3 STM8 core description

3.1 Introduction

The CPU has a full 8-bit architecture, with 16-bit operations on index registers (for address computation). Six internal registers allow efficient 8-bit data manipulation. The CPU is able to execute 80 basic instructions. It features 20 addressing modes and can address 6 internal registers and 16 Mbytes of memory/peripheral registers.

3.2 CPU registers

The 6 CPU registers are shown in the programming model in *Figure 1*. Following an interrupt, the register context is saved. The context is saved by pushing registers onto the stack in the order shown in *Figure 2.* They are popped from the stack in the reverse order.

Accumulator (A)

The accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations as well as data manipulations.

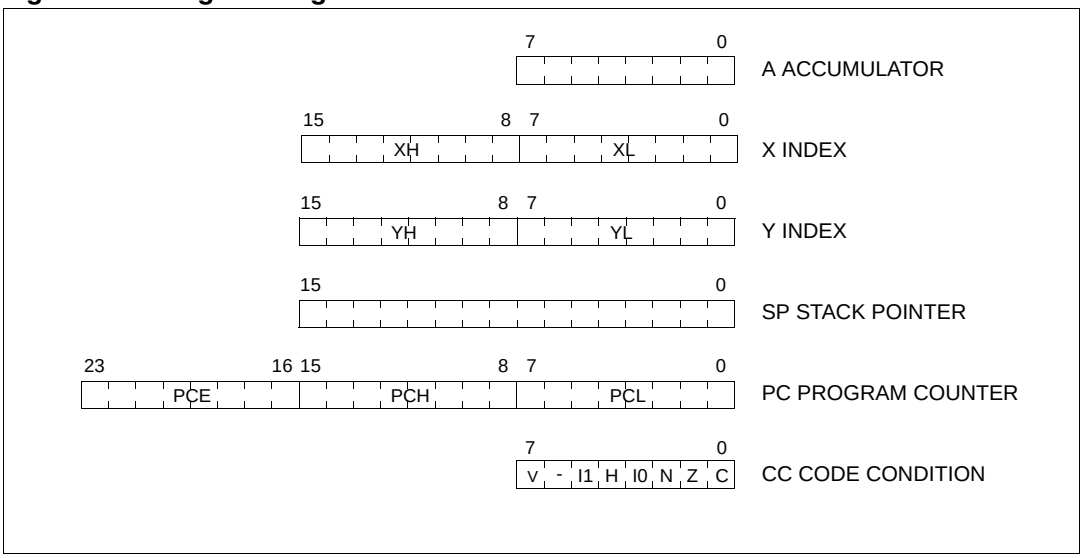
Index registers (X and Y)

These 16-bit registers are used to create effective addresses or as temporary storage area for data manipulations. In most of the cases, the cross assembler generates a PRECODE instruction (PRE) to indicate that the following instruction refers to the Y register. Both X and Y are automatically saved on interrupt routine branch.

Program Counter (PC)

The program counter is a 24-bit register used to store the address of the next instruction to be executed by the CPU. It is automatically refreshed after each processed instruction. As a result, the STM8 core can access up to 16-Mbytes of memory.

Figure 1. Programming model



Global configuration register (CFG_GCR)

The global configuration register is a memory mapped register. It controls the configuration of the processor. It contains the AL control bit:

AL: Activation level

If the AL bit is 0 (main), the IRET will cause the context to be retrieved from stack and the main program will continue after the WFI instruction.

If the AL bit is 1 (interrupt only active), the IRET will cause the CPU to go back to WFI/HALT mode without restoring the context.

This bit is used to control the low power modes of the MCU. In a very low power application, the MCU spends most of the time in WFI/HALT mode and is woken up (through interrupts) at specific moments in order to execute a specific task. Some of these recurring tasks are short enough to be treated directly in an ISR, rather than going back to the main program. In this case, by programming the AL bit to 1 before going to low power (by executing WFI/HALT instruction), the run time/ISR execution is reduced due to the fact that the register context is not saved/restored each time.

Condition Code register (CC)

The Condition Code register is a 8-bit register which indicates the result of the instruction just executed as well as the state of the processor. These bits can be individually tested by a program and specified action taken as a result of their state. The following paragraphs describe each bit.

- **V: Overflow**

  When set, V indicates that an overflow occurred during the last signed arithmetic operation, on the MSB operation result bit. See INC, INCW, DEC, DECW, NEG, NEGW, ADD, ADC, SUB, SUBW, SBC, CP, CPW instructions.

- **I1: Interrupt mask level 1**

  The I1 flag works in conjunction with the I0 flag to define the current interruptability level as shown in the following table. These flags can be set and cleared by software through the RIM, SIM, HALT, WFI, IRET, TRAP and POP instructions and are automatically set by hardware when entering an interrupt service routine.

Table 1. Interruptability levels

Interruptability	Priority	I1	I0
Interruptable Main	Lowest ↕ Highest	1	0
Interruptable Level 1		0	1
Interruptable Level 2		0	0
Non Interruptable		1	1

- **H: Half carry bit**

  The H bit is set to 1 when a carry occurs between the bits 3 and 4 of the ALU during an ADD or ADC instruction. The H bit is useful in BCD arithmetic subroutines.

  For ADDW, SUBW it is set when a carry occurs from bit 7 to 8, allowing to implement byte arithmetic on 16-bit index registers.

- **I0: Interrupt mask level 0**

  See Flag I1

- **N: Negative**

  When set to 1, this bit indicates that the result of the last arithmetic, logical or data manipulation is negative (i.e. the most significant bit is a logic 1).

- **Z: Zero**

  When set to 1, this bit indicates that the result of the last arithmetic, logical or data manipulation is zero.

- **C: Carry**

  When set, C indicates that a carry or borrow out of the ALU occurred during the last arithmetic operation on the MSB operation result bit (bit 7 for 8-bit result/destination or bit 15 for 16-bit result). This bit is also affected during bit test, branch, shift, rotate and load instructions. See ADD, ADC, SUB, SBC instructions.
  In bit test operations, C is the copy of the tested bit. See BTJF, BTJT instructions.
  In shift and rotates operations, the carry is updated. See RRC, RLC, SRL, SLL, SRA instructions.

  This bit can be set, reset or complemented by software using SCF, RCF, CCF instructions.

  Example: Addition

  $B5 + $94 = "C" + $49 = $149

	C	7						0
	0	1	0	1	1	0	1	0	1
+	0	1	0	0	1	0	1	0	0
=	1	0	1	0	0	1	0	0	1

The results of each instruction on the Condition Code register are shown by tables in *Section 7: STM8 instruction set*. The following table is an example:

V	I1	H	I0	N	Z	C
V	0		0	N	Z	1

where

Nothing = Flag not affected

Flag name = Flag affected

0 = Flag cleared

1 = Flag set

ADD — Addition — ADD

Syntax ADD A,src e.g. ADD A,#%11001010

Operation A <= A+ src

Description The source byte is added to the contents of the accumulator and the result is stored in the accumulator. The source is a memory or data byte.

Instruction overview

mnem	dst	src	Affected condition flags						
			V	I1	H	I0	N	Z	C
ADD	A	Mem	V	-	H	-	N	Z	C

V ⇒ $(A7.M7 + M7.\overline{R7} + \overline{R7}.A7) \oplus (A6.M6 + M6.\overline{R6} + \overline{R6}.A6)$
Set if the signed operation generates an overflow, cleared otherwise.

H ⇒ $A3.M3 + M3.\overline{R3} + \overline{R3}.A3$
Set if a carry occurred from bit 3 of the result, cleared otherwise.

N ⇒ R7
Set if bit 7 of the result is set (negative value), cleared otherwise.

Z ⇒ $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$
Set if the result is zero (0x00), cleared otherwise.

C ⇒ $A7.M7 + M7.\overline{R7} + \overline{R7}.A7$
Set if a carry occurred from bit 7 of the result, cleared otherwise.

Detailed description

dst	src	Asm	cy	lgth	Op-code(s)					ST7
A	#byte	ADD A,#$55	1	2		AB	XX			✗
A	shortmem	ADD A,$10	1	2		BB	XX			✗
A	longmem	ADD A,$1000	1	3		CB	MS	LS		✗
A	(X)	ADD A,(X)	1	1		FB				✗
A	(shortoff,X)	ADD A,($10,X)	1	2		EB	XX			✗
A	(longoff,X)	ADD A,($1000,X)	1	3		DB	MS	LS		✗
A	(Y)	ADD A,(Y)	1	2	90	FB				✗
A	(shortoff,Y)	ADD A,($10,Y)	1	3	90	EB	XX			✗
A	(longoff,Y)	ADD A,($1000,Y)	1	4	90	DB	MS	LS		✗
A	(shortoff,SP)	ADD A,($10,SP)	1	2		1B	XX			
A	[shortptr.w]	ADD A,[$10.w]	4	3	92	CB	XX			✗
A	[longptr.w]	ADD A,[$1000.w]	4	4	72	CB	MS	LS		
A	([shortptr.w],X)	ADD A,([$10.w],X)	4	3	92	DB	XX			✗
A	([longptr.w],X)	ADD A,([$1000.w],X)	4	4	72	DB	MS	LS		
A	([shortptr.w],Y)	ADD A,([$10.w],Y)	4	3	91	DB	XX			✗

See also: ADDW, ADC, SUB, SBC, MUL, DIV

CLR Clear CLR

Syntax CLR dst e.g. CLR A

Operation dst <= 00

Description The destination byte is forced to 00 value. The destination is either a memory byte location or the accumulator. This instruction is compact, and does not affect any register when used with RAM variables.

Instruction overview

mnem	dst	Affected condition flags						
		V	I1	H	I0	N	Z	C
CLR	Mem	-	-	-	-	0	1	-
CLR	A					0	1	

N: 0

Cleared

Z: 1

Set

Detailed description

dst	Asm	cy	lgth	Op-code(s)					ST7
A	CLR A	1	1		4F				✗
shortmem	CLR $10	1	2		3F	XX			✗
longmem	CLR $1000	1	4	72	5F	MS	LS		
(X)	CLR (X)	1	1		7F				✗
(shortoff.X)	CLR ($10,X)	1	2		6F	XX			✗
(longoff,X)	CLR ($1000,X)	1	4	72	4F	MS	LS		
(Y)	CLR (Y)	1	2	90	7F				✗
(shortoff,Y)	CLR ($10,Y)	1	3	90	6F	XX			✗
(longoff,Y)	CLR ($1000,Y)	1	4	90	4F	MS	LS		
(shortoff,SP)	CLR ($10,SP)	1	2		0F	XX			
[shortptr.w]	CLR [$10]	4	3	92	3F	XX			✗
[longptr.w]	CLR [$1000].w	4	4	72	3F	MS	LS		
([shortptr.w],X)	CLR ([$10],X)	4	3	92	6F	XX			✗
([longptr.w].X]	CLR ([$1000.w],X)	4	4	72	6F	MS	LS		
([shortptr.w],Y)	CLR ([$10],Y)	4	3	91	6F	XX			✗

See also: LD

JPF Jump far JPF

Syntax JPF dst e.g.:JPF test

Operation PC <= dst

Description The unconditional jump simply replaces the content of the PC by a destination with an extended address. Control then passes to the statement addressed by the program counter. For safe memory usage, this instruction must be used, when the operation crosses a memory section.

Instruction overview

mnem	dst	Affected condition flags						
		V	I1	H	I0	N	Z	C
JPF	Mem	-	-	-	-	-	-	-

Detailed description

dst	Asm	cy	lgth	Op-code(s)					ST7
extmem	JPF $2FFFFC	2	4		AC	ExtB	MS	LS	
[longptr.e]	JPF [$2FFC.e]	6	4	92	AC	MS	LS		

See also: JP, CALLF

JRA — Jump Relative Always — JRA

Syntax JRA dst e.g. JRA loop

Operation PC = PC+lgth

PC <= PC + dst, if Condition is True

Description Unconditional relative jump. PC is updated by the signed addition of PC and dst. Control then passes to the statement addressed by the program counter. Else, the program continues normally.

Instruction overview

mnem	dst	Affected condition flags						
		V	I1	H	I0	N	Z	C
JRA	Mem	-	-	-	-	-	-	-

Detailed description

dst	Asm	cy	lgth	Op-code(s)					ST7
shortoff	JRA $2B	2	2		20	XX			✗

See also: JP

JRxx — Conditional Jump Relative Instruction — JRxx

Syntax JRxx dst e.g. JRxx loop

Operation PC = PC+lgth

PC <= PC + dst, if Condition is True

Description Conditional relative jump. PC is updated by the signed addition of PC and dst, if the condition is true. Control, then passes to the statement addressed by the program counter. Else, the program continues normally.

Instruction overview

mnem	dst	Affected condition flags						
		V	I1	H	I0	N	Z	C
JRxx	Mem	-	-	-	-	-	-	-

Instruction List

mnem	meaning	sym	Condition	Op-code (OC)	
JRC	Carry		C = 1		25
JREQ	Equal	=	Z = 1		27
JRF	False		False		21
JRH	Half-Carry		H = 1	90	29
JRIH	Interrupt Line is High			90	2F
JRIL	Interrupt Line is Low			90	2E
JRM	Interrupt Mask		I = 1	90	2D
JRMI	Minus	< 0	N = 1		2B
JRNC	Not Carry		C = 0		24
JRNE	Not Equal	<> 0	Z = 0		26
JRNH	Not Half-Carry		H = 0	90	28
JRNM	Not Interrupt Mask		I = 0	90	2C
JRNV	Not Overflow		V = 0		28
JRPL	Plus	>= 0	N = 0		2A
JRSGE	Signed Greater or Equal	>=	(N XOR V) = 0		2E
JRSGT	Signed Greater Than	>	(Z OR (N XOR V)) = 0		2C
JRSLE	Signed Lower or Equal	<=	(Z OR (N XOR V)) = 1		2D
JRSLT	Signed Lower Than	<	(N XOR V) = 1		2F
JRT	True		True		20
JRUGE	Unsigned Greater or Equal		C = 0		24
JRUGT	Unsigned Greater Than	>	C = 0 and Z = 0		22
JRULE	Unsigned Lower or Equal	<=	C = 1 or Z = 1		23
JRC	Carry		C = 1		25
JRULT	Unsigned Lower Than		C = 1		25
JRV	Overflow		V = 1		29

Detailed description

dst	Asm	cy	lgth	Op-code(s)					ST7
shortoff	JRxx $15	1/2	2		Op-code	XX			✗
shortoff	JRxx $15	1/2	3	90	Op-code	XX			✗

LD Load LD

Syntax LD dst,src e.g. LD A,#$15

Operation dst <= src

Description Load the destination byte with the source byte. The dst and src can be a register, a byte (low/high) of an index register or a memory/data byte. When half of an index register is loaded, the other half remains unchanged.

Instruction overview

mnem	dst	src	Affected condition flags						
			V	I1	H	I0	N	Z	C
LD	Reg	Mem	-	-	-	-	N	Z	-
LD	Mem	Reg	-	-	-	-	N	Z	-
LD	Reg	Reg	-	-	-	-	-	-	-

N ⇒ R7
Set if bit 7 of the result is set (negative value), cleared otherwise.

Z ⇒ $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$
Set if the result is zero (0x00), cleared otherwise.

Detailed description

dst	src	Asm	cy	lgth	Op-code(s)					ST7
A	#byte	LD A,#$55	1	2		A6	XX			✗
A	shortmem	LD A,$50	1	2		B6	XX			✗
A	longmem	LD A,$5000	1	3		C6	MS	LS		✗
A	(X)	LD A,(X)	1	1		F6				✗
A	(shortoff,X)	LD A,($50,X)	1	2		E6	XX			✗
A	(longoff,X)	LD A,($5000,X)	1	3		D6	MS	LS		✗
A	(Y)	LD A,(Y)	1	2	90	F6				✗
A	(shortoff,Y)	LD A,($50,Y)	1	3	90	E6	XX			✗
A	(longoff,Y)	LD A,($5000,Y)	1	4	90	D6	MS	LS		✗
A	(shortoff,SP)	LD A,($50,SP)	1	2		7B	XX			
A	[shortptr.w]	LD A,[$50.w]	4	3	92	C6	XX			✗
A	[longptr.w]	LD A,[$5000.w]	4	4	72	C6	MS	LS		
A	([shortptr.w],X)	LD A,([$50.w],X)	4	3	92	D6	XX			✗
A	([longptr.w],X)	LD A,([$5000.w],X)	4	4	72	D6	MS	LS		
A	([shortptr.w],Y)	LD A,([$50.w],Y)	4	3	91	D6	XX			✗

LD detailed description (Continued)

dst	src	Asm	cy	lgth	Op-code(s)					ST7
shortmem	A	LD $50,A	1	2		B7	XX			✗
longmem	A	LD $5000,A	1	3		C7	MS	LS		✗
(X)	A	LD (X),A	1	1		F7				✗
(shortoff,X)	A	LD ($50,X),A	1	2		E7	XX			✗
(longoff,X)	A	LD ($5000,X),A	1	3		D7	MS	LS		✗
(Y)	A	LD (Y),A	1	2	90	F7				✗
(shortoff,Y)	A	LD ($50,Y),A	1	3	90	E7	XX			✗
(longoff,Y)	A	LD ($5000,Y),A	1	4	90	D7	MS	LS		✗
(shortoff,SP)	A	LD ($50,SP),A	1	2		6B	XX			
[shortptr.w]	A	LD [$50.w],A	4	3	92	C7	XX			✗
[longptr.w]	A	LD [$5000.w],A	4	4	72	C7	MS	LS		
([shortptr.w], X)	A	LD ([$50.w],X),A	4	3	92	D7	XX			✗
([longptr.w],X)	A	LD ([$5000.w],X),A	4	4	72	D7	MS	LS		
([shortptr.w], Y)	A	LD ([$50.w],Y),A	4	3	91	D7	XX			✗

dst	src	Asm	cy	lgth	Op-code(s)					ST7
XL	A	LD XL,A	1	1		97				✗
A	XL	LD A,XL	1	1		9F				✗
YL	A	LD YL,A	1	2	90	97				✗
A	YL	LD A,YL	1	2	90	9F				✗
XH	A	LD XH,A	1	1		95				
A	XH	LD A,XH	1	1		9E				
YH	A	LD YH,A	1	2	90	95				
A	YH	LD A,YH	1	2	90	9E				

See also: LDW, LDF, CLR

LDW Load word LDW

Syntax LDW dst,src e.g. LDW X,#$1500

Operation dst <= src

Description Load the destination word (16-bit value) with the source word. The dst and src can be a 16-bit register (X, Y or SP) or a memory/data 16-bit value.

Instruction overview

mnem	dst	src	Affected condition flags						
			V	I1	H	I0	N	Z	C
LD	Reg	Mem	-	-	-	-	N	Z	-
LD	Mem	Reg	-	-	-	-	N	Z	-
LD	Reg	Reg	-	-	-	-	-	-	-
LD	SP	Reg	-	-	-	-	-	-	-
LD	Reg	SP	-	-	-	-	-	-	-

N ⇒ R15
Set if bit 7 of the result is set (negative value), cleared otherwise.

Z ⇒ $\overline{R15}.\overline{R14}.\overline{R13}.\overline{R12}.\overline{R11}.\overline{R10}.\overline{R9}.\overline{R8}.\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$
Set if the result is zero (0x0000), cleared otherwise.

Detailed description

dst	src	Asm	cy	lgth	Op-code(s)					ST7
X	#word	LDW X,#$55AA	2	3		AE	MS	LS		✗
X	shortmem	LDW X,$50	2	2		BE	XX			✗
X	longmem	LDW X,$5000	2	3		CE	MS	LS		✗
X	(X)	LDW X,(X)	2	1		FE				✗
X	(shortoff,X)	LDW X,($50,X)	2	2		EE	XX			✗
X	(longoff,X)	LDW X,($5000,X)	2	3		DE	MS	LS		✗
X	(shortoff,SP)	LDW X,($50,SP)	2	2		1E	XX			
X	[shortptr.w]	LDW X,[$50.w]	5	3	92	CE	XX			✗
X	[longptr.w]	LDW X,[$5000.w]	5	4	72	CE	MS	LS		
X	([shortptr.w],X)	LDW X,([$50.w],X)	5	3	92	DE	XX			✗
X	([longptr.w],X)	LDW X,([$5000.w],X)	5	4	72	DE	MS	LS		

dst	src	Asm	cy	lgth	Op-code(s)					ST7
shortmem	X	LDW $50,X	2	2		BF	XX			✗
longmem	X	LDW $5000,X	2	3		CF	MS	LS		✗
(X)	Y	LDW (X),Y	2	1		FF				
(shortoff,X)	Y	LDW ($50,X),Y	2	2		EF	XX			
(longoff,X)	Y	LDW ($5000,X),Y	2	3		DF	MS	LS		

SUB Subtraction SUB

Syntax SUB A,src e.g. SUB A,#%11001010

Operation A <= A- src

Description The source byte is subtracted from the contents of the accumulator/SP and the result is stored in the accumulator/SP. The source is a memory or data byte.

Instruction overview

mnem	dst	src	Affected condition flags						
			V	I1	H	I0	N	Z	C
SUB	A	Mem	V	-	-	-	N	Z	C
SUB	SP	Imm	-	-	-	-	-	-	-

V ⇒ (A7.M7 + A7.R7 + A7.M7.R7) ⊕ (A6.M6 + A6.R6 + A6.M6.R6)
Set if the signed operation generates an overflow, cleared otherwise.

N ⇒ R7
Set if bit 7 of the result is set (negative value), cleared otherwise.

Z ⇒ R7.R6.R5.R4.R3.R2.R1.R0
Set if the result is zero (0x00), cleared otherwise.

C ⇒ $\overline{A7}$.M7 + $\overline{A7}$.R7 + A7.M7.R7
Set if a borrow request occurred from bit 7, cleared otherwise.

Detailed description

dst	src	Asm	cy	lgth	Op-code(s)					ST7
A	#byte	SUB A,#$55	1	2		A0	XX			✗
A	shortmem	SUB A,$10	1	2		B0	XX			✗
A	longmem	SUB A,$1000	1	3		C0	MS	LS		✗
A	(X)	SUB A,(X)	1	1		F0				✗
A	(shortoff,X)	SUB A,($10,X)	1	2		E0	XX			✗
A	(longoff,X)	SUB A,($1000,X)	1	3		D0	MS	LS		✗
A	(Y)	SUB A,(Y)	1	2	90	F0				✗
A	(shortoff,Y)	SUB A,($10,Y)	1	3	90	E0	XX			✗
A	(longoff,Y)	SUB A,($1000,Y)	1	4	90	D0	MS	LS		✗
A	(shortoff,SP)	SUB A,($10,SP)	1	2		10	XX			
A	[shortptr.w]	SUB A,[$10.w]	4	3	92	C0	XX			✗
A	[longptr.w]	SUB A,[$1000.w]	4	4	72	C0	MS	LS		
A	([shortptr.w],X)	SUB A,([$10.w],X)	4	3	92	D0	XX			✗
A	([longptr.w],X)	SUB A,([$1000.w],X)	4	4	72	D0	MS	LS		
A	([shortptr.w],Y)	SUB A,([$10.w],Y)	4	3	91	D0	XX			✗
SP	#byte	SUB SP,#$9	1	2		52	XX			

See also: SUBW, ADD, ADC, SBC, MUL