ST9 INSTRUCTION SET

REFERENCE GUIDE

1 SOFTWARE DESCRIPTION

1 SOFTWARE DESCRIPTION

1.1ADDRESSING MODES

The ST9 offers a wide variety of established and new addressing modes and combinations to facilitate full and rapid access to the address spaces while reducing program length. The available addressing modes are shown in Table 1-1:

Single operand arithmetic, logic and shift byte instructions have direct register and indirect register addressing modes. For a full list of the possible combinations for each instruction type, please refer to the ST9 Programming Manual.

Table 1-1. Addressing Modes

Operand is In	Addressing Mode	Destination Location	Notation
Instruction	Immediate	Byte Word	#N #NN
Register File	Direct	Byte Word	r rr
	Indirect Byte/Word	(r)	
	Indexed Byte/Word	N(r)	
	Indirect Post-Increment	Byte	(r)+
Program or Data Memory	Direct Byte/Word	NN	
	Indirect Byte/Word	(rr)	
	Indirect Post-Increment	Byte/Word	(rr)+
	Indirect Pre-Decrement	Byte/Word	-(rr)
	Short Indexed	Byte/Word N(rr)	
	Long Indexed	Byte/Word NN(rr)	
	Register Indexed	Byte/Word rr(rr)	
Any bit of any working register	Direct Bit	r.b	
Any bit in program or data memory	Indirect Bit	(rr).b	

ADDRESSING MODES (Continued)

Two Operands Arithmetic and Logic Instructions	
Destination	**Source**
Register Direct	Register Direct
Register Direct	Register Indirect
Register Direct	Memory Indirect
Register Direct	Memory Indexed
Register Direct	Memory Indirect with Post-Increment
Register Direct	Memory Indirect with Pre-Decrement
Register Direct	Memory Direct
Register Indirect	Register Direct
Memory Indirect	Register Direct
Memory Indexed	Register Direct
Memory Indirect with Post-Increment	Register Direct
Memory Indirect with Pre-Decrement	Register Direct
Memory Direct	Register Direct
Register Direct	Immediate
Memory Direct	Immediate
Memory Indirect	Immediate

Two Operands Arithmetic, Logic and Load Instructions	
Destination	**Source**
Memory Indirect	Memory Direct

ADDRESSING MODES (Continued)

Two Operands Load Instructions	
Destination	**Source**
Register Direct	Register Direct
Register Direct	Register Indirect
Register Direct	Register Indexed
Register Direct	Memory Indirect
Register Direct	Memory Indexed
Register Direct	Memory Indirect with Post-Increment
Register Direct	Memory Indirect with Pre-Decrement
Register Direct	Memory Direct
Register Indirect	Register Direct
Register Indexed	Register Direct
Memory Indirect	Register Direct
Memory Indexed	Register Direct
Memory Indirect with Post-Increment	Register Direct
Memory Indirect with Pre-Decrement	Register Direct
Memory Direct	Register Direct
Register Direct	Immediate
Memory Direct	Immediate
Memory Indirect	Immediate
Long Indexed Memory (1)	Immediate

Two Operands Load Instructions (2)	
Destination	**Source**
Register Indirect with Post-Increment	Memory Indirect with Post-Increment
Memory Indirect with Post-Increment	Register Indirect with Post-Increment
Memory Indirect with Post-Increment	Memory Indirect with Post-Increment

Notes:
1. Word Instructions Only
2. Load Byte Only

ADDRESSING MODES (Continued)

1.1.1 Register Addressing Modes

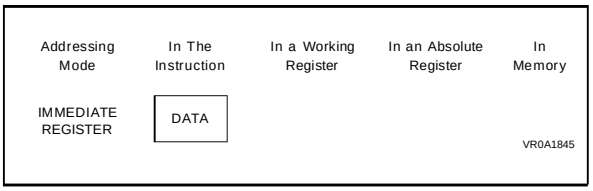
Immediate Addressing Mode

In the Immediate addressing mode, the data is found in the instruction. When using immediate data, a hash-mark (#) is used to distinguish it from an absolute address in memory.

Example: `ldw RR42,#65536`
loads the immediate value 65536 into the register pair R42 & R43. While the example shows decimal data, hexadecimal and binary values may also be used.

Example: `ldw RR42,#0FFFFh`.

Figure 1-1. Immediate Register

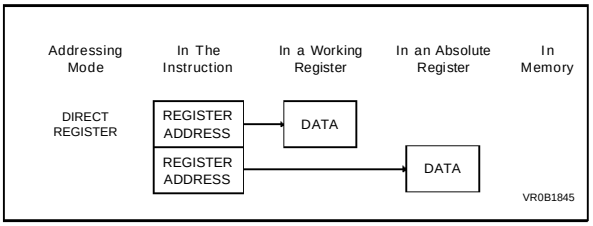


Direct Addressing Mode

In the direct addressing mode, a register can be addressed by using its absolute address in the Register File (in decimal, hexadecimal or binary form). Alternatively a register can be addressed directly as a working register;

Example: `xch R0A2h,r4`
exchanges the values in the register RA2h and working register number 4.

Figure 1-2. Direct Register



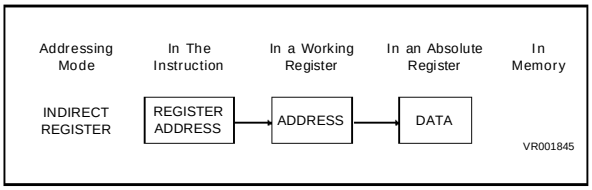
Indirect Addressing Mode

In the Indirect Register Addressing mode, the address of the data does not appear in the instruction but is located in a working register. The address of this register is given in the instruction. The indirect addressing mode is indicated by the use of parentheses.

Example:
If register 200 contains 178 and working register 11 contains 86 then the instruction `ld (r11),R200` loads the value 178 into register 86.

Note: the indirect address can only be contained in a working register.

Figure 1-3. Indirect Register



Indexed Addressing Mode

To address a register using the Indexed mode, an offset value is used to add to an index value (which acts as a base or starting value). The offset value is the Immediate value given in the instruction while the index value is given by the contents of the working register.

Example: if working register 10 contains 55 then the instruction

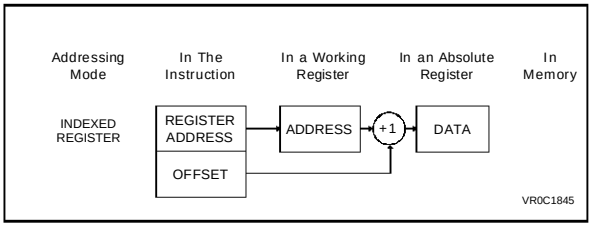
ld 40(r10),r18

loads register 95 (i.e.55+40) with the contents of working register 18.

The Register File never needs an absolute value requiring more than one byte and therefore only requires a short offset and a single register to contain the index.

Note: The index value can only be contained in a working register.

Figure 1-4. Indexed Register



ADDRESSING MODES (Continued)

Indirect Register Post-increment Addressing Mode

In this addressing mode, both destination and source addresses are given by the contents of working registers which are then post-incremented. The address of the memory location is contained in a working register pair, and the address of the register is contained into a single working register. Only working registers may be used to contain the addresses, this mode being indicated by both source and destination using parentheses followed by plus sign.

Example: if working register 8 contains the value 44, working register pair rr2 contains the value 2000, and register 44 contains the value 56, then by using the instruction

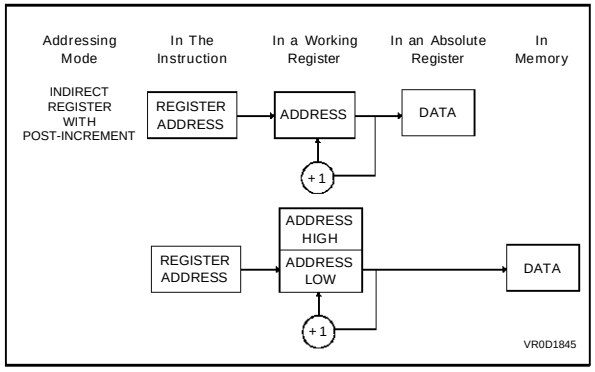
ld (rr2)+,(r8)+

the memory location 2000 will be loaded with the value 56. Immediately following this, the contents of r8 is incremented to 45 and the contents of rr2 is incremented to 2001.

This addressing mode is useful for moving blocks of data either from Register File to Memory or from Memory to Register File.

Direct Bit Addressing Mode

Figure 1-5. Register Indirect Post-Increment



In the direct bit addressing mode, any bit in any working register can be addressed

Examples: `bset r7.3`

This instruction sets the bit 3 of the working register 7.

bld r7.3,r12.6

This instruction loads the bit 6 of the working register 12 in bit 3 of working register 7

1.1.2 Memory Addressing Modes

The memory addressing modes described in this section are available to data and program memory. Thus before addressing the memory, it is necessary to indicate by use of the Set Program/Data Memory instructions, `spm` and `sdm`, in which memory the instructions are working. Since each memory space is 64K byte long, a word address is necessary to specify memory locations.

Direct Addressing Mode

The Memory Direct addressing mode requires the specific location within the memory. This only needs the absolute offset value which can be given in decimal, hex or binary form.

Thus the instruction

ld 12345,r9

loads working register 9 data into memory location 12345

In the memory direct mode, it is possible to use an immediate addressing mode for the source operand.

Examples:

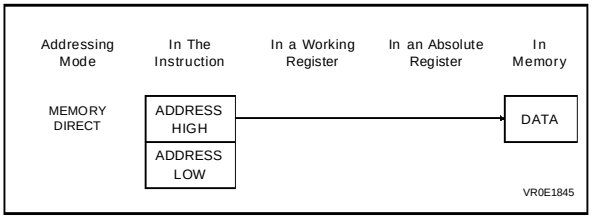
ld 12354,#34

will load the value 34 into the memory location 12354.

ldw 12354,#3457

will load the location pair 12354 and 12355 with the value 3457.

Figure 1-6. Memory Direct



ADDRESSING MODES (Continued)

Indirect Addressing Mode

When using the indirect addressing mode to access memory, the address is contained in a pair of working registers.

Example: if the working register pair r8 and r9 contains the value 2000 then the instruction

ld (rr8),#34

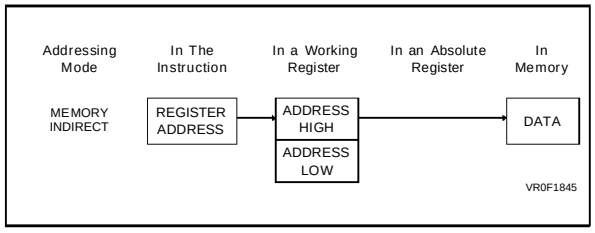
loads the value 34 into memory location 2000.

If the data to be stored is a word then the instruction ldw will automatically interpret the address as a pair of memory locations. So if rr8 contains 2000 then the instruction

ldw (rr8),#3467

loads the memory locations 2000 and 2001 with the value 3467.

Figure 1-7. Memory Indirect



Indirect With Post-increment Adressing Mode

The indirect with post-increment addressing mode is similar to the memory indirect addressing mode but, in addition, after accessing the data in the currently pointed address, the value in the pointing working register pair is incremented. This mode is indicated by a plus sign following a working register pair in parentheses, e.g. (rr4)+.

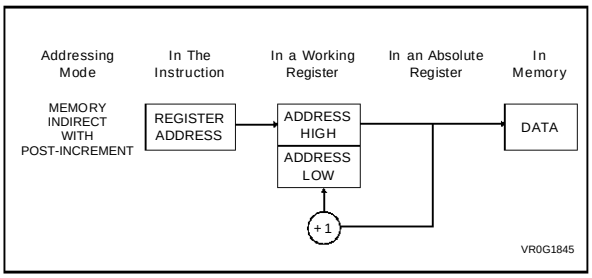
Example:
If the working register pair rr4 (working registers r4 and r5) contains the value 3000 and memory location 3000 contains the value 88, then the instruction

ld R50,(rr4)+

loads register 50 with the value 88 and then the value in rr4 to be incremented to 3001.

This mode uses only working registers to contain the address. Thus the Indirect with Post-Increment addressing mode is most useful in repeated situations when a number of adjacent items of data are required in succession. The use of this addressing mode saves both time and program memory space since it cuts the usual increment instruction.

Figure 1-8. Memory Indirect Post-Increment



Indirect With Pre-decrement Addressing Mode

This indirect memory addressing mode has an automatic pre-decrement. The address can only be contained in working registers and the mode is indicated by a minus sign in front of the working registers which are in parentheses, e.g. -(rr6).

Thus if the working register pair rr6 contains the value 1111 and location 1110 contains the value 40 then the instruction

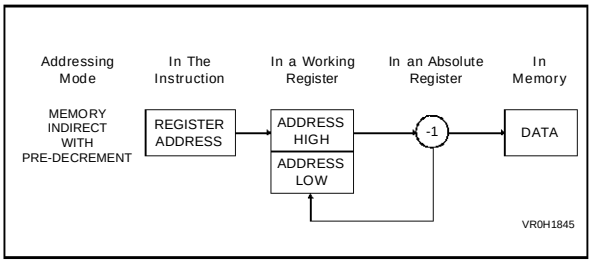
ld R56,-(rr6)

decrements the value in rr6 to 1110 and then loads the value 40 into register 56.

This addressing mode allows the ST9 to deal in the reverse order with data previously managed using the indirect post-increment mode without resetting the pointing registers (of the last post-increment).

The pre-decrement mode has the same benefits of time and program memory saving as the post-increment mode.

Figure 1-9. Memory Indirect Pre-Decrement



ADDRESSING MODES (Continued)

Indexed Addressing Modes

There are three indexed addressing modes, each using an indirect address plus offset format. The index address is given as an indirect address contained in a working register pair, while the offset can be long or short (a word or a byte). The address of the data required is given by the value of the working register pair indicated (the index), plus the value of the given offset. The specification of this offset which differentiates the three modes, is as follows:

- Indexed with an Immediate Short and Long Offset

In these indexed modes the offset is a fixed and Immediate value included in the instruction. It may be either a short or long index as required, this immediate value being added to the address given by the working register pair.

Example: if the working register pair, rr6, contains the value 8000 and memory location 8034 contains the value 254 then the instruction

ld R55,34(rr6)

loads the value 254 into register 55.

Or, as another example, if the working register pair rr2 contains the value 2000 and register 78 contains the value 34 then the instruction.

ld 322(rr2),r78

loaded the value 34 into memory location 2322.

- Indexed with a Register Offset

In this addressing mode, the index is supplied by one pair of working registers and the offset is supplied by a second pair of working registers. The format is rrx(rry), x and y being in the range 0,2,4...12,14.

Example
If working register pair rr0 contains the value 2222 and working register pair rr4 contains 3333 while register 45 contains the value 78 then the instruction

ld rr4(rr0),R45

loads the value 78 into memory location 5555.

Figure 1-10. Memory Indexed with Immediate Short Offset

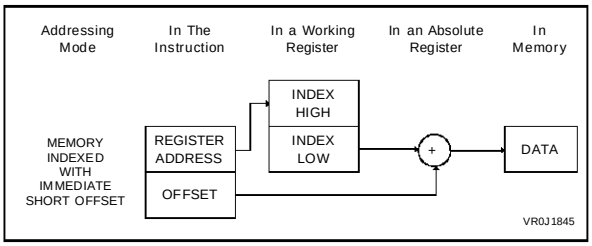


Figure 1-11. Memory Indexed with Immediate Long Offset

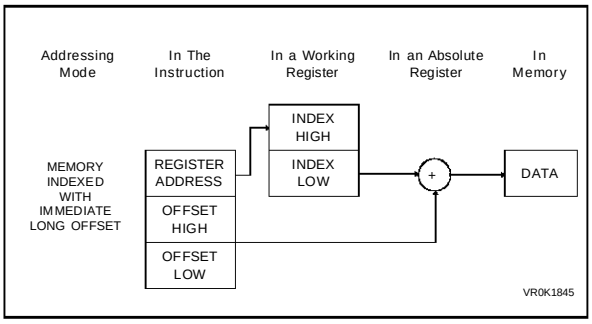
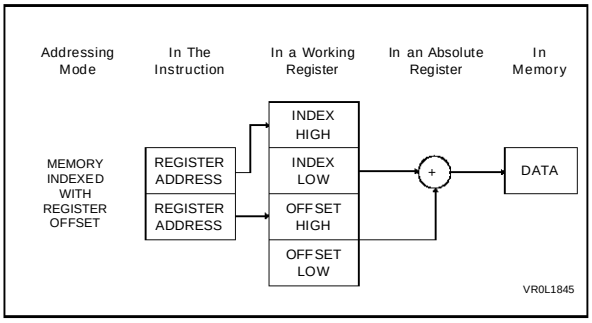


Figure 1-12. Memory Indexed with Register Offset



Indirect Memory Bit Addressing Mode

In the indirect memory bit addressing mode, any bit of Program/Data memory location can be addressed with the `btset` (Bit Test and SET) instruction.

Example

btset (rr8).3

This instruction sets bit 3 of the memory location addressed by the working registers r8, r9 contents.

1.2 INSTRUCTION SET

The ST9 instruction set consists of 87 instruction types which can be divided into eight groups:

- Load (two operands)
- Arithmetic & logic (two operands)
- Arithmetic Logic and Shift (one operand)
- Stack (one operand)
- Multiply & Divide (two operands)
- Boolean (one or two operands)
- Program Control (zero to three operands)
- Miscellaneous (zero to two operands)

The wide range of instructions eases use of the register file and address spaces, reducing operation times, while the register pointers mechanism allows an unmatched code efficiency and ultrafast context switching. A particularly notable feature is the comprehensive "Any Bit, Any Register" (ABAR) addressing capability of the Boolean instructions.

The ST9 can operate with a wide range of data lengths from single bits, 4-bit nibbles which can be in the form of Binary Coded Decimal (BCD) digits, 8-bit bytes, and 16-bit words.

The following summary shows the instructions belonging to each group and the number of operands required for each instruction. The source operand is "src", "dst" is the destination operand, and "cc" is a condition code.

INSTRUCTION SET (Continued)

Load Instructions (Two Operands)

Mnemonic	Operands	Instruction
LD LDW	dst,src dst,src	Load Load Word
LDPP LDPD LDDP LDDD	dst,src dst,src dst,src dst,src	Load Program Memory -> Program Memory Load Data Memory -> Program Memory Load Program Memory -> Data Memory Load Data Memory -> Data Memory

Arithmetic and Logic Instructions (Two Operands)

Mnemonic	Operands	Instruction
ADD ADDW	dst,src dst,src	Add Add Word
ADC ADCW	dst,src dst,src	Add With carry Add Word With Carry
SUB SUBW	dst,src dst,src	Substract Substract Word
SBC SBCW	dst,src dst,src	Substract With Carry Substract Word With Carry
AND ANDW	dst,src dst,src	Logical AND Logical Word AND
OR ORW	dst,src dst,src	Logical OR Logical Word OR
XOR XORW	dst,src dst,src	Logical Exclusive OR Logical Word Exclusive OR
CP CPW	dst,src dst,src	Compare Compare Word
TM TMW	dst,src dst,src	Test Under Mask Test Word Under Mask
TCM TCMW	dst,src dst,src	Test Complement Under Mask Test Word Complement Under Mask

INSTRUCTION SET (Continued)

Arithmetic Logic and Shift Instructions (One Operand)

Mnemonic	Operands	Instruction
INC	dst	Increment
INCW	dst	Increment Word
DEC	dst	Decrement
DECW	dst	Decrement Word
SLA	dst	Shift Left Arithmetic
SLAW	dst	Shift Word Left Arithmetic
SRA	dst	Shift Right Arithmetic
SRAW	dst	Shift Word Right Arithmetic
RRC	dst	Rotate Right Through Carry
RRCW	dst	Rotate Word Right Through Carry
RLC	dst	Rotate Left Through Carry
RLCW	dst	Rotate Word Left Through Carry
ROR	dst	Rotate Right
ROL	dst	Rotate Left
CLR	dst	Clear Register
CPL	dst	Complement Register
SWAP	dst	Swap Nibbles
DA	dst	Decimal Adjust

Stack Instructions (One Operand)

Mnemonic	Operands	Instruction
PUSH	src	Push on System Stack
PUSHW	src	Push Word on System Stack
PEA	src	Push Effective Address on System Stack
POP	dst	Pop From System Stack
POPW	dst	Pop Word from System Stack
PUSHU	src	Push on User Stack
PUSHUW	src	Push Word on User Stack
PEAU	src	Push Effective Address on User Stack
POPU	dst	Pop From User Stack
POPUW	dst	Pop Word From User Stack

INSTRUCTION SET (Continued)

Multiply and Divide Instructions (Two Operands)

Mnemonic	Operands	Instruction
MUL	dst,src	Multiply 8x8
DIV	dst,src	Divide 16/8
DIVWS	dst,src	Divide Word Stepped 32/16

Boolean Instructions (One or Two Operands)

Mnemonic	Operands	Instruction
BSET	dst	Bit Set
BRES	dst	Bit Reset
BCPL	dst	Bit Complement
BTSET	dst	Bit Test and Set
BLD	dst,src	Bit Load
BAND	dst,src	Bit AND
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR

INSTRUCTION SET (Continued)

Program Control Instructions (One, Two or Three Operands)

Mnemonic	Operands	Instruction
RET	Return from Subroutine	
IRET	Return from Interrupt	
WFI		Stop Program Execution and Wait for the next Enabled Interrupt. If a DMA request is present, the CPU executes the DMA service routine and then automatically returns to the WFI
HALT	Stop Program Execution Until Next System Reset	
JR	cc,dst	Jump Relative If Condition is Met
JP	cc,dst	Jump if Condition is Met
JP	dst	Unconditional Jump
CALL	dst	Unconditional Call
BTJF	dst,N	Bit Test and Jump if False
BTJT	dst,N	Bit Test and Jump if True
DJNZ	dst,N	Decrement a Working Register and Jump if Non Zero
DWJNZ	dst,N	Decrement a Register Pair and Jump if Non Zero
CPJFI	dst,N	Compare and Jump on False. Otherwise Post Increment
CPJTI	dst,N	Compare and Jump on True. Otherwise Post Increment

INSTRUCTION SET (Continued)

Miscellaneous (None, One or Two Operands)

Mnemonic	Operands	Instruction
XCH	dst,src	Exchange Registers
SRP	src	Set Register Pointer Long (16 working registers)
SRP0	src	Set Register Pointer 0 (8 LSB working register)
SRP1	src	Set Register Pointer 1 (8 MSB working register)
SPP	src	Set Page Pointer
EXT	dst	Sign Extend
EI	Enable Interrupts	
DI	Disable Interrupts	
SCF	Set Carry Flag	
RCF	Reset Carry Flag	
CCF	Complement Carry Flag	
SPM	Select Program Memory	
SDM	Select Data Memory	
NOP	No Operation	

INSTRUCTION SET (Continued)

1.2.1 ST9 Processor Flags

An important feature of a single chip microcomputer is the ability to test data and make the appropriate action based on the results. In order to provide this facility, FLAGR (register 231) in the register file is used as a flag register. Six bits of this register are used as the following flags:

C - Carry

Z - Zero

S - Sign

V - Overflow

D - Decimal Adjust

H - Half Carry

Bit 1 is available to the user. Bit 0 is the Program/Data Memory selector bit.

The Flag Register is further described in the Architecture Chapter.

1.2.2 Condition Codes

Flags C, Z, S, and OV control the operation of the "conditional" Jump instructions. The next table shows the condition codes and the flag settings.

Note : Some of the Status flags are used to indicate more than one condition e.g . Zero and Equal. In such cases the condition code is the same for both conditions.

Table 1-2. Condition Codes Table

Mnemonic code	Meaning	Flag setting	Hex. value	Binary value
F	Always False	_ _ _ _	0	0000
T	Always True	_ _ _ _	8	1000
C	Carry	C=1	7	0111
NC	Not carry	C=0	F	1111
Z	Zero	Z=1	6	0011
NZ	Not Zero	Z=0	E	1110
PL	Plus	S=0	D	1101
MI	Minus	S=1	5	0101
OV	Overflow	V=1	4	0100
NOV	No Overflow	V=0	C	1100
EQ	Equal	Z=1	6	0110
NE	Not Equal	Z=0	E	1110
GE	Greater Than or Equal	(S xor V)=0	9	1001
LT	Less Than	(S xor V)=1	1	0001
GT	Greater Than	(Z or(S xor V))=0	A	1010
LE	Less Than or Equal	(Z or(S xor V))=1	2	0010
UG	Unsigned Greater Than or Equal	C=0	F	1111
UL	Unsigned Less Than	C=1	7	0111
UGT	Unsigned Greater Than	(C=0 and Z=0)=1	B	1011
ULE	Unsigned Less Than or Equal	(C or Z)=1	3	0011

INSTRUCTION SET (Continued)

1.2.3 Notation

Operands and status flags are represented by a notational shorthand in the detailed instruction description (see programming manual).

The notation for operands (condition codes and address modes) and the actual operands they represent are as follows:

Table 1-3. Notation (Part 1)

Notation	Significance	Actual Operand/Range	
cc	Condition Code		
#N #NN	Immediate Byte Immediate Word	# data # data	where data is a byte expression where data is a word expression
r	Direct Working Register	rn	where n=0-15
R	Direct Register	Rn	where n=0-255
rr	Direct Working Register Pair	rrn	where n is an even number in the range 0-15. (n=0,2,4,6....14)
RR	Direct Register Pair	RRn	where n is an even number in the range 0-254. (n=0,2,4,6....254)
(r)	Indirect Working Register	(rn)	where n=0-15
(R)	Indirect register	(Rn)	where n=0-255
(r)+	Indirect working register post increment	(rn)+	where n=0-15
N(rx)	Indexed register	N(rx)	where x=0-15; N=0-255 (one byte)
N	Memory relative Short Address		Program label or expression in the range +127/-128 starting from the address of the next instruction
NN	Direct Memory Long Address		Program label or expression in the range 0-65535 in memory area
(rr)	Indirect Pair of Working Register Pointers	(rrn)	Where n is an even number in the range 0-15.(n=0,2,4,6....14)
(rr)+	Indirect Pair of Working Register Pointers with Post Increment	(rrn)+	where n is an even number in the range 0-15.(n=0,2,4,6....14)
-(rr)	Indirect Pair of Working Register Pointers with Pre Decrement	-(rrn)	where n is an even number in the range 0-15.(n=0,2,4,6....14)

INSTRUCTION SET (Continued)

Table 1-4. Notation (Part 2)

Notation	Significance	Actual Operand/Range	
N(rrx)	Indexed Pair of Working Register Pointers with Short Offset	N(rrx)	where x is an even number in the range 0-15.(n=0,2,4,6....14) and N is a signed one byte expression between +127/-128
NN(rrx)	Indexed Pair of Working Register Pointers with Long Offset	NN(rrx)	where x is an even number in the range 0-15.(n=0,2,4,6....14) and NN is word expression in the range between 0 and 65535
N(RRx)	Indexed Pair of Register Pointers with Short Offset	N(RRx)	where x is an even number in the range 0-255.(n=0,2,4,6...254) and N is a one byte signed expression in the range +127/-128
NN(RRx)	Indexed Pair of Register Pointers with Long Offset	NN(RRx)	where x is an even number in the range 0-255.(n=0,2,4,6....14) and NN is word expression in the range between 0 and 65535
rr(rrx)	Indexed Pair of Working Registers with a Pair of Working Registers used as Offset	rrn(rrx)	where n and x are two even numbers in the range 0-15. (n,x=0,2,4,6....14)
r.b	Bit pointer in a direct working register	rn.b	n=0.15 and b is a number between 0-7;0 LSB 7 MSB
(rr).b	Bit pointer in a Memory Location using a Pair of Indirect Working Registers as Address Pointer	(rrn).b	where n is an even number in the range 0-15.(n=0,2,4,6....14) and b is a number between 0-7 0 LSB 7 MSB
(RR)	Indirect pair of Register Pointer	(RRn)	where n is an even number in the range 0-255.(n=0,2,4,6....254)

1.3 INSTRUCTION SUMMARY

The following tables summarize the operation for each of the instructions which are listed with their corresponding mnemonic codes, addressing modes, byte counts, timing information, and affected flags.

GENERAL NOTES:

FLAGS STATUS:

- ^ : affected
- - : not affected
- 0 : reset to zero
- 1 : set to one
- ? : undefined

Note: for detailed information on the instruction set refer to the ST9 programming manual.

- dst: destination operand
- src: source operand
- SSP: system stack pointer
- USP: user stack pointer
- PC: program counter
- cc: condition code
- C: carry flag
- Z: zero flag
- S: sign flag
- V: overflow flag
- D: decimal adjust flag
- CIC: central interrupt control register
- DP : data/program memory flag

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ADC : Addition of 2 bytes with carry						
ADC	r	r	2	6	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	R	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	R	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	r	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(r)	2	6	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	(r)	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(rr)	3	12	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	(rr)	3	12	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	NN	4	18	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	N(rrx)	4	24	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	N(rrx)	4	24	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	NN(rrx)	5	26	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	NN(rrx)	5	26	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	rr(rrx)	3	22	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	(rr)+	3	16	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	R	(rr)+	3	16	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	r	-(rr)	3	16	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	-(rr)	3	16	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(r)	r	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(r)	R	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	r	3	18	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	R	3	18	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)+	r	3	22	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	(rr)+	R	3	22	dst<-dst+src+C rr<-rr+1	^ ^ ^ ^ 0 ^
ADC	NN	r	4	20	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	N(rrx)	r	4	26	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	N(rrx)	R	4	26	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN(rrx)	r	5	28	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN(rrx)	R	5	28	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	rr(rrx)	r	3	24	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	-(rr)	r	3	24	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	-(rr)	R	3	22	rr<-rr-1 dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	r	#N	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	R	#N	3	10	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	#N	3	16	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	NN	#N	5	24	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(rr)	(rr)	3	20	dst<-dst+src+C	^ ^ ^ ^ 0 ^
ADC	(RR)	(rr)	3	20	dst<-dst+src+C	^ ^ ^ ^ 0 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ADCW : Add word with carry						
ADCW	rr	rr	2	10	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	RR	3	12	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	RR	3	12	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	rr	3	12	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(r)	3	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	(r)	3	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(rr)	2	16	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	(rr)	3	18	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	NN	4	22	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	N(rrx)	4	28	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	N(rrx)	4	28	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	NN(rrx)	5	30	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	NN(rrx)	5	30	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	rr(rrx)	3	26	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	(rr)+	3	22	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	RR	(rr)+	3	22	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(r)	rr	3	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(r)	RR	3	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	rr	2	30	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	RR	3	30	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)+	rr	3	32	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	(rr)+	RR	3	32	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
ADCW	NN	rr	4	32	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	rr	4	38	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	RR	4	38	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	rr	5	38	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	RR	5	38	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr(rrx)	rr	3	34	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	-(rr)	rr	3	34	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	rr	#NN	4	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	RR	#NN	4	14	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	#NN	4	32	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN	#NN	6	36	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	N(rrx)	#NN	5	36	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	NN(rrx)	#NN	6	38	dst<-dst+src+C	^ ^ ^ ^ ? ?
ADCW	(rr)	(rr)	2	32	dst<-dst+src+C	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ADD : Addition of 2 bytes without carry						
ADD	r	r	2	6	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	R	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	R	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	r	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(r)	2	6	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	(r)	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(rr)	3	12	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	(rr)	3	12	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	NN	4	18	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	N(rrx)	4	24	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	N(rrx)	4	24	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	NN(rrx)	5	26	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	NN(rrx)	5	26	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	rr(rrx)	3	22	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	(rr)+	3	16	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	R	(rr)+	3	16	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	r	-(rr)	3	16	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	-(rr)	3	16	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(r)	r	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(r)	R	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	r	3	18	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	R	3	18	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)+	r	3	22	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	(rr)+	R	3	22	dst<-dst+src rr<-rr+1	^ ^ ^ ^ 0 ^
ADD	NN	r	4	20	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	N(rrx)	r	4	26	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	N(rrx)	R	4	26	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN(rrx)	r	5	28	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN(rrx)	R	5	28	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	rr(rrx)	r	3	24	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	-(rr)	r	3	22	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	-(rr)	R	3	22	rr<-rr-1 dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	r	#N	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	R	#N	3	10	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	#N	3	16	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	NN	#N	5	24	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(rr)	(rr)	3	20	dst<-dst+src	^ ^ ^ ^ 0 ^
ADD	(RR)	(rr)	3	20	dst<-dst+src	^ ^ ^ ^ 0 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ADDW : Add word without carry						
ADDW	rr	rr	2	10	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	RR	3	12	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	RR	3	12	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	rr	3	12	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(r)	3	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	(r)	3	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(rr)	2	16	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	(rr)	3	18	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	NN	4	22	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	N(rrx)	4	28	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	N(rrx)	4	28	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	NN(rrx)	5	30	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	NN(rrx)	5	30	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	rr(rrx)	3	26	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	(rr)+	3	22	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	RR	(rr)+	3	22	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(r)	rr	3	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(r)	RR	3	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	rr	2	30	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	RR	3	30	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)+	rr	3	32	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	(rr)+	RR	3	32	dst<-dst+src rr<-rr+2	^ ^ ^ ^ ? ?
ADDW	NN	rr	4	32	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	rr	4	38	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	RR	4	38	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	rr	5	38	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	RR	5	38	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr(rrx)	rr	3	34	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	rr	#NN	4	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	RR	#NN	4	14	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	#NN	4	32	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN	#NN	6	36	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	N(rrx)	#NN	5	36	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	NN(rrx)	#NN	6	38	dst<-dst+src	^ ^ ^ ^ ? ?
ADDW	(rr)	(rr)	2	32	dst<-dst+src	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
AND : Logical AND between 2 bytes						
AND	r	r	2	6	dst<-dst AND src	- ^ ^ 0 - -
AND	R	R	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	r	R	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	R	r	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(r)	2	6	dst<-dst AND src	- ^ ^ 0 - -
AND	R	(r)	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(rr)	3	12	dst<-dst AND src	- ^ ^ 0 - -
AND	R	(rr)	3	12	dst<-dst AND src	- ^ ^ 0 - -
AND	r	NN	4	18	dst<-dst AND src	- ^ ^ 0 - -
AND	r	N(rrx)	4	24	dst<-dst AND src	- ^ ^ 0 - -
AND	R	N(rrx)	4	24	dst<-dst AND src	- ^ ^ 0 - -
AND	r	NN(rrx)	5	26	dst<-dst AND src	- ^ ^ 0 - -
AND	R	NN(rrx)	5	26	dst<-dst AND src	- ^ ^ 0 - -
AND	r	rr(rrx)	3	22	dst<-dst AND src	- ^ ^ 0 - -
AND	r	(rr)+	3	16	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	R	(rr)+	3	16	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	r	-(rr)	3	16	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	R	-(rr)	3	16	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	(r)	r	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	(r)	R	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	r	3	18	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	R	3	18	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)+	r	3	22	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	(rr)+	R	3	22	dst<-dst AND src rr<-rr+1	- ^ ^ 0 - -
AND	NN	r	4	20	dst<-dst AND src	- ^ ^ 0 - -
AND	N(rrx)	r	4	26	dst<-dst AND src	- ^ ^ 0 - -
AND	N(rrx)	R	4	26	dst<-dst AND src	- ^ ^ 0 - -
AND	NN(rrx)	r	5	28	dst<-dst AND src	- ^ ^ 0 - -
AND	NN(rrx)	R	5	28	dst<-dst AND src	- ^ ^ 0 - -
AND	rr(rrx)	r	3	24	dst<-dst AND src	- ^ ^ 0 - -
AND	-(rr)	r	3	22	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	-(rr)	R	3	22	rr<-rr-1 dst<-dst AND src	- ^ ^ 0 - -
AND	r	#N	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	R	#N	3	10	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	#N	3	16	dst<-dst AND src	- ^ ^ 0 - -
AND	NN	#N	5	24	dst<-dst AND src	- ^ ^ 0 - -
AND	(rr)	(rr)	3	20	dst<-dst AND src	- ^ ^ 0 - -
AND	(RR)	(rr)	3	20	dst<-dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ANDW : Logical AND between two words						
ANDW	rr	rr	2	10	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	RR	3	12	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	RR	3	12	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	rr	3	12	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(r)	3	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	(r)	3	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(rr)	2	16	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	(rr)	3	18	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	NN	4	22	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	N(rrx)	4	28	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	N(rrx)	4	28	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	NN(rrx)	5	30	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	NN(rrx)	5	30	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	rr(rrx)	3	26	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	(rr)+	3	22	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	RR	(rr)+	3	22	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	(r)	rr	3	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(r)	RR	3	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	rr	2	30	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	RR	3	30	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)+	rr	3	32	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	(rr)+	RR	3	32	dst<-dst AND src rr<-rr+2	- ^ ^ 0 - -
ANDW	NN	rr	4	32	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	rr	4	38	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	RR	4	38	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	rr	5	38	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	RR	5	38	dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr(rrx)	rr	3	34	dst<-dst AND src	- ^ ^ 0 - -
ANDW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst AND src	- ^ ^ 0 - -
ANDW	rr	#NN	4	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	RR	#NN	4	14	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	#NN	4	32	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN	#NN	6	36	dst<-dst AND src	- ^ ^ 0 - -
ANDW	N(rrx)	#NN	5	36	dst<-dst AND src	- ^ ^ 0 - -
ANDW	NN(rrx)	#NN	6	38	dst<-dst AND src	- ^ ^ 0 - -
ANDW	(rr)	(rr)	2	32	dst<-dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
BAND : Bit AND						
BAND BAND	r.b r.b	r.b r.!b	3 3	14 14	dst bit<-dst bit AND src bit dst bit<-dst bit AND complemented src bit	- - - - - - - - - - - -
BCPL : Bit Complement						
BCPL	r.b		2	6	dst bit<-dst bit complemented	- - - - - -
BLD : Bit Load						
BLD BLD	r.b r.b	r.b r.!b	3 3	14 14	dst bit<-src bit dst bit<-src bit complemented	- - - - - - - - - - - -
BOR : Bit OR						
BOR BOR	r.b r.b	r.b r.!b	3 3	14 14	dst bit<-dst bit OR src bit dst bit<-dst bit OR complemented src bit	- - - - - - - - - - - -
BRES : Bit Reset						
BRES	r.b		2	6	dst bit<- 0 - - - - - -	
BSET : Bit Set						
BSET	r.b		2	6	dst bit<- 1 - - - - - -	
BTJF, BTJT : Bit test and jump						
BTJF BTJT	r.b r.b	N N	3 3	14/16 14/16	If test bit is 0, PC<-PC+N If test bit is 1, PC<-PC+N	- - - - - - - - - - - -
BXOR : Bit Exclusive OR						
BXOR BXOR	r.b r.b	r.b r.!b	3 3	14 14	dst bit<-dst bit XOR src bit dst bit<-dst bit XOR complemented src bit	- - - - - - - - - - - -
BTSET : Bit Test and Set						
BTSET BTSET	r.b (rr).b		2 2	8 20	If test bit = 0, test bit <-1,Z<-1 If test bit = 0, test bit <-1,Z<-1	- ^ ^ 0 - - - ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
CALL : Call a subroutine						
CALL	NN	3	18		SSP<-SSP-2,(SP)< PC, PC<-dst	- - - - - -
CALL	(rr)	2	16		" "	- - - - - -
CALL	(RR)	2	16		" "	- - - - - -
CCF : Complement Carry Flag						
CCF		1 6	C <- C		complemented	- - - - - -
CLR : Clear register						
CLR	r		2	6	dst<-0	- - - - - -
CLR	R		2	6	dst<-0	- - - - - -
CLR	(r)		2	6	dst<-0	- - - - - -
CLR	(R)		2	6	dst<-0	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
CP : Compare bytes						
CP	r	r	2	6	dst-src	^ ^ ^ ^ - -
CP	R	R	3	10	dst-src	^ ^ ^ ^ - -
CP	r	R	3	10	dst-src	^ ^ ^ ^ - -
CP	R	r	3	10	dst-src	^ ^ ^ ^ - -
CP	r	(r)	2	6	dst-src	^ ^ ^ ^ - -
CP	R	(r)	3	10	dst-src	^ ^ ^ ^ - -
CP	r	(rr)	3	12	dst-src	^ ^ ^ ^ - -
CP	R	(rr)	3	12	dst-src	^ ^ ^ ^ - -
CP	r	NN	4	18	dst-src	^ ^ ^ ^ - -
CP	r	N(rrx)	4	24	dst-src	^ ^ ^ ^ - -
CP	R	N(rrx)	4	24	dst-src	^ ^ ^ ^ - -
CP	r	NN(rrx)	5	26	dst-src	^ ^ ^ ^ - -
CP	R	NN(rrx)	5	26	dst-src	^ ^ ^ ^ - -
CP	r	rr(rrx)	3	22	dst-src	^ ^ ^ ^ - -
CP	r	(rr)+	3	16	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	R	(rr)+	3	16	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	r	-(rr)	3	16	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	R	-(rr)	3	16	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	(r)	r	3	10	dst-src	^ ^ ^ ^ - -
CP	(r)	R	3	10	dst-src	^ ^ ^ ^ - -
CP	(rr)	r	3	18	dst-src	^ ^ ^ ^ - -
CP	(rr)	R	3	18	dst-src	^ ^ ^ ^ - -
CP	(rr)+	r	3	22	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	(rr)+	R	3	22	dst-src,rr<-rr+1	^ ^ ^ ^ - -
CP	NN	r	4	20	dst-src	^ ^ ^ ^ - -
CP	N(rrx)	r	4	26	dst-src	^ ^ ^ ^ - -
CP	N(rrx)	R	4	26	dst-src	^ ^ ^ ^ - -
CP	NN(rrx)	r	5	28	dst-src	^ ^ ^ ^ - -
CP	NN(rrx)	R	5	28	dst-src	^ ^ ^ ^ - -
CP	rr(rrx)	r	3	24	dst-src	^ ^ ^ ^ - -
CP	-(rr)	r	3	22	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	-(rr)	R	3	22	rr<-rr-1,dst-src	^ ^ ^ ^ - -
CP	r	#N	3	10	dst-src	^ ^ ^ ^ - -
CP	R	#N	3	10	dst-src	^ ^ ^ ^ - -
CP	(rr)	#N	3	16	dst-src	^ ^ ^ ^ - -
CP	NN	#N	5	22	dst-src	^ ^ ^ ^ - -
CP	(rr)	(rr)	3	18	dst-src	^ ^ ^ ^ - -
CP	(RR)	(rr)	3	18	dst-src	^ ^ ^ ^ - -
CPL : Complement register						
CPL	r		2	6	dst<- NOT dst	- ^ ^ 0 - -
CPL	R		2	6	dst<- NOT dst	- ^ ^ 0 - -
CPL	(r)		2	6	dst<- NOT dst	- ^ ^ 0 - -
CPL	(R)		2	6	dst<- NOT dst	- ^ ^ 0 - -
CPJFI, CPJTI : Compare with post-increment						
CPJFI	(rr)	r,N	3	22/24	If compare not verified jump otherwise post-increment	- - - - - -
CPJTI	(rr)	r,N	3	22/24	If compare verified jump otherwise post-increment	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
CPW : Compare word						
CPW	rr	rr	2	10	dst-src	^ ^ ^ ^ - -
CPW	RR	RR	3	12	dst-src	^ ^ ^ ^ - -
CPW	rr	RR	3	12	dst-src	^ ^ ^ ^ - -
CPW	RR	rr	3	12	dst-src	^ ^ ^ ^ - -
CPW	rr	(r)	3	14	dst-src	^ ^ ^ ^ - -
CPW	RR	(r)	3	14	dst-src	^ ^ ^ ^ - -
CPW	rr	(rr)	2	16	dst-src	^ ^ ^ ^ - -
CPW	RR	(rr)	3	18	dst-src	^ ^ ^ ^ - -
CPW	rr	NN	4	22	dst-src	^ ^ ^ ^ - -
CPW	rr	N(rrx)	4	28	dst-src	^ ^ ^ ^ - -
CPW	RR	N(rrx)	4	28	dst-src	^ ^ ^ ^ - -
CPW	rr	NN(rrx)	5	30	dst-src	^ ^ ^ ^ - -
CPW	RR	NN(rrx)	5	30	dst-src	^ ^ ^ ^ - -
CPW	rr	rr(rrx)	3	26	dst-src	^ ^ ^ ^ - -
CPW	rr	(rr)+	3	22	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	RR	(rr)+	3	22	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	rr	-(rr)	3	24	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	RR	-(rr)	3	24	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	(r)	rr	3	14	dst-src	^ ^ ^ ^ - -
CPW	(r)	RR	3	14	dst-src	^ ^ ^ ^ - -
CPW	(rr)	rr	2	26	dst-src	^ ^ ^ ^ - -
CPW	(rr)	RR	3	28	dst-src	^ ^ ^ ^ - -
CPW	(rr)+	rr	3	30	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	(rr)+	RR	3	30	dst-src rr<-rr+2	^ ^ ^ ^ - -
CPW	NN	rr	4	30	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	rr	4	36	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	RR	4	36	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	rr	5	36	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	RR	5	36	dst-src	^ ^ ^ ^ - -
CPW	rr(rrx)	rr	3	32	dst-src	^ ^ ^ ^ - -
CPW	-(rr)	rr	3	30	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	-(rr)	RR	3	30	rr<-rr-2 dst-src	^ ^ ^ ^ - -
CPW	rr	#NN	4	14	dst-src	^ ^ ^ ^ - -
CPW	RR	#NN	4	14	dst-src	^ ^ ^ ^ - -
CPW	(rr)	#NN	4	30	dst-src	^ ^ ^ ^ - -
CPW	NN	#NN	6	34	dst-src	^ ^ ^ ^ - -
CPW	N(rrx)	#NN	5	34	dst-src	^ ^ ^ ^ - -
CPW	NN(rrx)	#NN	6	36	dst-src	^ ^ ^ ^ - -
CPW	(rr)	(rr)	2	32	dst-src	^ ^ ^ ^ - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
DA : Decimal adjust						
DA	r		2	6	dst<- DA dst	^ ^ ^ ? - -
DA	R		2	6	dst<- DA dst	^ ^ ^ ? - -
DA	(r)		2	6	dst<- DA dst	^ ^ ^ ? - -
DA	(R)		2	6	dst<- DA dst	^ ^ ^ ? - -
DEC : Decrement						
DEC	r		2	6	dst<- dst-1	- ^ ^ ^ - -
DEC	R		2	6	dst<- dst-1	- ^ ^ ^ - -
DEC	(r)		2	6	dst<- dst-1	- ^ ^ ^ - -
DEC	(R)		2	6	dst<- dst-1	- ^ ^ ^ - -
DECW : Decrement Word						
DECW	rr		2	8	dst<-dst-1	- ^ ^ ^ - -
DECW	RR		2	8	dst<-dst-1	- ^ ^ ^ - -
DI : Disable Interrupts						
DI		1		6	Bit 4 of the CIC Register is set to 0	- - - - - -
DIV : Divide 16 by 8						
DIV	rr	r	2	28/20	dst / src <- dst high=remainder 16/8 <- dst low=result	note 1
DIVWS : Divide Word Stepped 32 by 16						
DIVWS	rrhigh rrlow	rr	3	28	32/16 note 1	
DJNZ : Decrement a working register and Jump if Non Zero						
DJNZ	r	N	2	10/12	r <- r-1, If r=0 then PC<-PC+N	note 2
DWJNZ : Decrement a register pair and Jump if Non Zero						
DWJNZ	rr	N	3	12/16	rr<-rr-1, If rr=0 then PC<-PC+N	note 2
DWJNZ	RR	N	3	12/16	RR<-RR-1,If RR=0 then PC<-PC+N	

Notes :
1. Refer to the ST9 Programming Manual for detailed information.
2. Working registers in groups D, E and F are not allowed.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
EI : Enable Interrupts						
EI		1		6	Bit 4 of the CICR register is set to 1	- - - - - -
EXT : Sign extend						
EXT	rr		2	10	r(7) —> r(n) n=8-15	- - - - - -
EXT	RR		2	10	R(7) —> R(n) n=8-15	- - - - - -
HALT : Halt Operation						
HALT			2	6	Stops all internal clocks until next system reset if not in Watchdog Mode	- - - - - -
INC : Increment						
INC	r		2	6	dst<- dst+1	- ^ ^ ^ - -
INC	R		2	6	dst<- dst+1	- ^ ^ ^ - -
INC	(r)		2	6	dst<- dst+1	- ^ ^ ^ - -
INC	(R)		2	6	dst<- dst+1	- ^ ^ ^ - -
INCW : Increment Word						
INCW	rr		2	8	dst<-dst+1	- ^ ^ ^ - -
INCW	RR		2	8	dst<-dst+1	- ^ ^ ^ - -
IRET : Return from Interrupt Routine						
IRET			1	16	FLAGS<-(SSP),SSP<-SSP+1, PC<-(SSP), SSP<-SPP+2, CIC(4)<-1	note 1
JP : Jump to a Routine						
JP	NN		3	10	PC<-dst	- - - - - -
JP	(rr)		2	8	PC<-dst	- - - - - -
JP	(RR)		2	8	PC<-dst	- - - - - -
JPcc	NN		3	10	IF cc(condition code) is true, PC<-dst	- - - - - -
JRcc : Conditional Relative Jump to a Routine						
JRcc	N		2	10/12	IF cc(condition code)is true, PC<-PC+dst	- - - - - -

Note 1 : All flags are restored to original setting (before interrupt occured).

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
LD : Load byte instructions						
LD	r	r	2	6	dst<-src	- - - - - -
LD	R	R	3	10	dst<-src	- - - - - -
LD	r	R	2	6	dst<-src	- - - - - -
LD	R	r	2	6	dst<-src	- - - - - -
LD	r	(r)	2	6	dst<-src	- - - - - -
LD	R	(r)	3	10	dst<-src	- - - - - -
LD	r	(rr)	2	10	dst<-src	- - - - - -
LD	R	(rr)	3	12	dst<-src	- - - - - -
LD	r	NN	4	18	dst<-src	- - - - - -
LD	r	N(rx)	3	10	dst<-src	- - - - - -
LD	r	N(rrx)	4	24	dst<-src	- - - - - -
LD	R	N(rrx)	4	24	dst<-src	- - - - - -
LD	r	NN(rrx)	5	26	dst<-src	- - - - - -
LD	R	NN(rrx)	5	26	dst<-src	- - - - - -
LD	r	rr(rrx)	3	22	dst<-src	- - - - - -
LD	r	(rr)+	3	16	dst<-src,rr<-rr+1	- - - - - -
LD	R	(rr)+	3	16	dst<-src,rr<-rr+1	- - - - - -
LD	r	-(rr)	3	16	rr<-rr-1,dst<-src	- - - - - -
LD	R	-(rr)	3	16	rr<-rr-1,dst<-src	- - - - - -
LD	(r)	r	2	6	dst<-src	- - - - - -
LD	(r)	R	3	10	dst<-src	- - - - - -
LD	(rr)	r	2	10	dst<-src	- - - - - -
LD	(rr)	R	3	14	dst<-src	- - - - - -
LD	(rr)+	r	3	18	dst<-src,rr<-rr+1	- - - - - -
LD	(rr)+	R	3	18	dst<-src,rr<-rr+1	- - - - - -
LD	NN	r	4	18	dst-src	- - - - - -
LD	N(rx)	r	3	10	dst-src	- - - - - -
LD	N(rrx)	r	4	24	dst-src	- - - - - -
LD	N(rrx)	R	4	24	dst-src	- - - - - -
LD	NN(rrx)	r	5	26	dst-src	- - - - - -
LD	NN(rrx)	R	5	26	dst-src	- - - - - -
LD	rr(rrx)	r	3	22	dst-src	- - - - - -
LD	-(rr)	r	3	18	rr<-rr-1,dst<-src	- - - - - -
LD	-(rr)	R	3	18	rr<-rr-1,dst<-src	- - - - - -
LD	r	#N	2	6	dst<-src	- - - - - -
LD	R	#N	3	10	dst<-src	- - - - - -
LD	(rr)	#N	3	12	dst<-src	- - - - - -
LD	NN	#N	5	20	dst<-src	- - - - - -
LD	(rr)	(rr)	3	16	dst<-src	- - - - - -
LD	(RR)	(rr)	3	16	dst<-src,	- - - - - -
LD	(r)+	(rr)+	2	14	rr<-rr+1,r<-r+1	- - - - - -
LD	(rr)+	(r)+	2	18	rr<-rr+1,r<-r+1	- - - - - -
LDPP,LDDP,LDPD, LDDD : Load from / to program / data memory						
LDPP	(rr)+	(rr)+	2	16	dst<-src (1),rr<-rr+1	- - - - - -
LDDP	(rr)+	(rr)+	2	16	dst<-src (2),rr<-rr+1	- - - - - -
LDPD	(rr)+	(rr)+	2	16	dst<-src (3),rr<-rr+1	- - - - - -
LDDD	(rr)+	(rr)+	2	16	dst<-src (4),rr<-rr+1	- - - - - -

Notes:
1. dst in Program Memory, src in Program Memory
2. dst in Data Memory, src in Program Memory
3. dst in Program Memory, src in Data Memory
4. dst in Data Memory, src in Data Memory

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
LDW : Load word instructions						
LDW	rr	rr	2	10	dst<-src	- - - - - -
LDW	RR	RR	3	10	dst<-src	- - - - - -
LDW	rr	RR	3	10	dst<-src	- - - - - -
LDW	RR	rr	3	10	dst<-src	- - - - - -
LDW	rr	(r)	3	10	dst<-src	- - - - - -
LDW	RR	(r)	3	10	dst<-src	- - - - - -
LDW	rr	(rr)	2	16	dst<-src	- - - - - -
LDW	RR	(rr)	3	18	dst<-src	- - - - - -
LDW	rr	NN	4	22	dst<-src	- - - - - -
LDW	rr	N(rx)	3	16	dst<-src	- - - - - -
LDW	rr	N(rrx)	4	28	dst<-ssc	- - - - - -
LDW	RR	N(rrx)	4	28	dst<-src	- - - - - -
LDW	rr	NN(rrx)	5	30	dst<-src	- - - - - -
LDW	RR	NN(rrx)	5	30	dst<-src	- - - - - -
LDW	rr	rr(rrx)	3	24	dst<-src	- - - - - -
LDW	rr	(rr)+	3	20	dst<-src,rr<-rr+2	- - - - - -
LDW	RR	(rr)+	3	20	dst<-src,rr<-rr+2	- - - - - -
LDW	rr	-(rr)	3	22	rr<-rr-2,dst<-src	- - - - - -
LDW	RR	-(rr)	3	22	rr<-rr-2,dst<-src	- - - - - -
LDW	(r)	rr	3	10	dst<-src	- - - - - -
LDW	(r)	RR	3	10	dst<-src	- - - - - -
LDW	(rr)	rr	2	18	dst<-src	- - - - - -
LDW	(rr)	RR	3	20	dst<-src	- - - - - -
LDW	(rr)+	rr	3	24	rr<-rr+2,dst<-src	- - - - - -
LDW	(rr)+	RR	3	24	rr<-rr+2,dst<-src	- - - - - -
LDW	NN	rr	4	22	dst<-src	- - - - - -
LDW	N(rx)	rr	3	14	dst<-src	- - - - - -
LDW	N(rrx)	RR	4	26	dst<-src	- - - - - -
LDW	N(rrx)	rr	4	26	dst<-src	- - - - - -
LDW	NN(rrx)	RR	5	28	dst<-src	- - - - - -
LDW	NN(rrx)	rr	5	28	dst<-src	- - - - - -
LDW	rr(rrx)	rr	3	24	dst<-src	- - - - - -
LDW	-(rr)	rr	3	26	rr<-rr-2,dst<-src	- - - - - -
LDW	-(rr)	RR	3	26	rr<-rr-2,dst<-src	- - - - - -
LDW	rr	#NN	4	12	dst<-src	- - - - - -
LDW	RR	#NN	4	12	dst<-src	- - - - - -
LDW	(rr)	#NN	4	22	dst<-src	- - - - - -
LDW	N(rrx)	#NN	5	28	dst<-src	- - - - - -
LDW	NN(rrx)	#NN	6	30	dst<-src	- - - - - -
LDW	NN	#NN	6	26	dst<-src	- - - - - -
LDW	(rr)	(rr)	2	22	dst<-src	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
MUL : Multiply						
MUL	rr	r	2	22	dst <- dst x src, 8 x 8 multiply	note 1
NOP : No operation						
NOP		1 6	No Operation		- - - - - -	
OR : Logical OR between 2 bytes						
OR	r	r	2	6	dst<-dst OR src	- ^ ^ 0 - -
OR	R	R	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	r	R	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	R	r	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(r)	2	6	dst<-dst OR src	- ^ ^ 0 - -
OR	R	(r)	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(rr)	3	12	dst<-dst OR src	- ^ ^ 0 - -
OR	R	(rr)	3	12	dst<-dst OR src	- ^ ^ 0 - -
OR	r	NN	4	18	dst<-dst OR src	- ^ ^ 0 - -
OR	r	N(rrx)	4	24	dst<-dst OR src	- ^ ^ 0 - -
OR	R	N(rrx)	4	24	dst<-dst OR src	- ^ ^ 0 - -
OR	r	NN(rrx)	5	26	dst<-dst OR src	- ^ ^ 0 - -
OR	R	NN(rrx)	5	26	dst<-dst OR src	- ^ ^ 0 - -
OR	r	rr(rrx)	3	22	dst<-dst OR src	- ^ ^ 0 - -
OR	r	(rr)+	3	16	dst<-dst OR src rr<-rr+1	- ^ ^ 0 - -
OR	R	(rr)+	3	16	dst<-dst OR src rr<-rr+1	- ^ ^ 0 - -
OR	r	-(rr)	3	16	rr<-rr-1 dst<-dst OR src	- ^ ^ 0 - -
OR	R	-(rr)	3	16	rr<-rr-1 dst<-dst OR src	- ^ ^ 0 - -
OR	(r)	r	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	(r)	R	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	r	3	18	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	R	3	18	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)+	r	3	22	dst<-dst OR src rr<-rr+1	- ^ ^ 0 - -
OR	(rr)+	R	3	22	dst<-dst OR src rr<-rr+1	- ^ ^ 0 - -
OR	NN	r	4	20	dst<-dst OR src	- ^ ^ 0 - -
OR	N(rrx)	r	4	26	dst<-dst OR src	- ^ ^ 0 - -
OR	N(rrx)	R	4	26	dst<-dst OR src	- ^ ^ 0 - -
OR	NN(rrx)	r	5	28	dst<-dst OR src	- ^ ^ 0 - -
OR	NN(rrx)	R	5	28	dst<-dst OR src	- ^ ^ 0 - -
OR	rr(rrx)	r	3	24	dst<-dst OR src	- ^ ^ 0 - -
OR	-(rr)	r	3	22	dst<-dst OR src rr<-rr-1	- ^ ^ 0 - -
OR	-(rr)	R	3	22	dst<-dst OR src rr<-rr-1	- ^ ^ 0 - -
OR	r	#N	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	R	#N	3	10	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	#N	3	16	dst<-dst OR src	- ^ ^ 0 - -
OR	NN	#N	5	24	dst<-dst OR src	- ^ ^ 0 - -
OR	(rr)	(rr)	3	20	dst<-dst OR src	- ^ ^ 0 - -
OR	(RR)	(rr)	3	20	dst<-dst OR src	- ^ ^ 0 - -

Note 1. Refer to ST9 programming manual for detailed information.

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
ORW : Logical OR between two words						
ORW	rr	rr	2	10	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	RR	3	12	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	RR	3	12	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	rr	3	12	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(r)	3	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	(r)	3	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(rr)	2	16	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	(rr)	3	18	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	NN	4	22	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	N(rrx)	4	28	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	N(rrx)	4	28	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	NN(rrx)	5	30	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	NN(rrx)	5	30	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	rr(rrx)	3	26	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	(rr)+	3	22	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	RR	(rr)+	3	22	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	(r)	rr	3	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	(r)	RR	3	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	rr	2	30	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	RR	3	30	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)+	rr	3	32	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	(rr)+	RR	3	32	dst<-dst OR src rr<-rr+2	- ^ ^ 0 - -
ORW	NN	rr	4	32	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	rr	4	38	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	RR	4	38	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	rr	5	38	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	RR	5	38	dst<-dst OR src	- ^ ^ 0 - -
ORW	rr(rrx)	rr	3	34	dst<-dst OR src	- ^ ^ 0 - -
ORW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst OR src	- ^ ^ 0 - -
ORW	rr	#NN	4	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	RR	#NN	4	14	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	#NN	4	32	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN	#NN	6	36	dst<-dst OR src	- ^ ^ 0 - -
ORW	N(rrx)	#NN	5	36	dst<-dst OR src	- ^ ^ 0 - -
ORW	NN(rrx)	#NN	6	38	dst<-dst OR src	- ^ ^ 0 - -
ORW	(rr)	(rr)	2	32	dst<-dst OR src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
PEA : Push effective address on system stack						
PEA		N(rrx)	4	20	SSP<-USP-2, (SSP)<-rrx+N	- - - - - -
PEA		NN(rrx)	5	26	SSP<-USP-2, (SSP)<-rrx+N	- - - - - -
PEA		N(RRx)	4	20	SSP<-USP-2, (SSP)<-RRx+N	- - - - - -
PEA		NN(RRx)	5	26	SSP<-USP-2, (SSP)<-RRx+N	- - - - - -
PEAU : Push effective address on user stack						
PEAU		N(rrx)	4	20	USP<-USP-2, (USP)<-rrx+N	- - - - - -
PEAU		NN(rrx)	5	26	USP<-USP-2, (USP)<-rrx+N	- - - - - -
PEAU		N(RRx)	4	20	USP<-USP-2, (USP)<-RRx+N	- - - - - -
PEAU		NN(RRx)	5	26	USP<-USP-2, (USP)<-RRx+N	- - - - - -
POP : Pop system stack						
POP	r		2	10	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	R		2	10	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	(r)		2	10	dst<-(SSP), SSP<-SSP+1	- - - - - -
POP	(R)		2	10	dst<-(SSP), SSP<-SSP+1	- - - - - -
POPU : Pop user stack						
POPU	r		2	10	dst<-(USP), USP<-USP+1	- - - - - -
POPU	R		2	10	dst<-(USP), USP<-USP+1	- - - - - -
POPU	(r)		2	10	dst<-(USP), USP<-USP+1	- - - - - -
POPU	(R)		2	10	dst<-(USP), USP<-USP+1	- - - - - -
POPUW : Pop word from user stack						
POPUW	rr		2	14	dst<-(USP), USP<-USP+2	- - - - - -
POPUW	RR		2	14	dst<-(USP), USP<-USP+2	- - - - - -
POPW : Pop word from system stack						
POPW	rr		2	14	dst<-(SSP), SSP<-SSP+2	- - - - - -
POPW	RR		2	14	dst<-(SSP), SSP<-SSP+2	- - - - - -
PUSH : Push system stack						
PUSH		r	2	10	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		R	2	10	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		(r)	2	10	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		(R)	2	10	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSH		#N	3	16	SSP<-SSP-1, (SSP)<-src	- - - - - -
PUSHU : Push user stack						
PUSHU		r	2	10	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		R	2	10	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		(r)	2	10	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		(R)	2	10	USP<-USP-1, (USP)<-src	- - - - - -
PUSHU		#N	3	16	USP<-USP-1, (USP)<-src	- - - - - -
PUSHUW : Push word on user stack						
PUSHUW		rr	2	12	USP<-USP-2, (USP)<-src	- - - - - -
PUSHUW		RR	2	12	USP<-USP-2, (USP)<-src	- - - - - -
PUSHUW		#NN	4	20	USP<-USP-2, (USP)<-src	- - - - - -
PUSHW : Push Word on System Stack						
PUSHW		rr	2	12	SSP<-SSP-2, (SSP)<-src	- - - - - -
PUSHW		RR	2	12	SSP<-SSP-2, (SSP)<-src	- - - - - -
PUSHW		#NN	4	20	SSP<-SSP-2, (SSP)<-src	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
RCF : Reset carry flag						
RCF			1	6	C <- 0	0 - - - - -
RET : Return from subroutine						
RET			1	12	PC <- (SSP), SSP <- SPP+2	- - - - - -
RLC : Rotate left through carry						
RLC	r		2	6	dst(0)<-C, C<-dst(7) dst(n+1)<-dst(n) n=0-6	^ ^ ^ ^ - -
RLC	R		2	6	" "	^ ^ ^ ^ - -
RLC	(r)		2	6	" "	^ ^ ^ ^ - -
RLC	(R)		2	6	" "	^ ^ ^ ^ - -
RLCW : Rotate word left through carry						
RLCW	rr		2	8	dst(0)<-C, C<-dst(15) dst(n+1)<-dst(n) n=0-14	
RLCW	RR		2	8	" "	
ROL : Rotate left						
ROL	r		2	6	C<-dst(7), dst(0)<-dst(7) dst(n+1)<-dst(n) n=0-6	^ ^ ^ ^ - -
ROL	R		2	6	" "	^ ^ ^ ^ - -
ROL	(r)		2	6	" "	^ ^ ^ ^ - -
ROL	(R)		2	6	" "	^ ^ ^ ^ - -
ROR : Rotate right						
ROR	r		2	6	C<-dst(0), dst(7)<-dst(0) dst(n)<-dst(n+1) n=0-6	^ ^ ^ ^ - -
ROR	R		2	6	" "	^ ^ ^ ^ - -
ROR	(r)		2	6	" "	^ ^ ^ ^ - -
ROR	(R)		2	6	" "	^ ^ ^ ^ - -
RRC : Rotate right through carry						
RRC	r		2	6	dst(7)<-C, C<-dst(0) dst(n)<-dst(n+1) n=0-6	^ ^ ^ ^ - -
RRC	R		2	6	" "	^ ^ ^ ^ - -
RRC	(r)		2	6	" "	^ ^ ^ ^ - -
RRC	(R)		2	6	" "	^ ^ ^ ^ - -
RRCW : Rotate word right through carry						
RRCW	rr		2	8	dst(15)<-C, C<-dst(0) dst(n)<-dst(n+1) n=0-14	^ ^ ^ ^ - -
RRCW	RR		2	8	" "	^ ^ ^ ^ - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SBC : Subtraction of 2 bytes with carry						
SBC	r	r	2	6	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	R	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	R	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	r	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(r)	2	6	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	(r)	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(rr)	3	12	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	(rr)	3	12	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	NN	4	18	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	N(rrx)	4	24	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	N(rrx)	4	24	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	NN(rrx)	5	26	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	NN(rrx)	5	26	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	rr(rrx)	3	22	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	(rr)+	3	16	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	R	(rr)+	3	16	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	r	-(rr)	3	16	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	-(rr)	3	16	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(r)	r	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(r)	R	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	r	3	18	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	R	3	18	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)+	r	3	22	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	(rr)+	R	3	22	dst<-dst-src-C rr<-rr+1	^ ^ ^ ^ 1 ^
SBC	NN	r	4	20	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	N(rrx)	r	4	26	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	N(rrx)	R	4	26	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN(rrx)	r	5	28	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN(rrx)	R	5	28	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	rr(rrx)	r	3	24	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	-(rr)	r	3	22	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	-(rr)	R	3	22	rr<-rr-1 dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	r	#N	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	R	#N	3	10	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	#N	3	16	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	NN	#N	5	24	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(rr)	(rr)	3	20	dst<-dst-src-C	^ ^ ^ ^ 1 ^
SBC	(RR)	(rr)	3	20	dst<-dst-src-C	^ ^ ^ ^ 1 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SBCW : Subtract word with carry						
SBCW	rr	rr	2	10	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	RR	3	12	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	RR	3	12	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	rr	3	12	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(r)	3	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	(r)	3	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(rr)	2	16	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	(rr)	3	18	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	NN	4	22	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	N(rrx)	4	28	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	N(rrx)	4	28	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	NN(rrx)	5	30	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	NN(rrx)	5	30	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	rr(rrx)	3	26	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	(rr)+	3	22	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	RR	(rr)+	3	22	dst<-dst+src+C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(r)	rr	3	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(r)	RR	3	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	rr	2	30	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	RR	3	30	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)+	rr	3	32	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	(rr)+	RR	3	32	dst<-dst-src-C rr<-rr+2	^ ^ ^ ^ ? ?
SBCW	NN	rr	4	32	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	rr	4	38	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	RR	4	38	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	rr	5	38	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	RR	5	38	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr(rrx)	rr	3	34	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	rr	#NN	4	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	RR	#NN	4	14	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	#NN	4	32	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN	#NN	6	36	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	N(rrx)	#NN	5	36	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	NN(rrx)	#NN	6	38	dst<-dst-src-C	^ ^ ^ ^ ? ?
SBCW	(rr)	(rr)	2	32	dst<-dst-src-C	^ ^ ^ ^ ? ?

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SCF : Set carry flag						
SCF			1	6	C <- 1 1 - - - - -	
SDM : Set data memory						
SDM			1	6	Set Data Memory DP<-1 Note 1	- - - - - -
SLA : Shift left arithmetic						
SLA	r		2	6	dst C<-dst(7), dst (0)<-0 dst(n+1)<-dst(n)n=0-6	^ ^ ^ ^ 0 -
	R		3	10	" "	^ ^ ^ ^ 0 -
	(rr)		3	20	" "	^ ^ ^ ^ 0 -
SLAW : Shift word left arithmetic						
SLAW	rr		2	10	C<-dst(15), dst (0)<-0 dst(n+1)<-dst(n)n=1-14	^ ^ ^ ^ - -
	RR		3	12	" "	^ ^ ^ ^ - -
	(rr)		2	32	" "	^ ^ ^ ^ - -
SPM : Set program memory						
SPM			1	6	Set Program Memory DP<-0 Note 2	- - - - - -
SPP : Set page pointer						
SPP		#N	2	6	Set Page Pointer - - - - - -	
SRA : Shift right arithmetic						
SRA	r		2	6	dst(7)<-dst(7), C<-dst(0) dst(n)<-dst(n+1)n=0-6	^ ^ ^ ^ 0 ^
SRA	R		2	6	" "	^ ^ ^ ^ 0 ^
SRA	(r)		2	6	" "	^ ^ ^ ^ 0 ^
SRA	(R)		2	6	" "	^ ^ ^ ^ 0 ^
SRAW : Shift word right arithmetic						
SRAW	rr		2	6	dst(15)<-dst(15), C<-dst(0) dst(n)<-dst(n+1)n=0-14	^ ^ ^ 0 - -
SRAW	RR		2	8	" "	^ ^ ^ 0 - -

Notes:
1 Following this instruction, all addressing modes referring to address spaces will refer to the Data Space.
2 Following this instruction, all addressing modes referring to address spaces will refer to the Program Space, except for the following instructions which operate with Dataspace independently of the setting of the DP flag :
PUSH(W)/PUSHU(W), POP(W)/POPU(W), PEA/PEAU, and CALL, RET, IRET and interrupt execution
(assuming the Stack Pointers are not pointing to the Register File).

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SRP : Set register pointer						
SRP		#N	2	6	Set Register Pointer	- - - - - -
SRP0 : Set register pointer 0						
SRP0		#N	2	6	Set Register Pointer 0	- - - - - -
SRP1 : Set register pointer 1						
SRP1		#N	2	6	Set Register Pointer 1	- - - - - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SUB : Subtraction of 2 bytes without carry						
SUB	r	r	2	6	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	R	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	R	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	r	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(r)	2	6	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	(r)	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(rr)	3	12	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	(rr)	3	12	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	NN	4	18	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	N(rrx)	4	24	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	N(rrx)	4	24	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	NN(rrx)	5	26	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	NN(rrx)	5	26	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	rr(rrx)	3	22	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	(rr)+	3	16	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	R	(rr)+	3	16	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	r	-(rr)	3	16	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	-(rr)	3	16	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(r)	r	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(r)	R	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	r	3	18	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	R	3	18	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)+	r	3	22	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	(rr)+	R	3	22	dst<-dst-src rr<-rr+1	^ ^ ^ ^ 1 ^
SUB	NN	r	4	20	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	N(rrx)	r	4	26	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	N(rrx)	R	4	26	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN(rrx)	r	5	28	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN(rrx)	R	5	28	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	rr(rrx)	r	3	24	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	-(rr)	r	3	22	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	-(rr)	R	3	22	rr<-rr-1 dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	r	#N	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	R	#N	3	10	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	#N	3	16	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	NN	#N	5	24	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(rr)	(rr)	3	20	dst<-dst-src	^ ^ ^ ^ 1 ^
SUB	(RR)	(rr)	3	20	dst<-dst-src	^ ^ ^ ^ 1 ^

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
SUBW : Subtract words						
SUBW	rr	rr	2	10	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	RR	3	12	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	RR	3	12	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	rr	3	12	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(r)	3	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	(r)	3	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(rr)	2	16	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	(rr)	3	18	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	NN	4	22	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	N(rrx)	4	28	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	N(rrx)	4	28	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	NN(rrx)	5	30	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	NN(rrx)	5	30	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	rr(rrx)	3	26	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	(rr)+	3	22	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	RR	(rr)+	3	22	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(r)	rr	3	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(r)	RR	3	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	rr	2	30	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	RR	3	30	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)+	rr	3	32	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	(rr)+	RR	3	32	dst<-dst-src rr<-rr+2	^ ^ ^ ^ ? ?
SUBW	NN	rr	4	32	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	rr	4	38	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	RR	4	38	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	rr	5	38	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	RR	5	38	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr(rrx)	rr	3	34	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	rr	#NN	4	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	RR	#NN	4	14	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	#NN	4	32	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN	#NN	6	36	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	N(rrx)	#NN	5	36	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	NN(rrx)	#NN	6	38	dst<-dst-src	^ ^ ^ ^ ? ?
SUBW	(rr)	(rr)	2	32	dst<-dst-src	^ ^ ^ ^ ? ?
SWAP : Swap nibbles						
SWAP	r		2	8	dst(0-3)<—->dst(4-7)	? ^ ^ ? - -
SWAP	R		2	8	dst(0-3)<—->dst(4-7)	? ^ ^ ? - -
SWAP	(r)		2	8	dst(0-3)<—->dst(4-7)	? ^ ^ ? - -
SWAP	(R)		2	8	dst(0-3)<—->dst(4-7)	? ^ ^ ? - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
TCM : Test and complement byte under mask						
TCM	r	r	2	6	NOT dst AND src	- ^ ^ 0 - -
TCM	R	R	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	r	R	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	R	r	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(r)	2	6	NOT dst AND src	- ^ ^ 0 - -
TCM	R	(r)	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(rr)	3	12	NOT dst AND src	- ^ ^ 0 - -
TCM	R	(rr)	3	12	NOT dst AND src	- ^ ^ 0 - -
TCM	r	NN	4	18	NOT dst AND src	- ^ ^ 0 - -
TCM	r	N(rrx)	4	24	NOT dst AND src	- ^ ^ 0 - -
TCM	R	N(rrx)	4	24	NOT dst AND src	- ^ ^ 0 - -
TCM	r	NN(rrx)	5	26	NOT dst AND src	- ^ ^ 0 - -
TCM	R	NN(rrx)	5	26	NOT dst AND src	- ^ ^ 0 - -
TCM	r	rr(rrx)	3	22	NOT dst AND src	- ^ ^ 0 - -
TCM	r	(rr)+	3	16	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	R	(rr)+	3	16	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	r	-(rr)	3	16	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	R	-(rr)	3	16	rr<-rr-1 NOT dst AND src	- ^ ^ 0 - -
TCM	(r)	r	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	(r)	R	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	r	3	18	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	R	3	18	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)+	r	3	22	NOT dst AND src rr<-rr+1	- ^ ^ 0 - -
TCM	(rr)+	R	3	22	dst<-ds AND src rr<-rr+1	- ^ ^ 0 - -
TCM	NN	r	4	20	NOT dst AND src	- ^ ^ 0 - -
TCM	N(rrx)	r	4	26	NOT dst AND src	- ^ ^ 0 - -
TCM	N(rrx)	R	4	26	NOT dst AND src	- ^ ^ 0 - -
TCM	NN(rrx)	r	5	28	NOT dst AND src	- ^ ^ 0 - -
TCM	NN(rrx)	R	5	28	NOT dst AND src	- ^ ^ 0 - -
TCM	rr(rrx)	r	3	24	NOT dst AND src	- ^ ^ 0 - -
TCM	-(rr)	r	3	22	NOT dst AND src rr<-rr-1	- ^ ^ 0 - -
TCM	-(rr)	R	3	22	NOT dst AND src rr<-rr-1	- ^ ^ 0 - -
TCM	r	#N	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	R	#N	3	10	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	#N	3	16	NOT dst AND src	- ^ ^ 0 - -
TCM	NN	#N	5	22	NOT dst AND src	- ^ ^ 0 - -
TCM	(rr)	(rr)	3	18	NOT dst AND src	- ^ ^ 0 - -
TCM	(RR)	(rr)	3	18	NOT dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
TCMW : Test and complement word under mask						
TCMW	rr	rr	2	10	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	RR	3	12	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	RR	3	12	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	rr	3	12	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(r)	3	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	(r)	3	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(rr)	2	16	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	(rr)	3	18	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	NN	4	22	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	N(rrx)	4	28	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	N(rrx)	4	28	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	NN(rrx)	5	30	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	NN(rrx)	5	30	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	rr(rrx)	3	26	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	(rr)+	3	22	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	RR	(rr)+	3	22	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	rr	-(rr)	3	24	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	-(rr)	3	24	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	(r)	rr	3	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	(r)	RR	3	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	rr	2	30	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	RR	3	28	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)+	rr	3	30	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	(rr)+	RR	3	30	NOT dst AND src rr<-rr+2	- ^ ^ 0 - -
TCMW	NN	rr	4	30	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	rr	4	36	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	RR	4	36	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	rr	5	36	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	RR	5	36	NOT dst AND src	- ^ ^ 0 - -
TCMW	rr(rrx)	rr	3	32	NOT dst AND src	- ^ ^ 0 - -
TCMW	-(rr)	rr	3	30	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	-(rr)	RR	3	30	rr<-rr-2 NOT dst AND src	- ^ ^ 0 - -
TCMW	rr	#NN	4	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	RR	#NN	4	14	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	#NN	4	30	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN	#NN	6	34	NOT dst AND src	- ^ ^ 0 - -
TCMW	N(rrx)	#NN	5	34	NOT dst AND src	- ^ ^ 0 - -
TCMW	NN(rrx)	#NN	6	36	NOT dst AND src	- ^ ^ 0 - -
TCMW	(rr)	(rr)	2	32	NOT dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
TM : Test byte under mask						
TM	r	r	2	6	dst AND src	- ^ ^ 0 - -
TM	R	R	3	10	dst AND src	- ^ ^ 0 - -
TM	r	R	3	10	dst AND src	- ^ ^ 0 - -
TM	R	r	3	10	dst AND src	- ^ ^ 0 - -
TM	r	(r)	2	6	dst AND src	- ^ ^ 0 - -
TM	R	(r)	3	10	dst AND src	- ^ ^ 0 - -
TM	r	(rr)	3	12	dst AND src	- ^ ^ 0 - -
TM	R	(rr)	3	12	dst AND src	- ^ ^ 0 - -
TM	r	NN	4	18	dst AND src	- ^ ^ 0 - -
TM	r	N(rrx)	4	24	dst AND src	- ^ ^ 0 - -
TM	R	N(rrx)	4	24	dst AND src	- ^ ^ 0 - -
TM	r	NN(rrx)	5	26	dst AND src	- ^ ^ 0 - -
TM	R	NN(rrx)	5	26	dst AND src	- ^ ^ 0 - -
TM	r	rr(rrx)	3	22	dst AND src	- ^ ^ 0 - -
TM	r	(rr)+	3	16	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	R	(rr)+	3	16	dst AND -src rr<-rr+1	- ^ ^ 0 - -
TM	r	-(rr)	3	16	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	R	-(rr)	3	16	rr<-rr-1 dst AND src	- ^ ^ 0 - -
TM	(r)	r	3	10	dst AND src	- ^ ^ 0 - -
TM	(r)	R	3	10	dst AND src	- ^ ^ 0 - -
TM	(rr)	r	3	18	dst AND src	- ^ ^ 0 - -
TM	(rr)	R	3	18	dst AND src	- ^ ^ 0 - -
TM	(rr)+	r	3	22	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	(rr)+	R	3	22	dst AND src rr<-rr+1	- ^ ^ 0 - -
TM	NN	r	4	20	dst AND src	- ^ ^ 0 - -
TM	N(rrx)	r	4	26	dst AND src	- ^ ^ 0 - -
TM	N(rrx)	R	4	26	dst AND src	- ^ ^ 0 - -
TM	NN(rrx)	r	5	28	dst AND src	- ^ ^ 0 - -
TM	NN(rrx)	R	5	28	dst AND src	- ^ ^ 0 - -
TM	rr(rrx)	r	3	24	dst AND src	- ^ ^ 0 - -
TM	-(rr)	r	3	22	rr->rr-1 dst AND src	- ^ ^ 0 - -
TM	-(rr)	R	3	22	rr->rr-1 dst AND src	- ^ ^ 0 - -
TM	r	#N	3	10	dst AND src	- ^ ^ 0 - -
TM	R	#N	3	10	dst AND src	- ^ ^ 0 - -
TM	(rr)	#N	3	16	dst AND src	- ^ ^ 0 - -
TM	NN	#N	5	22	dst AND src	- ^ ^ 0 - -
TM	(rr)	(rr)	3	18	dst AND src	- ^ ^ 0 - -
TM	(RR)	(rr)	3	18	dst AND src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
TMW : Test word under mask						
TMW	rr	rr	2	10	dst AND src	- ^ ^ 0 - -
TMW	RR	RR	3	12	dst AND src	- ^ ^ 0 - -
TMW	rr	RR	3	12	dst AND src	- ^ ^ 0 - -
TMW	RR	rr	3	12	dst AND src	- ^ ^ 0 - -
TMW	rr	(r)	3	14	dst AND src	- ^ ^ 0 - -
TMW	RR	(r)	3	14	dst AND src	- ^ ^ 0 - -
TMW	rr	(rr)	2	16	dst AND src	- ^ ^ 0 - -
TMW	RR	(rr)	3	18	dst AND src	- ^ ^ 0 - -
TMW	rr	NN	4	22	dst AND src	- ^ ^ 0 - -
TMW	rr	N(rrx)	4	28	dst AND src	- ^ ^ 0 - -
TMW	RR	N(rrx)	4	28	dst AND src	- ^ ^ 0 - -
TMW	rr	NN(rrx)	5	30	dst AND src	- ^ ^ 0 - -
TMW	RR	NN(rrx)	5	30	dst AND src	- ^ ^ 0 - -
TMW	rr	rr(rrx)	3	26	dst AND src	- ^ ^ 0 - -
TMW	rr	(rr)+	3	22	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	RR	(rr)+	3	22	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	rr	-(rr)	3	24	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	RR	-(rr)	3	24	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	(r)	rr	3	14	dst AND src	- ^ ^ 0 - -
TMW	(r)	RR	3	14	dst AND src	- ^ ^ 0 - -
TMW	(rr)	rr	2	28	dst AND src	- ^ ^ 0 - -
TMW	(rr)	RR	3	28	dst AND src	- ^ ^ 0 - -
TMW	(rr)+	rr	3	30	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	(rr)+	RR	3	30	dst AND src rr<-rr+2	- ^ ^ 0 - -
TMW	NN	rr	4	30	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	rr	4	36	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	RR	4	36	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	rr	5	36	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	RR	5	36	dst AND src	- ^ ^ 0 - -
TMW	rr(rrx)	rr	3	32	dst AND src	- ^ ^ 0 - -
TMW	-(rr)	rr	3	30	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	-(rr)	RR	3	30	rr<-rr-2 dst AND src	- ^ ^ 0 - -
TMW	rr	#NN	4	14	dst AND src	- ^ ^ 0 - -
TMW	RR	#NN	4	14	dst AND src	- ^ ^ 0 - -
TMW	(rr)	#NN	4	30	dst AND src	- ^ ^ 0 - -
TMW	NN	#NN	6	34	dst AND src	- ^ ^ 0 - -
TMW	N(rrx)	#NN	5	34	dst AND src	- ^ ^ 0 - -
TMW	NN(rrx)	#NN	6	36	dst AND src	- ^ ^ 0 - -
TMW	(rr)	(rr)	2	32	dst AND src	- ^ ^ 0 - -
WFI : Wait for Interrupt						
WFI		2 18	wait for interrupt		- - - - - -	

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
XCH : Exchange Register						
XCH	r	r	3	12	dst <-> src	- - - - - -
XCH	R	R	3	12	dst <-> src	- - - - - -
XCH	r	r	3	12	dst <-> src	- - - - - -
XCH	R	R	3	12	dst <-> src	- - - - - -
XOR : Logical exclusive OR						
XOR	r	r	2	6	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	R	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	R	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	r	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(r)	2	6	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	(r)	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(rr)	3	12	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	(rr)	3	12	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	NN	4	18	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	N(rrx)	4	24	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	N(rrx)	4	24	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	NN(rrx)	5	26	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	NN(rrx)	5	26	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	rr(rrx)	3	22	dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	(rr)+	3	16	dst<-dst XOR src rr->rr+1	- ^ ^ 0 - -
XOR	R	(rr)+	3	16	dst<-dst XOR src rr->rr+1	- ^ ^ 0 - -
XOR	r	-(rr)	3	16	rr->rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	-(rr)	3	16	rr->rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	(r)	r	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(r)	R	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	r	3	18	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	R	3	18	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)+	r	3	22	dst<-dst XOR src rr->rr+1	- ^ ^ 0 - -
XOR	(rr)+	R	3	22	dst<-dst XOR src rr->rr+1	- ^ ^ 0 - -
XOR	NN	r	4	20	dst<-dst XOR src	- ^ ^ 0 - -
XOR	N(rrx)	r	4	26	dst<-dst XOR src	- ^ ^ 0 - -
XOR	N(rrx)	R	4	26	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN(rrx)	r	5	28	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN(rrx)	R	5	28	dst<-dst XOR src	- ^ ^ 0 - -
XOR	rr(rrx)	r	3	24	dst<-dst XOR src	- ^ ^ 0 - -
XOR	-(rr)	r	3	22	rr->rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	-(rr)	R	3	22	rr->rr-1 dst<-dst XOR src	- ^ ^ 0 - -
XOR	r	#N	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	R	#N	3	10	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	#N	3	16	dst<-dst XOR src	- ^ ^ 0 - -
XOR	NN	#N	5	24	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(rr)	(rr)	3	20	dst<-dst XOR src	- ^ ^ 0 - -
XOR	(RR)	(rr)	3	20	dst<-dst XOR src	- ^ ^ 0 - -

INSTRUCTION SUMMARY (Continued)

Mnemo.	dst	src	Bytes	Clock cycles	Operation	Flags C Z S V D H
XORW : Logical exclusive OR between words						
XORW	rr	rr	2	10	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	RR	3	12	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	RR	3	12	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	rr	3	12	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(r)	3	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	(r)	3	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(rr)	2	16	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	(rr)	3	18	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	NN	4	22	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	N(rrx)	4	28	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	N(rrx)	4	28	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	NN(rrx)	5	30	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	NN(rrx)	5	30	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	rr(rrx)	3	26	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	(rr)+	3	22	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	RR	(rr)+	3	22	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	rr	-(rr)	3	24	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	-(rr)	3	24	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	(r)	rr	3	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(r)	RR	3	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	rr	2	30	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	RR	3	30	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)+	rr	3	32	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	(rr)+	RR	3	32	dst<-dst XOR src rr<-rr+2	- ^ ^ 0 - -
XORW	NN	rr	4	32	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	rr	4	38	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	RR	4	38	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	rr	5	38	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	RR	5	38	dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr(rrx)	rr	3	34	dst<-dst XOR src	- ^ ^ 0 - -
XORW	-(rr)	rr	3	32	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	-(rr)	RR	3	32	rr<-rr-2 dst<-dst XOR src	- ^ ^ 0 - -
XORW	rr	#NN	4	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	RR	#NN	4	14	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	#NN	4	32	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN	#NN	6	36	dst<-dst XOR src	- ^ ^ 0 - -
XORW	N(rrx)	#NN	5	36	dst<-dst XOR src	- ^ ^ 0 - -
XORW	NN(rrx)	#NN	6	38	dst<-dst XOR src	- ^ ^ 0 - -
XORW	(rr)	(rr)	2	32	dst<-dst XOR src	- ^ ^ 0 - -