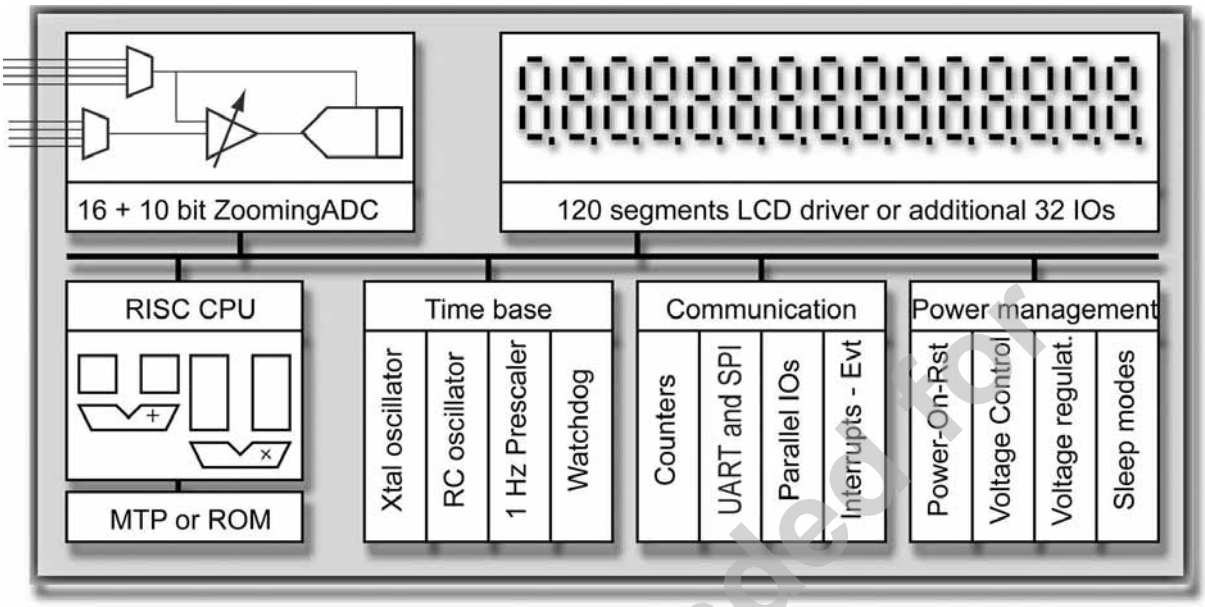
XE8802 Sensing Machine Data Acquisition MCU with ZoomingADC™ and LCD driver



XE8802 Sensing Machine

Data Acquisition SoC with 16+10 bit ZoomingADC™

Embedded RISC controller and LCD driver

General Description

The XE8802 includes a high resolution acquisition path with the 16+10 bits ZoomingADC™ and an LCD driver for up to 120 segments. The LCD lines can be used as additional I/Os.

The embedded core of the XE8802 is an ultra low-power low-voltage microcontroller unit (MCU) with a strict 1 instruction per clock processing and branch speed, allowing for a sustained 4MIPS at 1200µA under 2.4 to 5.5 V.

The XE8802 is available with on chip ROM or Multiple-Time-Programmable (MTP) program memory.

Applications

- Portable, battery operated instruments
- Medical portable instrument
- Remote control
- HVAC control
- Metering
- Sports watches, wrist instruments

Key product Features

- Low-power, high resolution ZoomingADC
  - 16-bits sigma-delta type ADC
  - 0.5 to 1000 gain with offset cancellation
  - up to 13 inputs multiplexer, 8 dedicated inputs
- 4 low power comparators
- Low-voltage low-power controller operation
  - 2 or 4 MIPS max, 2.4 to 5.5 V operation
  - 300µA per 1MIPS over full voltage range
  - 7MIPS or 1.2V operation in ROM
  - Hardware MULT
- up to 22kByte (8 kInstruction) MTP
- 1032Byte RAM data memory
- RC and crystal oscillators
- 5 reset, 22 interrupt, 8 event sources
- 36 I/O lines, hardware UART and SPI
- 120 segments LCD driver, can be used as 32 extra I/O lines
- 100 years MTP Flash retention at 55°C

Ordering Information

Product	Temperature range	Memory type	Package
XE8802MI000	-40°C to 85 °C	MTP	die
XE8802MI035LF	-40°C to 85 °C	MTP	LQFP100

TABLE OF CONTENTS

1. General overview

1.1 Top schematic

The top-level block schematic of the circuit is shown in Figure 1-1. The heart of the circuit consists of the Coolrisc816 CPU core. This core includes an 8x8 multiplier and 16 internal registers.

The bus controller generates all control signals for access to all data registers other than the CPU internal registers.

The reset block generates the adequate reset signals for the rest of the circuit as a function of the set-up contained in its control registers. Possible reset sources are the power-on-reset (POR), the external pin NRESET, the watchdog (WD), a bus error detected by the bus controller or a programmable pattern on Port A. Different low power modes are implemented.

The clock generation and power management block sets up the clock signals and generates internal supplies for different blocks. The clock can be generated from the RC oscillator (this is the start-up condition), the crystal oscillator (XTAL) or an external clock source (given on the XIN pin).

The test controller generates all set-up signals for different test modes. In normal operation, it is used as a set of 8 low power data registers. If power consumption is important for the application, the variables that need to be accessed frequently should be stored in these registers rather than in the RAM.

The IRQ handler routes the interrupt signals of the different peripherals to the IRQ inputs of the CPU core. It allows masking of the interrupt sources and it flags which interrupt source is active.

Events are generally used to restart the processor after a HALT period without jumping to a specified address, i.e. the program execution resumes with the instruction following the HALT instruction. The EVN handler routes the event signals of the different peripherals to the EVN inputs of the CPU core. It allows masking of the interrupt sources and it flags which interrupt source is active.

The Port B is an 8-bit parallel IO port with analog capabilities. The URST, UART, PWM and CMPD block also make use of this port.

The instruction memory is a 22-bit wide flash or ROM memory depending on the circuit version. In case of the ROM version, the VPP and NFASTREAD pins are not used. ROM versions have 8k instruction memory. Flash versions have 8k or 4k instruction memories depending on the selected operation speed.

The data memory on this product is a 1024 byte SRAM.

The Acquisition Chain is a high-resolution acquisition path with the 16+10 bits ZoomingADC™. The VMULT (voltage multiplier) powers a part of the Acquisition Chain.

The SPI is a serial interface with a master or slave configuration capability. When unused, the 4 SPI pads can be used as 4-bit wide general-purpose I/O port.

The port A is an 8 bit parallel input port. It can also generate interrupts, events or a reset. It can be used to input external clocks for the timer/counter/PWM block.

The Port D1 and the Port D2 are two general-purpose 8 bit parallel I/O ports.

The LCD driver can support a direct drive display (up to 32 segments), or multiplex 1/2, 1/3, 1/4 displays (up to 120 segments). The driver contains an on chip low-power voltage generation device VGEN. The LCD lines can be used as additional I/O pins.

The USRT (universal synchronous receiver/transmitter) contains some simple hardware functions in order to simplify the software implementation of a synchronous serial link.

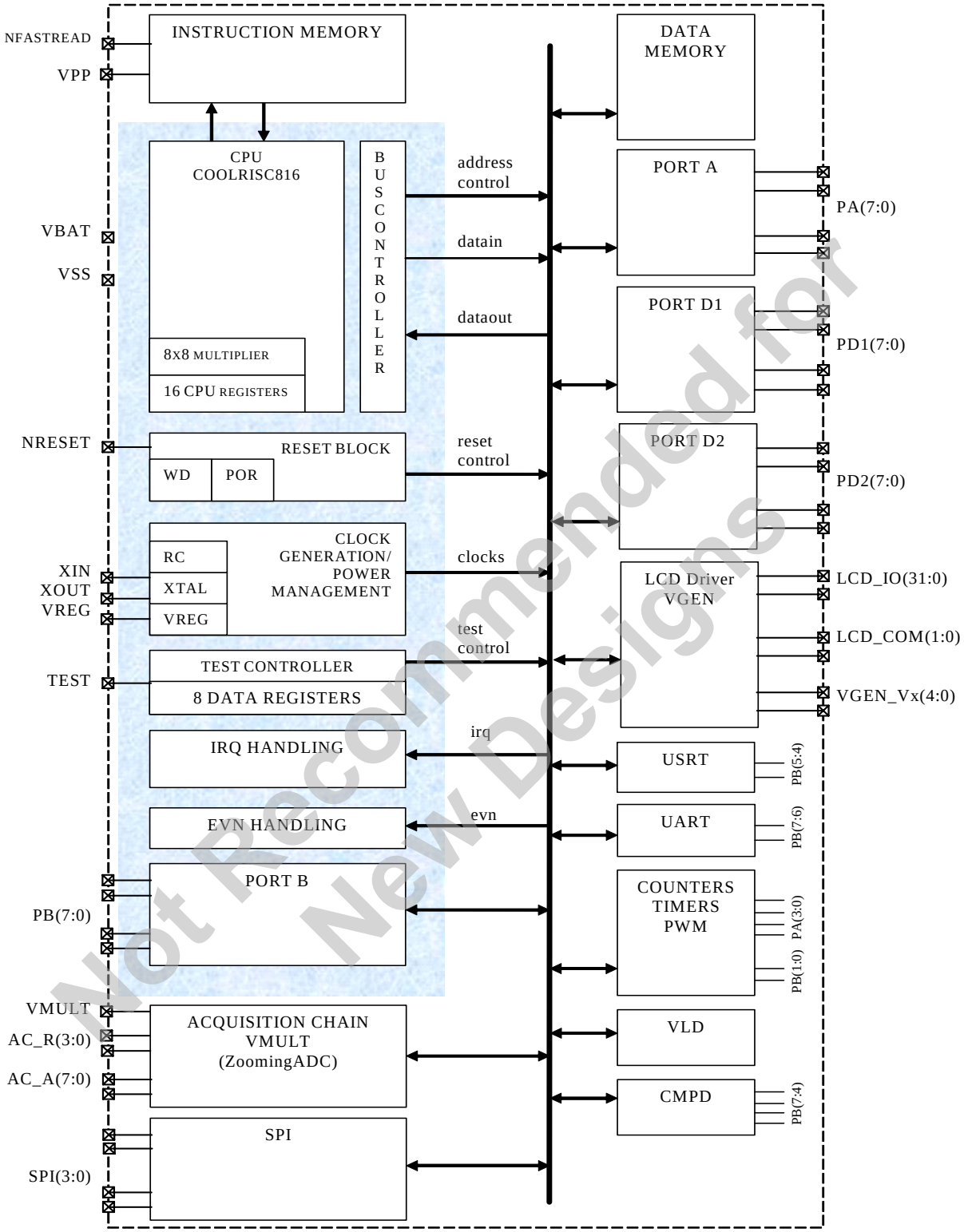


Figure 1-1. Block schematic of the XE8802 circuit.

The UART (universal asynchronous receiver/transmitter) contains a full hardware implementation of the asynchronous serial link.

The counters/timers/PWM can take their clocks from internal or external sources (on Port A) and can generate interrupts or events. The PWM is output on Port B.

The VLD (voltage level detector) detects the battery end of life with respect to a programmable threshold.

The CMPD contains a 4-channel comparator. It is intended to monitor analog or digital signals with very low power consumption.

1.2 Pin map

The 02 can be delivered in different packages. The pin maps for the different packages are given below.

1.2.1 LQFP-100

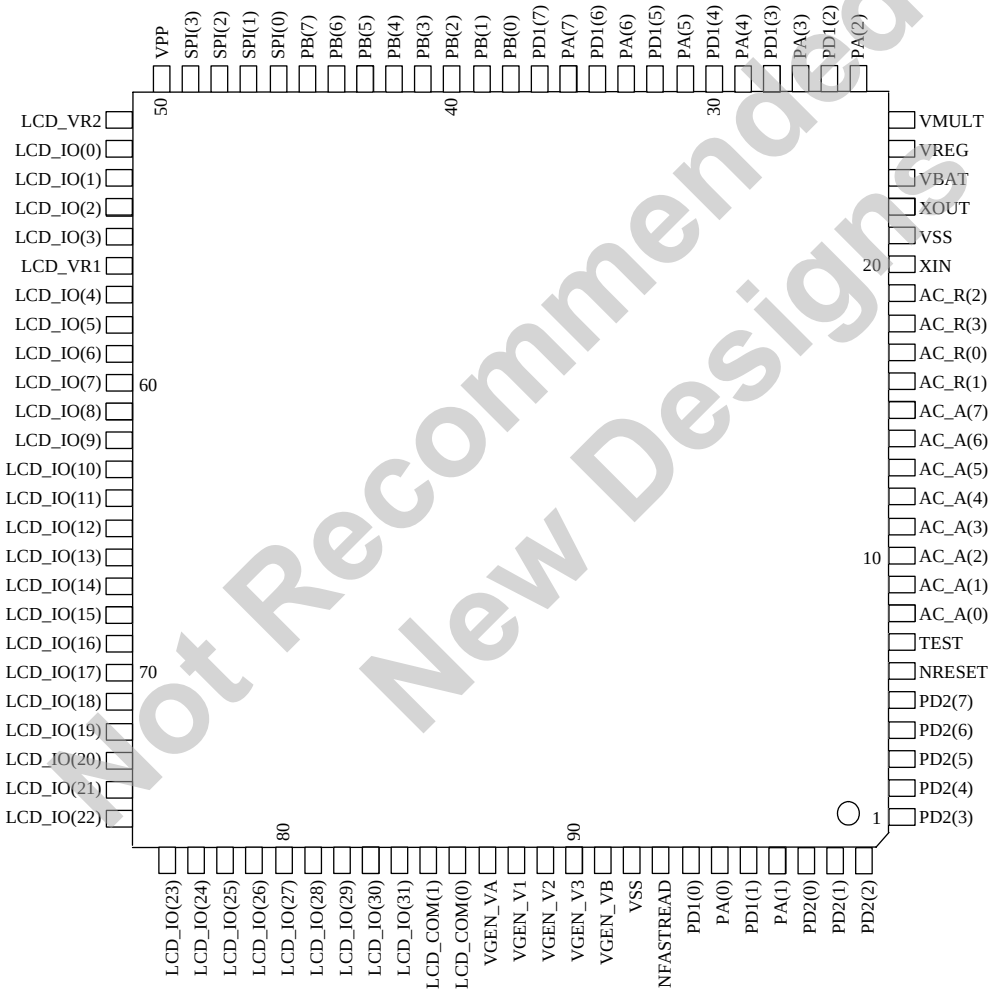


Figure 1-2. LQFP-100 pin map

Package pin	name	Package pin	name
1	PD2(3)	51	LCD_VR2
2	PD2(4)	52	LCD_IO(0)

3	PD2(5)	53	LCD_IO(1)
4	PD2(6)	54	LCD_IO(2)
5	PD2(7)	55	LCD_IO(3)
6	NRESET	56	LCD_VR1
7	TEST	57	LCD_IO(4)
8	AC_A(0)	58	LCD_IO(5)
9	AC_A(1)	59	LCD_IO(6)
10	AC_A(2)	60	LCD_IO(7)
11	AC_A(3)	61	LCD_IO(8)
12	AC_A(4)	62	LCD_IO(9)
13	AC_A(5)	63	LCD_IO(10)
14	AC_A(6)	64	LCD_IO(11)
15	AC_A(7)	65	LCD_IO(12)
16	AC_R(1)	66	LCD_IO(13)
17	AC_R(0)	67	LCD_IO(14)
18	AC_R(3)	68	LCD_IO(15)
19	AC_R(2)	69	LCD_IO(16)
20	XIN	70	LCD_IO(17)
21	VSS	71	LCD_IO(18)
22	XOUT	72	LCD_IO(19)
23	VBAT	73	LCD_IO(20)
24	VREG	74	LCD_IO(21)
25	VMULT	75	LCD_IO(22)
26	PA(2)	76	LCD_IO(23)
27	PD1(2)	77	LCD_IO(24)
28	PA(3)	78	LCD_IO(25)
29	PD1(3)	79	LCD_IO(26)
30	PA(4)	80	LCD_IO(27)
31	PD1(4)	81	LCD_IO(28)
32	PA(5)	82	LCD_IO(29)
33	PD1(5)	83	LCD_IO(30)
34	PA(6)	84	LCD_IO(31)
35	PD1(6)	85	LCD_COM(1)
36	PA(7)	86	LCD_COM(0)
37	PD1(7)	87	VGEN_VA
38	PB(0)	88	VGEN_V1
39	PB(1)	89	VGEN_V2
40	PB(2)	90	VGEN_V3
41	PB(3)	91	VGEN_VB
42	PB(4)	92	VSS
43	PB(5)	93	NFASTREAD
44	PB(6)	94	PD1(0)
45	PB(7)	95	PA(0)
46	SPI(0)	96	PD1(1)
47	SPI(1)	97	PA(1)
48	SPI(2)	98	PD2(0)
49	SPI(3)	99	PD2(1)
50	VPP	100	PD2(2)

Table 1-1. Bonding plan of the LQFP-100 package (LQFP 100L 14x14mm thick 1.6 mm)

1.2.2 LQFP-80

Package pin	name	Package pin	name
1	NRESET	41	LCD_VR1
2	TEST	42	LCD_IO(4)
3	AC_A(0)	43	LCD_IO(5)
4	AC_A(1)	44	LCD_IO(6)
5	AC_A(2)	45	LCD_IO(7)
6	AC_A(3)	46	LCD_IO(8)
7	AC_A(4)	47	LCD_IO(9)
8	AC_A(5)	48	LCD_IO(10)
9	AC_A(6)	49	LCD_IO(11)
10	AC_A(7)	50	LCD_IO(12)
11	AC_R(1)	51	LCD_IO(13)
12	AC_R(0)	52	LCD_IO(14)
13	AC_R(3)	53	LCD_IO(15)
14	AC_R(2)	54	LCD_IO(16)
15	XIN	55	LCD_IO(17)
16	VSS	56	LCD_IO(18)
17	XOUT	57	LCD_IO(19)
18	VBAT	58	LCD_IO(20)
19	VREG	59	LCD_IO(21)
20	VMULT	60	LCD_IO(22)
21	PA(2)/ PD1(2)	61	LCD_IO(23)
22	PA(3)/ PD1(3)	62	LCD_IO(24)
23	PA(4)/ PD1(4)	63	LCD_IO(25)
24	PA(5)/ PD1(5)	64	LCD_IO(26)
25	PA(6)/ PD1(6)	65	LCD_IO(27)
26	PA(7)	66	LCD_IO(28)
27	PD1(7)	67	LCD_IO(29)
28	PB(0)	68	LCD_IO(30)
29	PB(1)	69	LCD_IO(31)
30	PB(2)	70	LCD_COM(1)
31	PB(3)	71	LCD_COM(0)
32	PB(4)	72	VGEN_VA
33	PB(5)	73	VGEN_V1
34	PB(6)	74	VGEN_V2
35	PB(7)	75	VGEN_V3
36	SPI(0)	76	VGEN_VB
37	SPI(1)	77	VSS
38	SPI(2)	78	NFASTREAD
39	SPI(3)	79	PA(0)/PD1(0)
40	VPP	80	PA(1)/ PD1(1)

Table 1-2. Bonding plan of the LQFP-80 package (LQFP 80L 14x14mm thick 1.6 mm)

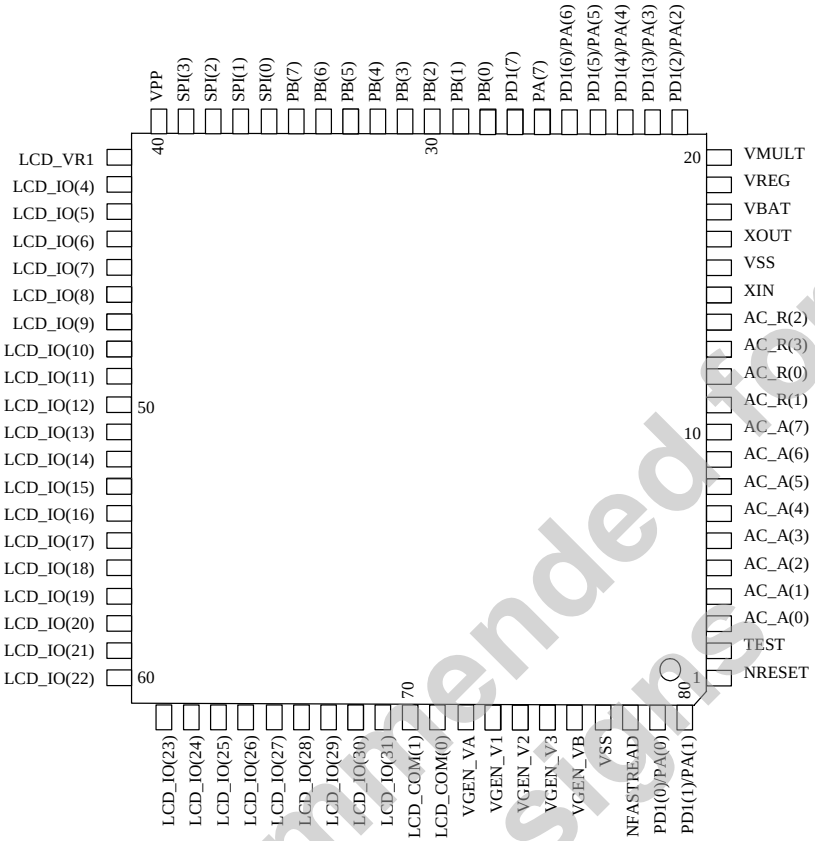


Figure 1-3. LQFP- 80 pin map

1.3 Bare die

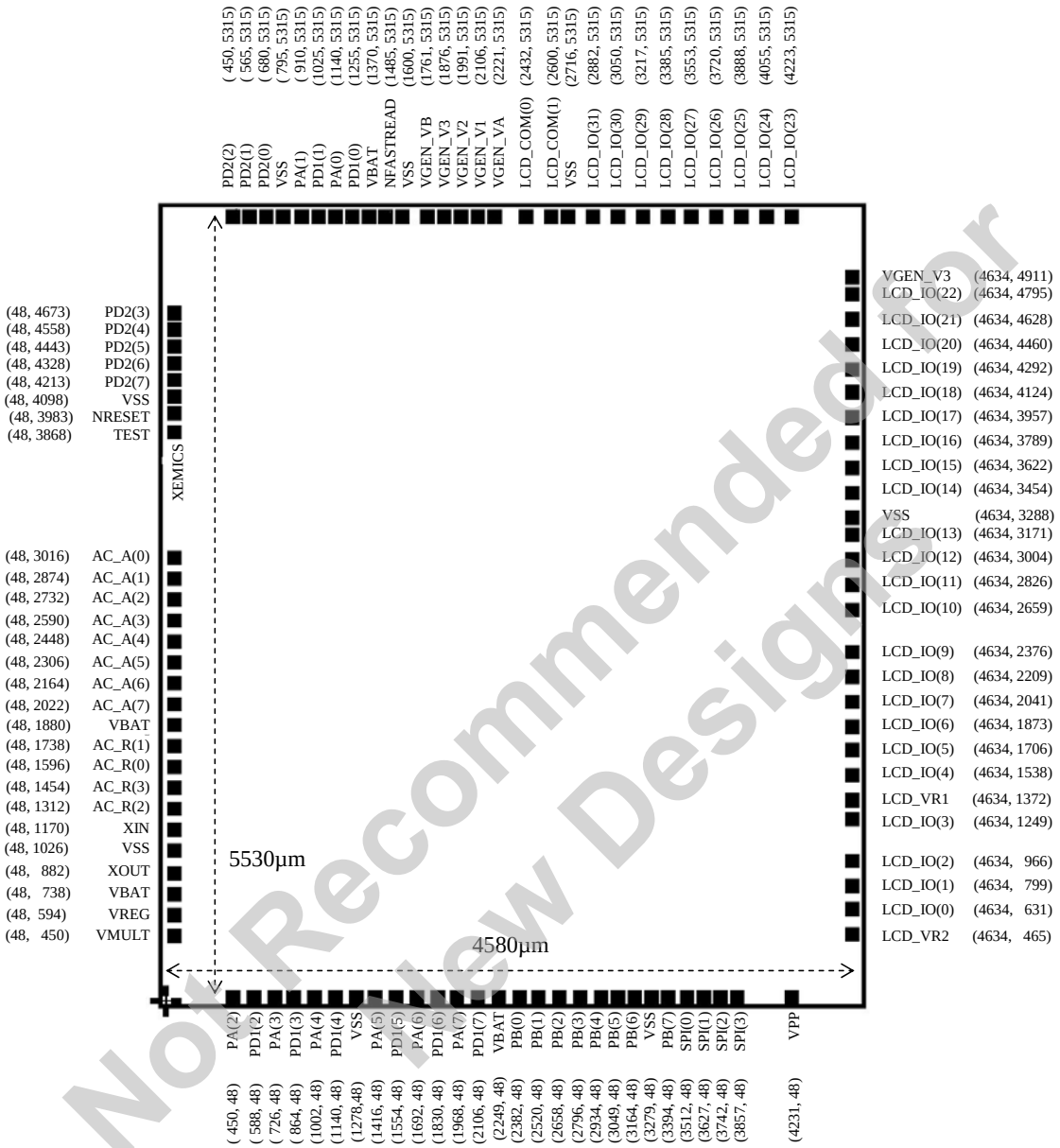


Figure 1-4. Bare die dimensions and pin locations.

1.4 Pin assignment

The table below gives a short description of the different pin assignments.

Pin	Assignment
VBAT	Positive power supply
VSS	Negative power supply
VREG	Connection for the mandatory external capacitor of the voltage regulator
VPP	High voltage supply for flash memory programming (NC in ROM versions)
NFASTREAD	Selects the maximal operation speed and the flash memory size (NC in ROM versions)
NRESET	Resets the circuit when the voltage is low
TEST	Sets the pin to flash programming mode
XIN/XOUT	Quartz crystal connections, also used for flash memory programming
PA(7:0)	Parallel input port A pins
PB(7:0)	Parallel I/O port B pins
PD1(7:0)	Parallel I/O port D1 pins
PD2(7:0)	Parallel I/O port D2 pins
SPI(3:0)	Serial SPI port or general purpose I/O port pins
LCD_IO(29:0)	LCD segment driver or general purpose I/O port pins
LCD_IO(31:30)	LCD segment driver or LCD back plane driver or general purpose I/O port pins
LCD_COM(1:0)	LCD back plane driver pins
LCD_VR1/LCD_VR2	LCD supply voltage
VGEN_Vx	LCD driver voltage generation pins
AC_A(7:0)	Acquisition chain input pins
AC_R(3:0)	Acquisition chain reference pins
VMULT	Connection for external capacitor of the voltage multiplier

Table 1-3. Pin assignment

Table 1-4 gives a more detailed pin map for the different pins in the different packages. It also indicates the possible I/O configuration of these pins. The indications in blue bold are the configuration at start-up. Please note that in the LQFP-80 package several functions are routed to the same package pins. These pins are indicated in red italic.

Pin number		Function			I/O configuration									
lqfp-100	lqfp-80	first	second	third	AI	AO	DI	DO	OD	PU	PD	SNAP	LCD	POWER
1		PD2(3)					**X**	X		**X**		X		
2		PD2(4)					**X**	X		**X**		X		
3		PD2(5)					**X**	X		**X**		X		
4		PD2(6)					**X**	X		**X**		X		
5		PD2(7)					**X**	X		**X**		X		
6	1	NRESET					**X**			**X**				
7	2	TEST					**X**				**X**			
8	3	AC_A(0)			**X**									
9	4	AC_A(1)			**X**									
10	5	AC_A(2)			**X**									
11	6	AC_A(3)			**X**									
12	7	AC_A(4)			**X**									
13	8	AC_A(5)			**X**									
14	9	AC_A(6)			**X**									
15	10	AC_A(7)			**X**									
16	11	AC_R(1)			**X**									

17	12	AC_R(0)			X									
18	13	AC_R(3)			X									
19	14	AC_R(2)			X									
20	15	XIN			X		X							
21	16	VSS												X
22	17	XOUT				X	X							
23	18	VBAT												X
24	19	VREG				X								
25	20	VMULT				X								
26	*21*	PA(2)	CNTC				X			X		X		
27	*21*	PD1(2)					X	X		X		X		
28	*22*	PA(3)	CNTD				X			X		X		
29	*22*	PD1(3)					X	X		X		X		
30	*23*	PA(4)					X			X		X		
31	*23*	PD1(4)					X	X		X		X		
32	*24*	PA(5)					X			X		X		
33	*24*	PD1(5)					X	X		X		X		
34	*25*	PA(6)					X			X		X		
35	*25*	PD1(6)					X	X		X		X		
36	26	PA(7)					X			X		X		
37	27	PD1(7)					X	X		X		X		
38	28	PB(0)	PWM0		X	X	X	X	X	X				
39	29	PB(1)	PWM1		X	X	X	X	X	X				
40	30	PB(2)			X	X	X	X	X	X				
41	31	PB(3)			X	X	X	X	X	X				
42	32	PB(4)	USRT_S0	CMPD(0)	X	X	X	X	X	X				
43	33	PB(5)	USRT_S1	CMPD(1)	X	X	X	X	X	X				
44	34	PB(6)	UART_Tx	CMPD(2)	X	X	X	X	X	X				
45	35	PB(7)	UART_Rx	CMPD(3)	X	X	X	X	X	X				
46	36	SPI(0)					X	X		X				
47	37	SPI(1)					X	X		X				
48	38	SPI(2)					X	X		X				
49	39	SPI(3)					X	X		X				
50	40	VPP												X
51		LCD_VR2			X									
52		LCD_IO(0)					X	X		X			X	
53		LCD_IO(1)					X	X		X			X	
54		LCD_IO(2)					X	X		X			X	
55		LCD_IO(3)					X	X		X			X	
56	41	LCD_VR1			X									
57	42	LCD_IO(4)					X	X		X			X	
58	43	LCD_IO(5)					X	X		X			X	
59	44	LCD_IO(6)					X	X		X			X	
60	45	LCD_IO(7)					X	X		X			X	
61	46	LCD_IO(8)					X	X		X			X	
62	47	LCD_IO(9)					X	X		X			X	
63	48	LCD_IO(10)					X	X		X			X	
64	49	LCD_IO(11)					X	X		X			X	
65	50	LCD_IO(12)					X	X		X			X	
66	51	LCD_IO(13)					X	X		X			X	
67	52	LCD_IO(14)					X	X		X			X	

68	53	LCD_IO(15)					X	X		X			**X**	
69	54	LCD_IO(16)					X	X		X			**X**	
70	55	LCD_IO(17)					X	X		X			**X**	
71	56	LCD_IO(18)					X	X		X			**X**	
72	57	LCD_IO(19)					X	X		X			**X**	
73	58	LCD_IO(20)					X	X		X			**X**	
74	59	LCD_IO(21)					X	X		X			**X**	
75	60	LCD_IO(22)					X	X		X			**X**	
76	61	LCD_IO(23)					X	X		X			**X**	
77	62	LCD_IO(24)					X	X		X			**X**	
78	63	LCD_IO(25)					X	X		X			**X**	
79	64	LCD_IO(26)					X	X		X			**X**	
80	65	LCD_IO(27)					X	X		X			**X**	
81	66	LCD_IO(28)					X	X		X			**X**	
82	67	LCD_IO(29)					X	X		X			**X**	
83	68	LCD_IO(30)					X	X		X			**X**	
84	69	LCD_IO(31)					X	X		X			**X**	
85	70	LCD_COM(1)											**X**	
86	71	LCD_COM(0)											**X**	
87	72	VGEN_VA			**X**									
88	73	VGEN_V1			**X**	X								
89	74	VGEN_V2			**X**	X								
90	75	VGEN_V3			**X**	X								
91	76	VGEN_VB			**X**									
92	77	VSS												**X**
93	78	NFASTREAD					**X**					**X**		
94	*79*	PD1(0)					**X**	X		**X**		X		
95	*79*	PA(0)	CNTA				**X**			**X**		X		
96	*80*	PD1(1)					**X**	X		**X**		X		
97	*80*	PA(1)	CNTB				**X**			**X**		X		
98		PD2(0)					**X**	X		**X**		X		
99		PD2(1)					**X**	X		**X**		X		
100		PD2(2)					**X**	X		**X**		X		

Pin map table legend:

blue bold: configuration at start up

AI: analog input
AO: analog output
DI: digital input
DO: digital output
OD: nMOS open drain output
PU: pull-up resistor
PD: pull-down resistor
SNAP: snap-to-rail function (see peripheral description for detailed description)
POWER: power supply

Table 1-4. Pin description table

2 XE8802 Performance

2.1 Absolute maximum ratings

Table 2-1. Absolute maximal ratings

	Min.	Max.		Note
Voltage applied to VBAT with respect to VSS	-0.3	6.0	V	
Voltage applied to VPP with respect to VSS	VBAT-0.3	12	V	
Voltage applied to all pins except VPP and VBAT	VSS-0.3	VBAT+0.3	V	
Storage temperature (ROM device or unprogrammed flash device)	-55	150	°C	
Storage temperature (programmed flash device)	-40	85	°C	

Stresses beyond the absolute maximal ratings may cause permanent damage to the device. Functional operation at the absolute maximal ratings is not implied. Exposure to conditions beyond the absolute maximal ratings may affect the reliability of the device.

2.2 Operating range

Table 2-2. Operating range for the flash device

	Min.	Max.		Note
Voltage applied to VBAT with respect to VSS	2.4	5.5	V	
Voltage applied to VBAT with respect to VSS during the flash programming	4.5	5.5	V	1
Voltage applied to VPP with respect to VSS	VBAT	11.5	V	
Voltage applied to all pins except VPP and VBAT	VSS	VBAT	V	
Operating temperature range	-40	85	°C	
Capacitor on VREG (flash version)	0.8	1.2	µF	2
Capacitor on VMULT	1.0	3.0	nF	3

1. During the programming of the device the temperature must be between 10°C and 40°C.
2. The capacitor on VREG is mandatory.
3. The capacitor on VMULT is optional. The capacitor has to be present if the multiplier is enabled. The multiplier has to be enabled if VBAT<3.0V.

Table 2-3. Operating range for the ROM device

			Min.	Max.		Note
Voltage applied to VBAT with respect to VSS	Acquisition chain off	VREG by-passed	1.2	5.5	V	2
		VREG on	1.5	5.5	V	
	Acquisition chain on	VMULT on	2.4	5.5	V	
		VMULT off	3.0	5.5	V	
Voltage applied to all pins except VPP and VBAT			VSS	VBAT	V	
Operating temperature range			-40	125	°C	
Capacitor on VREG			0.1	1.2	µF	1
Capacitor on VMULT			1.0	3.0	nF	3

1. The capacitor may be omitted when VREG is connected to VBAT.
2. The voltage reference for the LCD drivers starts operating at 1.5 V.
3. The capacitor on VMULT is optional. The capacitor has to be present if the multiplier is enabled. The multiplier has to be enabled if VBAT<3.0V.

All specifications in this document are valid for the complete operating range unless otherwise specified.

Table 2-4. Operating range of the Flash memory

	Min.	Max.		Note
Retention time at 85°C	10		years	1
Retention time at 55°C	100		years	1
Number of programming cycles	10			2

1. Valid only if programmed using a programming tool that is qualified
2. Circuits can be programmed more than 10 times but in that case, the retention time is no longer guaranteed.

2.3 Current consumption

The tables below give the current consumption for the circuit in different configurations. The figures are indicative only and may change as a function of the actual software implemented in the circuit.

Table 2-5 gives the current consumption for the flash version of the circuit. The peripherals are disabled. The parallel ports are configured in input with pull up. Their pins are not connected externally.

Table 2-5. Typical current consumption of the XE8802M version (8k instructions flash memory)

Operation mode	CPU	RC	Xtal	Consumption	comments	Note
High speed CPU	1 MIPS	1 MHz	Off	200 µA	2.4V<>5.5V, 27°C	1
				320 µA		2
				410 µA		3
				310 µA		4
Low speed CPU	.1 MIPS	100 kHz	Off	21 µA	2.4V <>5.5V, 27°C	1
				33 µA		2
				42 µA		3
Low power CPU	32 kIPS	Off	32 kHz	7.5 µA	2.4V <>5.5V, 27°C	1
				11.0 µA		2
				14.5 µA		3
Low power time keeping	HALT	Off	32 kHz	1.9 µA	2.4V <>5.5V, 27°C	
Fast wake-up time keeping	HALT	Ready	32kHz	2.3 µA	2.4V <>5.5V, 27°C	
Immediate wake-up time keeping	HALT	1 MHz	Off	35 µA	2.4V <>5.5V, 27°C	
VLD static current				15 µA	2.4V <>5.5V, 27°C	
CMPD static current				2 µA	2.4V <>5.5V, 27°C	

1. Software without data access
2. 100% low power RAM access
3. 100% RAM access
4. typical software

Table 2-6. Current consumption of the XE8802R version (8k instructions ROM memory)

Operation mode	CPU	RC	Xtal	Consumption	comments	Note
High speed CPU	1 MIPS	1 MHz	Off	200	2.4V<>5.5V, 27°C	1
Max. Speed CPU	4 MIPS	4 MHz	Off	800	2.4V<>5.5V, 27°C	1
Low speed CPU	.1 MIPS	100 kHz	Off	21	2.4V <>5.5V, 27°C	1
Low power CPU	32 kIPS	Off	32 kHz	7	2.4V <>5.5V, 27°C	1
Low voltage CPU	32 kIPS	Off	32 kHz	1	1.2V, 27°C	1
Low power time keeping	HALT	Off	32 kHz	1.3	2.4V <>5.5V, 27°C	

1. Software using MOVE instruction using internal CPU registers and peripheral registers

Hints for low power operation:

1. Use the low power RAM instead of the RAM for all parameters that are accessed frequently. The average current consumption for the low power RAM is about 40 times lower than for the RAM.
2. Rather than using the circuit at low speed, it is better to use the circuit at higher speed and switch off the blocks when not needed.
3. The power consumption of the program memory is an important part of the overall power consumption. In case you intend to use a ROM version and power consumption is too high, please ask us to provide you with a circuit version with smaller ROM size.

2.4 Operating speed

2.4.1 Flash circuit version

The speed of the flash devices is not highly dependent upon the supply voltage. However, by limiting the temperature range, the speed can be increased. The minimal guaranteed speed as a function of the supply voltage and maximal temperature operating temperature is given in Figure2-2.

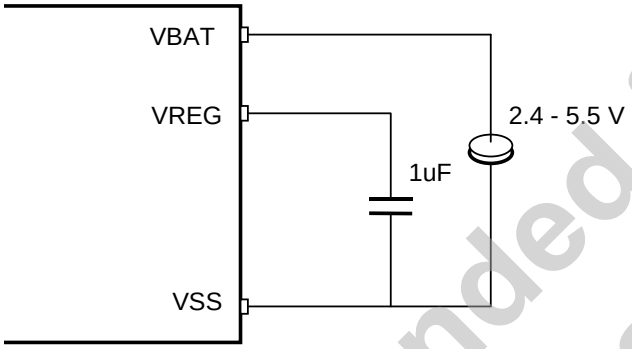


Figure 2-1. Supply configuration for flash circuit operation.

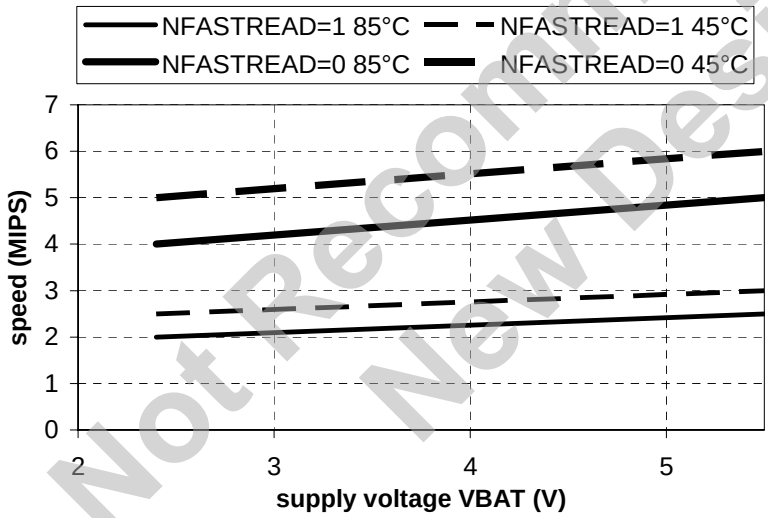


Figure 2-2. Guaranteed speed as a function of the supply voltage and maximal temperature.

The maximal speed of the device depends on the setting of the NFASTREAD pin. If NFASTREAD=1, the full 8k instructions of the memory are available and the speed is limited to 2-3MIPS. If NFASTREAD=0, only 4k instructions are available but the speed is doubled.

Important Note
The circuit has to be programmed for the correct option of the NFASTREAD pin. If not, the execution of the software may not be correct.

2.4.2 ROM circuit version with regulator on

For the ROM version, two possible operating modes exist: with and without voltage regulator. Using the voltage regulator, low power consumption will be obtained even with supply voltages above 2.4V. Without the voltage regulator (i.e. VREG short-circuited to VBAT), a higher speed can be obtained.

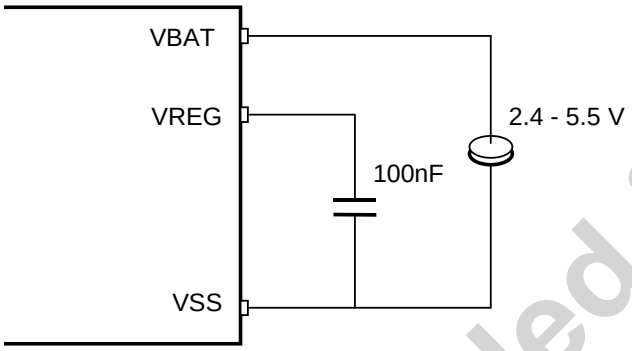


Figure 2-3. Supply configuration for ROM circuit operation using the internal regulator.

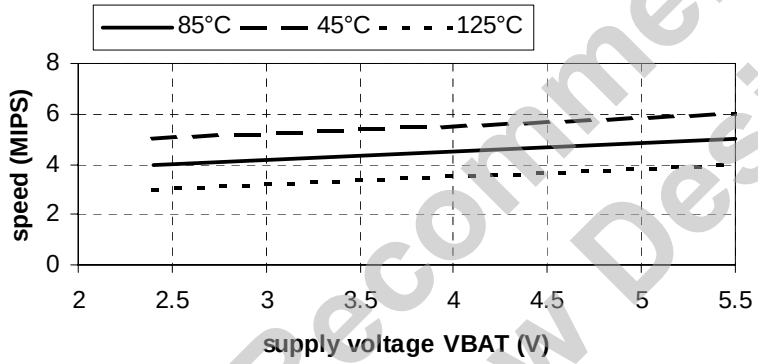


Figure 2-4. Guaranteed speed as a function of supply voltage and for different maximal temperatures using the voltage regulator.

2.4.3 ROM circuit version with regulator off

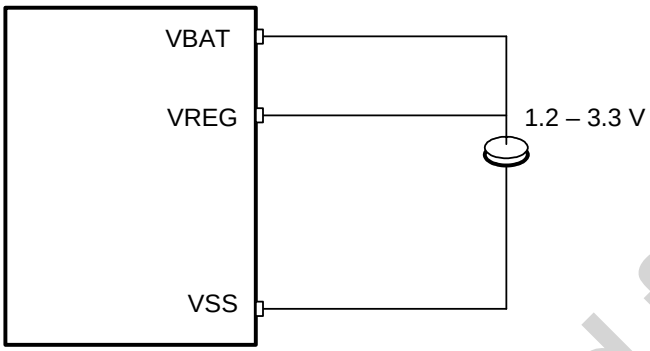


Figure 2-5. Supply configuration for ROM circuit operation by-passing the internal regulator.

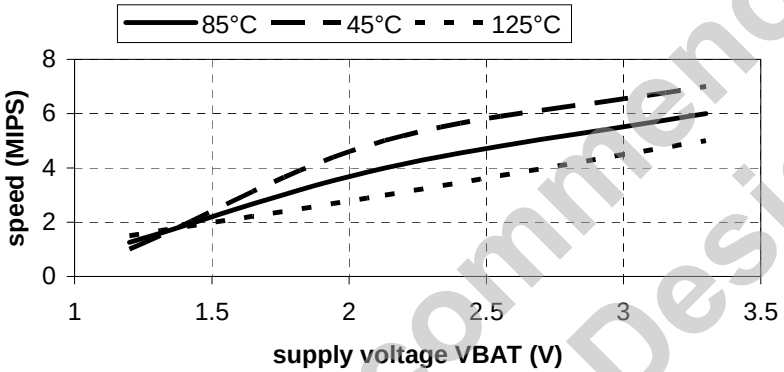


Figure 2-6. Guaranteed speed as a function of supply voltage and for two temperature ranges when VREG=VBAT.

Important Note
Note that the acquisition chain will not operate if VBAT is below 2.4V. The internal reference voltage for the LCD will not operate below 1.5V. If the internal reference is not used, the LCD voltage generator and the LCD driver will operate down to 1.2V. The operation range of the different blocks is summarized in Figure 2-7.

2.5 Simplified supply selection criteria

- MTP devices always require the capacitor on VREG and VREG cannot be shorted to VBAT on MTP devices.
- ROM devices can operate 1.5 V to 5.5 V with lowest current requirement with the capacitor on VREG, and VREG not shorted to VBAT.
- ROM devices can operate 1.2 V to 3.3 V at highest speed with VREG shorted to VBAT.
- If operation is always above 3.0 V, the capacitor on VMULT is not needed and VMULT can be always off.
- If the acquisition chain is used between 2.4 V and 3.0 V, then the capacitor on VMULT must be present and VMULT must be set on during operation below 3.0 V.
- The acquisition chain does not operate below 2.4 V.
- The internal reference voltage for the LCD does not operate below 1.5 V.

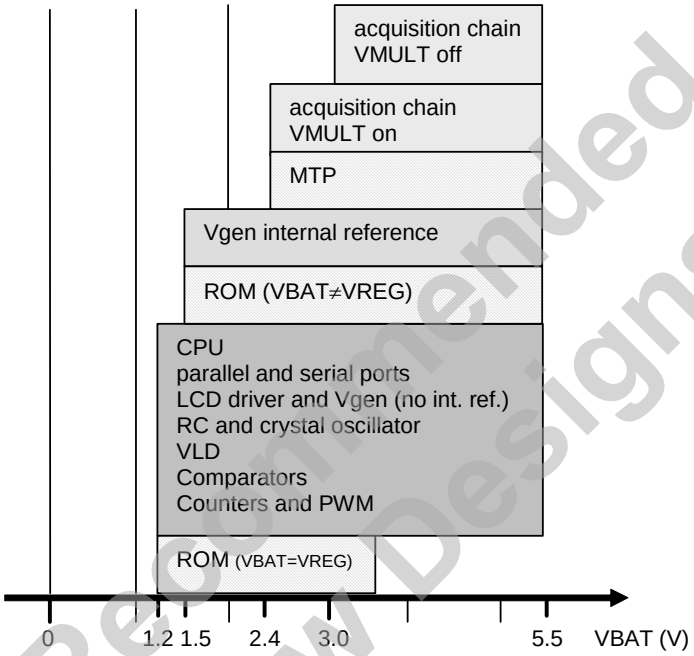


Figure 2-7. Operating voltage range of the different circuit blocks. MTP devices do not operate below 2.4V. ROM devices can operate in different voltage ranges if VREG and VBAT are short circuited or not.

3. CPU

3.1 CPU description

The CPU of the XE8000 series is a low power RISC core. It has 16 internal registers for efficient implementation of the C compiler. Its instruction set is made up of 35 generic instructions, all coded on 22 bits, with 8 addressing modes. All instructions are executed in one clock cycle, including conditional jumps and 8x8 multiplication. The circuit therefore runs on 1 MIPS on a 1MHz clock.

The CPU hardware and software description is given in the document “Coolrisc816 Hardware and Software Reference Manual”. A short summary is given in the following paragraphs.

The good code efficiency of the CPU core makes it possible to compute a polynomial like $Z = (A_0 + A_1 \cdot Y) \cdot X + B_0 + B_1 \cdot Y$ in less than 300 clock cycles (software code generated by the XEMICS C-compiler, all numbers are signed integers on 16 bits).

3.2 CPU internal registers

As shown in Figure 3-1, the CPU has 16 internal 8-bit registers. Some of these registers can be concatenated to a 16-bit word for use in some instructions. The function of these registers is defined in Table 3-1. The status register stat (Table 3-2) is used to manage the different interrupt and event levels. An interrupt or an event can both be used to wake up after a HALT instruction. The difference is that an interrupt jumps to a special interrupt function whereas an event continues the software execution with the instruction following the HALT instruction.

The program counter (PC) is a 16 bit register that indicates the address of the instruction that has to be executed. The stack (ST_n) is used to memorise the return address when executing subroutines or interrupt routines.

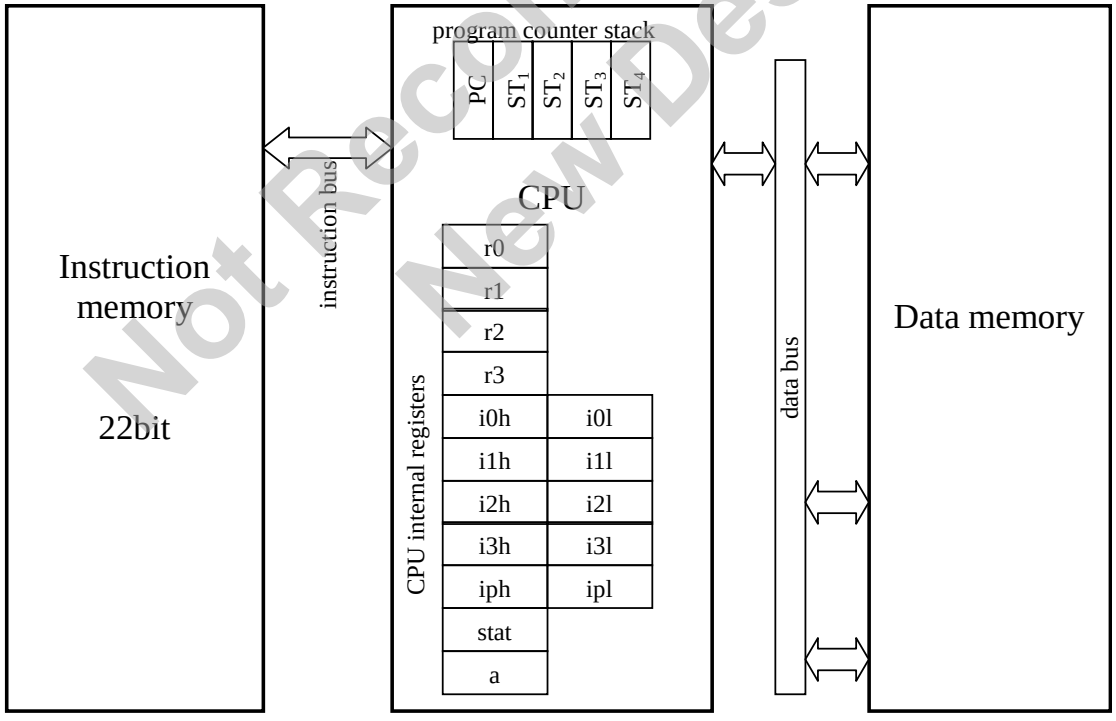


Figure 3-1. CPU internal registers

Register name	Register function
r0	general purpose
r1	general purpose
r2	general purpose
r3	data memory offset
i0h	MSB of the data memory index i0
i0l	LBS of the data memory index i0
i1h	MSB of the data memory index i1
i1l	LBS of the data memory index i1
i2h	MSB of the data memory index i2
i2l	LBS of the data memory index i2
i3h	MSB of the data memory index i3
i3l	LBS of the data memory index i3
iph	MSB of the program memory index ip
ipl	LBS of the program memory index ip
stat	status register
a	accumulator

Table 3-1. CPU internal register definition

bit	name	function
7	IE2	enables (when 1) the interrupt request of level 2
6	IE1	enables (when 1) the interrupt request of level 1
5	GIE	enables (when 1) all interrupt request levels
4	IN2	interrupt request of level 2. The interrupts labelled “low” in the interrupt handler are routed to this interrupt level. This bit has to be cleared when the interrupt is served.
3	IN1	interrupt request of level 1. The interrupts labelled “mid” in the interrupt handler are routed to this interrupt level. This bit has to be cleared when the interrupt is served.
2	IN0	interrupt request of level 0. The interrupts labelled “hig” in the interrupt handler are routed to this interrupt level. This bit has to be cleared when the interrupt is served.
1	EV1	event request of level 1. The events labelled “low” in the event handler are routed to this event level. This bit has to be cleared when the event is served.
0	EV0	event request of level 1. The events labelled “hig” in the event handler are routed to this event level. This bit has to be cleared when the event is served.

Table 3-2. Status register description

The CPU also has a number of flags that can be used for conditional jumps. These flags are defined in Table 3-3.

symbol	name	function
Z	zero	Z=1 when the accumulator a content is zero
C	carry	This flag is used in shift or arithmetic operations. For a shift operation, it has the value of the bit that was shifted out (LSB for shift right, MSB for shift left). For an arithmetic operation with unsigned numbers: it is 1 at occurrence of an overflow during an addition (or equivalent). it is 0 at occurrence of an underflow during a subtraction (or equivalent).
V	overflow	This flag is used in shift or arithmetic operations. For arithmetic or shift operations with signed numbers, it is 1 if an overflow or underflow occurs.

Table 3-3. Flag description

3.3 CPU instruction short reference

Table 3-4 shows a short description of the different instructions available on the Coolrisc816. The notation cc in the conditional jump instruction refers to the condition description as given in Table 3-6. The notation reg, reg1, reg2, reg3 refers to one of the CPU internal registers of Table 3-1. The notation eaddr and DM(eaddr) refer to one of the extended address modes as defined in Table 3-5. The notation DM(xxx) refers to the data memory location with address xxx.

Instruction	Modification	Operation
Jump addr[15:0]	-,-,-, -	PC := addr[15:0]
Jump ip	-,-,-, -	PC := ip
Jcc addr[15:0]	-,-,-, -	if cc is true then PC := addr[15:0]
Jcc ip	-,-,-, -	if cc is true then PC := ip
Call addr[15:0]	-,-,-, -	ST_{n+1} := ST_n (n>1); ST_1 := PC+1; PC := addr[15:0]
Call ip	-,-,-, -	ST_{n+1} := ST_n (n>1); ST_1 := PC+1; PC := ip
Calls addr[15:0]	-,-,-, -	ip := PC+1; PC := addr[15:0]
Calls ip	-,-,-, -	ip := PC+1; PC := ip
Ret	-,-,-, -	PC := ST_1; ST_n := ST_{n+1} (n>1)
Rets	-,-,-, -	PC := ip
Reti	-,-,-, -	PC := ST_1; ST_n := ST_{n+1} (n>1); GIE :=1
Push	-,-,-, -	PC := PC+1; ST_{n+1} := ST_n (n>1); ST_1 := ip
Pop	-,-,-, -	PC := PC+1; ip := ST_1; ST_n := ST_{n+1} (n>1)
Move reg,#data[7:0]	-,-, Z, a	a := data[7:0]; reg := data[7:0]
Move reg1, reg2	-,-, Z, a	a := reg2; reg1 := reg2
Move reg, eaddr	-,-, Z, a	a := DM(eaddr); reg := DM(eaddr)
Move eaddr, reg	-,-,-, -	DM(eaddr) := reg
Move addr[7:0],#data[7:0]	-,-,-, -	DM(addr[7:0]) := data[7:0]
Cmvd reg1, reg2	-,-, Z, a	a := reg2; if C=0 then reg1 := a;
Cmvd reg, eaddr	-,-, Z, a	a := DM(eaddr); if C=0 then reg := a
Cmvs reg1, reg2	-,-, Z, a	a := reg2; if C=1 then reg1 := a;
Cmvs reg, eaddr	-,-, Z, a	a := DM(eaddr); if C=1 then reg := a
Shl reg1, reg2	C, V, Z, a	a := reg2<<1; a[0] := 0; C := reg2[7]; reg1 := a
Shl reg	C, V, Z, a	a := reg<<1; a[0] := 0; C := reg[7]; reg := a
Shl reg, eaddr	C, V, Z, a	a := DM(eaddr)<<1; a[0] :=0; C := DM(eaddr)[7]; reg := a
Shlc reg1, reg2	C, V, Z, a	a := reg2<<1; a[0] := C; C := reg2[7]; reg1 := a
Shlc reg	C, V, Z, a	a := reg<<1; a[0] := C; C := reg[7]; reg := a
Shlc reg, eaddr	C, V, Z, a	a := DM(eaddr)<<1; a[0] := C; C := DM(eaddr)[7]; reg := a
Shr reg1, reg2	C, V, Z, a	a := reg2>>1; a[7] := 0; C := reg2[0]; reg1 :=a
Shr reg	C, V, Z, a	a := reg>>1; a[7] := 0; C := reg[0]; reg := a
Shr reg, eaddr	C, V, Z, a	a := DM(eaddr)>>1; a[7] := 0; C := DM(eaddr)[0]; reg := a
Shrc reg1, reg2	C, V, Z, a	a := reg2>>1; a[7] := C; C := reg2[0]; reg1 := a
Shrc reg	C, V, Z, a	a := reg>>1; a[7] := C; C := reg[0]; reg := a
Shrc reg, eaddr	C, V, Z, a	a := DM(eaddr)>>1; a[7] := C; C := DM(eaddr)[0]; reg := a
Shra reg1, reg2	C, V, Z, a	a := reg2>>1; a[7] := reg2[7]; C := reg2[0]; reg1 := a
Shra reg	C, V, Z, a	a := reg>>1; a[7] := reg[7]; C := reg[0]; reg := a
Shra reg, eaddr	C, V, Z, a	a := DM(eaddr)>>1; a[7] := DM(eaddr)[7]; C := DM(eaddr)[0]; reg := a
Cpl1 reg1, reg2	-,-, Z, a	a := NOT(reg2); reg1 := a
Cpl1 reg	-,-, Z, a	a := NOT(reg); reg := a
Cpl1 reg, eaddr	-,-, Z, a	a := NOT(DM(eaddr)); reg := a
Cpl2 reg1, reg2	C, V, Z, a	a := NOT(reg2)+1; if a=0 then C:=1 else C := 0; reg1 := a
Cpl2 reg	C, V, Z, a	a := NOT(reg)+1; if a=0 then C:=1 else C := 0; reg := a
Cpl2 reg, eaddr	C, V, Z, a	a := NOT(DM(eaddr))+1; if a=0 then C:=1 else C := 0; reg := a
Cpl2c reg1, reg2	C, V, Z, a	a := NOT(reg2)+C; if a=0 and C=1 then C:=1 else C := 0; reg1 := a
Cpl2c reg	C, V, Z, a	a := NOT(reg)+C; if a=0 and C=1 then C:=1 else C := 0; reg := a
Cpl2c reg, eaddr	C, V, Z, a	a := NOT(DM(eaddr))+C; if a=0 and C=1 then C:=1 else C := 0; reg := a
Inc reg1, reg2	C, V, Z, a	a := reg2+1; if a=0 then C := 1 else C := 0; reg1 := a
Inc reg	C, V, Z, a	a := reg+1; if a=0 then C := 1 else C := 0; reg := a
Inc reg, eaddr	C, V, Z, a	a := DM(eaadr)+1; if a=0 then C := 1 else C := 0; reg := a
Incc reg1, reg2	C, V, Z, a	a := reg2+C; if a=0 and C=1 then C := 1 else C := 0; reg1 := a
Incc reg	C, V, Z, a	a := reg+C; if a=0 and C=1 then C := 1 else C := 0; reg := a
Incc reg, eaddr	C, V, Z, a	a := DM(eaadr)+C; if a=0 and C=1 then C := 1 else C := 0; reg := a
Dec reg1, reg2	C, V, Z, a	a := reg2-1; if a=hFF then C := 0 else C := 1; reg1 := a

Dec reg	C, V, Z, a	a := reg-1; if a=hFF then C := 0 else C := 1; reg := a
Dec reg, eaddr	C, V, Z, a	a := DM(eaddr)-1; if a=hFF then C := 0 else C := 1; reg := a
Decc reg1, reg2	C, V, Z, a	a := reg2-(1-C); if a=hFF and C=0 then C := 0 else C := 1; reg1 := a
Decc reg	C, V, Z, a	a := reg-(1-C); if a=hFF and C=0 then C := 0 else C := 1; reg := a
Decc reg, eaddr	C, V, Z, a	a := DM(eaddr)-(1-C); if a=hFF and C=0 then C := 0 else C := 1; reg := a
And reg,#data[7:0]	-,-, Z, a	a := reg and data[7:0]; reg := a
And reg1, reg2, reg3	-,-, Z, a	a := reg2 and reg3; reg1 := a
And reg1, reg2	-,-, Z, a	a := reg1 and reg2; reg1 := a
And reg, eaddr	-,-, Z, a	a := reg and DM(eaddr); reg := a
Or reg,#data[7:0]	-,-, Z, a	a := reg or data[7:0]; reg := a
Or reg1, reg2, reg3	-,-, Z, a	a := reg2 or reg3; reg1 := a
Or reg1, reg2	-,-, Z, a	a := reg1 or reg2; reg1 := a
Or reg, eaddr	-,-, Z, a	a := reg or DM(eaddr); reg := a
Xor reg,#data[7:0]	-,-, Z, a	a := reg xor data[7:0]; reg := a
Xor reg1, reg2, reg3	-,-, Z, a	a := reg2 xor reg3; reg1 := a
Xor reg1, reg2	-,-, Z, a	a := reg1 xor reg2; reg1 := a
Xor reg, eaddr	-,-, Z, a	a := reg or DM(eaddr); reg := a
Add reg,#data[7:0]	C, V, Z, a	a := reg+data[7:0]; if overflow then C:=1 else C := 0; reg := a
Add reg1, reg2, reg3	C, V, Z, a	a := reg2+reg3; if overflow then C:=1 else C := 0; reg1 := a
Add reg1, reg2	C, V, Z, a	a := reg1+reg2; if overflow then C:=1 else C := 0; reg1 := a
Add reg, eaddr	C, V, Z, a	a := reg+DM(eaddr); if overflow then C:=1 else C := 0; reg := a
Addc reg,#data[7:0]	C, V, Z, a	a := reg+data[7:0]+C; if overflow then C:=1 else C := 0; reg := a
Addc reg1, reg2, reg3	C, V, Z, a	a := reg2+reg3+C; if overflow then C:=1 else C := 0; reg1 := a
Addc reg1, reg2	C, V, Z, a	a := reg1+reg2+C; if overflow then C:=1 else C := 0; reg1 := a
Addc reg, eaddr	C, V, Z, a	a := reg+DM(eaddr)+C; if overflow then C:=1 else C := 0; reg := a
Subd reg,#data[7:0]	C, V, Z, a	a := data[7:0]-reg; if underflow then C := 0 else C := 1; reg := a
Subd reg1, reg2, reg3	C, V, Z, a	a := reg2-reg3; if underflow then C := 0 else C := 1; reg1 := a
Subd reg1, reg2	C, V, Z, a	a := reg2-reg1; if underflow then C := 0 else C := 1; reg1 := a
Subd reg, eaddr	C, V, Z, a	a := DM(eaddr)-reg; if underflow then C := 0 else C := 1; reg := a
Subdc reg,#data[7:0]	C, V, Z, a	a := data[7:0]-reg-(1-C); if underflow then C := 0 else C := 1; reg := a
Subdc reg1, reg2, reg3	C, V, Z, a	a := reg2-reg3-(1-C); if underflow then C := 0 else C := 1; reg1 := a
Subdc reg1, reg2	C, V, Z, a	a := reg2-reg1-(1-C); if underflow then C := 0 else C := 1; reg1 := a
Subdc reg, eaddr	C, V, Z, a	a := DM(eaddr)-reg-(1-C); if underflow then C := 0 else C := 1; reg := a
Subs reg,#data[7:0]	C, V, Z, a	a := reg-data[7:0]; if underflow then C := 0 else C := 1; reg := a
Subs reg1, reg2, reg3	C, V, Z, a	a := reg3-reg2; if underflow then C := 0 else C := 1; reg1 := a
Subs reg1, reg2	C, V, Z, a	a := reg1-reg2; if underflow then C := 0 else C := 1; reg1 := a
Subs reg, eaddr	C, V, Z, a	a := reg-DM(eaddr); if underflow then C := 0 else C := 1; reg := a
Subsc reg,#data[7:0]	C, V, Z, a	a := reg-data[7:0]-(1-C); if underflow then C := 0 else C := 1; reg := a
Subsc reg1, reg2, reg3	C, V, Z, a	a := reg3-reg2-(1-C); if underflow then C := 0 else C := 1; reg1 := a
Subsc reg1, reg2	C, V, Z, a	a := reg1-reg2-(1-C); if underflow then C := 0 else C := 1; reg1 := a
Subsc reg, eaddr	C, V, Z, a	a := reg-DM(eaddr)-(1-C); if underflow then C := 0 else C := 1; reg := a
Mul reg,#data[7:0]	u, u, u, a	a := (data[7:0]*reg)[7:0]; reg := (data[7:0]*reg)[15:8]
Mul reg1, reg2, reg3	u, u, u, a	a := (reg2*reg3)[7:0]; reg1 := (reg2*reg3)[15:8]
Mul reg1, reg2	u, u, u, a	a := (reg2*reg1)[7:0]; reg1 := (reg2*reg1)[15:8]
Mul reg, eaddr	u, u, u, a	a := (DM(eaddr)*reg)[7:0]; reg := (DM(eaddr)*reg)[15:8]
Mula reg,#data[7:0]	u, u, u, a	a := (data[7:0]*reg)[7:0]; reg := (data[7:0]*reg)[15:8]
Mula reg1, reg2, reg3	u, u, u, a	a := (reg2*reg3)[7:0]; reg1 := (reg2*reg3)[15:8]
Mula reg1, reg2	u, u, u, a	a := (reg2*reg1)[7:0]; reg1 := (reg2*reg1)[15:8]
Mula reg, eaddr	u, u, u, a	a := (DM(eaddr)*reg)[7:0]; reg := (DM(eaddr)*reg)[15:8]
Mshl reg,#shift[2:0]	u, u, u, a	a := (reg*2^{shift})[7:0]; reg := (reg*2^{shift})[15:8]
Mshr reg,#shift[2:0]	u, u, u, a	a := (reg*$2^{(8-shift)}$)[7:0]; reg := (reg*$2^{(8-shift)}$)[15:8]
Mshra reg,#shift[2:0]	u, u, u, a*	a := (reg*$2^{(8-shift)}$)[7:0]; reg := (reg*$2^{(8-shift)}$)[15:8]
Cmp reg,#data[7:0]	C, V, Z, a	a := data[7:0]-reg; if underflow then C :=0 else C:=1; V := C and (not Z)
Cmp reg1, reg2	C, V, Z, a	a := reg2-reg1; if underflow then C :=0 else C:=1; V := C and (not Z)
Cmp reg, eaddr	C, V, Z, a	a := DM(eaddr)-reg; if underflow then C :=0 else C:=1; V := C and (not Z)
Cmpa reg,#data[7:0]	C, V, Z, a	a := data[7:0]-reg; if underflow then C :=0 else C:=1; V := C and (not Z)
Cmpa reg1, reg2	C, V, Z, a	a := reg2-reg1; if underflow then C :=0 else C:=1; V := C and (not Z)
Cmpa reg, eaddr	C, V, Z, a	a := DM(eaddr)-reg; if underflow then C :=0 else C:=1; V := C and (not Z)
Tstb reg,#bit[2:0]	-, -, Z, a	a[bit] := reg[bit]; other bits in a are 0
Setb reg,#bit[2:0]	-, -, Z, a	reg[bit] := 1; other bits unchanged; a := reg
Clrb reg,#bit[2:0]	-, -, Z, a	reg[bit] := 0; other bits unchanged; a := reg
Invb reg,#bit[2:0]	-, -, Z, a	reg[bit] := not reg[bit]; other bits unchanged; a := reg

Sflag	-,-,-, a	a[7] := C; a[6] := C xor V; a[5] := ST full; a[4] := ST empty
Rflag reg	C, V, Z, a	a := reg << 1; ; a[0] := 0; C := reg[7]
Rflag eaddr	C, V, Z, a	a := DM(eaddr)<<1; a[0] :=0; C := DM(eaddr)[7]
Freq divn	-,-,-, -	reduces the CPU frequency (divn=nodiv, div2, div4, div8, div16)
Halt	-,-,-, -	halts the CPU
Nop	-,-,-, -	no operation

- = unchanged, u = undefined, *MSHR reg,# 1 doesn't shift by 1

Table 3-4. Instruction short reference

The Coolrisc816 has 8 different addressing modes. These modes are described in Table 3-5. In this table, the notation ix refers to one of the data memory index registers i0, i1, i2 or i3. Using eaddr in an instruction of Table 3-4 will access the data memory at the address DM(eaddr) and will simultaneously execute the index operation.

extended address eaddr	accessed data memory location DM(eaddr)	index operation	
addr[7:0]	DM(h00&addr[7:0])	-	direct addressing
(ix)	DM(ix)	-	indexed addressing
(ix, offset[7:0])	DM(ix+offset)	-	indexed addressing with immediate offset
(ix,r3)	DM(ix+r3)	-	indexed addressing with register offset
(ix)+	DM(ix)	ix := ix+1	indexed addressing with index post-increment
(ix,offset[7:0])+	DM(ix+offset)	ix := ix+offset	indexed addressing with index post-increment by the offset
-(ix)	DM(ix-1)	ix := ix-1	indexed addressing with index pre-decrement
-(ix,offset[7:0])	DM(ix-offset)	ix := ix -offset	indexed addressing with index pre-decrement by the offset

Table 3-5. Extended address mode description

Eleven different jump conditions are implemented as shown in Table 3-6. The contents of the column CC in this table should replace the CC notation in the instruction description of Table 3-4.

CC	condition
CS	C=1
CC	C=0
ZS	Z=1
ZC	Z=0
VS	V=1
VC	V=0
EV	(EV1 or EV0)=1
After CMP op1,op2	
EQ	op1=op2
NE	op1≠op2
GT	op1>op2
GE	op1≥op2
LT	op1<op2
LE	op1≤op2

Table 3-6. Jump condition description

4 Memory Mapping

4.1 Memory organisation

The XE8802 CPU is built with Harvard architecture. Harvard architecture uses separate instruction and data memories. The instruction bus and data bus are also separated. The advantage of such a structure is that the CPU can get a new instruction and read/write data simultaneously. The circuit configuration is shown in Figure 4-1. The CPU has its 16 internal registers. The instruction memory has a capacity of 8192 22-bit instructions if the pin NFASTREAD=1. The instruction memory has a capacity of 4096 22-bit instructions if the pin NFASTREAD=0. The data memory space has 8 low power registers, the peripheral register space, 1024 bytes of RAM and the LCD control register space.

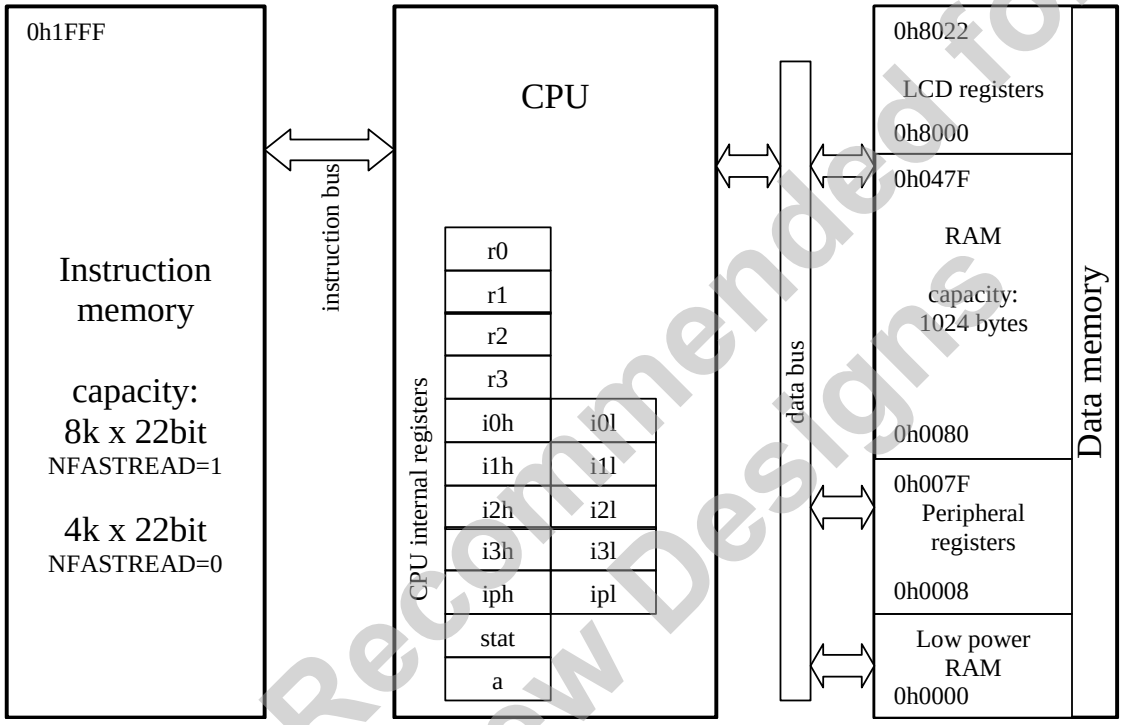


Figure 4-1. Memory mapping

The CPU internal registers are described in the CPU chapter. A short reference of the low power registers and peripheral registers is given in 4.3.

4.2 Instruction memory and NFASTREAD

As indicated in chapter 4.1, the instruction memory can be organized in two ways: 4096 instructions with a speed of 4 to 6 MIPS or 8192 instructions with a speed of 2 to 3 MIPS (see XE8802 performance chapter for more details).

Important Note:
When programming the device, the correct NFASTREAD option has to be selected in the programmer. Failure to do so may result in non-functional software. The default option in the programme is NFASTREAD=1, i.e. the 8k instruction slower operation mode.

Important Note:
In the actual version of the Prostart, the NFASTREAD pin is hard wired to VBAT. Nevertheless, the Prostart can be used to program the device for both modes by (un)checking the NFASTREAD selection box in the software loader. If the user wants to run the software programmed in the NFASTREAD=0 mode, he will need to build his own application board which connects NFASTREAD=VSS.

4.3 Quick reference data memory register map

The data register map is given in the tables below. A more detailed description of the different registers is given in the detailed description of the different peripherals.

The tables give the following information:

1. The register name and register address
2. The different bits in the register
3. The access mode of the different bits (see Table 4-4-1 for code description)
4. The reset source and reset value of the different bits

The reset source coding is given in Table 4-4-2. To get a full description of the reset sources, please refer to the reset block chapter.

code	access mode
r	bit can be read
w	bit can be written
r0	bit always reads 0
r1	bit always reads 1
c	bit is cleared by writing any value
c1	bit is cleared by writing a 1
ca	bit is cleared after reading
s	special function, verify the detailed description in the respective peripherals

Table 4-4-1. Access mode codes used in the register definitions

code	reset source
glob	nresetglobal
cold	nresetcold
pconf	nresetpconf
sleep	nresetsleep

Table 4-4-2. Reset source coding used in the register definitions

4.3.1 Low power data registers (h0000-h0007)

Name / Address	7	6	5	4	3	2	1	0
Reg00 h0000	Reg00[7:0] rw, 00000000, glob							
Reg01 h0001	Reg01[7:0] rw,00000000,glob							
Reg02 h0002	Reg02[7:0] rw,00000000,glob							
Reg03 h0003	Reg03[7:0] rw,00000000,glob							
Reg04 h0004	Reg04[7:0] rw,00000000,glob							
Reg05 h0005	Reg05[7:0] rw,00000000,glob							
Reg06 h0006	Reg06[7:0] rw,00000000,glob							
Reg07 h0007	Reg07[/:0] rw,0000000,glob							

Table 4-4-3. Low power data registers

4.3.2 System, clock configuration and reset configuration (h0010-h001F)

Name / Address	7	6	5	4	3	2	1	0
RegSysCtrl h0010	SleepEn rw,0,cold	EnResetPConf rw,0,cold	EnBusError rw,0,cold	EnResetWD rw,0,cold	r0	r0	r0	r0
RegSysReset h0011	Sleep rw,0,glob	SleepFlag rc,0,cold	ResetBusError rc, 0, cold	ResetWD rc, 0, cold	ResetfromportA rc, 0, cold	r0	r0	r0
RegSysClock h0012	CpuSel rw,0,sleep	r0	EnExtClock rw,0,cold	BiasRC rw,1,cold	ColdXtal r,1,sleep	r0	EnableXtal rw,0,sleep	EnableRC rw,1,sleep
RegSysMisc h0013	r0	r0	r0	r0	r0	r0	Output16k rw,0,sleep	OutputCpuCk rw,0,sleep
RegSysWd h0014	r0	r0	r0	r0	WatchDog[3:0] s,0000,glob			
RegSysPre0 h0015	r0	r0	r0	r0	r0	r0	r0	ClearLowPrescal c1r0,0,-
RegSysRcTrim1 h001B	r0	r0	r0	RcFreqRange rw,0,cold	RcFreqCoarse[3:0] rw,0001,cold			
RegSysRcTrim2 h001C	r0	r0	RcFreqFine[5:0] rw,00000,cold					

Table 4-4-4. Reset block and clock block registers

4.3.3 Port A (h0020-h0027)

Name / Address	7	6	5	4	3	2	1	0
RegPAIn h0020	PAIn[7:0] r							
RegPADebounce h0021	PADebounce[7:0] rw,00000000,pconf							
RegPAEdge h0022	PAEdge[7:0] rw,00000000,glob							
RegPAPullup h0023	PAPullup[7:0] rw,11111111,pconf							
RegPARes0 h0024	PARes0[7:0] rw, 00000000, glob							
RegPARes1 h0025	PARes1[7:0] rw,00000000,glob							
RegPACtrl h0026	r0	r0	r0	r0	r0	r0	r0	DebFast rw,0,pconf
RegPASnaptorail h0027	PASnaptorail[7:0] rw,00000000,pconf							

Table 4-4-5. Port A registers

4.3.4 Port B (h0028-h002F)

Name / Address	7	6	5	4	3	2	1	0
RegPBOut h0028	PBOut[7:0] rw,00000000,pconf							
RegPBIn h0029	PBIn[7:0] r							
RegPBDir h002A	PBDir[7:0] rw,00000000,pconf							
RegPBOpen h002B	PBOpen[7:0] rw,00000000,pconf							
RegPBPullup h002C	PBPullup[7:0] rw,11111111,pconf							
RegPBAna h002D	PBAna[7:0] rw,00000000,pconf							

Table 4-4-6. Port B registers

4.3.5 Port D1 (h0030-h0033)

Name / Address	7	6	5	4	3	2	1	0
RegPD1Out h0030	PD1Out[7:0] rw,00000000,pconf							
RegPD1In h0031	PD1In[7:0] r							
RegPD1Dir h0032	PD1Dir[7:0] rw,00000000,pconf							
RegPD1Pullup h0033	PD1SnapToRail[3:0] rw,0000,pconf				PD1Pullup[3:0] rw,1111,pconf			

Table 4-4-7. Port D1 registers

4.3.6 Port D2 (h0034-h0037)

Name / Address	7	6	5	4	3	2	1	0
RegPD2Out h0034	PD2Out[7:0] rw,00000000,pconf							
RegPD2In h0035	PD2In[7:0] r							
RegPD2Dir h0036	PD2Dir[7:0] rw,00000000,pconf							
RegPD2Pullup h0037	PD2SnapToRail[3:0] rw,0000,pconf				PD2Pullup[3:0] rw,1111,pconf			

Table 4-4-8. Port D2 registers

4.3.7 Flash programming (h0038-003B)

These four registers are used during flash programming only. Refer to the flash programming algorithm documentation for more details.

4.3.8 Event handler (h003C-h003F)

Name / Address	7	6	5	4	3	2	1	0
RegEvn h003C	CntIrqA rc1,0,glob	CntIrqC rc1,0,glob	128Hz rc1,0,glob	PAEvn[1] rc1,0,glob	CntIrqB rc1,0,glob	CntIrqD rc1,0,glob	1Hz rc1,0,glob	PAEvn[0] rc1,0,glob
RegEvnEn h003D	EvnEn[7:0] rw,00000000,glob							
RegEvnPriority h003E	EvnPriority[7:0] r,11111111,glob							
RegEvnEvn h003F	r0	r0	r0	r0	r0	r0	EvnHigh r,0,glob	EvnLow r,0,glob

Table 4-4-9. Event handler registers

The origin of the different events is summarised in the table below.

Event	Event source
CntIrqA	Counter/Timer A (counter block)
CntIrqB	Counter/Timer B (counter block)
CntIrqC	Counter/Timer C (counter block)
CntIrqD	Counter/Timer D (counter block)
128Hz	Low prescaler (clock block)
1Hz	Low prescaler (clock block)
PAEvn[1:0]	Port A

Table 4-4-10. Event source description

4.3.9 Interrupt handler (h0040-h0047)

Name Address	7	6	5	4	3	2	1	0
RegIrqHig h0040	IrqAC rc1,0,glob	128Hz rc1,0,glob	IrqSPI rc1,0,glob	CntIrqA rc1,0,glob	CntIrqC rc1,0,glob	CmpdIrq rc1,0,glob	UartIrqTx rc1,0,glob	UartIrqRx rc1,0,glob
RegIrqMid h0041	UsrtCond2 rc1,0,glob	UrstCond1 rc1,0,glob	PAIrq[5] rc1,0,glob	PAIrq[4] rc1,0,glob	1Hz rc1,0,glob	VldIrq rc1,0,glob	PAIrq[1] rc1,0,glob	PAIrq[0] rc1,0,glob
RegIrqLow h0042	PAIrq[7] rc1,0,glob	PAIrq[6] rc1,0,glob	CntIrqB rc1,0,glob	CntIrqD rc1,0,glob	PAIrq[3] rc1,0,glob	PAIrq[2] rc1,0,glob	r0	r0
RegIrqEnHig h0043	IrqEnHig[7:0] rw,0000000,glob							
RegIrqEnMid h0044	IrqEnMid[7:0] rw,0000000,glob							
RegIrqEnLow h0045	IrqEnLow[7:0] rw,0000000,glob							
RegIrqPriority h0046	IrqPriority[7:0] r,11111111,glob							
RegIrqIrq h0047	r0	r0	r0	r0	r0	IrqHig r,0,glob	IrqMid r,0,glob	IrqLow r,0,glob

Table 4-4-11. Interrupt handler registers

The origin of the different interrupts is summarised in the table below.

Event	Event source
CmpdIrq	Low power comparators
CntIrqA	Counter/Timer A (counter block)
CntIrqB	Counter/Timer B (counter block)
CntIrqC	Counter/Timer C (counter block)
CntIrqD	Counter/Timer D (counter block)
128Hz	Low prescaler (clock block)
1Hz	Low prescaler (clock block)
PAIrq[7:0]	Port A
UartIrqRx	UART reception
UartIrqTx	UART transmission
UrstCond1	USRT condition 1
UsrtCond2	USRT condition 2
VldIrq	Voltage level detector
IrqAC	Acquisition chain end of conversion interrupt
IrqSPI	SPI end of reception/transmission interrupt

Table 4-4-12. Interrupt source description

4.3.10 USRT (h0048-h004F)

Name Address	7	6	5	4	3	2	1	0
RegUsrtS1 h0048	r0	r0	r0	r0	r0	r0	r0	UsrtS1 s,1,glob
RegUsrtS0 h0049	r0	r0	r0	r0	r0	r0	r0	UsrtS0 s,1,glob
RegUsrtCond1 h004A	r0	r0	r0	r0	r0	r0	r0	UsrtCond1 rc,0,glob
RegUsrtCond2 h004B	r0	r0	r0	r0	r0	r0	r0	UsrtCond2 rc,0,glob
RegUsrtCtrl h004C	r0	r0	r0	r0	UsrtWaitS0 r,0,glob	UsrtEnWaitCond1 rw,0,glob	UsrtEnWaitS0 rw,0,glob	UsrtEnable rw,0,glob
RegUsrtBufferS1 h004D	r0	r0	r0	r0	r0	r0	r0	UsrtBufferS1 r,0,glob
RegUsrtEdgeS0 h004E	r0	r0	r0	r0	r0	r0	r0	UsrtEdgeS0 r,0,glob

Table 4-4-13. USRT register description

4.3.11 UART (h0050-h0057)

Name / Address	7	6	5	4	3	2	1	0
RegUartCtrl h0050	UartEcho rw,0,glob	UartEnRx rw,0,glob	UartEnTx rw,0,glob	UartXRx rw,0,glob	UartXTx rw,0,glob	UartBR[2:0] rw,101,glob		
RegUartCmd h0051	SelXtal rw,0,glob	r0	UartRcSel[2:0] rw,000,glob			UartPM rw,0,glob	UartPE rw,0,glob	UartWL rw,1,glob
RegUartTx h0052	UartTx[7:0] rw,0000000,glob							
RegUartTxSta h0053	r0	r0	r0	r0	r0	r0	UartTxBusy r,0,glob	UartTxFull r,0,glob
RegUartRx h0054	UartRx[7:0] r,00000000,glob							
RegUartRxSta h0055	r0	r0	UartRxSErr r,0,glob	UartRxPErr r,0,glob	UartRxFErr r,0,glob	UartRxOerr rc,0,glob	UartRxBusy r,0,glob	UartRxFull r,0,glob

Table 4-14. UART register description

4.3.12 Counter/Timer/PWM registers (h0058-h005F)

Name / Address	7	6	5	4	3	2	1	0
RegCntA h0058	CounterA[7:0] s,00000000,glob							
RegCntB h0059	CounterB[7:0] s,00000000,glob							
RegCntC h005A	CounterC[7:0] s,00000000,glob							
RegCntD h005B	CounterD[7:0] s,00000000,glob							
RegCntCtrlCk h005C	CntDCkSel[1:0] rw,00,glob		CntCCkSel[1:0] rw,00,glob		CntBCkSel[1:0] rw,00,glob		CntACkSel[1:0] rw,00,glob	
RegCntConfig1 h005D	CntDDownUp rw,0,glob	CntCDownUp rw,0,glob	CntBDownUp rw,0,glob	CntADownUp rw,0,glob	CascadeCD rw,0,glob	CascadeAB rw,0,glob	CntPWM1 rw,0,glob	CntPWM0 rw,0,glob
RegCntConfig2 h005E	CapSel[1:0] rw,00,glob		CapFunc[1:0] rw,00,glob		Pwm1Size[1:0] rw,00,glob		Pwm0Size[1:0] rw,00,glob	
RegCntOn h005F	CntDExtDiv rw,0,glob	CntCExtDiv rw,0,glob	CntBExtDiv rw,0,glob	CntAExtDiv rw,0,glob	CntDEnable rw,0,glob	CntCEnable rw,0,glob	CntBEnable rw,0,glob	CntAEnable rw,0,glob

Table 4-15. Counter/timer/PWM register description.

4.3.13 Acquisition chain registers (h0060-h0067)

Name / Address	7	6	5	4	3	2	1	0
RegAcOutLsb h0060	OUT[7:0] r,0,glob							
RegAcOutMsb h0061	OUT[15:8] r,0,glob							
RegAcCfg0 h0062	START w r0,0,glob	SET_NELCONV[1:0] rw,01,glob		SET_OSR[2:0] rw,010,glob			CONT rw,0,glob	r0
RegAcCfg1 h0063	IB_AMP_ADC[1:0] rw,11,glob		IB_AMP_PGA[1:0] rw,11,glob		ENABLE[3:0] rw,0000,glob			
RegAcCfg2 h0064	FIN rw,00,glob		PGA2_GAIN[1:0] rw,00,glob		PGA2_OFFSET[3:0] rw,0000,glob			
RegAcCfg3 h0065	PGA1_GAIN Rw,0,glob	PGA3_GAIN[6:0] rw,0000000,glob						
RegAcCfg4 h0066	r0	PGA3_OFFSET rw,0000000,glob						
RegAcCfg5 h0067	BUSY r,0,glob	DEF w r0	AMUX[4:0] rw,00000,glob					VMUX rw,0,glob

Table 4-16. Acquisition chain register description.

4.3.14 SPI registers (h0068-h006F)

Name / Address	7	6	5	4	3	2	1	0
RegSpiControl h0068	ClearCounter c1 r0	NotSlaveSelect rw,1,glob	SpiMaster rw,1,glob	SpiEnable rw,0,glob	ClockPhase rw,1,glob	ClockPolarity rw,0,glob	BaudRate[1:0] rw,00,glob	
RegSpiStatus h0069	r0	r0	r0	r0	r0	SpiOverflow r c1,0,glob	SpiRxFull r,0,glob	SpiTxEmpty r w1,1,glob
RegSpiDataOut h006A	SpiDataOut[7:0] rw,00000000,glob							
RegSpiDataIn h006B	SpiDataIn[7:0] r,00000000,glob							
RegSpiPullup h006C	r0	r0	r0	r0	SpiPullup[3:0] rw,1111,pconf			
RegSpiDir h006D	r0	r0	r0	r0	SpiDir[3:0] rw,0000,pconf			

Table 4-17. SPI register description.

4.3.15 LCD voltage generator registers (h0070)

Name / Address	7	6	5	4	3	2	1	0
RegVgenCfg0 h0070	r0	r0	VgenClkSel[1:0] rw,10,glob		VgenOff rw,1,glob	VgenMode rw,0,glob	VgenStdb rw,0,glob	VgenRefEn rw,0,glob

Table 4-18. LCD voltage generator register.

4.3.16 Comparator registers (h0072-h0073)

Name / Address	7	6	5	4	3	2	1	0
RegCmpdStat h0072	CmpdStat[3:0] rca,0000,glob				CmpdOut[3:0] r,0000,glob			
RegCmpdCtrl h0073	IrqOnRising[2:0] rw,000,glob			EnIrqCh[3:0] rw,0000,glob				Enable rw,0,glob

Table 4-19. Low power comparator registers

4.3.17 Voltage multiplier (h007C)

Name / Address	7	6	5	4	3	2	1	0
RegVmultCfg0 h007C	r0	r0	r0	r0	r0	Enable rw,0,glob	Fin[1:0] rw,00,glob	

Table 4-20. VMULT register.

4.3.18 Voltage Level Detector registers (h007E-h007F)

Name / Address	7	6	5	4	3	2	1	0
RegVldCtrl h007E	r0	r0	r0	r0	VldRange rw,0,glob	VldTune[2:0] rw,000,glob		
RegVldStat h007F	r0	r0	r0	r0	r0	VldResult r,0,glob	VldValid r,0,glob	VldEn rw,0,glob

Table 4-21. Voltage level detector register description

4.3.19 RAM (h0080-h047F)

The 1024 RAM bytes can be accessed for read and write operations. The RAM has no reset function. Variables stored in the RAM should be initialised before use since they can have any value at circuit start up.

4.3.20 LCD driver (h8000-8022)

Name / Address	7	6	5	4	3	2	1	0
RegLcdData0 h8000	LcdData0[7:0] rw,00000000,pconf							
RegLcdData1 h8001	LcdData1[7:0] rw,00000000,pconf							
RegLcdData2 h8002	LcdData2[7:0] rw,00000000,pconf							
RegLcdData3 h8003	LcdData3[7:0] rw,00000000,pconf							
RegLcdData4 h8004	LcdData4[7:0] rw,00000000,pconf							
RegLcdData5 h8005	LcdData5[7:0] rw,00000000,pconf							
RegLcdData6 h8006	LcdData6[7:0] rw,00000000,pconf							
RegLcdData7 h8007	LcdData7[7:0] rw,00000000,pconf							
RegLcdData8 h8008	LcdData8[7:0] rw,00000000,pconf							
RegLcdData9 h8009	LcdData9[7:0] rw,00000000,pconf							
RegLcdData10 h800A	LcdData10[7:0] rw,00000000,pconf							
RegLcdData11 h800B	LcdData11[7:0] rw,00000000,pconf							
RegLcdData12 h800C	LcdData12[7:0] rw,00000000,pconf							
RegLcdData13 h800D	LcdData13[7:0] rw,00000000,pconf							
RegLcdData14 h800E	LcdData14[7:0] rw,00000000,pconf							
RegLcdData15 h800F	LcdData15[7:0] rw,00000000,pconf							
RegPLcdOut0 h8010	PLcdOut0[7:0] rw,00000000,pconf							
RegPLcdOut1 h8011	PLcdOut1[7:0] rw,00000000,pconf							
RegPLcdOut2 h8012	PLcdOut2[7:0] rw,00000000,pconf							
RegPLcdOut3 h8013	PLcdOut3[7:0] rw,00000000,pconf							
RegPLcdDir0 h8014	PLcdDir0[7:0] rw,00000000,pconf							
RegPLcdDir1 h8015	PLcdDir1[7:0] rw,00000000,pconf							
RegPLcdDir2 h8016	PLcdDir2[7:0] rw,00000000,pconf							
RegPLcdDir3 h8017	PLcdDir3[7:0] rw,00000000,pconf							
RegPLcdPullup0 h8018	PLcdPullup0[7:0] rw,00000000,pconf							
RegPLcdPullup1 h8019	PLcdPullup1[7:0] rw,00000000,pconf							
RegPLcdPullup2 h801A	PLcdPullup2[7:0] rw,00000000,pconf							
RegPLcdPullup3 h801B	PLcdPullup3[7:0] rw,00000000,pconf							
RegPLcdIn0 h801C	PLcdIn0[7:0] r							
RegPLcdIn1 h801D	PLcdIn1[7:0] r							
RegPLcdIn2 h801E	PLcdIn2[7:0] r							
RegPLcdIn3 h801F	PLcdIn3[7:0] r							

RegLcdOn h8020	r0	r0	r0	r0	r0	LcdSleep rw,1,glob	LcdMux[1:0] rw,00,glob	
RegLcdSe h8021	LcdSe3 rw,1,glob	LcdSe7 rw,1,glob	LcdSe11 rw,1,glob	LcdSe15 rw,1,glob	LcdSe19 rw,1,glob	LcdSe23 rw,1,glob	LcdSe27 rw,1,glob	LcdSe31 rw,1,glob
RegLcdClkFrame h8022	LcdDivFreq[2:0] rw,000,glob			r0	r0	r0	LcdFreq[1:0] rw,00,glob	

Table 4-22. LCD driver registers.

5. Low Power Modes

5.1 Features

5.1.1 Overview

The XE8000 chips have three operating modes. These are the normal, low current and very low current modes (see Figure 5-1). The different modes are controlled by the reset and clock blocks (see the documentation of the respective blocks).

5.2 Operating mode

Start-up
All bits are reset in the design when a POR or padnreset is active.
RC is enabled, Xtal is disabled and CPU is reset (pmaddr = 0000).
If the port A is used to return from the sleep mode, all bits with nresetcold do not change (see sleep mode)

Start-up
All bits with nresetglobal and nresetpconf(if enabled) are reset. Clock configuration doesn't change except cpuck (freqdiv is reset, see clock block). CPU is reset

Active mode
This is the mode where the CPU and all peripherals can work and execute the embedded software.

Standby mode
Executing a HALT instruction moves the XE8000 into the Standby mode. The CPU is stopped, but the clocks remain active. Therefore, the enabled peripherals remain active e.g. for time keeping. A reset or an interrupt/event request (if enabled) cancels the standby mode.

Sleep mode
This is a very low-power mode because all circuit clocks and all peripherals are stopped. Only some service blocks remain active. No time-keeping is possible. Two instructions are necessary to move into sleep mode. First, the **SleepEn** (sleep enable) bit in **RegSysCtrl** has to be set to 1. The sleep mode can then be activated by setting the **Sleep** bit in **RegSysReset** to 1.

There are three possibe ways to wake-up from the sleep mode:

1. The POR (power-on-reset caused by a power-down followed by power-on). The RAM information is lost.
2. The padnreset
3. The Port A reset combination (if the Port A is present in the product). See Port A documentation for more details.

Note: If the Port A is used to return from the sleep mode, all bits with nresetcold do not change (**RegSysCtrl**, **RegSysReset** (except bit **sleep**), **Enextclock** and **Biasrc** in **RegSysClock**, **RegSysRcTrim1** and **RegSysRcTrim2**). The **SleepFlag** bit in **RegSysReset**, reads back a 1 if the circuit was in sleep mode since the flag was last cleared (see reset block for more details).

Note: It is recommended to insert a NOP instruction after the instruction that sets the circuit in sleep mode because this instruction can be executed when the sleep mode is left using the resetfromportA.

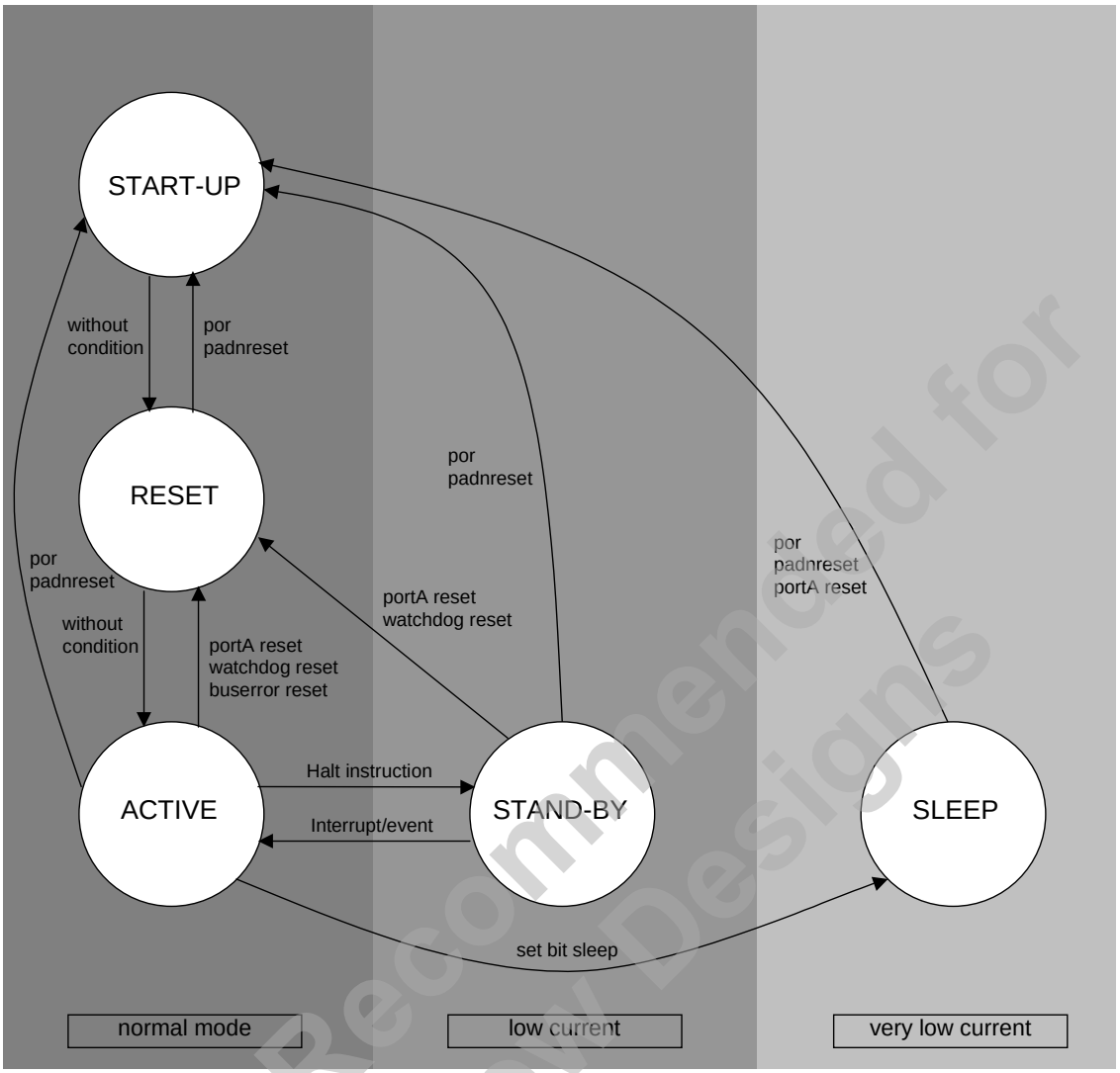


Figure 5-1. XE8000 operating modes.

6. Reset Generator

6.1 Features

- Power On Reset (POR)
- External reset from the NRESET pin
- Programmable Watchdog timer reset
- Programmable BusError reset
- Sleep mode management

Product dependant:
- Programmable Port A input combination reset

6.2 Overview

The reset block is the reset manager. It handles the different reset sources and distributes them through the system. It also controls the sleep mode of the circuit.

6.3 Register map

register name
RegSysCtrl
RegSysReset
RegSysWd

Table 6-1. Reset registers

Table 6-1 gives the different registers used by this block.

Pos.	RegSysCtrl	Rw	Reset	Function
7	SleepEn	r w	0 nresetcold	enables Sleep mode 0: sleep mode is disabled 1: sleep mode is enabled
6	EnResetPConf	r w	0 nresetcold	enables the nresetpconf signal when the nresetglobal is active 0: nresetpconf is disabled 1: nresetpconf is enabled
5	EnBusError	r w	0 nresetcold	enables reset from BusError 0: BusError reset source is disabled 1: BusError reset source is enabled
4	EnResetWD	r w	0 nresetcold	enables reset from Watchdog 0: Watchdog reset source is disabled 1: Watchdog reset source is enabled this bit can not be set to 0 by SW
3 – 0	-	r	0000	unused

Table 6-2. **RegSysCtrl** register.

Pos.	RegSysReset	Rw	Reset	Function
7	Sleep	rw	0 nresetglobal	Sleep mode control (reads always 0)
6	SleepFlag	r c	0 nresetcold	Sleep mode was active before
5	ResetBusError	r c	0 nresetcold	reset source was BusError
4	ResetWD	r c	0 nresetcold	reset source was Watchdog
3	ResetfromportA	r c	0 nresetcold	reset source was Port A combination
2 – 0		r	000	unused

Table 6-3. **RegSysReset** register

Pos.	RegSysWd	Rw	Reset	Function
7 - 4	-	r	0000	unused
3	WDKey[3]	w	0 nresetglobal	Watchdog Key bit 3
	WDCounter[3]	r		Watchdog counter bit 3
2	WDKey[2]	w	0 nresetglobal	Watchdog Key bit 2
	WDCounter[2]	r		Watchdog counter bit 2
1	WDKey[1]	w	0 nresetglobal	Watchdog Key bit 1
	WDCounter[1]	r		Watchdog counter bit 1
0	WDKey[0]	w	0 nresetglobal	Watchdog Key bit 0
	WDCounter[0]	r		Watchdog counter bit 0

Table 6-4. **RegSysWd** register

6.4 Reset handling capabilities

There are 5 reset sources:

- Power On Reset (POR)
- External reset from the NRESET pin
- Programmable port A input combination
- Programmable watchdog timer reset
- Programmable BusError reset on processor access outside the allocated memory map

Another reset source is the bit **Sleep** in the **RegSysReset** register. This source is fully controlled by software and is only used during the sleep mode.

Four internal reset signals are generated from these sources and distributed through the system:

- nresetcold: is asserted on POR or by the NRESET pin
- nresetglobal: is asserted when nresetcold or any other enabled reset source is active
- nresetsleep: is asserted when the circuit is in sleep mode
- nresetpconf: is asserted when nresetglobal is active and if the **EnResetPConf** bit in the **RegSysCtrl** register is set. This reset is generally used in the different ports. It allows to maintain the port configuration unchanged while the rest of the circuit is reset.

Table 6-5 shows a summary of the dependency of the internal reset signals on the various reset sources.

In all the tables describing the different registers, the reset source is indicated.

Asserted reset source	Internal reset signals				
	nresetglobal	nresetpconf		nresetsleep	nresetcold
		when EnResetPConf is set to 0	when EnRestPConf is set to 1		
POR	Asserted	Asserted	Asserted	Asserted	Asserted
NRESET pin	Asserted	Asserted	Asserted	Asserted	Asserted
PortA input	Asserted	-	Asserted	-	-
Watchdog	Asserted	-	Asserted	-	-
BusError	Asserted	-	Asserted	-	-
Sleep	-	-	-	Asserted	-

Table 6-5. Internal reset assertion as a function of the reset source.

6.5 Reset source description

6.5.1 Power On Reset

The power on reset (POR) monitors the external supply voltage. It activates a reset on a rising edge of this supply voltage. The reset is inactivated only if the internal voltage regulator has started up. The POR block performs no precise voltage level detection.

6.5.2 NRESET pin

Applying a low input state on the NRESET pin can activate the reset.

6.5.3 Programmable Port A input combination

Port A (if present in the product) can generate a reset signal. See the description of the Port A for further information.

6.5.4 Watchdog reset

The Watchdog will generate a reset if the **EnResetWD** bit in the **RegSysCtrl** register has been set and if the watchdog is not cleared in time by the processor. See chapter 6.8 describing the watchdog for further information.

6.5.5 BusError reset

The address space is assigned as shown in the register map of the product. If the **EnBusError** bit in the **RegSysCtrl** register is set and the software accesses an unused address, a reset is generated.

6.6 Sleep mode

Entering the sleep mode will reset a part of the circuit. The reset is used to configure the circuit for correct wake-up after the sleep mode. If the **SleepEn** bit in the **RegSysCtrl** register has been set, the sleep mode can be entered by setting the bit **Sleep** in **RegSysReset**. During the sleep mode, the nresetsleep signal is active. For detailed information on the sleep mode, see the system documentation.

6.7 Control register description and operation

Two registers are dedicated for reset status and control, **RegSysReset** and **RegSysCtrl**. The bits **Sleep, SleepFlag** and **SleepEn** are also located in those registers and are described in the chapter dedicated to the different operating modes of the circuit (system block).

The **RegSysReset** register gives information on the source that generated the last reset. It can be read at the beginning of the application program to detect if the circuit is recovering from an error or exception condition, or if the circuit is starting up normally.

- when **ResetBusError** is 1, a forbidden address access generated the reset.
- when **ResetWD** is 1, the watchdog generated the reset.
- when **ResetfromPortA** is 1, a PortA combination generated the reset.

Note: If no bit is set to 1, the reset source was either the NRESET pin or the internal POR.
Note: Several bits might be set or not, if the register was not cleared in between 2 reset occurrences.

The two other bits concern the sleep mode control and information (see system documentation for the sleep mode description).

- When **SleepFlag** is 1, the sleep mode was active before the reset occurred. This bit will always appear together with the **ResetfromPortA** bit since all other possibilities to leave the sleep mode (POR and NRESET pin) will clear the **SleepFlag**.
- When **Sleep** is set to 1, and **SleepEn** is 1, the sleep mode is entered. The bit always reads back a 0.

The **RegSysCtrl** register enables the different available reset sources and the sleep mode.

- **EnBusError** enables the reset due to a bus error condition.
- **EnResetWD** enables the reset due to the watchdog (can not be disabled once enabled).
- **EnResetPConf** enables the reset of the port configurations when reset by Port A, a Bus Error or the watchdog.
- **SleepEn** unlocks the **Sleep** bit. As long as **SleepEn** is 0, the **Sleep** bit has no effect.

6.8 Watchdog

The watchdog is a timer, which has to be cleared at least every 2 seconds by the software to prevent a reset to be generated by the timeout condition.

The watchdog can be enabled by software by setting the **EnResetWD** bit in the **RegSysCtrl** register to 1. It can then only be disabled by a power on reset or by setting the NRESET pin to a low state.

The watchdog timer can be cleared by writing consecutively the values Hx0A and Hx03 to the **RegSysWd** register. The sequence must strictly be respected to clear the watchdog.

In assembler code, the sequence to clear the watchdog is:

```
move AddrRegSysWd, #0x0A
move AddrRegSysWd, #0x03
```

Only writing Hx0A followed by Hx03 resets the WD. If some other write instruction is done to the **RegSysWd** between the writing of the Hx0A and Hx03 values, the watchdog timer will not be cleared.

It is possible to read the status of the watchdog in the **RegSysWd** register. The watchdog is a 4 bit counter with a count range between 0 and 7. The system reset is generated when the counter is reaching the value 8.

6.9 Start-up and watchdog specifications

At start-up of the circuit, the POR block generates a reset signal during t_{POR}. The circuit starts software execution after this period (see system chapter). The POR is intended to force the circuit into a correct state at start-up. For precise monitoring of the supply voltage, the voltage level detector (VLD) has to be used.

Symbol	Parameter	Min	Typ	Max	Unit	Comments
T_{POR}	POR reset duration	5		20	ms	
Vbat_sl	Supply ramp up	0.5			V/ms	1
WDtime	Watchdog timeout period	2			s	2

Table 6. Electrical and timing specifications

Note: 1) The Vbat_sl defines the minimum slope required on VBAT. Correct start-up of the circuit is not guaranteed if this slope is too slow. In such a case, a delay has to be built using the NRESET pin.

Note: 2) The minimal watchdog timeout period is guaranteed when the internal oscillators are used. In case an external clock source is used, the watchdog timeout period will be correct in so far the contents of the **RegSysRCTrim1** and **RegSysRCTrim2** registers are correct (see clock block documentation for more details).

7. Clock Generation

7.1 Features

3 available clock sources (RC oscillator, quartz oscillator and external clock).

- 2 divider chains: high-prescaler (8 bits) and low-prescaler (15 bits).
- CPU clock disabling in halt mode.

7.2 Overview

The XE88LCxx chips can work on different clock sources (RC oscillator, quartz oscillator and external clock). The clock generator block is in charge of distributing the necessary clock frequencies to the circuit.

Figure 7-1 represents the functionality of the clock block.

The internal RC oscillator or an external clock source can be selected to drive the high prescaler. This prescaler generates frequency divisions down to 1/256 of its input frequency. A 32kHz clock is generated by enabling the quartz oscillator (if present in the product) or by selecting the appropriate tap on the high prescaler. The low prescaler generates clock signals from 32kHz down to 1Hz. The clock source for the CPU can be selected from the RC oscillator, the external clock or the 32kHz clock.

7.3 Register map

pos.	RegSysClock	rw	reset	function
7	CpuSel	r/w	0 nresetsleep	Select speed for cpuck
6	-	r	0	Unused
5	EnExtClock	r/w	0 nresetcold	Enable for external clock
4	BiasRc	r/w	1 nresetcold	Enable Rcbias (reduces start-up time of RC).
3	ColdXtal	r	1 nresetsleep	Xtal in start phase
2	-	r	0	Unused
1	EnableXtal	r/w	0 nresetsleep	Enable Xtal oscillator
0	EnableRc	r/w	1 nresetsleep	Enable RC oscillator

Table 7-1: **RegSysClock** register

pos.	RegSysMisc	rw	reset	function
7-2	--	r	000000	Unused
1	Output16k	r/w	0 nresetsleep	Output 16 kHz signal on PB[3]
0	OutputCpuCk	r/w	0 nresetsleep	Output CPU clock on PB[2]

Table 7-2: **RegSysMisc** register

pos.	RegSysPre0	rw	reset	function
7-1	--	r	0000000	Unused
0	ClearLowPrescal	w1 r0	0	Write 1 to reset low prescaler, but always reads 0

Table 7-3: **RegSysPre0** register

pos.	RegSysRcTrim1	rw	reset	function
7-5	--	r	000	Unused
4	RcFreqRange	r/w	0 nresetcold	Low/high freq. range (low=0)
3	RcFreqCoarse[3]	r/w	0 nresetcold	RC coarse trim bit 3
2	RcFreqCoarse[2]	r/w	0 nresetcold	RC coarse trim bit 2
1	RcFreqCoarse[1]	r/w	0 nresetcold	RC coarse trim bit 1
0	RcFreqCoarse[0]	r/w	1 nresetcold	RC coarse trim bit 0

Table 7-4: **RegSysRCTrim1** register

pos.	RegSysRcTrim2	rw	reset	function
7-6	--	r	00	Unused
5	RcFreqFine[5]	r/w	0 nresetcold	RC fine trim bit 5
4	RcFreqFine[4]	r/w	0 nresetcold	RC fine trim bit 4
3	RcFreqFine[3]	r/w	0 nresetcold	RC fine trim bit 3
2	RcFreqFine[2]	r/w	0 nresetcold	RC fine trim bit 2
1	RcFreqFine[1]	r/w	0 nresetcold	RC fine trim bit 1
0	RcFreqFine[0]	r/w	0 nresetcold	RC fine trim bit 0

Table 7-5: **RegSysRCTrim2** register

pos.	RegSysPtckmode	rw	reset	function
7-1	--	r	0000000	Unused
0	Reserved	r/w	0 nresetglobal	Reserved

Table 7-6: **RegSysPtckmode** register

7.4 Interrupts and events map

interrupt source	Default mapping in the interrupt manager	Default mapping in the event manager
ck128Hz	RegIrqHig(6)	RegEvn(5)
ck1Hz	RegIrqMid(3)	RegEvn(1)

Table 7-7: Interrupts and events map

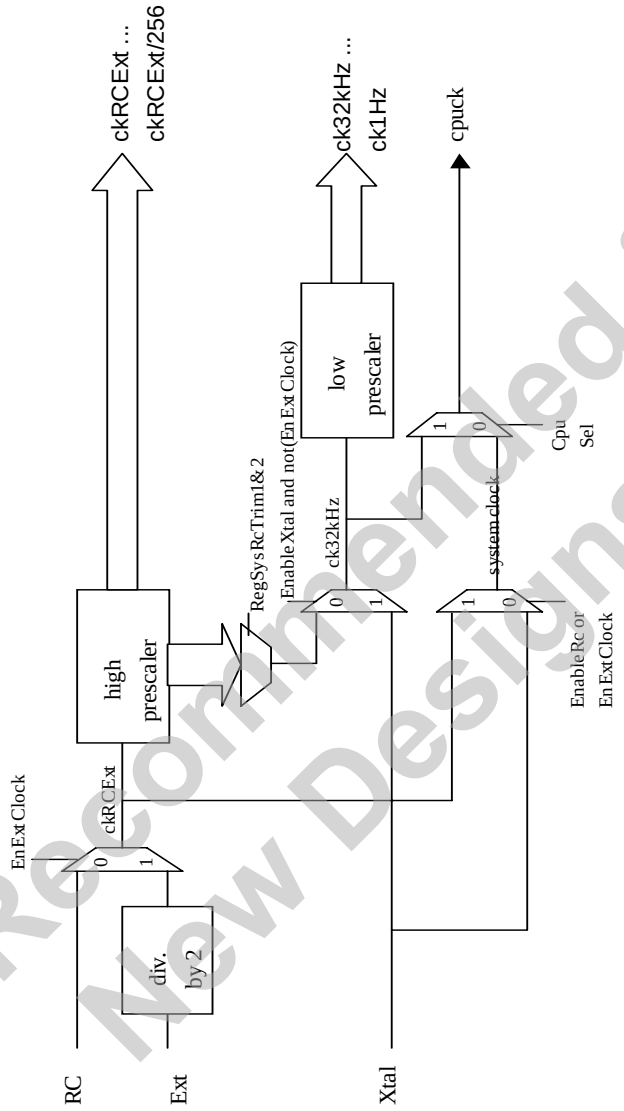


Figure 7-1. Clock block structure

7.5 Clock sources

7.5.1 RC oscillator Configuration

The RC oscillator is always turned on and selected for CPU and system operation at power-on reset, pad NRESET, and when exiting sleep mode. It can be turned off after the Xtal (quartz oscillator) has been started, after selection of the external clock or by entering sleep mode.

The RC oscillator has two frequency ranges: sub-MHz (50 kHz to 0.5 MHz) and above-MHz (0.5 MHz to 5 MHz). Inside a range, the frequency can be tuned by software for coarse and fine adjustment. See registers **RegSysRcTrim1** and **RegSysRcTrim2**.

Bit **EnableRc** in register **RegSysClock** controls the propagation of the RC clock signal and the operation of the oscillator. The user can stop the RC oscillator by resetting the bit **EnableRc**. Entering the sleep mode disables the RC oscillator.

Note: The RC oscillator bias can be maintained while the oscillator is disabled by setting the bit **BiasRc** in **RegSysClock**. This allows a faster restart of the RC oscillator at the cost of increased power consumption (see section 7.5.3).

7.5.2 RC oscillator frequency tuning

The RC oscillator frequency can be set using the bits in the **RegSysRcTrim1** and **RegSysRcTrim2** registers. Figure 7-2 shows the nominal frequency of the RC oscillator as a function of these bits. The absolute value of the frequency for a given register content may change by ±35% from chip to chip due to the tolerances on the integrated capacitors and resistors. However, the modification of the frequency as a function of a modification of the register content is fairly precise. This means that the curves in Figure 7-2 can shift up and down but that the slope remains unchanged.

The bit **RcFreqRange** modifies the oscillator frequency by a factor of 10. The upper curve in the figure corresponds to **RcFreqRange**=1.

The **RcFreqCoarse** modifies the frequency of the oscillator by a factor (**RcFreqCoarse**+1). The figure represents the frequency for 5 different values of the bits **RcFreqCoarse**: for each value the frequency is multiplied by 2.

Incrementing the **RcFreqFine** code, increases the frequency by about 1.4%.

The frequency of the oscillator is therefor given by:

$$f_{RC}=f_{Rcmin}\cdot(1+9\cdot\mathbf{RcFreqRange})\cdot(1+\mathbf{RcFreqCoarse})\cdot(1.014)^{\mathbf{RcFreqFine}}$$

with f_{Rcmin} the RC oscillator frequency if the registers are all 0.

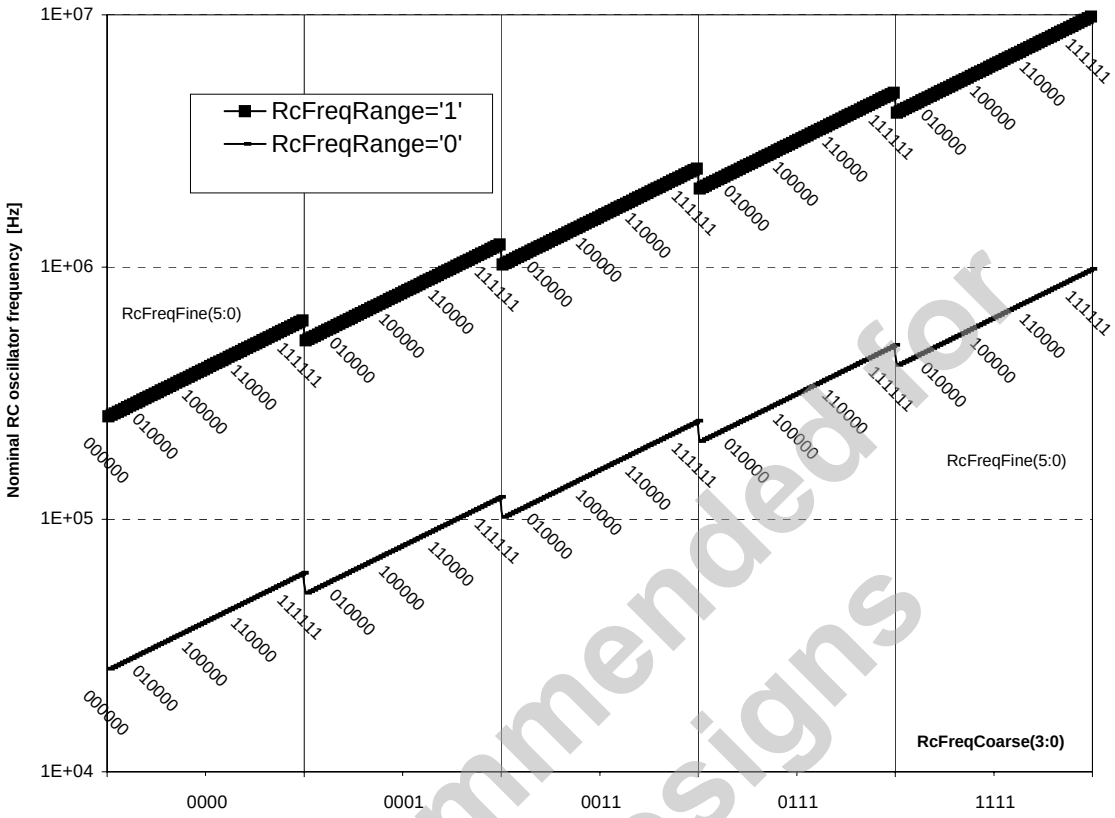


Figure 7-2. RC oscillator nominal frequency tuning.

7.5.3 RC oscillator specifications

sym	description	min	typ	max	unit	Comments
f_{RCmin}	Lowest RC frequency	25	40	55	kHz	Note 1
RcFreqFine	fine tuning step		1.4	2.0	%	
RC_su	startup time		30	50	us	**BiasRc**=0
			3	5	us	**BiasRc**=1
PSRR @ DC	Supply voltage dependence		TBD		%/V	Note 2
			TBD		%/V	Note 3
$\Delta f/\Delta T$	Temperature dependence		0.1		%/°C	

Table 7-8. RC oscillator specifications

Note 1: this is the frequency tolerance when all trimming codes are 0. The frequency at start-up is about twice as high.

Note 2: frequency shift as a function of VBAT with normal regulator function.

Note 3: frequency shift as a function of VBAT while the regulator is short-circuited to VBAT.

The tolerances on the minimal frequency and the drift with supply or temperature can be cancelled using the software or hardware DFLL (digital frequency locked loop) which uses the crystal oscillator as a reference frequency.

7.6 Xtal oscillator

7.6.1 Xtal configuration

The Xtal operates with an external crystal of 32'768 Hz. During Xtal oscillator start-up, the first 32768 cycles are masked. The two bits **EnableXtal** and **ColdXtal** in register **RegSysClock** control the oscillator.

At power-on reset, a pad NRESET pulse or during sleep mode, **EnableXtal** is reset and **ColdXtal** is set (Xtal oscillator is not selected at start-up). The user can start Xtal oscillator by setting **EnableXtal**. When the Xtal oscillator starts, bit **ColdXtal** is reset after 32768 cycles. Before **ColdXtal** is reset by the system, the Xtal frequency precision is not guaranteed. The Xtal oscillator can be stopped by the user by resetting bit **EnableXtal**.

When the user enters into sleep mode, the Xtal is stopped.

7.6.2 Xtal oscillator specifications

The crystal oscillator has been designed for a crystal with the specifications given in Table 7-9. The oscillator precision can only be guaranteed for this crystal.

Symbol	Description	Min	Typ	Max	Unit	Comments
Fs	Resonance frequency		32768		Hz	
CL	CL for nominal frequency		8.2	15	pF	
Rm	Motional resistance		40	100	kΩ	
Cm	Motional capacitance	1.8	2.5	3.2	fF	
C0	Shunt capacitance	0.7	1.1	2.0	pF	
Rmp	Motional resistance of 6^{th} overtone (parasitic)	4	8		kΩ	
Q	Quality factor	30k	50k	400k	-	

Table 7-9. Crystal specifications.

For safe operation, low power consumption and to meet the specified precision, careful board layout is required:

- Keep lines XIN and XOUT short and insert a VSS line in between them.
- Connect the crystal package to VSS.
- No noisy or digital lines near XIN or XOUT.
- Insert guards where needed.
- Respect the board specifications of Table 7-10.

Symbol	Description	Min	Typ	Max	Unit	Comments
Rh_xin	Resistance XIN-VSS	10			MΩ	
Rh_xout	Resistance XOUT-VSS	10			MΩ	
Rh_xin_xout	Resistance XIN-XOUT	50			MΩ	
Cp_xin	Capacitance XIN-VSS	0.5		3.0	pF	
Cp_xout	Capacitance XOUT-VSS	0.5		3.0	pF	
Cp_xin_xout	Capacitance XIN-XOUT	0.2		1.0	pF	

Table 7-10. Board layout specifications.

The oscillator characteristics are given in Table 7-11. The characteristics are valid only if the crystal and board layout meet the specifications above.

Symbol	Description	Min	Typ	Max	Unit	Comments
f_{Xtal}	Nominal frequency		32768		Hz	
St_xtal	Start-up time		1	2	s	
Fstab	Frequency deviation	-100		300	ppm	Note 1

Table 7-11. Crystal oscillator characteristics.

Note 1. This gives the relative frequency deviation from nominal for a crystal with CL=8.2pF and within the temperature range -40°C to 85°C. The crystal tolerance, crystal aging and crystal temperature drift are not included in this figure.

7.7 External clock

7.7.1 External clock configuration

The user can provide an external clock instead of the internal oscillators. The external provided frequency is internally divided by two. The external clock input pin is XIN.

The system is configured for external clock by bit **EnExtClock** in register **RegSysClock**. Using the bits in the registers **RegSysRcTrim1** and **RegSysRcTrim2**, the ck32kHz clock frequency can be controlled (see section 7.10).

Note: when using the external clock, the Xtal is not available.

7.7.2 External clock specification

The external clock has to satisfy the specifications in the table below. Correct behavior of the circuit can not be guaranteed if the external clock signal does not respect the specifications below.

Symbol	Description	Min	Typ	Max	Unit	Comments
F_{EXT}	External clock frequency			8	MHz	Note 1
PW_1	Pulse 1 width	0.06			µs	Note 1
PW_0	Pulse 0 width	0.03		20	µs	Note 1
F_{EXT_LV}	External clock frequency			TBD	kHz	Note 2
PW_1_LV	Pulse 1 width	TBD			µs	Note 2
PW_0_LV	Pulse 0 width	TBD		20	µs	Note 2

Table 7-12. External clock specifications.

Note 1. For VBAT≥2.4V

Note 2. For VBAT=VREG=1.2V

7.8 Clock source selection

There are three possible clock sources available for the CPU clock. The RC clock is always selected after power-up, a negative pulse on NRESET or after Sleep mode. The CPU clock selection is done with the bit **CpuSel** in **RegSysClock** (0= fastest clock, 1= 32 kHz from Xtal if **EnableXtal** =1 and **EnExtClock** = 0 else from high prescaler 32 kHz output).

Switching from one clock source to another is glitch free.

The next table summarizes the different clock configurations of the circuit:

Clock Sources				Clock targets			
Mode name	**EnExtClock**	**EnableRc**	**EnableXtal**	Cpuck		High Prescaler Clock input	Low Prescaler Clock input
				CpuSel=0	**CpuSel**=1		
Sleep	0	0	0	off	off	off	off
Xtal	0	0	1	Xtal	Xtal	off	Xtal
RC	0	1	0	RC NOTE 2	High presc.	RC	High presc.
RC + Xtal	0	1	1	RC NOTE 1 and 2	Xtal	RC NOTE 1	Xtal
External	1	X	X	External NOTE 2	High presc.	External	High presc.

Table 7-13: Table of clocking modes.

Note 1: The frequency of the RC must be higher than 100 kHz when Xtal is enabled in order to ensure a proper 32 kHz operation.

Note 2: The clock RC can be divided by the value of freq instruction (see coolrisc instruction information)

freq instruction	cpuck
nodiv	RC or external
div2	RC/2 or external/2
div4	RC/4 or external/4
div8	RC/8 or external/8
div16	RC/16 or external/16

Note 3: Switching from one clock source to another and stopping the unused clock source must be performed using 2 MOVE instructions to **RegSysClock**. First select the new clock source and then stop the unused one.

7.9 Prescalers

The clock generator block embeds two divider chains: the high prescaler and the low one.

The high prescaler is made of an 8 stage dividing chain and the low prescaler of a 15 stage dividing chain.

Features:

- High prescaler can only be driven with RC clock or external clock (bits **EnableRc** or **EnExtClock** have to be set, see Table 7-13).
- Low prescaler can be driven from the high prescaler or directly with the Xtal clock when bit **EnableXtal** is set to 1 and bit **EnExtClock** is set to 0.
- Bit **ClearLowPrescal** in the **RegSysPre0** register allows to reset synchronously the low prescaler, the low prescaler is also automatically cleared when bit **EnableXtal** is set. Both dividing chains are reset asynchronously by the *nresetglobal* signal.
- Bit **ColdXtal**=1 indicates the Xtal is in its start phase. It is active for 32768 Xtal cycles after setting **EnableXtal**.

7.10 32 kHz frequency selector

A decoder is used to select from the high prescaler, the frequency tap that is the closest to 32 kHz to operate the low prescaler when the Xtal is not running. In this case, the RC oscillator frequency of ±35% will also be valid for the low prescaler frequency outputs.

The next table shows how the RC trimming values in the **RegSysRcTrim1** and **RegSysRcTrim2** registers select the 32 kHz frequency. The least significant bits of the **RcFreqFine** word are not used.

In order to ensure the correct frequency selection for the low prescaler when having an external clock, a proper value must be set in the RC trim registers. The code can be selected from the table below as a function of the frequency ratio between half the frequency of the external clock and 32kHz. If the frequency is not set correctly, all timings derived from the low prescaler will be shifted accordingly (e.g. watchdog frequencies) and some peripherals may no longer function correctly if the deviation from 32kHz is too large (e.g. the voltage level detector).

RcFreqRange&RcFreqCoarse(3:0)&RcFreqFine(5:3)	Selected high prescaler tap
Default case (0'0001'000)	Ckrcext/2
From 0'0000'000 to 0'0000'100	Ckrcext
From 0'0000'101 to 0'0001'100	Ckrcext/2
From 0'0001'101 to 0'0001'111	Ckrcext/4
0'0010'000	Ckrcext/2
From 0'0010'001 to 0'0010'110	Ckrcext/4
0'0010'111	Ckrcext/8
From 0'0011'000 to 0'0011'100	Ckrcext/4
From 0'0011'101 to 0'0011'111	Ckrcext/8
From 0'0100'000 to 0'1000'010	Ckrcext/4
From 0'0100'011 to 0'0100'111	Ckrcext/8
0'0101'000	Ckrcext/4
From 0'0101'001 to 0'0101'110	Ckrcext/8
0'0101'111	Ckrcext/16
From 0'0110'000 to 0'0110'101	Ckrcext/8
From 0'0110'110 to 0'0110'111	Ckrcext/16
From 0'0111'000 to 0'0111'100	Ckrcext/8
From 0'0111'101 to 0'0111'111	Ckrcext/16
From 0'1000'000 to 0'1000'011	Ckrcext/8
From 0'1000'100 to 0'1000'111	Ckrcext/16
From 0'1001'000 to 0'1001'010	Ckrcext/8
From 0'1001'011 to 0'1001'111	Ckrcext/16
From 0'1010'000 to 0'1010'001	Ckrcext/8
From 0'1010'010 to 0'1010'111	Ckrcext/16
0'1011'000	Ckrcext/8
From 0'1011'001 to 0'1011'110	Ckrcext/16
0'1011'111	Ckrcext/32
From 0'1100'000 to 0'1100'110	Ckrcext/16
0'1100'111	Ckrcext/32
From 0'1101'000 to 0'1101'101	Ckrcext/16
From 0'1101'110 to 0'1101'111	Ckrcext/32
From 0'1110'000 to 0'1110'100	Ckrcext/16
From 0'1110'101 to 0'1110'111	Ckrcext/32
From 0'1111'000 to 0'1111'100	Ckrcext/16
From 0'1111'101 to 0'1111'111	Ckrcext/32
From 1'0000'000 to 1'0000'010	Ckrcext/8
From 1'0000'011 to 1'0001'010	Ckrcext/16
From 1'0001'011 to 1'0010'100	Ckrcext/32
From 1'0010'101 to 1'0010'111	Ckrcext/64
From 1'0011'000 to 1'0011'010	Ckrcext/32
From 1'0011'011 to 1'0011'111	Ckrcext/64
1'0100'000	Ckrcext/32
From 1'0100'001 to 1'0100'110	Ckrcext/64
1'0100'111	Ckrcext/128
From 1'0101'000 to 1'0101'100	Ckrcext/64
From 1'0101'101 to 1'0101'111	Ckrcext/128
From 10110'000 to 1'0110'011	Ckrcext/64
From 1'0110'100 to 1'0110'111	Ckrcext/128
From 1'0111'000 to 1'0111'010	Ckrcext/64
From 1'0111'011 to 1'0111'111	Ckrcext/128
From 1'1000'000 to 1'1000'001	Ckrcext/64
From 1'1000'010 to 1'1000'111	Ckrcext/128
1'1001'000	Ckrcext/64
From 1'1001'001 to 1'1111'111	Ckrcext/128

Table 7-14: Table of 32kHz high prescaler tap decoder.

8. Interrupt Handler

8.1 Features

The XE8000 chips support 24 interrupt sources, divided into 3 levels of priority.

8.2 Overview

The interrupt handler allows to manage 24 interrupt sources individually.

The 24 interrupt sources are divided into 3 levels of priority: High (8 interrupt sources), Mid (8 interrupt sources), and Low (8 interrupt sources). Those 3 levels of priority are directly mapped to those supported by the CoolRisc (IN0, IN1 and IN2; see CoolRisc documentation for more information).

Additional functions are given that allow fast detection of the highest priority interrupt that has been activated.

8.3 Register map

Register name
RegIrqHig
RegIrqMid
RegIrqLow
RegIrqEnHig
RegIrqEnMid
RegIrqEnLow
RegIrqPriority
RegIrqIrq

Table 8-1: IRQ handler registers

pos.	RegIrqHig	rw	reset	function
7	RegIrqHig[7]	r c1	0 nresetglobal	interrupt #23 (high priority) clear interrupt #23 when 1 is written
6	RegIrqHig[6]	r c1	0 nresetglobal	interrupt #22 (high priority) clear interrupt #22 when 1 is written
5	RegIrqHig[5]	r c1	0 nresetglobal	interrupt #21 (high priority) clear interrupt #21 when 1 is written
4	RegIrqHig[4]	r c1	0 nresetglobal	interrupt #20 (high priority) clear interrupt #20 when 1 is written
3	RegIrqHig[3]	r c1	0 nresetglobal	interrupt #19 (high priority) clear interrupt #19 when 1 is written
2	RegIrqHig[2]	r c1	0 nresetglobal	interrupt #18 (high priority) clear interrupt #18 when 1 is written
1	RegIrqHig[1]	r c1	0 nresetglobal	interrupt #17 (high priority) clear interrupt #17 when 1 is written
0	RegIrqHig[0]	r c1	0 nresetglobal	interrupt #16 (high priority) clear interrupt #16 when 1 is written

Table 8-2: **RegIrqHig**

pos.	RegIrqMid	rw	reset	function
7	RegIrqMid[7]	r c1	0 nresetglobal	interrupt #15 (mid priority) clear interrupt #15 when 1 is written
6	RegIrqMid[6]	r c1	0 nresetglobal	interrupt #14 (mid priority) clear interrupt #14 when 1 is written
5	RegIrqMid[5]	r c1	0 nresetglobal	interrupt #13 (mid priority) clear interrupt #13 when 1 is written
4	RegIrqMid[4]	r c1	0 nresetglobal	interrupt #12 (mid priority) clear interrupt #12 when 1 is written
3	RegIrqMid[3]	r c1	0 nresetglobal	interrupt #11 (mid priority) clear interrupt #11 when 1 is written
2	RegIrqMid[2]	r c1	0 nresetglobal	interrupt #10 (mid priority) clear interrupt #10 when 1 is written
1	RegIrqMid[1]	r c1	0 nresetglobal	interrupt #9 (mid priority) clear interrupt #9 when 1 is written
0	RegIrqMid[0]	r c1	0 nresetglobal	interrupt #8 (mid priority) clear interrupt #8 when 1 is written

Table 8-3: **RegIrqMid**

pos.	RegIrqLow	rw	reset	function
7	RegIrqLow[7]	r c1	0 nresetglobal	interrupt #7 (low priority) clear interrupt #7 when 1 is written
6	RegIrqLow[6]	r c1	0 nresetglobal	interrupt #6 (low priority) clear interrupt #6 when 1 is written
5	RegIrqLow[5]	r c1	0 nresetglobal	interrupt #5 (low priority) clear interrupt #5 when 1 is written
4	RegIrqLow[4]	r c1	0 nresetglobal	interrupt #4 (low priority) clear interrupt #4 when 1 is written
3	RegIrqLow[3]	r c1	0 nresetglobal	interrupt #3 (low priority) clear interrupt #3 when 1 is written
2	RegIrqLow[2]	r c1	0 nresetglobal	interrupt #2 (low priority) clear interrupt #2 when 1 is written
1	RegIrqLow[1]	r c1	0 nresetglobal	interrupt #1 (low priority) clear interrupt #1 when 1 is written
0	RegIrqLow[0]	r c1	0 nresetglobal	interrupt #0 (low priority) clear interrupt #0 when 1 is written

Table 8-4: **RegIrqLow**

pos.	RegIrqEnHig	rw	reset	function
7	RegIrqEnHig[7]	rw	0	1= enable interrupt #23
6	RegIrqEnHig[6]	rw	0	1= enable interrupt #22
5	RegIrqEnHig[5]	rw	0	1= enable interrupt #21
4	RegIrqEnHig[4]	rw	0	1= enable interrupt #20
3	RegIrqEnHig[3]	rw	0	1= enable interrupt #19
2	RegIrqEnHig[2]	rw	0	1= enable interrupt #18
1	RegIrqEnHig[1]	rw	0	1= enable interrupt #17
0	RegIrqEnHig[0]	rw	0	1= enable interrupt #16

Table 8-5: **RegIrqEnHig**

pos.	RegIrqEnMid	rw	reset	function
7	RegIrqEnMid[7]	rw	0	1= enable interrupt #15
6	RegIrqEnMid[6]	rw	0	1= enable interrupt #14
5	RegIrqEnMid[5]	rw	0	1= enable interrupt #13
4	RegIrqEnMid[4]	rw	0	1= enable interrupt #12
3	RegIrqEnMid[3]	rw	0	1= enable interrupt #11
2	RegIrqEnMid[2]	rw	0	1= enable interrupt #10
1	RegIrqEnMid[1]	rw	0	1= enable interrupt #9
0	RegIrqEnMid[0]	rw	0	1= enable interrupt #8

Table 8-6: **RegIrqEnMid**

pos.	RegIrqEnLow	rw	reset	function
7	RegIrqEnLow[7]	rw	0	1= enable interrupt #7
6	RegIrqEnLow[6]	rw	0	1= enable interrupt #6
5	RegIrqEnLow[5]	rw	0	1= enable interrupt #5
4	RegIrqEnLow[4]	rw	0	1= enable interrupt #4
3	RegIrqEnLow[3]	rw	0	1= enable interrupt #3
2	RegIrqEnLow[2]	rw	0	1= enable interrupt #2
1	RegIrqEnLow[1]	rw	0	1= enable interrupt #1
0	RegIrqEnLow[0]	rw	0	1= enable interrupt #0

Table 8-7: **RegIrqEnLow**

pos.	RegIrqPriority	rw	reset	function
7-0	RegIrqPriority	r	11111111	code of highest priority set

Table 8-8: **RegIrqPriority**

pos.	RegIrqIrq	rw	reset	function
7-3	-	r	00000	unused
2	IrqHig	r	0	one or more high priority interrupts is set
1	IrqMid	r	0	one or more mid priority interrupts is set
0	IrqLow	r	0	one or more low priority interrupts is set

Table 8-9: **RegIrqIrq**

8.4 Detailed description

The CoolRISC core has 3 different interrupt levels IN0, IN1 and IN2 (Figure 8-1). When these interrupts are triggered, the program counter (PC) is loaded with a fixed address. In case more than one interrupt occurs simultaneously, the execution order is IN0, IN1, IN2.

The masking, setting and clearing of these interrupts can be done in the stat register (see chapter describing the CPU).

The interrupt handler bundles a certain number of interrupt sources and routes them to one of these three interrupts and provides the possibility to enable/disable each of them individually. The definition of the interrupt sources is given in the memory mapping chapter.

RegIrqHig, RegIrqMid, and RegIrqLow are 8-bit registers containing flags for the interrupt sources. Those flags are set when the interrupt is enabled (i.e. if the corresponding bit in the registers **RegIrqEnHig**, **RegIrqEnMid** or **RegIrqEnLow** is set) and a rising edge is detected on the corresponding interrupt source.

Once memorized, an interrupt flag can be cleared by writing a '1' in the corresponding bit of **RegIrqHig**, **RegIrqMid** or **RegIrqLow**. Writing a '0' does not modify the flag. To definitively clear the interrupt, one has to clear the CoolRISC interrupt in the CoolRISC stat register. All interrupts are automatically cleared after a reset.

Two registers are provided to facilitate the writing of interrupt service software. **RegIrqPriority** contains the number of the highest priority set (its value is 0xFF when no interrupt is memorized). **RegIrqIrq** indicates the priority level of the currently activated interrupts.

All interrupt sources are sampled by the highest frequency in the system. A CPU interruption is generated and memorized when an interrupt becomes high. Between the rising edge of the interrupt on the peripheral and the rising edge on the CoolRISC core, there is a latency of one clock cycle.

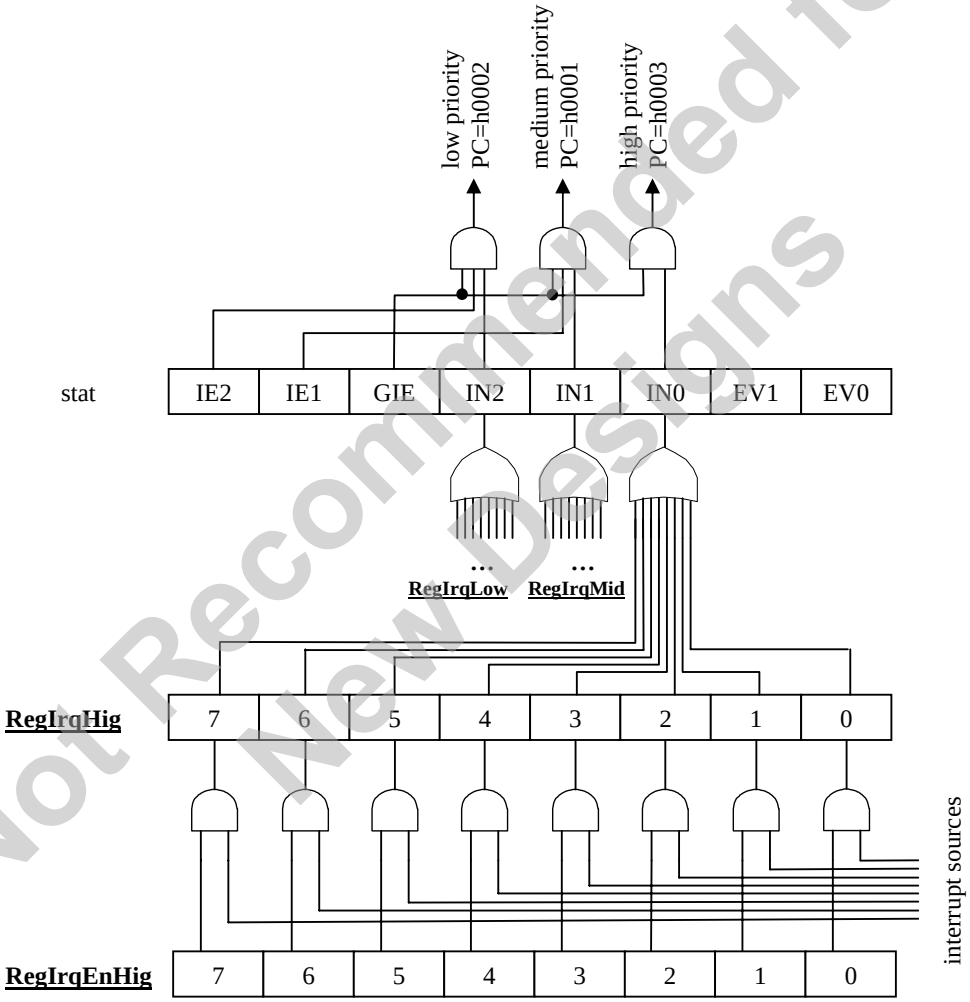


Figure 8-1. Principle of the interrupt handler.

8.5 Interrupt handling software

This chapter describes an example of the software used for the interrupt handler. This software is present by default in the software development environments. It represents only one of several possible ways of handling the interrupts.

First of all, the jump addresses are defined at the beginning of the crt0.s file. In our case, all three interrupt levels jump to the same place (defined by the _interrupt label), but this can be changed if required.

```
##############################################################################
## Reset & interrupt vectors
##############################################################################
_start:
    jump  main_init   ; reset
    jump  _interrupt  ; IN1
    jump  _interrupt  ; IN2
    jump  _interrupt  ; IN0
```

The first thing to do when an interrupt is activated is to save the context. You have to start with saving the contents of the accumulator, then the flags and finally the internal CPU registers. You will find this part of the code in the IRQComon_xx.s file.

```
_interrupt:
##############################################################################
## Save all registers and flags
##############################################################################
      move    -(i3), a
      move    a, r0
      sflag
      move    -(i3), a
      move    -(i3), ipl
      move    -(i3), iph
      move    -(i3), i0l
      move    -(i3), i0h
      move    -(i3), i1l
      move    -(i3), i1h
      move    -(i3), i2l
      move    -(i3), i2h
      move    -(i3), r0
      move    -(i3), r1
      move    -(i3), r2
      move    -(i3), r3
```

Next step is to determine which interrupt is activated. In this case, we use the value in the **RegIrqPriority** register to determine the highest priority interrupt that was activated. Other ways can be used, especially when the priority order fixed in the hardware needs to be changed. You will find this part of the code in the IRQComon_xx.s file. In this example, the labels are used as defined for the XE8802.

```
##############################################################################
## The following lines enables the adress calculation of the interrupt
## table. Where RegIrqPriority is the addres offset for the table.
## The RegIrqPriority valid values are between 0x00 until 0x017. The
## 0xFF value should never exist.
##############################################################################
    move  r0,RegIrqPriority
    calls _interrupttab                ; save pc+1 in ip
_interrupttab:
    add   ipl,#0x05                  ; add the offset, nb instr. before table
    addc  iph,#0x00                    ; propagate carry
    add   ipl,r0                       ; add the offset of the regirqpriority
    addc  iph,#0x00                    ; propagate carry
    rets                               ; put ip in pc

    ; interrupt table
    jump  ret_int                      ;  RegIrqPriority = 0x00
    jump  ret_int                      ;  RegIrqPriority = 0x01
    jump  Irq_Pa2                      ;  RegIrqPriority = 0x02
    jump  Irq_Pa3                      ;  RegIrqPriority = 0x03
    jump  Irq_CntD                     ;  RegIrqPriority = 0x04
    jump  Irq_CntB                     ;  RegIrqPriority = 0x05
    jump  Irq_Pa6                      ;  RegIrqPriority = 0x06
    jump  Irq_Pa7                      ;  RegIrqPriority = 0x07
    jump  Irq_Pa0                      ;  RegIrqPriority = 0x08
```

```
        jump  Irq_Pa1                   ;  RegIrqPriority = 0x09
        jump  Irq_Vld                   ;  RegIrqPriority = 0x0A
        jump  Irq_1Hz                   ;  RegIrqPriority = 0x0B
        jump  Irq_Pa4                   ;  RegIrqPriority = 0x0C
        jump  Irq_Pa5                   ;  RegIrqPriority = 0x0D
        jump  Irq_UsrtCond1             ;  RegIrqPriority = 0x0E
        jump  Irq_UsrtCond2             ;  RegIrqPriority = 0x0F
        jump  Irq_UartRx                ;  RegIrqPriority = 0x10
        jump  Irq_UartTx                ;  RegIrqPriority = 0x11
        jump  Irq_Cmpd                  ;  RegIrqPriority = 0x12
        jump  Irq_CntC                  ;  RegIrqPriority = 0x13
        jump  Irq_CntA                  ;  RegIrqPriority = 0x14
        jump  Irq_Spi                   ;  RegIrqPriority = 0x15
        jump  Irq_128Hz                 ;  RegIrqPriority = 0x16
        jump  Irq_AC                    ;  RegIrqPriority = 0x17
```

The next steps are to clear the interrupt flag in the interrupt handler, to call the specific function for the identified interrupt source and to clear the interrupt in the stat register. This code can be found in the file IRQSave0_xx.s.

```
#############################################################################
Irq_AC:
    move  RegIrqHig, #0x80
    calls Handle_Irq_AC
    jump  ret_int0
#############################################################################
Irq_128Hz:
    move  RegIrqHig, #0x40
    calls Handle_Irq_128Hz
    jump  ret_int0
#############################################################################
Irq_Spi:
    move  RegIrqHig, #0x20
    calls Handle_Irq_Spi
    jump  ret_int0

…

ret_int0:
    clrb  stat, #2
    jump  ret_int
```

Finally, the context and the PC have to be restored. This code can be found in the IRQComon_xx.s file.

```
ret_int:
#############################################################################
## Restore all registers and flags
#############################################################################
      move    r3, (i3)+
      move    r2, (i3)+
      move    r1, (i3)+
      move    r0, (i3)+
      move    i2h, (i3)+
      move    i2l, (i3)+
      move    i1h, (i3)+
      move    i1l, (i3)+
      move    i0h, (i3)+
      move    i0l, (i3)+
      move    iph, (i3)+
      move    ipl, (i3)+
      rflag   (i3)+
      move    a, (i3)+
      reti
#############################################################################
## End of interrupt handlers
#############################################################################
```

9. Event Handler

9.1 Features

The XE8000 chips support 8 event sources, divided into 2 levels of priority.

9.2 Overview

An event is different from an interrupt in that it does not modify the program counter (PC). Events are used by two microcontroller instructions.

First of all, events are useful to wake-up the microcontroller when it is in HALT. The software execution then simply resumes at the instruction next to the HALT instruction. The second instruction is the conditional jump on event (JEV). The jump is executed if one of the event flags in the stat register is set. In all other cases, the occurrence of an event has no effect.

The event handler allows to manage 8 event sources individually. The 8 event sources are divided into 2 levels of priority: High (4 event sources) and Low (4 event sources). Those 2 levels of priority are directly mapped to those supported by the CoolRisc™ (EV0and IN1; see CoolRisc™ documentation for more information).

Additional functions are given that allow fast detection of the highest priority event that has been activated.

9.3 Register map

The addresses given in Table 9-1 are the default values and may be different in some products.

Register name
RegEvn
RegEvnEn
RegEvnPriority
RegEvnEvn

Table 9-1: EVN handler registers.

pos.	RegEvn	rw	reset	function
7	RegEvn[7]	r c1	0 nresetglobal	event #7 (high priority) clear event #7 when written 1
6	RegEvn[6]	r c1	0 nresetglobal	event #6 (high priority) clear event #6 when written 1
5	RegEvn[5]	r c1	0 nresetglobal	event #5 (high priority) clear event #5 when written 1
4	RegEvn[4]	r c1	0 nresetglobal	event #4 (high priority) clear event #4 when written 1
3	RegEvn[3]	r c1	0 nresetglobal	event #3 (low priority) clear event #3 when written 1
2	RegEvn[2]	r c1	0 nresetglobal	event #2 (low priority) clear event #2 when written 1
1	RegEvn[1]	r c1	0 nresetglobal	event #1 (low priority) clear event #1 when written 1
0	RegEvn[0]	r c1	0 nresetglobal	event #0 (low priority) clear event #0 when written 1

Table 9-2: **RegEvn**

pos.	RegEvnEn	rw	reset	function
7	RegEvnEn[7]	rw	0 nresetglobal	1= enable event #7
6	RegEvnEn[6]	rw	0 nresetglobal	1= enable event #6
5	RegEvnEn[5]	rw	0 nresetglobal	1= enable event #5
4	RegEvnEn[4]	rw	0 nresetglobal	1= enable event #4
3	RegEvnEn[3]	rw	0 nresetglobal	1= enable event #3
2	RegEvnEn[2]	rw	0 nresetglobal	1= enable event #2
1	RegEvnEn[1]	rw	0 nresetglobal	1= enable event #1
0	RegEvnEn[0]	rw	0 nresetglobal	1= enable event #0

Table 9-3: **RegEvnEn**

pos.	RegEvnPriority	rw	reset	function
7-0	RegEvnPriority	r	11111111 nresetglobal	code of highest event set FF if no event present.

Table 9-4: **RegEvnPriority**

pos.	RegEvnEvn	rw	reset	function
7-2	-	r	00000	unused
1	EvnHig	r	0 nresetglobal	one or more high priority event is set
0	EvnLow	r	0 nresetglobal	one or more low priority event is set

Table 9-5: **RegEvnEvn**

9.4 Detailed description

The CoolRISC core has 2 different event levels EV0 and EV1 (Figure 9-1).

The setting and clearing of these events can be done in the stat register (see chapter describing the CPU).

The event handler bundles a certain number of event sources and routes them to one of these two events and provides the possibility to enable/disable each of them individually. The definition of the event sources is given in the memory mapping chapter.

RegEvn is an 8-bit register containing flags for the event sources. Those flags are set when the event is enabled (i.e. if the corresponding bit in the registers **RegEvnEn** is set) and a rising edge is detected on the corresponding event source.

Once memorized, writing a '1' in the corresponding bit of **RegEvn** clears an event flag. Writing a '0' does not modify the flag. All interrupts are automatically cleared after a reset.

Two registers are provided to facilitate the writing of event service software. **RegEvnPriority** contains the number of the highest event set (its value is 0xFF when no event is memorized). **RegEvnEvn** indicates the priority level of the current events.

All event sources are sampled by the highest frequency in the system. A CPU event is generated and memorized when an event becomes high. The 8 event sources are divided into 2 levels of priority: High (4 event sources) and Low (4 event sources). Those 2 levels of priority are directly mapped to those supported by the CoolRisc (EV0 and EV1; see CoolRisc documentation for more information).

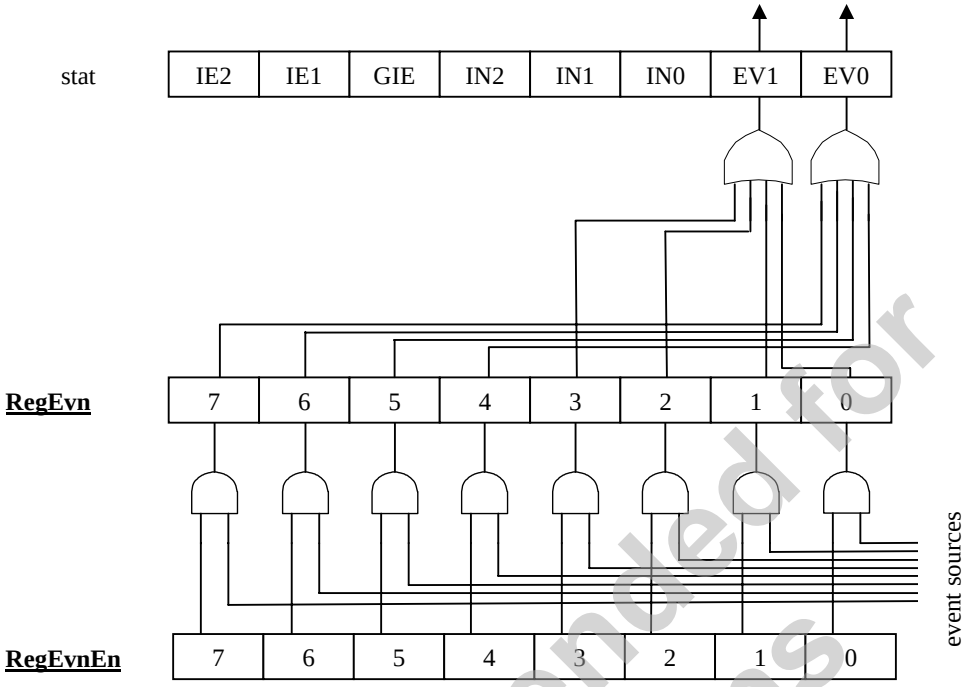


Figure 9-1. Event handler principle.

10. Low Power RAM

10.1 Features

10.1.1 Overview

In order to save power consumption, 8 8-bit registers are provided in page 0. These memory locations should be reserved for often-updated variables. As they are real registers and not RAM, power consumption is greatly reduced.

10.2 Register map

pos.	Reg00	rw	reset	function
7-0	Reg00	rw	0	low-power data memory
7-0	Reg01	rw	0	low-power data memory
7-0	Reg02	rw	0	low-power data memory
7-0	Reg03	rw	0	low-power data memory
7-0	Reg04	rw	0	low-power data memory
7-0	Reg05	rw	0	low-power data memory
7-0	Reg06	rw	0	low-power data memory
7-0	Reg07	rw	0	low-power data memory

Table 10-1: Low Power RAM

11. Port A

11.1 Features

- input port, 8 bits wide
- each bit can be set individually for debounced or direct input
- each bit can be set individually for pullup or not
- snap-to-rail option for each input
- each bit is an interrupt request source on the rising or falling edge
- a system reset can be generated on an input pattern
- PA[0] and PA[1] can generate two events for the CPU, individually maskable
- PA[0] to PA[3] can be used as clock inputs for the counters/timers/PWM (product dependent)

11.2 Overview

PortA is a general purpose 8 bit wide digital input port, with interrupt capability. Figure 11-1 shows its structure.

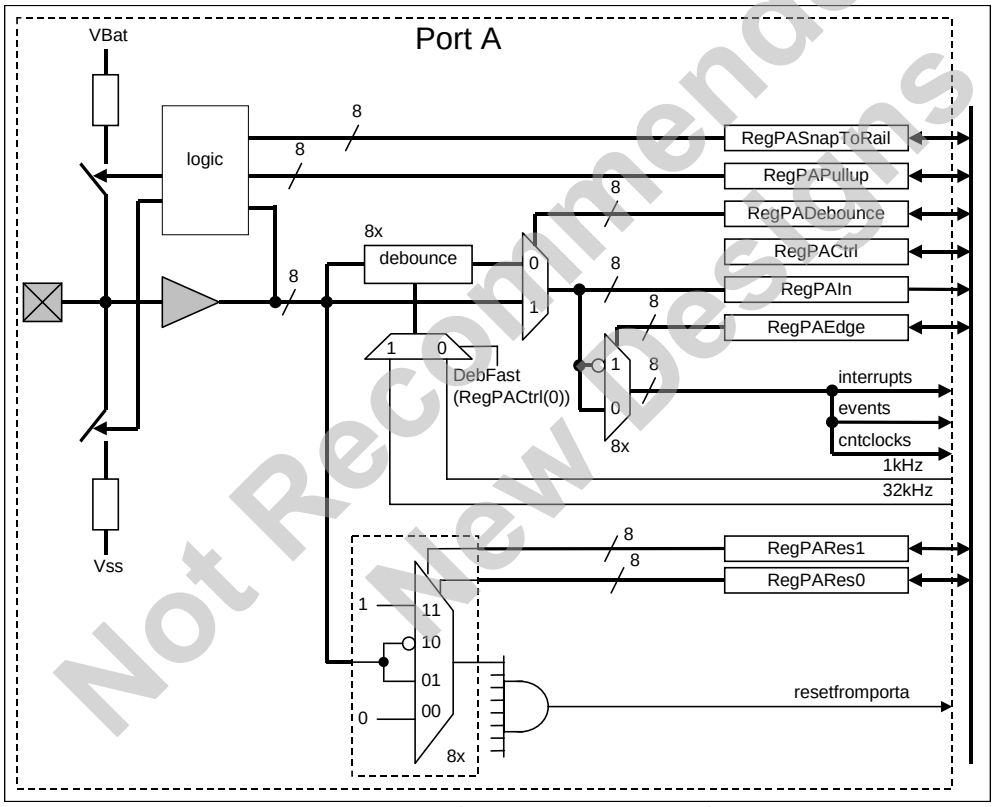


Figure 11-1: structure of Port A

11.3 Register map

There are eight registers in Port A (PA), namely **RegPAIn**, **RegPADebounce**, **RegPAPullup**, **RegPAEdge**, **RegPARes0**, **RegPARes1**, **RegPACtrl** and **RegPASnaptorail**. Table 11-2 to Table 11-9 show the mapping of control bits and functionality of these registers

register name
RegPAIn
RegPADebounce
RegPAEdge
RegPAPullup
RegPARes0
RegPARes1
RegPACtrl
RegPASnaptorail

Table 11-1: PA registers

pos.	RegPAIn	rw	reset	description
7:0	PAIn[7:0]	r	x	pad PA[7] to PA[0] input value

Table 11-2: RegPAIn

pos.	RegPADebounce	rw	reset	description
7:0	PADebounce[7:0]	r w	0 nresetpconf	PA[7] to PA[0] 1: debounce enabled 0: debounce disabled

Table 11-3: RegPADebounce

pos.	RegPAEdge	rw	reset	description
7:0	PAEdge[7:0]	r w	0 nresetglobal	PA[7] to PA[0] edge configuration 0: positive edge 1: negative edge

Table 11-4: RegPAEdge

pos.	RegPAPullup	rw	reset	description
7:0	PAPullup[7:0]	r w	1 nresetpconf	PA[7] to PA[0] pullup enable 0: pullup disabled 1: pullup enabled

Table 11-5: RegPAPullup

pos.	RegPARes0	rw	reset	description
7:0	PARes0[7:0]	r w	0 nresetglobal	PA[7] to PA[0] reset configuration

Table 11-6: RegPARes0

pos.	RegPARes1	rw	reset	description
7:0	PARes1[7:0]	r w	0 nresetglobal	PA[7] to PA[0] reset configuration

Table 11-7: RegPARes

pos.	RegPACtrl	rw	reset	description
7:1	[7:1]	r	0000000	Unused
0	DebFast	r w	0 nresetpconf	0 = slow debounce, 1 = fastdebounce

Table 11-8: RegPACtrl

pos.	RegPASnaptorail	rw	reset	description
7:0	PASnaptorail[7:0]	rw	0 nresetpconf	set snap-to-rail input on

Table 11-9: RegPASnaptorail

Note*:* Depending on the status of the **EnResetPConf** bit in **RegSysCtrl**, **RegPAEdge**, **RegPADebounce** and **RegPACtrl** can be reset by any of the possible system resets or only with power-on reset and NRESET pad.

11.4 Interrupts and events map

Interrupt source	Default mapping in the interrupt manager	Default mapping in the event manager
pa_irqbus[5]	RegIrqMid[5]	
pa_irqbus[4]	RegIrqMid[4]	
pa_irqbus[1]	RegIrqMid[1]	RegEvn[4]
pa_irqbus[0]	RegIrqMid[0]	RegEvn[0]
pa_irqbus[7]	RegIrqLow[7]	
pa_irqbus[6]	RegIrqLow[6]	
pa_irqbus[3]	RegIrqLow[3]	
pa_irqbus[2]	RegIrqLow[2]	

11.5 Port A (PA) Operation

The Port A input status (debounced or not) can be read from **RegPAin**.

Debounce mode:
Each bit in Port A can be individually debounced by setting the corresponding bit in **RegPADebounce**. After reset, the debounce function is disabled. After enabling the debouncer, the change of the input value is accepted only if height consecutive samples are identical. Selection of the clock is done by bit **DebFast** in Register **RegPACtrl**.

DebFast	Clock filter
0	1kHz
1	32kHz

Table 10: debounce frequency selection

Note: The tolerance on the debounce frequency depends on the selected clock source. When the external clock is used, the pulse width will be correct if the input of the low prescaler is set to a frequency close to 32kHz (see clock block documentation).

Pullups/Snap-to-rail:
Different functions are possible depending on the value of the registers **RegPAPullup** and **RegPASnaptorail**. When the corresponding bit in **RegPAPullup** is set to 0, the inputs are floating (pullup and pulldown resistors are disconnected). When the corresponding bit in **RegPAPullup** is 1 and in **RegPASnaptorail** is 0, a pullup resistor is connected to the input pin. Finally, when the corresponding bit in **RegPAPullup** is 1 and in **RegPASnaptorail** is 1, the snap-to-rail function is active.

The snap-to-rail function connects a pullup or pulldown resistor to the input pin depending on the value forced on the input pin. This function can be used for instance when the input port is connected to a tristate bus. When the bus is floating, the pullup or pulldown maintains the bus in the last low impedance state before it became floating until another low impedance output drives the bus. It also reduces the power consumption with respect to a classic pullup since it selects the pullup or pulldown resistor so that it confirms the detected input state.

The state of input pin is summarized in the table below.

PAPullup[x]	PASnaptorail[x]	(last) externally forced PA[x] value	PA[x] pull
0	x	x	floating
1	0	x	pullup
1	1	0	*pulldown*
1	1	1	*pullup*

Table 11: Snap-to-rail

Port A starts up with the pullup resistor connected and the snap-to-rail function disabled.

Port A as an interrupt source:
Each Port A input is an interrupt request source and can be set on rising or falling edge with the corresponding bit in **RegPAEdge**. After reset, the rising edge is selected for interrupt generation by default. The interrupt source can be debounced by setting register **RegPADebounce**. The interrupt signals are sampled on the fastest clock in the circuit. In order to guarantee that the circuit detects the interrupt, the minimal pulse length should be 1 cycle of this clock.

Note: care must be taken when modifying **RegPAEdge** because this register performs an edge selection. The change of this register may result in a transition, which may be interpreted as a valid interruption.

Port A as an event source:
The interrupt signals of the pins PA[0] and PA[1] are also available as events on the event controller.

Port A as a clock source (product dependent):
Images of the PA[0] to PA[3] input ports (debounced or not) are available as clock sources for the counter/timer/PWM peripheral.

Port A as a reset source:
Port A can be used to generate a system reset by placing a predetermined word on Port A externally. The reset is built using a logical and of the 8 PARes[x] signals:

resetfromportA = PAReset[7] AND PAReset[6] AND PAReset[5] AND ... AND PAReset[0]

PAReset[x] is itself a logical function of the corresponding pin PA[x]. One of four logical functions can be selected for each pin by writing into two registers **RegPARes0** and **RegPARes1** as shown in Table 11-12.

PARes1[x]	PARes0[x]	PAReset[x]
0	0	0
0	1	PA[x]
1	0	not(PA[x])
1	1	1

Table 11-12: Selection bits for reset signal

A reset from Port A can be inhibited by placing a 0 on both **PARes1[x]** and **PARes0[x]** for at least 1 pin. Setting both **PARes1[x]** and **PARes0[x]** to 1, makes the reset independent of the value on the corresponding pin. Setting both registers to hFF, will reset the circuit independent from the Port A input value. This makes it possible to do a reset by software.

Note: depending of the value of PA[0] to PA[7], changes to **RegPARes0** and **RegPARes1** can cause a reset. Therefore it is safe to have always one (RegPARes0[x], RegPARes1[x]) equal to '00' during the setting operations.

11.6 Port A electrical specification

sym	description	min	typ	max	unit	Comments
V_{INH}	Input high voltage	0.7*VBAT		VBAT	V	VBAT≥2.4V
V_{INL}	Input low voltage	VSS		0.2*VBAT	V	VBAT≥2.4V
R_{PU}	Pull-up resistance	20	50	80	kΩ	
Cin	Input capacitance		2.5		pF	Note 1

Note 1: this value is indicative only since it depends on the package.

Table 11-13. Electrical specification

12. Port B

12.1 Features

- Input / output / analog port, 8 bits wide
- Each bit can be set individually for input or output
- Each bit can be set individually for open-drain or push-pull
- Each bit can be set individually for pull-up or not (for input or open-drain mode)
- In open-drain mode, pull-up is not active when corresponding pad is set to zero
- The 8 pads can be connected individually to four internal analog lines (4 line analog bus)
- Two internal freq. (16 kHz and cpuck) can be output on PB[2] and PB[3]

Product dependant:

- Two PWM signal can be output on pads PB[0] and PB[1]
- The synchronous serial interface (USRT) uses pads PB[5], PB[4]
- The UART interface uses pads PB[6] and PB[7] for Tx and Rx

12.2 Overview

Port B is a multi-purpose 8 bit Input/output port. In addition to digital behavior, all pins can be used for analog signals. Each port terminal can be individually selected as digital input or output or as analog for sharing one of four possible analog lines.

12.3 Register map

Table 12-1 shows the Port B registers.

register name
RegPBOut
RegPBIn
RegPBDir
RegPBOpen
RegPBPullup
RegPBAna

Table 12-1: Port B registers

pos.	RegPBOut	rw	reset	description in digital mode	description in analog mode
7 – 0	PBOut[7-0]	r w	0 nresetpconf	Pad PB[7-0] output value	Analog bus selection for pad PB[7-0]

Table 12-2: RegPBOut

pos.	RegPBIn	rw	reset	description in digital mode	description in analog mode
7 – 0	PBIn[7-0]	r w	X	Pad PB[7-0] input status	Unused

Table 12-3: RegPBIn

pos.	RegPBDir	rw	reset	description in digital mode	description in analog mode
7 – 0	PBDir [7-0]	r w	0 nresetpconf	Pad PB[7-0] direction (0=input)	Analog bus selection for pad PB[7-0]

Table 12-4**:** RegPBDir

pos.	RegPBOpen	rw	reset	description in digital mode	description in analog mode
7 – 0	PBOpen[7-0]	r w	0 nresetpconf	Pad PB[7-0] open drain (1 = open drain)	Unused

Table 12-5: RegPBOpen

pos.	RegPBPullup	rw	reset	description in digital mode	description in analog mode
7 –0	PBPullup[7]	r w	1 nresetpconf	Pull-up for pad PB[7-0] (1=active)	Connect pad PB[7-0] on selected ana bus

Table 12-6: RegPBPullup

pos.	RegPBAna	rw	reset	description in digital mode	description in analog mode
7 – 0	PBAna [7-0]	r w	0 nresetpconf	Set PB[7-0] in analog mode	Set PB[7-0] in analog mode

Table 12-7: RegPBAna

Note: Depending on the status of the **EnResPConf** bit in **RegSysCtrl**, the reset conditions of the registers are different. See the reset block documentation for more details on the nresetpconf signal.

12.4 Port B capabilities

Port B	utilization (priority)		
name	high (analog)	medium (functions)	low (digital) (default)
PB[7]	analog	uart Rx	I/O (with pull-up)
PB[6]	analog	uart Tx	I/O (with pull-up)
PB[5]	analog	usrt S1	I/O (with pull-up)
PB[4]	analog	usrt S0	I/O (with pull-up)
PB[3]	analog	16 kHz	I/O (with pull-up)
PB[2]	analog	clock CPU	I/O (with pull-up)
PB[1]	analog	PWM1 Counter C (C+D)	I/O (with pull-up)
PB[0]	analog	PWM0 Counter A (A+B)	I/O (with pull-up)

Table 12-8: Different Port B functions

Table 12-8 shows the different usages that can be made of the port B with the order of priority. If a pin is selected to be analog, it overwrites the function and digital set-up. If the pin is not selected as analog, but a function is enabled, it overwrites the digital set-up. If neither the analog nor function is selected for a pin, it is used as an ordinary digital I/O. This is the default configuration at start-up.

Note: the presence of the functions is product dependent.

12.5 Port B analog capability

12.5.1 Port B analog configuration

Port B terminals can be attached to a 4 line analog bus by setting the **PBAna[x]** bits to 1 in the **RegPBAna** register.

The other registers then define the connection of these 4 analog lines to the different pads of Port B. These can be used to implement a simple LCD driver or A/D converter. Analog switching is available only when the circuit is

powered with sufficient voltage (see specification below). Below the specified supply voltage, only voltages that are close to VSS or VBAT can be switched.

When **PBAna[x]** is set to 1, one pad of the Port B terminals is changed from digital I/O mode to analog. The usage of the registers **RegPBPullup**, **RegPBOut** and **RegPBDir** define the analog configuration (see Table 12-9).

When **PBAna[x]** = 1**,** then **PBPullup[x]** connects the pin to the analog bus. **PBDir[x]** and **PBPOut[x]** select which of the 4 analog lines is used.

analog bus selection		PBPullup[x]	PB[x] selection on
PBDir[x]	PBout[x]		
0	0	1	analog line 0
0	1	1	analog line 1
1	0	1	analog line 2
1	1	1	analog line 3
X	X	0	High impedance

Table 12-9: Selection of the analog lines with **RegPBDir**, **RegPBout** and **RegPBPullup** when **PBAna[x]** = 1

Example:

Set the pads PB[2] and PB[5] on the analog line 3. (the values X depend on the configuration of others pads)

- apply high impedance in the analog mode (move RegPBPullup,#0bXX0XX0XX)
- go to analog mode (move RegPBAna,#0bXX1XX1XX)
- select the analog line3 (move RegPBDir,#0bXX1XX1XX and move RegPBOut,#0bXX1XX1XX)
- apply the analog line to the output (move RegPBPullup,#0bXX1XX1XX)

12.5.2 Port B analog function specification

The table below defines the on-resistance of the switches between the pin and the analog bus for different conditions. The series resistance between 2 pins of Port B connected to the same analog line is twice the resistance given in the table.

sym	description	min	typ	max	unit	Comments
Ron	switch resistance			11	kΩ	Note 1
Ron	switch resistance			15	kΩ	Note 2
Cin	input capacitance (off)		3.5		pF	Note 3
Cin	input capacitance (on)		4.5		pF	Note 4

Table 12-10. Analog input specifications.

Note 1: This is the series resistance between the pad and the analog line in 2 cases
1. VBAT ≥ 2.4V and the VMULT peripheral is present on the circuit and enabled.
2. VBAT ≥ 3.0V and the VMULT peripheral is not present on the circuit.

Note 2: This is the series resistance in case VBAT ≥ 2.8V and the peripheral VMULT is not present on the circuit.
Note 3: This is the input capacitance seen on the pin when the pin is not connected to an analog line. This value is indicative only since it is product and package dependent.
Note 4: This is the input capacitance seen on the pin when the pin is connected to an analog line and no other pin is connected to the same analog line. This value is indicative only since it is product and package dependent.

12.6 Port B function capability

The Port B can be used for different functions implemented by other peripherals. The description below is applicable only in so far the circuit contains these peripherals.

When the counters are used to implement a PWM function (see the documentation of the counters), the PB[0] and PB[1] terminals are used as outputs (PB[0] is used if **CntPWM0** in **RegCntConfig1** is set to 1, PB[1] is used if **CntPWM1** in **RegCntConfig1** is set to 1) and the PWM generated values override the values written in **RegPBout**. However, **PBDir(0)** and **PBDir(1)** are not automatically overwritten and have to be set to 1.

If **Output16k** is set in **RegSysMisc**, the frequency is output on PB[3]. This overrides the value contained in **PBOut(3)**. However, **PBDir(3)** must be set to 1. The frequency and duty cycle of the clock signal are given in Figure 12-1. f_{max} is the frequency of fastest clock present in the circuit.

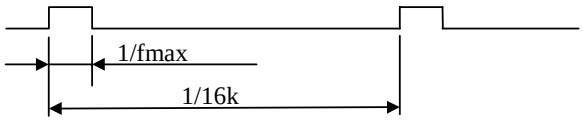


Figure 12-1. 16 kHz output clock timing

Similarly, if **OutputCkCpu** is set in **RegSysMisc**, the CPU frequency is output on PB[2]. This overrides the value contained in **PBOut(2)**. However, **PBDir(2)** must be set to 1.

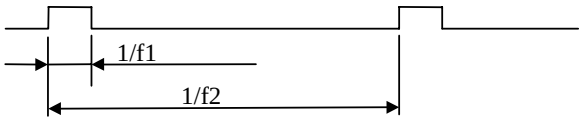


Figure 12-2. CPU output clock timing.

The timing of the CPU clock (Figure 12-2) depends on the selection of the **CpuSel** bit in the **RegSysClock** register and is given in Table 12-11. f_{max} is the frequency of fastest clock present in the circuit. Note that the tolerance on the 32 kHz depends on the selected clock source (see clock block documentation).

CpuSel	f1	f2
0	$f_{max}/4$	f_{max}
1	f_{max}	32 kHz

Table 12-11. CPU clock timing parameters.

Pins PB[5] and PB[4] can be used for S1 and S0 of the USRT (see USRT documentation) when the **UsrtEnable** bit is set in **RegUsrtCtrl**. The PB[5] and PB[4] then become open-drain. This overrides the values contained in **PBOpen(5:4)**, **PBOut(5:4)** and **PBDir(5:4)**. If there is no external pull-up resistor on these pins, internal pull-ups should be selected by setting **PBPullup(5:4)**. When S0 is an output, the pin PB[4] takes the value of **UsrtS0** in **RegUsrtS0**. When S1 is an output, the pin PB[5] takes the value of **UsrtS1** in **RegUsrtS1**.

Pins PB[6] and PB[7] can be used by the UART (see UART documentation). When **UartEnTx** in **RegUartCtrl** is set to 1, PB[6] is used as output signal Tx. When **UartEnRx** in **RegUartCtrl** is set to 1, PB[7] is used as input signal Rx. This overrides the values contained in **PBOut(7:6)** and **PBDir(7:6)**.

12.7 Port B digital capabilities

12.7.1 Port B digital configuration

The direction of each bit within Port B (input only or input/output) can be individually set using the **RegPBDir** register. If **PBDir[x]** = 1, both the input and output buffer are active on the corresponding Port B. If **PBDir[x]** = 0, the corresponding Port B pin is an input only and the output buffer is in high impedance. After reset (nresetpconf) Port B is in input only mode **(PBDir[x]** are reset to 0).

The input values of Port B are available in **RegPBIn** (read only). Reading is always direct - there is no debounce function in Port B. In case of possible noise on input signals, a software debouncer with polling or an external hardware filter have to be realized. The input buffer is also active when the port is defined as output and the effective value on the pin can be read back.

Data stored in **RegPBOut** are outputted at Port B if **PBDir[x]** is 1. The default values after reset is low (0).

When a pin is in output mode (**PBDir[x]** is set to 1), the output can be a conventional CMOS (Push-Pull) or a N-channel Open-drain, driving the output only low. By default, after reset (nresetpconf) the **PBOpen[x]** in **RegPBOpen** is cleared to 0 (push-pull). If **PBOpen[x]** in **RegPBOpen** is set to 1 then the internal P transistor in the output buffer is electrically removed and the output can only be driven low (**PBOut[x]**=0). When **PBOut[x]**=1, the pin is high Impedance. The internal pull-up or an external pull-up resistor can be used to drive to pin high.

Note: Because the P transistor actually exists (this is not a real Open-drain output) the pull-up range is limited to VDD + 0.2V (avoid forward bias the P transistor / diode).

Each bit can be set individually for pull-up or not using register **RegPBPullup**. Input is pulled up when its corresponding bit in this register is set to 1. Default status after (nresetpconf) is 1, which means with pull up. To

limit power consumption, pull-up resistors are only enabled when the associated pin is either a digital input or an N-channel open-drain output with the pad set to 1. In the other cases (push-pull output or open-drain output driven low), the pull up resistors are disabled independent of the value in **RegPBPullup**.

After power-on reset, the Port B is configured as an input port with pull-up. During power-on reset (see reset block documentation) however, the pin PB[1] is pulled down in stead of pulled up. Once the power-on reset completed, the pin PB[1] is pulled up, exactly as the other Port B pins.

The input buffer is always active, except in analog mode. This means that the Port B input should be a valid digital value at all times unless the pin is set in analog mode. Violating this rule may lead to high power consumption.

12.7.2 Port B digital function specification

sym	description	min	typ	max	unit	Comments
V_{INH}	Input high voltage	0.7*VBAT		VBAT	V	VBAT≥2.4V
V_{INL}	Input low voltage	VSS		0.2*VBAT	V	VBAT≥2.4V
V_{OH}	Output high voltage	VBAT-0.4		VBAT	V	VBAT=1.2V, I_{OH} =0.3mA VBAT=2.4V, I_{OH} =5.0mA VBAT=4.5V, I_{OH} =8.0mA
V_{OL}	Output low voltage	VSS		VSS+0.4	V	VBAT=1.2V, I_{OL} =0.3mA VBAT=2.4V, I_{OL} =12.0mA VBAT=4.5V, I_{OL} =15.0mA
R_{PU}	Pull-up resistance	20	50	80	kΩ	
Cin	Input capacitance		3.5		pF	Note 1

Note 1: this value is indicative only since it depends on the package.

12.8 Low power comparators

If the low power comparator (CMPD) peripheral is present in the circuit, the signals on the pins PB[7:4] can be used as inputs for these low power comparators. Although the comparators are functional independent of the Port B configuration, it is recommended to set the pins that are used for the CMPD in analog mode without selecting any analog lines. This is to avoid high power consumption in the digital input buffer when analog or slowly varying digital signals are applied.

13. Port D

13.1 Features

- input / output port, 8 bits wide
- each bit can be set individually for input or output
- pull-ups are available in input mode
- snap-to-rail option in input mode

13.2 Overview

Port D (PD) is a general purpose 8 bit input/output digital port.
Figure 13-1 shows its structure.

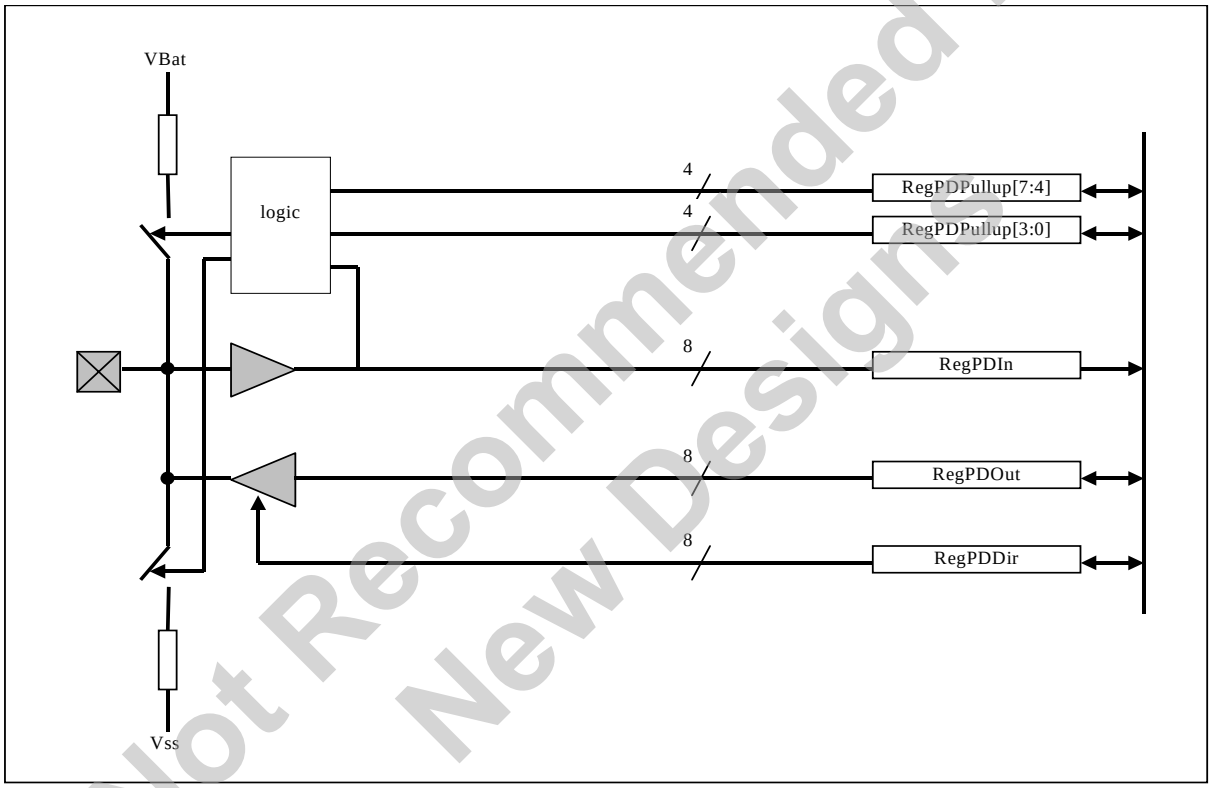


Figure 13-1 : structure of PortD

13.3 Register map

There are four registers in the Port D (PD), namely **RegPDIn**, **RegPDOut**, **RegPDDir** and **RegPDPullup**. Table 13-3 to Table 13-6 show the mapping of control bits and functionality of these registers.

register name
RegPDIn
RegPDOut
RegPDDir
RegPDPullup

Table 13-1 : PD registers

Pos.	RegPDIn	Rw	Reset	Description
7:0	PDIn[7:0]	r	-	pad PD[7:0] input value

Table 13-2 : RegPDIn

Pos.	RegPDOut	Rw	Reset	Description
7:0	PDOut[7:0]	r w	0 nresetpconf	pad PD[7:0] output value

Table 13-3 : RegPDOut

Pos.	RegPDDir	Rw	Reset	Description
7:0	PDDir[7:0]	r w	0 nresetpconf	pad PD[7:0] direction (0=input)

Table 13-4 : RegPDDir

Pos.	RegPDPullup	Rw	Reset	Description
7	PDSnapToRail[3]	r w	1 nresetpconf	snap-to-rail for pad PD[7] and PD[6] (1=active)
6	PDSnapToRail[2]	r w	1 nresetpconf	snap-to-rail for pad PD[5] and PD[4] (1=active)
5	PDSnapToRail[1]	r w	1 nresetpconf	snap-to-rail for pad PD[3] and PD[2] (1=active)
4	PDSnapToRail[0]	r w	1 nresetpconf	snap-to-rail for pad PD[1] and PD[0] (1=active)
3	PDPullup[3]	r w	1 nresetpconf	pullup for pad PD[7] and PD[6] (1=active)
2	PDPullup[2]	r w	1 nresetpconf	pullup for pad PD[5] and PD[4] (1=active)
1	PDPullup[1]	r w	1 nresetpconf	pullup for pad PD[3] and PD[2] (1=active)
0	PDPullup[0]	r w	1 nresetpconf	pullup for pad PD[1] and PD[0] (1=active)

Table 13-5 : RegPDPullup

13.4 Port D (PD) Operation

The direction of each pin of Port D (input or input/output) can be individually set by using the **RegPDDir** register. If **PDDir[x]** = 1, the output buffer on the corresponding Port D pin is enabled. After reset, Port D is in input only mode (**PDDir[x]** are reset to 0). The input buffer is always enabled independently from the **RegPDDir** contents.

Output data:
Data are stored in **RegPDOut** prior to output at Port D.

Input data:
The status of Port D is available in **RegPDIn** (read only). Reading is always direct - there is no digital debounce function associated with Port D. In case of possible noise on input signals, a software debouncer or an external filter must be realised.

Pull-up/Snap to Rail:
When configured as an input (**PDDir[x]**=0), pull-ups are available on every pin. The pull-up function of the pins is controlled two by two by the **PDPullup** and **PDSnapToRail** bits in the register **RegPDPullup**. When a bit **PDPullup[x]** is 0, the pull-ups on the pins PD[2x] and PD[2x+1] are disabled. When a bit **PDPullup[x]** is set to 1 and the bit **PDSnapToRail[x]** is set to 0, the pull-up resistor is connected to the pins PD[2x] and PD[2x+1]. When both **PDPullup[x]** and **PDSnapToRail[x]** are 1, the snap-to-rail function is active on the pins PD[2x] and PD[2x+1].

The snap-to-rail function connects a pullup or pulldown resistor to the input pin depending on the value forced on the input pin. This function can be used for instance when the input port is connected to a tristate bus. When the bus is floating, the pullup or pulldown maintains the bus in the last low impedance state before it became floating until another low impedance output is driving the bus. It also reduces the power consumption with respect to a classic pullup since it selects the pullup or pulldown resistor so that it confirms the detected input state.

The function is summarised in the table below as a function of the different register settings.

PDDir[2x(+1)]	PDPullup[x]	PDSnapToRail[x]	(last) externally forced PD[2x(+1)] value	PD[2x(+1)] pull resistor
1	x	x	x	not connected
0	0	x	x	not connected
0	1	0	x	pullup
0	1	1	0	pulldown
0	1	1	1	pullup

Table 13-6: Snap-to-rail and pullup function

At power-on reset, Port D is configured as an input port with all pull-ups active.

13.5 Port D electrical specification

sym	description	min	typ	max	unit	Comments
V_{INH}	Input high voltage	0.7*VBAT		VBAT	V	VBAT≥2.4V
V_{INL}	Input low voltage	VSS		0.2*VBAT	V	VBAT≥2.4V
V_{OH}	Output high voltage	VBAT-0.4		VBAT	V	VBAT=1.2V, I_{OH} =0.3mA VBAT=2.4V, I_{OH} =5.0mA VBAT=4.5V, I_{OH} =8.0mA
V_{OL}	Output low voltage	VSS		VSS+0.4	V	VBAT=1.2V, I_{OL} =0.3mA VBAT=2.4V, I_{OL} =12.0mA VBAT=4.5V, I_{OL} =15.0mA
R_{PU}	Pull-up resistance	20	50	80	kΩ	
Cin	Input capacitance		3.0		pF	Note 1

Note 1: this value is indicative only since it depends on the package.

Table 13-7. Port D electrical specification

14. Universal Asynchronous Receiver/Transmitter (UART)

14.1 Features

- Full duplex operation with buffered receiver and transmitter.
- Internal baudrate generator with 10 programmable baudrates (300 - 153600).
- 7 or 8 bits word length.
- Even, odd, or no-parity bit generation and detection
- 1 stop bit
- Error receive detection: Start, Parity, Frame and Overrun
- Receiver echo mode
- 2 interrupts (receive full and transmit empty)
- Enable receive and/or transmit
- Invert pad Rx and/or Tx

14.2 Overview

The Uart pins are PB[7], which is used as Rx - receive and PB[6] as Tx - transmit.

14.3 Registers map

register name
RegUartCtrl
RegUartCmd
RegUartTx
RegUartTxSta
RegUartRx
RegUartRxSta

Table 14-1: Uart registers

pos.	RegUartCmd	rw	reset	description
7	SelXtal	r/w	0 nresetglobal	Select input clock: 0 = RC/external, 1 = xtal
6	-	r	0	Unused
5-3	UartRcSel(2:0)	r/w	000 nresetglobal	RC prescaler selection
2	UartPM	r/w	0 nresetglobal	Select parity mode: 1 = odd, 0 = even
1	UartPE	r/w	0 nresetglobal	Enable parity: 1 = with parity, 0 = no parity
0	UartWL	r/w	1 nresetglobal	Select word length: 1 = 8 bits, 0 = 7 bits

Table 14-2: RegUartCmd

pos.	RegUartCtrl	rw	reset	description
7	UartEcho	r/w	0 nresetglobal	Enable echo mode: 1 = echo Rx->Tx, 0 = no echo
6	UartEnRx	r/w	0 nresetglobal	Enable uart reception
5	UartEnTx	r/w	0 nresetglobal	Enable uart transmission
4	UartXRx	r/w	0 nresetglobal	Invert pad Rx
3	UartXTx	r/w	0 nresetglobal	Invert pad Tx
2-0	UartBR(2:0)	r/w	101 nresetglobal	Select baud rate

Table 14-3: RegUartCtrl

pos.	RegUartTx	rw	reset	description
7-0	UartTx	r/w	00000000 nresetglobal	Data to be send

Table 14-4: RegUartTx

pos.	RegUartTxSta	rw	reset	description
7-2	-	r	000000	Unused
1	UartTxBusy	r	0 nresetglobal	Uart busy transmitting
0	UartTxFull	r	0 nresetglobal	**RegUartTx** full Set by writing to **RegUartTx** Cleared when transferring **RegUartTx** into internal shift register

Table 14-5: RegUartTxSta

pos.	RegUartRx	rw	reset	description
7-0	UartRx	r	00000000 nresetglobal	Received data

Table 14-6: RegUartRx

pos.	RegUartRxSta	rw	reset	description
7-6	-	r	00	Unused
5	UartRxSErr	r	0 nresetglobal	Start error
4	UartRxPErr	r	0 nresetglobal	Parity error
3	UartRxFErr	r	0 nresetglobal	Frame error
2	UartRxOErr	r/c	0 nresetglobal	Overrun error Cleared by writing **RegUartRxSta**
1	UartRxBusy	r	0 nresetglobal	Uart busy receiving
0	UartRxFull	r	0 nresetglobal	**RegUartRx** full Cleared by reading **RegUartRx**

Table 14-7: RegUartRxSta

14.4 Interrupts map

interrupt source	default mapping in the interrupt manager
Irq_uart_Tx	**IrqHig(1)**
Irq_uart_Rx	**IrqHig(0)**

Table 14-8: Interrupts map

14.5 Uart baud rate selection

In order to have correct baud rates, the Uart interface has to be fed with a stable and trimmed clock source. The clock source can be an external clock source, the RC oscillator or the crystal oscillator. The precision of the baud rate will depend on the precision of the selected clock source.

14.6 Uart on the RC oscillator or external clock source

To select the external clock or RC oscillator for the Uart, the bit **SelXtal** in **RegUartCmd** has to be 0. The choice between the RC oscillator and the external clock source is made with the bit **EnExtClock** in **RegSysClock**.

In order to obtain a correct baud rate, the RC oscillator or external clock frequency have to be set to one of the frequencies given in the table below. The precision of the obtained baud rate is directly proportional to the frequency deviation of the used clock source with respect to the values in the table below.

Frequency selection for correct Uart baud rates	
RC oscillator (Hz)	**External clock (Hz)**
2'457'600	4'915'200
1'228'800	2'457'600
614'400	1'228'800
307'200	614'400
153'600	307'200
76'800	153'600

Table 14-9a

For each of these frequencies, the baud rate can be selected with the bits **UartBR(2:0)** in **RegUartCtrl** and **UartRcSel(2:0)** in **RegUartCmd** as shown in Table 14-9.

RC frequency (Hz)		2457600	1228800	614400	307200	153600	76800
External clock freq. (Hz)		4915200	2457600	1228800	614400	307200	153600
UartRcSel	**UartBR**						
000	111	153600	76800	38400	19200	9600	4800
	110	76800	38400	19200	9600	4800	2400
	101	38400	19200	9600	4800	2400	1200
	100	19200	9600	4800	2400	1200	600
	011	9600	4800	2400	1200	600	300
	010	4800	2400	1200	600	300	-
	001	2400	1200	600	300	-	-
	000	1200	600	300	-	-	-
010	001	600	300	-	-	-	-
	000	300	-	-	-	-	-

Table 14-9: Uart baud rate with RC clock or external clock

Note 1 : Although not documented here, the coding of the baud rate used in the circuits XE8801, XE8803 and XE8805 can also be used.

Note 2 : The precision of the baud rate is directly proportional to the frequency deviation of the used clock from the ideal frequency given in the table. In order to increase the precision and stability of the RC oscillator, the DFLL (digital frequency locked loop) can be used with the crystal oscillator as a reference.

14.7 Uart on the crystal oscillator

In order to use the crystal oscillator as the clock source for the Uart, the bit **SelXtal** in **RegUartCmd** has to be set. The crystal oscillator has to be enabled by setting the **EnableXtal** bit in **RegSysClock**. The baud rate selection is done using the **UartBR** bits as shown in Table 14-10.

Xtal freq. (Hz)	32768
UartBR	
011	2400
010	1200
001	600
000	300

Table 14-10: Uart baud rate with Xtal clock

Due to the odd ratio between the crystal oscillator frequency and the baud rate, the generated baud rate has a systematic error of –2.48%.

14.8 Function description

14.8.1 Configuration bits

The configuration bits of the Uart serial interface can be found in the registers **RegUartCmd** and **RegUartCtrl**.

The bit **SelXtal** is used to select the clock source (see chapter 14.5). The bits **UartSelRc** and **UartBR** select the baud rate (see chapter 14.5).

The bits **UartEnRx** and **UartEnTx** are used to enable or disable the reception and transmission.

The word length (7 or 8 data bits) can be chosen with **UartWL**. A parity bit is added during transmission or checked during reception if **UartPE** is set. The parity mode (odd or even) can be chosen with **UartPM**.

Setting the bits **UartXRx** and **UartXTx** inverts the Rx respectively Tx signals.

The bit **UartEcho** is used to send the received data automatically back. The transmission function becomes then: Tx = Rx XOR **UartXRx** XOR **UartXTx**.

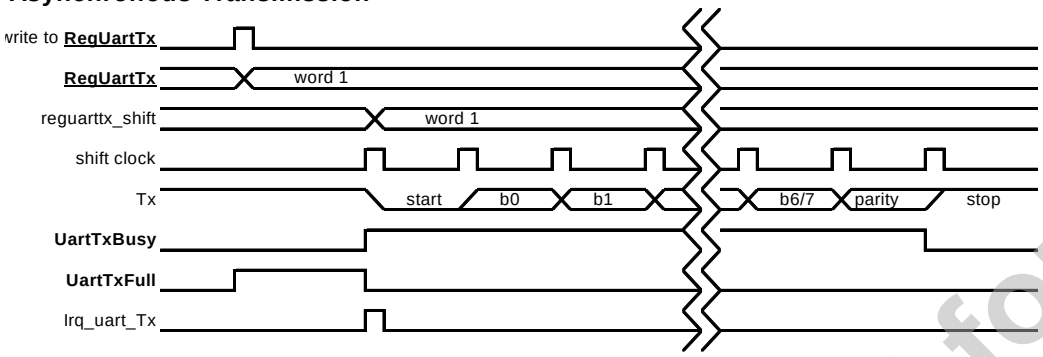
14.8.2 Transmission

In order to send data, the transmitter has to be enabled by setting the bit **UartEnTx**. Data to be sent have to be written to the register **RegUartTx**. The bit **UartTxFull** in **RegUartTxSta** then goes to 1, indicating to the transmitter that a new word is available. As soon as the transmitter has finished sending the previous word, it then loads the contents of the register **RegUartTx** to an internal shift register and clears the **UartTxFull** bit. An interrupt is generated on Irq_uart_Tx at the falling edge of the **UartTxFull** bit. The bit **UartTxBusy** in **RegUartTxSta** shows that the transmitter is busy transmitting a word.

A timing diagram is shown in Figure 14-1. Data is sent LSB first.

New data should be written to the register **RegUartTx** only while **UartTxBusy** is 0, otherwise data will be lost.

Asynchronous Transmission



Asynchronous Transmission (back to back)

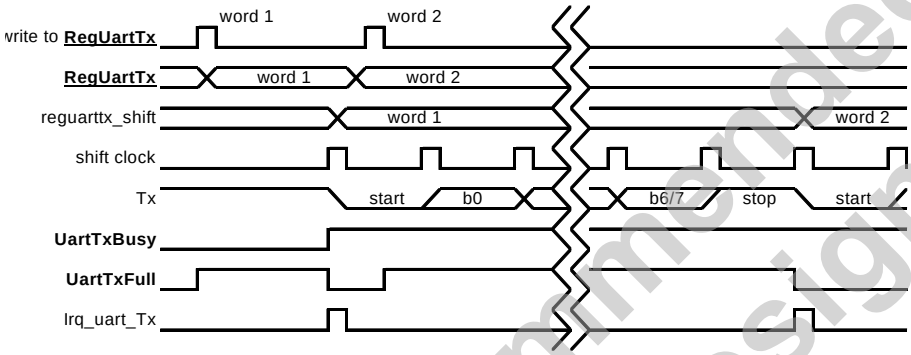


Figure 14-1. Uart transmission timing diagram.

14.8.3 Reception

On detection of the start bit, the **UartRxBusy** bit is set. On detection of the stop bit, the received data are transferred from the internal shift register to the register **RegUartRx**. At the same time, the **UartRxFull** bit is set and an interrupt is generated on Irq_uart_Rx. This indicates that new data is available in **RegUartRx**. The timing diagram is shown in Figure 14-2.

The **UartRxFull** bit is cleared when **RegUartRx** is read. If the register was not read before the receiver transfers a new word to it, the bit **UartRxOErr** (overflow error) is set and the previous contents of the register are lost. **UartRxOErr** is cleared by writing any data to **RegUartRxSta**.

The bit **UartRxSErr** is set if a start error has been detected. The bit is updated at data transfer to **RegUartRx**.

The bit **UartRxPErr** is set if a parity error has been detected, i.e. the received parity bit is not equal to the calculated parity of the received data. The bit is updated at data transfer to **RegUartRx**.

The bit **UartRxFErr** in **RegUartRxSta** shows that a frame error has been detected. No stop bit has been detected.

Asynchronous Reception

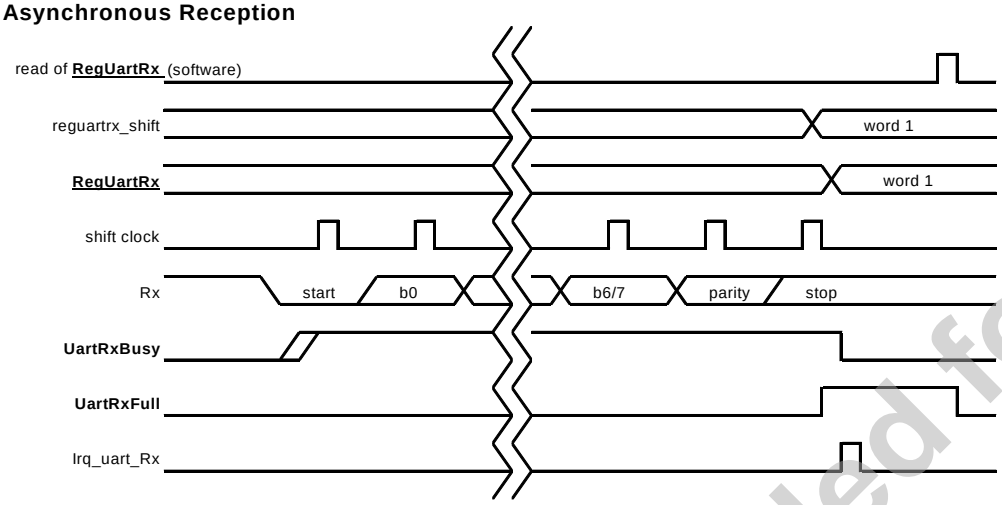


Figure 14-2. Uart reception timing diagram.

14.8.4 Interrupt or polling

The transmission and reception software can be driven by interruption or by polling the status bits.

Interrupt driven reception: each time an Irq_uart_Rx interrupt is generated, a new word is available in **RegUartRx**. The register has to be read before a new word is received.
Interrupt driven transmission: each time the contents of **RegUartTx** is transferred to the transmission shift register, an Irq_uart_Tx interrupt is generated. A new word can then be written to **RegUartTx**.

Reception driven by polling: the **UartRxFull** bit is to be read and checked. When it is 1, the **RegUartRx** register contains new data and has to be read before a new word is received.
Transmission driven by polling: the **UartTxFull** bit is to read and checked. When it is 0, the **RegUartTx** register is empty and a new word can be written to it.

14.9 Software hints

Example of program for a transmission with polling:

1. The **RegUartCmd** register and the **RegUartCtrl** register are initialized (for example: 8 bit word length, odd parity, 9600 baud, enable Uart transmission).
2. Write a byte to **RegUartTx**.
3. Wait untill the **UartTxFull** bit in **RegUartTxSta** register equals 0.
4. Jump to 2 to writing the next byte if the message is not finished.
5. End of transmission.

Example of program for a transmission with interrupt:

1. The **RegUartCmd** register and the **RegUartCtrl** register are initialized (for example: 8 bit word length, odd parity, 9600 baud, enable Uart transmission).
2. Write a byte to **RegUartTx**.
3. After an interrupt and if the message is not finished, jump to 2
4. End of transmission.

Example of program for a reception with polling:

1. The **RegUartCmd** register and the **RegUartCtrl** register are initialized (for example: 8 bit word length, odd parity, 9600 baud, enable Uart reception).
2. Wait until the **UartRxFull** bit in the **RegUartRxSta** register equals 1.
3. Read the **RegUartRxSta** and check if there is no error.
4. Read data in **RegUartRx**.
5. If data is not equal to End-Of-Line, then jump to 2.
6. End of reception.

Example of program for a reception with interrupt:

1. The **RegUartCmd** register and the **RegUartCtrl** register are initialized (for example: 8 bit word length, odd parity, 9600 baud, enable Uart reception).
2. When there is an interrupt, jump to 3
3. Read **RegUartRxSta** and check if there is no error.
4. Read data in **RegUartRx**.
5. If data is not equal to End-Of-Line, then jump to 2.
6. End of reception.

15. Universal Synchronous Receiver/Transmitter (URST)

15.1 Features

The USRT implements a hardware support for software implemented serial protocols:

- Control of two external lines S0 and S1 (read/write).
- Conditional edge detection generates interrupts.
- S0 rising edge detection.
- S1 value is stored on S0 rising edge.
- S0 signal can be forced to 0 after a falling edge on S0 for clock stretching in the low state.
- S0 signal can be stretched in the low state after a falling edge on S0 and after a S1 conditional detection.

15.2 Overview

The USRT block supports software universal synchronous receiver and transmitter mode interfaces.

External lines S0 and S1 respectively correspond to clock line and data line. S0 is mapped to PB[4] and S1 to PB[5] when the USRT block is enabled. It is independent of **RegPBdir** (Port B can be input or output). When USRT is enabled, the configurations in port B for PB[4] and PB[5] are overwritten by the USRT configuration. Internal pull-ups can be used by setting the **PBPullup[5:4]** bits.

Conditional edge detections are provided.

RegUsrtS1 can be used to read the S1 data line from PB[5] in receive mode or to drive the output S1 line PB[5] by writing it when in transmit mode. It is advised to read S1 data when in receive mode from the **RegUsrtBufferS1** register, which is the S1 value sampled on a rising edge of S0.

15.3 Register map

Register name
RegUsrtS1
RegUsrtS0
RegUsrtCtrl
RegUsrtCond1
RegUsrtCond2
RegUsrtBufferS1
RegUsrtEdgeS0

Table 15-1: USRT registers

Block configuration registers:

pos.	RegUsrtS1	rw	reset	function
7-1	“0000000”	r	-	Unused
0	UsrtS1	r/w	1 nresetglobal	Write: data S1 written to pad PB[5]), Read: value on PB[5] (not **UsrtS1** value).

Table 15-2: RegUsrtS1

pos.	RegUsrtS0	rw	reset	function
7-1	"0000000"	r	-	Unused
0	UsrtS0	r/w	1 nresetglobal	Write: clock S0 written to pad PB[4], Read: value on PB[4] (not **UsrtS0** value).

Table 15-3: RegUsrtS0

The values that are read in the registers **RegUsrtS1** and **RegUsrtS0** are not necessarily the same as the values that were written in the register. The read value is read back on the circuit pins not in the registers themselves. Since the outputs are open drain, an external circuit on the circuit pins may force a value different from the register value.

pos.	RegUsrtCtrl	rw	reset	function
7-4	"0000"	r	-	Unused
3	UsrtWaitS0	r	0 nresetglobal	Clock stretching flag (0=no stretching), cleared by writing **RegUsrtBufferS1**
2	UsrtEnWaitCond1	r/w	0 nresetglobal	Enable stretching on UsrtCond1 detection (0=disable)
1	UsrtEnWaitS0	r/w	0 nresetglobal	Enable stretching operation (0=disable)
0	UsrtEnable	r/w	0 nresetglobal	Enable USRT operation (0=disable)

Table 15-4: RegUsrtCtrl

pos.	RegUsrtCond1	rw	reset	function
7-1	"0000000"	r	-	Unused
0	UsrtCond1	r/c	0 nresetglobal	State of condition 1 detection (1 =detected), cleared when written.

Table 15-5: RegUsrtCond1

pos.	RegUsrtCond2	rw	reset	function
7-1	"0000000"	r	-	Unused
0	UsrtCond2	r/c	0 nresetglobal	State of condition 2 detection (1 =detected), cleared when written.

Table 15-6: RegUsrtCond2

pos.	RegUsrtBufferS1	rw	reset	function
7-1	"0000000"	r	-	Unused
0	UsrtBufferS1	r	0 nresetglobal	Value on S1 at last S0 rising edge.

Table 15-7: RegUsrtBufferS1

pos.	RegUsrtEdgeS0	rw	reset	function
7-1	"0000000"	r	-	Unused
0	UsrtEdgeS0	r	0 nresetglobal	State of rising edge detection on S0 (1=detected). Cleared by reading **RegUsrtBufferS1**

Table 15-8: RegUsrtEdgeS0

15.4 Interrupts map

interrupt source	default mapping in the interrupt manager
Irq_cond2	RegIrqMid(7)
Irq_cond1	RegIrqMid(6)

Table 15-9: Interrupts map

15.5 Conditional edge detection 1

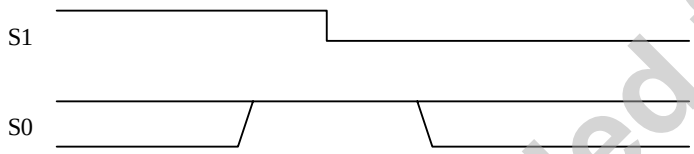


Figure 15-1: Condition 1

Condition 1 is satisfied when S0=1 at the falling edge of S1. The bit **UsrtCond1** in **RegUsrtCond1** is set when the condition 1 is detected and the USRT interface is enabled (**UsrtEnable**=1). Condition 1 is asserted for both modes (receiver and transmitter). The **UsrtCond1** bit is read only and is cleared by all reset conditions and by writing any data to its address.

Condition 1 occurrence also generates an interrupt on Irq_cond1.

15.6 Conditional edge detection 2

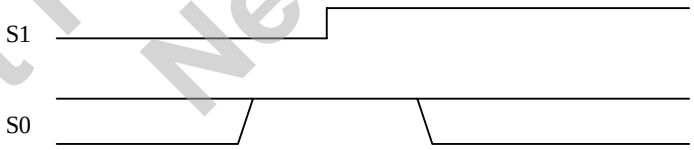


Figure 15-2: Condition 2

Condition 2 is satisfied when S0=1 at the rising edge of S1. The bit **UsrtCond2** in **RegUsrtCond2** is set when the condition 2 is detected and the USRT interface is enabled. Condition 2 is asserted for both modes (receiver and transmitter). The **UsrtCond2** bit is read only and is cleared by all reset conditions and by writing any data to its address.

Condition 2 occurrence also generates an interrupt on Irq_cond2.

15.7 Interrupts or polling

In receive mode, there are two possibilities to detect condition 1 or 2: the detection of the condition can generate an interrupt or the registers can be polled (reading and checking the **RegUsrtCond1** and **RegUsrtCond2** registers for the status of USRT communication).

15.8 Function description

The bit **UsrtEnable** in **RegUsrtCtrl** is used to enable the USRT interface and controls the PB[4] and PB[5] pins. This bit puts these two port B lines in the open drain configuration requested to use the USRT interface.

If no external pull-ups are added on PB[4] and PB[5], the user can activate internal pull-ups by setting **PBPullup[4]** and **PBPullup[5]** in **RegPBPullup**.

The bits **UsrtEnWaitS0**, **UsrtEnWaitCond1**, **UsrtWaitS0** in **RegUsrtCtrl** are used for transmitter/receiver control of USRT interface.

Figure 15-3 shows the unconditional clock stretching function which is enabled by setting **UsrtEnWaitS0.**

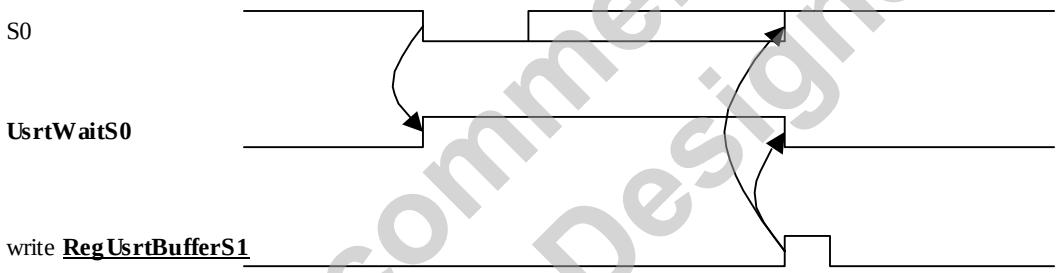


Figure 15-3: S0 Stretching (UsrtEnWaitS0=1)

When **UsrtEnWaitS0** is 1, the S0 line will be maintained at 0 after its falling edge (clock stretching). **UsrtWaitS0** is then set to 1, indicating that the S0 line is forced low. One can release S0 by writing to the **RegUsrtBufferS1** register.

The same can be done in combination with condition 1 detection by setting the **UsrtEnWaitCond1** bit. Figure 15-4 shows the conditional clock stretching function, which is enabled by setting **UsrtEnWaitCond1**.

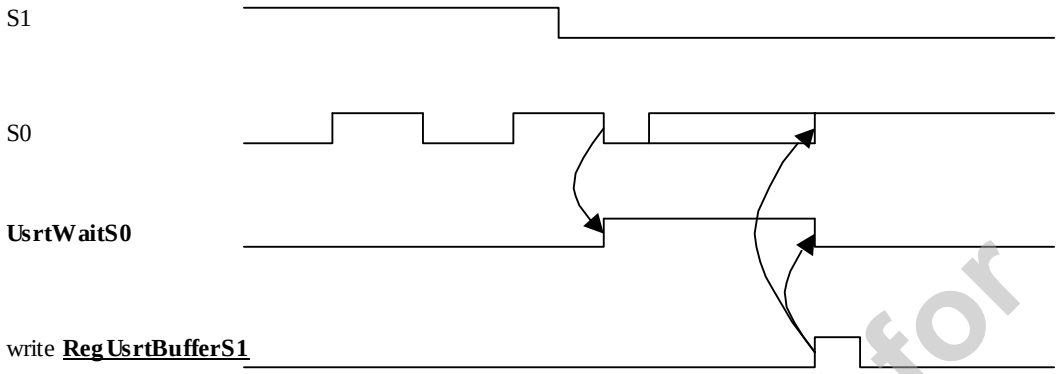


Figure 15-4: Conditional stretching (UsrtEnWaitCond1=1)

When **UsrtEnWaitCond1** is 1, the S0 signal will be stretched in its low state after its falling edge if the condition 1 has been detected before (**UsrtCond1**=1). **UsrtWaitS0** is then set to 1, indicating that the S0 line is forced low. One can release S0 by writing to the **RegUsrtBufferS1** register.

Figure 15-5 shows the sampling function implemented by the **UsrtBufferS1** bit. The bit **UsrtBufferS1** in **RegUsrtBufferS1** is the value of S1 sampled on PB[4] at the last rising edge of S0. The bit **UsrtEdgeS0** in **RegUsrtEdgeS0** is set to one on the same S0 rising edge and is cleared by a read operation of the **RegUsrtBufferS1** register. The bit therefor indicates that a new value is present in the **RegUsrtBufferS1** which was not yet read.

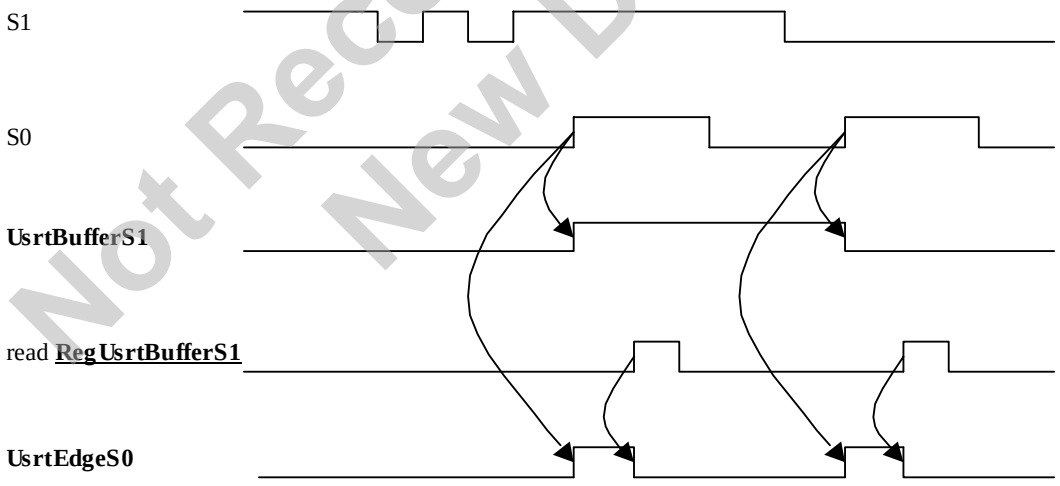


Figure 15-5: S1 sampling

16 Serial Peripheral Interface

16.1 Features

The SPI block implements the following functionality's:

- Full duplex operating mode.
- Master or slave configuration capability.
- Separate transmissions data, shift data and receive data registers in order to perform back-to-back transmissions.
- Four master mode frequencies available to generate serial clock.
- Serial clock with programmable polarity and phase.
- One enabled interrupt: SPI receive register full.
- Overflow detection flag.
- 4 I/O dedicated pads with 8mA drive and pull-up programmable.
- Multi-slave configuration capability.
- General purpose 4 bit wide digital input/output port mode.

16.2 Overview

The SPI can communicate with other external SPI devices. It provides flexibility to communicate with different SPI compatible circuits from several manufacturers (Serial EEPROMs, display drivers, A/D converters, audio device). Four dedicated input or output pads are attached to the SPI block: MISO, MOSI, SCK, NSS.

Six registers are used to run the SPI block. **RegSpiControl**, **RegSpiStatus**, **RegSpiDataOut**, **RegSpiDataIn**, **RegSpiPullup**, **RegSpiDir** are used to configure the communication settings, read the flags, write and read the exchanged data, and for the pad settings.

The SPI device can also be used as a 4 bit general purpose input/output port.

16.3 Register map

Register name
RegSpiControl
RegSpiStatus
RegSpiDataOut
RegSpiDataIn
RegSpiPullup
RegSpiDir

Table 16-1: Address mapping for SPI

When the peripheral is used as a general-purpose parallel I/O port, the SPI pads are mapped in the registers as follows (**RegSpiDataOut**, **RegSpiDatain**, **RegSpiPullup** and **RegSpiDir**):

SPI pad names	SPI register bits
NSS	3
MOSI	2
MISO	1
SCK	0

Table 16-2: Pin mapping in general purpose port mode

Block configuration registers:

pos.	RegSpiControl	Rw	Reset	Function
7	ClearCounter	w1	-	Writing 1 clears transmission control counters.
6	NotSlaveSelect	r/w	1 nresetglobal	In master mode, this bit drives the NSS output pad. Unused in slave mode. It must be asserted (to 0) during a byte transfer.
5	SpiMaster	r/w	1 nresetglobal	0: SPI slave mode. 1: SPI master mode.
4	SpiEnable	r/w	0 nresetglobal	peripheral configuration: 0: general purpose digital I/O port 1: SPI interface
3	ClockPhase	r/w	1 nresetglobal	Controls the timing relationship between the serial clock and SPI data (cf. Figure 16-2 and Figure 16-3).
2	ClockPolarity	r/w	0 nresetglobal	Determines the idle-state of the SPI clock signal (cf. Figure 16-2 and Figure 16-3).
1-0	BaudRate	r/w	00 nresetglobal	Selects the baud rate in master mode. 00 => ckRcExt/2 01 => ckRcExt/8 10 => ckRcExt/16 11 => 4kHz

Table 16-3: RegSpiControl

Note that the precision of the 4kHz depends on the selected clock source (see documentation of the clock block). In slave mode, the fastest clock of the circuit should be at least 4 times faster than the baud rate of the master.

pos.	RegSpiStatus	Rw	Reset	Function
7-3	--	r	00000	Unused
2	SpiOverflow	r c1	0 nresetglobal	This flag is set when a new byte is loaded in **SpiDataIn** before the previous byte was read. Writing 1 clears the flag.
1	SpiRxFull	r	0 nresetglobal	This flag is set each time a byte transfers from the shift register to **SpiDataIn**. It is cleared by reading **SpiDataIn**.
0	SpiTxEmpty	r w1	1 nresetglobal	This flag is cleared each time the CPU writes a byte in **SpiDataOut**. It is set when a byte transfers from **SpiDataOut** to the shift register. In the master mode, writing 1 to this bit performs the data transfer **SpiDataOut** to the shift register and enables the start of transmission. In the slave mode, the byte transfer is done automatically except for the very first byte after nresetglobal (cf.16.6 Software hints).

Table 16-4: RegSpiStatus

Pos.	RegSpiDataOut	Rw	Reset	Function
7-0	SpiDataOut[7:0]	r/w	00000000 nresetglobal	SPI mode: transmission data buffer I/O mode: output data (only bits 3:0, see Table 16-2)

Table 16-5: RegSpiDataOut

Pos.	RegSpiDataIn	Rw	Reset	Function
7-0	SpiDataIn[7:0]	r	00000000 nresetglobal	SPI mode: reception data byte I/O mode: input data (only bits 3:0, see Table 16-2, bits 7:4 read 0)

Table 16-6: RegSpiDataIn

Pos.	RegSpiPullup	Rw	Reset	Function
7-4	--	r	0000	Unused.
3-0	SpiPullup	r/w	1 nresetpconf	Pullup configuration in both SPI and I/O mode (1=active). Mapping as in Table 16-2.

Table 16-7: RegSpiPullup

Note that pull-ups are disconnected independent from the value in **RegSpiPullup** if the corresponding pin is configured as an output.

Pos.	RegSpiDir	Rw	Reset	Function
7-4	--	r	0000	Unused.
3-0	SpiDir[3:0]	r/w	0 nresetpconf	I/O mode only (mapping in Table 16-2) 1: output enabled 0: output disabled

Table 16-8: RegSpiDir

16.4 Interrupts map

interrupt source	Default mapping in the interrupt manager
Irq_spi(0)	irq_bus(21)

Table 16-9: Interrupts map

Irq_spi(0) interrupt is activated at the assertion of the **SpiRxFull** flag (cf. **RegSpiStatus** description).

16.5 Function description

Depending on the value of the bit **SpiEnable** in the **RegSpiControl** register, the SPI peripheral can work either as a real SPI interface (master or slave) or as a general purpose 4 bit wide digital input/output port.

16.5.1 SPI interface

16.5.1.1 General operation

The peripheral is configured as an SPI by setting the bit **SpiEnable**=1 in **RegSpiControl**. The bit **SpiMaster** in **RegSpiControl** selects the master or slave mode.

The SPI interface supports 4 physical wires between one master device and one or more slave devices (Figure 16-1): MISO (Master In Slave Out), MOSI (Master Out Slave In), SCK (Serial Clock), NSS (Slave Select).

A byte transmission performs a rotate operation between the value stored in the 8 bit shift register of the master device and the value stored in the 8 bit shift register of the (selected) slave device. The SCK line is used to synchronize both SPI interfaces.

The transmission format depends on the two configuration bits **ClockPhase** and **ClockPolarity**. These 2 bits should be configured in the same way in the slave and the master device in order to properly run the SPI transmission. The transmission baud rate depends on the two bits **BaudRate**, these 2 bits are only used in the master device to generate the serial clock SCK signal at the selected frequency. Data are transferred in a duplex way from master to slave through the MOSI wire and from slave to master through the MISO wire. Data are always sent most significant bit first. Each SPI device sequentially operates in two times: one clock edge to sample the received bit, and the other clock edge to shift the byte inside the shift register. The NSS signal is software controlled. It must be driven by the SPI master device by writing to the bit **NotSlaveSelect** in **RegSpiControl**. NSS should remain low during the byte transmission.

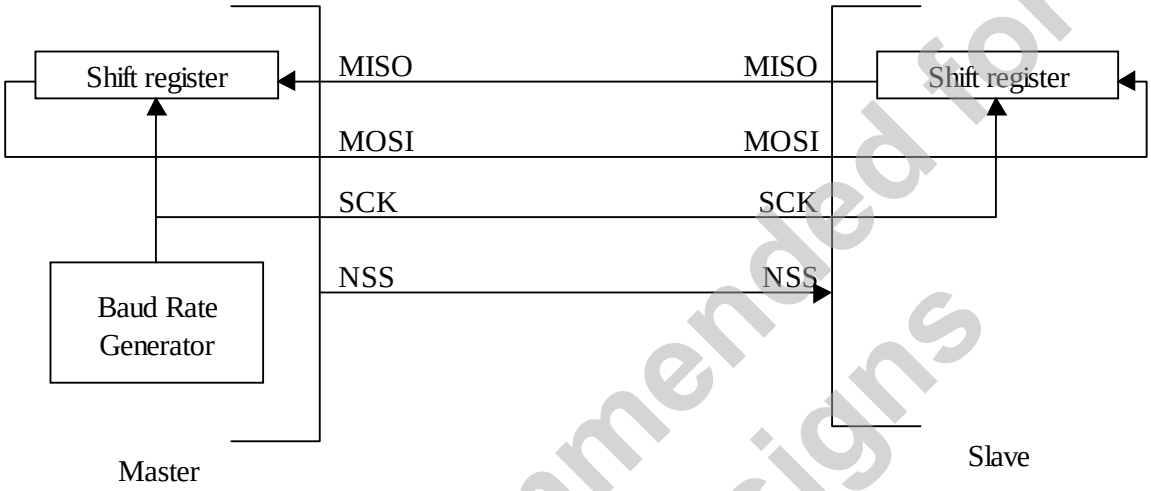


Figure 16-1: Connection of master and slave device

The next figure shows the timing diagrams for a SPI transmission with **ClockPhase** equal to 0. This means the active state of the serial clock SCK signal occurs on the 2nd half of the SCK cycle.

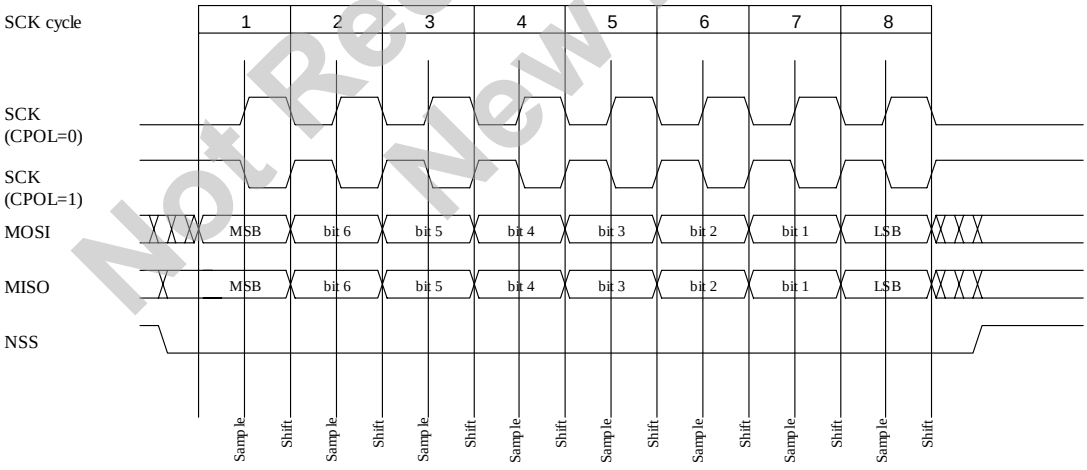


Figure 16-2: SPI transmission format with ClockPhase=0

The next figure shows the timing diagrams for a SPI transmission with **ClockPhase** equal to 1. This means the active state of the serial clock SCK signal occurs on the 1st half of the SCK cycle.

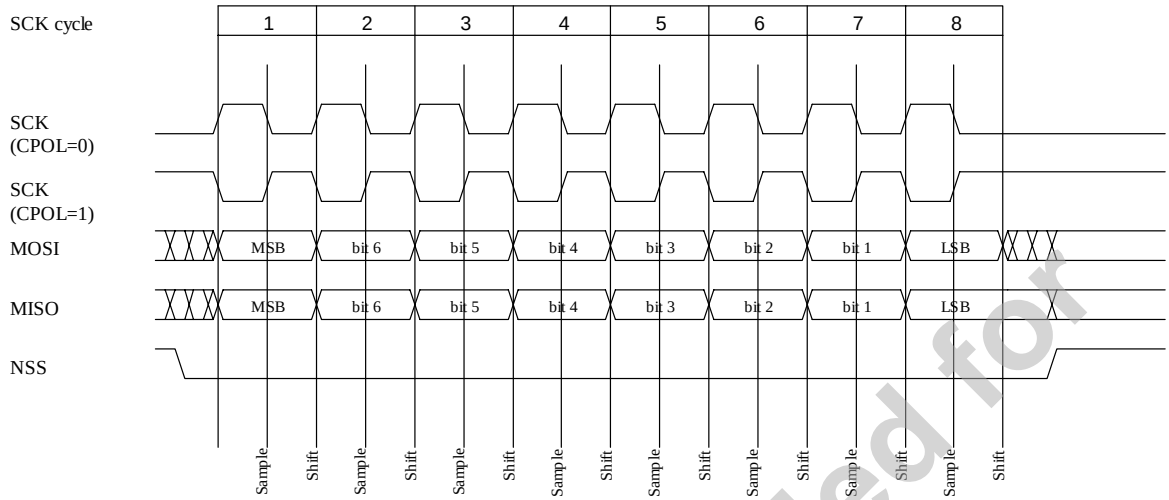


Figure 16-3: SPI transmission format with ClockPhase=1

Note: for both cases, it is not required to toggle the NSS signal back to high and back to low between each byte transmitted.

The SPI interface can be operated by polling the **RegSpiStatus** register or interrupt driven. The interrupt is active on reception of a new byte.

In case of a multi-slave configuration, any digital output pin of any parallel port can be used to select the different slaves. In some cases, it might be easier to have DC signals on these pins and to derive the timing from the single NSS pin independently from the selected slave. This can be realized by combining the NSS wire from the master device with signals coming from an output port as shown in Figure 16-4.

Pull-up resistors can be added on the input pads (MISO in master mode, MOSI, NSS and SCK in slave mode) by setting the corresponding bits in **RegSpiPullup**. Use Table 16-2 for the correspondence between the pads and register bits.

16.5.1.2 Master/Slave synchronization

In the master mode, a transmission is started by writing a 1 to the bit **SpiTxEmpty**. This automatically loads the data of the register **RegSpiDataOut** to the shift register and starts the clock and shifting. At the end of the transmission, the clock stops and the received data are copied to **RegSpiDataIn**. The bit **SpiTxEmpty** should not be asserted while the previous transmission is still running, otherwise, the transmitted and received data will be corrupted.

In slave mode, the fastest clock in the circuit should be at least 4 times faster than the baud rate of the transmission. The transmission is synchronized by the NSS input signal. While the NSS signal is high, the counters controlling the transmission are reset. The reception starts at the first clock cycle after the falling edge of NSS. At the end of the transmission, the received data are copied to **RegSpiDataIn** and the contents of the register **RegSpiDataOut** are copied automatically to the shift register. The data in the shift register can be overwritten by writing 1 to **SpiTxEmpty.** This should not be done while a transmission is running, otherwise, the transmitted and received data will be corrupted.

The counters controlling the timing of the transmission can be reset by writing a 1 to the **ClearCounters** bit in the **RegSpiControl** register. In master mode, it restarts a complete transmission cycle. In slave mode, it has the same effect as a rising edge of NSS. This bit should be used with caution.

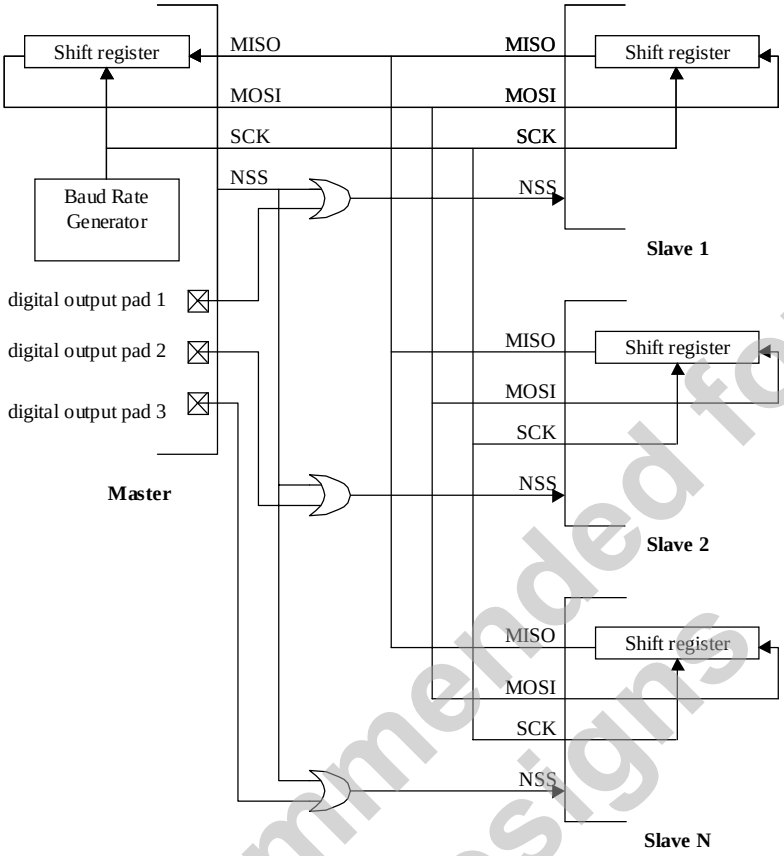


Figure 16-4: Connection of master and slaves in the case of a multi slave configuration

16.6 Software hints

16.6.1 Master mode

The following routine must be executed by the CoolRISC in order to run properly the SPI interface in master mode:

INITIALIZATION

1- Write **RegSpiControl** to enable the SPI interface and to configure the master mode (configured by default) and communication settings.
2- Write the data to send in the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.
3- Write 1 to the **SpiTxEmpty** bit to load the shift register with the value inside the **RegSpiDataOut** register and to *start the byte transmission*. The **SpiTxEmpty** flag toggles back high.
4- Write the next data to send to the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.

NORMAL OPERATION

5- Wait for the *end of transmission*, i.e. **SpiRxFull** flag is one or the interrupt of the receiver is asserted.
6- Write 1 to the **SpiTxEmpty** bit to load the shift register with the value inside the **RegSpiDataOut** register and to *start the byte transmission*. The SPI **SpiTxEmpty** flag toggles back high.
7- Read **RegSpiDataIn**. The **SpiRxFull** flag returns to 0.
8- Write the next data to send in the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.
9- Jump to 5.

16.6.2 Slave mode

The following routine must be executed by the CoolRISC in order to run properly the SPI interface in slave mode:

INITIALIZATION

1- Write **RegSpiControl** to enable the SPI interface and to configure the slave mode and communication settings (as configured in the master device).
2- Write the data to send in the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.
3- Write 1 to **SpiTxEmpty** bit to load the shift register with the value of the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles back high. This load is only required for the first byte to transmit after nresetglobal. It is automatic for the following bytes.
4- Write the next data to send in the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.

NORMAL OPERATION

5- Wait for the *end of transmission*, i.e. the **SpiRxFull** flag is one or the interrupt of the receiver asserted. The shift register is automatically loaded with the value inside the **RegSpiDataOut** register and the **SpiTxEmpty** flag toggles back high.
6- Read **RegSpiDataIn**. The **SpiRxFull** flag returns to 0.
7- Write the next data to send in the **RegSpiDataOut** register. The **SpiTxEmpty** flag toggles low.
8- Jump to 5.

16.6.3 General purpose port.

This mode is enabled when **SpiEnable** value is 0 (default value).

The SPI dedicated pads are used as a general purpose 4 bit input/output digital port. Next figure shows the structure of the SPI in this mode.

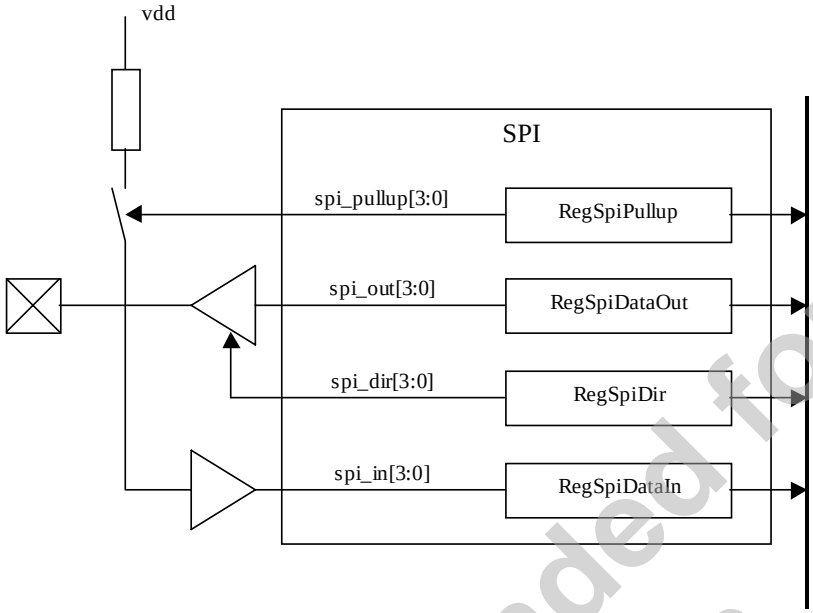


Figure 16-5: Structure of the SPI general purpose port

The direction of each bit within the SPI (input or input/output) can be individually set by using the **RegSpiDir** register. If **RegSpiDir[x]** = 1, the corresponding SPI pad becomes an output. After reset (*nresetpconf*), the SPI pads are in input configuration (**RegSpiDir[x]** are reset to 0).

In output configuration, the data are stored in **RegSpiDataOut** prior to output at SPI pads.

In input configuration, the status of SPI pads is available in **RegSpiIn** (read only). Reading is always direct –there is no digital debounce function associated with SPI pads. In case of possible noise on input signals, a software debouncer or an external filter must be realized.

If a bit in **RegSpiPullup** is set, the pull-up of the corresponding input pad is active. The pull-ups are disabled in output configuration independently of the **RegSpiPullup** content. By default after reset, the SPI pads are configured as input ports with all pull-ups active. Note that the pull-up resistors can be used for the input pads in SPI mode also.

Next table shows the link between SPI pads and SPI data bits.

SPI pad names	SPI data bits
NSS	SpiDataOut[3] or SpiDataIn[3]
MOSI	SpiDataOut[2] or SpiDataIn[2]
MISO	SpiDataOut[1] or SpiDataIn[1]
SCK	SpiDataOut[0] or SpiDataIn[0]

Table 16-10: SPI pad/bit relationship

17. Acquisition Chain

17.1 ZoomingADC™ Features

The ZoomingADC™ is a complete and versatile low-power analog front-end interface typically intended for sensing applications. The key features of the ZoomingADC™ are:

Programmable 6 to 16-bit dynamic range oversampled ADC
- Flexible gain programming between 0.5 and 1000
- Flexible and large range offset compensation
- 4-channel differential or 8-channel single-ended input multiplexer
- 2-channel differential reference inputs
- Power saving modes
- Direct interfacing to CoolRisc™ microcontroller

17.2 Overview

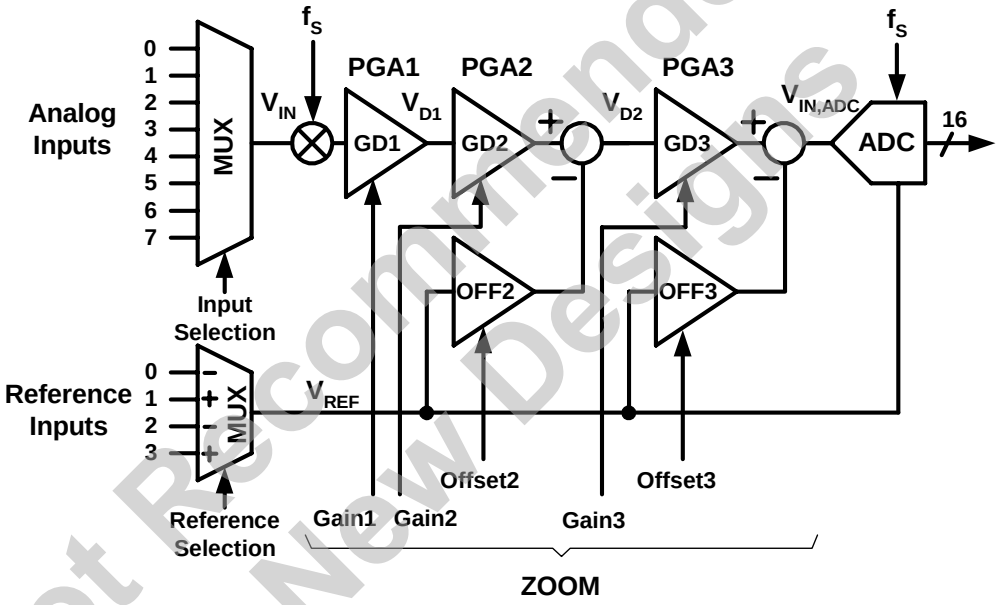


Figure 17-1. ZoomingADC™ general functional block diagram

The total acquisition chain consists of an input multiplexer, 3 programmable gain amplifier stages and an oversampled A/D converter. The reference voltage can be selected on two different channels. Two offset compensation amplifiers allow for a wide offset compensation range. The programmable gain and offset give the ability to zoom in on a small portion of the reference voltage defined input range.

17.3 Register map

There are eight registers in the acquisition chain (AC), namely **RegAcOutLsb**, **RegAcOutMsb**, **RegAcCfg0**, **RegAcCfg1**, **RegAcCfg2**, **RegAcCfg3**, **RegAcCfg4** and **RegAcCfg5**. Table 17-2 to Table 17-9 show the mapping of control bits and functionality of these registers while Table 17-1 gives an overview of these eight.

The register map only gives a short description of the different configuration bits. More detailed information is found in subsequent sections.

register name
RegAcOutLsb
RegAcOutMsb
RegAcCfg0
RegAcCfg1
RegAcCfg2
RegAcCfg3
RegAcCfg4
RegAcCfg5

Table 17-1: AC registers

pos.	RegAcOutLsb	rw	reset	description
7:0	Out[7:0]	r	00000000 nresetglobal	LSB of the output code

Table 17-2: RegAcOutLsb

pos.	RegAcOutMsb	rw	reset	description
7:0	Out[15:8]	r	00000000 nresetglobal	MSB of the output code

Table 17-3: RegAcOutMsb

pos.	RegAcCfg0	rw	reset	description
7	Start	w r0	0 nresetglobal	starts a conversion
6:5	SET_NELCONV[1:0]	r w	01 nresetglobal	sets the number of elementary conversions
4:2	SET_OSR[2:0]	r w	010 nresetglobal	sets the oversampling rate of an elementary conversion
1	CONT	r w	0 nresetglobal	continuous conversion mode
0	reserved	r w	0 nresetglobal	

Table 17-4: RegAcCfg0

pos.	RegAcCfg1	rw	reset	description
7:6	IB_AMP_ADC[1:0]	r w	11 nresetglobal	Bias current selection of the ADC converter
5:4	IB_AMP_PGA[1:0]	r w	11 nresetglobal	Bias current selection of the PGA stages
3:0	ENABLE[3:0]	r w	0000 nresetblobal	Enables the different PGA stages and the ADC

Table 17-5: RegAcCfg1

pos.	RegAcCfg2	rw	reset	description
7:6	FIN[1:0]	r w	00 nresetglobal	Sampling frequency selection
5:4	PGA2_GAIN[1:0]	r w	00 nresetglobal	PGA2 stage gain selection
3:0	PGA2_OFFSET[3:0]	r w	0000 nresetglobal	PGA2 stage offset selection

Table 17-6: RegAcCfg2

pos.	RegAcCfg3	rw	reset	description
7	PGA1_GAIN	r w	0 nresetglobal	PGA1 stage gain selection
6:0	PGA3_GAIN[6:0]	r w	0000000 nresetglobal	PGA3 stage gain selection

Table 17-7: RegAcCfg3

pos.	RegAcCfg4	rw	reset	description
7	reserved	r	0	Unused
6:0	PGA3_OFFSET[6:0]	r w	0000000 nresetglobal	PGA3 stage offset selection

Table 17-8: RegAcCfg4

pos.	RegAcCfg5	rw	reset	description
7	BUSY	r	0 nresetglobal	Activity flag
6	DEF	w r0	0	Selects default configuration
5:1	AMUX[4:0]	r w	00000 nresetglobal	Input channel configuration selector
0	VMUX	r w	0 nresetglobal	Reference channel selector

Table 17-9: RegAcCfg5

17.4 ZoomingADC™Description

Figure 17-2 gives a more detailed description of the acquisition chain.

17.4.1 Acquisition Chain

Figure 17-1 shows the general block diagram of the acquisition chain (AC). A control block (not shown in Figure 17-1) manages all communications with the CoolRisc™ microcontroller.

Analog inputs can be selected among eight input channels, while reference input is selected between two differential channels.

The core of the zooming section is made of three differential programmable amplifiers (PGA). After selection of a combination of input and reference signals V_{IN} and V_{REF}, the input voltage is modulated and amplified through stages 1 to 3. Fine gain programming up to 1'000V/V is possible. In addition, the last two stages provide programmable offset. Each amplifier can be bypassed if needed.

The output of the PGA stages is directly fed to the analog-to-digital converter (ADC), which converts the signal $V_{IN,ADC}$ into digital.

Like most ADCs intended for instrumentation or sensing applications, the ZoomingADC™ is an over-sampled converter (See Note[1]). The ADC is a so-called incremental converter, with bipolar operation (the ADC accepts both positive and negative input voltages). In first approximation, the ADC output result relative to full-scale (*FS*) delivers the quantity:

$$\frac{OUT_{ADC}}{FS/2} \cong \frac{V_{IN,ADC}}{V_{REF}/2} \qquad \text{(Eq. 1)}$$

in two's complement (see Sections 17.4 and 17.7 for details). The output code OUT_{ADC} is $-FS/2$ to $+FS/2$ for $V_{IN,ADC} \cong -V_{REF}/2$ to $+V_{REF}/2$ respectively. As will be shown in section 17.6, $V_{IN,ADC}$ is related to input voltage V_{IN} by the relationship:

$$V_{IN,ADC} = GD_{TOT} \cdot V_{IN} - GDoff_{TOT} \cdot V_{REF} \text{ (V)} \qquad \text{(Eq. 2)}$$

where GD_{TOT} is the total PGA gain, and $GDoff_{TOT}$ is the total PGA offset.

[1] Note: Over-sampled converters are operated with a sampling frequency f_S much higher than the input signal's Nyquist rate (typically f_S is 20-1'000 times the input signal bandwidth). The sampling frequency to throughput ratio is large (typically 10-500). These converters include digital decimation filtering. They are mainly used for high resolution, and/or low-to-medium speed applications.

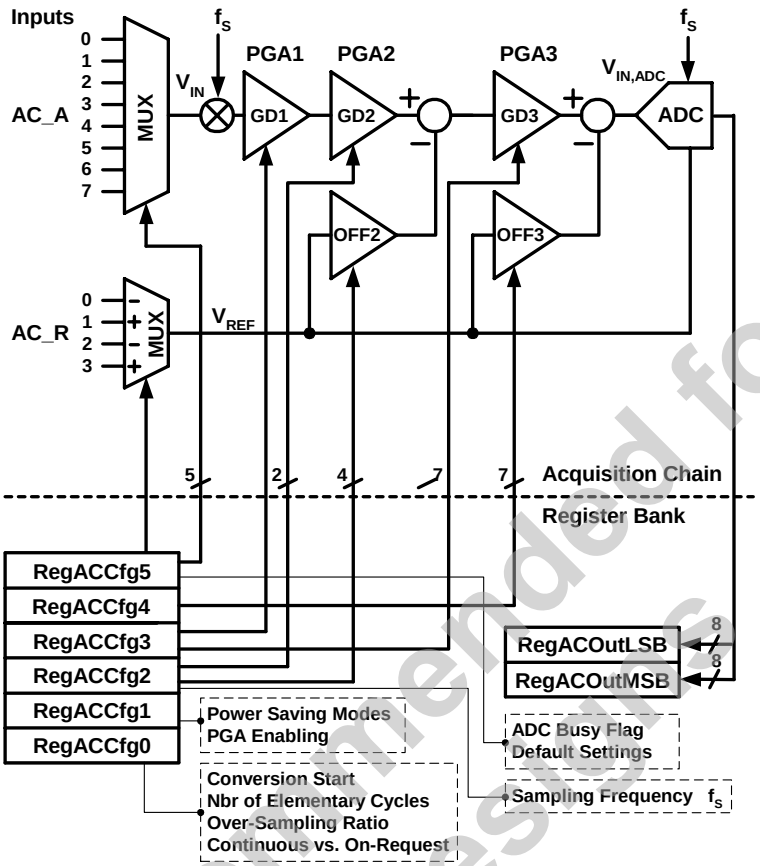


Figure 17-2. ZoomingADC™ detailed functional block diagram

17.4.2 Peripheral Registers

Figure 17-2 shows a detailed functional diagram of the ZoomingADC™.

In Table 17-10 the configuration of the peripheral registers is detailed. The system has a bank of eight 8-bit registers: six registers are used to configure the acquisition chain (**RegAcCfg0 to 5**), and two registers are used to store the output code of the analog-to-digital conversion (**RegAcOutMsb & Lsb**). The register coding of the ADC parameters and performance characteristics are detailed in Section 17.7.

Table 17-10. Peripheral registers to configure the acquisition chain (AC) and to store the analog-to-digital conversion (ADC) result

Register Name	Bit Position							
	7	6	5	4	3	2	1	0
RegAcOutLsb	OUT[7:0]							
RegAcOutMsb	OUT[15:8]							
RegAcCfg0 Default values:	START 0	SET_NELC[1:0] 01		SET_OSR[2:0] 010			CONT 0	TEST 0
RegAcCfg1 Default values:	IB_AMP_ADC[1:0] 11		IB_AMP_PGA[1:0] 11		ENABLE[3:0] 0001			

Cont'

Register Name	Bit Position							
	7	6	5	4	3	2	1	0
RegAcCfg2 Default values:	FIN[1:0] 00		PGA2_GAIN[1:0] 00		PGA2_OFFSET[3:0] 0000			
RegAcCfg3 Default values:	PGA1_G 0	PGA3_GAIN[6:0] 0000000						
RegAcCfg4 Default values:	0	PGA3_OFFSET[6:0] 0000000						
RegAcCfg5 Default values:	BUSY 0	DEF 0	AMUX[4:0] 00000					VMUX 0

With:

- OUT: (r) digital output code of the analog-to-digital converter. (MSB = OUT[15])
- START: (w) setting this bit triggers a single conversion (after the current one is finished). This bit always reads back 0.
- SET_NELC: (rw) sets the number of elementary conversions to $2^{SET\_NELC[1:0]}$. To compensate for offsets, the input signal is chopped between elementary conversions (1,2,4,8).
- SET_OSR: (rw) sets the over-sampling rate (*OSR*) of an elementary conversion to $2^{(3+SET\_OSR[2:0])}$. *OSR* = 8, 16, 32, ..., 512, 1024.
- CONT: (rw) setting this bit starts a conversion. A new conversion will automatically begin as long as the bit remains at 1.
- TEST: bit only used for test purposes. In normal mode, this bit is forced to 0 and cannot be overwritten.
- IB_AMP_ADC: (rw) sets the bias current in the ADC to 0.25*(1+ IB_AMP_ADC[1:0]) of the normal operation current (25, 50, 75 or 100% of nominal current). To be used for low-power, low-speed operation.
- IB_AMP_PGA: (rw) sets the bias current in the PGAs to 0.25*(1+IB_AMP_PGA[1:0]) of the normal operation current (25, 50, 75 or 100% of nominal current). To be used for low-power, low-speed operation.
- ENABLE: (rw) enables the ADC modulator (bit 0) and the different stages of the PGAs (PGAi by bit i=1,2,3). PGA stages that are disabled are bypassed.
- FIN: (rw) These bits set the sampling frequency of the acquisition chain. Expressed as a fraction of the oscillator frequency, the sampling frequency is given as: 00 → 1/4 f_{RC}, 01 → 1/8 f_{RC}, 10 → 1/32 f_{RC}, 11→ ~8kHz.
- PGA1_GAIN: (rw) sets the gain of the first stage: 0 → 1, 1 → 10.
- PGA2_GAIN: (rw) sets the gain of the second stage: 00 → 1, 01 → 2, 10 → 5, 11 → 10.
- PGA3_GAIN: (rw) sets the gain of the third stage to PGA3_GAIN[6:0]·1/12.
- PGA2_OFFSET: (rw) sets the offset of the second stage between –1 and +1, with increments of 0.2. The MSB gives the sign (0 → positive, 1 → negative); amplitude is coded with the bits PGA2_OFFSET[5:0].
- PGA3_OFFSET: (rw) sets the offset of the third stage between –5.25 and +5.25, with increments of 1/12. The MSB gives the sign (0 → positive, 1 → negative); amplitude is coded with the bits PGA3_OFFSET[5:0].
- BUSY: (r) set to 1 if a conversion is running.
- DEF: (w) sets all values to their defaults (PGA disabled, max speed, nominal modulator bias current, 2 elementary conversions, over-sampling rate of 32) and starts a new conversion without waiting the end of the preceding one.
- AMUX(4:0): (rw) AMUX[4] sets the mode (0 → 4 differential inputs, 1 → 7 inputs with A(0) = common reference) AMUX(3) sets the sign (0 → straight, 1→ cross) AMUX[2:0] sets the channel.
- VMUX: (rw) sets the differential reference channel (0 → R(1) and R(0), 1 → R(3) and R(2)).

(r = read; w = write; rw = read & write)

17.4.3 Continuous-Time vs. On-Request

The ADC can be operated in two distinct modes: "continuous-time" and "on-request" modes (selected using the bit **CONT**).

In "continuous-time" mode, the input signal is repeatedly converted into digital. After a conversion is finished, a new one is automatically initiated. The new value is then written in the result register, and the corresponding internal trigger pulse is generated. This operation is sketched in Figure 17-3. The conversion time in this case is defined as T_{CONV}.

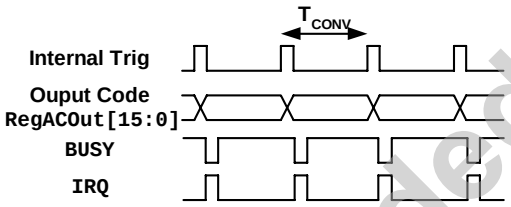


Figure 17-3. ADC "continuous-time" operation

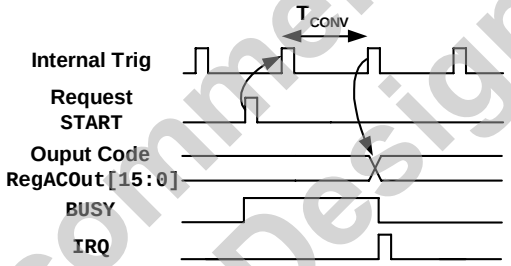


Figure 17-4. ADC "on-request" operation

In the "on-request" mode, the internal behavior of the converter is the same as in the "continuous-time" mode, but the conversion is initiated on user request (with the **START** bit). As shown in Figure 17-4, the conversion time is also T_{CONV}.

17.5 Input Multiplexers

The ZoomingADC™ has eight analog inputs AC_A(0) to AC_A(7) and four reference inputs AC_R(0) to AC_R(3). Let us first define the differential input voltage V_{IN} and reference voltage V_{REF} respectively as:

$$V_{IN} = V_{INP} - V_{INN} \quad \text{(V)} \qquad \text{(Eq. 3)}$$

and:

$$V_{REF} = V_{REFP} - V_{REFN} \quad \text{(V)} \qquad \text{(Eq. 4)}$$

As shown in Table 17-11 the inputs can be configured in two ways: either as 4 differential channels (V_{IN1} = AC_A(1) - AC_A(0),..., V_{IN4} = AC_A(7) - AC_A(6)), or AC_A(0) can be used as a common reference, providing 7 signal paths all referenced to AC_A(0). The control word for the analog input selection is **AMUX[4:0]**. Notice that the bit **AMUX[3]** controls the sign of the input voltage.

AMUX[4:0] (RegAcCfg5[5:1])	V_{INP}	V_{INN}
00x00	AC_A(1)	AC_A(0)
00x01	AC_A(3)	AC_A(2)
00x10	AC_A(5)	AC_A(4)
00x11	AC_A(7)	AC_A(6)
10000	AC_A(0)	AC_A(0)
10001	AC_A(1)	
10010	AC_A(2)	
10011	AC_A(3)	
10100	AC_A(4)	
10101	AC_A(5)	
10110	AC_A(6)	
10111	AC_A(7)	

AMUX[4:0] (RegAcCfg5[5:1])	V_{INP}	V_{INN}
01x00	AC_A(0)	AC_A(1)
01x01	AC_A(2)	AC_A(3)
01x10	AC_A(4)	AC_A(5)
01x11	AC_A(6)	AC_A(7)
11000	AC_A(0)	AC_A(0)
11001		AC_A(1)
11010		AC_A(2)
11011		AC_A(3)
11100		AC_A(4)
11101		AC_A(5)
11110		AC_A(6)
11111		AC_A(7)

Table 17-11. Analog input selection

Similarly, the reference voltage is chosen among two differential channels (V_{REF1} = `AC_R(1)-AC_R(0)` or V_{REF2} = `AC_R(3)-AC_R(2)`) as shown in Table 17-12. The selection bit is **VMUX**. The reference inputs V_{REFP} and V_{REFN} (common-mode) can be up to the power supply range.

VMUX (RegAcCfg5[0])	V_{REFP}	V_{REFN}
0	AC_R(1)	AC_R(0)
1	AC_R(3)	AC_R(2)

Table 17-12. Analog Reference input selection

17.6 Programmable Gain Amplifiers

As seen in Figure 17-1, the zooming function is implemented with three programmable gain amplifiers (PGA). These are:

- PGA1: coarse gain tuning
- PGA2: medium gain and offset tuning
- PGA3: fine gain and offset tuning

All gain and offset settings are realized with ratios of capacitors. The user has control over each PGA activation and gain, as well as the offset of stages 2 and 3. These functions are examined hereafter.

ENABLE[3:0]	Block
xxx0	ADC disabled
xxx1	ADC enabled
xx0x	PGA1 disabled
xx1x	PGA1 enabled
x0xx	PGA2 disabled
x1xx	PGA2 enabled
0xxx	PGA3 disabled
1xxx	PGA3 enabled

Table 17-13. ADC & PGA enabling

PGA1_GAIN	PGA1 Gain GD_1 (V/V)
0	1
1	10

Table 17-14. PGA1 Gain Settings

PGA2_GAIN[1:0]	PGA2 Gain GD_2 (V/V)
00	1
01	2
10	5
11	10

Table 17-15. PGA2 gain settings

PGA2_OFFSET[3:0]	PGA2 Offset $GDoff_2$ (V/V)
0000	0
0001	+0.2
0010	+0.4
0011	+0.6
0100	+0.8
0101	+1
1001	-0.2
1010	-0.4
1011	-0.6
1100	-0.8
1101	-1

Table 17-16. PGA2 offset settings

PGA3_GAIN[6:0]	PGA3 Gain GD_3 (V/V)
0000000	0
0000001	1/12(=0.083)
...	...
0000110	6/12
...	...
0001100	12/12
0010000	16/12
...	
0100000	32/12
...	
1000000	64/12
...	
1111111	127/12(=10.58)

Table 17-17. PGA3 gain settings

PGA3_OFFSET[6:0]	PGA3 Offset $GDoff_3$ (V/V)
0000000	0
0000001	+1/12(=+0.083)
0000010	+2/12
...	...
0010000	+16/12
...	...
0100000	+32/12
...	...
0111111	+63/12(=+5.25)
1000000	0
1000001	-1/12(=-0.083)
1000010	-2/12
...	...
1010000	-16/12
...	...
1100000	-32/12
...	...
1111111	-63/12(=-5.25)

Table 17-18. PGA3 offset settings

17.6.1 PGA & ADC Enabling

Depending on the application objectives, the user may enable or bypass each PGA stage. This is done using the word **ENABLE** and the coding given in Table 17-13. To reduce power dissipation, the ADC can also be inactivated while idle.

17.6.2 PGA1

The first stage can have a buffer function (unity gain) or provide a gain of 10 (see Table 17-14). The voltage V_{D1} at the output of PGA1 is:

$$V_{D1} = GD_1 \cdot V_{IN} \quad \text{(V)} \qquad \text{(Eq. 5)}$$

where GD_1 is the gain of PGA1 (in V/V) controlled with the bit **PGA1_GAIN**.

17.6.3 PGA2

The second PGA has a finer gain and offset tuning capability, as shown in Table 17-15 and Table 17-16. The voltage V_{D2} at the output of PGA2 is given by:

$$V_{D2} = GD_2 \cdot V_{D1} - GDoff_2 \cdot V_{REF} \quad \text{(V)} \qquad \text{(Eq. 6)}$$

where GD_2 and $GDoff_2$ are respectively the gain and offset of PGA2 (in V/V). These are controlled with the words **PGA2_GAIN[1:0]** and **PGA2_OFFSET[3:0]** .

17.6.4 PGA3

The finest gain and offset tuning is performed with the third and last PGA stage, according to the coding of Table 17-17 and Table 17-18. The output of PGA3 is also the input of the ADC. Thus, similarly to PGA2, we find that the voltage entering the ADC is given by:

$$V_{IN,ADC} = GD_3 \cdot V_{D2} - GDoff_3 \cdot V_{REF} \quad \text{(V)} \qquad \text{(Eq. 7)}$$

where GD_3 and $GDoff_3$ are respectively the gain and offset of PGA3 (in V/V). The control words are **PGA3_GAIN[6:0]** and **PGA3_OFFSET[6:0]** . To remain within the signal compliance of the PGA stages, the condition:

$$V_{D1}, V_{D2} < V_{DD} \quad \text{(V)} \qquad \text{(Eq. 8)}$$

must be verified.

Finally, combining equations Eq. 5 to Eq. 7 for the three PGA stages, the input voltage $V_{IN,ADC}$ of the ADC is related to V_{IN} by:

$$V_{IN,ADC} = GD_{TOT} \cdot V_{IN} - GDoff_{TOT} \cdot V_{REF} \quad \text{(V)} \qquad \text{(Eq. 9)}$$

where the total PGA gain is defined as:

$$GD_{TOT} = GD_3 \cdot GD_2 \cdot GD_1 \quad \text{(V/V)} \qquad \text{(Eq. 10)}$$

and the total PGA offset is:

$$GDoff_{TOT} = GDoff_3 + GD_3 \cdot GDoff_2 \quad \text{(V/V)} \qquad \text{(Eq. 11)}$$

17.7 ADC Characteristics

The main performance characteristics of the ADC (resolution, conversion time, etc.) are determined by three programmable parameters:

- sampling frequency f_S,
- over-sampling ratio *OSR*, and
- number of elementary conversions N_{ELCONV}.

The setting of these parameters and the resulting performances are described hereafter.

17.7.1 Conversion Sequence

A conversion is started each time the bit **START** or the bit **DEF** is set. As depicted in Figure 17-5, a complete analog-to-digital conversion sequence is made of a set of N_{ELCONV} elementary incremental conversions and a final quantitative step. Each elementary conversion is made of (*OSR*+1) sampling periods T_S=1/f_S, i.e.:

$$T_{ELCONV} = (OSR+1)/f_S \quad \text{(s)} \qquad \text{(Eq. 12)}$$

The result is the mean of the elementary conversion results. An important feature is that the elementary conversions are alternatively performed with the offset of the internal amplifiers contributing in one direction and the other to the output code. Thus, converter internal offset is eliminated if at least two elementary sequences are performed (i.e. if $N_{ELCONV} \geq 2$). A few additional clock cycles are also required to initiate and end the conversion properly.

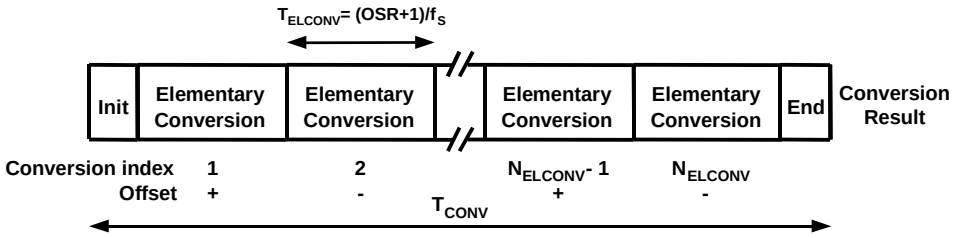


Figure 17-5. Analog-to-digital conversion sequence

17.7.2 Sampling Frequency

The word **FIN[1:0]** is used to select the sampling frequency f_S (Table 17-19). Three sub-multiples of the internal RC-based frequency f_{RCEXT} can be chosen. For **FIN** = "11", sampling frequency is about 8kHz. Additional information on oscillators and their control can be found in the clock block documentation.

FIN[1:0]	Sampling Frequency f_S (Hz)	
	XE8801/05	XE8802
00	$1/4 \cdot f_{RC}$	$1/8 \cdot f_{RCEXT}$
01	$1/8 \cdot f_{RC}$	$1/16 \cdot f_{RCEXT}$
10	$1/32 \cdot f_{RC}$	$1/64 \cdot f_{RCEXT}$
11	~8kHz	~4kHz

Table 17-19. Sampling frequency settings (f_{RC}= RC-based frequency)

17.7.3 Over-Sampling Ratio

The over-sampling ratio (*OSR*) defines the number of integration cycles per elementary conversion. Its value is set with the word **SET_OSR[2:0]** in power of 2 steps (see Table 17-20) given by:

$$OSR = 2^{3+\mathrm{SET\_OSR}[2:0]} \quad (-) \qquad \text{(Eq. 13)}$$

SET_OSR[2:0] (RegAcCfg0[4:2])	Over-Sampling Ratio *OSR* (-)
000	8
001	16
010	32
011	64
100	128
101	256
110	512
111	1024

Table 17-20. Over-sampling ratio settings

17.7.4 Elementary Conversions

As mentioned previously, the whole conversion sequence is made of a set of N_{ELCONV} elementary incremental conversions. This number is set with the word **SET_NELC[1:0]** in binary steps (see Table 17-21) given by:

$$N_{ELCONV} = 2^{\mathrm{SET\_NELC}[1:0]} \quad (-) \qquad \text{(Eq. 14)}$$

SET_NELC[1:0] (RegAcCfg0[6:5])	# of Elementary Conversions N_{ELCONV} (-)
00	1
01	2
10	4
11	8

Table 17-21. Number of elementary conversion settings

As already mentioned, N_{ELCONV} must be equal or greater than 2 to reduce internal amplifier offsets.

17.7.5 Resolution

The theoretical resolution of the ADC, without considering thermal noise, is given by:

$$n = 2 \cdot \log_2(OSR) + \log_2(N_{ELCONV}) \quad \text{(Bits)} \qquad \text{(Eq. 15)}$$

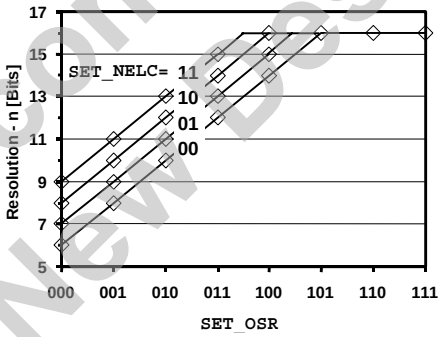


Figure 17-6. Resolution vs. SET_OSR[2:0] and SET_NELC[2:0]

SET_OSR [2:0]	SET_NELC			
	00	01	10	11
000	6	7	8	9
001	8	9	10	11
010	10	11	12	13
011	12	13	14	15
100	14	15	16	16
101	16	16	16	16
110	16	16	16	16
111	16	16	16	16

(shaded area: resolution truncated to 16 bits due to output register size **RegAcOut[15:0]**)

Table 17-22. Resolution vs. SET_OSR[2:0] and SET_NELC[1:0] settings

Using Table 17-22 or the graph plotted in Figure 17-6, resolution can be set between 6 and 16 bits. Notice that, because of 16-bit register use for the ADC output, **practical resolution is limited to 16 bits**, i.e. $n \leq 16$. Even if the resolution is truncated to 16 bit by the output register size, it may make sense to set OSR and N_{ELCONV} to higher values in order to reduce the influence of the thermal noise in the PGA (see section 17.8.4).

17.7.6 Conversion Time & Throughput

As explained using Figure 17-5, conversion time is given by:

$$T_{CONV} = (N_{ELCONV} \cdot (OSR + 1) + 1) / f_S \text{ (s)} \qquad \text{(Eq. 16)}$$

and throughput is then simply $1/T_{CONV}$. For example, consider an over-sampling ratio of 256, 2 elementary conversions, and a sampling frequency of 500kHz (**SET_OSR** = "101", **SET_NELC** = "01", f_{RC} = 2MHz, and **FIN** = "00"). In this case, using Table 17-23, the conversion time is 515 sampling periods, or 1.03ms. This corresponds to a throughput of 971Hz in continuous-time mode. The plot of Figure 17-7 illustrates the classic trade-off between resolution and conversion time.

SET_OSR [2:0]	SET_NELC[1:0]			
	00	01	10	11
000	10	19	37	73
001	18	35	69	137
010	34	67	133	265
011	66	131	261	521
100	130	259	517	1033
101	258	515	1029	2057
110	514	1027	2053	4105
111	1026	2051	4101	8201

Table 17-23. Normalized conversion time ($T_{CONV} \cdot f_S$) vs. SET_OSR[2:0] and SET_NELC[1:0] (normalized to sampling period $1/f_S$)

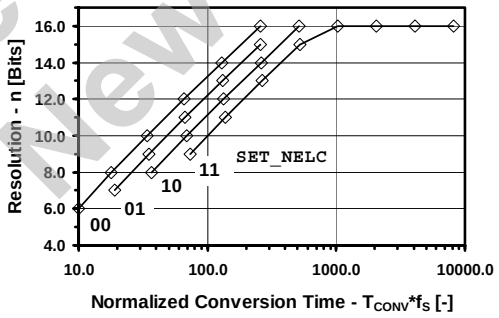


Figure 17-7. Resolution vs. normalized conversion time for different SET_NELC[1:0]

17.7.7 Output Code Format

The ADC output code is a 16-bit word in two's complement format (see Table 17-24). For input voltages outside the range, the output code is saturated to the closest full-scale value (i.e. 0x7FFF or 0x8000). For resolutions smaller than 16 bits, the non-significant bits are forced to the values shown in Table 17-25. The output code, expressed in LSBs, corresponds to:

$$OUT_{ADC} = 2^{16} \cdot \frac{V_{IN,ADC}}{V_{REF}} \cdot \frac{OSR + 1}{OSR} \text{ (LSB)} \qquad \text{(Eq.17)}$$

Recalling equation Eq. 9, this can be rewritten as:

$$OUT_{ADC} = 2^{16} \cdot \frac{V_{IN}}{V_{REF}} \cdot \left(GD_{TOT} - GDoff_{TOT} \cdot \frac{V_{REF}}{V_{IN}} \right) \cdot \frac{OSR+1}{OSR} \quad \text{(LSB)} \quad \text{(Eq. 18)}$$

where, from Eq. 10 and Eq. 11, the total PGA gain and offset are respectively:

$$GD_{TOT} = GD_3 \cdot GD_2 \cdot GD_1 \quad \text{(V/V)}$$

and:

$$GDoff_{TOT} = GDoff_3 + GD_3 \cdot GDoff_2 \quad \text{(V/V)}$$

ADC Input Voltage $V_{IN,ADC}$	% of Full Scale (FS)	Output in LSBs	Output Code in Hex
+2.49505V	+0.5·FS	$+2^{15}$-1 =+32'767	7FFF
+2.49497V	...	$+2^{15}$-2 =+32'766	7FFE
...	...	...	...
+76.145µV	...	+1	0001
0V	0	0	0000
-76.145µV	...	-1	FFFF
...	...	...	...
-2.49505V	...	-2^{15}-1 =-32'767	8001
-2.49513V	-0.5·FS	-2^{15} =-32'768	8000

Table 17-24. Basic ADC Relationships (example for: V_{REF} = 5V, *OSR* = 512, *n* = 16 bits)

SET_OSR [2:0]	SET_NELC = 00	SET_NELC = 01	SET_NELC = 10	SET_NELC = 11
000	1000000000	100000000	10000000	1000000
001	10000000	1000000	100000	10000
010	100000	10000	1000	100
011	1000	100	10	1
100	10	1	-	-
101	-	-	-	-
110	-	-	-	-
111	-	-	-	-

Table 17-25. Last forced LSBs in conversion output registers for resolution settings smaller than 16 bits (*n* < 16) (`RegAcOutMsb[7:0]` & `RegAcOutLsb[7:0]`)

The equivalent LSB size at the input of the PGA chain is:

$$LSB = \frac{1}{2^n} \cdot \frac{V_{REF}}{GD_{TOT}} \cdot \frac{OSR}{OSR+1} \quad \text{(V)} \quad \text{(Eq. 19)}$$

Notice that the input voltage $V_{IN,ADC}$ of the ADC must satisfy the condition:

$$\left|V_{IN,ADC}\right| \le \frac{1}{2}\cdot(V_{REFP}-V_{REFN})\cdot\frac{OSR}{OSR+1} \quad \text{(V)} \qquad \text{(Eq. 20)}$$

to remain within the ADC input range.

17.7.8 Power Saving Modes

During low-speed operation, the bias current in the PGAs and ADC can be programmed to save power using the control words **IB_AMP_PGA[1:0]** and **IB_AMP_ADC[1:0]** (see Table 17-26). If the system is idle, the PGAs and ADC can even be disabled, thus, reducing power consumption to its minimum. This can considerably improve battery lifetime.

IB_AMP_ADC [1:0]	IB_AMP_PGA [1:0]	ADC Bias Current	PGA Bias Current	Max. f_S [kHz]
00 01 10 11	x	1/4·I_{ADC} 1/2·I_{ADC} 3/4·I_{ADC} I_{ADC}	x	62.5 125 250 500
x	00 01 10 11	x	1/4·I_{PGA} 1/2·I_{PGA} 3/4·I_{PGA} I_{PGA}	62.5 125 250 500

Table 17-26. ADC & PGA power saving modes and maximum sampling frequency

17.8 Specifications and Measured Curves

This section presents measurement results for the acquisition chain. A summary table with circuit specifications and measured curves are given.

17.8.1 Default Settings

Unless otherwise specified, the measurement conditions are the following:

- Temperature T_A = +25°C
- V_{DD} = +5V, GND = 0V, V_{REF} = +5V, V_{IN} = 0V
- RC frequency f_{RC} = 2MHz, sampling frequency f_S = 500kHz
- Offsets $GDOff_2$ = $GDOff_3$ = 0
- Power operation: normal (**IB_AMP_ADC[1:0]** = **IB_AMP_PGA[1:0]** = '11')
- Resolution: for n = 12 bits: OSR = 32 and N_{ELCONV} = 4
  - for n = 16 bits: OSR = 512 and N_{ELCONV} = 2

17.8.2 Specifications

Unless otherwise specified: Temperature T_A = +25°C, V_{DD} = +5V, GND = 0V, V_{REF} = +5V, V_{IN} = 0V, RC frequency f_{RC} = 2MHz, sampling frequency f_S = 500kHz, Overall PGA gain GD_{TOT} = 1, offsets $GDOff_2$ = $GDOff_3$ = 0. Power operation: normal (`IB_AMP_ADC[1:0]` = `IB_AMP_PGA[1:0]` = '11'). For resolution *n* = 12 bits: *OSR* = 32 and N_{ELCONV} = 4. For resolution *n* = 16 bits: *OSR* = 512 and N_{ELCONV} = 2.

PARAMETER	VALUE			UNITS	COMMENTS/CONDITIONS
	MIN	TYP	MAX		
ANALOG INPUT CHARACTERISTICS					
Differential Input Voltage Ranges	-2.42		+2.42	V	Gain = 1, *OSR* = 32 (Note 1)
$V_{IN} = (V_{INP} - V_{INN})$	-24.2		+24.2	mV	Gain = 100, *OSR* = 32
	-2.42		+2.42	mV	Gain = 1000, *OSR* = 32
Reference Voltage Range					
$V_{REF} = (V_{REFP} – V_{REFN})$			V_{DD}	V	
PROGRAMMABLE GAIN AMPLIFIERS (PGA)					
Total PGA Gain, GD_{TOT}	0.5		1000	V/V	
PGA1 Gain, GD_1	1		10	V/V	See Table 17-14
PGA2 Gain, GD_2	1		10	V/V	See Table 17-15
PGA3 Gain, GD_3	0		127/12	V/V	Step=1/12 V/V, See Table 17-17
Gain Setting Precision (each stage)	-3	±0.5	+3	%	
Gain Temperature Dependence		±5		ppm/°C	
Offset					
PGA2 Offset, $GDoff_2$	-1		+1	V/V	Step=0.2 V/V, See Table 17-16
PGA3 Offset, $GDoff_3$	-127/12		+127/12	V/V	Step=1/12 V/V, See Table 17-18
Offset Setting Precision (PGA2 or 3)	-3	±0.5	+3	%	(Note 2)
Offset Temperature Dependence		±5		ppm/°C	
Input Impedance					
PGA1	1500			kΩ	PGA1 Gain = 1 (Note 3)
	150			kΩ	PGA1 Gain = 10 (Note 3)
PGA2, PGA3	150			kΩ	Maximal gain (Note 3)
Output RMS Noise					
PGA1		205		µV	(Note 4)
PGA2		340		µV	(Note 5)
PGA3		365		µV	(Note 6)
ADC STATIC PERFORMANCE					
Resolution, n	6		16	Bits	(Note 7)
No Missing Codes					(Note 8)
Gain Error		±0.15		% of FS	(Note 9)
Offset Error		±1		LSB	n = 16 bits (Note 10)
Integral Non-Linearity, INL					
Resolution n = 16 Bits		±1.0		LSB	(Note 11)
Differential Non-Linearity, DNL					
Resolution n = 16 Bits		±0.5		LSB	(Note 12)
Power Supply Rejection Ratio, PSRR		78		dB	V_{DD} = 5V ± 0.3V (Note 13)
		72		dB	V_{DD} = 3V ± 0.3V (Note 13)
DYNAMIC PERFORMANCE					
Sampling Frequency, f_S	3			kHz	
Conversion Time, T_{CONV}		133		cycles/f_S	n = 12 bits (Note 14)
		1027		cycles/f_S	n = 16 bits (Note 14)
Throughput Rate (Continuous Mode), $1/T_{CONV}$		3.76		kSps	n = 12 bits, f_S = 500kHz
		0.49		kSps	n = 16 bits, f_S = 500kHz
Nbr of Initialization Cycles, N_{INIT}	0		2	cycles	
Nbr of End Conversion Cycles, N_{END}	0		5	cycles	
PGA Stabilization Delay		*OSR*		cycles	(Note 15)
DIGITAL OUTPUT					
ADC Output Data Coding					Binary Two's Complement See Table 17-24 and Table 17-25

Specifications (Cont'd)

PARAMETER	VALUE MIN	VALUE TYP	VALUE MAX	UNITS	COMMENTS/CONDITIONS
POWER SUPPLY					
Voltage Supply Range, V_{DD}	+2.4	+5	+5.5	V	
Analog Quiescent Current Consumption, Total (I_Q)		720/620		µA	Only Acquisition Chain V_{DD} = 5V/3V
ADC Only		250/190		µA	V_{DD} = 5V/3V
PGA1		165/150		µA	V_{DD} = 5V/3V
PGA2		130/120		µA	V_{DD} = 5V/3V
PGA3		175/160		µA	V_{DD} = 5V/3V
Analog Power Dissipation					All PGAs & ADC Active
Normal Power Mode		3.6/1.9		mW	V_{DD} = 5V/3V (Note 16)
3/4 Power Reduction Mode		2.7/1.4		mW	V_{DD} = 5V/3V (Note 17)
1/2 Power Reduction Mode		1.8/0.9		mW	V_{DD} = 5V/3V (Note 18)
1/4 Power Reduction Mode		0.9/0.5		mW	V_{DD} = 5V/3V (Note 19)
TEMPERATURE					
Specified Range	-40		+85	°C	
Operating Range	-40		+125	°C	

Notes:

(1) Gain defined as overall PGA gain GD_{TOT} = $GD_1 \cdot GD_2 \cdot GD_3$. Maximum input voltage is given by: $V_{IN,MAX} = \pm(V_{REF}/2) \cdot (OSR/OSR+1)$.
(2) Offset due to tolerance on $GDoff_2$ or $GDoff_3$ setting. For small intrinsic offset, use only ADC and PGA1.
(3) Measured with block connected to inputs through AMUX block. Normalized input sampling frequency for input impedance is f_S = 512kHz. This figure must be multiplied by 2 for f_S = 256kHz, 4 for f_S = 128kHz. Input impedance is proportional to $1/f_S$.
(4) Figure independent from PGA1 gain and sampling frequency f_S. See model of Figure 17-18(a). See equation Eq. 21 to calculate equivalent input noise.
(5) Figure independent on PGA2 gain and sampling frequency f_S. See model of Figure 17-18(a). See equation Eq. 21 to calculate equivalent input noise.
(6) Figure independent on PGA3 gain and sampling frequency f_S. See model of Figure 17-18(a) and equation Eq. 21 to calculate equivalent input noise.
(7) Resolution is given by n = 2·log2(*OSR*) + log2(N_{ELCONV}). *OSR* can be set between 8 and 1024, in powers of 2. N_{ELCONV} can be set to 1, 2, 4 or 8.
(8) If a ramp signal is applied to the input, all digital codes appear in the resulting ADC output data.
(9) Gain error is defined as the amount of deviation between the ideal (theoretical) transfer function and the measured transfer function (with the offset error removed). (See Figure 17-19)
(10) Offset error is defined as the output code error for a zero volt input (ideally, output code = 0). For ± 1 LSB offset, N_{ELCONV} must be ≥2.
(11) INL defined as the deviation of the DC transfer curve of each individual code from the best-fit straight line. This specification holds over the full scale.
(12) DNL is defined as the difference (in LSB) between the ideal (1 LSB) and measured code transitions for successive codes.
(13) Figures for Gains = 1 to 100. PSRR is defined as the amount of change in the ADC output value as the power supply voltage changes.
(14) Conversion time is given by: $T_{CONV} = (N_{ELCONV} \cdot (OSR + 1) + 1) / f_S$. *OSR* can be set between 8 and 1024, in powers of 2. N_{ELCONV} can be set to 1, 2, 4 or 8.
(15) PGAs are reset after each writing operation to registers **ReqAcCfg1-5**. The ADC must be started after a PGA or inputs common-mode stabilisation delay. This is done by writing bit `Start` several cycles after PGA settings modification or channel switching. Delay between PGA start or input channel switching and ADC start should be equivalent to *OSR* (between 8 and 1024) number of cycles. This delay does not apply to conversions made without the PGAs.
(16) Nominal (maximum) bias currents in PGAs and ADC, i.e. **IB_AMP_PGA[1:0]** = '11' and **IB_AMP_ADC[1:0]** = '11'.
(17) Bias currents in PGAs and ADC set to 3/4 of nominal values, i.e. **IB_AMP_PGA[1:0]** = '10', **IB_AMP_ADC[1:0]** = '10'.
(18) Bias currents in PGAs and ADC set to 1/2 of nominal values, i.e. **IB_AMP_PGA[1:0]** = '01', **IB_AMP_ADC[1:0]** = '01'.
(19) Bias currents in PGAs and ADC set to 1/4 of nominal values, i.e. **IB_AMP_PGA[1:0]** = '00', **IB_AMP_ADC[1:0]** = '00'.

17.8.3 Linearity

17.8.3.1 Integral non-linearity

The integral non-linearity depends on the selected gain configuration. First of all, the non-linearity of the ADC (all PGA stages bypassed) is shown in Figure 17-8.

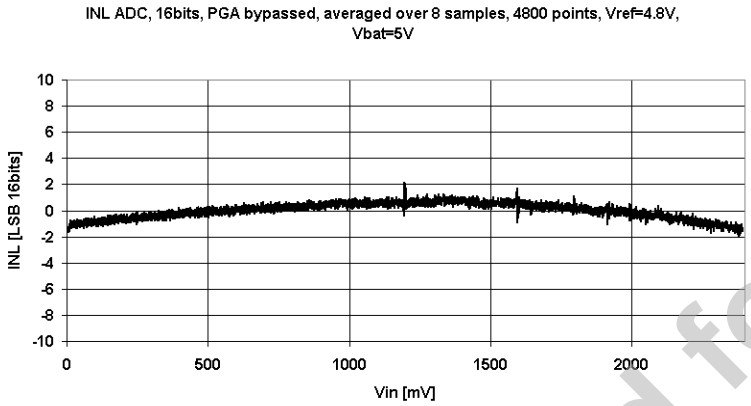


Figure 17-8. Integral non-linearity of the ADC (PGA disabled, reference voltage of 4.8V)

The different PGA stages have been designed to find the best compromise between the noise performance, the integral non-linearity and the power consumption. To obtain this, the first stage has the best noise performance and the third stage the best linearity performance. For large input signals (small PGA gains, i.e. up to about 50), the noise added by the PGA is very small with respect to the input signal and the second and third stage of the PGA should be used to get the best linearity. For small input signals (large gains, i.e. above 50), the noise level in the PGA is important and the first stage of the PGA should be used.

The following figures give the non-linearity for different gain settings of the PGA, selecting the appropriate stage to get the best noise and linearity performance. Figure 17-9 shows the non-linearity when the third stage is used with a gain of 1. It is of course not very useful to use the PGA with a gain of 1 unless it is used to compensate offset. By increasing the gain, the integral non-linearity becomes even smaller since the signal in the amplifiers reduces.

Figure 17-10 shows the non-linearity for a gain of 2. Figure 17-11 shows the non-linearity for a gain of 5. Figure 17-12 shows the non-linearity for a gain of 10. By comparing these figures to Figure 17-8, it can be seen that the third stage of the PGA does not add significant integral non-linearity.

Figure 17-13 shows the non-linearity for a gain of 20 and Figure 17-14 shows the non-linearity for a gain of 50. In both cases the PGA2 is used at a gain of 10 and the remaining gain is realized by the third stage. It can be seen again that the second stage of the PGA does not add significant non-linearity.

For gains above 50, the first stage PGA1 should be selected instead of PGA2. Although the non-linearity in the first stage of the PGA is larger than in stage 2 and 3, the gain in stage 3 is now sufficiently high so that the non-linearity of the first stage does become negligible as is shown in Figure 17-15 for a gain of 100. Therefore, the first stage is preferred over the second stage since it has less noise.

Increasing the gain further up to 1000 will further increase the linearity since the signal becomes very small in the first two stages. The signal is full scale at the output of stage 3 and as shown in Figure 17-9 to Figure 17-12, this stage has very good linearity.

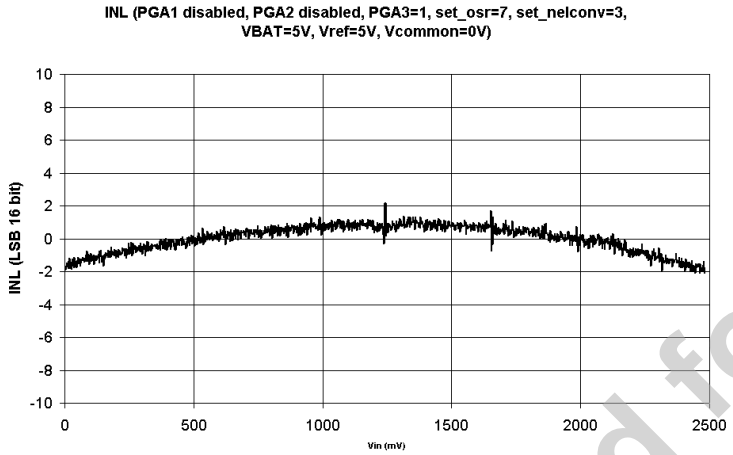


Figure 17-9. Integral non-linearity of the ADC and with gain of 1 (PGA1 and PGA2 disabled, PGA3=1, reference voltage of 5V)

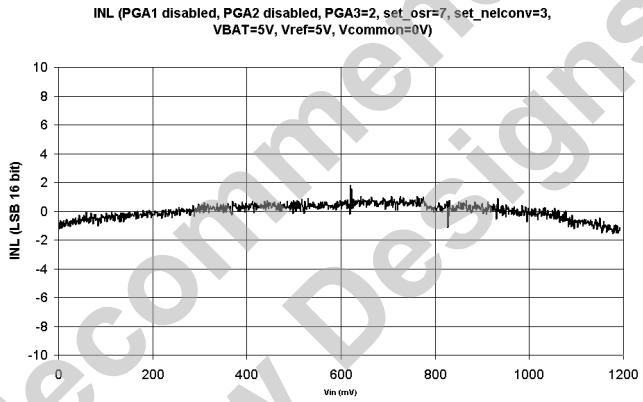


Figure 17-10. Integral non-linearity of the ADC and gain of 2 (PGA1 and PGA2 disabled, PGA3=2 reference voltage of 5V)

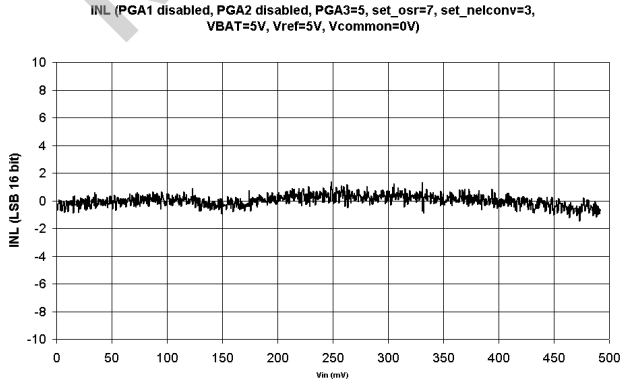


Figure 17-11. Integral non-linearity of the ADC and gain of 5 (PGA1 and PGA2 disabled, PGA3=5, reference voltage of 5V)

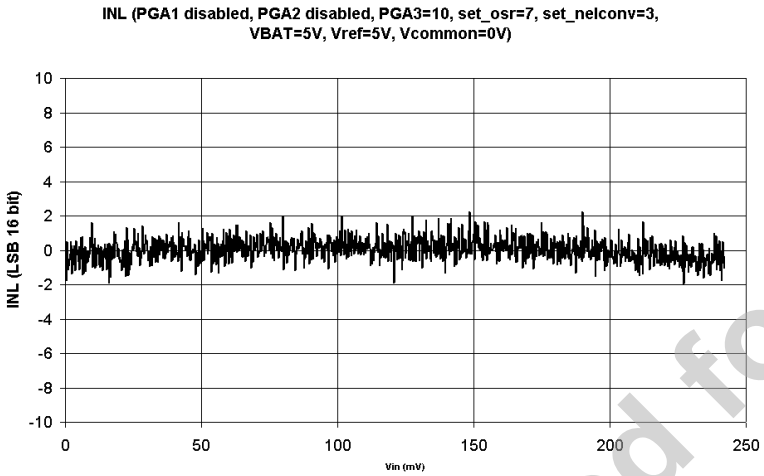


Figure 17-12. Integral non-linearity of the ADC and gain of 10 (PGA1 and PGA2 disabled, PGA3=10, reference voltage of 5V)

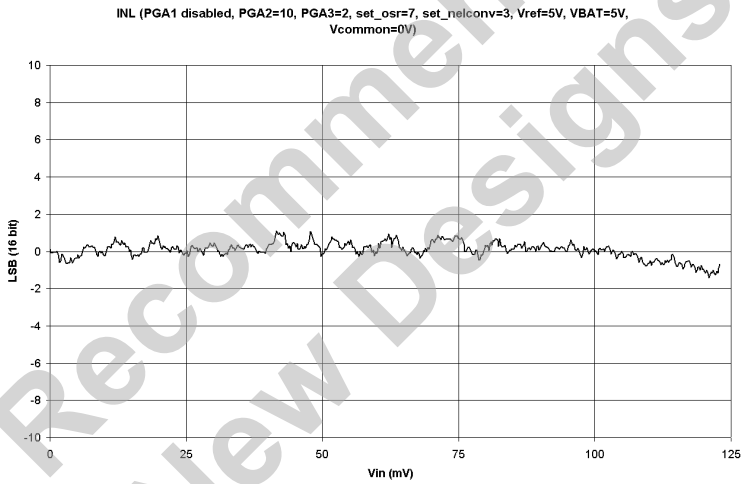


Figure 17-13. Integral non-linearity of the ADC and gain of 20 (PGA1 and PGA2=10, PGA3=2, reference voltage of 5V)

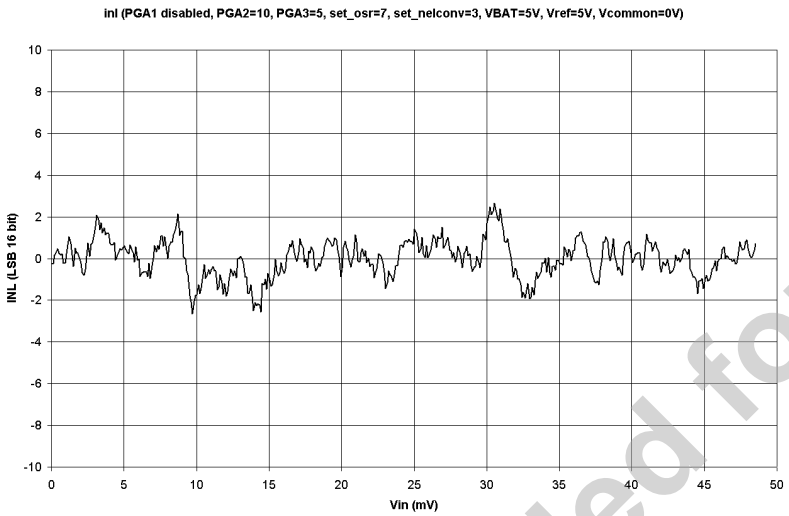


Figure 17-14. Integral non-linearity of the ADC and gain of 50 (PGA1 disabled, PGA2=10, PGA3=5, reference voltage of 5V)

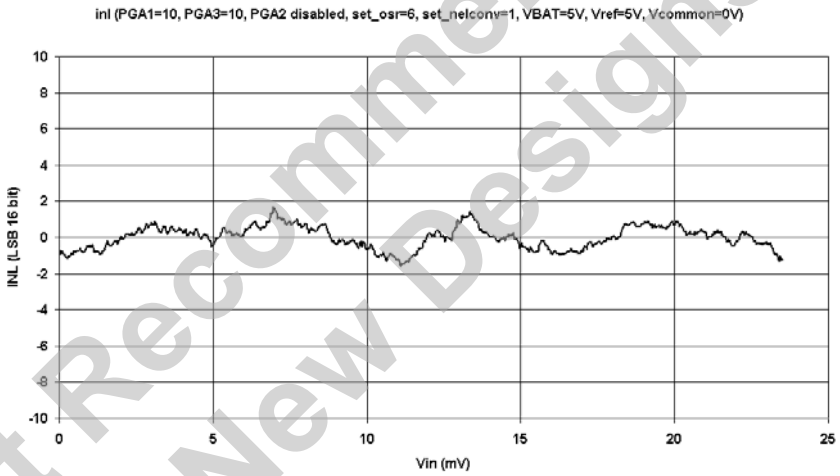


Figure 17-15. Integral non-linearity of the ADC and gain of 100 (PGA1=10 and PGA3=10, PGA2 disabled, reference voltage of 5V)

17.8.3.2 Differential non-linearity

The differential non-linearity is generated by the ADC. The PGA does not add differential non-linearity. Figure 17-16 shows the differential non-linearity.

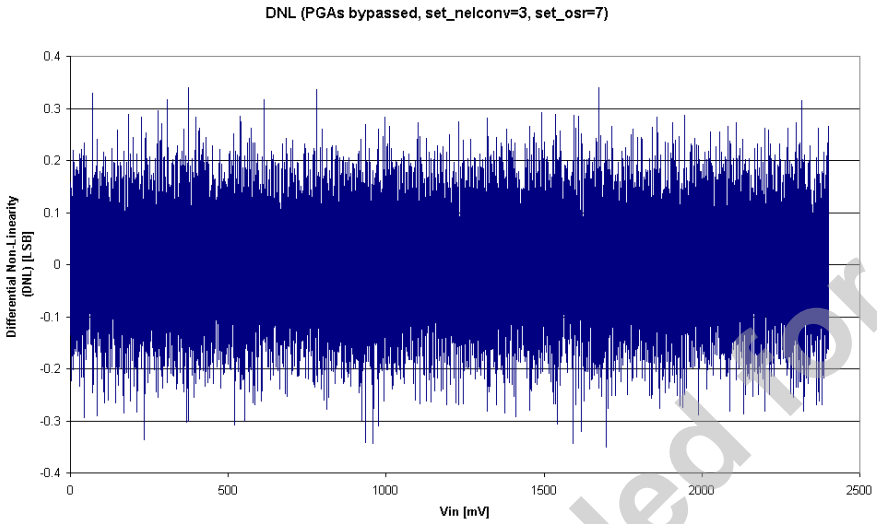


Figure 17-16. Differential non-linearity of the ADC converter.

17.8.4 Noise

Ideally, a constant input voltage V_{IN} should result in a constant output code. However, because of circuit noise, the output code may vary for a fixed input voltage. Thus, a statistical analysis on the output code of 1200 conversions for a constant input voltage was performed to derive the equivalent noise levels of PGA1, PGA2, and PGA3. The extracted rms output noise of PGA1, 2, and 3 are given in Table 17-27: standard output deviation and output rms noise voltage. Figure 17-17 shows the distribution for the ADC alone (PGA1, 2, and 3 bypassed). Quantitative noise is dominant in this case, and, thus, the ADC thermal noise is below 16 bits.

The simple noise model of Figure 17-18(a) is used to estimate the equivalent input referred rms noise $V_{N,IN}$ of the acquisition chain in the model of Figure 17-18(b). This is given by the relationship:

$$V_{N,IN}{}^2 = \frac{(V_{N1}/GD_1)^2 + (V_{N2}/(GD_1 \cdot GD_2))^2 + (V_{N3}/(GD_1 \cdot GD_2 \cdot GD_3))^2}{(OSR \cdot N_{ELCONV})} \quad (V^2rms) \qquad (Eq.\ 21)$$

where V_{N1}, V_{N2}, and V_{N3} are the output rms noise figures of Table 17-27, GD_1, GD_2, and GD_3 are the PGA gains of stages 1 to 3 respectively. As shown in this equation, noise can be reduced by increasing OSR and N_{ELCONV} (increases the ADC averaging effect, but reduces noise).

Parameter	PGA1	PGA2	PGA3
Standard deviation at ADC output (LSB)	0.85	1.4	1.5
Output rms noise (µV) [1]	205 (V_{N1})	340 (V_{N2})	365 (V_{N3})

Note: see noise model of Figure 17-18 and equation Eq. 21.

Table 17-27. PGA noise measurements (n = 16 bits, OSR = 512, N_{ELCONV} = 2, V_{REF} = 5V)

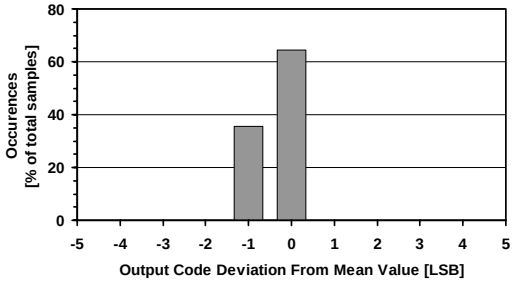


Figure 17-17. ADC noise (PGA1, 2 & 3 bypassed, OSR=512,N_{ELCONV}=2)

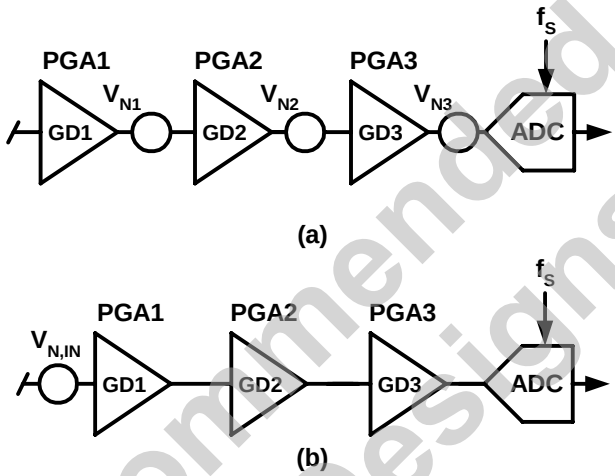


Figure 17-18. (a) Simple noise model for PGAs and ADC and (b) total input referred noise

As an example, consider the system where: GD_2 = 10 (GD_1 = 1; PGA3 bypassed), OSR = 512, N_{ELCONV} = 2, V_{REF} = 5V. In this case, the noise contribution V_{N1} of PGA1 is dominant over that of PGA2. Using equation Eq. 21, we get: $V_{N,IN}$ = 6.4µV (rms) at the input of the acquisition chain, or, equivalently, 0.85 LSB at the output of the ADC. Considering a 0.2V (rms) maximum signal amplitude, the signal-to-noise ratio is 90dB.

Implementing a software filter can also reduce noise. By taking an average on a number of subsequent measurements, the apparent noise is reduced by the square root of the number of measurement used to make the average.

17.8.5 Gain Error and Offset Error

Gain error is defined as the amount of deviation between the ideal transfer function (theoretical equation Eq. 18) and the measured transfer function (with the offset error removed).

The actual gain of the different stages can vary relating to the fabrication tolerances of the different elements. Although these tolerances are specified to a maximum of ±3%, most of the time they will be around ±0.5%. Moreover, the tolerances between the different stages are not correlated and the probability of getting the maximal error in the same direction in all stages is very low. Finally, the software can calibrate these gain errors at the same time as the gain errors of the sensor for instance.

Figure 17-19 shows gain error drift vs. temperature for different PGA gains. The curves are expressed in % of Full-Scale Range (FSR) normalized to 25°C.

Offset error is defined as the output code error for a zero volt input (ideally, output code = 0). The offset of the ADC and the PGA1 stage are completely suppressed if $N_{ELCONV} > 1$.

The measured offset drift vs. temperature curves for different PGA gains are depicted in Figure 17-20. The output offset error, expressed in LSB for 16-bit setting, is normalized to 25°C. Notice that if the ADC is used alone, the output offset error is below ±1 LSB and has no drift.

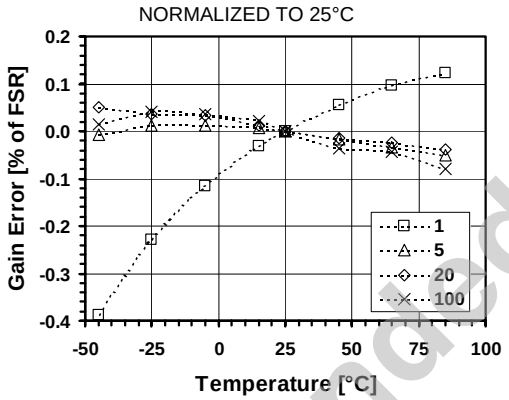


Figure 17-19. Gain error vs. temperature for different PGA gains

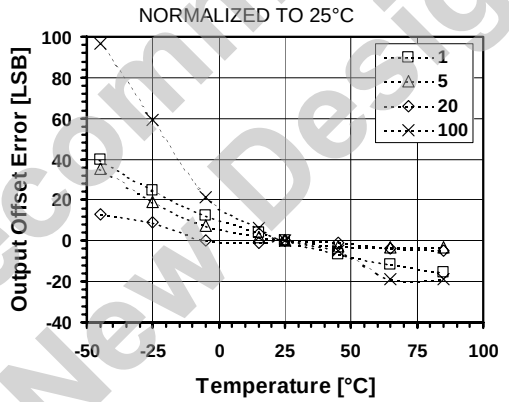


Figure 17-20. Offset error vs. temperature for different PGA gains

17.8.6 Power Consumption

Figure 17-21 plots the variation of quiescent current consumption with supply voltage V_{DD}, as well as the distribution between the 3 PGA stages and the ADC (see Table 17-28). As shown in Figure 17-22, if lower sampling frequency is used, the quiescent current consumption can be lowered by reducing the bias currents of the PGAs and the ADC with registers `IB_AMP_PGA [1:0]` and `IB_AMP_ADC [1:0]`. (In Figure 17-22, `IB_AMP_PGA/ADC[1:0]` = '11', '10', '00' for f_S = 500, 250, 62.5kHz respectively.)

Quiescent current consumption vs. temperature is depicted in Figure 17-23, showing a relative increase of nearly 40% between -45 and +85°C. Figure 17-24 shows the variation of quiescent current consumption for different frequency settings of the internal RC oscillator. It can be seen that the quiescent current varies by about 20% between 100kHz and 2MHz.

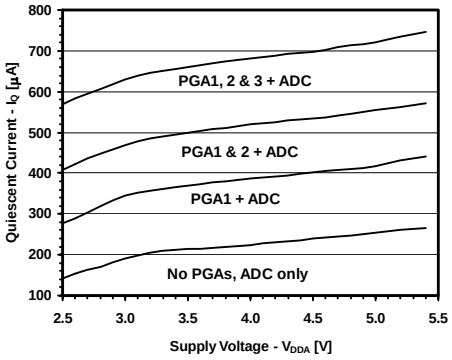


Figure 17-21. Quiescent current consumption vs. supply voltage

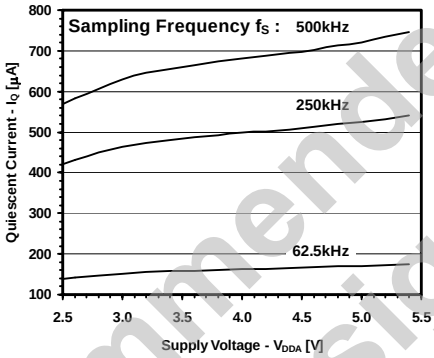
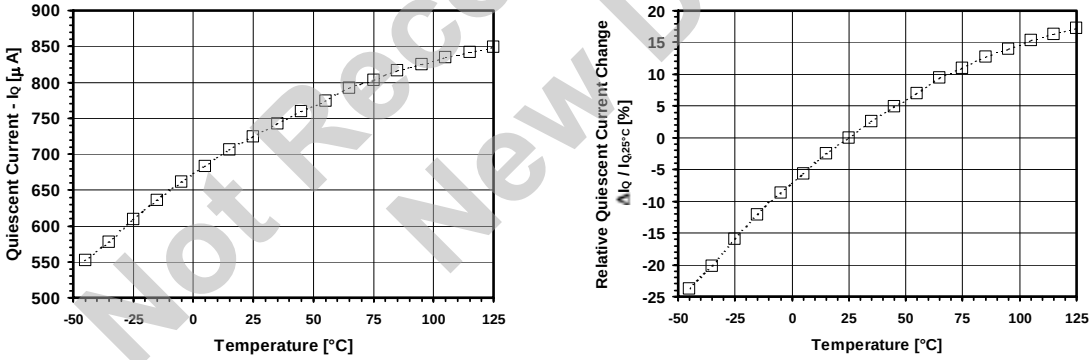


Figure 17-22. Quiescent current consumption vs. supply voltage for different sampling frequencies



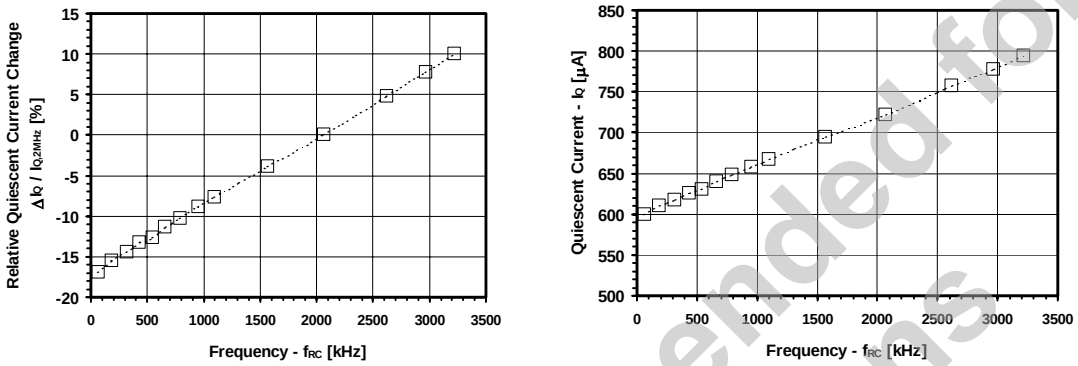
(a) **(b)**

Figure 17-23. (a) Absolute and (b) relative change inquiescent current consumption vs. temperature

Supply	ADC	PGA1	PGA2	PGA3	TOTAL	Unit
V_{DD} = 5V	250	165	130	175	720	µA
V_{DD} = 3V	190	150	120	160	620	µA

Table 17-28. Typical quiescent current distributions in acquisition chain

(n = 16 bits, f_S = 500kHz)



(a) **(b)**

Figure 17-24. (a) Absolute and (b) relative change in quiescent curent consumption vs. RC oscillator frequency (all PGAs active, V_{DD} = 5V)

17.8.7 Power Supply Rejection Ratio

Figure 17-25 shows power supply rejection ratio (PSRR) at a 3V and a 5V supply voltage, and for various PGA gains. PSRR is defined as the ratio (in dB) of voltage supply change (in V) to the change in the converter output (in V). PSRR depends on both PGA gain and supply voltage V_{DD}.

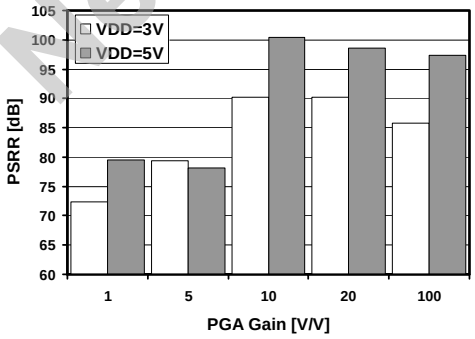


Figure 17-25. Power supply rejection ratio (PSRR)

Supply	GAIN = 1	GAIN =5	GAIN = 10	GAIN = 20	GAIN =100	Unit
V_{DD} = 5V	79	78	100	99	97	dB
V_{DD} = 3V	72	79	90	90	86	dB

Table 17-29. PSRR (n = 16 bits, V_{IN} = V_{REF} = 2.5V, f_S = 500kHz)

17.9 Application Hints

17.9.1 Input Impedance

The PGAs of the acquisition chain employ switched-capacitor techniques. For this reason, while a conversion is done, the input impedance on the selected channel of the PGAs is inversely proportional to the sampling frequency f_S and to stage gain as given in equation 22.

$$Z_{in} \geq \frac{768 \cdot 10^9 \, \Omega Hz}{f_s \cdot gain} \quad \text{(Eq. 22)}$$

The input impedance observed is the input impedance of the first PGA stage that is enabled or the input impedance of the ADC if all three stages are disabled.

PGA1 (with a gain of 10), PGA2 (with a gain of 10) and PGA3 (with a gain of 10) each have a minimum input impedance of 150kΩ at f_S = 512kHz (see Specification Table). Larger input impedance can be obtained by reducing the gain and/or by reducing the sampling frequency. Therefore, with a gain of 1 and a sampling frequency of 100kHz, Z_{in} > 7.6MΩ.

The input impedance on channels that are not selected is very high (>100MΩ).

17.9.2 PGA Settling or Input Channel Modifications

PGAs are reset after each writing operation to registers **RegAcCfg1-5**. Similarly, input channels are switched after modifications of **AMUX[4:0]** or **VMUX**. To ensure precise conversion, the ADC must be started after a PGA or inputs common-mode stabilization delay. This is done by writing bit **START** several cycles after modification of PGA settings or channel switching. Delay between PGA start or input channel switching and ADC start should be equivalent to *OSR* (between 8 and 1024) number of cycles. This delay does not apply to conversions made without the PGAs.

If the ADC is not settled within the specified period, there is probably an input impedance problem (see previous section).

17.9.3 PGA Gain & Offset, Linearity and Noise

Hereafter are a few design guidelines that should be taken into account when using the ZoomingADC™:

1) Keep in mind that increasing the overall PGA gain, or "zooming" coefficient, improves linearity but degrades noise performance.
2) Use the minimum number of PGA stages necessary to produce the desired gain ("zooming") and offset. Bypass unnecessary PGAs.
3) For high gains (>50), use PGA stage 1. For low gains (<50) use stages 2 and 3.
4) For the lowest noise, set the highest possible gain on the first (front) PGA stage used in the chain. For example, in an application where a gain of 20 is needed, set the gain of PGA2 to 10, set the gain of PGA3 to 2.
4) For highest linearity and lowest noise performance, bypass all PGAs and use the ADC alone (applications where no "zooming" is needed); i.e. set **ENABLE[3:0]** = '0001'.
5) For low-noise applications where power consumption is not a primary concern, maintain the largest bias currents in the PGAs and in the ADC; i.e. set **IB_AMP_PGA[1:0]** = **IB_AMP_ADC[1:0]** = '11'.
6) For lowest output offset error at the output of the ADC, bypass PGA2 and PGA3. Indeed, PGA2 and PGA3 typically introduce an offset of about 5 to 10 LSB (16 bit) at their output. Note, however, that the ADC output offset is easily calibrated out by software.

17.9.4 Frequency Response

The incremental ADC is an over-sampled converter with two main blocks: an analog modulator and a low-pass digital filter. The main function of the digital filter is to remove the quantitative noise introduced by the modulator. As shown in Figure 17-26, this filter determines the frequency response of the transfer function between the output of the ADC and the analog input V_{IN}. Notice that the frequency axes are normalized to one elementary conversion period OSR/f_S. The plots of Figure 17-26 also show that the frequency response changes with the number of elementary conversions N_{ELCONV} performed. In particular, notches appear for $N_{ELCONV} \geq 2$. These notches occur at:

$$f_{NOTCH}(i) = \frac{i \cdot f_S}{OSR \cdot N_{ELCONV}} \quad \text{(Hz)} \quad \text{for } i = 1,2,...,(N_{ELCONV} - 1) \quad \text{(Eq. 23)}$$

and are repeated every f_S/OSR.

Information on the location of these notches is particularly useful when specific frequencies must be filtered out by the acquisition system. For example, consider a 5Hz-bandwidth, 16-bit sensing system where 50Hz line rejection is needed. Using the above equation and the plots below, we set the 4th notch for N_{ELCONV} = 4 to 50Hz, i.e. $1.25 \cdot f_S/OSR$ = 50Hz. The sampling frequency is then calculated as f_S = 20.48kHz for OSR = 512. Notice that this choice yields also good attenuation of 50Hz harmonics.

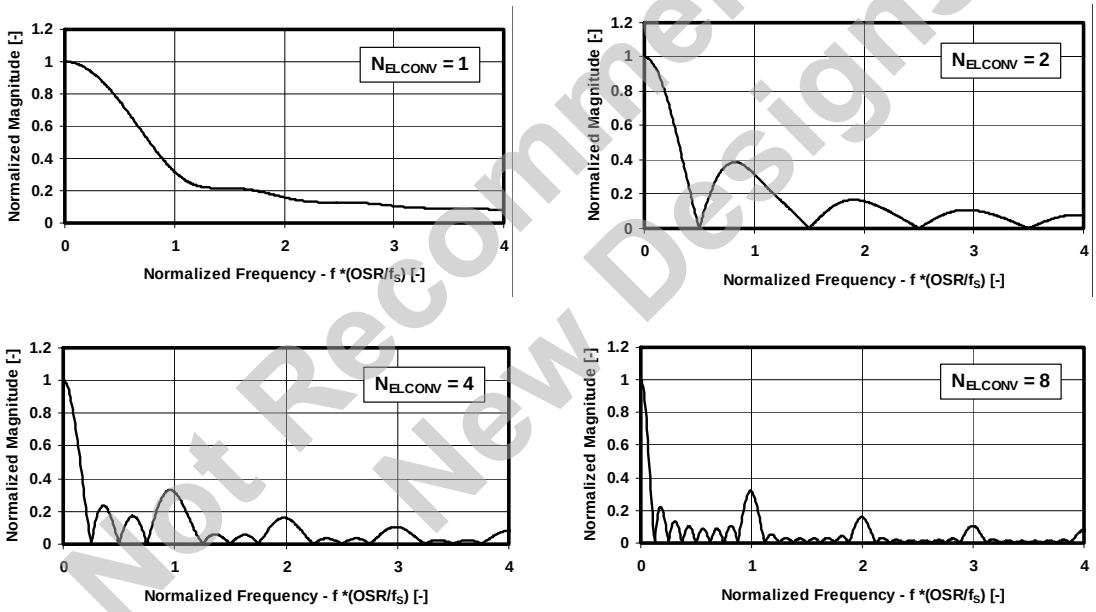


Figure 17-26. Frequency response: normalized magnitude vs. frequency for different N_{ELCONV}

17.9.5 Power Reduction

The ZoominADC™ is particularly well suited for low-power applications. When very low power consumption is of primary concern, such as in battery operated systems, several parameters can be used to reduce power consumption as follows:

1) Operate the acquisition chain with a reduced supply voltage V_{DD}.
2) Disable the PGAs which are not used during analog-to-digital conversion with **`ENABLE[3:0]`**.
3) Disable all PGAs and the ADC when the system is idle and no conversion is performed.

4) Use lower bias currents in the PGAs and the ADC using the control words **`IB_AMP_PGA[1:0]`** and **`IB_AMP_ADC[1:0]`**. (This reduces the maximum sampling frequency according to Table 17-26.)
5) Reduce internal RC oscillator frequency and/or sampling frequency.

Finally, remember that power reduction is typically traded off with reduced linearity, larger noise and slower maximum sampling speed.

18. Voltage Multiplier

18.1 Features

- Generates a voltage that is higher or equal to the supply voltage, but at maximum 4.8 V
- Can be easily enabled or disabled

18.2 Overview

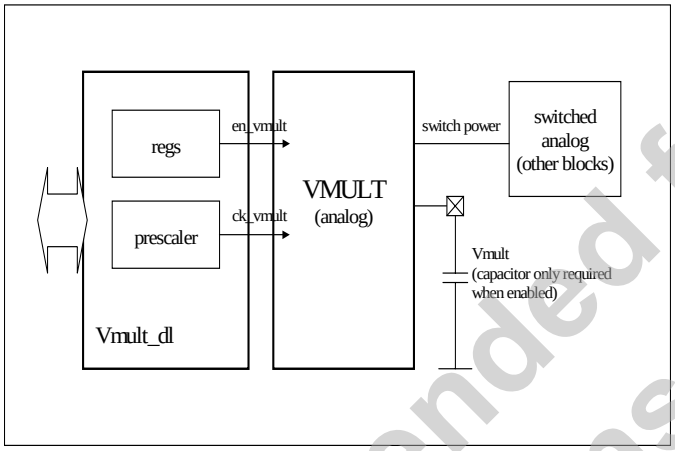


Figure 17-1 : Structure of the Vmult peripheral

The Vmult block generates a voltage (called "Vmult") that is higher or equal to the supply voltage. This output voltage is intended for use in analog switch drivers, for example in the ADC and PGA block.

- In normal use its value is 4.8 V, this voltage is available on the pad Vmult.
- When the main voltage is below 2.6 V, Vmult is twice the main voltage minus 0.4 V.
- When the main voltage is above 4.8 V, Vmult remains 4.8 V but the internal voltage intended for the analog switch drivers equal the main voltage.

The voltage multiplier should be on (bit **Enable** in **RegVmultCfg0**) when using switched analog blocks, like ADC, DAC or analog properties of Port B when Vbat is below 3V. The source clock of Vmult is selected from **Fin[1:0]** in **RegVmultCfg0**. It is strongly recommended to use the same **Fin** bit code as in the ADC.

The external capacitor on the pin VMULT has to be connected if the block is enabled. The size of the capacitor has to be 2nF ± 50%.

18.3 Control part Registers

There is only one register in the Vmult. Table 18-1 describes the bits in the register.

Pos.	RegVmultCfg0	rw	Reset	Function
2	Enable	rw	0	enable of the vmult '1' : enabled '0' : disabled
1-0	Fin	rw	0	system clock division factor '00' : 1/2, '01' : 1/4, '10' : 1/16, '11' : 1/64

Table 18-1. RegVmultCfg0

19. LCD Driver

19.1 Features

- Supports direct drive for 32 segments.
- Supports multiplexed drive for up to 120 segments
- Multiplex 1/2, 1/3 and 1/4.
- Possibility to use pads as an input/output port or as LCD driver pin.
- Multiple frame frequencies.
- Sleep mode.
- On chip low-power voltage generation.
- LCD driver voltage is independent from the circuit supply voltage.
- Parallel IO with voltage different from the circuit supply voltage.

19.2 Overview

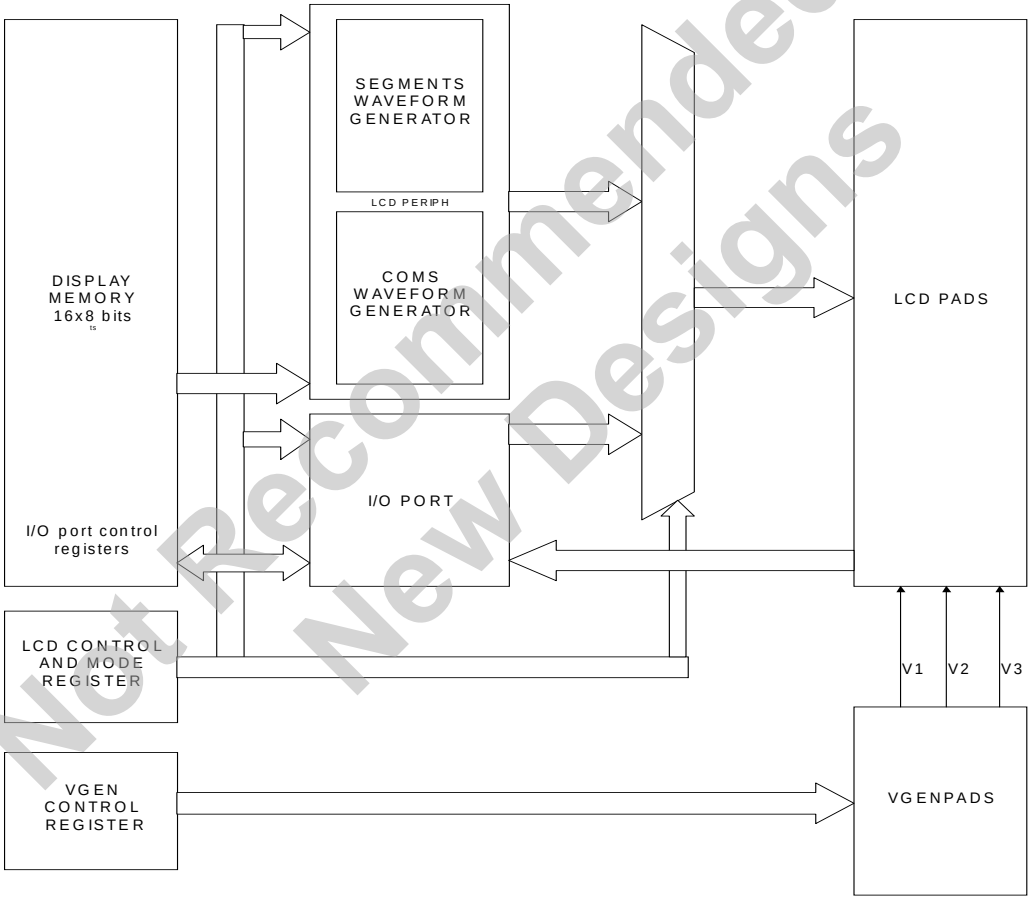


Figure 19-1: General Structure

The LCD driver generates all waveforms to drive the display. The user has only to set a 0 (segment off) or a 1 (segment on) in the bit location corresponding to the segment. The LCD driver supports the wave form generation for different multiplexing schemes: direct drive (no multiplexing), multiplexing by 2, by 3 and by 4. The frame frequency is software programmable. Low power on-chip voltage generation is provided for each of the different multiplexing schemes. The LCD driver voltage is completely independent from the circuit supply voltage: the LCD driver voltage can be below or above the circuit supply voltage. All or part of the driver can be configured as a

general-purpose parallel IO-port. When used as a parallel port, it can be connected to circuits with a different supply voltage.

Section 19.4 describes the basic functions of the LCD driver using the on-chip voltage generator. Section 19.5 describes more advanced the features of the LCD driver and especially different ways of generating the voltage for the LCD. Section 19.6 describes how to use the peripheral as a general-purpose parallel IO-port. Section 19.7 shows how to partition when used as a partial LCD driver and partial IO port. Finally, the electrical specifications of the driver are given in section 19.8.

19.3 Register map

There are thirty-six registers in the LCD driver, namely **RegVgenCfg0**, **RegLcdOn**, **RegLcdSe**, **RegLcdClkFrame**, **RegLcdDataN** (0≤N≤15), **RegPLcdInN** (0≤N≤3), **RegPLcdOutN** (0≤N≤3), **RegPLcdDirN** (0≤N≤3) and **RegPLcdPullupN** (0≤N≤3). Table 19-2 to Table 19-11 show the mapping of control bits and functionality of these registers while Table 19-1 gives the default address of these thirty-six registers.

Register name
Vgen registers
RegVgenCfg0
Control registers
RegLcdOn
RegLcdSe
RegLcdClkFrame
LCD registers
RegLcdData0
RegLcdData1
RegLcdData2
RegLcdData3
RegLcdData4
RegLcdData5
RegLcdData6
RegLcdData7
RegLcdData8
RegLcdData9
RegLcdData10
RegLcdData11
RegLcdData12
RegLcdData13
RegLcdData14
RegLcdData15
Port Registers
RegPLcdOut0
RegPLcdOut1
RegPLcdOut2
RegPLcdOut3
RegPLcdDir0
RegPLcdDir1
RegPLcdDir2
RegPLcdDir3
RegPLcdPull0
RegPLcdPull1
RegPLcdPull2
RegPLcdPull3
RegPLcdIn0
RegPLcdIn1
RegPLcdIn2
RegPLcdIn3

Table 19-1: LCD registers default addresses

pos.	RegVgenCfg0	RW	reset	function
7-6	--	r	00	Unused
5-4	VgenClkSel	r w	10 nresetglobal	Voltage generator frequency selection. 00 – 256 Hz 01 – 512 Hz 10 – 1 kHz 11 – 2 kHz
3	VgenOff	r w	1 nresetglobal	Voltage generator disable signal. 0 – enabled 1 – disabled
2	VgenMode	r w	0 nresetglobal	Mode selection. 0 – 1/3 bias 1 – 1/2 bias
1	VgenStdb	r w	0 nresetglobal	Stand-by mode
0	VgenRefEn	r w	0 nresetglobal	Internal 1.2V voltage reference enable. 0 – disabled 1 – enabled

Table 19-2: RegVgenCfg0

pos.	RegLcdOn	Rw	Reset	Function
7-3		-	00000	
2	LcdSleep	r w	1 nresetglobal	1 = stop operating 0 = continue operating
1-0	LcdMux	r w	00 nresetglobal	00 = direct drive 01 = 1:2 mux drive 10 = 1:3 mux drive 11 = 1:4 mux drive

Table 19-3: RegLcdOn

pos.	RegLcdSe	rw	Reset	Function
7	LcdSe3	r w	1 nresetglobal	1 = pins pad_lcd_io(0-3) have LCD drive function 0 = pins pad_lcd_io(0-3) have digital function
6	LcdSe7	r w	1 nresetglobal	1 = pins pad_lcd_io(4-7) have LCD drive function 0 = pins pad_lcd_io(4-7) have digital function
5	LcdSe11	r w	1 nresetglobal	1 = pins pad_lcd_io(8-11) have LCD drive function 0 = pins pad_lcd_io(8-11) have digital function
4	LcdSe15	r w	1 nresetglobal	1 = pins pad_lcd_io(12-15) have LCD drive function 0 = pins pad_lcd_io(12-15)have digital function
3	LcdSe19	r w	1 nresetglobal	1 = pins pad_lcd_io(16-19) have LCD drive function 0 = pins pad_lcd_io(16-19) have digital function
2	LcdSe23	r w	1 nresetglobal	1 = pins pad_lcd_io(20-23) have LCD drive function 0 = pins pad_lcd_io(20-23) have digital function
1	LcdSe27	r w	1 nresetglobal	1 = pins pad_lcd_io(24-27) have LCD drive function 0 = pins pad_lcd_io(24-27) have digital function
0	LcdSe31	r w	1 nresetglobal	1 = pins pad_lcd_io(28-31) have LCD drive function 0 = pins pad_lcd_io(28-31) have digital function

Table 19-4: RegLcdSe

pos.	RegLcdClkFrame	rw	reset	Function
7-5	LcdDivFreq	r w	000 nresetglobal	000 = LcdFreq 001 = LcdFreq / 2 010 = LcdFreq / 3 011 = LcdFreq / 4 100 = LcdFreq / 5 101 = LcdFreq / 6 110 = LcdFreq / 7 111 = LcdFreq / 8
4-2	Reserved	-	000	
1-0	LcdFreq	r w	00 nresetglobal	00 = 512 Hz 01 = 1024 Hz 10 = 2048 Hz 11 = 4096 Hz

Table 19-5: RegLcdClkFrame

Pos.	RegLcdDataN	Rw	Reset	Description	Map Pin
7	LcdDataN[7]	rw	0 nresetpconf	Segment[3] value in 1:4 MUX	pad_lcd_io[2N+1]
6	LcdDataN[6]	rw	0 nresetpconf	Segment[2] value in 1:3 MUX	
5	LcdDataN[5]	rw	0 nresetpconf	Segment[1] value in 1:2 MUX	
4	LcdDataN[4]	rw	0 nresetpconf	Segment[0] value in DD	
3	LcdDataN[3]	rw	0 nresetpconf	Segment[3] value in 1:4 MUX	pad_lcd_io[2N]
2	LcdDataN[2]	rw	0 nresetpconf	Segment[2] value in 1:3 MUX	
1	LcdDataN[1]	rw	0 nresetpconf	Segment[1] value in 1:2 MUX	
0	LcdDataN[0]	rw	0 nresetpconf	Segment[0] value in DD	

Table 19-6: **RegLcdDataN** with 0≤N≤14

Pos.	RegLcdData15	Rw	Reset	Description	Map Pin
7	LcdData15[7]	rw	0 nresetpconf	--	pad_lcd_io[31]
6	LcdData15[6]	rw	0 nresetpconf	--	
5	LcdData15[5]	rw	0 nresetpconf	Segment[1] value in 1:2 MUX	
4	LcdData15[4]	rw	0 nresetpconf	Segment[0] value in DD	
3	LcdData15[3]	rw	0 nresetpconf	--	pad_lcd_io[30]
2	LcdData15[2]	rw	0 nresetpconf	Segment[2] value in 1:3 MUX	
1	LcdData15[1]	rw	0 nresetpconf	Segment[1] value in 1:2 MUX	
0	LcdData15[0]	rw	0 nresetpconf	Segment[0] value in DD	

Table 19-7: **RegLcdData15**

Pos.	RegPLcdInN	Rw	Reset	Description
7	PLcdInN[7]	r	x	pad_lcd_io[8N+7] input value
6	PLcdInN[6]	r	x	pad_lcd_io[8N+6] input value
5	PLcdInN[5]	r	x	pad_lcd_io[8N+5] input value
4	PLcdInN[4]	r	x	pad_lcd_io[8N+4] input value
3	PLcdInN[3]	r	x	pad_lcd_io[8N+3] input value
2	PLcdInN[2]	r	x	pad_lcd_io[8N+2] input value
1	PLcdInN[1]	r	x	pad_lcd_io[8N+1] input value
0	PLcdInN[0]	r	x	pad_lcd_io[8N+0] input value

Table 19-8: **RegPLcdInN** with 0≤N≤3

Pos.	RegPLcdOutN	Rw	Reset	Description
7	PLcdOuN[7]	r w	0 nresetpconf	pad_lcd_io[8N+7] output value
6	PLcdOutN[6]	r w	0 nresetpconf	pad_lcd_io[8N+6] output value
5	PLcdOutN[5]	r w	0 nresetpconf	pad_lcd_io[8N+5] output value
4	PLcdOutN[4]	r w	0 nresetpconf	pad_lcd_io[8N+4] output value
3	PLcdOutN[3]	r w	0 nresetpconf	pad_lcd_io[8N+3] output value
2	PLcdOutN[2]	r w	0 nresetpconf	pad_lcd_io[8N+2] output value
1	PLcdOutN[1]	r w	0 nresetpconf	pad_lcd_io[8N+1] output value
0	PLcdOutN[0]	r w	0 nresetpconf	pad_lcd_io[8N+0] output value

Table 19-9: **RegPLcdOutN** with 0≤N≤3

Pos.	RegPLcdDirN	Rw	Reset	Description
7	PLcdDirN[7]	r w	0 nresetpconf	pad_lcd_io[8N+7] direction (0=input)
6	PLcdDirN[6]	r w	0 nresetpconf	pad_lcd_io[8N+6] direction (0=input)
5	PLcdDirN[5]	r w	0 nresetpconf	pad_lcd_io[8N+5] direction (0=input)
4	PLcdDirN[4]	r w	0 nresetpconf	pad_lcd_io[8N+4] direction (0=input)
3	PLcdDirN[3]	r w	0 nresetpconf	pad_lcd_io[8N+3] direction (0=input)
2	PLcdDirN[2]	r w	0 nresetpconf	pad_lcd_io[8N+2] direction (0=input)
1	PLcdDirN[1]	r w	0 nresetpconf	pad_lcd_io[8N+1] direction (0=input)
0	PLcdDirN[0]	r w	0 nresetpconf	pad_lcd_io[8N+0] direction (0=input)

Table 19-10: RegPLcdDirN with 0≤N≤3

Pos.	RegPLcdPullupN	Rw	Reset	Description
7	PLcdPullupN[7]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+7] (1=active)
6	PLcdPullupN[6]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+6] (1=active)
5	PLcdPullupN[5]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+5] (1=active)
4	PLcdPullupN[4]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+4] (1=active)
3	PLcdPullupN[3]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+3] (1=active)
2	PLcdPullupN[2]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+2] (1=active)
1	PLcdPullupN[1]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+1] (1=active)
0	PLcdPullupN[0]	r w	0 nresetpconf	pullup for pad_lcd_io[8N+0] (1=active)

Table 19-11: RegPLcdPullupN with 0≤N≤3

19.4 Basic LCD capabilities

This section shows how to use the LCD driver. For each multiplexing configuration it gives a basic example explaining how to set-up the driver, the generated waveforms, how to connect the display and how to use the on-chip voltage generator. Other connection schemes and more advanced control functions will be given in the next section.

The LCD module generates the timing control to drive a static or multiplexed LCD panel, with support for up to 32 segment lines multiplexed with up to four common lines. The table below summarises the multiplexing scheme for LCD operation.

Multiplex scheme	Max #segments
direct drive (DD)	32
1:2	32 x 2
1:3	31 x 3
1:4	30 x 4

Table 19-12: Multiplexing Scheme

19.4.1 Direct Drive Mode

With direct drive mode, each pad_lcd_io pin drives one segment of the display. In this mode, up to 32 segments of the display can be connected directly to the pad_lcd_io[31:0] pins. The common (or backplane) node is to be connected to pad_lcd_com[0]. This connection is shown in Figure 19-2.

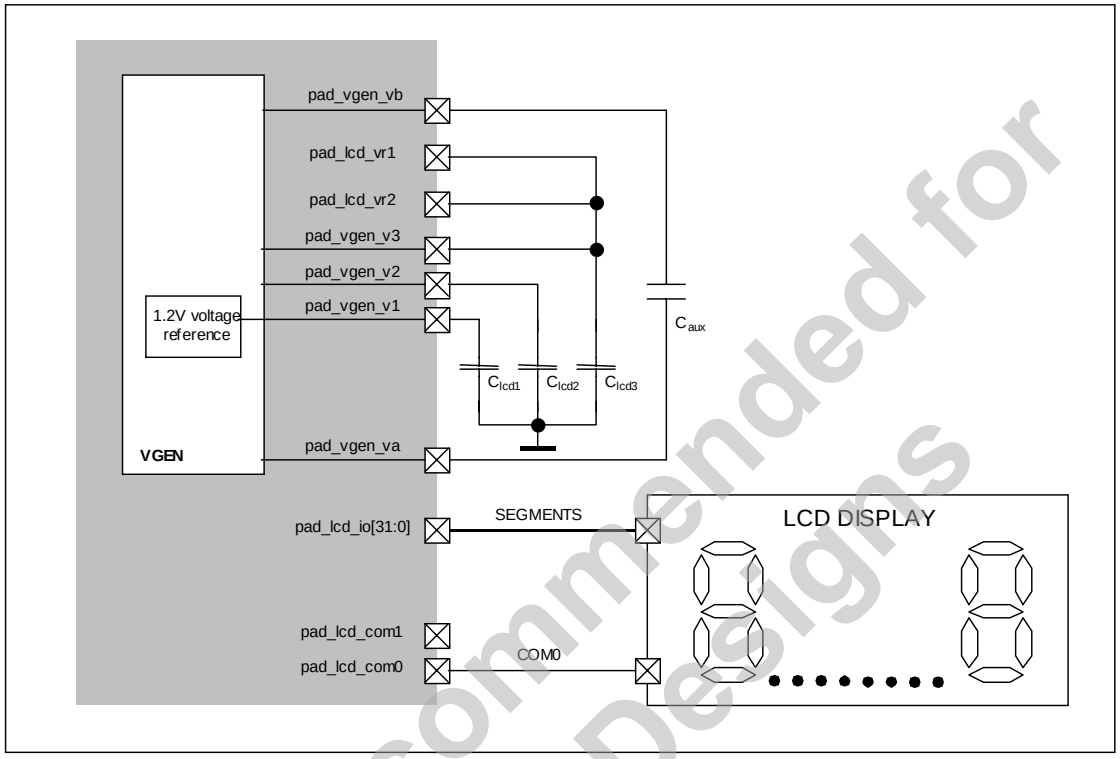


Figure 19-2: LCD with direct drive and on-chip voltage generation.

Lots of different configurations for the voltage generation can be used. Figure 19-2 shows the connections and external elements that are required if the display is to be driven from V3=3.6V independently from the circuit supply voltage. Other possible configurations are given in section 19.5.4. The recommended value for the external elements is 470nF (see also section 19.5.4.1).

To operate the driver in this configuration, the registers should be loaded with the values in Table 19-13.

Register	Contents[7:0]
RegVgenCfg0	xx110001
RegLcdOn	xxxxx000
RegLcdSe	11111111
RegLcdClkFrame	100xxx00

Table 19-13. Register contents for direct drive example

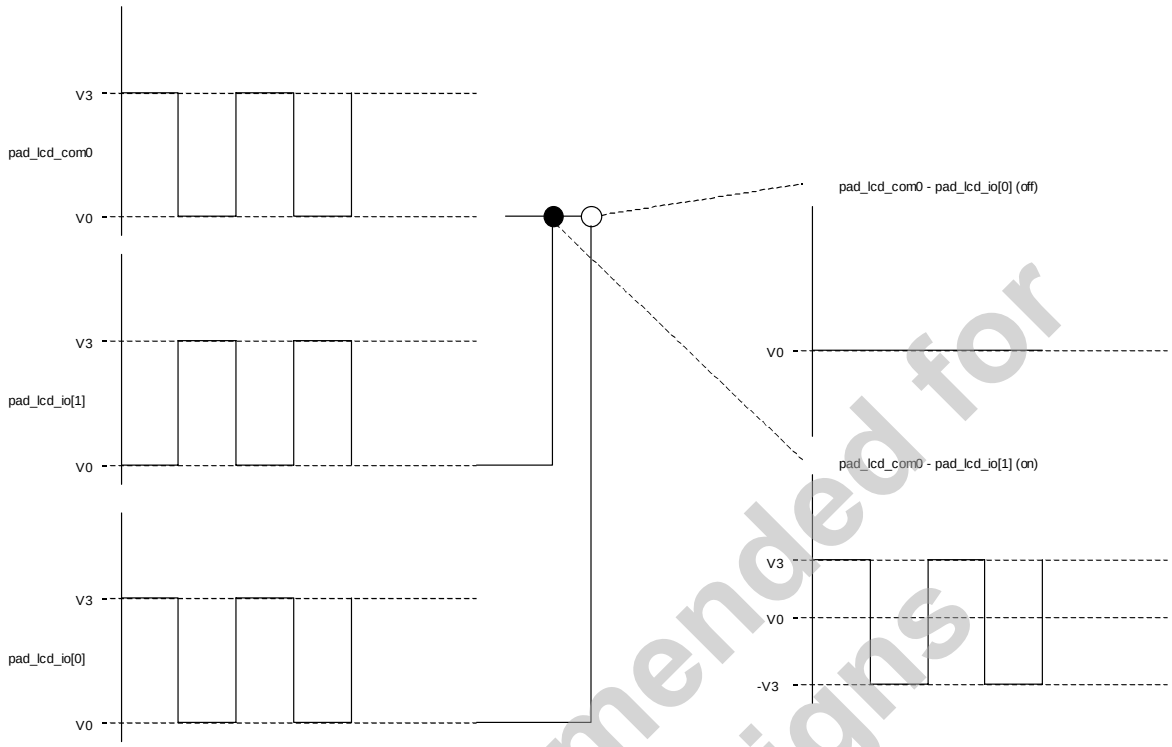


Figure 19-3: Direct Drive mode waveforms

The voltage generator has to be configured for the use of the internal reference (**VgenRefEn**=1 in **RegVgenCfg0**). The generator has to run (**VgenOff**=0 and **VgenStdb**=0 in **RegVgenCfg0**). It has to multiply by 3 (**VgenMode**=0 in **RegVgenCfg0)** and the generator output impedance is set to minimum (**VgenClkSel**=11 in **RegVgenCfg0)**.

The LCD driver is set to direct drive (**LcdMux**=00 in **RegLcdOn**) and the waveform generator is enabled (**LcdSleep**=0 in **RegLcdOn**). All pads are set into the LCD driver mode by setting all bits of register **RegLcdSe** to 1. To select a frame rate of about 50Hz, set the bits **LcdFreq**=00 and **LcdDivFreq**=100 in **RegLcdClkFrame**. Note that the precision of the frame frequency depends on the selected clock source (see clock block documentation).

The 32 segments are on or off depending on the bits set in the registers **RegLcdDataN**. Only the bits 0 and 4 of these registers are used in direct drive. All other bits are don't care. Figure 19-3 shows the generated waveforms for two segments. Segment0 is off (**LcdData0[0]**=0 in **RegLcdData0**) and segment1 is on (**LcdData0[4]**=1 in **RegLcdData0**). The figure shows on the left side the waveforms on the circuit pins, in the middle the segment status and on the right the voltage on the segment (difference between the pad_lcd_io pin and pad_lcd_com0 pin).

19.4.2 1:2 Multiplex scheme

With the 1:2 multiplex scheme, each pad_lcd_io pin drives two segments of the display. The segments of the display are connected to the pad_lcd_io[31:0] pins. The common (or backplane) nodes are to be connected to pad_lcd_com0 and pad_lcd_com1. This connection is shown in Figure 19-4.

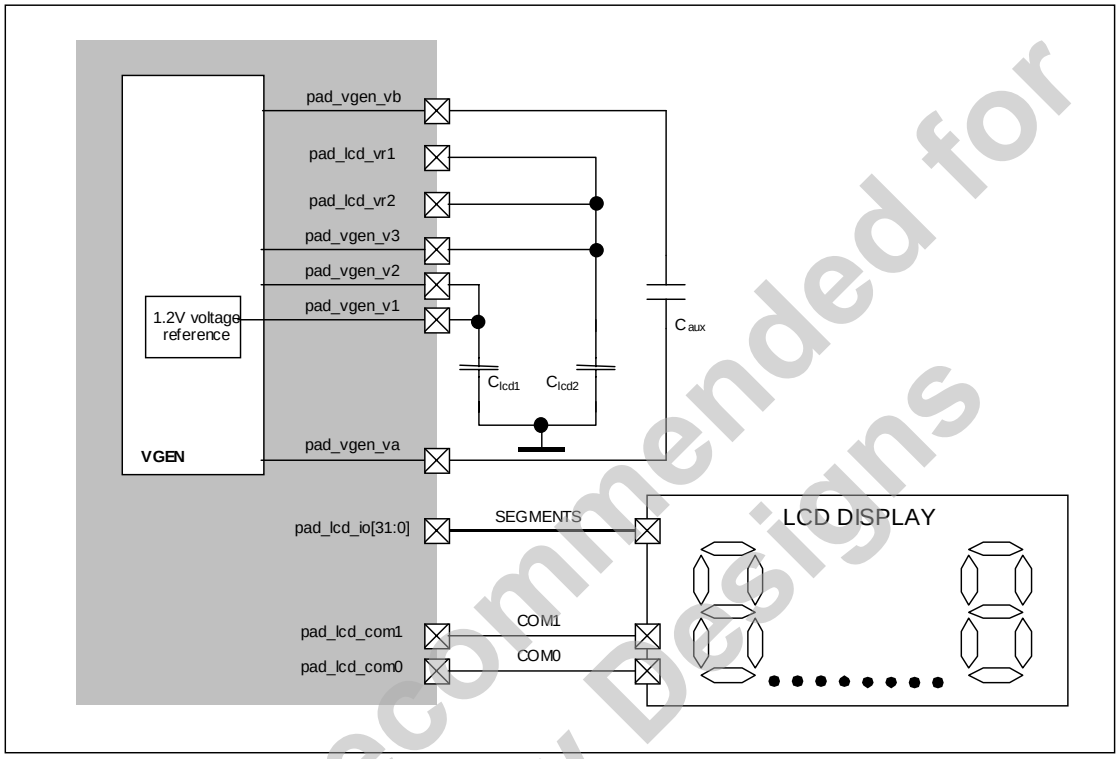


Figure 19-4. LCD with 1:2 multiplexing and on-chip 1/2 bias voltage generation.

Lots of different configurations for the voltage generation can be used. Figure 19-4 shows the connections and external elements that are required if the display is to be driven from V2=1.2V and V3=2.4V independently from the circuit supply voltage. Other possible configurations are given in section 19.5.4. Recommended values for the external capacitors are 470nF.

To operate the driver in this configuration, the registers should be loaded with the values in Table 19-14.

Register	Contents[7:0]
RegVgenCfg0	xx110101
RegLcdOn	xxxxx001
RegLcdSe	11111111
RegLcdClkFrame	100xxx01

Table 19-14. Register contents for 1:2 mux example

The voltage generator has to be configured for the use of the internal reference (**VgenRefEn**=1 in **RegVgenCfg0**). The generator has to run (**VgenOff**=0 and **VgenStdb**=0 in **RegVgenCfg0**). It has to be in 1/2 bias (**VgenMode**=1 in **RegVgenCfg0)** and the generator output impedance is set to minimum (**VgenClkSel**=11 in **RegVgenCfg0)**.

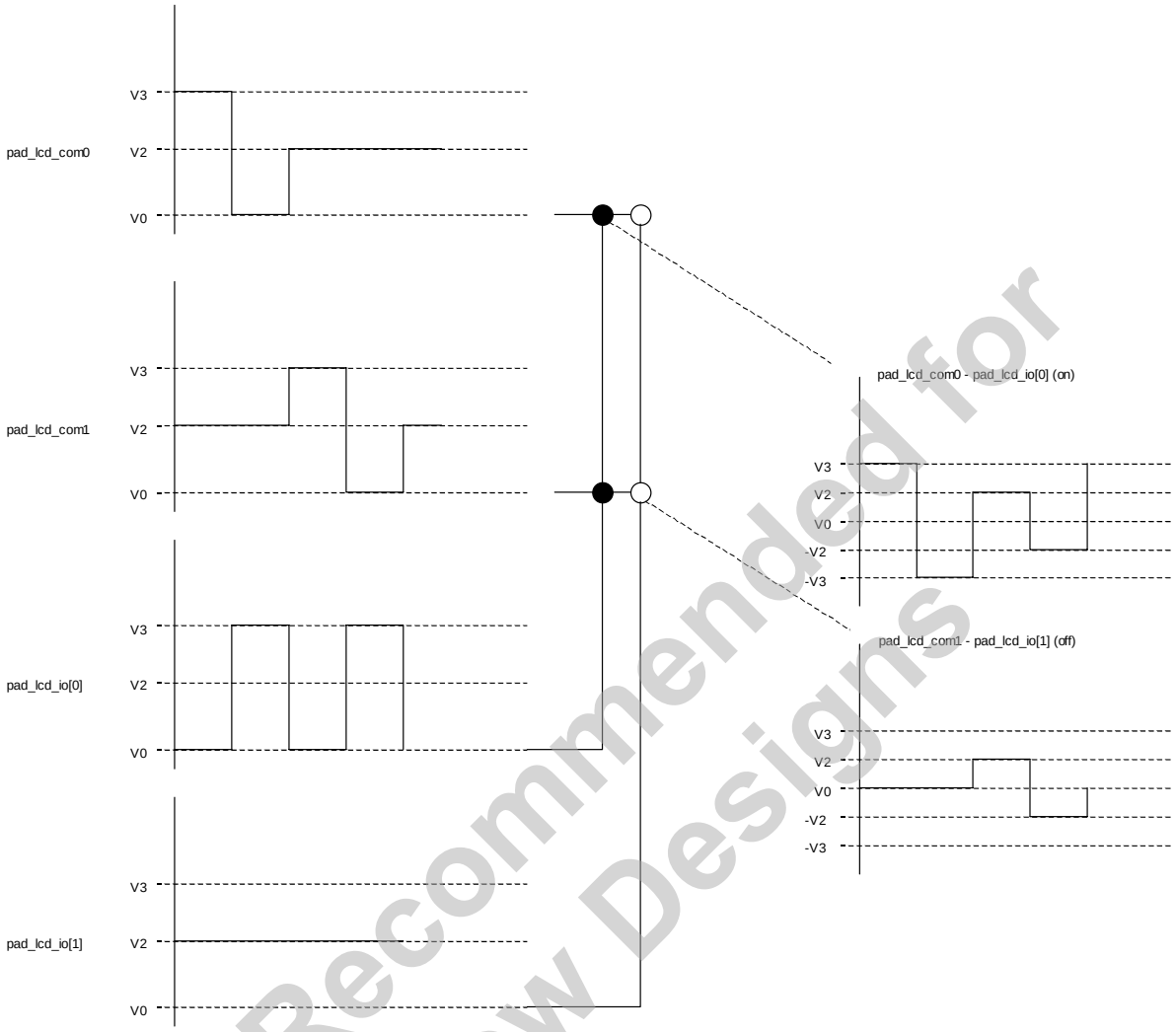


Figure 19-5. 1:2 MUX mode waveforms

The LCD driver is set to 1:2 multiplexing (**LcdMux**=01 in **RegLcdOn**) and the waveform generator is enabled (**LcdSleep**=0 in **RegLcdOn**). All pads are set into the LCD driver mode by setting all bits of register **RegLcdSe** to 1. To select a frame rate of about 50Hz, set the bits **LcdFreq**=01 and **LcdDivFreq**=100 in **RegLcdClkFrame**. Note that the precision of the frame frequency depends on the selected clock source (see clock block documentation).

The 64 segments are on or off depending on the bits set in the registers **RegLcdDataN**. Only the bits 0,1 and 4,5 of these registers are used in 1:2 multiplexing. Figure 19-5 shows the generated waveforms for four segments. Segments connected to pad_lcd_io[0] are on (**LcdData0[1:0]**=11 in **RegLcdData0**) and segments connected to pad_lcd_io[1] are off (**LcdData0[5:4]**=00 in **RegLcdData0**). The figure shows on the left side the waveforms on the circuit pins, in the middle the segment status and on the right the voltage on the segment (difference between the segment pin pad_lcd_io and common signal pad_lcd_com0 or pad_lcd_com1).

19.4.3 1:3 Multiplex scheme

With 1:3 multiplexing, each pad_lcd_io pin can drive three segments of the display. In this mode, up to 93 segments of the display can be driven by the pad_lcd_io[30:0] pins. The common nodes are to be connected to the pad_lcd_com0, pad_lcd_com1 and pad_lcd_io[31]. These connections are shown in Figure 19-6.

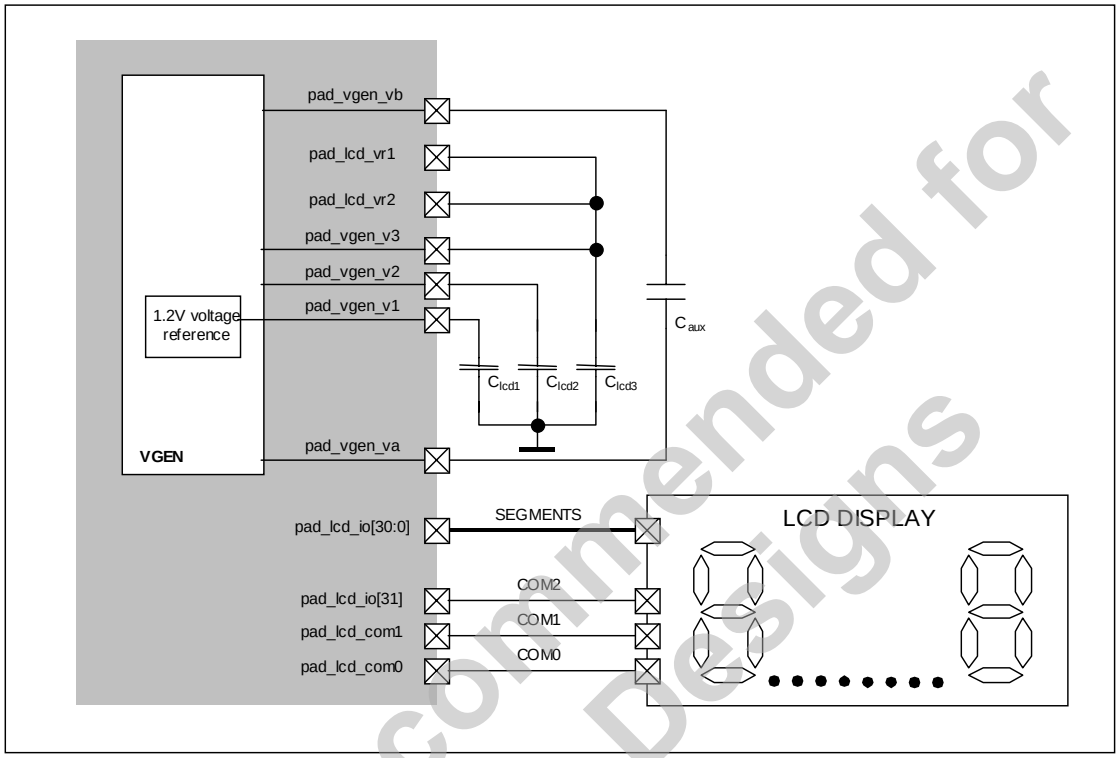


Figure 19-6: LCD with 1:3 multiplexing and on-chip 1/3 bias voltage generation.

Lots of different configurations for the voltage generation can be used. Figure 19-6 shows the connections and external elements that are required if the display is to be driven from 3.6V independently from the circuit supply voltage. Other possible configurations are given in section 19.5.4. The recommended value for the external capacitors is 470nF.

To operate the driver in this configuration, the registers should be loaded with the values in Table 19-15.

Register	Contents[7:0]
RegVgenCfg0	xx110001
RegLcdOn	xxxxx010
RegLcdSe	11111111
RegLcdClkFrame	110xxx10

Table 19-15. Register contents for 1:3 mux example

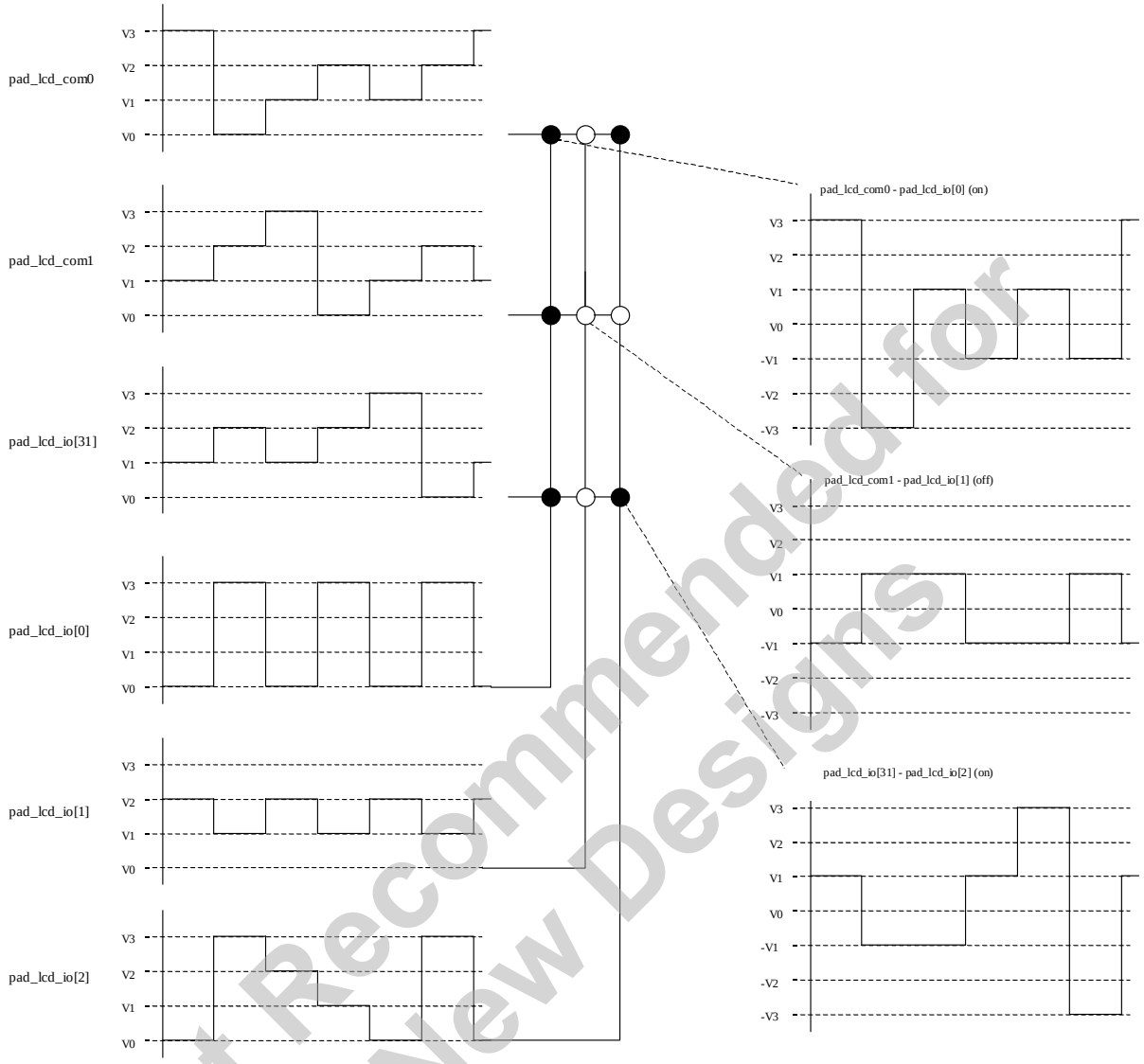


Figure 19-7. 1:3 MUX mode waveforms

The voltage generator has to be configured for the use of the internal reference (**VgenRefEn**=1 in **RegVgenCfg0**). The generator has to run (**VgenOff**=0 and **VgenStdb**=0 in **RegVgenCfg0**). It has to generate 1/3 bias (**VgenMode**=0 in **RegVgenCfg0)** and the generator output impedance is set to minimum (**VgenClkSel**=11 in **RegVgenCfg0)**.

The LCD driver is set to 1:3 multiplexing (**LcdMux**=10 in **RegLcdOn**) and the waveform generator is enabled (**LcdSleep**=0 in **RegLcdOn**). All pads are set into the LCD driver mode by setting all bits of register **RegLcdSe** to 1. To select a frame rate of about 50Hz, set the bits **LcdFreq**=10 and **LcdDivFreq**=110 in **RegLcdClkFrame**. Note that the precision of the frame frequency depends on the selected clock source (see clock block documentation).

The 93 segments are on or off depending on the bits set in the registers **RegLcdDataN**. Only the bits 0,1,2 and 4,5,6 of these registers are used in 1:3 multiplexing. The bits 3 and 7 are not important. Figure 19-7 shows the generated waveforms for nine segments. The segments connected to pad_lcd_io[0] are all on (**LcdData0[2:0]**=111 in **RegLcdData0**), the segments connected to pad_lcd_io[1] are all off (**LcdData0[6:4]**=000 in **RegLcdData0**) and the segments connected to pad_lcd_io[2] are partially on, partially off (**LcdData1[2:0]**=101 in **RegLcdData1**). The

figure shows on the left side the waveforms on the circuit pins, in the middle the segment status and on the right the voltage on some segments (difference between the segment pin pad_lcd_io and common signals: pad_lcd_com0, pad_lcd_com1 or pad_lcd_io[31]).

19.4.4 1:4 Multiplex scheme

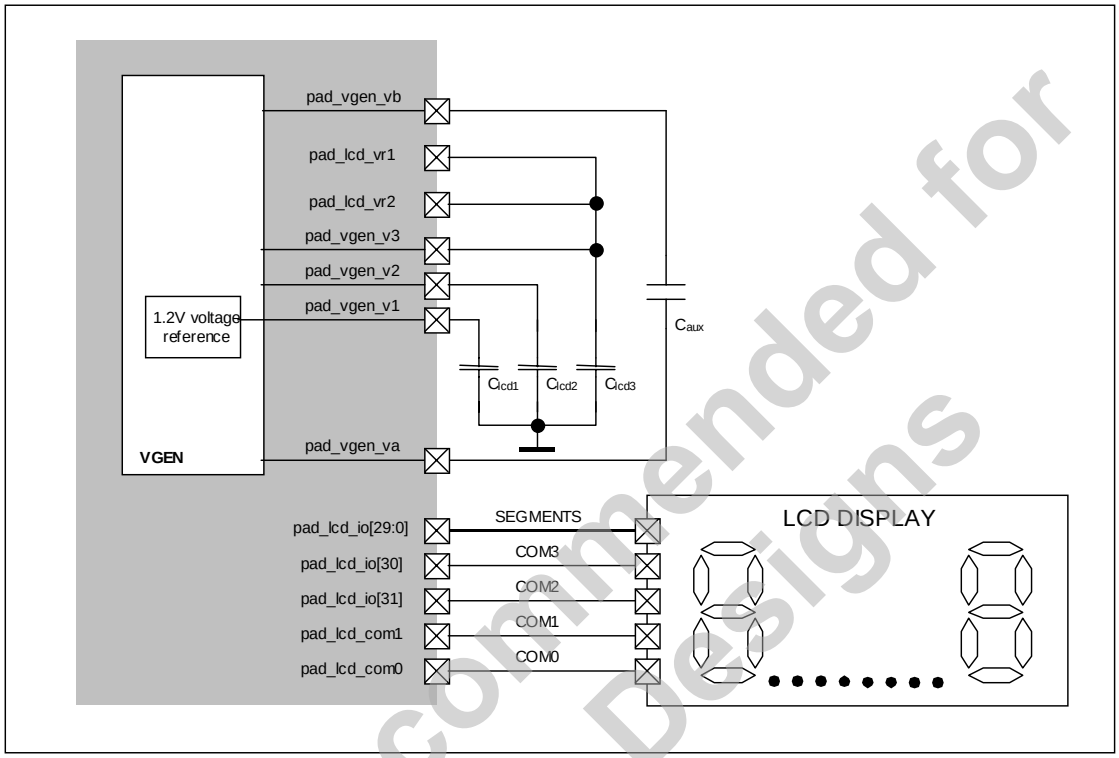


Figure 19-8: LCD with 1:4 multiplexing and on-chip 1/3 bias voltage generation.

With 1:4 multiplexing, each pin pad_lcd_io can drive four segments of the display. In this mode, up to 120 segments of the display can be driven by the pad_lcd_io[29:0] pins. The common nodes are to be connected to the pad_lcd_com0, pad_lcd_com1, pad_lcd_io[31] and pad_lcd_io[30]. These connections are shown in Figure 19-8.

Lots of different configurations for the voltage generation can be used. Figure 19-8 shows the connections and external elements that are required if the display is to be driven from V3=3.6V, V2=2.4V and V1=1.2V independently from the circuit supply voltage. Other possible configurations are given in section 19.5.4. The recommended value for the external capacitors is 470nF.

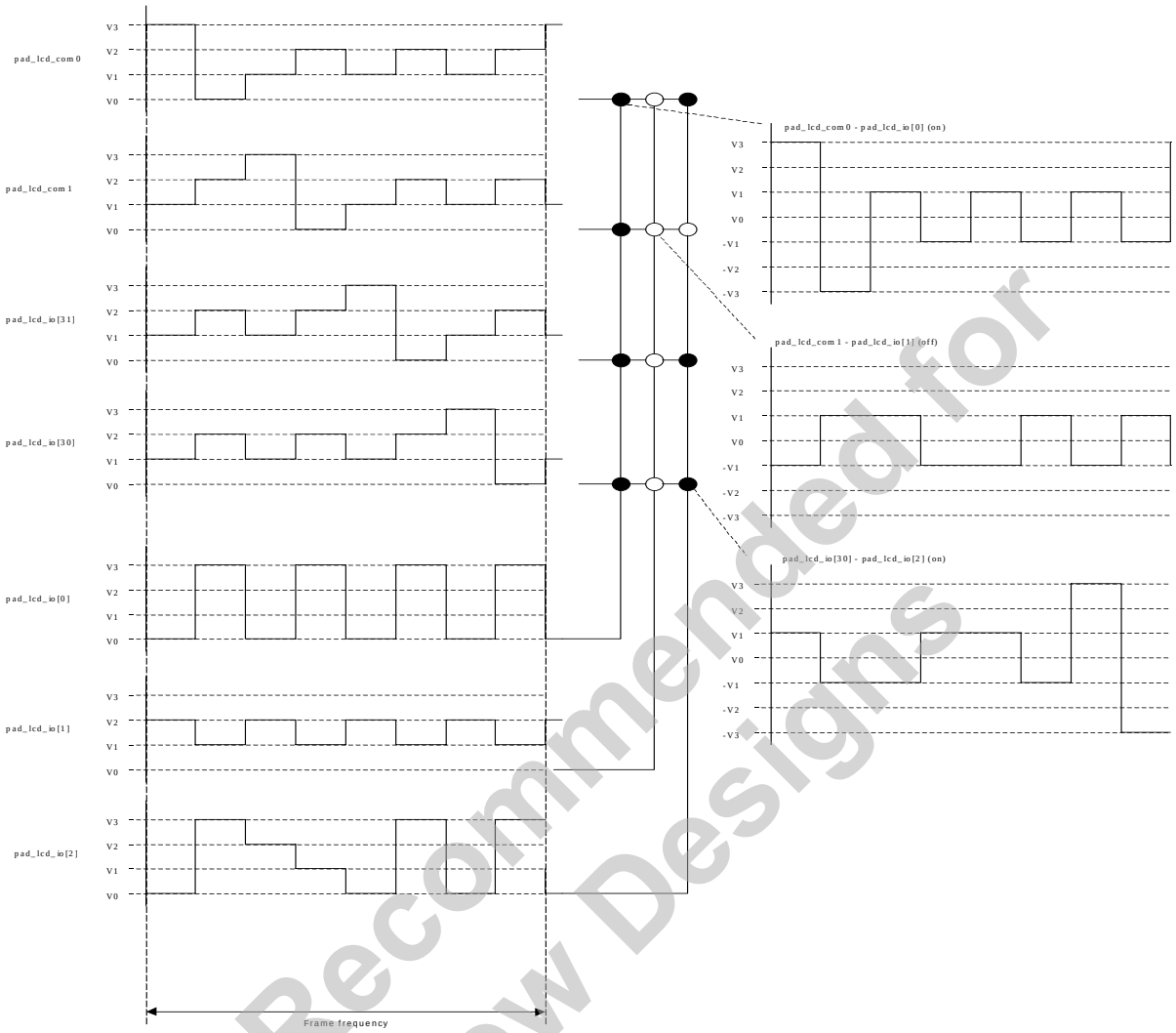


Figure 19-9. 1:4 MUX mode waveforms

To operate the driver in this configuration, the registers should be loaded with the values in Table 19-16.

The voltage generator has to be configured for the use of the internal reference (**VgenRefEn**=1 in **RegVgenCfg0**). The generator has to run (**VgenOff**=0 and **VgenStdb**=0 in **RegVgenCfg0**). It has to generate 1/3 bias (**VgenMode**=0 in **RegVgenCfg0)** and the generator output impedance is set to minimum (**VgenClkSel**=11 in **RegVgenCfg0)**.

Register	Contents[7:0]
RegVgenCfg0	xx110001
RegLcdOn	xxxxx011
RegLcdSe	11111111
RegLcdClkFrame	100xxx10

Table 19-16. Register contents for 1:4 mux example

The LCD driver is set to 1:4 multiplexing (**LcdMux**=11 in **RegLcdOn**) and the waveform generator is enabled (**LcdSleep**=0 in **RegLcdOn**). All pads are set into the LCD driver mode by setting all bits of register **RegLcdSe** to

1. To select a frame rate of about 50Hz, set the bits **LcdFreq**=10 and **LcdDivFreq**=100 in **RegLcdClkFrame**. Note that the precision of the frame frequency depends on the selected clock source (see clock block documentation).

The 120 segments are on or off depending on the bits set in the registers **RegLcdDataN**. All bits of these registers are used in 1:4 multiplexing. Figure 19-9 shows the generated waveforms for twelve segments. The segments connected to pad_lcd_io[0] are all on (**LcdData0[3:0]**=1111 in **RegLcdData0**), the segments connected to pad_lcd_io[1] are all off (**LcdData0[7:4]**=0000 in **RegLcdData0**) and the segments connected to pad_lcd_io[2] are partially on, partially off (**LcdData1[2:0]**=1101 in **RegLcdData1**). The figure shows on the left side the waveforms on the circuit pins, in the middle the segment status and on the right the voltage on some segments (difference between the pad_lcd_io pin and common signal: pad_lcd_com0, pad_lcd_com1, pad_lcd_io[31] or pad_lcd_io[30]).

19.5 Advanced LCD features

19.5.1 Register usage

To set the peripheral in LCD mode, all bits in **RegLcdSe** have to be set to 1.

The status of the segments has to be written to the registers **RegLcdDataN** (with 0<N<15). Each register drives the segments connected to the pins pad_lcd_io[2N] (bits 0 to 3) and pad_lcd_io[2N+1] (bits 4 to 7). Depending on the selected multiplexing (**LcdMux** in **RegLcdOn**), not all bits in the data registers **RegLcdDataN** are used as shown in Table 19-17. The unused locations will always read back '0'. When reading back these registers, it will give the actual value on the display. When writing to these registers, the modifications will be taken into account only at the start of a new frame.

Note that in the 1:3 multiplexing, the pad_lcd_io[31] pin is used as the third common signal and the data **LcdData15[7:4]** are not used. In the 1:4 multiplexing, the pad_lcd_io[31:30] are used as common signals and the data **LcdData15[7:0]** are not used.

RegLcdDataN[x]	**DD**	**1:2**	**1:3**	**1:4**
3, 7	x	x	x	used
2, 6	x	x	used	used
1, 5	x	used	used	used
0, 4	used	used	used	used

Table 19-17. Useful data.

In the LCD mode, all pads have LCD driver capabilities. The parallel port configuration registers (**RegPLcdPullupN**, **RegPLcdDirN** and **RegPLcdOutN**) are unused. Writing to these registers will have no influence on the peripheral activity. These registers and **RegPLcdInN** will read back '0'.

19.5.2 Sleep mode or blinking mode

When putting the LCD driver in sleep mode (**LcdSleep** = 1 in **RegLcdOn**), the peripheral will stop the waveform generation at the end of a frame. Entering sleep mode does not modify the LCD set-up but during sleep mode, the user is still allowed to write data into the **RegLcdDataN** registers. Figure 19-10 shows an example of the sleep mode synchronisation. The sleep signal can be used to blink the display.

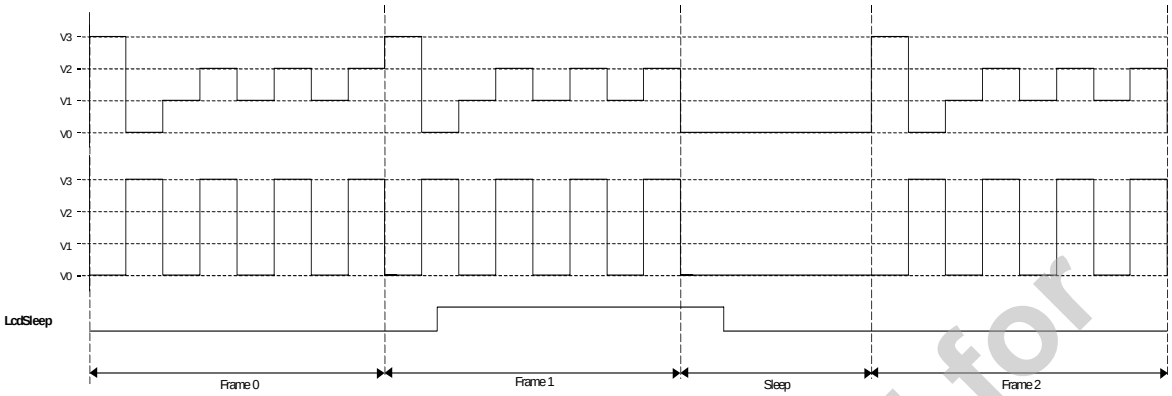


Figure 19-10. Sleep mode synchronisation

19.5.3 LCD frame frequency selection

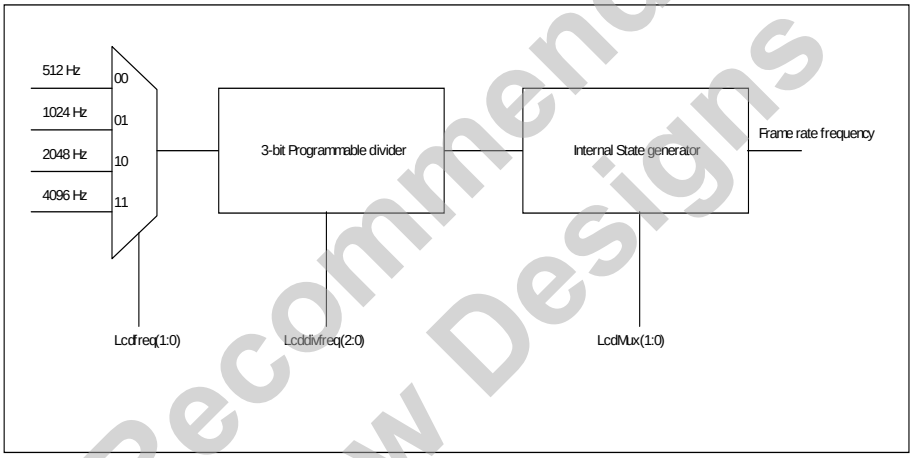


Figure 19-11: Frame Rate generation

The LCD driver frame frequency depends on three parameters: the selected clock source frequency, the selected division factor and the multiplexing scheme. As shown in Figure 19-11, four different clock source frequencies can be selected by setting the bits **LcdFreq** in **RegLcdClkFrame**. This frequency is further divided by a factor between 1 and 8 depending on the bits **LcdDivFreq** in **RegLcdClkFrame**. It further depends on the selected multiplexing scheme (bits **LcdMux** in **RegLcdOn**). The selections are defined in Table 19-18 to Table 19-20. The frame rate FR is then given by:

$$FR = \frac{f_{LCD}}{2 \cdot div \cdot mux}$$

Finally, Table 19-21 shows the minimal and maximal frame rate for each multiplexing. The source frequency should be chosen as low as possible to reduce the power consumption. Note that the precision of the frame rate depends on the precision of the selected clock source (see clock block documentation).

LcdFreq	f_{LCD} [Hz]
00	512
01	1048
10	2048
11	4096

Table 19-18. LCD driver source frequency selection

LcdDivFreq	*div*
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

Table 19-19. LCD driver frequency division

LcdMux	multiplexing scheme	*mux*
00	direct drive	1
01	1:2	2
10	1:3	3
11	1:4	4

Table 19-20. LCD multiplexing

LcdMux	LcdFreq	LcdDivFreq	*FR* [Hz]
00	00	111	32
00	11	000	2048
01	00	111	16
01	11	000	1024
10	00	111	10.6
10	11	000	682.7
11	00	111	8
11	11	000	512

Table 19-21. Minimal and maximal Frame Rate

19.5.4 Voltage generation

The different voltages used in the LCD driver can be generated in several ways. Depending on the selected multiplexing scheme, the LCD display driver needs 2, 3 or 4 different voltages.

The reference voltage can be derived from an internal 1.2V band gap reference or from an externally supplied voltage. The circuit also contains a voltage multiplier/divider that can be configured to generate a 1/2 bias or a 1/3 bias. This multiplier/divider needs external capacitors. Instead of the internal multiplier/divider, an external resistive ladder can be used. The use of the internal circuitry will in most applications give the lowest power solution.

The voltages V0, V1 and V2 are connected internally in the circuit. The voltage V3 is not connected internally but externally by short-circuiting the pins pad_lcd_vr1 and pad_lcd_vr2 to pad_vgen_v3.

The voltages used in the LCD driver for the different multiplexing schemes are given in Table 19-22.

multpilexing	V0	V1	V2	V3
direct drive	used	-	-	used
1:2	used	-	used	used
1:3	used	used	used	used
1:4	used	used	used	used

Table 19-22. V0, V1, V2, V3 usage in the LCD driver

19.5.4.1 Generating LCD voltage with the internal multiplier/divider.

To generate LCD voltage with the internal multiplier/divider, some external capacitors have to be connected to the circuit. The value of the capacitor C_{aux} can be calculated depending on the output impedance that is required on the V3 voltage:

$$C_{aux} \cong \frac{6}{f_{vgen} \cdot Z_{outV3}}$$

with f_{vgen} the operating frequency of the multiplier/divider (set by the bits **VgenClkSel** in **RegVgenCfg0**, see Table 19-23) and Z_{outV3} the required output impedance of V3. The equation is valid for C_{aux} <5µF.

Note that the operating frequency depends on the selected clock source (see clock block documen-tation).

For a capacitor C_{aux} of 470nF and a frequency of 1024Hz (default value), an output impedance of 12kΩ is obtained. The capacitors C_{LCD1}, C_{LCD2} and C_{LCD3} can be chosen equal to C_{aux}.

VgenClkSel	f_{vgen} (Hz)
00	256
01	512
10	1024
11	2048

Table 19-23. multplier/divider operating frequency

To enable the voltage multiplier/divider, the bits **VgenOff** and **VgenStdb** in **RegVgenCfg0** are set to 0. The difference between the two bits is that **VgenOff** stops the generator and forces the nodes pad_vgen_v1, pad_vgen_v2, pad_vgen_v3, pad_vgen_va and pad_vgen_vb to predefined values and therefor discharges the capacitors C_{LCD1}, C_{LCD2} and C_{LCD3}. The bit **VgenStdb** only stops the operating clock without changing the voltage on C_{LCD1}, C_{LCD2} and C_{LCD3}. The capacitors will be discharged by leakage and by the switching of the LCD if the bit **LcdSleep** is not set.

The multiplier/divider can generate 1/2 bias (**VgenMode**=1 in **RegVgenCfg0**) or 1/3 bias (**VgenMode**=0 in **RegVgenCfg0**).

Finally, the multiplier/divider can use the internal 1.2V bandgap reference (**VgenRefEn**=1 in **RegVgenCfg0**) or an external reference (**VgenRefEn**=0 in **RegVgenCfg0**). The internal voltage reference can not be used in case the circuit voltage supply is below 1.5V.

Figure 19-12 shows the external connections to be made in 1/3 bias mode. Part (a) of the figure shows the use of the internal voltage reference. In this case, V1=1.2V, V2=2.4V and V3=3.6V. Part (b) of the figure shows the use of an external voltage reference. The external reference can be connected to one of the pins pad_vgen_v1, pad_vgen_v2, pad_vgen_v3. Table 19-24 shows the voltage generated on V1, V2, V3 depending on the connection of the external reference voltage Vext. The external voltage may or may not be identical to the circuit

supply voltage VBAT. The voltage on V3 should not exceed 5.5V. The voltage on pad_vgen_v1 should not exceed VBAT.

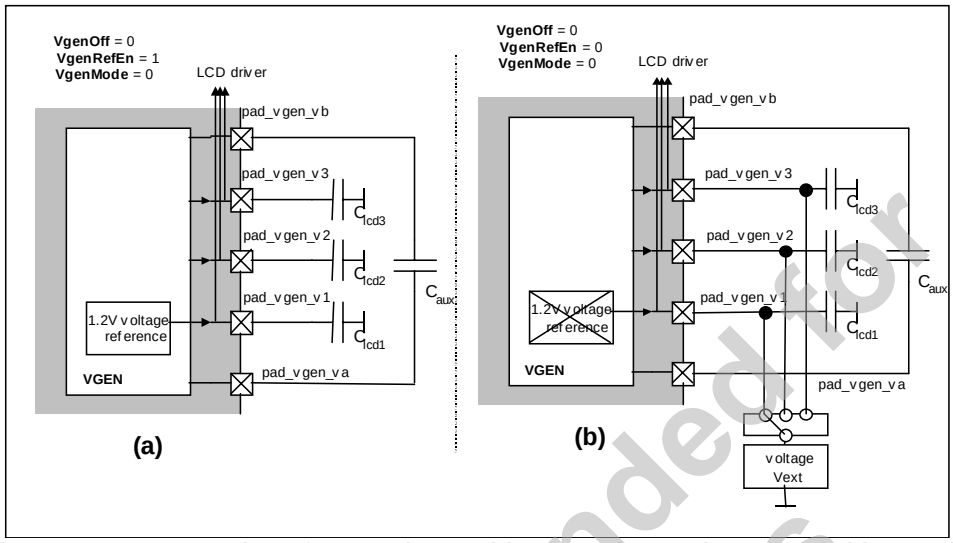


Figure 19-12: Generation of LCD voltage with external capacitors for 1/3 bias mode.

Vext connection	V1 (V)	V2 (V)	V3 (V)
pad_vgen_v1	Vext	2·Vext	3·Vext
pad_vgen_v2	(1/2)·Vext	Vext	(3/2)·Vext
pad_vgen_v3	(1/3) ·Vext	(2/3) ·Vext	Vext

Table 19-24. V1, V2 and V3 as a function of the external reference connection in 1/3 bias mode

Figure 19-13 shows the external connections to be made in 1/2 bias mode. Part (a) of the figure shows the use of the internal voltage reference. In this case, V1=V2=1.2V and V3=2.4V. Part (b) of the figure shows the use of an external voltage reference. The external reference can be connected to one of the pins pad_vgen_v1 or pad_vgen_v3. Table 19-25 shows the voltage generated on V1 and V3 depending on the connection of the external reference voltage Vext. The external voltage may or may not be identical to the circuit supply voltage VBAT. The voltage on V3 should not exceed 5.5V.

Vext connection	V1 (V)	V3 (V)
pad_vgen_v1	Vext	2·Vext
pad_vgen_v3	(1/2) ·Vext	Vext

Table 19-25. V1 and V3 as a function of the external reference connection in 1/2 bias mode

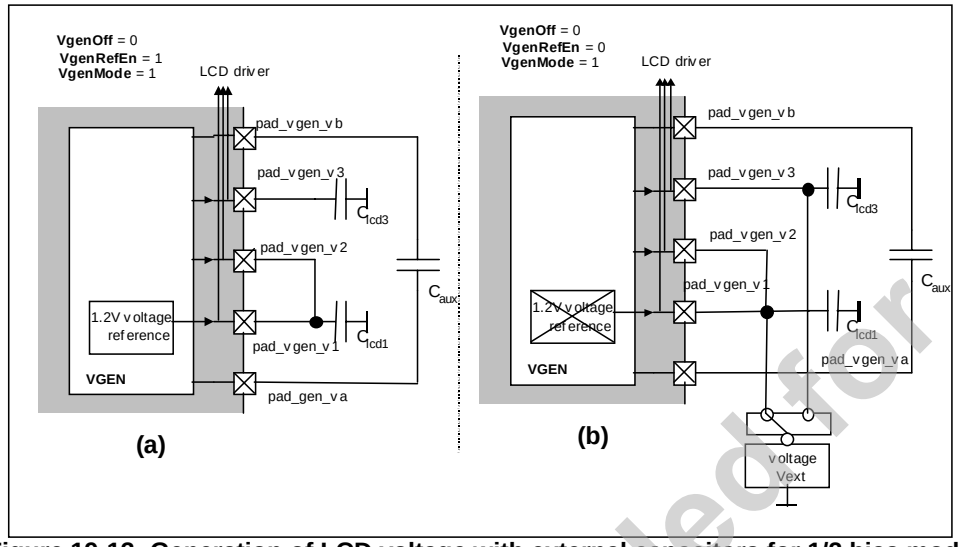


Figure 19-13: Generation of LCD voltage with external capacitors for 1/2 bias mode.

Finally, in direct drive, there is no need for intermediate voltages. The configurations of Figure 19-12(a) or Figure 19-13(a) can still be used (depending on the required voltage: 2.4V or 3.6V) but the configurations in Figure 19-12(b) or Figure 19-13(b) can be replaced by the simple schematic of Figure 19-14. The external voltage Vext may or may not be identical to the circuit supply voltage VBAT.

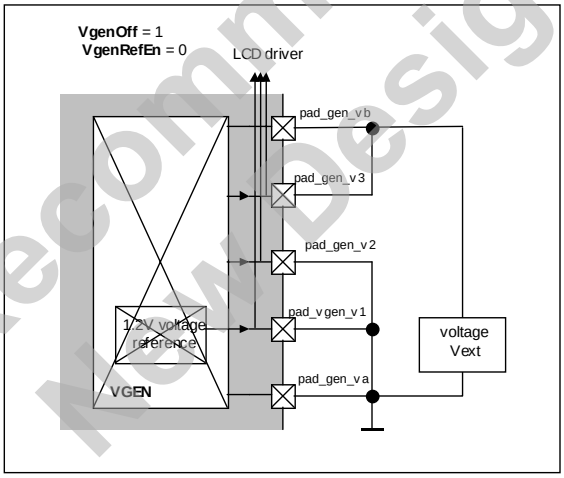


Figure 19-14. Generation of LCD voltage for direct drive using external voltage.

19.5.4.2 Generating LCD voltages with an external resistor ladder

To generate the LCD voltages with an external R-ladder (Figure 19-15), the internal voltage reference and multiplier/divider block are not used (**VgenOff**=1 and **VgenRefEn**=0 in **RegVgenCfg0**). All other bits in the register **RegVgenCfg0** are unused.

All resistor values are equal and sufficiently small in order to have small LCD voltage impedance. For direct drive, no resistors are required as shown in Figure 19-14. The voltage Vext can be connected to the circuit supply voltage VBAT or any other external voltage.

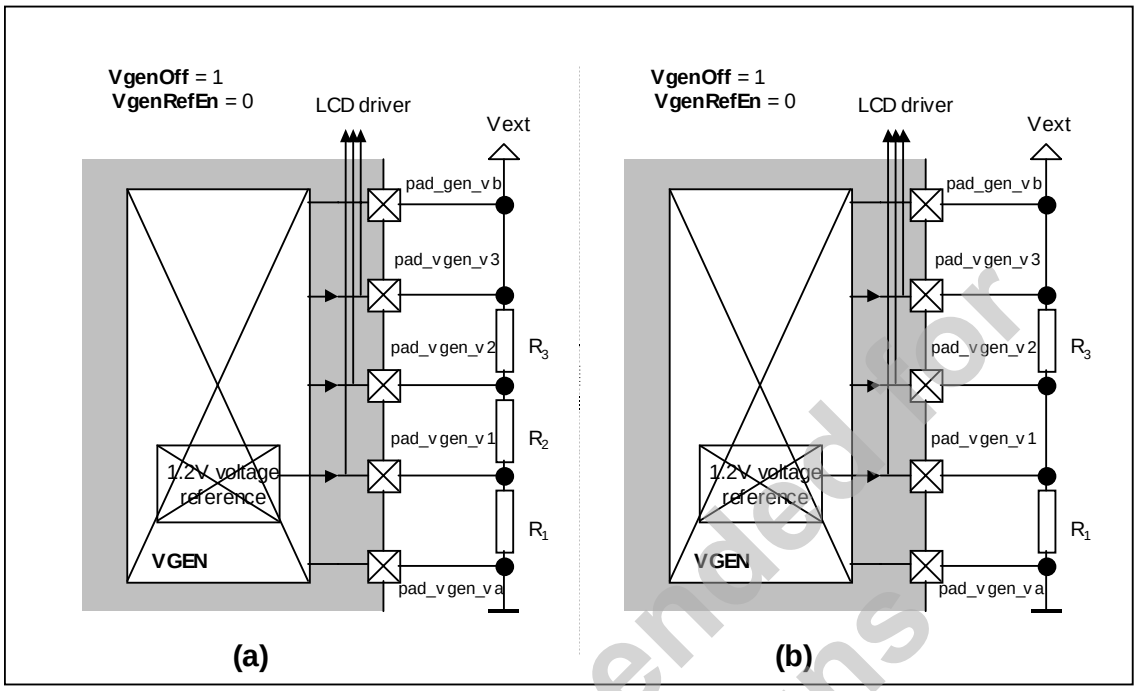


Figure 19-15. Generation of LCD voltages with external resistors ladder in 1/3 (a) and 1/2 (b) bias mode.

19.6 Parallel I/O port capabilities

19.6.1 Parallel port function

All pins pad_lcd_io[31:0] can be used as a general input/output digital port. The peripheral is set to this mode by writing all bits of the register **RegLcdSe** to '0'. Figure 19-16 shows the structure for one pin and the registers used in this mode. Since the peripheral has 32 pins, 4 of each of the registers in Figure 19-16 exist. They are labelled with the suffix 0 to 3. The mapping of the register bits to the I/O pins is given in Table 19-26. As an example, the bit **PLcdPullup2[6]** in **RegPLcdPullup2** controls the pull up resistor of the pin pad_lcd_io[22].

suffix of the register 0≤N≤3	bit in the register 0≤n≤7	pin
N	n	pad_lcd_io[8N+n]

Table 19-26. Register bit to pin mapping

The direction of each pin pad_lcd_io[31:0] (input or output) can be individually set by using the **RegPLcdDirN** registers. If **PLcdDirN[n]** = 1, the output buffer of the corresponding pad_lcd_io[8N+n] is enabled.

Output mode:
The data to be output is stored in **RegPLcdOutN**. In output mode, the pull-up resistors should be switched off by writing '0' to the registers **RegPLcdPullupN**. If not, current will flow through the pull-up resistors when the output is forced to '0'.

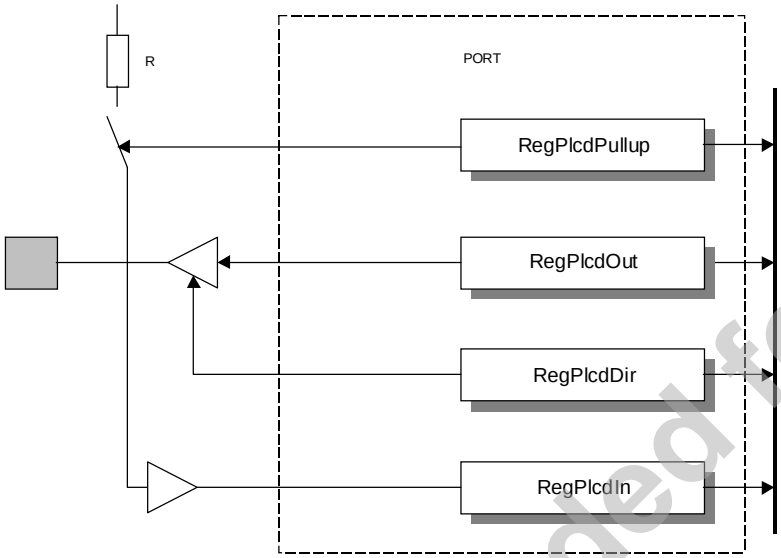


Figure 19-16. Structure of the LCD driver used as parallel port

Input mode:
The status of the pins pad_lcd_io is available in **RegPLcdInN** (read only). Reading is always direct, there is no digital debounce function. In case of noisy input signals, a software debouncer or an external filter must be realised.

The pull-up resistors are individually controllable for each pin by setting the corresponding bit in the registers **RegPLcdPullN** (1=active, 0=inactive).

19.6.2 Parallel Port Voltage

When the peripheral is used as a parallel port, the internal voltage generator is not required (**VgenOff**=1 and **VgenRefEn**=0 in **RegVgenCfg0**). For normal operation as a standard parallel port, the circuit is connected as shown in Figure 19-17. In this case, the logical '1' corresponds to the voltage VBAT.

The parallel port can also be driven from another voltage than VBAT. This allows the circuit to be interfaced to other circuits that have a different supply voltage domain VDD1 that can be either lower or higher than VBAT without adding any extra hardware as shown in Figure 19-18.

The parallel port can be split in several voltage domains. The pins pad_lcd_io[31:12] are supplied from the pad_vgen_v3 pin. The pins pad_lcd_io[11:4] are supplied from the pin pad_lcd_vr1 and the pins pad_lcd_io[3:0] are supplied from the pin pad_lcd_vr2. Each of these can be supplied with a different voltage as shown in Figure 19-19. As an example, VBAT could be supplied at 2.7V, VDD1 at 2.2V, VDD2 at 3.3V and VDD3 at 5V. The only limitation is that VDD1>VREG, VDD2>VREG, VDD3>VREG.

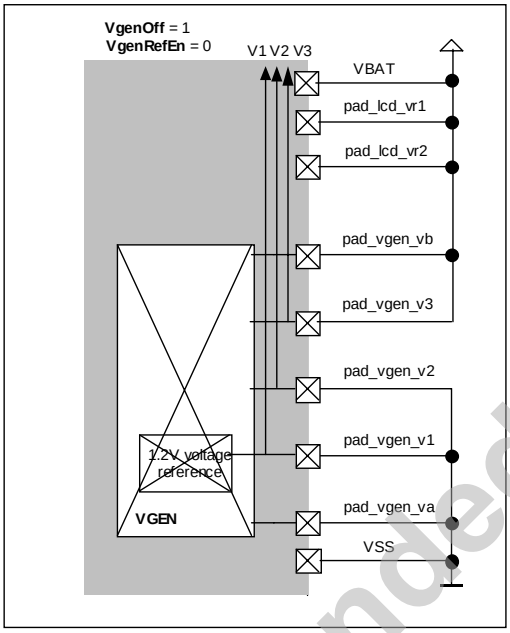


Figure 19-17. Parallel port voltage connection

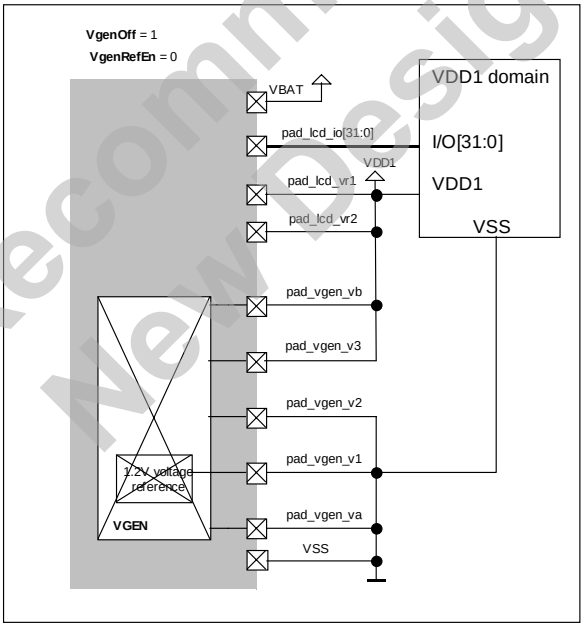


Figure 19-18. Parallel port voltage connection with I/O levelshifting

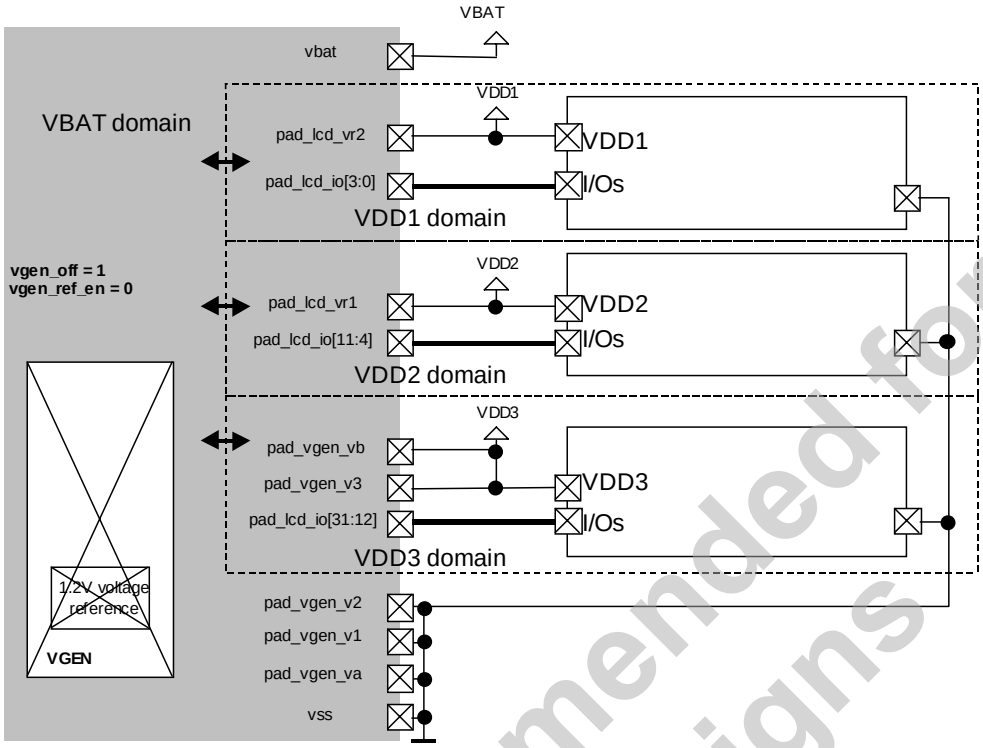


Figure 19-19. Multi voltage parallel port interface

19.7 Partial LCD Driver / Partial Parallel I/O Port

It is perfectly possible to combine different modes described in the previous sections on different pins. Using the bits **LcdSe** in the register **RegLcdSe**, the pins pad_lcd_io[31:0] can be set as parallel port or as LCD drivers. Each bit sets the mode for 4 pins (see Table 19-4).

When combining the LCD and digital port functions, care has to be taken with the voltage generation. The logical '1' of the digital I/O is driven from the V3 rail and needs to be low impedance. The V3 of the pins used for digital I/O can not be supplied by the on-circuit voltage multiplier. The next sections show different possibilities.

19.7.1 Single low impedance voltage for V3 of LCD and digital I/O

If V3 in the LCD driver is supplied from a low impedance voltage supply (as e.g. in Figure 19-12(b) or Figure 19-13(b) with connection of Vext to pad_vgen_v3, or in Figure 19-14 and Figure 19-15), the partitioning of the pins pad_lcd_io[31:0] between the LCD driver function and the digital parallel port function can be chosen freely by setting the bits **LcdSe**. Vext may be equal to the circuit supply voltage VBAT but this is not required. The digital I/O pins use the voltages V0=VSS and V3=Vext for the logical '0' and logical '1' respectively.

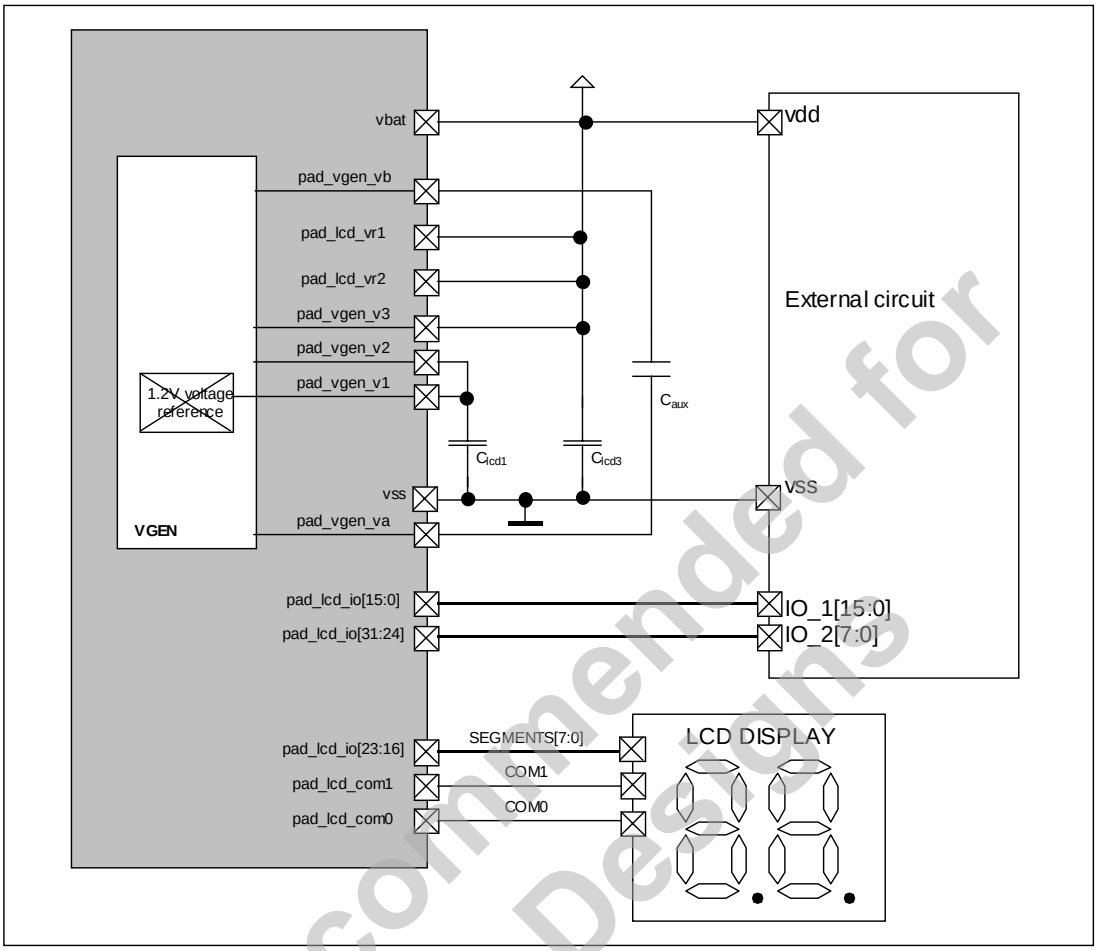


Figure 19-20: Sharing LCD (1:2 mux) and digital I/O (low impedance V3)

Figure 19-20 and Table 19-27 show a possible example for such a configuration. In this case, the LCD driver voltage V3=VBAT and V1=VBAT/2 (1/2 bias mode, **VgenMode**=1, **VgenRefEn**=0 and **VgenOff**=0 in **RegVgenCfg0**). The LCD is in 1:2 multplexing (**LcdMux**=01 in **RegLcdOn**) and **LcdSe19**=**LcdSe23**=1 while all other bits in **RegLcdSe** are 0.

Register	Contents[7:0]
RegVgenCfg0	xx110100
RegLcdOn	xxxxx001
RegLcdSe	11110011

Table 19-27. Register contents for configuration of Figure 19-20.

19.7.2 Different Voltages for V3 of LCD and Digital I/O

When configured as a digital I/O, the logical '1' on the pins pad_lcd_io[31:0] is identical to V3. For a part of the pad_lcd_io[31:0] pins, the V3 connection is not made inside the circuit but has to be done externally by connecting the pins pad_vgen_v3, pad_lcd_vr1 and pad_lcd_vr2 together. This feature allows for the use of different voltages on V3 for the LCD display and for I/O parallel pins. Table 19-28 shows the partitioning of the pins. The voltages that can be applied on pad_vgen_v3, pad_lcd_vr1, pad_lcd_vr2 and VBAT are completely independent from each other.

pin	V3 connection
pad_lcd_com0 pad_lcd_com1 pad_lcd_io[31:12]	pad_vgen_v3
pad_lcd_io[11:4]	pad_lcd_vr1
pad_lcd_io[3:0]	pad_lcd_vr2

Table 19-28. V3 connection of the different pad_lcd_io pins

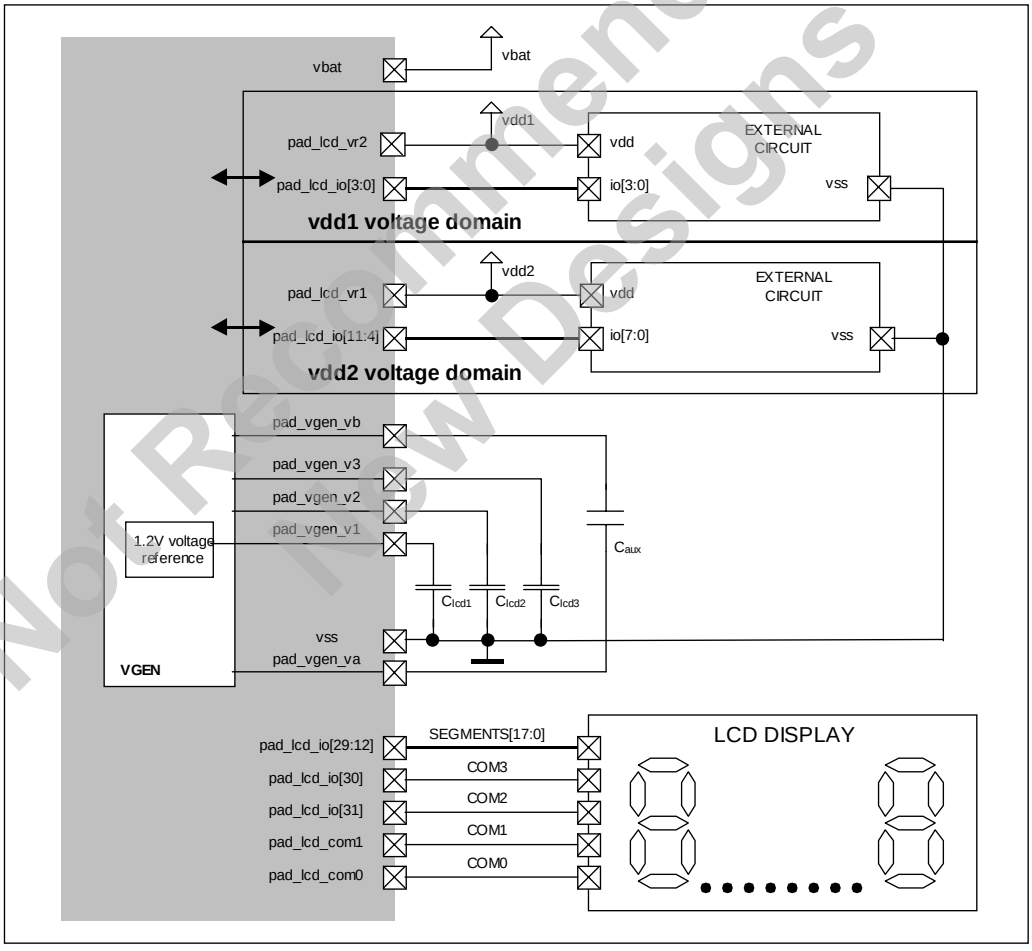


Figure 19-21. Sharing LCD driver (1:4 mux) and digital I/O with V3 ≠ logic '1'

As can be seen from Table 19-28, the voltage V3 on the pins pad_lcd_io[31:12] can not be dissociated from the voltage V3 on the pins pad_lcd_com0, pad_lcd_com1 and the internal voltage multiplier/divider. It means that, if V3 is not a low impedance external voltage as in the previous section, they can be used for the LCD driver only and not for digital I/O.

Figure 19-21 and Table 19-29 show an example. In this case, the pins pad_lcd_io[29:12] are used to drive a display with 1:4 multiplexing (**LcdSe15**=**LcdSe19**=**LcdSe23**=**Lcd27**=**Lcd31**=1 in **RegLcdSe** and **LcdMux**=11 in **RegLcdOn**). In 1:4 multiplexing, the lines pad_lcd_io[31:30] are used for COM2 and COM3. The voltage V3 for the display is generated by the internal voltage multiplier/divider using the internal reference (**VgenOff**=0, **VgenRefEn**=1, **VgenMode**=0 in **RegVgenCfg0**). The segment status is set by using the **RegLcdDataN** registers with 6≤N≤15. Writing in the registers with 0≤N≤5 will have no effect. The pins pad_lcd_io[11:0] are used as digital I/O (**LcdSe3**=**LcdSe7**=**LcdSe11**=0 in **RegLcdSe**). The control of the digital I/O is done using the registers **RegPLcdInN**, **RegPLcdOutN**, **RegPLcdDirN** and **RegPLcdPullupN** with 0≤N≤1. Writing in the registers with 2≤N≤3 will have no effect. The pins pad_lcd_io[11:4] and pad_lcd_io[3:0] can further be split into two different voltage domains. The voltage domains VDD1, VDD2, V3 and VBAT are independent. The only limitation is that VDD1>VREG and VDD2>VREG. In the example V3=3.6V, VBAT could be at 2.7V, VDD1 at 2.4V and VDD2 at 5V.

Register	Contents[7:0]
RegVgenCfg0	xx110001
RegLcdOn	xxxxx011
RegLcdSe	00011111

Table 19-29. Register contents for configuration of Figure 19-21.

19.8 Specifications

19.8.1 pad_lcd_io used in LCD mode

Specification	Min	Typ	Max	Unit	Description	Comments
V1	1.1		VBAT	V		
V2, V3	1.1		5.5	V		
$t_{rise-fall}$			25	µs	Rise/Fall time (LCD mode)	(1) (2) (3)

(1) rise or fall time from 10% to 90% of the output signal
(2) Cload=5000pF
(3) V1=V2/2=V3/3=1.1V (1/3 bias) or V1=V2=V3/2=1.1V (1/2 bias)

19.8.2 pad_lcd_io used in digital I/O mode

Specification	Min	Typ	Max	Unit	Description	Comments
pad_lcd_vr1	VREG		5.5	V	pad supply voltage	
pad_lcd_vr2	VREG		5.5	V	pad supply voltage	
pad_vgen_v3	VREG		5.5	V	pad supply voltage	
$R_{\_pullud}$	35		100	kΩ	Pull up/down resistance	
$t_{rise-fall}$		1		µs	Rise/Fall time	(1) (2)
I_{OD}	8			mA	Output current drive	(3)

(1) rise or fall time from 10% to 90% of the output signal
(2) with Cload=5nF, pad_vgen_v3=pad_lcd_vr1=pad_lcd_vr2=2.4V
(3) pad_vgen_v3=pad_lcd_vr1=pad_lcd_vr2=4.5V, voltage on pad_lcd_io=0.4V for sink current and 4.1V for source current.

19.8.3 Voltage Reference

Specification	Min	Typ	Max	Unit	Comment
Vref	1.0	1.17	1.34	V	@20°C, VBAT>2.4V
ΔVref/ΔT		0.2		mV/°C	
Power supply VBAT	1.5		5.5	V	
I_load on V1			2	mA	
Zout on V1			1000	Ω	

19.8.4 LCD multiplier/divider

Specification	Min	Typ	Max	Unit	Comments
Settling time to 90%			30	ms	(2), (4), (5)
$Z_{out,v2}$ on V2	3 (1)	6 (2)	25 (3)	kΩ	(4), (5)
$Z_{out,v3}$ on V3	7 (1)	14 (2)	60 (3)	kΩ	(4), (5)
pad_vgen_v3			5.5	V	

(1) f(vgen_clk) = 2 kHz.
(2) f(vgen_clk) = 1 kHz.
(3) f(vgen_clk) = 0.25 kHz.
(4) 1/3 bias mode.
(5) with 0.47μF external capacitors.

20. Counters/PWM

20.1 Features

- 4 x 8-bits timer/counter modules or 2 x 16-bits timers/counter modules
- Each with 4 possible clock sources
- Up/down counter modes
- Interrupt and event generation
- Capture function (internal or external source)
- Rising, falling or both edge of capture signal (except for xtal 32 kHz, only rising edge)
- PA[3:0] can be used as clock inputs (debounced or direct, frequency divided by 2 or not)
- 2 x 8 bits PWM or 2 x 16 bits PWM
- PWM resolution of 8, 10, 12, 14 or 16 bits
- Complex mode combinations are possible

20.2 Overview

Counter A and Counter B are 8-bits counters and can be combined to form a 16-bit counter. Counter C and Counter D exhibit the same feature.

The counters can also be used to generate two PWM outputs on PB[0] and PB[1]. In PWM mode one can generate PWM functions with 8, 10, 12, 14 or 16 bits wide counters.

The counters A and B can be captured by events on an internal or an external signal. The capture can be performed on both 8-bit counters running individually on two different clock sources or on both counters chained to form a 16-bit counter. In any case, the same capture signal is used for both counters.

When the counters A and B are not chained, they can be used in several configurations: A and B as counters, A and B as captured counters, A as PWM and B as counter, A as PWM and B as captured counter.

When the counters C and D are not chained, they can be used either both as counters or counter C as PWM and counter D as counter.

20.3 Register map

register name
RegCntA
RegCntB
RegCntC
RegCntD
RegCntCtrlCk
RegCntConfig1
RegCntConfig2
RegCntOn

Table 20-1. Counter registers

bit	RegCntA	rw	reset	function
7-0	CounterA	R	00000000 nresetglobal	8-bits counter value
7-0	CounterA	W	00000000 nresetglobal	8-bits comparison value

Table 20-2. **RegCntA**

bit	RegCntB	rw	reset	function
7-0	CounterB	R	00000000 nresetglobal	8-bits counter value
7-0	CounterB	W	00000000 nresetglobal	8-bits comparison value

Table 20-3. **RegCntB**

Note: When writing to **RegCntA** or **RegCntB**, the processor writes the counter comparison values. When reading these locations, the processor reads back either the actual counter value or the last captured value if the capture mode is active.

bit	RegCntC	rw	reset	function
7-0	CounterC	R	00000000 nresetglobal	8-bits counter value
7-0	CounterC	W	00000000 nresetglobal	8-bits comparison value

Table 20-4. **RegCntC**

bit	RegCntD	rw	reset	function
7-0	CounterD	R	00000000 nresetglobal	8-bits counter value
7-0	CounterD	W	00000000 nresetglobal	8-bits comparison value

Table 20-5. **RegCntD**

Note: When writing **RegCntC** or **RegCntD**, the processor writes the counter comparison values. When reading these locations, the processor reads back the actual counter value.

bit	RegCntCtrlCk	rw	reset	function
7-6	CntDCkSel(1:0)	R/W	00 nresetglobal	Counter d clock selection
5-4	CntCCkSel(1:0)	R/W	00 nresetglobal	Counter c clock selection
3-2	CntBCkSel(1:0)	R/W	00 nresetglobal	Counter b clock selection
1-0	CntACkSel(1:0)	R/W	00 nresetglobal	Counter a clock selection

Table 20-6. **RegCntCtrlCk**

bit	RegCntConfig1	rw	reset	function
7	CntDDownUp	R/W	0 nresetglobal	Counter d up or down counting (0=down)
6	CntCDownUp	R/W	0 nresetglobal	Counter c up or down counting (0=down)
5	CntBDownUp	R/W	0 nresetglobal	Counter b up or down counting (0=down)
4	CntADownUp	R/W	0 nresetglobal	Counter a up or down counting (0=down)
3	CascadeCD	R/W	0 nresetglobal	Cascade counter c & d (1=cascade)
2	CascadeAB	R/W	0 nresetglobal	Cascade counter a & b (1=cascade)
1	CntPWM1	R/W	0 nresetglobal	Activate pwm1 on counter c or c+d (PB(1))
0	CntPWM0	R/W	0 nresetglobal	Activate pwm0 on counter a or a+b (PB(0))

Table 20-7. **RegCntConfig1**

bit	RegCntConfig2	rw	reset	function
7-6	CapSel(1:0)	R/W	00 nresetglobal	Capture source selection
5-4	CapFunc(1:0)	R/W	00 nresetglobal	Capture function
3-2	Pwm1Size(1:0)	R/W	00 nresetglobal	Pwm1 size selection
1-0	Pwm0Size(1:0)	R/W	00 nresetglobal	Pwm0 size selection

Table 20-8. **RegCntConfig2**

bit	RegCntOn	rw	reset	function
7	CntDExtDiv	R/W	0 nresetglobal	Divide PA(3) frequency by 2 (1=divide)
6	CntCExtDiv	R/W	0 nresetglobal	Divide PA(2) frequency by 2 (1=divide)
5	CntBExtDiv	R/W	0 nresetglobal	Divide PA(1) frequency by 2 (1=divide)
4	CntAExtDiv	R/W	0 nresetglobal	Divide PA(0) frequency by 2 (1=divide)
3	CntDEnable	R/W	0 nresetglobal	Enable counter d
2	CntCEnable	R/W	0 nresetglobal	Enable counter c
1	CntBEnable	R/W	0 nresetglobal	Enable counter b
0	CntAEnable	R/W	0 nresetglobal	Enable counter a

Table 20-9. **RegCntOn**

20.4 Interrupts and events map

Interrupt source	Default mapping in the interrupt manager	Default mapping in the event manager
IrqA	RegIrqHigh(4)	RegEvn(7)
IrqB	RegIrqLow(5)	RegEvn(3)
IrqC	RegIrqHigh(3)	RegEvn(6)
IrqD	RegIrqLow(4)	RegEvn(2)

Table 20-10. Default interrupt and event mapping.

20.5 Block schematic

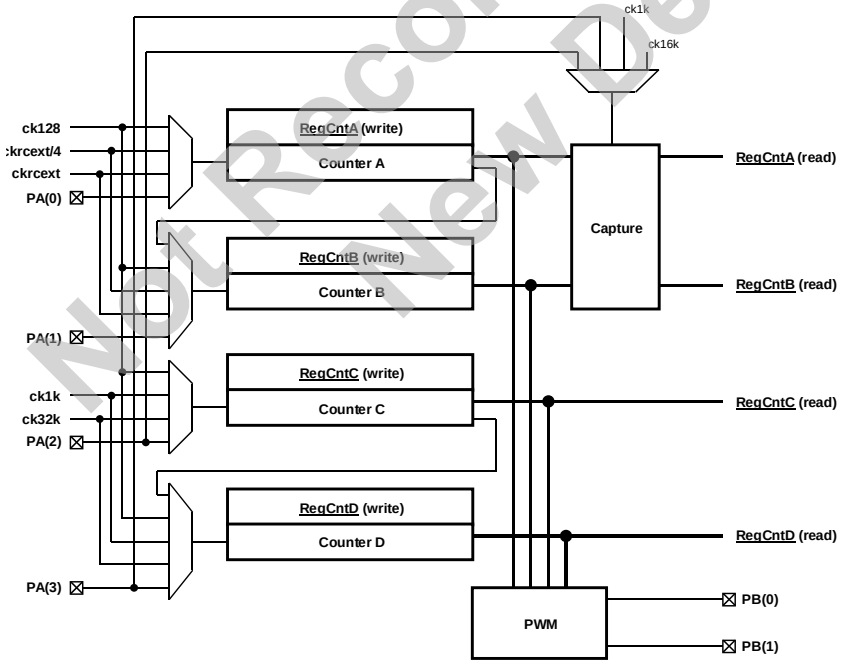


Figure 20-1: Counters/timers block schematic

20.6 General counter registers operation

Counters are enabled by **CntAEnable**, **CntBEnable**, **CntCEnable**, and **CntDEnable** in **RegCntOn**.

To stop the counter X, **Cnt**X**Enable** must be reset. To start the counter X, **Cnt**X**Enable** must be set. When counters are cascaded, **CntAEnable** and **CntCEnable** also control respectively the counters B and D.

WARNING: THERE SHOULD BE AT LEAST ONE CPU INSTRUCTION BETWEEN THE CONFIGURATION OF THE COUNTERS AND THE ENABLING.

All counters have a corresponding 8-bit read/write register: **RegCntA**, **RegCntB**, **RegCntC**, and **RegCntD**. When read, these registers contain the counter value (or the captured counter value). When written, they modify the counter comparison values.

It is possible to read any counter at any time, even when the counter is running. The value is guaranteed to be correct when the counter is running on an internal clock source. For a correct acquisition of the counter value when running on an external clock source, use one of the three following methods:

1) For slow operating counters (typically at least 8 times slower than the CPU clock), oversample the counter content and perform a majority operation on the consecutive read results to select the correct actual content of the counter.

2) Stop the concerned counter, perform the read operation and restart the counter. While stopped, the counter content is frozen and the counter does not take into account the clock edges delivered on the external pin.

3) Use the capture mechanism.

When a value is written into the counter register while the counter is in counter mode, both the comparison value is updated and the counter value is modified. In upcount mode, the register value is reset to zero. In downcount mode, the comparison value is loaded into the counter. Due to the synchronization mechanism between the processor clock domain and the external clock source domain, this modification of the counter value can be postponed until the counter is enabled and that it receives it's first valid clock edge.

In the PWM mode or in the capture mode, the counter value is not modified by the write operation in the counter register. Changing to the counter mode, does not update the counter value (no reset in upcount, no load in downcount mode).

20.7 Clock selection

The clock source for each counter can be individually selected by writing the appropriate value in the register **RegCntCtrlCk**.

CntX**CkSel(1:0)**	**Clock source for**			
	CounterA	**CounterB**	**CounterC**	**CounterD**
11	Ck128			
10	CkRcExt/4		Ck1k	
01	CkRcExt		Ck32k	
00	PA(0)	PA(1)	PA(2)	PA(3)

Table 20-11: Clock sources for counters A, B, C and D

Table 20-11 gives the correspondence between the binary codes used for the configuration bits **CntACkSel(1:0)**, **CntBCkSel(1:0), CntCCkSel(1:0)** or **CntDCkSel(1:0)** and the clock source selected respectively for the counters A, B, C or D.

The CkRcExt clock is the RC oscillator or external clock. The clocks below 32kHz can be derived from the RC oscillator, the external clock source or the crystal oscillator (see the documentation of the clock block). A separate external clock source can be delivered on PortA for each individula counter.

The external clock sources can be debounced or not by properly setting the PortA configuration registers. Additionally, the external clock sources can be divided by two in the counter block, thus enabling higher external clock frequencies, by setting the **Cnt**X**ExtDiv** bits in the **RegCntOn** register.

Switching between an internal and an external clock source can only be performed while the counter is stopped. The enabling or disabling of the external clock frequency division can only be performed while the counter using this clock is stopped, or when this counter is running on an internal clock source.

20.8 Counter mode selection

Each counter can work in one of the following modes:

1) Counter, downcount & upcount
2) Captured counter, downcount & upcount (only counters A&B)
3) PWM, downcount & upcount
4) Captured PWM, downcount and upcount

The counters A and B or C and D can be cascaded or not. In cascaded mode, A and C are the LSB counters while B and D are the MSB counters.

Table 20-12 shows the different operation modes of the counters A and B as a function of the mode control bits. For all counter modes, the source of the down or upcount selection is given (either the bit **CntADownUp** or the bit **CntBDownUp**). Also, the mapping of the interrupt sources IrqA and IrqB and the PWM output on PB(0) in these different modes is shown.

CascadeAB	CountPWM0	CapFunc(1:0)	Counter A mode	Counter B mode	IrqA source	IrqB source	PB(0) function
0	0	00	Counter 8b Downup: A	Counter 8b Downup: B	Counter A	Counter B	PB(0)
1	0	00	Counter 16b AB Downup: A		Counter AB	-	PB(0)
0	1	00	PWM 8b Downup: A	Counter 8b Downup: B	-	Counter B	PWM A
1	1	00	PWM 10 – 16b AB Downup A		-	-	PWM AB
0	0	1x or x1	Captured counter 8b Downup: A	Captured counter 8b Downup: B	Capture A	Capture B	PB(0)
1	0	1x or x1	Captured counter 16b AB Downup: A		Capture AB	Capture AB	PB(0)
0	1	1x or x1	Captured PWM 8b Downup: A	Captured counter 8b Downup: B	Capture A	Capture B	PWM A
1	1	1x or x1	Captured 10 – 16b PWM (captured value on 16b) Downup: A		Capture AB	Capture AB	PWM AB

Table 20-12: Operating modes of the counters A and B

Table 20-13 shows the different operation modes of the counters C and D as a function of the mode control bits. For all counter modes, the source of the down or upcount selection is given (either the bit **CntCDownUp** or the bit

CntDDownUp). The mapping of the interrupt sources IrqC and IrqD and the PWM output on PB(1) in these different modes is also shown.

The switching between different modes must be done while the concerned counters are stopped. While switching capture mode on and off, unwanted interrupts can appear on the interrupt channels concerned by this mode change.

CascadeCD	CountPWM1	Counter C mode	Counter D mode	IrqC source	IrqD source	PB(1) function
0	0	Counter 8b Downup: C	Counter 8b Downup: D	Counter C	Counter D	PB(1)
1	0	Counter 16b CD Downup: C		Counter CD	-	PB(1)
0	1	PWM 8b Downup: C	Counter 8b Downup: D	-	Counter D	PWM C
1	1	PWM 10 – 16b CD Downup: C		-	-	PWM CD

Table 20-13: Operating modes of the counters C and D

20.9 Counter / Timer mode

The counters in counter / timer mode are generally used to generate interrupts after a predefined number of clock periods applied on the counter clock input.

Each counter can be set individually either in upcount mode by setting **Cnt**X**DownUp** in the register **RegCntConfig1** or in downcount mode by resetting this bit. Counters A and B can be cascaded to behave as a 16 bit counter by setting **CascadeAB** in the **RegCntConfig1** register. Counters C and D can be cascaded by setting **CascadeCD**. When cascaded, the up/down count modes of the counters B and D are defined respectively by the up/down count modes set for the counters A and C.

When in upcount mode, the counter will start incrementing from zero up to the target value which has been written in the corresponding **RegCnt**X register(s). When the counter content is equal to the target value, an interrupt is generated at the next counter clock pulse and the counter is loaded again with the zero value (Figure 20-2).

When in downcount mode, the counter will start decrementing from the initial load value which has been written in the corresponding **RegCnt**X register(s) down to the zero value. Once the counter content is equal to zero, an interrupt is generated at the next counter clock pulse and the counter is loaded again with the load value (Figure 20-2).

Be careful to select the counter mode (no capture, not PWM, specify cascaded or not and up or down counting mode) before writing any target or load value to the **RegCnt**X register(s). This ensures that the counter will start from the correct initial value. When counters are cascaded, both counter registers must be written to ensure that both cascaded counters will start from the correct initial values.

The stopping and consecutive starting of a counter in counter mode without a target or load value write operation in between can generate an interrupt if this counter has been stopped at the zero value (downcount) or at it's target value (upcount). This interrupt is additional to the interrupt which has already been generated when the counter reached the zero or the target value.

Due to the synchronization between the CPU clock and the counter clock source, it may take up to 1 CPU clock cycle before the configuration changes written in the **RegCntConfig**X or **RegCnt**X registers are becoming effective. In order to get correct operation of the counters, there should be at least 1 software instruction between the modification of **RegCntConfig**X or **RegCnt**X and the enabling of the counters.

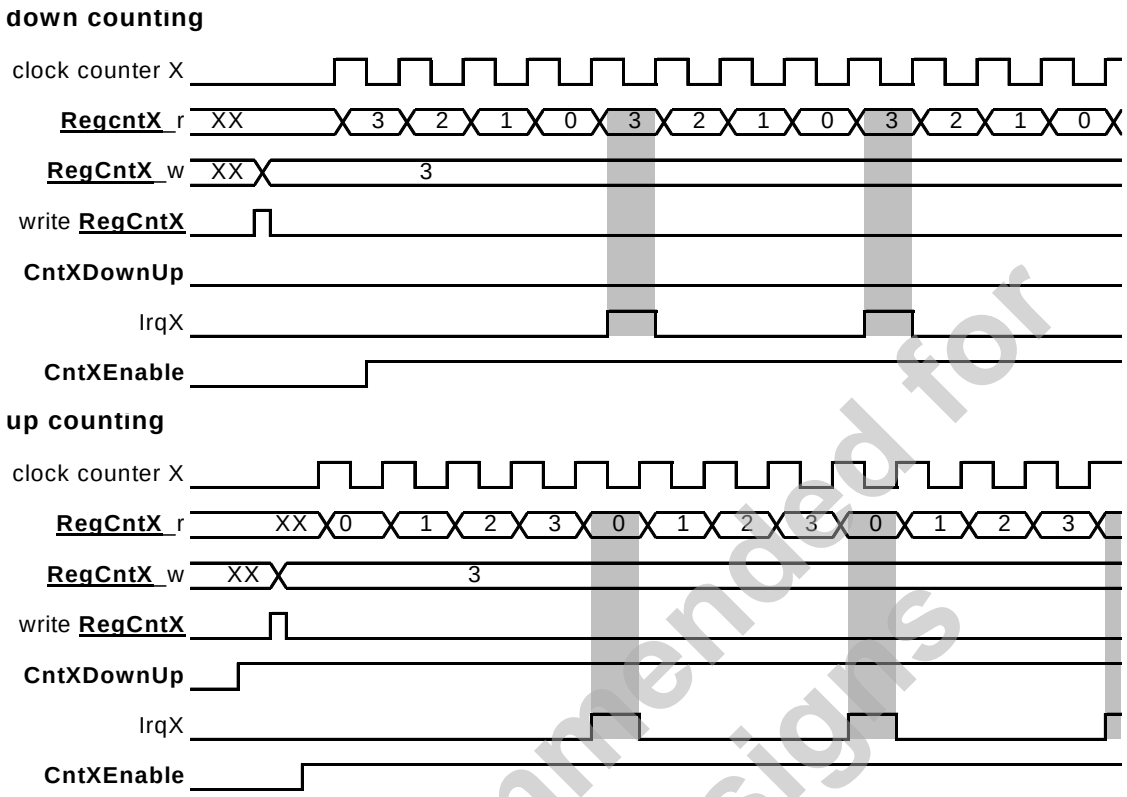


Figure 20-2. Up and down count interrupt generation.

20.10 PWM mode

The counters can generate PWM signals (Pulse Width Modulation) on the PortB outputs PB(0) and PB(1).

The PWM mode is selected by setting **CntPWM1** and **CntPWM0** in the **RegCntConfig1** register. See Table 20-12 and Table 20-13 for an exact description of how the setting of **CntPWM1** and **CntPWM0** affects the operating mode of the counters A, B, C and D according to the other configuration settings.

When **CntPWM0** is enabled, the PWMA or PWMAB output value overrides the value set in bit 0 of **RegPBOut** in the Port B peripheral. When **CntPWM1** is enabled, the PWMC or PWMCD output value overrides the value set in bit 1 of **RegPBOut**. The corresponding ports (0 and/or 1) of Port B must be set in digital mode and as output and either open drain or not and pull up or not through a proper setting of the control registers of the Port B.

Counters in PWM mode count down or up, according to the **Cnt**X**DownUp bit** setting. No interrupts and events are generated by the counters that are in PWM mode. Counters do count circularly: they restart at zero or at the maximal value (either 0xFF when not cascaded or 0xFFFF when cascaded) when respectively an overflow or an underflow condition occurs in the counting.

The internal PWM signals are low as long as the counter contents are higher than the PWM code values written in the **RegCnt**X registers. They are high when the counter contents are smaller or equal to these PWM code values. In order to have glitch free outputs, the PWM outputs on PB(0) and PB(1) are sampled versions of these internal PWM signals, therefore delayed by one counter clock cycle.

The PWM resolution is always 8 bits when the counters used for the PWM signal generation are not cascaded. **PWM0Size(1:0)** and **PWM1Size(1:0)** in the **RegCntConfig2** register are used to set the PWM resolution for the counters A and B or C and D respectively when they are in cascaded mode. The different possible resolutions in cascaded mode are shown in Table 20-14. Choosing a 16 bit PWM code which is higher than the maximum value that can be represented by the number of bits chosen for the resolution results in a PWM output which is always tied to 1.

PwmXsize(1:0)	Resolution
11	16 bits
10	14 bits
01	12 bits
00	10 bits

Table 20-14: Resolution selection in cascaded PWM mode

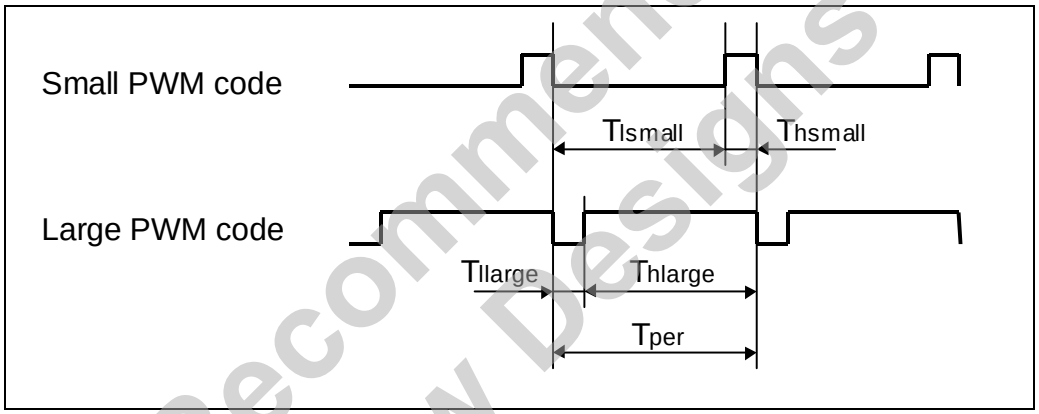


Figure 20-3: PWM modulation examples

The period of the PWM signal is given by the formula:

$$Tper = \frac{2^{resolution}}{f_{ckcnt}}$$

The duty cycle ratio DCR of the PWM signal is defined as:

$$DCR = \frac{Th}{Tper}$$

DCR can be selected between $\frac{100}{2^{resolution}}$ % and 100 %.

DCR in % in function of the RegCntX content(s) is given by the relation:

$$DCR = MIN\left(\frac{100(1+\mathrm{RegCntX})}{2^{resolution}}, 100\right)$$

20.11 Capture function

The 16-bit capture register is provided to facilitate frequency measurements. It provides a safe reading mechanism for the counters A and B when they are running. When the capture function is active, the processor does not read anymore the counters A and B directly, but instead reads shadow registers located in the capture block. An interrupt is generated after a capture condition has been met when the shadow register content is updated. The capture condition is user defined by selecting either internal capture signal sources derived from the prescaler or from the external PA(2) or PA(3) ports. Both counters use the same capture condition.

When the capture function is active, the A and B counters can either upcount or downcount. They do not count circularly: they restart at zero or at the maximal value (either 0xFF when not cascaded or 0xFFFF when cascaded) when respectively an overflow or an underflow condition occurs in the counting. The capture function is also active on the counters when used to generate PWM signals.

CapFunc(1:0) in register **RegCntConfig2** determines if the capture function is enabled or not and selects which edges of the capture signal source are valid for the capture operation. The source of the capture signal can be selected by setting **CapSel(1:0)** in the **RegCntConfig2** register. For all sources, rising, falling or both edge sensitivity can be selected. Table 20-15 shows the capture condition as a function of the setting of these configuration bits.

CapSel(1:0)	Selected capture signal	CapFunc	Selected condition	Capture condition
11	1 K	00	**Capture disabled**	-
		01	Rising edge	1 K rising edge
		10	Falling edge	1 K falling edge
		11	Both edges	2 K
10	16 K	00	**Capture disabled**	-
		01	Rising edge	16 K rising edge
		10	Falling edge	16 K falling edge
		11	Both edges	32 K
01	PA3	00	**Capture disabled**	-
		01	Rising edge	PA3 rising edge
		10	Falling edge	PA3 falling edge
		11	Both edges	PA3 both edges
00	PA2	00	**Capture disabled**	-
		01	Rising edge	PA2 rising edge
		10	Falling edge	PA2 falling edge
		11	Both edges	PA2 both edges

Table 20-15: Capture condition selection

CapFunc(1:0) and **CapSel(1:0)** can be modified only when the counters are stopped otherwise data may be corrupted during one counter clock cycle.

Due to the synchronization mechanism of the shadow registers and depending on the frequency ratio between the capture and counter clocks, the interrupts may be generated one or only two counter clock pulses after the effective capture condition occurred. When the counters A and B are not cascaded and do not operate on the same clock, the interruptions on IrqA and IrqB which inform that the capture condition was met, may appear at different moments. In this case, the processor should read the shadow register associated to a counter only if the interruption related to this counter has been detected.

An edge is detected on the capture signals only if the minimal pulse widths of these signals in the low and high states are higher than a period of the counter clock source.

20.12 Specifications

Parameter	Min	Typ	Max	Unit	Conditions
Pulse width in the low and high states for an external clock source, frequency division by 2 disabled	500			ns	@ 1.2V
	125			ns	@ 2.4V
Pulse width in the low and high states for an external clock source, frequency division by 2 enabled	100			ns	@ 1.2V
	25			ns	@ 2.4V
Pulse width of external capture signals	$\frac{1}{fckcnt}$			s	

Table 20-16: Timing specifications for the counters

21. The Voltage Level Detector

21.1 Features

- can be switched off, on or simultaneously with CPU activities
- generates an interrupt if power supply is below a pre-determined level

21.2 Overview

The Voltage Level Detector monitors the state of the system battery. It returns a logical high value (an interrupt) in the status register if the supplied voltage drops below the user defined level (Vsb).

21.3 Register map

There are two registers in the VLD, namely **RegVldCtrl** and **RegVldStat**. Table 21-2 shows the mapping of control bits and functionality of **RegVldCtrl** while Table 21-3 describes that for **RegVldStat**.

register name
RegVldCtrl
RegVldStat

Table 21-1: Vld registers

pos.	RegVldCtrl	rw	reset	function
7-4	--	r	0000	reserved
3	VldRange	r w	0 nresetglobal	VLD detection voltage range for VldTune = “011”: 0 : 1.3V 1 : 2.55V
2-0	VldTune[2:0]	r w	000 nresetglobal	VLD tuning: 000 : +19 % 111 : -18 %

Table 21-2: RegVldCtrl

pos.	RegVldStat	rw	reset	function
7-3	--	r	00000	reserved
2	VldResult	r	0 nresetglobal	is 1 when battery voltage is below the detection voltage
1	VldValid	r	0 nresetglobal	Indicates when VldResult can be read
0	VldEn	r w	0 nresetglobal	VLD enable

Table 21-3: RegVldStat

21.4 Interrupt map

interrupt source	default mapping in the interrupt manager
IrqVld	RegIrqMid(2)

Table 21-4: Interrupt map

21.5 VLD operation

The VLD is controlled by **VldRange**, **VldTune** and **VldEn**. **VldRange** selects the voltage range to be detected, while **VldTune** is used to fine-tune this voltage level in 8 steps. **VldEn** is used to enable (disable) the VLD with a 1(0) value respectively. Disabled, the block will dissipate no power.

symbol	description	min	typ	max	unit	comments	
Vth	Threshold voltage	Note 1			V	trimming values:	
						VldRange	VldTune
			1.53			0	000
			1.44			0	001
			1.36			0	010
			1.29			0	011
			1.22			0	100
			1.16			0	101
			1.11			0	110
			1.06			0	111
			3.06			1	000
			2.88			1	001
			2.72			1	010
			2.57			1	011
			2.44			1	100
			2.33			1	101
			2.22			1	110
			2.13			1	111
T_{EOM}	duration of measurement		2.0	2.5	ms	Note 2	
T_{PW}	Minimum pulse width detected		875	1350	us	Note 2	

Table 21-5: Voltage level detector operation

Note 1: absolute precision of the threshold voltage is ±10%.
Note 2: this timing is respected in case the internal RC or crystal oscillators are selected. Refer to the clock block documentation in case the external clock is used.

To start the voltage level detection, the user sets bit **VldEn**. The measurement is started. After 2ms, the bit **VldValid** is set to indicate that the measurement results are valid. From that time on, as long as the VLD is enabled, a maskable interrupt request is sent if the voltage level falls below the threshold. One can also poll the VLD and monitor the actual measurement result by reading the **VldResult** bit of the **RegVldStat**. This result is only valid as long as the **VldValid** bit is '1'.

Figure 21-1 shows the timing of the VLD. An interrupt is generated on each rising edge of **VldResult**.

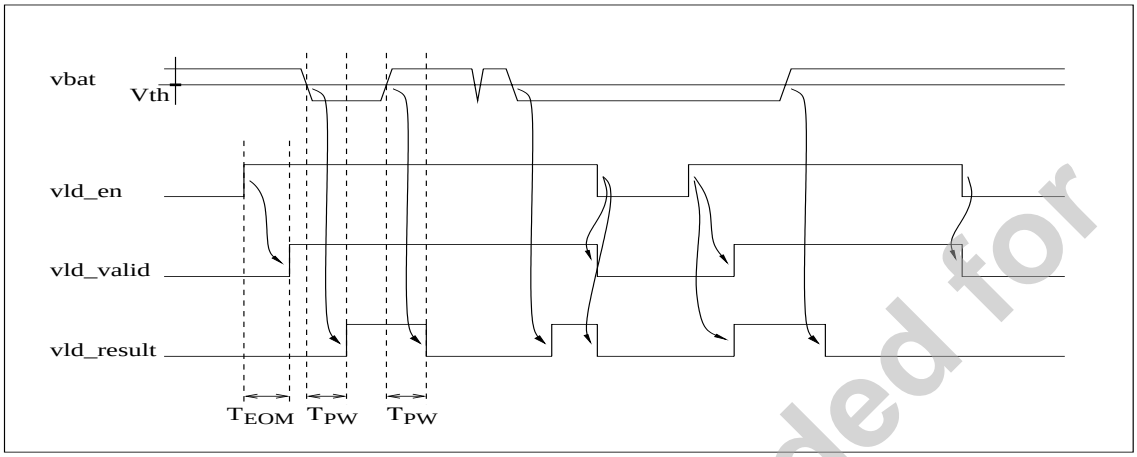


Figure 21-1: VLD timing

The threshold value should not be changed during the measurement.

22. Low Power Comparators

22.1 Features

The cmpd peripheral implements four low power comparators.

- Quiescent current consumption of 1.5µA
- Very low switching current
- Per channel configurable interrupt
- Hysteresis
- 1 MHz operation

22.2 Overview

Figure 22-1 gives an overview of this block:

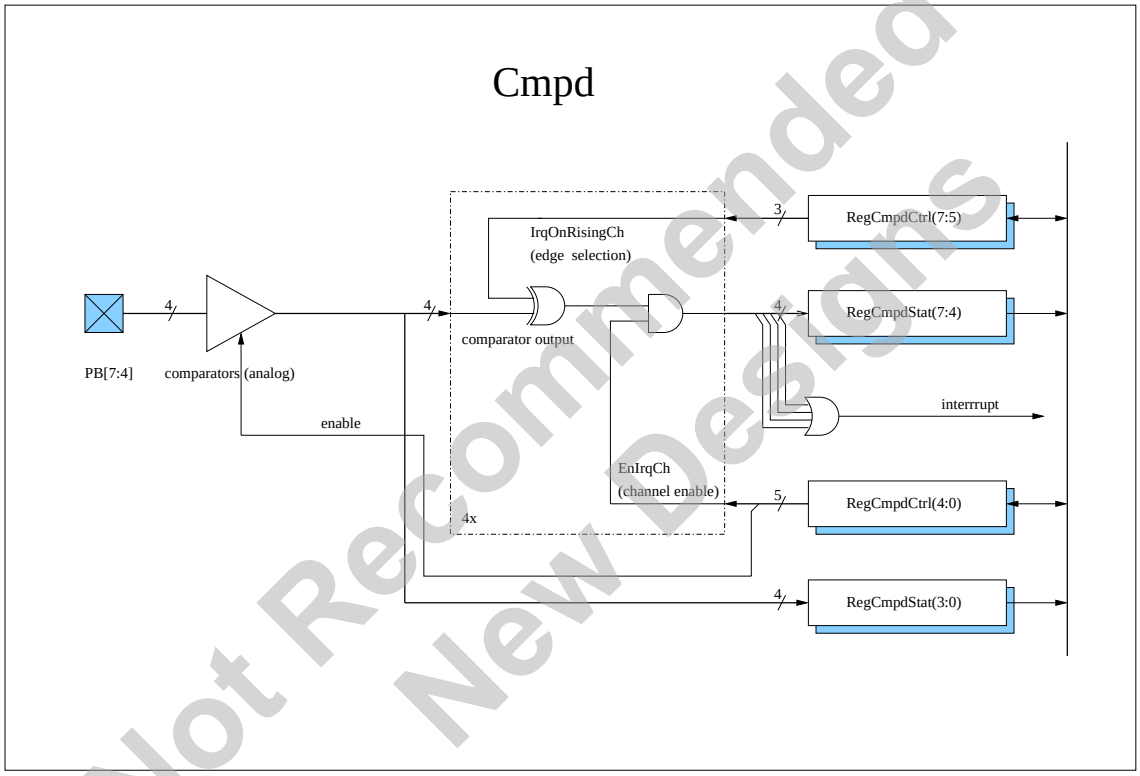


Figure 22-1: Structure of Cmpd

The cmpd peripheral is a 4-channel low power comparator. It is intended to compare analog input signals with an internally set threshold voltage. The comparator maintains low current consumption even if the input signal is very close to the threshold. The comparison result of each channel can be used to generate an interrupt and/or is available for polling.

The comparator can be enabled or disabled by programming the **Enable** bit in the **RegCmpdCtrl** register. When disabled, the block consumes no current.

The peripheral has a single interrupt output which is a combination of the four channels. The combination can be chosen by programming the **RegCmpdCtrl** register. The **EnIrqCh[3:0]** bits select the channel that can activate the interrupt. The **IrqOnRisingCh[2:0]** bits indicate if the interrupt is generated on detection of the rising or falling edge of the channel.

The comparison results of the peripheral can be read in the **RegCmpdStat** register. The bits **CmpdOut[3:0]** are the value of the comparisons at the moment the register is read. The **CmpdStat[3:0]** indicates which channel generated an interrupt since the register was last read.

Comparator specifications:

Sym	description	min	typ	max	unit	comments
t_{pulse}	Required input pulse width	500			ns	VBAT≥1.2V
IDD_q	Quiescent current		0.8	1.5	µA	1
IDD_{stat}	Maximal static current		1.5		µA	2
V_{th}	Threshold voltage	0.7		1.1	V	3
$\Delta V_{th}/\Delta T$	Threshold temperature drift		-0.9		mV/°C	
V_{hyst}	Threshold hysteresis		13		mV	

Table 22-1: Comparator specifications

Comments:
1. The quiescent current is defined for a static input voltage <0.5V or >1.3V. The specified consumption is the sum for all 4 channels.
2. The maximal static current is defined for any static input voltage between VDD and VSS. The specified consumption is the sum for all 4 channels.
3. Defined with respect to VSS.

How to start the cmpd:
To avoid unwanted irqs one has first to configure the rising / falling edge of the detection (bit IrqOnRisingCh[2:0]) and to enable the comparator (bit Enable). Only after that may the user enable the channel interrupts with bit EnIrqCh[3:0].

22.3 Register map

There are two registers in the Cmpd, namely **RegCmpdStat** and **RegCmpdCtrl**. Table 22-3 and Table 22-4 show the mapping of the control bits and the functionality of these registers.

register name
RegCmpdStat
RegCmpdCtrl

Table 22-2: Cmpd registers

pos.	RegCmpdStat	rw	reset	function
7	CmpdStat[3]	rc	0 nresetglobal	1: if the channel 3 generated an interrupt since last read of this register
6	CmpdStat[2]	rc	0 nresetglobal	1: if the channel 2 generated an interrupt since last read of this register
5	CmpdStat[1]	rc	0 nresetglobal	1: if the channel 1 generated an interrupt since last read of this register
4	CmpdStat[0]	rc	0 nresetglobal	1: if the channel 0 generated an interrupt since last read of this register
3	CmpdOut[3]	r	0 nresetglobal	Channel 3 comparator output
2	CmpdOut[2]	r	0 nresetglobal	Channel 2 comparator output
1	CmpdOut[1]	r	0 nresetglobal	Channel 1 comparator output
0	CmpdOut[0]	r	0 nresetglobal	Channel 0 comparator output

Table 22-3: RegCmpdStat

pos.	RegCmpdCtrl	rw	reset	function
7	IrqOnRisingCh[2]	rw	0 nresetglobal	1: an interrupt is generated on the rising edge of channels 2 and 3. 0: an interrupt is generated on the falling edge of channels 2 and 3.
6	IrqOnRisingCh[1]	rw	0 nresetglobal	1: an interrupt is generated on the rising edge of channel 1. 0: an interrupt is generated on the falling edge of channel 1.
5	IrqOnRisingCh[0]	rw	0 nresetglobal	1: an interrupt is generated on the rising edge of channel 0. 0: an interrupt is generated on the falling edge of channel 0.
4	EnIrqCh[3]	rw	0 nresetglobal	1 enables interrupt on channel 3
3	EnIrqCh[2]	rw	0 nresetglobal	1 enables interrupt on channel 2
2	EnIrqCh[1]	rw	0 nresetglobal	1 enables interrupt on channel 1
1	EnIrqCh[0]	rw	0 nresetglobal	1 enables interrupt on channel 0
0	Enable	rw	0 nresetglobal	Enables the comparator

Table 22-4: RegCmpdCtrl

22.4 Interrupt map

interrupt source	default mapping in the interrupt manager
cmpd_irq	RegIrqHigh[2]

Table 22-5: Interrupt map

23 Physical Dimensions

23.1 QFP type package

The QFP package dimensions are given in Figure 23-1 and Table 23-1

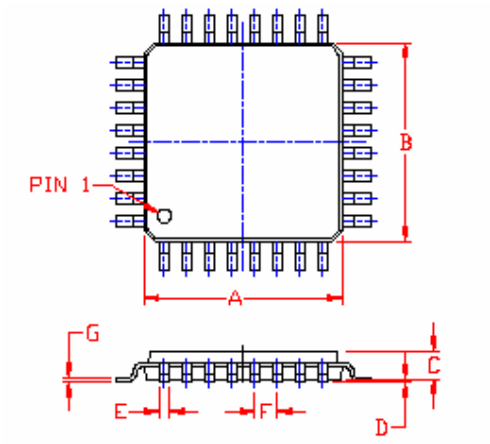


Figure 23-1. QFP type package

package	A mm	B mm	C mm	D mm	E mm	F mm
LQFP-80	14.0	14.0	1.4	0.10	0.32	0.65
LQFP-100	14.0	14.0	1.4	0.10	0.22	0.5

Table 23-1. QFP package dimensions

Contact Information

Semtech Corporation
Wireless and Sensing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone (805) 498-2111 Fax : (805) 498-3804