" when read.
3	-	R/W	OV (Overflow Flag). This bit is set to be 1 whenever the sign operation is overflow.
2	-	R/W	AC (Auxiliary Carry Flag). There are two conditions to set this bit, the first one is carry out of low nibble in addition operation and the other one is borrow from the high nibble into low nibble in subtraction operation.
1	-	R/W	C (Carry Flag). There are two conditions to set this bit, the first one is carry out in addition operation, and the other one is borrow in subtraction operation. Carry is also affected by shift with carry instruction.
0	-	R/W	Z (Zero Flag). This bit will be set when the result of arithmetic or logic operation is zero; Otherwise, it is cleared.

#### 6.2. Stack Pointer Register (*sp*), IO address =0x02

Bit	Reset	R/W	Description
7 - 0	-	R/W	Stack Pointer Register. Read out the current stack pointer, or write to change the stack pointer. Please notice that bit 0 should be kept 0 due to program counter is 16 bits.

#### 6.3. Clock Mode Register (*clkmd*), IO address =0x03

Bit	Reset	R/W	Description
7 - 5	111	R/W	System clock (CLK) selection: 000: IHRC/16 001: IHRC/8 010: ILRC/8 (ICE does NOT support) 011: IHRC/32 100: IHRC/64 101: ILRC/4 110: ILRC/2 (ICE does NOT support) 111: ILRC/1
4	0	R/W	Internal High RC Enable. 0 / 1: disable / enable
3	0	R/W	Reserved
2	1	R/W	Internal Low RC Enable. 0 / 1: disable / enable If ILRC is disabled, watchdog timer is also disabled.
1	1	R/W	Watch Dog Enable. 0 / 1: disable / enable
0	0	R/W	Pin PA5/PRSTB function. 0 / 1: PA5 / PRSTB.

#### 6.4. Interrupt Enable Register (*inten*), IO address =0x04

Bit	Reset	R/W	Description
7	0	-	Reserved
6	0	R/W	Enable interrupt from ILRC/8. 0 / 1: disable / enable
5	0	-	Reserved
4	0	-	Reserved
3	0	-	Reserved
2	0	R/W	Enable interrupt from Timer16 overflow. 0 / 1: disable / enable
1	0	R/W	Enable interrupt from PB6. 0 / 1: disable / enable
0	0	R/W	Enable interrupt from PA0. 0 / 1: disable / enable

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 6.5. Interrupt Request Register (*intrq*), IO address =0x05

Bit	Reset	R/W	Description
7	-	-	Reserved
6	-	R/W	Interrupt Request from ILRC/8, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
5	-	-	Reserved
4	-	-	Reserved
3	-	-	Reserved
2	-	R/W	Interrupt Request from Timer16, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
1	-	R/W	Interrupt Request from pin PB6, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	R/W	Interrupt Request from pin PA0, this bit is set by hardware and cleared by software. 0 / 1: No Request / request

### 6.6. Timer16 mode Register (*t16m*), IO address =0x06

Bit	Reset	R/W	Description
7 - 5	000	R/W	Timer16 Clock source selection. 000: disable 001: CLK (system clock) 010: reserved 011: PA4 falling edge (from external pin) 100: IHRC 101: reserved 110: ILRC 111: PA0 falling edge (from external pin)
4 - 3	00	R/W	Timer16 clock pre-divider. 00: ÷1 01: ÷4 10: ÷16 11: ÷64
2 - 0	000	R/W	Interrupt source selection. Interrupt event happens when the selected bit status is changed. 0: bit 8 of Timer16 1: bit 9 of Timer16 2: bit 10 of Timer16 3: bit 11 of Timer16 4: bit 12 of Timer16 5: bit 13 of Timer16 6: bit 14 of Timer16 7: bit 15 of Timer16

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 6.7. MISC Registe (*misc*), IO address = 0x1b

Bit	Reset	R/W	Description
7 - 6	-	-	Reserved. (keep 0 for future compatibility)
5	0	WO	Enable fast Wake up. 0: Normal wake up. The wake-up time is 1K ILRC clocks (Not for fast boot-up) 1: Fast wake up The wake-up time is sum of 16 ILRC clocks and time of stable oscillation If wake-up from STOPEXE, time of stable oscillation is 0. If wake-up from STOPSYS, time of stable oscillation is from IHRC or ILRC stable time with power on.
4 - 2	-	-	Reserved. (keep 0 for future compatibility)
1 - 0	00	WO	Watch dog time out period: 00: 4k ILRC clock period 01: 8k ILRC clock period 10: 32k ILRC clock period 11: 128k ILRC clock period

### 6.8. Interrupt Edge Select Register (*integs*), IO address =0x0c

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved. (keep 0 for future compatibility)
4	0	WO	Timer16 edge selection. 0: rising edge of the selected bit to trigger interrupt. 1: falling edge of the selected bit to trigger interrupt.
3 - 2	00	WO	PB6 edge selection. 00: both rising edge and falling edge of the selected bit to trigger interrupt 01: rising edge of the selected bit to trigger interrupt 10: falling edge of the selected bit to trigger interrupt 11: Reserved
1 - 0	00	WO	PA0 edge selection. 00: both rising edge and falling edge of the selected bit to trigger interrupt 01: rising edge of the selected bit to trigger interrupt 10: falling edge of the selected bit to trigger interrupt 11: Reserved

### 6.9. Port A Digital Input Enable Register (*padier*), IO address =0x0d

Bit	Reset	R/W	Description
7 - 3	11111	WO	Enable PA7~PA3 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PA7~PA3 toggling.
2 - 1	-	-	Reserved.
0	1	WO	Enable PA0 digital input and and wake-up event and interrupt request. 1 / 0 : enable / disable These bits can be set to low to disable wake up from PA0 toggling and interrupt request from this pin.

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 6.10. Port B Digital Input Enable Register (*pbdier*), IO address =0x0e

Bit	Reset	R/W	Description
7	1	WO	Enable PB7 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB7 toggling.
6	1	WO	Enable PB6 digital input and wake up event and interrupt request. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB6 toggling and interrupt request from this pin.
5	1	WO	Enable PB5 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB5 toggling.
4	1	WO	Enable PB4 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB4 toggling.
3	1	WO	Enable PB3 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB3 toggling.
2	1	WO	Enable PB2 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB2 toggling.
1	1	WO	Enable PB1 digital input and wake up event. 1 / 0: enable / disable These bits can be set to low to disable wake up from PB1 toggling.
0	-	WO	Reserved

### 6.11. Port A Data Register (*pa*), IO address =0x10

Bit	Reset	R/W	Description
7 - 3	5'h00	R/W	Data register for Port A.
2 - 1	-	-	Reserved
0	0	R/W	Data register for Port A.

### 6.12. Port A Control Register (*pac*), IO address =0x11

Bit	Reset	R/W	Description
7 - 3	5'h00	R/W	Port A control registers. This register is used to define input mode or output mode for each corresponding pin of port A. 0 / 1: input / output
2 - 1	-	-	Reserved
0	0	R/W	Port A control registers. This register is used to define input mode or output mode for each corresponding pin of port A. 0 / 1: input / output

### 6.13. Port A Pull-High Register (*paph*), IO address =0x12

Bit	Reset	R/W	Description
7 - 0	5'h00	WO	Port A pill-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port A. 0 / 1 : disable / enable
2 - 1	-	-	Reserved
0	0	WO	Port A pill-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port A. 0 / 1 : disable / enable

### 6.14. Port B Data Register (*pb*), IO address =0x14

Bit	Reset	R/W	Description
7 - 1	7'h00	R/W	Data register for Port B.
0	0	R/W	IRTX carrier output enable register. 0: REM outputs IRTX Carrier. 1: REM output disable.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 6.15. Port B Control Register (*pbc*), IO address =0x15

Bit	Reset	R/W	Description
7 - 1	7'h00	R/W	Port B control registers. This register is used to define input mode or output mode for each corresponding pin of port B. 0 / 1: input / output
0	0	R/W	REM output enable register. 0 / 1 : disable / enable

### 6.16. Port B Pull-High Register (*pbph*), IO address =0x16

Bit	Reset	R/W	Description
7 - 1	7'h00	R/W	Port B pill-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port B. 0 / 1 : disable / enable
0	-	-	Reserved

### 6.17. Port A T-scan Select Register (*pats*), IO address =0x13

Bit	Reset	R/W	Description
7 - 3	5'h00	R/W	Select PA7~PA3 to be T-scan or normal IO wake-up port in STOPEXE. 0 / 1 : normal / T-scan
2 - 1	-	-	Reserved
0	0	R/W	Select PA0 to be T-scan or normal IO wake-up port in STOPEXE. 0 / 1 : normal / T-scan

### 6.18. Port B T-scan Select Register (*pbts*), IO address =0x17

Bit	Reset	R/W	Description
7 - 1	7'h00	R/W	Select PB7~PB1 to be T-scan or normal IO wake-up port in STOPEXE. 0 / 1 : normal / T-scan
0	0	R/W	Reserved

### 6.19. T-scan Control Register (*tscnc*), IO address =0x1E

Bit	Reset	R/W	Description
7	0	R/W	Enable hardware T-scan in STOPEXE. 0 / 1 : disable / enable
6 - 2	-	-	Reserved
1 - 0	2'h0	R/W	Select T-scan frequency : 00: ILRC/8 01: ILRC/16 10: ILRC/32 11: ILRC/64

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 6.20. IR Timer Control Register (*irtmc*), IO address =0x1C

Bit	Reset	R/W	Description
7 - 6	2'h0	R/W	IRTX Carrier Duty Cycle selection. 00 : IRTX Carrier is always 0. 01 : IRTX Carrier is 1/2 Duty. The IRTX Carrier Period is irtmb*2. 10 : IRTX Carrier is 1/3 Duty. The IRTX Carrier Period is irtmb*3. 11 : IRTX Carrier is 1/4 Duty. The IRTX Carrier Period is irtmb*4.
5 - 4	2'h0	R/W	IR timer frequency selection 00: IHRC/1 01: IHRC/2 10: IHRC/16 11: IHRC/64
3 - 1	-	-	Reserved
0	0	R/W	Enable to inverse the polarity of IRTX Carrier output. 0 / 1: disable / enable

### 6.21. IR Timer Bound Register (*irtmb*), IO address =0x1D

Bit	Reset	R/W	Description
7 - 0	-	WO	IRTX Carrier Timer Bound register.



# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 7. Instructions

Symbol	Symbol
ACC	Accumulator (Abbreviation of accumulator)
a	Accumulator (symbol of accumulator in program)
sp	Stack pointer
flag	ACC status flag register
I	Immediate data
&	Logical AND
	Logical OR
←	Movement
^	Exclusive logic OR
+	Add
—	Subtraction
-~	NOT (logical complement, 1's complement)
⌋	NEG (2's complement)
OV	Overflow (The operational result is out of range in signed 2's complement number system)
Z	Zero (If the result of ALU operation is zero, this bit is set to 1)
C	Carry (The operational result is to have carry out for addition or to borrow carry for subtraction in unsigned number system)
AC	Auxiliary Carry (If there is a carry out from low nibble after the result of ALU operation, this bit is set to 1)
word	Only addressed in 0~0x1F (0~31) is allowed
M.n	Only addressed in 0~0x7F (0~127) is allowed

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 7.1. Data Transfer Instructions

<i>mov</i> a, l	<p>Move immediate data into ACC.</p> <p>Example: <i>mov</i> a, 0x0f;</p> <p>Result: <math>a \leftarrow 0fh</math>;</p> <p>Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>mov</i> M, a	<p>Move data from ACC into memory</p> <p>Example: <i>mov</i> MEM, a;</p> <p>Result: <math>MEM \leftarrow a</math></p> <p>Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>mov</i> a, M	<p>Move data from memory into ACC</p> <p>Example: <i>mov</i> a, MEM ;</p> <p>Result: <math>a \leftarrow MEM</math>; Flag Z is set when MEM is zero.</p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p>
<i>mov</i> a, IO	<p>Move data from IO into ACC</p> <p>Example: <i>mov</i> a, pa ;</p> <p>Result: <math>a \leftarrow pa</math>; Flag Z is set when pa is zero.</p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p>
<i>mov</i> IO, a	<p>Move data from ACC into IO</p> <p>Example: <i>mov</i> pb, a;</p> <p>Result: <math>pb \leftarrow a</math></p> <p>Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>ldt16</i> word	<p>Move 16-bit counting values in Timer16 to memory in word.</p> <p>Example: <i>ldt16</i> word;</p> <p>Result: <math>word \leftarrow 16\text{-bit timer}</math></p> <p>Affected flags: [N] Z [N] C [N] AC [N] OV</p> <p>Application Example:</p> <pre> word      T16val ;           // declare a RAM word ... clear     lb@ T16val ;       // clear T16val (LSB) clear     hb@ T16val ;       // clear T16val (MSB) stt16     T16val ;           // initial T16 with 0 ... set1      t16m.5 ;           // enable Timer16 ... set0      t16m.5 ;           // disable Timer 16 ldt16     T16val ;           // save the T16 counting value to T16val .... </pre>

# IRC12x

## 1KW OTP type IR Remote Control MCU

<i>stt16</i> word	<p>Store 16-bit data from memory in word to Timer16.</p> <p>Example: <i>stt16</i> word;</p> <p>Result: 16-bit timer ← word</p> <p>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV</p> <p>Application Example:</p> <hr/> <pre> word    T16val ;           // declare a RAM word ... mov     a, 0x34 ; mov     lb@ T16val , a ;    // move 0x34 to T16val (LSB) mov     a, 0x12 ; mov     hb@ T16val , a ;    // move 0x12 to T16val (MSB) stt16   T16val ;           // initial T16 with 0x1234 ... </pre> <hr/>
<i>idxm</i> a, index	<p>Move data from specified memory to ACC by indirect method. It needs 2T to execute this instruction.</p> <p>Example: <i>idxm</i> a, index;</p> <p>Result: a ← [index], where index is declared by word.</p> <p>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV</p> <p>Application Example:</p> <hr/> <pre> word    RAMIndex ;         // declare a RAM pointer ... mov     a, 0x5B ;           // assign pointer to an address (LSB) mov     lb@RAMIndex, a ;    // save pointer to RAM (LSB) mov     a, 0x00 ;           // assign 0x00 to an address (MSB), should be 0 mov     hb@RAMIndex, a ;    // save pointer to RAM (MSB) ... idxm    a, RAMIndex ;       // move memory data in address 0x5B to ACC </pre> <hr/>
<i>ldxm</i> index, a	<p>Move data from ACC to specified memory by indirect method. It needs 2T to execute this instruction.</p> <p>Example: <i>ldxm</i> index, a;</p> <p>Result: [index] ← a; where index is declared by word.</p> <p>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV</p> <p>Application Example:</p> <hr/> <pre> word    RAMIndex ;         // declare a RAM pointer ... mov     a, 0x5B ;           // assign pointer to an address (LSB) mov     lb@RAMIndex, a ;    // save pointer to RAM (LSB) mov     a, 0x00 ;           // assign 0x00 to an address (MSB), should be 0 mov     hb@RAMIndex, a ;    // save pointer to RAM (MSB) ... mov     a, 0xA5 ; ldxm    RAMIndex, a ;       // mov 0xA5 to memory in address 0x5B </pre> <hr/>

# IRC12x

## 1KW OTP type IR Remote Control MCU

<i>xch</i> M	<p>Exchange data between ACC and memory</p> <p>Example: <code>xch MEM ;</code></p> <p>Result:    <math>MEM \leftarrow a, a \leftarrow MEM</math></p> <p>Affected flags: <math>\{N\} Z \quad \{N\} C \quad \{N\} AC \quad \{N\} OV</math></p>
<i>pushaf</i>	<p>Move the ACC and flag register to memory that address specified in the stack pointer.</p> <p>Example: <code>pushaf;</code></p> <p>Result:    <math>[sp] \leftarrow \{flag, ACC\};</math>  <math>sp \leftarrow sp + 2 ;</math></p> <p>Affected flags: <math>\{N\} Z \quad \{N\} C \quad \{N\} AC \quad \{N\} OV</math></p> <p>Application Example:</p> <hr style="border-top: 1px dashed black;"/> <pre>.romadr 0x10 ;           // ISR entry address     pushaf ;           // put ACC and flag into stack memory     ...                // ISR program     ...                // ISR program     popaf ;            // restore ACC and flag from stack memory     reti ;</pre> <hr style="border-top: 1px dashed black;"/>
<i>popaf</i>	<p>Restore ACC and flag from the memory which address is specified in the stack pointer.</p> <p>Example: <code>popaf;</code></p> <p>Result:    <math>sp \leftarrow sp - 2 ;</math>  <math>\{Flag, ACC\} \leftarrow [sp] ;</math></p> <p>Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>

## 7.2. Arithmetic Operation Instructions

<i>add</i> a, I	Add immediate data with ACC, then put result into ACC
-----------------	---

# IRC12x

## 1KW OTP type IR Remote Control MCU

	<p>Example: <code>add a, 0x0f ;</code>  Result: <math>a \leftarrow a + 0fh</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>add a, M</code>	<p>Add data in memory with ACC, then put result into ACC  Example: <code>add a, MEM ;</code>  Result: <math>a \leftarrow a + MEM</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>add M, a</code>	<p>Add data in memory with ACC, then put result into memory  Example: <code>add MEM, a;</code>  Result: <math>MEM \leftarrow a + MEM</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>addc a, M</code>	<p>Add data in memory with ACC and carry bit, then put result into ACC  Example: <code>addc a, MEM ;</code>  Result: <math>a \leftarrow a + MEM + C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>addc M, a</code>	<p>Add data in memory with ACC and carry bit, then put result into memory  Example: <code>addc MEM, a ;</code>  Result: <math>MEM \leftarrow a + MEM + C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>addc a</code>	<p>Add carry with ACC, then put result into ACC  Example: <code>addc a ;</code>  Result: <math>a \leftarrow a + C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>addc M</code>	<p>Add carry with memory, then put result into memory  Example: <code>addc MEM ;</code>  Result: <math>MEM \leftarrow MEM + C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>sub a, I</code>	<p>Subtraction immediate data from ACC, then put result into ACC  Example: <code>sub a, 0x0f;</code>  Result: <math>a \leftarrow a - 0fh \text{ ( } a + [2's \text{ complement of } 0fh] \text{ )}</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>sub a, M</code>	<p>Subtraction data in memory from ACC, then put result into ACC  Example: <code>sub a, MEM ;</code>  Result: <math>a \leftarrow a - MEM \text{ ( } a + [2's \text{ complement of } M] \text{ )}</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>sub M, a</code>	<p>Subtraction data in ACC from memory, then put result into memory  Example: <code>sub MEM, a;</code>  Result: <math>MEM \leftarrow MEM - a \text{ ( } MEM + [2's \text{ complement of } a] \text{ )}</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>subc a, M</code>	<p>Subtraction data in memory and carry from ACC, then put result into ACC  Example: <code>subc a, MEM;</code>  Result: <math>a \leftarrow a - MEM - C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>subc M, a</code>	<p>Subtraction ACC and carry bit from memory, then put result into memory  Example: <code>subc MEM, a ;</code>  Result: <math>MEM \leftarrow MEM - a - C</math>  Affected flags: <math>\{Y\} Z \quad \{Y\} C \quad \{Y\} AC \quad \{Y\} OV</math></p>
<code>subc a</code>	<p>Subtraction carry from ACC, then put result into ACC  Example: <code>subc a;</code>  Result: <math>a \leftarrow a - C</math></p>

# IRC12x

## 1KW OTP type IR Remote Control MCU

	Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>subc</i> M	<p>Subtraction carry from the content of memory, then put result into memory</p> <p>Example: <i>subc</i> MEM;</p> <p>Result: MEM ← MEM - C</p> <p>Affected flags: [Y] Z [Y] C [Y] AC [Y] OV</p>
<i>inc</i> M	<p>Increment the content of memory</p> <p>Example: <i>inc</i> MEM ;</p> <p>Result: MEM ← MEM + 1</p> <p>Affected flags: [Y] Z [Y] C [Y] AC [Y] OV</p>
<i>dec</i> M	<p>Decrement the content of memory</p> <p>Example: <i>dec</i> MEM;</p> <p>Result: MEM ← MEM - 1</p> <p>Affected flags: [Y] Z [Y] C [Y] AC [Y] OV</p>
<i>clear</i> M	<p>Clear the content of memory</p> <p>Example: <i>clear</i> MEM ;</p> <p>Result: MEM ← 0</p> <p>Affected flags: [N] Z [N] C [N] AC [N] OV</p>

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 7.3. Shift Operation Instructions

<i>sr a</i>	Shift right of ACC, shift 0 to bit 7 Example: <i>sr a</i> ; Result: $a(0, b7, b6, b5, b4, b3, b2, b1) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b0)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>src a</i>	Shift right of ACC with carry bit 7 to flag Example: <i>src a</i> ; Result: $a(c, b7, b6, b5, b4, b3, b2, b1) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b0)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>sr M</i>	Shift right the content of memory, shift 0 to bit 7 Example: <i>sr MEM</i> ; Result: $MEM(0, b7, b6, b5, b4, b3, b2, b1) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b0)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>src M</i>	Shift right of memory with carry bit 7 to flag Example: <i>src MEM</i> ; Result: $MEM(c, b7, b6, b5, b4, b3, b2, b1) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b0)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>sl a</i>	Shift left of ACC shift 0 to bit 0 Example: <i>sl a</i> ; Result: $a(b6, b5, b4, b3, b2, b1, b0, 0) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b7)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>slc a</i>	Shift left of ACC with carry bit 0 to flag Example: <i>slc a</i> ; Result: $a(b6, b5, b4, b3, b2, b1, b0, c) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b7)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>sl M</i>	Shift left of memory, shift 0 to bit 0 Example: <i>sl MEM</i> ; Result: $MEM(b6, b5, b4, b3, b2, b1, b0, 0) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b7)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>slc M</i>	Shift left of memory with carry bit 0 to flag Example: <i>slc MEM</i> ; Result: $MEM(b6, b5, b4, b3, b2, b1, b0, c) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b7)$ Affected flags: $\{N\} Z \{Y\} C \{N\} AC \{N\} OV$
<i>swap a</i>	Swap the high nibble and low nibble of ACC Example: <i>swap a</i> ; Result: $a(b3, b2, b1, b0, b7, b6, b5, b4) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ Affected flags: $\{N\} Z \{N\} C \{N\} AC \{N\} OV$

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 7.4. Logic Operation Instructions

<i>and</i> a, I	Perform logic AND on ACC and immediate data, then put result into ACC Example: <i>and</i> a, 0x0f ; Result: $a \leftarrow a \& 0fh$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>and</i> a, M	Perform logic AND on ACC and memory, then put result into ACC Example: <i>and</i> a, RAM10 ; Result: $a \leftarrow a \& RAM10$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>and</i> M, a	Perform logic AND on ACC and memory, then put result into memory Example: <i>and</i> MEM, a ; Result: $MEM \leftarrow a \& MEM$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>or</i> a, I	Perform logic OR on ACC and immediate data, then put result into ACC Example: <i>or</i> a, 0x0f ; Result: $a \leftarrow a   0fh$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>or</i> a, M	Perform logic OR on ACC and memory, then put result into ACC Example: <i>or</i> a, MEM ; Result: $a \leftarrow a   MEM$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>or</i> M, a	Perform logic OR on ACC and memory, then put result into memory Example: <i>or</i> MEM, a ; Result: $MEM \leftarrow a   MEM$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>xor</i> a, I	Perform logic XOR on ACC and immediate data, then put result into ACC Example: <i>xor</i> a, 0x0f ; Result: $a \leftarrow a \wedge 0fh$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>xor</i> IO, a	Perform logic XOR on ACC and IO register, then put result into IO register Example: <i>xor</i> pa, a ; Result: $pa \leftarrow a \wedge pa$ ; // pa is the data register of port A Affected flags: [N] Z [N] C [N] AC [N] OV
<i>xor</i> a, M	Perform logic XOR on ACC and memory, then put result into ACC Example: <i>xor</i> a, MEM ; Result: $a \leftarrow a \wedge RAM10$ Affected flags: [Y] Z [N] C [N] AC [N] OV
<i>xor</i> M, a	Perform logic XOR on ACC and memory, then put result into memory Example: <i>xor</i> MEM, a ; Result: $MEM \leftarrow a \wedge MEM$ Affected flags: [Y] Z [N] C [N] AC [N] OV



# IRC12x

## 1KW OTP type IR Remote Control MCU

<i>not</i> a	<p>Perform 1's complement (logical complement) of ACC</p> <p>Example: <i>not</i> a ;</p> <p>Result: <math>a \leftarrow \sim a</math></p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p> <p>Application Example:</p> <hr/> <pre> mov    a, 0x38 ;    // ACC=0X38 not    a ;          // ACC=0XC7 </pre> <hr/>
<i>not</i> M	<p>Perform 1's complement (logical complement) of memory</p> <p>Example: <i>not</i> MEM ;</p> <p>Result: <math>MEM \leftarrow \sim MEM</math></p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p> <p>Application Example:</p> <hr/> <pre> mov    a, 0x38 ; mov    mem, a ;    // mem = 0x38 not    mem ;       // mem = 0xC7 </pre> <hr/>
<i>neg</i> a	<p>Perform 2's complement of ACC</p> <p>Example: <i>neg</i> a ;</p> <p>Result: <math>a \leftarrow \overline{a}</math></p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p> <p>Application Example:</p> <hr/> <pre> mov    a, 0x38 ;    // ACC=0X38 neg    a ;          // ACC=0XC8 </pre> <hr/>
<i>neg</i> M	<p>Perform 2's complement of memory</p> <p>Example: <i>neg</i> MEM ;</p> <p>Result: <math>MEM \leftarrow \overline{MEM}</math></p> <p>Affected flags: [Y] Z [N] C [N] AC [N] OV</p> <p>Application Example:</p> <hr/> <pre> mov    a, 0x38 ; mov    mem, a ;    // mem = 0x38 neg    mem ;       // mem = 0xC8 </pre> <hr/>

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 7.5. Bit Operation Instructions

<i>set0</i> IO.n	Set bit n of IO port to low Example: <i>set0</i> pa.5 ; Result: set bit 5 of port A to low Affected flags: [N] Z [N] C [N] AC [N] OV
<i>set1</i> IO.n	Set bit n of IO port to high Example: <i>set1</i> pb.5 ; Result: set bit 5 of port B to high Affected flags: [N] Z [N] C [N] AC [N] OV
<i>set0</i> M.n	Set bit n of memory to low Example: <i>set0</i> MEM.5 ; Result: set bit 5 of MEM to low Affected flags: [N] Z [N] C [N] AC [N] OV
<i>set1</i> M.n	Set bit n of memory to high Example: <i>set1</i> MEM.5 ; Result: set bit 5 of MEM to high Affected flags: [N] Z [N] C [N] AC [N] OV

### 7.6. Conditional Operation Instructions

<i>ceqsn</i> a, I	Compare ACC with immediate data and skip next instruction if both are equal. Flag will be changed like as ( $a \leftarrow a - I$ ) Example: <i>ceqsn</i> a, 0x55 ; <i>inc</i> MEM ; <i>goto</i> error ; Result: If a=0x55, then "goto error"; otherwise, "inc MEM". Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>ceqsn</i> a, M	Compare ACC with memory and skip next instruction if both are equal. Flag will be changed like as ( $a \leftarrow a - M$ ) Example: <i>ceqsn</i> a, MEM; Result: If a=MEM, skip next instruction Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>t0sn</i> IO.n	Check IO bit and skip next instruction if it's low Example: <i>t0sn</i> pa.5; Result: If bit 5 of port A is low, skip next instruction Affected flags: [N] Z [N] C [N] AC [N] OV
<i>t1sn</i> IO.n	Check IO bit and skip next instruction if it's high Example: <i>t1sn</i> pa.5 ; Result: If bit 5 of port A is high, skip next instruction Affected flags: [N] Z [N] C [N] AC [N] OV
<i>t0sn</i> M.n	Check memory bit and skip next instruction if it's low Example: <i>t0sn</i> MEM.5 ; Result: If bit 5 of MEM is low, then skip next instruction Affected flags: [N] Z [N] C [N] AC [N] OV
<i>t1sn</i> M.n	Check memory bit and skip next instruction if it's high EX: <i>t1sn</i> MEM.5 ; Result: If bit 5 of MEM is high, then skip next instruction Affected flags: [N] Z [N] C [N] AC [N] OV
<i>izsn</i> a	Increment ACC and skip next instruction if ACC is zero

# IRC12x

## 1KW OTP type IR Remote Control MCU

	Example: <i>izsn</i> a; Result: $a \leftarrow a + 1$ , skip next instruction if $a = 0$ Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>dzn</i> a	Decrement ACC and skip next instruction if ACC is zero Example: <i>dzn</i> a; Result: $A \leftarrow A - 1$ , skip next instruction if $a = 0$ Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>izsn</i> M	Increment memory and skip next instruction if memory is zero Example: <i>izsn</i> MEM; Result: $MEM \leftarrow MEM + 1$ , skip next instruction if $MEM = 0$ Affected flags: [Y] Z [Y] C [Y] AC [Y] OV
<i>dzn</i> M	Decrement memory and skip next instruction if memory is zero Example: <i>dzn</i> MEM; Result: $MEM \leftarrow MEM - 1$ , skip next instruction if $MEM = 0$ Affected flags: [Y] Z [Y] C [Y] AC [Y] OV

### 7.7. System control Instructions

<i>call</i> label	Function call, address can be full range address space Example: <i>call</i> function1; Result: $[sp] \leftarrow pc + 1$ $pc \leftarrow \text{function1}$ $sp \leftarrow sp + 2$ Affected flags: [N] Z [N] C [N] AC [N] OV
<i>goto</i> label	Go to specific address which can be full range address space Example: <i>goto</i> error; Result: Go to error and execute program. Affected flags: [N] Z [N] C [N] AC [N] OV
<i>ret</i> I	Place immediate data to ACC, then return Example: <i>ret</i> 0x55; Result: $A \leftarrow 55h$ $ret$ ; Affected flags: [N] Z [N] C [N] AC [N] OV
<i>ret</i>	Return to program which had function call Example: <i>ret</i> ; Result: $sp \leftarrow sp - 2$ $pc \leftarrow [sp]$ Affected flags: [N] Z [N] C [N] AC [N] OV
<i>reti</i>	Return to program that is interrupt service routine. After this command is executed, global interrupt is enabled automatically. Example: <i>reti</i> ; Affected flags: [N] Z [N] C [N] AC [N] OV
<i>nop</i>	No operation Example: <i>nop</i> ; Result: nothing changed Affected flags: [N] Z [N] C [N] AC [N] OV
<i>pcadd</i> a	Next program counter is current program counter plus ACC. Example: <i>pcadd</i> a;

# IRC12x

## 1KW OTP type IR Remote Control MCU

	<p>Result: <math>pc \leftarrow pc + a</math>  Affected flags: [N] Z [N] C [N] AC [N] OV  Application Example:</p> <hr style="border-top: 1px dashed black;"/> <pre> ... mov      a, 0x02 ; pcadd    a ;           // PC &lt;- PC+2 goto     err1 ; goto     correct ;     // jump here goto     err2 ; goto     err3 ; ... correct:           // jump here ... </pre> <hr style="border-top: 1px dashed black;"/>
<i>engint</i>	<p>Enable global interrupt enable  Example: <i>engint</i>;  Result: Interrupt request can be sent to CPU  Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>disgint</i>	<p>Disable global interrupt enable  Example: <i>disgint</i>;  Result: Interrupt request is blocked from CPU  Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>stopsys</i>	<p>System halt.  Example: <i>stopsys</i>;  Result: Stop the system clocks and halt the system  Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>stopexe</i>	<p>CPU halt. The oscillator module is still active to output clock, however, system clock is disabled to save power.  Example: <i>stopexe</i>;  Result: Stop the system clocks and keep oscillator modules active.  Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>reset</i>	<p>Reset the whole chip, its operation will be same as hardware reset.  Example: <i>reset</i>;  Result: Reset the whole chip.  Affected flags: [N] Z [N] C [N] AC [N] OV</p>
<i>wdreset</i>	<p>Reset Watchdog timer.  Example: <i>wdreset</i>;  Result: Reset Watchdog timer.  Affected flags: [N] Z [N] C [N] AC [N] OV</p>

# IRC12x

## 1KW OTP type IR Remote Control MCU

### 7.8. Summary of Instructions Execution Cycle

2T		<i>goto, call, idxm, pcadd, ret, reti,</i>
2T	Condition is fulfilled.	<i>ceqsn, t0sn, t1sn, dzsn, izsn</i>
1T	Condition is not fulfilled.	
1T		Others

### 7.9. Summary of affected flags by Instructions

Instruction	Z	C	AC	OV	Instruction	Z	C	AC	OV	Instruction	Z	C	AC	OV
<i>mov a, l</i>	-	-	-	-	<i>mov M, a</i>	-	-	-	-	<i>mov a, M</i>	Y	-	-	-
<i>mov a, IO</i>	Y	-	-	-	<i>mov IO, a</i>	-	-	-	-	<i>ldt16 word</i>	-	-	-	-
<i>stt16 word</i>	-	-	-	-	<i>idxm a, index</i>	-	-	-	-	<i>idxm index, a</i>	-	-	-	-
<i>xch M</i>	-	-	-	-	<i>pushaf</i>	-	-	-	-	<i>popaf</i>	Y	Y	Y	Y
<i>add a, l</i>	Y	Y	Y	Y	<i>add a, M</i>	Y	Y	Y	Y	<i>add M, a</i>	Y	Y	Y	Y
<i>addc a, M</i>	Y	Y	Y	Y	<i>addc M, a</i>	Y	Y	Y	Y	<i>addc a</i>	Y	Y	Y	Y
<i>addc M</i>	Y	Y	Y	Y	<i>sub a, l</i>	Y	Y	Y	Y	<i>sub a, M</i>	Y	Y	Y	Y
<i>sub M, a</i>	Y	Y	Y	Y	<i>subc a, M</i>	Y	Y	Y	Y	<i>subc M, a</i>	Y	Y	Y	Y
<i>subc a</i>	Y	Y	Y	Y	<i>subc M</i>	Y	Y	Y	Y	<i>inc M</i>	Y	Y	Y	Y
<i>dec M</i>	Y	Y	Y	Y	<i>clear M</i>	-	-	-	-	<i>sra</i>	-	Y	-	-
<i>src a</i>	-	Y	-	-	<i>sr M</i>	-	Y	-	-	<i>src M</i>	-	Y	-	-
<i>sl a</i>	-	Y	-	-	<i>slc a</i>	-	Y	-	-	<i>sl M</i>	-	Y	-	-
<i>slc M</i>	-	Y	-	-	<i>swap a</i>	-	-	-	-	<i>and a, l</i>	Y	-	-	-
<i>and a, M</i>	Y	-	-	-	<i>and M, a</i>	Y	-	-	-	<i>or a, l</i>	Y	-	-	-
<i>or a, M</i>	Y	-	-	-	<i>or M, a</i>	Y	-	-	-	<i>xor a, l</i>	Y	-	-	-
<i>xor IO, a</i>	-	-	-	-	<i>xor a, M</i>	Y	-	-	-	<i>xor M, a</i>	Y	-	-	-
<i>not a</i>	Y	-	-	-	<i>not M</i>	Y	-	-	-	<i>neg a</i>	Y	-	-	-
<i>neg M</i>	Y	-	-	-	<i>set0 IO.n</i>	-	-	-	-	<i>set1 IO.n</i>	-	-	-	-
<i>set0 M.n</i>	-	-	-	-	<i>set1 M.n</i>	-	-	-	-	<i>ceqsn a, l</i>	Y	Y	Y	Y
<i>ceqsn a, M</i>	Y	Y	Y	Y	<i>t0sn IO.n</i>	-	-	-	-	<i>t1sn IO.n</i>	-	-	-	-
<i>t0sn M.n</i>	-	-	-	-	<i>t1sn M.n</i>	-	-	-	-	<i>izsn a</i>	Y	Y	Y	Y
<i>dzsn a</i>	Y	Y	Y	Y	<i>izsn M</i>	Y	Y	Y	Y	<i>dzsn M</i>	Y	Y	Y	Y
<i>call label</i>	-	-	-	-	<i>goto label</i>	-	-	-	-	<i>ret l</i>	-	-	-	-
<i>ret</i>	-	-	-	-	<i>reti</i>	-	-	-	-	<i>nop</i>	-	-	-	-
<i>pcadd a</i>	-	-	-	-	<i>engint</i>	-	-	-	-	<i>disgint</i>	-	-	-	-
<i>stopsys</i>	-	-	-	-	<i>stopexe</i>	-	-	-	-	<i>reset</i>	-	-	-	-
<i>wdreset</i>	-	-	-	-										

### 7.10. BIT definition

Bit defined: Only addressed at 0x00 ~ 0x0F.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 8. Code Options

Option	Selection	Description
Security	Enable	OTP content is protected 7/8 words
	Disable	OTP content is not protected so program can be read back
Boot-up_Time	Slow	Slow boot up
	Fast	Fast boot up

### 9. Special Notes

This chapter is to remind user who use IRC12x series IC in order to avoid frequent errors upon operation.

#### 9.1. Using IC

##### 9.1.1. IO pin usage and setting

(1) IO pin is set to be digital input

- ◆ When IO is set as digital input, the level of  $V_{ih}$  and  $V_{il}$  would changes with the voltage and temperature. Please follow the minimum value of  $V_{ih}$  and the maximum value of  $V_{il}$ .
- ◆ The value of internal pull high resistor would also changes with the voltage, temperature and pin voltage. It is not the fixed value.

(2) If IO pin is set to be digital input and enable wake-up function

- ◆ Configure IO pin as input.
- ◆ Set corresponding bit to "1" in PADIER and PBDIER.

(3) PA5 is set to be output pin

- ◆ PA5 can be set to be Open-Drain output pin only, output high requires adding pull-high resistor.

(4) PA5 is set to be PRST

- ◆ Configure PA5 as input.
- ◆ Set CLKMD.0=1 to enable PA5 as PRST# input pin.

(5) PA5 is set to be input pin and to connect with a push button or a switch by a long wire

- ◆ Needs to put a  $>33\Omega$  resistor in between PA5 and the long wire.
- ◆ Avoid using PA5 as input in such application.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 9.1.2. Interrupt

(1) When using the interrupt function, the procedure should be:

Step1: Set INTEN register, enable the interrupt control bit

Step2: Clear INTRQ register

Step3: In the main program, using ENGINT to enable CPU interrupt function

Step4: Wait for interrupt. When interrupt occurs, enter to Interrupt Service Routine

Step5: After the Interrupt Service Routine being executed, return to the main program

\* Use DISGINT in the main program to disable all interrupts

\* When interrupt service routine starts, use PUSHAF instruction to save ALU and FLAG register. POPAF instruction is to restore ALU and FLAG register before RETI as below:

```
void Interrupt (void)    // Once the interrupt occurs, jump to interrupt service routine
{
    // Enter DISGINT status automatically, no more interrupt is
    // accepted

    PUSHAF;
    ...
    POPAF;
} // RETI will be added automatically. After RETI being executed, ENGINT status
// will be restored
```

(2) INTEN and INTRQ have no initial values. Please set required value before enabling interrupt function.

### 9.1.3. System clock switching

System clock can be switched by CLKMD register. Please notice that, NEVER switch the system clock and turn off the original clock source at the same time. For example: When switching from clock A to clock B, please switch to clock B first; and after that turn off the clock A oscillator through CLKMD.

- ◆ Example : Switch system clock from ILRC to IHRC/8  
CLKMD = 0x34; // switch to IHRC, ILRC can not be disabled here  
CLKMD.2 = 0; // ILRC can be disabled at this time
- ◆ **ERROR:** Switch ILRC to IHRC/8 and turn off ILRC simultaneously  
CLKMD = 0x30; // MCU will hang

### 9.1.4. Power-Down mode, Wake-up and Watchdog

Watchdog will be inactive once ILRC is disabled.

# IRC12x

## 1KW OTP type IR Remote Control MCU

---

### 9.1.5. TIMER time out

When select \$ INTEGS BIT\_R (default value) and T16M counter BIT8 to generate interrupt, if T16M counts from 0, the first interrupt will occur when the counter reaches to 0x100 (BIT8 from 0 to 1) and the second interrupt will occur when the counter reaches 0x300 (BIT8 from 0 to 1). Therefore, selecting BIT8 as 1 to generate interrupt means that the interrupt occurs every 512 counts. Please notice that if T16M counter is restarted, the next interrupt will occur once Bit8 turns from 0 to 1.

If select \$ INTEGS BIT\_F (BIT triggers from 1 to 0) and T16M counter BIT8 to generate interrupt, the T16M counter changes to an interrupt every 0x200/0x400/0x600/. Please pay attention to two differences with setting INTEGS methods.

### 9.1.6. IHRC

- (1) The IHRC frequency calibration is performed when IC is programmed by the writer.
- (2) Because the characteristic of the Epoxy Molding Compound (EMC) would some degrees affects the IHRC frequency (either for package or COB), if the calibration is done before molding process, the actual IHRC frequency after molding may be deviated or becomes out of spec. Normally , the frequency is getting slower a bit.
- (3) It usually happens in COB package or Quick Turnover Programming (QTP). This company would not take any responsibility for this situation.
- (4) Users can make some compensatory adjustments according to their own experiences. For example, users can set IHRC frequency to be 0.5% ~ 1% higher and aim to get better re-targeting after molding.

### 9.1.7. Program writing of IRC12x

Please use 5S-P-003 for program writing. 3S-P-002 or previous version doesn't support writing IRC12x. For details, please refer to 5S-P-003-UM and connect the jumper.

◆ Special notes about voltage and current while Multi-Chip-Package(MCP) or On-Board Programming

- (1) PA5 ( $V_{PP}$ ) may be higher than 8.5V.
- (2)  $V_{DD}$  may be higher than 7V, and its maximum current may reach about 20mA.
- (3) All other signal pins level (except GND) are the same as  $V_{DD}$ .

User should confirm when using this product in MCP or On-Board Programming, the peripheral circuit or components will not be destroyed or limit the above voltages.

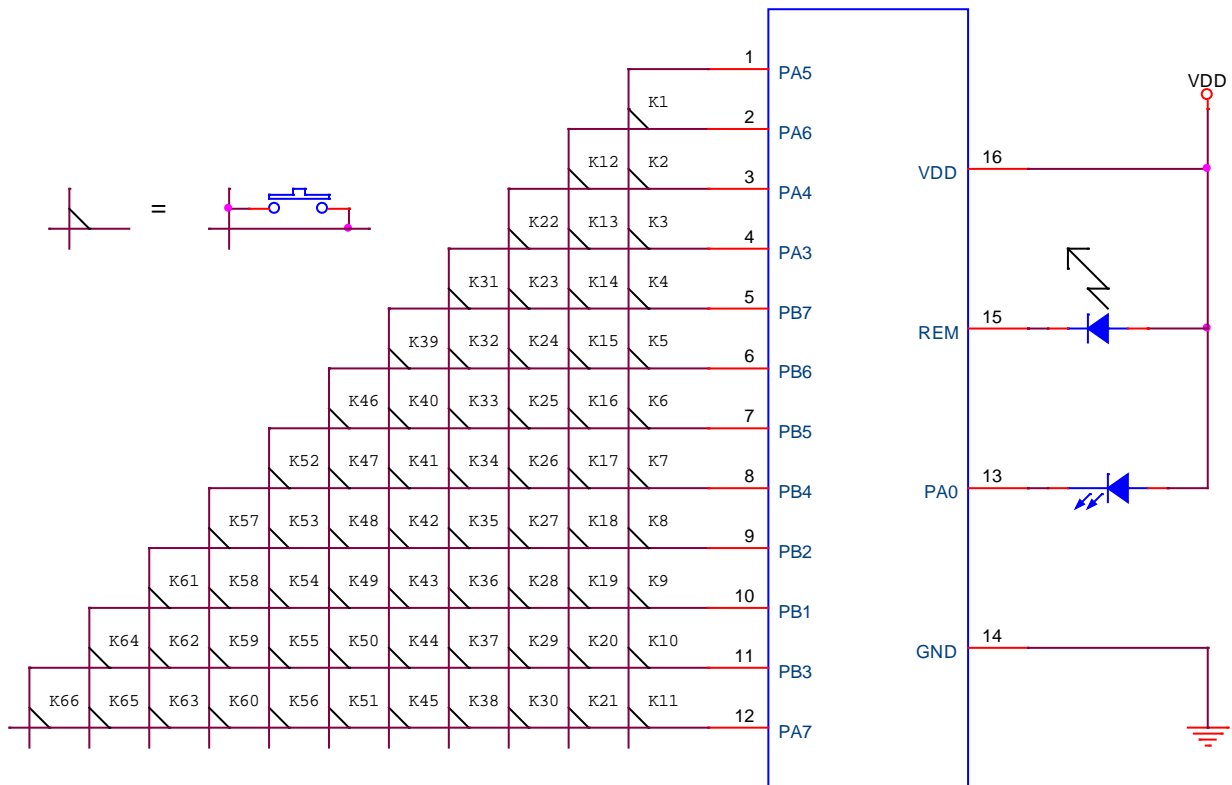


# IRC12x

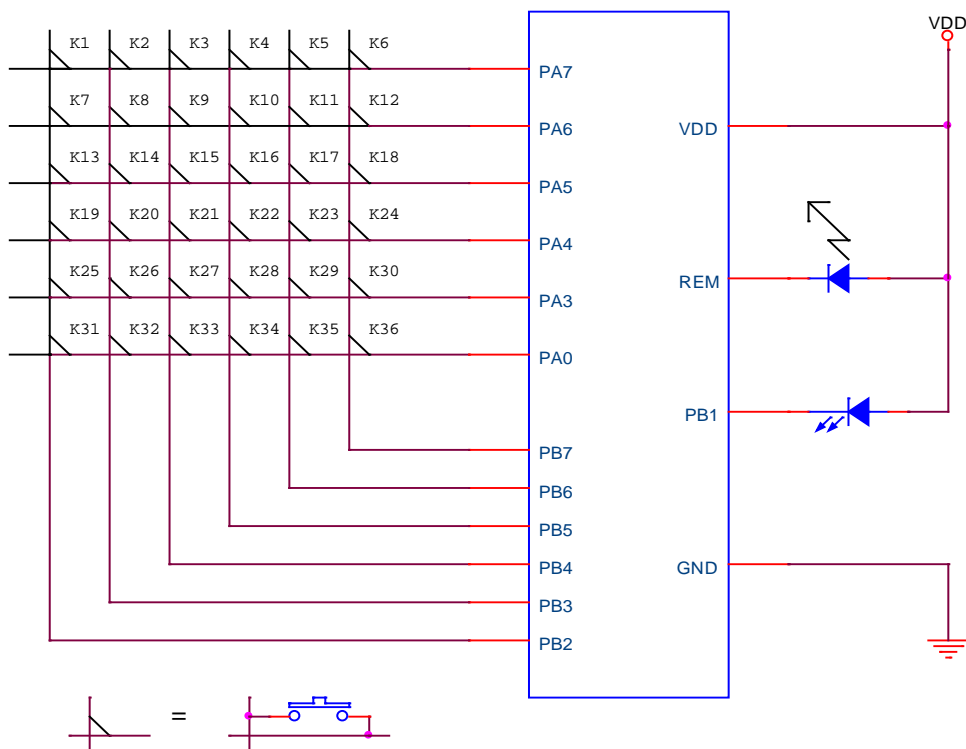
## 1KW OTP type IR Remote Control MCU

### 9.1.8 Application Schematic for IRC12x

(1) 66 keys with LED indicator, for reference only.



(2) 36 keys with LED indicator (SOP16x)

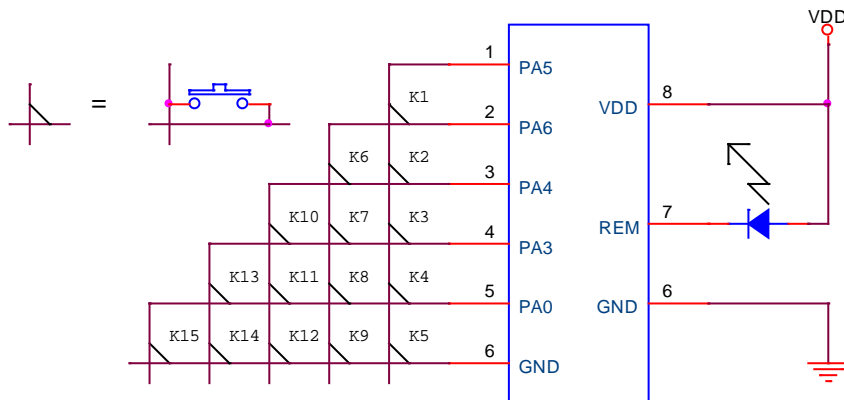


# IRC12x

## 1KW OTP type IR Remote Control MCU

---

(3) 15 keys Schematic, for reference only.



◆ **Special notes about drawing PCB board**

- (1) Avoid too long power lines and ground lines.
- (2) It is recommended that the power line of the emission circuit and IC power line should be separated or thickened.

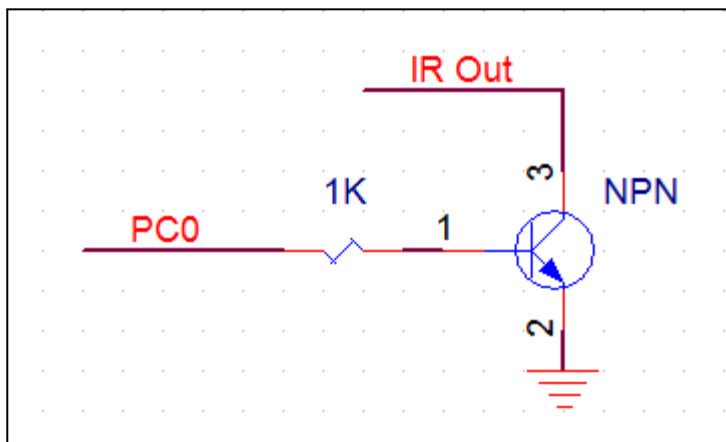
# IRC12x

## 1KW OTP type IR Remote Control MCU

### 9.2 Using ICE

(1) Please use 5S-I-S01/2(B) to emulate of IRC12x function except as the list below:

- ICE doesn't support SYSCLK = ILRC/2, ILRC/8 and IHRC/64
- ICE doesn't support PB6 as the source of interrupt
- Emulator rewrites and replaces all functions, like IRTMC / IRTMB / TSCNC / PAPH / PBPH / PATS / PBTS / PADIER / PBDIER / STOPEXE. We expect to be able to simulate Key Scan and IR functions, but as a result, it will cause slower execution timing.
- Output REM (IR Out) of IRTMB / IRTMC can simulate inverse output for PC0 with connection NPN by simulator.



- Fast wakeup time is different from emulator : 128 SysClk, IC : 32 ILRC
- Watchdog time out period is different from emulate.

WDT period	5S-I-S01/2(B)	IRC12x
misc[1:0]=00	$2k * T_{ILRC}$	$4k * T_{ILRC}$
misc[1:0]=01	$4k * T_{ILRC}$	$8k * T_{ILRC}$
misc[1:0]=10	$16k * T_{ILRC}$	$32k * T_{ILRC}$
misc[1:0]=11	$256 * T_{ILRC}$	$128k * T_{ILRC}$

ICE cycle	IO = A	A = IO	IO. bit = 0/1	If (IO. bit)
PADIER	6	-	-	-
PBDIER	7	-	-	-
TSCNC	8	1	1	1 or 2
PATS	1	1	1	1 or 2
PBTS	6	1	1	1 or 2
IRTMC	67	1	67	1 or 2
IRTMB	67	-	67	-
PB.0	-	-	67	-
PADIER	6	-	-	-