MF324

Industrial Grade –
8 bits MTP BLDC Controller

Datasheet

Version 0.01

June 2, 2026

6F-6, No.1, Sec. 3, Gongdao 5th Rd., Hsinchu City 30069, Taiwan, R.O.C.
TEL: 886-3-572-8688 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 that may involve potential risks of death, personal injury, fire or severe property damage.

Any programming software provided by PADAUK Technology to customers is of a service and reference nature and does not have any responsibility for software vulnerabilities. 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.

Table of Contents

Revision History **7**
Usage Warning **7**
1. Features **8**
1.1. Special Features 8
1.2. System Features 8
1.3. High Performance RISC CPU Array 9
1.4. Ordering/ Package Information 9
2. General Description and Block Diagram **10**
3. Pin Definition and Functional Description **11**
4. Device Characteristics **12**
4.1. Absolute Maximum Ratings 12
4.2. DC/AC Characteristics 12
4.3. Typical ILRC frequency vs. VDD and temperature 14
4.4. Typical IHRC frequency deviation vs. VDD and temperature 14
4.5. Typical ILRC frequency vs. VDD 15
4.6. Typical IHRC frequency deviation vs. VDD 15
4.7. Typical Operating Current vs. VDD and CLK=IHRC/n 16
4.8. Typical Operating Current vs. VDD and CLK=ILRC/n 16
4.9. Typical IO pull high / low resistance 17
4.10. Typical IO driving current (IOH) and sink current (IOL) 18
4.11. Typical IO input high/low threshold voltage (VIH/VIL) 19
5. Central Processing Unit (CPU) **19**
5.1. Functional Description 19
5.1.1. Processing Units 20
5.1.2. Program Counter 22
5.1.3. Program Structure 23
5.1.4. Arithmetic and Logic Unit 23
5.2. Storage Memory 24
5.2.1. Program Memory – ROM 24
5.2.2. Data Memory – SRAM 26
5.2.3. System Register 27
5.2.3.1. ACC Status Flag Register (FLAG), address = 0x00 27
5.2.3.2. Multi-Core Enable Register (MCUEN), address = 0x01 28
5.2.3.3. Option 2 Register (OPR2), address = 0x47 28
5.2.3.4. MISC Register (MISC), address = 0x34 28
5.3. The Stack 29
5.3.1. Stack Pointer Register (*SP*), address = 0x02 30
5.4. Code Options 30
6. Oscillator and System Clock **30**

6.1. Internal High RC Oscillator and Internal Low RC Oscillator....30
6.2. System Clock and IHRC Calibration....31
6.2.1. System Clock....31
6.2.1.1. Clock Mode Register (CLKMD), address = 0x03....31
6.2.2. Frequency Calibration....32
6.2.2.1. Special Statement....33
6.2.3. System Clock Switching....33
7. Reset....34
7.1. Power On Reset - POR....34
7.2. Low Voltage Reset - LVR....35
7.3. Watch Dog Timeout Reset....36
7.4. External Reset Pin - PRSTB....37
8. Interrupt....38
8.1. Interrupt Enable Register (*INTEN*), address = 0x04....39
8.2. Interrupt Enable 2 Register (INTEN2), address = 0x08....39
8.3. Interrupt Request Register (INTRQ), address = 0x05....40
8.4. Interrupt Request 2 Register (INTRQ2), address = 0x09....40
8.5. Interrupt Edge Select Register (INTEGS), address = 0x0C....41
8.6. Interrupt Work Flow....41
8.7. General Steps to Interrupt....42
8.8. Example for Using Interrupt....43
9. I/O Port....44
9.1. IO Related Registers....44
9.1.1. Port A Digital Input Enable Register (*PADIER*), address = 0x0D....44
9.1.2. Port A Data Registers (PA), address = 0x0F....44
9.1.3. Port A Control Registers (*PAC*), address = 0x10....44
9.1.4. Port A Pull-High Registers (*PAPH*), address = 0x11....44
9.1.5. Port A Pull-Low Registers (*PAPL*), address = 0x12....44
9.1.6. Port B Data Registers (PB), address = 0x13....44
9.1.7. Port B Control Registers (*PBC*), address = 0x14....44
9.2. IO Structure and Functions....45
9.2.1. IO Pin Structure....45
9.2.2. IO Pin Functions....46
9.2.3. IO Pin Usage and Setting....46
10. Timer / PWM Counter....47
10.1. 16-bit Timer (Timer16)....47
10.1.1. Timer16 Introduction....47
10.1.2. Timer16 Time Out....48
10.1.3. Timer16 Mode Register (*T16M*), address = 0x06....49
10.2. 16-bit Timer (Timer4)....50
10.2.1. Timer4 Introduction....50
10.2.2. Timer4 mode Register (TM4C), address = 0x22....51
10.2.3. Timer4 Counter High Byte Register (TM4CTH), address = 0x23....51
10.2.4. Timer4 Counter Low Byte Register (TM4CTL), address = 0x24....51

10.2.5. Timer4 Bound High Byte Register (TM4BH), address = 0x25 51
10.2.6. Timer4 Bound Low Byte Register (TM4BL), address = 0x26 51
10.3. 8-bit Timer (Timer2, Timer3) 52
10.3.1. Timer2 Control Register (*TM2C*), address = 0x27 53
10.3.2. Timer2 Counter Register (TM2CT), address = 0x28 53
10.3.3. Timer2 Scalar Register (TM2S), address = 0x29 53
10.3.4. Timer2 Bound Register (TM2B), address = 0x2A 53
10.3.5. Timer3 Control Register (TM3C), address = 0x2B 53
10.3.6. Timer3 Counter Register (TM3CT), address = 0x2C 54
10.3.7. Timer3 Scalar Register (TM3S), address = 0x2D 54
10.3.8. Timer3 Bound Register (TM3B), address = 0x2E 54
10.4. 12-bit PWM Generation 54
10.4.1. Hardware PWM with dead zone 54
10.4.2. Timing Diagram 55
10.4.3. Equations for 12-bit PWM Generator 55
10.4.4. 12-bit PWM Related Registers 56
10.4.4.1. PWM PB control 1 Register (PWMPBC1), address = 0x18 56
10.4.4.2. PWM PB control 2 Register (PWMPBC2), address = 0x19 56
10.4.4.3. PWM Generator control 1 Register (PWMGC1), address = 0x1A 57
10.4.4.4. PWM Generator control 2 Register (PWMGC2), address = 0x1B 57
10.4.4.5. PWM Generator Control Register (PWMGC), address = 0x1C 58
10.4.4.6. Miscellaneous Setting Register (MISC), address = 0x34 58
10.4.4.7. PWM Counter Upper Bound High Register (PWMUPBH), address = 0x3D 58
10.4.4.8. PWM Counter Upper Bound Low Register (PWMUPBL), address = 0x3E 58
10.4.4.9. PWMG0 Duty Value High Register (PWM0DTH), address = 0x3F 59
10.4.4.10. PWMG0 Duty Value Low Register (PWM0DTL), address = 0x40 59
10.4.4.11. PWMG ADC trigger High Register (PWMADCH), address = 0x41 59
10.4.4.12. PWMG ADC trigger Low Register (PWMADCL), address = 0x42 59
10.4.4.13. PWM Front Dead-zone Register (PWMDDZVF), address = 0x43 59
10.4.4.14. PWM Rear Dead-zone Register (PWMDDZVR), address = 0x44 59
11. Special Function 59
11.1. General Purpose Comparator 59
11.1.1. General Purpose Comparator Hardware Diagram 59
11.1.2. General Purpose Comparator Control Register (*GPCC*), address = 0x15 60
11.1.3. General Purpose Comparator Result Register (*GPCR*), address = 0x16 61
11.1.4. Analog Inputs 61
11.1.5. Internal Reference Voltage ($V_{internal\ R}$) 62
11.1.6. Using the Comparator 64
11.2. Analog-to-Digital Conversion (ADC) module 65
11.2.1. The input requirement for ADC conversion 66
11.2.2. ADC clock selection 66
11.2.3. AD conversion 67
11.2.4. Configure the analog pins 67
11.2.5. Using the ADC 68
11.2.6. ADC Related Registers 69
11.2.6.1. ADC Result High Register (ADCRH), address = 0x1E 69

11.2.6.2. ADC Result Low Register (ADCRL), address = 0x1F 69
11.2.6.3. ADC Control Register (ADCC), address = 0x20 69
11.2.6.4. ADC Mode Register (ADCM), address = 0x21 70
11.3. PWM generator Trigger ADC Conversion 70
11.3.1. PWM Trigger ADC - PWM Counter High Register (*PWMADCH*), address = 0x41 71
11.3.2. PWM Trigger ADC - PWM Counter Low Register (*PWMADCL*), address = 0x42 71
11.4. Input Pulse Capture 71
11.4.1. Pulse Capture Control Register (*PLSCC*), address = 0x32 72
11.4.2. Pulse Capture Scalar Register (*PLSCS*), address = 0x33 72
11.4.3. Pulse Capture Pulse Width High Register (*PLSPWH*), address = 0x39 72
11.4.4. Pulse Capture Pulse Width Low Register (*PLSPWL*), address = 0x3A 73
11.4.5. Pulse Capture Pulse High High Register (*PLSPHH*), address = 0x3B 73
11.4.6. Pulse Capture Pulse High Low Register (*PLSPHL*), address = 0x3C 73
11.5. Special Comparator 73
11.5.1. Limit current comparator 73
11.5.1.1. Limit Current Setting Register (*LCS*), address = 0x2F 73
11.5.2. Zero crossing point comparator 74
11.5.2.1. Zero Crossing Point Control Register (*ZCPC*), address = 0x30 74
11.5.2.2. Zero Crossing Point Setting Register (ZCPS), address = 0x31 75
11.6. Multiplier 75
11.6.1. 8×8 multiplier 75
11.6.2. Arithmetic Operation Register (*EARITH*), address = 0x1D 76
11.6.3. 8X8 Multiplier Operand 1 High Byte Register (M8OP1), address = 0x35 76
11.6.4. 8X8 Multiplier Operand 2 Register (M8OP2), address = 0x36 76
11.6.5. 8X8 Multiplier Result1 Register (M8RS1), address = 0x37 76
11.6.6. 8X8 Multiplier Result0 Register (M8RS0), address = 0x38 76
12. Program Writing 76
13. Instructions 80
13.1. Instruction Table 81

Revision History

Revision	Date	Description
0.00	2026/01/16	Preliminary version
0.01	2026/06/02	1. Chapter11.2.2: ADC clock: Change ADC clock period from 0.5us to 0.2us. 2. Chapter11.2.2: Update Fig. 27: ADC Block Diagram 3. Chapter11.5.1: Modify PA4 & PADIER.4 to PA7 & PADIER.7.

Usage 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:

https://www.padauk.com.tw/en/product/search_list.aspx?kw=MF

1. Features

1.1. Special Features

- High EFT series:

  Especially fit for the products that are AC powered with even using RC step-down circuit, or require strong noise immunity, or required high EFT capability (±4KV) for passing safety regulation tests.
- Operating temperature range: -40°C ~ 105°C

1.2. System Features

- 3KW MTP program memory for 8 FPPA units (programming cycle at least 1,000 times)
- 192 Bytes data RAM for all FPPA units
- Two hardware 16-bit timers
- Two hardware 8-bit timers
- One 12-bit hardware PWM generator with programmable period and duty
- Support PWM generator trigger ADC conversion
- Up to 11-channel 12-bit resolution ADC:
  - 1-channel for internal Bandgap reference voltage
  - 1-channel for internal VDD/4 voltage
  - 1-channel for internal BEMF
- One BEMF detector with de-glitch
- Built-in BEMF COM for PA0, PA1, PA2
- One general purpose comparator
- One Hardware Pulse Capture
- One 8X8 hardware multiplier
- Support 3-Phase brushless DC motor
- 7 IO pins with I_{OH} 10mA and I_{OL} 10mA capability and optional pull-high/pull-low resistor
- 1 IO pin with I_{OL} 10mA and optional pull-high resistor
- 6 specifics PWM pins with I_{OH} 40mA and I_{OL} 20mA capability
- 16 levels of VDD voltage reset
- Selectable three external interrupt pins: PA3, PA4, PA6
- Operating voltage range: 2.2V ~ 5V

1.3. High Performance RISC CPU Array

- Patented Field Programmable Processor Array (FPPA™) Technology
- Operating modes: 8 processing units FPPA™ mode
- 87 powerful instructions
- Most instructions are 1T execution cycle
- Programmable stack pointer (address: 0~128) and adjustable stack level
- Direct and indirect addressing modes for data access. Data memories are available for use as an index pointer of Indirect addressing mode
- Support security function to protect MTP data
- Register space, memory space and MTP space are independent

1.4. Ordering/ Package Information

- MF324-1J16A: QFN(3*3*0.75mm)
- MF324-S16A: SOP16(150mil)

- Please refer to the official website file for package size information：”Package information “

2. General Description and Block Diagram

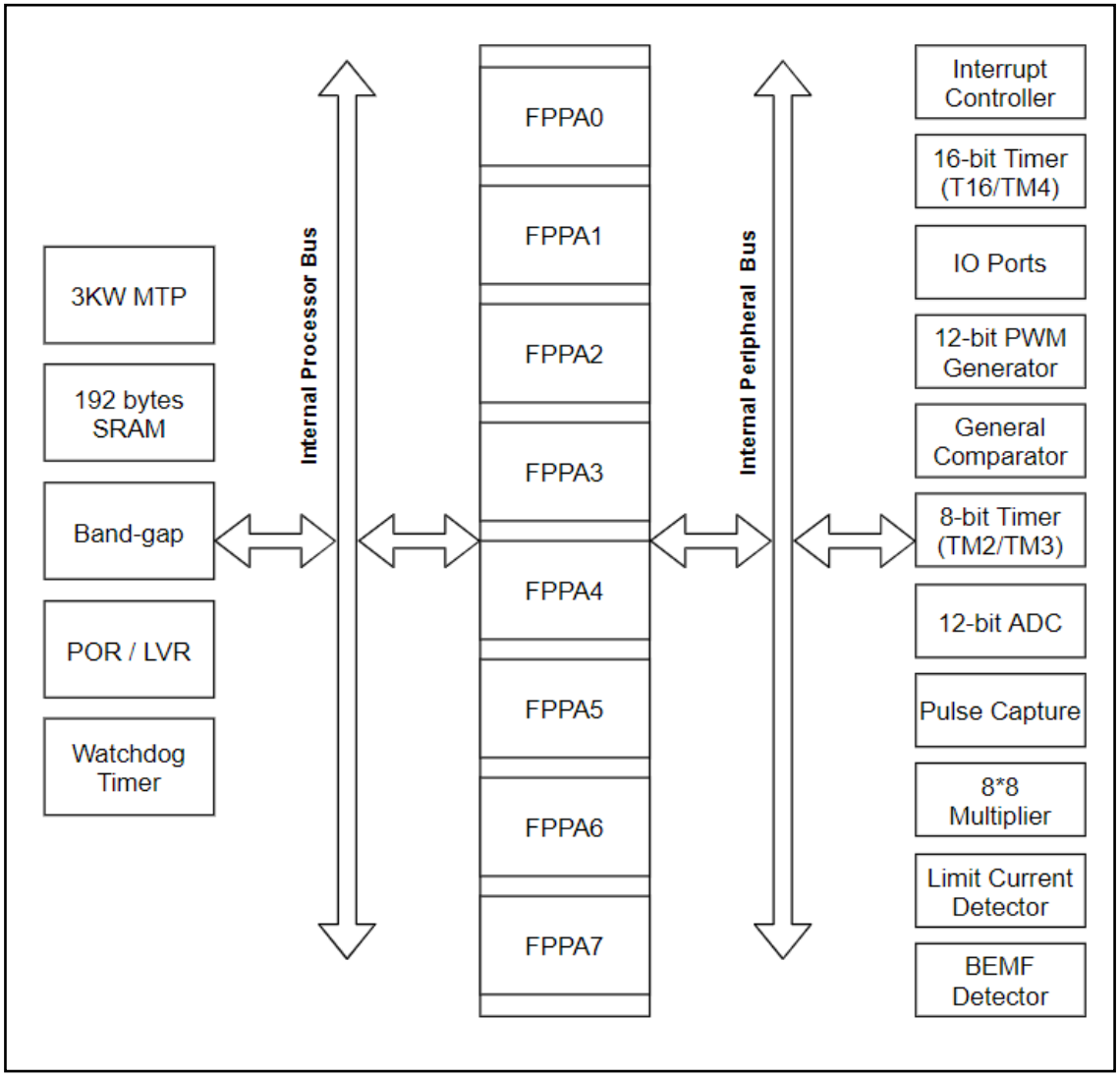
The MF324 is a 3-Phase BLDC of PADAUK's parallel processing, fully static, MTP-based CMOS 8×8 bit processor array that can execute eight peripheral functions in parallel. It employs RISC architecture based on patent pending FPPA™ (Field Programmable Processor Array) technology and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access.

3KW MTP program memory and 192 bytes data SRAM are inside for 8 FPPA units using.

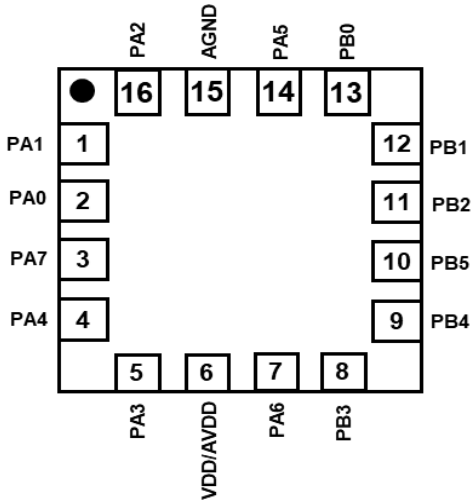
One up to 11 channels 12-bit ADC is a precise comparator.

MF324 provide one general purpose comparators and four hardware timers: Two are 16-bit timers and two are 8-bit timers.

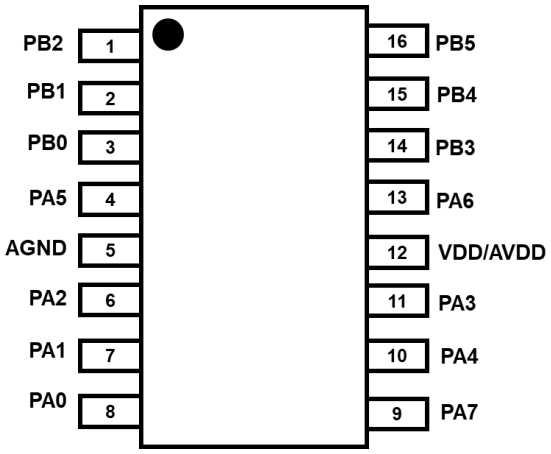
One hardware Pulse Capture, one BEMF detectors, one 12-bit hardware PWM generator and PWM protection modules are also built inside the MF324 in order to provide the best solution for BLDC controller.



3. Pin Definition and Functional Description



MF324-1J16A: QFN(3*3*0.75mm)



MF324-S16A: SOP16(150mil)

Pin description:

Pin Name	Input / output			Special Functions						
	I / O	Pull High / Low	Wake Up	GPC BEMF/LC	GPC	PWM	Pulse Capture	ADC	External INT / RST	Program
PA0	I / O	H / L		BEMF-	CMP-			AD0		
PA1	I / O	H / L		BEMF-	CMP-			AD1		
PA2	I / O	H / L		BEMF-	CMP-			AD2		
PA3	I / O	H / L	V		CMP-		PCIN	AD3	INT0	Program
PA4	I / O	H / L	V	LC Flag	CMP-		PCIN	AD4	INT1	
PA5	I	H	V	BEMF-	CMP+			AD5	RST	Program
PA6	I / O	H / L	V	BEMF Flag	CMP Flag			AD6	INT2	Program
PA7	I / O	H / L	V		CMP-		PCIN	AD7		
PB0	O					V				
PB1	O					V				
PB2	O					V				
PB3	O					V				
PB4	O					V				
PB5	O					V				
VDD / AVDD										Program
GND / AGND										Program
Notice	1. All the I/O pins have: Schmitt Trigger input and CMOS voltage level. 2. IO function is automatically deactivated when a pin is used as PWM output port. 3. Please put 33Ω resistor in series to have high noise immunity when PA5 is in input mode.									

4. Device Characteristics

4.1. Absolute Maximum Ratings

Name	Min	Typ.	Max	Unit	Notes
Supply Voltage (VDD)	2.2	5.0	5.5	V	**Exceed the maximum rating may cause permanent damage!!**
Operating Temperature	-40		105	°C	
Storage Temperature	-50		125	°C	
Junction Temperature		150		°C	

4.2. DC/AC Characteristics

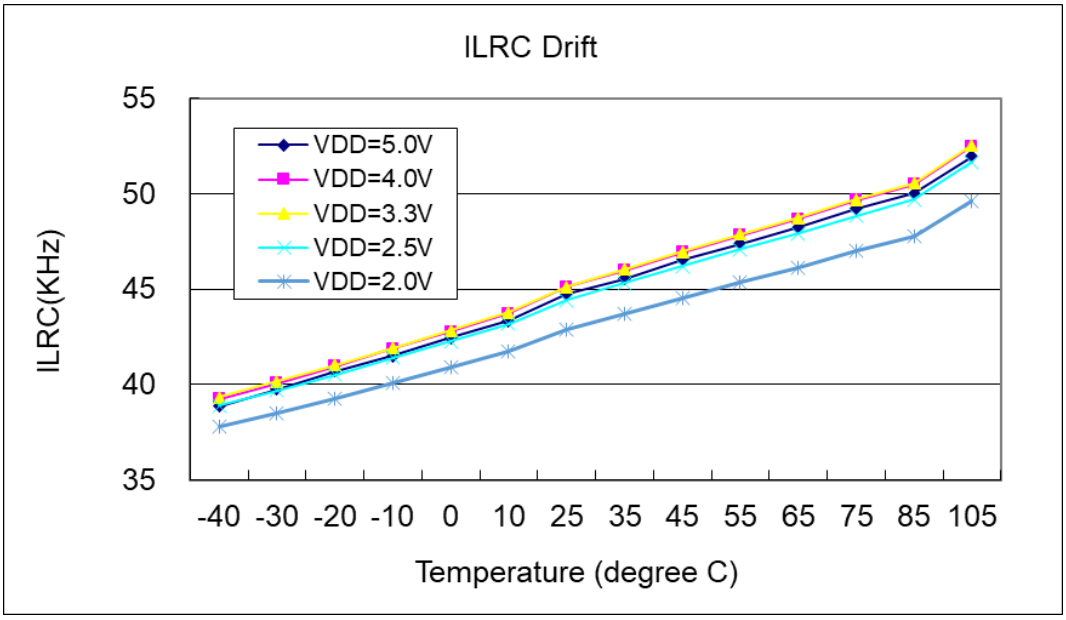
Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25℃)
LVR%	Low Voltage Reset Tolerance	-10		10	%	
f_{SYS}	System clock (CLK)* = IHRC/2		8M		Hz	V_{DD} ≧ 2.75V@105 °C V_{DD} ≧ 2.5V@35 °C
	IHRC/4 IHRC/8 Internal low RC oscillator		4M 2M 45K			V_{DD} ≧ 2.0V V_{DD} ≧ 1.8V V_{DD} = 5.0V
P_{cycle}	Program cycle	1000			cycles	
V_{POR}	Power On Reset Voltage		2.1*		V	* Subject to LVR tolerance
I_{OP}	Operating Current		2 1.5 0.4		mA	f_{SYS}=8MIPS@5.0V f_{SYS}=4MIPS@5.0V f_{SYS}=ILRC= 45KHz@5.0V
I_{PD}	Power Down Current (by ***stopsys*** command)		0.6		uA	V_{DD}=5.0V; Only ILRC module is ON.
I_{PS}	Power Save Current (by ***stopexe*** command)		8		uA	V_{DD}=5.0V; Bandgap, LVR, IHRC, ILRC, Timer16 modules are ON.
V_{IL}	Input low voltage for IO lines	0		0.2VDD	V	
V_{IH}	Input high voltage for IO lines	0.8VDD 0.7VDD		VDD	V	PA5 Other IO
I_{OL}	**IO lines Sink current**					
	PB0 – PB5 PA5 Other pins		47 X 16		mA	V_{DD}=5.0V, V_{OL}=0.5V
I_{OH}	**IO lines Drive current**					
	PB0 – PB5 PA5 Other pins		18 X 12		mA	V_{DD}=5.0V, V_{OH}=4.5V
R_{PH}	Pull-high Resistance		120 71		KΩ	PA5 V_{DD}=5.0V, Other IO
R_{PL}	Pull-low Resistance		X 71		KΩ	PA5 V_{DD}=5.0V, Other IO
V_{BG}	Bandgap Reference Voltage	1.12	1.20	1.28	V	V_{DD}=5V -40°C <Ta<105°C*

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25℃)
f_{IHRC}	Frequency of IHRC after calibration *	15.52*	16*	16.48*	MHz	25°C, V_{DD}=3.15V~5.5V
		14*	16*	17.28*		V_{DD}=3.15V~5.5V, -40°C <Ta<105°C*
V_{ADC}	Workable ADC operating Voltage	2.4		VDD	V	V_{DD}=5V
V_{AD}	AD Input Voltage	0		$ADCV_{REF}$	V	
ADrs	ADC resolution			12	bit	
ADclk	ADC clock period		2		us	V_{ADC}=2.4V ~ VDD
t_{ADCONV}	ADC conversion time (T_{ADCLK} is the period of the selected AD conversion clock)		16		T_{ADCLK}	
AD DNL	ADC Differential NonLinearity		±3*		LSB	
AD INL	ADC Integral NonLinearity		±3*		LSB	
ADos	ADC offset*		3		LSB	-40°C <Ta<105°C*
t_{INT}	Interrupt pulse width	30			ns	V_{DD} = 5.0V
V_{DR}	RAM data retention voltage*	1.5			V	In power-down mode
t_{WDT}	Watchdog timeout period (T_{ILRC} is the clock period of ILRC)		4096		T_{ILRC}	misc[1:0]=01
			16384			misc[1:0]=10
t_{SBP}	System boot-up period from power-on		2600		T_{ILRC}	Where T_{ILRC} is the clock period of ILRC
System Wake-up Time						
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	
LCP_{thr}	OCP threshold	50	100	210	mV	Limit current comparator

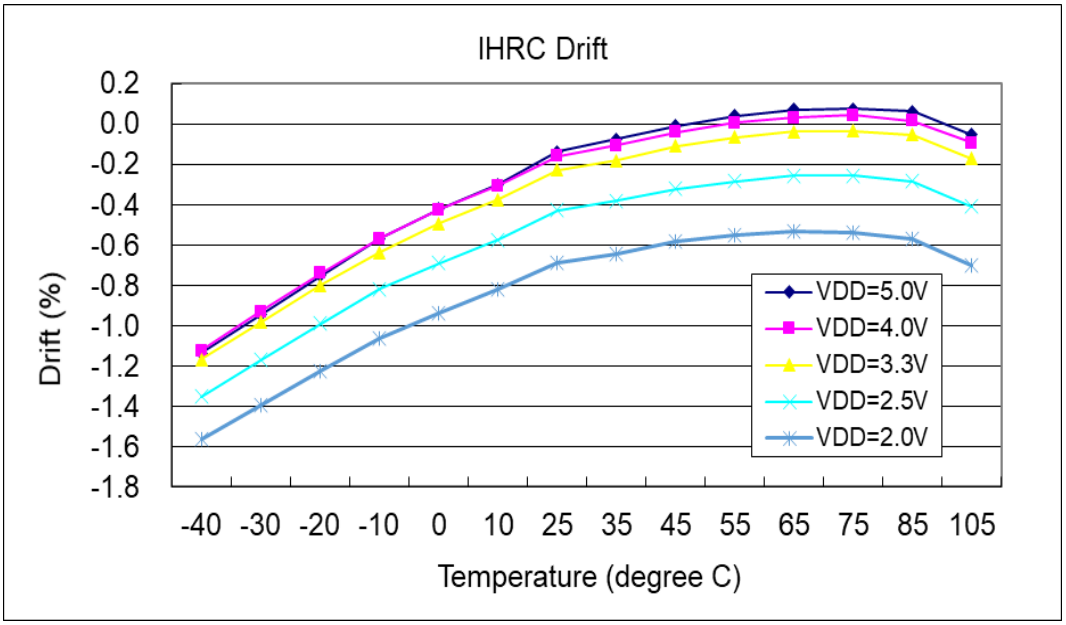
*These parameters are for design reference, not tested for every chip.

**The characteristic diagrams are the actual measured values. Considering the influence of production drift and other factors, the data in the table are within the safety range of the actual measured values.

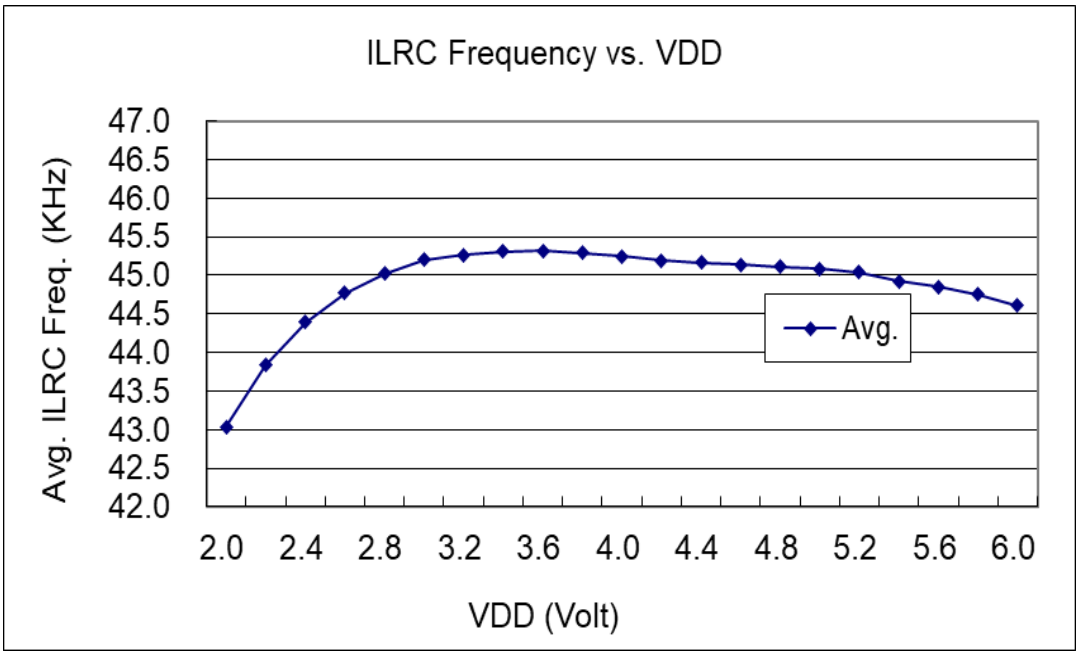
4.3. Typical ILRC frequency vs. VDD and temperature



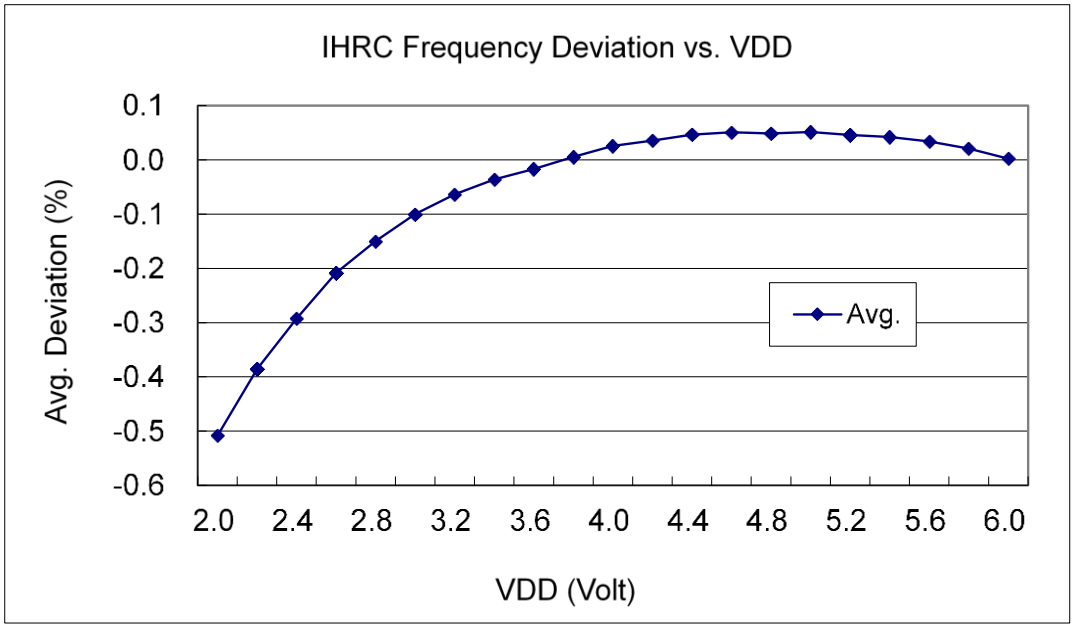
4.4. Typical IHRC frequency deviation vs. VDD and temperature



4.5. Typical ILRC frequency vs. VDD



4.6. Typical IHRC frequency deviation vs. VDD

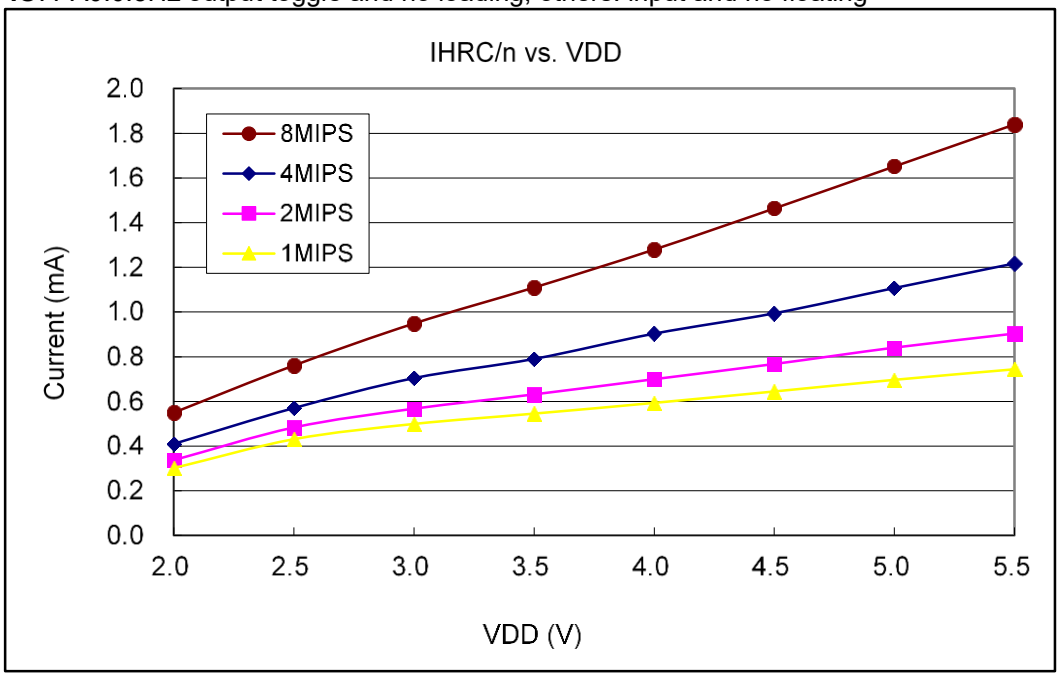


4.7. Typical Operating Current vs. VDD and CLK=IHRC/n

Conditions: 1-FPPA (code option)

ON: Bandgap, LVR, IHRC; **OFF**: ILRC, T16, TM2, ADC, PWM;

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

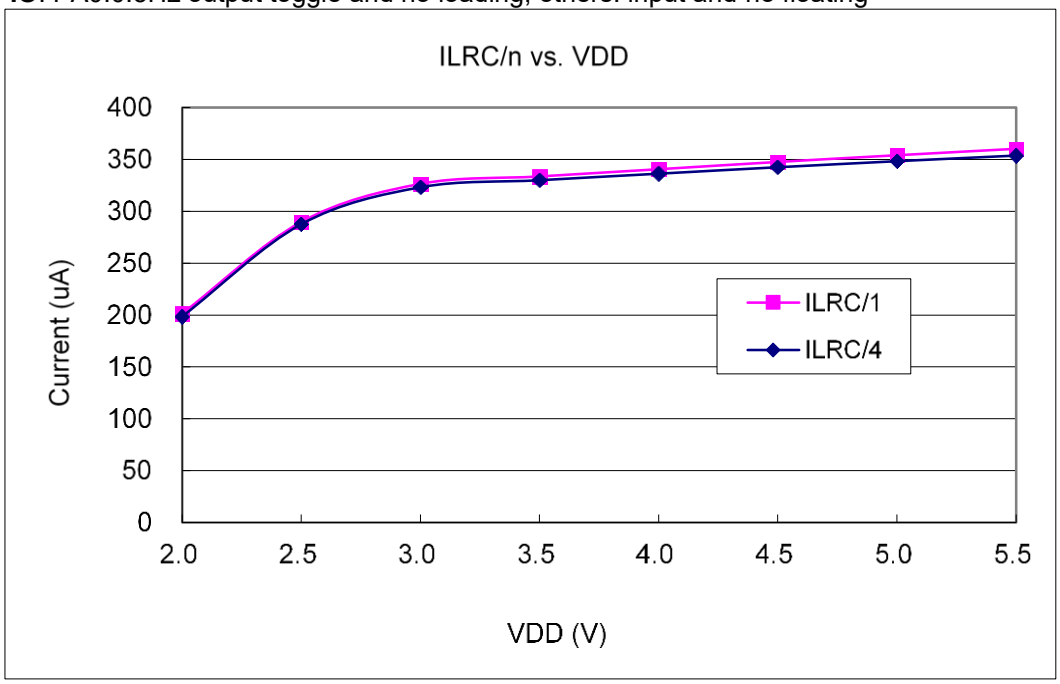


4.8. Typical Operating Current vs. VDD and CLK=ILRC/n

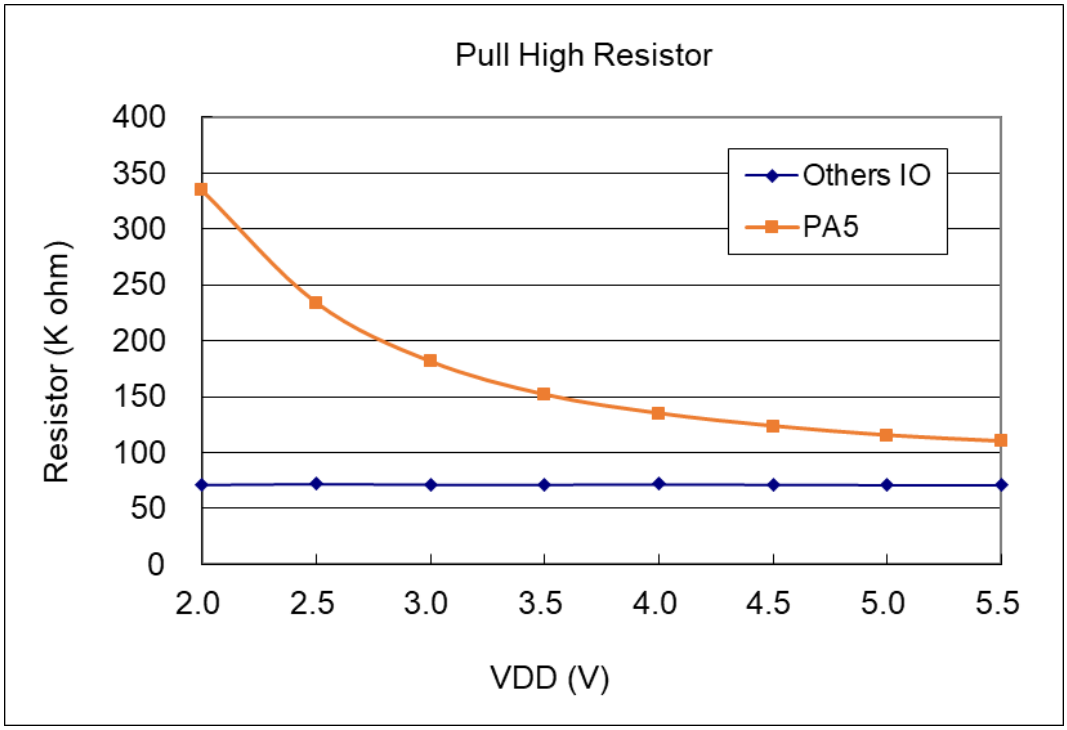
Conditions: 1-FPPA (code option)

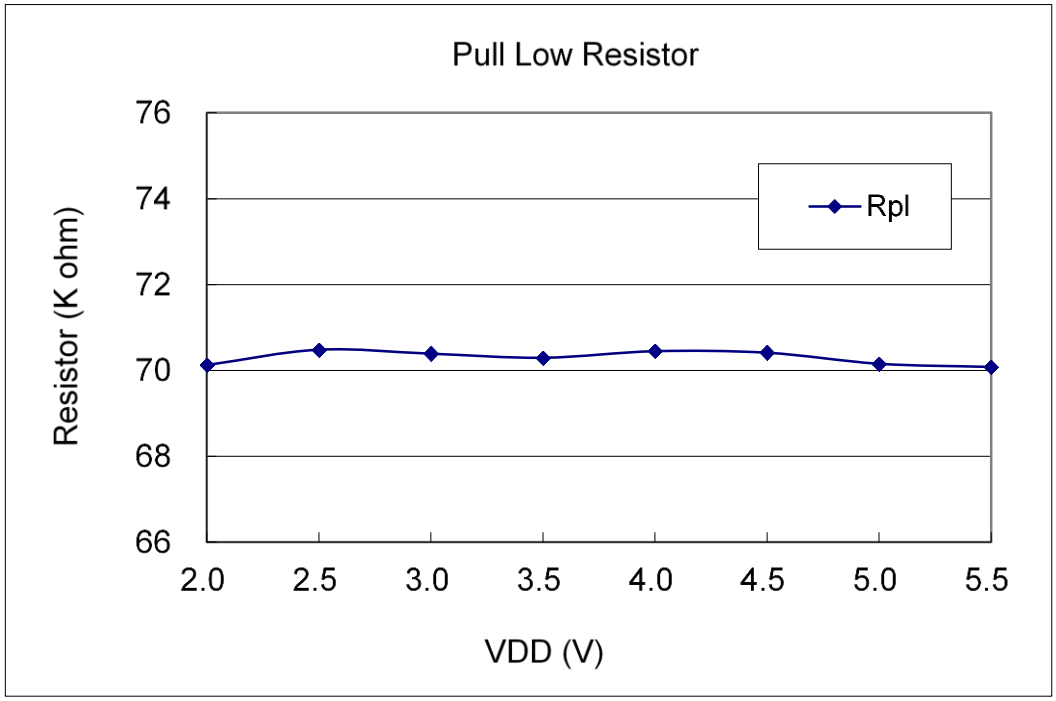
ON: ILRC; **OFF**: Bandgap, LVR, IHRC, T16, TM2, ADC, PWM;

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



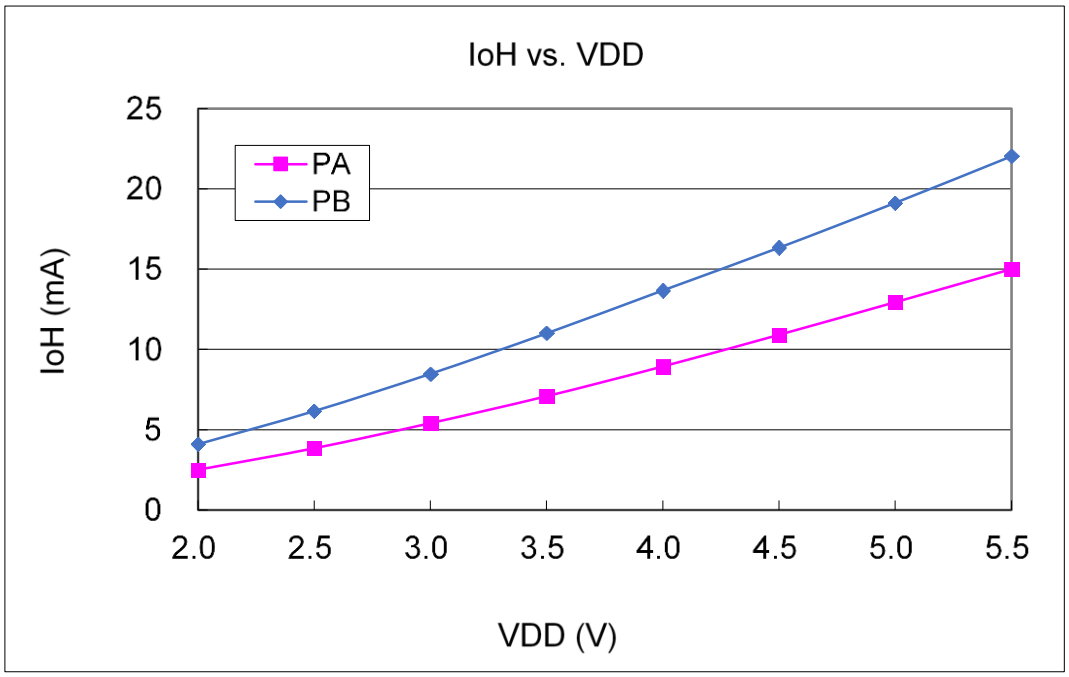
4.9. Typical IO pull high / low resistance

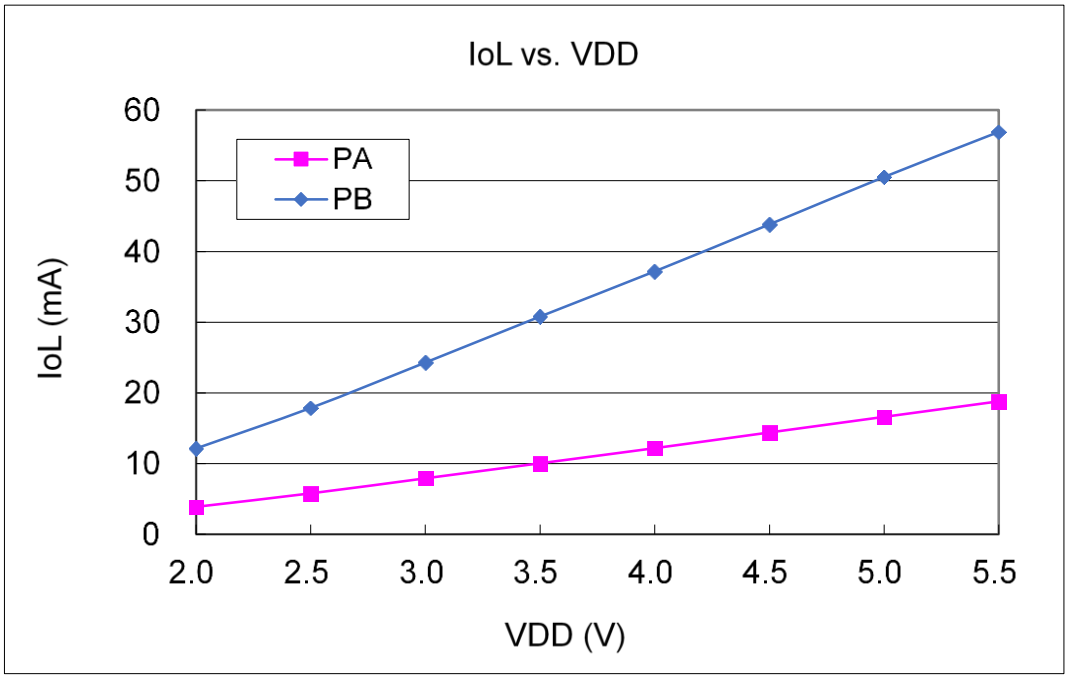




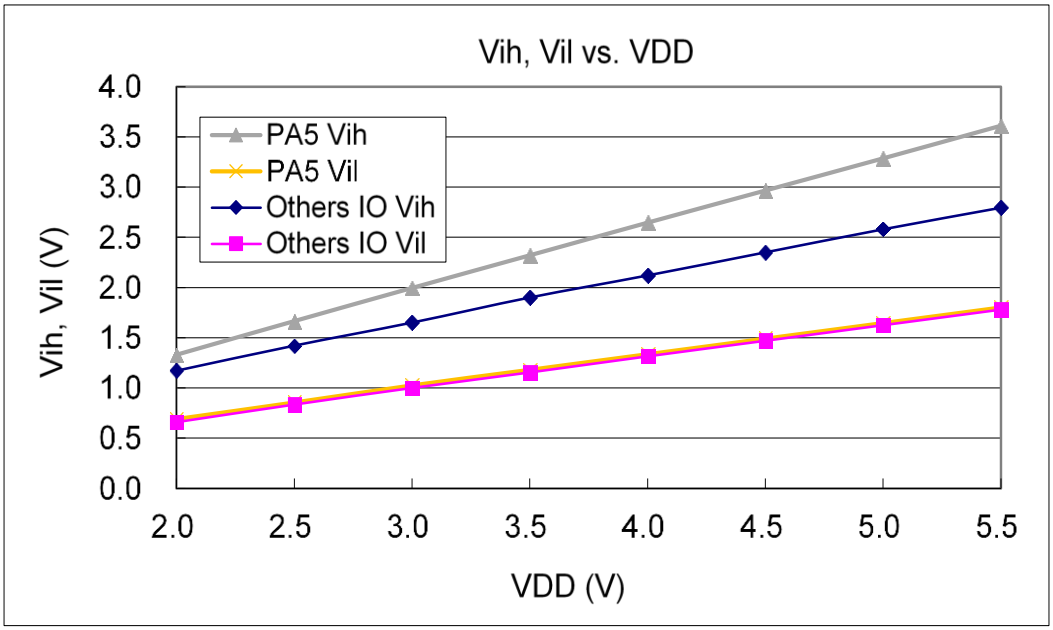
4.10. Typical IO driving current (IOH) and sink current (IOL)

(VOH=0.9*VDD, VOL=0.1*VDD)





4.11. Typical IO input high/low threshold voltage (VIH/VIL)



5. Central Processing Unit (CPU)

5.1. Functional Description

There are eight processing units (FPPA unit) inside the chip of MF324. In each processing unit, it includes:

a. Its own Program Counter to control the program execution sequence
b. Its own Stack Pointer to store or restore the program counter for program execution
c. Its own accumulator
d. Status Flag to record the status of program execution.

Each FPPA unit has its own program counter and accumulator for program execution, flag register to record the status, and stack pointer for jump operation. Based on such architecture, FPPA unit can execute its own program independently, thus parallel processing can be expected.

5.1.1. Processing Units

These eight FPPA units share the same 3Kx15 bits MTP user program memory, 192 bytes data SRAM and all the IO ports, these eight FPPA units are operated at mutual exclusive clock cycles to avoid interference. One task switch is built inside the chip to decide which FPPA unit should be active for the corresponding cycle. The hardware diagram of processing units is illustrated in Fig. 1.

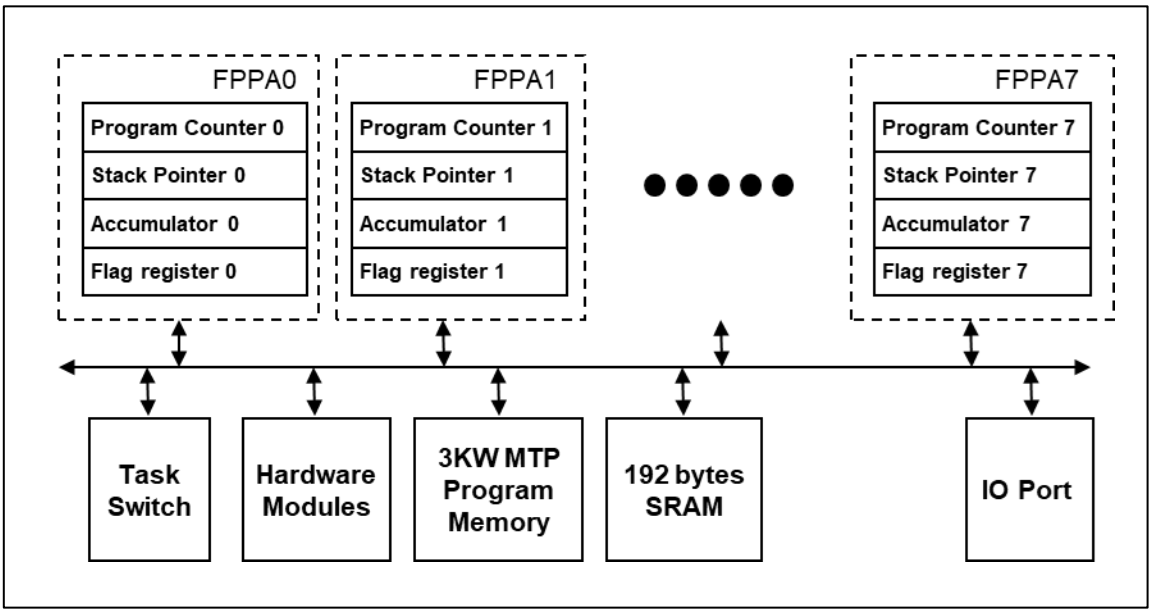


Fig. 1: Hardware Diagram of FPPA units

These eight FPPA units are operated at mutual exclusive clock cycles and can be enabled independently.

The system performance is shared to the assigned FPPA units via *pmode* command; please refer to the description of *pmode* instruction. The bandwidth assignment is nothing to do with FPPA enable, means that the bandwidth is also allocated to the assigned FPPA unit even though it is disabled.

Two FPPA units

Fig. 2 shows the timing sequence of FPPA units for *pmode*=0 which will assign the bandwidth to two FPPA units only. FPPA0 and FPPA1 each have half computing power of whole system; for *pmode*=0, FPPA0 and FPPA1 will be operated at 4MHz if system clock is 8MHz.

For FPPA0 unit, its program will be executed in sequence every other system clock, shown as (M-1)th, Mth, …. (M+4)th instructions. For FPPA1 unit, its program will be also executed in sequence every other system clock, shown as (N-1)th, Nth, …. (N+3)th instructions.

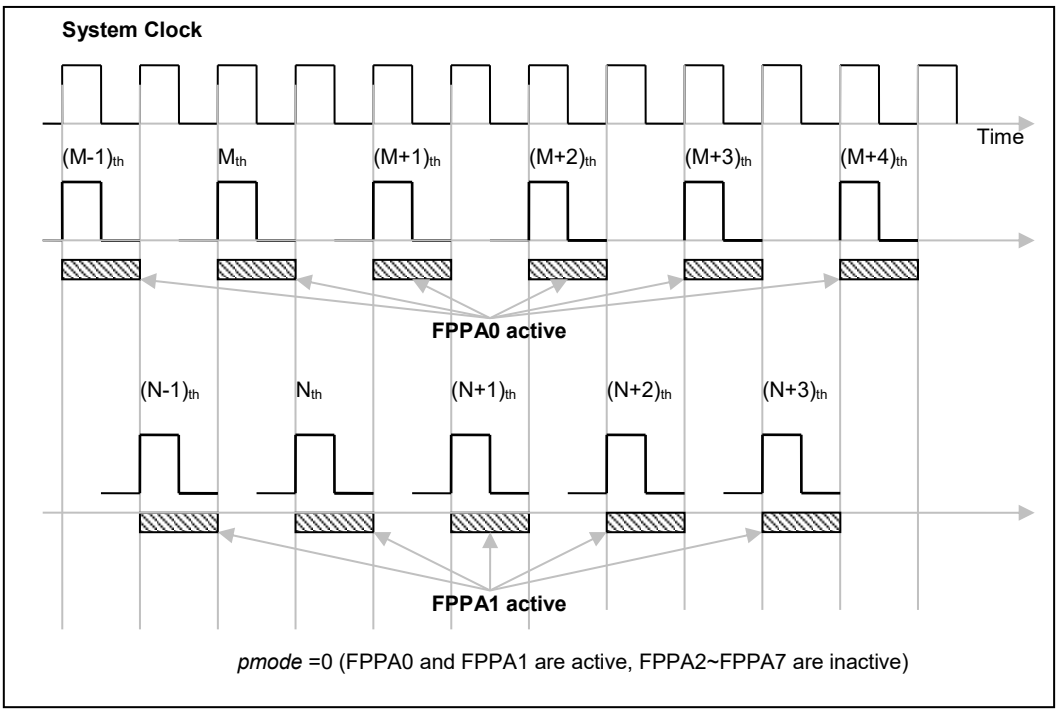


Fig. 2: Timing Sequence of Processing units for *pmode*=0

Four FPPA units

Fig. 3 shows the timing sequence of FPPA units for *pmode*=6 which will assign the bandwidth to four FPPA units (FPPA0, FPPA1, FPPA2, FPPA3); for *pmode*=6, FPPA0, FPPA1, FPPA2 and FPPA3 will be operated at 2MHz if system clock is 8MHz, means that each FPPA unit has quarter computing power of whole system, however, FPPA4, FPPA5, FPPA6 and FPPA7 are inactive.

For FPPA0 unit, its program will be executed once in sequence every four system clock, shown as (M-1)th, Mth, …. (M+4)th instructions. For FPPA1 unit, its program will be also executed once in sequence every four system clock, shown as (N-1)th, Nth, …. (N+3)th instructions. For FPPA2 unit, its program will be also executed once in sequence every four system clock, shown as (O-1)th, Oth, …. (O+3)th instructions. For FPPA3 unit, its program will be also executed once in sequence every four system clock, shown as (P-1)th, Pth, …. (P+3)th instructions.

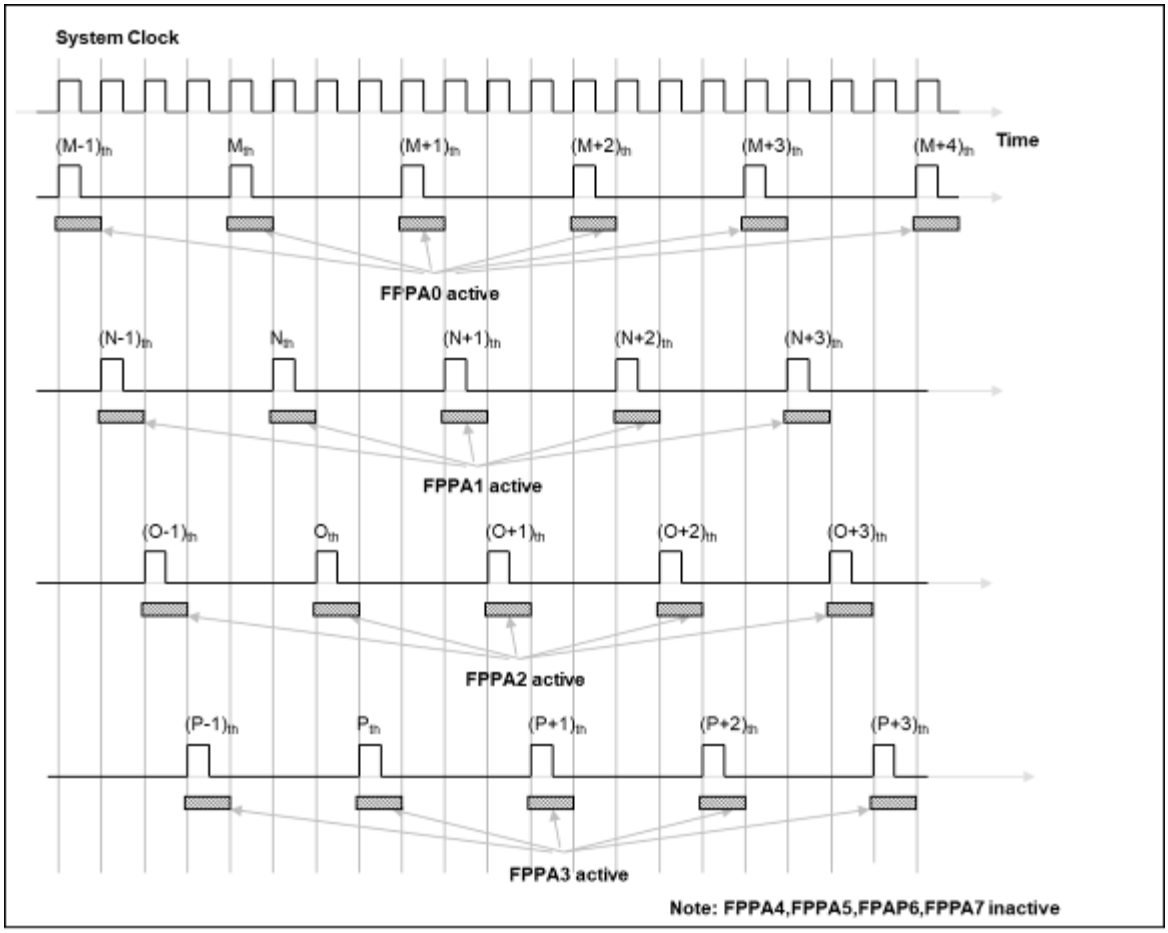


Fig. 3: Timing Sequence of Processing units for *pmode*=6

The FPPA unit can be enabled or disabled by programming the FPPA unit Enable Register, only FPPA0 is enabled after power-on reset. The system initialization will be started from FPPA0 and other units can be enabled by user's program if necessary. All the FPPA units can be enabled or disabled by using any one FPPA unit, including itself.

5.1.2. Program Counter

Program Counter (PC) is the unit that contains the address of an instruction to be executed next. The program counter is automatically incremented at each instruction cycle so that instructions are retrieved sequentially from the program memory. Certain instructions, such as branches and subroutine calls, interrupt the sequence by placing a new value in the program counter.

The bit length of the program counter is 13 for MF324. The program counter of FPPA0 is 0 after hardware reset, 1 for FPPA1, 2 for FPPA2, 3 for FPPA3, 4 for FPPA4, 5 for FPPA5, 6 for FPPA6 and 7 for FPPA7. Whenever interrupt event happens, only FPPA0 will be informed and its program counter will jump to 0x10 for interrupt service routine. All the FPPA units have its own program counter to control the program execution sequence.

5.1.3. Program Structure

After power-up, the program starting address of FPPA0 is 0x000, 0x001 for FPPA1, 0x002 for FPPA2, 0x003 for FPPA3, 0x004 for FPPA4, 0x005 for FPPA5, 0x006 for FPPA6 and 0x007 for FPPA7. The 0x010 is the entry address of interrupt service routine, which belongs to FPPA0 only. The basic firmware structure for MF324 is shown as Fig. 4, it shows that there are four FPPA units are used; the program codes of four FPPA units are placed in one whole program space. Except for the initial addresses of processing units and entry address of interrupt, the memory location is not specially specified; the program codes of processing unit can be resided at any location no matter what the processing unit is. After power-up, the FPPA0Boot will be executed first, which will include the system initialization and other FPPA units enabled.

```
// Page 1
.romadr    0x00
// Program Begin
goto       FPPA0Boot;
goto       FPPA1Boot;
goto       FPPA2Boot;
goto       FPPA3Boot;
//------Interrpt service Routine----------------
.romadr     0x010
pushaf ;
t0sn        intrq.0;       //PA.7 ISR
goto        ISR_PA7;
//------End of ISR---------
//------ Begin of FPPA0 ----------
FPPA0Boot :
 //--- Initialize FPPA0 SP and so on…
…
FPPA0Loop:
…
goto FPPA0Loop:
//------ End of FPPA0 --------
```

```
// Page 2
//------ Begin of FPPA1 ----------
FPPA1Boot :
//--- Initialize FPPA1 SP and so on…
FPPA1Loop:
…
goto FPPA1Loop:
//--------- End of FPPA1 --------
//------ Begin of FPPA2 ----------
FPPA2Boot :
//--- Initialize FPPA2 SP and so on…
FPPA2Loop:
…
goto FPPA2Loop:
//--------- End of FPPA2 --------
//------ Begin of FPPA3 ----------
FPPA3Boot :
//--- Initialize FPPA3 SP and so on…
FPPA3Loop:
…
goto FPPA3Loop:
//--------- End of FPPA3 --------
```

Fig. 4: Program Structure

5.1.4. Arithmetic and Logic Unit

Arithmetic and Logic Unit (ALU) is the computation element to operate integer arithmetic, logic, shift and other specialized operations. The operation data can be from instruction, accumulator or SRAM data memory. Computation result could be written into accumulator or SRAM. All the FPPA units share ALU for its corresponding operation.

5.2. Storage Memory

5.2.1. Program Memory – ROM

The MF324 program memory is MTP (Multiple Time Programmable), used to store data (including: data, tables and interrupt entry) and program instructions to be executed. All the user program codes for all FPPA units are stored in this 3KW MTP memory which is partitioned as Table 1.

After reset, the initial address for FPPA0 is 0x000, 0x001 for FPPA1, 0x002 for FPPA2, 0x003 for FPPA3, 0x004 for FPPA4, 0x005 for FPPA5, 0x006 for FPPA6, 0x007 for FPPA7. The interrupt entry is 0x010 if used and interrupt function is for FPPA0 only.

The MTP memory from address 0xBE0 to 0xBFF are for system using, address space from 0x008 to 0x00F and from 0x011 to 0xBDF are user program spaces. And the address 0x000 to 0x007 is the FPPA units initial address.

Address	Function
0x000	FPPA0 reset – *goto* instruction
0x001	FPPA1 reset – *goto* instruction
0x002	FPPA2 reset – *goto* instruction
0x003	FPPA3 reset – *goto* instruction
0x004	FPPA4 reset – *goto* instruction
0x005	FPPA5 reset – *goto* instruction
0x006	FPPA6 reset – *goto* instruction
0x007	FPPA7 reset – *goto* instruction
0x008	User program memory
•	•
0x00F	User program memory
0x010	Interrupt entry address
0x011	User program memory
•	•
•	•
0xBDF	User program memory
0xBE0	System using
•	•
0xBFF	System using

Table 1: Program Memory Organization

Example of Using Program Memory for Two FPPA mode

In order to have maximum flexibility for user program using, the user program memory is shared for all FPPA units, and the program space allocation is done by program compiler automatically, user does not need to specify the address if not necessary. Table 2 shows one example of program memory using which two FPPA units are used.

Address	Function
0x000	FPPA0 reset – goto instruction (*goto* 0x020)
0x001	FPPA1 reset – goto instruction (*goto* 0x7A1)
0x002	Reserved
•	•
0x007	Reserved
0x008	Not used
•	•
0x00F	Not used
0x010	Interrupt entry address(FPPA0 only)
•	•
0x01F	End of ISR (depend on user code size)
0x020	Begin of FPPA0 user program
•	•
•	•
0x7A0	End of FPPA0 user program
0x7A1	Begin of FPPA1 program
•	•
•	•
0xB37	End of FPPA1 program
0xB38	Not used
•	•
•	•
0xBDF	Not used
0xBE0	System Using
•	•
0xBFF	System Using

Table 2: Example of Program Memory Using

5.2.2. Data Memory – SRAM

Fig. 5 shows the SRAM data memory organization of MF324, all the SRAM data memory could be accessed by every FPPA unit directly with 1T clock cycle. 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 for all FPPA units.

The stack memory for each processing unit should be independent from each other, and defined in the data memory. The stack pointer is defined in the stack pointer register of each processing unit; 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. All the 192 bytes SRAM data memory of MF324 can be accessed by indirect access mechanism.

Bit defined: Only addressed at 0x00 ~ 0x7F.

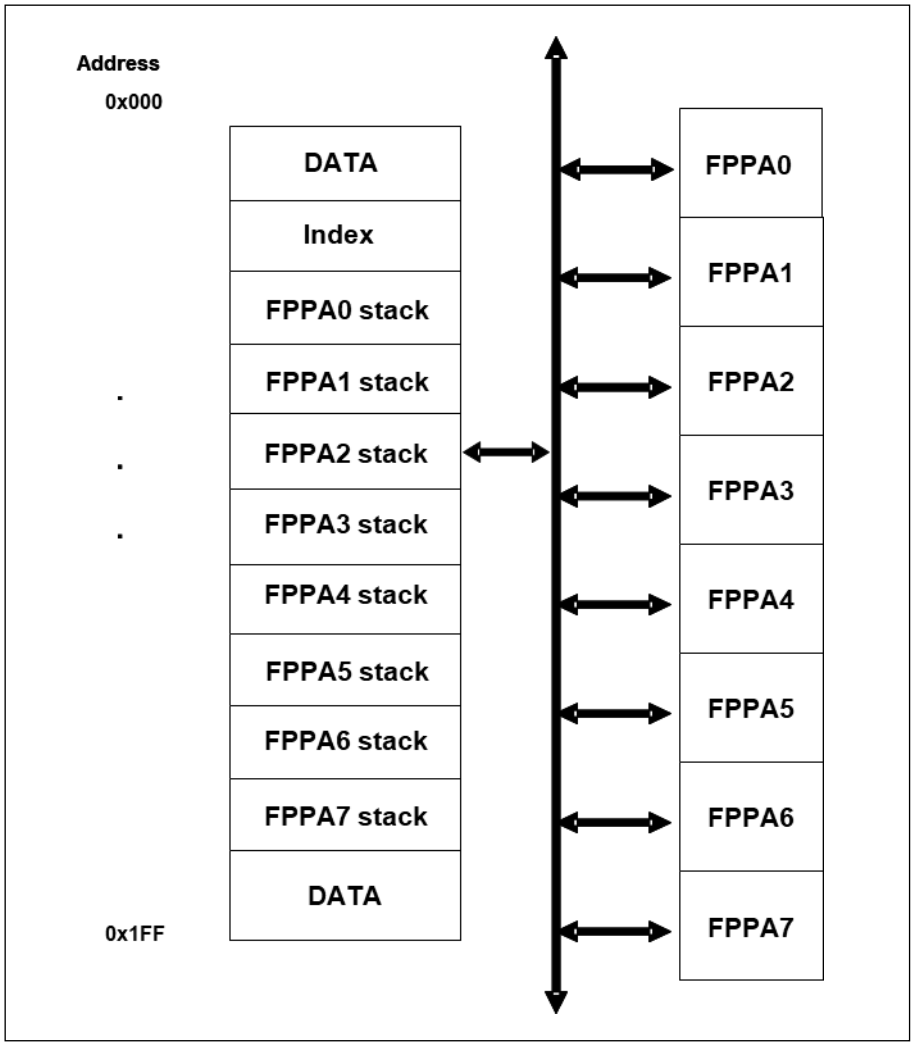


Fig. 5: SRAM Data Memory Organization

5.2.3. System Register

The register space of MF324 is independent of SRAM space and MTP space.

The following is the MF324 register address and brief description:

	+0	+1	+2	+3	+4	+5	+6	+7
0x00	*FLAG*	*FPPAEN*	*SP*	*CLKMD*	*INTEN*	*INTRQ*	*T16M*	
0x08	*INTEN2*	*INTRQ2*			*INTEGS*	*PADIER*		*PA*
0x10	*PAC*	*PAPH*	*PAPL*	*PB*	*PBC*	*GPCC*	*GPCR*	
0x18	*PWMPBC1*	*PWMPBC2*	*PWMGC1*	*PWMGC2*	*PWMGC*	*EARITH*	*ADCRH*	*ADCRL*
0x20	*ADCC*	*ADCM*	*TM4C*	*TM4CTH*	*TM4CTL*	*TM4BH*	*TM4BL*	*TM2C*
0x28	*TM2CT*	*TM2S*	*TM2B*	*TM3C*	*TM3CT*	*TM3S*	*TM3B*	*LCS*
0x30	*ZCPC*	*ZCPS*	*PLSCC*	*PLSCS-*	*MISC*	*M8OP1-*	*M8OP2*	*M8RS1*
0x38	*M8RS0*	*PLSPWH*	*PLSPWL*	*PLSPHH*	*PLSPHL*	*PWMUPBH*	*PWMUPBL*	*PWM0DTH*
0x40	*PWM0DTL*	*PWMADCH*	*PWMADCL*	*PWMDDZVF*	*PWMDDZVR*			*OPR2*

5.2.3.1. ACC Status Flag Register (*FLAG*), address = 0x00

Bit	Reset	R/W	Description
7 - 4	1111	-	Reserved. These four bits are “1” when read.
3	0	R/W	OV (Overflow). This bit is set whenever the sign operation is overflow.
2	0	R/W	AC (Auxiliary Carry). 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). 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). This bit will be set when the result of arithmetic or logic operation is zero; Otherwise, it is cleared.

5.2.3.2. Multi-Core Enable Register (*MCUEN*), address = 0x01

Bit	Reset	R/W	Description
7	0	R/W	FPPA7 enable. This bit is used to enable FPPA7. 0 / 1: disable / enable
6	0	R/W	FPPA6 enable. This bit is used to enable FPPA6. 0 / 1: disable / enable
5	0	R/W	FPPA5 enable. This bit is used to enable FPPA5. 0 / 1: disable / enable
4	0	R/W	FPPA4 enable. This bit is used to enable FPPA4. 0 / 1: disable / enable
3	0	R/W	FPPA3 enable. This bit is used to enable FPPA3. 0 / 1: disable / enable
2	0	R/W	FPPA2 enable. This bit is used to enable FPPA2. 0 / 1: disable / enable
1	0	R/W	FPPA1 enable. This bit is used to enable FPPA1. 0 / 1: disable / enable
0	1	R/W	FPPA0 enable. This bit is used to enable FPPA0. 0 / 1: disable / enable

5.2.3.3. Option 2 Register (*OPR2*), address = 0x47

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved. Please keep 0
4	-	-	Reserved. Please keep 0
3	-	-	Reserved. Please keep 0
2	0	WO	Option for Timer16 clock pre-divider selection. Please see the *T16M* register.
1	-	-	Reserved. Please keep 0
0	0	WO	Option for external interrupt 0 pin selection. 0 / 1: PA4 / PA3

5.2.3.4. MISC Register (*MISC*), address = 0x34

Bit	Reset	R/W	Description
7	0	WO	PWMO0 / PWMO2 / PWMO4, Emergency Stop Disable / Enable
6	0	WO	PWMO1 / PWMO3 / PWMO5, Emergency Stop Disable / Enable
5 - 3	-	-	Reserved
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 - 0	10	WO	Watch dog time out period 00: Reserved 01: 4096 ILRC clock period 10: 16384 ILRC clock period 11: Reserved

5.3. The Stack

The stack pointer in each processing unit is used to point the top of the stack area where the local variables and parameters to subroutines are stored; the stack pointer register (*SP*) is located in register address 0x02. The bit number of stack pointer is 7 bit; the stack memory cannot be accessed over 128 bytes and should be defined within 128 bytes from 0x00 address. The stack memory of MF324 for each FPPA unit can be assigned by user via stack pointer register, means that the depth of stack pointer for each FPPA unit is adjustable in order to optimize system performance. The following example shows how to define the stack in the ASM (assembly language) project:

```
        .ROMADR0
        GOTO        FPPA0
        GOTO        FPPA1
        ...
        .RAMADR0                    // Address must be less than 0x100
        WORD        Stack0 [1]      // one WORD
        WORD        Stack1 [2]      // two WORD
        ...
FPPA0:
    SP      =       Stack0;         // assign Stack0 for FPPA0,
                                    // one level call because of Stack0[1]
        ...
        call        function1
        ...
FPPA1:
        SP  =       Stack1;         // assign Stack1 for FPPA1,
                                    // two level call because of Stack1[2]
        ...
        call        function2
        ...
```

In Mini-C project, the stack calculation is done by system software, user will not have effort on it, and the example is shown as below:

```
void        FPPA0 (void)
{
            ...
}
```

User can check the stack assignment in the window of program disassembling, Fig. 6 shows that the status of stack before FPPA0 execution, system has calculated the required stack space and has reserved for the program.

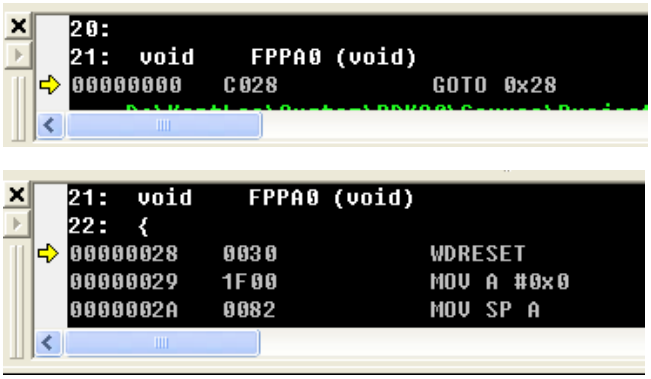


Fig. 6: Stack Assignment in Mini-C project

5.3.1. Stack Pointer Register (*SP*), 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.

5.4. Code Options

Option	Selection	Description
Security	Enable	MTP content is protected and program cannot be read back
	Disable(default)	MTP content is not protected so program can be read back
LVR	64us	LVR IHRC de-glitch > 64us
	32us	LVR IHRC de-glitch > 32us
	16us	LVR IHRC de-glitch > 16us
	8us	LVR IHRC de-glitch > 8us
	4us	LVR IHRC de-glitch > 4us
	2us	LVR IHRC de-glitch > 2us
	Disable	LVR IHRC de-glitch Disable

6. Oscillator and System Clock

There are two oscillator circuits provided by MF324: internal high RC oscillator (IHRC) and internal low RC oscillator (ILRC). These three oscillators are enabled or disabled by registers *CLKMD*.4 and *CLKMD*.2 independently.

User can choose one of these two oscillators as system clock source and use *CLKMD* register to target the desired frequency as system clock to meet different applications.

Oscillator Module	Enable / Disable	Default after boot-up
IHRC	*CLKMD*.4	Enabled
ILRC	*CLKMD*.2	Enabled

Table 3: Two Oscillator Circuits provided by MF324

6.1. 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 2% normally after calibration and it still drifts slightly with supply voltage and operating temperature.

The frequency of ILRC will also vary by process, supply voltage and temperature. Please refer to the measurement chart for IHRC / ILRC frequency verse V_{DD} and IHRC / ILRC frequency verse temperature.

The MF324 writer tool provides IHRC frequency calibration (usually up to 16MHz) to eliminate frequency drift caused by factory production. ILRC has no calibration operation. For applications that require accurate timing, please do not use the ILRC clock as a reference time.

6.2. System Clock and IHRC Calibration

6.2.1. System Clock

The clock source of system clock comes from IHRC and ILRC, the hardware diagram of system clock in the MF324 is shown as Fig. 7.

After power up, the running system clock will be ILRC/1, user can change the system clock any time by setting *CLKMD* register and the new system clock will be changed into the new one immediately after writing the *CLKMD* register.

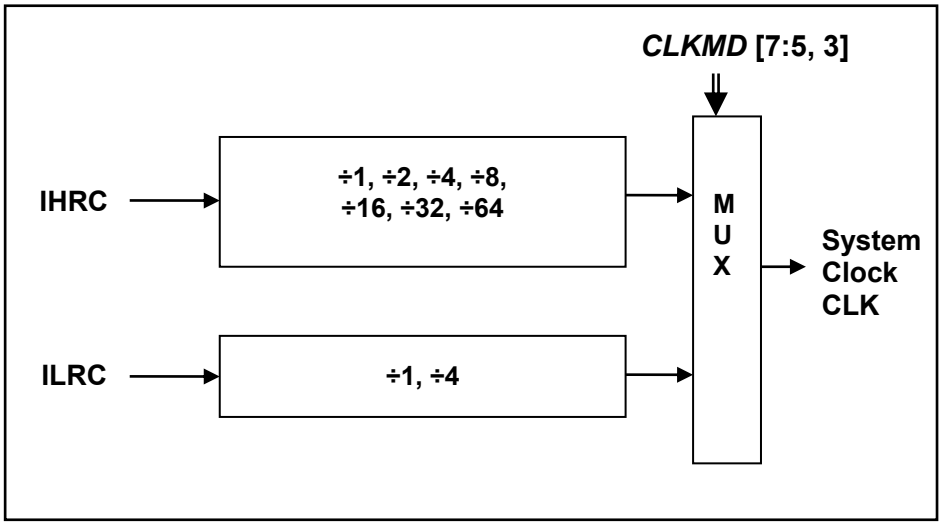


Fig. 7: Options of System Clock

6.2.1.1. Clock Mode Register (CLKMD), address = 0x03

Bit	Reset	R/W	Description	
7 - 5	111	R/W	System clock selection	
			Type 0, *CLKMD*[3]=0	Type 1, *CLKMD*[3]=1
			000: IHRC/4 001: IHRC/2 010 / 011 / 100 / 101: reserved 110: ILRC/4 111: ILRC (default)	000: IHRC/16 001: IHRC/8 010: reserved 011: IHRC/32 100: IHRC/64 101 / 11x: reserved
4	1	R/W	IHRC oscillator 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	ILRC 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	

6.2.2. Frequency Calibration

The IHRC frequency and Bandgap reference voltage may be different chip by chip due to manufacturing variation, MF324 provide both the IHRC frequency calibration and Bandgap 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, Bandgap=(**p4**);

Where, **p1**=2, 4, 8, 16, 32, 64; In order to provide different system clock.
p2=16 ~ 18; In order to calibrate the chip to different frequency, 16MHz is the usually one.
p3=2.7 ~ 5.5; In order to calibrate the chip under different supply voltage.
p4= On or Off; Bandgap calibration is On or Off.

Usually, .ADJUST_IC will be the first command after boot up, in order to set the target operating frequency whenever stating the system. The program code for IHRC frequency calibration is executed only one time that occurs in writing the codes into MTP 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. As shown in table 4:

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 4: Options for IHRC Frequency Calibration

The following shows the different states of MF324 under different options:

(1) *.ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, V_{DD}=5V*

After boot, *CLKMD* = 0x14:

a. IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is enabled
b. System CLK = IHRC/4 = 4MHz
c. Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

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

After boot, *CLKMD* = 0xE4:

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

(3) *.ADJUST_IC DISABLE*

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

a. IHRC is not calibrated and IHRC module is disabled
b. System CLK = ILRC
c. Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode

6.2.2.1. Special Statement

(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.

6.2.3. System Clock Switching

After IHRC calibration, the system clock of MF324 can be switched between IHRC and ILRC by setting the *CLKMD* register at any time, **but please notice that the original clock module can NOT be turned off at the same time as writing command to *CLKMD* register.** For example, when switching from A clock source to B clock source, you should first switch the system clock source to B and then close the A clock source. 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/4

```
...                             //  system clock is ILRC
CLKMD.4     =   1;              //  turn on IHRC first to improve anti-interference ability
CLKMD       =   0x14;           //  switch to IHRC/4, ILRC CAN NOT be disabled here
// CLKMD.2  =   0;              //  if need, ILRC CAN be disabled at this time
...
```

Case 2: Switching system clock from IHRC/8 to IHRC/4

```
...                             //  system clock is IHRC/8, ILRC is enabled here
CLKMD       =   0x14;           //  switch to IHRC/4
...
```

Case 3: System may hang if it is to switch clock and turn off original oscillator at the same time

```
...                             //  system clock is ILRC
CLKMD       =   0x10;           //  CAN NOT switch clock from ILRC to IHRC/4 and turn off
                                //  ILRC oscillator at the same time
...
```

7. Reset

There are many causes to reset the MF324, the main four factors are: power-on reset, LVR reset, watchdog timeout overflow reset, and PRSTB pin reset. After the reset, the system will restart. The program counter will jump to address 0x000 and most of all the registers in MF324 will be set to the default values.

7.1. Power On Reset - POR

POR (Power-On-Reset) is used to reset MF324 when power up, however, the supply voltage may be not stable. To ensure the stability of supply voltage after power up, it will wait 2547 ILRC clock cycles before first instruction being executed, which is T_{SBP} and shown in the Fig. 8. After boot up procedure, the default system clock is ILRC. If user wants to switch the system clock source from ILRC to IHRC, user must enable the corresponding oscillator module and make sure clock is already stable.

The MF324 data memory is in an uncertain state when the power on reset occurs.

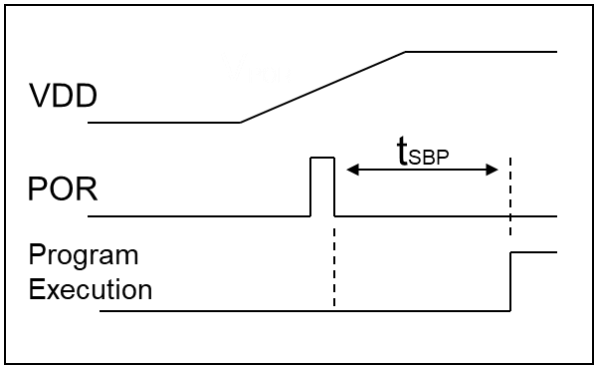


Fig.8: Power-On Sequence

Fig. 8 shows the typical program flow after boot up, it shows all the FPPA units are used. Please notice that the FPPA1~FPPA7 is disabled after reset and recommend NOT enabling FPPA1~FPPA7 before system and FPPA0 initialization.

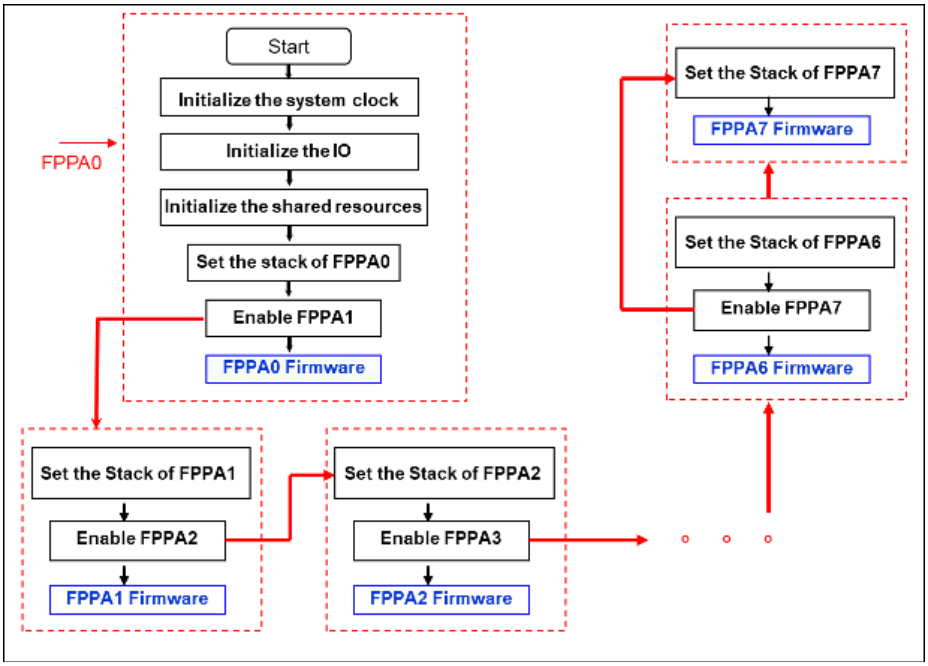


Fig. 9: Boot Procedure

7.2. Low Voltage Reset - LVR

User can choose different operating system clock depends on its requirement; the selected operating system clock should be in conjunction with supply voltage and LVR level to ensure system stable. If VDD drops below the LVR level, then the LVR Reset will occur in the system, and its timing diagram is shown in Fig.10.

After LVR reset, the SRAM data will be kept when VDD>VDR (SRAM data retention voltage). However, if SRAM is cleared after power-on again, the data cannot be kept, the data memory is in an uncertain state when VDD<VDR.

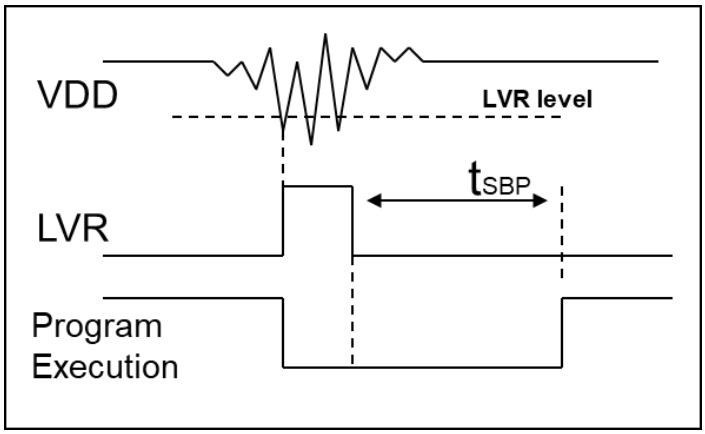


Fig. 10: Low Voltage Reset Sequence

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, and the lowest LVR levels can be chosen for different operating frequencies. Please refer to Section 13.1.

The following are suggestions for setting operating frequency, power supply voltage and LVR level:

SYSCLK	VDD	LVR
8MHz	≧2.75V	2.75V
4MHz	≧ 2.0V	2.0V

Table 5: LVR setting for reference

Users can set *MISC*.2 as 1 to disable the LVR function. At this time, it should be ensured that the VDD is above the minimum working voltage of the chip, otherwise the IC may not work properly.

MISC Register (*MISC*), address = 0x34			
Bit	**Reset**	**R/W**	**Description**
7	0	WO	PWMO0 / PWMO2 / PWMO4 Emergency Stop Disable / Enable
6	0	WO	PWMO1 / PWMO3 / PWMO5 Emergency Stop Disable / Enable
5 - 3	-	-	Reserved
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 - 0	00	WO	Watch dog time out period 00: Reserved 01: 4096 ILRC clock period 10: 16384 ILRC clock period 11: Reserved

7.3. Watch Dog Timeout Reset

The watchdog timer (WDT) is a counter with clock coming from ILRC, so it will be invalid if ILRC is off. The frequency of ILRC may drift a lot due to the variation of manufacture, supply voltage and temperature. User should reserve guard band for safe 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.

To ensure the watchdog is cleared before the timeout overflow, the instruction *wdreset* can be used to clear the WDT within a safe time. WDT can be cleared by power-on-reset or by command *wdreset* at any time.

When WDT is timeout, MF324 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig. 11.

The MF324 data memory will be reserved when the WDT reset occurs.

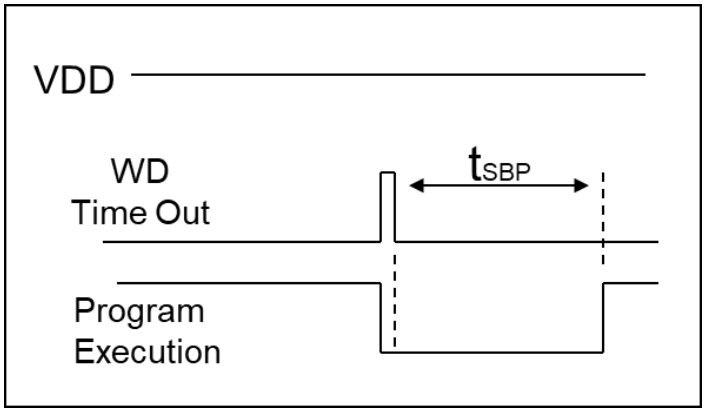


Fig. 11: Sequence of Watch Dog Timeout reset

There are two different timeout periods of watchdog timer can be chosen by setting the *MISC*[1:0]. And watchdog timer can be disabled by *CLKMD*.1.

Clock Mode Register (*CLKMD*), address = 0x03			
Bit	**Reset**	**R/W**	**Description**
7 – 5	111	R/W	System clock selection
4	1	R/W	IHRC oscillator Enable. 0 / 1: disable / enable
3	0	R/W	Clock Type Select.
2	**1**	**R/W**	**ILRC 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**

7.4. External Reset Pin - PRSTB

The MF324 supports external reset and its external reset pin shares the same IO port with PA5. Using external reset function requires:

(1) Set PA5 as input;

(2) Set *CLKMD*.0 =1 to make PA5 as the external PRSTB input pin.

When the PRSTB pin is high, the system is in normal working state. Once the reset pin detects a low level, the system will be reset. The timing diagram of PRSTB reset is shown in Fig. 12.

The MF324 data memory will be reserved when the PRSTB reset occurs.

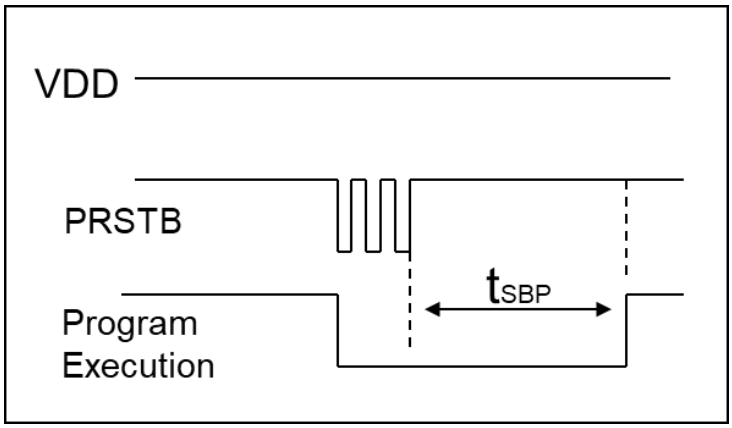


Fig. 12: Sequence of PRSTB reset

8. Interrupt

There are 9 interrupt lines for MF324:

1. Three external interrupt lines (PA3, PA4, PA6 the edge type is specified by *INTEGS* register)
2. GPC interrupt (the edge type is specified by *INTEGS* register)
3. Timer16 interrupt (the edge type is specified by *INTEGS* register)
4. PWM generator interrupt
5. ADC interrupt
6. Timer2 interrupt
7. LC interrupt
8. Timer3 interrupt
9. Timer4 interrupt

Every interrupt request line to CPU has its own corresponding interrupt control bit to enable or disable it. The hardware diagram of interrupt controller is shown as Fig. 13. All the interrupt request flags are set by hardware and cleared by writing *INTRQ / INTRQ2* register. When the request flags are set, it can be rising edge, falling edge or both, depending on the setting of register *INTEGS / INTEGS2*. 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.

Only FPPA0 can accept the interrupt request, other FPPA unit will not be interfered by interrupt. The stack memory for interrupt is shared with data memory and its address is specified by stack register *SP*. Since the program counter is 15 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 FPPA unit could be fully specified by user to achieve maximum flexibility of system.

During the interrupt service routine, the interrupt source can be determined by reading the *INTRQ / INTRQ2* register.

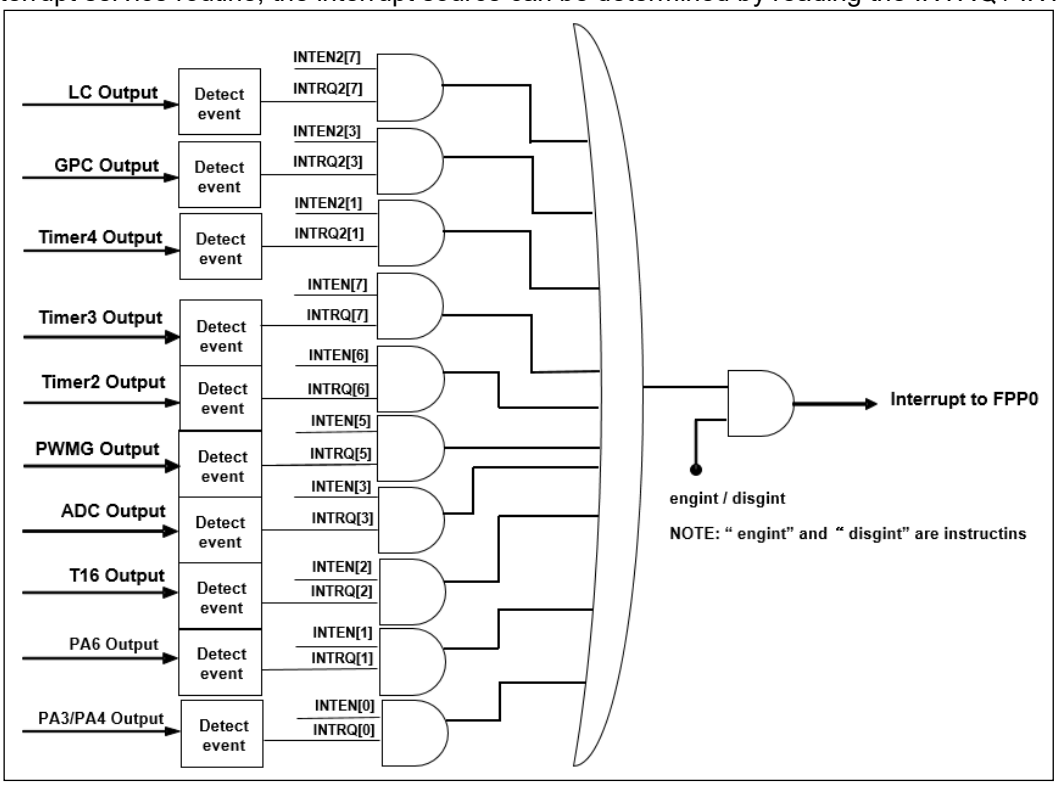


Fig. 13: Hardware diagram of Interrupt controller

Option 2 Register (*OPR2*), address = 0x47			
Bit	**Reset**	**R/W**	**Description**
7 – 5	-	-	Reserved. Please keep 0.
4	-	-	Reserved. Please keep 0.
3	-	-	Reserved. Please keep 0.
2	0	WO	Option for Timer16 clock pre-divider selection. Please see the *T16M* register
1	-	-	Reserved. Please keep 0.
0	**0**	**WO**	**Option for external interrupt 0 pin selection. 0 / 1: PA4 / PA3**

8.1. Interrupt Enable Register (*INTEN*), 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 PWM generator. 0 / 1: disable / enable	
4	-	-	Reserved. Please keep 0.	
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 PA6 pin. 0 / 1: disable / enable	
0	0	R/W	*OPR2*.0=0	Enable interrupt from PA4 pin. 0 / 1: disable / enable
			OPR2.0=1	Enable interrupt from PA3 pin. 0 / 1: disable / enable
Where, *OPR2* is an optional register with address 0x47.				

8.2. Interrupt Enable 2 Register (*INTEN2*), address = 0x08

Bit	Reset	R/W	Description
7	0	R/W	Enable interrupt from LC. 0 / 1: disable / enable
6	-	-	Reserved. Please keep 0.
5	-	-	Reserved. Please keep 0.
4	-	-	Reserved. Please keep 0.
3	0	R/W	Enable interrupt from GPC. 0 / 1: disable / enable
2	-	-	Reserved. Please keep 0.
1	0	R/W	Enable interrupt from Timer4. 0 / 1: disable / enable
0	-	-	Reserved. Please keep 0.

8.3. Interrupt Request Register (*INTRQ*), 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 PWM generator, this bit is set by hardware and cleared by software. 0 / 1: No request / Request	
4	-	-	Reserved. Please keep 0.	
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 PA6, this bit is set by hardware and cleared by software. 0 / 1: No request / Request	
0	-	R/W	*OPR2*.0=0	Interrupt Request from PA4 pin, this bit is set by hardware and cleared by software. 0 / 1: No Request / request
			OPR2.0=1	Interrupt Request from PA3 pin, this bit is set by hardware and cleared by software. 0 / 1: No Request / request
Where, *OPR2* is an optional register with address 0x47.				

8.4. Interrupt Request 2 Register (*INTRQ2*), address = 0x09

Bit	Reset	R/W	Description
7	-	R/W	Interrupt Request from LC, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
6	-	-	Reserved. Please keep 0.
5	-	-	Reserved. Please keep 0.
4	-	-	Reserved. Please keep 0.
3	-	R/W	Interrupt Request from GPC, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
2	-	-	Reserved. Please keep 0.
1	-	R/W	Interrupt Request from Timer4, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	-	Reserved. Please keep 0.

8.5. Interrupt Edge Select Register (*INTEGS*), address = 0x0C

Bit	Reset	R/W	Description
7	0	R/W	PWM interrupt selection. 0 : PWM counter Boundary mode 1 : PWM counter Zero mode
6 - 5	00	R/W	GPC interrupt 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.
4	0	R/W	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	R/W	PA6 interrupt 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	R/W	PA3 or PA4 interrupt 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 Note: If *OPR2*.0=0, PA4 is selected; If *OPR2*.0=1, PA3 is selected

Note:

INTEN/INTEN2 and *INTRQ/INTRQ2* have no initial values. Please set required value before enabling interrupt function. Even if *INTEN*=0, *INTRQ* will be still triggered by the interrupt source, so are the *INTEN2* and *INTRQ2*.

8.6. Interrupt Work Flow

Once the interrupt occurs, its operation will be:

(1) The program counter will be stored automatically to the stack memory specified by register *SP.*
(2) New *SP* will be updated to SP+2.
(3) Global interrupt will be disabled automatically.
(4) The next instruction will be fetched from address 0x010.

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

(1) The program counter will be restored automatically from the stack memory specified by register *SP.*
(2) New *SP* will be updated to SP-2.
(3) Global interrupt will be enabled automatically.
(4) The next instruction will be the original one before interrupt.

8.7. General Steps to Interrupt

When using the interrupt function, the procedure should be:

Step1: Set *INTEN / INTEN2* register, enable the interrupt control bit.

Step2: Clear *INTRQ / INTRQ2* 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.

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
```

* Use *disgint* in the main program can disable all interrupts.

8.8. Example for Using 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 PA4;        // INTEN =1; interrupt request when PA4 level changed
    INTRQ = 0;          // clear INTRQ
    INTRQ2 = 0;         // clear INTRQ2

    ENGINT              // global interrupt enable
    ...
    DISGINT             // global interrupt disable
    ...
}

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

    // If INTEN.PA4 will be opened and closed dynamically,
    // user can judge whether INTEN.PA0 =1 or not.
    // Example:  If (INTEN.PA4 && INTRQ.PA4)  {...}

    // If INTEN.PA4 is always enable,
    // user can omit the INTEN.PA4 judgement to speed up interrupt service routine.

    If (INTRQ.PA4)
    {                       // Here for PA4 interrupt service routine
        INTRQ.PA4 = 0;      // Delete corresponding bit (take PA4 for example)
        ...
    }
    ...
  // (X:) INTRQ = 0;       // It is not recommended to use INTRQ = 0 to clear all at the end of the
                           // interrupt service routine.
                           // It may accidentally clear out the interrupts that have just occurred
                           // and are not yet processed.
    POPAF                  // restore ALU and FLAG register
}
```

9. I/O Port

9.1. IO Related Registers

9.1.1. Port A Digital Input Enable Register (*PADIER*), address = 0x0D

Bit	Reset	R/W	Description
7 - 0	0xFF	WO	Enable PA7 ~ PA0 digital input 0 / 1 : disable / enable

9.1.2. Port A Data Registers (PA), address = 0x0F

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

9.1.3. Port A Control Registers (*PAC*), address = 0x10

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. Note: Besides PA5, PA5 is only input mode.

9.1.4. Port A Pull-High Registers (*PAPH*), address = 0x11

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. 0 / 1: disable / enable

9.1.5. Port A Pull-Low Registers (*PAPL*), address = 0x12

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A pull-low register. This register is used to enable the internal pull-low device on each corresponding pin of port A. 0 / 1: disable / enable Note: PA5 is for input only, PA5 isn't supported pull-low.

9.1.6. Port B Data Registers (PB), address = 0x13

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

9.1.7. Port B Control Registers (*PBC*), address = 0x14

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B control registers. This register is used to define only output mode for each corresponding pin of port B. 0 / 1: no function / output.

9.2. IO Structure and Functions

9.2.1. IO Pin Structure

All the IO pins of MF324 has the same structure. The hardware diagram of IO buffer is shown as Fig. 14.

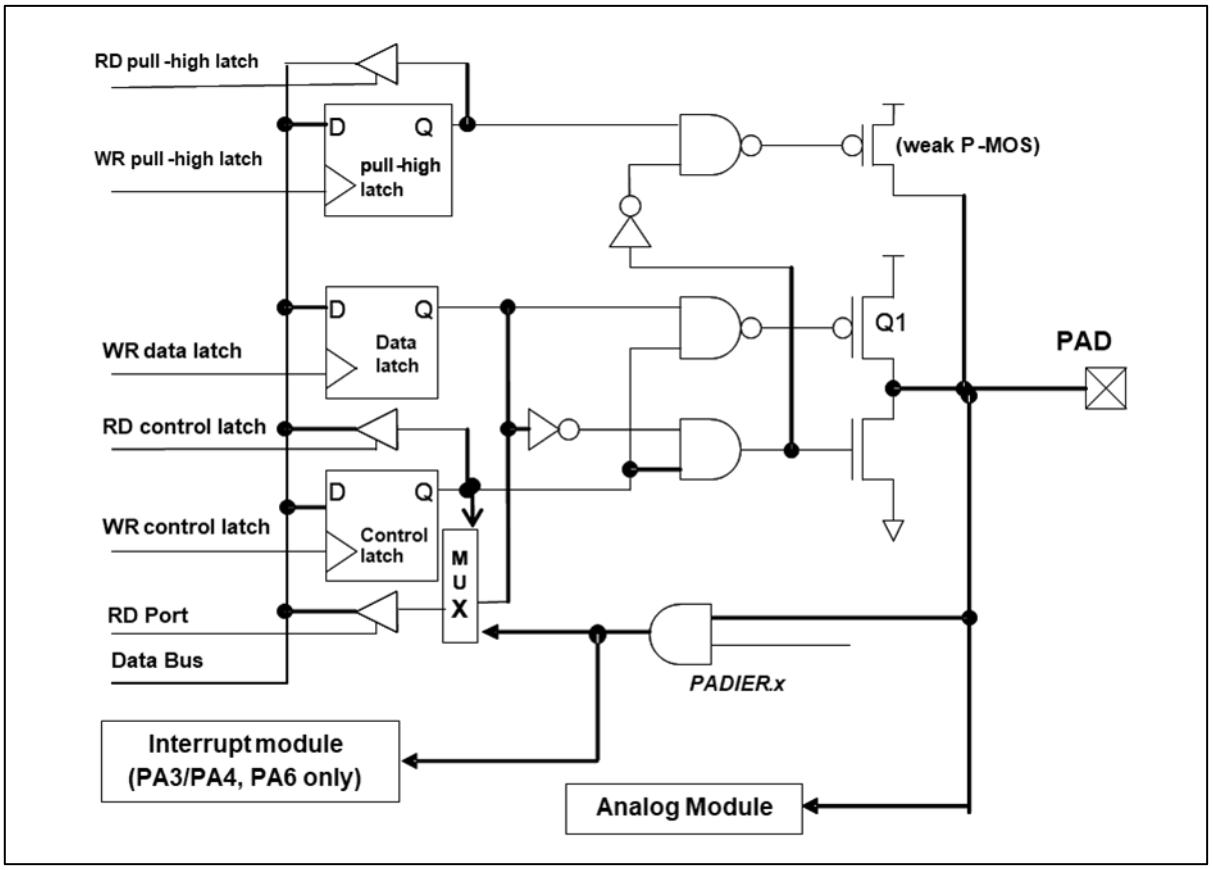


Fig. 14: Hardware diagram of IO buffer

9.2.2. IO Pin Functions

(1) Input / output function:

MF324 all the pins beside port B and PA5 can be independently set into digital input, analog input, output low, output high.

Each IO pin can be independently configured for different state by configuring the data registers (*PA*), control registers (*PAC*), pull-high registers (*PAPH)*, and pull-low registers (*PAPL).*

The corresponding bits in registers *PxDIER* should be set to low to prevent leakage current for those pins are selected to be analog function. When it is set to output low, the pull-high resistor is turned off automatically.

If user wants to read the pin state, please notice that it should be set to input mode before reading the data port. If user reads the data port when it is set to output mode, the reading data comes from data register, NOT from IO pad.

As an example, Table 6 shows the configuration table of bit 0 of port A.

PA.0	*PAC.0*	*PAPH.0*	Description
X	0	0	Input without pull-high resistor
X	0	1	Input with pull-high resistor
0	1	X	Output low without pull-high resistor
1	1	0	Output high without pull-high resistor
1	1	1	Output high with pull-high resistor

Table 6: PA0 Configuration Table

Digital output port B can be independently configured for different state by configuring the data registers (*PBC*).

(2) External interrupt function:

When the IO acts as an external interrupt pin, the corresponding bit of *PxDIER* should be set to high. For example, *PADIER.0* should be set to high when PA0 is used as external interrupt pin.

9.2.3. 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 change with the voltage, temperature and pin voltage. It is not the fixed value.

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

- ◆ Configure IO pin as input mode by *PxC* register.
- ◆ Set corresponding bit to “1” in *PxDIER.*
- ◆ For those IO pins of PA that are not used, *PxDIER* should be set low in order to prevent them from leakage.

(3) PA5 is set to be PRSTB input pin.

- ◆ Configure PA5 as input.
- ◆ Set *CLKMD*.0=1 to enable PA5 as PRSTB input pin.

10. Timer / PWM Counter

10.1. 16-bit Timer (Timer16)

10.1.1. Timer16 Introduction

MF324 provide a 16-bit hardware timer (Timer16) and its block diagram is shown in Fig. 15.

The clock source of Timer16 is selected by register *T16M*[7:5]. Before sending clock to the 16-bit counter (counter16), a pre-scaling logic with divided-by-1/2/4/8/16/32/64 or 128 is selected by *T16M*[4:3] and *OPR2*.2 for wide range counting.

Note: The system clock cannot be slower than the timer16 clock.

T16M[2:0] is used to select the interrupt request. The interrupt request from Timer16 will be triggered by the selected bit which comes from bit[15:8] of this 16-bit counter. Rising edge or falling edge can be optional chosen by register *INTEGS*.4.

The 16-bit counter performs up-counting operation only, the counter initial values can be stored from data memory by issuing the *stt16* instruction and the counting values can be loaded to data memory by issuing the *ldt16* instruction.

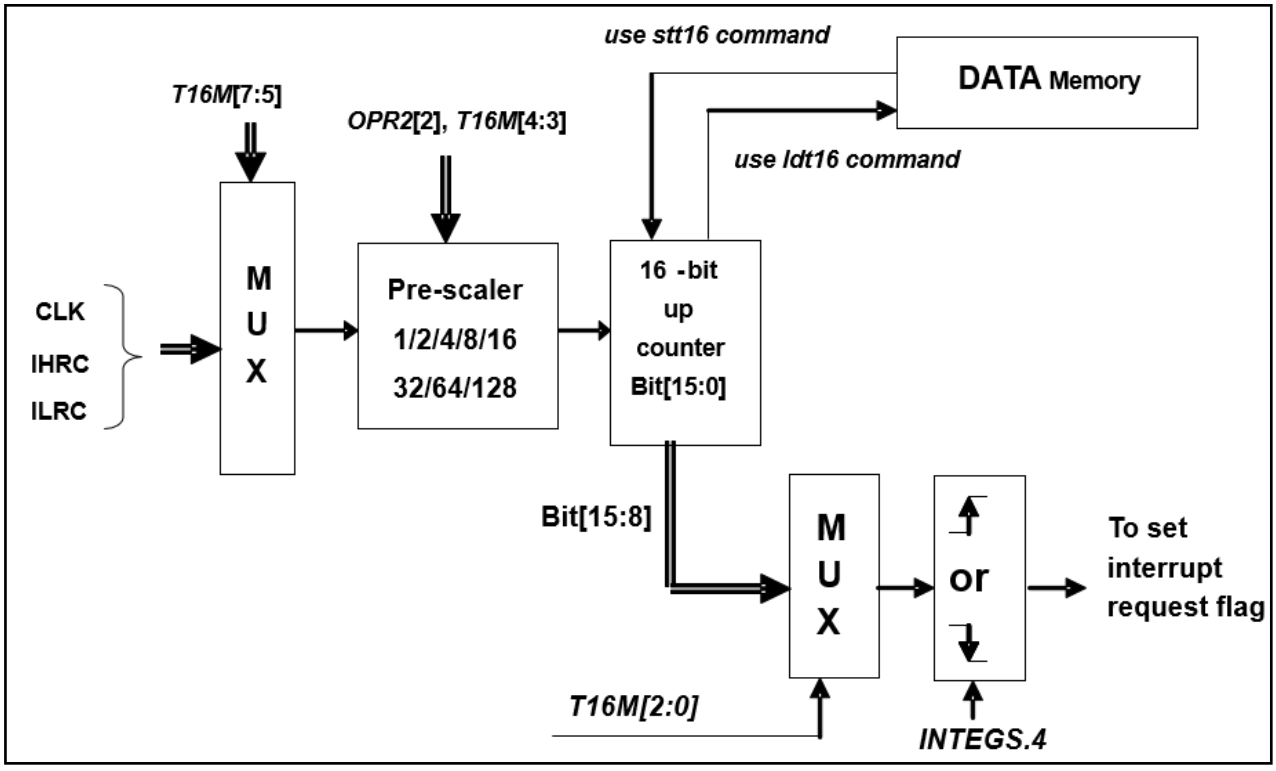


Fig. 15: Hardware diagram of Timer16

There are three parameters to define the Timer16 using; 1st parameter is used to define the clock source of Timer16, 2nd parameter is used to define the pre-scalar and the 3rd one is to define the interrupt source.

```
T16M    IO_RW    0x06
$ 7~5:      STOP, SYSCLK, IHRC, ILRC                          // 1st par.
$ 4~3:      /1, /4, /16, /64(OPR2[2]=0), /2, /8, /32, /128(OPR2[2]=1)   // 2nd par.
$ 2~0:      BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15        // 3rd par.
```

User can choose the proper parameters of *T16M* to meet system requirement, examples as below:

```
$    T16M    SYSCLK, /64, BIT15;
     // choose (SYSCLK/64) as clock source, every 2^16 clock to set INTRQ.2=1
     // if system clock SYSCLK = IHRC / 4 = 4 MHz
     // SYSCLK/64 = 4 MHz/64 = 16 uS, about every 1 S to generate INTRQ.2=1

$    T16M    IHRC, /1, BIT8;
     // choose IHRC as clock source, every 2^9 to generate INTRQ.2=1
     // receiving every 512 times IHRC 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:

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

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

P is the selection of *OPR2*[2] and *T16M* [4:3]; (1, 4, 16, 32, 64, 128)

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

10.1.2. Timer16 Time Out

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

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

10.1.3. Timer16 Mode Register (*T16M*), address = 0x06

Bit	Reset	R/W	Description	
7 - 5	000	R/W	Timer Clock source selection. 000: Timer16 is disabled 001: CLK (system clock) 010: Reserved 011: Reserved 100: IHRC 101: Reserved 110: ILRC 111: Reserved	
4 - 3	00	R/W	*OPR2*.2=0	Timer16 clock pre-divider. 00: /1 01: /4 10: /16 11: /64
			OPR2.2=1	Timer16 clock pre-divider. 00: /2 01: /8 10: /32 11: /128
2 - 0	000	R/W	Interrupt source selection. Interrupt event happens when selected bit goes high. 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	
Where, *OPR2* is an optional register with address 0x47.				

Option 2 Register (*OPR2*), address = 0x47			
Bit	**Reset**	**R/W**	**Description**
7 – 5	-	-	Reserved. Please keep 0.
4	0	WO	PWM Same-Arm protector. 0 / 1: Enable / Disable
3	-	-	Reserved. Please keep 0.
2	**0**	**WO**	**Option for Timer16 clock pre-divider selection. Please see the T16M register.**
1	0	WO	Option for external interrupt 1 pin selection. 0 / 1: X / PA4
0	0	WO	Option for external interrupt 0 pin selection. 0 / 1: PA7 / PA4

10.2. 16-bit Timer (Timer4)

10.2.1. Timer4 Introduction

MF324 provide an another 16-bit hardware timer (Timer4) and its block diagram is shown in Fig. 16.

The clock source of Timer4 is selected by register *TM4C*[7:5]. Before sending clock to the 16-bit counter (counter16), a pre-scaling logic with divided-by-1/2/4/8/16/32/64 or 128 is selected by *TM4C*[4:2].

The 16-bit counter will be clear to zero and generate interrupt request automatically whenever its values reach to that of bound register *TM4BH and TM4BL*. The bound register is used to define the period of Timer4 and need to write *TM4BH* first.

The initial value of 16-bit counter can be set by registers *TM4CTH* and *TM4CTL*. It can also be read back from registers to obtain the value of 16-bit counter. The counter register is used to define the period of Timer4 and need to write *TM4CTH* first.

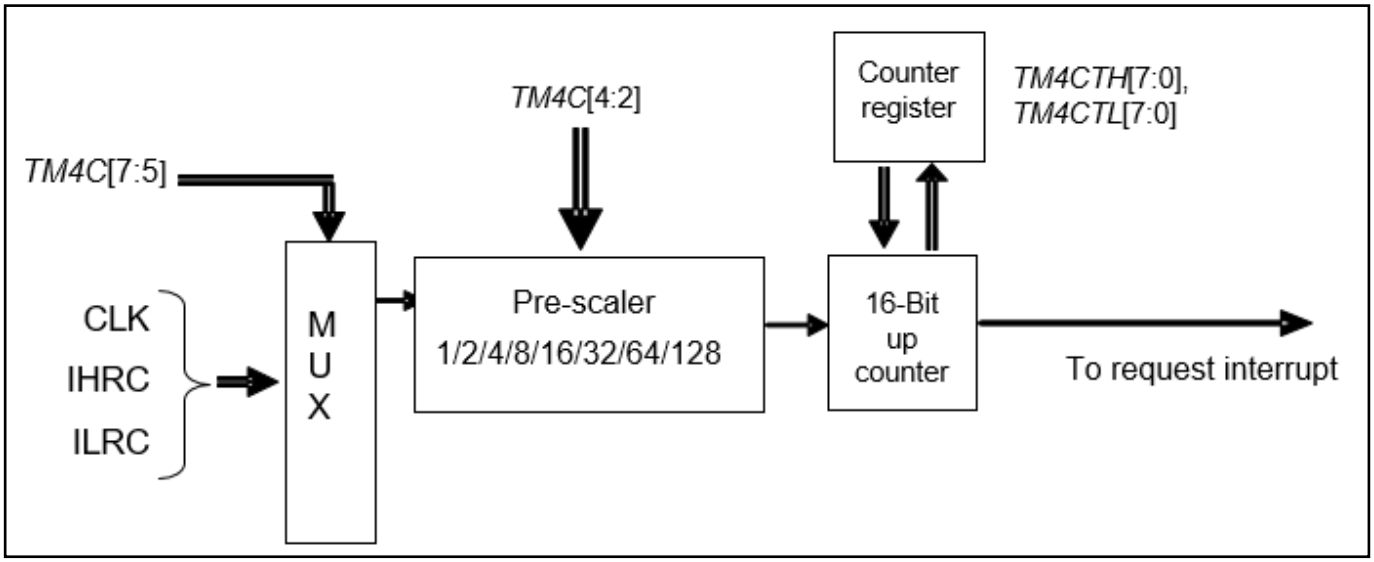


Fig. 16: Timer4 hardware diagram

10.2.2. Timer4 mode Register (*TM4C*), address = 0x22

Bit	Reset	R/W	Description
7 - 5	000	W/O	Timer Clock source selection. 000: Timer4 is disabled. 001: CLK (system clock) 010: Reserved 011: Reserved 100: IHRC 101: Reserved 110: ILRC 111: Reserved
4 - 2	000	W/O	Timer Clock pre-divider selection. 000: /1 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128
1 - 0	-	W/O	Reserved

10.2.3. Timer4 Counter High Byte Register (*TM4CTH*), address = 0x23

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Write: Timer4 initial value of counter high byte register Read: Timer4 counter high byte register

10.2.4. Timer4 Counter Low Byte Register (*TM4CTL*), address = 0x24

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Write: Timer4 initial value of counter low byte register Read: Timer4 counter low byte register

10.2.5. Timer4 Bound High Byte Register (*TM4BH*), address = 0x25

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer4 bound high byte register

10.2.6. Timer4 Bound Low Byte Register (*TM4BL*), address = 0x26

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer4 bound low byte register

10.3. 8-bit Timer (Timer2, Timer3)

Two 8-bit hardware timers (Timer2/TM2, Timer3/TM3) are implemented in the MF324. Timer2 is used as the example to describe its function due to these two 8-bit timers are the same Fig. 21 shows the Timer2 hardware diagram.

Bit[7:4] of register *TM2C* are used to select the clock source of Timer2. The clock frequency divide module is controlled by bit[7:5] of *TM2S* register. And *TM2S*[4:0] provide one scaling module with divided-by-1~32. In conjunction of pre-scaling function and scaling function, the frequency of Timer2 clock (TM2_CLK) can be wide range and flexible. The counter values can be set or read back by *TM2CT* register.

The Timer2 counter performs 8-bit up-counting operation only. The 8-bit counter will be clear to zero and generate interrupt request automatically whenever its values reach to that of bound register *TM2B*, the bound register is used to define the period of Timer2.

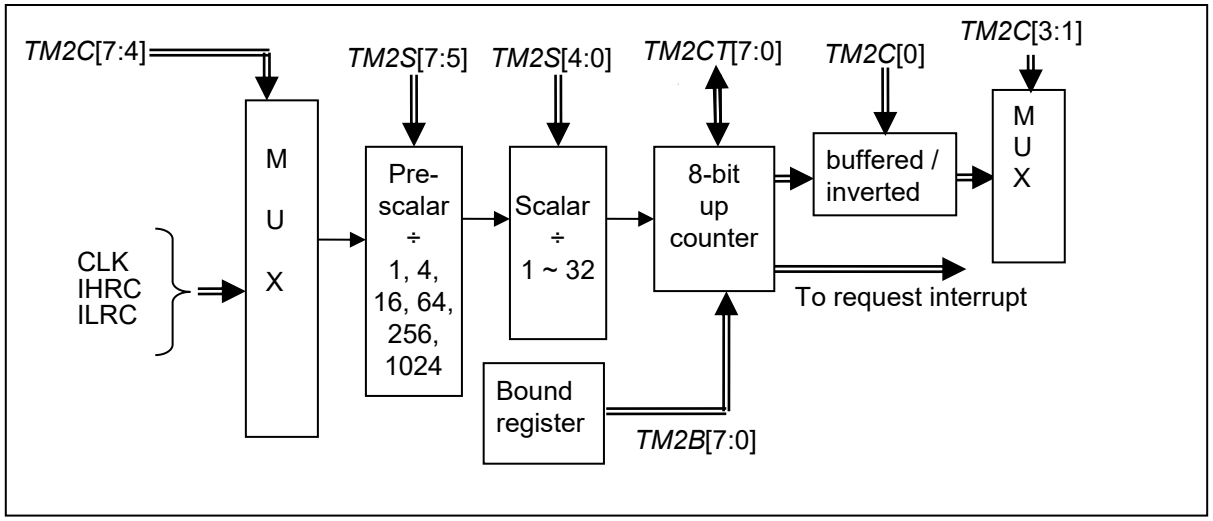


Fig. 17: Timer2 hardware diagram

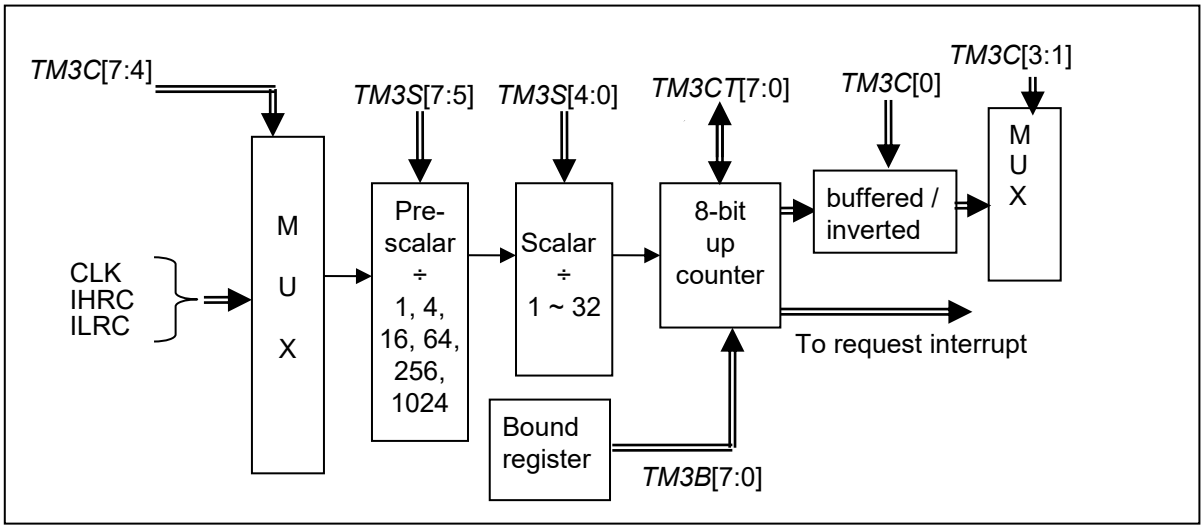


Fig. 18: Timer3 hardware diagram

10.3.1. Timer2 Control Register (*TM2C*), address = 0x27

Bit	Reset	R/W	Description
7 - 4	0000	WO	Timer2 clock selection. 0000 : disable 0001 : system clock 0010 : IHRC 0011 : Reserved 0100 : ILRC 0101 - 1111 : Reserved
3 - 0	-	-	Reserved. Please keep 0.

10.3.2. Timer2 Counter Register (*TM2CT*), address = 0x28

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Write: Timer2 initial value of counter register. Read: Timer2 counter register.

10.3.3. Timer2 Scalar Register (*TM2S*), address = 0x29

Bit	Reset	R/W	Description
7 - 5	000	WO	Timer2 clock pre-scalar 000 : ÷ 1 001 : ÷ 4 010 : ÷ 16 011 : ÷ 64 100 : ÷ 256 101 : ÷ 1024 11X : Reserved
4 - 0	00000	WO	Timer2 clock scalar

10.3.4. Timer2 Bound Register (*TM2B*), address = 0x2A

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

10.3.5. Timer3 Control Register (*TM3C*), address = 0x2B

Bit	Reset	R/W	Description
7 - 4	0000	R/W	Timer3 clock selection. 0000 : disable 0001 : system clock 0010 : IHRC 0011 : Reserved 0100 : ILRC 0101 - 1111 : Reserved
3 - 0	-	-	Reserved. Please keep 0.

10.3.6. Timer3 Counter Register (*TM3CT*), address = 0x2C

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Write : Timer3 initial value of counter register Read : Timer3 counter register

10.3.7. Timer3 Scalar Register (*TM3S*), address = 0x2D

Bit	Reset	R/W	Description
7 - 5	000	WO	Timer3 clock pre-scalar. 000 : ÷ 1 001 : ÷ 4 010 : ÷ 16 011 : ÷ 64 100 : ÷ 256 101 : ÷ 1024 11X : Reserved
4 - 0	00000	WO	Timer3 clock scalar

10.3.8. Timer3 Bound Register (*TM3B*), address = 0x2E

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

10.4. 12-bit PWM Generation

Three 12-bit hardware PWM generators with programmable period and duty with dead zone control is built inside the MF324. The PWM is configured as complementary(center) mode. The clock source can be select as clock source, System Clock, IHRC, IHRC*2 by *PWMGC*[3:2]. Includes a pre-scaling logic with division by 1, 2, 4, 8 is selected by register *PWMGCS*[1:0]. Built-in basic same-arm protection hardware.

Additionally, it includes a flexible control combination to support a 3-phase brushless DC motor.

10.4.1. Hardware PWM with dead zone

The PWM generator with dead zone control is built inside the MF324. The following figure 19 shows its hardware diagram. The period of PWM waveform as defined in the PWM upper bound registers *PWMUpBH* and *PWMUpBL*. The duty cycle of the PWM waveform as defined in register *PWM0DtH* and *PWM0DtL,* and the dead zone as defined in register *PWMDdZVF* and *PWMDdZVR*.

The PWM output signal can be switched to high/low/PWM+/PWM- by register *PWMGC1* and *PWMGC2*. When the IO PBX outputs PWM signal, it can be switched by register *PWMPBC1* and *PWMPBC2*.

The PWM can trigger ADC by register *PWMADCH* and *PWMADCL*.

Note: PWMGC1 needs to be set first, while setting PWMGC2 will make the register effective.

In the MF324, PWM have a special circuit for emergency stop. Refer to *MISC*[7:6].

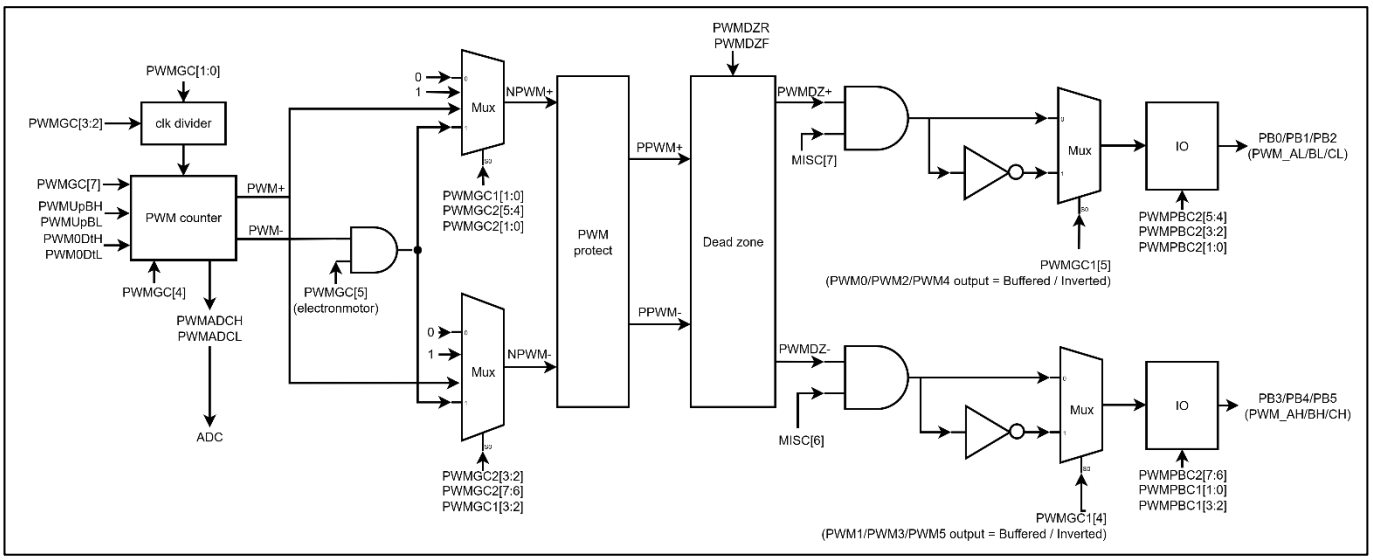


Fig. 19: 12-bit PWM Generator

10.4.2. Timing Diagram

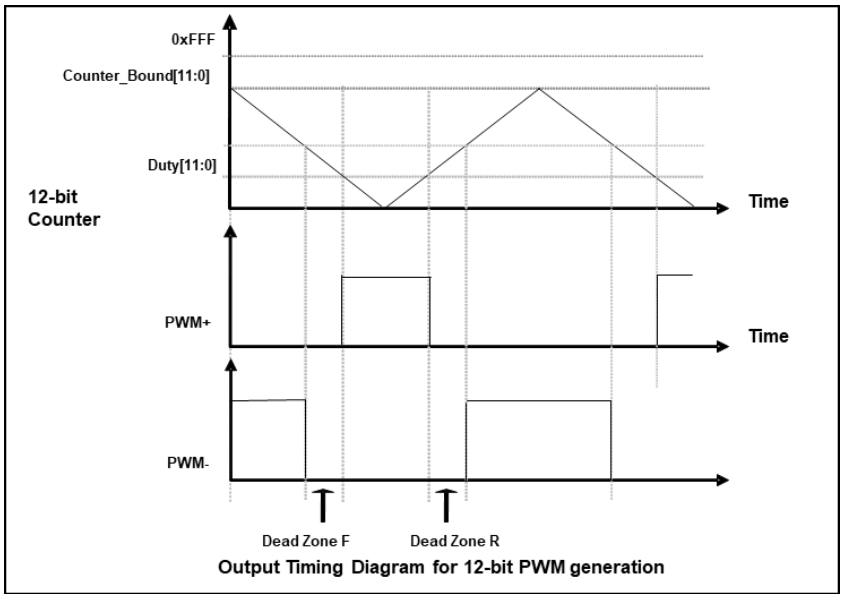


Fig. 20: Output Timing Diagram of Center mode of 12-bit PWM Generator

10.4.3. Equations for 12-bit PWM Generator

If F_{IHRC} is the frequency of IHRC oscillator and IHRC is the chosen clock source for 12-bit PWM generator, the PWM frequency and duty cycle in time will be:

Frequency of PWM Output = F_{IHRC} ÷ [P × CB]

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

Where, *PWMGC*[1:0] = P ; pre-scalar

12-bit Duty_Bound[11:0] = {pwmdth[7:0],pwmdtl[7:4]} = **DB**; duty bound

11-bit Counter_Bound[11:1] X2 = {pwmcubh[7:0], pwmcubl[7:5]} X2 = **CB**; counter bound

10.4.4. 12-bit PWM Related Registers

10.4.4.1. PWM PB control 1 Register (*PWMPBC1*), address = 0x18

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved
3 - 2	10	R/W	PB5 output PWM1/ PWM3/ PWM5 type. 00: PWM1 01: PWM3 10: PWM5 11: Reserved
1 - 0	01	R/W	PB4 output PWM1/ PWM3/ PWM5 type. 00: PWM1 01: PWM3 10: PWM5 11: Reserved

10.4.4.2. PWM PB control 2 Register (*PWMPBC2*), address = 0x19

Bit	Reset	R/W	Description
7 - 6	00	R/W	PB3 output PWM1/ PWM3/ PWM5 type. 00: PWM1 01: PWM3 10: PWM5 11: Reserved
5 - 4	10	R/W	PB2 output PWM0/ PWM2/ PWM4 type. 00: PWM0 01: PWM2 10: PWM4 11: Reserved
3 - 2	01	R/W	PB1 output PWM0/ PWM2/ PWM4 type. 00: PWM0 01: PWM2 10: PWM4 11: Reserved
1 - 0	00	R/W	PB0 output PWM0/ PWM2/ PWM4 type. 00: PWM0 01: PWM2 10: PWM4 11: Reserved

10.4.4.3. PWM Generator control 1 Register (*PWMGC1*), address = 0x1A

Bit	Reset	R/W	Description
7 - 6	-	-	Reserved
5	0	R/W	PWM0 / PWM2 / PWM4 output Buffered / Inverted. 0: Buffered 1: Inverted
4	0	R/W	PWM1 / PWM3 / PWM5 output Buffered / Inverted. 0: Buffered 1: Inverted
3 - 2	00	R/W	PWM5 output type. 00: force low 01: PWM+ 10: PWM- 11: force high
1 - 0	00	R/W	PWM4 output type. 00: force low 01: PWM+ 10: PWM- 11: force high

10.4.4.4. PWM Generator control 2 Register (PWMGC2), address = 0x1B

Bit	Reset	R/W	Description
7 - 6	00	R/W	PWM3 output type. 00: force low 01: PWM+ 10: PWM- 11: force high
5 - 4	00	R/W	PWM2 output type. 00: force low 01: PWM+ 10: PWM- 11: force high
3 - 2	00	R/W	PWM1 output type. 00: force low 01: PWM+ 10: PWM- 11: force high
1 - 0	00	R/W	PWM0 output type. 00: force low 01: PWM+ 10: PWM- 11: force high

10.4.4.5. PWM Generator Control Register (PWMGC), address = 0x1C

Bit	Reset	R/W	Description
7	0	R/W	Enable PWM generator. 0 / 1: disable / enable
6	0	RO	Output of PWMG0(PWM+)
5	0	R/W	PWM- disable electromotor mode. 0: Disable 1: Enable
4	0	WO	PWM counter reset Writing “1” to clear PWM counter and this bit will be self-clear to 0 after counter reset
3 - 2	00	R/W	PWM generator clock source 00: system clock 01: IHRC 10: IHRC*2 11: Reserved
1 - 0	00	R/W	PWMGx clock pre-scalar 00: ÷ 1 01: ÷ 2 10: ÷ 4 11: ÷ 8

10.4.4.6. Miscellaneous Setting Register (MISC), address = 0x34

Bit	Reset	R/W	Description
7	0	WO	Enable emergency stop of output of PWM0/PWM2/PWM4. 0 / 1: disable / enable
6	0	WO	Enable emergency stop of output of PWM1/PWM3/PWM5. 0 / 1: disable / enable
5 - 3	-	-	Reserved. Please keep 0.
2			Reserved. Please keep 0.
1 - 0	10	WO	Watchdog time out period. 01: 4K ILRC 10: 16K ILRC

10.4.4.7. PWM Counter Upper Bound High Register (PWMUPBH), address = 0x3D

Bit	Reset	R/W	Description
7 - 0	8’h00	WO	Bit[11:4] of PWM counter upper bound

10.4.4.8. PWM Counter Upper Bound Low Register (PWMUPBL), address = 0x3E

Bit	Reset	R/W	Description
7 - 5	3’h0	WO	Bit[3:1] of PWM counter upper bound
4 - 0	-	-	Reserved

10.4.4.9. PWMG0 Duty Value High Register (PWM0DTH), address = 0x3F

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Duty values bit[11:4] of PWM0 generator

10.4.4.10. PWMG0 Duty Value Low Register (PWM0DTL), address = 0x40

Bit	Reset	R/W	Description
7 - 4	4'h0	WO	Duty values bit[3:0] of PWM0 generator
3 - 0	-	-	Reserved

10.4.4.11. PWMG ADC trigger High Register (PWMADCH), address = 0x41

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Duty values bit[11:4] of PWM2 generator

10.4.4.12. PWMG ADC trigger Low Register (PWMADCL), address = 0x42

Bit	Reset	R/W	Description
7 - 4	4'h0	WO	Duty values bit[3:0] of PWM2 generator
3 - 0	-	-	Reserved

10.4.4.13. PWM Front Dead-zone Register (PWMDDZVF), address = 0x43

Bit	Reset	R/W	Description
7 - 1	-	WO	Front dead-zone values bit[7:1] of PWM generator
0	-	-	Reserved

10.4.4.14. PWM Rear Dead-zone Register (PWMDDZVR), address = 0x44

Bit	Reset	R/W	Description
7 - 1	-	WO	Rear dead-zone values bit[7:1] of PWM generator
0	-	-	Reserved

11. Special Function

11.1. General Purpose Comparator

11.1.1. General Purpose Comparator Hardware Diagram

One general purpose comparator is built inside the MF324. It can compare signals between two pins or with either internal reference voltage $V_{internal\ R}$ or external PA5 voltage. The two signals to be compared, one will be the plus input of comparator and the other one is the minus input of comparator. For general purpose comparator, the plus input pin is selected by register *GPCC*.0, and the minus input pin is selected by *GPCC*[3:1].

The comparator result can be:

(1) Read back by *GPCC*.6.

(2) Inversed the polarity by *GPCC*.4.

(3) Interrupt edge select by INTEGS[6:5].

The comparator is disabled after power-on reset and can be enabled by setting *GPCC*.7=1.

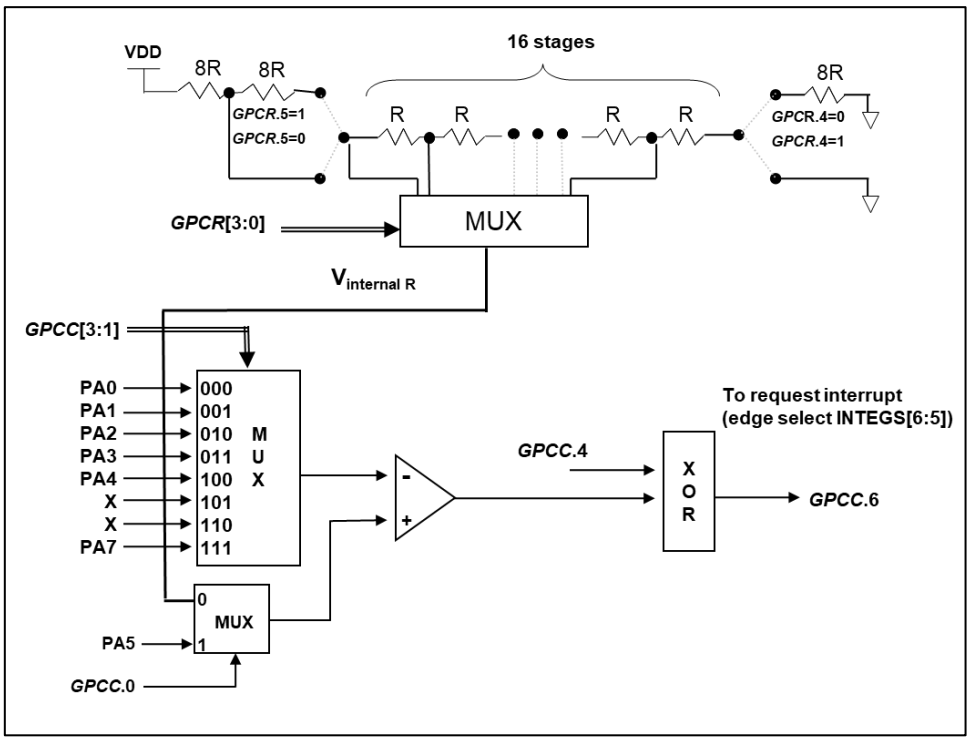


Fig. 21: Hardware diagram of comparator 1

11.1.2. General Purpose Comparator Control Register (*GPCC*), address = 0x15

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 input pins to be digital disable to prevent IO leakage.
6	-	RO	Comparator result 0: plus input < minus input 1: plus input > minus input
5	-	-	Reserved
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 general-purpose comparator 000 : PA0 001 : PA1 010 : PA2 011 : PA3 100 : PA4 101 : Reserved 110 : Reserved 111 : PA7
0	0	R/W	Selection the plus input (+) of comparator 1 0 : $V_{internal\ R}$ 1 : PA5

11.1.3. General Purpose Comparator Result Register (*GPCR*), address = 0x16

Bit	Reset	R/W	Description
7	0	WO	GPC compare result output to PA6 0: Disable 1: Enable
6	-	-	Reserved
5	0	WO	Selection of high range of general-purpose comparator
4	0	WO	Selection of low range of general-purpose comparator
3 - 0	0000	WO	Selection the voltage level of general-purpose comparator 0000 (lowest) ~ 1111 (highest)

11.1.4. Analog Inputs

A simplified circuit for the analog inputs is shown in the Fig. 22. All the analog input pins for general purpose comparator are shared function with a digital input which has reverse biased ESD protection diodes to VDD and GND, therefore, the analog input signal must be between VDD and GND.

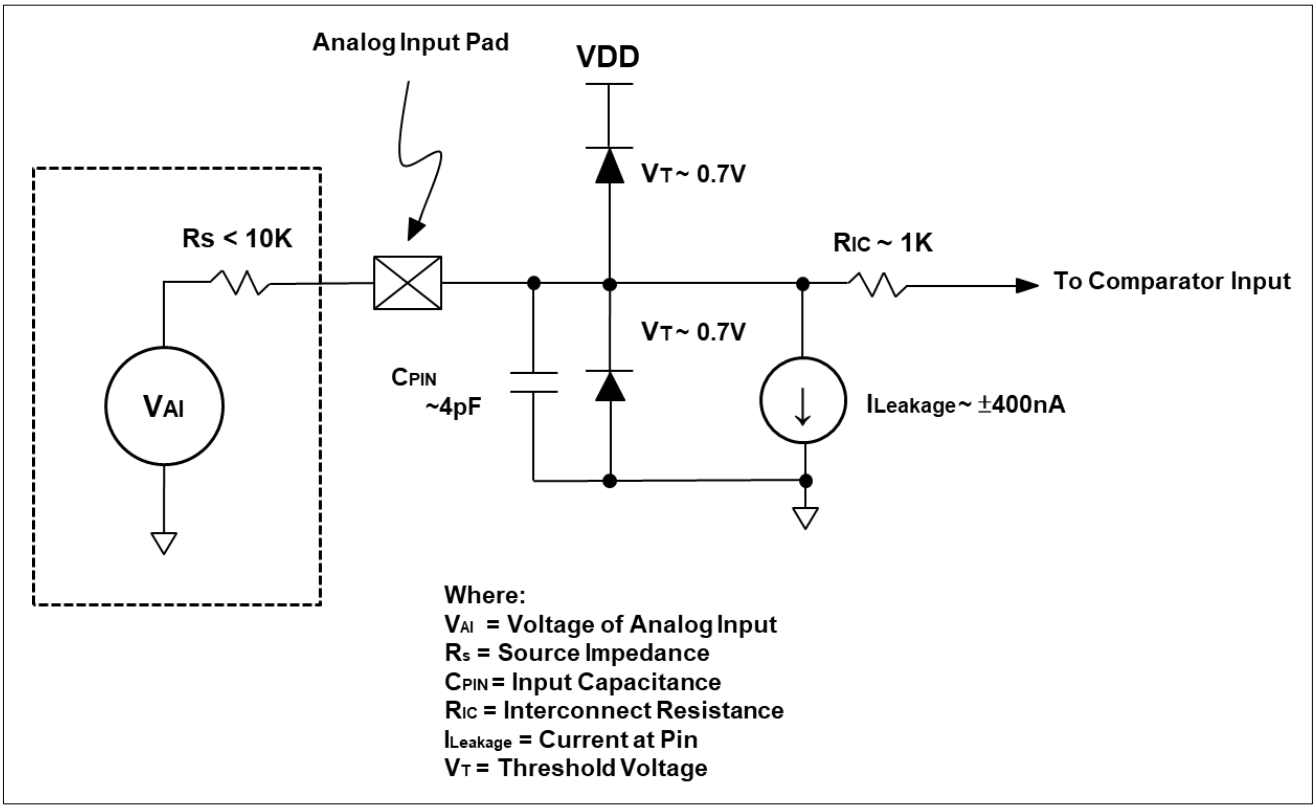


Fig.22: Analog Input Model of General-Purpose Comparator

11.1.5. Internal Reference Voltage ($V_{internal\ R}$)

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

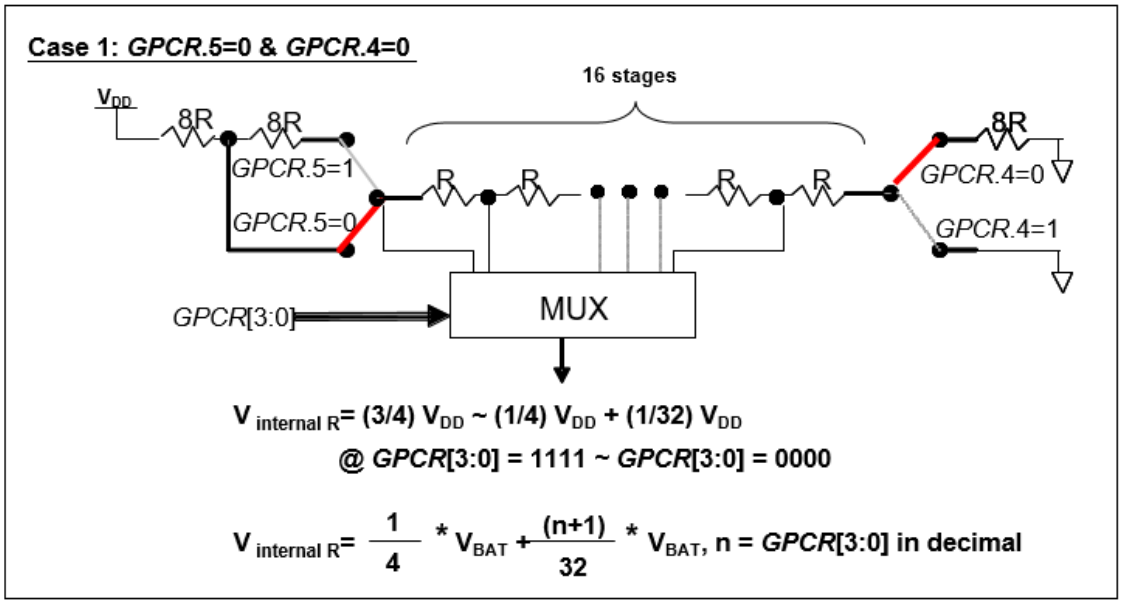


Fig. 23: $V_{internal\ R}$ hardware connection if *GPCR*.5=0 and *GPCR*.4=0

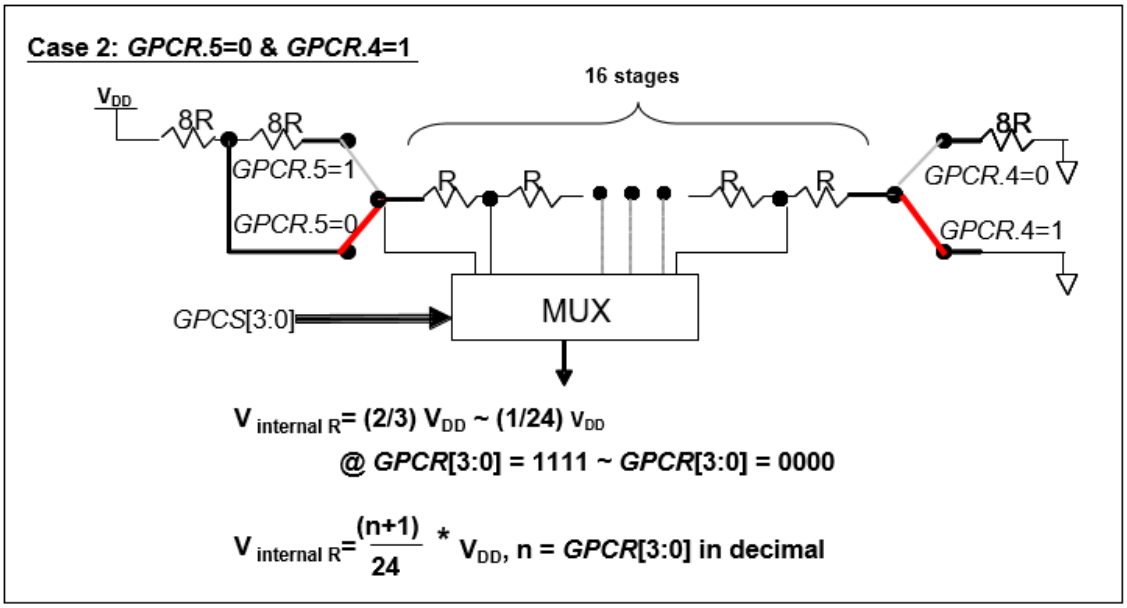


Fig. 24: $V_{internal\ R}$ hardware connection if *GPCR*.5=0 and *GPCR*.4=1

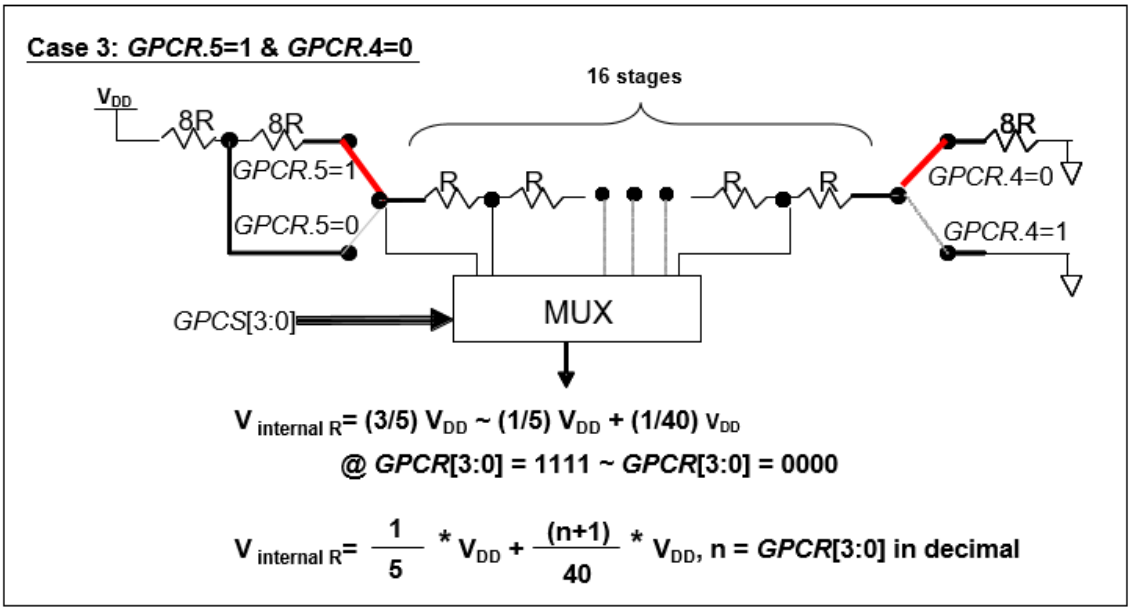


Fig. 25: $V_{internal\ R}$ hardware connection if *GPCR*.5=1 and *GPCR*.4=0

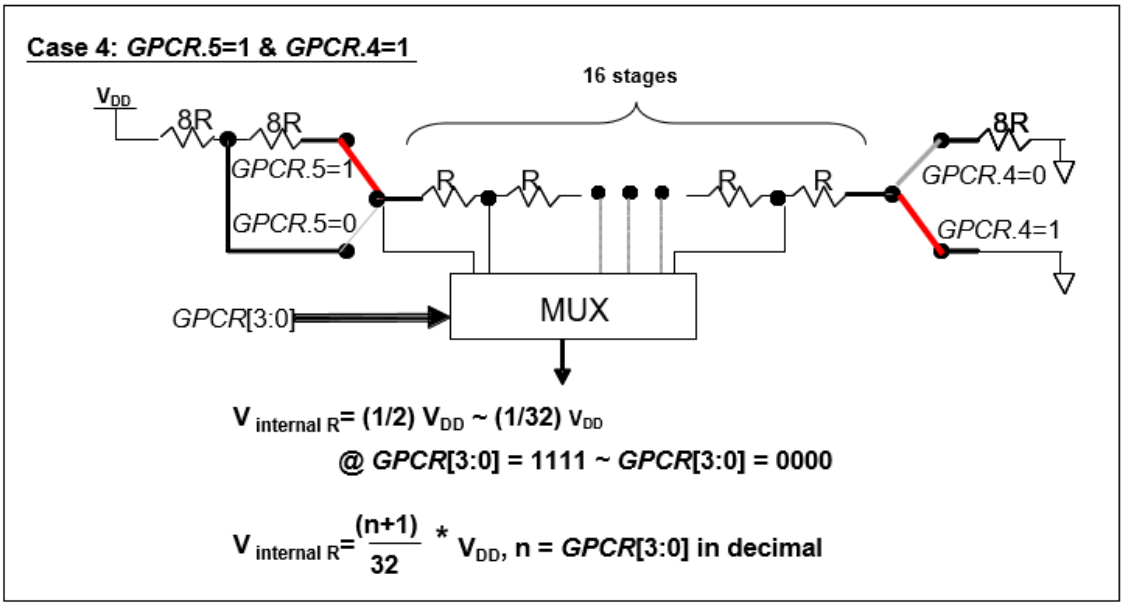


Fig. 26: $V_{internal\ R}$ hardware connection if *GPCR*.5=1 and *GPCR*.4=1

11.1.6. Using the Comparator

Case 1:

Choosing PA0 as minus input and $V_{internal\ R}$ with (18/32)*VDD voltage level as plus input. $V_{internal\ R}$ is configured as the above Figure "*GPCR*[5:4] = 2b'00" and *GPCR*[3:0] = 4b'1001 (n=9) to have $V_{internal\ R}$ = (1/4)*VDD + [(9+1)/32]*VDD = [(9+9)/32]*VDD = (18/32)*VDD.

```
GPCC    = 0b1_0_0_0_000_0;    // enable comp, input - : PA0, input + : Vinternal R,
GPCR    = 0b00_0_0_1001;      // Vinternal R = VDD*(18/32)
PADIER  = 0bxxxx_xxx_0;       // disable PA0 digital input to prevent leakage current
```

Case 2:

Choosing $V_{internal\ R}$ as plus input with (22/40)*VDD voltage level and PA2 as minus input, the comparator result will be inversed and without output to PA2. $V_{internal\ R}$ is configured as the above Figure "*GPCR*[5:4] = 2b'10" and *GPCR*[3:0] = 4b'1101 (n=13) to have $V_{internal\ R}$ = (1/5)*VDD + [(13+1)/40]*VDD = [(13+9)/40]*VDD = (22/40)*VDD.

```
GPCC    = 0b1_0_0_1_010_0;    // enable comp, Inverse output, input - : PA2, input + : Vinternal R,
GPCR    = 0b00_1_0_1101;      // Vinternal R = VDD*(22/32)
PADIER  = 0bxxx_x_0_xx;       // disable PA2 digital input to prevent leakage current
```

11.2. Analog-to-Digital Conversion (ADC) module

The MF324 provides one 12-bit resolution analog-to-digital conversion module with 8 external channels and 3 internal channels respectively for Bandgap reference voltage, BEMF and one-quarter of the VDD.

For the conversion process, analog signal will be sampled and held first, then sending into the converter to generate the result via successive approximation. The analog reference high voltage of ADC is the positive supply voltage (VDD) and the reference low is always the GND.

Higher than 1uF capacitor is recommended to be placed between VDD and GND to have better AD conversion result.

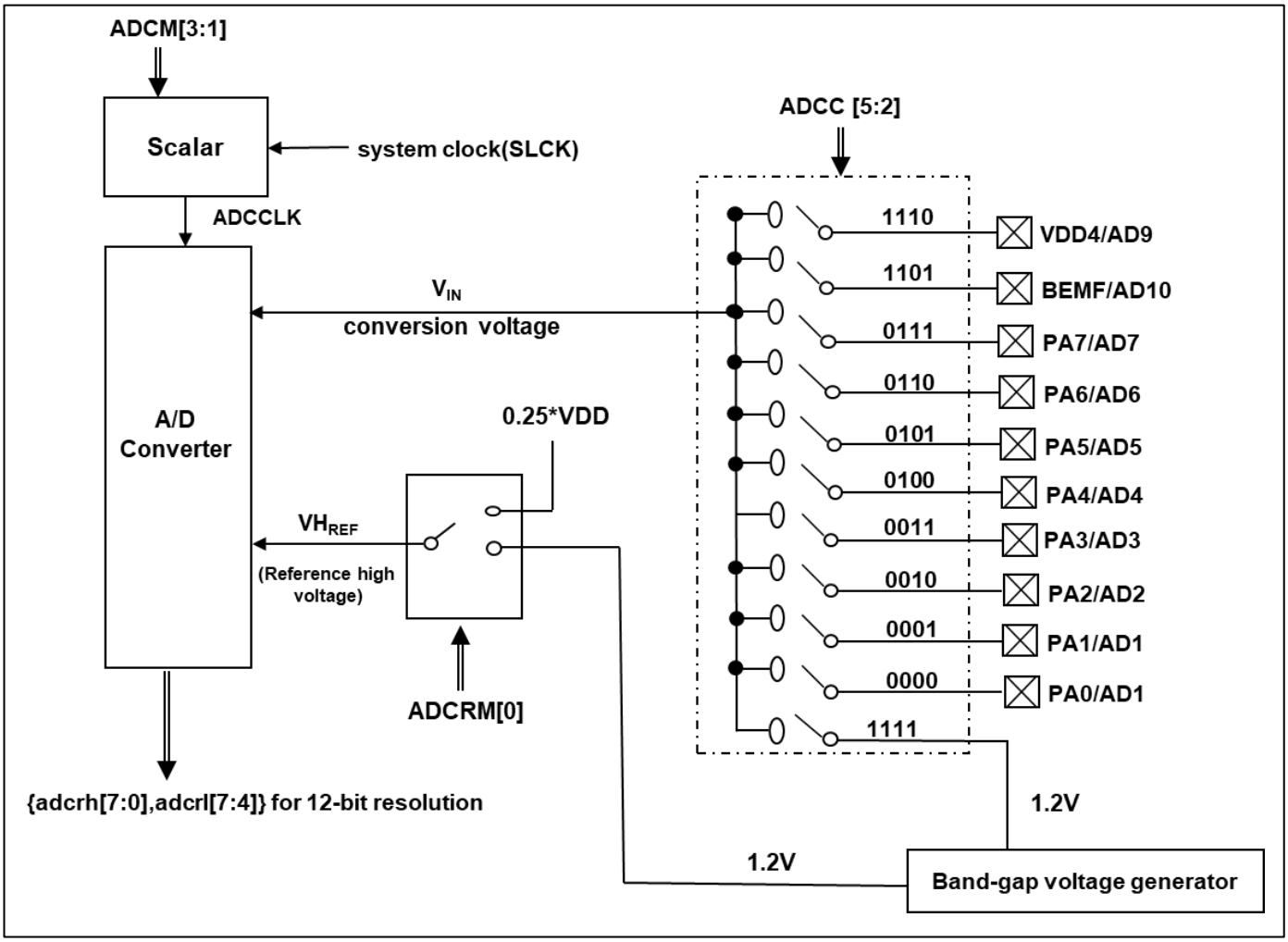


Fig. 27: ADC Block Diagram

11.2.1. The input requirement for ADC conversion

For the ADC conversion, to meet its specified accuracy, the charge holding capacitor (C_{HLD}) must be allowed to fully charge to the voltage reference high level (VDD) and discharge to the voltage reference low level (GND). The analog input model is shown as Fig.32, the signal driving source impedance (RS) and the internal sampling switch impedance (R_{ON}) will affect the required time to charge the capacitor C_{HLD} 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.

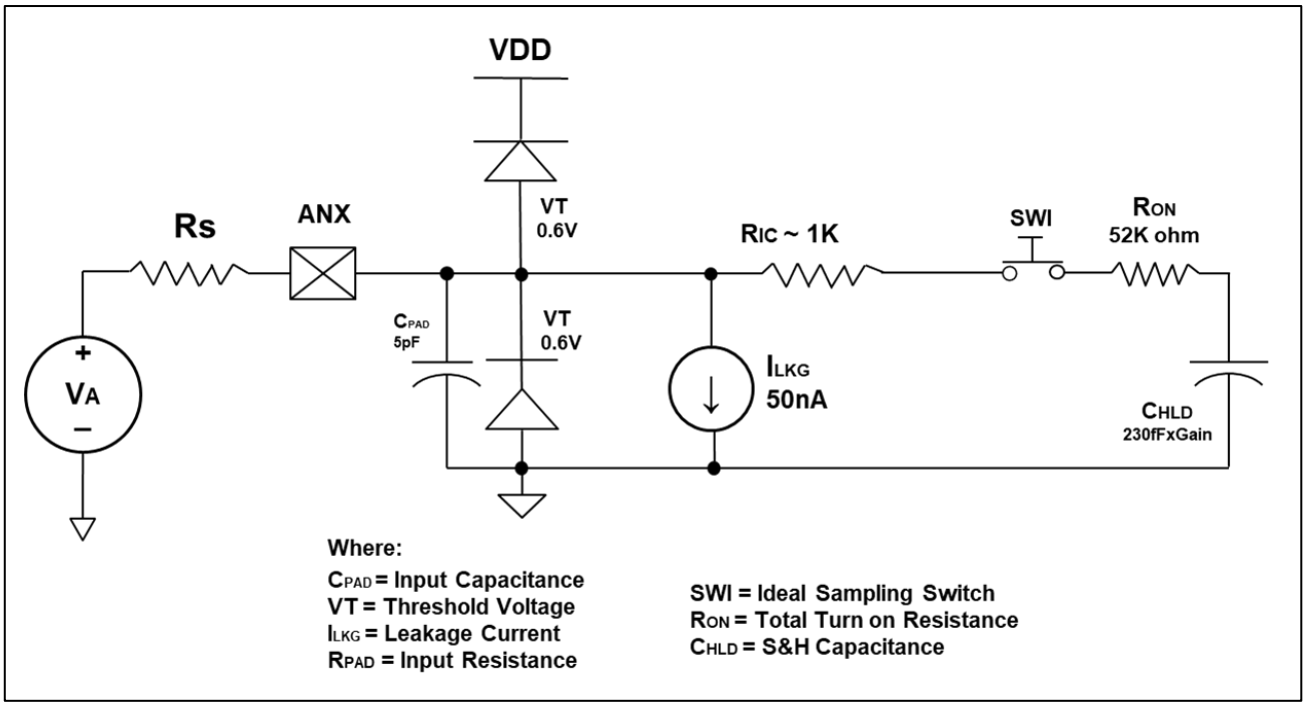


Fig. 28: Analog Input Model

Before starting the ADC conversion, the minimum signal acquisition time should be met for the selected analog input signal. The signal acquisition time (T_{ACQ}) of ADC in MF324 series is fixed to one clock period of ADCLK, the selection of ADCLK must be met the minimum signal acquisition time.

11.2.2. 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 0.2us.

11.2.3. AD conversion

The process of ADC conversion starts from setting START/DONE (ADCC.6) to high, and the flag is cleared automatically when starting the ADC conversion. Then converting analog signal bit by bit and finally setting START/DONE high to indicate the completion of ADC conversion. If ADCLK is selected, TADCLK is the period of ADCLK and the ADC conversion time can be calculated as follows:

- 12-bit resolution: ADC conversion time = 16 T_{ADCLK}

11.2.4. Configure the analog pins

There are 11 analog signals can be selected for ADC conversion, 8 analog input signals come from external pins and one is from internal Bandgap reference voltage, one is from BEMF and another one from one-quarter of the VDD. To avoid leakage current at the digital circuit, external pins defined for analog input should disable the digital input function (set the corresponding bit of *PADIER* 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 and pull-low resistor
(3) set the corresponding pin to analog input by port A/B digital input disable register (*PADIER*)

The following steps are recommended to do the ADC conversion procedure:

(1) Configure the ADC module:
- Configure the voltage reference high by ADCM register, recommend it is to operate at Bandgap
- Select the ADC input channel by *ADCC* register
- Configure the ADC conversion clock by *ADCM* register
- Configure the pin as analog input by *PADIER* register
- Enable the ADC module by *ADCC* register

(2) Configure interrupt for ADC: (if desired)
- Clear the ADC interrupt request flag in bit 3 of INTRQ register
- Enable the ADC interrupt request in bit 3 of INTEN register
- Enable global interrupt by issuing engine command

(3) Start ADC conversion:
- Set ADC process control bit in the ADCC register to start the conversion (set1 ADCC.6)

(4) Wait for the completion flag of ADC conversion, by either:
- Waiting for the completion flag by using command “wait1 ADCC.6”; or Waiting for the ADC interrupt

(5) Read the ADC result registers:
- Read *ADCRH* and *ADCRL* the result registers

(6) For next conversion, go to step 1 or step 2 as required

11.2.5. Using the ADC

The following example shows how to use ADC with PA3~PA4.

First, defining the selected pins:

```
PAC     =    0B_XXX0_0XXX;        //    PA3 ~ PA4 as Input
PAPH    =    0B_XXX0_0XXX;        //    PA3 ~ PA4 without pull-high
PADIER      =    0B_XXX0_0XXX;   //    PA3 ~ PA4 digital input is disabled
```

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

```
$   ADCC    Enable, PA3;         //    set PA3 as ADC input
$   ADCC    Enable, PA4;         //    set PA4 as ADC input
```

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

```
$   ADCM    /16;                 //    recommend /16 @System Clock=8MHz
$   ADCM    /8;                  //    recommend /8 @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;    //    or Data    =    (ADCRH << 8) | ADCRL;
```

The ADC can be disabled by using the following method:

```
$   ADCC    Disable;
```

or

```
ADCC        =    0;
```

11.2.6. ADC Related Registers

11.2.6.1. ADC Result High Register (ADCRH), address = 0x1E

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.

11.2.6.2. ADC Result Low Register (ADCRL), address = 0x1F

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

11.2.6.3. ADC Control Register (*ADCC*), address = 0x20

Bit	Reset	R/W	Description
7	0	R/W	Enable ADC function. 0/1: Disable/Enable
6	-	R/W	ADC process control bit. Write “1” to start ADC conversion, and the flag is cleared automatically when starting the ADC conversion. Read “1” to indicate the completion of ADC conversion and “0” is in progressing.
5 - 2	0000	R/W	Channel selector. These four bits are used to select input signal for ADC conversion: 0000: PA0/AD0 0001: PA1/AD1 0010: PA2/AD2 0011: PA3/AD3 0100: PA4/AD4 0101: PA5/AD5 0110: PA6/AD6 0111: PA7/AD7 1000-1100: Reserved 1101: BEMF(COM is selected by ZCPC[3]) 1110: VDD/4 1111: Bandgap reference voltage
1 - 0	-	-	Reserved. Please keep 0

11.2.6.4. ADC Mode Register (ADCM), address = 0x21

Bit	Reset	R/W	Description
7	0	R/W	ADC Trigger mode selection. 0/1: Command / PWM Trigger mode
6	0	R/W	Center mode PWM Trigger ADC. 0/1: Up / Down count
5 - 4	-	-	Reserved
3 - 1	000	R/W	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	0	WO	ADC Reference Voltage Select. 0: VDD 1: Bandgap Voltage

11.3. PWM generator Trigger ADC Conversion

The PWM generator also supports triggering ADC conversion. It can trigger ADC conversion by setting the trigger enable bit of register (*ADCM*.7) and ADC enable bit of ADCC register(*ADCC*.7). When the PWM counter count matches the set value of the *PWMADCH* and *PWMADCL* registers, it will issue a control signal to trigger the ADC start, as shown in Fig. 29. The trigger enable bit will be cleared automatically after the ADC conversion is completed.

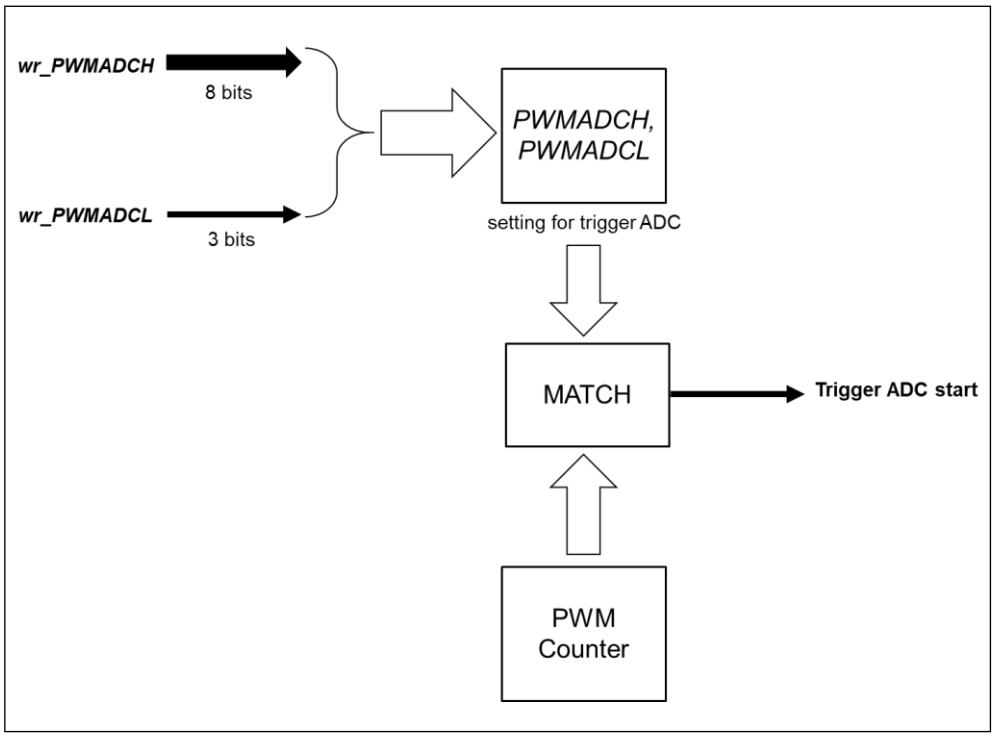


Fig. 29: Block Diagram of PWM generator trigger ADC conversion

11.3.1. PWM Trigger ADC - PWM Counter High Register (*PWMADCH*), address = 0x41

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Trigger ADC - PWM counter values bit[11:4] setting

11.3.2. PWM Trigger ADC - PWM Counter Low Register (*PWMADCL*), address = 0x42

Bit	Reset	R/W	Description
7 - 5	3'h0	WO	Trigger ADC - PWM counter values bit[3:1] setting
4 - 0	-	-	Reserved

11.4. Input Pulse Capture

The feature of input Pulse Capture is useful in applications which requiring frequency and pulse measurement. Fig. 30 shows the hardware diagram of input Pulse Capture in MF324, the time base of Pulse Capture module can be system clock or IHRCX2, the input signals for measurement can be PA3, PA4 or PA6.

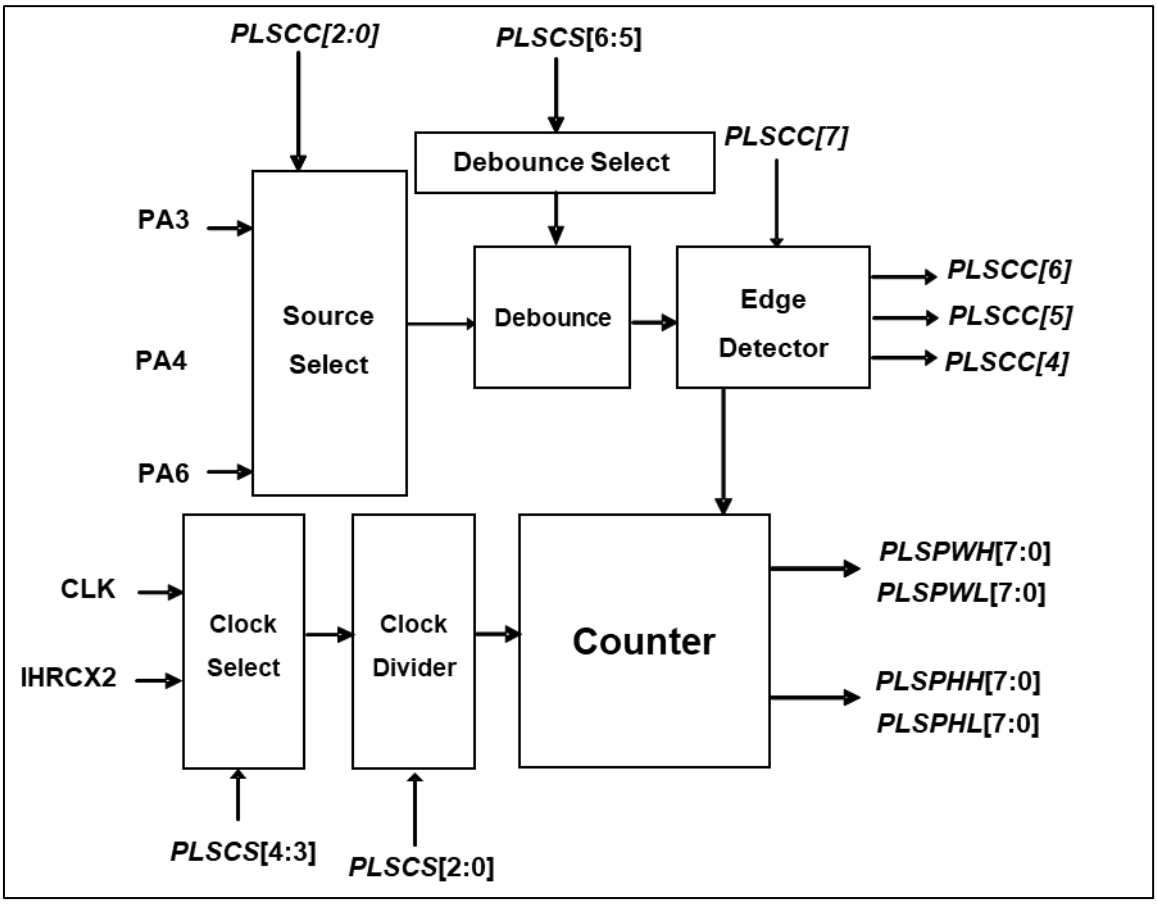


Fig. 30: Hardware Diagram of Input Pulse Capture

11.4.1. Pulse Capture Control Register (*PLSCC*), address = 0x32

Bit	Reset	R/W	Description
7	0	WO	0: Stop to do Pulse Capture 1: Start to do Pulse Capture and clear Pulse Capture Overflow or Done Flag
6		RO	Pulse Capture Overflow Flag. clears this bit automatically after finishing the Pulse Capture operation.
5		RO	Pulse Capture Done Flag
4		RO	Full Cycle Period Capture Done Flag
3	1	R/W	0 / 1: Capture Low / High Pulse
2 - 0	000	R/W	Sources for Pulse Capture 000: PA3 001: PA4 010: PA6 011-111 : Reserved

11.4.2. Pulse Capture Scalar Register (*PLSCS*), address = 0x33

Bit	Reset	R/W	Description
7	-	-	Reserved, please keep 0.
6 - 5	00	R/W	Debounce of Pulse Capture input source. 00 : None 01 : 1 Pulse Capture clock 10 : 2 Pulse Capture clocks 11 : 3 Pulse Capture clocks
4 - 3	00	R/W	Pulse Capture clock source 00: system CLK 01: IHRC*2 1X: Reserved
2 - 0	000	R/W	Clock divider of Pulse Capture 000 : /1 001 : /2 010 : /4 011 : /8 100 : /16 101 : /32 110 : /64 111 : /128

11.4.3. Pulse Capture Pulse Width High Register (*PLSPWH*), address = 0x39

Bit	Reset	R/W	Description
7 - 0	0xFF	W/R	R: High byte of Pulse Capture Pulse Width W: High byte of Pulse Capture Overflow Bound

11.4.4. Pulse Capture Pulse Width Low Register (*PLSPWL*), address = 0x3A

Bit	Reset	R/W	Description
7 - 0	0xFF	W/R	R: Low byte of Pulse Capture Pulse Width W: Low byte of Pulse Capture Overflow Bound

11.4.5. Pulse Capture Pulse High High Register (*PLSPHH*), address = 0x3B

Bit	Reset	R/W	Description
7 - 0	-	RO	High byte of Pulse Capture Pulse High

11.4.6. Pulse Capture Pulse High Low Register (*PLSPHL*), address = 0x3C

Bit	Reset	R/W	Description
7 - 0	-	RO	Low byte of Pulse Capture Pulse High

11.5. Special Comparator

MF324 has two built-in special comparators, providing better support for 3-phase brushless DC motor applications.

11.5.1. Limit current comparator

MF324 provides a comparator current detection. It can compare signals between PA7 and the internal reference voltage. If using the compare function, please turn off the digital input by setting the *PADIER*.7 register. The internal reference voltage can be selected 50mV – 210mV each 10mV step. Refer to register *LCS*[3:0] more details.

The comparator is disabled after power-on reset and can be enabled by setting *LCS*.7=1.

The comparator result can be read back by *LCS*.5, and it also includes a de-glitch function. For more details, please refer to register *ZCPS*[2:0].

It is worth mentioning that they also have the ability to turn off the PWM, which can effectively reduce the motor current. To use this function, *LCS*.6 must be enabled.

11.5.1.1. Limit Current Setting Register (*LCS*), address = 0x2F

Bit	Reset	R/W	Description
7	0	R/W	Enable Limit comparators. 0 / 1: Disable / Enable
6	0	WO	Enable Limit current comparator result off PWM function. 0 / 1: Disable / Enable
5	0	R/W	Limit current comparator result.
4	0	R/W	ADC reference Voltage. 0 / 1: Disable / Enable Note: Please turn on it if setting the LCS enable
3 - 0	0x05	R/W	The internal reference voltage of Limit current comparator = 50mV + (LCS[3:0] * 10 mV)

11.5.2. Zero crossing point comparator

MF324 provides one comparator for detecting the back electromotive force (BEMF) zero-crossing point (ZCP). It can compare signals between one phase of the three-phase BEMF and the zero point. One phase of the three-phase BEMF can be selected by register *ZCPC*[7:6]. The comparator result (ZCP_X, X = A or B or C.) can be read back by *ZCPC*.5, and it also includes reverse or buffer read back signal by register *ZCPC*[2] and a de-glitch function refer to register *ZCPS*[5:3]. Additionally, it can be input external voltage and internal ZCP common voltage by register *ZCPC*[3].

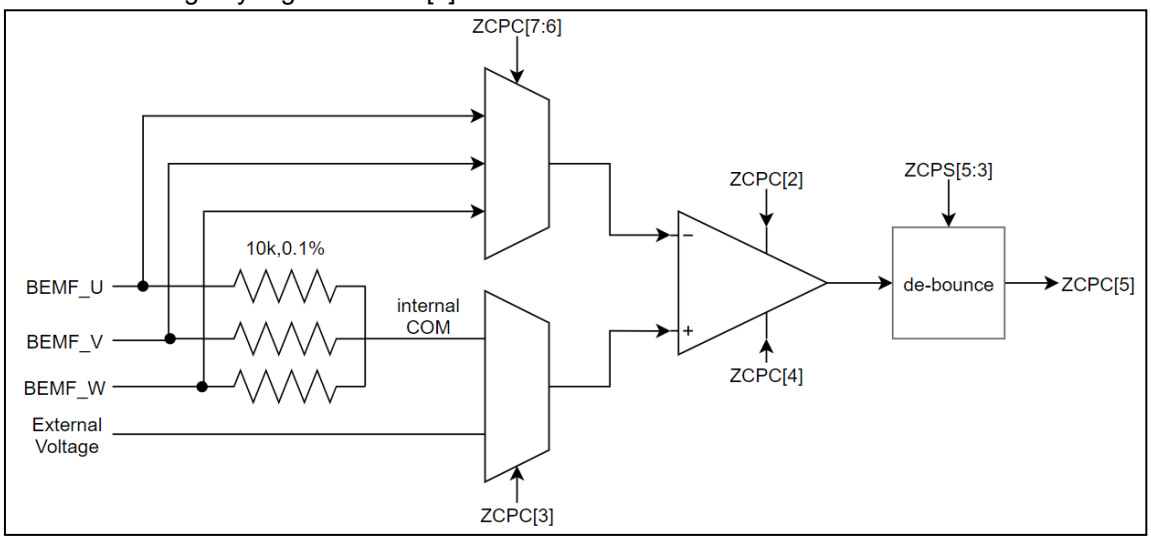


Fig. 31: Hardware Diagram of Zero crossing point comparator

11.5.2.1. Zero Crossing Point Control Register (*ZCPC*), address = 0x30

Bit	Reset	R/W	Description
7 - 6	00	R/W	Enable and select one phase of the three-phase BEMF 00 : BEMF of PA0 phase 01 : BEMF of PA1 phase 10 : BEMF of PA2 phase 11 : Disable
5	0	RO	Zero crossing point comparator result.
4	0	R/W	Zero crossing point comparator result data rate. 0: 2M 1: 4M Note: It can promise 5mV offset comparator choosing the 2M data rate. If choosing the 4M the offset would be higher.
3	0	R/W	Select the ZCP comparator input Voltage. 0: internal ZCP COM voltage 1: external voltage by PA5.
2	0	R/W	Used to control the output state of comparator result. 0 / 1 : buffered / inverted
1 - 0	00	R/W	Reserved.

11.5.2.2. Zero Crossing Point Setting Register (ZCPS), address = 0x31

Bit	Reset	R/W	Description
7	0	R/W	Limit current comparator result data rate. 0: 2M 1: 4M Note: It can promise 5mV offset comparator choosing the 2M data rate. If choosing the 4M the offset would be higher.
6	-	-	Reserved, please keep 0.
5 - 3	000	WO	De-glitch for ZCP comparator result. 000 : disable 001 : 1 us 010 : 2 us 011 : 4 us 100 : 8 us 101 : 16 us 11X : Reserved
2 - 0	000	WO	De-glitch for LC comparator result. 000 : disable 001 : 1 us 010 : 2 us 011 : 4 us 100 : 8 us 101-111 : Reserved

11.6. Multiplier

11.6.1. 8×8 multiplier

There is an 8x8 multiplier on the MF324 that enhances the hardware capabilities of the arithmetic function. The multiplication is an 8x8 unsigned operation and can be completed in 9 system clock cycles. Before setting the start bit (*EARITH*.6), the multiplicand and multiplier must be placed in registers *M8OP1H/M8OP1L and M8OP2*; after 8x8 is completed, the done bit will be set (*EARITH*.1) and the result will be placed in registers *M8RS1/M8RS0*. The hardware diagram of this multiplier is shown as Fig. 32.

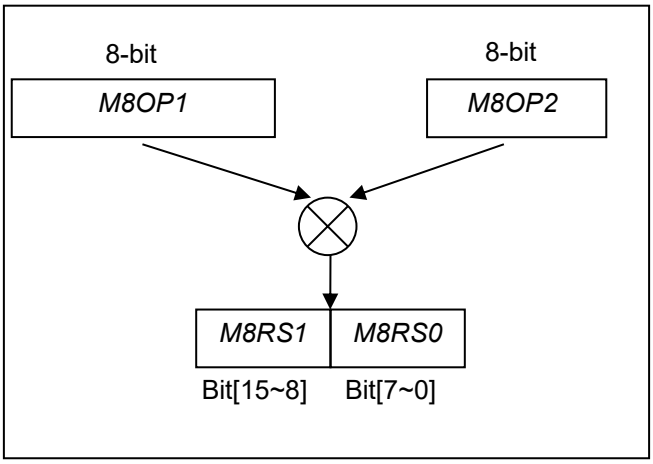


Fig. 32: Block diagram of hardware multiplier

11.6.2. Arithmetic Operation Register (*EARITH*), address = 0x1D

Bit	Reset	R/W	Description
7	-	-	Reserved
6	-	WO	8X8 Multiplier start bit and clear by hardware
5	-	-	Reserved.
4 - 3	-	-	Reserved.
2	-	-	Reserved.
1	-	RO	8X8 Multiplier done flag and clear by start bit
0	-	-	Reserved.

11.6.3. 8X8 Multiplier Operand 1 High Byte Register (M8OP1), address = 0x35

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 1 Byte for hardware multiplication operation

11.6.4. 8X8 Multiplier Operand 2 Register (M8OP2), address = 0x36

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 2 Byte for hardware multiplication operation

11.6.5. 8X8 Multiplier Result1 Register (M8RS1), address = 0x37

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 15~8 of multiplication operation

11.6.6. 8X8 Multiplier Result0 Register (M8RS0), address = 0x38

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 7~0 of multiplication operation

12. Program Writing

There are 5 pins for using the writer to program: PA3, PA5, PA6, VDD and GND.

Please use 5S-P-003x to program MF324 real chip. 3S-P-002x or older versions do not support programming MF324.

Normal Programming Mode

Range of application:

- Single-Chip-Package IC with programming at the writer IC socket or on the handler.
- Multi-Chip-Package (MCP) with MF324. Be sure its connected IC and devices will not be damaged by the following voltages, and will not clam the following voltages.

The voltage conditions in normal programming mode:

(1) VDD is 5.5 V, and the maximum supply current is up to about 20mA.

(2) The voltages of other program pins (except GND) are the same as VDD.

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

Important Cautions:

- You MUST follow the instructions on APN004 and APN011 for programming IC on the handler.
- Connecting a 0.01uF capacitor between VBAT and GND at the handler port to the IC is always good for suppressing disturbance. But please DO NOT connect with > 0.01uF capacitor, otherwise, programming may fail.

For Writer to write MF324, use Jumper7 to adapt program signal connection. The connection of signal depends on the IC package. Please refer to. Chapter 5 of the Writer user manual to find example and make the Jumper-7 adaptive board for target IC package. User can get the user manual from the following link:

PADAUK MCU Program Writer User Manual
http://www.padauk.com.tw/en/technical/index.aspx?kind=27

For example, make JP7 writer signal connections for the QFN-16 package. As shown in Fig.33, the program pins locate in pin 4 (GND), pin6 (PA5), pin 9 (PA3), pin 11 (VDD), and pin 13 (PA6). Jumper7 connection method as the following.

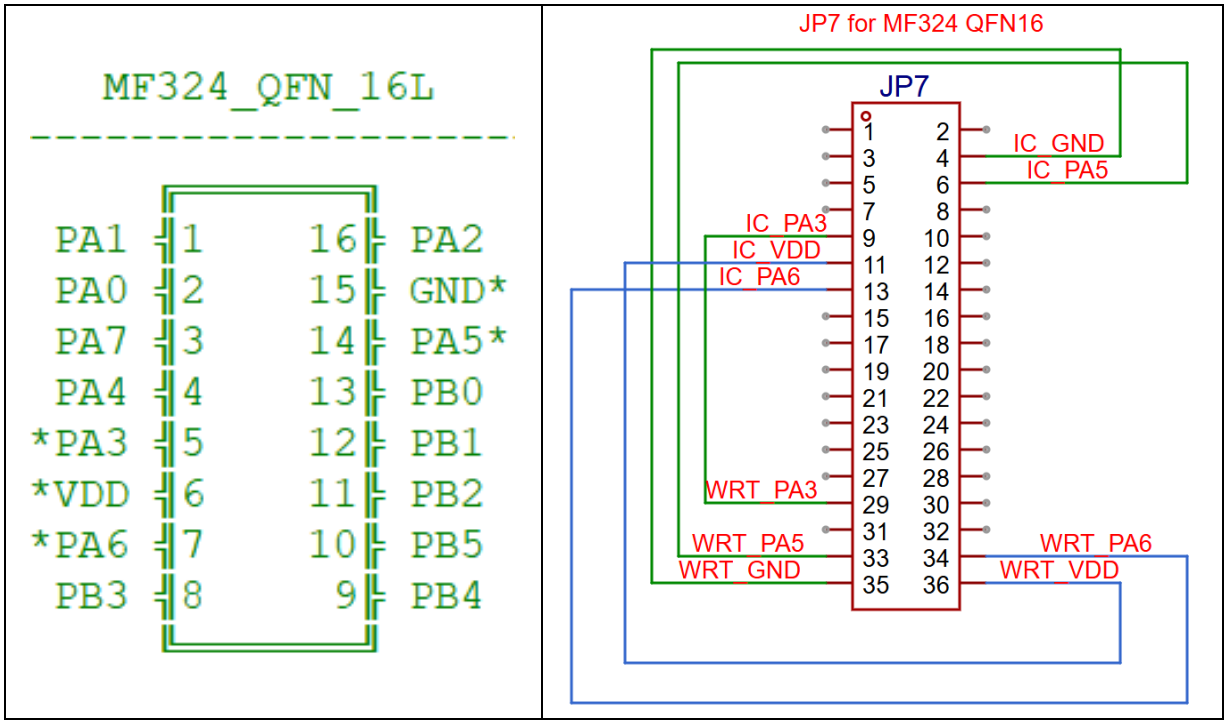
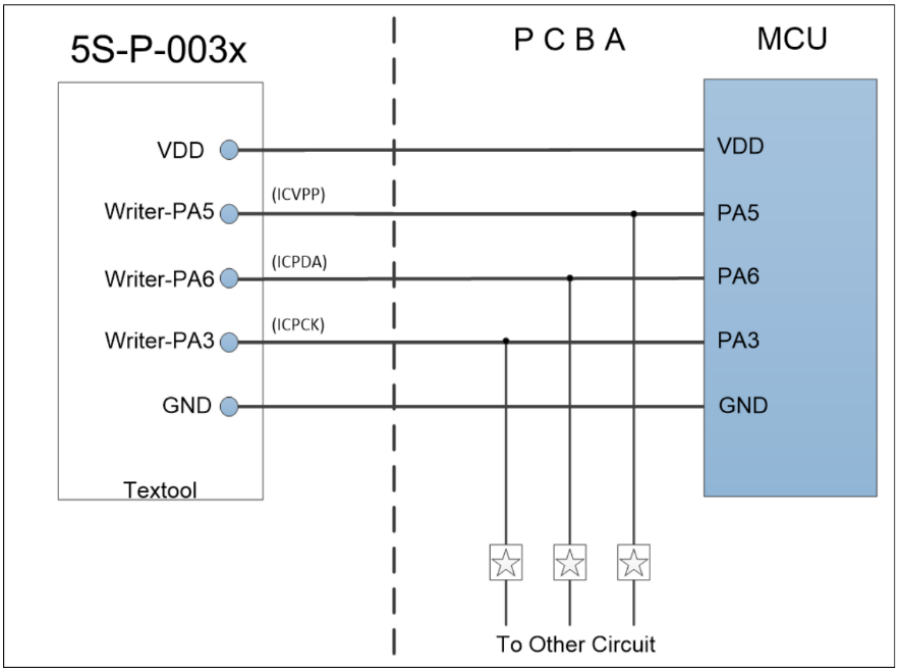


Fig. 33: QFN-16 schematic diagram of Jumper7 for P003x

After installing the JP7 adaptive board on the writer and downloading the PDK file, click <Convert> ➔ <To Package> ➔ Select the PDK to download ➔ Select Package ➔ Saving the PDK file.

On-board Writing

MF324 can support On-board writing. On-Board Writing is known as the situation that the IC has to be programmed when the IC itself and other peripheral circuits and devices have already been mounted on the PCB. Five wires of 5S-P-003x are used for On-Board Writing: ICPCK, ICPDA, VDD, GND and ICVPP. They are used to connect PA3, PA6, VDD, GND and PA5 of the IC correspondingly.



The above figure shows the connection for MF324 on-board writing. In this figure, ☆ can be either resistors or capacitors. They are used to isolate the programming signal wires from the peripheral circuit. It should be ≧ 10KΩ for resistance while ≦ 220pF for capacitance.

Notice:

- Any zener diode ≦ 5.0V, or any circuitry which clamps the 5.0V to be created SHOULD NOT be connected between VDD and GND of the PCB.
- Any capacitor ≧ 500uF SHOULD NOT be connected between VDD and GND of the PCB.
- In general, the writing signal pins PA3, PA5 and PA6 SHOULD NOT be considered as strong output pins.

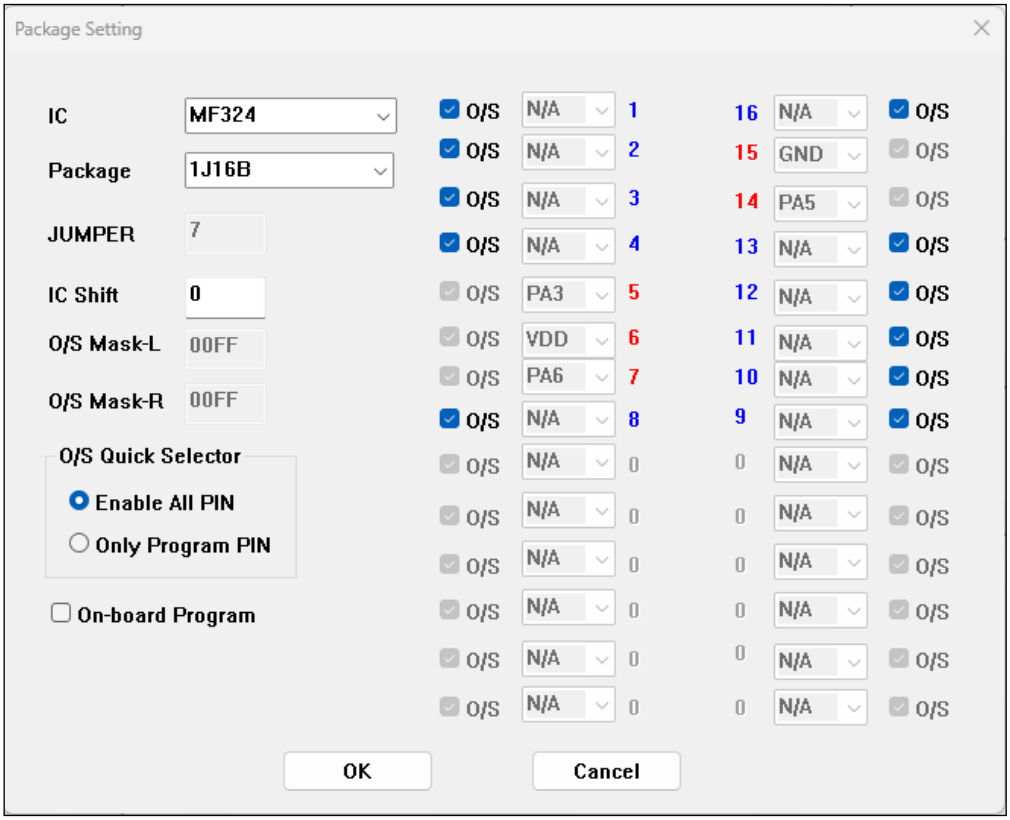


Fig.34: IDE Package Setting GUI

ext, load new PDK from GUI, insert JP7, and then input IC on the socket without shifting. After LCDM displays “IC ready”, click <Auto Program> to write.

13. Instructions

Symbol	Description
ACC	Accumulator
a	Accumulator
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)
pc0	Program counter for FPPA0
pc1	Program counter for FPPA1
pc2	Program counter for FPPA2
pc3	Program counter for FPPA3
pc4	Program counter for FPPA4
pc5	Program counter for FPPA5
pc6	Program counter for FPPA6
pc7	Program counter for FPPA7

13.1. Instruction Table

Instructions	Function	Cycles	Z	C	AC	OV
Data Transfer Instructions						
mov a, I	*mov* a, 0x0f; a ← 0fh;	1	-	-	-	-
mov M, a	*mov* MEM, a; MEM ← a	1	-	-	-	-
mov a, M	*mov* a, MEM ; a ← MEM; Flag Z is set when MEM is zero.	1	Y	-	-	-
mov a, IO	*mov* a, pa ; a ← pa; Flag Z is set when pa is zero.	1	Y	-	-	-
mov IO, a	*mov* pa, a; pa ← a ;	1	-	-	-	-
nmov M, a	*nmov* MEM, a; MEM ← ∓a	1	-	-	-	-
nmov a, M	*mov* a, MEM ; a ← ∓MEM; Flag Z is set when ∓MEM is zero.	1	-	-	-	-
ldtabh index	*ldtabh index;* a ← {bit 15~8 of MTP [index]};	**2**	-	-	-	-
ldtabl index	*ldtabl index;* a ← {bit7~0 of MTP [index]};	**2**	-	-	-	-
ldt16 word	*ldt16* word; word ← 16-bit timer	1	-	-	-	-
stt16 word	*stt16* word; 16-bit timer ← word	1	-	-	-	-
idxm a, index	*idxm* a, index; a ← [index], where index is declared by word.	**2**	-	-	-	-
idxm index, a	*idxm* index, a; [index] ← a; where index is declared by word.	**2**	-	-	-	-
xch M	*xch* MEM ; MEM ← a , a ← MEM	1	-	-	-	-
pushaf	*pushaf*; [sp] ← {flag, ACC}; sp ← sp + 2 ;	1	-	-	-	-
popaf	*popaf;* sp ← sp - 2 ; {Flag, ACC} ← [sp] ;	1	Y	Y	Y	Y
pushw word	*pushw word*; [sp] ← *word* ; sp ← sp + 2	**2**	-	-	-	-
popw word	*popw word*; sp ← sp - 2 ; word ← [sp] ;	**2**	-	-	-	-

Instructions	Function	Cycles	Z	C	AC	OV
Arithmetic Operation Instructions						
add a, I	*add* a, 0x0f ; a ← a + 0fh	1	Y	Y	Y	Y
add a, M	*add* a, MEM ; a ← a + MEM	1	Y	Y	Y	Y
add M, a	*add* MEM, a; MEM ← a + MEM	1	Y	Y	Y	Y
addc a, M	*addc* a, MEM ; a ← a + MEM + C	1	Y	Y	Y	Y
addc M, a	*addc* MEM, a ; MEM ← a + MEM + C	1	Y	Y	Y	Y
addc a	*addc* a ; a ← a + C	1	Y	Y	Y	Y
addc M	*addc* MEM ; MEM ← MEM + C	1	Y	Y	Y	Y
nadd a, M	*nadd* a, MEM ; a ← ∓a + MEM	1	Y	Y	Y	Y
nadd M, a	*nadd* MEM, a; MEM ← ∓MEM + a	1	Y	Y	Y	Y
sub a, I	*sub* a, 0x0f; a ← a - 0fh (a + [2's complement of 0fh])	1	Y	Y	Y	Y
sub a, M	*sub* a, MEM ; a ← a - MEM (a + [2's complement of M])	1	Y	Y	Y	Y
sub M, a	*sub* MEM, a; MEM ← MEM - a (MEM + [2's complement of a])	1	Y	Y	Y	Y
subc a, M	*subc* MEM, a ; a ← a – MEM - C	1	Y	Y	Y	Y
subc M, a	*subc* MEM, a ; MEM ← MEM – a - C	1	Y	Y	Y	Y
subc a	*subc* a; a ← a - C	1	Y	Y	Y	Y
subc M	*subc* MEM; MEM ← MEM - C	1	Y	Y	Y	Y
inc M	*inc* MEM ; MEM ← MEM + 1	1	Y	Y	Y	Y
dec M	*dec* MEM; MEM ← MEM - 1	1	Y	Y	Y	Y
clear M	*clear* MEM ; MEM ← 0	1	-	-	-	-

Instructions	Function	Cycle	Z	C	AC	OV
Shift Operation Instructions						
sr a	*sr* a ; a (0,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
src a	*src* a ; a (c,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
sr M	*sr* MEM ; MEM(0,b7,b6,b5,b4,b3,b2,b1) ← MEM(b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
src M	*src* MEM ; MEM(c,b7,b6,b5,b4,b3,b2,b1) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
sl a	*sl* a ; a (b6,b5,b4,b3,b2,b1,b0,0) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a (b7)	1	-	Y	-	-
slc a	*slc* a ; a (b6,b5,b4,b3,b2,b1,b0,c) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b7)	1	-	Y	-	-
sl M	*sl* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,0) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b7)	1	-	Y	-	-
slc M	*slc* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,C) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM (b7)	1	-	Y	-	-
swap a	*swap* a ; a (b3,b2,b1,b0,b7,b6,b5,b4) ← a (b7,b6,b5,b4,b3,b2,b1,b0)	1	-	-	-	-
swap M	*swap* MEM ; MEM (b3,b2,b1,b0,b7,b6,b5,b4) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0)	1	-	-	-	-
Logic Operation Instructions						
and a, I	*and* a, 0x0f ; a ← a & 0fh	1	Y	-	-	-
and a, M	*and* a, RAM10 ; a ← a & RAM10	1	Y	-	-	-
and M, a	*and* MEM, a ; MEM ← a & MEM	1	Y	-	-	-
or a, I	*or* a, 0x0f ; a ← a | 0fh	1	Y	-	-	-
or a, M	*or* a, MEM ; a ← a | MEM	1	Y	-	-	-
or M, a	*or* MEM, a ; MEM ← a | MEM	1	Y	-	-	-
xor a, I	*xor* a, 0x0f ; a ← a ^ 0fh	1	Y	-	-	-
xor IO, a	*xor* *pa, a ;* pa ← a ^ pa ;	1	-	-	-	-
xor a, M	*xor* a, MEM ; a ← a ^ RAM10	1	Y	-	-	-
xor M, a	*xor* MEM, a ; MEM ← a ^ MEM	1	Y	-	-	-
not a	*not* a ; a ← ~a	1	Y	-	-	-
not M	*not* MEM ; MEM ← ~MEM	1	Y	-	-	-
neg a	*neg* a; a ← ∓a	1	Y	-	-	-
neg M	*neg* MEM; MEM ← ∓MEM	1	Y	-	-	-
comp a, I	*comp* a, 0x55; Flag will be changed by regarding as (a - 0x55)	1	Y	Y	Y	Y
comp a, M	*comp* a,MEM; Flag will be changed by regarding as (a - MEM)	1	Y	Y	Y	Y
comp M, a	*comp* MEM,a; Flag will be changed by regarding as (MEM - a)	1	Y	Y	Y	Y

Instructions	Function	Cycles	Z	C	AC	OV
Bit Operation Instructions						
set0 IO.n	*set0* pa.5 ; PA5=0	1	-	-	-	-
set1 IO.n	*set1* pa.5 ; PA5=1	1	-	-	-	-
set0 M.n	*set0* MEM.5 ; set bit 5 of MEM to low	1	-	-	-	-
set1 M.n	*set1* MEM.5 ; set bit 5 of MEM to high	1	-	-	-	-
swapc *IO.n*	*swapc* IO.0; 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;	1	-	Y	-	-
Conditional Operation Instructions						
ceqsn a, I	*ceqsn* a, 0x55 ; *inc* MEM ; *goto* error ; If a=0x55, then “goto error”; otherwise, “inc MEM”.	1	Y	Y	Y	Y
ceqsn a, M	*ceqsn* a, MEM; If a=MEM, skip next instruction	1	Y	Y	Y	Y
ceqsn M, a	*ceqsn* MEM, a; If a=MEM, skip next instruction	1	Y	Y	Y	Y
cneqsn a, M	*cneqsn* a, MEM; If a≠MEM, skip next instruction	1	Y	Y	Y	Y
cneqsn M, a	*cneqsn* MEM,a; If a≠MEM, skip next instruction	1	Y	Y	Y	Y
cneqsn a, I	*cneqsn* a, 0x55 ; *inc* MEM ; *goto* error ; If a≠0x55, then “goto error”; Otherwise, “inc MEM”	1	Y	Y	Y	Y
t0sn IO.n	*t0sn* pa.5; If bit 5 of port A is low, skip next instruction	1	-	-	-	-
t1sn IO.n	*t1sn* pa.5; If bit 5 of port A is high, skip next instruction	1	-	-	-	-
t0sn M.n	*t0sn* MEM.5 ; If bit 5 of MEM is low, then skip next instruction	1	-	-	-	-
t1sn M.n	*t1sn* MEM.5 ; If bit 5 of MEM is high, then skip next instruction	1	-	-	-	-
izsn a	*izsn* a; a ← a + 1,skip next instruction if a = 0	1	Y	Y	Y	Y
dzsn a	*dzsn* a; a ← a – 1，skip next instruction if a = 0	1	Y	Y	Y	Y
izsn M	*izsn* MEM; MEM ← MEM + 1，skip next instruction if MEM= 0	1	Y	Y	Y	Y
dzsn M	*dzsn* MEM; MEM ← MEM - 1，skip next instruction if MEM= 0	1	Y	Y	Y	Y

Instructions	Function	Cycles	Z	C	AC	OV
System Control Instructions						
call label	*call* function1; [sp] ← pc + 1, pc ← function1, sp ← sp + 2	**1**	-	-	-	-
goto label	*goto* routine1; Go to routine1 and execute program.	**1**	-	-	-	-
delay I	*delay* 0x05; Delay 6 cycles here	1	-	-	-	-
delay a	*delay* a; Delay 16 cycles here if ACC=0fh	1	-	-	-	-
delay M	*delay* M; Delay 256 cycles here if M=ffh	1	-	-	-	-
Notes for *delay* instruction: (1) Because ACC is the temporarily buffer for counting, please make sure that it will not be interrupted when executing *delay* instruction. Otherwise, it may be not the expected delay time. (2) This instruction is not supported in single FPPA mode.						
ret I	*ret* 0x55; A ← 55h ret ;	**2**	-	-	-	-
ret	*ret;* sp ← sp - 2 pc ← [sp]	**2**	-	-	-	-
reti	*reti*; Return to program from interrupt service routine. After this command is executed, global interrupt is enabled automatically.	**2**	-	-	-	-
nop	*nop*; Nothing changed.	1	-	-	-	-
pcadd a	*pcadd* *a*; pc ← pc + a	**2**	-	-	-	-
engint	*engint*; Interrupt request can be sent to FPPA0	1	-	-	-	-
disgint	*disgint* ; Interrupt request is blocked from FPPA0	1	-	-	-	-
reset	*reset*; Reset the whole chip.	1	-	-	-	-
wdreset	*wdreset* ; Reset Watchdog timer.	1	-	-	-	-
pmode n	Operational mode selection for each FPPA unit *pmode* 0; FPPA units bandwidth sharing is set to mode 0	1	-	-	-	-

Instructions	Function	Cycles	Z	C	AC	OV
System Control Instructions						
pmode n	Operational mode selection for each FPPA unit *pmode* 0; FPPA units bandwidth sharing is set to mode 0 Mode FPPA0 ~ FPPA7 bandwidth sharing 0: /2, /2 1: /2, /4, /4 2: /4, /2, /4 3: /2, /4, /8, /8 4: /4, /2, /8, /8 5: /8, /2, /4, /8 6: /4, /4, /4, /4 7: /8, /4, /4, /4, /8 8: /2, /8, /8, /8, /8 9: /4, /4, /4, /8, /8 10: /8, /2, /8, /8, /8 11: /2, /8, /8, /8, /16, /16 12: /16, /2, /8, /8, /8, /16 13: /4, /4, /8, /8, /8, /8 14: /8, /4, /4, /8, /8, /8 15: /4, /4, /4, /8, /16, /16 16: /8, /4, /4, /4, /16, /16 17: /16, /4, /4, /4, /8, /16 18: /2, /8, /8, /16, /16, /16, /16 19: /8, /2, /8, /16, /16, /16, /16 20: /16, /2, /8, /8, /16, /16, /16 21: /4, /4, /4, /16, /16, /16, /16 22: /16, /4, /4, /4, /16, /16, /16 23: /4, /8, /8, /8, /8, /8, /8 24: /8, /2, /16, /16, /16, /16, /16, /16 25: /4, /8, /4, /8, /16, /16, /16, /16 26: /8, /4, /4, /8, /16, /16, /16, /16 27: /2, /8, /16, /16, /16, /16, /16, /16 28: /4, /4, /8, /8, /16, /16, /16, /16 29: /16, /2, /8, /16, /16, /16, /16, /16 30: /8, /4, /4, /8, /16, /16, /16, /16 31: /8, /8, /8, /8, /8, /8, /8, /8	1	-	-	-	-