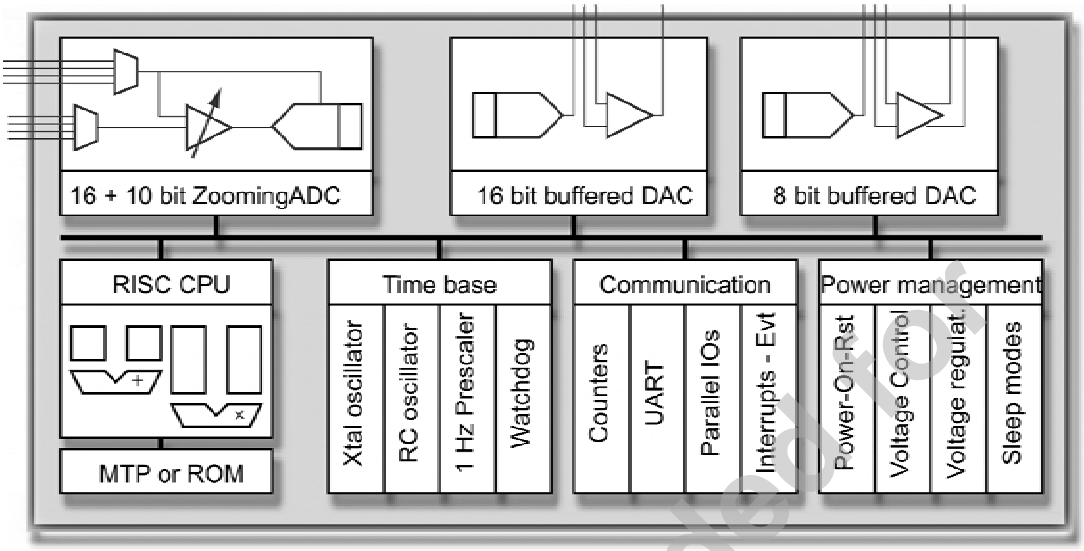
Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



XE8805/05A – SX8805R Sensing Machine – Data Acquisition with 16+10 bit ZoomingADC™ and buffered DACs

General Description

The XE8805A is a data acquisition ultra low-power low-voltage system on a chip (SoC) with a high efficiency microcontroller unit embedded (MCU), allowing for 1 MIPS at 300uA and 2.4 V, and multiplying in one clock cycle.

The XE8805A includes a high resolution acquisition path with the 16+10 bits ZoomingADC and two buffered DACs.

The XE8805A is available with on chip ROM (the SX8805) or Multiple-Time-Programmable (MTP) program memory.

Applications

- Portable, battery operated instruments
- Current loop powered instruments
- Wheatstone bridge interfaces
- Pressure and chemical sensors
- HVAC control
- Metering
- Sports watches, wrist instruments

Key product Features

- Low-power, high resolution ZoomingADC
  - 0.5 to 1000 gain with offset cancellation
  - up to 16 bits analog to digital converter
  - up to 13 inputs multiplexer
- Low-voltage low-power controller operation
  - 2 MIPS with 2.4 V to 5.5 V operation
  - 300 µA at 1 MIPS over voltage range
- 22 kByte (8 kInstruction) MTP
- 520 Byte RAM data memory
- RC and crystal oscillators
- 5 reset, 22 interrupt, 8 event sources
- 8 bit and 16 bit buffered DACs
- 100 years MTP Flash retention at 55°C

Ordering Information

Product	Temperature range	Memory type	Package
XE8805MI028*	-40°C to 85 °C	MTP	LQFP64
XE8805AMI000	-40°C to 85 °C	MTP	die
XE8805AMI028LF	-40°C to 85 °C	MTP	LQFP64
SX8805Rxxx	-40°C to 85 °C**	ROM	

*Not for new designs
**Extended temperature range on request

TABLE OF CONTENTS

1. General Overview

CONTENTS

1.1 Top schematic

1.1.1 General description

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 RESET, 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 OSCIN 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 very often 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, and PWM block also make use of this port.

The instruction memory is a 22-bit wide flash or ROM memory depending on the circuit version. Flash and ROM versions have both 8k instruction memory. The data memory of this product is 512 byte SRAM.

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

The signal D/A (DAS) is a 16 bit D/A based on sigma-delta modulation. It includes a stand-alone amplifier that can be used for analog output filtering.

The bias D/A (DAB) is an 8 bit low frequency D/A. It includes a stand-alone amplifier that is used to drive large currents. It can be used to bias a sensor.

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 C is a general purpose 8 bit parallel I/O port.

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

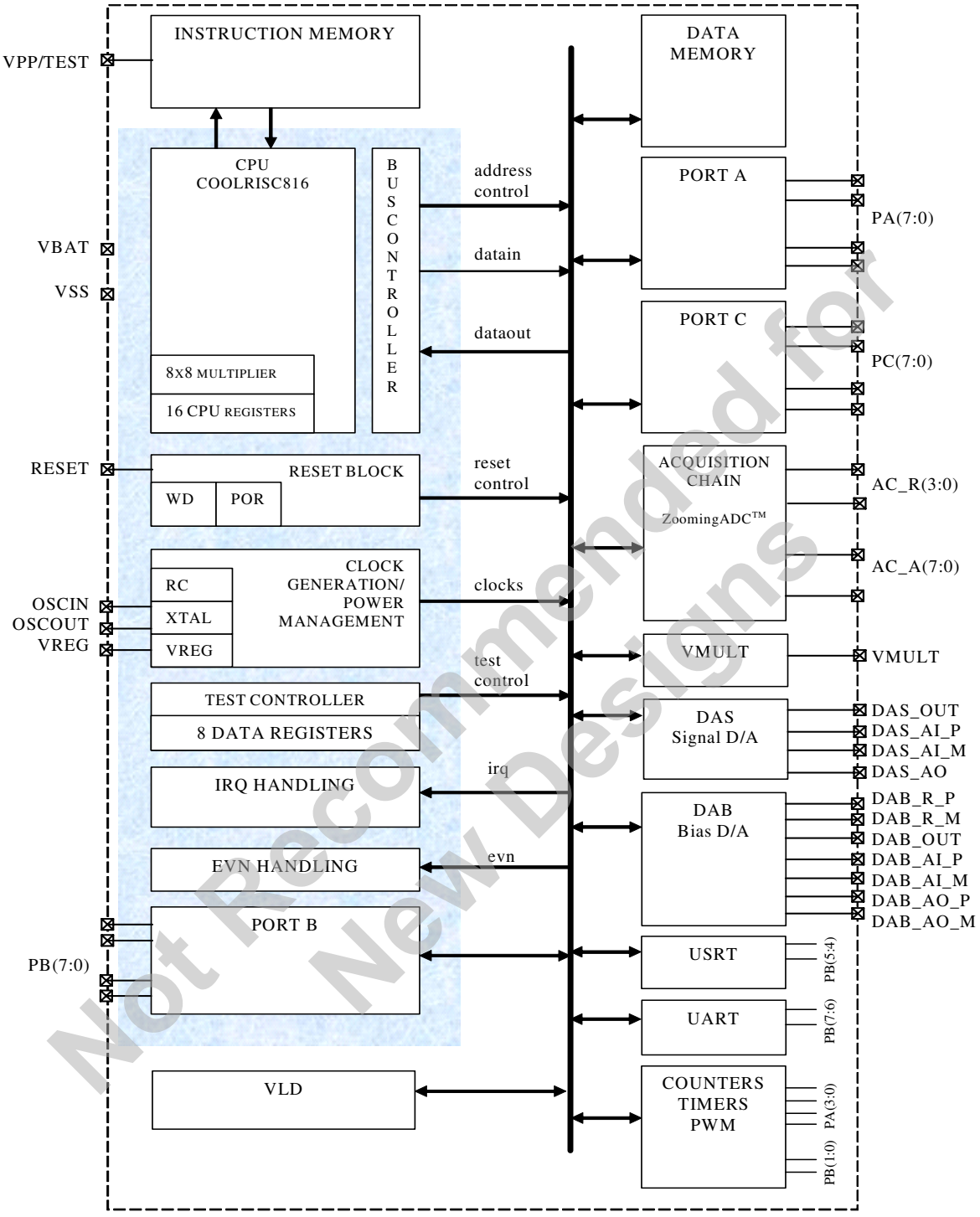


Figure 1-1. Block schematic of the XE8805/05A circuit.

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.

1.1.2 XE8805 vs XE8805A

The XE8805A has a new RESET pin function. The action of the RESET pin of the XE8805A resets the clock registers too and creates an additional short delay. See the RESET chapter for more information.

1.2 Pin map

1.2.1 Bare die

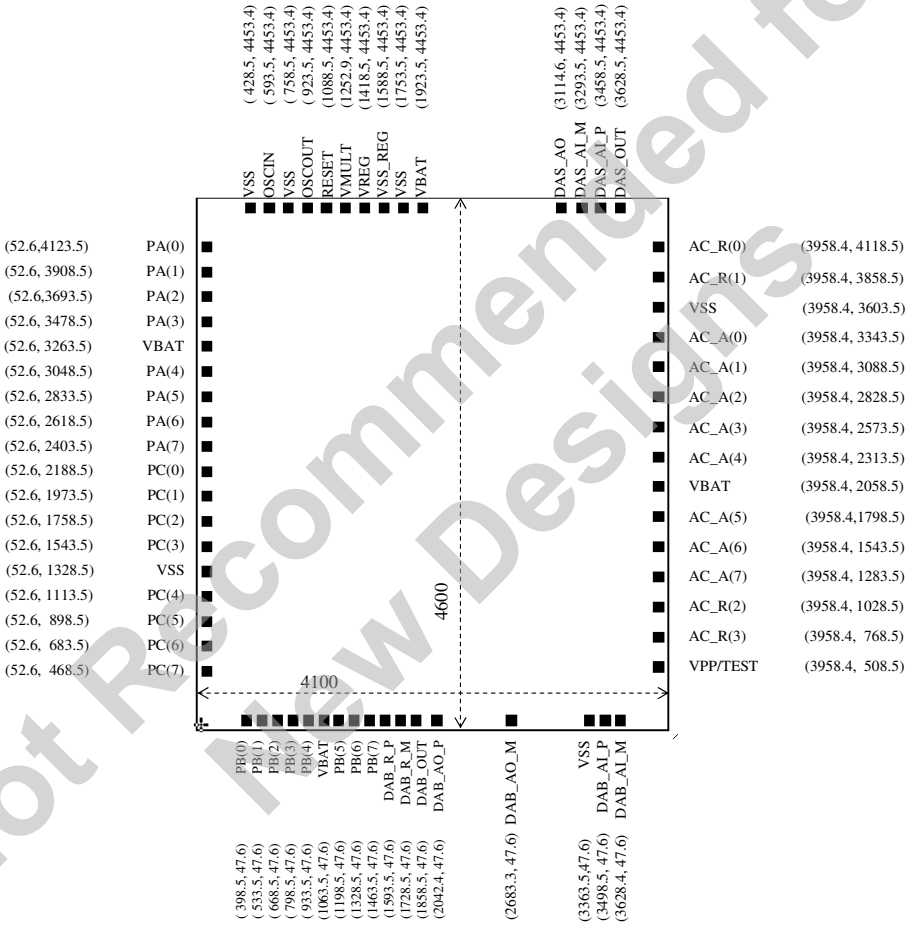


Figure 1-2. Die dimensions and pin coordinates (in µm)

1.2.2 LQFP-64

The XE8805/05A is delivered in a LQFP-64 package. The pin map is given below.

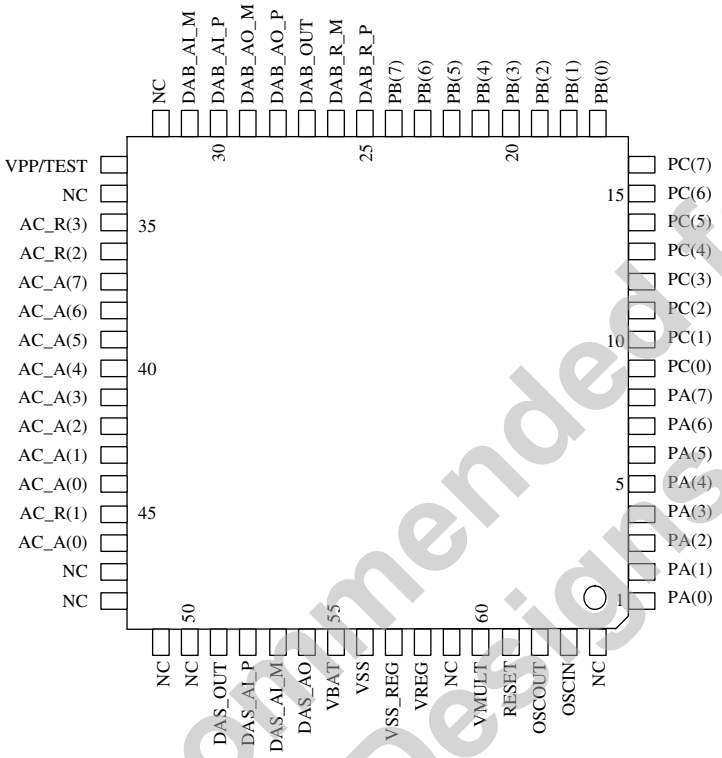


Figure 1-3. LQFP-64 pin map

Package pin	name	Package pin	name
1	PA(0)	33	VPP/TEST
2	PA(1)	34	NC
3	PA(2)	35	AC_R(3)
4	PA(3)	36	AC_R(2)
5	PA(4)	37	AC_A(7)
6	PA(5)	38	AC_A(6)
7	PA(6)	39	AC_A(5)
8	PA(7)	40	AC_A(4)
9	PC(0)	41	AC_A(3)
10	PC(1)	42	AC_A(2)
11	PC(2)	43	AC_A(1)
12	PC(3)	44	AC_A(0)
13	PC(4)	45	AC_R(1)
14	PC(5)	46	AC_R(0)
15	PC(6)	47	NC
16	PC(7)	48	NC
17	PB(0)	49	NC
18	PB(1)	50	NC
19	PB(2)	51	DAS_OUT
20	PB(3)	52	DAS_AI_P

Package pin	name	Package pin	name
21	PB(4)	53	DAS_AI_M
22	PB(5)	54	DAS_AO
23	PB(6)	55	VBAT
24	PB(7)	56	VSS
25	DAB_R_P	57	VSS_REG
26	DAB_R_M	58	VREG
27	DAB_OUT	59	NC
28	DAB_AO_P	60	VMULT
29	DAB_AO_M	61	RESET
30	DAB_AI_P	62	OSCOUT
31	DAB_AI_M	63	OSCIN
32	NC	64	NC

Table 1-1. Bonding plan of the LQFP-64 package (LQFP 64L 10x10mm thick 1.6 mm)

1.3 Pin assignment

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

Pin	Assignment
VBAT	Positive power supply
VSS VSS_REG	Negative power supply
VREG	Connection for the mandatory external capacitor of the voltage regulator
VPP/TEST	High voltage supply for flash memory programming (NC in ROM versions)
RESET	Resets the circuit when the voltage is high
OSCIN/OSCOUT	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
PC(7:0)	Parallel I/O port C pins
AC_A(7:0)	Acquisition chain input pins
AC_R(3:0)	Acquisition chain reference pins
VMULT	Connection for the external capacitor if VBAT is below 3V
DAB_OUT	Bias D/A output
DAB_R_x	Bias D/A reference (x=P: plus, x=M: minus)
DAB_Ax_y	Bias D/A amplifier IO (x=I: input, x=O: output ; y=P: plus, y=M: minus)
DAS_OUT	Signal D/A output
DAS_AI_x	Signal D/A amplifier inputs (x=P: plus, x=M: minus)
DAS_AO	Signal D/A amplifier output

Table 1-2. Pin assignment

Table 1-3 gives a more detailed pin map for the different pins. It also indicates the possible I/O configuration of these pins. The indications in blue bold are the configuration at start-up.

The pins CNTx pins are possible counter inputs, PWMx are possible PWM outputs.

pin	function			I/O configuration						
lqfp-64	first	second	third	AI	AO	DI	DO	OD	PU	POWER
1	PA(0)	CNTA				X			X	
2	PA(1)	CNTB				X			X	
3	PA(2)	CNTC				X			X	
4	PA(3)	CNTD				X			X	
5	PA(4)					X			X	
6	PA(5)					X			X	
7	PA(6)					X			X	
8	PA(7)					X			X	
9	PC(0)					X	X			
10	PC(1)					X	X			
11	PC(2)					X	X			
12	PC(3)					X	X			
13	PC(4)					X	X			
14	PC(5)					X	X			
15	PC(6)					X	X			
16	PC(7)					X	X			
17	PB(0)	PWM0		X	X	X	X	X	X	
18	PB(1)	PWM1		X	X	X	X	X	X	
19	PB(2)			X	X	X	X	X	X	
20	PB(3)			X	X	X	X	X	X	
21	PB(4)	USRT_S0		X	X	X	X	X	X	
22	PB(5)	USRT_S1		X	X	X	X	X	X	
23	PB(6)	UART_Tx		X	X	X	X	X	X	
24	PB(7)	UART_Rx		X	X	X	X	X	X	
25	DAB_R_P			X						
26	DAB_R_M			X						
27	DAB_OUT				X					
28	DAB_AO_P				X					
29	DAB_AO_M				X					
30	DAB_AI_P			X						
31	DAB_AI_M			X						
33	VPP	TEST								X
35	AC_R(3)			X						
36	AC_R(2)			X						
37	AC_A(7)			X						
38	AC_A(6)			X						
39	AC_A(5)			X						
40	AC_A(4)			X						
41	AC_A(3)			X						
42	AC_A(2)			X						
43	AC_A(1)			X						
44	AC_A(0)			X						
45	AC_R(1)			X						
46	AC_R(0)			X						
51	DAS_OUT				X					
52	DAS_AI_P			X						
53	DAS_AI_M			X						
54	DAS_AO				X					
55	VBAT									X

pin	function			I/O configuration						
lqfp-64	first	second	third	AI	AO	DI	DO	OD	PU	POWER
56	**VSS**									X
57	**VSS_REG**									X
58	**VREG**				X					
60	**VMULT**				X					
61	**RESET**					X				
62	**OSCOUT**				X					
63	**OSCIN**			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
POWER: power supply

Table 1-3. Pin description table

2 XE8805/05A Performance

2.1 Absolute maximum ratings

Table 2-1. Absolute maximum 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	3.3	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 supply voltage should at least be equal to the supply voltage used during normal operation.
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	2.4	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	2

1. The capacitor may be omitted when VREG is connected to VBAT.
2. 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 after that, the retention time is no longer guaranteed. All qualification tests have been done after 10 cycles.

2.3 Supply configurations

2.3.1 Flash circuit

The flash version of the circuit can be run from a supply between 2.4V and 5.5V (Figure 2-1). The capacitor on VREG has to be connected at all times (value in Table 2-2) to guarantee proper operation of the device. The capacitor on VMULT is only required if the circuit is to be operated below 3V.

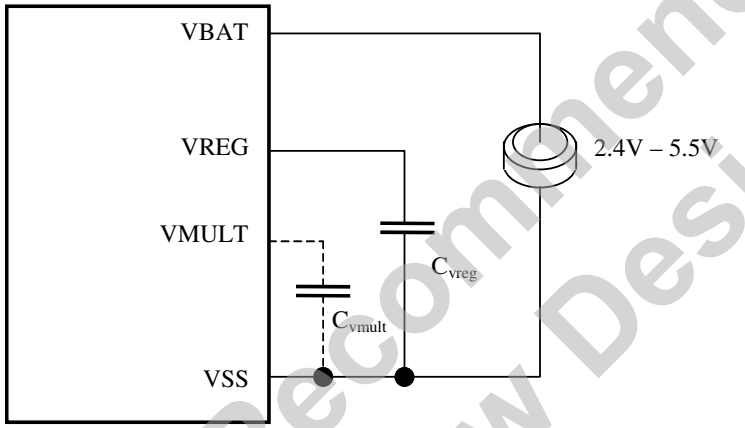


Figure 2-1. Supply configuration for the flash circuit.

2.3.2 ROM circuit

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.

2.3.2.1 Low power operation

In this case, the internal voltage regulator is used in order to maintain low power consumption independent from the supply voltage. The capacitor on VREG has to be connected at all times (value in Table 2-3) to guarantee proper operation of the device. The capacitor on VMULT has to be connected only when VBAT<3V.

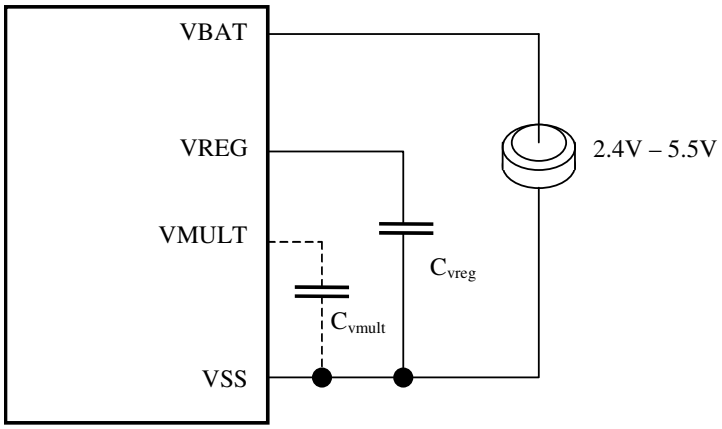


Figure 2-2. Supply voltage connections for low power operation of the ROM version.

2.3.2.2 High speed operation

In this case, the internal voltage regulator is not used. The operation speed of the circuit can be increased with increasing supply voltage but the supply current will also increase. The capacitor on VMULT has to be connected only when VBAT<3V. In this case, the supply voltage can decrease down to 2.15V. Beware however that the zoomingADC™ will not run below 2.4V (see Figure 2-4). In this configuration, the circuit can not be used above 3.3V.

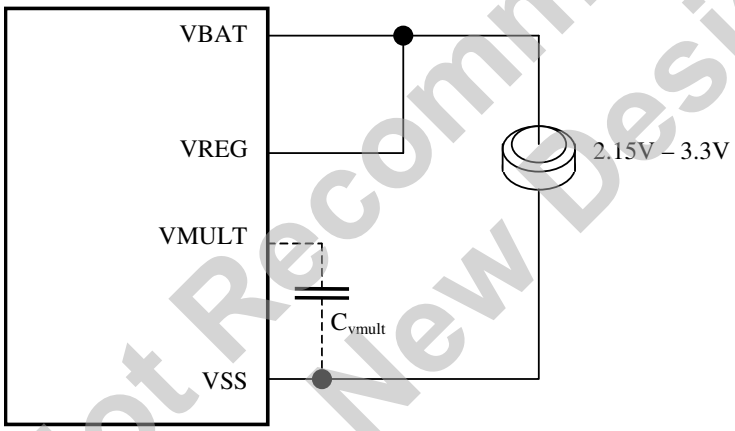


Figure 2-3. Supply voltage connections for high speed operation of the ROM version.

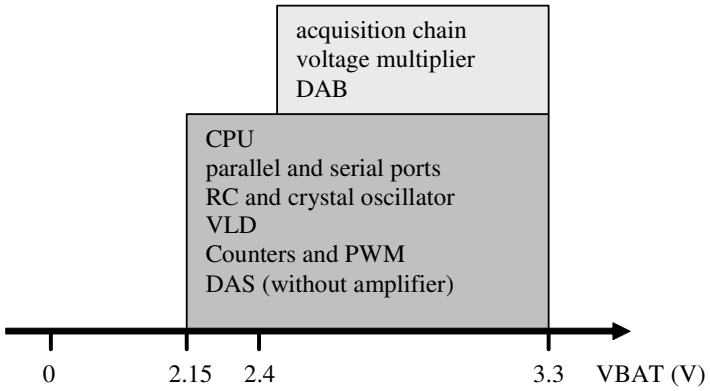


Figure 2-4. Operation range of the different circuit blocks

2.4 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 A and B are configured in input with pull up, the parallel port C is configured as an output. Their pins are not connected externally. The pin RESET is connected to VSS and the pin VPP/TEST is connected to VBAT. The inputs of the acquisition chain are connected to VSS.

Table 2-5. Typical current consumption of the XE8805 version (8k instructions flash memory)

Operation mode	CPU	RC	Xtal	Consumption	comments	Note
High speed CPU	1 MIPS	1 MHz	Off	310 µA	2.4V<>5.5V, 27°C	
Low power CPU	32 kIPS	Off	32 kHz	10 µA	2.4V <>5.5V, 27°C	
Low power time keeping	HALT	Off	32 kHz	1.0 µA	2.4V <>5.5V, 27°C	
Fast wake-up time keeping	HALT	Ready	32kHz	1.7 µA	2.4V <>5.5V, 27°C	
Immediate wake-up time keeping	HALT	100 kHz	Off	1.4 µA	2.4V <>5.5V, 27°C	
VLD static current				15 µA	2.4V <>5.5V, 27°C	
16 bit resolution data acquisition	HALT	2 MHz	Off	190 µA	3.0V, 27°C	1
12 bit , gain 100, data acquisition	HALT	2 MHz	Off	460 µA	3.0V, 27°C	2

1. PGA disabled, ADC enabled, 16 bit resolution
2. PGA 1 disabled, PGA 2 and 3 enabled, ADC enabled, 12 bit resolution

For more information concerning the current consumption of the zoomingADC™, see the chapter power consumption in the acquisition chain documentation which shows the current consumption of this block as a function of temperature and voltage and for different configurations of the PGA and ADC.

The power consumption of the ROM version of the circuit is identical if it is configured as shown in Figure 2-2. In the high speed configuration, the current consumption will increase proportional with VBAT.

2.5 Operating speed

2.5.1 Flash version

The speed of the 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 Figure 2-5.

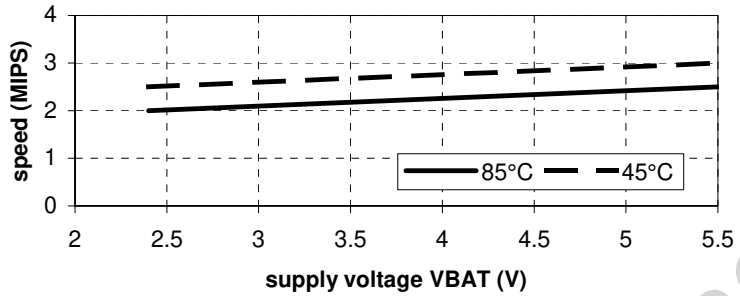


Figure 2-5. Guaranteed speed as a function of the supply voltage and maximal temperature.

2.5.2 ROM circuit version

2.5.2.1 Low power supply configuration

In the low power supply configuration as shown in Figure 2-2, the operating speed does not depend highly on the supply voltage as shown in Figure 2-6.

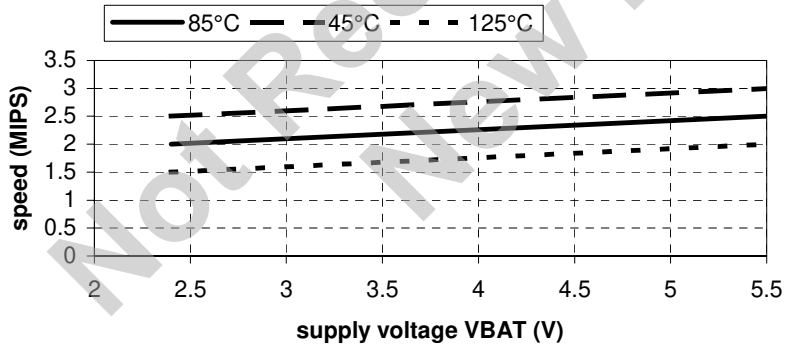


Figure 2-6. Guaranteed speed as a function of supply voltage and for different maximal temperatures using the voltage regulator.

2.5.2.2 High speed supply configuration

In the high speed supply configuration of Figure 2-3, the guaranteed speed of the circuit is shown in Figure 2-7.

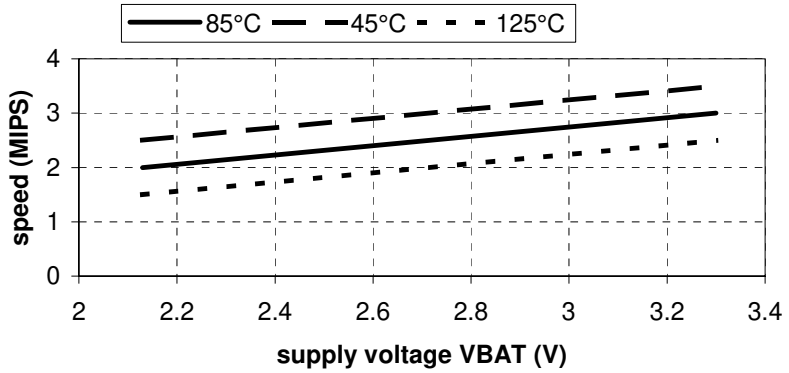


Figure 2-7. Guaranteed speed as a function of supply voltage and for three temperature ranges when VREG=VBAT.

3. CPU

CONTENTS

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.

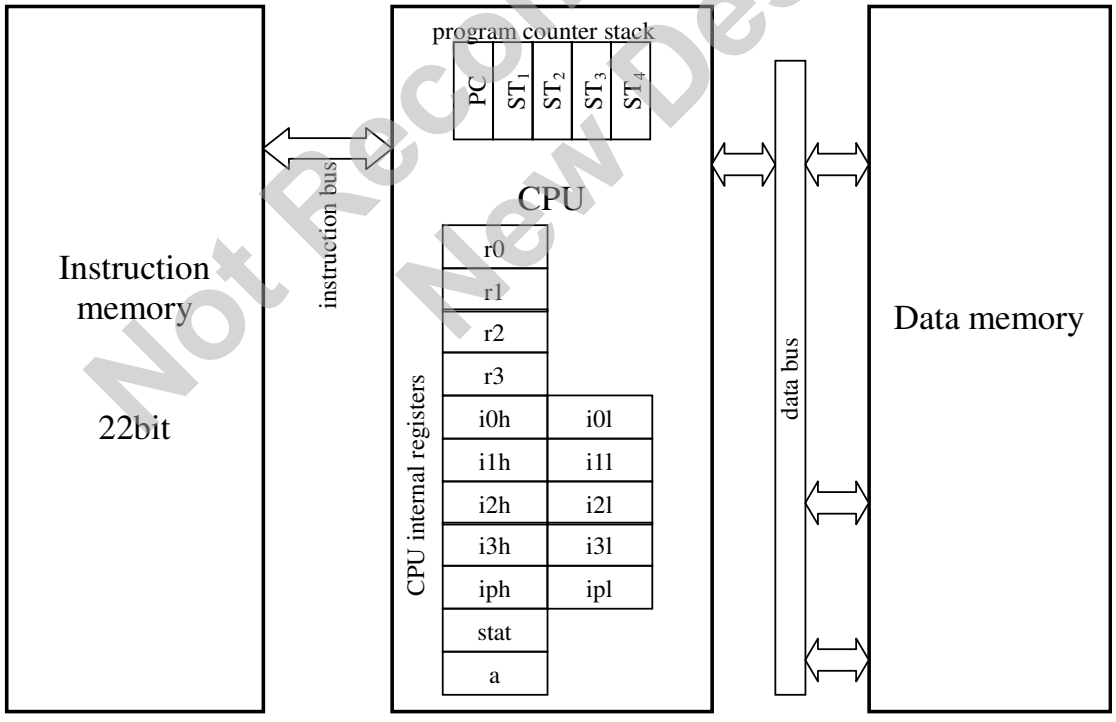


Figure 3-1. CPU internal registers

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 XE8805 CPU is built with a Harvard architecture. The 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. The data memory space has 8 low power registers, the peripheral register space and 512 bytes of RAM.

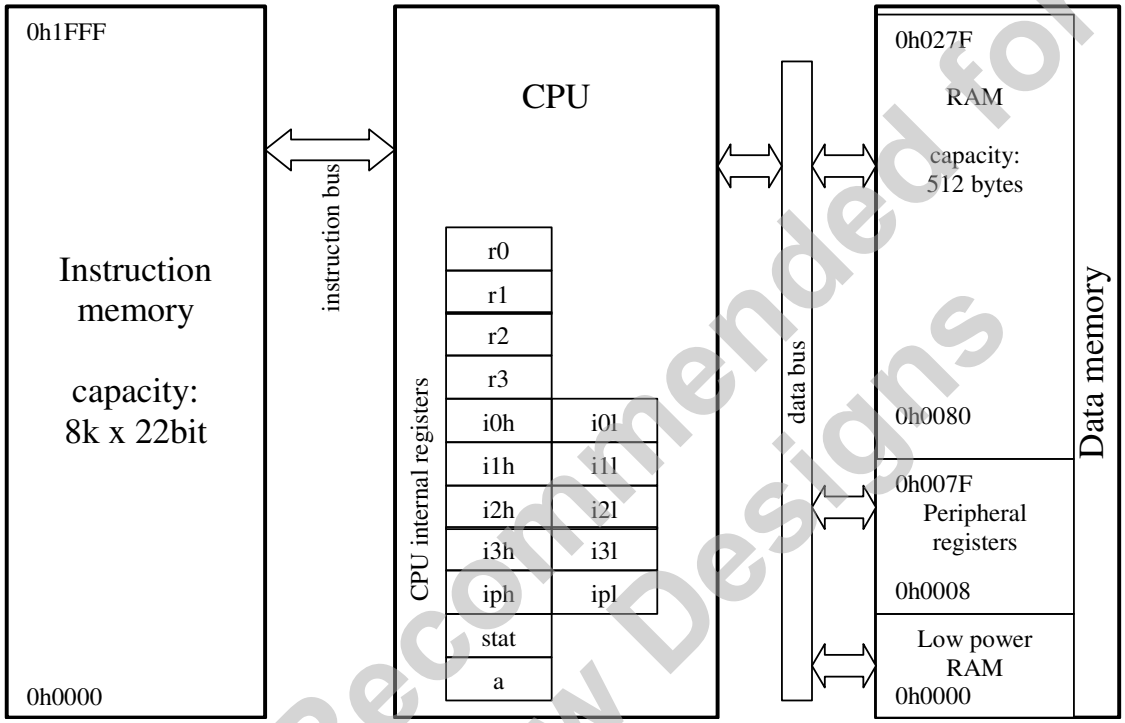


Figure 4-1. Memory mapping

The CPU internal registers are described in the CPU chapter. A short reference of the low power registers and peripheral registers is given in 4.2.

4.2 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 4-1 for code description)
4. The reset source and reset value of the different bits

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