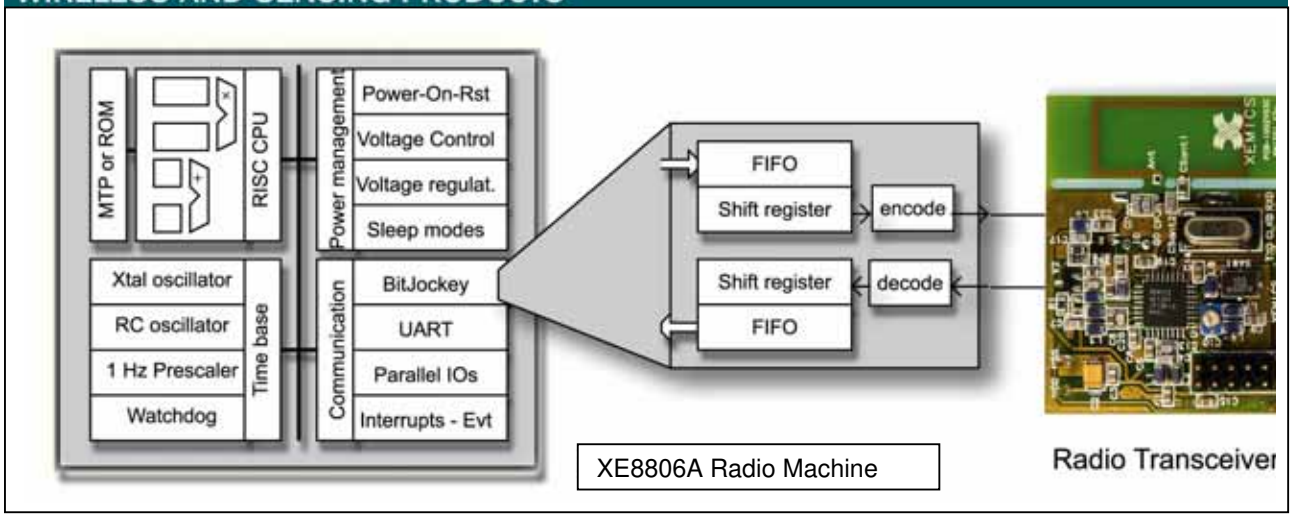
Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete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



XE8806A and XE8807A

Ultra Low-Power Low-Voltage Radio Machines

General Description

The XE8806A and XE8807A are ultra low-power low-voltage microcontroller based Radio Machines. They include the revolutionary BitJockey™, UART type of peripheral specialized for radio communication.

The XE8806A and XE8807A are available with on chip ROM or Multiple-Time-Programmable (MTP) program memory.

Key product Features

- Ultra low-power MCU, up to 7 MIPS
- 300 uA at 1 MIPS operation
- 6 uA at 32 kHz operation
- 1 uA time keeping
- Low-voltage operation (1.2 - 5.5 V supply voltage)
- 22 kB (8 kW) ROM/MTP (XE8806A)
  11 kB (4kW) MTP (XE8807A)
- 520 B RAM
- 4 counters
- PWM, UART, BitJockey™
- Analog matrix switching
- 4 low-power analog comparators
- independant RC and crystal oscillators
- 5 reset, 15 interrupt, 8 event sources
- 100 years MTP Flash retention at 55°C

Applications

- RF companion chip
- RF system supervisor
- Portable, battery operated instruments
- Metering
- Remote control
- HVAC control

Ordering Information

Product	Temperature range	Memory type	Package
XE8806AMI000	-40°C to 85 °C	MTP	die
XE8806AMI026LF	-40°C to 85 °C	MTP	TQFP32
XE8806ARI000	-40°C to 125°C	ROM	die
XE8806ARI026LF	-40°C to 125°C	ROM	TQFP32
XE8807AMI000	-40°C to 85 °C	MTP	die
XE8807AMI026LF	-40°C to 85 °C	MTP	TQFP32

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 (central processing unit) 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.

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 RAM. 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 event sources and it flags which event source is active.

The Port B is an 8 bit parallel IO port with analog capabilities. The USRT, UART, PWM and CMPD blocks 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 pin is not used. The maximal number of instructions in the XE8806A is 8192. The maximal number of instructions in the XE8807A is 4096.

The data memory on this product is a 512 byte SRAM.

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 D is a general purpose 8 bit parallel IO port.

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

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

The RFIF interface is a serial interface dedicated to communication with RF circuits. From the CPU side, it very much looks like an ordinary UART but it also implements low level coding/decoding and frame synchronisation. The input/output pins are multiplexed on port D.

The counters/timers/PWM can take its 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 whilst having a very low power consumption.

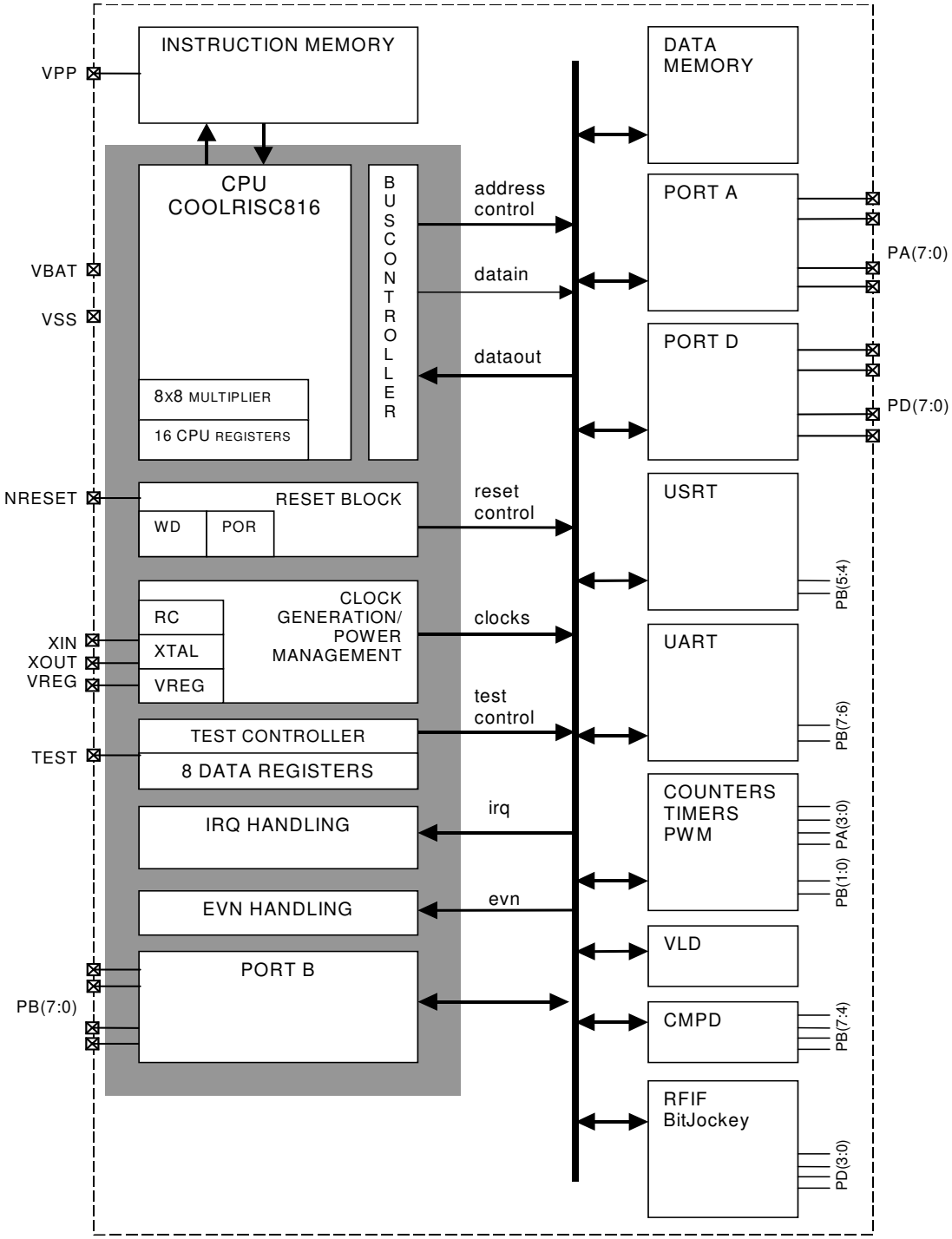


Figure 1-1. Block schematic of the XE8806A and XE8807A circuits.

1.2 Pin map

The XE8806A and XE8807A can be delivered in different packages. The pin maps for the different packages are given below.

1.2.1 TQFP-32

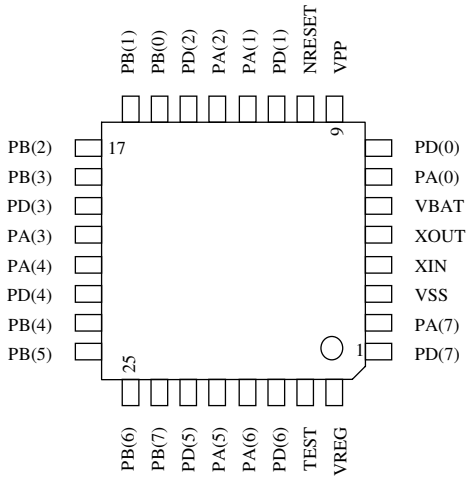


Figure 1-2. TQFP-32 pin map

1.2.2 SO-28

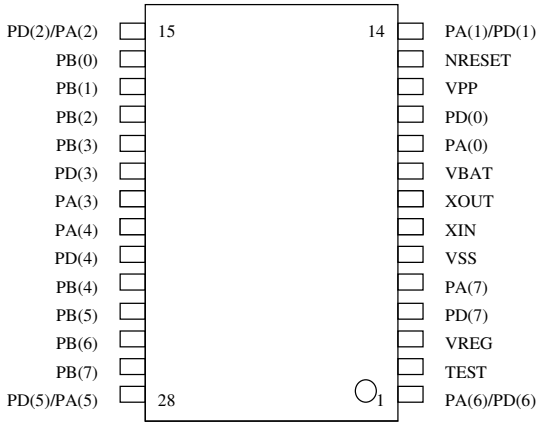


Figure 1-3. SO28 pin map

In the SO-28 package, 4 pins of Port A and Port D are connected together. It is up to the user to choose between the functionality of Port A or Port D for these pins.

Note: if one of the pins PD(1), PD(2), PD(5), PD(6) is used as output, the pull up of the corresponding pin of Port A should be disabled in order to have low power consumption.

1.2.3 SO-24

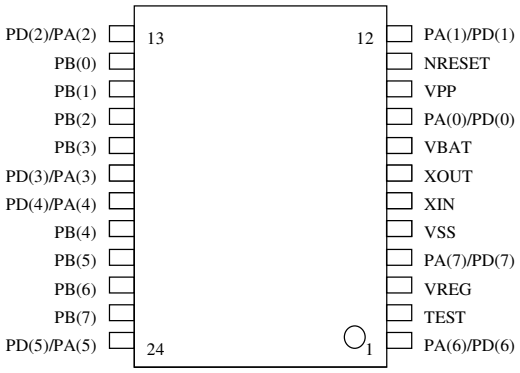


Figure 1-4. SO24 pin map

In the SO-24 package, all pins of Port A and Port D are connected together. It is up to the user to choose between the functionality of Port A or Port D.

Note: if one of the pins of Port D is used as output, the pull up of the corresponding pin of Port A should be disabled in order to have low power consumption.

1.2.4 Bare die XE8806A

The circuit is also available in bare die for chip on board assembly. All VBAT pins and all VSS pins should be connected together. The substrate of the circuit is connected to VSS.

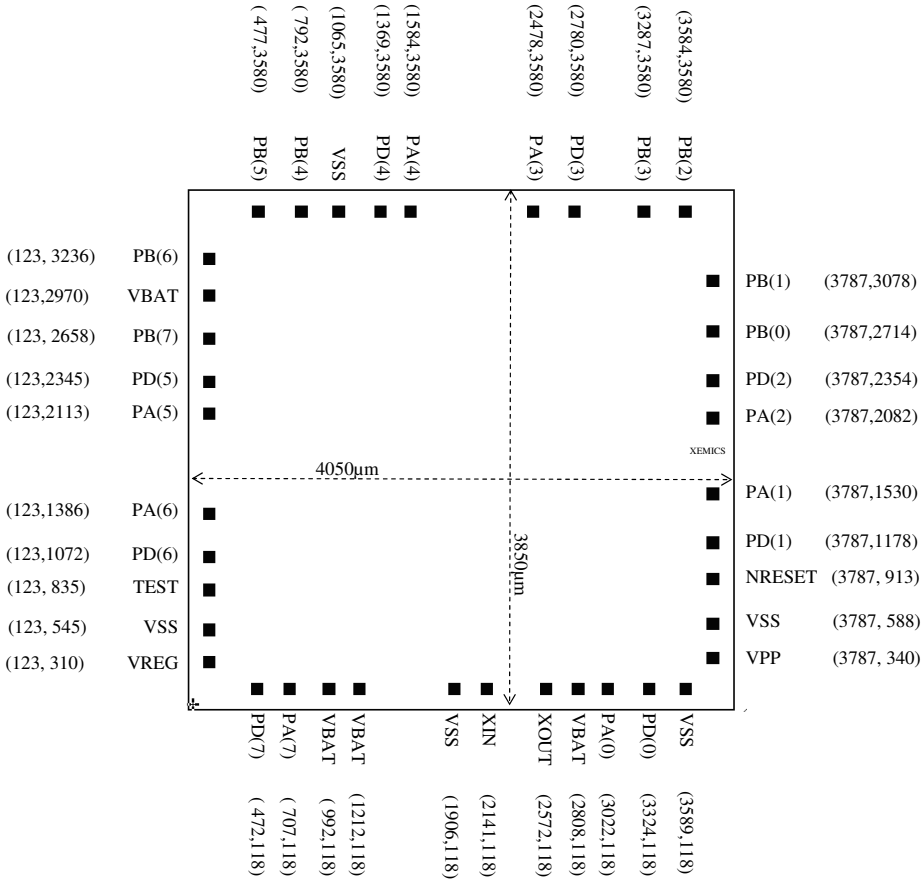


Figure 1-5. Die dimension and pin location of the XE8806A

1.2.5 Bare die XE8807A

The circuit is also available in bare die for chip on board assembly. All VBAT pins and all VSS pins should be connected together. The substrate of the circuit is connected to VSS.

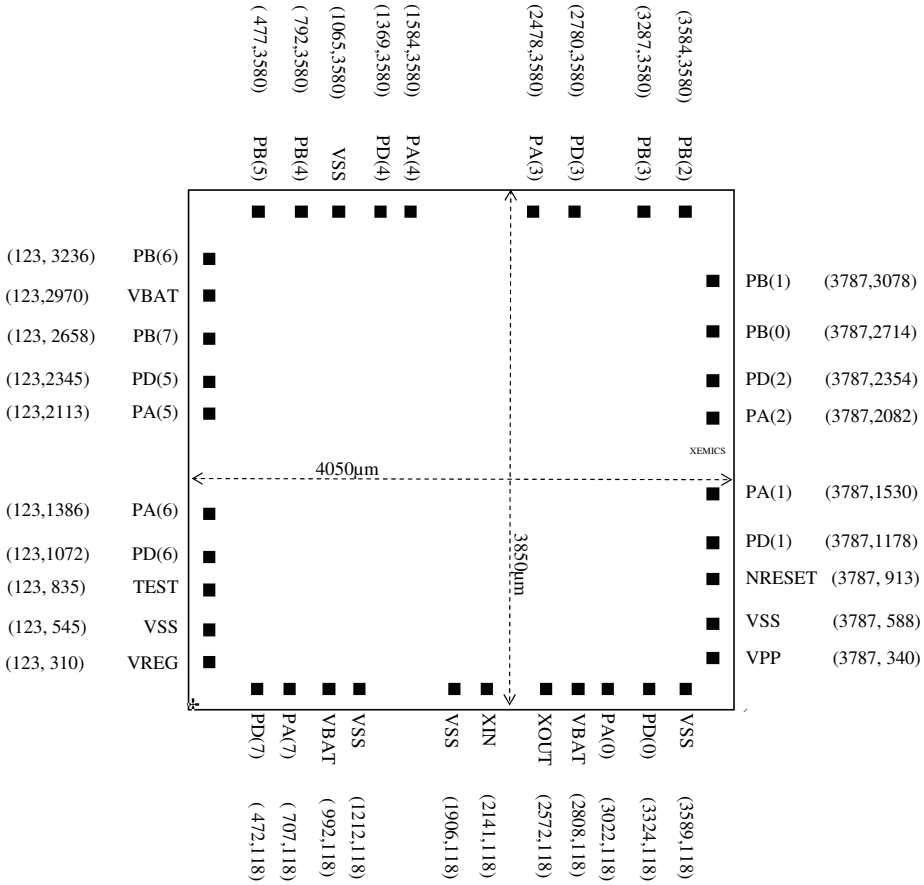


Figure 1-6. Die dimension and pin location of the XE8807A

1.3 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)
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
PD(7:0)	Parallel I/O port D pins

Table 1-1. Pin assignment

Table 1-2 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 SO-28 and SO-24 package several functions are routed to the same package pins. These pins are indicated in red italic. The pins RFIF(3:0) are the I/O pins of the RF interface, the CNTx pins are possible counter inputs, PWMx are possible PWM outputs, the CMPD pins are comparator inputs.

pin number			function			I/O configuration								
tqfp-32	SO-28	SO-24	first	second	third	AI	AO	DI	DO	OD	PU	PD	SNAP	POWER
1	4	*4*	**PD(7)**					**X**	X		**X**		X	
2	5	*4*	**PA(7)**					**X**			**X**		X	
3	6	5	**VSS**											**X**
4	7	6	**XIN**			**X**								
5	8	7	**XOUT**				**X**							
6	9	8	**VBAT**											**X**
7	10	*9*	**PA(0)**	CNTA				**X**			**X**		X	
8	11	*9*	**PD(0)**	RFIF(0)				**X**	X		**X**		X	
9	12	10	**VPP**											**X**
10	13	11	**NRESET**					**X**			**X**			
11	*14*	*12*	**PD(1)**	RFIF(1)				**X**	X		**X**		X	
12	*14*	*12*	**PA(1)**	CNTB				**X**			**X**		X	
13	*15*	*13*	**PA(2)**	CNTC				**X**			**X**		X	
14	*15*	*13*	**PD(2)**	RFIF(2)				**X**	X		**X**		X	
15	16	14	**PB(0)**	PWM0		X	X	**X**	X	X	**X**			
16	17	15	**PB(1)**	PWM1		X	X	**X**	X	X	**X**			
17	18	16	**PB(2)**			X	X	**X**	X	X	**X**			
18	19	17	**PB(3)**			X	X	**X**	X	X	**X**			
19	20	*18*	**PD(3)**	RFIF(3)				**X**	X		**X**		X	
20	21	*18*	**PA(3)**	CNTD				**X**			**X**		X	
21	22	*19*	**PA(4)**					**X**			**X**		X	
22	23	*19*	**PD(4)**					**X**	X		**X**		X	
23	24	20	**PB(4)**	USRT_S0	CMPD(0)	X	X	**X**	X	X	**X**			
24	25	21	**PB(5)**	USRT_S1	CMPD(1)	X	X	**X**	X	X	**X**			
25	26	22	**PB(6)**	UART_Tx	CMPD(2)	X	X	**X**	X	X	**X**			
26	27	23	**PB(7)**	UART_Rx	CMPD(3)	X	X	**X**	X	X	**X**			
27	*28*	*24*	**PD(5)**					**X**	X		**X**		X	
28	*28*	*24*	**PA(5)**					**X**			**X**		X	
29	*1*	*1*	**PA(6)**					**X**			**X**		X	
30	*1*	*1*	**PD(6)**					**X**	X		**X**		X	
31	2	2	**TEST**					**X**				**X**		
32	3	3	**VREG**				**X**							

Table 1-2. Pin description table

Pin map table legend:

red italic: pin shared with another peripheral in a specific package
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

2 XE8806A and XE8807A Performance

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	

Table 2-1. Absolute maximal ratings

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

	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	0.8	1.2	μF	

Table 2-2. Operating range for the flash device

Note 1. During the programming of the device, the supply voltage should at least be equal to the supply voltage used during normal operation, and temperature between 10°C and 40°C.

		Min.	Max.		Note
Voltage applied to VBAT with respect to VSS	VREG by-passed	1.2	5.5	V	
	VREG on	1.5	3.6	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

Table 2-3. Operating range for the ROM device

Note 1. The capacitor may be omitted when VREG is connected to VBAT.

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

	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

Table 2-4. Operating range of the Flash memory

Note 1. Valid only if programmed using a programming tool that is qualified
Note 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 (USRT, UART, CNT, VLD, CMPD) are disabled. The parallel ports are configured in input with pull up. Their pins are not connected externally.

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	

Table 2-5. Typical current consumption of the XE8806AM version (8k instructions flash memory) and XE8807AM version (4k instructions flash memory)

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

Table 2-6 shows the typical current consumption for the ROM version with 8k instructions. Two possible modes are possible: a 2.4V-5.5V operating range using the internal regulator and a 1.2V-3.3V operating range short circuiting the voltage regulator (i.e. connect VREG to VBAT).

Operation mode	CPU	RC	Xtal	Consumption	comments	Note
High speed CPU	1 MIPS	1 MHz	Off	200	2.4V<>5.5V, 27°C	1,2
Max. Speed CPU	4 MIPS	4 MHz	Off	800	2.4V<>5.5V, 27°C	1,2
Low speed CPU	.1 MIPS	100 kHz	Off	21	2.4V <>5.5V, 27°C	1,2
Low power CPU	32 kIPS	Off	32 kHz	7	2.4V <>5.5V, 27°C	1,2
Low voltage CPU	32 kIPS	Off	32 kHz	1	1.2V, 27°C	1,3
Low power time keeping	HALT	Off	32 kHz	1.3	2.4V <>5.5V, 27°C	2

Table 2-6. Current consumption of the XE8806AR version (8k instructions ROM memory)

1. Software using MOVE instruction using internal CPU registers and peripheral registers.
2. Using the internal voltage regulator (see Figure 2-5).
3. With the internal regulator short circuited (i.e. by connecting VREG to VBAT, see Figure 2-7). In this case, the current consumption will increase with VBAT.

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 XE8806AM

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 Figure 2-2.

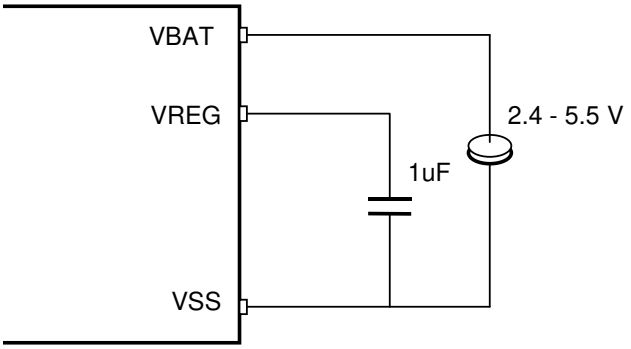


Figure 2-1. Supply configuration for flash circuit operation.

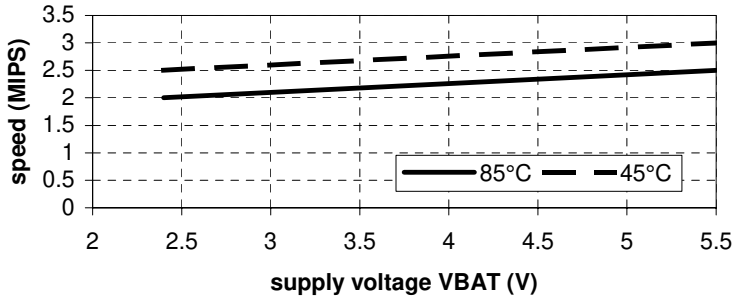


Figure 2-2. Guaranteed speed as a function of the supply voltage and maximal temperature.

Note that the speed of the flash circuit version is limited by the flash memory. All other peripherals of the device can run at the same speed as the ROM version (see Figure 2-6). The maximal speed of the peripherals can be exploited by reducing the CPU frequency by a factor of 2 with respect to the clock source by executing the instruction “FREQ div2”. Take care to execute this instruction before increasing the clock speed above the figures given in Figure 2-2.

2.4.2 Flash circuit version XE8807AM

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 Figure 2-4.

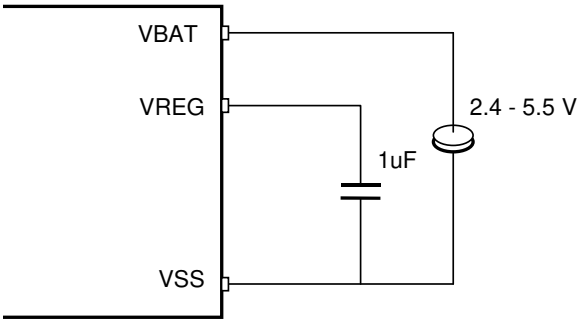


Figure 2-3. Supply configuration for flash circuit operation.

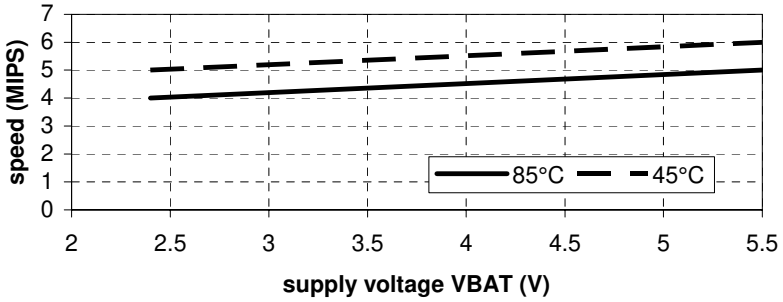


Figure 2-4. Guaranteed speed as a function of the supply voltage and maximal temperature.

2.4.3 ROM circuit version, regulator on

For the ROM version, two possible operating modes exist: with and without voltage regulator. Using the voltage regulator, a 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.

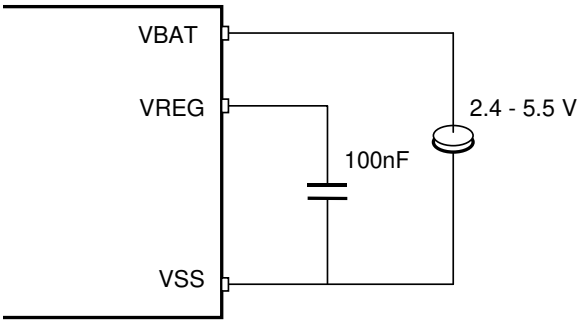


Figure 2-5. Supply configuration for ROM circuit operation using the internal regulator.

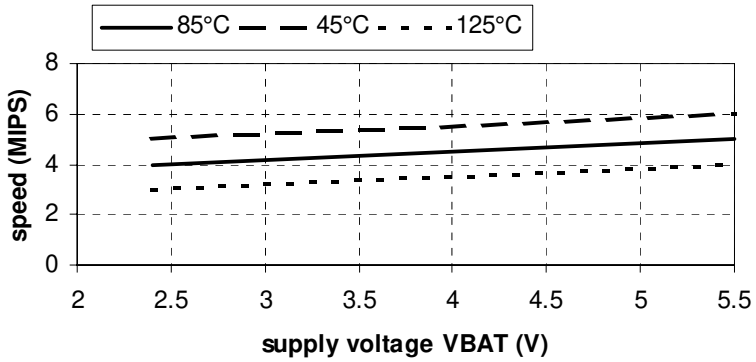


Figure 2-6. Guaranteed speed as a function of supply voltage and for different maximal temperatures using the voltage regulator.

2.4.4 ROM circuit version, regulator by-passed

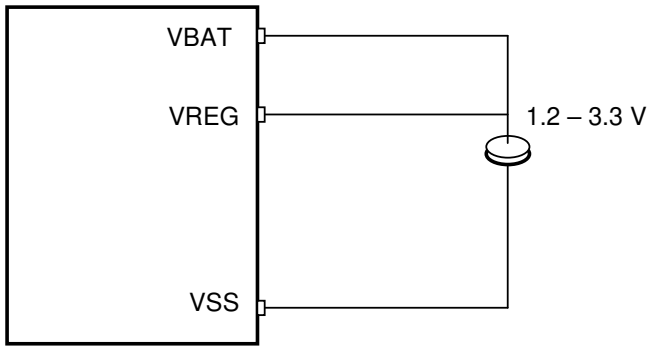


Figure 2-7. Supply configuration for ROM circuit operation by-passing the internal regulator.

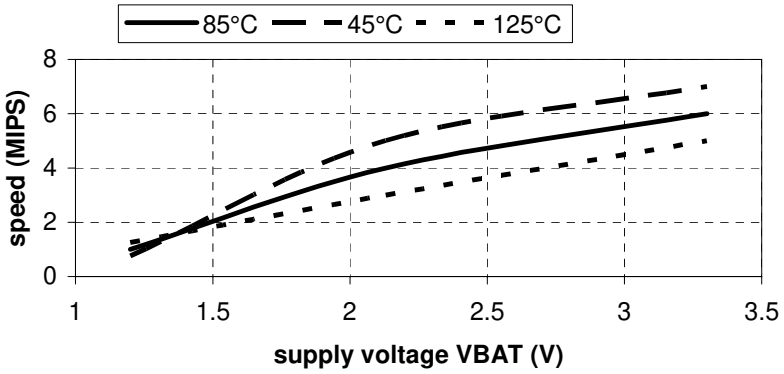


Figure 2-8. Guaranteed speed as a function of supply voltage and for two temperature ranges when VREG=VBAT.

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.

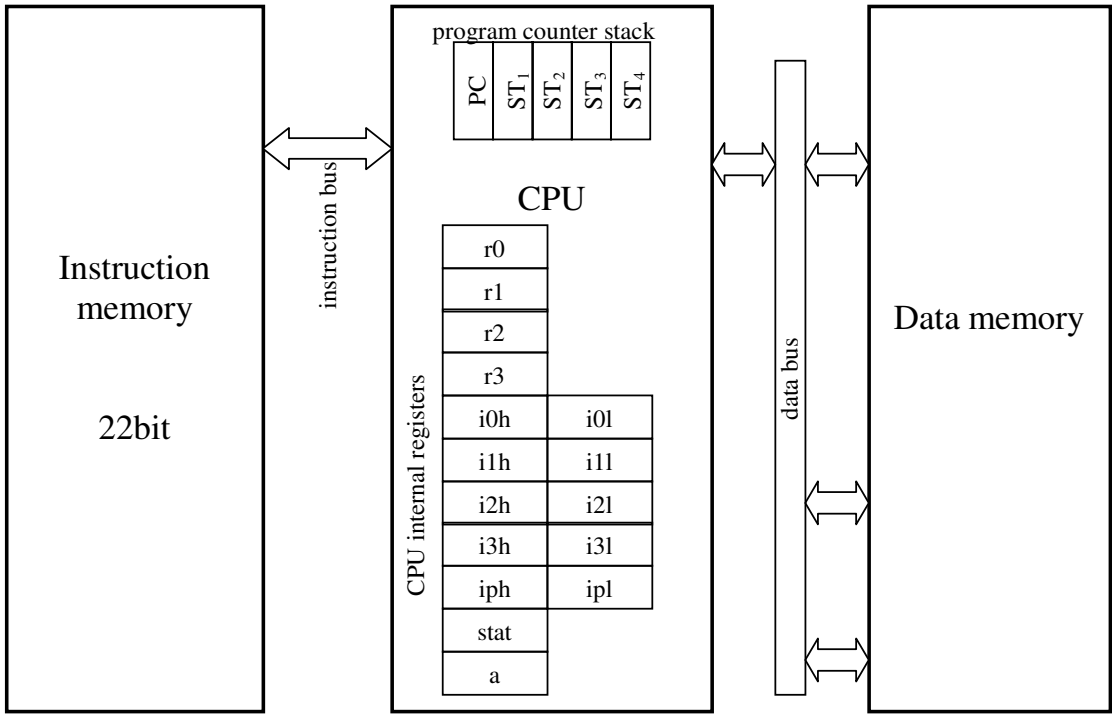


Figure 3-1. CPU internal registers

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 XE8806A and XE8807A CPU are built with 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 fetch 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 in the XE8806A and 4096 22-bit instructions in the XE8807A. The data memory space has 8 low power registers, the peripheral register space and 512 bytes of RAM.

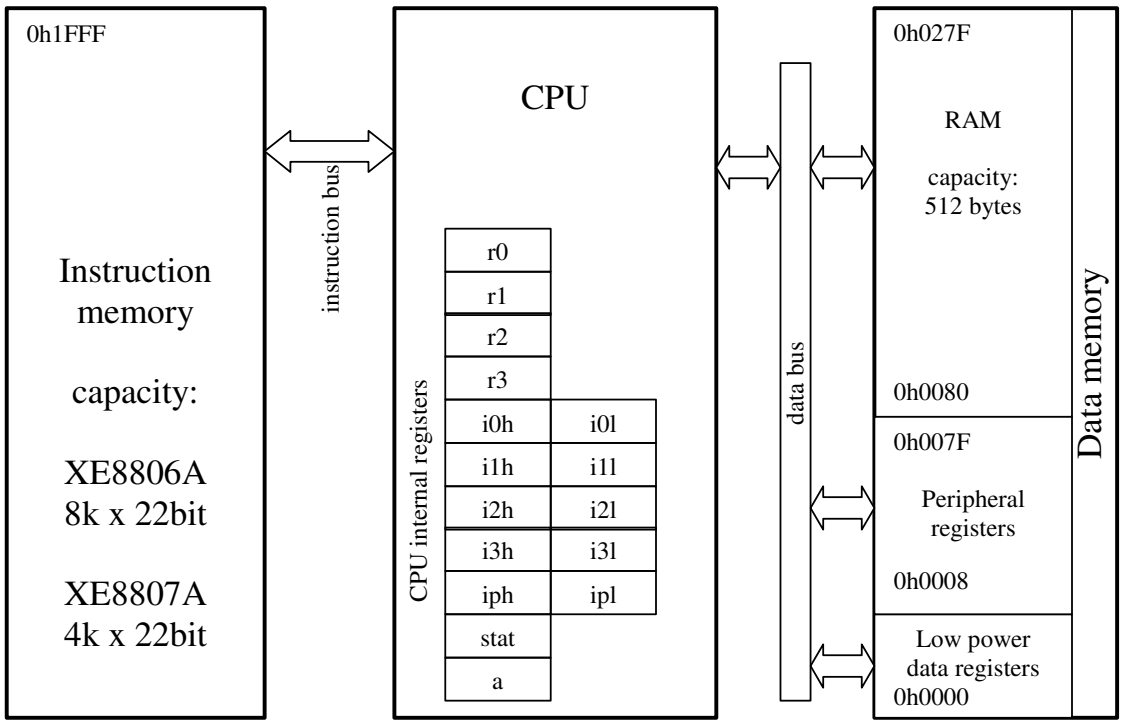


Figure 4-1. Memory mapping

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.

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

code	reset source
glob	nresetglobal
cold	nresetcold
pconf	nresetpconf
sleep	nresetsleep

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

4.2.1 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[7:0] rw,0000000,glob							

Table 4-3. Low power data registers

4.2.2 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	ClearLowPresca l 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. Reset block and clock block registers