Registers(draft)

General

COFFEE has two different register sets. The first set (SET 1) is intended to be used by application programs. The second set of registers (SET 2) is for privileged software which could be an operating system or similar. SET 2 is protected from application program. Privileged software can access both sets. There's a total of 32 registers in both sets including general purpose registers (GPRs) and special purpose registers (SPRs).

In addition COFFEE has eight condition registers (CRs) which are used with conditional branches or when executing instructions conditionally. These are visible to application software as well as to privileged software.

Besides the register bank described here, COFFEE has another register bank, CCB(core control block), which is mapped to memory (accessed using ld and st –instructions). CCB is for controlling the processor operation and as such should be configured by boot code. CCB also contains few status registers. Note that, CCB can be extended with an external configuration block!

The usage of general purpose registers is not restricted by hardware in any way. In any case, good programming means fixing some registers for a certain purpose.

table 1, *Registers*

SET 1			**SET 2**		
R0	GPR	32 bits	PR0	GPR	32 bits
R1	GPR	32 bits	PR1	GPR	32 bits
...			...		
R28	GPR	32 bits	PR28	GPR	32 bits
R29	GPR	32 bits	PR29	PSR	8 bits
R30	GPR	32 bits	PR30	SPSR	32 bits
R31	GPR/LR	32 bits	PR31	GPR/LR	32 bits

SET 1 GPRs

SET 1 has 32 identical general purpose registers R0...R31 with one exception: R31 is used as a link register(LR) with some instructions. The programmer is free to use R31 for any other purpose as long as it's special behaviour is taken into account. All general purpose registers (and the link register) are 32 bits wide.

SET 2 GPRs

SET 2 has 30 identical general purpose registers PR0...PR28 and PR31 with one exception: PR31 is used as a link register by some instructions. The programmer is free to use PR31 for any other purpose as long as it’s special behaviour is taken into account. All general purpose registers (and the link register) are 32 bits wide.

SET 2 SPRs

There’s two special purpose registers in SET 2: PSR and SPSR. PSR is eight bits wide. When reading data from PSR the ‘non existent’ bits are read as zeros. Writing to a read only register(PSR) is ignored.

PSR(register index 29)

Processor Status Register is a read only register and contains the flags explained below. Bits 7 downto 5 are reserved for future extensions.

RESERVED	IE	IL	RSWR	RSRD	UM
7...5	4	3	2	1	0

IE = 1: Interrupts enabled, IE = 0: Interrupts disabled.
IL = 1: Instruction word length is 32 bits, IL = 0: Instruction word length is 16 bits.
RSWR bit selects which register set to use as target:
RSWR = 1: SET2, super users set; RSWR = 0: SET1, users set.
RSRD bit selects which register set to use as source:
RSRD = 1: SET2, super users set; RSRD = 0: SET1, users set.
UM indicates which user mode the processor is in:
UM = 0: super user mode, UM = 1 : user mode.
RESERVED: Read as zeros.

SPSR(register index 30)

SPRS is used to save PSR flags when changing user mode by executing scall – instruction. It can be also used to set mode flags for the user: IE and IL flags are copied from SPSR to PSR when retu –instruction is executed. Note that bits 31 downto 5 are writable but only bits 7 downto 0 are saved in case of scall.

CRs

There’s eight three bit wide condition registers C0...C7 (visible both to application software and privileged software). Condition registers are used with conditional branches or when executing instructions conditionally. Each register contains three flags: Z (Zero), N (Negative) and C (Carry). When executing compare instructions or some arithmetic instructions these three flags are calculated and saved to the selected CR (arithmetic instructions always save flags to C0). When conditionally branching or executing, flags from the selected CR are compared to match a certain condition given by the programmer. See chapters ‘conditional execution’ and ‘instruction specifications’ for more infomation.

CCB registers

Note, that ‘byte’ addresses (that is consecutive addresses) are used in table below. 256 consecutive addresses are reserved for core configuration block. Addresses beyond CCB_BASE + ffh can be configured to point to an external peripheral configuration block (PCB), if present.

Registers which are shorter than 32 bits:

- LSB of a GPR corresponds to LSB of the short register in CCB.
- Unused bits read as zeros.
- For code combatibility with future versions, you should write unused bits as you would if there were more bits (interrupt masking, for example).

Core control block (CCB)

Offset	**mnemonic**	**Width**	**description/usage**	**notes**
00h	CCB_BASE	32	Start address of this relocatable configuration block (address of the CCB_BASE itself)	Has to be aligned to 256B boundary! That is, bits 7 downto 0 must be zeros
01h	CCB_END	32	End address of configuration register space.	See note 2 below
02h	COP0_INT_VEC	32	Co-processor 0 interrupt service routine start address.	See interrupts
03h	COP1_INT_VEC	32	Co-processor 1 interrupt service routine start address.	See interrupts
04h	COP2_INT_VEC	32	Co-processor 2 interrupt service routine start address.	See interrupts
05h	COP3_INT_VEC	32	Co-processor 3 interrupt service routine start address.	See interrupts
06h	EXT_INT0_VEC	32	External interrupt 0 service routine base address.	See interrupts
07h	EXT_INT1_VEC	32	External interrupt 1 service routine base address.	See interrupts
08h	EXT_INT2_VEC	32	External interrupt 2 service routine base address.	See interrupts
09h	EXT_INT3_VEC	32	External interrupt 3 service routine base address.	See interrupts
0ah	EXT_INT4_VEC	32	External interrupt 4 service routine base address.	See interrupts
0bh	EXT_INT5_VEC	32	External interrupt 5 service routine base address.	See interrupts

0ch	EXT_INT6_VEC	32	External interrupt 6 service routine base address.	See interrupts
0dh	EXT_INT7_VEC	32	External interrupt 7 service routine base address.	See interrupts
0eh	INT_MODE_IL	12	Instruction decoding mode flags for interrupt routines (PSR:IL is set accordingly when entering routine).	Bit associations: See note 3 below. See interrupts and processor status register.
0fh	INT_MODE_UM	12	User mode flags for interrupt routines(PSR:UM, RSRD, RSRW are set accordingly when entering routine).	
10h	INT_MASK	12	Register for masking external and cop interrupts individually. A low bit (‘0’) means blocking an interrupt source, a high bit enables an interrupt.	
11h	INT_SERV	12	Interrupt service status bits.	Read only. See interrupts.
12h	INT_PEND	12	Pending interrupt requests.	
13h	EXT_INT_PRI	32	Interrupt priorites: Bits 31 downto 28 : INT 7 priority Bits 27 downto 24 : INT 6 priority ... Bits 7 downto 4 : INT 1 priority Bits 3 downto 0 : INT 0 priority	0 – highest priority 15 – lowest priority Priorities for external interrupts can only be set if external handler is not used.
14h	COP_INT_PRI	16	Bits 15 downto 12 : COP3 priority Bits 11 downto 8 : COP2 priority Bits 7 downto 4 : COP1 priority Bits 3 downto 0 : COP0 priority	
15h	EXCEPTION_CS	8	Exception cause code.	Read only. See exceptions.
16h	EXCEPTION_PC	32	Address of the instruction which caused the exception.	
17h	EXCEPTION_PSR	8	Copy of the processor status flags which were used when decoding the violating instruction.	
18h	DMEM_BOUND_LO	32	start of protected/allowed address space for data memory	See user modes: super user. See also register MEM_PCONF. Note that bounds are included in the address space.
19h	DMEM_BOUND_HI	32	end of protected/allowed address space for data memory	
1ah	IMEM_BOUND_LO	32	start of protected/allowed address space for instruction memory	
1bh	IMEM_BOUND_HI	32	end of protected/allowed address space for instruction memory	

1ch	MEM_PCONF	32	Defines whether the space between addresses set by XMEM_BOUND_LO and XMEM_BOUND_HI is protected from user or allowed for user. Bit 0 controls instruction memory protection, bit 1 data memory protection. Bits 31 downto 2 are reserved. Bit high => area is protected Bit low => area is allowed (and the rest is protected)	See note 4.
1dh	SYSTEM_ADDR	32	System code entry address. (used by scall)	See instruction specifications: scall
1eh	EXCEP_ADDR	32	Exception handler entry address.	See exceptions
1fh	WAIT_STATES	12	Number of wait cycles for coprocessor and memory accesses. Can be set between 0 and 15 bits 11 downto 8: coprocessor access wait cycles. bits 7 downto 4 : data memory and PCB access wait cycles. bits 3 downto 0: instruction memory access wait cycles.	See core interface description.
20h	CREG_I_INDX	20	**Specifying register index for coprocessor instruction word.** bits 19 downto 15: Coprocessor number 3 register index used by cop –instruction bits 14 downto 10: Coprocessor number 2 register index used by cop –instruction bits 9 downto 5: Coprocessor number 1 register index used by cop –instruction bits 4 downto 0: Coprocessor number 0 register index used by cop –instruction	
21h	TMR0_CNT	32	Current timer value of timer 0	See document about timers.
22h	TMR0_MAX_CNT	32	Maximum value of timer 0	
23h	TMR1_CNT	32	Current timer value of timer 1	
24h	TMR1_MAX_CNT	32	Maximum value of timer 1	

25h	TMR_CONF	32	Common configuration register for timers 0 and 1 bits 31 downto 16 : timer 1 configuration bits. bits 15 downto 0 : timer 0 configuration bits.	
26h	COP_IF_MODE	8	Coprocessor interface configuration.	To be implemented later
27...ffh	**RESERVED FOR FUTURE EXTENSIONS**			

[2] Address range ([CCB_BASE] + 100h) to [CCB_END] is used to access an external configuration block directly. This makes it possible to connect peripherals directly to data cache bus instead of system bus.

[3] Bit index and interrupt source associations:

bit	source	bit	source	bit	source
0	coprocessor 0 int (exception)	4	ext int 0	8	ext int 4
1	coprocessor 1 int (exception)	5	ext int 1	9	ext int 5
2	coprocessor 2 int (exception)	6	ext int 2	10	ext int 6
3	coprocessor 3 int (exception)	7	ext int 3	11	ext int 7

[4] Memory protection can be dynamically configured which is convenient in multitasking system. Most secure way is to set the limits always when switching task and to allow one task to access only address space reserved for it(data and instruction memory). If different tasks share global data(dangerous!) address spaces can overlap. In most cases communication between tasks should follow schemes offered by operating system. In simple systems only vital part of the the memory might be protected and the rest of the memory is ‘free’ to everyone. In both cases it is recommended that CCB is mapped to protected area!

Register usage of a privileged user

When processor starts executing instructions after boot (see interface document) following conditions are assumed: 32 bit instruction word length, super user mode, register set SET2 for reading and writing and all interrupts (also cop exceptions) disabled. Boot code has the responsibility to initialize the special purpose registers to guarantee proper handling of interrupts and coprocessor exceptions. User mode can be entered by issuing the command *retu* (see ‘instruction defininions’ for details). Before passing the control, registers SPSR and PR31 must be set appropriately. Executing *retu* causes PSR to be overwritten by SPSR(not all flags though) and PC(program counter) overwritten by PR31. That is, execution will start at address saved to PR31 and with status flags saved in SPSR.

When an application program issues the command *scall* (requesting some system/kernel service, for example), SPSR is overwritten with PSR and PR31 is overwritten with link address (an address to return when resuming application code). In practise this means that super user is able to see the state in which the user was before calling system code and is able to resume execution from the correct address. Also the super user has full control over the user and the possibility to read and alter the status bits of the user.
An application program can pass parameters to privileged software (and the other way around) in some general purpose registers RXX , if desired , since privileged software can read and write both sets of registers with the help of *chrs* command. For more information about instructions *scall*, *retu* and *chrs* see ‘instruction defininions’.

Register limitations in 16 bit mode

In 16 bit mode only the last eight registers from both sets are available, that is registers R24...R31 from set 1 and PR24...PR31 from set 2. Registers are mapped so that referring to register R0/PR0 in 16 bit mode means referring to register R24/PR24 in 32 bit mode and in general referring to Rx/PRx in 16 bit mode means referring to R(x+24)/PR(x+24) in 32 bit mode where x is an integer in the range 0...7. Of course, assembler should provide straight forward notion to access registers.
Condition registers C1...C7 are disabled in 16 bit mode. Register C0 is always used (automatically selected) with conditional branches and arithmetic.

Register values after reset

PSR start value is 0000 1110b. SPSR is set to 0000 0009h Other registers in RF and CR are set to zero upon reset.

RESERVED	IE	IL	RSWR	RSRD	UM
7...5	4	3	2	1	0

CCB (internal) register values after reset

mnemonic	value after reset	Notes
CCB_BASE	0001 0000h	64KB offset from the 'start'. Depending on the actual memory implementation, data and instruction cache may or may not point to the same physical memory.
CCB_END	0001 00ffh	Must be set if an external configuration block is present.
COP0_INT_VEC	0000 0000h	
COP1_INT_VEC	0000 0000h	
COP2_INT_VEC	0000 0000h	
COP3_INT_VEC	0000 0000h	
EXT_INT0_VEC	0000 0000h	
EXT_INT1_VEC	0000 0000h	
EXT_INT2_VEC	0000 0000h	
EXT_INT3_VEC	0000 0000h	
EXT_INT4_VEC	0000 0000h	
EXT_INT5_VEC	0000 0000h	
EXT_INT6_VEC	0000 0000h	
EXT_INT7_VEC	0000 0000h	
INT_MODE_IL	fffh	32 bit mode for all routines
INT_MODE_UM	000h	Super user mode for all routines
INT_MASK	fffh	All interrupts disabled
EXT_INT_PRI	0000 0000h	See 'Interrupts and exceptions' for default priorites.
COP_INT_PRI	0000h	
INT_SERV	000h	
INT_PEND	000h	
EXCEPTION_CS	00h	
EXCEPTINON_PC	0000 0000h	
EXCEPTION_PSR	00h	
DMEM_BOUND_LO	0000 0000h	All the address space reserved for super user. Cannot run in user mode before configuring these register appropriately.
DMEM_BOUND_HI	ffff ffffh	
IMEM_BOUND_LO	0000 0000h	
IMEM_BOUND_HI	ffff ffffh	
MEM_PCONF	0000 0003h	

SYSTEM_ADDR	00000000h	
EXCEP_ADDR	00000000h	
WAIT_STATES	fffh	Assuming the slowest memories possible. Sixteen clock cycles per memory and cop access. (1 basic cycle + 15 wait cycles)
CREG_INDX_I	0 0000h	cop –instruction accesses register index 0 of the coprocessor.
TMR0_CNT	00000000h	
TMR0_MAX_CNT	00000000h	
TMR1_CNT	00000000h	
TMR1_MAX_CNT	00000000h	
TMR_CONF	00000000h	
COP_IF_MODE	00000000h	