PMS132

12-bit ADC Enhanced Controller

Datasheet

Version 0.02 – Nov. 23, 2017

10F-2, No. 1, Sec. 2, Dong-Da Road, Hsin-Chu 300, Taiwan, R.O.C.

TEL: 886-3-532-7598 www.padauk.com.tw

IMPORTANT NOTICE

PADAUK Technology reserves the right to make changes to its products or to terminate production of its products at any time without notice. Customers are strongly recommended to contact PADAUK Technology for the latest information and verify whether the information is correct and complete before placing orders.

PADAUK Technology products are not warranted to be suitable for use in life-support applications or other critical applications. PADAUK Technology assumes no liability for such applications. Critical applications include, but are not limited to, those which may involve potential risks of death, personal injury, fire or severe property damage.

PADAUK Technology assumes no responsibility for any issue caused by a customer's product design. Customers should design and verify their products within the ranges guaranteed by PADAUK Technology. In order to minimize the risks in customers' products, customers should design a product with adequate operating safeguards.

PMS132 is NOT designed for AC RC step-down powered, high power ripple or high EFT requirement application. Please do NOT apply PMS132 to those application products.

Table of content

Revision History:

Revision	Date	Description
0.00	2016/08/08	Preliminary version
0.01	2016/12/01	1. Revise Chapter 3: Pin Assignment and Description 2. Revise Section 5.8.2: Hardware and Timing Diagram 3. Revise Section 5.8.3: Frequency of PWM Output 4. Add Section 5.8: 11-bit PWM Generator Functional Description 5. Add Section 6.43~6.51: IO Registers
0.02	2017/11/23	1. Revise Section 1.2 System Features 2. Add Section 1.4 Package Information 3. Revise Chapter 2 General Description and Block Diagram 4. Add Chapter 3 Pin Assignment and Description:PMS132-S08,PMS132-U06 5. Revise Section 4.1 AC/DC Device Characteristics: “V_{IL}” and “V_{IH}” 6. Revise Section 4.3 Typical ILRC frequency vs. VDD and temperature 7. Revise Section 4.4 Typical IHRC frequency deviation vs. VDD 8. Revise Section 4.5 Typical ILRC Frequency vs. Temperature 9. Revise Section 4.6 Typical IHRC Frequency vs. Temperature 10. Revise Section 4.7 Typical operating current vs. VDD @ system clock = ILRC/n 11. Revise Section 4.8 Typical operating current vs. VDD @ system clock = IHRC/n 12. Revise Section 4.9 Typical operating current vs.VDD@system clock=4MHz EOSC / n 13. Revise Section 4.10 Typical operating current vs.VDD@system clock=32KHz EOSC / n 14. Revise Section 4.11 Typical operating current vs.VDD@system clock=1MHz EOSC / n 15. Revise Section 4.12 Typical IO driving current (I_{OH}) and sink current (I_{OL}) 16. Revise Section 4.13 Typical IO input high/low threshold voltage (V_{IH}/V_{IL}) 17. Revise Section 4.14 Typical resistance of IO pull high device 18. Add Section 4.15 Typical power down current (I_{PD}) and power save current (I_{PS}) 19. Revise Section 5.1 Program Memory - OTP 20. Revise Table 2: Three oscillation circuits 21. Revise Section 5.4.3 IHRC Frequency Calibration and System Clock 22. Revise Section 5.4.4 External Crystal Oscillator 23. Revise Fig. 3: Options of System Clock 24. Revise Section 5.5.2 Using the comparator 25. Revise Section 5.6 16-bit Timer 26. Revise Section 5.8 11-bit PWM Generator 27. Revise Section 5.11.1 Power-Save mode 28. Revise Fig. 20: ADC Block Diagram 29. Revise Section 5.14.4 Configure the analog pin 30. Revise Section 5.14.5 Using the ADC 31. Revise Section 6.3 Clock Mode Register 32. Revise Section 6.4 Interrupt Enable Register 33. Revise Section 6.5 Interrupt Request Register 34. Revise Section 6.11 Port A Digital Input Enable Register 35. Revise Section 6.12 Port B Digital Input Enable Register 36. Delete Section 6.13 MISC2 Register 37. Revise Section 6.21 Comparator Control Register 38. Revise Section 6.28 PWMG0 control Register 39. Revise Section 6.29 PWMG0 Counter Upper Bound High Register

		40. Revise Section 6.30 PWMG0 Counter Upper Bound Low Register 41. Revise Section 6.31 PWMG0 Duty Value High Register 42. Revise Section 6.32 PWMG0 Duty Value Low Register 43. Revise Section 6.38 ADC Control Register 44. Revise Section 6.40 ADC Regulator Control Register 45. Revise Section 6.43 PWMG1 control Register 46. Revise Section 6.49 PWMG2 control Register 47. Delete Chapter 7: Instructions symbol “word” and “pc0” 48. Revise Section 7.5 “*swapc* IO.n” Bit Operation Instruction 49. Add Chapter 8 Code Options 50. Revise Section 9.2.1 IO pin usage and setting 51. Revise Section 9.2.3 System clock switching 52. Add Section 9.2.6 IHRC 53. Revise Section 9.2.7 LVR 54. Revise Section 9.2.8 The result of Comparator controls the PWM output pins 55. Revise Section 9.2.10 BIT definition 56. Revise Section 9.2.11 Programming the PMS132 57. Revise Section 9.3 Using ICE 58. Revise all PWMG registers from ” R/W” to “WO”

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

- 2KW OTP program memory
- 128 Bytes data RAM
- One hardware 16-bit timer
- Two hardware 8-bit timers with PWM generation
- Three hardware 11-bit PWM generators (PWMG0, PWMG1 & PWMG2)
- Provide 1T 8x8 hardware multiplier
- 14 IO pins with optional pull-high resistor
- Every IO pin can be configured to enable wake-up function
- Band-gap circuit to provide 1.20V reference voltage
- Up to 12-channel 12-bit resolution ADC
- Provide ADC reference high voltage: external input, internal V_{DD}, Band-gap(1.20V), 4V, 3V, 2V
- Clock sources: internal high RC oscillator, internal low RC oscillator and external crystal oscillator
- For every wake-up enabled IO, two optional wake-up speed are supported: normal and fast
- Eight levels of LVR reset: 4.0V, 3.5V, 3.0V, 2.75V, 2.5V, 2.2V, 2.0V, 1.8V
- Four selectable external interrupt pins

1.3. CPU Features

- One processing unit operating mode
- 87 powerful instructions
- Most instructions are 1T execution cycle
- Programmable stack pointer to provide adjustable stack level
- Direct and indirect addressing modes for data and instructions
- All data memories are available for use as an index pointer
- Separated IO and memory space

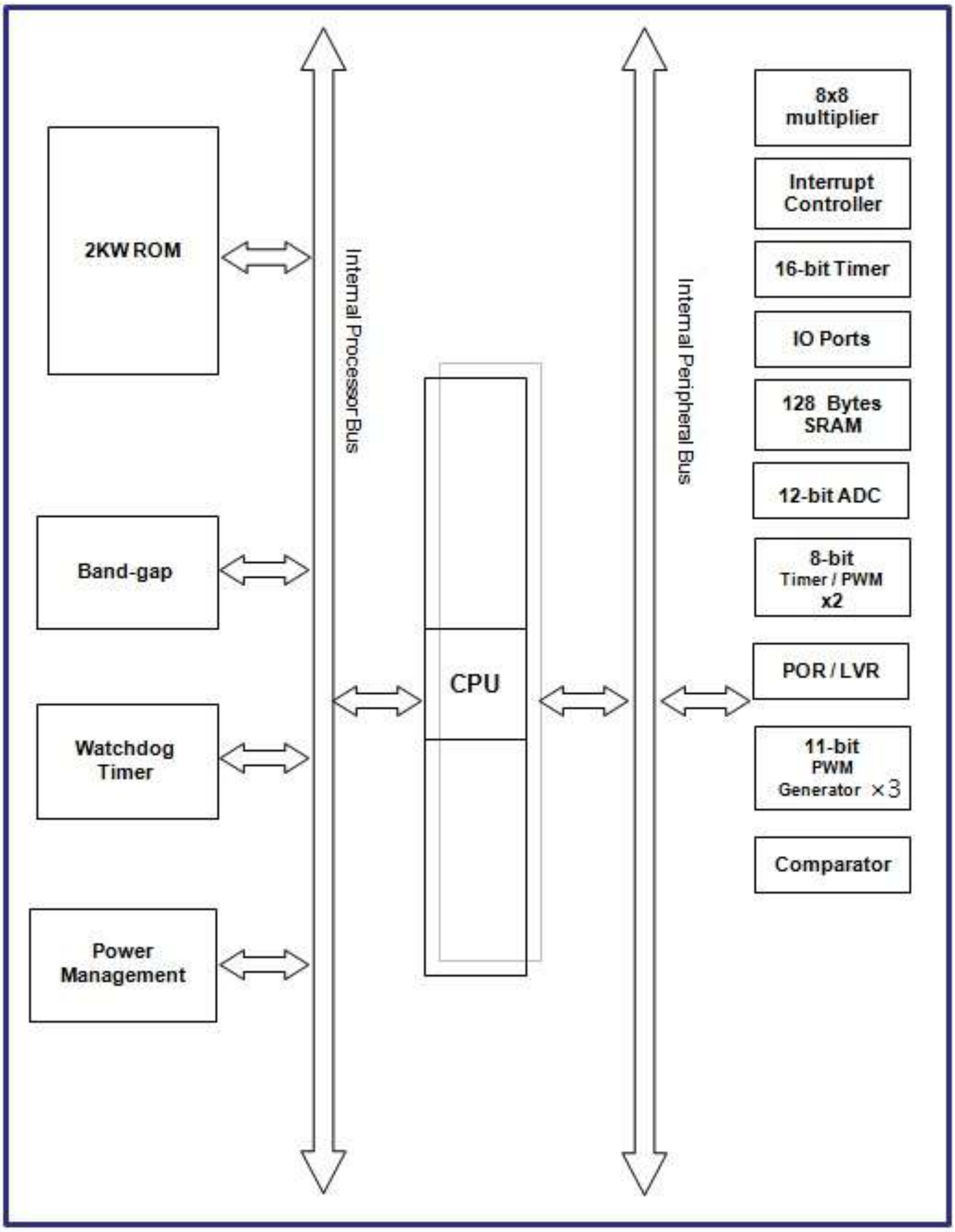
1.4. Package Information

- PMS132-U06: SOT23-6 (60mil);
- PMS132-S08: SOP8 (150mil);
- PMS132-M10: MSOP10 (118mil);
- PMS132-4N10: DFN3*3-10P (0.5pitch);
- PMS132-S14: SOP14 (150mil);
- PMS132-S16A: SOP16A (150mil);
- PMS132-S16B: SOP16B (150mil);
- PMS132-2J16A: QFN4*4-16P (0.65pitch);
- PMS132-1J16A: QFN3*3-16P (0.5pitch)

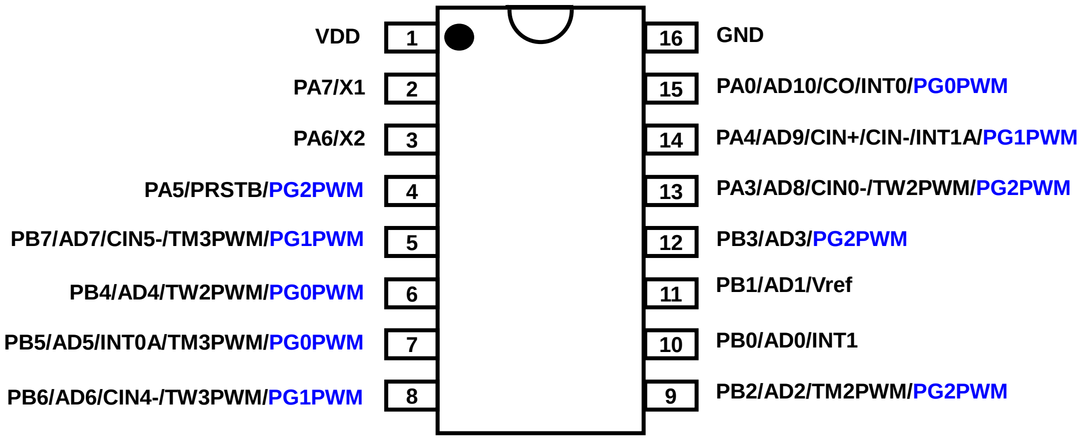
2. General Description and Block Diagram

The PMS132 family is an ADC-Type, fully static, OTP-based CMOS 8-bit microcontroller. It employs RISC architecture and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access.

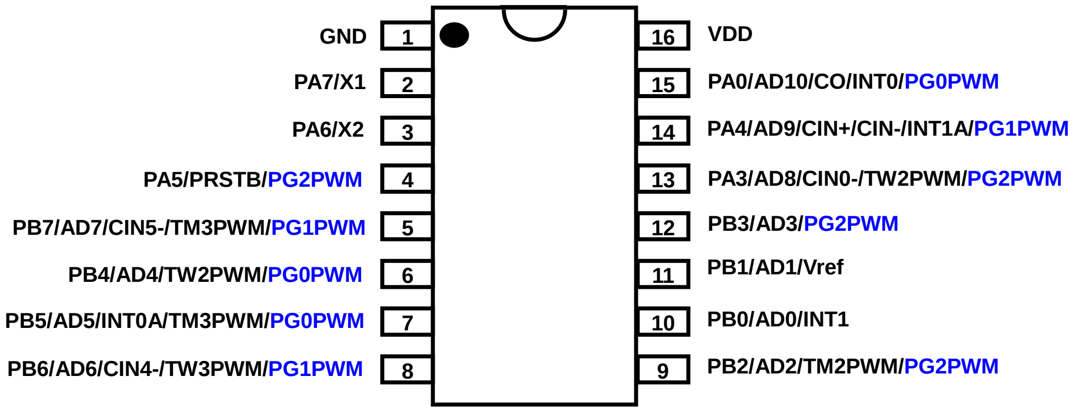
2KW bits OTP program memory and 128 bytes data SRAM are inside, one up to 12 channels 12-bit ADC is built inside the chip with one channel for internal band-gap reference voltage or 0.25*V_{DD}. PMS132 also provides six hardware timers: one is 16-bit timer, two are 8-bit timers with PWM generation, and three hardware 11-bit timers with PWM generation are also included.



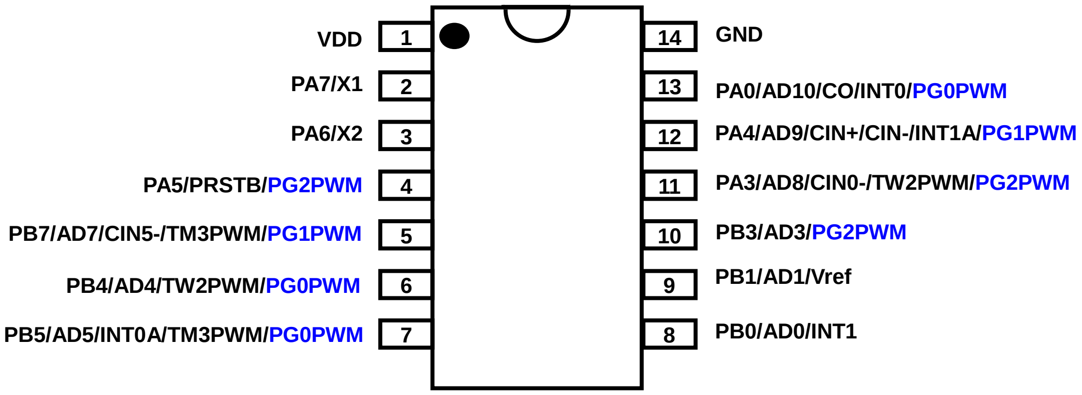
3. Pin Assignment and Description


VDD 1
PA7/X1 2
PA6/X2 3
PA5/PRSTB/PG2PWM 4
PB7/AD7/CIN5-/TM3PWM/PG1PWM 5
PB4/AD4/TW2PWM/PG0PWM 6
PB5/AD5/INT0A/TM3PWM/PG0PWM 7
PB6/AD6/CIN4-/TW3PWM/PG1PWM 8
16 GND
15 PA0/AD10/CO/INT0/PG0PWM
14 PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
13 PA3/AD8/CIN0-/TW2PWM/PG2PWM
12 PB3/AD3/PG2PWM
11 PB1/AD1/Vref
10 PB0/AD0/INT1
9 PB2/AD2/TM2PWM/PG2PWM

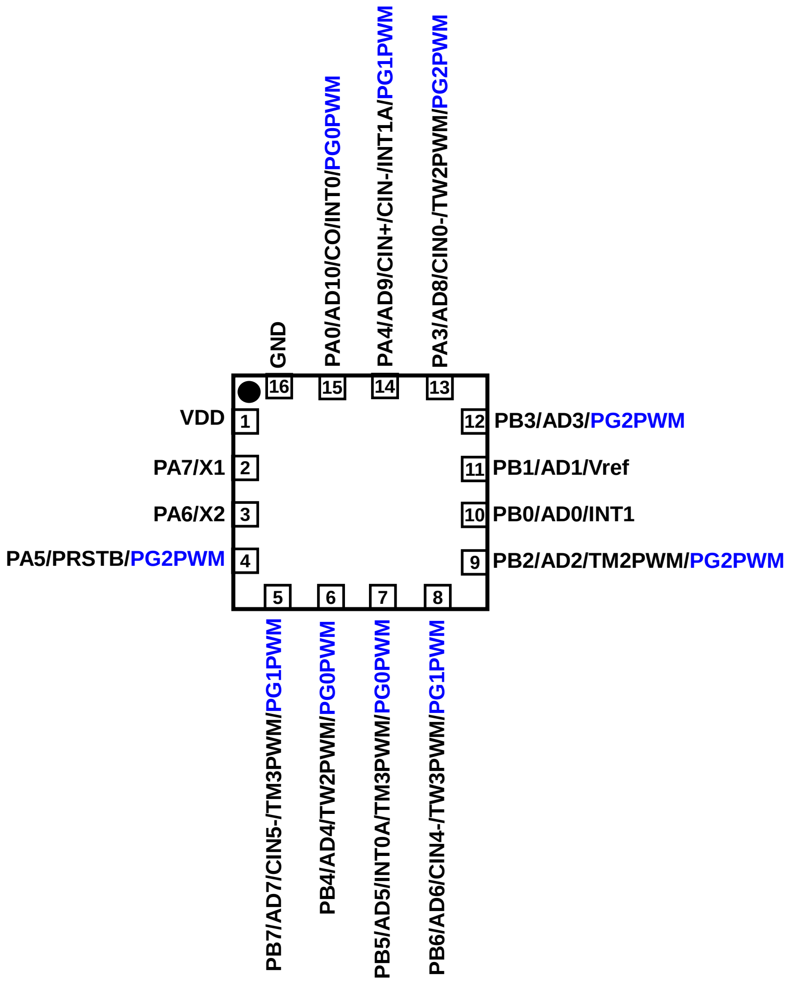
PMS132-S16A (SOP16A-150mil)


GND 1
PA7/X1 2
PA6/X2 3
PA5/PRSTB/PG2PWM 4
PB7/AD7/CIN5-/TM3PWM/PG1PWM 5
PB4/AD4/TW2PWM/PG0PWM 6
PB5/AD5/INT0A/TM3PWM/PG0PWM 7
PB6/AD6/CIN4-/TW3PWM/PG1PWM 8
16 VDD
15 PA0/AD10/CO/INT0/PG0PWM
14 PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
13 PA3/AD8/CIN0-/TW2PWM/PG2PWM
12 PB3/AD3/PG2PWM
11 PB1/AD1/Vref
10 PB0/AD0/INT1
9 PB2/AD2/TM2PWM/PG2PWM

PMS132-S16B (SOP16B-150mil)

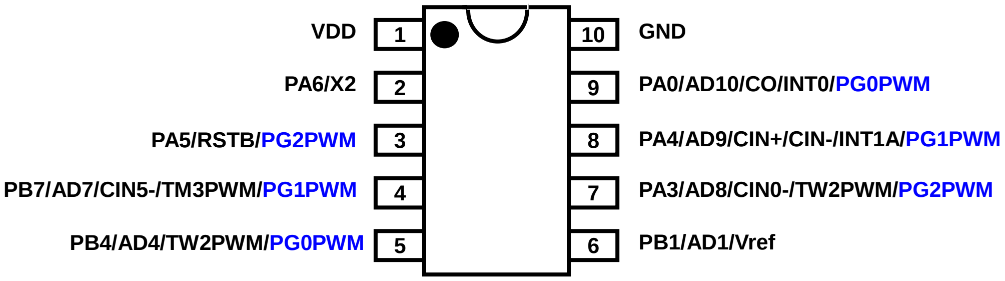

VDD 1
PA7/X1 2
PA6/X2 3
PA5/PRSTB/PG2PWM 4
PB7/AD7/CIN5-/TM3PWM/PG1PWM 5
PB4/AD4/TW2PWM/PG0PWM 6
PB5/AD5/INT0A/TM3PWM/PG0PWM 7
14 GND
13 PA0/AD10/CO/INT0/PG0PWM
12 PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
11 PA3/AD8/CIN0-/TW2PWM/PG2PWM
10 PB3/AD3/PG2PWM
9 PB1/AD1/Vref
8 PB0/AD0/INT1

PMS132-S14 (SOP14-150mil)

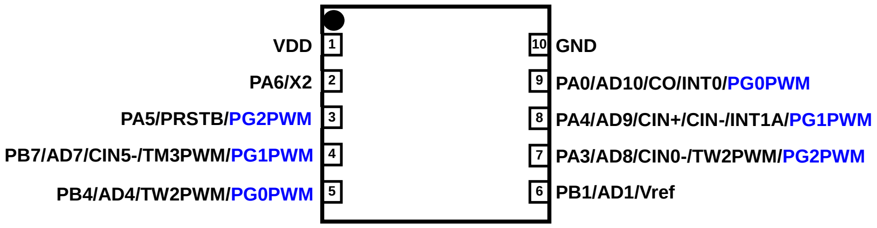


PMS132-2J16A(QFN4*4-16P-0.65pitch)

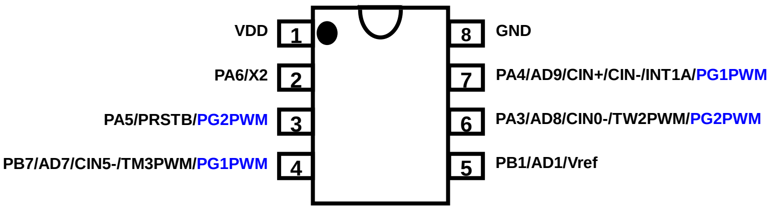
PMS132-1J16A(QFN3*3-16P-0.5pitch)


VDD 1
PA6/X2 2
PA5/RSTB/PG2PWM 3
PB7/AD7/CIN5-/TM3PWM/PG1PWM 4
PB4/AD4/TW2PWM/PG0PWM 5
10 GND
9 PA0/AD10/CO/INT0/PG0PWM
8 PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
7 PA3/AD8/CIN0-/TW2PWM/PG2PWM
6 PB1/AD1/Vref

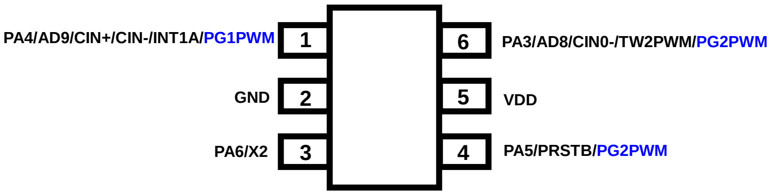
PMS132-M10 (MSOP10-118mil)


VDD 1
PA6/X2 2
PA5/PRSTB/PG2PWM 3
PB7/AD7/CIN5-/TM3PWM/PG1PWM 4
PB4/AD4/TW2PWM/PG0PWM 5
10 GND
9 PA0/AD10/CO/INT0/PG0PWM
8 PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
7 PA3/AD8/CIN0-/TW2PWM/PG2PWM
6 PB1/AD1/Vref

PMS132-4N10 (DFN3*3-10P-0.5pitch)


VDD
1
PA6/X2
2
PA5/PRSTB/PG2PWM
3
PB7/AD7/CIN5-/TM3PWM/PG1PWM
4
GND
8
PA4/AD9/CIN+/CIN-/INT1A/PG1PWM
7
PA3/AD8/CIN0-/TW2PWM/PG2PWM
6
PB1/AD1/Vref
5

PMS132-S08 (SOP8-150mil)



PMS132-U06 (SOT23-6 60mil)

Pin Description

Pin Name	Pin Type & Buffer Type	Description
PA7 / X1	IO ST / CMOS	The functions of this pin can be: (1) Bit 7 of port A. It can be configured as input or output with pull-up resistor. (2) X1 when crystal oscillator is used. If this pin is used for crystal oscillator, bit 7 of ***padier*** register must be programmed "0" to avoid leakage current. This pin can be used to wake-up system during sleep mode; however, wake-up function is also disabled if bit 7 of ***padier*** register is "0".
PA6 / X2	IO ST / CMOS	The functions of this pin can be: (1) Bit 6 of port A. It can be configured as input or output with pull-up resistor. (2) X2 when crystal oscillator is used. If this pin is used for crystal oscillator, bit 6 of ***padier*** register must be programmed "0" to avoid leakage current. This pin can be used to wake-up system during sleep mode; however, wake-up function is also disabled if bit 6 of ***padier*** register is "0".
PA5 / PRSTB / PG2PWM	IO (OD) 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 pin (2) Hardware reset (3) Output of 11-bit PWM generator PWMG2 This pin can be used to wake-up system during sleep mode; however, wake-up function is also disabled if bit 5 of ***padier*** register is "0". Please put 33Ω resistor in series to have high noise immunity when this pin is in input mode.

Pin Name	Pin Type & Buffer Type	Description
PA4 / AD9 / CIN+ / CIN1- / INT1A / PG1PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 4 of port A. It can be configured as digital input, two-state output with pull-up resistor by software independently (2) Channel 9 of ADC analog input (3) Plus input source of comparator (4) Minus input source 1 of comparator (5) External interrupt line 1A. It can be used as an external interrupt line 1. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting (6) Output of 11-bit PWM generator PWMG1 When this pin is configured as analog input, please use bit 4 of register ***padier*** to disable the digital input to prevent current leakage. The bit 4 of ***padier*** register can be set to “0” to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PA3 / AD8 / CIN0- / TM2PWM / PG2PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 3 of port A. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 8 of ADC analog input (3) Minus input source 0 of comparator (4) PWM output from Timer2 (5) Output of 11-bit PWM generator PWMG2 When this pin is configured as analog input, please use bit 3 of register ***padier*** to disable the digital input to prevent current leakage. The bit 3 of ***padier*** register can be set to “0” to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PA0 / AD10 / CO / PG0PWM / INT0	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 0 of port A. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 10 of ADC analog input (3) Output of comparator (4) Output of 11-bit PWM generator PWMG0 (5) External interrupt line 0. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting The bit 0 of ***padier*** register can be set to “0” to disable wake-up from power-down by toggling this pin.

Pin Name	Pin Type & Buffer Type	Description
PB7 / AD7 / CIN5- / TM3PWM/ PG1PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 7 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 7 of ADC analog input (3) Minus input source 5 of comparator (4) PWM output from Timer3 (5) Output of 11-bit PWM generator PWMG1 When this pin is configured as analog input, please use bit 7 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 7 of ***pbdier*** register can be set to “0” to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB6 / AD6 / CIN4- / TM3PWM/ PG1PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 6 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 6 of ADC analog input (3) Minus input source 4 of comparator. (4) PWM output from Timer3 (5) Output of 11-bit PWM generator PWMG1 When this pin is configured as analog input, please use bit 6 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 6 of ***pbdier*** register can be set to “0” to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB5 / AD5 / TM3PWM / PG0PWM / INT0A	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 5 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 5 of ADC analog input (3) PWM output from Timer3 (4) Output of 11-bit PWM generator PWMG0. (5) External interrupt line 0A. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting. When this pin is configured as analog input, please use bit 5 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 5 of ***pbdier*** register can be set to “0” to disable digital input; wake-up from power-down by toggling this pin is also disabled.

Pin Name	Pin Type & Buffer Type	Description
PB4 / AD4 / TM2PWM / PG0PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 4 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 4 of ADC analog input (3) PWM output from Timer2 (4) Output of 11-bit PWM generator PWMG0. When this pin is configured as analog input, please use bit 4 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 4 of ***pbdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB3 / AD3 / PG2PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 3 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 3 of ADC analog input (3) Output of 11-bit PWM generator PWMG2 When this pin is configured as analog input, please use bit 3 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 3 of ***pbdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB2 / AD2 / TM2PWM / PG2PWM	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 2 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 2 of ADC analog input (3) PWM output from Timer2 (4) Output of 11-bit PWM generator PWMG2 When this pin is configured as analog input, please use bit 2 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 2 of ***pbdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB1 / AD1 / Vref	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 1 of port B. It can be configured as digital input, two-state output with pull-up resistor independently by software (2) Channel 1 of ADC analog input (3) External reference high voltage for ADC. When this pin is configured as analog input, please use bit 1 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 1 of ***pbdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.

Pin Name	Pin Type & Buffer Type	Description
PB0 / AD0 / INT1	IO ST / CMOS / Analog	The functions of this pin can be: (1) Bit 0 of port B. It can be configured as analog input, digital input, and two-state output mode with pull-up resistor independently by software. (2) Channel 0 of ADC analog input. (3) External interrupt line 1. It can be used as an external interrupt line 1. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting. When this pin acts as analog input, please use bit 0 of register ***pbdier*** to disable the digital input to prevent current leakage. If bit 0 of ***pbdier*** register is set to "0" to disable digital input, wake-up from power-down by toggling this pin is also disabled.
VDD	VDD	Positive power
GND	GND	Ground
Notes: **IO**: Input/Output; **ST**: Schmitt Trigger input; **OD**: Open Drain; **Analog**: Analog input pin **CMOS**: CMOS voltage level		

4. Device Characteristics

4.1. AC/DC Device Characteristics

All data are acquired under the conditions of V_{DD}=5.0V, f_{SYS} =2MHz unless noted.

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25^{o}C)
V_{DD}	Operating Voltage	2.2	5.0	5.5	V	
LVR%	Low Voltage Reset Tolerance	-5		5	%	
f_{SYS}	System clock (CLK)* = IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0	 55K	 8M 4M 2M	Hz	 V_{DD} ≧ 3.5V V_{DD} ≧ 2.5V V_{DD} ≧ 2.2V V_{DD} =5.0V
I_{OP}	Operating Current		1 15		mA uA	f_{SYS}=IHRC/16=1MIPS@5.0V f_{SYS}=ILRC=55KHz@3.3V
I_{PD}	Power Down Current (by ***stopsys*** command)		1 0.6		uA uA	f_{SYS}= 0Hz, V_{DD} =5.0V f_{SYS}= 0Hz, V_{DD} =3.3V
I_{PS}	Power Save Current (by ***stopexe*** command)		5		uA	V_{DD} =5.0V; f_{SYS}= ILRC Only ILRC module is enabled.
V_{IL}	Input low voltage for IO lines	0 0		0.1 V_{DD} 0.2 V_{DD}	V	PA5 Others IO
V_{IH}	Input high voltage for IO lines	0.8 V_{DD} 0.7 V_{DD}		V_{DD} V_{DD}	V	PA5 Others IO
I_{OL}	IO lines sink current PA5 PA0, PA3, PA4 PA6, PA7, PB0, PB1, PB3 PB2, PB5, PB6 PB4, PB7 (Normal) PB4, PB7 (Low)		 23 20 13 13 40 20		mA	V_{DD}=5.0V, V_{OL}=0.5V
I_{OH}	IO lines drive current PA5 PB4, PB7 (Normal) PB4, PB7 (Low) Others IO		 0 -20 -10 -10		mA	V_{DD}=5.0V, V_{OH}=4.5V
V_{IN}	Input voltage	-0.3		V_{DD} +0.3	V	
$I_{INJ\ (PIN)}$	Injected current on pin			1	mA	V_{DD} +0.3≧V_{IN}≧ -0.3
R_{PH}	Pull-high Resistance		100 200 450		KΩ	V_{DD} =5.0V V_{DD} =3.3V V_{DD} =2.2V
V_{BG}	Band-gap Reference Voltage	1.145*	1.20*	1.255*	V	V_{DD} =2.2V ~ 5.5V -20^{o}C <Ta<70^{o}C*
f_{IHRC}	Frequency of IHRC after calibration *****	15.76*	16*	16.24*	MHz	25^{o}C, V_{DD} =2.2V~5.5V
		15.20*	16*	16.80*		V_{DD} =2.2V~5.5V, 0^{o}C <Ta<70^{o}C*
t_{INT}	Interrupt pulse width	30			ns	V_{DD} = 5.0V

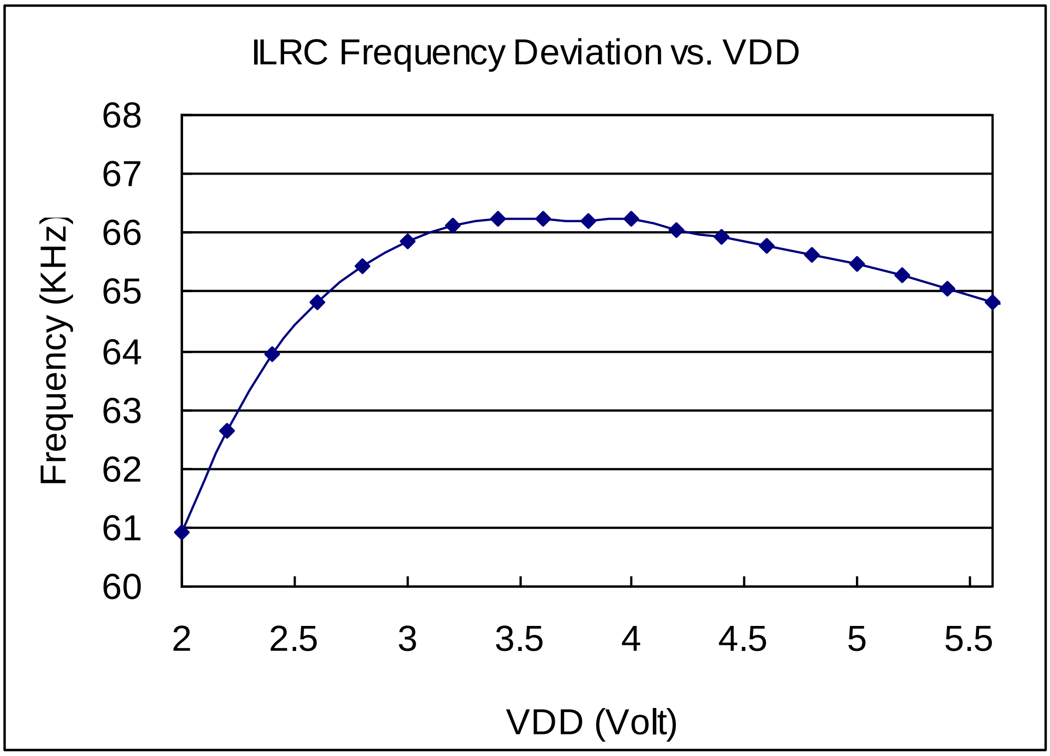
Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{ADC}	Supply voltage for workable ADC	2.2		V_{DD}	V	
V_{AD}	AD Input Voltage	0		V_{DD}	V	
ADrs	ADC resolution		12		bit	
ADcs	ADC current consumption		0.9 0.8		mA	@5V @3V
ADclk	ADC clock period		2		us	2.2V ~ 5.5V
t_{ADCONV}	ADC conversion time (T_{ADCLK} is the period of the selected AD conversion clock)		16		T_{ADCLK}	12-bit resolution
AD DNL	ADC Differential NonLinearity		±2*		LSB	
AD INL	ADC Integral NonLinearity		±4*		LSB	
ADos	ADC offset*		2		mV	@ V_{DD} =3V
V_{REFH}	ADC reference high voltage 4V 3V 2V	 3.90 2.93 1.95	 4 3 2	 4.10 3.07 2.05		@ V_{DD} =5V, 25 °C
V_{DR}	RAM data retention voltage*	1.5			V	in stop mode
t_{WDT}	Watchdog timeout period		8k		T_{ILRC}	misc[1:0]=00 (default)
			16k			misc[1:0]=01
			64k			misc[1:0]=10
			256k			misc[1:0]=11
t_{WUP}	Wake-up time period for fast wake-up		45		T_{ILRC}	Where T_{ILRC} is the time period of ILRC
	Wake-up time period for normal wake-up		3000			
t_{SBP}	System boot-up period from power-on for Normal boot-up		60		ms	V_{DD} =5V
	System boot-up period from power-on for Fast boot-up		600		us	V_{DD} =5V
t_{RST}	External reset pulse width	120			us	@ V_{DD} =5V
CPos	Comparator offset*	-	±10	±20	mV	
CPcm	Comparator input common mode*	0		V_{DD} -1.5	V	
CPspt	Comparator response time**		100	500	ns	Both Rising and Falling
CPmc	Stable time to change comparator mode		2.5	7.5	us	
CPcs	Comparator current consumption		20		uA	V_{DD} = 3.3V

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

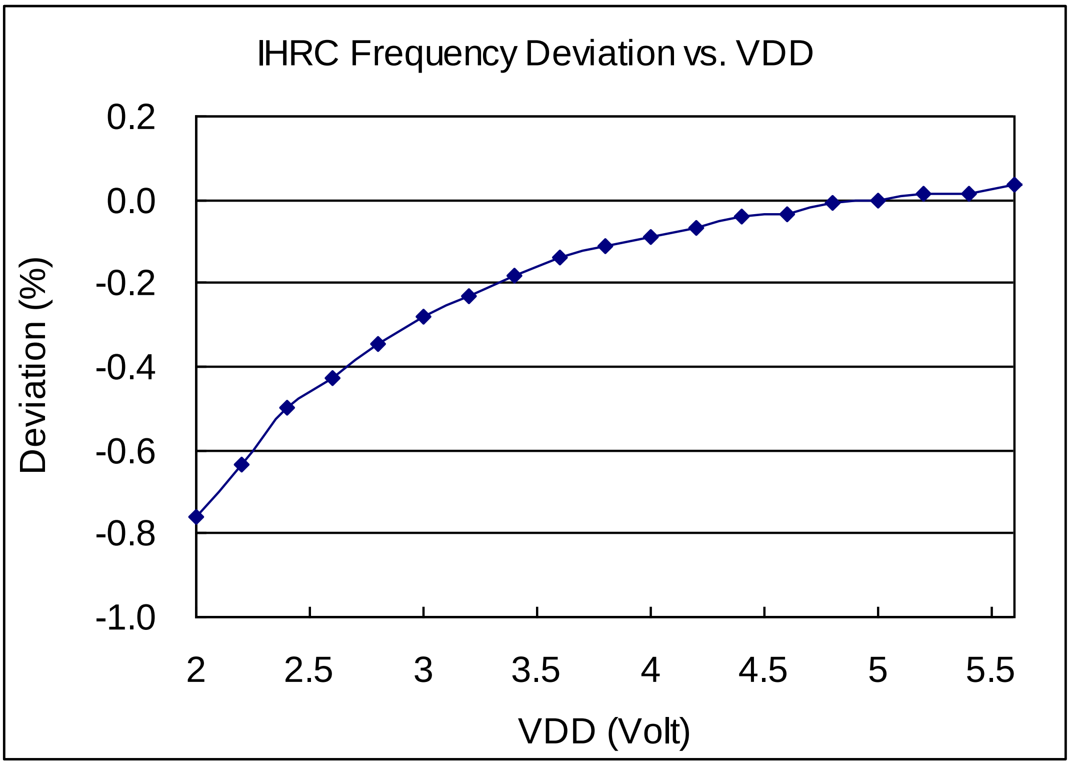
4.2. Absolute Maximum Ratings

Supply Voltage ………………………………......	2.2V ~ 5.5V
Input Voltage …………………………………..	-0.3V ~ V_{DD} + 0.3V
Operating Temperature	-20^{o}C ~ 70^{o}C
Junction Temperature …………………………	150°C
Storage Temperature …………………………	-50°C ~ 125°C

4.3. Typical ILRC frequency vs. VDD and temperature

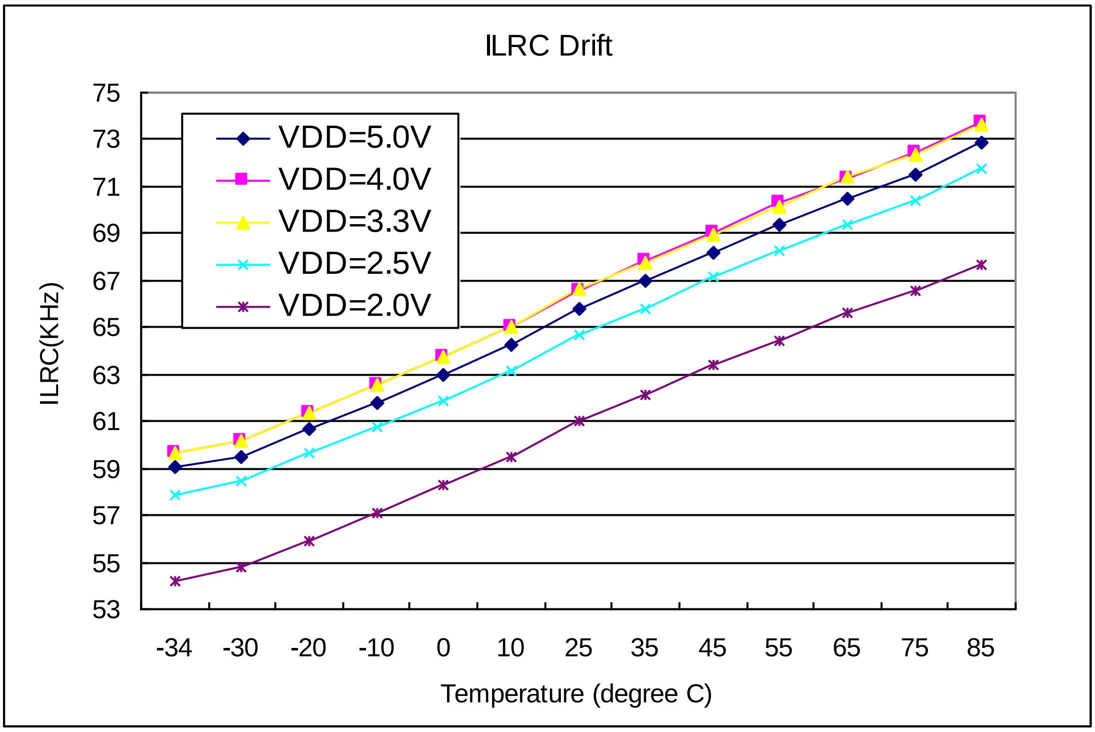


4.4. Typical IHRC frequency deviation vs. VDD

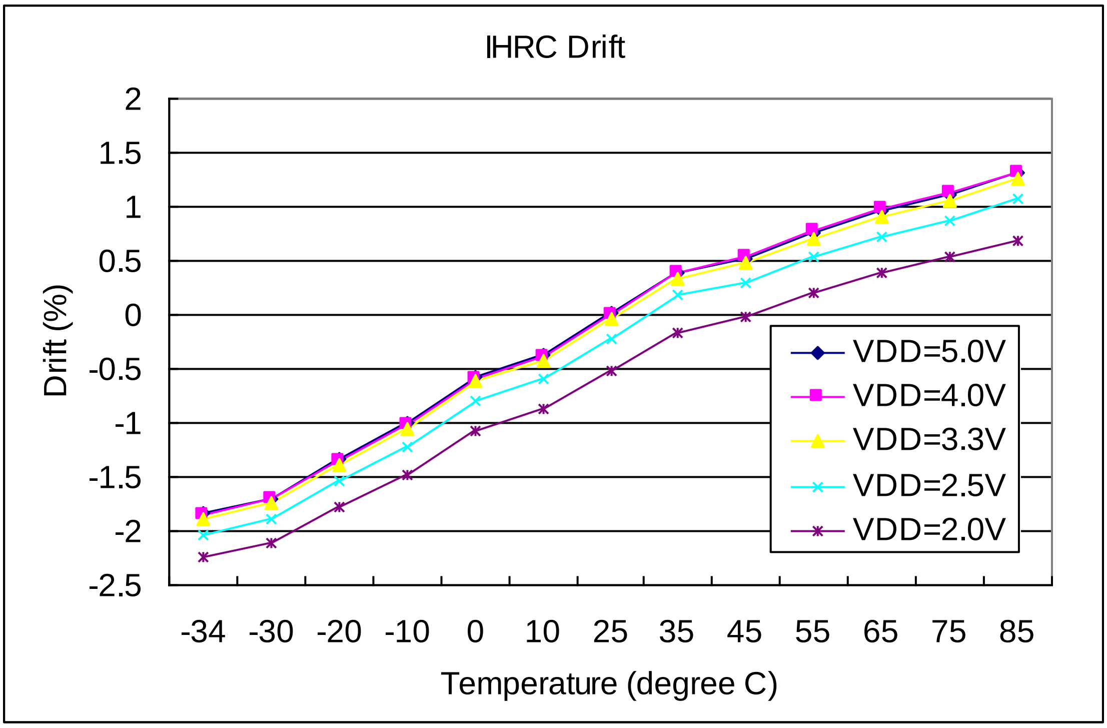


Note: IHRC is calibrated to 16MHz

4.5. Typical ILRC Frequency vs. Temperature



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

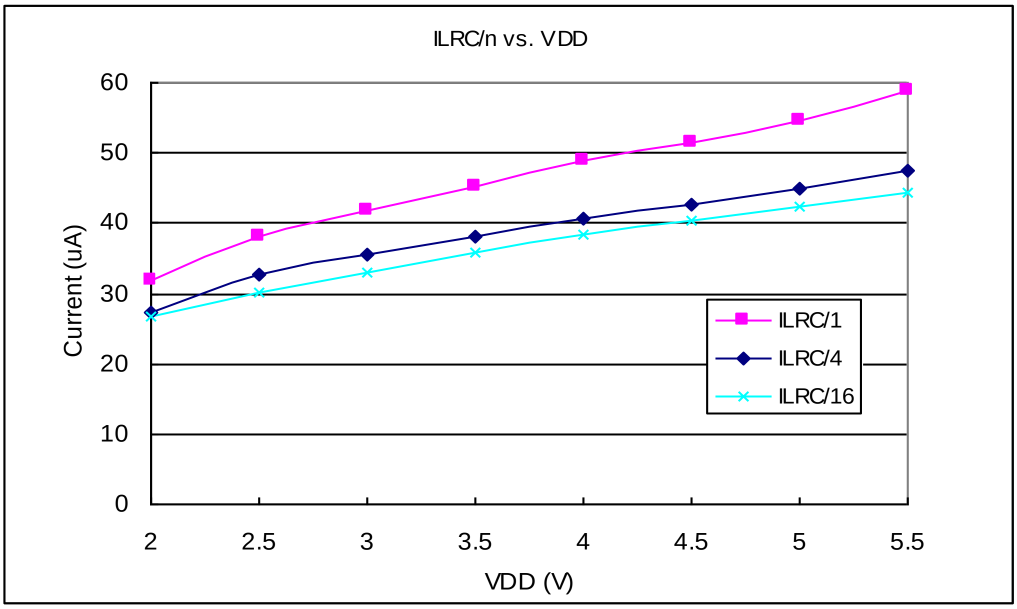


4.7. Typical operating current vs. VDD @ system clock = ILRC/n

Conditions:

ON: ILRC, Band-gap, LVR; **OFF**: IHRC, EOSC, T16, TM2, TM3, ADC modules;

IO: PA0:0.5Hz output toggle and no loading**, others**: input and no floating

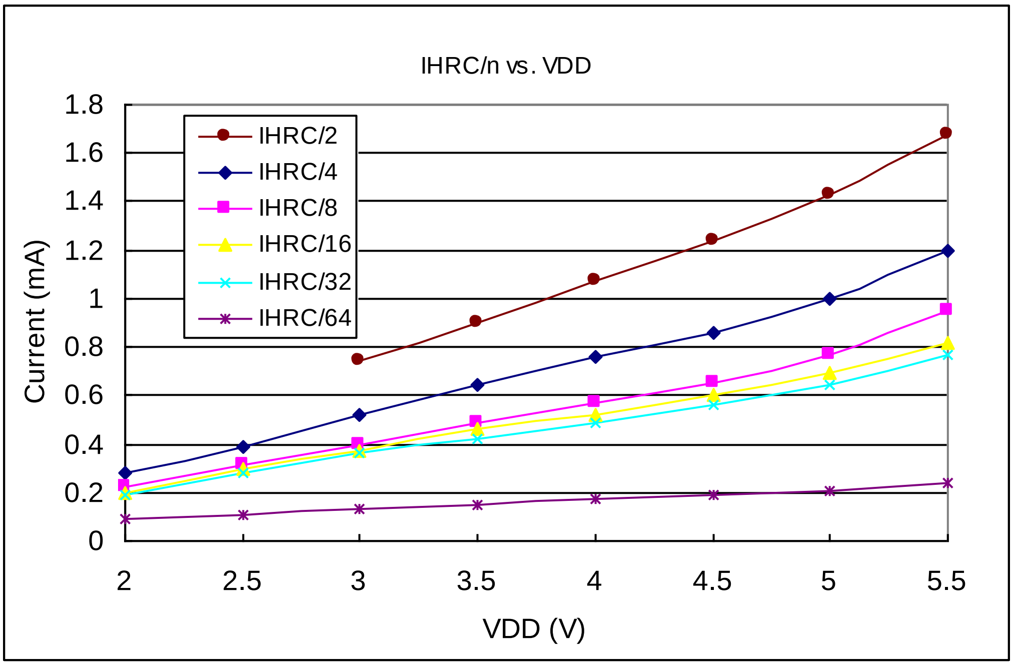


4.8. Typical operating current vs. VDD @ system clock = IHRC/n

Conditions:

ON: Band-gap, LVR, IHRC; **OFF**: ILRC, EOSC, LVR, T16, TM2, TM3, ADC modules;

IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating

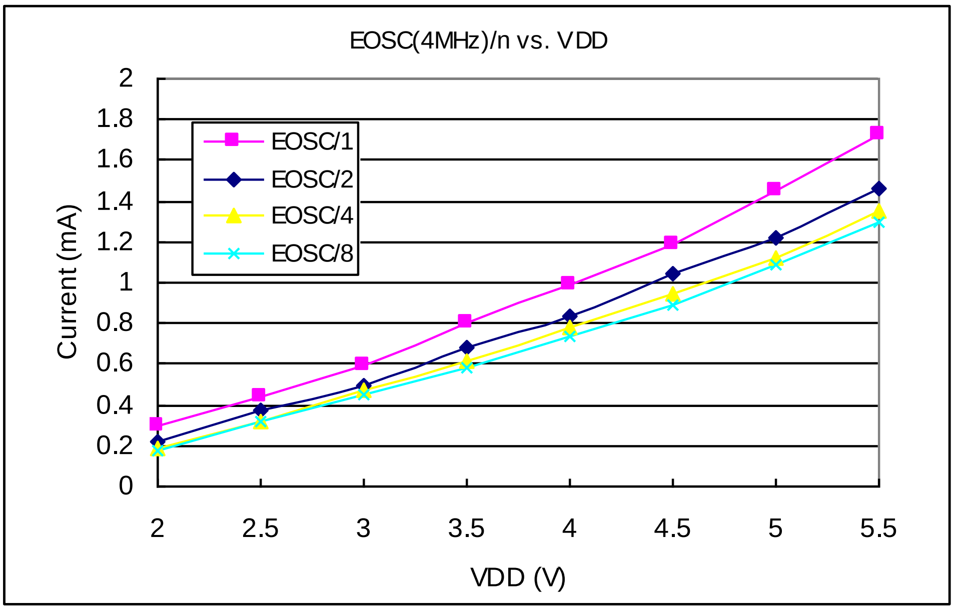


4.9. Typical operating current vs. VDD @ system clock = 4MHz EOSC / n

Conditions:

ON: EOSC, MISC.6 = 1, Band-gap, LVR; **OFF**: IHRC, ILRC, T16, TM2, TM3, ADC modules;

IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating

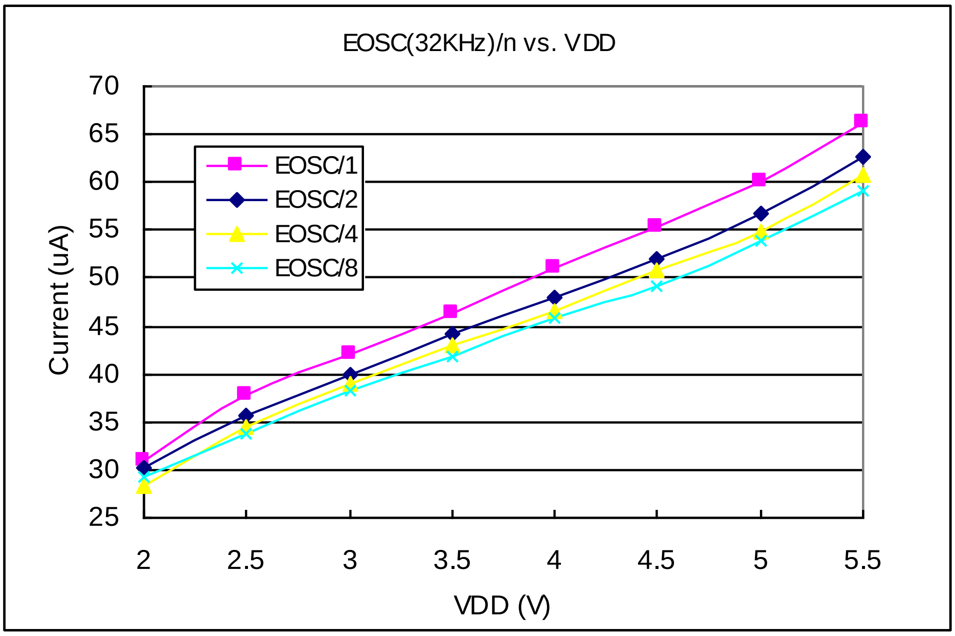


4.10.Typical operating current vs. VDD @ system clock = 32KHz EOSC / n (reserved[1])

Conditions:

ON: EOSC, MISC.6 = 1, Band-gap, LVR; **OFF**: IHRC, ILRC, T16, TM2, TM3, ADC modules;

IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



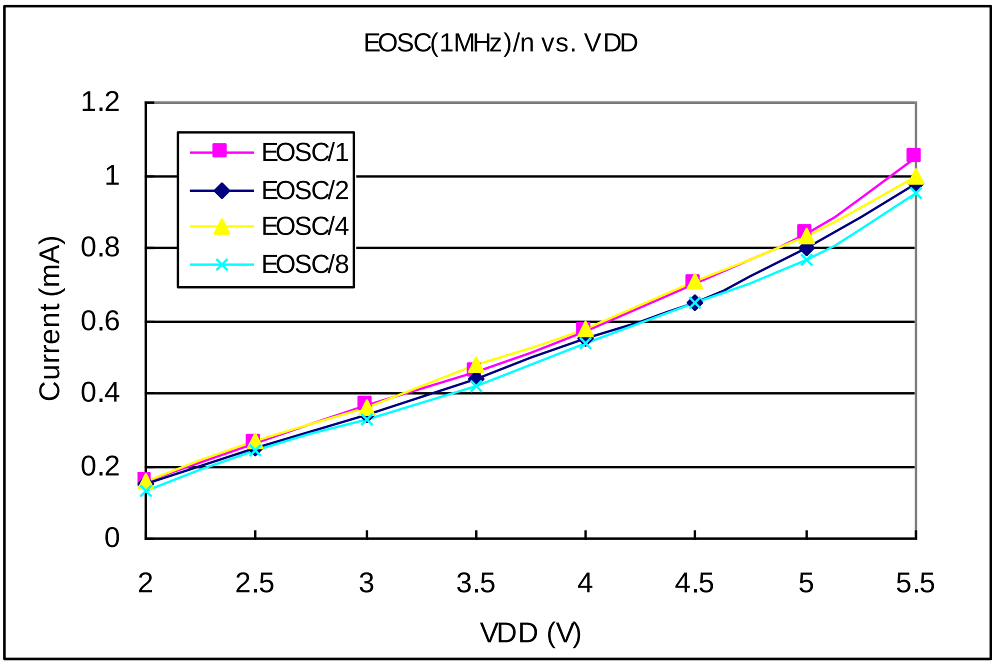
[1] Please contact our sales representative.

4.11. Typical operating current vs. VDD @ system clock = 1MHz EOSC / n

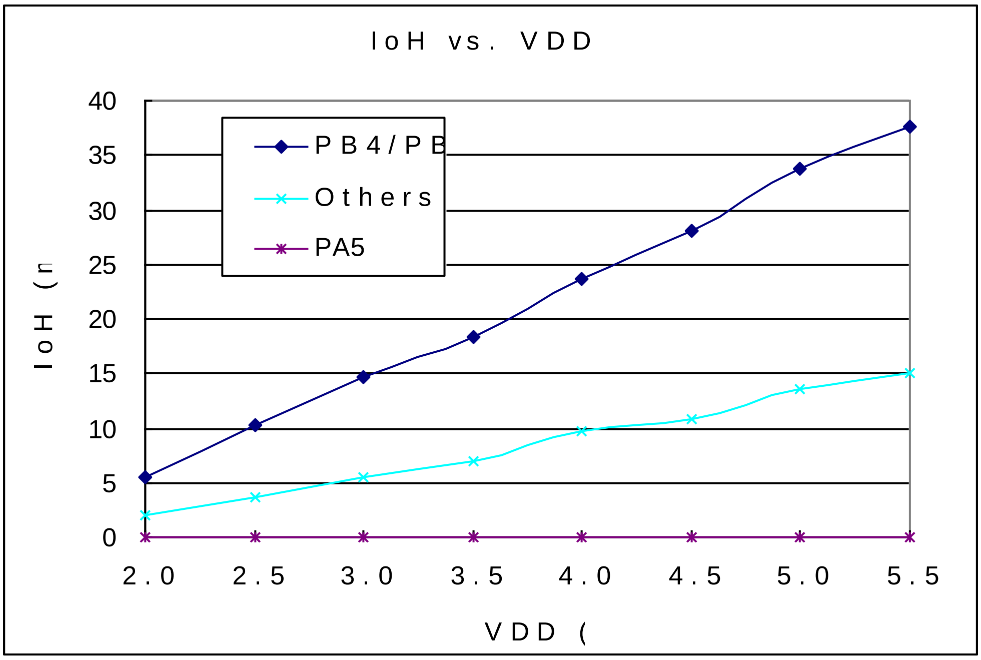
Conditions:

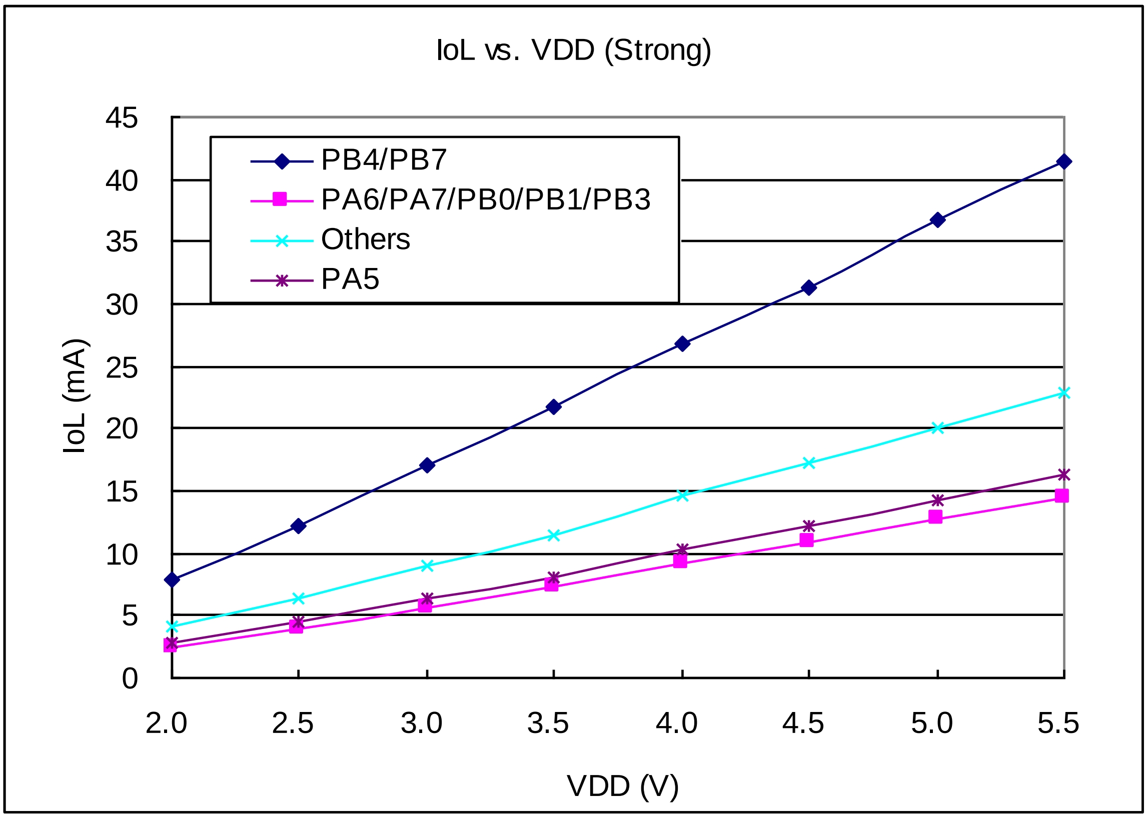
ON: EOSC, MISC.6 = 1, Band-gap, LVR; **OFF**: IHRC, ILRC, T16, TM2, TM3, ADC modules;

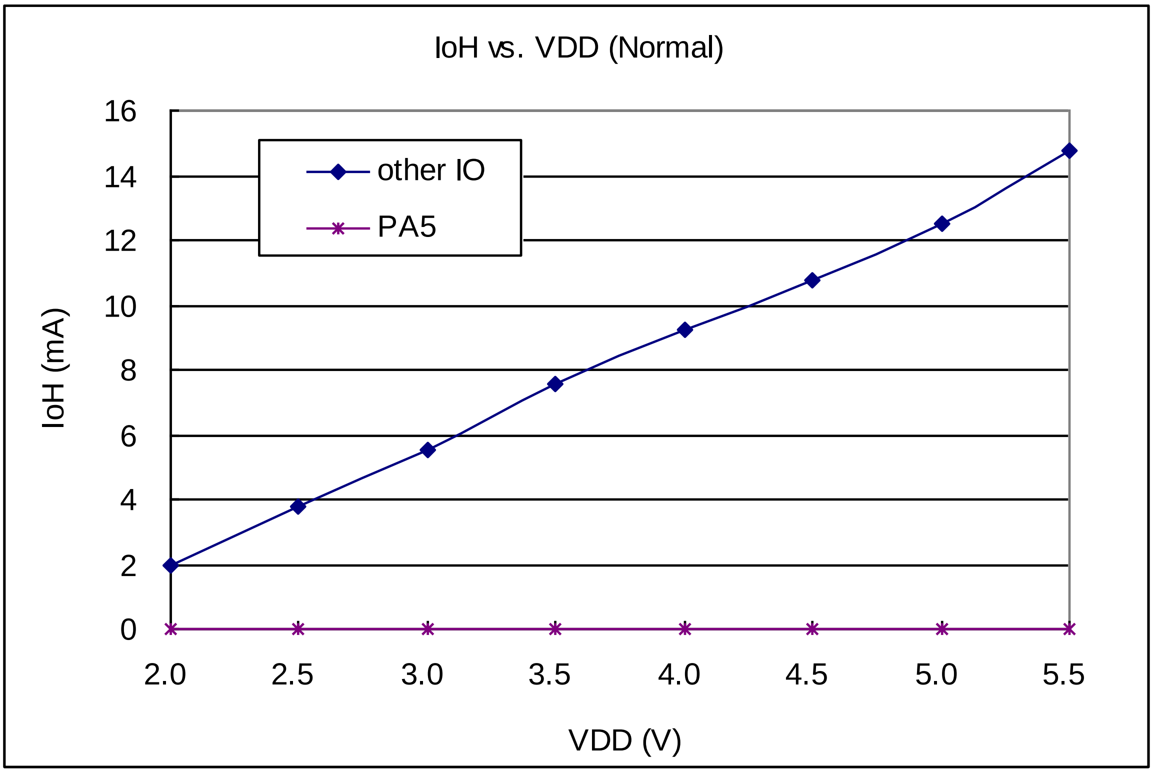
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating

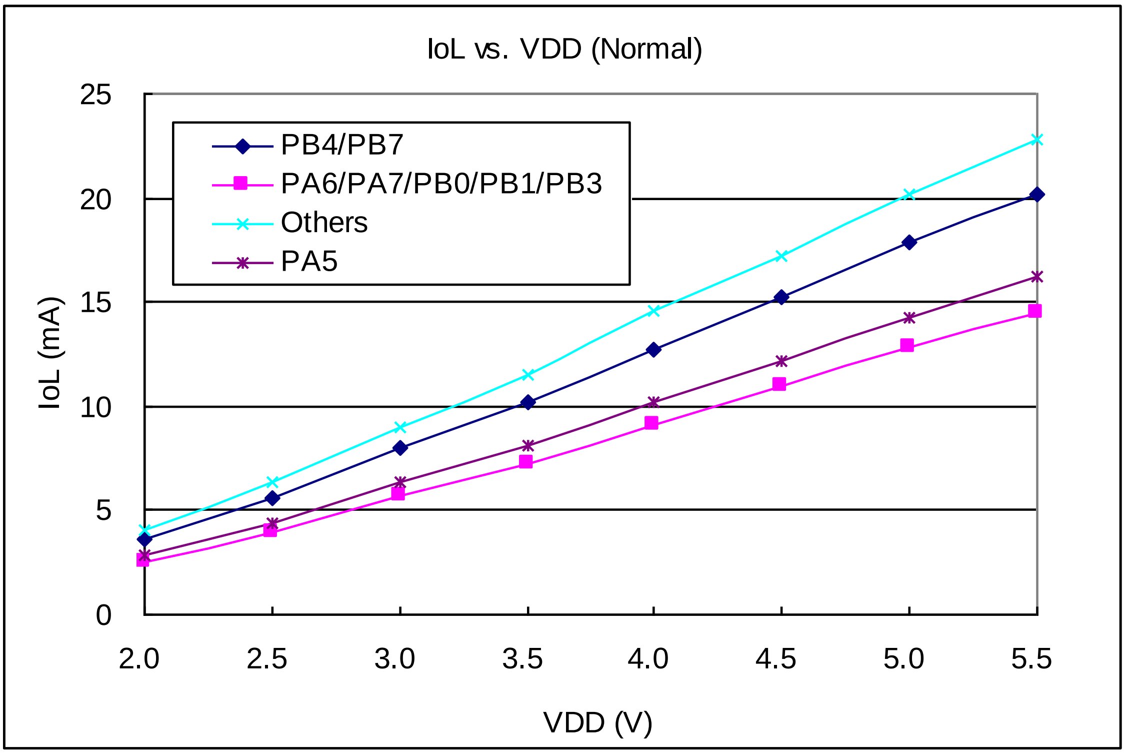


4.12. Typical IO driving current (I_{OH}) and sink current (I_{OL})

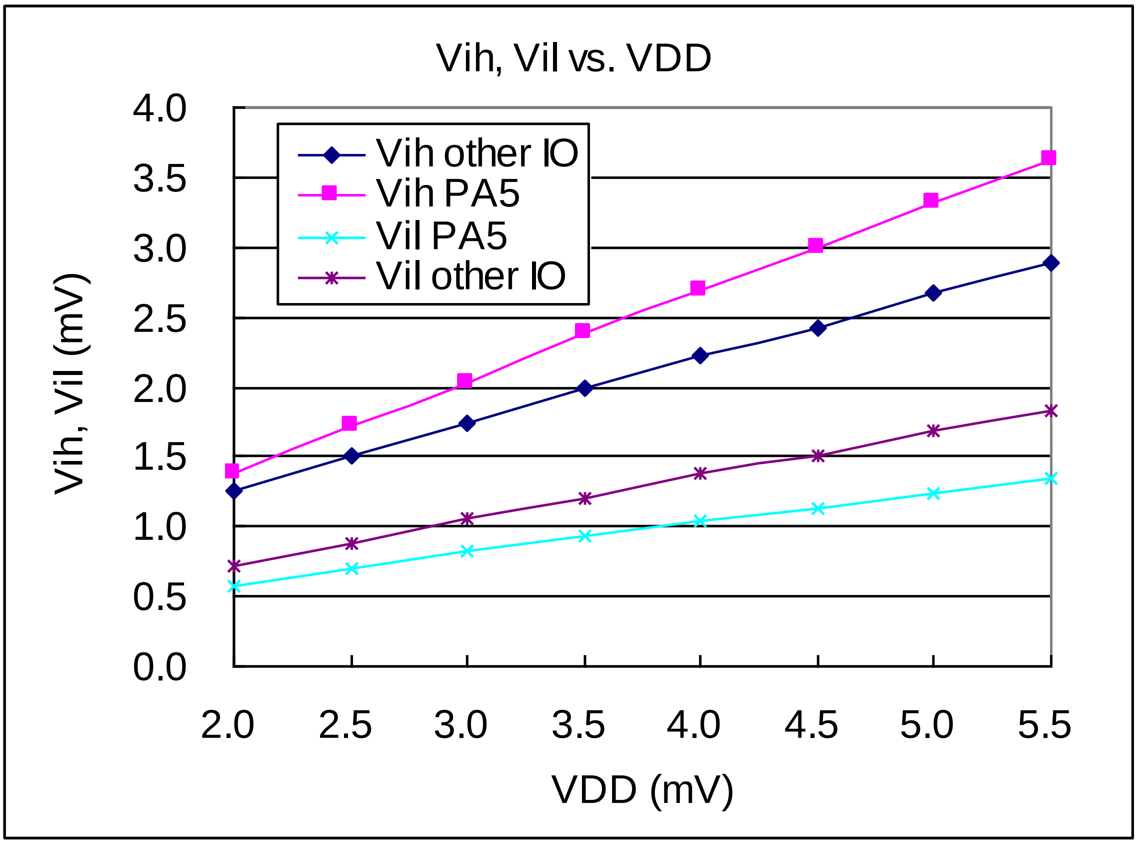



IoL vs. VDD (Strong)
IoL (mA)
VDD (V)
0
5
10
15
20
25
30
35
40
45
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
PB4/PB7
PA6/PA7/PB0/PB1/PB3
Others
PA5

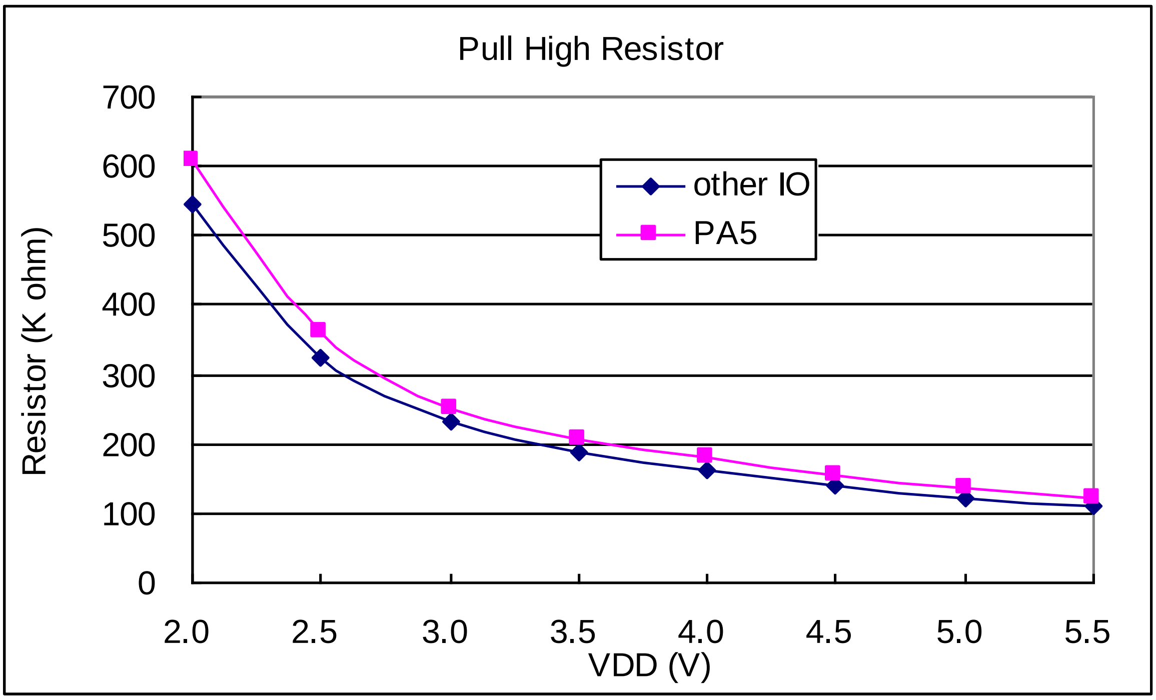

IoH vs. VDD (Normal)
IoH (mA)
VDD (V)
0
2
4
6
8
10
12
14
16
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
other IO
PA5



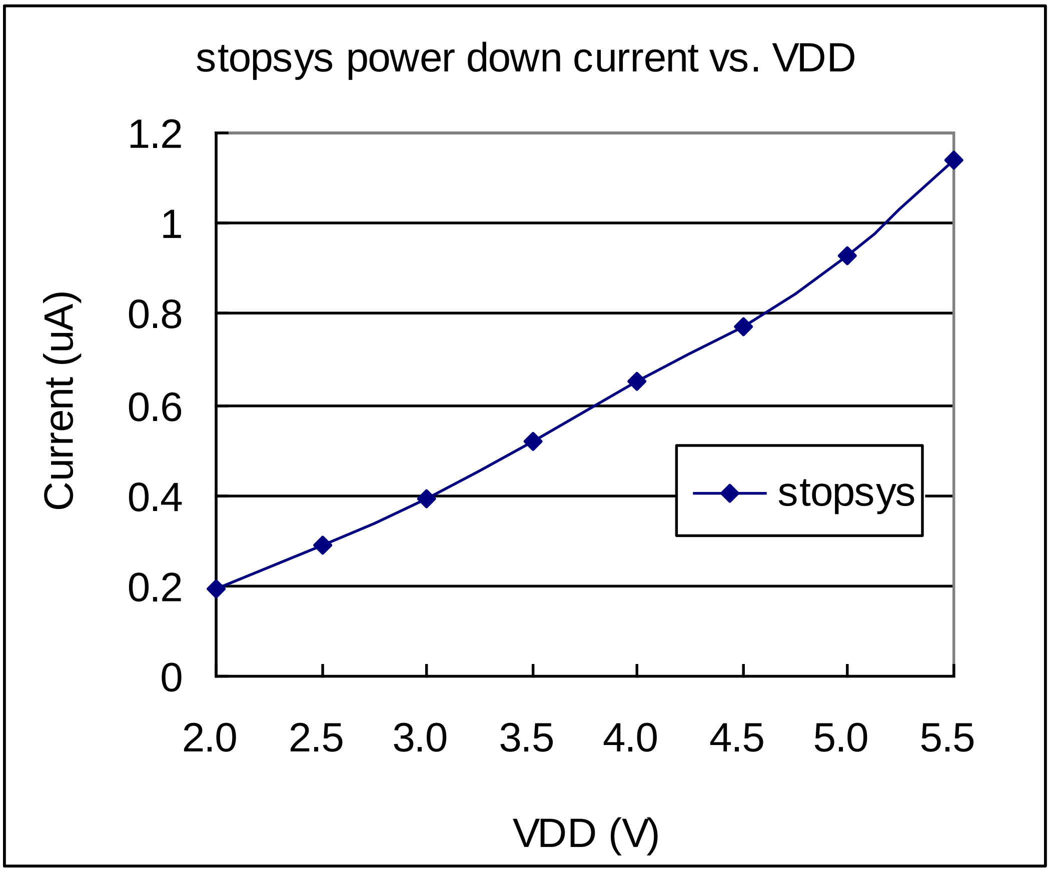
4.13. Typical IO input high/low threshold voltage (V_{IH}/V_{IL})

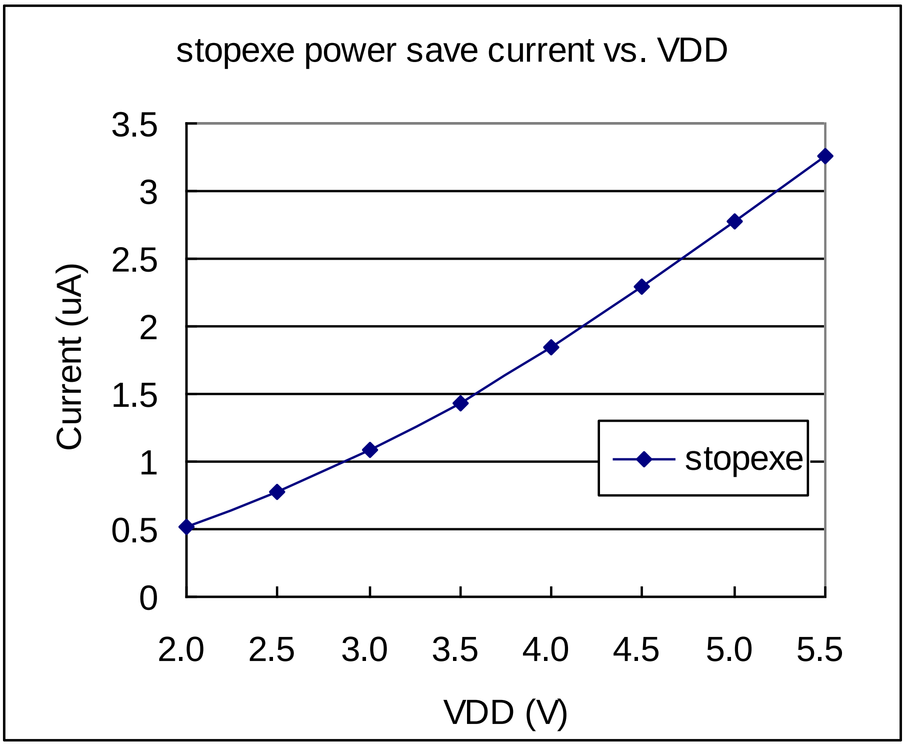


4.14. Typical resistance of IO pull high device

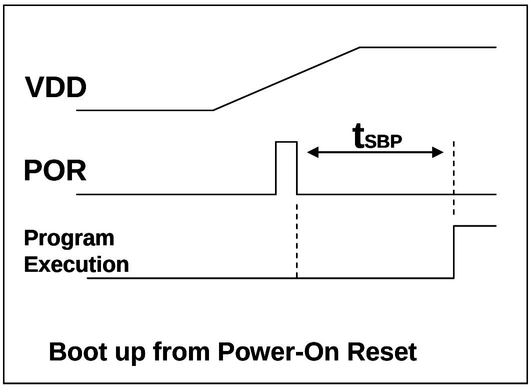


4.15. Typical power down current (I_{PD}) and power save current (I_{PS})

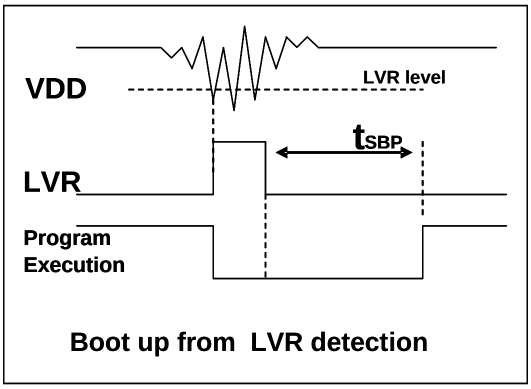




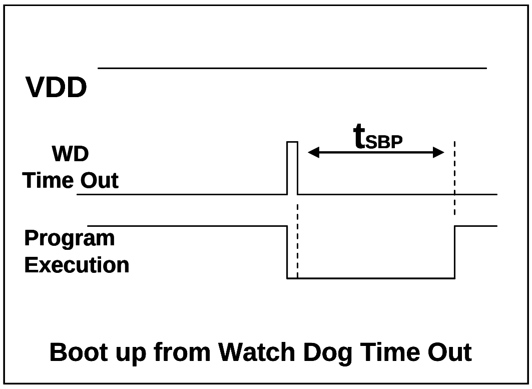
4.16. Timing charts for boot up conditions



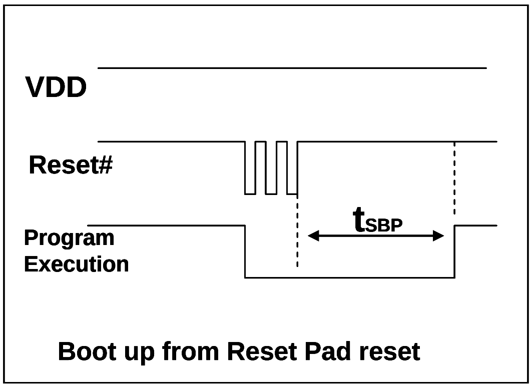
Boot up from Power-On Reset



Boot up from LVR detection



Boot up from Watch Dog Time Out



Boot up from Reset Pad reset

5. Functional Description

5.1. Program Memory - OTP

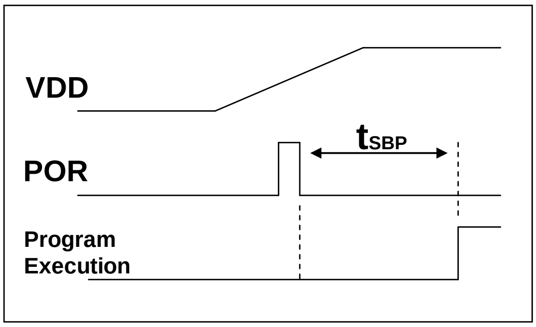
The OTP (One Time Programmable) program memory is used to store the program instructions to be executed. The OTP program memory may contains the data, tables and interrupt entry. After reset, the initial address 0x000 is reserved for system using, so the program will start from 0x001 which is GOTO FPPA0 instruction usually. The interrupt entry is 0x10 if used, the last 16 addresses are reserved for system using, like checksum, serial number, etc. The OTP program memory for PMS132 is a 2Kx14 bit that is partitioned as Table 1. The OTP memory from address ‘0x7E8 to 0x7FF is for system using, address space from0x002 to 0x00F and from 0x011 to 0x7E7 are user program spaces.

Address	Function
0x000	System Using
0x001	GOTO FPPA0 instruction
0x002	User program
•	•
0x00F	User program
0x010	Interrupt entry address
0x011	User program
•	•
0x7E7	User program
0x7E8	System Using
•	•
0x7FF	System Using

Table 1: Program Memory Organization

5.2. Boot Procedure

POR (Power-On-Reset) is used to reset PMS132 when power up. The boot up time can be optional fast or normal. Time for fast boot-up is about 45 ILRC clock cycles whereas 3000 ILRC clock cycles for normal boot-up. 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

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 of each processing unit 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 128 bytes data memory of PMS132 can be accessed by indirect access mechanism.

5.4. Oscillator and clock

There are three oscillator circuits provided by PMS132: external crystal oscillator (EOSC), internal high RC oscillator (IHRC) and internal low RC oscillator (ILRC), and these three oscillators are enabled or disabled by registers eoscr.7, clkmd.4 and clkmd.2 independently. User can choose one of these three oscillators as system clock source and use ***clkmd*** register to target the desired frequency as system clock to meet different applications.

Oscillator Module	Enable/Disable
EOSC	**eoscr**.7
IHRC	**clkmd**.4
ILRC	**clkmd**.2

Table 2: Three 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. The frequency deviation can be within 1.5% normally after calibration under the calibrated voltage, however, it still drifts slightly with supply voltage and operating temperature, the total drift rate is around ±5% for V_{DD}=2.2V~5.5V and -20^{o}C~70^{o}C operating conditions. 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.

5.4.2. Chip calibration

The IHRC frequency and band-gap reference voltage may be different chip by chip due to manufacturing variation, PMS132 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_{DD}=(**p3**)V;

Where, **p1**=2, 4, 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=2.5 ~ 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:

SYSCLK	CLKMD	IHRCR	Description
○ Set IHRC / 2	= 34h (IHRC / 2)	Calibrated	IHRC calibrated to 16MHz, CLK=8MHz (IHRC/2)
○ Set IHRC / 4	= 14h (IHRC / 4)	Calibrated	IHRC calibrated to 16MHz, CLK=4MHz (IHRC/4)
○ Set IHRC / 8	= 3Ch (IHRC / 8)	Calibrated	IHRC calibrated to 16MHz, CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 1Ch (IHRC / 16)	Calibrated	IHRC calibrated to 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 7Ch (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 PMS132 for different option:

(1) .ADJUST_IC SYSCLK=IHRC/2, IHRC=16MHz, V_{DD}=5V

After boot up, CLKMD = 0x34：

- ◆ IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is enabled
- ◆ System CLK = IHRC/2 = 8MHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(2) .ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, V_{DD}=3.3V

After boot up, CLKMD = 0x14：

- ◆ IHRC frequency is calibrated to 16MHz@V_{DD}=3.3V and IHRC module is enabled
- ◆ System CLK = IHRC/4 = 4MHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(3) .ADJUST_IC SYSCLK=IHRC/8, IHRC=16MHz, V_{DD}=2.5V

After boot up, CLKMD = 0x3C：

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

(4) .ADJUST_IC SYSCLK=IHRC/16, IHRC=16MHz, V_{DD}=2.5V

After boot up, CLKMD = 0x1C：

- ◆ IHRC frequency is calibrated to 16MHz@V_{DD}=2.5V and IHRC module is enabled
- ◆ System CLK = IHRC/16 = 1MHz
- ◆ Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(5) .ADJUST_IC SYSCLK=IHRC/32, IHRC=16MHz, V_{DD}=5V

After boot up, CLKMD = 0x7C：

- IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is enabled
- System CLK = IHRC/32 = 500KHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(6) .ADJUST_IC SYSCLK=ILRC, IHRC=16MHz, V_{DD}=5V

After boot up, CLKMD = 0XE4：

- IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is disabled
- System CLK = ILRC
- Watchdog timer is enabled, ILRC is enabled, PA5 is input mode

(7) .ADJUST_IC DISABLE

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

- IHRC is not calibrated and IHRC module is to be enabled or disabled by Boot-up Time.
- System CLK = ILRC or IHRC/64 (by Boot-up_Time)
- Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode.

5.4.4. External Crystal Oscillator

If crystal oscillator is used, a crystal or resonator is required between X1 and X2. Fig.2 shows the hardware connection under this application; the range of operating frequency of crystal oscillator can be from 32 KHz (reserved) to 4MHz, depending on the crystal placed on; higher frequency oscillator than 4MHz is NOT supported.

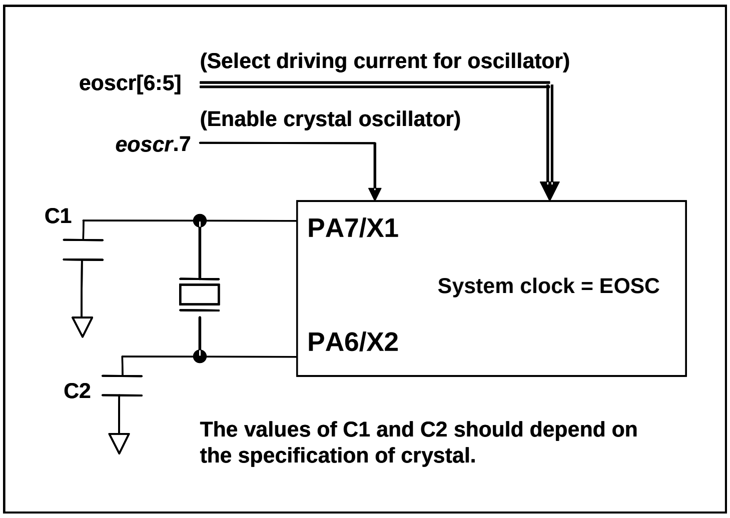

(Select driving current for oscillator)
eoscr[6:5]
(Enable crystal oscillator)
eoscr.7
C1
PA7/X1
System clock = EOSC
PA6/X2
C2
The values of C1 and C2 should depend on the specification of crystal.

Fig.2: Connection of crystal oscillator

Besides crystal, external capacitor and options of PMS132 should be fine tuned in *eoscr* (0x0b) register to have good sinusoidal waveform. The *eoscr*.7 is used to enable crystal oscillator module, *eoscr*.6 and *eoscr*.5 are used to set the different driving current to meet the requirement of different frequency of crystal oscillator:

- *eoscr*.[6:5]=01 : Low driving capability, for lower frequency, ex: 32KHz crystal oscillator (reserved)
- *eoscr*.[6:5]=10 : Middle driving capability, for middle frequency, ex: 1MHz crystal oscillator
- *eoscr*.[6:5]=11 : High driving capability, for higher frequency, ex: 4MHz crystal oscillator

Table 4 shows the recommended values of C1 and C2 for different crystal oscillator; the measured start-up time under its corresponding conditions is also shown. Since the crystal or resonator had its own characteristic, the capacitors and start-up time may be slightly different for different type of crystal or resonator, please refer to its specification for proper values of C1 and C2.

Frequency	C1	C2	Measured Start-up time	Conditions
4MHz	4.7pF	4.7pF	6ms	(*eoscr*[6:5]=11, *misc*.6=0)
1MHz	10pF	10pF	11ms	(*eoscr*[6:5]=10, *misc*.6=0)
32KHz (reserved)	22pF	22pF	450ms	(*eoscr*[6:5]=01, *misc*.6=0)

Table 4: Recommend values of C1 and C2 for crystal and resonator oscillators

When using the crystal oscillator, user must pay attention to the stable time of oscillator after enabling it, the stable time of oscillator will depend on frequency, crystal type, external capacitor and supply voltage. Before switching the system to the crystal oscillator, user must make sure the oscillator is stable; the reference program is shown as below:

```
void    FPPA0 (void)
{
        . ADJUST_IC   SYSCLK=IHRC/16, IHRC=16MHz, VDD=5V
        $       EOSCR       Enable, 4MHz;       // EOSCR = 0b110_00000;
        $ T16M        EOSC，  /1，  BIT13;       // while T16.Bit13 0 => 1, Intrq.T16 => 1
                                                // suppose crystal osc. is stable
        WORD          count =      0;
        stt16         count;
        Intrq.T16     =      0;
        while (¡ Intrq.T16)   NULL;             // count fm 0x0000 to 0x2000, then trigger INTRQ.T16

        clkmd=        0xb4;                     // switch system clock to EOSC;

        clkmd.4 = 0;                            //disable IHRC
        ...
```

Please notice that the crystal oscillator should be fully turned off before entering the power-down mode, in order to avoid unexpected wakeup event.

5.4.5. System Clock and LVR level

The clock source of system clock comes from EOSC, IHRC and ILRC, the hardware diagram of system clock in the PMS132 is shown as Fig.3.

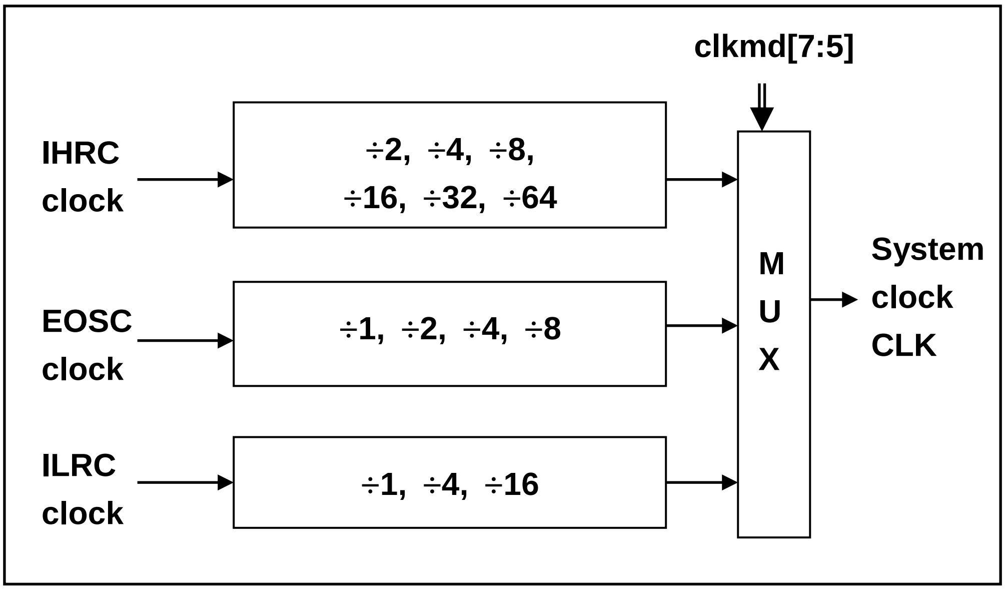


Fig.3: Options of System Clock

User can choose different operating system clock depends on its requirement; the selected operating system clock should be combined with supply voltage and LVR level to make system stable. The LVR level will be selected during compilation, the following operating frequency and LVR level is recommended:

- system clock = 8MHz with LVR=3.5V
- system clock = 4MHz with LVR=2.5V
- system clock = 2MHz with LVR=2.2V

5.4.6. 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 PMS132 can be switched among IHRC, ILRC and EOSC 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/2

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

Case 2: Switching system clock from ILRC to EOSC

```
…                                //    system clock is ILRC
CLKMD         =     0xA6；       //    switch to IHRC, ILRC CAN NOT be disabled here
CLKMD.2       =     0；          //    ILRC CAN be disabled at this time
…
```

Case 3: Switching system clock from IHRC/2 to ILRC

```
…                                //    system clock is IHRC/2
CLKMD         =     0xF4；       //    switch to ILRC, IHRC CAN NOT be disabled here
CLKMD.4       =     0；          //    IHRC CAN be disabled at this time
…
```

Case 4: Switching system clock from IHRC/2 to EOSC

```
…                                //    system clock is IHRC/2
CLKMD         =     0XB0；       //    switch to EOSC, IHRC CAN NOT be disabled here
CLKMD.4       =     0；          //    IHRC CAN be disabled at this time
…
```

Case 5: Switching system clock from IHRC/2 to IHRC/4

```
…                                //    system clock is IHRC/2, ILRC is enabled here
CLKMD         =     0X14；       //    switch to IHRC/4
…
```

Case 6: 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/2 and
                                       turn off ILRC oscillator at the same time
```

5.5. Comparator

One hardware comparator is built inside the PMS132; Fig.4 shows its hardware diagram. It can compare signals between two pins or with either internal reference voltage $V_{\text{internal R}}$ or internal band-gap reference voltage. The two signals to be compared, one is the plus input and the other one is the minus input. For the minus input of comparator can be PA3, PA4, Internal band-gap 1.20 volt, PB6, PB7 or $V_{\text{internal R}}$ selected by bit [3:1] of gpcc register, and the plus input of comparator can be PA4 or $V_{\text{internal R}}$ selected by bit 0 of gpcc register. The output result can be enabled to output to PA0 directly, or sampled by Time2 clock (TM2_CLK) which comes from Timer2 module. The output can be also inversed the polarity by bit 4 of ***gpcc*** register, the comparator output can be used to request interrupt service.

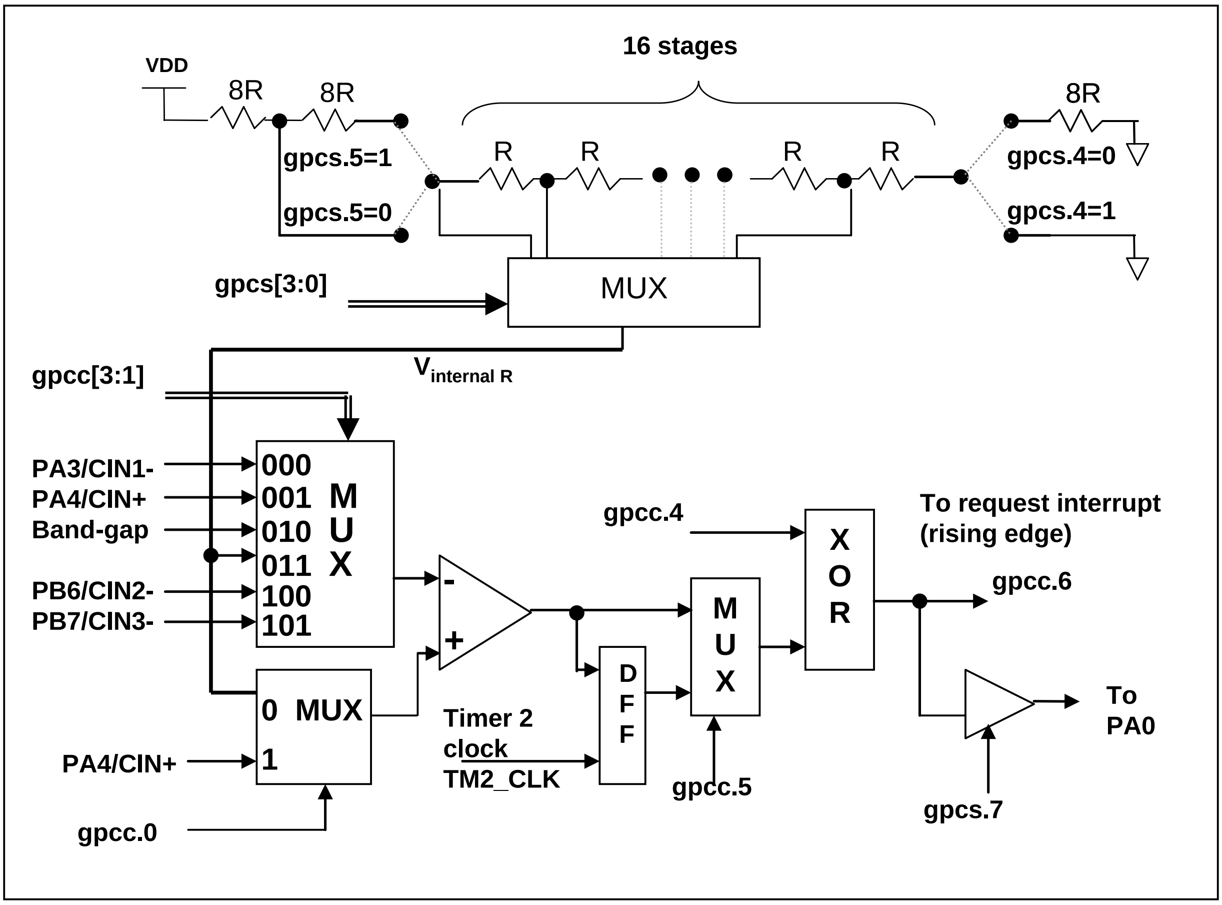
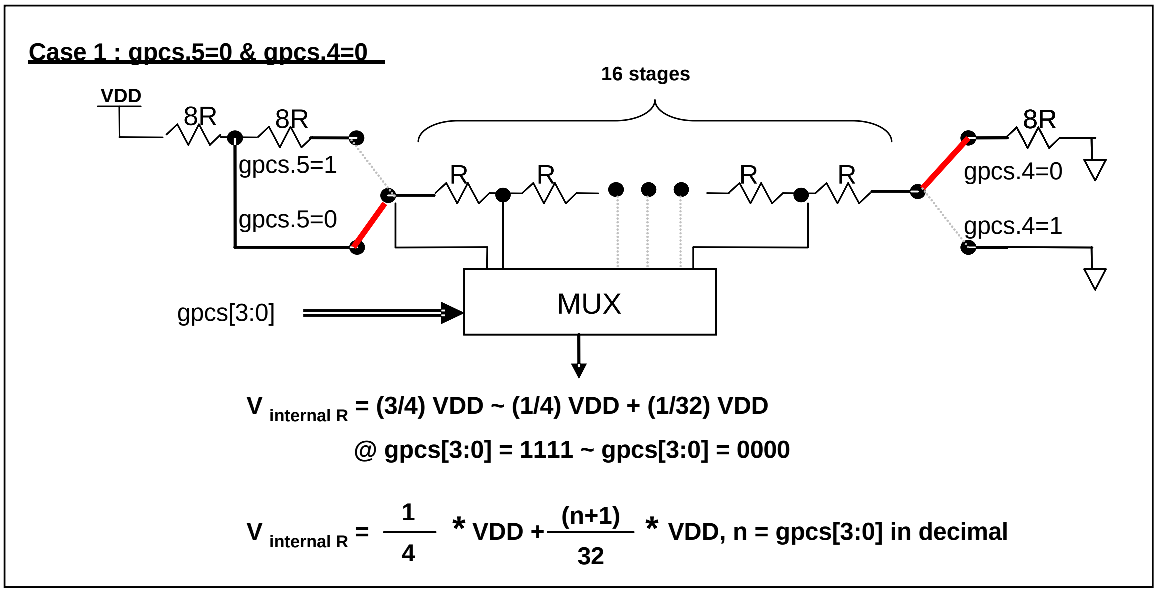


Fig.4: Hardware diagram of comparator

5.5.1 Internal reference voltage ($V_{\text{internal R}}$)

The internal reference voltage $V_{\text{internal R}}$ is built by series resistance to provide different level of reference voltage, bit 4 and bit 5 of ***gpcs*** register are used to select the maximum and minimum values of $V_{\text{internal R}}$ and bit [3:0] of ***gpcs*** register are used to select one of the voltage level which is deivided-by-16 from the defined maximum level to minimum level. Fig.5 to Fig.8 shows four conditions to have different reference voltage $V_{\text{internal R}}$. By setting the ***gpcs*** register, the internal reference voltage $V_{\text{internal R}}$ can be ranged from (1/32)*V_{DD} to (3/4)*V_{DD}.

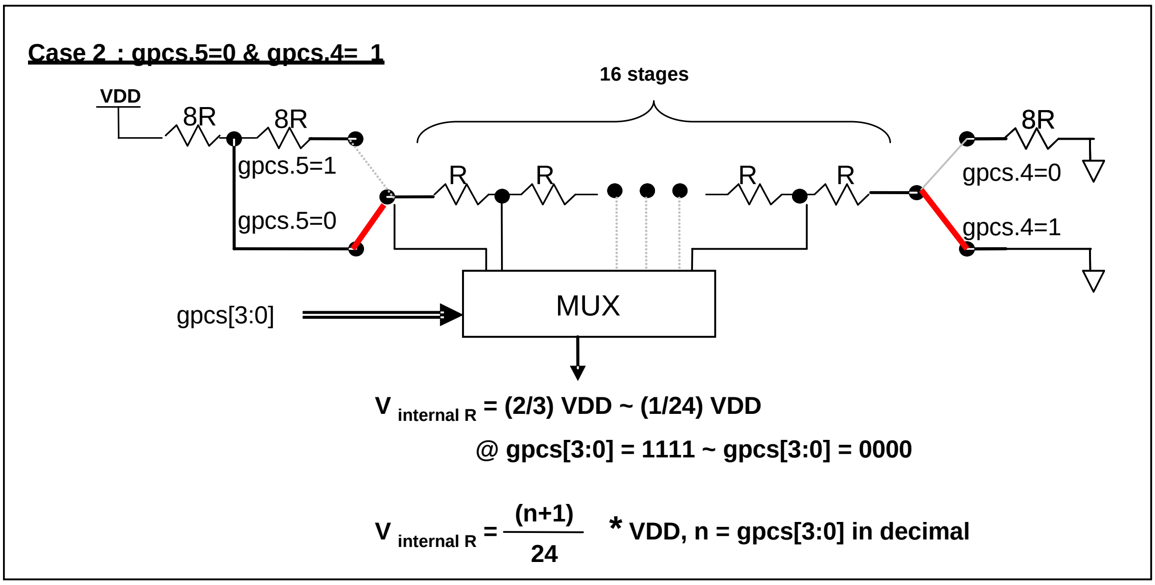


$$V_{\text{internal R}} = \text{(3/4) VDD} \sim \text{(1/4) VDD} + \text{(1/32) VDD}$$

@ gpcs[3:0] = 1111 ~ gpcs[3:0] = 0000

$$V_{\text{internal R}} = \frac{1}{4} * \text{VDD} + \frac{(n+1)}{32} * \text{VDD, n = gpcs[3:0] in decimal}$$

Fig.5: $V_{\text{internal R}}$ hardware connection if gpcs.5=0 and gpcs.4=0

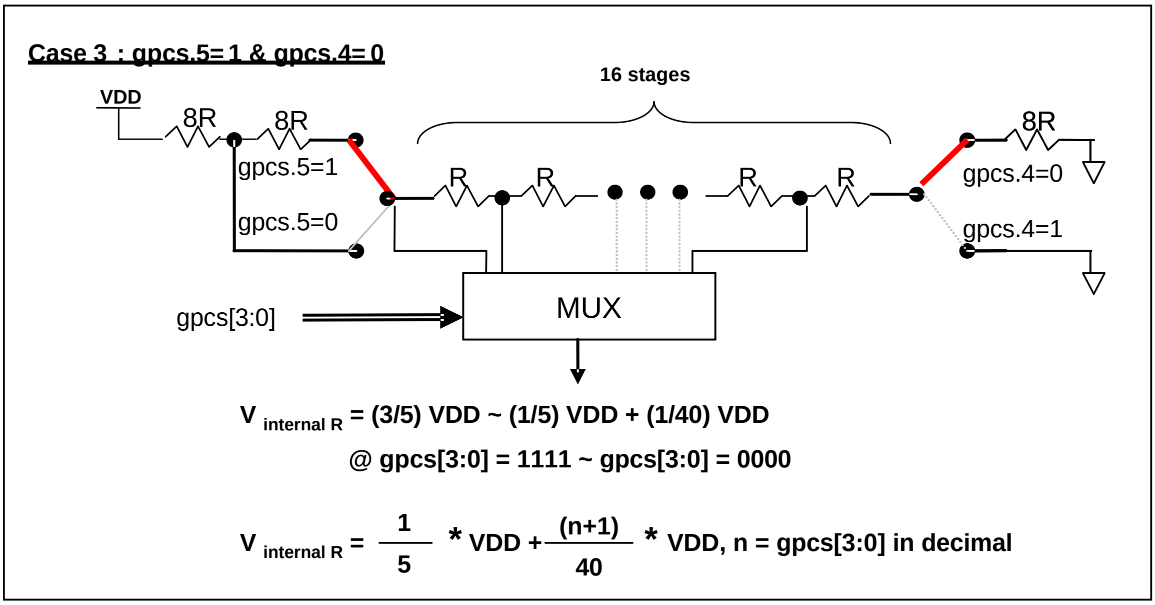


$$V_{\text{internal R}} = \text{(2/3) VDD} \sim \text{(1/24) VDD}$$

@ gpcs[3:0] = 1111 ~ gpcs[3:0] = 0000

$$V_{\text{internal R}} = \frac{(n+1)}{24} * \text{VDD, n = gpcs[3:0] in decimal}$$

Fig.6: $V_{\text{internal R}}$ hardware connection if gpcs.5=0 and gpcs.4=1

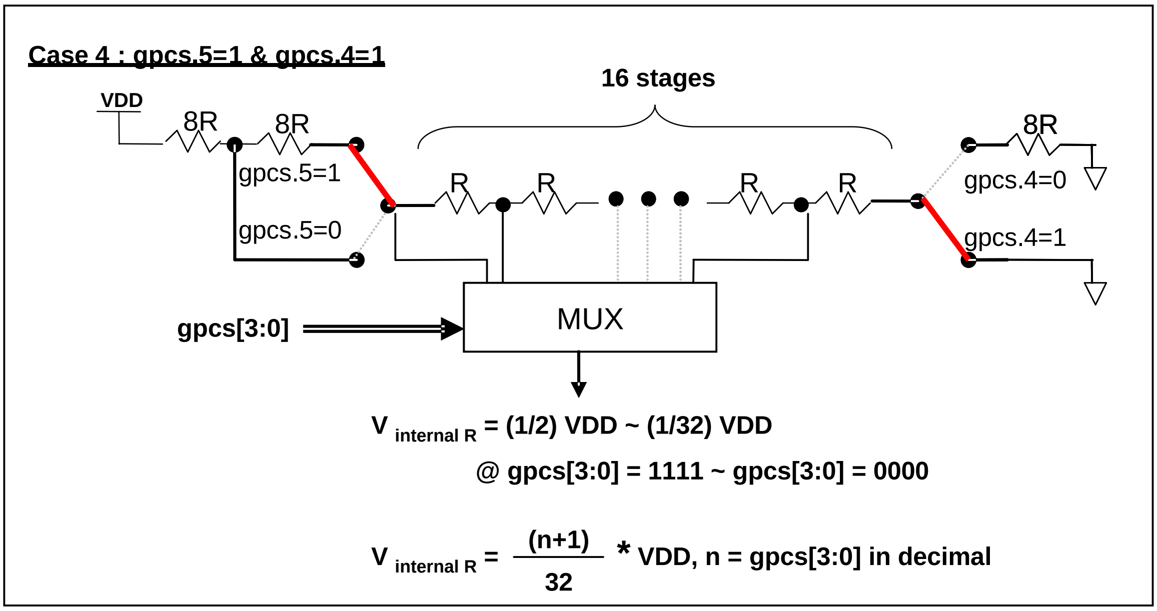


$$V_{\text{internal R}} = (3/5)\ \text{VDD} \sim (1/5)\ \text{VDD} + (1/40)\ \text{VDD}$$

@ gpcs[3:0] = 1111 ~ gpcs[3:0] = 0000

$$V_{\text{internal R}} = \frac{1}{5} * \text{VDD} + \frac{(n+1)}{40} * \text{VDD},\ n = \text{gpcs[3:0] in decimal}$$

Fig.7: $V_{\text{internal R}}$ hardware connection if gpcs.5=1 and gpcs.4=0



$$V_{\text{internal R}} = (1/2)\ \text{VDD} \sim (1/32)\ \text{VDD}$$

@ gpcs[3:0] = 1111 ~ gpcs[3:0] = 0000

$$V_{\text{internal R}} = \frac{(n+1)}{32} * \text{VDD},\ n = \text{gpcs[3:0] in decimal}$$

Fig.8: $V_{\text{internal R}}$ hardware connection if gpcs.5=1 and gpcs.4=1

5.5.2 Using the comparator

Case I:

Choosing PA3 as minus input and $V_{internal\ R}$ with (18/32)*V_{DD} voltage level as plus input, the comparator result will be output to PA0, the comparator result will be output to PA0. $V_{internal\ R}$ is configured as Fig. 10 and gpcs [3:0] = 4b'1001 (n=9) to have $V_{internal\ R}$ = (1/4)*V_{DD} + [(9+1)/32]*V_{DD} = (18/32)*V_{DD}.

```
gpcs   = 0b1_0_00_1001;        // output to PA0, Vinternal R = VDD*(18/32)
gpcc   = 0b1_0_0_0_000_0;      // enable comp, - input: PA3, + input: Vinternal R
padier = 0bxxxx_0_xxx;         // disable PA3 digital input to prevent leakage current
```

Case 2:

Choosing $V_{internal\ R}$ as minus input with (14/32)*V_{DD} voltage level and PA4 as plus input, the comparator result will be inversed and then output to PA0. $V_{internal\ R}$ is configured as Fig. 11 and gpcs [3:0] = 4b'1101 (n=13) to have $V_{internal\ R}$ = [(13+1)/32]*V_{DD} = (14/32)*V_{DD}.

```
gpcs   = 0b1_1_1_1_1101;       // Vinternal R = VDD*(14/32)
gpcc   = 0b1_0_0_1_011_1;      // Inverse output, - input: Vinternal R, + input: PA4
padier = 0bxxx_0_xxxx;         // disable PA4 digital input to prevent leakage current
```

5.5.3 Using the comparator and band-gap 1.20V

The internal band-gap module can provide 1.20 volt, it can measure the external supply voltage level. The band-gap 1.20 volt is selected as minus input of comparator and $V_{internal\ R}$ is selected as plus input, the supply voltage of $V_{internal\ R}$ is V_{DD}, the V_{DD} voltage level can be detected by adjusting the voltage level of $V_{internal\ R}$ to compare with band-gap. If N (gpcs[3:0] in decimal) is the number to let $V_{internal\ R}$ closest to band-gap 1.20 volt, the supply voltage V_{DD} can be calculated by using the following equations:

For using Case 1: V_{DD} = [32 / (N+9)] * 1.20 volt ;

For using Case 2: V_{DD} = [24 / (N+1)] * 1.20 volt ;

For using Case 3: V_{DD} = [40 / (N+9)] * 1.20 volt ;

For using Case 4: V_{DD} = [32 / (N+1)] * 1.20 volt ;

More information and sample code, please refer to IDE utility.

5.6 16-bit Timer (Timer16)

A 16-bit hardware timer (Timer16) is implemented in the PMS132, the clock sources of Timer16 may come from system clock (CLK), clock of external crystal oscillator (EOSC), 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. A selector is used to select the interrupt condition of Timer16, whenever overflow occurs, the Timer16 interrupt can be triggered. The hardware diagram of Timer16 is shown as Fig.9. 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 bit 5 of ***integs*** register (IO address 0x0C).

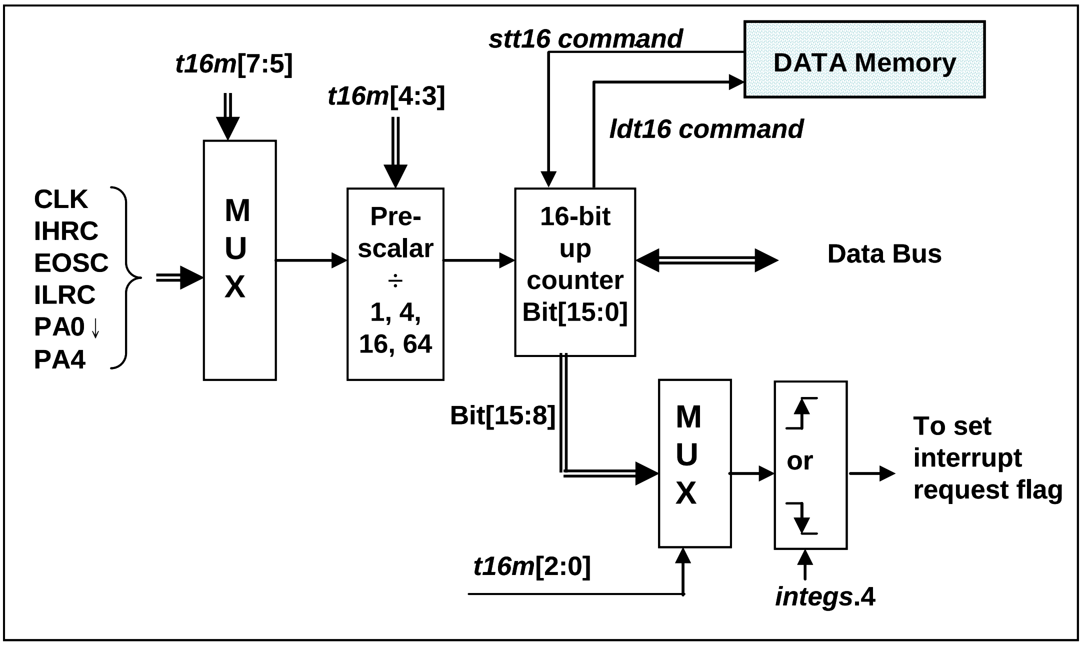


Fig.9: Hardware diagram of Timer16

When using the Timer16, the syntax for Timer16 has been defined in the .INC file. There are three parameters to define the Timer16; 1st parameter is used to define the clock source of Timer16, 2nd parameter is used to define the pre-scalar 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, EOSC, 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
    // if using System Clock = IHRC / 2 = 8 MHz
    // SYSCLK/64 = 8 MHz/64 = 125KHz, about every 512 mS to generate INTRQ.2=1

$ T16M EOSC, /1, BIT13;
    // choose (EOSC/1) as clock source, every 2^14 clocks to generate INTRQ.2=1
    // if EOSC=32768 Hz, 32768 Hz/(2^14) = 2Hz, every 0.5S to generate INTRQ.2=1

$ T16M PA0_F, /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
```

If Timer16 is operated at free running, the frequency of interrupt can be described as below:

$$\mathbf{F_{INTRQ\_T16M} = F_{clock\ source} \div P \div 2^{n+1}}$$

Where, F is the frequency of selected clock source to Timer16;

P is the selection of t16m [4:3]; (1, 4, 16, 64)

N is the nth bit selected to request interrupt service, for example: n=10 if bit 10 is selected.

5.7 8-bit Timer (Timer2/Timer3) with PWM generation

Two 8-bit hardware timers (Timer2 and Timer3) with PWM generation are implemented in the PMS132. The following descriptions thereinafter are for Timer2 only. It is because Timer3 have same structure with Timer2. Please refer to Fig.10 shown the hardware diagram of Timer2, the clock sources of Timer2 may come from system clock, internal high RC oscillator (IHRC), internal low RC oscillator (ILRC), external crystal oscillator (EOSC), PA0, PB0,PA4 and comparator. Bit [7:4] of register tm2c are used to select the clock of Timer2. If IHRC is selected for Timer2 clock source, the clock sent to Timer2 will keep running when using ICE in halt state. The output of Timer2 can be sent to pin PB2, PA3 or PB4, depending on bit [3:2] of tm2c register. A clock pre-scaling module is provided with divided-by- 1, 4, 16, and 64 options, controlled by bit [6:5] of tm2s register; one scaling module with divided-by-1~31 is also provided and controlled by bit [4:0] of tm2s register. In conjunction of pre-scaling function and scaling function, the frequency of Timer2 clock (TM2_CLK) can be wide range and flexible.

The Timer2 counter performs 8-bit up-counting operation only; the counter values can be set or read back by tm2ct register. The 8-bit counter will be clear to zero automatically when its values reach for upper bound register, the upper bound register is used to define the period of timer or duty of PWM. There are two operating modes for Timer2: period mode and PWM mode; period mode is used to generate periodical output waveform or interrupt event; PWM mode is used to generate PWM output waveform with optional 6-bit or 8-bit PWM resolution, Fig.11 shows the timing diagram of Timer2 for both period mode and PWM mode.

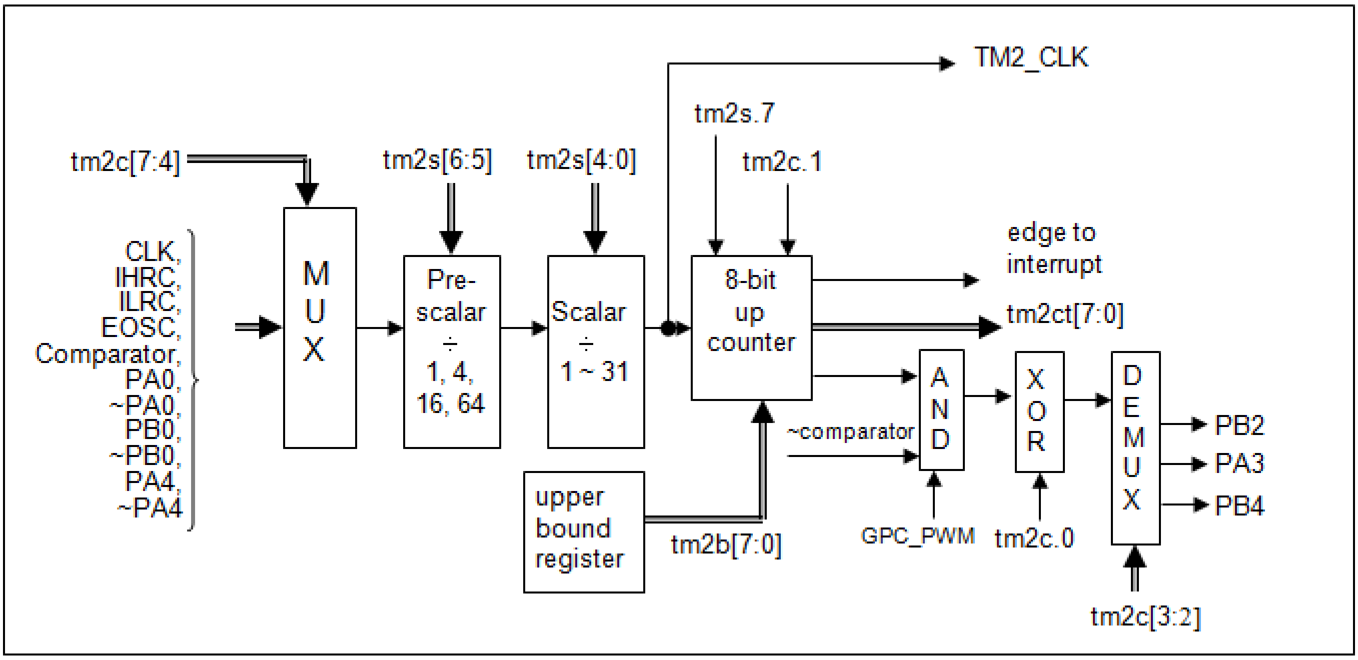
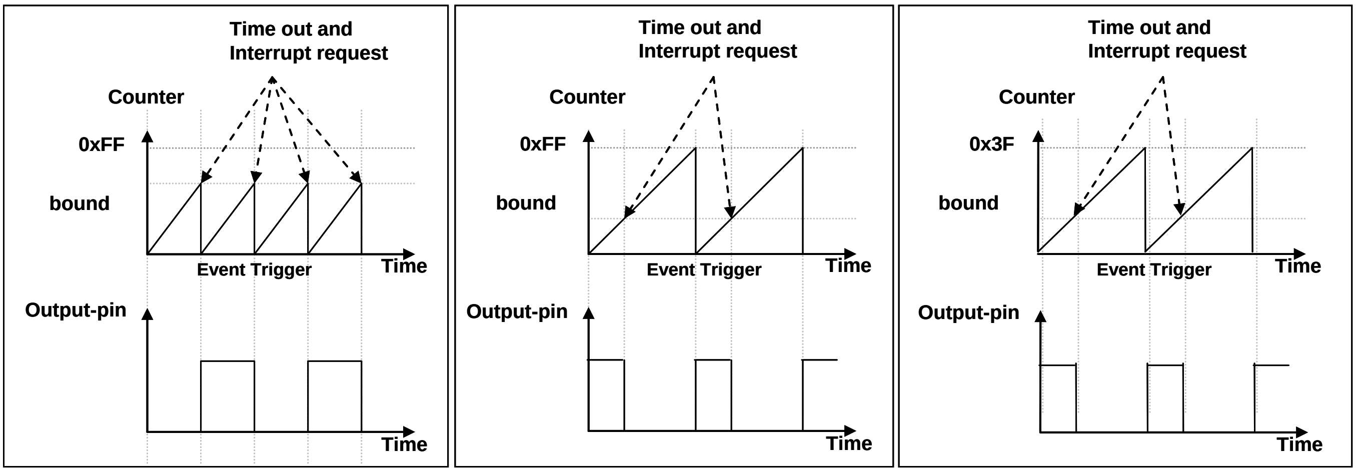


Fig.10: Timer2 hardware diagram

The output of Timer3 can be sent to pin PB5, PB6 or PB7.



Mode 0 – Period Mode **Mode 1 – 8-bit PWM Mode** **Mode 1 – 6-bit PWM Mode**

Fig.11: Timing diagram of Timer2 in period mode and PWM mode (tm2c.1=1)

5.7.1 Using the Timer2 to generate periodical waveform

If periodical mode is selected, the duty cycle of output is always 50%; its frequency can be summarized as below:

Frequency of Output = Y ÷ [2 × (K+1) × S1 × (S2+1)]

Where, Y = tm2c[7:4] : frequency of selected clock source

K = tm2b[7:0] : bound register in decimal

S1 = tm2s[6:5] : pre-scalar (1, 4, 16, 64)

S2 = tm2s[4:0] : scalar register in decimal (1 ~ 31)

Example 1:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s = 0b0000_00000, S1=1, S2=0

➔ frequency of output = 8MHz ÷ [2 × (127＋1) × 1 × (0＋1)] = 31.25KHz

Example 2:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s[7:0] = 0b0111_11111, S1=64 , S2 = 31

➔ frequency = 8MHz ÷ (2 × (127＋1) × 64 × (31＋1)) =15.25Hz

Example 3:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_1111, K=15

tm2s = 0b0000_00000, S1=1, S2=0

➔ frequency = 8MHz ÷ (2 × (15＋1) × 1 × (0＋1)) = 250KHz

Example 4:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_0001, K=1

tm2s = 0b0000_00000, S1=1, S2=0

➔ frequency = 8MHz ÷ (2 × (1＋1) × 1 × (0＋1)) =2MHz

The sample program for using the Timer2 to generate periodical waveform from PA3 is shown as below:

```
Void   FPPA0 (void)
{
        . ADJUST_IC     SYSCLK=IHRC/2, IHRC=16MHz, VDD=5V
        …
        tm2ct = 0x0;
        tm2b = 0x7f;
        tm2s = 0b0_00_00001;                //      8-bit PWM, pre-scalar = 1, scalar = 2
        tm2c = 0b0001_10_0_0;               //      system clock, output=PA3, period mode
        while(1)
        {
                nop;
        }
}
```

5.7.2 Using the Timer2 to generate 8-bit PWM waveform

If 8-bit PWM mode is selected, it should set ***tm2c***[1]=1 and ***tm2s***[7]=0, the frequency and duty cycle of output waveform can be summarized as below:

Frequency of Output = Y ÷ [256 × S1 × (S2+1)]
Duty of Output = (K＋1) ÷ 256

Where, Y = tm2c[7:4] : frequency of selected clock source

K = tm2b[7:0] : bound register in decimal

S1= tm2s[6:5] : pre-scalar (1, 4, 16, 64)

S2 = tm2s[4:0] : scalar register in decimal (1 ~ 31)

Example 1:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s = 0b0000_00000, S1=1, S2=0

➔ frequency of output = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz

➔ duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 2:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s = 0b0111_11111, S1=64, S2=31

➔ frequency of output = 8MHz ÷ (256 × 64 × (31+1)) = 15.25Hz

➔ duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 3:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b1111_1111, K=255

tm2s = 0b0000_00000, S1=1, S2=0

➔ PWM output keep high

➔ duty of output = [(255+1) ÷ 256] × 100% = 100%

Example 4:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0000_1001, K = 9

tm2s = 0b0000_00000, S1=1, S2=0

➔ frequency of output = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz

➔ duty of output = [(9+1) ÷ 256] × 100% = 3.9%

The sample program for using the Timer2 to generate PWM waveform from PA3 is shown as below:

```
void    FPPA0 (void)
{
    .ADJUST_IC      SYSCLK=IHRC/2, IHRC=16MHz, VDD=5V
    wdreset;
    tm2ct = 0x0;
    tm2b = 0x7f;
    tm2s = 0b0_00_00001;              //      8-bit PWM, pre-scalar = 1, scalar = 2
    tm2c = 0b0001_10_1_0;             //      system clock, output=PA3, PWM mode
    while(1)
    {
         nop;
    }
}
```

5.7.3 Using the Timer2 to generate 6-bit PWM waveform

If 6-bit PWM mode is selected, it should set ***tm2c***[1]=1 and ***tm2s***[7]=1, the frequency and duty cycle of output waveform can be summarized as below:

Frequency of Output = Y ÷ [64 × S1 × (S2+1)]
Duty of Output = [(K＋1) ÷ 64] × 100%

Where, tm2c[7:4] = Y : frequency of selected clock source

tm2b[7:0] = K : bound register in decimal

tm2s[6:5] = S1 : pre-scalar (1, 4, 16, 64)

tm2s[4:0] = S2 : scalar register in decimal (1 ~ 31)

Example 1:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0001_1111, K=31

tm2s = 0b1000_00000, S1=1, S2=0

➔ frequency of output = 8MHz ÷ (64 ✕ 1 ✕ (0+1)) = 125KHz

➔ duty = [(31+1) ÷ 64] × 100% = 50%

Example 2:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0001_1111, K=31

tm2s = 0b1111_11111, S1=64, S2=31

➔ frequency of output = 8MHz ÷ (64 × 64 × (31+1)) = 61.03 Hz

➔ duty of output = [(31+1) ÷ 64] **×** 100% = 50%

Example 3:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0011_1111, K=63

tm2s = 0b1000_00000, S1=1, S2=0

➔ PWM output keep high

➔ duty of output = [(63+1) ÷ 64] × 100% = 100%

Example 4:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0000_0000, K=0

tm2s = 0b1000_00000, S1=1, S2=0

➔ frequency = 8MHz ÷ (64 × 1 × (0+1)) = 125KHz

➔ duty = [(0+1) ÷ 64] × 100% =1.5%

5.8 11-bit PWM Generator

Three 11-bit hardware PWM generators (PWMG0, PWMG1 & PWMG2) are implemented in the PMS132. The following descriptions thereinafter are for PWMG0 only. It is because PWMG1 & PWMG2 have the same structures and functions with PWMG0.

Their individual outputs are listed as below:

- PWMG0 – PA0, PB4, PB5
- PWMG1 – PA4, PB6, PB7
- PWMG2 – PA3, PB2, PB3, PA5 (Only PA5 open drain output, and ICE does not support.)

5.8.1 PWM Waveform

A PWM output waveform (Fig.12) has a time-base (T_{Period} = Time of Period) and a time with output high level (Duty Cycle). The frequency of the PWM output is the inverse of the period ($f_{PWM} = 1/T_{Period}$), the resolution of the PWM is the clock count numbers for one period (N bits resolution, $2^N \times T_{clock} = T_{Period}$).

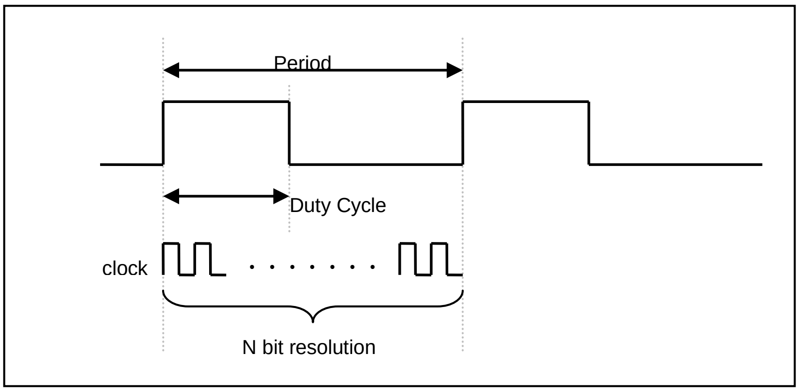


Fig.12: PWM Output Waveform

5.8.2 Hardware and Timing Diagram

Three 11-bit hardware PWM generators are built inside the PMS132; Fig.13 shows the hardware diagram PWMG0 as an example. The clock source can be IHRC or system clock and output pin can be selected via ***pwmc*** register selection. The period of PWM waveform is defined in the PWM upper bond high and low registers, the duty cycle of PWM waveform is defined in the PWM duty high and low registers.

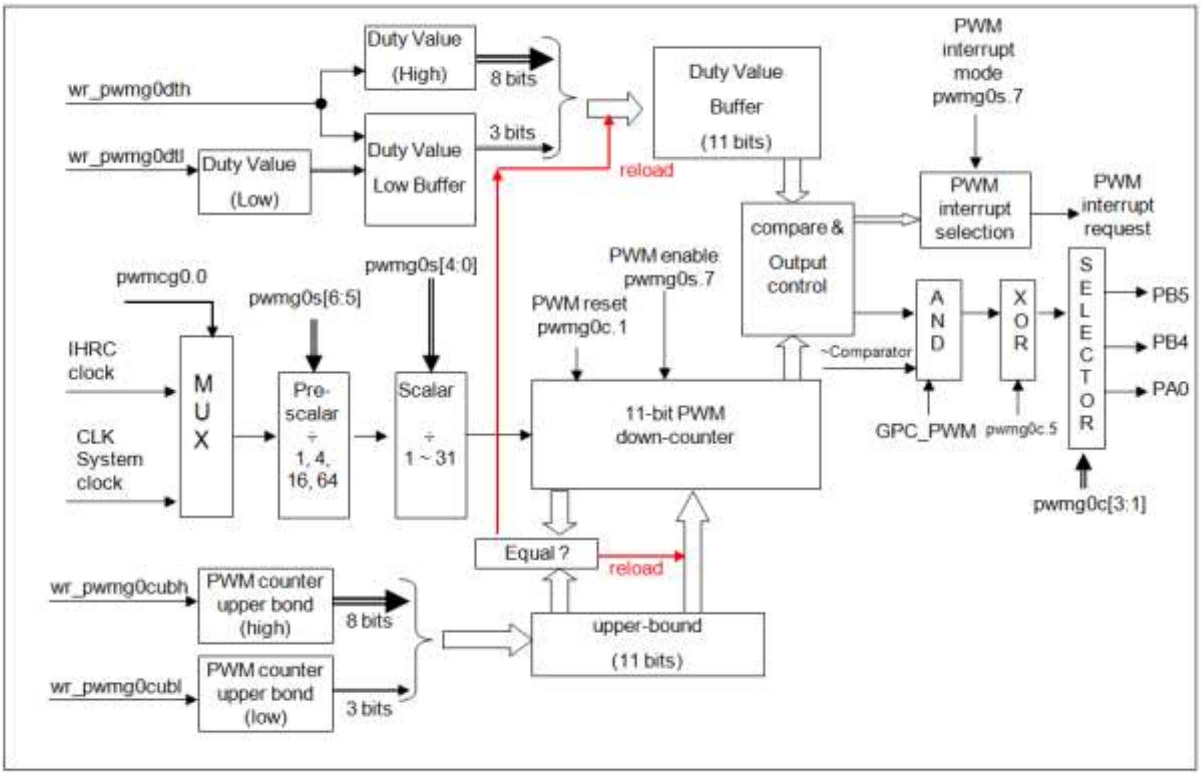


Fig.13: Hardware Diagram of 11-bit PWM Generator

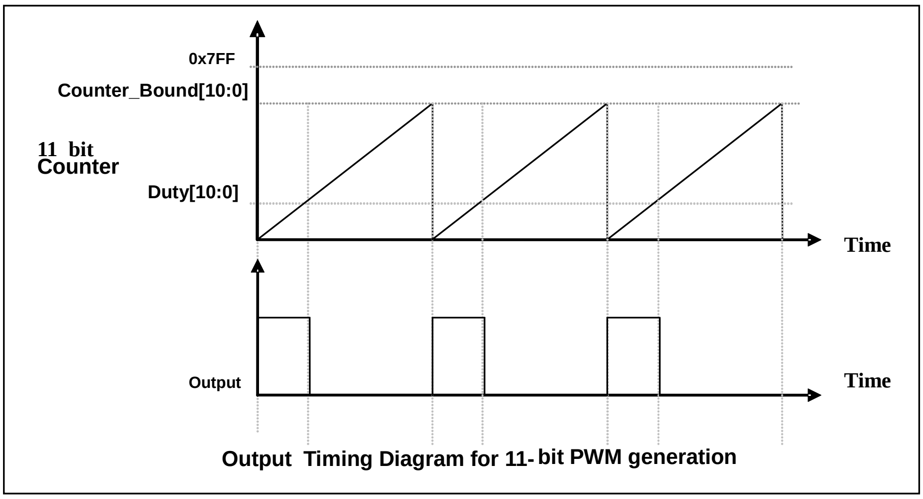


Fig.14: Output Timing Diagram of 11-bit PWM Generator

5.8.3 Equations for 11-bit PWM Generator

If F_{IHRC} is the frequency of IHRC oscillator and IHRC is the chosen clock source for 11-bit PWM generator, the PWM frequency and duty cycle in time will be:

$$\textbf{Frequency of PWM Output} = F_{IHRC} \div [P \times K \times CB]$$

$$\textbf{Duty Cycle of PWM Output (in time)} = (1/F_{IHRC}) * [DB \div CB]$$

Where, pwms[6:5] = **P** ; pre-scalar

pwms[4:0] = **K** ; scalar

Duty_Bound[10:0] = {pwmdth[7:0],pwmdtl[7:5]} = **DB**; duty bound

Counter_Bount[10:0] = {pwmcubh[7:0], pwmcubl[7:5]} = **CB**; counter bount

5.9 WatchDog Timer

The watchdog timer (WDT) is a counter with clock coming from ILRC. WDT can be cleared by power-on-reset or by command ***wdreset*** at any time. There are four different timeout periods of watchdog timer to be chosen by setting the ***misc*** register, it is:

- 8k ILRC clocks period if register misc[1:0]=00 (default)
- 16k ILRC clocks period if register misc[1:0]=01
- 64k ILRC clocks period if register misc[1:0]=10
- 256k 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, PMS132 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig.15.

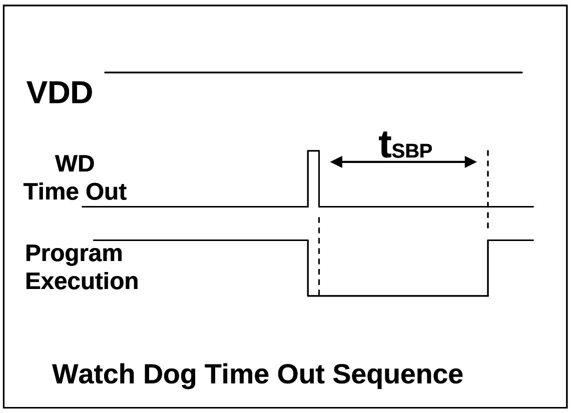


Fig.15: Sequence of Watch Dog Time Out

5.10 Interrupt

There are eight interrupt lines for PMS132:

- External interrupt PA0/PB5
- External interrupt PB0/PA4
- ADC interrupt
- Timer16 interrupt
- GPC interrupt
- PWMG0 interrupt
- Timer2 interrupt
- Timer3 interrupt

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.16. 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 ***integs***. 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, user should manipulate the memory using carefully. By adjusting the memory location of stack point, the depth of stack pointer for every CPU could be fully specified by user to achieve maximum flexibility of system.

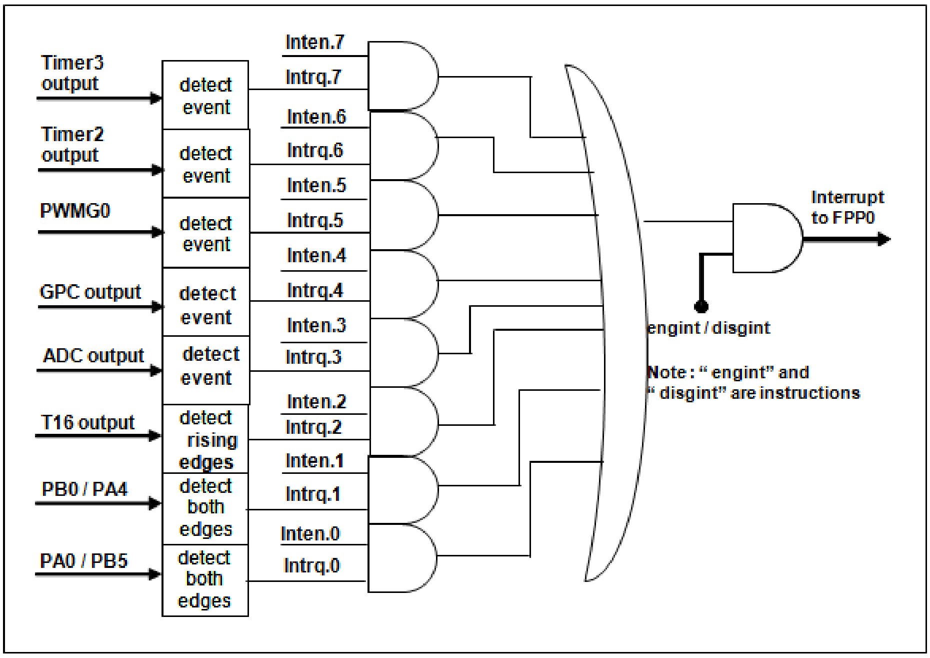


Fig.16: Hardware diagram of interrupt controller

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.

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    (INTRQ.0)
    {                               //    Here for PA0 interrupt service routine
          INTRQ.0   =    0;
          ...
    }
    ...
    POPAF                           //    restore ALU and FLAG register
}
```

5.11 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 occasionally and Power-Down mode is used in the system which needs power down deeply with seldom wake-up. Fig.17 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	EOSC
STOPSYS	Stop	Stop	Stop
STOPEXE	No Change	No Change	No Change

Fig.17: Differences in oscillator modules between STOPSYS and STOPEXE

5.11.1 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. Wake-up from input pins can be considered as a continuation of normal execution, the detail information for Power-Save mode shows below:

- IHRC, ILRC and EOSC oscillator modules: No change, keep active if it was enabled
- 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 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;

…
```

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

5.11.2 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 PMS132 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: ANY IO toggle.
- If PA or PB is input mode and set to analog input by ***padier*** or ***pbdier*** register, it can NOT be used to wake-up the system.

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
  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/2
```

5.11.3 Wake-up

After entering the Power-Down or Power-Save modes, the PMS132 can be resumed to normal operation by toggling IO pins, Timer16 interrupt is available for Power-Save mode ONLY. Fig.18 shows the differences in wake-up sources between STOPSYS and STOPEXE.

Differences in wake-up sources between STOPSYS and STOPEXE		
	IO Toggle	T16 Interrupt
STOPSYS	Yes	No
STOPEXE	Yes	Yes

Fig.18: Differences in wake-up sources between Power-Save mode and Power-Down mode

When using the IO pins to wake-up the PMS132, registers ***padier*** should be properly set to enable the wake-up function for every corresponding pin. The time for normal wake-up is about 3000 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 45 ILRC clocks from IO toggling.

Suspend mode	Wake-up mode	Wake-up time (t_{WUP}) from IO toggle
STOPEXE suspend or STOPSYS suspend	Fast wake-up	45 * $T_{ILRC,}$ Where T_{ILRC} is the time period of ILRC
STOPEXE suspend or STOPSYS suspend	Normal wake-up	3000 * T_{ILRC}, Where T_{ILRC} is the clock period of ILRC

Please notice that when Fast boot-up is selected, no matter which wake-up mode is selected in misc.5, the wake-up mode will be forced to be FAST. If Normal boot-up is selected, the wake-up mode is determined by misc.5.

5.12 IO Pins

All the pins can be independently set into two states output or input by configuring the data registers (***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 output low, the pull-up 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 5 shows the configuration table of bit 0 of port A. The hardware diagram of IO buffer is also shown as Fig.19.

pa.0	*pac.0*	*paph.0*	Description
X	0	0	Input without pull-up resistor
X	0	1	Input with pull-up resistor
0	1	X	Output low without pull-up resistor
1	1	0	Output high without pull-up resistor
1	1	1	Output high with pull-up resistor

Table 5: PA0 Configuration Table

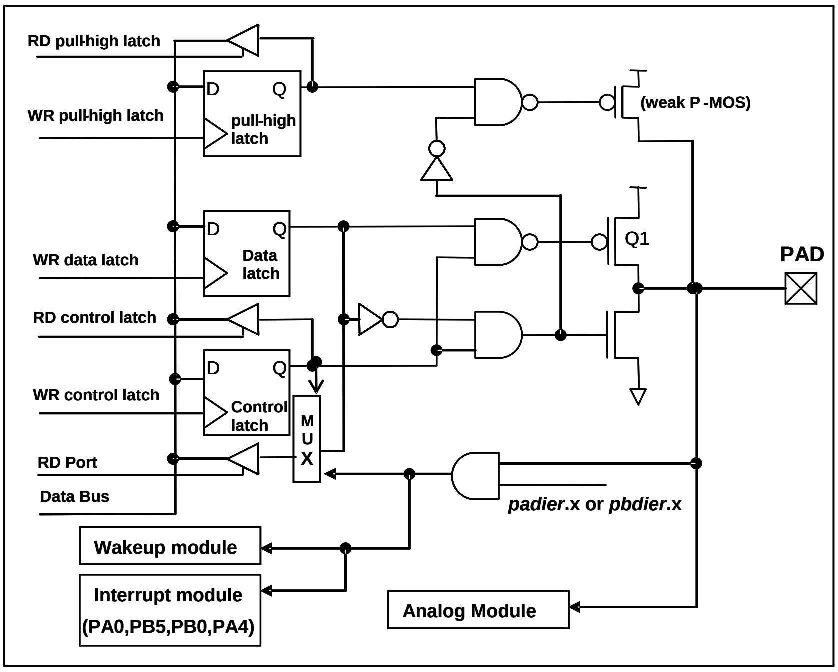


Fig.19: Hardware diagram of IO buffer

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 PMS132 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*** and ***pbdier*** to high***.*** The same reason, ***padier***.0 should be set high when PA0 is used as external interrupt pin, ***pbdier***.0 for PB0, ***padier***.4 for PA4 and ***pbdier***.5 for PB5.

5.13 Reset and LVR

5.13.1 Reset

There are many causes to reset the PMS132, once reset is asserted, most of all the registers in PMS132 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.

5.13.2 LVR reset

By code option, there are 8 different levels of LVR for reset: 4.0V, 3.5V, 3.0V, 2.75V, 2.5V, 2.2V, 2.0V and 1.8V; usually, user selects LVR reset level to be in conjunction with operating frequency and supply voltage.

5.14 Analog-to-Digital Conversion (ADC) module

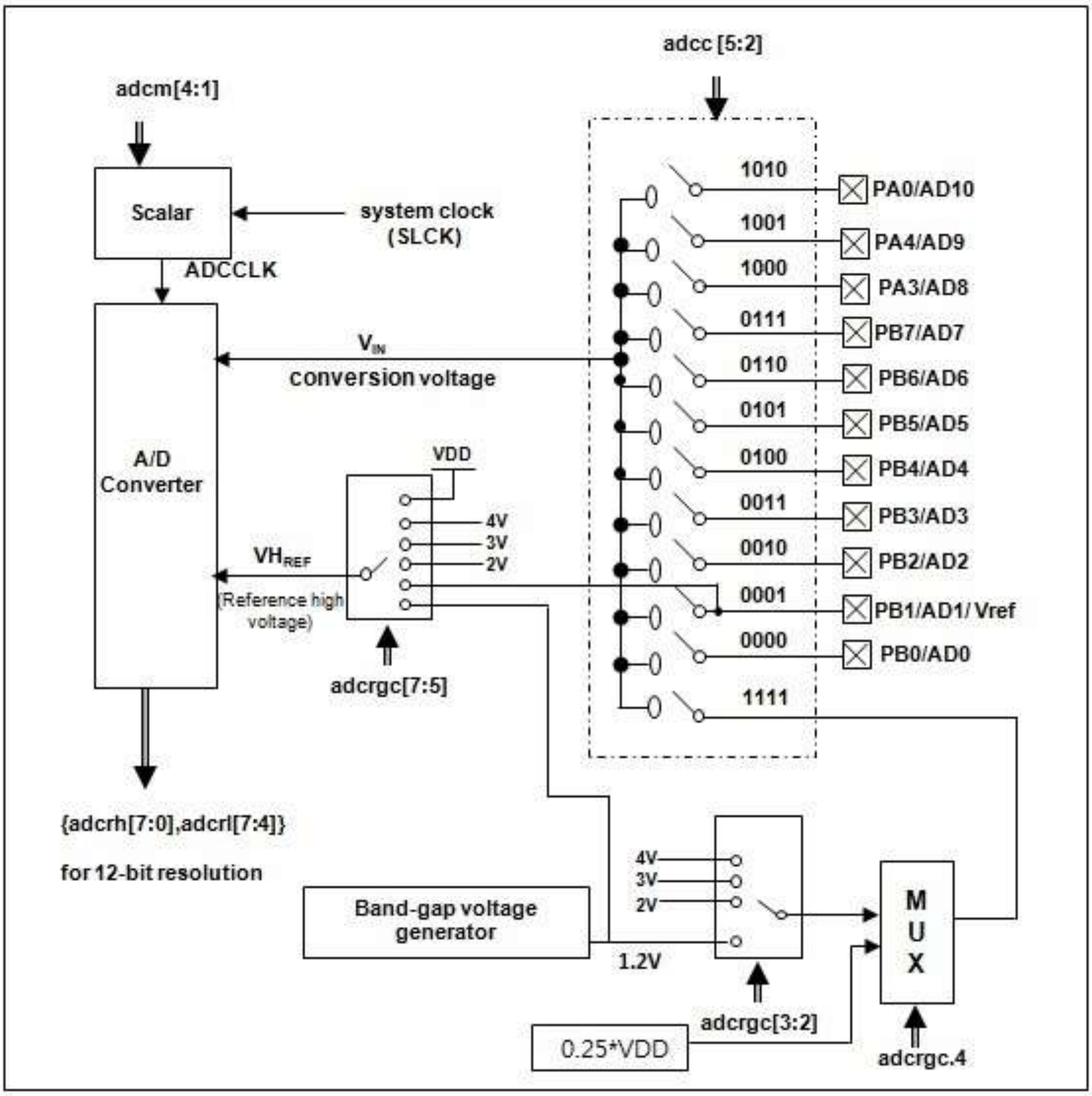


Fig.20: ADC Block Diagram

There are seven registers when using the ADC module, which are:

- ADC Control Register (***adcc***)
- ADC Regulator Control Register (***adcrgc***)
- ADC Mode Register (***adcm***)
- ADC Result High/Low Register (***adcrh***, ***adcrl***)
- Port A/B Digital Input Enable Register (***padier***, ***pbdier***)

The following steps are required to do the AD conversion procedure:

(1) Configure the voltage reference high by ***adcrgc*** register

(2) Configure the AD conversion clock by ***adcm*** register

(3) Configure the pin as analog input by ***padier***, ***pbdier*** register

(4) Select the ADC input channel by ***adcc*** register

(5) Enable the ADC module by ***adcc*** register

(6) Delay a certain amount of time after enabling the ADC module
Condition 1: using the internal voltage reference high which are 2V, 3V, 4V or the input channel is bandgap
It must delay more than 1 ms when the time of 200 AD clocks is less than 1ms. Or it must delay 200 AD clocks when the time of 200 AD clocks is larger than 1ms
Condition 2: without using any internal 2V, 3V, 4V, bandgap voltage
It needs to delay 200 AD clocks only.

(7) Execute the AD conversion and check if ADC data is ready
set ‘1’ to ***addc***.6 to start the conversion and check whether ***addc***.6 is ‘1’

(8) Read the ADC result registers:
First read the ***adcrh*** register and then read the ***adcrl*** register.

If user power down the ADC and enable the ADC again, be sure to go to step 6 to confirm the ADC becomes ready before the conversion.

5.14.1 The input requirement for AD conversion

For the AD conversion to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the voltage reference high level and discharge to the voltage reference low level. The analog input model is shown as Fig.21, the signal driving source impedance (Rs) and the internal sampling switch impedance (Rss) will affect the required time to charge the capacitor C_{HOLD} directly. The internal sampling switch impedance may vary with ADC supply voltage; the signal driving source impedance will affect accuracy of analog input signal. User must ensure the measured signal is stable before sampling; therefore, the maximum signal driving source impedance is highly dependent on the frequency of signal to be measured. The recommended maximum impedance for analog driving source is about 10KΩ under 500KHz input frequency.

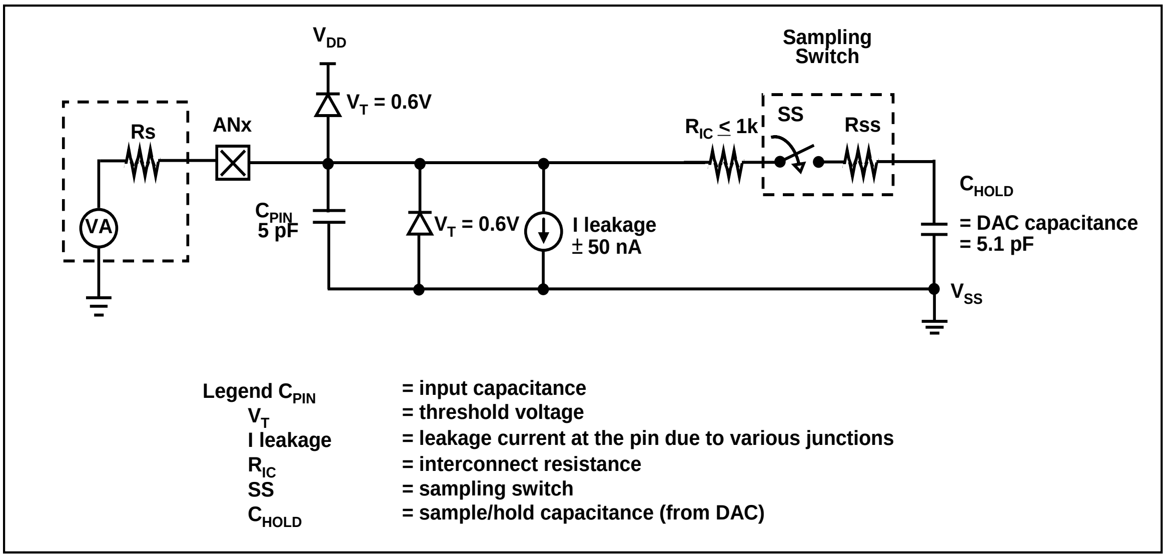

V_{DD}
Sampling Switch
V_T = 0.6V
Rs
ANx
R_{IC} ≤ 1k
SS
Rss
C_{HOLD}
VA
C_{PIN} 5 pF
V_T = 0.6V
I leakage ± 50 nA
= DAC capacitance = 5.1 pF
V_{SS}
Legend C_{PIN} = input capacitance
V_T = threshold voltage
I leakage = leakage current at the pin due to various junctions
R_{IC} = interconnect resistance
SS = sampling switch
C_{HOLD} = sample/hold capacitance (from DAC)

Fig.21: Analog Input Model

Before starting the AD conversion, the minimum signal acquisition time should be met for the selected analog input signal, the selection of ADCLK must be met the minimum signal acquisition time.

5.14.2 Select the reference high voltage

The ADC reference high voltage can be selected via bit[7:5] of register ***adcrgc*** and its option can be V_{DD}, 4V, 3V, 2V, band-gap (1.20V) reference voltage or PB1 from external pin.

5.14.3 ADC clock selection

The clock of ADC module (ADCLK) can be selected by ***adcm*** register; there are 8 possible options for ADCLK from CLK÷1 to CLK÷128 (CLK is the system clock). Due to the signal acquisition time T_{ACQ} is one clock period of ADCLK, the ADCLK must meet that requirement. The recommended ADC clock is to operate at 2us.

5.14.4 Configure the analog pins

There are 12 analog signals can be selected for AD conversion, 11 analog input signals come from external pins and one is from internal band-gap reference voltage or 0.25*V_{DD}. There are 4 voltage levels selectable for the internal band-gap reference, they are 1.2V, 2V, 3V and 4V. For external pins, the analog signals are shared with Port A[0], Port A[3], Port A[4], and Port B[7:0]. To avoid leakage current at the digital circuit, those pins defined for analog input should disable the digital input function (set the corresponding bit of ***padier or pbdier*** register to be 0).

The measurement signals of ADC belong to small signal; it should avoid the measured signal to be interfered during the measurement period, the selected pin should (1) be set to input mode (2) turn off weak pull-high resistor (3) set the corresponding pin to analog input by port A/B digital input disable register (***padier / pbdier***).

5.14.5 Using the ADC

The following example shows how to use ADC with PB0~PB3.

First, defining the selected pins:

```
PBC       =    0B_XXXX_0000;        //    PB0 ~ PB3 as Input
PBPH      =    0B_XXXX_0000;        //    PB0 ~ PB3 without pull-high
PBDIER    =    0B_XXXX_0000;        //    PB0 ~ PB3 digital input is disabled
```

Next, setting ***ADCC*** register, example as below:

```
$   ADCC   Enable, PB3;             //    set PB3 as ADC input
$   ADCC   Enable, PB2;             //    set PB2 as ADC input
$   ADCC   Enable, PB0;             //    set PB0 as ADC input
```

Next, setting ***ADCM*** register, example as below:

```
$   ADCM   12BIT, /16;              //    recommend /16 @System Clock=8MHz
$   ADCM   12BIT, /8;               //    recommend /8 @System Clock=4MHz
```

Next, delay 400us, example as below:

```
.Delay 8*2*200;                         //      System Clock=8MHz
.Delay 4*2*200;                         //      System Clock=4MHz
```

Then, start the ADC conversion:

```
AD_START =      1;                      //      start ADC conversion
while (! AD_DONE)   NULL;               //      wait ADC conversion result
```

Finally, it can read ADC result when AD_DONE is high:

```
WORD            Data;                   //      two bytes result: ADCRH and ADCRL
Data$1     =    ADCRH
Data$0     =    ADCRL;
Data       =    Data >> 4;
```

The ADC can be disabled by using the following method:

```
$   ADCC   Disable;
```

or

```
ADCC            =       0;
```

5.15 Multiplier

There is an 8x8 multiplier on-chip to enhance hardware capability in arithmetic function, its multiplication is an 8 x8 unsigned operation and can be finished in one clock cycle. Before issuing the *mul* command, both multiplicand and multiplicator must be put on ACC and register *mulop* (0x08); After *mul* command, the high byte result will be put on register *mulrh* (0x09) and low byte result on ACC. The hardware diagram of this multiplier is shown as Fig.22.

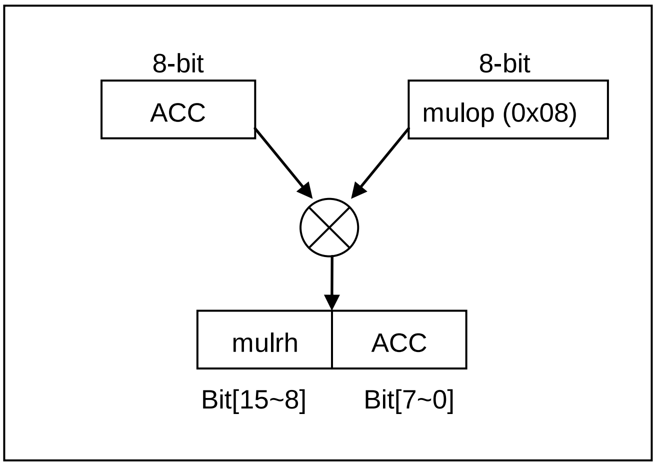


Fig.22: Block diagram of hardware multiplier

6. IO Registers

6.1. ACC Status Flag Register (*flag*), IO address = 0x00

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. Please do not use.
3	0	R/W	OV (Overflow Flag). This bit is set to be 1 whenever the sign operation is overflow.
2	0	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	0	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	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.

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

Bit	Reset	R/W	Description	
7 - 5	111	R/W	System clock (CLK) selection:	
			Type 0, clkmd[3]=0	Type 1, clkmd[3]=1
			000: IHRC÷4 001: IHRC÷2 010: IHRC 011: EOSC÷4 100: EOSC÷2 101: EOSC 110: ILRC÷4 111: ILRC (default)	000: IHRC÷16 001: IHRC÷8 010: ILRC÷16 (ICE does NOT Support.) 011: IHRC÷32 100: IHRC÷64 101: EOSC÷8 11x: reserved.
4	1	R/W	Internal High RC Enable. 0 / 1: disable / enable	
3	0	R/W	Clock Type Select. This bit is used to select the clock type in bit [7:5]. 0 / 1: Type 0 / Type 1	
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	R/W	Enable interrupt from Timer3. 0 / 1: disable / enable.
6	0	R/W	Enable interrupt from Timer2. 0 / 1: disable / enable.
5	0	R/W	Enable interrupt from PWMG0. 0 / 1: disable / enable.
4	0	R/W	Enable interrupt from comparator. 0 / 1: disable / enable.
3	0	R/W	Enable interrupt from ADC. 0 / 1: disable / enable.
2	0	R/W	Enable interrupt from Timer16 overflow. 0 / 1: disable / enable.
1	0	R/W	Enable interrupt from PB0/PA4. 0 / 1: disable / enable.
0	0	R/W	Enable interrupt from PA0/PB5. 0 / 1: disable / enable.

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

Bit	Reset	R/W	Description
7	-	R/W	Interrupt Request from Timer3, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
6	-	R/W	Interrupt Request from Timer2, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
5	-	R/W	Interrupt Request from PWMG0, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
4	-	R/W	Interrupt Request from comparator, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
3	-	R/W	Interrupt Request from ADC, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
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 PB0/PA4, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	R/W	Interrupt Request from pin PA0/PB5, this bit is set by hardware and cleared by software. 0 / 1: No Request / request

6.6. Multiplier Operand Register (*mulop*), IO address = 0x08

Bit	Reset	R/W	Description
7 - 0	-	R/W	Operand for hardware multiplication operation.

6.7. Multiplier Result High Byte Register (*mulrh*), IO address = 0x09

Bit	Reset	R/W	Description
7 - 0	-	RO	High byte result of multiplication operation (read only).

6.8. 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: EOSC 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

6.9. External Oscillator setting Register (*eoscr*), IO address = 0x0a

Bit	Reset	R/W	Description
7	0	WO	Enable external crystal oscillator. 0 / 1 : Disable / Enable
6 - 5	00	WO	External crystal oscillator selection. 00 : reserved 01 : Low driving capability, for lower frequency, ex: 32KHz crystal oscillator (reserved) 10 : Middle driving capability, for middle frequency, ex: 1MHz crystal oscillator 11 : High driving capability, for higher frequency, ex: 4MHz crystal oscillator
4 - 1	-	-	Reserved. Please keep 0 for future compatibility.
0	0	WO	Power-down the Band-gap and LVR hardware modules. 0 / 1: normal / power-down.

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

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved.
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	PB0/PA4 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/PB5 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.

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

Bit	Reset	R/W	Description
7	1	WO	Enable PA7 digital input and wake-up event. 1 / 0 : enable / disable. This bit should be set to low to prevent leakage current when external crystal oscillator is used. If this bit is set to low, PA7 can NOT be used to wake-up the system.
6	1	WO	Enable PA6 digital input and wake-up event. 1 / 0 : enable / disable. This bit should be set to low to prevent leakage current when external crystal oscillator is used. If this bit is set to low, PA6 can NOT be used to wake-up the system.
5	1	WO	Enable PA5 digital input and wake-up event. 1 / 0 : enable / disable. This bit can be set to low to disable wake-up from PA5 toggling.
4	1	WO	Enable PA4 digital input and wake-up event. 1 / 0 : enable / disable. This bit should be set to low when PA4 is assigned as AD input to prevent leakage current. If this bit is set to low, PA4 can NOT be used to wake-up the system.
3	1	WO	Enable PA3 digital input and wake-up event. 1 / 0 : enable / disable. This bit should be set to low when PA3 is assigned as AD input to prevent leakage current. If this bit is set to low, PA3 can NOT be used to wake-up the system.
2 - 1	1	WO	Reserved
0	1	WO	Enable PA0 digital input and wake-up event and interrupt request. 1 /0: enable / disable. This bit can be set to low to prevent leakage current when PA0 is assigned as AD input, and to disable wake-up from PA0 toggling and interrupt request from this pin.

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

Bit	Reset	R/W	Description
7 - 0	0xFF	WO	Enable PB7~PB0 digital input to prevent leakage when the pin is assigned for AD input. When disable is selected, the wakeup function from this pin is also disabled. 0 / 1 : disable / enable

.

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Data register for Port A.

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

Bit	Reset	R/W	Description
7 - 0	0x00	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 Please note that PA5 can be INPUT or OUTPUT LOW ONLY, the output state will be tri-state when PA5 is programmed into output mode with data 1.

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port A and this pull high function is active only for input mode. 0 / 1 : disable / enable

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Data register for Port B.

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B control register. This register is used to define input mode or output mode for each corresponding pin of port B. 0 / 1: input / output

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port B and this pull high function is active only for input mode. 0 / 1 : disable / enable

6.19. Miscellaneous Register (*misc*), IO address = 0x17

Bit	Reset	R/W	Description
7 - 6	-	-	Reserved. (keep 0 for future compatibility)
5	0	WO	Enable fast Wake-up. Fast wake-up is NOT supported when EOSC is enabled. 0: Normal wake-up. The wake-up time is 3000 ILRC clocks (Not for fast boot-up) 1: Fast wake-up. The wake-up time is 45 ILRC clocks + oscillator stable time. If wake-up from STOPEXE suspend, there is no oscillator stable time; If wake-up from STOPSYS suspend, it will be IHRC or ILRC stable time from power-on.
4 - 3	-	-	Reserved. (keep 0 for future compatibility)
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 - 0	00	WO	Watch dog time out period. 00: 8k ILRC clock period 01: 16k ILRC clock period 10: 64k ILRC clock period 11: 256k ILRC clock period

6.20. Comparator Control Register (*gpcc*), IO address = 0x18

Bit	Reset	R/W	Description
7	0	R/W	Enable comparator. 0 / 1 : disable / enable When this bit is set to enable, please also set the corresponding analog input pins to be digital disable to prevent IO leakage.
6	-	RO	Comparator result of comparator. 0: plus input < minus input 1: plus input > minus input
5	0	R/W	Select whether the comparator result output will be sampled by TM2_CLK? 0: result output NOT sampled by TM2_CLK 1: result output sampled by TM2_CLK
4	0	R/W	Inverse the polarity of result output of comparator. 0: polarity is NOT inversed. 1: polarity is inversed.
3 - 1	000	R/W	Selection the minus input (-) of comparator. 000 : PA3 001 : PA4 010 : Internal 1.20 volt band-gap reference voltage 011 : $V_{\text{internal R}}$ 100 : PB6 101 : PB7 11X: reserved
0	0	R/W	Selection the plus input (+) of comparator. 0 : $V_{\text{internal R}}$ 1 : PA4

6.21. Comparator Selection Register (*gpcs*), IO address = 0x19

Bit	Reset	R/W	Description
7	0	WO	Comparator output enable (to PA0). 0 / 1 : disable / enable
6	0	-	Reserved
5	0	WO	Selection of high range of comparator.
4	0	WO	Selection of low range of comparator.
3 - 0	0000	WO	Selection the voltage level of comparator. 0000 (lowest) ~ 1111 (highest)

6.22. Reset Status Register (*rstst*), IO address = 0x1b

Bit	Reset (POR only)	R/W	Description
7	0	R/W	MCU had been reset by Watch-Dog time-out? This bit is set to high whenever reset occurs from watch-dog time-out, and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes..
6	0	R/W	MCU had been reset by invalid code? This bit is set to high whenever reset occurs from invalid instruction code, and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes..
5	0	-	Reserved. Please keep 0.
4	-	-	Reserved. Please keep 1.
3	-	R/W	MCU reset from external reset pin (PA5)? This bit is set to high whenever reset occurs from PA5 pin, and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes.
2	-	R/W	V_{DD} had been lower than 4V? This bit is set to high whenever VDD under 4V and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes.
1	-	R/W	V_{DD} had been lower than 3V? This bit is set to high whenever VDD under 3V and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes.
0	-	R/W	V_{DD} had been lower than 2V? This bit is set to high whenever VDD under 2V and reset only when writing “0” to clear this bit or POR (power-on-reset) happens. 0 / 1 : No / Yes.

6.23. Timer2 Control Register (*tm2c*), IO address = 0x1c

Bit	Reset	R/W	Description
7 - 4	0000	R/W	Timer2 clock selection. 0000 : disable 0001 : CLK (system clock) 0010 : IHRC 0011 : EOSC 0100 : ILRC 0101 : comparator output 011x : reserved 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge) **Notice:** In ICE mode and IHRC is selected for Timer2 clock, the clock sent to Timer2 does NOT be stopped, Timer2 will keep counting when ICE is in halt state.
3 - 2	00	R/W	Timer2 output selection. 00 : disable 01 : PB2 10 : PA3 11 : PB4
1	0	R/W	Timer2 mode selection. 0 / 1 : period mode / PWM mode
0	0	R/W	Enable to inverse the polarity of Timer2 output. 0 / 1: disable / enable.

6.24. Timer2 Counter Register (*tm2ct*), IO address = 0x1d

Bit	Reset	R/W	Description
7–0	0x00	R/W	Bit [7:0] of Timer2 counter register.

6.25. Timer2 Scalar Register (*tm2s*), IO address = 0x1e

Bit	Reset	R/W	Description
7	0	WO	PWM resolution selection. 0 : 8-bit 1 : 6-bit
6 - 5	00	WO	Timer2 clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
4 - 0	00000	WO	Timer2 clock scalar.

6.26. Timer2 Bound Register (*tm2b*), IO address = 0x09

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer2 bound register.

6.27. PWMG0 control Register (*pwmg0c*), IO address = 0x20

Bit	Reset	R/W	Description
7	0	WO	Enable PWMG0 generator. 0 / 1 : disable / enable.
6	-	RO	Output status of PWMG0 generator.
5	0	WO	Enable to inverse the polarity of PWMG0 generator output. 0 / 1 : disable / enable.
4	0	WO	PWMG0 counter reset. Writing “1” to clear PWMG0 counter and this bit will be self clear to 0 after counter reset.
3 - 1	0	WO	Select PWM output pin for PWMG0. 000: none 001: PB5 011: PA0 100: PB4 Others: reserved
0	0	WO	Clock source of PWMG0 generator. 0 : SYSCLK 1 : IHRC or IHRC * 2 (by Code Option: PWM_Source)

6.28. PWMG0 Scalar Register (*pwmg0s*), IO address = 0x21

Bit	Reset	R/W	Description
7	0	WO	PWMG0 interrupt mode. 0: Generate interrupt when counter is 0. 1: Generate interrupt when counter matches the duty value
6 - 5	0	WO	PWMG0 clock pre-scalar. 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 - 0	0	WO	PWMG0 clock divider.

6.29. PWMG0 Counter Upper Bound High Register (*pwmg0cubh*), IO address = 0x24

Bit	Reset	R/W	Description
7 - 0	-	WO	Bit[10:3] of PWMG0 counter upper bound.

6.30. PWMG0 Counter Upper Bound Low Register (*pwmg0cubl*), IO address = 0x25

Bit	Reset	R/W	Description
7 - 5	-	WO	Bit[2:0] of PWMG0 counter upper bound.
4 - 0	-	-	Reserved

6.31. PWMG0 Duty Value High Register (*pwmg0dth*), IO address = 0x22

Bit	Reset	R/W	Description
7 - 0	-	WO	Duty values bit[10:3] of PWMG0.

6.32. PWMG0 Duty Value Low Register (*pwmg0dtl*), IO address = 0x23

Bit	Reset	R/W	Description
7 - 5	-	WO	Duty values bit [2:0] of PWMG0.
4 - 0	-	-	Reserved

Note: It’s necessary to write PWMG0 Duty_Value Low Register before writing PWMG0 Duty_Value High Register.

6.33. Timer3 Control Register (*tm3c*), IO address = 0x32

Bit	Reset	R/W	Description
7 - 4	0000	R/W	Timer3 clock selection. 0000 : disable 0001 : CLK (system clock) 0010 : IHRC 0011 : EOSC 0100 : ILRC 0101 : comparator output 011x : reserved 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge) **Notice:** In ICE mode and IHRC is selected for Timer3 clock, the clock sent to Timer3 does NOT be stopped, Timer3 will keep counting when ICE is in halt state.
3 - 2	00	R/W	Timer3 output selection. 00 : disable 01 : PB5 10 : PB6 11 : PB7
1	0	R/W	Timer3 mode selection. 0 / 1 : period mode / PWM mode
0	0	R/W	Enable to inverse the polarity of Timer3 output. 0 / 1: disable / enable

6.34. Timer3 Counter Register (*tm3ct*), IO address = 0x33

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Bit [7:0] of Timer3 counter register.

6.35. Timer3 Scalar Register (*tm3s*), IO address = 0x34

Bit	Reset	R/W	Description
7	0	WO	PWM resolution selection. 0 : 8-bit 1 : 6-bit
6 - 5	00	WO	Timer3 clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
4 - 0	00000	WO	Timer3 clock scalar.

6.36. Timer3 Bound Register (*tm3b*), IO address = 0x3f

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer3 bound register.

6.37. ADC Control Register (*adcc*), IO address = 0x3b

Bit	Reset	R/W	Description
7	0	R/W	Enable ADC function. 0/1: Disable/Enable.
6	0	R/W	ADC process control bit. Read “1” to indicate the ADC is ready.
5 - 2	0000	R/W	Channel selector. These four bits are used to select input signal for AD conversion. 0000: PB0/AD0, 0001: PB1/AD1, 0010: PB2/AD2, 0011: PB3/AD3, 0100: PB4/AD4, 0101: PB5/AD5, 0110: PB6/AD6, 0111: PB7/AD7, 1000: PA3/AD8, 1001: PA4/AD9, 1010: PA0/AD10, 1111: (Channel F) Band-gap reference voltage or 0.25*V_{DD} Others: reserved
0 - 1	-	-	Reserved. (keep 0 for future compatibility)

6.38. ADC Mode Register (*adcm*), IO address = 0x3c

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved (keep 0 for future compatibility)
3 - 1	000	WO	ADC clock source selection. 000: CLK (system clock) ÷ 1, 001: CLK (system clock) ÷ 2, 010: CLK (system clock) ÷ 4, 011: CLK (system clock) ÷ 8, 100: CLK (system clock) ÷ 16, 101: CLK (system clock) ÷ 32, 110: CLK (system clock) ÷ 64, 111: CLK (system clock) ÷ 128,
0	-	-	Reserved

6.39. ADC Regulator Control Register (*adcrgc*), IO address = 0x3d

Bit	Reset	R/W	Description
7 - 5	000	WO	These three bits are used to select input signal for ADC reference high voltage. 000: V_{DD}, 001: 2V, 010: 3V, 011: 4V, 100: PB1, 101: Band-gap 1.20 volt reference voltage Others: reserved
4	0	WO	ADC channel F selector: 0: Band-gap reference voltage 1: 0.25*V_{DD}. The deviation is within ±0.01*V_{DD} mostly.
3 - 2	00	WO	Band-gap reference voltage selector for ADC channel F: 00: 1.2V 01: 2V 10: 3V 11: 4V
1 - 0	-	-	Reserved. Please keep 0.

6.40. ADC Result High Register (*adcrh*), IO address = 0x3e

Bit	Reset	R/W	Description
7 - 0	-	RO	These eight read-only bits will be the bit [11:4] of AD conversion result. The bit 7 of this register is the MSB of ADC result for any resolution.

6.41. ADC Result Low Register (*adcrl*), IO address = 0x3f

Bit	Reset	R/W	Description
7 - 4	-	RO	These four bits will be the bit [3:0] of AD conversion result.
3 - 0	-	-	Reserved

6.42. PWMG1 control Register (*pwmg1c)*, IO address = 0x26

Bit	Reset	R/W	Description
7	0	WO	Enable PWMG1. 0 / 1 : disable / enable.
6	-	RO	Output of PWMG1.
5	0	WO	Enable to inverse the polarity of PWMG1 output. 0 / 1 : disable / enable.
4	0	WO	PWMG1 counter reset. Writing “1” to clear PWMG1 counter.
3 - 1	0	WO	Select PWMG1 output pin. 000: none 001: PB6 011: PA4 100: PB7 Others: reserved
0	0	WO	Clock source of PWMG1. 0 : SYSCLK 1 : IHRC or IHRC * 2 (by Code Option : PWM_Source)

6.43. PWMG1 Scalar Register *(pwmg1s)*, IO address = 0x27

Bit	Reset	R/W	Description
7	0	WO	PWMG1 interrupt mode. 0: Generate interrupt when counter matches the duty value. 1: Generate interrupt when counter is 0.
6 - 5	0	WO	PWMG1 clock pre-scalar. 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 - 0	0	WO	PWMG1 clock divider.

6.44. PWMG1 Counter Upper Bound High Register *(pwmg1cubh)*, IO address = 0x2A

Bit	Reset	R/W	Description
7 - 0	8’h00	WO	Bit[10:3] of PWMG1 counter upper bound.

6.45. PWMG1 Counter Upper Bound Low Register *(pwmg1cubl)*, IO address = 0x2B

Bit	Reset	R/W	Description
7 - 5	000	WO	Bit[2:0] of PWMG1 counter upper bound.
4 - 0	-	-	Reserved

6.46. PWMG1 Duty Value High Register (*pwmg1dth*), IO address = 0x28

Bit	Reset	R/W	Description
7 - 0	8’h00	WO	Duty values bit[10:3] of PWMG1.

6.47. PWMG1 Duty Value Low Register (*pwmg1dtl*), IO address = 0x29

Bit	Reset	R/W	Description
7 - 5	000	WO	Duty values bit[2:0] of PWMG1.
4 - 0	-	-	Reserved

Note: It’s necessary to write PWMG1 Duty_Value Low Register before writing PWMG1 Duty_Value High Register.

6.48. PWMG2 control Register (*pwmg2c*), IO address = 0x2C

Bit	Reset	R/W	Description
7	0	WO	Enable PWMG2. 0 / 1 : disable / enable.
6	-	RO	Output of PWMG2.
5	0	WO	Enable to inverse the polarity of PWMG2 output. 0 / 1 : disable / enable.
4	0	WO	PWMG2 counter reset. Writing “1” to clear PWMG2 counter.
3 - 1	0	WO	Select PWMG2 output pin. 000: disable 001: PB3 011: PA3 100: PB2 101: PA5 Others: reserved
0	0	WO	Clock source of PWMG2. 0 : SYSCLK 1 : IHRC or IHRC * 2 (by Code Option : PWM_Source)

6.49. PWMG2 Scalar Register (*pwmg2s*), IO address = 0x2D

Bit	Reset	R/W	Description
7	0	WO	PWMG2 interrupt mode. 0: Generate interrupt when counter matches the duty value. 1: Generate interrupt when counter is 0.
6 - 5	0	WO	PWMG2 clock pre-scalar. 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 - 0	0	WO	PWMG2 clock divider.

6.50. PWMG2 Counter Upper Bound High Register (*pwmg2cubh*), IO address = 0x30

Bit	Reset	R/W	Description
7 - 0	8’h00	WO	Bit[10:3] of PWMG2 counter upper bound.

6.51. PWMG2 Counter Upper Bound Low Register (*pwmg2cubl*), IO address = 0x31

Bit	Reset	R/W	Description
7 - 5	000	WO	Bit[2:0] of PWMG2 counter upper bound.
4 - 0	-	-	Reserved

6.52. PWMG2 Duty Value High Register (*pwmg2dth*), IO address = 0x2E

Bit	Reset	R/W	Description
7 - 0	8’h00	WO	Duty values bit[10:3] of PWMG2.

6.53. PWMG2 Duty Value Low Register (*pwmg2dtl*), IO address = 0x2F

Bit	Reset	R/W	Description
7 - 5	000	WO	Duty values bit[2:0] of PWMG2.
4 - 0	-	-	Reserved

Note: It’s necessary to write PWMG2 Duty_Value Low Register before writing PWMG2 Duty_Value High Register.

7. Instructions

Symbol	Description
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)
IO.n	The bit of register
M.n	Only addressed in 0~0x3F (0~63) is allowed

7.1. Data Transfer Instructions

mov a, I	Move immediate data into ACC.
	Example: *mov* a, 0x0f;
	Result: a ← 0fh;
	Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

mov M, a	Move data from ACC into memory Example: *mov* MEM, a; Result: MEM ← a Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
mov a, M	Move data from memory into ACC Example: *mov* a, MEM ; Result: a ← MEM; Flag Z is set when MEM is zero. Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
mov a, IO	Move data from IO into ACC Example: *mov* a, pa ; Result: a ← pa; Flag Z is set when pa is zero. Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
mov IO, a	Move data from ACC into IO Example: *mov* pb, a; Result: pb ← a Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

ldt16 word	Move 16-bit counting values in Timer16 to memory in word.
	Example: *ldt16* word;
	Result: word ← 16-bit timer
	Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
	
	Application Example:
	-- 	
	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
	….
	-- 	

stt16 word

Store 16-bit data from memory in word to Timer16.

Example: *stt16* word;

Result: 16-bit timer ←word

Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

Application Example:

```
-------------------------------------------------------------------------------------------------------------------
      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
      …
-------------------------------------------------------------------------------------------------------------------
```

idxm a, index

Move data from specified memory to ACC by indirect method. It needs 2T to execute this instruction.

Example: *idxm* a, index;

Result: a ← [index], where index is declared by word.

Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

Application Example:

```
-------------------------------------------------------------------------------------------------------------------
      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 ;          // mov memory data in address 0x5B to ACC
-------------------------------------------------------------------------------------------------------------------
```

Idxm index, a

Move data from ACC to specified memory by indirect method. It needs 2T to execute this instruction.

Example: *idxm* index, a;

Result: [index] ← a; where index is declared by word.

Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

Application Example:

```
-------------------------------------------------------------------------------------------------------------------
      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 ;
      idxm        RAMIndex, a ;          // mov 0xA5 to memory in address 0x5B
-------------------------------------------------------------------------------------------------------------------
```

xch *M*	Exchange data between ACC and memory Example: xch MEM ; Result: MEM ← a , a ← MEM Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

pushaf	Move the ACC and flag register to memory that address specified in the stack pointer.
	Example: pushaf;
	Result: [sp] ← {flag, ACC};
	sp ← sp + 2 ;
	Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
	
	Application Example:
	-- 	
	.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 ;
	-- 	

popaf	Restore ACC and flag from the memory which address is specified in the stack pointer. Example: popaf; Result: sp ← sp - 2 ; {Flag, ACC} ← [sp] ; Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

7.2. Arithmetic Operation Instructions

add a, I	Add immediate data with ACC, then put result into ACC Example: *add* a, 0x0f ; Result: a ← a + 0fh Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
add a, M	Add data in memory with ACC, then put result into ACC Example: *add* a, MEM ; Result: a ← a + MEM Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
add M, a	Add data in memory with ACC, then put result into memory Example: *add* MEM, a; Result: MEM ← a + MEM Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
addc a, M	Add data in memory with ACC and carry bit, then put result into ACC Example: *addc* a, MEM ; Result: a ← a + MEM + C Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
addc M, a	Add data in memory with ACC and carry bit, then put result into memory Example: *addc* MEM, a ; Result: MEM ← a + MEM + C Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

addc a	Add carry with ACC, then put result into ACC Example: addc a ; Result: a ← a + C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
addc M	Add carry with memory, then put result into memory Example: addc MEM ; Result: MEM ← MEM + C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
nadd a, M	Add negative logic (2's complement) of ACC with memory Example: nadd a, MEM ; Result: a ← 〒a + MEM Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
nadd M, a	Add negative logic (2's complement) of memory with ACC Example: nadd MEM, a ; Result: MEM ← 〒MEM + a Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
sub a, I	Subtraction immediate data from ACC, then put result into ACC. Example: sub a, 0x0f; Result: a ← a - 0fh (a + [2's complement of 0fh]) Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
sub a, M	Subtraction data in memory from ACC, then put result into ACC Example: sub a, MEM ; Result: a ← a - MEM (a + [2's complement of M]) Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
sub M, a	Subtraction data in ACC from memory, then put result into memory Example: sub MEM, a; Result: MEM ← MEM - a (MEM + [2's complement of a]) Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
subc a, M	Subtraction data in memory and carry from ACC, then put result into ACC Example: subc a, MEM; Result: a ← a – MEM - C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
subc M, a	Subtraction ACC and carry bit from memory, then put result into memory Example: subc MEM, a ; Result: MEM ← MEM – a - C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
subc a	Subtraction carry from ACC, then put result into ACC Example: subc a; Result: a ← a - C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
subc M	Subtraction carry from the content of memory, then put result into memory Example: subc MEM; Result: MEM ← MEM - C Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV

inc M	Increment the content of memory Example: inc MEM ; Result: MEM ← MEM + 1 Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
dec M	Decrement the content of memory Example: dec MEM; Result: MEM ← MEM - 1 Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
clear M	Clear the content of memory Example: clear MEM ; Result: MEM ← 0 Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV

mul	Multiplication operation, 8x8 unsigned multiplications will be executed.
	Example: mul ;
	Result: {MulRH,ACC} ← ACC * MulOp
	Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
	Application Example :
	-- 	
	…
	mov a, 0x5a ;
	mov mulop, a ;
	mov a, 0xa5 ;
	mul // 0x5A * 0xA5 = 3A02 (mulrh + ACC)
	mov ram0, a ; // LSB, ram0=0x02
	mov a, mulrh ; // MSB, ACC=0X3A
	…
	-- 	

7.3. Shift Operation Instructions

sr a	Shift right of ACC, shift 0 to bit 7 Example: *sr* a ; Result: a (0,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0) Affected flags: 『N』 Z 『Y』 C 『N』 AC 『N』 OV
src a	Shift right of ACC with carry bit 7 to flag Example: *src* a ; Result: a (c,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0) Affected flags: 『N』 Z 『Y』 C 『N』 AC 『N』 OV
sr M	Shift right the content of memory, shift 0 to bit 7 Example: *sr* MEM ; Result: MEM(0,b7,b6,b5,b4,b3,b2,b1) ← MEM(b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0) Affected flags: 『N』 Z 『Y』 C 『N』 AC 『N』 OV
src M	Shift right of memory with carry bit 7 to flag Example: *src* MEM ; Result: MEM(c,b7,b6,b5,b4,b3,b2,b1) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0) Affected flags: 『N』 Z 『Y』 C 『N』 AC 『N』 OV

sl a	Shift left of ACC shift 0 to bit 0 Example: *sl* a ; Result: a (b6,b5,b4,b3,b2,b1,b0,0) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a (b7) Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
slc a	Shift left of ACC with carry bit 0 to flag Example: *slc* a ; Result: a (b6,b5,b4,b3,b2,b1,b0,c) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b7) Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
sl M	Shift left of memory, shift 0 to bit 0 Example: *sl* MEM ; Result: MEM (b6,b5,b4,b3,b2,b1,b0,0) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b7) Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
slc M	Shift left of memory with carry bit 0 to flag Example: *slc* MEM ; Result: MEM (b6,b5,b4,b3,b2,b1,b0,C) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM (b7) Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
swap a	Swap the high nibble and low nibble of ACC Example: *swap* a ; Result: a (b3,b2,b1,b0,b7,b6,b5,b4) ← a (b7,b6,b5,b4,b3,b2,b1,b0) Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

7.4. Logic Operation Instructions

and a, I	Perform logic AND on ACC and immediate data, then put result into ACC Example: *and* a, 0x0f ; Result: a ← a & 0fh Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
and a, M	Perform logic AND on ACC and memory, then put result into ACC Example: *and* a, RAM10 ; Result: a ← a & RAM10 Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
and M, a	Perform logic AND on ACC and memory, then put result into memory Example: *and* MEM, a ; Result: MEM ← a & MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
or a, I	Perform logic OR on ACC and immediate data, then put result into ACC Example: *or* a, 0x0f ; Result: a ← a | 0fh Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
or a, M	Perform logic OR on ACC and memory, then put result into ACC Example: *or* a, MEM ; Result: a ← a | MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
or M, a	Perform logic OR on ACC and memory, then put result into memory Example: *or* MEM, a ; Result: MEM ← a | MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV

xor a, I	Perform logic XOR on ACC and immediate data, then put result into ACC Example: xor a, 0x0f ; Result: a ← a ^ 0fh Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV

xor IO, a	Perform logic XOR on ACC and IO register, then put result into IO register
	Example: xor pa, a ;
	Result: pa ← a ^ pa ; // pa is the data register of port A
	Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

xor a, M	Perform logic XOR on ACC and memory, then put result into ACC Example: xor a, MEM ; Result: a ← a ^ RAM10 Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
xor M, a	Perform logic XOR on ACC and memory, then put result into memory Example: xor MEM, a ; Result: MEM ← a ^ MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV

not a	Perform 1's complement (logical complement) of ACC
	Example: not a ;
	Result: a ← ～a
	Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
	Application Example:
	-- 	
	mov a, 0x38 ; // ACC=0X38
	not a ; // ACC=0XC7
	-- 	

not M	Perform 1's complement (logical complement) of memory
	Example: not MEM ;
	Result: MEM ← ～MEM
	Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
	
	Application Example:
	-- 	
	mov a, 0x38 ;
	mov mem, a ; // mem = 0x38
	not mem ; // mem = 0xC7
	-- 	

neg a	Perform 2's complement of ACC
	Example: neg a;
	Result: a ← 〒a
	Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
	
	Application Example:
	-- 	
	mov a, 0x38 ; // ACC=0X38
	neg a ; // ACC=0XC8
	-- 	

neg M	Perform 2’s complement of memory
	Example: neg MEM;
	Result: MEM ← 〒MEM
	Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
	
	Application Example:
	-- 	
	mov a, 0x38 ;
	mov mem, a ; // mem = 0x38
	not mem ; // mem = 0xC8
	-- 	

comp a, M	Compare ACC with the content of memory
	Example: comp a, MEM;
	Result: Flag will be changed by regarding as (a - MEM)
	Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
	Application Example:
	-- 	
	mov a, 0x38 ;
	mov mem, a ;
	comp a, mem ; // Z flag is set as 1
	mov a, 0x42 ;
	mov mem, a ;
	mov a, 0x38 ;
	comp a, mem ; // C flag is set as 1
	-- 	

comp M, a	Compare ACC with the content of memory Example: comp MEM, a; Result: Flag will be changed by regarding as (MEM - a) Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

7.5. Bit Operation Instructions

set0 IO.n	Set bit n of IO port to low Example: *set0* pa.5 ; Result: set bit 5 of port A to low Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
set1 IO.n	Set bit n of IO port to high Example: *set1* pb.5 ; Result: set bit 5 of port B to high Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

swapc IO.n	Swap the nth bit of IO port with carry bit
	Example: swapc IO.0;
	Result: C ← IO.0 , IO.0 ← C
	When IO.0 is a port to output pin, carry C will be sent to IO.0;
	When IO.0 is a port from input pin, IO.0 will be sent to carry C;
	Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
	Application Example1 (serial output) :
	-- 	
	...
	set1 pac.0 ; // set PA.0 as output
	...
	set0 flag.1 ; // C=0
	swapc pa.0 ; // move C to PA.0 (bit operation), PA.0=0
	set1 flag.1 ; // C=1
	swapc pa.0 ; // move C to PA.0 (bit operation), PA.0=1
	...
	-- 	
	Application Example2 (serial input) :
	-- 	
	...
	set0 pac.0 ; // set PA.0 as input
	...
	swapc pa.0 ; // read PA.0 to C (bit operation)
	src a ; // shift C to bit 7 of ACC
	swapc pa.0 ; // read PA.0 to C (bit operation)
	src a ; // shift new C to bit 7, old C
	...
	-- 	

set0 M.n	Set bit n of memory to low Example: *set0* MEM.5 ; Result: set bit 5 of MEM to low Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
set1 M.n	Set bit n of memory to high Example: *set1* 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

ceqsn a, I	Compare ACC with immediate data and skip next instruction if both are equal. Flag will be changed like as (a ← a – I) Example: *ceqsn* a, 0x55 ; *inc* MEM ; *goto* error ; Result: If a=0x55, then “goto error”; otherwise, “inc MEM”. Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
ceqsn a, M	Compare ACC with memory and skip next instruction if both are equal. Flag will be changed like as (a ← a - M) Example: *ceqsn* a, MEM; Result: If a=MEM, skip next instruction Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
cneqsn a, M	Compare ACC with memory and skip next instruction if both are not equal. Flag will be changed like as (a ← a - M) Example: *cneqsn a, MEM;* Result: If a≠MEM, skip next instruction Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
cneqsn a, I	Compare ACC with immediate data and skip next instruction if both are no equal. Flag will be changed like as (a ← a - I) Example: *cneqsn a,0x55 ;* *inc MEM ;* *goto error ;* Result: If a≠0x55, then “goto error”; Otherwise, “inc MEM”. Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
t0sn IO.n	Check IO bit and skip next instruction if it’s low Example: *t0sn* pa.5; Result: If bit 5 of port A is low, skip next instruction Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
t1sn IO.n	Check IO bit and skip next instruction if it’s high Example: *t1sn* pa.5 ; Result: If bit 5 of port A is high, skip next instruction Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
t0sn M.n	Check memory bit and skip next instruction if it’s low Example: *t0sn* MEM.5 ; Result: If bit 5 of MEM is low, then skip next instruction Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
t1sn M.n	Check memory bit and skip next instruction if it’s high EX: *t1sn* MEM.5 ; Result: If bit 5 of MEM is high, then skip next instruction Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
izsn a	Increment ACC and skip next instruction if ACC is zero Example: *izsn* a; Result: a ← a + 1,skip next instruction if a = 0 Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

dzsn a	Decrement ACC and skip next instruction if ACC is zero Example: dzsn a; Result: A ← A - 1,skip next instruction if a = 0 Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
izsn M	Increment memory and skip next instruction if memory is zero Example: izsn MEM; Result: MEM ← MEM + 1, skip next instruction if MEM= 0 Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV
dzsn M	Decrement memory and skip next instruction if memory is zero Example: dzsn MEM; Result: MEM ← MEM - 1, skip next instruction if MEM = 0 Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV

7.7. System control Instructions

call label	Function call, address can be full range address space
	Example: *call* function1;
	Result: [sp] ← pc + 1
	pc ← function1
	sp ← sp + 2
	Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV

goto label	Go to specific address which can be full range address space Example: *goto* error; Result: Go to error and execute program. Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
ret I	Place immediate data to ACC, then return Example: *ret* 0x55; Result: A ← 55h ret ; Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
ret	Return to program which had function call Example: *ret;* Result: sp ← sp - 2 pc ← [sp] Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
reti	Return to program that is interrupt service routine. After this command is executed, global interrupt is enabled automatically. Example: *reti*; Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
nop	No operation Example: *nop*; Result: nothing changed Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV

pcadd a	Next program counter is current program counter plus ACC.
	Example: pcadd a;
	Result: pc ← pc + a
	Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
	Application Example:
	-- 	
	…
	mov a, 0x02 ;
	pcadd a ; // PC <- PC+2
	goto err1 ;
	goto correct ; // jump here
	goto err2 ;
	goto err3 ;
	…
	correct: // jump here
	…
	-- 	

engint	Enable global interrupt enable Example: engint; Result: Interrupt request can be sent to CPU Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
disgint	Disable global interrupt enable Example: disgint ; Result: Interrupt request is blocked from CPU Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
stopsys	System halt. Example: stopsys; Result: Stop the system clocks and halt the system Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
stopexe	CPU halt. The oscillator module is still active to output clock, however, system clock is disabled to save power. Example: stopexe; Result: Stop the system clocks and keep oscillator modules active. Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
reset	Reset the whole chip, its operation will be same as hardware reset. Example: reset; Result: Reset the whole chip. Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
wdreset	Reset Watchdog timer. Example: wdreset ; Result: Reset Watchdog timer. Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV

7.8. Summary of Instructions Execution Cycle

2T	*goto, call, idxm*
1T/2T	*ceqsn, t0sn, t1sn, dzsn, izsn*
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
mov a, I	-	-	-	-	*mov* M, a	-	-	-	-	*mov* a, M	Y	-	-	-
mov a, IO	Y	-	-	-	*mov* IO, a	-	-	-	-	*ldt16* word	-	-	-	-
stt16 word	-	-	-	-	*idxm* a, index	-	-	-	-	*idxm* index, a	-	-	-	-
xch M	-	-	-	-	*pushaf*	-	-	-	-	*popaf*	Y	Y	Y	Y
add a, I	Y	Y	Y	Y	*add* a, M	Y	Y	Y	Y	*add* M, a	Y	Y	Y	Y
addc a, M	Y	Y	Y	Y	*addc* M, a	Y	Y	Y	Y	*addc* a	Y	Y	Y	Y
addc M	Y	Y	Y	Y	*nadd a, M*	Y	Y	Y	Y	*nadd M, a*	Y	Y	Y	Y
sub a, I	Y	Y	Y	Y	*sub* a, M	Y	Y	Y	Y	*sub* M, a	Y	Y	Y	Y
subc a, M	Y	Y	Y	Y	*subc* M, a	Y	Y	Y	Y	*subc* a	Y	Y	Y	Y
subc M	Y	Y	Y	Y	*inc* M	Y	Y	Y	Y	*dec* M	Y	Y	Y	Y
clear M	-	-	-	-	*mul*	-	-	-	-	*sr* a	-	Y	-	-
src a	-	Y	-	-	*sr* M	-	Y	-	-	*src* M	-	Y	-	-
sl a	-	Y	-	-	*slc* a	-	Y	-	-	*sl* M	-	Y	-	-
slc M	-	Y	-	-	*swap* a	-	-	-	-	*and* a, I	Y	-	-	-
and a, M	Y	-	-	-	*and* M, a	Y	-	-	-	*or* a, I	Y	-	-	-
or a, M	Y	-	-	-	*or* M, a	Y	-	-	-	*xor* a, I	Y	-	-	-
xor IO, a	-	-	-	-	*xor* a, M	Y	-	-	-	*xor* M, a	Y	-	-	-
not a	Y	-	-	-	*not* M	Y	-	-	-	*neg* a	Y	-	-	-
neg M	Y	-	-	-	*comp a, M*	Y	Y	Y	Y	*comp M, a*	Y	Y	Y	Y
set0 IO.n	-	-	-	-	*set1* IO.n	-	-	-	-	*set0* M.n	-	-	-	-
set1 M.n	-	-	-	-	*swapc IO.n*	-	Y	-	-	*ceqsn* a, I	Y	Y	Y	Y
ceqsn a, M	Y	Y	Y	Y	*cneqsn a,M*	Y	Y	Y	Y	*cneqsn a, I*	Y	Y	Y	Y
t0sn IO.n	-	-	-	-	*t1sn* IO.n	-	-	-	-	*t0sn* M.n	-	-	-	-
t1sn M.n	-	-	-	-	*izsn* a	Y	Y	Y	Y	*dzsn* a	Y	Y	Y	Y
izsn M	Y	Y	Y	Y	*dzsn* M	Y	Y	Y	Y	*call* label	-	-	-	-
goto label	-	-	-	-	*ret* I	-	-	-	-	*ret*	-	-	-	-
reti	-	-	-	-	*nop*	-	-	-	-	*pcadd* a	-	-	-	-
engint	-	-	-	-	*disgint*	-	-	-	-	*stopsys*	-	-	-	-
stopexe	-	-	-	-	*reset*	-	-	-	-	*wdreset*	-	-	-	-

8. Code Options

Option	Selection	Description
Security	Enable	Security 7/8 words Enable
	Disable	Security Disable
LVR	4.0V	Select LVR = 4.0V
	3.5V	Select LVR = 3.5V
	3.0V	Select LVR = 3.0V
	2.75V	Select LVR = 2.75V
	2.5V	Select LVR = 2.5V
	2.2V	Select LVR = 2.2V
	2.0V	Select LVR = 2.0V
	1.8V	Select LVR = 1.8V
Boot-up_Time	Slow	Please refer to t_{WUP} and t_{SBP} in Section 4.1
	Fast	Please refer to t_{WUP} and t_{SBP} in Section 4.1
PWM_Source	16MHZ	When pwmg0c.0= 1, PWMG0 clock source = IHRC = 16MHZ When pwmg1c.0= 1, PWMG1 clock source = IHRC = 16MHZ When pwmg2c.0= 1, PWMG2 clock source = IHRC = 16MHZ
	32MHZ	When pwmg0c.0= 1, PWMG0 clock source = IHRC*2 = 32MHZ When pwmg1c.0= 1, PWMG1 clock source = IHRC*2 = 32MHZ When pwmg2c.0= 1, PWMG2 clock source = IHRC*2 = 32MHZ (ICE does NOT Support.)
GPC_PWM	Disable	GPC/ PWM are independent
	Enable	GPC output control PWM output (ICE does NOT Support.)
Interrupt Src0	PA.0	INTEN/ INTRQ.Bit0 is from PA.0
	PA.5	INTEN/ INTRQ.Bit0 is from PB.5
Interrupt Src1	PB.0	INTEN/ INTRQ.Bit1 is from PB.0
	PA.4	INTEN/ INTRQ.Bit1 is from PA.4
PB4_PB7_Drive	Normal	PB4 & PB7 Drive/ Sink Current is Normal
	Strong	PB4 & PB7 Drive/ Sink Current is Strong
Comparator_Edge	All_Edge	The comparator will trigger an interrupt on the rising edge or falling edge
	Rising_Edge	The comparator will trigger an interrupt on the rising edge
	Falling_Edge	The comparator will trigger an interrupt on the falling edge

9. Special Notes

This chapter is to remind user who use PMS132 series IC in order to avoid frequent errors upon operation.

9.1. Warning

User must read all application notes of the IC by detail before using it. Please download the related application notes from the following link:

http://www.padauk.com.tw/technical-application.php

9.2. Using IC

9.2.1 IO pin usage and setting

(1) IO pin as digital input

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

(2) IO pin as digital input and enable wakeup function

- ◆ Configure IO pin as input
- ◆ Set PADIER and PBDIER registers to set the corresponding bit to 1.

(3) PA5 is set to be output pin

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

(4) PA5 is set to be PRSTB input pin

- ◆ Configure PA5 as input
- ◆ Set CLKMD.0=1 to enable PA5 as PRSTB 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 >33Ω resistor in between PA5 and the long wire
- ◆ Avoid using PA5 as input in such application.

(6) PA7 and PA6 as external crystal oscillator

- ◆ Configure PA7 and PA6 as input
- ◆ Disable PA7 and PA6 internal pull-up resistor
- ◆ Configure PADIER register to set PA6 and PA7 as analog input
- ◆ EOSCR register bit [6:5] selects corresponding crystal oscillator frequency :
  - ✧ 01 : for lower frequency, ex : 32KHz (reserved)
  - ✧ 10 : for middle frequency, ex : 455KHz, 1MHz
  - ✧ 11 : for higher frequency, ex : 4MHz
- ◆ Program EOSCR.7 =1 to enable crystal oscillator
- ◆ Ensure EOSC working well before switching from IHRC or ILRC to EOSC, refer to 9.2.3.(2)

9.2.2 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

(3) PA4 and PB5 can be used as external interrupt pins. When using the PA4 as external interrupt pin, the setting method of **inten/intrq/integs** registers are same as that of PB0, the only difference is to choose PB0 or PA4 as source of interrupt_Src1 in PADAUK_CODE_OPTION. Similarly, when using the PB5 as external interrupt pin, the setting method of **inten/intrq/integs** registers are same as that of PA0, the only difference is to choose PA0 or PB5 as source of interrupt_Src0 in PADAUK_CODE_OPTION.

9.2.3 System clock switching

(1) 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.

- Case 1 : Switch system clock from ILRC to IHRC/2

  CLKMD = 0x36; // switch to IHRC, *ILRC can not be disabled here*

  CLKMD.2 = 0; // ILRC can be disabled at this time

- Case 2 : Switch system clock from ILRC to EOSC

  CLKMD = 0xA6; // switch to EOSC, *ILRC can not be disabled here*

  CLKMD.2 = 0; // ILRC can be disabled at this time

- **ERROR.** Switch ILRC to IHRC and turn off ILRC simultaneously

  CLKMD = 0x50; // MCU will hang

(2) Please ensure the EOSC oscillation has established before switching from ILRC or IHRC to EOSC. MCU will not check its status. Please wait for a while after enabling EOSC. System clock can be switched to EOSC afterwards. Otherwise, MCU will hang. The example for switching system clock from ILRC to 4MHz EOSC after boot up is as below:

```
.ADJUST_IC     SYSCLK=ILRC;
$   EOSCR       Enable, 4MHz;              // 4MHz EOSC start to oscillate.
                                           // delay time to wait crystal oscillator stable
$ T16M EOSC, /1, BIT10
Word Count = 0;
Stt16 Count;
Intrq.T16 = 0;
do
{ nop; }while(!Intrq.T16);
CLKMD    = 0xA4;                           // ILRC -> EOSC;
CLKMD.2 = 0;                               // turn off ILRC only if necessary
```

The delay duration should be adjusted in accordance with the characteristic of the crystal and PCB. To measure the oscillator signal by the oscilloscope, please select (x10) on the probe and measure through PA6(X2) pin to avoid the interference on the oscillator.

9.2.4 Power down mode, wakeup and watchdog

Watchdog will be inactive once ILRC is disabled.

9.2.5 TIMER time out

When select T16M counter BIT8 as 1 to generate interrupt, 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.

9.2.6 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). And PADAUK 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.

9.2.7 LVR

(1) V_{DD} must reach or above 2.0V for successful power-on process; otherwise IC will be inactive.

(2) The setting of LVR (1.8V, 2.0V, 2.2V etc.) will be valid just after successful power-on process.

(3) User can set MISC.2 as "1" to disable LVR. However, V_{DD} must be kept as exceeding the lowest working voltage of chip; Otherwise IC may work abnormally.

9.2.8 The result of Comparator controls the PWM output pins

The special function of GPC_PWM in PADAUK_CODE_OPTION is used to control the output pins of PWM modules including TM2, TM3 and PWMG0 / PWMG1 / PWMG2 according to the status of gpcc.6. Any output pins of those PWM modules will go to 0 when gpcc.6 is 1 and go back to normal PWM function when gpcc.6 is 0.

9.2.9 Instructions

(1) PMS132 supports 87 instructions.

(2) The instruction execution cycle of PMS132 is shown as below:

Instruction	Condition	CPU
goto, call, pcadd, ret, reti		2T
ceqsn, cneqsn, t0sn, t1sn, dzsn, izsn	Condition is fulfilled	2T
	Condition is not fulfilled	1T
idxm		2T
Others		1T

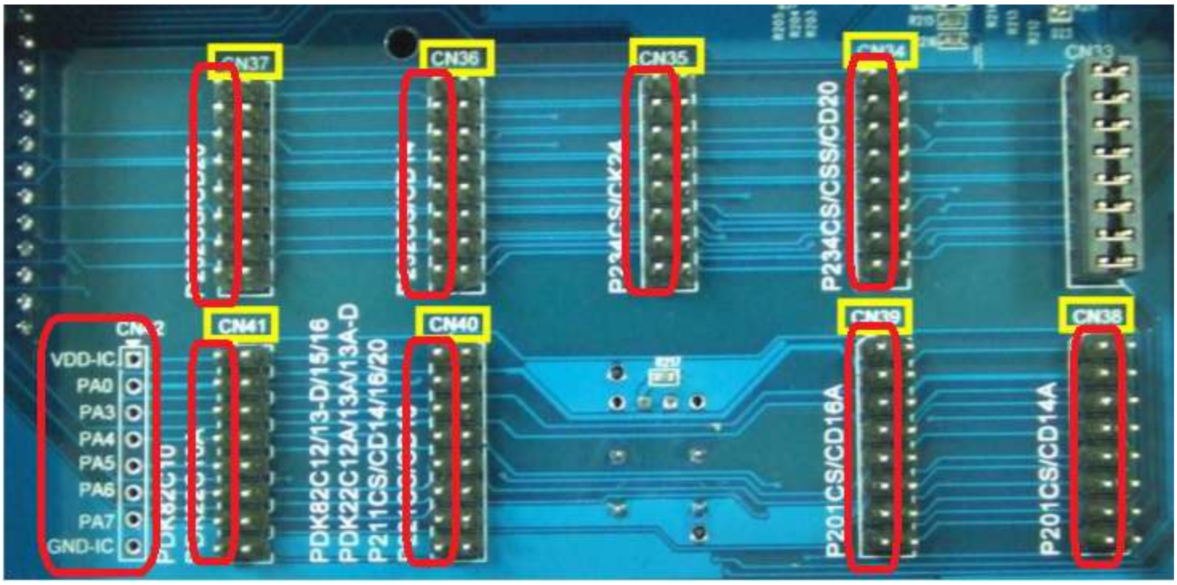
9.2.10 BIT definition

Bit access of RAM is only available for address from 0x00 to 0x3F.

9.2.11 Programming the PMS132

There are 6 pins for using the writer to program: PA3, PA4, PA5, PA6, V_{DD}, and GND.

Please use PDK3S-P-002 to program and put the PMS132-S16A/ S16B /S14 to move down three spaces over the CN38. Other packages could be programmed by user’s way. All the left signs behind the jumper are the same (there are V_{DD}, PA0(not required), PA3, PA4, PA5, PA6, PA7(not required), and GND).The following picture is shown:



If user use PDK5S-P-002 or above to program, please follow the instruction.

9.3 Using ICE

(1) It is recommended to use PDK5S-I-S01 for emulation of PMS132. PDK5S-I-S01 supports PMS132 1-FPPA MCU emulation work, the following items should be noted when using PDK5S-I-S01 to emulate PMS132:

- PDK5S-I-S01 doesn’t support the instruction NADD/COMP of PMS132.
- PDK5S-I-S01 doesn’t support SYSCLK=ILRC/16 of PMS132.
- PDK5S-I-S01 doesn’t support the function TM2.GPCRS/TM3.GPCRS of PMS132.
- PDK5S-I-S01 doesn’t support the function PWMG2C.PA5.
- PDK5S-I-S01 doesn’t support the function ADCRGC.BG_2V/BG_3V_BG_4V, and fix BG_1V2 only.
- PDK5S-I-S01 doesn’t support the code options: GPC_PWM, PWM_Source, and TMx_bit.
- Fast Wakeup time is different from PDK5S-I-S01: 128 SysClk, PMS132: 45 ILRC
- Watch dog time out period is different from PDK5S-I-S01:

WDT period	PMS132	PDK5S-I-S01
misc[1:0]=00	8K* T_{ILRC}	2048* T_{ILRC}
misc[1:0]=01	16K* T_{ILRC}	4096* T_{ILRC}
misc[1:0]=10	64K* T_{ILRC}	16384* T_{ILRC}
misc[1:0]=11	256K* T_{ILRC}	256* T_{ILRC}