PUD310

8bit OTP Type PD Controller

Datasheet

Version 0.04 – June 25, 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 which 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 content

Revision History **7**
Usage Warning **8**
1. Features **9**
1.1. Special Features 9
1.2. Fast Charging Specification Application Features 9
1.3. System Features 10
1.4. CPU Features 10
1.5. Ordering/ Package Information 10
2. General Description and Block Diagram **11**
3. Pin Definition and Functional Description **12**
4. Device Characteristics **15**
4.1. AC/DC Device Characteristics 15
4.2. Absolute Maximum Ratings 16
4.3. Typical ILRC frequency vs. VDD 17
4.4. Typical IHRC frequency deviation vs. VDD (calibrated to 12MHz) 17
4.5. Typical ILRC Frequency vs. Temperature 18
4.6. Typical IHRC Frequency vs. Temperature (calibrated to 12MHz) 18
4.7. Typical operating current vs. VDD @ system clock = ILRC/n 19
4.8. Typical operating current vs. VDD @ system clock = IHRC/n 19
4.9. Typical IO driving current (I_{OH}) and sink current (I_{OL}) 20
4.10. Typical IO input high/low threshold voltage (V_{IH}/V_{IL}) 21
4.11. Typical resistance of IO Pull High Resistor 21
4.12. Typical resistance of IO Pull Low Resistor 22
4.13. Typical power down current (I_{PD}) and power save current (I_{PS}) 22
4.14. Different VDD Voltage vs. VREG 23
4.15. DP/DM Pin Characteristics 24
4.16. CC1/CC2 Pin Characteristics 24
4.17. GATE Pin Characteristics 25
4.18. VBUS (VDD) Comparator Characteristics 25
5. Functional Description **26**
5.1. Program Memory - OTP 26
5.2. Boot Procedure 26
5.2.1. Timing charts for reset conditions 27
5.3. Data Memory - SRAM 28
5.4. Oscillator and Clock 28
5.4.1. Internal High RC oscillator and Internal Low RC oscillator 28
5.4.2. Chip calibration 29
5.4.3. IHRC Frequency Calibration and System Clock 29

5.4.4. System Clock and LVR level 30
5.4.5. System Clock Switching 31
5.5 16-bit Timer (Timer16) 32
5.6 8-bit Timer (Timer2) with PWM generation 33
5.6.1 Using the Timer2 to generate periodical waveform 35
5.6.2 Using the Timer2 to generate 8-bit PWM waveform 36
5.6.3 Using the Timer2 to generate 6-bit PWM waveform 37
5.6.4 PWM Waveform 38
5.7 WatchDog Timer 38
5.8 Interrupt 39
5.9 Power-Save and Power-Down 41
5.9.1 Power-Save mode (“*stopexe*”) 41
5.9.2 Power-Down mode (“*stopsys*”) 42
5.9.3 Wake-up 43
5.10 IO Pins 43
5.11 Reset and LVR 44
5.11.1 Reset 44
5.11.2 LVR reset 44
5.12 PD PHY Controller 45
5.12.1 Features 45
5.12.2 Block Diagram 45
5.12.3 Function Description 47
5.12.3.1 TX/RX Hardware State Diagram 47
5.12.3.2 TX Flow Control 48
5.12.3.3 RX Flow Control 49
5.12.3.4 PD PHY Interrupt Type 51
5.12.3.5 TX/RX FIFO Access 51
5.12.3.6 CRC32 Features 52
5.12.3.7 TX and RX Collision Process 52
5.12.3.8 Received Signal Quality Adjustment 52
5.12.3.9 Inner Hardware Signal Probe 53
5.12.4. Programming Sequence 53
5.12.4.1 Transmit one SOP Packet 54
5.12.4.2 Transmit one Hard Reset Signaling 54
5.12.4.3 Receive one SOP Packet 55
5.12.4.4 Receive one Hard Reset Signaling 55
5.12.4.5 CC Wire Manual Control 55
5.13 Analog Front End (DP/DM/CC1/CC2) 56
5.13.1 Features 56
5.13.2 Function Description 57
5.13.2.1 DP and DM Output Control 57
5.13.2.2 DP and DM Input Control 58
5.13.2.3 DP and DM GPIO Control Mode 58
5.13.2.4 CC1 and CC2 Control 62
5.13.2.4.1 USB PD Data Stream Path 63

5.13.2.4.2 USB Type-C 5V/1.5A and 5V/3A Detection 63
5.13.2.4.3 CC1/CC2 Attached Detection 63
5.14 GATE Pin Control and Connection 64
5.14.1 Function Description 64
5.14.2 GATE Pin Application Connection 65
5.15 VBUS (VDD) Voltage Comparator 65
5.15.1 Function Description 65

6. IO Registers 66

6.1. ACC Status Flag Register (*flag*), IO address = 0x00 66
6.2. Reset Control Register (*rstc*), IO address = 0x01 66
6.3. Stack Pointer Register (*sp*), IO address = 0x02 66
6.4. Clock Mode Register (*clkmd*), IO address = 0x03 67
6.5. Interrupt Enable Register (*inten*), IO address = 0x04 67
6.6. Interrupt Request Register (*intrq*), IO address = 0x05 67
6.7. LVR Reset Control Register (lvrc), IO address = 0x08 68
6.8. Interrupt Edge Select Register (*integ*), IO address = 0x09 68
6.9. Timer16 mode Register (*t16m*), IO address = 0x0B 68
6.10. DP/DM IO Control Register (dp_dm_io_ctrl), IO address = 0x0C 69
6.11. Port A Digital Input Enable Register (*padier*), IO address = 0x0D 70
6.12. Port B Digital Input Enable Register (*pbdier*), IO address = 0x0E 71
6.13. Port A Data Register (*pa*), IO address = 0x10 71
6.14. Port A Control Register (*pac*), IO address = 0x11 71
6.15. Port A Pull-High Register (*paph*), IO address = 0x12 71
6.16. Port A Pull-Low Register (*papl*), IO address = 0x13 71
6.17. Port B Data Register (*pb*), IO address = 0x14 71
6.18. Port B Control Register (*pbc*), IO address = 0x15 71
6.19. Port B Pull-High Register (*pbph*), IO address = 0x16 72
6.20. Port B Pull-Low Register (*pbpl*), IO address = 0x17 72
6.21. Timer2 PWM Control Register (*tpw2c*), IO address = 0x1A 72
6.22. Timer2 PWM Scalar Register (*tpw2s*), IO address = 0x1B 73
6.23. Timer2 PWM Bound Register (*tpw2b*), IO address = 0x1C 73
6.24. Operation Register (*opr1*), IO address = 0x1D 73
6.25. Miscellaneous Register (*misc2*), IO address = 0x1F 73
6.26. PD PHY FIFO Register (*pd_fifo*), IO address = 0x30 74
6.27. PD PHY Status Register (*pd_stat*), IO address = 0x31 74
6.28. PD PHY TXRX Register (*pd_txrx*), IO address = 0x32 75
6.29. PD PHY Control Register (*pd_ctrl*), IO address = 0x33 76
6.30. PD PHY Decoder Register (*pd_decoder*), IO address = 0x34 77
6.31. DP Output Register (*dp_out*), IO address = 0x38 77
6.32. DP Input Register (*dp_in*), IO address = 0x39 78
6.33. DM Output Register (*dm_out*), IO address = 0x3A 79
6.34. DM Input Register (*dm_in*), IO address = 0x3B 80
6.35. CC Control Register (*cc_ctrl*), IO address = 0x3D 80
6.36. CC Status Register (*cc_stat*), IO address = 0x3E 81
6.37. CC Miscellaneous Register (*cc_misc*), IO address = 0x3F 81

7. Instructions 82
7.1. Data Transfer Instructions 83
7.2. Arithmetic Operation Instructions 85
7.3. Shift Operation Instructions 87
7.4. Logic Operation Instructions 88
7.5. Bit Operation Instructions 90
7.6. Conditional Operation Instructions 91
7.7. System Control Instructions 92
7.8. Summary of Instructions Execution Cycle 94
7.9. Summary of affected flags by Instructions 94
7.10. BIT definition 95
8. Code Options 95
9. Special Notes 95
9.1. Using IC 95
9.1.1. IO pin usage and setting 95
9.1.2. Interrupt 96
9.1.3. System clock switching 97
9.1.4. Watchdog 97
9.1.5. TIMER time out 97
9.1.6. IHRC 97
9.1.7. LVR 98
9.1.8. Programming Writing 98

Revision History

Revision	Date	Description
0.02	2025/11/28	1. Chapter 1.2**:** Fast Charging Specification Application Features, delete the description “24V high voltage tolerance”. 2. Chapter 4.1: AC/DC Device Characteristics, delete the “±20” value. 3. Chapter 4.15~4.18: Remove “CP Test Range” description. 4. Chapter 4.15: Remove Max and Min definition of V_{ODP5}. 5. Chapter 4.15: Remove Max and Min definition of V_{ODM5}. 6. Chapter 4.17: Remove Max and Min definition of V_{OLGATE}. 7. Chapter 4.17: Remove Max and Min definition of V_{OHGATE}. 8. Chapter 4.17: Remove Max definition of I_{GATE} and add Typ definition of I_{GATE}. 9. Remove Chapter 5.12.3.9: Internal and External Loopback Mode. 10. Remove Chapter 5.12.4.5: Transmit the BIST Signaling. 11. Remove Chapter 5.12.4.6: Internal and External Loopback Testing. 12. Remove Chapter 5.13.2 Block Diagram. 13. Chapter 6.28. PD PHY TXRX Register (pd_txrx), IO address = 0x32, remove Bit7 SEND_BIST definition. 14. Chapter 6.29. PD PHY Control Register (*pd_ctrl*), IO address = 0x33, remove Bit7 RX_REQ definition, Bit6 RX_ACK definition, Bit5 LOOP_SEL definition and Bit4 TEST_SEL definition.
0.03	2026/04/23	1. Remove PA1, PA2, and related information. 2. Update code option descriptions.
0.04	2026/06/25	1. Section 6.4 Delete "Ratio" and "Source".

Usage Warning

User must read all application notes of the IC by detail before using it.

Please visit the official website to download and view the latest APN information associated with it.

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

Feature Documents Software &Tools Application Note

Content	Description	Download (CN)	Download (EN)
APN002	Over voltage protection		
APN003	Over voltage protection		
APN004	Semi-Automatic writing handler		
APN007	Setting up LVR level		
APN011	Semi-Automatic writing Handler improve writing stability		
APN019	E-PAD PCB layout guideline		

1. Features

1.1. Special Features

- General purpose series
- USB Power Delivery (PD) Sink application
- USB Battery Charging 1.2 (BC1.2) detection
- This product can display multiple breathing light modes
- Operating temperature range: -40°C ~ 85°C

1.2. Fast Charging Specification Application Features

- Particular USB DP and DM pins can support different charging specification applications:
  (1) Selective drive voltage level: 5.0V, 3.3V, 0.6V, 0V
  (2) Selective voltage comparator: 3.0V, 2.4V, 1.2V, 0.35V
  (3) 13V high voltage tolerance
  (4) Can be configured as OTP download cable pins at test mode
- Particular USB CC1 and CC2 pins can support USB PD specification application:
  (1) One USB PD PHY transceiver
  (2) One 0.66V voltage comparator for USB Type-C 5V/1.5A detection
  (3) One 1.23V voltage comparator for USB Type-C 5V/3A detection
  (4) Precise 1.125V CC1/CC2 drive voltage
- One digital PD PHY controller can support USB PD specification application
- One particular GATE pin can be connected to external power MOSFET to turn on/off power line:
  (1) Open-drain IO (20mA sink current at V_{OL} = 0.5V)
  (2) 21V high voltage tolerance
- One VBUS (VDD) 7V voltage comparator for charging specification application
- 4V ~ 21V VBUS (VDD) work voltage range

1.3. System Features

- 1.5KW OTP program memory
- 96 Bytes data RAM
- One hardware 16-bit timer
- One hardware 8-bit timers with 6/7/8-bit PWM generation
- 5 IO pins with optional pull-high and pull-low resistor (PA0, PA3, PA5, PB0, PB1)
- Three different IO Driving capability group to meet different application requirements
  (1) PA4, PA6 Drive/ Sink Current= 0.33mA / 0.37mA
  (2) Other IOs (except PA7) Drive/ Sink Current = 16mA / 18mA
- Every IO pin (except PA4 & PA6) can be configured to enable wake-up function
- Clock sources: IHRC, ILRC & NILRC
- For every wake-up enabled IO, two optional wake-up speed are supported: normal and fast
- 16 levels of LVR:
  4.5V, 4.0V, 3.75V, 3.5V, 3.3V, 3.15V, 3.0V, 2.7V, 2.5V, 2.4V, 2.3V, 2.2V, 2.1V, 2.0V, 1.9V, and 1.8V
- One external interrupt pin: PA0
- Bandgap circuit to provide 1.20V reference voltage

1.4. CPU Features

- One processing unit operating mode
- 95 powerful instructions
- Most instructions are 1T execution cycle
- Programmable stack pointer to provide 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
- IO space and memory space are independent

1.5. Ordering/ Package Information

- PUD310-U06A: SOT23-6 (60mil)
- PUD310-2N08A: DFN (2*2mm)
- PUD310-EY10A: ESSOP10 (150mil)
- PUD310-2N06A: DFN (2*2mm)
- PUD310-S08A: SOP8 (150mil)
- PUD310-4N12A: DFN (3*3mm)

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

2. General Description and Block Diagram

The PUD310 family is an IO-Type, fully static, OTP-based CMOS 8-bit microcontroller. It employs RISC architecture and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access.

1.5KW bits OTP program memory and 96 bytes data SRAM are inside. PUD310 also provides two hardware timers: one 16-bit timer and one 8-bit timer with PWM generation, refer to Fig. 2.1.

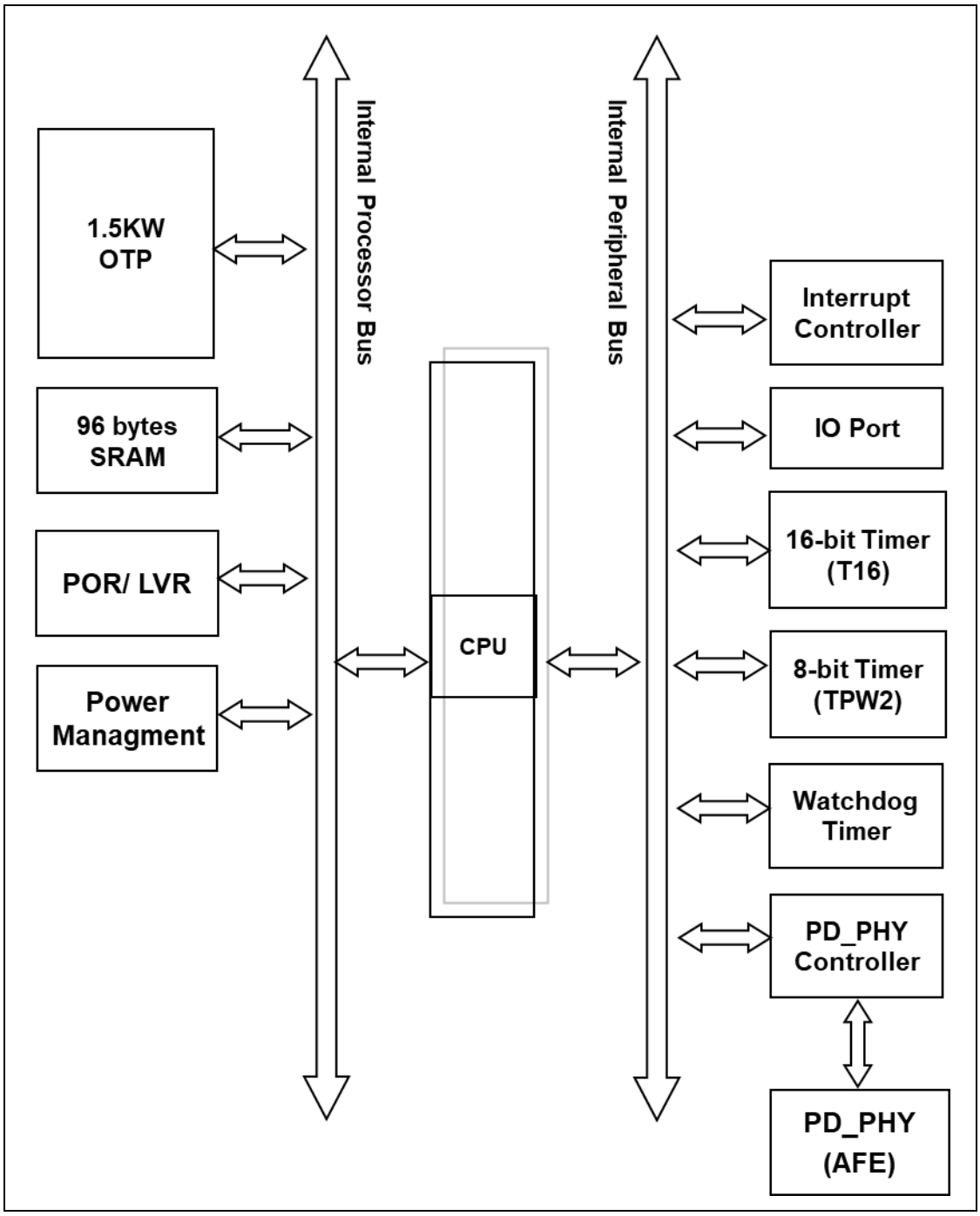
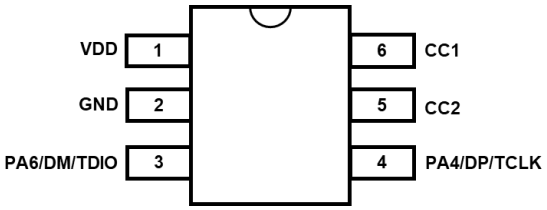
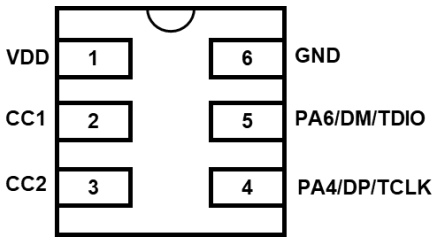


Fig. 2.1: PUD310 architecture

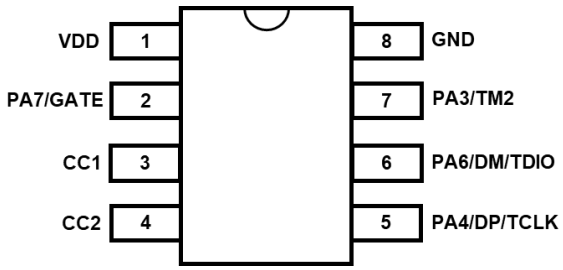
3. Pin Definition and Functional Description



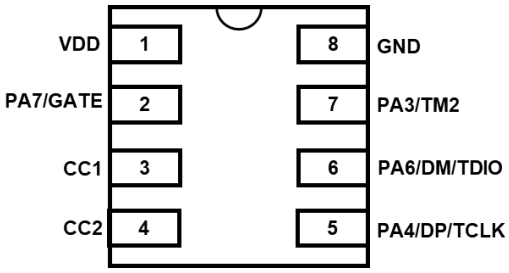
PUD310-U06A: SOT23-6 (60mil)



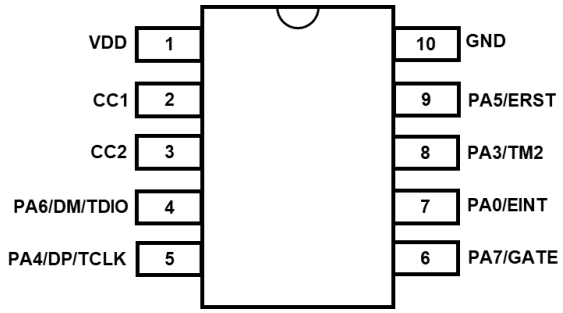
PUD310-2N06A: DFN (2*2mm)



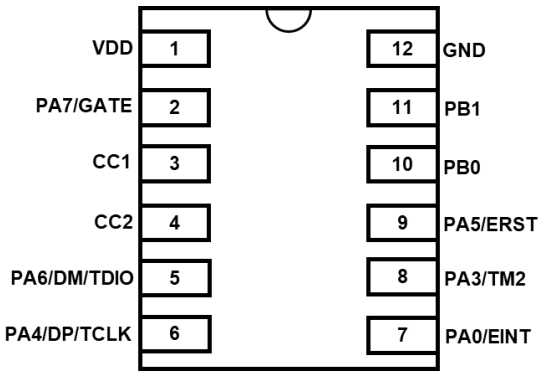
PUD310-S08A: SOP8 (150mil)



PUD310-2N08A: DFN (2*2mm)



PUD310-EY10A: ESSOP10 (150mil)



PUD310-4N12A: DFN (3*3mm)

Pin Name	Pin Type & Buffer Type	Description
PA7 / GATE	OD HVT	The functions of this pin can be: (1) Used to connect to external power MOSFET to turn on/off power line (PA7 can't be GPIO). High voltage tolerance = 21V. A 20mA sink current at V_{OL} = 0.5V.
PA6 / DM / TDIO	IO HVT	The functions of this pin can be: (1) USB DM pin (PA6 can't be GPIO). (2) Download cable TDIO pin for OTP program. If this pin is used for download cable pin, it must work at test mode. High voltage tolerance = 13V.
PA5 / ERST	IO (OD) ST / CMOS	The functions of this pin can be: (1) Bit 5 of port A. It can be configured as digital input or open-drain output, with pull-high resistor. (2) Hardware reset. This pin can be used to wake-up system during sleep mode; however, wake-up function is also disabled if bit 5 of ***padier*** register is "0". Please put 33Ω resistor in series to have high noise immunity when this pin is in input mode.
PA4 / DP / TCLK	IO HVT	The functions of this pin can be: (1) USB DP pin (PA4 can't be GPIO). (2) Download cable TCLK pin for OTP program. If this pin is used for download cable pin, it must work at test mode. High voltage tolerance = 13V.
PA3 / TM2	IO ST / CMOS	The functions of this pin can be: (1) Bit 3 of port A. It can be configured as digital input or two-state output, with pull-high resistor. (2) PWM output from Timer2 When this pin is configured as analog input, please use bit 3 of register ***padier*** to disable the digital input to prevent current leakage. The bit 3 of ***padier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.
CC2	HVT	The functions of this pin can be: (1) USB CC2 pin. High voltage tolerance = 24V. [NOTE 2]
CC1	HVT	The functions of this pin can be: (1) USB CC1 pin. High voltage tolerance = 24V. [NOTE 2]

Pin Name	Pin Type & Buffer Type	Description
PA0 / EINT	IO ST / CMOS	The functions of this pin can be: (1) Bit 0 of port A. It can be configured as digital input or two-state output, with pull-high resistor. (2) External interrupt line 0. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting The bit 0 of ***padier*** register can be set to "0" to disable wake-up from power-down by toggling this pin.
PB1	IO ST / CMOS	The functions of this pin can be: Bit 1 of port B. It can be configured as digital input or two-state output, with pull-high resistor. The bit 1 of ***pbdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled.
PB0	IO ST / CMOS	The functions of this pin can be: Bit 0 of port B. It can be configured as digital input or two-state output, with pull-high resistor. The bit 0 of ***pbdier*** register can be set to "0" to disable digital input, wake-up from power-down by toggling this pin is also disabled.
VDD	VDD	Positive power. [NOTE 2]
GND	GND	Ground

NOTE 1: **IO**: Input/Output; **ST**: Schmitt Trigger input; **OD**: Open Drain; **Analog**: Analog input pin
CMOS: CMOS voltage level; **HVT**: High Voltage Tolerance

NOTE 2: CC1 and CC2 tolerance is tested per below waveforms. (shown in the Fig. 3.1)
(1) Supply VDD=24V power source at T1.
(2) Supply CC1/CC2=24V power source at T1+30ms, to emulate CC1/CC2 short to VDD (VBUS).
(3) Disable VDD and CC1/CC2 power source at T1+60ms.
(4) Per USB PD specification, SOURCE shall disable VBUS within 20ms when detect SINK detached.
(5) Maximum rating 24V is typical value and is tested in LAB under room temperature.

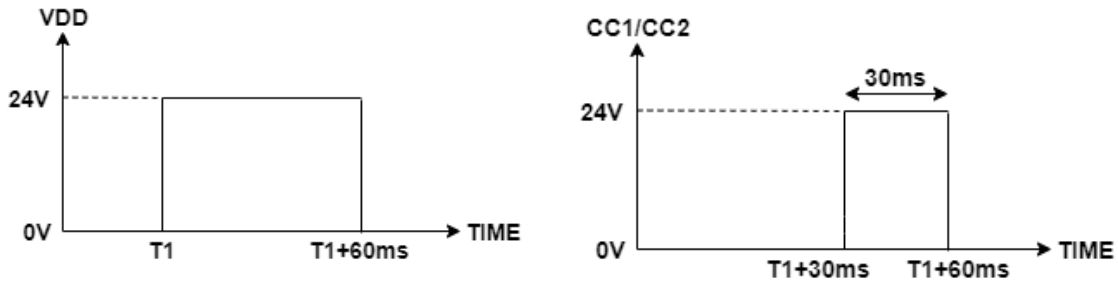


Fig. 3.1 Test Method for CC1/CC2 Maximum Rating

4. Device Characteristics

4.1. AC/DC Device Characteristics

All data are acquired under the conditions of Ta= -40 °C ~ 85 °C, V_{DD}=5.0V, f_{SYS} =3MHz unless noted.

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{DD}	Operating Voltage	4#		21	V	# Subject to LVR tolerance
LVR%	Low Voltage Reset Tolerance	-7		7	%	
f_{SYS}	System clock (CLK)* = IHRC/1 IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0 0	 41K	 12M 6M 3M 1.5M	Hz	 V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} = 5.0V
V_{POR}	Power On Reset Voltage		1.8*		V	* Subject to LVR tolerance
I_{OP}	Operating Current		1.7 30		mA uA	f_{SYS} =IHRC/8 =1.5MIPS@5.0V f_{SYS} =ILRC =41KHz@5.0V
I_{PD}	Power Down Current (by ***stopsys*** command)		0.24 0.15		uA uA	f_{SYS} =0Hz, V_{DD} =5.0V f_{SYS} =0Hz, V_{DD} =3.3V
I_{PS}	Power Save Current (by ***stopexe*** command)		3.0		uA	V_{DD} =5.0V; f_{SYS} =ILRC Only ILRC module is enabled.
V_{IL}	Input low voltage for IO lines	0		0.2 V_{REG}	V	
V_{IH}	Input high voltage for IO lines	0.7 V_{REG}		V_{REG}	V	
I_{OL}	IO lines sink current					
	PA6, PA4 PA5, PA3, PA0, PB1, PB0		0.4 17		mA	V_{DD} =5.0V, V_{OL} =0.5V V_{DD} =5.0V, V_{OL} =0.5V
I_{OH}	IO lines drive current					
	PA5, PA3, PA0, PB1, PB0 PA6, PA4		-13 -0.15		mA	V_{DD} =5.0V, V_{OH} =4.5V
V_{IN}	Input voltage	-0.3		V_{DD} +0.3	V	
$I_{INJ\ (PIN)}$	Injected current on pin			1	mA	V_{REG} +0.3 ≧ V_{IN} ≧ -0.3
R_{PH}	Pull-high Resistance		75		KΩ	V_{DD} =5.0V
R_{PL}	Pull-low Resistance		75		KΩ	V_{DD} =5.0V
V_{BG}	Bandgap Reference Voltage	1.145*	1.20*	1.255*	V	V_{DD} =4.5V ~ 21V -40°C <Ta< 85°C*
f_{IHRC}	Frequency of IHRC after calibration *	11.82*	12*	12.18*	MHz	25°C, V_{DD} =4.5V~21V
		11.40*	12*	12.60*		V_{DD} =4.5V~21V, -40°C <Ta< 85°C*
V_{REG}	Regulator output voltage		4.55			V_{DD} =5.0V
t_{INT}	Interrupt pulse width	30			ns	V_{DD} =5.0V
V_{DR}	RAM data retention voltage*	1.5			V	in stop mode

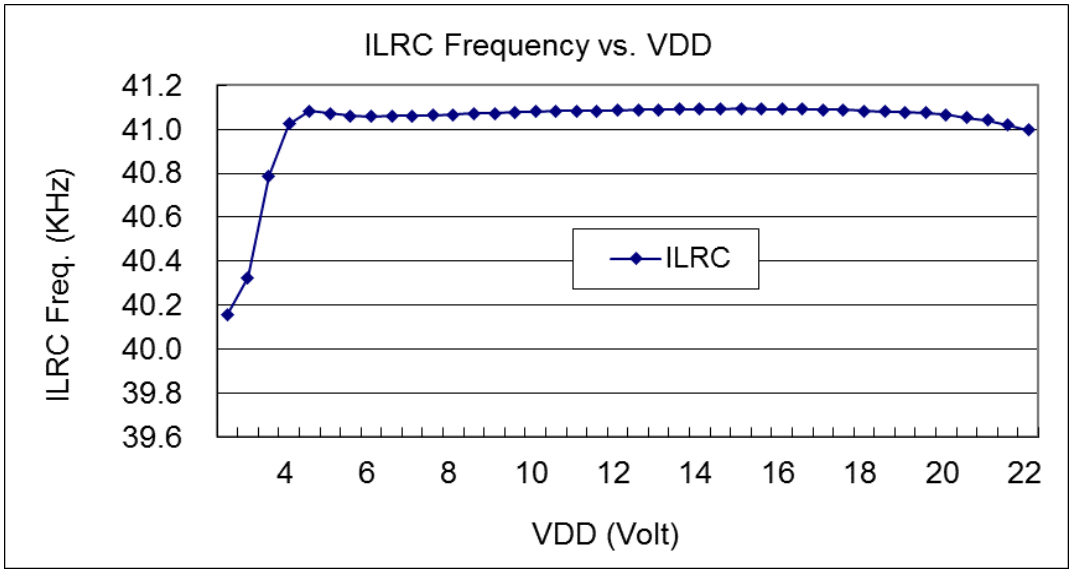
Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
t_{WDT}	Watchdog timeout period		8K		T_{ILRC}	misc[1:0]=00 (default)
			16K			misc[1:0]=01
			64K			misc[1:0]=10
			256K			misc[1:0]=11
t_{WUP}	Wake-up time period for fast wake-up		15		T_{ILRC}	Where T_{ILRC} is the time period of ILRC
	Wake-up time period for slow wake-up		1024			
t_{SBP}	System boot-up period from power-on for Slow boot-up		25		ms	V_{DD} =5.0V
t_{RST}	External reset pulse width	120			us	@ V_{DD} =5.0V
CPos	Comparator offset*		±10		mV	
CPcm	Comparator input common mode*	0		V_{DD} -1.5	V	
CPspt	Comparator response time**		100	500	ns	Both Rising and Falling
CPmc	Stable time to change comparator mode		2.5	7.5	us	
CPcs	Comparator current consumption		20		uA	V_{DD} = 3.3V

*These parameters are not tested for each chip.

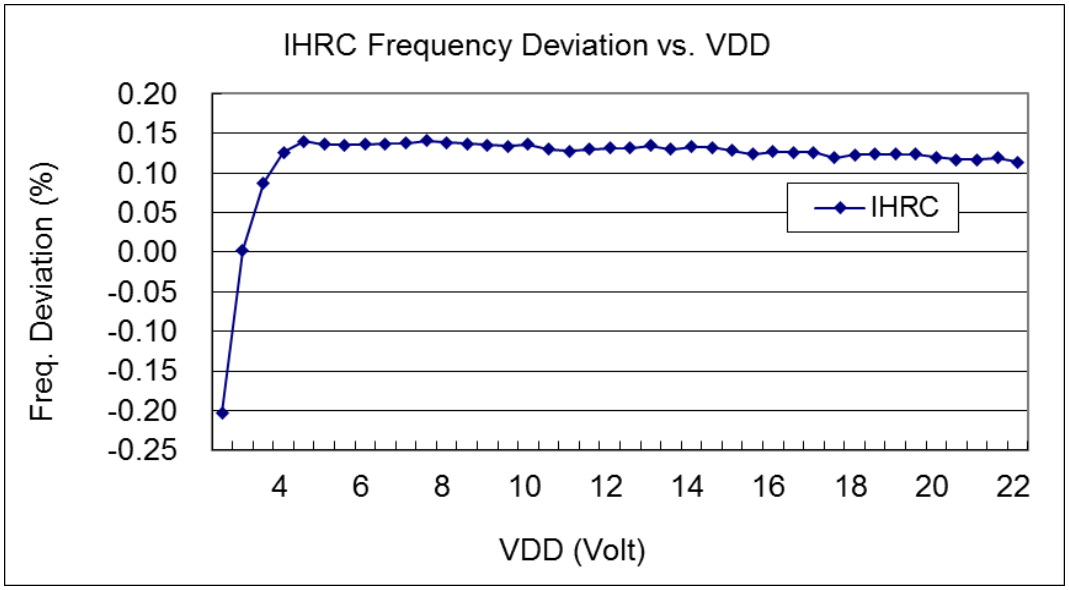
4.2. Absolute Maximum Ratings

Parameter	Maximum Rating	Notes
Supply Voltage (VDD)	24V	Maximum rating tested per Figure 3.1. Overstress in voltage (24V) or time (30ms) may cause permanent damage.
GATE Voltage	21V	If GATE is forced to the maximum rating, it may lead to permanent damage to the IC.
CC1, CC2 Voltage	24V	Maximum rating tested per Figure 3.1. Overstress in voltage (24V) or time (30ms) may cause permanent damage.
DP, DM Voltage	13V	If DP or DM are forced to the maximum rating, it may lead to permanent damage to the IC.
Maximum CLOAD on CC1/CC2	1.4nF	If CC1 or CC2 are loaded with total capacitance more than 1.4nF, it may cause USB PD transaction failure.
Input Voltage Range	-0.3V ~ V_{REG} + 0.3V	Exceeding this range may cause device damage.
Operating Temperature	-40°C ~ 85°C	
Storage Temperature	-50°C ~ 125°C	
Junction Temperature	150°C	

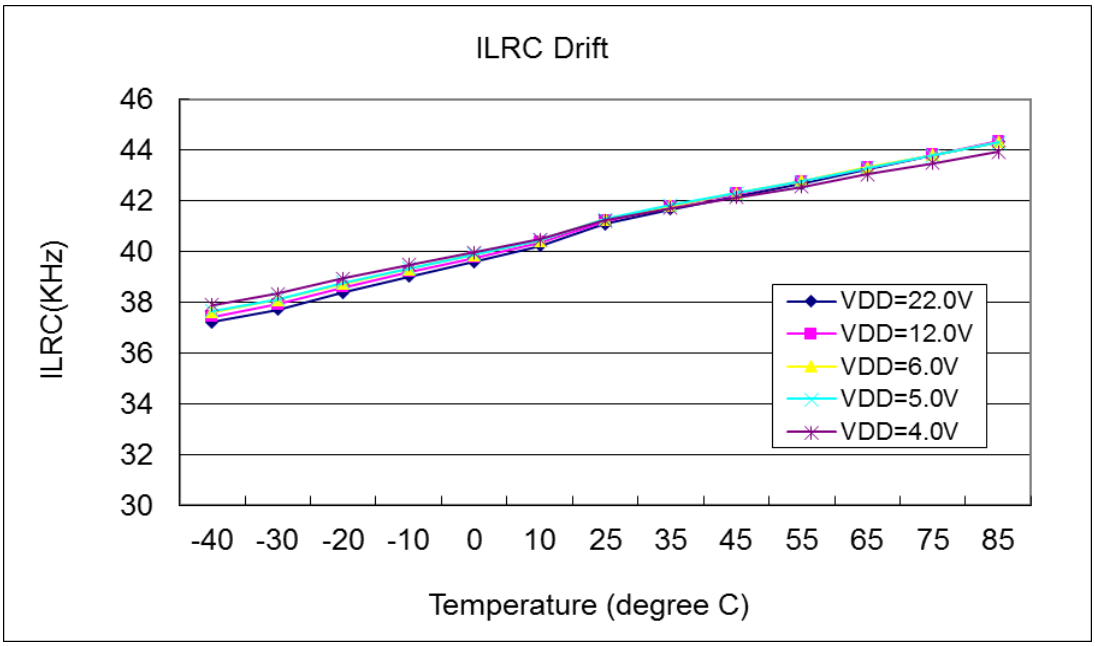
4.3. Typical ILRC frequency vs. VDD



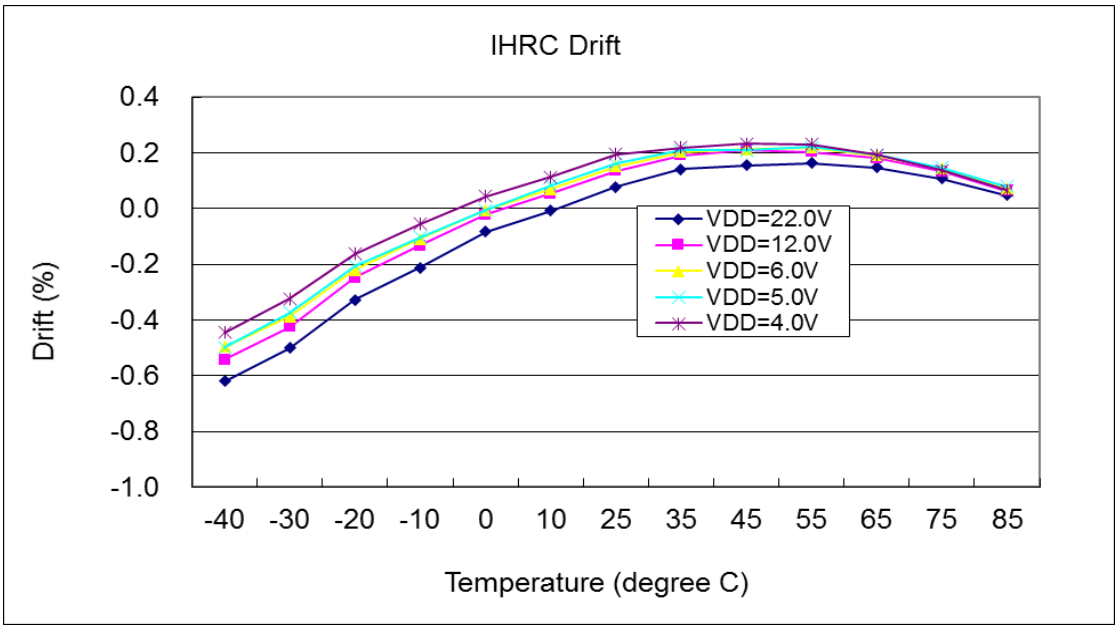
4.4. Typical IHRC frequency deviation vs. VDD (calibrated to 12MHz)



4.5. Typical ILRC Frequency vs. Temperature



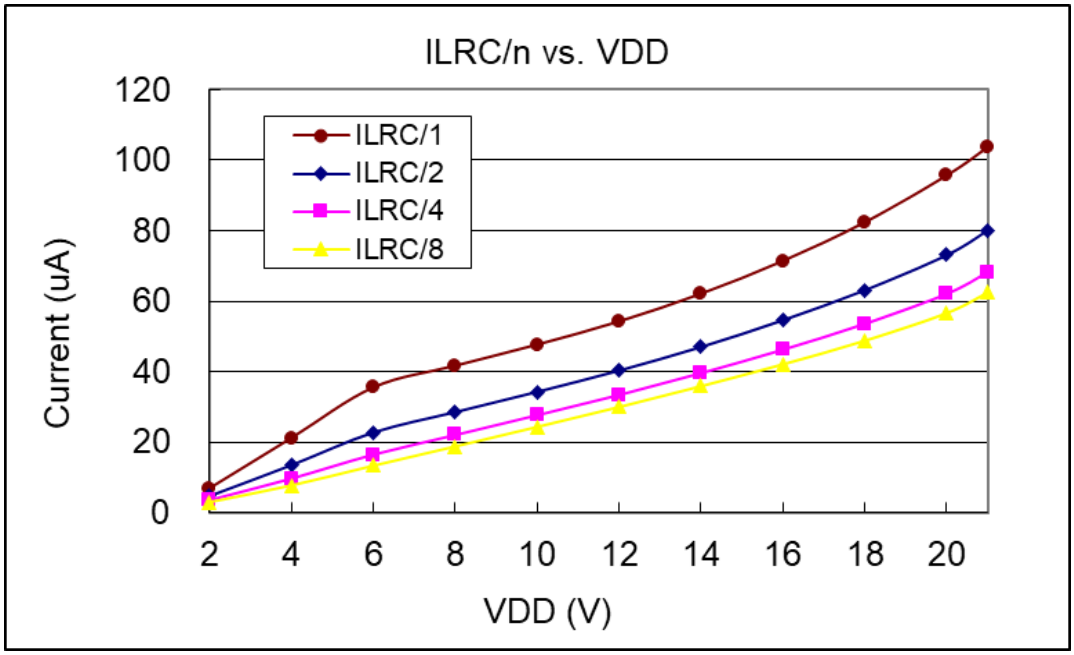
4.6. Typical IHRC Frequency vs. Temperature (calibrated to 12MHz)



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

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

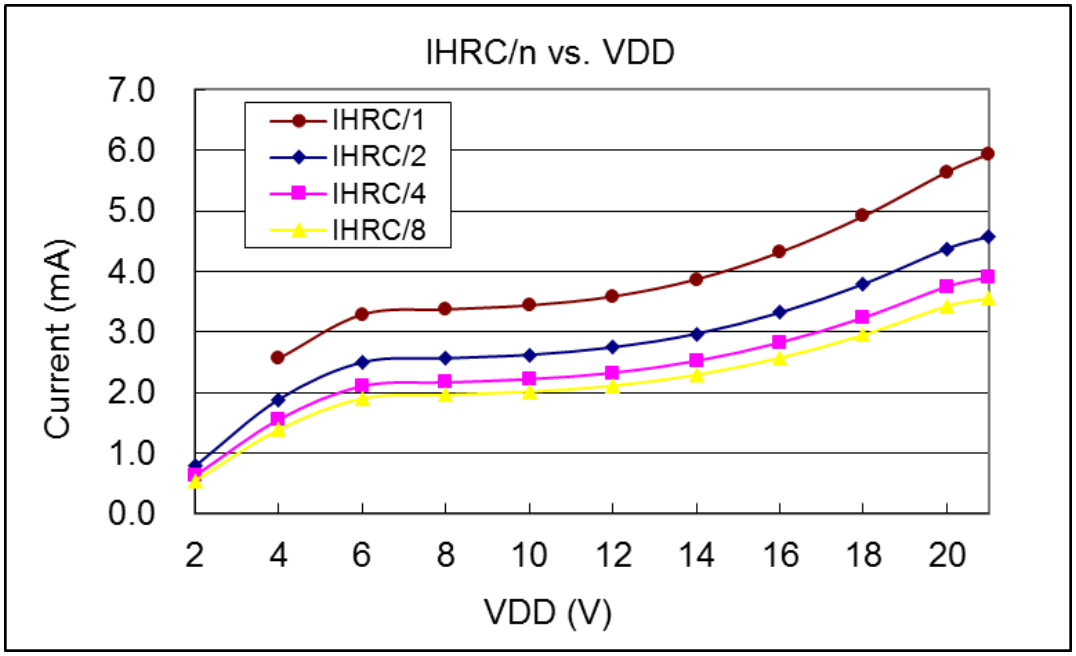
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



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

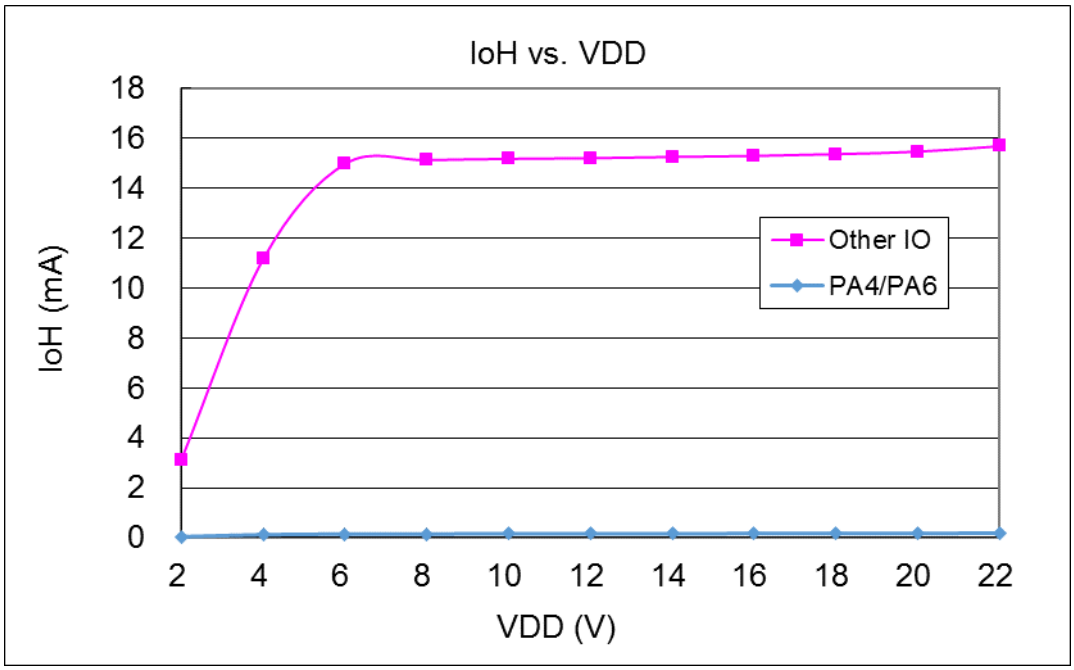
Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16, TM2;

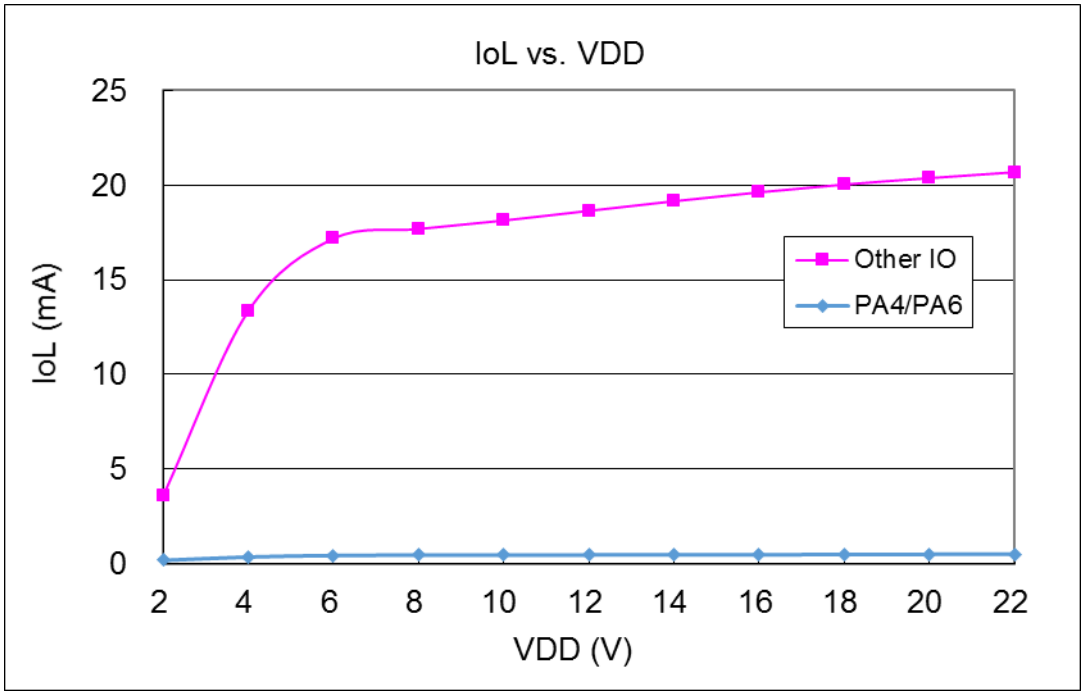
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



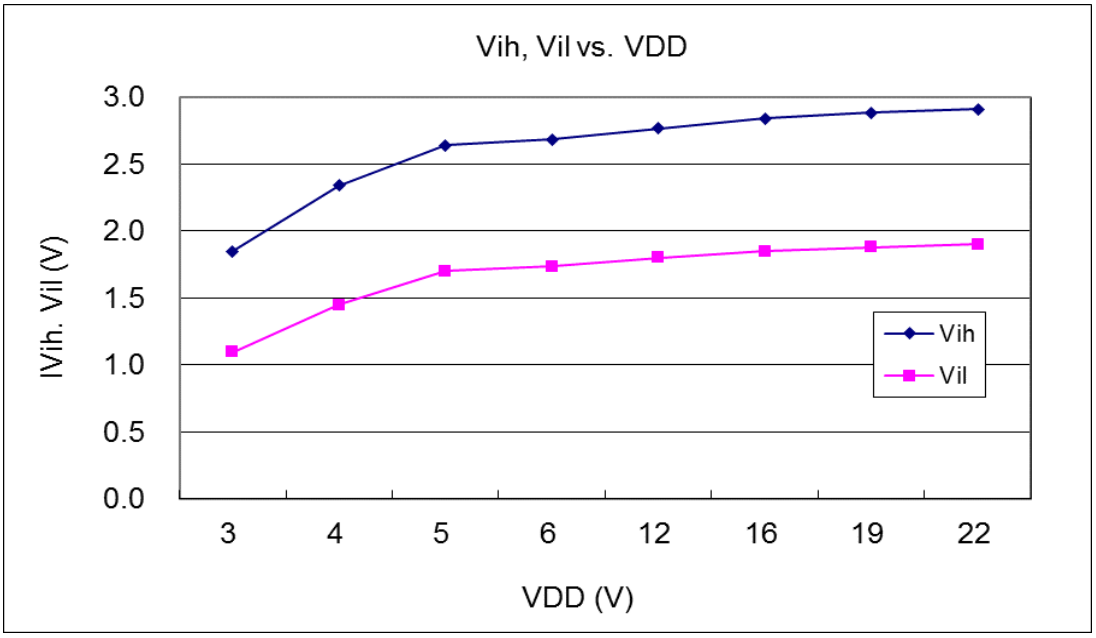
4.9. Typical IO driving current (I_{OH}) and sink current (I_{OL})

(V_{OH}=0.9*V_{REG}, V_{OL}=0.1*V_{REG})

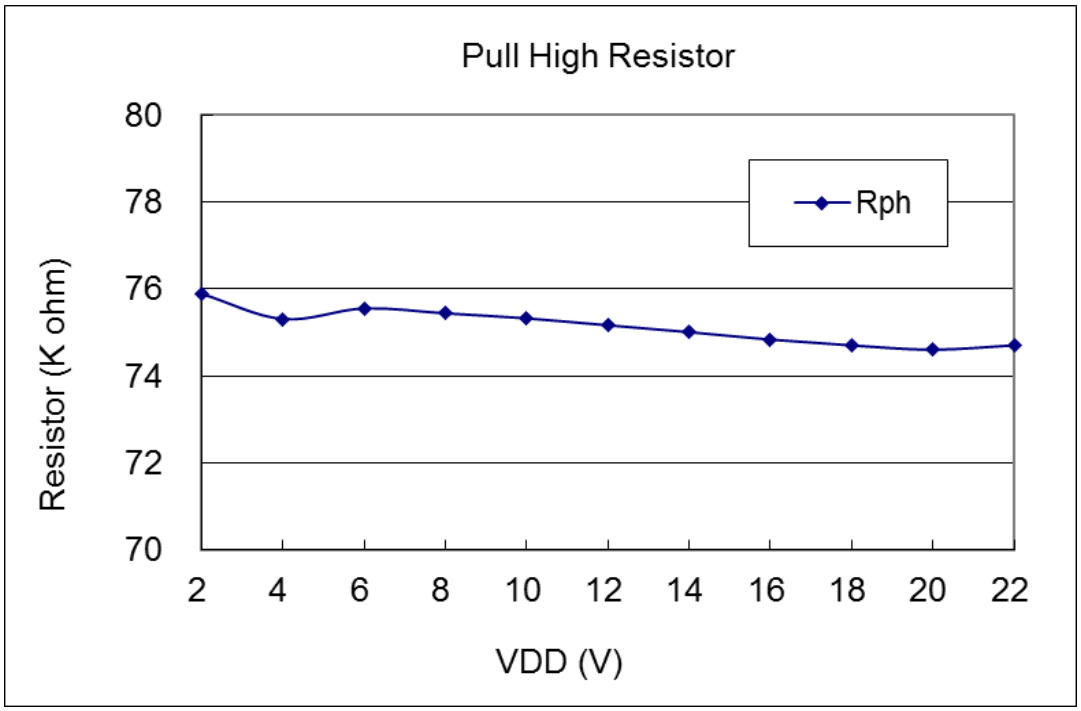




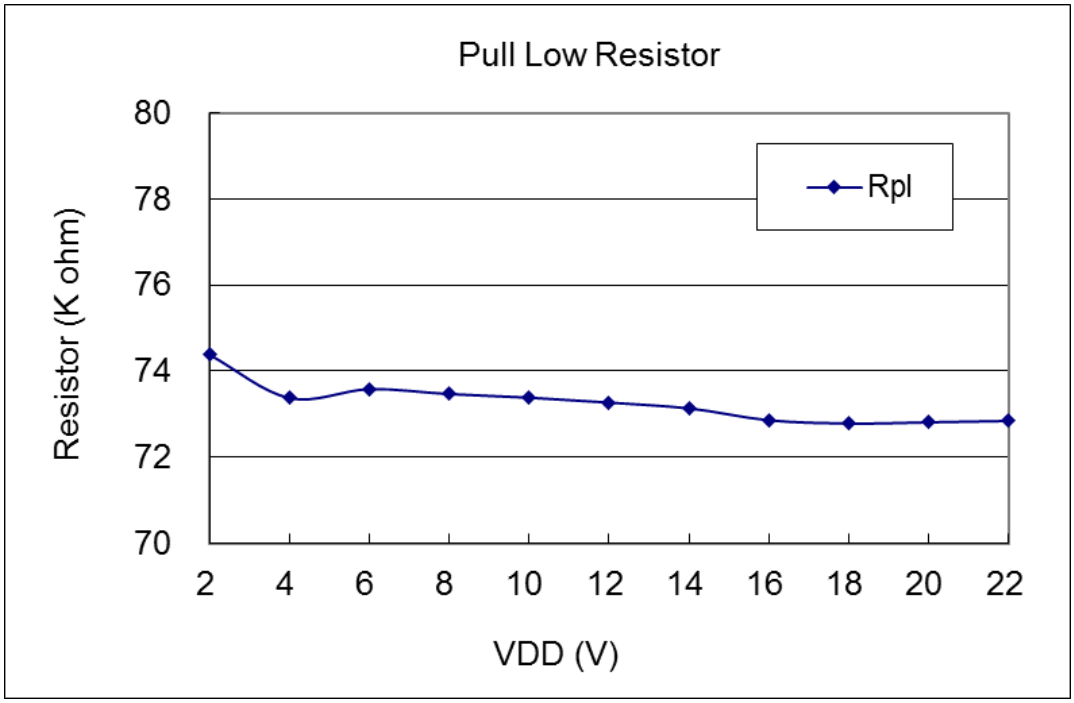
4.10. Typical IO input high/low threshold voltage (V_{IH}/V_{IL})



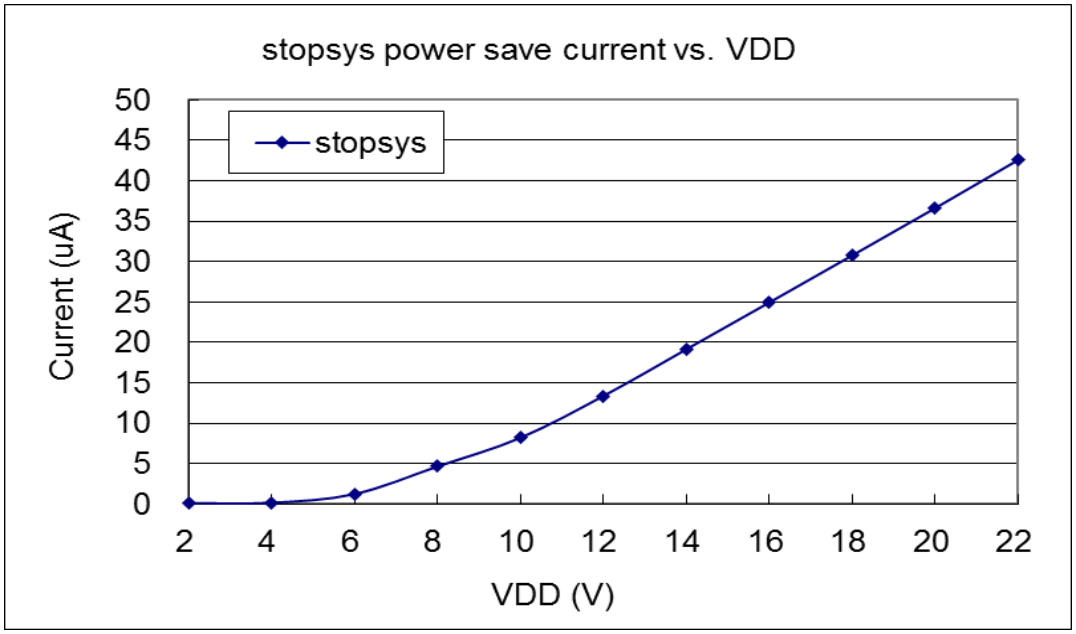
4.11. Typical resistance of IO Pull High Resistor

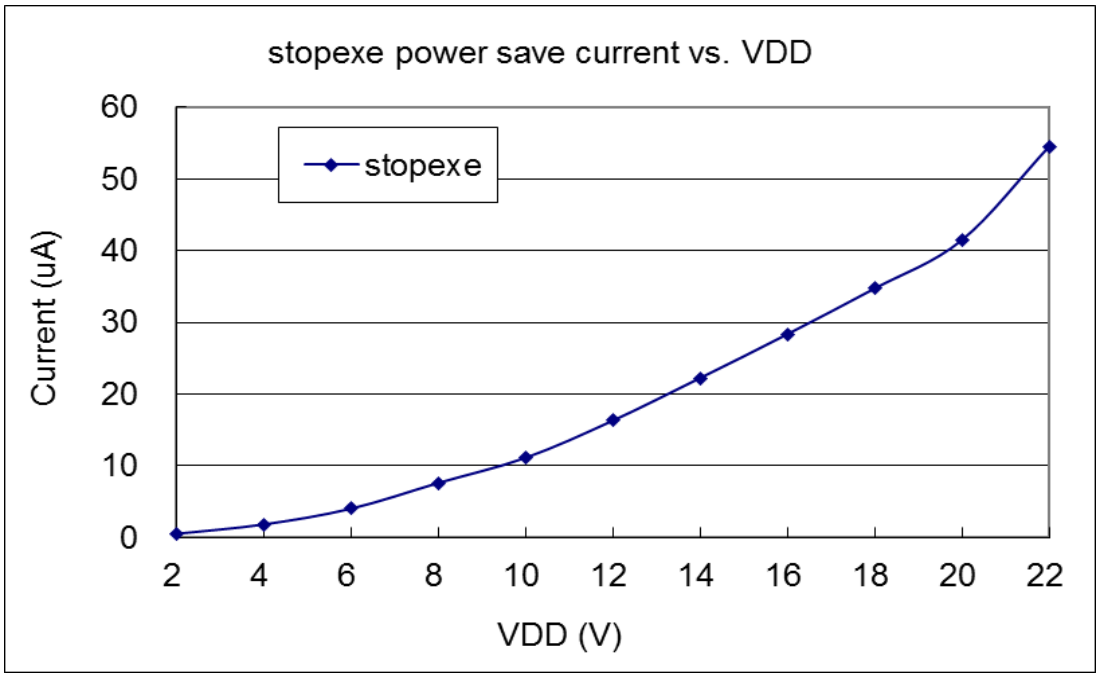


4.12. Typical resistance of IO Pull Low Resistor

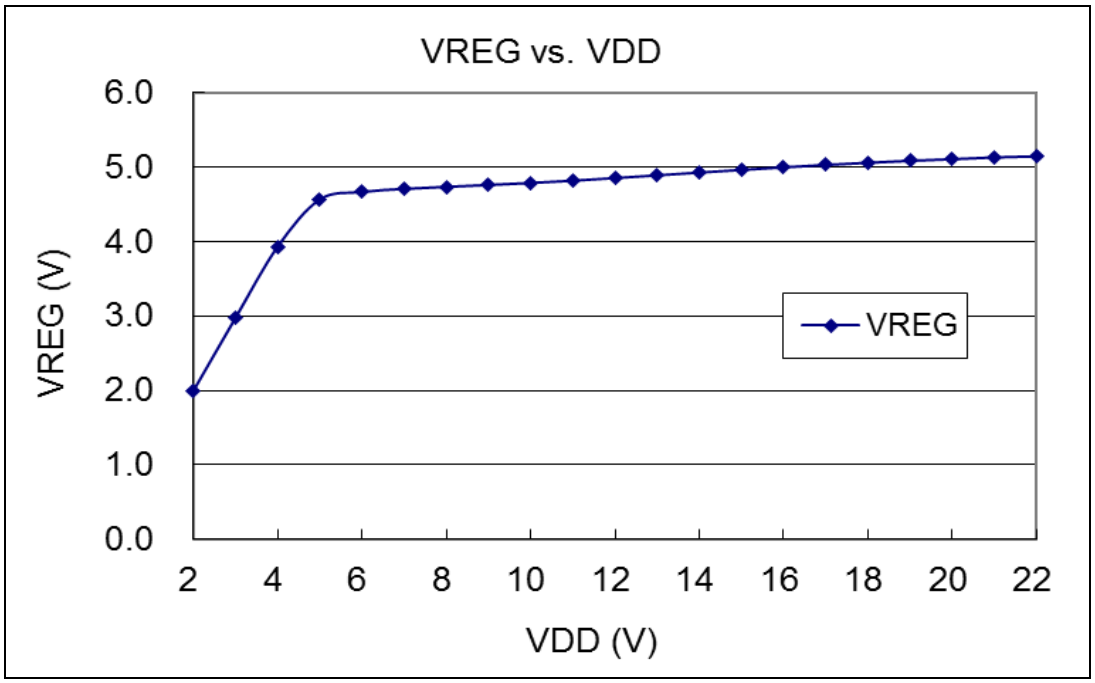


4.13. Typical power down current (I_{PD}) and power save current (I_{PS})





4.14. Different VDD Voltage vs. VREG



4.15. DP/DM Pin Characteristics

Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16

All data are acquired under the conditions of Ta= 25 °C, V_{DD}=5.0V, f_{SYS} =12MHz unless noted.

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{ODP5}	DP 5V Drive Voltage		V_{REG}*		V	See section 4.14 Different VDD vs. V_{REG}
$V_{ODP3.3}$	DP 3.3V Drive Voltage	3.0	3.3	3.6	V	
$V_{ODP0.6}$	DP 0.6V Drive Voltage	0.5	0.6	0.7	V	
V_{ODP0}	DP 0V Drive Voltage	-0.1	0.0	0.1	V	
V_{IDP3}	DP Comparator 3V Threshold	2.7	3.0	3.3	V	
$V_{IDP2.4}$	DP Comparator 2.4V Threshold	2.0	2.4	2.7	V	
$V_{IDP1.2}$	DP Comparator 1.2V Threshold	1.08	1.2	1.32	V	
$V_{IDP0.35}$	DP Comparator 0.35V Threshold	0.25	0.35	0.45	V	
V_{ODM5}	DM 5V Drive Voltage		V_{REG}*		V	See section 4.14 Different VDD vs. V_{REG}
$V_{ODM3.3}$	DM 3.3V Drive Voltage	3.0	3.3	3.6	V	
$V_{ODM0.6}$	DM 0.6V Drive Voltage	0.5	0.6	0.7	V	
V_{ODM0}	DM 0V Drive Voltage	-0.1	0.0	0.1	V	
V_{IDM3}	DM Comparator 3V Threshold	2.7	3.0	3.3	V	
$V_{IDM2.4}$	DM Comparator 2.4V Threshold	2.0	2.4	2.7	V	
$V_{IDM1.2}$	DM Comparator 1.2V Threshold	1.08	1.2	1.32	V	
$V_{IDM0.35}$	DM Comparator 0.35V Threshold	0.25	0.35	0.45	V	

*This parameter is subject to Regulator output voltage (V_{REG}).

4.16. CC1/CC2 Pin Characteristics

Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16

All data are acquired under the conditions of Ta= 25 °C, V_{DD}=5.0V, f_{SYS} =12MHz unless noted.

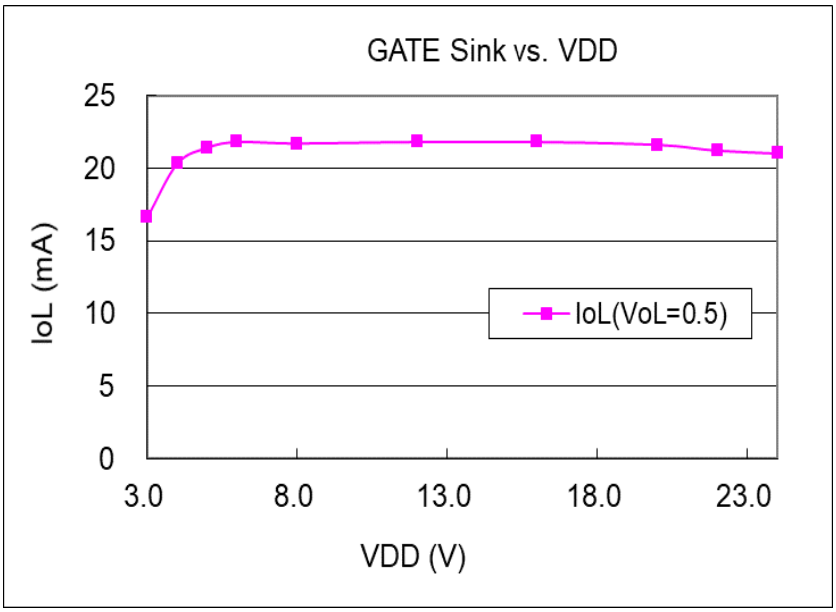
Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{OHCC1}	CC1 1.125V Drive Voltage	1.05	1.125	1.20	V	
V_{OLCC1}	CC1 0V Drive Voltage	-0.1	0.0	0.1	V	
V_{ATTCC1}	CC1 Comparator 0.2V Threshold	0.15	0.20	0.35	V	
V_{OHCC2}	CC2 1.125V Drive Voltage	1.05	1.125	1.20	V	
V_{OLCC2}	CC2 0V Drive Voltage	-0.1	0.0	0.1	V	
V_{ATTCC2}	CC2 Comparator 0.2V Threshold	0.15	0.20	0.35	V	
$V_{1.5ACC}$	CC Comparator 0.66V Threshold	0.61	0.66	0.80	V	
V_{3ACC}	CC Comparator 1.23V Threshold	1.16	1.23	1.31	V	

4.17. GATE Pin Characteristics

Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16

All data are acquired under the conditions of Ta= 25 °C, V_{DD}=5.0V, f_{SYS} =12MHz unless noted.

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{OLGATE}	GATE 0V Drive Voltage		0.0		V	GATE connect a 31.5k resistor to 10V
V_{OHGATE}	GATE Floating Output Test		10.0		V	GATE connect a 31.5k resistor to 10V
V_{OHGATE}	GATE Floating Output Test		22		V	GATE connect a 55k resistor to 22V
I_{GATE}	GATE Sink Current		20		mA	At V_{OL} = 0.5V



4.18. VBUS (VDD) Comparator Characteristics

Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16

All data are acquired under the conditions of Ta= 25 °C, V_{DD}=5.0V, f_{SYS} =12MHz unless noted.

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V_{VBUS}	VBUS Comparator 7V Threshold	6.0	7.0	8.0	V	

5. Functional Description

5.1. Program Memory - OTP

The OTP (One Time Programmable) program memory is used to store the program instructions to be executed. The OTP program memory may contain the data, tables and interrupt entry. After reset, the initial address 0x000 is reserved for system using, so the program will start from 0x001 which is GOTO FPPA0 instruction usually. The interrupt entry is 0x10 if used, the last 16 addresses are reserved for system using, like checksum, serial number, etc. The OTP program memory for PUD310 is 1.5KW that is partitioned as Table 5.1. The OTP memory from address '0x5F0 to 0x5FF is for system using, address space from0x002 to 0x00F and from 0x011 to 0x5EF are user program spaces.

Address	Function
0x000	System Using
0x001	GOTO FPPA0 instruction
0x002	User program
•	•
0x00F	User program
0x010	Interrupt entry address
0x011	User program
•	•
0x5EF	User program
0x5F0	System Using
•	•
0x5FF	System Using

Table 5.1: Program Memory Organization

5.2. Boot Procedure

POR (Power-On-Reset) is used to reset PUD310 when power up. The boot up time is about 1024 ILRC clock cycles. Customer must ensure the stability of supply voltage after power up, the power up sequence is shown in the Fig. 5.1 and t_{SBP} is the boot up time.

Please noted, during Power-On-Reset, the V_{DD} must go higher than V_{POR} to boot-up the MCU.

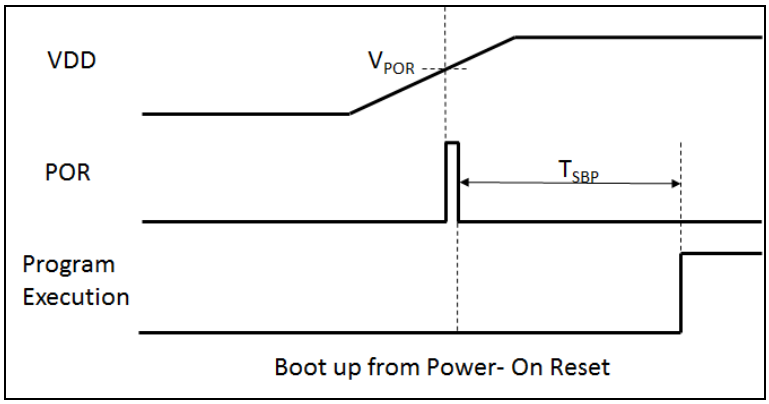
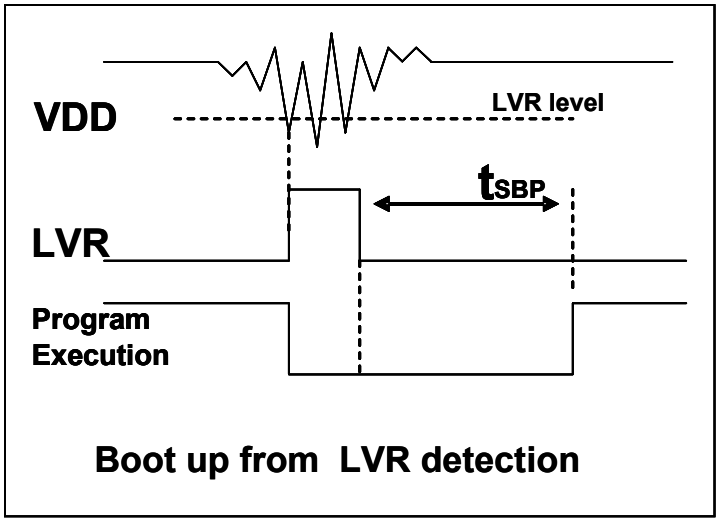
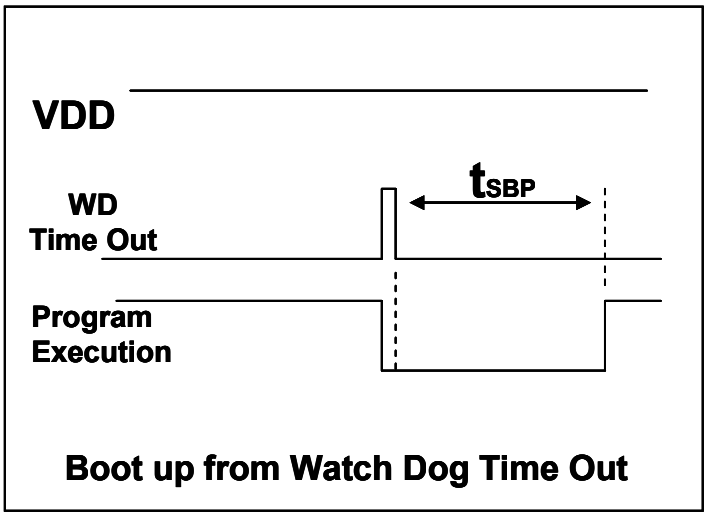
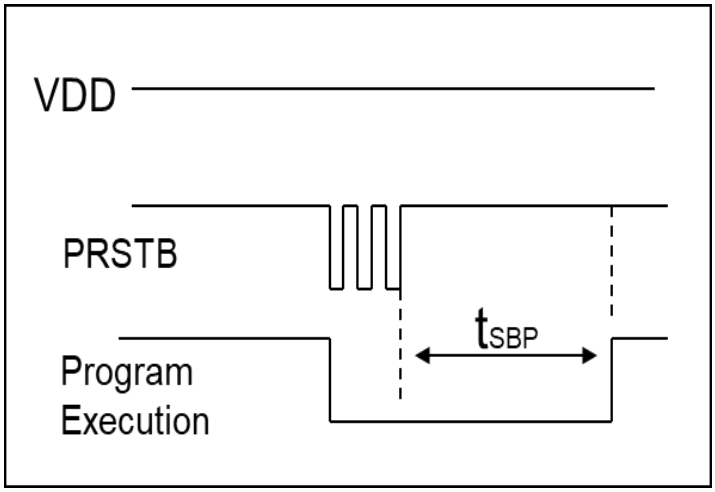


Fig. 5.1: Power-Up Sequence

5.2.1. Timing charts for reset conditions







5.3. Data Memory - SRAM

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

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

For indirect memory access mechanism, the data memory is used as the data pointer to address the data byte. All the data memory could be the data pointer; it's quite flexible and useful to do the indirect memory access. Since the data width is 8-bit, all the 96 bytes data memory of PUD310 can be accessed by indirect access mechanism.

5.4. Oscillator and Clock

There are three oscillator circuits provided by PUD310: internal high RC oscillator (IHRC), internal low RC oscillator (ILRC), and internal ultra-low RC oscillator (NILRC), and IHRC is enabled or disabled by registers clkmd.1 and clkmd.0 is used for ILRC and NILRC switch. User can choose one of these three oscillators as system clock source and use ***clkmd*** register to target the desired frequency as system clock to meet different applications, refer to Table 5.2.

Oscillator Module	Enable/Disable
IHRC	**clkmd**.1
Oscillator Module	**Switch**
ILRC	**clkmd**.0 = 0
NILRC	**clkmd**.0 = 1

Table 5.2: Three oscillation circuits

5.4.1. Internal High RC oscillator and Internal Low RC oscillator

After boot-up, the IHRC and ILRC oscillators are enabled. The frequency of IHRC can be calibrated to eliminate process variation by ***ihrcr*** register; normally it is calibrated to 12MHz. Please refer to the measurement chart for IHRC frequency verse V_{DD} and IHRC frequency verse temperature.

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

5.4.2. Chip calibration

The IHRC frequency and bandgap reference voltage may be different chip by chip due to manufacturing variation, PUD310 provide the IHRC frequency calibration to eliminate this variation, and this function can be selected when compiling user's program and the command will be inserted into user's program automatically. The calibration command is shown as below:

.ADJUST_IC SYSCLK=IHRC/(**p1**), IHRC=(**p2**)MHz, V_{DD}=(**p3**)V;

Where, **p1**=2, 4, 8; In order to provide different system clock.
p2=1.5~12; In order to calibrate the chip to different frequency, 12MHz is the usually one.
p3=1.8~5.5; In order to calibrate the chip under different supply voltage.

5.4.3. IHRC Frequency Calibration and System Clock

During compiling the user program, the options for IHRC calibration and system clock are shown as Table 5.3:

SYSCLK	CLKMD	IHRCR	Description
○ Set IHRC / 1	= 22h (IHRC / 1)	Calibrated	IHRC calibrated to 12MHz, CLK=12MHz (IHRC/1)
○ Set IHRC / 2	= 62h (IHRC / 2)	Calibrated	IHRC calibrated to 12MHz, CLK=6MHz (IHRC/2)
○ Set IHRC / 4	= A2h (IHRC / 4)	Calibrated	IHRC calibrated to 12MHz, CLK=3MHz (IHRC/4)
○ Set IHRC / 8	= E2h (IHRC / 8)	Calibrated	IHRC calibrated to 12MHz, CLK=1.5MHz (IHRC/8)
○ Set ILRC	= 02h (ILRC / 1)	Calibrated	IHRC calibrated to 12MHz, CLK=ILRC
○ Disable	No change	No Change	IHRC not calibrated, CLK not changed

Table 5.3: Options for IHRC Frequency Calibration

Usually, .ADJUST_IC will be the first command after boot up, in order to set the target operating frequency whenever starting the system. The program code for IHRC frequency calibration is executed only one time that occurs in writing the codes into OTP memory; after then, it will not be executed again. If the different option for IHRC calibration is chosen, the system status is also different after boot. The following shows the status of PUD310 for different option:

(1) .ADJUST_IC SYSCLK=IHRC/1, IHRC=12MHz, V_{DD}=5V

After boot up, RSTC = 0x04, CLKMD = 0x22：

- IHRC frequency is calibrated to 12MHz@V_{DD}=5V and IHRC module is enabled
- System CLK = IHRC/1 = 12MHz
- Watchdog timer is disabled, ILRC is selected, PA5 is in input mode

(2) .ADJUST_IC SYSCLK=IHRC/2, IHRC=12MHz, V_{DD}=3.3V

After boot up, RSTC = 0x04, CLKMD = 0x62：

- IHRC frequency is calibrated to 12MHz@V_{DD}=3.3V and IHRC module is enabled
- System CLK = IHRC/2 = 6MHz
- Watchdog timer is disabled, ILRC is selected, PA5 is in input mode

(3) .ADJUST_IC SYSCLK=IHRC/4, IHRC=12MHz, V_{DD}=2.5V

After boot up, RSTC = 0x04, CLKMD = 0xA2：

- IHRC frequency is calibrated to 12MHz@V_{DD}=2.5V and IHRC module is enabled
- System CLK = IHRC/4 = 3MHz
- Watchdog timer is disabled, ILRC is selected, PA5 is in input mode

(4) .ADJUST_IC SYSCLK=IHRC/8, IHRC=12MHz, V_{DD}=2.5V

After boot up, RSTC = 0x04, CLKMD = 0xE2：

- IHRC frequency is calibrated to 12MHz@V_{DD}=2.5V and IHRC module is enabled
- System CLK = IHRC/8 = 1.5MHz
- Watchdog timer is disabled, ILRC is selected, PA5 is in input mode

(5) .ADJUST_IC SYSCLK=ILRC, IHRC=12MHz, V_{DD}=5V

After boot up, RSTC = 0x04, CLKMD = 0x02：

- IHRC frequency is calibrated to 12MHz@V_{DD}=5V and IHRC module is enabled
- System CLK = ILRC = 32KHz
- Watchdog timer is disabled, ILRC is selected, PA5 is in input mode

(6) .ADJUST_IC DISABLE

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

- IHRC is not calibrated
- System CLK = ILRC
- Watchdog timer is enabled, ILRC is selected, PA5 is in input mode

5.4.4. System Clock and LVR level

The clock source of system clock comes from IHRC, ILRC, and NILRC, the hardware diagram of system clock in the PUD310 is shown as Fig. 5.2.

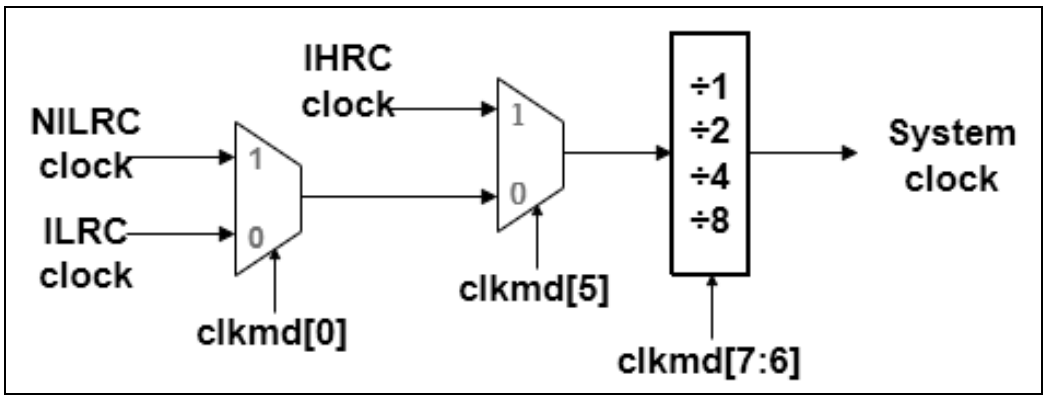


Fig. 5.2: Options of System Clock

User can choose different operating system clock depends on its requirement; the selected operating system clock should be combined with supply voltage and LVR level to make system stable. The LVR level will be selected during compilation. Please refer to Section 4.1.

5.4.5. System Clock Switching

After IHRC calibration, user may want to switch system clock to a new frequency or may switch system clock at any time to optimize the system performance and power consumption. Basically, the system clock of PUD310 can be switched among IHRC, ILRC and NILRC by setting the ***clkmd*** register at any time; system clock will be the new one after writing to ***clkmd*** register immediately. Please notice that the original clock module can NOT be turned off at the same time as writing command to ***clkmd*** register. The examples are shown as below and more information about clock switching, please refer to the "Help" → "Application Note" → "IC Introduction" → "Register Introduction" → CLKMD".

Case 1: Switching system clock from ILRC to IHRC/2

```
...
CLKMD.1     =   1;      //  turn on IHRC first to improve anti-interference ability
CLKMD       =   0x62 ;  //  switch to IHRC/2
...
```

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

```
...
CLKMD       =   0x02 ;  //  switch to ILRC, IHRC CAN NOT be disabled here
CLKMD.1     =   0 ;     //  IHRC CAN be disabled at this time
...
```

Case 3: Switching system clock from IHRC/2 to IHRC/4

```
...                     //  system clock is IHRC/2, ILRC is enabled here
CLKMD       =   0XA2 ;  //  switch to IHRC/4
...
```

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

```
...                     //  system clock is IHRC/2
CLKMD       =   0x00 ;  //  CAN NOT switch clock from IHRC/2 to ILRC and
                        //  turn off IHRC oscillator at the same time
```

5.5 16-bit Timer (Timer16)

A 16-bit hardware timer (Timer16) is implemented in the PUD310, the clock sources of Timer16 may come from system clock (CLK), internal high RC oscillator (IHRC), internal low RC oscillator (ILRC), and PA0, a multiplex is used to select clock output for the clock source. Before sending clock to the counter16, a pre-scaling logic with divided-by-1, 4, 16, and 64 is used for wide range counting. The 16-bit counter performs up-counting operation only, the counter initial values can be stored from memory by *stt16* instruction and the counting values can be loaded to memory by *ldt16* instruction. A selector is used to select the interrupt condition of Timer16, whenever overflow occurs, the Timer16 interrupt can be triggered. The hardware diagram of Timer16 is shown as Fig. 5.3. The interrupt source of Timer16 comes from one of bit 8 to 15 of 16-bit counter, and the interrupt type can be rising edge trigger or falling edge trigger which is specified in the bit 4 of ***integ*** register (address 0x09).

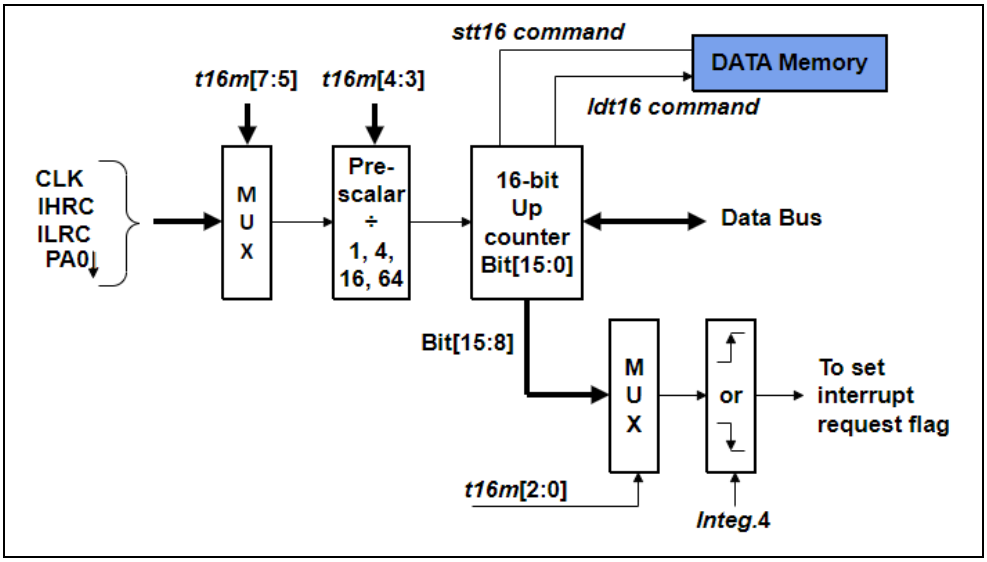


Fig. 5.3: Hardware diagram of Timer16

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

```
T16M        IO_RW        0x0B
$ 7~5: STOP, SYSCLK, X, X, IHRC, X, ILRC, PA0_F        // 1st par.
$ 4~3: /1, /4, /16, /64                                // 2nd par.
$ 2~0: BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15    // 3rd par.
```

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

```
$ T16M SYSCLK, /64, BIT15;
    // choose (SYSCLK/64) as clock source, every 2^16 clock to set INTRQ.2=1
    // if using System Clock = IHRC / 2 = 6 MHz
    // SYSCLK/64 = 6 MHz/64 = 93.75KHz, about every 700 mS to generate INTRQ.2=1

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

$ T16M PA0_F, /1, BIT8;
    // choose PA0 as clock source, every 2^9 to generate INTRQ.2=1
    // receiving every 512 times PA0 to generate INTRQ.2=1

$ T16M STOP;
    // stop Timer16 counting
```

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

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

Where, F is the frequency of selected clock source to Timer16;
P is the selection of t16m [4:3]; (1, 4, 16, 64)
N is the n^{th} bit selected to request interrupt service, for example: n=10 if bit 10 is selected.

5.6 8-bit Timer (Timer2) with PWM generation

An 8-bit hardware timer (Timer2) with PWM generation is implemented in the PUD310. Please refer to Fig. 5.4 shown the hardware diagram of Timer2, the clock sources of Timer2 may come from internal high RC oscillator (IHRC), 2x internal high RC oscillator (IHRC*2), and internal low RC oscillator (ILRC). Bit [5:4] of register tpw2c are used to select the clock of Timer2 PWM. If IHRC is selected for Timer2 clock source, the clock sent to Timer2 will keep running when using ICE in halt state. The output of Timer2 can be sent to pin PA3, depending on bit [6:5] of tpw2s register. A clock pre-scaling module is provided with divided-by- 1, 4, 16, and 64 options, controlled by bit [3:2] of tpw2c register; one scaling module with divided-by-1~32 is also provided and controlled by bit [4:0] of tpw2s register. In conjunction of pre-scaling function and scaling function, the frequency of Timer2 PWM clock (TPW2_CLK) can be wide range and flexible.

The Timer2 counter performs 8-bit up-counting operation only. The 8-bit counter will be clear to zero automatically when its values reach for upper bound register in period mode. The upper bound register is used to define the period of timer or duty of PWM. There are two operating modes for Timer2: period mode and PWM mode; period mode is used to generate periodical output waveform or interrupt event; PWM mode is used to generate PWM output waveform with optional 6-bit, 7-bit or 8-bit PWM resolution, Fig. 5.5 shows the timing diagram of Timer2 for both period mode and PWM mode.

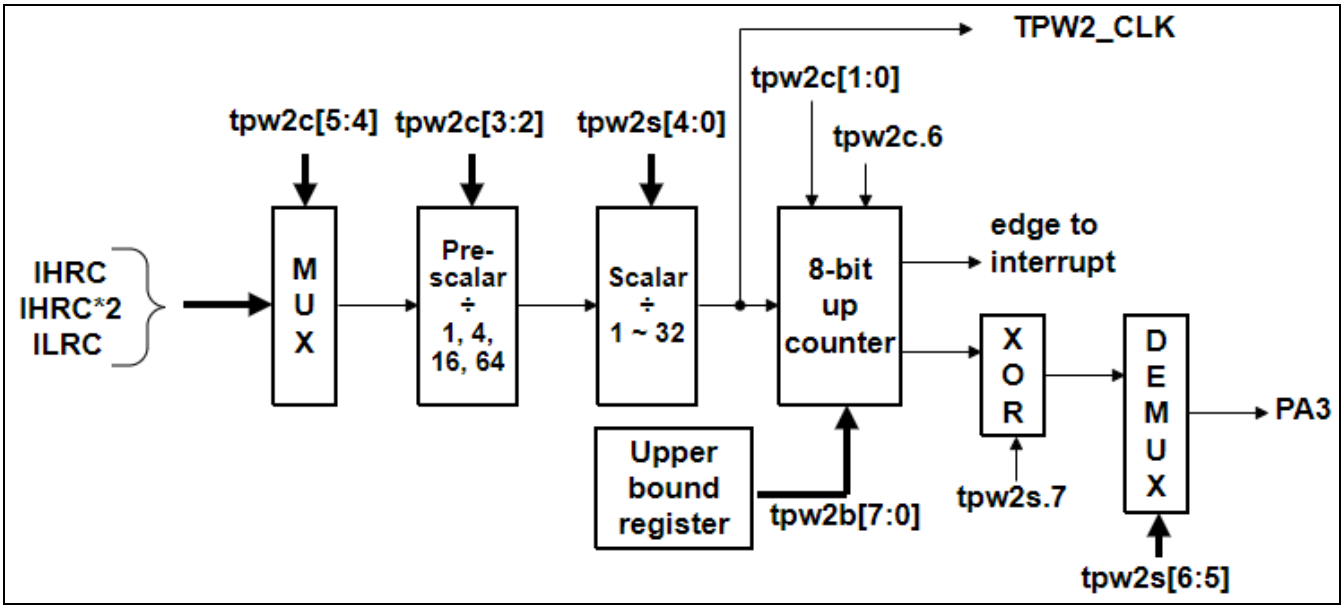
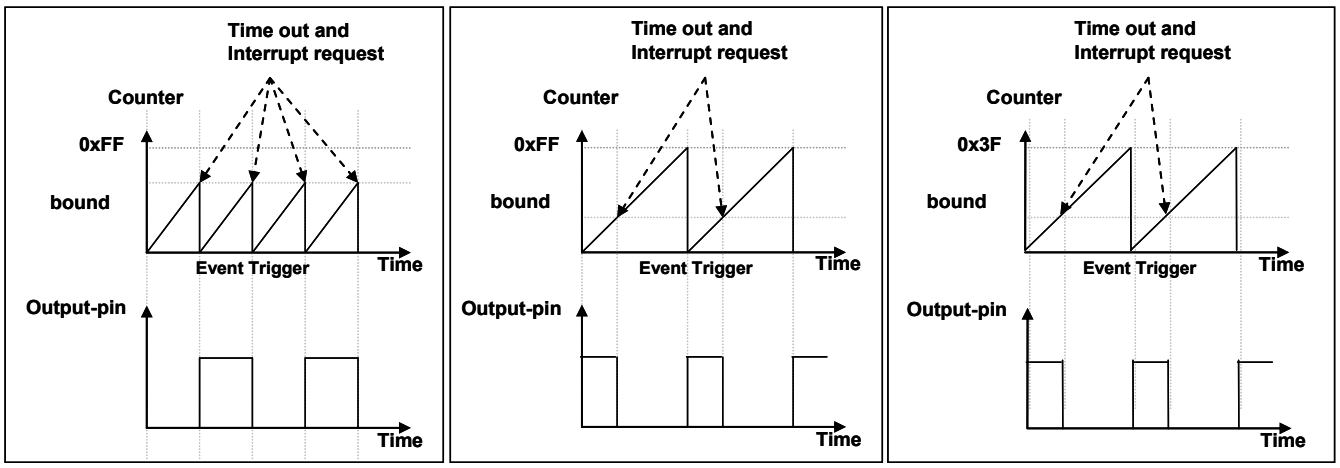


Fig. 5.4: Timer2 hardware diagram



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

Fig. 5.5: Timing diagram of Timer2 in period mode and PWM mode (tpw2c.6=1)

5.6.1 Using the Timer2 to generate periodical waveform

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

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

Where, Y = tpw2c[5:4] : frequency of selected clock source

K = tpw2b[7:0] : bound register in decimal

S1 = tpw2c[3:2] : pre-scalar (S1=1, 4, 16, 64)

S2 = tpw2s[4:0] : scalar register in decimal (S2=0 ~ 31)

Example 1:

tpw2c = 0b0000_0000, Y=12MHz, S1=1
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ [2 × (127＋1) × 1 × (0＋1)] = 46.875KHz

Example 2:

tpw2c = 0b0000_1100, Y=12MHz, S1=64
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_11111, S2 = 31
➔ frequency of output = 12MHz ÷ (2 × (127＋1) × 64 × (31＋1)) =22.89Hz

Example 3:

tpw2c = 0b0000_0000, Y=12MHz, S1=1
tpw2b = 0b0000_1111, K=15
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ (2 × (15＋1) × 1 × (0＋1)) = 375KHz

Example 4:

tpw2c = 0b0000_0000, Y=12MHz, S1=1
tpw2b = 0b0000_0001, K=1
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ (2 × (1＋1) × 1 × (0＋1)) =3MHz

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

```
Void   FPPA0 (void)
{
    . ADJUST_IC    SYSCLK=IHRC/2, IHRC=12MHz, VDD=5V
    ...
    tpw2c = 0x0;
    tpw2b = 0x7f;
    tpw2s = 0b0_10_00001;              //    output=PA3, scalar = 2
    tpw2c = 0b1000_00_00;              //    period mode, IHRC, pre-scalar = 1
    while(1)
    {
        nop;
    }
}
```

5.6.2 Using the Timer2 to generate 8-bit PWM waveform

If 8-bit PWM mode is selected, it should set ***tpw2c***[6]=1 and ***tpw2c***[1:0]=2'b00, the frequency and duty cycle of output waveform can be summarized as below:

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

Duty of Output = [(K＋1) ÷ 256]×100%

Where, Y = tpw2c[5:4] : frequency of selected clock source

K = tpw2b[7:0] : bound register in decimal

S1= tpw2c[3:2] : pre-scalar (S1=1, 4, 16, 64)

S2 = tpw2s[4:0] : scalar register in decimal (S2=0 ~ 31)

Example 1:

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ (256 × 1 × (0+1)) = 46.875KHz
➔ duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 2:

tpw2c = 0b0100_1100, Y=12MHz, S1=64
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_11111, S2=31
➔ frequency of output = 12MHz ÷ (256 × 64 × (31+1)) = 22.89Hz
➔ duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 3:

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b1111_1111, K=255
tpw2s = 0b010_00000, S2=0
➔ PWM output keep high
➔ duty of output = [(255+1) ÷ 256] × 100% = 100%

Example 4:

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b0000_1001, K = 9
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ (256 × 1 × (0+1)) = 46.875KHz
➔ duty of output = [(9+1) ÷ 256] × 100% = 3.9%

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

```
void    FPPA0 (void)
{
    .ADJUST_IC      SYSCLK=IHRC/2, IHRC=12MHz, VDD=5V
    wdreset;
    tpw2c = 0x0;
    tpw2b = 0x7f;
    tpw2s = 0b0_10_00001;           //      output=PA3, scalar = 2
    tpw2c = 0b1100_00_00;           //      PWM mode , IHRC, pre-scalar = 1
                                    //      8-bit PWM
    while(1)
    {
        nop;
    }
}
```

5.6.3 Using the Timer2 to generate 6-bit PWM waveform

If 6-bit PWM mode is selected, it should set ***tpw2c***[6]=1 and ***tpw2c***[1:0]=2'b10, the frequency and duty cycle of output waveform can be summarized as below:

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

Duty of Output = [(K＋1) ÷ 64] × 100%

Where, tpw2c[5:4] = Y : frequency of selected clock source
tpw2b[7:0] = K : bound register in decimal
tpw2c[3:2] = S1 : pre-scalar (S1=1, 4, 16, 64)
tpw2s[4:0] = S2 : scalar register in decimal (S2=0 ~ 31)

Example 1:

tpw2c = 0b0100_0010, Y=12MHz, S1=1
tpw2b = 0b0001_1111, K = 31
tpw2s = 0b010_00000, S2=0
➔ frequency of output = 12MHz ÷ (64 × 1 × (0+1)) = 187.5KHz
➔ duty = [(31+1) ÷ 64] × 100% = 50%

Example 2:

tpw2c = 0b0100_1110, Y=12MHz, S1=64
tpw2b = 0b0001_1111, K = 31
tpw2s = 0b010_11111, S2=31
➔ frequency of output = 12MHz ÷ (64 × 64 × (31+1)) = 91.55 Hz
➔ duty of output = [(31+1) ÷ 64] × 100% = 50%

Example 3:

tpw2c = 0b0100_0010, Y=12MHz, S1=1
tpw2b = 0b0011_1111, K=63
tpw2s = 0b010_00000, S2=0
➔ PWM output keep high
➔ duty of output = [(63+1) ÷ 64] × 100% = 100%

Example 4:

tpw2c = 0b0100_0010, Y=12MHz, S1=1

tpw2b = 0b0000_0000, K=0

tpw2s = 0b010_00000, S2=0

➔ frequency = 12MHz ÷ (64 × 1 × (0+1)) = 187.5KHz

➔ duty = [(0+1) ÷ 64] × 100% =1.5%

5.6.4 PWM Waveform

A PWM output waveform (Fig. 5.6) has a time-base (T_{Period} = Time of Period) and a time with output high level (Duty Cycle). The frequency of the PWM output is the inverse of the period ($f_{PWM} = 1/T_{Period}$).

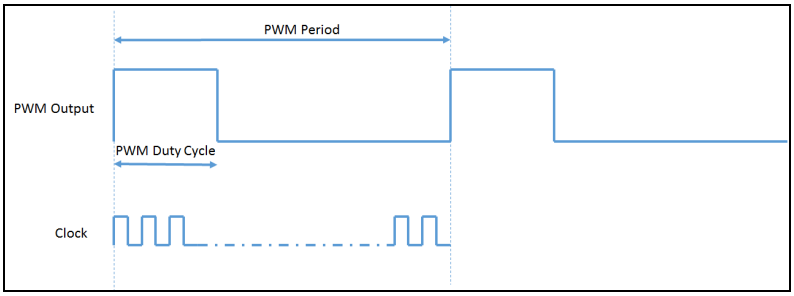


Fig. 5.6: PWM Output Waveform

5.7 WatchDog Timer

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

- 8k ILRC clocks period if register rstc[5:4]=00 (default)
- 16k ILRC clocks period if register rstc[5:4]=01
- 64k ILRC clocks period if register rstc[5:4]=10
- 256k ILRC clocks period if register rstc[5:4]=11

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

When WDT is timeout, PUD310 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig. 5.7.

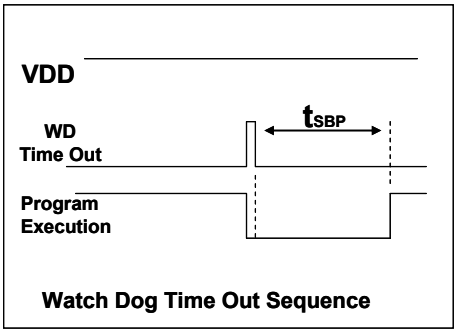


Fig. 5.7: Sequence of Watch Dog Time Out

5.8 Interrupt

There are eight interrupt lines for PUD310:

- External interrupt PA0
- PD_PHY interrupt
- Timer16 interrupt
- TPW2 interrupt

Every interrupt request line has its own corresponding interrupt control bit to enable or disable it; the hardware diagram of interrupt function is shown as Fig. 5.8. All the interrupt request flags are set by hardware and cleared by writing ***intrq*** register. When the request flags are set (External PA0 and Timer16 interrupt only), it can be rising edge, falling edge or both, depending on the setting of register ***integs***. All the interrupt request lines are also controlled by ***engint*** instruction (enable global interrupt) to enable interrupt operation and ***disgint*** instruction (disable global interrupt) to disable it.

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

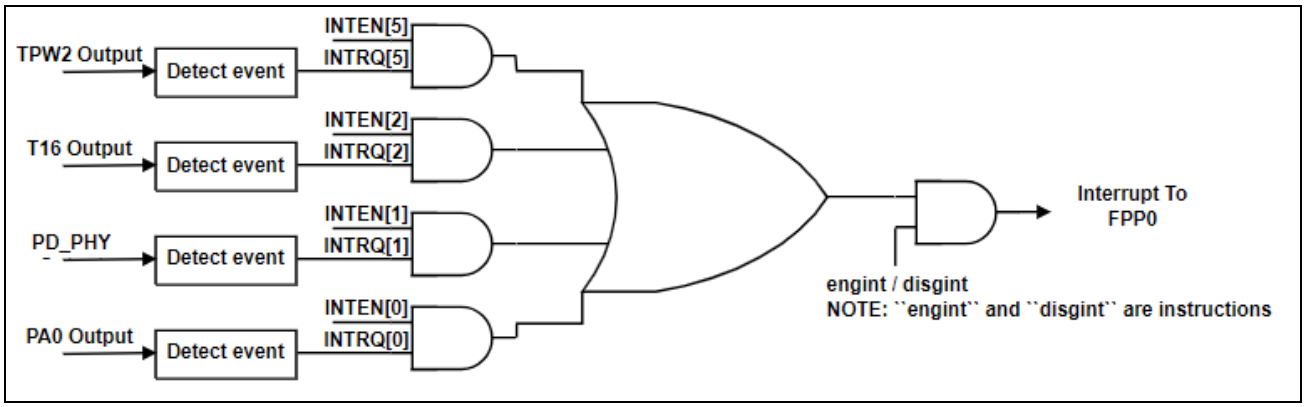


Fig. 5.8: Hardware diagram of interrupt controller

Once the interrupt occurs, its operation will be:

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

During the interrupt service routine, the interrupt source can be determined by reading the ***intrq*** register.
Note: Even if INTEN=0, INTRQ will be still triggered by the interrupt source.

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

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

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

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

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

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

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

  If (INTRQ.PA0)
  {                                  // Here for PA0 interrupt service routine
              INTRQ.PA0 = 0;         // Delete corresponding bit (take PA0 for example)
        ...
  }
    ...
    // X : INTRQ = 0;          // It is not recommended to use INTRQ = 0 to clear all at the end of the
                               // interrupt service routine.
                               // It may accidentally clear out the interrupts that have just occurred
                               // and are not yet processed.
POPAF                     // restore ALU and FLAG register
}
```

5.9 Power-Save and Power-Down

There are three operational modes defined by hardware: ON mode, Power-Save mode and Power-Down modes. ON mode is the state of normal operation with all functions ON, Power-Save mode ("***stopexe***") is the state to reduce operating current and CPU keeps ready to continue, Power-Down mode ("***stopsys***") is used to save power deeply. Therefore, Power-Save mode is used in the system which needs low operating power with wake-up occasionally and Power-Down mode is used in the system which needs power down deeply with seldom wake-up. Table 5.4 shows the differences in oscillator modules between Power-Save mode ("***stopexe***") and Power-Down mode ("***stopsys***").

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

Table 5.4: Differences in oscillator modules between STOPSYS and STOPEXE

5.9.1 Power-Save mode ("*stopexe*")

Using "***stopexe***" instruction to enter the Power-Save mode, only system clock is disabled, remaining all the oscillator modules active. For CPU, it stops executing; however, for Timer16, counter keep counting if its clock source is not the system clock. Wake-up from input pins can be considered as a continuation of normal execution, the detail information for Power-Save mode shows below:

- IHRC oscillator modules: No change, keep active if it was enabled.
- ILRC oscillator modules: must remain enabled, need to start with ILRC when be wakening up.
- System clock: Disable, therefore, CPU stops execution.
- OTP memory is turned off.
- Timer counter: Stop counting if its clock source is system clock or the corresponding oscillator module is disabled; otherwise, it keeps counting. (The Timer contains TM16, TPW2.)
- Wake-up sources:
  a. IO toggle wake-up: IO toggling in digital input mode (*PxC* bit is 0 and *PxDIER* bit is 1)
  b. Timer wake-up: If the clock source of Timer is not the SYSCLK, the system will be awakened when the Timer counter reaches the set value, it is awakened on both the rising and falling edges.

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

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

…
```

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

5.9.2 Power-Down mode ("*stopsys*")

Power-Down mode is the state of deeply power-saving with turning off all the oscillator modules. The following shows the internal status of PUD310 detail when "***stopsys***" command is issued:

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

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

```
CLKMD    =    0x22;      //    Change clock from IHRC to ILRC
CLKMD.1  =    0;         //    disable IHRC
...
while (1)
{
         STOPSYS;       //    enter power-down
         if (...) break; //    if wakeup happen and check OK, then return to high speed,
                        //    else stay in power-down mode again
}
CLKMD    =    0x62;      //    Change clock from ILRC to IHRC/2
```

5.9.3 Wake-up

After entering the Power-Down or Power-Save modes, the PUD310 can be resumed to normal operation by toggling IO pins. Wake-up from timer are available for Power-Save mode ONLY. Table 5.5 shows the differences in wake-up sources between STOPSYS and STOPEXE.

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

Table 5.5: Differences in wake-up sources between Power-Save mode and Power-Down mode

When using the IO pins to wake-up the PUD310, the IO pins must bigger than 1 operation instruction and registers ***pxdier*** should be properly set to enable the wake-up function for every corresponding pin. The time for normal wake-up is about 1024 ILRC clocks counting from wake-up event; fast wake-up can be selected to reduce the wake-up time by ***rstc*** register, and the time for fast wake-up is about 16 ILRC clocks from IO toggling, refer to Table 5.6.

Suspend mode	Wake-up mode	Wake-up time (t_{WUP}) from IO toggle
STOPEXE suspend or STOPSYS suspend	Fast wake-up	16 * T_{ILRC}, Where T_{ILRC} is the period of ILRC
STOPEXE suspend or STOPSYS suspend	Normal wake-up	1024 * T_{ILRC}, Where T_{ILRC} is the clock period of ILRC

Table 5.6: Differences in wake-up time between Fast/Normal wake-up

5.10 IO Pins

All the pins can be independently set into two states output or input by configuring the data registers (***pa***, ***pb***), control registers (***pac***, ***pbc***) and pull-high resistor (***paph***, ***pbph***). All these pins have Schmitt-trigger input buffer and output driver with CMOS level. When it is set to output low, the pull-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 5.7 shows the configuration table of bit 0 of port A. The hardware diagram of IO buffer is also shown as Fig. 5.9.

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 5.7: PA0 Configuration Table

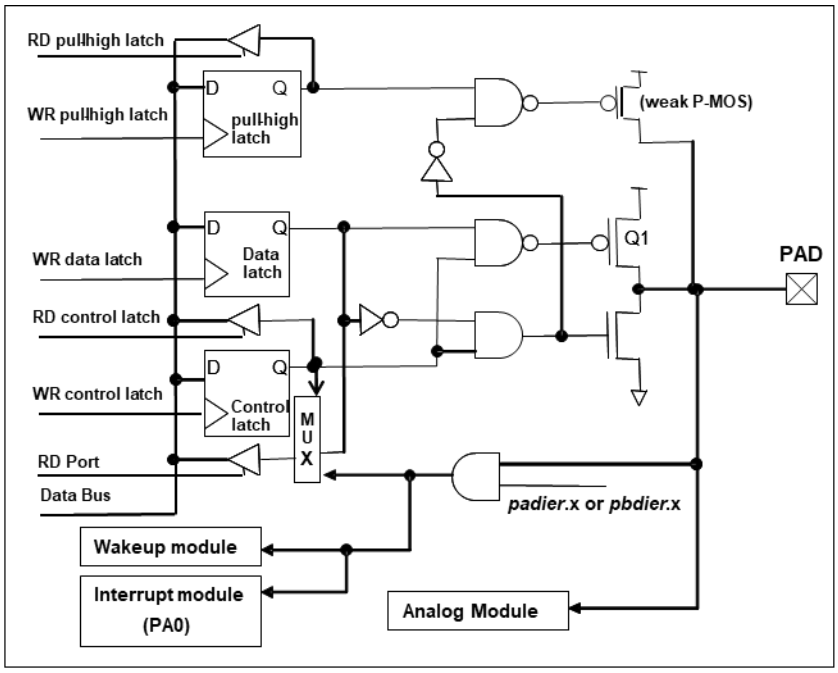


Fig. 5.9: Hardware diagram of IO buffer

Except than PA7, PA1, PA2, PA4, PA6, PA7 can be open-drain ONLY when setting to output mode; PA1, PA2, PA4, PA6 are PD special IO. Others IO pins have the same structure; The corresponding bits in registers ***padier*** / ***pbdier*** should be set to low to prevent leakage current for those pins are selected to be analog function. When PUD310 is put in power-down or power-save mode, every pin can be used to wake-up system by toggling its state. Therefore, those pins needed to wake-up system must be set to input mode and set the corresponding bits of registers ***padier*** and ***pbdier*** to high**.** The same reason, ***padier***.0 should be set high when PA0 is used as external interrupt pin, ***pbdier***.0 for PB0, and ***padier***.3 for PA3.

5.11 Reset and LVR

5.11.1 Reset

There are many causes to reset the PUD310, once reset is asserted, most of all the registers in PUD310 will be set to default values, system should be restarted once abnormal cases happen, or by jumping program counter to address 0x00. After power-up and LVR reset, the SRAM data will be kept when V_{DD}>V_{DR} (SRAM data retention voltage). However, if SRAM is cleared after power-on again, the data cannot be kept. And, the data memory is in uncertain state when V_{DD}<V_{DR}. The content will be kept when reset comes from PRSTB pin or WDT timeout.

5.11.2 LVR reset

By code option, there are many different levels of LVR for reset. Usually, user selects LVR reset level to be in conjunction with operating frequency and supply voltage.

5.12 PD PHY Controller

5.12.1 Features

PD PHY controller is a USB PD capable sink-end device responsible for sending and receiving messages on the CC wire. The main function executes the PD physical layer signal communications and supplies a simple access interface to system software, which let user can do the most effective management for the protocol layer. In addition, this function block is also responsible for all signal control of Analog Front End (AFE), please see section 5.13 for details.

- Hardware Features:
  - Standard MCU system bus interface for register access.
  - One PD PHY system interrupt comprises TX interrupt, RX interrupt and RX FIFO Half Full interrupt
  - One shared 4 bytes TX/RX FIFO.
  - Supports inner signal monitor capability through GPIOs for software protocol layer development.
- USB PD Physical Layer Features:
  - Transmit and receive only two kinds of PD signals, SOP packet and Hard Reset signaling. SOP', SOP'', and other signaling will be auto omitted by the controller, which reduce the unnecessary data flows between the physical layer and protocol layer.
  - Receiving data capability adjustment for cable and connector application.

5.12.2 Block Diagram

Fig. 5.10 shown below illustrates the block diagram of the PD PHY controller and Analog Front End (AFE) macro. This controller comprises a transmitting and receiving data flow controller (TX/RX CTRL), a data encoder and transmitter (Transmitter), a data decoder and receiver (Receiver), a shared TX/RX CRC generator (CRC32), a shared TX/RX data storage unit (TX/RX FIFO) and a register file (Regfile). Transmitting data path (TX data path) is composed of Transmitter, CRC32 and TX/RX FIFO, while receiving data path (RX data path) is composed of Receiver, CRC32, and TX/RX FIFO. It is noted that CRC32 and TX/RX FIFO are shared by both TX and RX data paths, so data transfer and data receiving are not allowed to operate at the same time.

All related registers of PD PHY controller and AFE macro are allocated at the Regfile. The Regfile supplies one standard MCU system bus interface as register access. The AFE registers in the Regfile are used to manage all operation of the AFE macro. The TX/RX CTRL controls the TX/RX data flow direction, handles all protocol errors, and communicates with application software responsible for managing all transactions from the protocol layer. The TX/RX FIFO holds data to be transferred in the TX operation mode, and stores received data in the RX operation mode. The TX/RX FIFO is a 4-byte storage unit. In the RX operation mode, one half-full interrupt will be asserted when $\geqq$2 bytes have been stored in the FIFO, and will be automatically de-asserted when less than 2 bytes stay in the FIFO by reading received data. It is noted that this half-full interrupt will not function in the TX operation mode. Based on the USB PD specification, the CRC32 is used for generating the checksum at the both TX and RX operation mode.

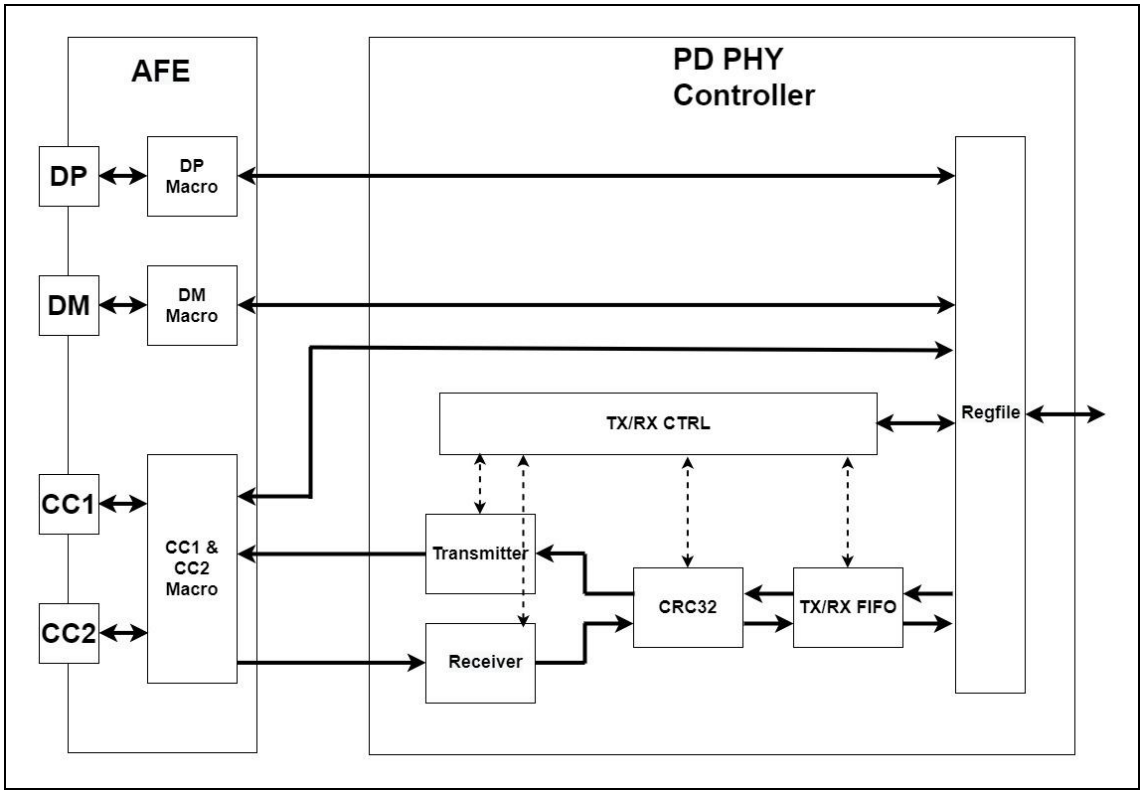


Fig. 5.10: Block Diagram of PD PHY Controller and AFE

For each SOP packet transmission, at first Transmitter generates a 64-bit Preamble automatically and then reads data from the TX/RX FIFO and CRC32 and executes the 4b5b encoding with them. The Transmitter also generates SOP and EOP symbols which will be pre-appended or appended to these encoded data. Finally, do the BMC encoding before sending this SOP packet. Besides, the Transmitter also can generate a Hard Reset signaling described in the USB PD specification. One TX interrupt will be asserted when a SOP transmission has been executed successfully, including Hard Reset signaling. TX interrupt must be cleared by user, please see section 5.12.3.4 for details.

When any signal occurs on the CC wire (from bus idle to bus active), the Receiver will start to try the BMC decoding. User also can acquire the bus idle or active information through reading the specified register. If one legal SOP packet is detected, all data payload and 32-bit CRC after 4b5b decoding will be transferred to the CRC32 and stored in the TX/RX FIFO. The first RX half-full interrupt will be asserted when two bytes have been stored in the FIFO. Whenever the RX half-full interrupt occurs, user must read these received data as soon as possible; otherwise, one FIFO overrun will cause this correct packet to be regarded as one failed transfer. Once the first RX half-full interrupt has been asserted, another RX interrupt will be also generated when the CC wire returns to the bus idle again. Whether the received data is correct or not, user must finish this packet transaction. In addition, if a Hard Reset signaling is received successfully, one RX interrupt can be also asserted. It is noted that the SOP', SOP'' packets and other signaling, including the meaningless signal or noise, will be auto omitted by the Receiver, and no message and interrupt will be generated to application software.

5.12.3 Function Description

This section illustrates all PD PHY controller functions and the TX/RX operation procedures. Some useful information also be mentioned.

5.12.3.1 TX/RX Hardware State Diagram

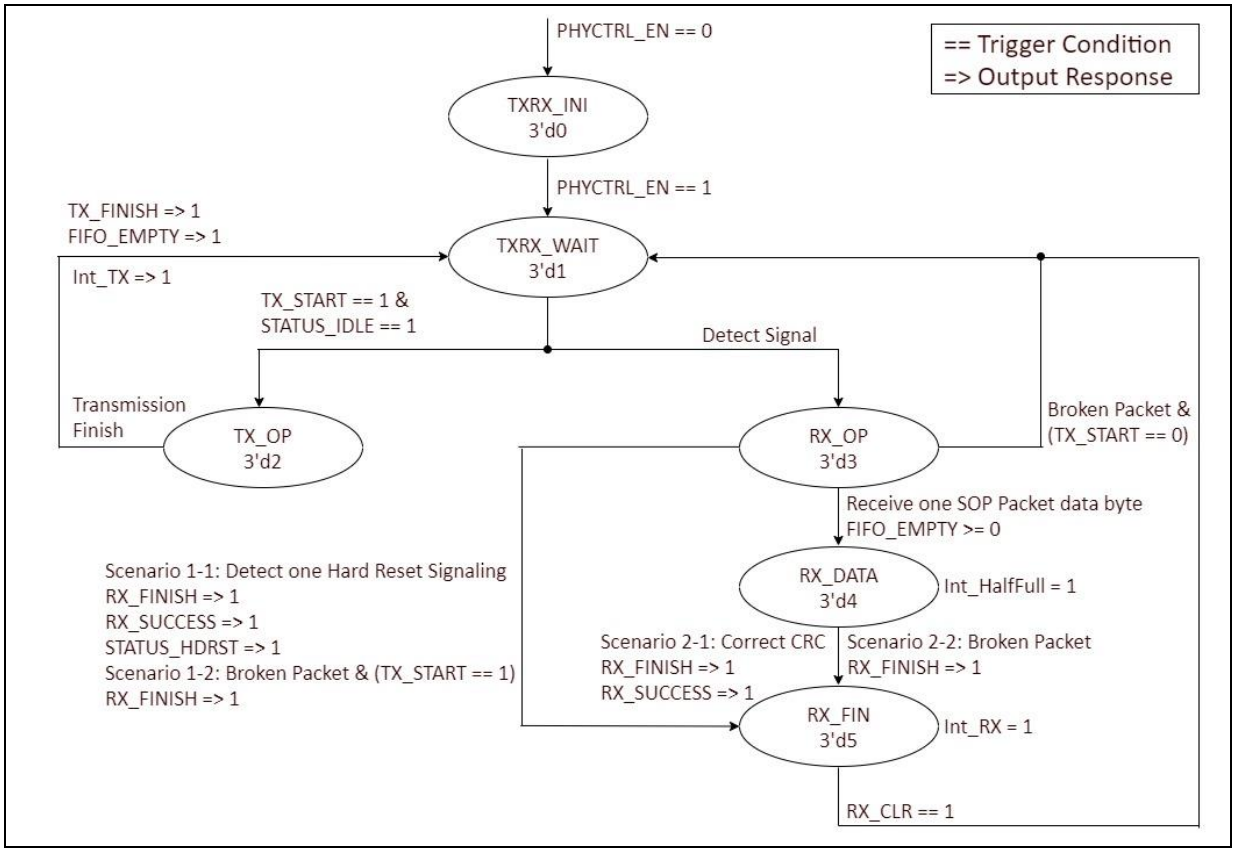


Fig. 5.11: TX and RX Hardware State Diagram

Operation Mode	Signal Type	Comment
TX Operation Mode	SOP packet	
TX Operation Mode	Hard Reset singling	
RX Operation Mode	SOP packet	
RX Operation Mode	Hard Reset singling	

Table 5.8: Accepted Packet and Signaling Types

Fig. 5.11 illustrates the hardware behaviors of the PD PHY controller when sending or receiving data. Table 5.8 describes which signal types can be accepted or operated under the control of the controller.

Before starting any operation, user must set PHYCTRL_EN of pd_txrx register to 1 to enable the controller. Setting PHYCTRL_EN to 0 will force the controller to the default state from any other one, including the PD PHY interrupt and all status flags of pd_stat register. It also can be used as a soft reset of the controller. Set the TX_START of pd_txrx register to 1 to start one new SOP packet transfer, including

one Hard Reset transmission. Once this transmission is complete, the Int_TX interrupt will be automatically asserted. If this transfer is successful, both TX_FINISH and FIFO_EMPTY of pd_stat will be set to 1. It is a rare possibility that when TX_START is enabled, one packet on the CC wire is detecting by the controller at the same time. Basically, the priority of RX operation is higher than TX operation in the hardware algorithm, so this transmission will be auto omitted and TX_FINISH will not be active forever, that is, one unexpected TX/RX collision may occur. This exception could come from a PD protocol error; user must provide one proper algorithm to solve this issue (please see section 5.12.3.7 for details).

When any signal occurs on the CC wire, the controller will try to detect and analyze this received data. The PD PHY controller only accepts the SOP packet and Hard Reset signaling, and other type of received data will be regarded as a meaningless signal, which will be ignored automatically by the controller. If one Hard Reset singling is detected successfully, the Int_RX interrupt will be asserted to tell system program, and the RX_FINISH, RX_SUCCESS and STATUS_HDRST of pd_stat register will be set to 1. If one SOP packet is detected, data payload and 32-bit CRC data will be stored in the FIFO. When two bytes have been received and stored in the FIFO, the first Int_HalfFull interrupt will be asserted until less than two bytes stay in the FIFO by reading the pd_fifo register. Each received data will continue to be stored in the FIFO hereafter. Each time two bytes, including 3 and 4, stay in the FIFO, the Int_HalfFull interrupt will be asserted again. Until the EOP symbol of packet or incorrect data is detected, this reception process will be stopped, and meantime one Int_RX interrupt will be asserted and RX_FINISH is also set to 1. RX_SUCCESS can be used to decide whether one correct packet has been received successfully or not. Finally, user must set the RX_CLR of pd_txrx register to make the controller return back to the initial state again. Now, another TX or RX operation can continue to be executed.

If user want to monitor the state transition and inner signals of the controller in the Fig. 5.11, they also can be probed on the GPIOs (please see section 5.12.3.9 for details).

5.12.3.2 TX Flow Control

Fig. 5.12 describes the basic flow chart of one standard SOP packet transfer, and Fig. 5.13 describe how to transmit one Hard Reset. (NOTE: It is not necessary to follow these flow charts if user want to develop own policy for the application.)

Read the STATUS_IDLE of pd_stat register to check whether the CC wire is bus idle or not, which avoids the collision occurrence of data transmission and reception at the same time. Enable the TX_START of pd_txrx register to start one new transfer, and meantime the controller also starts to generate automatically a 64-bit Preamble on the CC wire. The time of entire Preamble transmission is enough to prepare the transmitted data for software. Write the data payload of a SOP packet to the pd_fifo register, and check the FIFO_FULL of pd_stat register to decide the next data writing. The controller always monitors the FIFO status through the FIFO_EMPTY of pd_stat register during entire data transfer. Once the FIFO is found empty, the controller regards it as no data wanted to be transmitted again. When this transfer is complete, the Int_TX interrupt will be asserted. Check the TX_FINISH of pd_stat register to assure that data transfer has be executed successfully and no exception occurs. Finally, set TX_CLR of pd_txrx register to clear all hardware status and PD PHY interrupt.

The flow chat of a Hard Reset signaling transmission is identical to the one used in the SOP packet, except that no data payload is needed to be transmitted, and SEND_HDRST of pd_txrx register must be set first before the TX_START setting.

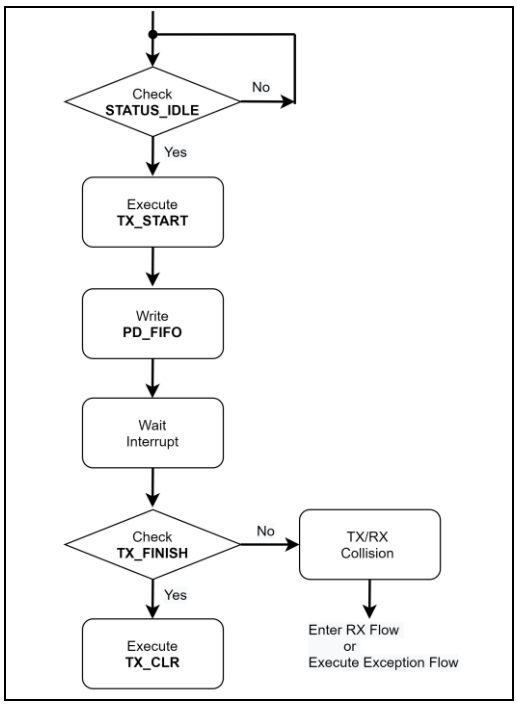


Fig. 5.12: TX Flow Chart (SOP Packet)

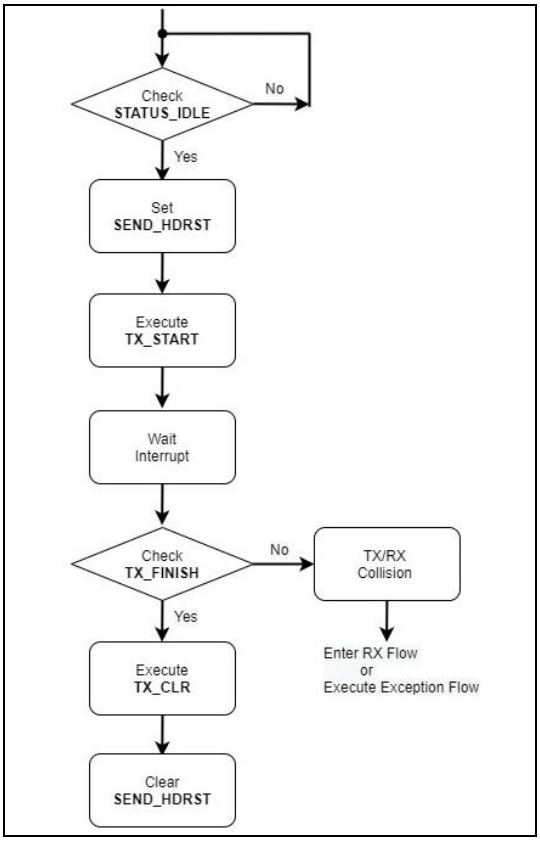


Fig. 5.13: TX Flow Chart (Hard Reset Signaling)

5.12.3.3 RX Flow Control

Fig. 5.14 shown below illustrates the flow chat of a SOP packet or Hard Reset signaling reception. (NOTE: It is not necessary to follow this flow chart if user want to develop own policy for the application.)

If the first PD PHY interrupt occurs, user must check the RX_FINISH of pd_stat register to decide whether it is an Int_RX or Int_HalfFull interrupt. If RX_FINISH is set to 1, it must be Int_RX, otherwise an Int_HalfFull interrupt occurs. If this is an Int_RX interrupt type, then read the RX_SUCCESS of pd_stat register to check whether it is set to 1 or not. If the RX_SUCCESS is set to 1, the received data must be one Hard Reset singling, otherwise it is regarded as a broken packet or signaling and no action need be taken. Also, check the STATUS_HDRST of pd_stat register to confirm that the received data is a Hard-Reset singling.

If the first interrupt is an Int_HalfFull interrupt, it must be a SOP packet. Read received data from the pd_fifo register until the FIFO_EMPTY of pd_stat is back to the default value. Wait the next interrupt occurs, check the RX_FINISH flag and read received data from the FIFO again. Once the final EOP symbol or the broken data has been detected by the controller, the Int_RX interrupt will be asserted and RX_FINISH is set to 1. Check RX_SUCCESS to decide whether the correct packet has been received or not. Finally, read the residual data in the FIFO and set RX_CLR of pd_txrx register to 1 to clear the PD PHY interrupt and all status flags.

Table 5.9 describes the relationship between the status flags and received signal types. According to this table content, it is easy to decide which packet or signaling is receiving by the controller. Table 5.10 describes how to clear the PD PHY interrupt sources, and Table 5.11 shows the clear method of all status flags in the pd_stat register.

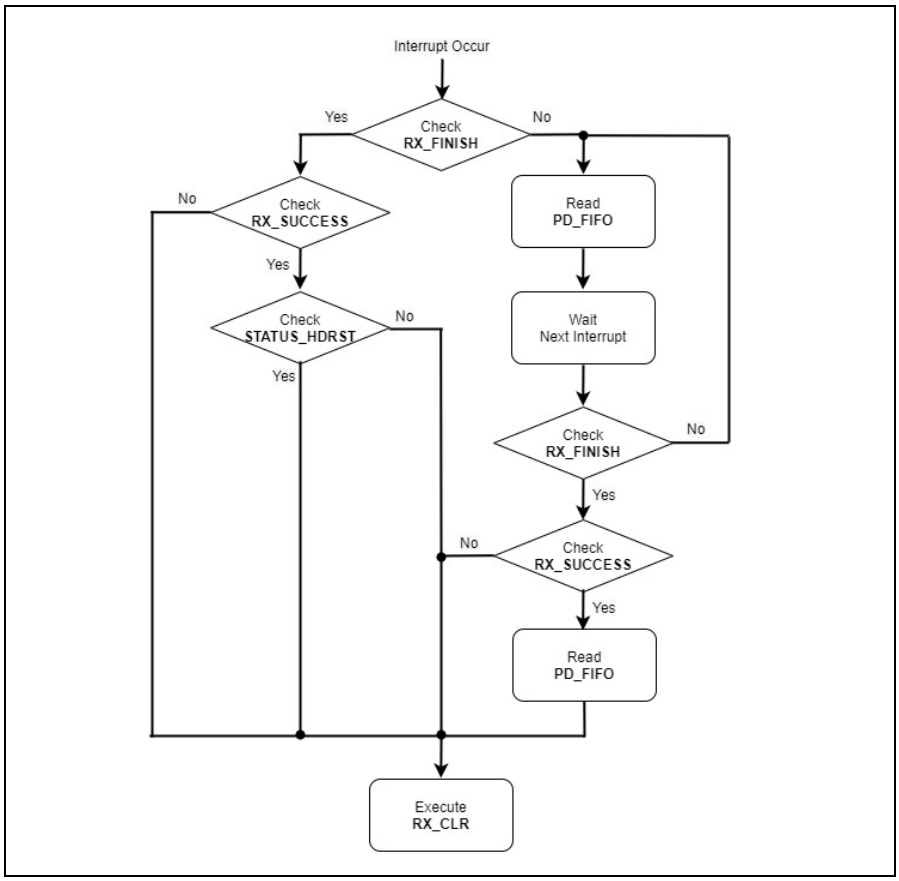


Fig. 5.14: RX Flow Chart

	RX_FINISH	RX_SUCCESS	STATUS_HDRST
SOP Packet	1	1	0
Hard Reset Singling	1	1	1
Broken Packet	1	0	0

Table 5.9: Relationship between Status Flag and Signal Type

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
Int_TX	Clear	Clear	Clear	N.A.
Int_RX	Clear	Clear	Clear	N.A.
Int_HalfFull	Clear	N.A.	Clear	Clear

Table 5.10: PD PHY Interrupt Clear Method

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
TX_FINISH	Clear	Clear	N.A.	N.A.
RX_FINISH	Clear	N.A.	Clear	N.A.
RX_SUCCESS	Clear	N.A.	Clear	N.A.
FIFO_EMPTY	Clear	N.A.	Clear	Clear

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
FIFO_FULL	Clear	N.A.	Clear	Clear
STATUS_HDRST	Clear	N.A.	Clear	N.A.
STATUS_IDLE	Clear	N.A.	N.A.	N.A.

Table 5.11: PD PHY Status Flag Clear Method

5.12.3.4 PD PHY Interrupt Type

PD PHY interrupt is composed of three independent interrupt sources and is a level-trigger mode, shown as Fig. 5.15. The system interrupt enable is defined in the bit1 of inten register. The system interrupt status is defined in the bit1 of intrq register. The interrupt reset method is described as Table 5.10 shown above. It is noted that there is no corresponding flag for Int_TX, Int_RX and Int_HalfFull for the sake of reducing the complexity of operation for software.

PD PHY Interrupt = ***Int_TX*** | ***Int_RX*** | ***Int_HalfFull***:

- ***Int_TX*** : will be active when TX has transmitted one packet or signaling.
- ***Int_RX*** : will be active when RX has received one legal packet or signaling, or the broken packet data.
- ***Int_HalfFull*** : will be active when FIFO has received 2/3/4 bytes at the RX operation mode.

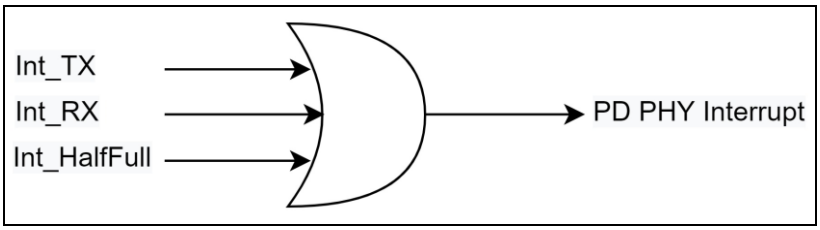


Fig. 5.15: PD PHY Interrupt Type

5.12.3.5 TX/RX FIFO Access

Here are some TX/RX FIFO features:

1. One fixed 4 bytes FIFO.
2. FIFO is shared by both TX and RX data paths, so transmitting and receiving data simultaneously are not allowed.
3. In the TX operation mode, FIFO is only allowed to be written by user; in the RX operation mode, FIFO is only allowed to be read by user.
4. After writing one byte into the FIFO, must set FIFO_WR of pd_txrx register to 1 first, then set 0 (Or, set 0 first, then set 1. It is a rising-edge trigger mode.). Do this operation to update the write index of the FIFO.
5. After reading one byte from the FIFO, must set FIFO_RD of pd_txrx register to 1 first, then set 0 (Or, set 0 first, then set 1. It is a rising-edge trigger mode.). Do this operation to update the read index of the FIFO.
6. The Int_Half Full interrupt is only used and asserted at the RX operation mode.
7. The last four bytes of received data must be the 32-bit CRC data if it is a correct SOP packet.

5.12.3.6 CRC32 Features

The following CRC information comes from the USB PD specification. Please see the related documents for details. CRC-32 is used to protect the data integrity of the data payload. CRC-32 is defined as follows:

- The CRC-32 polynomial shall be = 04C1 1DB7h
- The CRC-32 initial value shall be = FFFF FFFFh
- CRC-32 shall be calculated for all bytes of the payload not inclusive of any packet framing symbols (i.e. excludes the Preamble, SOP*, EOP).
- CRC-32 calculation shall begin at byte 0, bit 0 and continue to bit 7 of each of the bytes of the packet.
- The remainder of CRC-32 shall be complemented
- The residual of CRC-32 shall be C704 DD7Bh

Note: This inversion of the CRC-32 remainder adds an offset of FFFF FFFFh that will create a constant CRC-32 residual of C704 DD7Bh at the receiver side.

5.12.3.7 TX and RX Collision Process

Section 5.12.3.1 refers to the TX/RX collision condition. In the TX operation mode, set TX_START to start one new transmission; however, this transfer is omitted by the controller and the hardware state is force to switch into the RX operation mode. When this condition occurs, host (power source-end) and device (PD PHY controller) might lose the communication synchronization between both, and sometimes the root cause of this PD protocol error is hard to be found.

The USB PD specification presents many solutions to restore to the normal communication between the power source-end and sink-end, but this needs the aids of protocol layer, that is, user must support one robust software algorithm to deal with this exception. In fact, if one unexpected error occurs, do nothing and just wait one Hard Reset signaling from the host. This can make the device restore to its initial state and restart one new transaction between the host and the device.

5.12.3.8 Received Signal Quality Adjustment

The data receiving capability of the PD PHY controller is compliant to the USB PD standard specification. If user wants to adjust the quality of receiving data for the particular application. The controller supports the TH_DEB, TH_HIGH, and TH_LOW of pd_decoder register to meet these requirements.

The TH_DEB can deglitch the unknown signal toggles from some interferences such as spike noise. The BMC decoding principle used at the receiver is that detecting the duration of high/low voltage to decide the “Long One/Zero” or “Short One/Zero”. This “Long” or “Short” decision is judged by the threshold values of both TH_HIGH and TH_LOW. It is noted that the improper setting of these values will cause the occurrence of the BMC decoding error. The default value for the TH_DEB, TH_HIGH, and TH_LOW is typically good enough and do not need further adjustment.

5.12.3.9 Inner Hardware Signal Probe

Fig. 5.11 shown in the section 5.12.3.1 illustrates the state transition of hardware behaviors of the controller and the variation of the corresponding flags. This information is helpful to find the root cause of data communication errors, or is used to develop the protocol layer algorithm. These real-time signals can be probed on the GPIOs. Both PROBE_SEL of pd_ctrl registers and bit 6:4 of opr1 register are used to select these signals to be monitored. Table 5.12 shows the relationship between the probe signals, the register setting values, and the corresponding output GPIO pins. Table 5.13 explains the definition and function of these probe signals. The signals spelled with a capital letter represent the status register name and these signals description are defined in the pd_stat and cc_misc register. The state name and numbering of st_phy[2:0] signal is defined in the Table 5.14 shown below.

Bit 6:4 of opr1 = 001	PB0	PA5	PA3	PA0
PROBE_SEL = 00	int_pd_phy	st_phy[2]	st_phy[1]	st_phy[0]
PROBE_SEL = 01	fifo_ov	STATUS_HDRST	RX_SUCCESS	RX_FINISH
PROBE_SEL = 10	FIFO_FULL	FIFO_EMPTY	TX_FINISH	STATUS_IDLE
PROBE_SEL = 11	crc_cmp_out	sym_eop	pktdet_fail	BMC_BIT

Table 5.12: Probe Signal of PD PHY Controller

Probe Signal	Description
st_phy[2:0]	State machine name in the Fig. 5.11: TX and RX Hardware State Diagram
int_pd_phy	PD PHY controller interrupt
fifo_ov	FIFO overrun
crc_cmp_out	32-bit CRC check output; 1: received data without errors, 0: CRC compare error
sym_eop	Find a EOP symbol
pktdet_fail	The received data cannot be decoded correctly.

Table 5.13: Probe Signal Definition

st_phy[2:0]	0	1	2	3	4	5
State Name	TXRX_INI	TXRX_WAIT	TX_OP	RX_OP	RX_DATA	RX_FIN

Table 5.14: State Name and Numbering of st_phy[2:0]

5.12.4 Programming Sequence

This section scribes how to execute the proper register setting to control the PD PHY controller to transmit or receive data. These examples can be used as the basic programming sequences; however, according to the applications, user also can develop own program code to do the more efficient operation of the controller.

5.12.4.1 Transmit one SOP Packet

The following programming sequence transmits one SOP packet with interrupt:

Step 1 Set ***PHYCTRL_EN*** of ***pd_txrx*** register to enable the PD PHY controller.

Step 2 Wait for CC wire idle by looping until ***STATUS_IDLE*** of ***pd_stat*** register becomes 1.

Step 3 Set ***TX_START*** of ***pd_txrx*** register to start a new transfer.

Step 4 Wait for FIFO not full by looping until ***FIFO_FULL*** of ***pd_stat*** register becomes 0.

Step 5 Write one byte of data to the ***pd_fifo*** register.

Step 6 Set "1" first, then set "0" to ***FIFO_WR*** of ***pd_txrx*** register to update write index of FIFO.

Step 7 If there are more data to send, go to Step 4.

Step 8 Wait for an occurrence of PD PHY interrupt.

Step 9 Check ***TX_FINISH*** of ***pd_stat*** register:

1. ***TX_FINISH*** = 1: This transmission is successful.
2. ***TX_FINISH*** = 0: This transmission is auto omitted.

Step 10 Set ***TX_CLR*** of ***pd_txrx*** register to clear PD PHY interrupt and all hardware status.

5.12.4.2 Transmit one Hard Reset Signaling

The following programming sequence transmits one Hard Reset signaling with interrupt:

Step 1 Set ***PHYCTRL_EN*** of ***pd_txrx*** register to enable the PD PHY controller.

Step 2 Wait for CC wire idle by looping until ***STATUS_IDLE*** of ***pd_stat*** register becomes 1.

Step 3 Set ***SEND_HDRST*** of ***pd_txrx*** register to enable Hard Reset transmission.

Step 4 Set ***TX_START*** of ***pd_txrx*** register to start a new transmission.

Step 5 Wait for an occurrence of PD PHY interrupt.

Step 6 Check ***TX_FINISH*** of ***pd_stat*** register:

1. ***TX_FINISH*** = *1: This transmission is successful.*
2. ***TX_FINISH*** = *0: This transmission is auto omitted.*

Step 7 Set ***TX_CLR*** of ***pd_txrx*** register to clear PD PHY interrupt and all hardware status.

Step 8 Clear ***SEND_HDRST*** of ***pd_txrx*** register to disable Hard Reset transmission.

5.12.4.3 Receive one SOP Packet

The following programming sequence receives one SOP packet with interrupt:

Step 1 Set ***PHYCTRL_EN*** of ***pd_txrx*** register to enable the PD PHY controller.

Step 2 Wait for an occurrence of PD PHY interrupt.

Step 3 Read one byte of data from the ***pd_fifo*** register.

Step 4 Set "1" first, then set "0" to ***FIFO_RD*** of ***pd_txrx*** register to update read index of FIFO.

Step 5 Go to Step 3 to read more received data if ***FIFO_EMPTY*** of ***pd_stat*** register is not "1".
(Note: The last four bytes are 32-bit CRC data)

Step 6 Go to Step 2 to wait for the next interrupt if ***RX_FINISH*** of ***pd_stat*** register is not "1".

Step 7 Check ***RX_SUCCESS*** of ***pd_stat*** register:

1. ***RX_SUCCESS*** = *1: Receive a legal packet.*
2. ***RX_SUCCESS*** = *0: Receive a broken packet.*

Step 8 Set ***RX_CLR*** of ***pd_txrx*** register to clear PD PHY interrupt and all hardware status.

5.12.4.4 Receive one Hard Reset Signaling

The following programming sequence receives one Hard Reset with interrupt:

Step 1 Set ***PHYCTRL_EN*** of ***pd_txrx*** register to enable the PD PHY controller.

Step 2 Wait for an occurrence of PD PHY interrupt.

Step 3 Check ***STATUS_HDRST*** of ***pd_stat*** register:

1. ***STATUS_HDRST*** = 1: Receive one Hard Reset signaling.
2. ***STATUS_HDRST*** = 0: Receive other signals.

Step 4 Set ***RX_CLR*** of ***pd_txrx*** register to clear PD PHY interrupt and all hardware status.

5.12.4.5 CC Wire Manual Control

The following programming sequence describes how to control the CC wire directly:

Write logic zero/one on the CC wire:

Step 1 Set ***CC_SEL*** of ***pd_ctrl*** register to obtain the control of the CC wire.

Step 2 Set ***BMCTX_EN*** of ***cc_misc*** register to turn on the TX transmission.

Step 3 Set the desired transmitted data in the ***BMCTX_DOUT*** of ***cc_misc*** register.

Read logic zero/one on the CC wire:

Step 1 Set ***CC_SEL*** of ***pd_ctrl*** register to obtain the control of the CC wire.

Step 2 Set ***BMCRX_EN*** of ***cc_misc*** register to turn on the RX reception.

Step 3 Read ***BMC_BIT*** of ***cc_misc*** register to acquire the received data.

5.13 Analog Front End (DP/DM/CC1/CC2)

Analog Front End (AFE) is an analog macro for the application of general fast charging specifications. AFE comprises a DP macro, a DM macro, a CC macro, a control interface connecting to a PD PHY controller, and a USB communication interface with the DP, DM, CC1 and CC2 pins. The DP and DM macros are two independent circuits with the identical function for the DP and DM pins, while CC macro is a common hardware shared by the CC1 and CC2 pins. Fig. 5.16 illustrates the block diagram of the AFE macro.

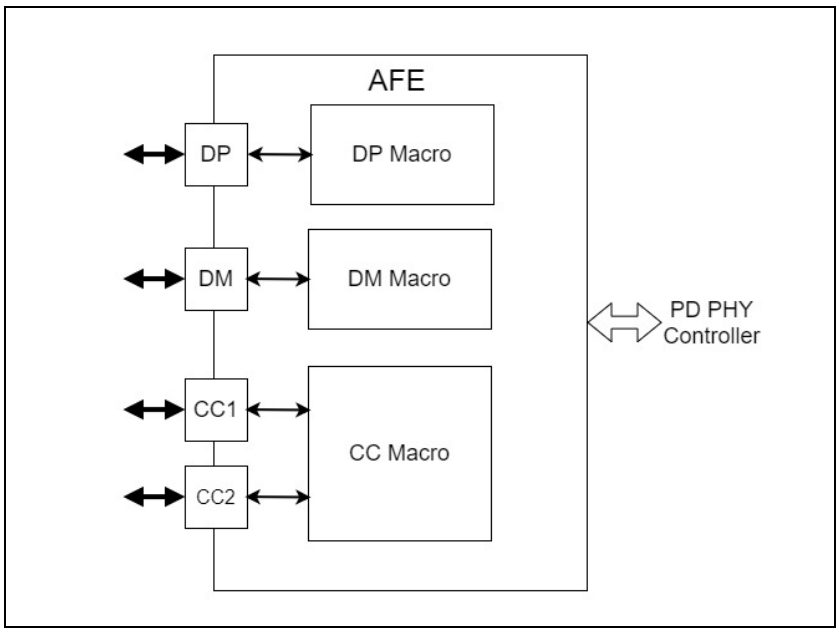


Fig. 5.16: Analog Front End (AFE) Block Diagram

5.13.1 Features

- DP and DM Macros Features:
  - Support four different drive voltage levels: 5V, 3.3V, 0.6V and 0V
  - Support one selective voltage comparators: 3V, 2.4V, 1.2V and 0.35V
  - DP and DM pins can be configured as OTP download cable pins in the test mode, which function is identical to a 5V GPIO.
  - DP and DM pins also can be operated through the GPIO control mode.
- CC Macro Features:
  - Supports a transmitter and receiver used as data transfer of the USB PD communication. Transmitter supplies a precise 1.125V drive voltage on the CC wire.
  - Supports a 0.66V comparator for USB Type-C 5V/1.5A detection.
  - Supports a 1.23V comparator for USB Type-C 5V/3A detection.
  - Supports two independent 0.2V comparators used as the CC1/CC2 attached detection.
  - Supports two independent CC1/CC2 5.1k R_d resistors.

5.13.2 Function Description

5.13.2.1 DP and DM Output Control

According to the various charging protocols, the requirement of the DP/DM output voltage level would be not the same. The AFE DP/DM macro supplies a selective drive voltage to meet one particular application. Table 5.15 and 5.16 describe all legal setting value of dp_out and dm_out registers and the corresponding drive voltage.

Note 1: 3.3V and 0V output voltage have two kinds of setting configuration, but user can select any one of them as the setting value.
Note 2: DP_ISINK_EN and DM_ISINK_EN of dp_out and dm_out register are used to control a pulldown resistor which supplies an initial value after power on reset. In the normal operation mode, it is suggested that user should turn off this function after finishing the system startup.

Bit [1:0] of dp_dm_io_ctrl	00	00	00	00	00	00	00
DP_ISINK_EN	0	0	0	0	0	0	0
DP_BUFVREF_SEL	X	X	X	X	0	1	X
DP_BUF_EN	0	0	0	0	1	1	0
DP_TX_EN	0	0	1	1	0	0	0
DP_OUT_EN	1	1	0	0	0	0	0
DP_OUT	0	1	0	1	X	X	X
DP Drive Voltage	0V	5V	0V	3.3V	0.6V	3.3V	Hi-Z

Note 1: Bit [1:0] of dp_dm_io_ctrl register must be set first before setting other register bits.
Note 2: Suggest that DP_ISINK_EN should be disabled in the normal DP operation.

Table 5.15: DP Legal Setting Value

Bit [3:2] of dp_dm_io_ctrl	00	00	00	00	00	00	00
DM_ISINK_EN	0	0	0	0	0	0	0
DM_BUFVREF_SEL	X	X	X	X	0	1	X
DM_BUF_EN	0	0	0	0	1	1	0
DM_TX_EN	0	0	1	1	0	0	0
DM_OUT_EN	1	1	0	0	0	0	0
DM_OUT	0	1	0	1	X	X	X
DM Drive Voltage	0V	5V	0V	3.3V	0.6V	3.3V	Hi-Z

Note 1: Bit [3:2] of dp_dm_io_ctrl register must be set first before setting other register bits.
Note 2: Suggest that DM_ISINK_EN should be disabled in the normal DM operation.

Table 5.16: DM Legal Setting Value

5.13.2.2 DP and DM Input Control

The DP and DM macros support a selective comparator which supplies 4-level reference voltage. Table 5.17 shows the relationship between the setting of DP_CMPVREF_SEL and DM_CMPVREF_SEL of dp_in and dm_in registers and the comparator reference voltage. The compare output of two voltage comparators will be represented individually in the DP_DIN and DM_DIN of dp_in and dm_in registers. The DP_CMP_EN and DM_CMP_EN of dp_in and dm_in registers are used to enable these two comparators individually.

DP_CMPVREF_SEL DM_CMPVREF_SEL	00	01	10	11
Compare Voltage	0.35V	1.2V	2.4V	3V

Table 5.17: Selective Reference Voltage Comparator

5.13.2.3 DP and DM GPIO Control Mode

The usage shown in the table 5.15 and 5.16 may be not suited to the operation of the DP and DM pins. Usually, not all drive voltage levels are necessary for a particular charging protocol in practice. In addition, it is not intuitive and convenient to control these related register setting. AFE macro also supports a GPIO control mode to simplify the DP and DM operation. The DP pin will be switched to the GPIO PA4, and the DM pin will be switched to the GPIO PA6. Most fields of dp_out, dp_in, dm_out and dm_in registers can be remapped to the ones of pa, pac and padier registers. Table 5.18 explains the transformation between the register fields of DP and PA4, and table 5.19 explains the transformation between the register fields of DM and PA6. It is noted that there are three different transformation types defined in the dp_dm_io_ctrl register. Table 5.20 and Table 5.21 describe the relationship between the DP/DM status and the corresponding register setting.

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
01	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	pa[4]
	DP_BUF_EN	pac[4]
	DP_TX_EN	0
	DP_OUT_EN	0
	DP_OUT	0
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
10	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	0
	DP_BUF_EN	0
	DP_TX_EN	pac[4]
	DP_OUT_EN	0
	DP_OUT	pa[4]
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
11	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	0
	DP_BUF_EN	0
	DP_TX_EN	0
	DP_OUT_EN	pac[4]
	DP_OUT	pa[4]
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

Table 5.18: Transformation between DP and PA4

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
01	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	pa[6]
	DM_BUF_EN	pac[6]
	DM_TX_EN	0
	DM_OUT_EN	0
	DM_OUT	0
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
10	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	0
	DM_BUF_EN	0
	DM_TX_EN	pac[6]
	DM_OUT_EN	0
	DM_OUT	pa[6]
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
11	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	0
	DM_BUF_EN	0
	DM_TX_EN	0
	DM_OUT_EN	pac[6]
	DM_OUT	pa[6]
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

Table 5.19: Transformation between DM and PA6

DP Status	pa[4]	pac[4]	padier[4]
Bit [1:0] of dp_dm_io_ctrl = 01			
Output High (3.3V)	1	1	X
Output Low (0.6V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0
Bit [1:0] of dp_dm_io_ctrl = 10			
Output High (3.3V)	1	1	X
Output Low (0V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0
Bit [1:0] of dp_dm_io_ctrl = 11			
Output High (5V)	1	1	X
Output Low (0V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0

Note: V_{th} is decided by DP_CMPVREF_SEL.

Table 5.20: Relationship between DP Status and PA4 Register Setting

DM Status	pa[6]	pac[6]	padier[6]
Bit [3:2] of dp_dm_io_ctrl = 01			
Output High (3.3V)	1	1	X
Output Low (0.6V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0
Bit [3:2] of dp_dm_io_ctrl = 10			
Output High (3.3V)	1	1	X
Output Low (0V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0
Bit [3:2] of dp_dm_io_ctrl = 11			
Output High (5V)	1	1	X
Output Low (0V)	0	1	X
Input High (> V_{th})	1	0	1
Input Low (< V_{th})	0	0	1
Output Hi-Z	X	0	X
Input Hi-Z	X	X	0

Note: V_{th} is decided by DM_CMPVREF_SEL

Table 5.21: Relationship between DM Status and PA6 Register Setting

5.13.2.4 CC1 and CC2 Control

The CC macro provides all necessary functions for the application of the USB PD specification. The CC macro comprises a PD data stream path (BMC_TX and BMC_RX), two voltage comparators with 0.66V and 1.23V reference voltage individually for the USB Type-C 5V/1.5A and 5V/3A application, a CC1/CC2 attached detection circuit, and two Rd pulldown resistors for CC1 and CC2 pins.

5.13.2.4.1 USB PD Data Stream Path

The BMC_TX and BMC_RX are designed individually for a transmitter and receiver which comprise the data stream path of the USB PD communication. The BMC_TX transmits the BMC encoding bit stream from the PD PHY controller. A precise 1.125V output voltage can be trimmed in mass production through searching for the best trimming value of Bandgap reference voltage. The BMC_RX decodes the incoming data on the CC wire, and these decoding data will be transferred to the PD PHY controller further.

The BMCTX_EN signal is used to enable the BMC_TX transmitter, and the BMCTX_DOUT signal is used as the transmitted data. The BMCRX_EN signal is used to enable the BMC_RX receiver, and the BMC_BIT presents the decoding output value of the BMC_RX. In the normal operation mode, all signals mentioned above are under the control of the PD PHY controller. However, these signals also can be managed directly by register access mode. All fields with the same naming in the cc_misc register can be used to access these signals. How to control this register, please see the related programming sequence in the section 5.12.4.5 for further details.

5.13.2.4.2 USB Type-C 5V/1.5A and 5V/3A Detection

There are two independent voltage comparators in the CC macro used for the Type-C 5V/1.5A and 5V/3A detection. One comparator with 0.66V reference voltage is used for 5V/1.5A application, and another with 1.23V is for 5V/3A application. The CC_CMPSNKXA_EN of cc_stat register is used to enable these two comparators. The CC_SNK1P5A and CC_SNK3P0A of cc_stat register are used to represent the output status of these two comparators individually.

5.13.2.4.3 CC1/CC2 Attached Detection

There are two independent CC attached detection circuits in the CC macro for the CC1 and CC2 pins individually. Each is composed of one comparator with 0.2V reference voltage and one sampling latch for sampling and storing the output value of the voltage comparator. The CC_CMPRA_EN of cc_ctrl register is used to enable these two comparators. And, the CC1_ATT and CC2_ATT of cc_ctrl register are used to represent the individual output status of these two sampling latches. The sampling operation of the latch is defined in the CC_CMPRA_LAT of cc_ctrl register, and the reset control of the latch is defined in the CC_REGRA_RESB of cc_ctrl register.

The following programming sequence shows how to detect the CC wire attached:

Step 1 Set ***CC_CMPRA_EN*** of ***cc_ctrl*** register to enable two voltage comparators.
Step 2 Set 1 first, then set 0 to the ***CC_CMPRA_LAT*** of ***cc_ctrl*** register to sample and hold the output value of these two comparators simultaneously.
Step 3 Read the ***CC1_ATT*** and ***CC2_ATT*** of ***cc_ctrl*** register to decide which active pin is connecting to the USB cable.

Table 5.22 shown below explains the attached detection result by reading the status of both CC1_ATT and CC2_ATT2.

CC1_ATT	CC2_ATT	Description
0	0	Reset value
1	0	CC1 attached
0	1	CC2 attached
1	1	N.A.

Table 5.22: Attached Detection Result by the Status of CC1_ATT and CC2_ATT2

5.14 GATE Pin Control and Connection

5.14.1 Function Description

One particular GATE pin is designed to manage the power delivery for the fast charging application. It can be connected to an external power MOSFET to turn on/off power line. The bit0 of misc2 register is used to control the GATE pin. Table 5.23 shown below describes this IO output characteristics. In addition, the GATE pin has the following features:

(1) Open-drain IO with 20mA sink current capability at V_{OL} = 0.5V.
(2) Supports 21V high voltage tolerance.

Bit0 of misc2	Gate Pin
0	Output Floating (Reset Value)
1	Output 0V

Table 5.23: GATE Pin Output Characteristics

5.14.2 GATE Pin Application Connection

Fig. 5.17 illustrates a power management application and the external connection of one power MOSFET. During power on reset, the output status of the GATE pin is floating, which disable the external MOSFET. In the meanwhile, no current can flow from VBUS to VGATE. Set 1 to the bit0 of misc2 register to force the GATE pin to output a zero voltage, which turn on the P-Channel MOSFET and allow the flow of charging current from VBUS to VGATE.

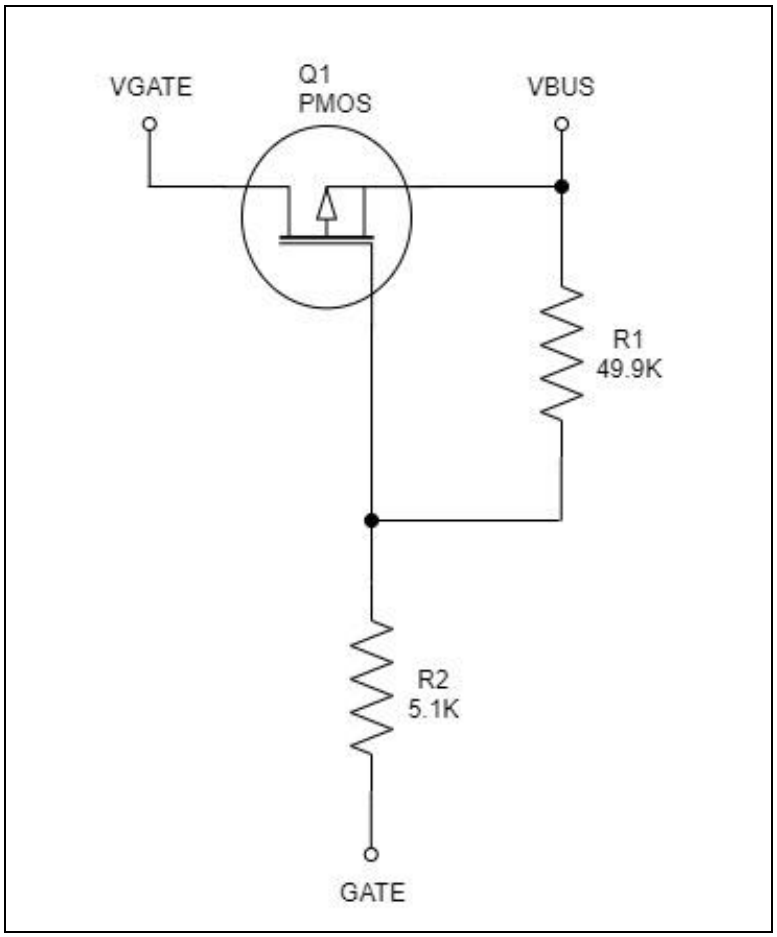


Fig. 5.17: GATE Pin Application Connection

5.15 VBUS (VDD) Voltage Comparator

5.15.1 Function Description

One VBUS (VDD) comparator with a fixed 7V reference voltage is designed for the particular charging specification application. The bit3 of misc2 register is used to enable the voltage comparator. And, the compare result can be read through the bit7 of misc2 register.

6. IO Registers

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

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. Please do not use.
3	0	R/W	OV (Overflow Flag). This bit is set to be 1 whenever the sign operation is overflow.
2	0	R/W	AC (Auxiliary Carry Flag). There are two conditions to set this bit, the first one is carry out of low nibble in addition operation and the other one is borrow from the high nibble into low nibble in subtraction operation.
1	0	R/W	C (Carry Flag). There are two conditions to set this bit, the first one is carry out in addition operation, and the other one is borrow in subtraction operation. Carry is also affected by shift with carry instruction.
0	0	R/W	Z (Zero Flag). This bit will be set when the result of arithmetic or logic operation is zero; Otherwise, it is cleared.

6.2. Reset Control Register (*rstc*), IO address = 0x01

Bit	Reset	R/W	Description
7	0	R/W	External reset (PA5) in test mode Enable. 0 / 1: disable / Enable
6	0	R/W	Enable fast Wake up. 0: Normal wake up. The wake-up time is 1024 ILRC clocks. 1: Fast wake up. The wake-up time is 16 ILRC clocks.
5 - 4	00	R/W	Watch dog time out period 00: 8k ILRC clock period 01: 16k ILRC clock period 10: 64k ILRC clock period 11: 256k ILRC clock period
3	0	R/W	Reserved
2	1	R/W	LVR Reset Enable. 0 / 1: disable / enable
1	1	R/W	Watch Dog Enable. 0 / 1: disable / enable
0	0	R/W	Pin PA5/PRSTB function. 0 / 1: PA5 / PRSTB

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

Bit	Reset	R/W	Description
7 - 0	-	R/W	Stack Pointer Register. Read out the current stack pointer, or write to change the stack pointer.

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

Bit	Reset	R/W	Description
7 - 6	00	R/W	System clock... 00: System clock / 1 01: System clock / 2 10: System clock / 4 11: System clock / 8
5	0	R/W	System clock selection. 0 / 1: ILRC / IHRC
4 - 2	-	R/W	Reserved
1	1	R/W	Internal High RC Enable. 0 / 1: disable / enable
0	0	R/W	Internal Low RC mode selection. 0 / 1: ILRC / NILRC

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

Bit	Reset	R/W	Description
7	0	R/W	Reserved
6	0	R/W	Reserved
5	0	R/W	Enable interrupt from Timer2. 0 / 1: disable / enable
4	0	R/W	Reserved
3	0	R/W	Reserved
2	0	R/W	Enable interrupt from Timer16. 0 / 1: disable / enable
1	0	R/W	Enable interrupt from PD_PHY. 0 / 1: disable / enable
0	0	R/W	Enable interrupt from PA0. 0 / 1: disable / enable

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

Bit	Reset	R/W	Description
7	-	R/W	Reserved
6	-	R/W	Reserved
5	-	R/W	Interrupt Request from Timer2, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
4	-	R/W	Reserved
3	-	R/W	Reserved
2	-	R/W	Interrupt Request from Timer16, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
1	-	R/W	Interrupt Request from PD_PHY, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	R/W	Interrupt Request from pin PA0, this bit is set by hardware and cleared by software. 0 / 1: No Request / request

6.7. LVR Reset Control Register (lvrc), IO address = 0x08

Bit	Reset	R/W	Description
7 – 4	0000	WO	LVR reset voltage selection. 0000: 1.8 V 0001: 1.9 V 0010: 2.0 V 0011: 2.1 V 0100: 2.2 V 0101: 2.3 V 0110: 2.4 V 0111: 2.5 V 1000: 2.7 V 1001: 3.0 V 1010: 3.15V 1011: 3.3 V 1100: 3.5 V 1101: 3.75V 1110: 4.0 V 1111: 4.5 V
3 – 0	-	WO	Reserved

6.8. Interrupt Edge Select Register (*integ*), IO address = 0x09

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

6.9. Timer16 mode Register (*t16m*), IO address = 0x0B

Bit	Reset	R/W	Description
7 - 5	000	R/W	Timer16 Clock source selection. 000: disable 001: CLK (system clock) 010: reserved 100: IHRC 110: ILRC 111: PA0 falling edge (from external pin)
4 - 3	00	R/W	Timer16 clock pre-divider. 00: ÷1

Bit	Reset	R/W	Description
			01: ÷4 10: ÷16 11: ÷64
2 - 0	000	R/W	Interrupt source selection. Interrupt event happens when the selected bit status is changed. 0 : bit 8 of Timer16 1 : bit 9 of Timer16 2 : bit 10 of Timer16 3 : bit 11 of Timer16 4 : bit 12 of Timer16 5: bit 13 of Timer16 6: bit 14 of Timer16 7: bit 15 of Timer16

6.10. DP/DM IO Control Register (dp_dm_io_ctrl), IO address = 0x0C

Bit	Reset	R/W	Description		
7 - 4	0000	R/W	Reserved		
3 - 2	00	R/W	DM IO mode switch.		
			00	O_DM_BUF_EN	= DM_BUF_EN
				O_DM_BUFVREF_SEL	= DM_BUFVREF_SEL
				O_DM_TX_EN	= DM_TX_EN
				O_DM_DOUT_EN	= DM_DOUT_EN
				O_DM_DOUT	= DM_DOUT
				O_DM_CMP_EN	= DM_CMP_EN
			01	O_DM_BUF_EN	= pac[6]
				O_DM_BUFVREF_SEL	= pa[6]
				O_DM_TX_EN	= 1'b0
				O_DM_DOUT_EN	= 1'b0
				O_DM_DOUT	= 1'b0
				O_DM_CMP_EN	= padier[6]
			10	O_DM_BUF_EN	= 1'b0
				O_DM_BUFVREF_SEL	= 1'b0
				O_DM_TX_EN	= pac[6]
				O_DM_DOUT_EN	= 1'b0
				O_DM_DOUT	= pa[6]
				O_DM_CMP_EN	= padier[6]
			11	O_DM_BUF_EN	= 1'b0
				O_DM_BUFVREF_SEL	= 1'b0
				O_DM_TX_EN	= 1'b0
				O_DM_DOUT_EN	= pac[6]
				O_DM_DOUT	= pa[6]
				O_DM_CMP_EN	= padier[6]

Bit	Reset	R/W	Description		
1 - 0	00	R/W	DP IO mode switch.		
			00	O_DP_BUF_EN	= DP_BUF_EN
				O_DP_BUFVREF_SEL	= DP_BUFVREF_SEL
				O_DP_TX_EN	= DP_TX_EN
				O_DP_DOUT_EN	= DP_DOUT_EN
				O_DP_DOUT	= DP_DOUT
				O_DP_CMP_EN	= DP_CMP_EN
			01	O_DP_BUF_EN	= pac[4]
				O_DP_BUFVREF_SEL	= pa[4]
				O_DP_TX_EN	= 1'b0
				O_DP_DOUT_EN	= 1'b0
				O_DP_DOUT	= 1'b0
				O_DP_CMP_EN	= padier[4]
			10	O_DP_BUF_EN	= 1'b0
				O_DP_BUFVREF_SEL	= 1'b0
				O_DP_TX_EN	= pac[4]
				O_DP_DOUT_EN	= 1'b0
				O_DP_DOUT	= pa[4]
				O_DP_CMP_EN	= padier[4]
			11	O_DP_BUF_EN	= 1'b0
				O_DP_BUFVREF_SEL	= 1'b0
				O_DP_TX_EN	= 1'b0
				O_DP_DOUT_EN	= pac[4]
				O_DP_DOUT	= pa[4]
				O_DP_CMP_EN	= padier[4]

6.11.Port A Digital Input Enable Register (*padier*), IO address = 0x0D

Bit	Reset	R/W	Description
7 - 6	-	WO	Reserved. (Please keep 00 for future compatibility)
5	0	WO	Enable PA5 digital input and wake-up event. 1 / 0: enable / disable This bit can be set to low to disable wake-up from PA5 toggling from this pin.
4	-	WO	Reserved.
3	0	WO	Enable PA3 digital input and wake-up event. 1 / 0: enable / disable This bit can be set to low to disable wake-up from PA3 toggling from this pin.
2 - 1	-	WO	Reserved. (Please keep 00 for future compatibility)
0	0	WO	Enable PA0 digital input, wake-up event and interrupt request. 1 / 0: enable / disable. This bit can be set to low to disable wake-up from PA0 toggling and interrupt request from this pin.

6.12.Port B Digital Input Enable Register (*pbdier*), IO address = 0x0E

Bit	Reset	R/W	Description
7 - 2	-	WO	Reserved. (Please keep 00 for future compatibility)
1	0	WO	Enable PB1 digital input and wake-up event. 1 / 0: enable / disable. This bit can be set to low to disable wake-up from PB1 toggling from this pin.
0	0	WO	Enable PB0 digital input and wake-up event. 1 / 0: enable / disable. This bit can be set to low to disable wake-up from PB0 toggling from this pin.

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

Bit	Reset	R/W	Description
7	-	R/W	Reserved
6 - 0	0x00	R/W	Data register for Port A. (PA7 is not available)

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A control registers. This register is used to define input mode or output mode for each corresponding pin of port A. 0 / 1: input / output Please note: PA5 is an open drain output. PA1, PA2, PA4, PA6, and PA7 are not available.

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

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port A and this pull high function is active only for input mode. 0 / 1 : disable / enable PA1, PA2, PA4, PA6, and PA7 are not available.

6.16. Port A Pull-Low Register (*papl*), IO address = 0x13

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 and this pull low function is active only for input mode. 0 / 1 : disable / enable PA1, PA2, PA4, PA6, and PA7 are not available.

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

Bit	Reset	R/W	Description
7 - 2	-	R/W	Reserved
1 - 0	00	R/W	Data register for Port B. (Only PB0 and PB1 are available.)

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

Bit	Reset	R/W	Description
7 - 2	-	R/W	Reserved
1 - 0	00	R/W	Port B control register. This register is used to define input mode or output mode for each corresponding pin of port B. 0 / 1: input / output Only PB0 and PB1 are available.

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

Bit	Reset	R/W	Description
7 – 2	-	R/W	Reserved.
1 – 0	00	R/W	Port B pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port B and this pull high function is active only for input mode. 0 / 1 : Disable / Enable

6.20. Port B Pull-Low Register (*pbpl*), IO address = 0x17

Bit	Reset	R/W	Description
7 – 2	-	R/W	Reserved
1 – 0	00	R/W	Port B pull-low register. This register is used to enable the internal pull-low device on each corresponding pin of port B and this pull low function is active only for input mode. 0 / 1 : Disable / Enable

6.21. Timer2 PWM Control Register (*tpw2c*), IO address = 0x1A

Bit	Reset	R/W	Description
7	0	R/W	Timer2 PWM Disable/ Enable 0 : TPW2 Disable 1 : TPW2 Enable
6	0	R/W	Timer2 PWM mode selection. 0 : Period mode 1 : PWM mode
5 - 4	00	R/W	Timer2 PWM clock source selection. 00 : IHRC 01 : IHRC * 2 10 : Reserved 11 : ILRC
3 - 2	00	R/W	Timer2 PWM clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
1 - 0	00	R/W	Timer2 PWM resolution selection. 00 : 8-bit 01 : 7-bit 10 : 6-bit 11 : Reserved

6.22. Timer2 PWM Scalar Register (*tpw2s*), IO address = 0x1B

Bit	Reset	R/W	Description
7	0	R/W	Inverse the polarity of result output of Timer2 PWM. 0: polarity is NOT inversed 1: polarity is inversed
6 - 5	00	R/W	Timer2 PWM output port selection. 00 : TPW2 output disable 01 : Reserved 10 : TPW2 output to PA3 11 : Reserved
4 - 0	0x00	R/W	Timer2 PWM clock scalar.

6.23. Timer2 PWM Bound Register (*tpw2b*), IO address = 0x1C

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

6.24. Operation Register (*opr1*), IO address = 0x1D

Bit	Reset	R/W	Description
6 - 4	000	R/W	Output probe signal. 000 : Disable output probe signal 001 : Output PD PHY probe signal

6.25. Miscellaneous Register (*misc2*), IO address = 0x1F

Bit	Reset	R/W	Description
7	0	RO	VBUS (VDD) Voltage Comparator Output. 0: VBUS (VDD) < 7V 1: VBUS (VDD) > 7V
6	0	R/W	ILRC switch in stopsys mode 0: Disable 1: Enable Note: Please make sure to set it to Disable.
5	0	R/W	LVR switch in stopsys mode 0: Disable 1: Enable Note: Please make sure to set it to Disable.
4	1	R/W	LVR power switch 0: Off 1: On (Note) Note: Please make sure to set it to ON.
3	1	R/W	VBUS (VDD) Voltage Comparator Control. 0: Disable 1: Enable
2	1	R/W	PD CTRL frequency 6MHz and 3MHz switch 0 : Disable 1 : Enable

Bit	Reset	R/W	Description
1	1	R/W	PD CTRL frequency 12MHz switch 0 : Disable 1 : Enable
0	0	R/W	GATE Drive Control: 0 : Disable (Output Floating) 1 : Enable (Output 0V)

6.26. PD PHY FIFO Register (*pd_fifo*), IO address = 0x30

Bit	Reset	R/W	Description
7 - 0	-	R/W	FIFO_DATA : One 4-byte FIFO is used for PD PHY TX/RX data storage. In TX mode (WO): used to write the data payload of a PD SOP packet. In RX mode (RO): used to read the received packet data, including 32-bit CRC data.

6.27. PD PHY Status Register (*pd_stat*), IO address = 0x31

Bit	Reset	R/W	Description
7	-	-	Reserved
6	1	RO	STATUS_IDLE: Reveal the USB CC wire status. It is used to avoid a signal collision occurrence on the CC wire. User must assure that the bus is at the idle status before starting to transmit data. 0: Bus Active 1: Bus Idle
5	0	RO	STATUS_HDRST: PD PHY controller has received one Hard Reset signaling successfully. This bit is valid only when both RX_FINISH and RX_SUCCESS are active. Auto-Clear when enabling RX_CLR or disabling PHYCTRL_EN of pd_txrx register. 0: Receive one PD SOP packet 1: Receive one PD Hard Reset signaling
4	0	RO	FIFO_FULL: Reveal PD PHY FIFO (pd_fifo register) status. Restore to the default value when enabling RX_CLR or disabling PHYCTRL_EN of pd_txrx register. 0: Non-Full 1: Full
3	1	RO	FIFO_EMPTY: Reveal PD PHY FIFO (pd_fifo register) status. Restore to the default value when enabling RX_CLR or disabling PHYCTRL_EN of pd_txrx register. 0: Non-Empty 1: Empty
2	0	RO	RX_SUCCESS: PD PHY controller has received one packet or signaling successfully. This bit is valid only when RX_FINISH is active.

Bit	Reset	R/W	Description
			Auto-Clear when enabling RX_CLR or disabling PHYCTRL_EN of pd_txrx register. 0: RX Fail (This received data may be illegal or broken, and it should be ignored.) 1: RX Success
1	0	RO	RX_FINISH: PD PHY controller has finished one packet or signaling reception. Auto-Clear when enabling RX_CLR or disabling PHYCTRL_EN of pd_txrx register. 0: RX Busy if PD PHY controller is starting to receive a packet. 1: RX Finish
0	0	RO	TX_FINISH: PD PHY controller has finished one packet or signaling transmission successfully. Auto-Clear when enabling TX_CLR or disabling PHYCTRL_EN of pd_txrx register. Note: If PD PHY controller is starting to receive a packet unexpectedly, this bit will not be active and this packet transmission will be cancelled. 0: TX Busy if TX_START of pd_txrx register has been enabled. 1: TX Finish

6.28. PD PHY TXRX Register (*pd_txrx*), IO address = 0x32

Bit	Reset	R/W	Description
7	-	-	Reserved
6	0	RW	SEND_HDRST: Enable PD PHY controller to send one Hard Reset signaling on the CC wire. 0: Disable 1: Enable
5	0	RW	PHYCTRL_EN: Enable PD PHY controller. Note: When user disable the PD PHY controller, the following signal will be also cleared simultaneously: (1) TX_FINISH, RX_FINISH, RX_SUCCESS and STATUS_HDRST of pd_stat register (2) PD PHY interrupt signal (3) FIFO_EMPTY and FIFO_FULL of pd_stat register restores to the default value 0: Disable 1: Enable
4	0	RW	TX_START: Enable PD PHY controller to transmit a packet or signaling, including the BIST signaling. 0: No effect 1: Enable (Auto-Clear after enabling this bit)
3	0	RW	RX_CLR: User must set this bit to clear all RX status and PD PHY interrupt after PD PHY controller has finished a packet or signaling reception. When this bit is enabled, the following signal will be also cleared simultaneously: (1) RX_FINISH, RX_SUCCESS and STATUS_HDRST of pd_stat register (2) PD PHY interrupt signal

Bit	Reset	R/W	Description
			(3) FIFO_EMPTY and FIFO_FULL of pd_stat register restores to the default value 0: No effect 1: Enable (Auto-Clear after enabling this bit)
2	0	RW	TX_CLR: User must set this bit to clear all TX status and PD PHY interrupt after PD PHY controller has finished a packet or signaling transmission. When this bit is enabled, the following signals will be also cleared simultaneously: (1) TX_FINISH of pd_stat register (2) PD PHY interrupt signal 0: No effect 1: Enable (Auto-Clear after enabling this bit)
1	0	RW	FIFO_RO: User must update the read index pointer of the PD PHY FIFO after reading one byte from FIFO (pd_fifo register). A correct operation is “write 1 first, then write 0” or “write 0 first, then write 1” (rising-edge trigger mode).
0	0	RW	FIFO_WR: User must update the write index pointer of the PD PHY FIFO after writing one byte into FIFO (pd_fifo register). A correct operation is “write 1 first, then write 0” or “write 0 first, then write 1” (rising-edge trigger mode).

6.29. PD PHY Control Register (*pd_ctrl*), IO address = 0x33

Bit	Reset	R/W	Description
7:4	-	-	Reserved
3:2	0	RW	PROBE_SEL: Select the inner signal of PD PHY controller onto the specified GPIO pins when all related GPIO pins have been switched to the probe mode. These signals are useful for debugging protocol errors during software development. Note: These bits are valid only when bit [6:4] of opr1 register = 001 (Output PD PHY probe signal). 00: int_pd_phy, st_phy[2:0] 01: fifo_ov, STATUS_HDRST, RX_SUCCESS, RX_FINISH 10: FIFO_FULL, FIFO_EMPTY, TX_FINISH, STATUS_IDLE 11: crc_cmp_out, sym_eop, pktdet_fail, BMC_BIT
1	0	RW	DIS_RX: PD PHY controller automatically turn off the received data path during transmitting data. 0: Disable 1: Enable
0	0	RW	CC_SEL: Obtain the control of transmitted and received data paths from the PD PHY controller. 0: Disable (PD PHY controller manages all TX/RX data paths automatically.) 1: Enable (User can control the entire data communication by setting BMCRX_EN, BMCTX_EN and BMCTX_DOUT of cc_misc register.)

6.30. PD PHY Decoder Register (*pd_decoder*), IO address = 0x34

Bit	Reset	R/W	Description
7:5	4	RW	TH_LOW: Used to adjust the PD PHY data receiving capability. Threshold selection of low-voltage signal length of BMC decoding: 000: 11T Clock Time (166.66ns X 11) 001: 12T Clock Time (166.66ns X 12) 010: 13T Clock Time (166.66ns X 13) 011: 14T Clock Time (166.66ns X 14) 100: 15T Clock Time (166.66ns X 15) 101: 16T Clock Time (166.66ns X 16) 110: 17T Clock Time (166.66ns X 17) 111: 18T Clock Time (166.66ns X 18)
4:2	4	RW	TH_HIGH: Used to adjust the PD PHY data receiving capability. Threshold selection of high-voltage signal length of BMC decoding: 000: 11T Clock Time (166.66ns X 11) 001: 12T Clock Time (166.66ns X 12) 010: 13T Clock Time (166.66ns X 13) 011: 14T Clock Time (166.66ns X 14) 100: 15T Clock Time (166.66ns X 15) 101: 16T Clock Time (166.66ns X 16) 110: 17T Clock Time (166.66ns X 17) 111: 18T Clock Time (166.66ns X 18)
1:0	1	RW	TH_DEB: Used to adjust the PD PHY data receiving capability. Debounce time selection for the received data on the CC wire: 00: Deglitch 1T Clock Time (166.66ns X 1) 01: Deglitch 2T Clock Time (166.66ns X 2) 10: Deglitch 3T Clock Time (166.66ns X 3) 11: Deglitch 4T Clock Time (166.66ns X 4)

6.31. DP Output Register (*dp_out*), IO address = 0x38

Bit	Reset	R/W	Description
7:6	-	-	Reserved
5	1	RW	DP_ISINK_EN: Enable DP sink current path (25uA~175uA, weak pulldown). Note: Strongly suggest that it should be turned off after system startup. Its function is only used for providing an initial value on the DP pin, which preventing the unknown state from entering system operation. 0: Disable 1: Enable
4	0	RW	DP_BUFVREF_SEL: Voltage selection of DP drive buffer; this bit is valid only when DP_BUF_EN = 1. 0: 0.6V 1: 3.3V

Bit	Reset	R/W	Description
3	0	RW	DP_BUF_EN: Enable DP drive buffer. 0: Disable 1: Enable
2	0	RW	DP_TX_EN: Enable DP drive buffer. 0: Disable 1: Enable
1	0	RW	DP_OUT_EN: Enable DP digital output. 0: Disable 1: Enable
0	0	RW	DP_OUT: DP drive voltage selection; this bit is valid only when DP_OUT_EN or DP_TX_EN = 1. 0: 0V when DP_OUT_EN or DP_TX_EN = 1 1: (1) 5V when DP_OUT_EN = 1 (Note) (2) 3.3V when DP_TX_EN = 1 Note: Actual output voltage is decided by a 5V Regulator output (V_{REG}). An additional voltage drop can cause the DP drive voltage less than the expected value (5V).

6.32. DP Input Register (*dp_in*), IO address = 0x39

Bit	Reset	R/W	Description
7:4	-	-	Reserved
3:2	2	RW	DP_CMPVREF_SEL: Reference voltage selection of DP voltage comparator. 00: 0.35V 01: 1.2V 10: 2.4V 11: 3.0V
1	1	RW	DP_CMP_EN: Enable DP voltage comparator. 0: Disable 1: Enable
0	0	R/O	DP_DIN: Output value of DP voltage comparator. 0: Logic Zero 1: Logic One

6.33. DM Output Register (*dm_out*), IO address = 0x3A

Bit	Reset	R/W	Description
7:6	-	-	Reserved
5	1	RW	DM_ISINK_EN: Enable DM sink current path (25uA~175uA, weak pulldown). Note: Strongly suggest that it should be turned off after system startup. Its function is only used for providing an initial value on the DM pin, which preventing the unknown state from entering system operation. 0: Disable 1: Enable
4	0	RW	DM_BUFVREF_SEL: Voltage selection of DM drive buffer; this bit is valid only when DM_BUF_EN = 1. 0: 0.6V 1: 3.3V
3	0	RW	DM_BUF_EN: Enable DM drive buffer. 0: Disable 1: Enable
2	0	RW	DM_TX_EN: Enable DM drive buffer. 0: Disable 1: Enable
1	0	RW	DM_OUT_EN: Enable DM digital output. 0: Disable 1: Enable
0	0	RW	DM_OUT: DM drive voltage selection; this bit is valid only when DM_OUT_EN or DM_TX_EN = 1. 0: 0V when DM_OUT_EN or DM_TX_EN = 1 1: (1) 5V when DM_OUT_EN = 1 (Note) (2) 3.3V when DM_TX_EN = 1 Note: Actual output voltage is decided by a 5V Regulator output (V_{REG}). An additional voltage drop can cause the DM drive voltage less than the expected value (5V).

6.34. DM Input Register (*dm_in*), IO address = 0x3B

Bit	Reset	R/W	Description
7:4	-	-	Reserved
3:2	2	RW	DM_CMPVREF_SEL: Reference voltage selection of DM voltage comparator. 00: 0.35V 01: 1.2V 10: 2.4V 11: 3.0V
1	1	RW	DM_CMP_EN: Enable DM voltage comparator. 0: Disable 1: Enable
0	0	R/O	DM_DIN: Output value of DM voltage comparator. 0: Logic Zero 1: Logic One

6.35. CC Control Register (*cc_ctrl*), IO address = 0x3D

Bit	Reset	R/W	Description
7:5	-	-	Reserved
4	0	RW	CC_CMPRA_LAT: Sample CC1/CC2 attached detection comparator output, and save the compare value into the CC1/CC2 sampling latch. Note: Correct operation is “Write 1 first, then write 0”. 0: Release 1: Latch comparator output
3	0	RW	CC_REGRA_RESB: Reset CC1/CC2 sampling latch. 0: Reset (Active Low) 1: Release
2	1	RW	CC_CMPRA_EN: Enable CC1/CC2 attached detection comparator. 0: Disable 1: Enable
1	0	RO	CC2_ATT: CC2 sampling latch output. 0: CC2 attached detection comparator output ＜0.2V 1: CC2 attached detection comparator output ≧0.2V
0	0	RO	CC1_ATT: CC1 sampling latch output. 0: CC1 attached detection comparator output ＜0.2V 1: CC1 attached detection comparator output ≧0.2V

6.36. CC Status Register (*cc_stat*), IO address = 0x3E

Bit	Reset	R/W	Description
7:3	-	-	Reserved
2	0	RW	CC_CMPSNKXA_EN: Enable two independent CC1/CC2 voltage comparators to detect the USB Type-C current mode. 0: Disable 1: Enable
1	0	RO	CC_SNK3P0A: CC1/CC2 voltage comparator output for the USB Type-C 5V/3A detection. 0: CC1/CC2 voltage comparator output ＜1.23V 1: CC1/CC2 voltage comparator output ≧1.23V
0	0	RO	CC_SNK1P5A: CC1/CC2 voltage comparator output for the USB Type-C 5V/1.5A detection. 0: CC1/CC2 voltage comparator output ＜0.66V 1: CC1/CC2 voltage comparator output ≧0.66V

6.37. CC Miscellaneous Register (*cc_misc*), IO address = 0x3F

Bit	Reset	R/W	Description
7:4	-	-	Reserved
3	0	RW	BMCTX_EN: Enable CC1/CC2 digital output. This bit is valid only when CC_SEL of pd_ctrl register = 1. 0: Disable 1: Enable
2	0	RW	BMCTX_DOUT: CC1/CC2 digital output data. This bit is valid only when CC_SEL of pd_ctrl register = 1. 0: Logic Zero 1: Logic One
1	0	RW	BMCRX_EN: Enable CC1/CC2 receiver edge detector. This bit is valid only when CC_SEL of pd_ctrl register = 1. 0: Disable 1: Enable
0	0	RO	BMC_BIT: CC1/CC2 receiver edge detector output. 0: Logic Zero 1: Logic One

7. Instructions

Symbol	Description
ACC	Accumulator (Abbreviation of accumulator)
a	Accumulator (symbol of accumulator in program)
sp	Stack pointer
flag	ACC status flag register
I	Immediate data
&	Logical AND
|	Logical OR
←	Movement
^	Exclusive logic OR
+	Add
−	Subtraction
~	NOT (logical complement, 1's complement)
∓	NEG (2's complement)
OV	Overflow (The operational result is out of range in signed 2's complement number system)
Z	Zero (If the result of ALU operation is zero, this bit is set to 1)
C	Carry (The operational result is to have carry out for addition or to borrow carry for subtraction in unsigned number system)
AC	Auxiliary Carry (If there is a carry out from low nibble after the result of ALU operation, this bit is set to 1)
M.n	Only addressed in 0~0x3F (0~63) is allowed
IO.n	Only addressed in 0~0x3F (0~63) is allowed

7.1. Data Transfer Instructions

mov a, I	Move immediate data into ACC. Example: *mov* a, 0x0f; Result: a ← 0fh; Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
mov M, a	Move data from ACC into memory Example: *mov* MEM, a; Result: MEM ← a Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
mov a, M	Move data from memory into ACC Example: *mov* a, MEM ; Result: a ← MEM; Flag Z is set when MEM is zero. Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
mov a, IO	Move data from IO into ACC Example: *mov* a, pa ; Result: a ← pa; Flag Z is set when pa is zero. Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV
mov IO, a	Move data from ACC into IO Example: *mov* pb, a; Result: pb ← a Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
ldt16 word	Move 16-bit counting values in Timer16 to memory in word. Example: *ldt16* word; Result: word ← 16-bit timer Affected flags: 『N』Z 『N』C 『N』AC 『N』OV Application Example: --- word T16val ; // declare a RAM word … *clear* lb@ T16val ; // clear T16val (LSB) *clear* hb@ T16val ; // clear T16val (MSB) *stt16* T16val ; // initial T16 with 0 … *set1* t16m.5 ; // enable Timer16 … *set0* t16m.5 ; // disable Timer 16 *ldt16* T16val ; // save the T16 counting value to T16val …. ---

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

	idxm RAMIndex, a ; // mov 0xA5 to memory in address 0x5B ----------
xch M	Exchange data between ACC and memory Example: *xch* MEM ; Result: MEM ← a , a ← MEM Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
pushaf	Move the *ACC* and *flag* register to memory that address specified in the stack pointer. Example: *pushaf*; Result: [sp] ← {flag, ACC}; sp ← sp + 2 ; Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV Application Example: ---------- .romadr 0x10 ; // ISR entry address *pushaf* ; // put ACC and flag into stack memory … // ISR program … // ISR program *popaf* ; // restore ACC and flag from stack memory *reti* ; ----------
popaf	Restore *ACC* and *flag* from the memory which address is specified in the stack pointer. Example: *popaf*; Result: sp ← sp - 2 ; {Flag, ACC} ← [sp] ; Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV

7.2. Arithmetic Operation Instructions

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

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

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

7.3. Shift Operation Instructions

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

7.4. Logic Operation Instructions

and a, I	Perform logic AND on ACC and immediate data, then put result into ACC Example: *and* a, 0x0f ; Result: a ← a & 0fh Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
and a, M	Perform logic AND on ACC and memory, then put result into ACC Example: *and* a, RAM10 ; Result: a ← a & RAM10 Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
and M, a	Perform logic AND on ACC and memory, then put result into memory Example: *and* MEM, a ; Result: MEM ← a & MEM Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
or a, I	Perform logic OR on ACC and immediate data, then put result into ACC Example: *or* a, 0x0f ; Result: a ← a | 0fh Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
or a, M	Perform logic OR on ACC and memory, then put result into ACC Example: *or* a, MEM ; Result: a ← a | MEM Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
or M, a	Perform logic OR on ACC and memory, then put result into memory Example: *or* MEM, a ; Result: MEM ← a | MEM Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
xor a, I	Perform logic XOR on ACC and immediate data, then put result into ACC Example: *xor* a, 0x0f ; Result: a ← a ^ 0fh Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
xor IO, a	Perform logic XOR on ACC and IO register, then put result into IO register Example: *xor pa, a ;* Result: pa ← a ^ pa ; // pa is the data register of port A Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV
xor a, M	Perform logic XOR on ACC and memory, then put result into ACC Example: *xor* a, MEM ; Result: a ← a ^ RAM10 Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
xor M, a	Perform logic XOR on ACC and memory, then put result into memory Example: *xor* MEM, a ; Result: MEM ← a ^ MEM Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV
not a	Perform 1's complement (logical complement) of ACC Example: *not* a ; Result: a ← ~a Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV Application Example: ----------

	mov a, 0x38 ; // ACC=0X38 *not* a ; // ACC=0XC7
not M	Perform 1's complement (logical complement) of memory Example: *not* MEM ; Result: MEM ← ～MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV Application Example: *mov* a, 0x38 ; *mov* mem, a ; // mem = 0x38 *not* mem ; // mem = 0xC7
neg a	Perform 2's complement of ACC Example: *neg* a; Result: a ← 〒a Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV Application Example: *mov* a, 0x38 ; // ACC=0X38 *neg* a ; // ACC=0XC8
neg M	Perform 2's complement of memory Example: *neg* MEM; Result: MEM ← 〒MEM Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV Application Example: *mov* a, 0x38 ; *mov* mem, a ; // mem = 0x38 *not* mem ; // mem = 0xC8
comp a, M	Compare ACC with the content of memory Example: *comp a, MEM;* Result: Flag will be changed by regarding as (a - MEM) Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV Application Example: *mov a, 0x38 ;* *mov mem, a ;* *comp a, mem ; // Z flag is set as 1* *mov a, 0x42 ;* *mov mem, a ;*

```
    mov     a, 0x38 ;
    comp    a, mem ;    // C flag is set as 1
```

---

comp *M, a*

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

7.5. Bit Operation Instructions

set0 IO.n

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

set1 IO.n

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

swapc IO.n

Swap the nth bit of IO port with carry bit
Example: swapc IO.0;
Result: C ← IO.0 , IO.0 ← C
When IO.0 is a port to output pin, carry C will be sent to IO.0;
When IO.0 is a port from input pin, IO.0 will be sent to carry C;
Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV
Application Example1 (serial output) :

```
    ...
    set1    pac.0 ;     // set PA.0 as output
    ...
    set0    flag.1 ;    // C=0
    swapc   pa.0 ;      // move C to PA.0 (bit operation), PA.0=0
    set1    flag.1 ;    // C=1
    swapc   pa.0 ;      // move C to PA.0 (bit operation), PA.0=1
    ...
```

Application Example2 (serial input) :

```
    ...
    set0    pac.0 ;     // set PA.0 as input
    ...
    swapc   pa.0 ;      // read PA.0 to C (bit operation)
    src     a ;         // shift C to bit 7 of ACC
    swapc   pa.0 ;      // read PA.0 to C (bit operation)
    src     a ;         // shift new C to bit 7, old C
    ...
```

---

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

7.6. Conditional Operation Instructions

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

	Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
t1sn M.n	Check memory bit and skip next instruction if it's high EX: *t1sn* MEM.5 ; Result: If bit 5 of MEM is high, then skip next instruction Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
izsn a	Increment ACC and skip next instruction if ACC is zero Example: *izsn* a; Result: a ← a + 1,skip next instruction if a = 0 Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
dzsn a	Decrement ACC and skip next instruction if ACC is zero Example: dzsn a; Result: A ← A - 1,skip next instruction if a = 0 Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
izsn M	Increment memory and skip next instruction if memory is zero Example: izsn MEM; Result: MEM ← MEM + 1, skip next instruction if MEM= 0 Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV
dzsn M	Decrement memory and skip next instruction if memory is zero Example: dzsn MEM; Result: MEM ← MEM - 1, skip next instruction if MEM = 0 Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

7.7. System Control Instructions

call label	Function call, address can be full range address space Example: *call* function1; Result: [sp] ← pc + 1 pc ← function1 sp ← sp + 2 Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
goto label	Go to specific address which can be full range address space Example: *goto* error; Result: Go to error and execute program. Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
ret I	Place immediate data to ACC, then return Example: *ret* 0x55; Result: A← 55h ret ; Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
ret	Return to program which had function call Example: *ret;* Result: sp ← sp - 2 Pc ← [sp] Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
reti	Return to program from interrupt service routine. After this command is executed, global interrupt is enabled automatically. Example: *reti*; Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
nop	No operation

	Example: *nop*; Result: nothing changed Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
wdreset	Reset Watchdog timer. Example: *wdreset* ; Result: Reset Watchdog timer. Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
pcadd a	Next program counter is current program counter plus ACC. Example: *pcadd a*; Result: pc ← pc + a Affected flags: 『N』Z 『N』C 『N』AC 『N』OV Application Example: --- … mov a, 0x02 ; pcadd a ; // PC <- PC+2 goto err1 ; goto correct ; // jump here goto err2 ; goto err3 ; … correct: // jump here … ---
engint	Enable global interrupt enable Example: *engint*; Result: Interrupt request can be sent to FPP0 Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
disgint	Disable global interrupt enable Example: *disgint* ; Result: Interrupt request is blocked from CPU Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
stopsys	System halt. Example: *stopsys*; Result: Stop the system clocks and halt the system Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
stopexe	CPU halt. The oscillator module is still active to output clock, however, system clock is disabled to save power. Example: *stopexe*; Result: Stop the system clocks and keep oscillator modules active. Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
reset	Reset the whole chip, its operation will be same as hardware reset. Example: *reset*; Result: Reset the whole chip. Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

7.8. Summary of Instructions Execution Cycle

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

7.9. Summary of affected flags by Instructions

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

7.10. BIT definition

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

8. Code Options

Option	Selection	Description
Security	Enable	OTP content is protected and program cannot be read back
	Disable	OTP content is not protected so program can be read back
LVR	4.5V	Select LVR = 4.5V
	4.0V	Select LVR = 4.0V
	3.75V	Select LVR = 3.75V
	3.5V	Select LVR = 3.5V
	3.3V	Select LVR = 3.3V
	3.15V	Select LVR = 3.15V
	3.0V	Select LVR = 3.0V
	2.7V	Select LVR = 2.7V
	2.5V	Select LVR = 2.5V
	2.4V	Select LVR = 2.4V
	2.3V	Select LVR = 2.3V
	2.2V	Select LVR = 2.2V
	2.1V	Select LVR = 2.1V
	2.0V	Select LVR = 2.0V
	1.9V	Select LVR = 1.9V
	1.8V	Select LVR = 1.8V

9. Special Notes

This chapter is intended to remind users of common mistakes to avoid when operating the PUD310 series ICs.

9.1. Using IC

9.1.1. IO pin usage and setting

(1) IO pin is set to be digital input

- ◆ When IO is set as digital input, the level of 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) IO pin as digital input and enable wakeup function
- ◆ Configure IO pin as input
- ◆ Set corresponding bit to “1” in PXDIER
- ◆ The IO toggle width must be more than one system clock cycle.

(3) PA7 is set to be output pin
- ◆ PA7 can be set to be Open-Drain output pin only, output high requires adding pull-high resistor.

(4) PA5 is set to be PRSTB input pin
- ◆ Configure PA5 as input
- ◆ Set RSTC.0=1 to enable PA5 as PRSTB input pin

Note: Please read the PMC-APN013 carefully. According to PMC-APN013, the crystal oscillator should be used reasonably. If the following situations happen to cause IC start-up slowly or non-startup, PADAUK Technology is not responsible for this: the quality of the user's crystal oscillator is not good, the usage conditions are unreasonable, the PCB cleaner leakage current, or the PCB layouts are unreasonable.

9.1.2. Interrupt

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

Step1: Set INTEN register, enable the interrupt control bit

Step2: Clear INTRQ register

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

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

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

*Use DISGINT in the main program to disable all interrupts

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

```
void Interrupt (void)    // Once the interrupt occurs, jump to interrupt service routine
{                        // enter DISGINT status automatically, no more interrupt is accepted
    PUSHAF;
    …
    POPAF;
}   // RETI will be added automatically. After RETI being executed, ENGINT status will be restored
```

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

9.1.3. System clock switching

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

- Example : Switch system clock from IHRC to ILRC

```
CLKMD   =   0x02;          // switch to ILRC, IHRC can not be disabled here
CLKMD.1 =   0;             // IHRC can be disabled at this time
```

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

```
CLKMD   =   0x00;          // MCU will hang
```

9.1.4. Watchdog

Watchdog is open by default, but when the program executes ADJUST_IC, the watchdog will be closed. To use the watchdog, you need to reconfigure the open. Watchdog will be inactive once ILRC is disabled.

9.1.5. TIMER time out

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

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

9.1.6. IHRC

(1) The IHRC frequency calibration is performed when IC is programmed by the writer.

(2) Because the characteristic of the Epoxy Molding Compound (EMC) would some degrees affects the IHRC frequency (either for package or COB), if the calibration is done before molding process, the actual IHRC frequency after molding may be deviated or becomes out of spec. Normally, the frequency is getting slower a bit.

(3) It usually happens in COB package or Quick Turnover Programming (QTP). And PADAUK would not take any responsibility for this situation.

(4) Users can make some compensatory adjustments according to their own experiences. For example, users can set IHRC frequency to be 0.5% ~ 1% higher and aim to get better re-targeting after molding.

9.1.7. LVR

LVR level selection is done at compile time. User must select LVR based on the system working frequency and power supply voltage to make the MCU work stably.

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

SYSCLK	VDD	LVR	Temp.
1.5MHz	≧ 1.8V	≧ 1.8V	25℃
3MHz	≧ 2.0V	≧ 2.0V	
6MHz	≧ 2.0V	≧ 2.0V	
12MHz	≧ 2.3V	≧ 2.3V	

(1) The setting of LVR (1.8V ~ 4.5V) will be valid just after successful power-on process.

(2) User can set MISC2.4 as "0" to disable LVR. However, VDD must be kept as exceeding the lowest working voltage of chip; Otherwise IC may work abnormally.

(3) The LVR function will be invalid when IC in stopexe or stopsys mode.

9.1.8. Programming Writing

There are 4 pins for using the writer to program: DP(PA4), DM(PA6), VDD and GND.
Please use 5S-P-003x or later version to program PUD310 real chip. (3S-P-002 or elder versions do not support programming PUD310)

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

(1) VDD_T may be higher than 9.5V, and its maximum current may reach about 20mA.

(2) All other signal pins level (except GND) are the same as VDD_T.

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.

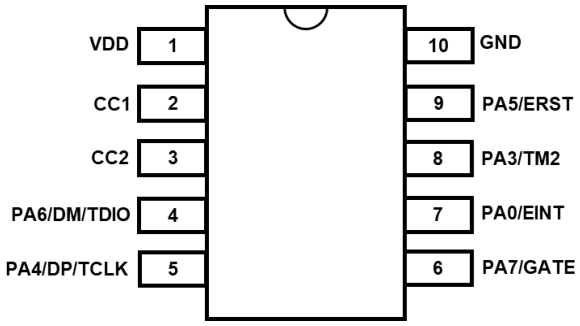
Important Cautions：

- **You MUST follow the instructions on APN004 and APN011 for programming IC on the handler.**
- **Connecting a 0.01uF capacitor between V_{BAT} 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 be fail.**

For 5S-P-003x to write PUD310, 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 linker web page.

http://www.padauk.com.tw/en/technical/index.aspx?kind=27

for example, make JP7 writer signal connection of ESSOP-10, as the following.



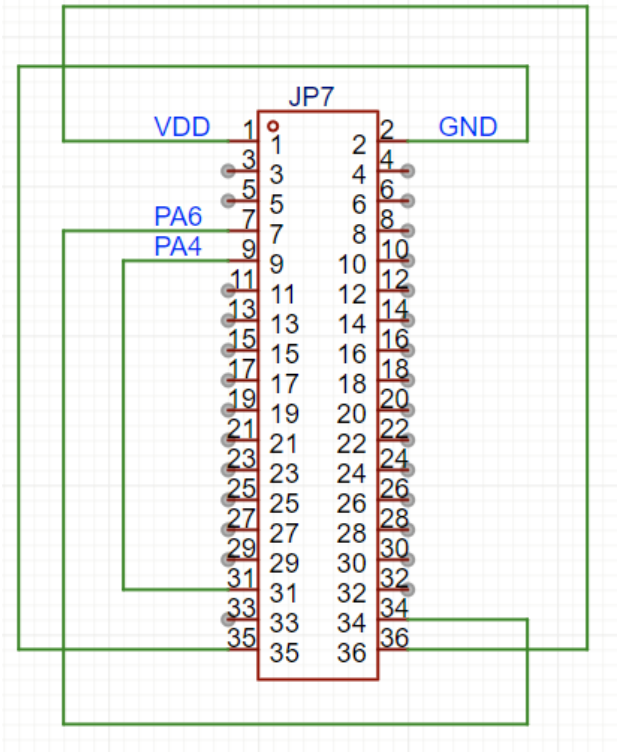
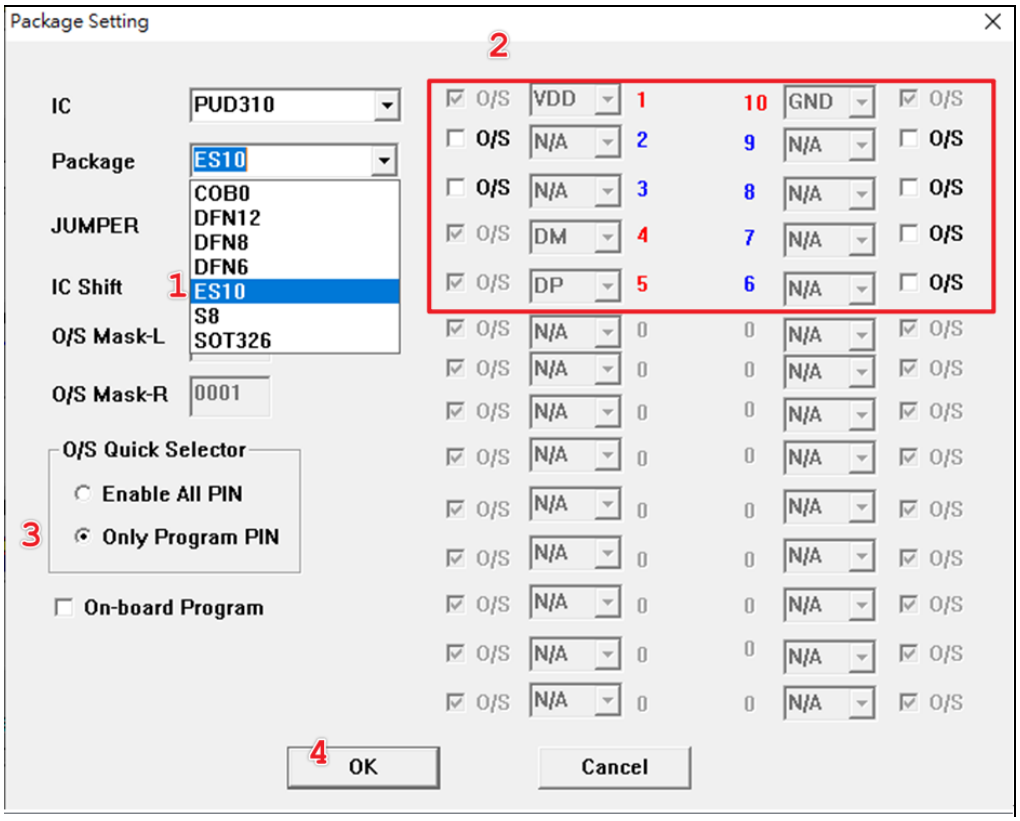


Fig. 9.1: schematic diagram of Jumper7 for P-003x

Load PDK from GUI, insert JP7 and then input IC on the socket without shift. After LCDM displays IC ready, it can be written.


Package Setting
2
IC
PUD310
Package
ES10
COB0
DFN12
DFN8
DFN6
ES10
S8
SOT326
JUMPER
IC Shift
1
O/S Mask-L
O/S Mask-R
0001
O/S Quick Selector
Enable All PIN
Only Program PIN
3
On-board Program
O/S
VDD
1
10
GND
N/A
2
9
3
8
DM
4
7
DP
5
6
0
4
OK
Cancel