## STL INSTRUCTIONS

Version 2.00 (August 1992)

Published by: Bernecker und Rainer Industrie-Elektronik GmbH

Model No.: MAAWLKB-E

The information contained in this booklet is believed to be accurate as of the date of publication, however, Bernecker und Rainer Industrie-Elektronik GmbH will not be liable for any damages, including indirect or consequential, or reliance upon the accuracy of this documentation. The information contained herein is subject to change without notice.

## **CONTENTS**

Syntax Explanation	9	4. Mathematic Routines	179
1.1. CPU Type	9	4.1. General	179
1.2. Page Layout	9	4.2. Basic Calculations	
1.3. Information	10	MADD - Addition in Floating Point Format	183
1.3.1. Processor	10	MSUB - Subtraction in Floating Point Format	183
1.3.2. Mnemonic (Command abbreviation)	10	MMUL - Multiplication in Floating Point Format	184
1.3.3. Function	11	MDIV - Division in Floating Point Format	184
1.3.4. Operating Mode	11	· · · · · · · · · · · · · · · · · · ·	
1.3.5. Addressing Modes / Opcode	12	MSQR - Square Root in Floating Point Format	185
1.3.6. Address Preselection	13	4.3. Operand Manipulation	
1.3.7. Condition Code Register (CCR)	14	MSGN - Change Sign of Operand 1	185
1.4. Detailed Description	15	MCOP - Copy Operand OP1 to Operand OP2	186
2. Addressing Modes	17	MEXG - Exchange Operands OP1 and OP2 with each other	186
2.1. IMPL Implied Addressing	17	4.4. Operand Loading	
2.2. DIR Direct Addressing	19	LAL1 - Load Operand OP1 with Number (Absolute long)	187
2.3. EXT Extended Addressing	21	LAL2 - Load Operand OP2 with Number (Absolute long)	187
2.4. IMMED Immediate Addressing	23	LAW1 - Load Operand OP1 with Number (Absolute word)	188
2.5. IND Indexed Addressing	25	LAW2 - Load Operand OP2 with Number (Absolute word)	188
2.6. REL Relative Addressing	29	LIL1 - Load Operand OP1 with Number (Integer long)	189
2.7. Indirect Addressing	30 32	LIL2 - Load Operand OP2 with Number (Integer long)	189
2.7. Negation 3. STL Instructions	33	LIW1 - Load Operand OP1 with Number (Integer word)	190
3.1. Load Instructions	33	LIW2 - Load Operand OP2 with Number (Integer word)	190
3.2. Store Instructions	51	LF1 - Load Operand OP1 with Number (IEEE format)	191
3.3. Data Exchange between Registers	59	LF2 - Load Operand OP2 with Number (IEEE format)	191
3.4. Stack Instructions	69	4.5. Operand Saving	131
3.5. Logical Functions	81	SAL - Store Operand OP1 in Absolute Long Format	192
3.6. Arithmetic Instructions	91		192
3.7. Test and Comparison Instructions	107	SAW - Store Operand OP1 in Absolute Word Format	
3.8. Increment and Decrement Instructions	119	SIL - Store Operand OP1 in Integer Long Format	193
3.9. Shift and Rotate Instructions	131	SIW - Store Operand OP1 in Integer Word Format	193
3.10. Branch Instructions	147	SFX - Store Operand OP1 in IEEE Format	194
3.11 Miscellaneous Instructions	163		

4.6. Operand Temporary Memory		4.10. Utilities	
SFM1 - Store Operand OP1 in Memory 1	195	FCOP - Function Copy	226
SFM2 - Store Operand OP1 in Memory 2	195	FSMB - Function Set Memory Byte	227
SFM3 - Store Operand OP1 in Memory 3	195	FSMW - Function Set Memory Word	228
RFM1 - Load Operand OP2 from Memory 1	196	FCLR - Function Clear Memory	229
RFM2 - Load Operand OP2 from Memory 2	196	MCMP - Compare OP1 and OP2	230
RFM3 - Load Operand OP2 from Memory 3	196	MHIL - Limitation to High Limit;	
4.7. Quick Multiplication		if OP1 > OP2, then OP2 ⇒ OP1	23′
FM2B - 2 Byte X 2 Byte Multiplication	197	MLOL - Limitation to Low Limit;	
FM3B - 3 Byte X 2 Byte Multiplication	197	if OP1 < OP2, then OP2 ⇒ OP1	232
FM4B - 4 Byte X 2 Byte Multiplication	198	5. Special Instructions	233
4.8. Number Conversion		5.1. Arithmetic Processor (only CP80)	233
CAF - Convert ASCII String to Floating Point (IEEE) Format	199	5.1.1. Operands	233
CFA - Convert OP1 to ASCII (without leading zeros)	200	5.1.2. APU Instruction Calls	234
CFA0 - Convert OP1 to ASCII (with leading zeros)	202	5.1.3. Number Formats	235
CFEA - Convert OP1 to ASCII String with Exponent Format	204	5.1.4. 68881 APU Instruction Set	236
CIA - Convert Integer to ASCII (without leading zeros)	206	5.1.5. Constants	238
CIA0 - Convert Integer to ASCII (with leading zeros)	208	5.2. Interface Operation (only CP80, PP60 and NTC)	
CBCD - Convert Binary to BCD	210	5.2.1. SOB - Output Character	24′
CBIN - Convert BCD to Binary	211	5.2.2. SIB - Read Character	242
4.9. Conversion Routines		5.2.3. SC - Request Interface Status	243
General Binary <=> Physical	212	5.2.4. SF - Interface Functions (e.g. Initialization)	244
CBPP - Calculation Factors k and d from		5.2.5. Handshake Techniques	255
two Straight Line Points		6. Appendix	259
CBPQ - Convert Binary => Physical, Quick	216	6.1. Alphabetical Overview of B&R Mnemonics	259
CPBQ - Convert Physical => Binary, Quick	217	6.2. Alphabetical Overview of Motorola Mnemonics	260
CBP - Convert Binary => Physical	218	6.3. Alphabetical Overview of Mathematic Routines	26′
CPB - Convert Physical => Binary	220	6.4. Alphabetical Overview of B&R Short Mnemonics	
CIM - Convert Inch => Millimeter	222	6.5. Index	262
CMI - Convert Millimeter => Inch	224		

# Register

Interface Operation (only CP80, PP60 and NTCP6#)	Special Instructions Arithmetic					Operar				Mathematic Routines	Miso	Jump a	Shift e	Increment and [	Test and C				Data Exchar		STL Instructions		
CP80, PP60 and NTCP6#)	Arithmetic Processor (only CP80)	Utilities	Conversion Routines	Number Conversion	Quick Multiplication	Operand Temporary Memory	Operand Saving	Operand Loading	Operand Manipulation	Basic Calculations	Miscellaneous Instructions	Jump and Branch Instructions	Shift and Rotate Instructions	Increment and Decrement Instructions	Test and Comparison Instructions	Arithmetic Instructions	Logical Functions	Stack Instructions	Exchange between Registers	Store Instructions	Load Instructions	Addressing Modes	Syntax Explanation

#### 1. SYNTAX EXPLANATION

#### 1.1. CPUTYPE

There are two different CPU types depending on the processor which is used.

Processor	CPU TYPE	Module
6303	Α	CP40, CP41, NTCP33, NTCP34, PSCP35, PP40,
6809	В	CP30, CP31, CP32 CP60, CP80, NTCP63, NTCP64, PSCP65, PP60

#### 1.2. PAGE LAYOUT

In writing the STL commands description the differentiation between the two types of CPU has been taken into consideration for the simple reason that some commands were written once for CPU type A and once for CPU type B. Therefore the following pages have been divided into two halves for easy reading:

Left side: 6303 commands description (CPU type A)
 Right side: 6809 commands description (CPU type B)

#### 1.3. INFORMATION

For every command an information table is given which looks like this:

Mot	orola			Fun	ction												C	CP	U	ty	/p	e A	Α/	6	30	3
B&F	ł																									
Sho	rt																									
Δ,	ddroes	ina n	ode /	/ Opcode Address preselection PG1								<b>31</b> 0	1000													
	au 1 6 3 3	ing ii	ioue /	Орсос	10	L		^'	uu		30	, ,	,,,	.31	-10			"					Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Γ		CC	R		П
						Ε	Α	М	F	z	#	Р	С	1	Υ	D	U	!	В	G	н	ī	N	z	٧	С
						Г	Г	Г	Г		Г	Г	Г	П					П		Г	Г	Г			П

Mote	orola			Fun	Function CPU type B / 6809																					
B&R	1																									
Sho	rt																									
۸	Addressing mode / Opcode Address preselection CP 80																									
	iui ess	ing in	ioue /	Орсос	16			~'	uu	16	33	, ,	,, e	.51	316	-		""					Р	CE	0	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	T	N	Z	٧	С
						Γ	Γ	Γ				Г	Γ													

#### 1.3.1. Processor

The CPU type which applies to the description given is shown in the top right hand corner of the table. **CPU TYPE A** is always shown on the left hand side and **CPU TYPE B** is on the right. If one side is shaded **gray** the description that follows only concerns that CPU type (in this case - CPU type B).

## 1.3.2. Mnemonic (Command abbreviation)

There are three ways a command can be written, corresponding to three mnemonics, which can be configured for use with the PROgramming SYStem:

1) Motorola: MOTOROLA® Mnemonics

2) B&R: B&R Mnemonics

3) Short: For some B&R mnemonics shorter abbreviations have been developed to enable quicker STL command entries. When

one of these short forms has been entered the corresponding B&R mnemonic (in B&R mode) or the corresponding

MOTOROLA® mnemonic (in MIXM mode) is automatically loaded.

#### 1.3.3. Function

Commands are represented by symbolic characters in this field. The symbols and characters used are as follows:

Accumulator A (8 Bit) Α 8 bit register (A, B, CCR, DP) В Accumulator B (8 Bit) 16 bit register (D. X. Y. SP!, SPU) D Accumulator D (16 Bit; A = HOB, B = LOB) IRQ Mask CCR Condition code register (8 Bit) Logical AND Λ DΡ Register for direct addressing (8 Bit) Logical OR (Direct Page Register) Logical exclusive-or (EXOR) Χ Index register X (16 Bit) HOB High order byte Υ Index register Y (16 Bit) LOB Low order byte SPI System stack pointer (16 Bit) FΑ Effective address SPU User stack pointer (16 Bit) IMM Immediate value (constant) PC Program counter Data bit n of a register or position in memory dn M Address of a memory location (M) Contents of the memory location adressed by M (8 Bit)

#### 1.3.4. Operating Mode

The symbol "O" shows that the displayed command can be used in the PROgramming SYStem in this operating mode.

Example:



M and M+1 (16 Bit)

(M:M+1) Contents of both memory locations addressed by

or



#### 1.3.5. Addressing modes / Opcode

In the addressing mode which is possible for a certain command, two values are given and separated by "/".

#### Example: 4/2

- The first value is the execution time for a command given in machine cycles.
- The second value is the *length* of the command in *bytes*.

A "+" can be found next to this value (e.g.: 4+/2+). In this case the meaning of the values are altered to: the execution time for the command is **four or more** machine cycles and the command is **two or more** bytes long.

A command's opcode is shown beneath in the table the command length and machine cycle values. If the opcode is longer than one byte it is connected by a blank character.

The following addressing modes are available:

IMPI	L. 1	lm	aı	li	е	d

The instruction needs no additional parameters. The opcode itself includes all necessary address information

#### DIR.1) Direct

The least significant byte of the address is given in addition to the instruction. The most significant byte is defined by the DP register (Direct Page register).

#### EXT. Absolute (Indirect Absolute 2)

The entire address (2 bytes) is given in addition to the instruction

#### IMMED. Immediate (Constant)

The effective address of the data is the location immediately following the opcode (i.e. the data to be used in the instruction immediately follows the opcode of the instruction).

#### IND. Indexed (Indirect Indexed 2)

With indexed addressing the effective address is calculated by adding the contents of the index register  $(X, Y^0, SP^{10}, SPU^0)$  with the offset which is given with the instruction.

#### REL. Relative

Relative addressing is used with conditional branches. The destination address is calculated by adding the actual PC (Program counter) and the entered 1 or 2<sup>1)</sup> byte offset.

- ocan only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode
- Indirect: For addressing modes EXT. and IND. there are additional possibilities for *indirect* addressing. The effective address which refers to the instruction is found at the indicated address (can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode).

## 1.3.6. Address preselection

Several address preselections are possible:

MOTOROLA	B&R	
I O F S T # P R X Y U ! D G в	E A M F Z # P C I Y U ! D G B	Input Output Flag Start timer Timer (time elapsed flag) Constant Peripheral address Register Index register X Index register Y¹) User stack pointer¹) Direct addressing¹) Global RAM in PP60 (extended dual port RAM)¹) Block memory in PP60¹)
		* **

The following symbols will be used to explain:

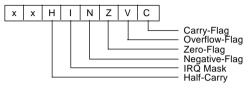
- O can be used in the specified operating mode
- can be used with PROgramming SYStem Version 5.00 in PC 80 operating mode

1) can be used with PROgramming SYStem Version 5.00 in PC 80 operating mode

#### 1.3.7. Condition Code Register (CCR)

Condition code register bits are changed according to the outcome of an instruction. The state of the condition code register influences e.g. conditional branches (depending on the condition code register a branch is executed or the next command is processed).

The condition code register format:



Carry-Flag: This bit is set to a one when the preceding

calculation creates a carry in the case of addition or

a borrow with subtraction i.e. the result is greater or smaller than the operation register can handle.

Overflow-Flag: This bit is set to a one by an operation which causes a signed two's complement arithmetic overflow i.e.

the result is outside of the range from -128 to +127 (with a 1 byte value) or outside of the range from -

32768 to +32767 (with a 2 byte value).

Zero-Flag: This bit is set to a one if the result of the previous

operation was identically zero.

Negative-Flag: This bit contains exactly the value of the MSB of the

result of the preceding operation.

IRQ Mask: If this bit is set to a one the processor will not

recognize interrupts from the IRQ line. All processor

interrupts are locked.

Half-Carry:

Bit 6 and 7:

This bit is set to a one and is used to indicate a carry from bit 3 to bit 4 as a result of an 8 bit addition (only ADCA, ADCB, ADDA and ADDB). This bit is used to

perform a decimal add adjust operation.

In the 6303 these bits are always set to one and are not used.

These bits should **not** be changed by the user in the 68091

The following symbols are used:

Flag is not influenced

Flag is changed according to the operation

Flag is set to a one

Flag is set to a zero

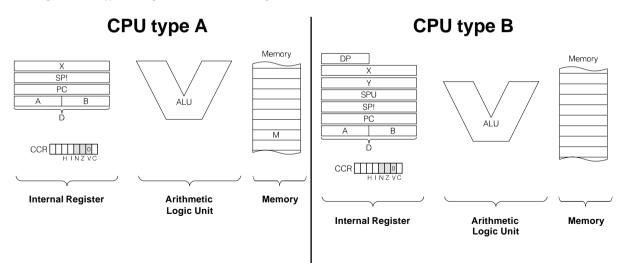
State of the flag is not defined

#### 1.4. Detailed Description

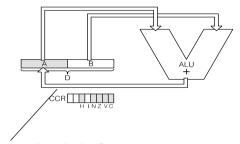
After the information table for each command the following is given:

- A detailed description of the command
- 2) A diagram showing the data flow

According to the CPU type the diagram can have the following elements:



Registers, memory locations or condition code register bits which can be changed with an instruction are **shaded in grey**.



Thick arrow shows the data flow.

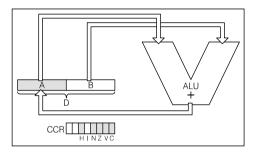
**Small arrow** represents a pointer (the contents of a register point to a memory location).

**Broken line** Register and pointer display the state of an instruction before it is sent.

#### 2. ADDRESSING MODES

#### 2.1. IMPL. - IMPLIED ADDRESSING

The opcode of the instruction contains all required information for executing the instruction. Entering parameters in addition to the instruction is not required. E.g.: ABA (B&R: A+B). Registers A and B are used by the processor. The data flow is as follows:



## **Instruction Overview / Implied Addressing**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ABA	A+B	DEX	DR	PULX	RVS
ABX	B+R	EXG <sup>1)</sup>	EXG <sup>1)</sup>	ROLA	RLA
ASLA	SLA	INCA	IA	ROLB	RLB
ASLB	SLB	INCB	IB	RORA	RRA
ASLD	SLD	INS	IS	RORB	RRB
CBA	AVB	INX	IR	RTS	RET
CLC	CLC	LSRA	SRA	SBA	A-B
CLI	CLI	LSRB	SRB	SEC	SEC
CLRA <sup>2)</sup>	CLA <sup>2)</sup>	LSRD	SRD	SEI	SEI
CLRB <sup>2)</sup>	CLB <sup>2)</sup>	MUL	A*B	TAB	MAB
COMA <sup>2)</sup>	COA <sup>2)</sup>	NOP	NOP	TBA	MBA
COMB <sup>2)</sup>	COB <sup>2)</sup>	PSHA	ANS	TFR <sup>1)</sup>	TFR <sup>1)</sup>
DAA	DK	PSHB	BNS	TPA	MCA
DECA	DA	PSHX	RNS	TSX	MSR
DECB	DB	PULA	AVS	TXS	MRS
DES	DS	PULB	BVS	XGDX	DXR

<sup>1)</sup> can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode

can only be used with PROgramming SYStem Version 5.00 or higher in PG-PC or PC 80 operating modes

#### 2.2. DIR. - DIRECT ADDRESSING

This mode of addressing is comparable with indexed addressing. The address which the instruction uses is set together in two bytes.



Example:

LDAA

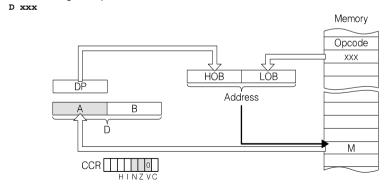
The memory range is broken down into pages of 256 bytes each. A page can be addressed directly with the preselection "D". This is written as follows. E.g.:

LDAA

D
000

D
200

The advantage of this mode of addressing is the shorter execution time of an instruction. This addressing mode is used when frequent and quick access of a certain address range is required.



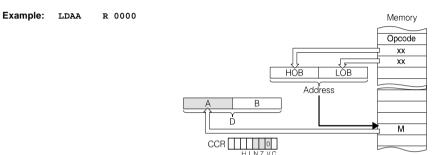
## **Instruction Overview / Direct Addressing**

This addressing mode can only be used with the PROgramming SYStem Version 5.00 or higher in PC 80 operating mode.

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA ADCB ADDA ADDB ADDD ANDA ANDB ASL BITA BITB	ADD ++B + +B +D UND UB SL B BB	CMPA CMPB COM DEC EORA EORB INC LDAA LDAB LDD	CMP VB K DEC EXO EB INC LAD LB LD	ORAB ROL ROR SBCA SBCB STAA STAB STD SUBA SUBB	OB SLI SRE SUB B = B = D
CLR	CLR	ORAA	OD	SUBD	-D

#### 2.3. EXT. - EXTENDED ADDRESSING

In this addressing mode two bytes representing the 16 bit address which is used by the instruction follow the opcode. Address preselections I, O, F, S, T, P, R, B and G can be used with extended addressing.



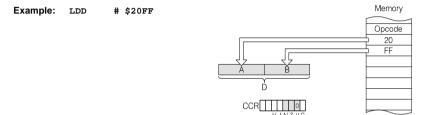
## **Instruction Overview / Extended Addressing**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA	ADD	DEC	DEC	ROL	SLI
ADCB	++B	EORA	EXO	ROR	SRE
ADDA	+	EORB	EB	RST	RST
ADDB	+B	INC	INC	SBCA	SUB
ADDD	+D	JMP	SPI	SBCB	B
ANDA	UND	JSR	SPU	SET	SET
ANDB	UB	LDAA	LAD	STAA	=
ASL	SL	LDAB	LB	STAB	=B
BITA	В	LDD	LD	STD	=D
BITB	BB	LDS	LS	STS	=S
CLR	CLR	LDX	LR	STX	=R
CMPA	CMP	LDY <sup>1)</sup>	LY <sup>1)</sup>	STY1)	=Y1)
CMPB	VB	LSR	SR	SUBA	-
COM	K	ORAA	OD	SUBB	-B
CPX	VR	ORAB	ОВ	SUBD	-D
CPY <sup>1)</sup>	VY <sup>1)</sup>	PRS	PRS		

can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode

#### 2.4. IMMED. - IMMEDIATE ADDRESSING

In this case one or two byte values which are processed by the instruction are given after the instruction. The value is stored in the area after the opcode.



## **Instruction Overview / Immediate Addressing**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA ADCB	ADD ++B	CPX# CPY#¹)	VRK VYK¹)	LDY# <sup>1)</sup> LDYL <sup>1)</sup>	LYK <sup>1)</sup> LYL <sup>1)</sup>
ADDA	+	EIM <sup>2)</sup>	EIM <sup>2)</sup>	OIM <sup>2)</sup>	OIM <sup>2)</sup>
ADDB	+B	EORA	EXO	ORAA	OD
ADDD	+D	EORB	EB	ORAB	OB
AIM <sup>2)</sup>	AIM <sup>2)</sup>	LDAA	LAD	SBCA	SUB
ANDA	UND	LDAB	LB	SBCB	B
ANDB	UB	LDD	LD	SUBA	-
BITA	В	LDK <sup>3)</sup>	LDK <sup>3)</sup>	SUBB	-B
BITB	BB	LDX#	LRK	SUBD	-D
CMPA	CMP	LDXL <sup>3)</sup>	LDL <sup>3)</sup>	TIM <sup>2)</sup>	TIM <sup>2)</sup>
CMPB	VB				

can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode

<sup>2)</sup> can only be used with PROgramming SYStem Version 5.00 or higher in PG-PC operating mode (CPU type A only)

<sup>3)</sup> can only be used with PROgramming SYStem Version 5.00 or higher in PG-PC or PC 80 operating modes

#### 2.5. IND. - INDEXED ADDRESSING

Indexed addressing is possible with the following address preselections: X, Y, U, !

In all indexed addressing, one of the pointer registers (X, Y, SPU, SP!) is used in a calculation of the effective address of the operand to be used by the instruction. Three modes of indexed addressing are possible:

#### 2.5.1. CONSTANT OFFSET INDEXED

Maximum and minimum value of the offset depend on the PROgramming SYStem operating mode:

Operating mode	PG1000	CP 80	PG-PC	PC 80
Offset	X 000 to X 255	X -128 to X 127	X 000 to X 255	X -32768 to X 32767 Y -32768 to Y 32767 U -32768 to U 32767 ! -32768 to ! 32767

In PC 80 mode the length of the instruction depends on the size of the offset:

Offset value	-16 to +15	-128 to -17 +16 to +127	-32768 to -129 +128 to +32767
Instruction length	Opcode + 0 byte	Opcode + 1 byte	Opcode + 2 bytes

The offset value is entered after the instruction and address preselection.

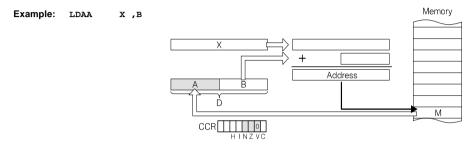
#### Example:

Calculating the effective instruction length for the LDAA instruction in bytes. The description for this instruction shows the instruction length for indexed addressing (IND.) given with 2 +:

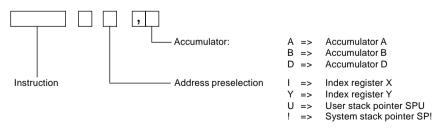
Offset value	-16 to +15	-128 to -17 +16 to +127	-32768 to -129 +128 to +32767
Instruction length	2 + 0 = 2 bytes	2 + 1 = 3 bytes	2 + 2 = 4 bytes

#### 2.5.2. ACCUMULATOR - OFFSET INDEXED

This mode is similar to constant offset indexed except that the two's complement value in one of the accumulators (A, B, or D) and the contents of one of the pointer registers are added to form the effective address of the operand. This addressing mode is only possible in CP 80 operating mode.

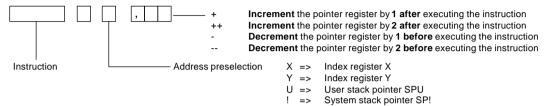


The following can be used for an accumulator offset:



#### 2.5.3. POSTINCREMENT / PREDECREMENT INDEXED

The pointer register can be decremented automatically before executing an instruction or incremented after executing the instruction. This mode of addressing is only possible with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode.



Example: LDAA X ,++

## **Instruction Overview / Indexed Addressing**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA ADCB ADDA	ADD ++B	DEC EORA EORB	DEC EXO EB	LSR ORAA ORAB	SR OD OB
ADDB ADDD	+ +B +D	INC JMP	INC SPI	ROL ROR	SLI SRE
ANDA ANDB	UND UB	JSR LDAA	SPU LAD	SBCA SBCB	SUB B
ASL BITA	SL B	LDAB LDD	LB LD	STAA STAB	= =B
BITB CLR	BB CLR	LDS LDX	LS LR	STD STS	=D =S
CMPA CMPB COM	CMP VB K	LDY <sup>1)</sup> LEA! <sup>1)</sup> LEAU <sup>1)</sup>	LY <sup>1)</sup> LEI <sup>1)</sup> LEU <sup>1)</sup>	STX STY <sup>1)</sup> SUBA	=R =Y <sup>1)</sup>
CPX CPY <sup>1)</sup>	VR VY <sup>()</sup>	LEAX¹) LEAY¹)	LER <sup>1)</sup> LEY <sup>1)</sup>	SUBB SUBD	- -В -D

can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode

#### 2.6. REL. - RELATIVE ADDRESSING

Relative addressing is only used with certain branch instructions. The byte(s) following the branch opcode is (are) traded as a signed offset which may be added to the program counter. Short (1 byte offset) and long (2 byte offset) relative addressing modes are available.

#### Instruction Overview / Relative Addressing

LONG1)

J<=L

SNOL

J-I

J+L

BLSL

BMII

BNFI

**BPLL** 

MOTOROLA	B&R	MOTOROLA	B&R
BCC	JC0	BCCL	JC0L
BEQ	SP0	BEQL	SP0L
BHI	SP>	BHIL	SP>L
BCS	SP<	BCSL	SP <l< th=""></l<>

J<=

SN<sub>0</sub>

J-

J+

SHORT

BLS

BMI

BNF

**BPL** 

can only be used with PROgramming SYStem Version 5.00 and higher in PC 80 operating mode

#### 2.7. INDIRECT ADDRESSING

In addition to extended and indexed addressing the option of using the resulting address as an "address of an address" is available. This means that if using **extended indirect** or **indexed indirect** addressing the effective address to which the instruction applies is found in the memory location that the original address points to. This mode of addressing is distinguished by square cornered brackets "[".

To select indirect addressing the cursor must be positioned in the address preselection field in the STL entry line.

Example: LDAA [ X ,++ Example: LDAA [ R 1000

Indirect addressing is only possible with 2 byte postincrement or predecrement.

## **Instruction Overview / Indirect Addressing**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA ADCB ADDA ADDB ADDD ANDA ANDB ASL BITA	ADD ++B + +B +D UND UB SL B	DEC EORA EORB INC JMP JSR LDAA LDAB LDD	DEC EXO EB INC SPI SPU LAD LB	LSR ORAA ORAB ROL ROR SBCA SBCB STAA STAB	SR OD OB SLI SRE SUB B =
BITA BITB CLR CMPA CMPB COM CPX CPY <sup>1)</sup>	BB CLR CMP VB K VR VY <sup>(1)</sup>	LDS LDX LDY <sup>1)</sup> LEA! <sup>1) 2)</sup> LEAU <sup>1) 2)</sup> LEAX <sup>1) 2)</sup> LEAY <sup>1) 2)</sup>	LS LR LY <sup>1)</sup> LEI <sup>1) 2)</sup> LEU <sup>1) 2)</sup> LER <sup>1) 2)</sup> LEY <sup>1) 2)</sup>	STD STS STX STY <sup>1)</sup> SUBA SUBB SUBD	=D =S =R =Y <sup>1)</sup> - -B

can only be used with PROgramming SYStem Version 5.00 or higher in PC 80 operating mode

<sup>2)</sup> not possible with extended indirect addressing

#### 2.7. NEGATION

Negation is only possible with 1 bit addresses. Negation is possible with the following address preselections:

- I Digital input
- O Digital output
- F Flag
- S Start timer
- T Timer (timer elapsed signal)

To select negation the cursor must be positioned in the address preselection field in the STL entry line.

## **Instruction Overview / Negation**

MOTOROLA	B&R	MOTOROLA	B&R	MOTOROLA	B&R
ADCA	ADD	CPX#	VRK	ORAA	OD
ADCB	++B	CPY#1)	VYK <sup>1)</sup>	ORAB	OB
ADDA	+	DEC	DEC	PRS	PRS
ADDB	+B	EORA	EXO	ROL	SLI
ADDD	+D	EORB	EB	ROR	SRE
ANDA	UND	INC	INC	RST	RST
ANDB	UB	LDAA	LAD	SBCA	SUB
ASL	SL	LDAB	LB	SBCB	B
BITA	В	LDD	LD	SET	SET
BITB	BB	LDK	LDK	STAA	=
CLR	CLR	LDX#	LRK	STAB	=B
CMPA	CMP	LDY'1)	LYK <sup>1)</sup>	STD	=D
CMPB	VB	LSR	SR	SUBA	-
COM	K			SUBB	-B

can only be used with PROgramming SYStem Version 5.00 in PC80 operating mode

## 3. STL INSTRUCTIONS 3.1. LOAD INSTRUCTIONS

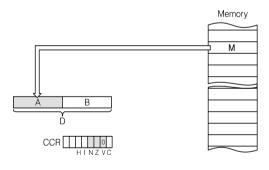
All instructions which load data (1 or 2 bytes) from a given memory position to a register.

Motorola	B&R		Operatin	g mode	
Wiotorola	Dak	PG1000	PG-PC	CP 80	PC 80
LDAA	LAD	0	0	0	0
LDAB	LB	0	0	0	0
LDD	LD	0	0	0	0
LDK	LDK		0		0
LDL	LDL		0		0
LDX	LR	0	0	0	0
LDX#	LRK	0	0	0	0
LDXL	LRL	0	0	0	0
LDY	LY				0
LDY#	LYK				0
LDYL	LYL				0
LDS	LS	0	0	0	0
LEA!	LE!				0
LEAU	LEU				0
LEAX	LER			0	0
LEAY	LEY				0

Mot	orola	LDA	٩A	Fun	ction								(	ЭP	U t	ype	e A	/ 6	30:	3	Mot	orola	LD.	AA	Fur	nction									C	Pι	J ty	/pe	В	/ 68	30
B&F	₹	LA		(M)	⇒ A															1	B&F	₹	LA	D	(M)	⇒ A															
Sho	ort	L																		ı	Sho	rt	L																		
Δ.	ddraes	ina n	node /	Onco	40		Δο	ldra		pre	260	مما	tic	'n		0	F	PG1	000	_		ddrae	eina r	node /	Onco	de	Т	_	dА	ro	_	pre	200	عما	tio	<u>_</u>		0		CP8	
	uuiesa	sing ii	ioue /	Орсо	46		~	uit		Pi,	-30		,,,,	•••		0		PG-	PC		^	uui 63.	anig i	noue,	Орсс	ue			uu		,,	Pic			,,,,	"		0	F	PC 8	30
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ΙO	F	S T	#	P R	х	ΥD	U	!	В	Г	(	CCR	1		IMPL.	DIR.	EXT.	IMMED	. IND.	REL.	ı	O F	s	Т	# F	P R	х	Y D	U	! E	3 G		С	CR	
	3/2	4/3	2/2	4/2		ΕA	М	F Z	#	РС	1	Y D	U	!	В	н	1 1	٧Z	٧	С		4/2	5/3	2/2	4+/2+		Е	ΑN	1 F	Z	# F	РС	П	Y D	U	! E	3 G	н	I N	Z	٧
	96	B6	86	A6		၁၁	0	ာ	0	၁၁	0					0	0	•	•	)		96	В6	86	A6		)	0	ာ	0	<b>o</b> c	)	ာ	• •	•	•	•	0	<b>•</b>	•	•
													_	_																	_				_	_				_	_

Accumulator A is loaded with the contents of location M in memory.

When loading 1 bit data (address preselections I, O, F, S, T) data bit 0 from accumulator A contains the respective information. Data bits 1 to 7 have a NULL value.

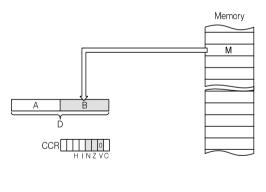


Mot	orola	LD/	۱B	Fun	ction												(	CP	U	ty	/p	е /	Α/	6	30	3
B&F	ł	LB		(M)	⇒B																					
Sho	rt																									
Δ,	dress	ina n	node /	Oncor	łe	Γ		Δι	44	re		: r	re		عاد	20	tic	'n			0	1		_	000	
		,g	iouc,	Ороос		L					-	<b>'</b> r						···			О	<u> </u>	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	z	٧	С
	D6	F6	C6	E6		0	0	0	0	0	0	0	0	0							0	0	•	•	•	0

Mot	orola	LDA	ιB	Fun	ction												C	CP	U	ty	/p	e E	3 /	6	В0	9
B&F	₹	LB		(M)	⇒В																					
Sho	rt																									
Δ,	ddraes	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		. r	ırc		عاد		tic	'n			O	1	С	P	30	
_^`	uui 633	ing ii	ioue /	Орсос	40	L		_,	uu		30	, ,	,,,		-			•••			С	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	CR		
	4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	D6	F6	C6	E6		Э	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

Accumulator B is loaded with the contents of location M in memory.

When loading 1 bit data (address preselections I, O, F, S, T) data bit 0 from accumulator B contains the respective information. Data bits 1 to 7 have a NULL value.

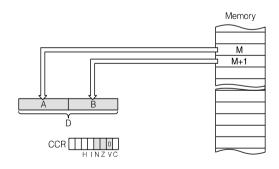


Mot	orola	LDE	)	Fun	ction												C	CP	U	ty	/p	e A	Α/	6	30	3
B&F	₹	LD		(M:N	/I+1) ⊏	۱ (	D																			
Sho	rt																									
	ddroes	ina n	node /	Oncor	10	Γ		Δ,	44	ro	ss	: r	ırc		ماد	201	Hio	'n			0	1	P	<b>G1</b> (	000	П
	uui 638	ing ii	ioue /	Орсос	10				uu		30	, ,	,, ,	.31	-	-		<b>,,,</b>			О	•	P	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3	3/3	5/2		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
	DC	FC	cc	EC		0	0	0	0	0	0	ာ	0	0							0	0	•	•	•	0

Mot	orola	LDE	)	Fun	ction												þ	P	U	ty	ďρ	e E	3 /	6	В0	9
B&F	ł	LD		(M:N	/l+1) ⊏	<b>&gt;</b> [	)																			
Sho	rt																									
۸,	droce	ina m	ode /	10	Γ		۸,	14	rn		s p			N/		·io	'n			0	7	С	Pδ	30		
Α.	Jui 638	ilig ii	ioue /	Орсос	16	L		~	Ju	16	33	, h	,, ,	-30	216	-		""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	5/2	6/3	3/3	5+/2+		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	DC	FC	CC	EC		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	۲	0

Accumulator D is loaded with the contents of locations M and M+1 in memory.

When loading 1 bit data (address preselections I, O, F, S, T) data bit 0 from accumulator A contains the contents of location M in memory and data bit 0 from accumulator B holds the contents of location M+1 which is the next higher address. Data bits 1 to 7 have a NULL value.



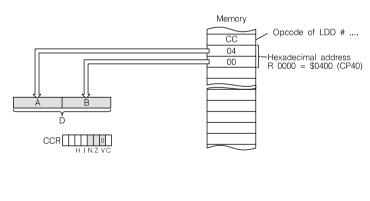
Mot	orola	LDF	(	Fun	ction												C	P	U	ty	/p	e .	A /	6	30	3
B&F	₹	LDF	(	M⊐	D																					
Sho	rt																									
	ddroes	ina n	node /	10	Γ		۸,	44	rn			re		٠		Hio	'n				T	PC	<b>G1</b> (	000	$\Box$	
Α.	uuies	ing ii	16			~	uu		33	, ,	,, e	.5	-10						O	<b>,</b>	P	G-F	S			
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	ĸ		Т
			3/3			Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
			cc			b	0	0	0	0		0	0	0							0	0	•	•	•	0

I	Mot	orola	LDF	(	Fun	ction												C	P	U	ty	/p	e I	3 /	6	80	9
	B&F	₹	LDF	(	Mఐ	D																					
	Sho	rt																									
ı	Δ,	ddraes	ina m	ode /	Oncor	40	Γ		Δ,	44	ro		s p	rc		عاد		Hio	'n				Τ	С	P	80	
ı		uui 633	ing ii	ioue /	Орсос	46	L			<i>a</i> u		30	, h	,, ,	.31	510			···			С	)	Р	C	80	
-	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		
Ī				3/3			Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
ı				CC			b	0	0	0	0		0	0	0					•	•	ા	0	•	•	•	0

Accumulator D is loaded with address M.

The command corresponds with the LDD # xxxx command. The PROgramming SYStem replaces the address given in the STL input line by the user with the effective address (hexadecimal value) which the B&R address (=address preselection + address position) holds in PLC memory.

Example: LDK R 0000 (in a CP40)



	Oi Oiu				011011												`			٠,	η.	٠,		•		•
B&F	₹	LDL		M⊏	D																					
Sho	rt																									
	Addressing mode / Opcode Address preselection																		P	<b>31</b> 0	000	П				
_ ^'	uuies	sing ii	loue /	Орсо	ue			^'	u		33	, ,	,, e	.50	316	-		"			О	•	Р	G-F	Š	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		٦
			3/3			Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	I	-	N	Z	٧	С

Mot	orola	LDL		Fun	ction												C	P	U	ty	р	e I	В/	6	В0	9
B&F	l	LDL		M□	D																					
Sho	rt																									
Δ,	dress	ina m	ode /	te at	Γ		Δι	44	re	99	: n	re	156	ale		tio	'n				Ι	_	P			
l ^``	.u. 000	g							-	, ,			•••			•••			а	1	Ρ	C	30			
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	CR		Т
			3/3			Ε	Α	М	F	Z	#	Р	С	Ι	Υ	D	U	!	В	G	н	I	N	Z	٧	С
			CC			Г	Г	Г				Г									0	0	•	•	•	ं

A label with address M is loaded to accumulator D.

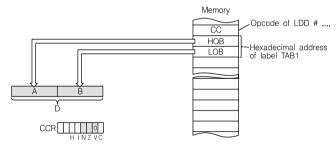
Eunotion

This instruction corresponds to the LDD # xxxx command. Only one label name can be entered in the STL input line by the user. The PROgramming SYStem replaces this name with the effective address (hexadecimal value) which the label stands for in PLC memory.

CBII tuno A / 6202

Example: LDL TAB1

Motorolo I DI



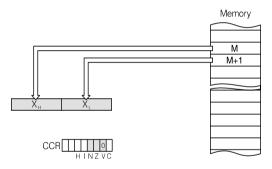
Note:

If a table name is given for the label accumulator D contains the address of the table header after executing the instruction. The first data byte is saved effectively in the memory location with address D + 15 (see the PROgramming SYStem user's manual, chapter 7 Table Editor).

Mot	orola	LDX	(	Fun	ction												(	CP	U	ty	р	е /	Α/	6	30	3
B&F	₹	LR		(M:N	VI+1) ⊏	> )	X																			
Sho	rt																									
	ddroe	ina n	node /	40	Г		۸,	44	rn			re		٠			'n			C	T	P	<b>G1</b> (	000	,	
	uui es	sing ii	ue .			^'	u		33	۰,	,, e	31	510	,,,		,,,,			С	1	Ρ	G-F	c			
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	R		
	4/2	5/3	3/3		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С	
	DE	FE	CE	EE								ာ	0	0							Э	0	•	•	•	0

Mote	orola	LDX		Fun	ction												þ	P	U	ty	/p	e I	3 /	6	80	9
B&R	1	LR		(M:N	/l+1) =	> )	K																			
Sho	rt																									
Ac	Addressing mode / Opcode Address presele														ct	tic	n			O	_		P			
,		g		оросс		<del>                                     </del>											•••			С	)	Ρ	C	30		
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	О	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R	Т	
	5/2	6/3	3/3	5+/2+		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
	9E	BE	8E	AE		Γ	П	П	П			0	0	0	•		•	•	•	•	0	0	•	•	▼	0

Index register X is loaded with the contents of memory locations M and M + 1.



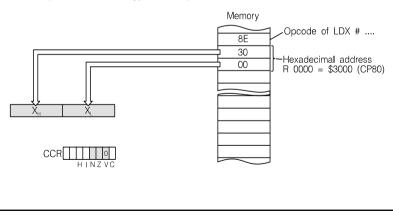
Mot	orola	LDX	Fun	ction												C	P	U	ty	/p	e A	Α/	6	30	3	
B&F	₹	LR		Mఐ	Х																					٦
Sho	rt																									-
Δ,	ddraes	ina m	ode /	Oncor	10	Г		Δ,	44	rΔ		· n	ırc		ماد		tic	'n			С	1	P	G1(	000	Л
Α.	uui ess	ilig ii	Орсос	16			^(	uu		33	۰,	,, e	.51	-10	,,,		""				<b>)</b>	Ρ	G-I	PC		
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G	Г		C	CR		
			Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С			
			CE				0	0	0	0		0	0								0	0	•	•	•	0

Mot	orola	LDX	(#	Fun	ction												þ	P	U	ty	/p	e I	3 /	6	В0	9
B&F	t	LR		M□	X																					
Sho	rt																									
Δ,	drace	ina m	ode /	Oncor	40	Γ		Δ,	44	rΔ		s p	rc		عاد		Hio	'n			О	1	С	Pδ	30	
^	aui ess	iiig ii	ioue /	Орсос	46				<i>a</i> u		30	, 1	,, ,	.30	,,,			···			О	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
			3/3			Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
			8E			0	0	0	0	0		0	0						•	•	0	0	•	•	•	0

Index register X is loaded with address M.

The command corresponds with the LDX # xxxx command. The PROgramming SYStem replaces the address given in the STL input line by the user with the effective address (hexadecimal value) which the B&R address (=address preselection + address position) holds in PLC memory.

Example: LDX# R 0000 (in a CP80 => CPU type B / 6809)



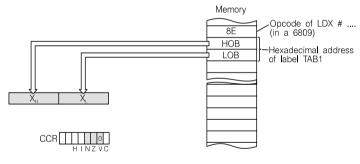
Mote	orola	a L	ЭX	Ľ		Fu	nction													CI	<b>?</b> U	l t	yp	е	A / 630
B&R	₹	LI	RL			М	⇒ X																		
Sho	rt																								
۸,	ddress	eeina	m	_	do /	Once	ndo.	Т		,	١d٥	ı.					۸l	~~	41,	'n			Ç	)	PG1000
Α.	uuies	SSIIIY			ue /	Opce	ue	L		_	·u	1116	. 33	, ,	,,,,	53	CII			,,,			C	)	PG-PC
IMPL.	DIR.	EX	. ]	IM	MED.	IND.	REL.	Ī	C	o [i	FS	Т	#	Р	R	х	Υ	D	υ	1	В	G	Γ		CCR

Mot	orola	LDX	(L	Fun	ction												C	P	U	ty	р	e E	3 /	6	В0	9
B&F	₹	LRL		Mఐ	Х																					
Sho	rt																									
	ddress	ina m	ode /	Oncor	10	Γ		Δ	44	rΔ		: n	rc		عاد		tio	'n			0	1	С	P	30	
l ^`	uui 633	ing ii	ioue /	Орсос	40	L		^'	u		30	, 1	,,,	.30	,,,			•••			a	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			C	CR		Τ
			3/3			Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	н	I	Ν	Z	٧	С
			8E			Г	Г	Г	П			Г	П	П							0	0	•	•	•	0

A label with address M is loaded to index register X.

This instruction corresponds to the LDX # xxxx command. Only one label name can be entered in the STL input line by the user. The PROgramming SYStem replaces this name with the effective address (hexadecimal value) which the label stands for in PLC memory.

Example: LDXL TAB1



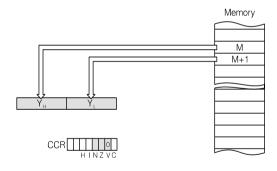
Note:

If a table name is given for the label index register X contains the address of the table header after executing the instruction. The first data byte is saved effectively in the memory location with address D + 15 (see the PROgramming SYStem user's manual, chapter 7 Table Editor).

Mot	orola			Fun	ction												C	P	U	ty	/p	e .	A A	6	30	3
B&F	1																									
Sho	rt																									
Δ,	ldraes	ina n	node /	Oncor	10	Γ		Αc	44	rΔ		· n	rc		ale	201	Hio	'n				Τ	P	G10	000	<u>,                                     </u>
~	uui ess	ing ii	ioue /	Орсос	40	L		^'	u		30	, 1	,,,	.30	,,,			•••				Ι	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		Ξ
						Ε	Α	М	F	z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С

Mot	orola	LDY	7	Fun	ction												þ	P	U	ty	γp	e E	3 /	6	В0	9
B&F	t	LY		(M:N	/l+1) ⊏	۰,	Y																			
Sho	rt																									
Δι	dress	ina m	ode /	Oncor	łe	Γ		Δ	14	re	٠,	s p	re	250	عاد	201	tio	'n				Ι	_	P		
í	au. 000	g	iouc,	Ороос		L			••		•	, ,			٠.,			•••			О	ı	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	6/3	7/4	4/4	6+/3+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	1	N	Z	٧	С
	10 9E	10 BE	10 8E	10 AE		Γ	Г					•	•	•	•		•	•	•	•	0	0	•	•	•	0

Index register Y is loaded with the contents of the given memory locations M and M + 1.



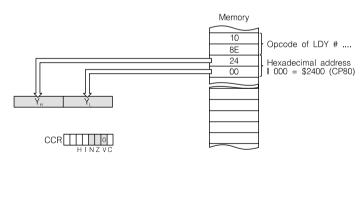
Mot	orola			Fun	ction												C	CP	U	ty	/p	e .	A.	/ 6	30	3
B&F	₹																									٦
Sho	rt																									١
Δ,	ddress	ina m	ode /	Oncor	łe	Γ		A	44	re		: r	re		۱e	ci	tio	'n				Ι		_	000	
	uu. 000	,g	.ouc,	Ороос		L		^,,		٠٠	-	'		-	•••			•••					Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		
						Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	Ι	N	Z	٧	С

Mot	orola	LDY	#	Fun	ction												þ	P	U	ty	/p	e I	В /	6	В0	9
B&F	t	LYK		M□	Υ																					
Sho	rt																									
	ddress	ina m	odo /	0000	10	Γ		۸.	14	re											Г	Τ	С	P	30	
_ A	uiess	ing ii	oue /	Opcoo	ie.			A	Ju	ıe	53	· P	,,,	:51	316	:6	LIC	""			О	7	Р	С	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			C	R		
			4/4			Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
			10 8E			•	•	•	•	•		•	•						•	•	0	0	•	•	•	0

Index register Y is loaded with address M.

The command corresponds with the LDY # xxxx command. The PROgramming SYStem replaces the address given in the STL input line by the user with the effective address (hexadecimal value) which the B&R address (=address preselection + address position) holds in PLC memory.

**Example:** LDY# I 000 (in a CP80 = CPU type B / 6809)



IVIOL	UlUla			ruii	CLIOII												_ ^	,,	U	·	·μ	· ,	٦,	U.	30	J
B&F	₹																									
Sho	rt																									
Λ.	ddroes	ina m	node /	Oncor	40	Г		۸,	14	rn			re		1			'n				Τ	P	<b>G1</b> (	000	
A	uures	sing ii	ioue /	Opcoo	Je			Α,	Ju	16	55	, ŀ	,, е	:56	316	:0	···	,,,				T	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
						Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
						Г	Г																			

T	Mot	orola	LDY	Ľ	Fun	ction												þ	P	U	ty	'n	e I	3 /	6	В0	9
Г	B&R	1	LYL		M□	Y																					
	Sho	rt																									
Г	Δ	ldrass	ina m	ode /	Oncor	10	Γ		Δ,	14	rΔ		s p	rc		عاد		Hio	'n				Τ	С	P	30	
L	Α(	iui caa	ilig ii	loue /	орсос	16				ıu	16	33	, h	,, ,	-30	316			""			О	1	Р	C	30	
Г	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
				4/4			Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
				10 8E																		0	0	•	•	•	0

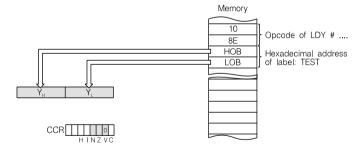
A label with address M is loaded to index register Y.

Eunotion

This instruction corresponds to the LDY # xxxx command. Only one label name can be entered in the STL input line by the user. The PROgramming SYStem replaces this name with the effective address (hexadecimal value) which the label stands for in PLC memory.

CBII tupo A / 6202

Example: LYL TEST



Note:

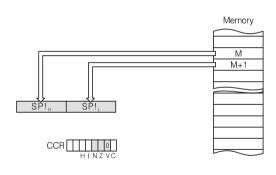
Motorolo

If a table name is given for the label index register Y contains the address of the table header after executing the instruction. The first data byte is saved effectively in the memory location with address D + 15 (see the PROgramming SYStem user's manual, chapter 7 Table Editor).

Mot	orola	LDS	;	Fun	ction												C	P	U	ty	/p	е /	<b>A</b> /	6	30	3
B&F	₹	LS		(M:N	<b>/</b> 1+1) ⊏	, ;	SP	!																		
Sho	rt																									
Δ,	ddroes	ina m	ode /	Oncor	10	Г		Δ,	44	re		: r	ırc		ماد		tio	'n			O	1	P	<b>G1</b> (	000	$\Box$
^	uui 638	ing ii	ioue /	Орсос	10				uu		30	, ,	,, ,	.31	-			···			U	•	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
	4/2	5/3	3/3	5/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	z	٧	С
	9E	BE	8E	AE								0	0	0							0	0	•	•	▼	0

Mot	orola	LDS	;	Fun	ction												C	P	U	ty	ďρ	e I	3 /	6	80	9
B&F	t	LS		(M:N	/l+1) =	> ;	SP	!																		
Sho	rt																									
Δι	dress	ina m	ode /	Oncor	łe	Γ		Δι	44	re		s p	re	250	عاد	20	tic	'n			0			P		
	au. 000	g	ouc,	Ороос		L					•	, ,			•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	CR		Т
	6/3	7/4	4/4	6+/3+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U		В	G	Ι	I	N	Z	٧	С
	10 DE	10 FE	10 CE	10 EE		Г	Г	П	Г			0	0	0	•		•	•		•	0	0	•	•	▼	ं

The system stack pointer SP! is loaded with the contents of memory locations M and M+1.



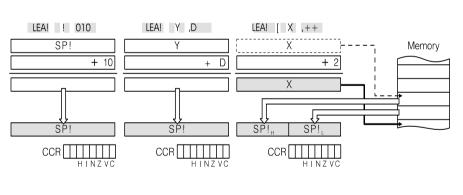
Mot	orola			Fun	ction								С	PU	l ty	ре	Α/	6303	3	Mot	orola	LEA	.!	Fun	ction									СР	Ut	уре	B / 68	09
B&F	2																		1	B&F	1	LE!		EA	⇒ SP!													
Sho	ort																		ı	Sho	rt																	
Δ,	ddress	ina m	node /	Oncor	de.	Γ	Δι	ldr	ess	nr	250	lec	tio	n	┒			1000	]	Δ,	ldress	ina m	ode /	Onco	de.	Г	Δ	hh	Ires	s n	res	ماه	ctio	nn			CP 8	-
	uu. coc	,g	.ouc,	ороос		ı				ρ.,					- 1		PG	-PC	1	^``	.u. 000	,g	.ouc,	Ороо		ı	•	····		J P		٠.٠	••••	٠		0	PC 8	0
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1 0	) F	ST	#	PR	x	Y D	U	! В	G		CCI	R	7	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0 1	s	T #	P	R X	Υ	D U	1	В G	Г	CCR	
						E/	АМ	F Z	#	РС	1	Y D	U	! В	G	ΗΙ	N 2	z v	С				4+/2+			Ε	ΑN	ΛF	Z #	Р	СІ	Υ	D U	!	В С	ΗΙ	N Z	v c
						П	Т		П	T	П	T	П	T	П	T	П	$\Box$	7				32			T	T	Т	П	П	•	•	•	•	T	0	00	o o

The effective address EA is calculated from the indexed addressing given by the instruction and is stored in system stack pointer SP!.

**Example:** LEA! ! 010  $SP! + 10 \Rightarrow SP!$ 

LEA! Y ,D  $Y + D \Rightarrow SP!$ 

LEA! [ X ,++  $(X:X+1) \Rightarrow SP!; X+2 \Rightarrow X$ 



Mot	orola			Fun	ction												C	PΙ	J	ty	ре	<i>,</i>	4/6	30	3
B&F	₹																								
Sho	rt																								
	ddroes	ina n	node /	Oncor	10	Γ		Ad	14	ra			·ro				i۸	n					PG	1000	$\Box$
Α.	uuies	onig ii	ioue /	Орсос	16			~	Ju	16	33	۰,	,, e	.5	316	-		"				Т	PG	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	! E	3	G			CCF	₹	
						F	Δ	м	F	7	#	Р	С	Т	Υ	D	u	1 F	3	G	н	П	N Z	v	С

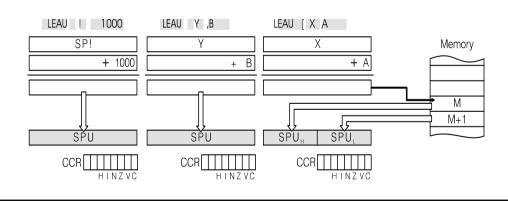
Mot	orola	LEA	Ü	Fun	ction												C	P	U	ty	γp	e I	3 /	6	В0	9
B&F	1	LEU		EA :	⇒ SPl	J																				
Sho	rt																									
Δ,	ldraes	ina m	ode /	Oncor	10	Γ		Δ	44	rΔ		· r	rc		عاد		tio	n				Τ	С	P	30	
^\	.u. 000	g	ouc,	Ороос						٠٠	•	' r			•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			C	CR		
			4+/2+			Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
			33			Г	Π	Γ						•	•		•	•			0	0	0	0	0	0

The effective address EA is calculated from the indexed addressing given by the instruction and is stored in user stack pointer SPU.

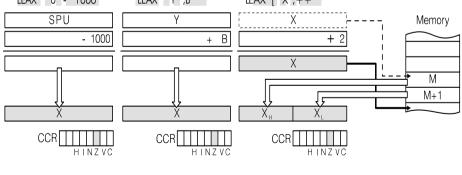
**Example: LEAU** ! **1000** SP! + **1000** ⇒ SPU

LEAU Y ,B  $Y + B \Rightarrow SPU$ 

LEAU [ X ,A (X+A:X+A+1) ⇒ SPU



	orola			Fun	ction							СР	U typ	e A / 6303	зП	Mote	orola	LE	AX	Fun	oction								СР	U ty	ре Е	3 / 680
B&R															1	B&R	1	LE	R	EA	⇒ X											
Shor	rt														П	Sho	rt															
Ad	ldress	ing n	node /	Орсо	de		Add	ress	pre	sel	ecti	on	H	PG1000 PG-PC	7	Ac	dres	sing ı	node /	Орсо	de		Ac	ldre	ss	pre	esel	ect	ion		<u>ာ</u>	CP 80 PC 80
MPL.	DIR.	EXT.	IMMED.	IND.	REL.		F S		P R		D U		ВБ	CCR	1	IMPL.	DIR.	EXT.		IND.	REL.	$\rightarrow$	O F		# F	-1	-	D	$\rightarrow$	В G		CCR
_						ΕA	M F	Z #	PC	ΙY	D U	1 !	B G F	I N Z V					4+/2+			E .	A M	F Z	# F	c	_	-	$\rightarrow$			N Z V
						Ш	ш					Ш	Ш		J				30			Ш					) ●	Ш	• •		၁၂၁	o   •   o
			LEA	х	Y,	<b>,</b> B			Υ	+ E	3 ⇒	Χ																				
			LEA	х [	х	,++			(X	:X+	⊦1)÷	⇒ )	κ; x	+ 2 X																		
				LEAX	U	-	100	00			LE/	4Х	Υ	,В		LE	AX	Χ	,++													
					S	PU	_						١	′				>	<		<u>-</u>		-			М	emc	ry				
								100	_							_					_					$\overline{}$	_					



Mot	orola				Fun	ction												C	P	U	ty	pε	e /	4/6	30	13
B&F	₹																									
Sho	rt	Г																								
	ddroes	ein	a m	ode /	Oncor	10	Γ		Αc	14	ro		· n	ro					n					PG	000	)
Α.	uuies	3111	9	oue /	Орсос	16			~	ıu		33	, 1	" 6	30	316	-		"				Т	PG	PC	
IMPL.	DIR.	E)	XT.	IMMED.	IND.	REL.	ī	0	F	S	т	#	Ρ	R	Х	Υ	D	U	!	в	G			CCF	1	П
							F	Δ	М	F	7	#	Р	С	_	V	n	ш	1	R	G	п	П	N Z	V	C

Mot	orola	LEA	·Υ	Fun	ction												C	P	U	ty	γp	e E	3 /	6	В0	9
B&F	ł	LEY	,	EA :	⇒Y																					
Sho	rt																									
Ad	dress	ina m	ode /	Oncod	de .	Γ		Δα	14	re	SS	s p	re	se	ele	c	tic	n				I	_	P		
		g		орос							_	, 1				~		•••			0	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
			4+/2+			Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U		В	G	Η	I	Ν	Z	٧	С
			31			Г	П	П					П	•	•		•	•			0	0	0	•	0	0

The effective address EA is calculated from the indexed addressing given by the instruction and is stored in index register Y.

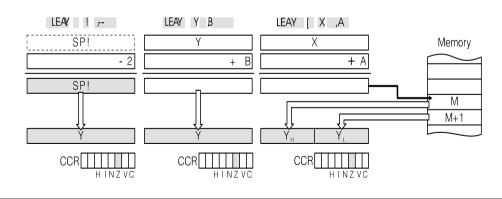
Example:

LEAY

LEAY Y,B

$$Y + B \Rightarrow Y$$

LEAY [ X ,A



# **3.2. STORE INSTRUCTIONS**

All instructions which store data (1 or 2 bytes) from a register to a designated location in memory are explained in this section.

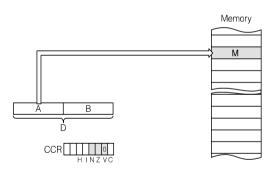
Market	D.0 D		Operatir	ıg mode	
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
STAA	=	0	0	0	0
STAB	=B	0	0	0	0
STD	=D	0	0	0	0
STX	=R	0	0	0	0
STY	=Y				0
STS	=S	0	0	0	0

Mot	orola	STA	AA.	Fun	ction												C	CP	U	ty	/p	e .	A A	6	30	3
B&F	ł	=		A⇒	(M)																					
Sho	rt	T																								
Δ,	drace	ina m	node /	Oncor	40	Г		Δ,	44	rΔ		· n	re		ماد	201	Hio	'n			U	)	P	G1(	000	)
ť	Jui 638	sing ii	loue /	Орсос	ue			^(	Ju	16	33	۰,	,, e	.5	-10	-		""			$\circ$	)	Р	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	т	#	Ρ	R	х	Υ	D	U	!	В	G			C	CR		П
	3/2	4/3		4/2		Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Η	ı	N	Z	٧	С
	97	B7		A7		Г	0	0	0			0	0	0							0	0	•	•	▼	ာ

Mot	orola	STA	Α	Fun	ction												þ	ЭP	U	ty	/p	e I	В/	6	В0	9
B&F	t	=		A⇔	(M)																					
Sho	rt	ı																								
Δd	drace	ina m	odes/	Onco	do	Γ		Δ,	44	rΔ		s p	ırc		عاد		tic	'n			0	1	С	P	30	
Α.	uiess	iiig iii	oues /	Орсо	ue	L			<i>1</i> u		30	, 1	,,,		,,,			"			О	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3		4+/2+		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
	97	В7		A7		Ĺ	0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	0

The contents of accumulator A is stored in the memory location with address M. Accumulator A remains unchanged.

If the destination address points to a 1 bit location (O, F, or S), only data bit 0 from accumulator A is stored.

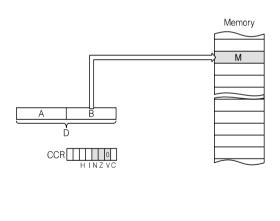


Mot	orola	STA	В	Fun	ction												(	CP	'n	ty	/p	e .	A A	6	30	3
B&F	ł	=B		В⇨	(M)																					
Sho	rt																									
Δ.	drace	ina m	odes/	Onco	da	Г		Δ,	44	ro		. r	re		ماد	200	tic	'n			U	)	P	<b>31</b> (	000	)
	uicoo	g	oues /	Орсо	ue			^	u			, ,	,,,		-10			"			C	)	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		П
	3/2	4/3		4/2		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	N	z	٧	С
	D7	F7		E7		Ĺ	0	0	0			0	0	0							0	0	•	•	•	0

Mot	orola	STA	В	Fun	ction												C	P	U	ty	/p	e I	3 /	6	В0	9
B&F	ł	=B		В⇨	(M)																					
Sho	rt																									
Δd	dross	ina m	odes /	Onco	da	Γ		Δ,	44	ro		s p	ırc		عاد		tic	'n			0	1	С	P	30	
ζ	uiess	iiig iii	oues /	Opco	ue				<i>1</i> u		3.	, h	,,,		,,,			<b></b>			О	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3		4+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	Ν	Z	٧	С
	D7	F7		E7		Γ	0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	0

The contents of accumulator B is stored in the memory location with address M. Accumulator B remains unchanged.

If the destination address points to a 1 bit location (O, F, or S), only data bit 0 from accumulator B is stored.

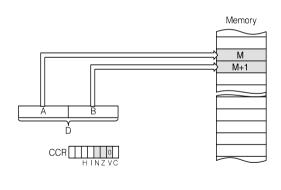


Mot	orola	STA	D	Fun	ction												C	P	U	ty	p	9 /	<b>A</b> /	6	30	3
B&F	ł	=D		D⇔	(M:M	+1	)																			
Sho	rt																									
	droce	ina m	ode /	Oncor	10	Γ		۸,	14	rn			re		N/		Hio	'n			C	7	P	31	000	$\overline{}$
^	Julesa	ilig ii	ioue /	орсос	16	L		^'	Ju		33	۰,	,, e	-30	216	-		""			C	1	P	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3		5/2		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	DD	FD		ED			0	0	0			0	0	0							Э	0	•	•	•	0

T	Mot	orola	STA	D	Fun	ction												(	CP	U	ty	ďρ	e E	3 /	6	В0	9
ſ	B&F	ł	=D		D⇒	(M:M	+1	)																			
	Sho	rt																									
ı	۸.	droce	ina m	odes /	Onco	do	Γ		۸,	14	ro		s p			٠		410	'n			0	7	С	P	30	
L	Au	uicss	iiig iii	oues /	Opco	ue	L		~'	Ju	16	33	, h	,, ,	.5	-16			"			О	1	Р	C	30	
I	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
Ī		5/2	6/3		5+/2+		Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
I		DD	FD		ED			0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	0

The contents of accumulator D is stored in the memory location with addresses M and M+1. D remains unchanged.

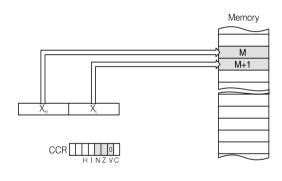
If the destination address points to a 1 bit memory location (O, F, or S), only data bit 0 from accumulator A is stored in location M and data bit 0 from accumulator B in location M+1.



Mot	orola	STX	(	Fun	ction												C	CP	U	ty	φ	e A	A /	6	30	3
B&F	₹	=R		X⇔	(M:M-	+1	)																			
Sho	rt																									
Δ.	droce	ina m	odes/	Onco	da	Г		Δ	44	re		: r	re		ماد	200	tic	'n			O	)	P	<b>G1</b> (	000	П
	uiess	iiig iii	oues /	Opco	ue				uu			, ,	,,,	.3	-			"			U	)	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
	4/2	5/3		5/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	DF	FF		EF								0	0	0							0	0	•	•	•	0

Mot	orola	STX		Fun	ction												C	P	U	ty	р	e E	3 /	6	В0	9
B&F	₹	=R		X⇔	(M:M	+1	)																			
Sho	rt																									
Δο	ldress	ina m	ndes /	Onco	do	Γ		Δ,	44	ro		: n	ırc		ماد		tic	'n			0	1	С	P	30	
	ui 633	iiig iii	oues ,	Opco	ue	L		_,	-		3	, 1	,,,	.3	-			•••			a	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		
	5/2	6/3		5+/2+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	н	I	Ν	Z	٧	С
	9F	BF		AF		Γ			Г			0	0	0	•	Г	•	•	П	•	0	0	•	•	▼	0

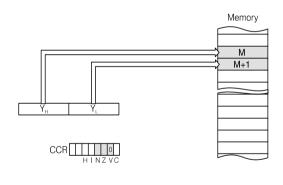
The contents of index register X will be stored in the memory locations with addresses M and M+1. X remains unchanged.



Mot	orola			Fun	ction												C	P	U	ty	/p	e.	A.	/ 6	30	3
B&F	ł																									1
Sho	rt																									١
Δ.	ddroes	ina m	ode /	Oncor	10	Г		Δ,	44	rΔ		· n	re		عاد	201	io	'n				T	Р	G1(	000	╗
ť	au 1 6 3 3	ing ii	ioue /	Орсос	10				<i>a</i> u		30	, 1	,, ,	.30	,,,			···					Р	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	!	В	G	Γ		C	CR		٦
						ш	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
						L															L					П

3	Т	Moto	orola	STY	,	Fun	ction												c	P	U	ty	ďρ	e E	3 /	68	30	9
1		B&R	1	=Y		Y⇔	(M:M	+1	)																			
		Sho	rt																									
]	Γ	Ac	dress	ina m	ode /	Oncor	le			Δα	łН	re	SS	s p	re	se	ele	ct	io	n				I	_	P8		
┚				9			-									_							О	l	Р	CE	30	
1	1	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	I	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
С	Г		6/3	7/4		6+/3+		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
]			10 9F	10 BF		10 AF								0	0	0	•		•	•		•	0	0	•	•	•	0

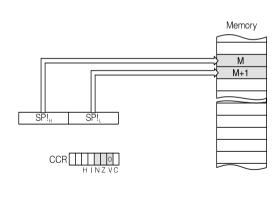
The contents of index register Y will be stored in the memory locations with addresses M and M+1. Y remains unchanged.



Mot	orola	STS	;	Fun	ction												C	P	U	ty	/p	е /	<b>A</b> /	6	30	3
B&F	₹	=S		SP!	<b>⇒</b> (M:	M	+1	1)																		٦
Sho	rt																									
Δd	droce	ina m	odes /	Onco	da	Г		Δ	44	ro		· r	\rc		ماد	٠.	tic	'n			U	7	P	<b>G1</b> (	000	╝
Λu	uiess	ing iii	oues /	Орсо	ue				uu		30	, ,	,,,		-			<b>'''</b>			$\cup$	)	Ρ	G-F	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3		5/2		Е	Α	М	F	Z	#	Ρ	С	_	Υ	D	U	!	В	G	Ι	1	Ν	Z	٧	С
	9F	BF		AF								0	0	0							0	0	•	•	•	0

	Mot	orola	STS	;	Fun	ction												C	P	U	ty	ďρ	e I	3 /	6	В0	9
	B&F	ł	=S		SP!	⇒ (M:	M	+1	)																		
	Sho	rt																									
	Δd	dress	ina m	odes /	Onco	de	Γ		Δι	44	re		s p	ıre		عاد	20	tic	'n			0	_		P	_	
ı		u. 000	g	ouco,	Орос	·uc	L			-		•	, ,			٠.,			•••			0	1	Ρ	C	30	
	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	CR		
		6/3	7/4		6+/3+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U		В	G	Ι	I	N	Z	٧	С
		10 DF	10 FF		10 EF		Г	Г	П	Г			0	0	0	•		•	•		•	0	0	•	•	•	0

The contents of system stack pointer SP! will be stored in the memory locations with addresses M and M+1. SP! remains unchanged.



# 3.3. DATA EXCHANGE BETWEEN REGISTERS

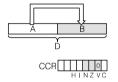
All instructions which store data (1 or 2 bytes) from a register to another or exchange data between two registers, are described in this section.

Markanala	505				
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
TAB	MAB	0	0	0	0
TBA	MBA	0	0	0	0
TAP	MAC	0	0	0	0
TPA	MCA	0	0	0	0
TSX	MSR	0	0	0	0
TXS	MRS	0	0	0	0
XGDX	DXR	0	0	0	0
TFR	TFR				0
EXG	EXG				О

Motorola	TAB	Function	CPU type A / 6303
B&R	MAB	A⇔B	
Short			

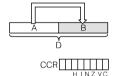
۸,	ddroes	ina m	ode /	Oncor	10	Г		۸,	44	ro					٠.,		Hio	'n			O	7	P	<b>G10</b>	000	П
Α.	uui es	sing ii	ioue /	Орсос	16			~	Ju		33	, ,	Preselection					O	7	Р	G-F	Š	٦			
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	ĸ		٦
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
16						Г	П	П	П			П	П						П		0	0	•	•	•	0

The contents of accumulator A will be copied to accumulator B.



Mot	orola	TAE	3	Fun	ction												C	P	U	ty	ре	e E	3 /	68	30	9
B&F	₹	MA	3	A⇒	В																					
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	łe	Г		Δι	44	re	99	: n	re		۱e	20	tio	'n			0	1		P	_	
^``	uu. 000	g	ouc,	Ороос						٠.	-	, ,		-				•••			0	ı	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
6/2						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U		В	G	Н	1	Ν	Z	٧	С
1F 89																					0	0	0	0	0	0

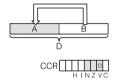
The contents of accumulator A will be copied to accumulator B.



Motorola	TBA	Function	CPU type	A / 6303
B&R	MBA	B⇔A		
Short				
				DO 4000

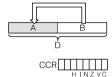
	ddroe	ina m	ode /	Oncor	40	Г		A	44	ro					٠			'n			O	7	P	<b>G10</b>	000	П
Α.	uui es	sing ii	ioue /	Орсос	16			^'	u		33	, ,	,, e	.5	516			"			O	7	Р	G-F	'n	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R	Т	٦
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
17						Г	П	Г	Г	Г	П	П	П	П	П				П		0	0	•	•	•	0

The contents of accumulator B will be copied to accumulator A.



Mot	orola	TBA	١	Fun	ction												C	P	U	ty	p	e E	3 /	68	30	9
B&F	~	MB	4	В⇨	Α																					
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		: n	rc		عاد		tio	'n			0	1	С	Pε	30	
_^`	aui 633	iiig ii	oue,	Орсос	40	L		^	u		30	, h	,,,	.30	,,,			•••			a	1	Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
6/2						Ε	Α	М	F	Z	#	Р	С	Ι	Υ	D	U	!	В	G	н	I	Ν	Z	٧	С
1F 98						Г	Г	Г	П			Г	П								0	0	0	0	0	0

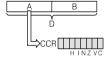
The contents of accumulator B will be copied to accumulator A.



	Motorola	TAP	Function	CPU type A / 6303
	B&R	MAC	A ⇒ CCR	
1	Short			

۸.	ddroed	ina n	node /	Oncor	40	ı		۸,	44	ro			ore		٠ı،	~~	41,	'n			C	<b>)</b>	PC	<b>310</b>	)00	,
_ ^'	uui es	sing ii	loue /	Орсос	ue			^'	uu		33	<b>,</b>	,, ,	-31	-10	-6		,,,			O	<b>7</b>	P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R	Т	٦
1/1						Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Η	1	Ν	Z	٧	С
06						Γ	Γ	Γ	Γ		Γ	Γ	Γ	Γ		Γ	Γ				•	•	•	•	•	•

The contents of accumulator A will be copied to condition code register CCR.



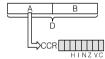
### Note:

If the Interrupt Inhibit Bit I (Bit 4 of the condition code register) is set to logic 1 by the user all interrupts are locked. This technique is used if parts of the user program are not to be broken by an interrupt in any case.

Before the END instruction is processed bit I absolutely must be set to logic 0 again.

Mot	orola	TAP	•	Fun	ction												C	P	U	ty	/p	e I	3 /	6	В0	9
B&F	ł	MAG		A⇔	CCR																					
Sho	rt																									
Α.	dress	ina m	ode /	Oncor	de .	Γ		Δı	44	re	SS	s p	re	S	ele	c	tic	n			O	-		P8	_	
						L					_					_					С		Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		_
6/2						Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U		В	G	Η	I	N	Z	٧	С
1F 8A																					•	•	•	•	•	•

The contents of accumulator A will be copied to condition code register CCR.



## Note:

If the Interrupt Inhibit Bit I (Bit 4 of the condition code register) is set to logic 1 by the user all interrupts are locked. This technique is used if parts of the user program are not to be broken by an interrupt in any case.

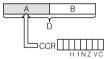
Before the END instruction is processed bit I absolutely must be set to logic 0 again.

Bit 7 and 6 of the condition code register may not be changed by the user.

Mot	orola	TPA	١.	Fun	ction												C	P	U	ty	/p	e A	<b>A</b> /	6	30	3
B&F	ł	MC	A	CCF	R⇒A																					٦
Sho	rt																									١
Δ,	ddroes	ina m	ode /	Oncor	10	Г		Δ	44	rΔ		: r	ırc		ماد		tic	'n			U	7	P	<b>31</b> 0	000	ቯ
_^\	au 1 6 3 3	ing ii	ioue /	Орсос	10				uu		30	, ,	,, ,	.31	-			···			$\cup$	)	Ρ	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	!	В	G			CC	R		
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
07																					0	0	0	0	0	ာ

Mot	orola	TPA		Fun	ction												¢	P	U	ty	/p	e E	3 /	6	В0	9
B&F	t	MCA	4	CCF	R⇔A																					
Sho	rt																									
	ddress	ina m	odo /	Oncor	10	Γ		۸,	44	ra							tio	'n			0	ī	С	P	30	
<u>۱</u> ^'	uui ess	ing in	loue /	орсос	16			~'	uu	16	33	۱ د	,, e	.50	316	-					О	7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	ļ	В	G	Г		C	CR		
6/2						Ε	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
1F A8						Γ	Γ	Г	П			Γ	Γ								0	0	0	0	0	0

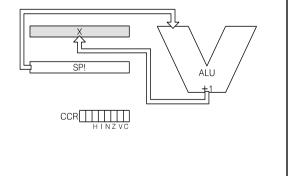
The contents of condition code register CCR will be copied to accumulator A.



Motorola	TSX	Function	CPU type A / 6303
B&R	MSR	SP! + 1 ⇒ X	
Short			

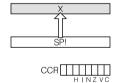
۸,	ddroe	ina m	node /	Oncor	40	ı		۸.	44	rn			ore		٠.,			'n			Ų	<u>'</u>	P	ااذ	JUU	
Α.	uuies	sing ii	loue /	Орсос	16			^'	uu		33	, ,	,, ,	-31	-10			,,,			0	ī	P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		٦
1/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	Τ	N	Z	٧	С
30							П													П	0	0	0	0	0	0

Index register X will be loaded with the address of the memory location which holds the last valid value of the stack. Since in 6303 processors the stack pointer points to the next free location in the stack a value of 1 is added to the system stack pointer SP! and the accumulator is stored in index register X.



		_																								_
Mote	orola	TSX		Fun	ction												C	P	U	ty	/p	e E	3 /	6	В0	9
B&R	<b>?</b>	MSF	₹	SP!	⇒ X																					
Sho	rt		g mode / O																							
۸۰	droce	ina m	odo /	Oncor	5	Г		۸,	44	rn		· n			N/a		io	n			0	T	С	P	30	
	uui ess	ing ii	loue /	Орсос	ie.			~'	uu		33	, h		.50	316			""			О	7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		٦
6/2						Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
1F 41						Г	Г		П												0	0	0	0	0	0

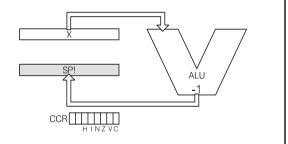
Index register X will be loaded with the address of the memory location which holds the last valid value of the stack Since in 6809 processors the stack pointer points to the last valid value in the stack no corrections which are required in the 6303 processor must be done. Therefore SP! is copied to index register X.



Motorola	TXS	Function	CPU type A / 6303
B&R	MRS	X - 1 ⇒ SP!	
Short			

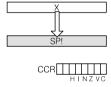
۸.	ddroed	ina m	ode /	Oncor	40	ı		۸,	44	ro	ss				٠.			'n			Ç	<b>)</b>	PC	<b>310</b>	)00	)
^'	uuies	sing ii	ioue /	Орсос	16			^'	u		33	, ,	,, ,	-31	CIC	-		,,,			С	<b>7</b>	P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	ĸ	Т	٦
1/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
35						Γ	Г	Г	Г			Г	Г	П				Г	П		0	0	0	0	0	0

This instruction is the opposite of the TSX instruction. The contents of index register X are reduced by 1 and loaded to the system stack pointer.



Mot	orola	TXS	;	Fun	ction												c	P	U	ty	ρę	e E	3 /	68	В0	9
B&F	₹	MRS	6	X⇔	SP!																					
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ	٠.	: n	re		ele		·io	n			0	1	С	Pε	30	
l ^`	uui 633	iiig ii	oue,	Орсос	40	L		_,	u		30	, 1		.30	,,,			•••			0	1	Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
6/2						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	ļ	В	G	Н	I	Ν	Z	٧	С
1F 14						Γ		Г			Г									П	0	0	0	0	0	0

This instruction is the opposite of the TSX instruction. The contents of index register X are loaded to the system stack pointer.

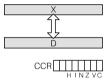


Attention: The system stack pointer SP! may only be changed if the running of stack operations is thoroughly understood. If the system stack pointer is not changed carefully the result may be a CPU error. This causes a HALT in the CPU; the error message "STACK ERROR" is shown in the status test.

CPU type B / 6809

Moto	rola	XGE	X	Fun	ction									CF	v	ty	ре	A	63	303	П	Moto	orola	XG	DX	Fun	ction										CF	PU	typ	oe I	3 / 680	9
B&R		DXR	~	D (=	> X																	B&R		DXI	R	D ¢	⇒ X															I
Short	t																					Sho	rt																			
Add	dress	ing m	ode /	Орсос	le		Ad	dre	ss	pre	se	le	ctic	on		Ŧ	<u>o</u>		G10 G-F	)00		Ac	ldress	sing n	node /	Орсо	de	T	,	Add	lre	ss	pr	es	ele	cti	ion	1		<u>၁</u>	CP 80 PC 80	_
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı o	F	S T	# !	R	X.	ΥC	U	!	В	G		C	CR		l	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	FS	Т	# F	R	x	Υ	DΙ	J !	В	G		CCR	7
2/1						E A	М	F Z	#	С	1	ΥC	) U	!	В	G	ΗΙ	N	Z	v c	[	7/2						Ε	Α	M F	z	# F	c	T	Υ	DΙ	J !	В	G H	1 1	N Z V	С
18																	)	ာ	0	၁		1E 10						L											С	)	000	ା

The contents of accumulator D are exchanged with the contents of index register X.



Mot	orola			Fun	ction												C	P	U	ty	рe	A/6	303
B&F	ł																						
Sho	rt																						
Addressing mode / Opcode															٠	ect	i۸	n		Т		PG1	000
^(	uui es	siiig	iiioue /	Орсо	ue			^'	u	16	33	۰,	,, e	.5	516	,,,,	10	••		I		PG-	PC
IMPL.	DIR.	EXT	IMMED	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	!	В	G		CCR	
						F	Α	М	F	7	#	Р	С	Т	Υ	D	u	1	В	G	н	1 N Z	v c

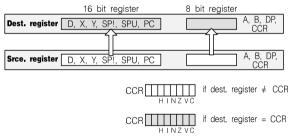
Mot	orola	TFR		Fun	ction												þ	P	U	ty	ďρ	e E	3 /	68	В0	9
B&F	t	TFR		r <sub>source</sub>	"⇔r <sub>o</sub>	est	tina	tion																		
Sho	rt																									
Δ,	Addressing mode / Opcode								44	rΔ	٠.	· n	re		عاد		tio	'n			0	1	С	Pε	30	
	au. 000	g	ouc,	Ороос						٠٠	•	, ,		-	•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	I	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
6/2				Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С		
1F						Г	Г														•	•	•	•	•	•

The contents of the indicated source register will be copied to the destination register. The source and destination registers must be the same size; either a 1 byte register (A; B, DP or CCR) or a 2 byte register (D, X, Y, SP!, SPU or PC).

Syntax: TFR T,r—Destination register
Source register

The entire instruction consists of two bytes. The first byte is the opcode (instruction) and the second byte (post byte) gives the source and destination byte.

		Post	byte
Dest. / Srce. register	r	Srce.	Dest.
D X Y P U S P! P A B D P C C R	DXYU!\$ABPC	0000 0001 0010 0011 0100 0101 1000 1001 1011 1010	0000 0001 0010 0011 0100 0101 1000 1001 1011 1010



Attention: This system stack pointer SP! may only be changed if the running of stack operations is thoroughly understood. If the system stack pointer is not changed carefully the result may be a CPU error. This causes a HALT in the CPU; the error message "STACK ERROR" is shown in the status test.

Changing the program counter PC causes an unconditional jump to the address loaded to the PC.

Mot	orola			Fun	ction										C	P	Ut	уре	A	/ 6303
B&F	₹																			
Sho	rt																			
	ddroed	ina n	node /	Oncor	40	Γ		Ad	dro		- n	rne	ماد	~~	410	'n		Т	ı	PG1000
^'	uuies	ing ii	ioue /	орсос	ue		,	-u	JI 6	33	P	168	CIC			""		Г		PG-PC
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	Г	lol	F	Т	#	РΙ	R Ix	Y	Ь	lυl	!	вЮ	3	_	CCR

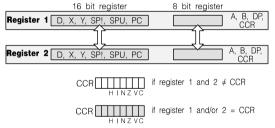
Mot	orola	EXG	<b>;</b>	Fun	ction												þ	P	U	ty	γp	e I	В/	6	В0	9
B&F	1	EXC	,	r₁	⇒r <sub>2</sub>																					
Sho	rt																									
Δ,	Addressing mode				10	Γ		Δ,	14	re		· n	re		عاد		io	'n			0	1	С	P	30	
_ ^`	iui 633	ilig ii	loue /	Орсос	10				<i>.</i> u		30	, 1	,,,	.31	510			""			О	·	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	S	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
7/2	7/2					Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Η	1	N	Z	٧	С
1E																					•	•	•	•	•	•

The contents of the indicated registers are exchanged. Both registers must be the same size; either a 1 byte register (A; B, DP or CCR) or a 2 byte register (D, X, Y, SP!, SPU or PC).

Syntax: EXG r,r—Register 2

The entire instruction consists of two bytes. The first byte is the opcode (instruction) and the second byte (post byte) gives the registers to be exchanged.

		Post	byte
Register 1 / 2	r	Register 1	Register 2
D X Y SPU SPI P C B DP CCR	DXYU!\$ABPC	0000 0001 0010 0011 0100 0101 1000 1001 1011 1010	0000 0001 0010 0011 0100 0101 1000 1001 1011 1010



Attention: This system stack pointer SP! may only be changed if the running of stack operations is thoroughly understood. If the system stack pointer is not changed carefully the result may be a CPU error. This causes a HALT in the CPU; the error message "STACK ERROR" is shown in the status test.

Changing the program counter PC causes an unconditional jump to the address loaded to the PC.

# 3.4. STACK INSTRUCTIONS

Instructions which put the contents of a register (1 or 2 bytes) on a stack or retrieve them from it are covered in this section.

Motorola	B&R	(	Operating	mode	
Motorola	Dak	PG1000	PG-PC	CP 80	PC 80
PSH	PSH				0
PUL	PUL				0
PSHA	ANS	0	0	0	0
PULA	AVS	0	0	0	0
PSHB	BNS	0	0	0	0
PULB	BVS	0	0	0	0
PSHX	RNS	0	0	0	0
PULX	RVS	0	0	0	О

Mot	orola			Fun	ction												c	P	U	ty	/p	е /	۹ /	6	30	3
B&F	ł																									٦
Sho	rt																									١
Δι	dress	sina n	node /	Oncor	łe	Г		Δι	44	re		: r	re	-56	ale	ct	in	n				Ι		<b>G10</b>		╗
		,g	iouc,	Ороос						٠.	-	'		-				•••					Р	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	ĸ		٦
						Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
						Г									П	T										П

Mot	orola	PSH	1	Fun	ction												C	P	U	ty	γp	e E	3 /	68	30	9
B&F	~	PSH			es con							in	dic	at	ec	l r	eg	is	te	r to	0 1	the	9			
Sho	rt			syst	em or	u	se	rs	ta	ck																
Δ,	ddress	ina m	ode /	Oncor	łe.	Γ		Δι	44	re	S 6	: r	re	250	ele	201	tio	'n				Ι	_	P		
^`	au. 000	9	ouc,	Ороос							•	' r			٠.٠			•••			0	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
5+/2						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
34 / 36						Г			П					П			•	•			0	0	0	0	0	0

This instruction is used to put one or more user indicated registers on a stack (system or user).

Syntax:

PSH | s ,r.... Register to be placed on the stack
Stack upon which the register shall be placed

The entire instruction consists of two bytes. The first byte (opcode) gives the stack. The registers to be placed on the stack are defined in second byte (postbyte). If more registers are defined in the postbyte they are placed on the stack in a certain order. The order is shown in the tables below in the direction the arrow points.

	= ! (Opcode = \$34 gister order on a \$	,
Register	r	Postbyte
CCR A B DP X Y SPU PC All except PC	C A B P X Y U \$	xxxxxxx1 xxxxxx1xx xxxxx1xxx xxx1xxxx xx1xxxxx xx1xxxxx x1xxxxx 1xxxxxx

	U (Opcode = \$36 ster order on a S	•
Register	r	Postbyte
CCR A B DP X Y SP! PC All except PC	C A B P X Y! \$ *	XXXXXXXI XXXXXXIXX XXXXIXXXX XXXIXXXX XXIXXXXX XIXXXXXX

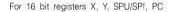
Different registers can be combined in the postbyte.

**Example:** PSH ! ,ABY This instruction causes registers A, B and Y to be placed on the stack.

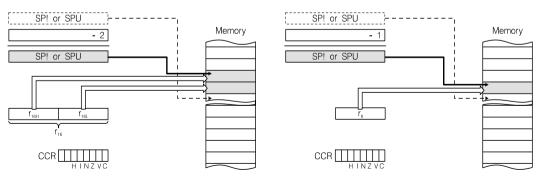
An entry of... is changed by the PROgramming SYStem to...

PSH ! ,\* PSH ! ,CABPXYU
PSH U ,\* PSH U ,CABPXY!
PSH U ,XBC PSH U ,CBX

The following data flow applies for each of the registers given:



For 8 bit registers CCR, A, B, DP



Mot	orola			Fun	ction												(	CP	U	ty	/p	e A	Α/	6	30	3
B&F	₹																									
Sho	rt																									
Δ,	ddroes	sina m	ode /	Oncor	10	Г		Ad	44	rΔ		: r	ırc		ماد	20	tic	'n					P	<b>31</b> 0	000	7
^,	uui 63	sing ii	ioue /	Орсос	40			^	u		30	, ,	,,,	.30	-10			′''				Ι	Р	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		Т
						Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
						Г	Г	Г				Г	Г				П	Г		Г	Г	Г	П			П

Mot	orola	PUL		Fun	ction												C	P	U	ty	γp	e I	3 /	6	В0	9
B&F	t	PUL			d indic		ed	re	eg	ist	eı	· W	/ith	1 (	la	а	fro	on	n t	he	s	ys	te	m	or	
Sho	rt			user	stack																					
Δ,	Addressing mode / Opcode					Γ		Δι	44	re	S 6	: n	re		۱e	·C	tio	n				Ι	_	P		
'''	Addressing mode /										_	, 1				~		•			0	1	Ρ	C	30	
IMPL.	DIR.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R				
5+/2						Ε	Α	М	F	Z	#	Р	С	-	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
35 / 37						Γ											•	•			•	•	•	•	•	•

This instruction is used to load one or more registers with data from a system or user stack.

Syntax: PUL s ,r..... Register with data to be loaded from the stack
Stack to load the data from

The entire instruction consists of two bytes. The first byte (opcode) gives the stack. The registers to be loaded from the stack are defined in second byte (postbyte). If more registers are defined in the postbyte they are loaded from the stack in a certain order. The order is shown in the tables below in the direction the arrow points.

	=! (Opcode = \$3 ad register from \$	•
Register	r	Postbyte
CCR A B DP X Y SPU PC All except PC	C A B P X Y ∪ \$ *	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

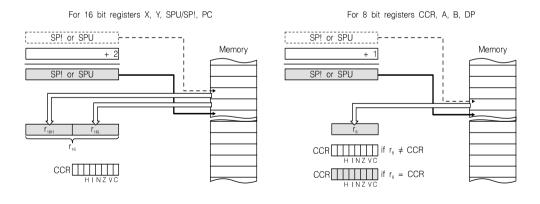
	U (Opcode = \$37 d register from SF	
Register	r	Postbyte
CCR A B DP X Y SP! PC All except PC	C A B P X Y ! \$*	xxxxxxx1 xxxxxxxxx xxxx1xxx xxx1xxxx xx1xxxxx x1xxxxxx

Different registers can be combined in the postbyte.

**Example:** PUL ! , CPX This instruction causes registers CCR, DP and X to be loaded with data from the system stack.

An en	try of	is cha	nged by the PROgramming SYStem to
PUL	! ,*	PSH	! ,CABPXYU
PUL	U ,*	PSH	U ,CABPXY!
PUL	U ,XBC	PSH	U ,CBX

The following data flow applies for each of the registers given:

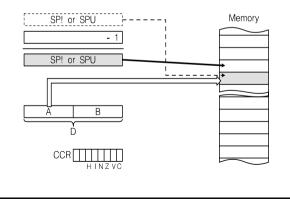


Motorola	PSHA	Function		CPU type A / 6303
B&R	ANS	A ⇒ (SP!);	; SP! - 1 ⇒ SP!	
Short				
				D PC1000

۸,	ddroe	ina n	node /	Oncor	40	ı		A	44	-					٠.,			'n			О	1	PC	<b>G10</b>	)00	'
_ ^'	uuies	sing ii	loue /	Орсос	16			^'	u		33	, ,	,, ,	-31	-10			,,,			О	ī	P	G-F	Š	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		PG-PC CCR			٦
4/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	1	Ν	Z	٧	С
36						Г	Г	Г	Г		П	П	Г	П				П	Г		0	0	0	0	0	0

The contents of accumulator A are saved in the memory position that the system stack pointer SP! points to.

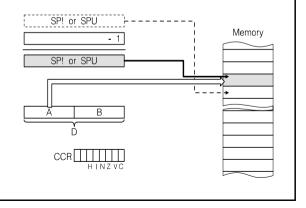
The SP! is then reduced by a value of 1 so that the SP! will again point to the next free location.



Mote	orola	PSH	IA	Fun	ction												c	P	U	ty	pe	e E	3 /	6	в0	9
B&R	l	ANS		SP!	-1 ⇨	SF	?!;	A	٦	<b>&gt;</b> (	S	P!	)													٦
Sho	rt																									
Ac	Addressing mode /				de	Г		Δα	44	re	ss	: n	re	se	ele	ect	io	n			0	1	_	Pδ	_	
,		g		оросс				•		. •	-			•				••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	_	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		٦
6/2						Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	1	N	Z	٧	С
34 02																					0	0	0	0	0	0

Since the system stack pointer SP! points to the last occupied memory position the SP! must first be decreased by the value 1.

The contents of accumulator A are saved to the position in memory that the system stack pointer points to.

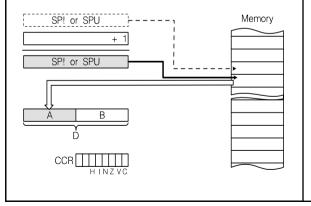


Motorola	PULA	Function	CPU type A / 6303
B&R	AVS	SP! + 1 ⇒ SP!; (SP!) ⇒ A	
Short			

	ddroes	sina n	node /	Oncor	40	Γ		A	44	ro					٠			'n			0	1	P	G1(	000	П
Α.	uuies	sing ii	loue /	Орсос	16	L		^'	u		33	, ,	,, ,	.5	-10	-		,,,			О	•	Р	G-F	SC	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г	PG-PC CCR			٦	
3/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
32						Γ	Γ	Γ	Γ	Γ		Γ	Г	Г					Γ	П	0	0	0	0	0	$\circ$

Since the system stack pointer points to the next free memory location in the stack the SP! must first be increased by a value of 1.

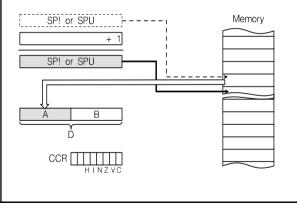
Accumulator A is then loaded with the contents of the memory location pointed to by the SP!.



Mote	orola	PUL	Α.	Fun	ction												C	P	U	ty	/p	e I	3 /	6	В0	9
B&R	ł	AVS		(SP	) <b>⇒</b> A	; 5	SP	! -	+ 1	١٤	> :	SF	P!													7
Sho	rt																									
Δ	Addressing mode / Opco					Г		Δ	44	rΔ		: n	re		عاد		tio	n			0	1	С	P	30	$\Box$
		,g	Ороос						٠٠	•	, ,			•••			•••			0	1	Ρ	C	30		
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	CR		٦
6/2						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
35 02						Г	Г	Г	Г			П		П					П		0	0	0	0		

Since the system stack pointer SP! points to the last occupied memory location accumulator A is loaded with the contents of the memory location pointed to by the SP!.

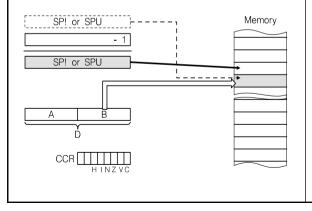
The SP! is then increased by a value of 1 so that the SP! will point again to the last occupied memory location of the stack.



Mot	orola	PSF	ΙВ	Fun	ction												C	CP	U	ty	/p	e .	A A	6	30	3
B&F	₹	BNS	S	В⊳	(SP!)	; 8	SP	! -	1	Ξ	> 5	SP	!													
Sho	rt																									
Δ,	Addressing mode /				40	Γ		Δ,	44	ro		. r	ore		ماد	201	Hic	'n			С	•	P	G1(	000	)
ť	uuies	ing ii	ioue /	Орсос	46	L			<i>a</i> u		30	, ŀ	,,,	.31	-	-		<b>,,,</b>			С	)	Ρ	G-I	C	
IMPL.	IMPL. DIR. EXT. IMMED. IND. RE								s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	CR		_
4/1	4/1					Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	Ν	Z	٧	С
37																					0	0	0	0	ာ	0

The contents of accumulator B will be saved to the location in memory which the system stack pointer SP! points to.

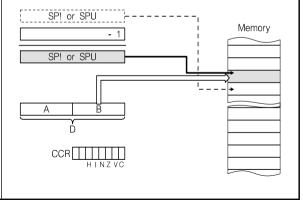
The SP! is then decreased by a value of 1 to enable the SP! to point again to the next free location in memory.



Mot	orola	PSH	ΙB	Fun	ction												c	P	U	ty	'n	e I	3 /	6	30	9
B&F	t	BNS		SP!	-1 ⇨	SF	?!;	В	3 5	> (	S	P!	)													П
Sho	rt																									
Δ,	Addressing mode / Opcode					Г		Δ,	44	re	•	: n	re	250	ale	·ci	io	n			0	1	С	P	30	
l ^``	au. 000	g	.ouc,	Ороос							-	'						•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	_	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		П
6/2						Е	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
34 04						П															0	0	0	0	0	0

Since the SP! points to the last occupied memory location in the stack the system stack pointer SP! must first be decreased by a value of 1.

The contents of accumulator B will then be saved to the position which the system stack pointer SP! points to.

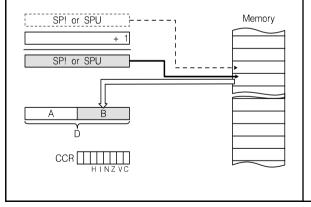


Motorola	PULB	Function	CPU type A / 6303
B&R	BVS	SP! + 1 ⇒ SP!; (SP!) ⇒ B	
Short			

Λ.	ddroes	ina m	ode /	Oncor	40	Г		۸,	44	ro					٠			'n			0	T	P	<b>G10</b>	00	П
Α.	uuies	sing ii	ioue /	Орсос	16			~	u		33	, ,	,, e	.5	-10	-		"			О	7	Р	G-F	,C	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	CCR   CCR									٦	
3/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
33						Г	П	П	Г		П	П	П	П					П		0	0	0	0	0	ା

Since the system stack pointer points to the next free memory location in the stack the SP! must first be increased by a value of 1.

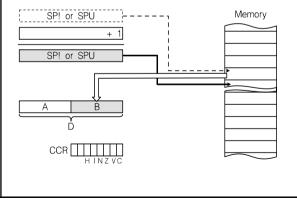
Accumulator B is then loaded with the contents of the memory location pointed to by the SP!.



Mot	orola	PUL	В	Fun	ction												c	P	U	ty	/p	e I	3 /	6	В0	9
B&F	ł	BVS		(SP	) <b>⇒</b> B	; 5	SP	! -	+ 1	١٤	<b>&gt;</b> :	SF	P!													П
Sho	rt			/ Opcode Address presolection O CP80															1							
Δ,	drace	ina m	ode /	Opcode Address preselection O CP80														30								
l ^`	aui 633	ing ii	oue,	Орсос	40			_,	uu		30	, 1	,,,	.30	,,,			•••			О	ı	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	ļ	В	G	Г		C	CR		
6/2						Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
35 04						Г	Г	Г	Г	Г	П		Г							П	0	0	0	0	0	0

Since the system stack pointer SP! points to the last occupied memory location accumulator B is loaded with the contents of the memory location pointed to by the SP!.

The SP! is then increased by a value of 1 so that the SP! will point again to the last occupied memory location of the stack.



Motorola	PSHX	Function	CPU type A / 6303
B&R	RNS	X <sub>L</sub> ⇒ (SP!); SP! - 1 ⇒ S	P!
Short		X <sub>H</sub> ⇒ (SP!); SP! - 1 ⇒ S	P!
			O PG1000

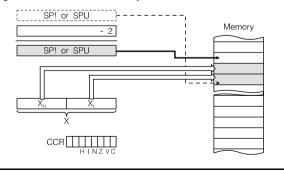
	ddrae	eina n	node /	Onco	40			Λ.	44	ro					n۱	ec.	fi,	'n			_	1				_
	uuies	sing ii	loue /	Орсо	ue			^'	u		33	, ,	,, ,	-31	-10	-6		,,,			О	ī	P	G-F	'n	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		٦
5/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	1	Ν	Z	٧	С
3C						Γ	Π														0	0	0	0	0	0

The contents of index register X will be placed in the system stack. Since the system stack pointer always points to the next available memory location in the stack the contents of index register X will be saved as follows:

1)  $X_i \Rightarrow (SPI)$ 

- X<sub>L</sub> ⇒ (SP!)
   SP! 1 ⇒ SP!
- 3) X<sub>H</sub> ⇒ (SP!)
- 4) SP! 1 ⇒ SP!

After the instruction is executed the system stack pointer SP! points again to the next available memory location in the stack.



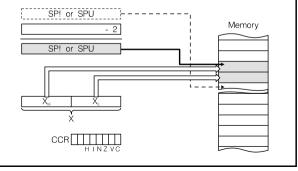
Mot	orola	PSF	IX	Fun	ction												C	P	U	ty	/p	e I	3 /	68	30	Э
B&F	ł	RNS		SP!	-1 ⇨	SI	P!;	X	<u>_</u> 1	⇒	(S	P	!)													
Sho	rt			SP!	Ocode Address preselection O CP8																					
Δ,	drace	ina m	ode /	Oncor	ncode I Address preselection IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII														30							
	aui 633	ing ii	oue,	Орсос	IO CP														30							
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	ļ	В	G	Г		CC	R		٦
7/2						Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
34 10						Г	П	П	П						П						0	0	0	0	0	0

The contents of index register X will be placed in the system stack. Since the system stack pointer SP! points to the last occupied memory location in the stack the contents of index register X are saved as follows:

1) SP! - 1 ⇒ SP!

- 2) X<sub>1</sub> ⇒ (SP!)
- 3) SP! 1 ⇒ SP!
- 4) X<sub>H</sub> ⇒ (SP!)

After the instruction is executed the system stack pointer SP! points again to the last occupied memory location in the stack.



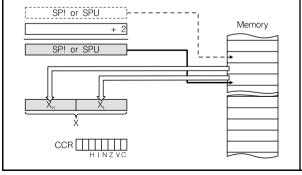
Motorola	PULX	Function	CPU type A / 6303
B&R	RVS	SP! + 1 ⇒ SP!; (SP!) ⇒ X <sub>H</sub>	
Short		SP! + 1 ⇒ SP!; (SP!) ⇒ X <sub>1</sub>	

۸.	ddroe	eina n	node /	Oncor	40	ı		۸.	44	ro					٠.,			ion   PG1000   PG-PC   U ! B G   CCR   U ! B G   H   N   Z   V		'						
^	uuies	sing ii	loue /	Орсос	16			^'	u		33	, ,	,, e	.5	-10			"			О	ī	P	G-F	'n	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г	CCR				٦
4/1						Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
38						Г	Π														0	0	0	0	0	$\circ$

Index register X will be loaded with data from the SP! system stack. Since the system stack pointer SP! always points to the next available location in the stack. X will be loaded from the stack as follows: SPI + 1 ⇒ SPI

- - $(SP!) \Rightarrow X_H$
- 3 SPI + 1 ⇒ SPI
- (SP!) ⇒ X<sub>1</sub>

After the instruction is executed the system stack pointer SP! points again to the next available memory location in the stack.

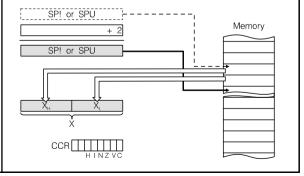


Mote	orola	PUL	.X	Fun	ction												þ	P	U	ty	/p	e E	3 /	6	80	9
B&R	}	RVS	;	(SP	) ⇔ X,	;	SI	P!	+	1	⇔	s	P!													
Sho	rt			(SP!) ⇒ X <sub>L</sub> ; SP! + 1 ⇒ SP!																						
Δ,	ldraes	ina m	ode /	Opcode Address preselection O CP														P	30							
_^`	iui 638	ning ii	oue,	Орсос	10				uu		30	, 1	,, ,	.31	510			···			О	ŀ	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		7
7/2						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
35 10						Г	Г		П			Г	Г								0	0	0	0	0	0

Index register X will be loaded with data from the SP! system stack. Since the system stack pointer always points to the last occupied location in the stack. X is loaded from the stack as follows:

- (SP!) ⇒ X<sub>H</sub> SP! + 1 ⇒ SP!
- (SP!) ⇒ X<sub>L</sub> SPI + 1 ⇒ SPI

After the instruction is executed the system stack pointer SP! points again to the last occupied memory location in the stack.



## 3.5. LOGICAL FUNCTIONS

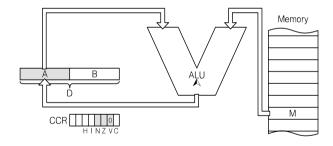
This section describes all instructions which combines the contents of registers and/or memory locations with a logical operation and saves the result.

Motorola	B&R	(	Operatin	g mode	
Wotorola	Dak	PG1000	PG-PC	CP 80	PC 80
ANDA	UND	0	0	0	0
ANDB	UB	0	0	0	0
AIM	AIM		0		
ORAA	OD	0	0	0	0
ORAB	ОВ	0	0	0	0
OIM	OIM		0		
EORA	EXO	0	0	0	0
EORB	EB	0	0	0	0
EIM	EIM		0		

Mot	orola	ANI	DA	Fun	ction												C	CP	U	ty	γp	e A	Α /	6	30	3
B&F	ł	UNE	)	Ал	(M) ⊏	•	١																			
Sho	rt	υ																								
	ddroes	ina m	ode /	Oncor	10	Г		۸,	14	rn			re		٠			'n			0	1	P	<b>G1</b> (	000	)
^	Julesa	sing ii	ioue /	орсос	16			^'	Ju		33	۰,	,, e	.5	-10	,,,		""			О	ŀ	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		П
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	Ν	z	٧	С
	94	B4	84	A4		0	0	0	0	0	0	0	0	0							0	0	•	•	•	0

Mot	orola	AND	PΑ	Fun	ction												þ	P	U	ty	/p	e E	3 /	6	В0	9
B&F	ł	UNE	)	Ал	(M) =	> 4	١																			
Sho	rt	U																								
A	dress	ina m	ode /	Oncor	de	Γ		Δα	łН	re	ss	: n	re	s	ele	c	tic	n			О	1		P	_	Ξ
ļ ,,,		9		-рос.	A ∧ (M) ⇒ A  Opcode Address preselect																О	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	S	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	R		Т
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	N	z	٧	С
	94	B4	84	A4		Э	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

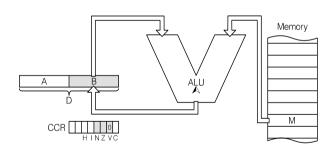
Accumulator A will be combined with the contents of memory location M through the use of an AND operation. The result will be stored in A.



Mot	orola	ANI	DВ	Fun	ction												C	CP	U	ty	p	e A	Α/	6	30	3
B&F	₹	UB		Вл	(M) ⊏	> E	3																			
Sho	rt																									
۸,	ddroes	ina n	node /	Oncor	10	Γ		۸,	44	rn			ore		٠		+i.c	'n			0	1	P	<b>G1</b> (	000	)
Α.	uui ess	sing ii	loue /	Орсос	16			^'	uu		33	۰,	,, ,	.51	-10			""			a	ŀ	Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	D4	F4	C4	E4		Ь	0	0	0	0	0	0	0	0		Γ	П				0	0	•	•	▼	0

Mot	orola	AND	В	Fun	ction												C	P	U	ty	/p	e E	3 /	68	В0	9
B&R		UB		Вл	(M) =	, E	3																			
Sho	rt																									
Δ	drass	ina m	ode /	Oncor	40	Γ		Δ,	44	rΔ		s p	re		عاد		Hio	'n			0	ī	С	Pε	30	
Υ΄	Jui 633	iiig ii	loue / ·	орсос	16			~(	ıu	16	33	, h		30	216	-		""			О	Ī	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R	_	_
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	-	Υ	D	U		В	G	Η	I	N	Z	٧	С
	D4	F4	C4	E4		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

Accumulator B will be combined with the contents of memory location M through the use of an AND operation. The result will be stored in B.

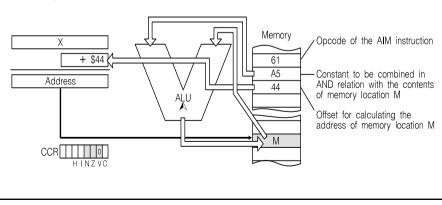


Mot	orola	AIM		Fun	ction												C	P	U	ty	/p	e A	A /	6	30	3
B&F	₹	AIM		(X +	Offse	t)	٨	IN	1M	١٤	⇒	(X	+	С	ff	se	t)									
Sho	rt																									
Δ,	ddroes	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		· r	ırc		ماد	ect	Hio	'n				Τ	P	G10	000	,
	uui 633	ing ii	ioue /	Орсос	16			^'	uu			, ,	,,,	.31	-10	-		•••			С	,	Ρ	G-F	S	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G		П	C	CR		٦
				7/3		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
				61							0										0	0	•	•	•	0

Mot	orola			Fun	ction												C	P	U	ty	/p	e I	В/	68	30	9
B&F	ł																									
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	40	Γ		Δ	44	re		: n	re		عاد		Hio	'n				Τ	С	Pε	30	
_^`	aui 633	iiig ii	oue,	Орсос	46	L		^'	u		30	, 1		.30	,,,			•••					Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	ļ	В	G	Г		CC	R		
						Ε	Α	М	F	Z	#	Р	С	Ι	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
						Γ	Г	Г				Г											Г			П

An immediate value \$xx (constant) given in the instruction will be combined with the contents of memory location M in AND relation. The result will be stored in the same memory location. The address of M is attained from the sum of index register X and the offset \$yy which must be given with the instruction.

Example: AIM # \$A544

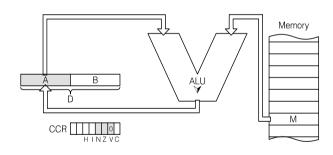


Motorola	ORAA	Function		CPU type	e A / 6303
B&R	OD	A ∨ (M) =	A		
Short	0				
Address	ing mode /	Oncode	Address preselection	on 0	PG1000

	ddroes	ina n	node /	Oncor	40	Г		۸,	44	ro	ss				٠.,			'n			O	7	P	<b>31</b> (	000	П
	uuies	sing ii	loue /	Орсос	16	L		^'	u		33	, ,	,, ,	-3	-10	-		,,,			O	•	Р	G-F	S	П
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	ĸ		٦
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	1	Ν	Z	٧	С
	9A	BA	8A	AA		0	ာ	0	0	0	0	ာ	0	0							0	0	•	•	•	0

Mot	orola	OR/	λA	Fun	ction												c	P	U	ty	ďρ	e E	3 /	68	30	9
B&F	ł	OD		Αv	(M) =	> /	4																			
Sho	rt	0																								
A	dress	ina m	ode /	Oncor	de	Γ		Δι	44	re	SS	: n	re	S	ele	ct	io	n			0	1	_	Pδ	_	
'"		9		орос.		L		•		. •	-	, 1				•		•••			0	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		Ī
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
	0.0	DΛ	0.0	ΔΔ.		5		$\overline{}$	$\overline{}$			$\overline{}$	$\overline{}$		_		_	_	•	•		$\overline{}$	•	•	_	$\overline{}$

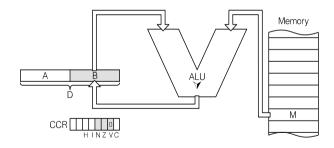
Accumulator A will be combined in OR relation with the contents of memory location M. The result will be stored in A.



Mot	orola	ORA	AΒ	Fun	ction												(	CP	U	ty	γp	е /	Α/	6	30	13
B&F	₹	ОВ		Вv	(M) ⊏	• E	3																			
Sho	rt																									
Δ,	ddress	ina m	node /	Oncor	łe	Γ		Δι	44	re		: r	ore	250	عاد	201	tic	'n			0	1		_	000	_
	uuiesa	ing ii	ioue /	Орсос	10	L			<i>a</i> u		30	, ŀ	,,,	.31	-			<b>,,,</b>			О	•	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		П
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	Ν	z	٧	С
	DA	FA	CA	EA		0	0	0	0	0	0	0	0	0							0	0	•	•	•	0

Mot	orola	ORA	AΒ	Fun	ction												þ	P	U	ty	/p	e I	В/	6	80	9
B&F	ł	ОВ		В∨	(M) =	> E	3																			
Sho	rt																									
Λ.	droce	ina m	node /	Oncor	10	Γ		۸,	14	rn		s p			٠		Hio	'n			0	7	С	P	30	
Α.	Jui 638	ilig ii	loue /	Орсос	16			~	Ju		33	, h	,, ,	.5	-16	-		""			О	<b>)</b>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
	DA	FA	CA	EA		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	ာ	•	•	•	0

Accumulator B will be combined in OR relation with the contents of memory location M. The result will be stored in B.

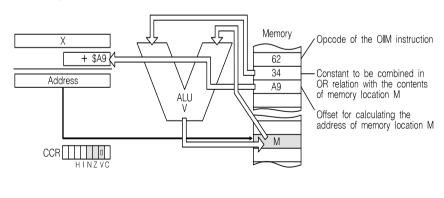


Mot	orola	OIM	l	Fun	ction												(	P	U	ty	/p	e .	A /	6	30	3
B&F	₹	OIM	l	(X +	Offse	t)	v	IN	IN	١٤	⇒	(X	+	С	ff	se	t)									
Sho	rt																									-
Δ	ddroes	ina n	node /	Oncor	10	Γ		Δ	44	rΔ		. r	re		ماد	200	Hic	'n				Ι	P	G1(	000	$\Box$
ť	uui 638	ing ii	ioue /	Орсос	16				<u> </u>		30	, ,	,, ,	.31	510	,,,		•			U	)	Ρ	G-I	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	CR		П
				7/3		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
				62		Γ	Π		П		0										0	0	•	•	▼	)

Mote	orola			Fun	ction												C	P	U	ty	/p	e E	3 /	68	80	9
B&R	ł																									
Sho	rt																									
Δ,	ddress	ina m	node / (	Oncor	de	Γ	_	Δ,	44	lre:			re		عاد	-	tic	'n				Ι	_	Pε		
	101 COO	g	ouc,	орсос	10	L		_	-			۰, ۲			310			<b></b>					P	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.		0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R	_	
						Е	Α	М	F	z	#	Р	С	П	Υ	D	U	!	В	G	Н	T	N	Z	٧	С
						Γ	Г	Г	Г	П	П	П	П	П		П			П		П	П	П			Г

An immediate value \$xx (constant) given in the instruction will be combined with the contents of memory location M in OR relation. The result will be stored in the same memory location. The address of M is attained from the sum of index register X and the offset \$yy which must be given with the instruction.

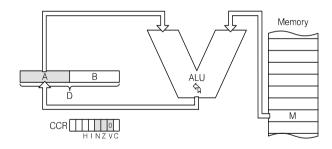
Example: OIM # \$34A9



Mot	orola	EO	₹A	Fun	ction												ľ	٦٢	'U	ty	/p	e .	Α/	6	30	3
B&F	₹	EXC	)	AΦ	(M) =	> /	4																			
Sho	rt	Е																								
	ddroes	ina n	node /	Oncor	40	Γ		۸,	44	rn	ss				٠			'n			O	)	P	<b>31</b> (	000	$\Box$
Α.	uuies	sing ii	loue /	Орсос	16			^'	u		33	٠,	,, e	.5	-10	-		""			O	•	Ρ	G-F	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Η	ı	N	Z	٧	С
	98	B8	88	A8		b	0	0	0	0	0	0	0	0							0	0	•	•	•	0

Mot	orola	EOF	RA	Fun	ction												þ	P	U	ty	ďρ	e E	3 /	6	В0	9
B&F	ł	EXC	)	ΑФ	(M) =	> ,	Α																			
Sho	rt	E																								
۸,	droce	ina m	ode /	Oncor	10	Γ		۸,	14	rn		s p			N/		Hio	'n			0	7	С	Pδ	30	
Α.	Jui 638	ilig ii	ioue /	Орсос	16			~	Ju		33	, h	,, ,	-30	216	-		""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	98	В8	88	A8		b	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

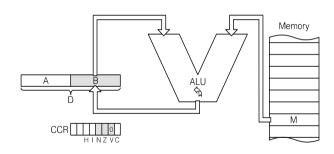
Accumulator A will be combined in EXCLUSIVE-OR relation with the contents of memory location M. The result will be stored in A.



Mot	orola	ola	EOF	RB	Fun	ction												С	Pl	J	y	pe	A / 630	3
B&F	₹		EB		В⊕	(M) =	۱ (	В																
Sho	rt																							
۸,	ddroe	lrocc	ina m	ode /	Oncor	40	Γ		۸,	14	ra			ro		٠	ect	i۸	n		T	0	PG1000	)
Α.	uui es	11 633	mg m	ioue /	Орсос	16			^(	Ju	16	33	۰,	,, ,	.5	-16	,,,,	10	"			0	PG-PC	
IMPL.	DIR.	DIR.	EXT.	IMMED.	IND.	REL.	Ī	0	F	s	Т	#	Ρ	R	Х	Υ	D	υ	! [	3 (	3		CCR	
	0.10	0.00					1	١.						_			_			_	_			

Mot	orola	EOF	≀В	Fun	ction												C	P	U	ty	/p	e E	3 /	6	В0	9
B&F	₹	ЕВ		В⊕	(M) =	>	В																			
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	łe	Γ		Δ	чч	re		: r	ıre	250	عاد		tic	'n			0	_		P	_	
l ^``	uu. 000	g	ouc,	Ороос		L			uu	٠.	•	'			•••			•••			О	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		
	4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
	D8	F8	C8	E8		О	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

Accumulator B will be combined in EXCLUSIVE-OR relation with the contents of memory location M. The result will be stored in B.

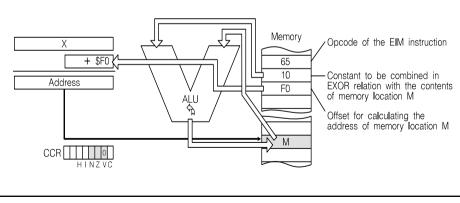


Mot	orola	EIM		Fun	ction												(	CP	U	ty	/p	e .	A A	6	30	3
B&F	₹	EIM		(X +	Offse	t)	Ф	IN	٨N	1	⇔	()	( 4	. C	Off	se	t)									
Sho	rt																									
Δ,	ddroes	ina n	node /	Oncor	10	Г		Δ,	44	ro		· r	ore		عاد	c	ic	'n				T	Р	G1(	000	$\Box$
^`	uui 638	ing ii	ioue /	Орсос	16				uu		30	, ,	,,,	.30	,,,			<b>,,,</b>			U	)	Р	G-I	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	I	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		П
				7/3		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
				65							0										0	0	•	•	•	0

Mot	orola			Fun	ction												C	P	U	ty	ďρ	e I	В/	68	30	9
B&F	₹																									
Sho	rt																									
Δ,	ddraes	ina m	ode /	Oncor	40	Γ		Δ	44	re	٠.	: n	rc		عاد		Hio	'n				Τ	С	Pε	30	
^,	uui 633	ing ii	ioue /	Орсос	46	L		^'	uu		30	, 1	,,,	.30	,,,			•••					Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		
						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
						Г	Г	Г	П		Г	Г	П	П									Г			П

An immediate value \$xx (constant) given in the instruction will be combined with the contents of memory location M in EXCLUSIVE-OR relation. The result will be stored in the same memory location. The address of M is attained from the sum of index register X and the offset \$yy which must be given with the instruction.

Example: EIM # \$10F0



## 3.6. ARITHMETIC INSTRUCTIONS

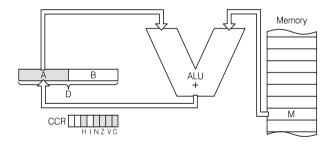
This section describes all instructions used to add, subtract or mutiply two operands.

		(	Operatin	g mode	
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
ADDA ADCA	+ ADD	0	0	0	0
ADDB ADCB	+B ++B	0	0	0	0
ADDD	+D	0	Ö	0	0
ABA SUBA	A+B -	0	0	0	0
SBCA	SUB	0	0	0	О
SUBB SBCB	-B B	0	0	0	0
SUBD SBA	-D A-B	0	0	0	0
ABX MUL	B+R A*B	0	0	000	000

Mote	orola	ADI	DA	Fun	ction												C	CP	U	ty	p	e A	<b>A</b> /	6	30	3
B&R	ł	+		A +	(M) ⊏	<b>,</b>	١																			
Sho	rt																									
	ddroes	ina m	node /	Oncor	10	Γ		۸,	14	rn			re		N/a			'n			0	7	P	<b>G1</b> (	000	,
	Julesa	sing ii	loue /	Орсос	16	L		^'	Ju		33	۰,	,, e	-30	316			""			a	ŀ	P	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	I	Ν	z	٧	С
	9B	BB	8B	AB		0	0	0	0	0	0	0	0	0							•	0	•	•	•	•

Mot	orola	ADE	PΑ	Fun	ction												þ	P	U	ty	/p	e I	В/	6	80	9
B&F	t	+		A +	(M) =	> 4	١																			
Sho	rt																									
Δ,	drace	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		s p	rc		عاد	c	io	'n			О	7	С	P	30	
^	aui ess	iiig ii	oue,	Орсос	10	L			<i>a</i> u		30	, 1	,, ,	.30	,,,			···			О	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
	9B	BB	8B	AB		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	•	ာ	•	•	•	•

The contents of accumulator A and memory location M will be added. The result will be stored in accumulator A.

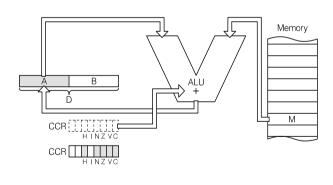


Motorola	ADCA	Function	CPU type A / 6303
B&R	ADD	A + (M) + C ⇒ A	
Short	A, ++		
			D PC1000

	ddroes	ina m	ode /	Oncor	40	Г		۸,	44	ro	ss				٠.,		41,	'n			O	)	P	<b>310</b>	)00	,
^'	uuies	sing ii	ioue /	Орсос	16			^'	u		33	, ,	,, e	.5	-10			,,,			O	7	P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	ĸ	Т	٦
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	Τ	Ν	Z	٧	С
	99	В9	89	A9		0	ာ	0	0	0	0	ာ	0	0							•	)	•	•	•	•

		_															_									_
Mot	orola	ADO	CA	Fun	ction												K	P	U	ty	/p	e I	В/	6	80	9
B&F	₹	ADI	)	A +	(M) +	С	П	<i>,</i>	١																	
Sho	rt	A, +	+																							
Δι	ddress	ina m	ode /	Oncor	łe	Γ		Δι	44	re		s p	re	250	عاد	20	tio	'n			O	<u> </u>	_	P		
	uu. 000	g	.ouc,	Ороос		L				٠.	•	, ,			٠.、			•••			С	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		_
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
	99	B9	89	A9		Ь	0	0	0	0	0	0	0	0	•	•	•	•	•	•	•	0	•	•	•	•

The contents of accumulator A, memory location M and the carry flag (possible from previous addition) will be added. The result will be stored in accumulator A.



WOT	oroia	ADL	סכ	run	ction													, P	U	τy	p	e /	4 /	0	ას	ა
B&F	₹	+B		B +	(M) =	E	3																			
Sho	rt																									
	ddroor	ina n	node /	Onco	10	Г		۸.	44				re								0	ī	P	<b>G1</b> (	000	)
A	uures	sing ii	ioue /	Opcoo	ie			Α,	Ju	16	53	, ŀ	,, e	:51	316	:0	···	,,,			О	1	Р	G-F	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		П
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
	DB	FB	CB	EB		0	ာ	0	0	0	0	ာ	0	0							•	0	•	•	•	•

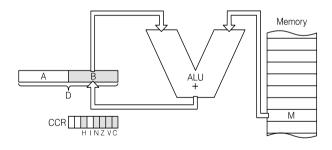
Eupotion

Motorolo ADDR

Mot	orola	ADI	В	Fun	ction												þ	P	υ	ty	/p	e I	3 /	6	80	9
B&F	ł	+B		B +	(M) =	<b>E</b>	3																			
Sho	rt																									
Δι	dress	ina m	ode /	Oncor	łe.	Γ		Δ,	44	re		s p	re		ale	20	tic	'n			О	_	_	P		
		g	.ouc,	Ороос		L				٠.	•	, ,		-	•••			•••			О	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	o	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		_
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Ρ	С	1	Υ	D	U		В	G	Н	ı	N	Z	٧	С
	DB	FB	CB	EB		Э	ာ	0	0	0	0	ာ	0	0	•	•	•	•	•	•	•	ာ	•	•	•	•

The contents of accumulator B and memory location M will be added. The result will be stored in accumulator B.

CBILture A / 6202



																	_ [*		_	-,	-			_		-
B&F	₹	++B		B +	(M) +	С	П	> E	3																	
Sho	rt																									
	ddroes	ina m	node /	Oncor	40	Г		۸,	44	ro					٠		tic	'n			0	1	P	<b>G1</b> (	000	$\Box$
Α.	uuies	sing ii	loue /	Орсос	16			^(	u		33	۰,	,, e	531	-10	-		""			О	ŀ	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	z	٧	С
	D9	F9	C9	E9		0	0	0	0	0	0	0	0	0							•	0	•	•	•	•

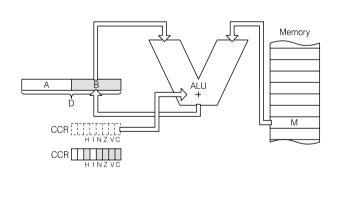
Function

Motorola ADCB

Mot	orola	ADO	В	Fun	ction												C	P	U	ty	ďρ	e E	3 /	6	80	9
B&F	ł	++B		B +	(M) +	С	П	• E	3																	
Sho	rt																									
Λ.	droce	ina m	ode /	Oncor	10	Γ		۸,	14			s p			N/		Hio	'n			0	7	С	P	30	
Α.	Jui 638	ilig ii	ioue /	Орсос	16	L		~	Ju	16	33	, h	16	-30	216			""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U		В	G	Ι	I	N	Z	٧	С
	D9	F9	C9	E9		િ	0	0	0	0	0	0	0	0	•	•	•	•	•	•	•	0	•	•	•	•

The contents of accumulator B, memory location M and the carry flag (possible from previous addition) will be added. The result will be stored in accumulator B.

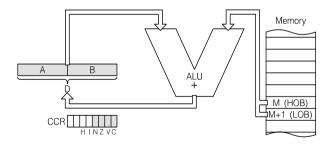
CPU type A / 6303



Mot	orola	ADL	)D	Fun	ction												C	P	U	ty	p	е /	4 /	6	30	3
B&F	₹	+D		D+	(M:M+	-1	) 1	⇒	D																	
Sho	rt																									
	ddroes	ina m	node /	Oncor	10	Γ		۸,	44	ro			re					'n		П	Q	T	P	<b>G1</b> (	000	
^'	uui ess	sing ii	loue /	Орсос	16			^'	u		33	۰,	,, e	.5	216	-		""			a	1	Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3	3/3	5/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	D3	F3	C3	E3		Г	П				0	0	0	0						П	Э	0	•	•	•	•

Mot	torola	ADI	OD	Fun	ction												¢	P	U	ty	/p	e I	3 /	6	80	9
B&I	R	+D		D+	(M:M+	-1	) [	⇒	D																	
Sho	ort			1																						
Δ	ddress	sina m	node /	Oncor	de de	Γ		Δι	44	re		s p	re	250	عاد	· C1	tic	'n			C	_		P		
(	uu. coc	,g	iouc,	Орос	40	L					-	, ,						•••			С	<u> </u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		
	6/2	7/3	4/3	6+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	Ν	Z	٧	С
	D3	F3	C3	E3		Γ					0	0	0	0	•	•	•	•	•	•	ા	0	•	•	•	•

The contents of accumulator D, memory location M and M+1 will be added. The result will be stored in accumulator D.

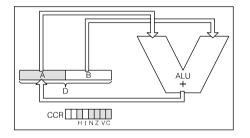


Motorola	ABA	Function	CPU type A / 6303
B&R	A+B	A+B ⇔ A	
Short			

Λ.	Addressing mode / Opcode Address preselection										'n			0	1	P	<b>G10</b>	)00	П							
Α.	uui es	sing ii	ioue /	Орсос	16			^'	u		33	, 1	,, e	.5	516						О	•	Р	G-F	'n	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	ĸ	Т	٦
1/1						Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
1B						Г	П	Г	Г	Г	П		П	П	П						•	0	•	•	•	•

Mot	orola	ABA	١	Fun	ction												C	P	U	ty	ďρ	e E	3 /	6	80	9
B&F	ł	A+E	3	A +	В ⇒ /	4																				
Sho	rt																									
Δι	dress	ina m	ode /	Oncor	de de	Γ		Δι	44	re		s p	re		۱e	·C	tio	'n			0	1		P	_	
	CO.	,g	ouc,	Ороос		L					-	, ,		-	•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		Τ
		25/3				Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U		В	G	Η	I	N	Z	٧	С
		BD FF 90				Г	Г	Г	Г			Г	П							П	•	0	•	•	•	•

The contents of accumulators A and B will be added. The result will be stored in accumulator A.



Since this instruction is not available in the 6809 the utilization of a sub-program in the operating system is required. This sub-program which simulates this instruction consists of three instructions:

PSH ! ,B The contents of B are placed on the stack:

Stack:

SP! - 1 ⇒ SP!

B ⇒ (SP!)

ADDA ! ,+ The contents of A and the memory

location which the system stack pointer points to are added. SP! is the increased

by 1. (Post increment):

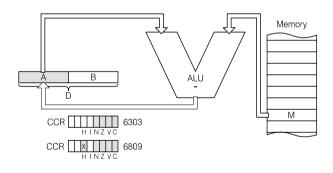
A + (SP!) ⇒ A SP! + 1 ⇒ SP!

RTS Leaves the sub-program.

Mot	orola	SUE	3A	Fun	ction												(	ЭP	U	ty	р	e /	Α/	6	30	3
B&F	₹	-		A - (	(M) ⇒	Α																				
Sho	rt																									
	ddroes	ina m	ode /	Oncor	10	Γ		Δ	44	ro		: r	re		ماد	201	tic	'n			0	1	P	<b>31</b> 0	000	)
	uui 638	ing ii	ioue /	Орсос	10				uu		30	, ,	,, ,	.31	-			<b>,,,</b>			a	•	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	Ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	н	Ι	N	Z	٧	С
	90	В0	80	A0		Э	0	0	0	0	0	0	0	0			П			П	Э	0	•	•	•	•

_																											
	Mot	orola	SUE	ЗА	Fun	ction												C	P	U	ty	/p	e E	3 /	68	В0	9
ı	B&F	₹	-		A - (	(M) ⇒	Α																				
	Sho	rt																									
	Δ,	ddraes	ina m	ode /	Oncor	40	Γ		Δ,	44	rΔ		s p	ro		عاد	c	·io	'n			0	ī	С	Pε	30	
	^	uui 633	ing ii	ioue /	Орсос	46	L			<i>a</i> u		30	, P		.30	510			···			О	)	P	CE	30	
	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
		4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	-	Υ	D	U		В	G	Η	I	Ν	Z	٧	С
		90	В0	80	A0		ि	0	ာ	ာ	0	0	0	0	0	•	•	•	•	•	•	×	0	•	•	•	•

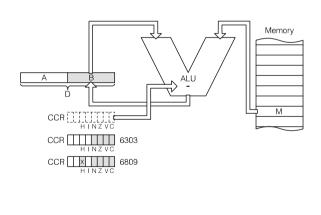
The contents of memory location M will be subtracted from accumulator A. The result will be stored in accumulator A.



Mot	orola	SBC	CA	Fun	ction												C	P	U	ty	p	е /	Α/	6	30	3
B&F	₹	SUI	В	A - (	M) - C	; 1	⇒	Α																		
Sho	rt																									
۸,	ddroes	ina n	node /	Oncor	10	Γ		۸,	44	rn			re				Hin	'n			Q	T	P	<b>G1</b> (	000	)
^(	uuies	sing ii	loue /	Орсос	16			~	uu		33	, ,	,, e	30	316	-		""			Q	7	Р	G-F	Š	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	92	B2	82	A2		Ь	0	0	0	0	0	0	0	0			П				Э	0	•	•	•	•

	Mot	orola	SBC	CA	Fun	ction												¢	CP	U	ty	/p	e I	3 /	6	В0	9
ı	B&F	1	SUE	3	A - (	M) - C	: ;	⇨	Α																		
ı	Sho	rt																									
ı	Δι	dress	ina m	ode /	Oncor	1e	Γ		Δι	44	re		s p	re		عاد	20	tic	'n			C	1		P		
ı			g	.ouc,	Ороос		L					•	, ,			٠.、			•••			C	)	Ρ	C	30	
ı	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	CR		
ı		4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
ı		92	B2	82	A2		Э	0	0	0	0	0	0	0	0	•	•	•	•	•	•	×	0	•	•	•	•

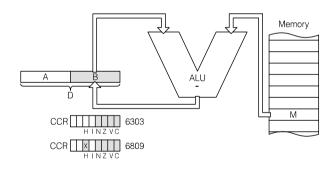
The contents of memory location M and the carry flag will be subtracted from accumulator A. The result will be stored in accumulator A.



Mot	orola	SUE	ВВ	Fun	ction												C	CP	U	ty	p	е /	Α/	6	30	3
B&F	₹	-В		В-(	(M) ⇒	В																				
Sho	rt																									
	ddroes	ina m	ode /	Oncor	10	Γ		۸,	44	rn			re		٠			'n			0	T	P	<b>G1</b> (	000	
	uui ess	sing ii	ioue /	орсос	16			^'	uu		33	۰,	,, ,	30	-16	-		""			a	1	Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Ρ	С	_	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	D0	F0	C0	E0		Э	0	0	0	0	0	0	0	0							)	0	•	•	•	•

Mot	orola	SUE	3B	Fun	ction												r	ìP	u	f١	m	e l	В/	6	80	9
B&F		-В			(M) ⇒	В												-	_	-,	-	_		_		_
Sho	rt																									
Ad	dress	sina m	node /	Oncor	de	Γ		Δα	hh	re	SS	s p	re	s	ele	c	tic	n			О	-	_	P8		
		g	,	орос.				•		. •	-	- 1		_	•	-		•••			О	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
	D0	F0	C0	E0		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	×	0	•	•	•	•

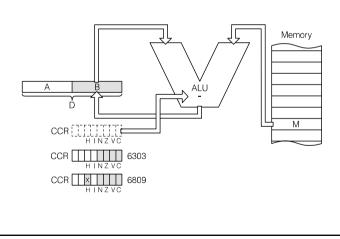
The contents of memory location M will be subtracted from accumulator B. The result will be stored in accumulator B.



Mot	orola	SBC	СВ	Fun	ction												C	CP	U	ty	ďρ	е /	<b>A</b> /	6	30	3
B&F	₹	В		В-(	M) - C	; 1	⇒	В																		
Sho	rt																									
	ddroes	ina n	node /	Oncor	10	Γ		۸,	44	rn			ore		٠		410	'n			C	7	P	<b>G1</b> (	000	)
Α.	uui ess	sing ii	loue /	Орсос	16			^(	uu		33	۰,	,, ,	.51	510	,,,		""			O	ŀ	Ρ	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
	D2	F2	C2	E2		b	0	0	0	0	0	0	0	0	Γ						0	0	•	•	•	•

Mot	orola	SBC	В	Fun	ction		CPU type B / 6809   Address preselection  O CP 80  O PC 80  I O F S T # P R X Y D U I B G E A M F Z # P C I Y D U I B G H I X V V C  D D D D D D D D D D S S N D S S D S D S D S S D S D S S D S D S S D S S D S S D S S D S S D S D S D S S D S D S S D S S D S S D S D S D S S D S D S S D S D S D S D S D S D S D S S D S D S D S D S D S D S D S D S D S D S D S D S D S																			
B&F	ł	В		В-(	(M) - C	;	₽	В																		
Sho	rt																									
Δι	dress	ina m	ode /	Oncor	de de	Γ		Δι	44	re		: n	re	250	عاد	20	tic	'n			ı.	_		_		
, , ,		g		орос.				•			-	, 1		_	•	-		•••			C	)	Ρ	С	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	ī	N	Z	٧	С
	D2	F2	C2	E2		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	×	0	•	•	•	•

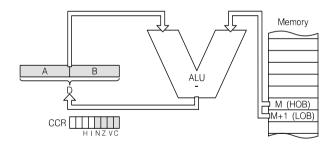
The contents of memory location M and the carry flag will be subtracted from accumulator B. The result will be stored in accumulator B.



Wot	orola	SUE	30	Fun	ction												4	٦,	'U	ty	p	e /	A /	6	30	3
B&F	₹	-D		D - (	M:M+	1)	Ε	> I	ס																	
Sho	rt																									
	ddroes	sina n	node /	Oncor	10	Г		Δ	44	rΔ		· n	re		عاد		Hic	'n			0	1	P	<b>31</b> 0	000	)
^'	uui ess	sing ii	loue /	орсос	16			^'	uu	16	33	۰,	,, e	-30	216			""			a	1	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			C	R		
	4/2	5/3	3/3	5/2		Е	Α	М	F	Z	#	Ρ	С	_	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	93	В3	83	А3		Γ	Γ	Г			0	0	0	0							0	0	•	•	•	•

Mot	orola	SUE	3D	Fun	ction												þ	ЭP	U	ty	ďρ	e E	3 /	6	В0	9
B&F	t	-D		D - (	M:M+	1)	Е	۱ (	D																	
Sho	rt																									
Λ.	droce	ina m	ode /	Oncor	10	Γ		۸,	14			s p			N/		Hio	'n			0	7	С	P	30	_
Α.	Jui 633	ilig ii	oue /	Орсос	16			~(	Ju	16	33	, h	,, ,	-30	216	-		""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	6/2	7/3	4/3	6+/2+		Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	93	В3	83	А3							0	0	ာ	0	•	•	•	•	•	•	0	0	•	•	•	•

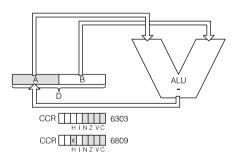
The contents of memory location M and M+1 will be subtracted from accumulator D. The result will be stored in accumulator D.



Mot	orola	SBA	4	Fun	ction												C	CP	U	ty	p	е.	A.	/ 6	30	3
B&F	₹	A-B		A - I	3 <b>⇒</b> A																					
Sho	rt																									
	ddress	ina n	node /	Oncor	łe.	Г		Δι	44	re		r	ıre	250	عاد	ect	tic	'n			0	1	Р	G10	000	
	uu. 000	,g	iouc,	Ороос						٠٠	-	'			٠.,			•••			a	1	Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		

	Mot	orola	SBA	1	Fun	ction												þ	P	U	ty	ďρ	e E	3 /	6	В0	9
	B&F	ł	A-B		A - I	B⇔A																					
	Sho	rt																									
	Δ,	ddraes	ina m	ode /	Oncor	10	Г		Δ,	44	rΔ		s p	rc		عاد	c	io	'n			0	1	С	Pδ	30	
	^	aui 638	ing in	oue,	Орсос	10				<i>a</i> u		30	, 1	,, ,	.30	,,,			···			О	•	Ρ	C	30	
	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
1			25/3				Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
			BD FF 9A																			X	0	•	•	•	•

The contents of accumulator A will be subtracted from accumulator B. The result will be storede in accumulator A.



Since this instruction is not available in the 6809 the utilization of a sub-program in the operating system is required. This sub-program which simulates this instruction consists of three instructions:

PSH ! ,B The contents of B are placed on the stack:

SP! - 1 ⇒ SP! B ⇒ (SP!)

SUBA ! ,+ T

The contents of the memory location which the system stack pointer points to are subtracted from accumulator A. SP! is then increased by 1 (Post increment):

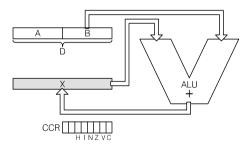
A - (SP!) ⇒ A SP! + 1 ⇒ SP!

RTS Leaves the sub-program.

Mot	orola	AB	X	Fun	ction	( ⇒ X																				
B&F	₹	B+F	₹	B +	X ⇒ 2	Address preselection																				
Sho	rt																									
	ddrocc	ina n	node /	Onco	40	Γ		۸,	44	ro					٠			'n			0	T	PC	<b>G10</b>	000	,
_ ^	uui ess	sing ii	ioue /	Орсо	ue .			^(	uu		3	۰,	,, e	.51	-10	-		""			О	ī	P	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
1/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
ЗА						Γ	Γ		Γ			Γ									0	0	0	0	0	0

Mot	orola	ABX	(	Fun	ction												þ	P	U	ty	/p	e I	3 /	6	80	9
B&F	₹	B+R	₹	B +	X ⇒ 2	K																				
Sho	rt																									
Δ,	ddraes	ina m	ode /	Oncor	10	Γ		Δ	44	rΔ		s p	rc		عاد		Hio	'n			O	7	С	Pδ	30	
l ^`	uui 633	ing ii	ioue /	Орсос	40	L		^'	uu		30	, ,	,,,	.30	-			•••			С	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	R		_
3/1						Ε	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
ЗА						Γ	Γ	Г	П			Γ				Г			Г		0	0	0	0	0	0

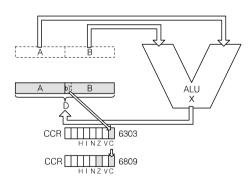
The contents of accumulator B and index register X will be added. The result will be stored in X.



Mot	orola	MUI	L	Fun	ction												C	CP	U	ty	p	е /	A /	6	30	3
B&F	₹	A*B		ΑX	B⇔l	D																				
Sho	rt																									
۸,	ddroes	ina n	node /	Oncor	40	Γ		۸,	44	rn			ore		٠			'n			0	T	P	<b>G10</b>	000	
^(	uui ess	ing ii	loue /	Орсос	ue			^'	uu		33	, ,	,, 6	.5	-10			"			0	7	Р	G-F	Š	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	O			CC	R		
7/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	Ι	Ν	Z	٧	С
3D						Г	П					Г		П							0	0	0	0	0	•

Mot	orola	MUL	-	Fun	ction												C	P	U	ty	ďρ	e E	3 /	6	В0	9
B&F	t	A*B		ΑX	В⇒	D																				
Sho	rt																									
Δι	ddress	ina m	ode /	Oncor	łe	Γ		Δι	44	re		s p	re		۱e	·C	tio	'n			0	1		Pδ	_	
	au. 000	g	ouc,	Ороос		L				٠.	•	, ,		-	•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		Ī
11/1						Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U		В	G	Η	I	N	Z	٧	С
3D						Г	П	Γ	П			П									0	0	0	•	0	•

The contents of accumulators A and B will be multiplied. The result will be stored in D.



## 3.7. TEST AND COMPARISON INSTRUCTIONS

This section describes the instructions which compare registers or memory locations with one another (subtraction) or which test a certain bit pattern (AND combination). Operations are completed without saving the result. The condition code register is changed according to the result of the operation. Depending on the change made to the condition code register (result) certain conditional branches can be made.

Motorola	B&R		Operatir	g mode	
Wotorola	Dar	PG1000	PG-PC	CP 80	PC 80
СВА	AVB	0	0	0	0
CMPA	CMP	0	0	0	0
CMPB	VB	0	0	0	0
CPX	VR	0	0	0	0
CPX#	VRK	0	0	0	0
CPY	VY				0
CPY#	VYK				0
BITA	В	0	0	0	0
BITB	BB	0	0	0	0
TIM	TIM		0		

Mot	orola	19	CRA	١.	Fun	ction												L	٦,	'U	ty	p	е /	A /	6	30	3
B&F	ł	T /	AVB	3	A - I	3																					٦
Sho	rt	Т																									1
	droce	ein	a m	ode /	Oncor	10	Г		۸,	14	rn			re		٦l/		Hio	'n			0	7	P	<b>31</b> 0	000	П
_ ^	uuiess	3111	9	loue /	Орсос	16			~	Ju		33	۰,	,, e	.5	-10			""			О	•	Р	G-F	Š	
IMPL.	DIR.	E)	XT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	Х	Υ	D	U	!	В	G			C	R		٦
1/1							Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
11								П														0	0	•	•	•	•

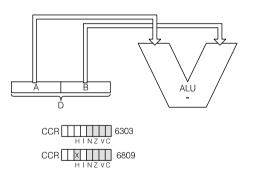
Francisco.

Matarala ODA

Mot	orola	CBA	*	Fun	ction												¢	P	U	ty	/p	e I	В/	6	80	9
B&F	t	AVE	3	A - I	В																					
Sho	rt																									
Δ,	ddrae	sina m	ode /	Oncor	40	Г		Δ,	44	rΔ		· n	re		عاد	c	io	'n			O	7	С	Pδ	30	
Α.	uui es	sing ii	ioue /	орсос	ue			~(	Ju	16	33	, h	,, e	30	316			""			O	7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		
		25/3				Е	Α	М	F	Z	#	Р	С	-	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
		BD FF 95				Г	П	П	П			П	П								×	0	•	•	•	•

The contents of accumulators A and B will be compared by subtracting one from the other (A - B). The result influences the condition code register but is not saved.

ODUL 1 .... A / 0000



## 6809:

Since this instruction is not available in the 6809 a sub-program jump in the operating system is utilized. This sub-program consists of three instructions:

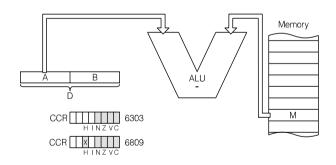
of three instr	ructions:	
PSH	! ,B	The contents of accumulator B are placed on the stack: SP! - 1 ⇒ SP! B ⇒ (SP!)
CMP	! ,+	The contents of A are compared with the contents of the memory location that the system stack pointer SP! points to. SP! is then increased by 1 (post increment): A - (SP!) SP! + 1 ⇒ SP!
RTS		Leaves the sub-program.

Motorola	CMPA	Function		CPU type	A / 6303
B&R	CMP	A - (M)			
Short	٧				
			T .		DO 4000

	ddroes	ina m	ode /	Oncor	10	Г		۸,	44	ro					٠.,		41,	'n			O	7	P	<b>31</b> (	000	
_ ^'	uuies	sing ii	ioue /	Орсос	16	Address preselection											O	7	P	G-F	SC					
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	ĸ		٦
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
	91	B1	81	A1			ာ	0	0	0	0	ာ	0	ာ							0	0	•	•	•	•

Mot	orola	СМ	PA	Fun	ction												þ	P	'n	ty	/p	e I	В /	6	В0	9
B&F	t	СМЕ	-	A - (	M)																					
Sho	rt	٧																								
	ddraed	ina n	node/	Oncor	ı۵	Γ		Δ	44	re		: n	\rc		ماد		tic	'n			О	7	С	P	30	
	uuies	onig n	ioue,	Opcoc	10	L		^'	uu		30	, 1	,,,	.3	-			•••			О	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		
	4/2	5/3	2/2	4+/2+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
	91	B1	81	A1		Ь	0	0	0	0	0	0	0	0	•	•	•	•	•	•	х	0	•	•	•	•

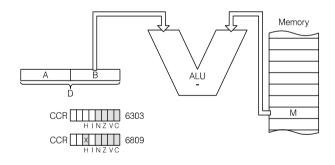
The contents of accumulator A will be compared with the contents of memory location M by subtracting one from the other (A -(M)). The results influences the condition code register but is not saved.



Mot	orola	CMI	РВ	Fun	ction												C	٦,	'U	ty	/p	е /	A /	6	30	3
B&F	₹	VВ		В-(	M)																					
Sho	rt																									
	ddroes	ina m	node /	Oncor	10	Γ		۸,	44	ro			re		٠		410	'n			O	1	P	G1(	000	)
_ ^	uuies	sing ii	loue /	Орсос	16	L		^'	uu		33	۰,	,, e	.51	-10	-		""			O	ŀ	Ρ	G-I	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	CR		Т
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
	D1	F1	C1	E1		b	0	0	0	0	0	0	0	0							0	0	•	•	•	•

Mot	orola	СМІ	РВ	Fun	ction												þ	P	υ	ty	/p	e I	3 /	6	80	9
B&F	₹	VВ		В-(	M)																					
Sho	rt																									
Δι	ddress	ina m	ode /	Oncor	1e	Γ		Δι	44	re		s p	re		ale	20	tic	'n			O	_		P		
	uu. 000	g	.ouc,	Ороос						٠٠	•	, ,		-	•••			•••			С	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	o	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		Ī
	4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Ρ	С	1	Υ	D	U		В	G	Н	ı	N	Z	٧	С
	D1	F1	C1	E1		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	x	ာ	•	•	•	•

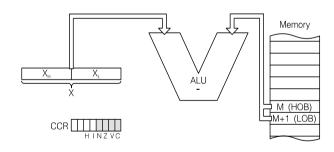
The contents of accumulator B wil be compared with the contents of memory location M by subtracting one from the other (B -(M)). The result influences the condition code register but is not saved.



Mot	orola	CP	(	Fun	ction												(	CP	U	ty	р	е /	۹/	6	30	3
B&F	₹	VR		X - (	M:M+	1)																				
Sho	rt																									
Δ,	ddress	ina n	node /	Oncor	łe	Γ		Δ	44	re		: r	re	250	عاد	20	tic	'n			0	1	PC	<b>G10</b>	000	)
	uu. 000	,g	iouc,	Ороос					uu	٠.	-	'			٠.、			•••			а	1	Р	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
	4/2	5/3	3/3	5/2		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	н	Ι	Ν	Z	٧	С
	9C	BC	8C	AC		Г	Г	П	П			0	0	0			П				0	0	•	•	•	•

Mot	orola	CPX	(	Fun	ction												C	P	U	ty	ďρ	e I	3 /	6	В0	9
B&F	ł	VR		X - (	M:M+	1)																				
Sho	rt																									
Λ.	ddroc	sina n	node/	Oncor	10	Γ		۸,	14	ro		s p			٠		Hic	'n			0	7	С	Pδ	30	
	uuies	sing ii	ioue/	орсос	16			~	Ju		33	• P	,, ,	531	-16			""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	6/2	7/3	4/3	6+/2+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	9C	BC	8C	AC								0	0	0	•		•	•	•	•	0	0	•	•	•	•

The contents of index register X will be compared with the contents of memory locations M and M+1 by subtracting one from the other (X - (M:M+1)). The result influences the condition code register but is not saved.

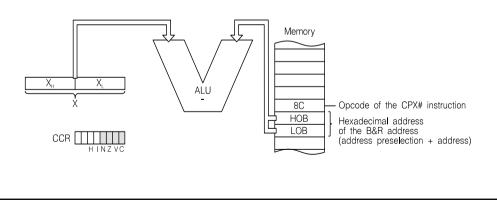


Mot	orola	CP)	<b>(</b> #	Fun	ction												C	P	U	ty	p	e .	A /	6	30	3
B&F	₹	VRI	<b>(</b>	X - I	M																					٦
Sho	rt																									
Λ.	ddroes	ina n	node /	Oncor	40	Γ		۸,	44	rn			ore		٠		Hio	'n			0	)	P	G1(	000	П
Α.	uuies	ing ii	loue /	Орсос	ue			^'	uu		33	, ,	,, 6	.5	-10	-					0	<b>,</b>	Р	G-F	S	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			C	CR		٦
			3/3			Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	ı	Ν	Z	٧	С
			8C			Э	0	0	0	0		0	0	П			П		П		0	0	•	•	•	•

٦	Motorola		CPX#		Fun	CPU type B / 6809																					
	B&R		VRK		X - I																						
	Short																										
Addressing mode / Opcode								Address preselection O CP 80 O PC 80														0	1	CP 80			
-	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CCR			
Ī				4/3			Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
ĺ				8C			b	0	0	0	0		ာ	0						•	•	0	0	•	•	•	•

The contents of index register X will be compared with the address entered for M by subtracting one from the other (X - M). The result influences the condition code register but is not saved.

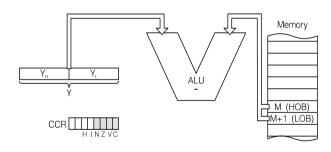
This instruction corresponds to the CPX # xxxx instruction. The PROgramming SYStem replaces the address, which the user enters in the STL input line, with the effective address (hexadecimal value), which the B&R address (= address preselection + address position) holds in the PLC memory.



Mot	orola			Fun	ction												C	P	J	ty	γpe	e /	۸/	630	03
B&F	₹																								
Sho	rt																								
۸,	ddroes	ina m	ode /	Oncor	40	Γ		۸.	44	ra					٠	ect		n					PG	100	0
^(	uuies	ing ii	ioue /	Орсос	16			^'	uu	16	33	, 1	,, e	-31	516	-		"				Τ	PG	-PC	;
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	! !	3	G			CCI	₹	
						F	Α	м	F	7	#	Р	С	ī	Υ	D	u	1 1	3	G	н	Т	N Z	, V	1

Mot	orola	CPY	,	Fun	ction												þ	CP	U	ty	/p	e I	3 /	6	80	9
B&F	1	VY		Y - (	M:M+	1)																				
Sho	rt																									
۸,	ldroce	ina m	ode /	Oncor	10	Γ		۸,	14	ra		s p	rc		٦le		Hic	'n				Τ	С	P	30	
Α.	iui 638	ilig ii	loue /	Орсос	16			~(	Ju	16	33	, h	,, ,	.5	-16			""			О	1	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	7/3	8/4	5/4	7+/3+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	N	z	٧	С
	10 9C	10 BC	10 8C	10 AC								•	•	•	•		•	•	•	•	0	0	•	•	•	•

The contents of index register Y will be compared with the contents of memory locations M and M+1 by subtracting one from the other (Y - (M:M+1)). The result influences the condition code register but is not saved.

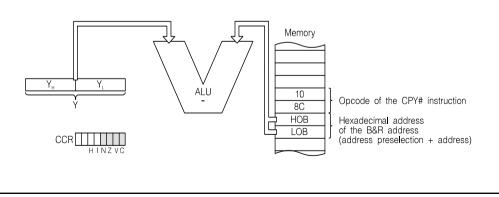


Mot	orola			Fun	ction												C	P	U	ty	pe	<b>3</b> /	A /	6	30	3
B&F	₹																									7
Sho	rt																									
	ddraed	sina n	node /	Oncor	10	Γ		A	44	rΔ		· n	ırc		عاد	201	·io	n		П		Τ	P	<b>310</b>	000	╗
	uui 63	onig ii	ioue /	Орсос	40			^'	uu		30	, 1	,,,	.30	,,,			•••				Ι	Р	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		П
						Ε	Α	М	F	Z	#	Р	С	_	Υ	О	U	!	В	G	Н	Ι	Ν	Ζ	٧	С

T	Mot	orola	CPY	<b>'</b> #	Fun	ction												þ	P	U	ty	'n	e I	3 /	6	В0	9
ſ	B&F	ï	VYK	7	Y - I	И																					
ſ	Sho	rt																									
ı	Λ.	ddroes	ina n	node/ (	)ncor	io.	Г		۸,	14			s p			N/		Hic	'n				Τ	С	P	30	
L	^	uuies	ilig ii	ioue/ v	opcoc	16				ıu	16	33	, h	,, ,	.5	316			""			О	1	Р	C	30	
-	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
I				5/4			Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
				10 8C			•	•	•	•	•		•	•						•	•	0	0	•	•	•	•

The contents of index register Y will be compared with address M by subtracting one from the other (Y - M). The result influences the condition code register but is not saved.

This instruction corresponds to the CPY # xxxx instruction. The PROgramming SYStem replaces the address, which the user enters in the STL input line, with the effective address (hexadecimal value), which the B&R address (= address preselection + address position) holds in the PLC memory.



WOL	oroia	J 5117	_	' u	CLIOII												-	٠.	v	·y	P		_	, ,	50	•
B&F	₹	В		Ал	(M)																					
Sho	rt																									
	ddroes	ina n	node /	Oncor	40	Γ		Δ,	44	rΔ		: r	ore		ماد	20	tic	'n			0	1	Р	G1	000	)
	uui 63	ing ii	ioue /	Орсос	46	L		^	u		30	, ,	,,,		-			"			О	,	F	PG-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	O			С	CR	Ξ	
	3/2	4/3	2/2	4/2		Ε	Α	М	F	Z	#	Р	С	ı	Υ	D	U	!	В	G	Н	T	N	Z	٧	С

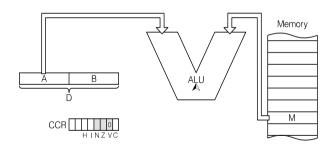
Function

Motorola BITA

	Mot	orola	BIT	A	Fun	ction												¢	P	U	ty	/p	e I	В/	6	80	9
	B&F	ł	В		Ал	(M)																					
	Sho	rt																									
ſ	Δ	ddrae	sina n	node/	Oncor	10	Γ		Δ,	44	ro		s p	ırc		ماد		tic	'n			C	)	С	P	30	
ı		uuies	sing ii	ioue/	орсос	16	L		^'	Ju		33	, h	,, ,	.5	-10			""			С	<b>)</b>	Р	C	30	
ı	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		
Ī		4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
ĺ		95	B5	85	A5		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

The contents of accumulator A will be combined in AND relation with the contents of memory location M. The result influences the condition code register but is not saved.

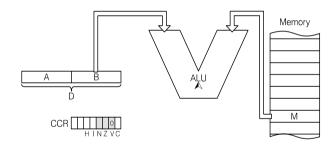
CPII type A / 6303



Mot	orola	BIII	В	Fun	ction												ľ	٦,	'U	ty	p	e A	A /	6	30	3
B&F	₹	ВВ		Вл	(M)																					
Sho	rt																									
	ddroes	ina n	node /	Oncor	10	Γ		۸,	44	ro			ore		٠			'n			0	7	P	<b>G1</b> (	000	)
_ ^'	uui ess	sing ii	loue /	Орсос	16	L		^'	uu		33	۰,	,, e	.5	-16	-		""			О	<b>)</b>	P	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	3/2	4/3	2/2	4/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	Ν	z	٧	С
	D5	F5	C5	E5		0	ာ	0	0	0	0	ာ	0	0							0	0	•	•	•	0

T	Mot	orola	BITI	В	Fun	ction												þ	CP	U	ty	/p	e I	В/	6	80	9
ſ	B&F	ł	ВВ		Вл	(M)																					
ſ	Sho	rt																									
ſ	Δ,	drace	ina m	ode /	Oncor	10	Г		Δ,	14	rΔ		s p	rc		عاد		tic	'n			С	1	С	P	30	
L		aui ess	iiig ii	ioue /	Орсос	10				<u> </u>		30	, 1	,, ,	.30	510			<b>,,,</b>			С	)	Ρ	C	30	
Ι	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
I		4/2	5/3	2/2	4+/2+		Е	Α	М	F	Z	#	Р	С	-	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
[		D5	F5	C5	E5		0	0	0	0	0	0	0	0	0	•	•	•	•	•	•	0	0	•	•	•	0

The contents of accumulator B will be combined in AND relation with the contents of memory location M. The result influences the condition code register but is not saved.



Mot	orola	TIM		Fun	ction												(	P	U	ty	/p	e .	A /	6	30	3
B&F	₹	TIM		(X +	Offse	t)	^	IN	١N	1																
Sho	rt																									
۸,	ddroes	ina n	node /	Oncor	10	Г		۸,	44	ro	ss				٦le		Hic	'n			Г	Τ	P	G10	000	Л
ί	uui ess	sing ii	loue /	Орсос	16			^'	Ju	16	33	٠,	,, e	.5	-10	-		"			C	•	Ρ	G-F	SC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		٦
			5/3			Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
			6B			Γ	Γ				0										b	0	•	•	•	0

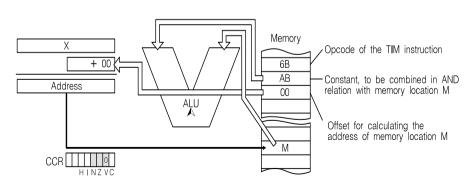
Mot	orola			Fun	ction												þ	CP	'n	ty	/p	e I	3 /	6	80	9
B&F	ł																									
Sho	rt																									
	ddres	sina n	node/	Oncor	łe	Γ		Δ	dА	lre		s p	ıre	250	عاد		tic	'n						P		
^	uu. 00.	Jg	iouc,	Оросс		L		_	uu			, ,			٠.,			•••			L		Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	υ	!	В	G	Г		CC	R		_
						Ε	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
						Г	Г	Т	Т	Г	Г	Г	Г	П		Г	Г	Г	Г		Г					Г

An immediate value \$xx (constant) given in the instruction will be combined in AND relation with the contents of memory location M. The result influences the condition code register but is not saved. Address M is attained from the sum of index register X and the offset \$yy which must be given in the instruction.

Syntax: TIM # \$xxyy xx => 1 byte constant

yy => 1 byte offset for index register X

Example: TIM # \$AB00



# 3.8. INCREMENT AND DECREMENT INSTRUCTIONS

This section describes instructions which increment or decrement the contents of a register or memory location by a value of 1.

Matauala	D.O.D.		Operatin	g mode	
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
INC	INC	0	0	0	0
INCA	IA	0	0	0	0
INCB	IB	0	0	0	0
INX	IR	0	0	0	0
INS	IS	0	0	0	0
DEC	DEC	0	0	0	0
DECA	DA	0	0	0	0
DECB	DB	0	0	0	0
DEX	DR	0	0	0	0
DES	DS	0	0	0	0

					••												٦.	••	_	٠,	Ρ.	٠.	• • •	•		_
B&F	₹	INC		(M)	+1 ⇨	(1	M)																			
Sho	rt																									
	ddroor	ina n	node /	Onco	40	Γ		۸.	44		ss				٠.,						0	ī	P	<b>G10</b>	000	)
A	uures	sing ii	ioue /	Opcoo	Je			A	Ju	16	55	١,	,, e	:51	316	30	LIC	""			О	•	P	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	ĸ		Ξ
		6/3		6/2		Ε	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
		7C		6C			0	0	0			0	0	0							0	0	•	•	•	0

Function

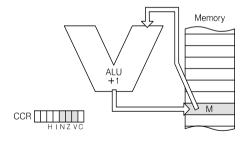
Motorola INC

T	Mot	orola	INC		Fun	ction												þ	ЭP	U	ty	ďρ	e E	3 /	6	80	9
ſ	B&F	1	INC		(M)	+1 ⇨	(1	M)																			
ſ	Sho	rt																									
Γ	Δι	ldress	ina m	ode /	Oncor	1e	Γ		Δι	44	re		s p	re	250	عاد	20	tic	'n			0	_		P		
L	٠.,	.u. 000	g	.ouc,	Ороос		L				٠.	•	, ,			•••			•••			0	1	Ρ	C	30	
Γ	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	CR		Ī
ſ		6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
I		0C	7C		6C			0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	0

The contents of the entered memory location will be incremented by a value of 1.

The V flag is set to logic 1 if the contents of the memory location are increased from \$7F to \$80, otherwise V is always set to 0.

CPU type A / 6303

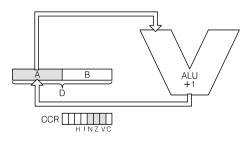


Mot	orola	a	INC	Α	Fun	ction												C	CP	U	ty	pe	Α	/ 6	303
B&F	t	Т	IA		A +	1 ⇒ A	١																		
Sho	rt	T																							
	ddroes	eei	na m	ode /	Oncor	40	Γ		Ac	ı	rn			·ro		٠		410	'n			0		PG1	000
_ ^	uui es	331	ng m	loue /	орсос	16			~	ıu		33	, 1	,, e	.5	-10	-	···	"			0		PG-	PC
IMPL.	DIR.	. [	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G		(	CCR	
							_			_			_		_							-	-	$\overline{}$	-

Mot	orola	INC	Α	Fun	ction												¢	P	U	ty	/p	e l	В	6	80	9
B&F	t	IA		A +	1 ⇒ 4	١																				
Sho	rt																									
	ddress	ina m	odo /	Oncor	10	Γ		۸,	14						٦l/		tio	'n			0	7	С	P	30	П
<u>۱</u> ^'	uui ess	iliy ii	loue /	орсос	16			~'	Ju	16	33	, ,	,, e	.5	-10						О	7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		
2/1						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
4C						Γ	Г	Г				Г	Г								0	0	•	•	•	0

The contents of accumulator A will be incremented by a value of 1.

The V flag is set to logic 1 if accumulator A is increased from \$7F to \$80, otherwise V is always set to 0.

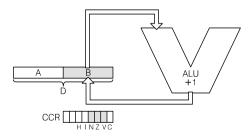


Mot	orola	INC	В	Fun	ction												C	CP	U	ty	p	e A	Α/	6	30	3
B&F	ł	IB		B +	1	3																				
Sho	rt																									
Δ,	ddraes	ina m	node /	Oncor	ło.	Г		Δ,	44	rΔ		: r	re		ماد	201	tic	'n			O	)	P	<b>31</b> (	000	<u> </u>
_^\	au 1 6 3 3	ing ii	ioue /	Орсос	10			^'	<i>a</i> u		30	, ,	,, ,	.31	-			<b>,,,</b>			U	)	Ρ	G-I	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	z	٧	С
5C																					0	0	•	•	•	0

Mot	orola	INC	В	Fun	ction												C	P	U	ty	'n	e I	В/	6	В0	9
B&F	t	IB		B +	1 ⊅ E	3																				
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	10	Γ		Δ	44	rΔ		: r	ırc		عاد		tio	'n			0	1	С	Pδ	30	
l ^`	aui 633	ing ii	oue,	Орсос	40	L		^'	uu		30	, ,	,,,	.30	-10			•••			О	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			C	R		_
2/1						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
5C						Г	Г	Г	П			П	Г	П					П		0	0	•	•	•	0

The contents of accumulator B will be incremented by a value of 1.

The V flag is set to logic 1 if accumulator B is increased from \$7F to \$80, otherwise V is always set to 0.

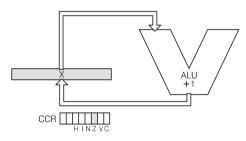


Motorola	INX	Function	CPU type A / 6303
B&R	IR	X + 1 ⇒ X	
Short			

	ddroes	ina m	ode /	Oncor	40	Г		Ad	44	ro					٠.,		Hic	'n			O	7	PC	<b>G10</b>	)00	
^'	uuies	sing ii	ioue /	Орсос	16			~	uu		33	, ,	,, ,	-31	-10			"			O	7	P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	ĸ	Т	٦
1/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
08						Γ	Γ														0	0	0	•	0	0

Mot	orola	INX		Fun	ction												C	P	U	ty	/p	e E	3 /	6	В0	9
B&F	1	IR		X +	1 ⇒ )	(																				
Sho	rt																									
Δ,	drace	ina m	ode /	Oncor	10	Γ		Δ	44	ro		s p	re		عاد	c	Hio	'n			О	1	С	Pδ	30	
_^`	aui 633	ing ii	ioue /	Орсос		L		^'	u		30	, P		.30	,,,						О	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	Ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Γ		CC	R		
				5/3		Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U		В	G	Н	1	Ν	Z	٧	С
				30 88 01		Г	Т	Г	Г	Г	Г	П	П				П	П		П	0	0	0	•	0	0

The contents of index register X will be increased by a value of 1.



## 6809:

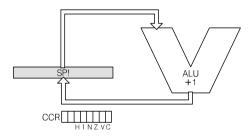
The "INX" instruction does not exist with the 6809. It is simulated with the "LEAX  $\,$  X 001" instruction.

Mot	orola	INS	i	Fun	ction												C	P	U	ty	/pe	A / 6	303
B&F	ł	IS		S+	1 ⇒ 5	;																	
Sho	rt																						
	droce	ina n	node /	Oncor	10	Г		۸,	14						\/\	ect	·io	'n			0	PG1	1000
^	Jui 63	sing ii	ioue /	Орсос	16			Α.	ıu		33	, 1	,, e	-30	310	501		""			0	PG-	-PC
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	т	#	Ρ	R	х	Υ	D	U	!	В	G		CCR	₹

31

Mot	orola	INS		Fun	ction												C	P	U	ty	р	e E	3 /	6	В0	9
B&F	₹	IS		S+	1 ⇒ 8	3																				
Sho	rt			1																						
Δ,	ddress	ina m	ode /	Oncor	10	Γ		Δ	44	rΔ		: n	rc		عاد		tio	'n			0	1	С	Pδ	30	
_^`	uui 633	ing ii	oue,	Орсос	40	L		^'	u		30	, h	,,,	.30	,,,			•••			a	•	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
				5/3		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	н	I	Ν	Z	٧	С
				32 E8 01		Γ	Г	Г					Г								0	0	0	0	0	0

The contents of the system stack pointer will be increased by a value of 1.



#### 6809:

The instruction "  ${\tt INS}$  " does not exist for the 6809. It is simulated by the "LEA!  $\,\,!\,$  001" command.

WIOL	oroia	oia	DEC	,	run	CHOIL												u	٢u	, ,	ype	•	A / 0303
B&R	ł		DEC	;	(M)	-1 ⇒	(1	A)															
Sho	rt																						
	ddroes	Irocci	ina m	ode /	Oncor	40	Γ		۸۰	ı			n	res		ı	nti	۸r			0	I	PG1000
	Julesa	11 6331	iiig ii	ioue /	орсос	16			Αι	ıu	-	99	μ	163		10	CII	01	٠		0	Ι	PG-PC
IMPL.	DIR.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	т	#	Р	R >	( )	1	υ	!	В	G	Г		CCR

Eupotion

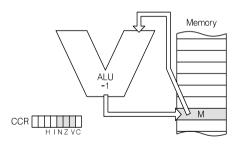
Motorolo DEC

Mot	orola	DEC	;	Fun	ction												C	CP	U	ty	/p	e E	3 /	6	В0	9
B&F	₹	DEC	;	(M)	-1 ⇨	(1	A)																			
Sho	rt																									
Δ,	ddraes	ina n	ode /	Oncor	10	Γ		Δ	44	rΔ		. r	ırc		ماد		tic	'n			O	1	С	P	30	
l ^`	uui 633	ing ii	ioue /	Орсос	40	L		^'	u		30	, ŀ	,,,	-3	-			•••			С	•	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	CR		
	6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
	0A	7A		6A		Г	0	0	0			0	0	0	•	•	•	•	П	•	0	0	•	•	•	0

The contents of the entered memory location M will be decremented by a value of 1.

The V flag is set to logic 1 if accumulator A is decreased from \$80 to \$7F, otherwise V is always set to 0.

CBILture A / 6202

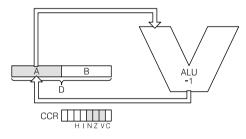


Mot	orola	DEC	SA.	Fun	ction												C	P	U	ty	p	e A	A /	6	30	3
B&F	₹	DA		A - 1	1 ⇒ A																					
Sho	rt																									
	Addressing mode / Opc					Г		۸,	14	rn			re					'n			0	1	P	<b>G1</b> (	000	,
_ ^	uui ess	sing ii	ioue /	орсос	16			^(	Ju		33	۰,	,, e	.5	216	-		""			a	ŀ	Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
4A																					0	0	•	•	•	0

Mot	orola	DEC	CA	Fun	ction												c	P	U	ty	/pe	e E	3 /	68	В0	9
B&F	₹	DA		Α-	1 <b>⇒</b> A																					
Sho	rt			1																						
Δ,	ddress	ina m	node /	Oncor	de de	Γ		Δ,	44	re		s p	re	-56	عاد	· C1	in	'n			0	Ī	_	Pε		
Α.	uu. 000	g	iouc,	Орос	40	L		_			-	, h						•••			0	ı	Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	o	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
2/1						Ε	Α	М	F	Z	#	Р	С	-	Υ	D	U		В	G	Ι	Τ	Ν	Z	٧	С
4A						Γ	Г	П	Г			П	П			П				П	0	0	•	•	•	0

The contents of accumulator A will be decremented by a value of 1.

The V flag is set to logic 1 if accumulator A is decreased from \$80 to \$7F, otherwise V is always set to 0.

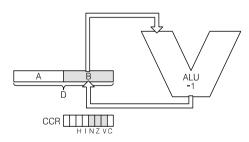


Mot	orola	ıla	DEC	В	Fun	ction												C	P	U	ty	pe	Α	63	303
B&F	₹		DB		B - 1	I ⇒ B																			
Sho	rt																								
	ddroes	neei	na m	ode /	Oncor	10	Γ		۸,	14	ra		. n	re	-			Hio	'n			0	P	G10	00
^'	uui es	CSSI	iiig iii	oue /	орсос	16			Α.	ıu	16	33	, h	16	30	,16	-		""			0	P	G-P	С
IMPL.	DIR.	R.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G		C	CR	
4.74							1	Ι.					_	- 1	. 1		_		. 1	_				I = I	

Mot	orola	DEC	В	Fun	ction												¢	P	U	ty	/p	e E	3 /	6	В0	9
B&F	l	DB		B - ′	1 ⇒ B																					
Sho	rt																									
$\Box$	ldroce	ina m	ode /	Oncor	10	Γ		۸,	14	re			ro				Hio	'n			0	T	С	P	30	
_ ^	iui caa	орсос	16			~	Ju	16	38	, h	,, ,	30	216			""			О	1	Р	C	30			
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		
1/1						Ε	Α	М	F	Z	#	Р	С	-	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
5A																					0	0	•	•	•	0

The contents of accumulator B will be decremented by a value of 1.

The V flag is set to logic 1 if accumulator B is decreased from \$80 to \$7F, otherwise V is always set to 0.

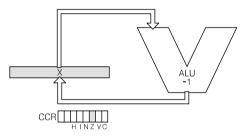


Motorola	DEX	Function		CPU type	A / 6303
B&R	DR	X - 1 ⇒ X			
Short					
Address	ing mode /	Oncode	Address preselecti	on O	PG1000
7144.000		opecus	/ ruanded processes	• IO	PG-PC

۸.	ddroed	ina m	node /	Oncor	40	ı		۸,	44	ro					ماہ	~~	410	'n			Ç	1	P	<b>31</b> 0	)00	וי
_ ^'	uui es	sing ii	loue /	Орсос	16	Address preselection										O	•	Р	G-F	SC	$\neg$					
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		٦
1/1						Ε	Α	М	F	z	#	Р	С	ı	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
09																					0	0	0	•	0	0

	Mot	orola	DEX	(	Fun	ction												C	P	U	ty	ďρ	e E	3 /	6	80	9
	B&F	ł	DR		X - 1	⇒ X																					
	Sho	rt																									
	Ad	dress	ing m	ode /	Орсос	le	Γ		Ac	ld	re	SS	s p	re	se	ele	ct	tio	n			00	_	_	P 8		
١	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Ť	_	CC	_	_	_
					5/3		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
					30 88 FF		Г															0	0	0	•	0	0

The contents of index register X will be decreased by a value of 1.



## 6809:

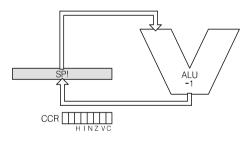
The "DEX" instruction does not exist for the 6809. It is simulated by the "LEAX  $\,$  X -001" instruction.

Motorola	DES	Function	CPU type A / 6303
B&R	DS	S-1 ⇒S	
Short			

	ddroe	ina m	ode /	Oncor	40	Г		۸,	44	ro					٠		Hio	'n			0	1	P	<b>310</b>	000	П
Α.	uui es	sing ii	ioue /	Орсос	16	Address preselection										О	•	Р	G-F	Š	П					
IMPL.	DIR.	EXT.	IMMED.	<del> </del>								В	G	Г		CC	R		٦							
1/1						Е	Α	М	F	z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
34						Г	П	П	Г		П		П	П	П		П		П		0	0	0	0	0	0

Mot	orola	DES	;	Fun	ction												C	P	U	ty	/p	e E	3 /	6	80	9
B&F	₹	DS		S - 1	⇒ S																					
Sho	rt																									
A	ddress	ina m	ode /	Opcod	le	Γ		Αc	bb	re	SS	s p	re	se	ele	ct	tic	n			О	_		P		
						L						•									О	<u> </u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R	_	
				5/3		Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
				32 E8 FF		Г	П	Г				П	П					П	П		0	0	0	0	0	0

The contents of the system stack pointer will be decreased by a value of 1.



## 6809:

The "DES" instruction does not exist for the 6809. It is simulated by the "LEA!  $\,!\,$  -001" instruction.

# 3.9. SHIFT AND ROTATE INSTRUCTIONS

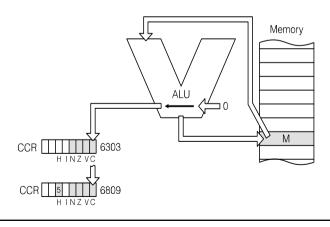
This section describes instructions which are used to shift or rotate the contents of a register or memory location 1 bit to the left or right.

	505		Operatin	g mode	
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
ASL	SL	0	0	0	0
ASLA	SLA	0	0	0	0
ASLB	SLB	0	0	0	0
ASLD	SLD	0	0	0	0
LSR	SR	0	0	0	0
LSRA	SRA	0	0	0	0
LSRB	SRB	0	0	0	0
LSRD	SRD	0	0	0	0
ROL	SLI	0	0	0	0
ROLA	RLA	0	0	0	0
ROLB	RLB	0	0	0	0
ROR	SRE	0	0	0	0
RORA	RRA	0	0	0	0
RORB	RRB	0	0	0	0

Mot	orola	ASL		Fun	ction												(	CP	U	ty	/p	e A	Α/	6	30	3
B&F	₹	SL		Shif	t cont	er	nts	6 0	of I	M								Г	H	Ŧ.	I	Ţ	Ţ	Ŧ	_	П
Sho	rt			1 bit	to th	e l	lef	t										C		d7	_	_		d0		١
Λ.	ddroed	ina m	ode /	Oncor	10	Γ		۸,	44	rn			re		٦l/		410	'n			С	7	P	<b>31</b> 0	000	П
Α.	uui ess	sing ii	ioue /	Орсос	16			^(	uu		33	۰,	,, e	.5	-10	-		""			U	)	Ρ	G-F	c	⅃
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		٦
		6/3		6/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
		78		68		Γ	0	0	0			0	0	0							0	0	•	•	•	•

Mot	orola	ASL		Fun	ction												þ	CP	U	ty	/p	e I	В/	6	80	9
B&F	₹	SL		Shif	/2+ E A M F Z # P C I					Γ	-	Ŧ	1	Ţ	†‡	Ŧ	_	0								
Sho	rt			1 bit	to th	e I	lef	t										C		d7				d0		
Δ,	ddress	ina m	ode /	Oncor	łe	Γ		Δι	44	re	S 6	: n	ıre	250	عاد		tic	'n			$^{\circ}$	7	С	P	30	
_^`	uui 633	ning ii	oue /	Орсос	10				uu		30	, 1	,,,	-31	510			<b>,,,</b>			V	)	Р	С	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	Address pre					х	Υ	D	U	!	В	G			C	R		
	6/2	7/3		6+/2+		Ε	Α	М	F	ress pre				1	Υ	D	U	!	В	G	Н	ī	N	Z	٧	С
	08	78		68		L	0	0	0			0	0	0	•	•	•	•		•	×	0	•	•	•	•

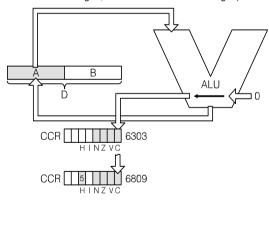
The contents of memory location M will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag. When the instruction is executed the V flag is assigned with the result of N  $\oplus$  C (V = N  $\oplus$  C). The V flag shows whether bit 7 has been changed or not (V = 1 => bit 7 was changed; V = 0 => bit 7 was not changed).



Mot	orola	ASL	Α	Fun	ction												C	P	U	ty	/p	e A	Α /	6	30	3
B&F	₹	SLA	`	Shif	t cont	er	nts	6	f	4								Γ	7-	Ŧ	ŀ	Ţ	#1	Ŧ	<u>.                                    </u>	0
Sho	rt			1 bit	to the	e I	ef	t										Ċ	5	d7				d0		
Δ,	ddraes	ina n	node /	Oncor	10	Г		Δ,	44	rΔ		· r	re		ماد	201	Hio	'n			U	7	P	<b>G1</b> (	000	<u>,                                     </u>
ť	uui 633	ning ii	ioue /	Орсос	10				<i>a</i> u		30	, ,	,, ,	.31	-			···			U	)	Ρ	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	т	#	Ρ	R	х	Υ	D	U	!	В	G	Г		C	R		
1/1						Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
48																					0	0	•	•	•	•

Mot	orola	ASL	Α.	<del> </del>									¢	CP	U	ty	/p	e I	3 /	6	80	9				
B&F	ł	SLA	١	Shif	t cont	er	nts	s c	of .	Α								Γ	-	f	1	1	†‡	+	_	0
Sho	rt			1 bit to the left  ode / Opcode Address presele												Č	;	d7				d0				
Δ,	drace	ina m	ode /	Shift contents of A   1 bit to the left									عاد		tic	'n			(	7	С	P	30			
_^	Jui 638	sing ii	ioue /	Орсос	16			^'	u		33	۰,	,, ,	531	-16	,,,		""				7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	Opcode         Address preselect           IND.         REL.         I   0   F   S   T   #   P   R   X   Y   E							D	U	!	В	G	Г		C	R							
2/1					ft contents of A  it to the left  de				Υ	D	U	!	В	G	Н	Ι	Ν	Z	٧	С						
48						Г		Π													×	0	•	•	•	•

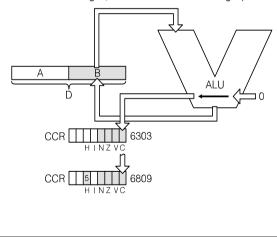
The contents of accumulator A will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag. When the instruction is executed the V flag is assigned with the result of N  $\oplus$  C (V = N  $\oplus$  C). The V flag shows whether bit 7 has been changed by the operation or not (V = 1 => bit 7 was changed; V = 0 => bit 7 was not changed).



Mot	1 bit to the left   Address preselva								(	CP	U	ty	ďρ	е /	Α/	6	30	3								
B&F	SLB												Γ	ŀ	1	Ŧ	Ţ	14	Ŧ	_	0					
Sho	rt			1 bit	to th	e I	lef	t										C		d7				d0		
	ddroes	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		·	ırc		ماد		tic	'n			O	7	P	<b>31</b> 0	000	)
	uui 638	ing ii	ioue /	Орсос	10	ontents of B the left  Address prese			-			<b>,,,</b>			U	)	Р	G-F	C							
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
1/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
58						Contents of B						П			П	0	0	•	•	•	•					

Mote	orola	ASL	В	Fun	ction												c	P	U	ty	р	e I	3 /	6	В0	9
B&R	t	SLB	-	Shif	t cont	er	nts	0	f I	3								Γ	1-	Ŧ	F	Ţ	Ţ	Ŧ	-	0
Sho	rt			1 bit	to the	e I	ef	t										Č	-	d7			_	d0		
۸۰	ddress	ina m	odo /	Oncor	5	Г		۸,	14	rn		· n			٠		io	n			C	7	С	P	30	
~	uui ess	ilig ili	loue /	Орсос	16			~	ıu		33	, P		.5	516	-		•••			0	7	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	О	F	S	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
2/1						Е	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	н	I	N	Z	٧	С
58																					×	0	•	•	•	•

The contents of accumulator B will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag. When the instruction is executed the V flag is assigned with the result of N  $\oplus$  C (V = N  $\oplus$  C). The V flag shows whether bit 7 has been changed by the operation or not (V = 1 => bit 7 was changed; V = 0 => bit 7 was not changed).

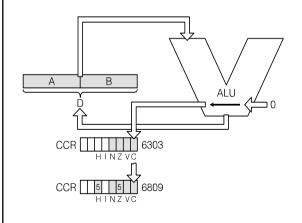


Mot	orola	ASL	_D	Fun	ction												(	CP	'n	ty	/p	e A	Α/	6	30	3
B&F	₹	SLE	)	Shif	t cont	er	ıts	6 0	f I	D				η.	4	f.I	Ŧ.	Τ,	I†	ŀ	Ţ.	ŦĮ	+.	IT.	ŀ	0
Sho	rt			1 bit	oit to the left															_				di	í	
Ad	ddress	sing n	node /	Орсос	de			Ad	dd	re	SS	s p	ore	ese	ele	ec	tic	n			0	)		G1( G-l		
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	CR		
1/1						Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
05																					0	0	•	•	•	•

Mote	orola	ASL	.D	Fun	ction												c	P	U	ty	/p	e I	В /	6	30	9
B&R	ł	SLD	)	Shif	t cont	er	nts	6	fΙ	D				Π	٠-[٠	F.I	11	1	I	T	FI	1,	+.	-	F	0
Sho	rt			1 bit	t to the	e I	ef	t						c	ď1	5				•				d(		
Δ	drace	1 bit to the left costs													عاد		·io	'n			V	)	С	P	30	Π
Ć	aui 633	ning ii	oue /	Орсос	10	30	, 1	,, ,	.31	510			···			U	)	Р	C	30						
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R		٦
4/2						Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
58 49						Г														П	×	0	•	×	•	•

The contents of accumulator D will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 15 is moved to the carry flag.

When the instruction is executed the V flag is assigned with the result of N  $\oplus$  C (V = N  $\oplus$  C). The V flag shows whether bit 15 has been changed by the operation or not (V = 1 => bit 15 was changed; V = 0 => bit 15 was not changed).



## 6809:

Since this instruction is not available in the 6809 it is simulated by the use of two other instructions.

ASLB T

The contents of B are moved 1 bit to the left. Bit 0 is set to logic 0 and bit 7 is moved to the carry flag.

ROLA

The contents of A are moved 1 bit to the left. The carry flag (=> bit 7 from B) is moved to bit 0. Bit 7

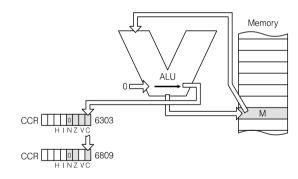
is moved to the carry flag.

Since the ASLD instruction is made up of two different instructions the state of the zero flag is undefined.

Mot	orola	LSF	}	Fun	ction												(	CP	'n	ty	/p	e A	Α/	6	30	3
B&F	ł	SR		Shif	t cont	er	nts	6 0	fΙ	М								0-	<b>-</b> [	+	F	Ţ	1		4	
Sho	rt			1 bit	to th	e	riç	jht	ŧ										d	17	_	_	_	d0	(	5
Δ	dress	ina n	ode /	Oncor	łe	Γ		Ad	44	re		: r	ıre	250	عاد	-	tic	'n			U	)		<b>G1</b> (		
		<b>g</b>	,	-рос.		L		•			-	7		_	•	•		•••			Ç		Ρ	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		
		6/3		6/2		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	z	٧	С
		74		64		Г	0	0	0	П		0	0	0	П		П		П	П	0	0	▼	•	•	•

Mot	orola	LSR		Fun	ction												(	CP	U	ty	ďρ	e I	3 /	6	80	9
B&F	₹	SR		Shif	t cont	er	nts	c	fΙ	M								0	<b>-</b> f	+	F	Ŧ	1	Ŧŀ	4	1
Sho	rt			1 bit	to th	e	rig	h	t										(	17	11-			d0	(	5
Δ,	ddroes	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		: n	rc		ماد		tic	'n			V	7	С	Pδ	30	
_^`	uui esa	ing ii	oue /	Орсос	10				<i>a</i> u		30	, 1	,, ,	.31	-	,,,		<b>,,,</b>			U	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
	6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
	04	74		64		Γ	0	0	0			0	0	0	•	•	•	•		•	0	0	▼	•	0	•

The contents of memory location M will be moved 1 bit to the right. Bit 0 is moved to the carry flag and bit 7 is replaced by logic 0.

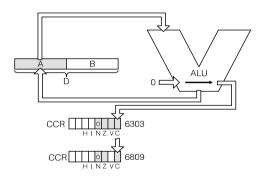


#### 6303:

Mot	orola	LSR	RA	Fun	ction												C	P	U	ty	pe	e /	۸/	63	30	3
B&F	₹	SRA	4	Shif	t cont	er	nts	so	f.	Α								0-	[	†-	7	F	Ŧ	F}-		1
Sho	rt			1 bit	to th	e	rig	jht	ŧ										d	7				d0	Ċ	Ī
A	ddress	sina m	node /	Oncor	de .	Γ		Αc	Нd	re	SS	: r	re	S	ele	ci	tic	n			0	7	PG			
		g	,	орос.		L		•			-	7						•••			Q	)	PG	-P	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
1/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	1	N.	z	٧	С

Mot	orola	LSR	A	Fun	ction												þ	P	U	ty	p	e E	3 /	6	В0	9
B&F	₹	SRA	١.	Shif	t cont	er	nts	c	f A	Α								(	) <del>~</del>	Π.	IJ	1-	1:	Ŧŀ	<u>-</u>	1
Sho	rt			1 bit	t to the	e I	rig	h	t											d7	_		_	d0	С	;
Δ,	ddress	ina m	ode /	Oncor	10	Г		Δ,	44	ro		· n	rc		عاد		tic	'n			V	7	С	P	30	
_^`	uui 633	iiig ii	oue /	Орсос	10				<i>a</i> u		30	, 1	,, ,	.31	,,,			···			U	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
2/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	Ν	Z	٧	С
44																					0	0	•	•	0	•

The contents of accumulator A will be moved 1 bit to the right. Bit 0 is moved to the carry flag and bit 7 is replaced by logic 0.

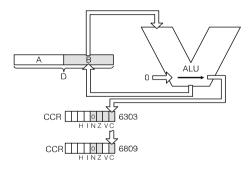


#### 6303:

Mot	orola	LSF	RB	Fun	ction												C	CP	U	ty	γp	е /	۸/	6	30	3
B&F	ł	SRE	3	Shif	t cont	er	nts	s o	f	В								0		†-	ļ	1	1	7	7	
Sho	rt			1 bit	t to th	e	rig	jht	ŧ										Ċ	d7				d0	Ċ	1
A	ddress	ing n	node /	Орсо	de	Γ		Ad	dd	re	SS	р	re	se	ele	ec	tio	n			0	)	PO	310 G-F		
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G			CC	R		
1/1						Ε	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	Η	Ι	Ν	Z	٧	С
54						Г	П		П								П				0	0	▼	•	•	•

Mot	orola	LSR	В	Fun	ction												C	P	U	ty	р	e E	3 /	68	30	9
B&F	₹	SRE	3	Shif	t cont	er	nts	6	f I	В									0-	П	I	1	Ħ	Ŧŀ	-	1
Sho	rt			1 bit	to th	e	rig	h	ŧ											d7				d0	C	,
$\Box$	ddress	ina m	odo /	Oncor	10	Γ		۸,	14	ra			ro		N/a		tio	'n			C	7	С	Pε	30	
_ ^	uui ess	oning in	loue /	Орсос	16			^'	Ju	16	38	, h	,, ,	30	316			""			C	)	Р	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		Τ
2/1						Ε	Α	М	F	Z	#	Р	С	-	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
54						Γ		Γ			Г										0	0	•	•	0	•

The contents of accumulator B will be moved 1 bit to the right. Bit 0 is moved to the carry flag and bit 7 is replaced by logic 0.

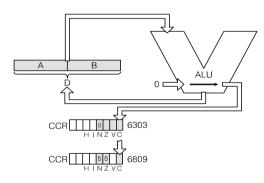


## **6303**:

Mote	orola	LSR	₹D	Fun	ction												C	CP	U	ty	p	e A	<b>A</b> /	63	30	3
B&R	ł	SRI	)	Shif	t cont	er	nts	s o	f I	D				0-	4	+1	+	It.	H	ŀŀ	ŀŀ	†]	+1	.†1	4	٦
Sho	rt			1 bit	t to the	e i	rig	jht	ŧ						ď.	15								d0	(	7
Δι	dress	ina m	node /	Oncor	de.	Г		Ad	44	re	•	: r	ıre		۱	201	tio	'n			V	1		<b>310</b>		╗
	au. 000	g	iouc,	Ороос	40						-	<b>'</b> Ի						···			U	<u> </u>	P	G-P	C	╝
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	O			CC	R		7
1/1						ш	Α	М	F	Z	#	Ρ	С	_	Υ	D	U	!	В	G	Ι	Ι	Ν	Z	٧	С

Mot	orola	LSR	D	Fun	ction												þ	P	U	ty	p	e E	3 /	68	30	9
B&F	₹	SRE	)	Shif	t cont	er	nts	6	f I	ס				0-	<b>-</b> F		+1	Ŧ	H	H	1	1	1	Ŧ	4	ī
Sho	rt			1 bit	to th	e	rig	ht	ŧ						d1	5	_				_			d0	(	5
A	ddress	ina m	ode /	Oncod	de	Γ		Δα	44	re	SS	: n	re	s	ele	c	tio	n			C	1		Pδ	_	
ļ ,,,		g		орос		L		•		. •	-	, 1		_	•	-		•••			C	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
4/2						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	н	Ι	Ν	Z	٧	С
44 56						Γ	Г	Г	П			Г	Г	П							0	0	x	x	0	•

The contents of accumulator D will be moved 1 bit to the right. Bit 0 is moved to the carry flag and bit 15 is replaced by logic 0.



#### 6303:

When the instruction is executed the V flag is assigned with the result of  $N \oplus C$  (V =  $N \oplus C$ ). The V flag shows whether bit 15 has been changed by the operation or not (V = 1 => bit 15 was changed;  $V = 0 \Rightarrow bit 15$  was not changed).

#### 6809:

Since this instruction is not available in the 6809 it is simulated by the use of two other instructions

The contents of A are moved 1 bit to the right. Bit 0 T.SRA is moved to the carry flag and bit 7 is set to logic 0.

The contents of B are moved 1 bit to the right. The RORB

carry flag (=> bit 0 from A) is moved to bit 7. Bit 0

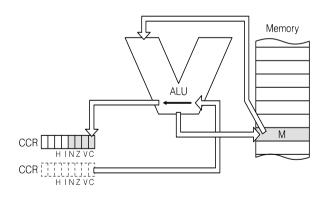
is moved to the carry flag.

Since the LSRD instruction is made up of two different instructions the states of the zero and negative flags are undefined.

Mot	orola	ROL	_	Fun	ction												C	P	U	ty	/p	e A	A A	6	30	3	l
B&F	₹	SLI		Rota	ate co	nt	er	nts	0	fΙ	VI								Щ		+	1+	1-	FT-	Ŧ	]	l
Sho	rt	RL		1 bit	t to th	e I	lef	t											C		17	+1:	+	1-1-	40		l
Δ	ddress	ina m	node /	Oncor	łe	Γ		A	44	re		: r	ıre	250	عاد	20	tio	'n			V	7	_	G1(		_	l
	uu. 000	,g	iouc,	Ороос		L			uu	٠.	•	'			٠.、			•••			C	)	Ρ	G-I	PC		ı
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR			ı
		6/3		6/2		Ε	Α	М	F	z	#	Р	С	I	Υ	D	U	!	В	G	Н	ī	N	z	٧	С	ı
		79		69		Г	0	0	0			0	0	0					П		0	0	•	•	•	•	ı

1	Mot	orola	ROL	-	Fun	ction												þ	CP	U	ty	/p	e I	В/	6	80	9
	B&F	₹	SLI		Rota	ate co	nt	er	ıts	c	fΙ	VI								ŀ	1+1	+		11	7-	FI.	j
	Sho	rt	RL		1 bit	t to th	e I	lef	t											c	, ,	17	T-1	T-1	_	d0	
ı	Δι	ddress	ina m	node /	IND. REL. I O F S T # P R X										عاد	20	tic	'n			0	)	_	P	_		
ı		uui 633	ing ii	ioue /	1 bit to the left / Opcode Address presele			510			<b>,,,</b>			V	)	Ρ	C	30									
ı	IMPL.	DIR.	EXT.	IMMED.	1 bit to the left  Opcode Ad  IND. REL. I O F 1 6+/2+ E A M I						Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
Ī		6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
ĺ		09	79		69			0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	•

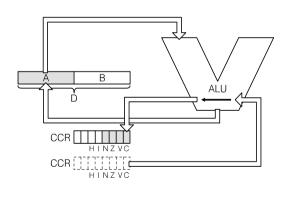
The contents of memory location M will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.



Mot	orola	ROI	-Α	Fun	ction												C	P	U	ty	/p	e A	A /	63	30	3
B&F	₹	RLA	4	Rota	ate co	nt	er	ıts	0	f /	4								rf.		+	1+	-1+	-1-	Ŧ	J
Sho	rt			1 bit	to the	e I	lef	t											닏	d	17	=1	+	+	d0	
A	ddress	sina n	node /	Орсос	de	Г		Αc	bb	re	SS	s r	re	ese	ele	c	tic	n			U	1	PC			_
												•				_					C		P	G-F	'n	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	-	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	O			CC	R		_
1/1						Ε	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С

Mot	orola	ROL	Α.	Fun	ction												þ	P	U	ty	/p	e E	3 /	6	80	9
B&F	ł	RLA	١	Rota	ate co	nt	er	ts	0	f A	١							1	<u>-</u>	٦.	ŦI	<del>-</del>	1+	.   -	FL	]
Sho	rt			1 bit	to th	e I	lef	t										ι	c	ď	7	-1-	<u>+1</u> :	+	d0	
Δ,	drace	ina m	ode /	Oncor	10	Γ		Δ,	44	rΔ		· n	rc		عاد		tic	'n			V	<u>1</u>	С	P	30	
_^`	aui 633	ning ii	oue /	Орсос	10				uu		30	, 1	,, ,	.31	510			···			V	<u>)</u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		
2/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
49						Г	Г														0	0	•	•	•	•

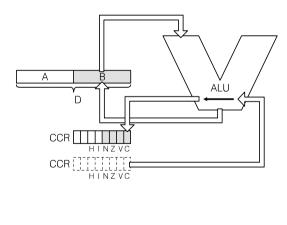
The contents of accumulator A will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.



Mot	orola	ROL	В	Fun	ction												C	CP	U	ty	/p	e /	۹/	6	30	3	Γ
B&F	₹	RLE	3	Rota	ate co	nt	en	ts	0	f E	3								Æ		+	+	14	-   -	=	]	l
Sho	rt			1 bit	to the	e I	ef	t											Ç.	ď	17	-1-	+1	+	90  -		l
Δ.	ddroes	ina m	node /	Oncor	ło.	Г		Δ,	44	rΔ		: r	re		ale	201	io	'n			U	1	P	<b>G10</b>	000	_	l
ť	uui 638	ing ii	ioue /	Орсос	10				<i>a</i> u		30	, ,	,, ,	.30	210			<b>,,,</b>			U	1	Р	G-F	'n		ı
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R			l
1/1						Ε	Α	М	F	Z	#	Р	С	Ι	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С	ı
59						Г													П		0	0	•	•	•	•	l

Mot	orola	ROL	В	Fun	ction												þ	P	U	ty	p	e I	3 /	6	30	9
B&F	ł	RLE		Rota	ate co	nt	en	ts	0	f E	3							ſ	۲.		ΗT	+	1+	1+	F].	]
Sho	rt			1 bit	to th	e I	ef	t										·	ਰ	d7	131	- 1-		**	d0	
Δ,	dress	ina m	ode /	Oncor	1e	Γ		Δι	44	re	٠,	: n	re		ele	·C	tic	'n			C	)		P	_	
^`	<b>24.00</b> 0	g	ouc,	Ороос		L						, ,						•••			C	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
2/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
59																					0	0	•	•	•	•

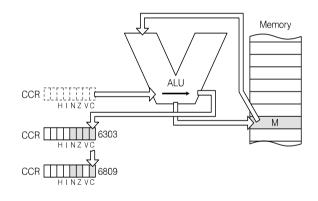
The contents of accumulator B will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.



Moto	orola	RO	R	Fun	ction										С	Pι	Jt	ype	•	A / 630	3
B&R		SRI	=	Rot	ate co	nte	nts	6 0	f M							ľ	<u>.</u>	<b>—</b>	+	1+1+1	$\Box$
Sho	rt	RR		1 bi	t to the	e riç	gh	t								(	7	d7	- 1-	d0	
۸,	Idrocc	ina n	node /	Onco	do		۸,	44	res	e n	ra		اما	·+i	or			0		PG1000	
Α.	iui ess	silig ii	loue /	Орсо	ue		^	uu	63	s h	16	30	10	<i>-</i> LI	UI.			0		PG-PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	RFI.	ıΩ	ΙF	s	Т #	Ι <sub>Ρ</sub> Ι	вT	×Τ	γſг	ıΓι	ıΤ	I <sub>E</sub>	G	г	_	CCR	7

Mot	orola	ROF	₹	Fun	ction												¢	CP	v	ty	/p	e E	3 /	6	80	9
B&F	ł	SRE	•	Rota	ate co	nt	er	ıts	0	f I	VI								f	Н	+	_ [ <del>]</del>	1	FI	Ŧì	]
Sho	rt	RR		1 bit	to th	e	rig	jh	t										С	' '	17	-1	_		d0	
$\Box$	droce	ina m	ode /	Oncor	10	Γ		۸,	44	ra					٠		tic	'n			0	ℷ	С	P	30	
_ ^	Jui 638	sing ii	ioue /	Орсос	16			^'	uu	16	33	۰,	,, ,	531	-10	,,,		""			V	<u>1</u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
	6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
	06	76		66		Γ	0	0	0			0	0	0	•	•	•	•	Γ	•	0	0	•	•	0	•

The contents of memory location M will be moved 1 bit to the right. The carry flag is moved to bit 7 and bit 0 to the carry flag.



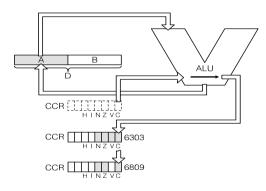
#### 6303:

76

Γ	Mot	orola	ROF	RA	Fun	ction												(	P	U	ty	ρę	e /	<b>A</b> /	63	30	3
Γ	B&F	ł	RRA	١.	Rota	ate co	nt	en	ts	0	f /	4								ď		+	П	T	FT-	Ŧl	
Ι	Sho	rt			1 bit	to the	е	rig	ht	:										c		±17	+ 1	+	-	ᇷ	
Γ	Ad	dress	ina m	ode /	Орсос	de	Г		Αc	ld	re	SS	s n	re	se	ele	c	tic	n			0	1	PG			
L			9		-р																	O	1	P	G-P	C	╝
E	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
I	1/1						Е	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Н	T	N	Z	٧	С

Mot	orola	ction												C	P	U	ty	р	e E	3 /	68	30	9			
B&F	₹	RRA	,	Rota	ate co	nt	en	ts	0	f A	١								Lf.	щ	+	17	T-1	-1-	Ŧl	]
Sho	rt			1 bit	to the	e i	rig	ht	ŧ										c	1 7	17	+ 1	+	+	ᇷ	
A	Addressing mode / Opcode							Α	44	re	SS	: n	re	S	ele	c	tio	n			C	1		Pε	_	
ļ ,,,		9		орос				•		. •	-	, 1		•		_		•••			C	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	MMED. IND. REL. I O F S T # P R X Y D U ! B G CCR																						
2/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	н	I	N Z V		٧	С
46						Г					Г		П								0	0	•	•	0	•

The contents of accumulator A will be moved 1 bit to the right. The carry flag is moved to bit 7 and bit 0 to the carry flag.

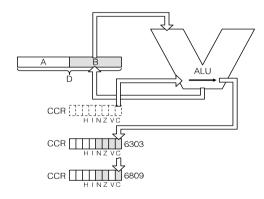


#### 6303:

Mot	orola	RO	RB	Fun	ction												C	P	U	ty	ре	<b>•</b> /	Α/	63	30	3					
B&F	₹	ate co	nt	er	nts	0	f E	3								f	JF	+	+	1-	- 1-	+1	]								
Short 1 bit to the right																			c	ď	7	-  -	+	+	d0						
Δ,	ddress	10	Γ		Ad	44	ro		: r	ırc		عاد		Hio	n		I	0	1	P	<b>310</b>	000									
	uui 63	sing ii	40			^	uu		30	, ,	,,,	.30	,,,			•••		_	0	1	Р	G-F	c								
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CCR								
1/1						Е	Α	М	F	z	#	Р	С	T	Υ	D	U	!	В	G	н	Ι	N Z V C								

Mote	orola	ROF	≀В	Fun	ction												þ	CP	U	ty	/p	e E	3 /	6	В0	9
B&R	ł	RRE	3	Rota	ate co	nt	en	ıts	0	f E	3								ď	-f	+	ET.	IT	I	= F]-	]
Sho	rt		t to the	e r	rig	jht	t										c	d	7		-		ᇷ			
Δ,	Addressing mode / Opcode							Δ,	44	lre					عاد		tic	'n			C	1	С	P	30	
_ ^`	Addressing mode / Opcode							_	14	16	30	, h	<i>,</i> 1 6	,50	510			".			0	<u> </u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	[	0	F	s	Т	#	Р	R	х	Υ	D	υ	!	В	G	Γ		CC	R		
2/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
56						Г	Г	Г		П	П	Г	Г								0	0	•	•	0	•

The contents of accumulator B will be moved 1 bit to the right. The carry flag is moved to bit 7 and bit 0 to the carry flag.



#### 6303:

## 3.10. BRANCH INSTRUCTIONS

Branch instructions are commands which change the program counter.

There are two types of branch instructions:

- unconditional branch instructions
- conditional branch instructions

Unconditional branch instructions are always executed.

**Conditional** branch instructions are only executed if certain conditions are true.

Motorola	B&R		Operatir	ng mode	
Wotorola	Dak	PG1000	PG-PC	CP 80	PC 80
BCC 1)	JC0 1)	0	0	0	0
BCS <sup>1)</sup>	SP<1)	0	0	0	0
BPL <sup>1)</sup>	J+ 1)	0	0	0	0
BMI <sup>1)</sup>	J- <sup>1)</sup>	0	0	0	0
BNE 1)	SN0 1)	0	0	0	0
BEQ 1)	SP0 <sup>1)</sup>	0	0	0	0
BHI <sup>1)</sup>	SP>1)	0	0	0 0 0	0
BLS <sup>1)</sup>	J<= 1)	0	0	0	0
SK0	SK0				0
SK1	SK1		0		0
JSR	SPU	0	0	0	0
RTS	RET	0	0	0	0
JMP	SPI	0	0	0	0
NOP	NOP	0	0	0 0 0	
END	END	О	О	0	0

Short relative branches (branch width: -128 to +127). By adding an "L" the short can be changed to a long branch instruction (branch width: -32768 to +32767). Only possible in PC 80 mode!

Example: BCCL or JC0L

Motorola	xxx	Function	CPU type A / 6303
B&R	xxx	Branch, if condition xx	x is true
Short	xxx		
			O 004000

۸,	ddroce	ina m	odo /	Oncor	40			۸,	44				ore		٠.,			'n			0	1	P	<b>310</b>	)00	,				
_ ^'	Addressing mode / Opcode							^'	u		33	, ,	,, ,	-31	-10	-6		,,,			О	ī	P	G-F	SC					
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	-	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CCR							
					3/2	Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	I	1	N Z V		٧	О				
					xxx		Г														0	0	0	0	0	0				

"xxx" in the table above stands for conditions which are defined below:

	Mnemonics		Condition	Oncodo
Motorola	B&R	Short	Condition	Opcode
BCC	JC0		C = 0	24
BCS	SP<	J<	C = 1	25
BEQ	SP0	J0	Z = 1	27
BHI	SP>	J>	$C \lor Z = 0$	22
BLS	J<=		C ∨ Z = 1	23
BMI	J-		N = 1	2B
BNE	SN0	J1	Z = 0	26
BPL	J+		N = 0	2A

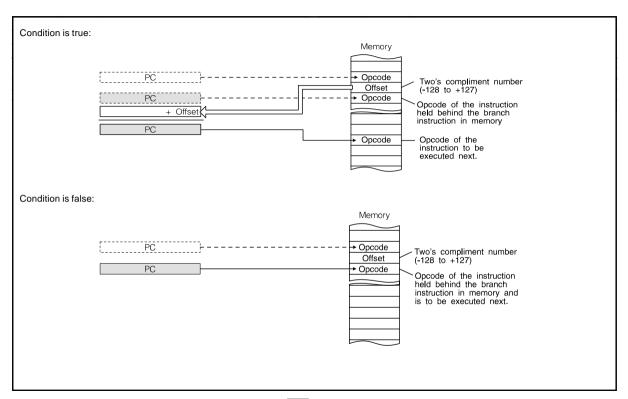
Mot	orola	xxx		Fun	ction												þ	P	U	ty	pe	e E	3 /	6	В0	9
B&F	t	xxx		Brai	nch, if	С	or	d	iti	on	X	XX	( is	s t	ru	e										
Sho	rt	xxx																								
Δ,	Addressing mode / Opcode							Δι	44	re	99	: n	re	250	عاد	20	tio	'n			0	1	С	P	30	
l ^``	au. 000	g	ouc,	Ороос		L					-	, ,			•••			•••			0	1	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	Y D U ! B G CCR										Τ
					3/2	Е	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	1	I N Z V		٧	С
					xxx																)	0	0	0	0	ા

Short relative branches are made up of two bytes:

- 1) The first byte is the instruction opcode.
- 2) The second byte is the offset which defines where the program should be re-routed when the condition is true. This offset is a two's complement number and can accept values from -128 to +127. The offset always refers back to the address of the next instruction after the conditional branch.
  A label is entered in the STL input line as the branch

A label is entered in the STL input line as the branch destination. The B&R PROgramming SYStem calculates the offset on its own. If a branch is impossible because of being outside of the range (-128 to +127) it is shown in the input line with a "+" before the label. In this case the program cannot be transferred to the PLC and the following error message is displayed:

E051 Invalid Branch



Motorola	Function	(	CPU type	A / 6303
B&R				
Short				
	 			PG1000

ן נ	G100	PC					'n	٠	ect	٠.,					-	44	۸.		ı	do	Oncor	ode /	ina m	ddroes	
П	G-P	P	Τ				"			-10	.5	,, e	, 1	33		Ju	~'		L	16	Орсос	ioue /	sing ii	uuies	Α.
⊐	R	CC			G	В	!	U	D	Υ	х	R	Ρ	#	Т	s	F	0	ı	REL.	IND.	IMMED.	EXT.	DIR.	IMPL.
С	Z١	Ν	1	Η	G	В	!	U	D	Υ	I	С	Ρ	#	Z	F	М	Α	Е						
П	П			П	П	Г		П			Г		Ī		Γ	Г	Г	Γ	Γ						

<sup>&</sup>quot;xxx" in the table above stands for the conditions defined below:

	Mnemonics		Condition	Opcode
Motorola	B&R	Short	Condition	Opcode
BCCL	JC0L		C = 0	10 24
BCSL	SP <l< td=""><td></td><td>C = 1</td><td>10 25</td></l<>		C = 1	10 25
BEQL	SP0L		Z = 1	10 27
BHIL	SP>L		$C \vee Z = 0$	10 22
BLSL	J<=L		C ∨ Z = 1	10 23
BMIL	J-L		N = 1	10 2B
BNEL	SN0L		Z = 0	10 26
BPLL	J+L		N = 0	10 2A
				•

Mot	orola	xxx		Fun	ction												þ	P	U	ty	γp	e E	3 /	6	80	9
B&F	1	xxx		Brai	nch, if	С	on	d	iti	on	X	XX	( is	s t	ru	e										
Sho	rt																									
Δ,	ldress	ina m	ode /	Oncor	łe			Δι	44	re	S 6	: n	re	250	ele	201	tio	n				Ι	_	P		
	au. 000	g	ouc,	ороос							•	, ,			٠.,			•••			О	ı	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	I	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		
					5(6)/4	Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
					xxx	Г															0	0	0	0	0	0

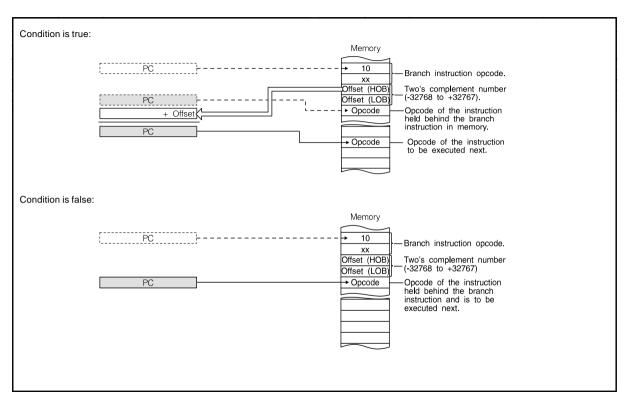
Long relative branch consists of four bytes:

- 1) Bytes 1 and 2 are the opcode of the instruction.
- 2) Bytes 3 and 4 are the offset which define where the program should be re-routed when the condition is true. This offset is a two's complement number and can accept values between -32768 and +32767. The offset always refers back to the address of the next instruction after the conditional branch. The entire memory range of a PLC can be covered with the two byte offset which means there is no limitation as with short relative branch instructions. A label is entered in the STL input line as the branch destination. The B&R PROgramming SYStem calculates the offset on its own.

Execution time: 5(6) Clock cycles

Condition true, branch executed.

Condition false, branch not executed.



																	_				•					
B&F	₹	SKO	)	Skip	next	in	st	ru	ct	io	n,	if	Z	=	1											
Sho	rt				ncode Address preselection																					
	ddroes	ina n	odo /	Oncor	pcode Address preselection														Г	Τ	P	G1(	000	5		
Α.	uuies	sing ii	loue /	Орсос	ue	L	Address preselection														O	<b>,</b>	Р	G-F	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		_
					3/2	Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	Ν	Z	٧	С
					27	Γ	Γ					Γ									0	0	0	0	0	0

Function

Mot	orola	SK0	)	Fun	ction												c	P	U	ty	pe	e E	3 /	6	В0	9
B&F	t	SK0		Skip	next	in	st	ru	ct	io	n,	if	Z	=	1											
Sho	rt																									
A	dress	ina m	ode /	Oncod	de .	Γ		Δc	44	re	SS	: n	re	s	ele	c	tio	n		╗		I		P8		
ļ ,,,		9		орос		L		•		. •	-	, 1		_	•	-		••			Q		Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
					3/2	Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
					27	Γ			Г					Г							Э	0	0	0	0	ာ

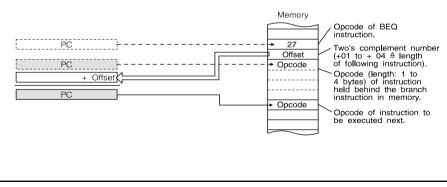
The following instruction is skipped if the zero flag is set to logic 1 (previous operation resulted in 0).

This instruction corresponds with the BEQ instruction. The offset for the branch instruction corresponds with the opcode length of the following instruction. The PROgramming SYStem sets this value automatically.

CPU type A / 6303

#### Condition is true:

Motorola SK0



## Condition is false: Memory Opcode of the BEQ instruction. PC 27 Two's complement number $(+01 \text{ to } +04 \triangleq \text{length of the following instruction}).$ Offset PC → Opcode Opcode (length: 1 to 4 bytes) of the instruction held behind the branch instruction to be executed Opcode next

					••												_ [*		_	٠,	-	-		_		_
B&F	₹	SK1		Skip	next	in	st	ru	ct	io	n,	if	Z	=	0											
Sho	rt																									
Δ,	ddroes	ina n	node /	Oncor	10	Г		Δ,	14	rΔ		: r	re		ماد	20	tic	'n				I	P	<b>G1</b> (	000	)
	uuiesa	ing ii	ioue /	Орсос	10				<u> </u>		30	, ,	,, ,	.31				<b>,,,</b>			V	)	Р	G-F	PC	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			C	R		
					3/2	Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ı	N	z	٧	С
					26	Γ	Γ								Γ	Γ	Π				0	0	0	0	0	0

Function

Mot	orola	SK1		Fun	ction												þ	P	U	ty	р	e I	3 /	6	В0	9
B&F	t	SK1		Skip	next	in	st	ru	ct	io	n,	if	Z	=	0											
Sho	rt																									
	droco	inan	ode /	0000	10	Г		۸,	14						ele							Т	С	P	30	Т
^	uuless	ing ii	ioue /	Opcoo	Je			Α(	Ju	ıe	58	· P	,, e	:51	316		LIC	""			C	)	Р	С	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	CR		Т
					3/2	Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
					26																0	0	0	0	0	0

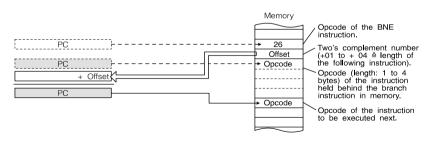
The following instruction is skipped if the zero flag is set to logic 0 (previous result of operation is not zero).

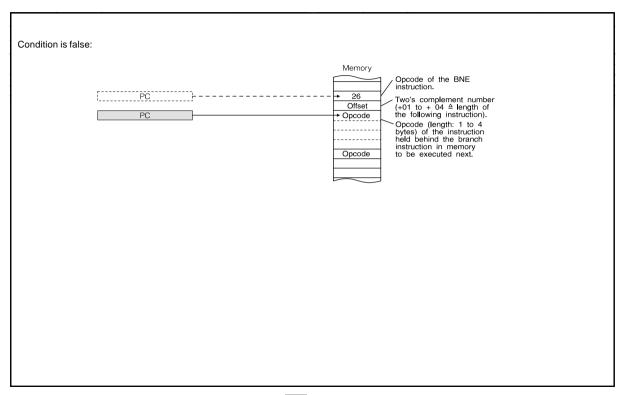
This instruction corresponds with the BNE instruction. The offset for the branch instruction corresponds with the opcode length of the following instruction. The PROgramming SYStem sets this value automatically.

CPU type A / 6303

#### Condition is true:

Motorola SK1





Mot	orola	JSF	ł	Fun	ction												C	P	U	ty	/p	e /	<b>A</b> /	6	30	3
B&F	₹	SPL	J	Unc	onditi	OI	na	۱k	ora	ın	ch	i	n a	1 5	ul	b-l	pr	og	jra	ım	1					٦
Sho	IO PG1000																									
Λ.	Addressing mode / Oncode   Address preselection																									
ť	uui ess	sing ii	loue /	орсос	16			^'	uu		33	۰,	,, e	.51	-10	-		""			О	ī	P	G-F	c	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		٦
	5/2	6/3		5/2		Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	1	Ν	Z	٧	С
	9D	BD		AD		Γ	Γ														0	0	0	0	0	0

Mot	orola	JSR		Fun	ction												C	P	U	ty	/p	e E	3 /	6	80	9
B&F	1	SPU	1	Unc	onditi	OI	na	Ιk	ra	ın	ch	i	ı a	1 5	ul	<b>)</b> -	pr	og	jra	am	1					
Sho	rt																									
$\Box$	ddress	ina m	odo /	Oncor	10	Γ		۸,	14						٠		tio	'n			a	1	С	Pδ	30	
_ ^	Jui 638	ing in	oue /	орсос	16			^'	Ju	16	38	۰,	,, ,	.51	-16	,,,		""			0		Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		Ī
	7/2	8/3		7+/2+		Ε	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
	9D	BD		AD		Γ															0	0	0	0	0	0

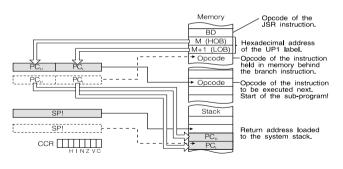
This instruction causes a jump to the desired location in the program. Before the jump the instruction address (=> return address) which is held behind the branch instruction is placed on the system stack.

If a label is given in the STL input line to which the program should jump to the PROgramming SYStem replaces it with the hexadecimal address of the label in the PLC memory.

Stack processing varies between the two types of processors:

Example: JSR UP1

6303:



### 6809: Memory Opcode of the JSR instruction. BD M (HOB) Hexadecimal address M+1 (LOB) of the UP1 label. Opcode of the instruction held in memory behind the branch instruction. → Opcode → Opcode Opcode of the instruction to be executed next. Start of the sub-program! Stack SP! PC, Return address loaded PC, to the system stack. HINZVC

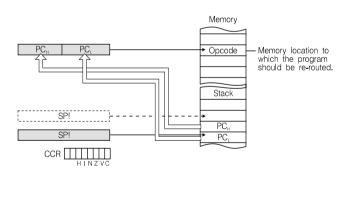
Mot	orola	RTS	3	Fun	ction												(	CP	U	ty	/p	e A	Α/	6	30	3
B&F	₹	RET	Г	Retu	ırn jur	mį	o f	ro	m	s	ub	)-p	orc	og	ra	m										٦
Sho	rt		add / Opposite Address presslession O PG10/															-1								
	ort OPG1000 Addressing mode / Opcode Address preselection OPGPC														П											
A	uures	sing ii	ioue /	Opcoo	Je			A	ıu	ıe	55	, ŀ	,, e	:51	316	30	···	"			a	•	Р	G-F	С	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	CR		٦
5/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
39																					0	0	0	0	0	0

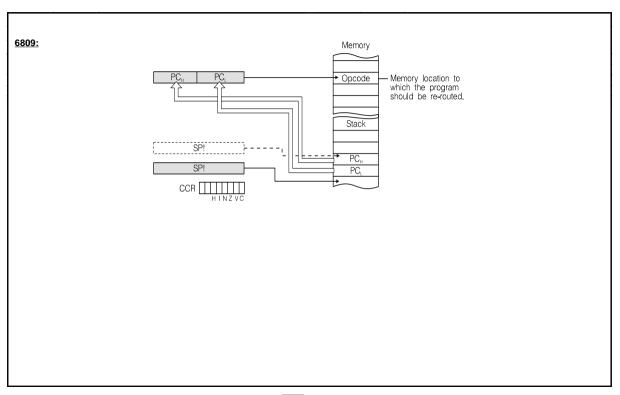
Mot	orola	RTS		Fun	ction												C	P	U	ty	ďρ	e E	3 /	68	30	9
B&F	t	RET		Retu	ırn jui	mį	p f	ro	m	s	uk	)-p	rc	g	ra	m										
Sho	rt																									
Δ,	ddress	ina m	ode /	Oncor	1e	Γ		Δι	44	re	٠,	: n	re	-56	ale	·C	tio	'n			0	1		Pε	_	
l ^``	au. 000	g	ouc,	Ороос		L				٠٠	٠.	, ,		-				•••			0		Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		Т
5/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	Ι	Ν	Z	٧	С
39						Γ					Г										0	0	0	0	0	0

This instruction causes a return jump to the address located on the system stack. If a so called sub-program is called with the JSR instruction it must be quit with RTS, otherwise a "STACK FAILURE" can occur.

Stack processing varies between the two types of processors:

#### 6303:





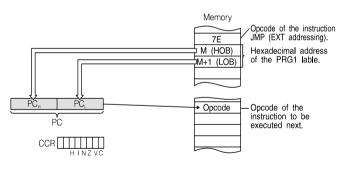
Mot	orola	JMF	•	Fun	ction												C	CP	U	ty	/p	e A	Α,	6	30	3	ı	M
B&F	₹	SPI		Unc	onditi	or	ıa	Ιjι	un	np	,																	В
Sho	rt																											s
Δ,	ddraed	ina n	node /	Onco	40	Γ		Αc	44	ro		: r	ırc	200	ماد	20	tic	'n			С	<u> </u>	P	<b>G1</b> (	000	0	1	Г
	uui 63	onig ii	ioue /	Орсо	46	L		^	u		30	, ,	,,,	-3	-			•••			C	)	Р	G-I	PC		l	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		C	R			1	IMF
		3/3		3/2		Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	N	z	٧	С	1	
		7E		6E																	0	0	0	0	0	0		

Mot	orola	JMF	•	Fun	ction												C	P	U	ty	p	e E	3 /	68	30	9
B&F	ł	SPI		Unc	onditi	OI	ıa	l jı	un	ıρ																
Sho	rt																									
Δ,	dress	ina m	ode /	Oncor	łe	Γ		Δι	44	re	٠,	: n	re	156	ale	·C	tio	'n		╗	O	1		Pε	_	
l ^``		g	ouc,	Ороос		L				٠٠	٠.	, ,						•••			a		Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		_
	3/2	4/3		3+/2+		Ε	Α	М	F	Z	#	Р	С	Ι	Υ	D	U	!	В	G	Н	Ι	Ν	Z	٧	С
	0E	7E		6E		Γ	Г					Г	Г							П	)	0	0	0	0	0

This instruction causes a jump to the desired location in the program.

If a label is given in the STL input line to which the program should jump to the PROgramming SYStem replaces it with the hexadecimal address of the label in the PLC memory.

Example: JMP PRG1



Motorola	NOP	Function	CPU type A / 6303
B&R	NOP	No operation	
Short			

	ddroe	ina m	ode /	Oncor	40	Г		۸,	44	rn			re		٠		Hio	'n			0	7	P	<b>310</b>	000	
Α.	uui es	sing ii	ioue /	Орсос	16			~	u		33	, ,	,, e	.5	-10			"			O	7	Р	G-F	Š	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		٦
1/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ī	Ν	Z	٧	С
01						Г	Г	П	Г			П	П	П			П		П		0	0	0	0	0	0

This instruction has **no** function. Its sole purpose is to cause a break where for a certain period of time (=> 1 processor cycle) nothing is done.

Mot	orola	NOF	•	Fun	ction												þ	CP	U	ty	/p	e I	3 /	6	80	9
B&F	ł	NOF	•	No	perat	io	n																			
Sho	rt																									
Δ,	drace	ina m	ode /	Onco	10	Γ		Δ	44	ro	٠.	s p	rc		عاد		Hic	'n			O	<u>,                                    </u>	С	P	30	
_^`	aui 633	ning ii	oue /	Орсо	10				uu		3.	, ŀ	,, ,	.31	510			<b>,,,</b>			U	)	Р	С	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		_
2/1						Ε	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
12						Γ	Γ	Г				Γ									0	0	0	0	0	0

This instruction has **no** function. Its sole purpose is to cause a break where for a certain period of time (=> 2 processor cycle) nothing is done.

Mot	orola	END	)	Fun	ction			_							CF	'n	ty	ре	Α	63 /	03	T	Mote	orola	E	ND		Fun	ction									C	PI	U ty	/pe	В/	680	,
B&F	'n	END		Prog	gram	en	d!															Γ	B&R	1	E	ND		Pro	gram	en	d!													
Sho	rt			Star	t in p	rog	gra	m	lin	e 0											- 1	Γ	Sho	rt				Sta	rt in p	ro	gra	m	line	9 0										
Δ,	ddress	ina m	node /	Oncor	łe.	Г	_	, de	dre		nr	256	lei	-ti	nη		Ţ	0		G100	_	Γ	Δι	ldres	sina	mο	de /	Орсо	de	Г	_	Λdc	ire	22	nre		lec	tio	n		0	_	P 80	_
1 ,,,		g		орос			•				۳.,				•		_ !	0	Ρ	G-P	C	- [			····9			ороо	ram end!   t in program line 0						•••		0	P	C 80					
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	П	0 1	s	т	# F	R	х	ΥC	U	!	В	G		C	CR	П	Γ	IMPL.	DIR.	EXT	. IN	MED.	IND.	REL.	ī	o l	FS	т	# P	R	X.	ΥD	U	! !	В G		CC	R	
						Е	A I	ΛF	z	# 8	c	П	ΥC	U	!	В	G	ΗΙ	N	Z١	/ C	Γ								Е	Αľ	ИF	Z	# P	С	1	Y D	U	! !	ВG	Н	N	z v	5
		7E C0 00				П	T	Т	П	T								)		<b>A</b> 1	7 ▼	I	11 3F							Π			П	Т				П			) <b>T</b>	0	ာ	5

This instruction causes a jump to the PLC operating system. This jump is done differently for each of the two CPU types:

**6303:** With an unconditional jump "JMP" to address \$C000 a jump is performed directly to the operating system.

 $\underline{\textbf{6809:}}\;\;$  The operating system is called by a software interrupt.

After several tests (e.g. test for stack failure) are run from the operating system a jump to line 0 of the user program is made.

### 3.11. MISCELLANEOUS INSTRUCTIONS

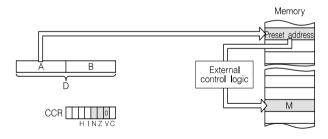
Matanala	Den		Operatin	g mode	
Motorola	B&R	PG1000	PG-PC	CP 80	PC 80
PRS	PRS	0	0	0	0
RST	RST	Ö	Ö	Ö	Ö
CLR	CLR	0	0	0	0
CLRA	CLA		0		0
CLRB	CLB		0		0
SET	SET	0	0	0	
CLC	CLC	0	0	0	0 0
SEC	SEC	0	0	0	0
CLI	CLI	0	0	0	0
SEI	SEI	0	0	0	0
COM	K	0	0	0	0
COMA	COA		0		0 0
COMB	COB		0		0
DAA	DK	0	О	О	О

Motorola PRS F	Function												C	P	U	ty	/p	е /	4 /	6	30	3
B&R PRS I	If d0 from	Α	=	1:	: 1	Ε	<del>)</del> (	M	)													1
Short P I	If d0 from	A	=	0:	(	M)	r	en	na	in	S	un	cl	na	ηę	јe	d					
Addressing mode / Op	ncodo			۸,	14	rn			re				Hio	n			О	•	P	<b>G1</b> (	000	П
Addressing mode / Op	pcode				ıu		33	۰,	,, e	30	216	-		""			O	ŀ	Ρ	G-F	Š	
IMPL. DIR. EXT. IMMED. IN	IND. REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		٦
4/3		Е	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
B7			0	0	0												0	0	•	•	•	0

Mot	orola	PRS	3	Fun	ction												c	P	U	ty	/p	e I	В	6	В0	9
B&F	₹	PRS		If do	) from	Α	=	1	: 1	п	> (	(M	)													٦
Sho	rt	Р		If do	from	Α	=	0	(	M)	r	en	na	in	S	un	cł	ıa	nç	јe	d					١
	ddroce	ina m	ode /	Oncor	10	Г		۸,	14	rn		. n			٠		io	n			C	7	С	P	30	٦
A	uuress	ing ii	ioue /	Opcoo	Je			Α(	Ju	ıe	58	· P	,, e	:51	316	:0	.10	""			C	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		٦
		5/3				Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
		В7					0	0	0												0	0	•	•	•	0

The 1 bit memory location M will be loaded with logic 1 if data bit 0 from accumulator A is logic 1, otherwise the contents of M remain unchanged.

The 6303 and 6809 do not hold the PRS instruction. PRS is actually a store instruction (STAA). This preset function is done through the hardware. For this purpose 1 bit memory locations (preset addresses) with address preselections O, F and S are available. The PROgramming SYStem replaces the entered M address with the corresponding preset address by transferring it in the PLC.



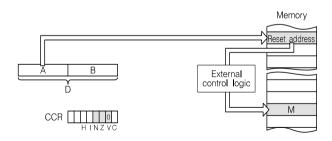
PRESETs for outputs G to N are impossible for memory reasons!

Mot	orola	RST		Fun	ction												C	CP	U	ty	/p	e A	Α/	6	30	3
B&F	₹	RST	_	If do	from	Α	=	1	: (	) =	<del>&gt;</del> (	М	)													٦
Sho	rt	R		If do	from	Α	=	0	: (	M)	r	en	na	in	s	un	cł	ha	ng	јe	d					-
۸,	ddroed	ina m	node /	Oncor	10	Г		۸,	14	ro		. ,	re		\/\		Hio	'n			С	•	P	<b>31</b> 0	000	П
A	uures	sing ii	ioue /	Opcoo	Je			A	ıu	16	55	١,	,, e	:51	316	30	LIC	"			O	•	Р	G-F	C	٦
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		٦
		4/3				Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
		B7					0	0	0												0	0	•	•	•	0

Mot	orola	RST		Fun	ction												þ	P	U	ty	/p	e I	В /	68	30	9
B&F	₹	RST		If do	from	Α	=	1	: 0	) =	> (	(M	)													٦
Sho	rt	R		If do	from	Α	=	0	(	M)	r	en	na	in	S	un	ıcł	na	nç	јe	d					-
Δ,	ddress	ina m	ode /	Oncor	łe	Г		Δ	44	re	٠,	: n	re	250	ale	20	tio	n			C	)		Pε	_	$\Box$
l ^``	uu. 000	g	ouc,	Ороос						٠٠	٠.	, ,			•••			•••			C	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	R		
		5/3				Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ī	N	Z	٧	С
		B7				Г	0	0	0		Г						П				ા	0	•	•	•	ା

The 1 bit memory location M will be loaded with logic 0 if data bit 0 from accumulator A is logic 1, otherwise the contents of M remain unchanged.

The 6303 and 6809 do not hold the RST instruction. RST is actually a store instruction (STAA) to a so called reset address. This reset function is done through the hardware. All 1 bit memory locations with address preselections O, F and S are assigned with reset addresses. The PROgramming SYStem replaces the entered M address with the corresponding reset address by transferring it in the PLC.



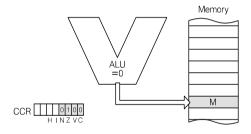
RESETs for outputs G to N are impossible for memory reasons!

Motorola	CLR	Function		CPU type	A / 6303
B&R	CLR	0 ⇒ (M)			
Short	С	1			
Address	ing mode /	Opcode	Address preselection	on O	PG1000

۸.	ddroed	ina n	node /	Oncor	40	ı		۸.	44	ro					٠ı،	ect	Hic	'n			C	1	P	G1(	)00	)
^	uuies	sing ii	loue /	Орсос	16			~'	u		33	, 1	,, e	.5	-10			"			O	7	Р	G-F	S	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		C	CR		П
		5/3		5/2		Е	Α	М	F	z	#	Ρ	С	I	Υ	D	U	!	В	G	Н	ī	N	Z	٧	С
		7F		6F			0	0	0			0	0	0							0	0	•	•	•	•

					_																						
	Mot	orola	CLF	₹	Fun	ction												C	CP	U	ty	/p	e I	В/	6	80	9
ı	B&F	₹	CLF	₹	0 ⇒	(M)																					
	Sho	rt	С																								
	Λ.	ddress	ina m	odo /	Oncor	10	Γ		۸,	14	ro		s p			۸۱,		Hio	'n			C	7	С	Р	30	
	ť	uui ess	ing ii	ioue /	Орсос	16			^(	Ju		33	• P	,, e	73	CIC	-		""			C	)	Р	С	30	
	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
		6/2	7/3		6+/2+		Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
ı		0F	7F		6F		Γ	0	0	0	Γ	Γ	0	0	0	•	•	•	•	Г	•	0	0	▼	•	▼	▼

Memory location M will be cleared (its contents are set to logic 0).

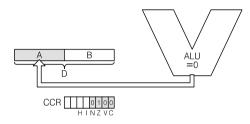


Motorola	CLRA	Function	CPU type A / 6303
B&R	CLA	0 ⇒ A	
Short			

A	ddress	sing m	ode /	Орсос	de			Ad	bb	re	ss	s p	re	se	ele	ec	tic	n			00		P(	310 G-P		Ξ
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R		
1/1						Е	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С
4F						Г	Г		П			Г	П				П				0	0	▼	•	•	•

Mot	orola	CLR	A	Fun	ction												þ	P	U	ty	/p	e I	3 /	6	80	9
B&F	ł	CLA	,	0 ₽	Α																					
Sho	rt																									
	ddress	ina m	odo /	Oncor	10	Γ		۸,	44	re					٠.,		Hic	'n			С	7	С	P	30	
Ĭ	uuress	ing ii	loue /	Opcoo	Je			A	Ju	ıe	58	, h	11 6	:50	sie	:0	LIC	""			O	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		CC	CR		Ī
2/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	I	N	Z	٧	С
4F						Г	Г		П		Г	П	П								0	0	▼	•	▼	▼

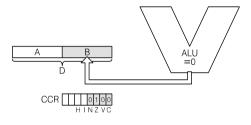
Accumulator A will be cleared (its contents are set to logic 0).



Mot	orola	CLF	≀В	Fun	ction												(	P	U	ty	pe	9 /	A /	63	30	3
B&F	₹	CLE	8	0 ⇒	В																					
Sho	rt																									
A	ddress	sina m	ode /	Oncor	de	Γ		Αc	44	re	SS	r	re	S	ele	ec:	tic	n			0	1	PG	_	_	_
1 '''		g		орос.		L		•		. •	-	r			•	-		•••			0	ı	P	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	1	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			CC	R		
1/1						-	Α.		-	7	44	٥	С	-	W	D			_	$\sim$		-	N	7	11	$\overline{}$

Mot	orola	CLF	≀В	Fun	ction												þ	P	U	ty	p	e E	3 /	6	В0	9
B&F	t	CLE	3	0 ⇒	В																					
Sho	rt																									
Δ,	dress	ina m	ode /	Oncor	łe	Γ		Δι	44	re	S 6	s p	re	256	۱e	·C	tio	'n			С	<u> </u>		P		
l ^``	au. 000	g	.ouc,	Ороос		L					٠.	, ,						•••			C	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	CR		_
2/1						Ε	Α	М	F	Z	#	Ρ	С	Ι	Υ	D	U	!	В	G	н	I	Ν	Z	٧	С
5F						Г	Г	Г	Г		Г	Г	Г					П			0	0	•	•	•	▼

Accumulator B will be cleared (its contents are set to logic 0).



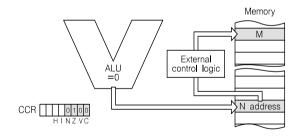
Motorola	SET	Function	(	CPU type	A / 6303
B&R	SET	1 ⇒ (M)			
Short					
				О	PG1000

۸,	ddroes	ina m	ode /	Oncor	10	Г		۸,	44						٠.,	ect	Hic	'n			О	7	P	310	000	П
Α.	uuies	ing ii	ioue /	Орсос	16			^'	u		33	۰,	,, e	.5	-10			"			О	7	Р	G-F	Š	П
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		٦
		5/3				Е	Α	М	F	Z	#	Ρ	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
		7F					0	0	0												0	0	•	•	•	▼

Mot	orola	SET	•	Fun	ction												C	Р	U	ty	'n	e I	В /	6	30	9
B&F	ł	SET	•	1 ⇨	(M)																					٦
Sho	rt																									1
Δ,	dress	ina m	ode /	Oncor	łe	Г		Δ,	14	re	•	: n	re	-56	ale	·ci	io	n			С	1	С	P	30	┒
l ^``		g	.ouc,	Ороос					•		-	, 1		-							С	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	_	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			C	CR		П
		7/3				Е	Α	М	F	Z	#	Ρ	С	1	Υ	D	U	!	В	G	Η	I	N	Z	٧	С
		7F					0	0	0				П								0	0	•	•	•	₹

1 bit memory location M will be loaded with logic 1.

The 6303 and 6809 processors do not contain the SET instruction in their basic command set. The SET function is performed through hardware. SET is actually a CLR instruction which corresponds with an N address (negation). All 1 bit memory locations with address preselections O, F and S are assigned with an N address. The PROgramming SYStem replaces the entered address M with the corresponding N address by transferring it in the PLC.



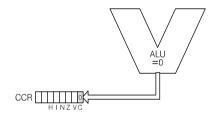
SET for outputs G to N is impossible for memory reasons!

Mot	orola	ola	CLC	;	Fun	ction												С	Pι	J t	уp	е	A / 6303
B&F	ł		CLC	;	0 ⇒	С																	
Sho	rt																						
۸,	droce	rocc	ina m	ode /	Oncor	10	Γ		۸,	14	rn		. n	ro		J.	ct	in	$\overline{}$		C	)	PG1000
Α.	uui es	1633	mg m	ioue /	орсос	16			~(	Ju		33	, P		30	,,,	·CL	101	•		C	7	PG-PC
IMPL.	DIR.	IR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	х	Υ	D	U !	Е	3 0	ī		CCR
							1	٠.			П		$\overline{}$	_			_		7.			Τ.	

0C

Mot	orola	CLC	;	Fun	ction												C	P	U	ty	ďρ	e E	3 /	68	30	9
B&F	t	CLC	;	0 ⇒	С																					
Sho	rt																									
Δι	ddress	ina m	ode /	Oncor	łe	Γ		Δι	44	re		s p	re		۱e	c	tio	'n			С	<u> </u>		Pδ	_	
	au. 000	g	ouc,	Ороос		L				٠.	•	, ,		-				•••			С	)	Ρ	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
3/2						Е	Α	М	F	Z	#	Ρ	С	1	Υ	D	U		В	G	Η	I	Ν	Z	٧	С
1C FE						Γ							П							П	0	0	0	0	0	▼

The carry flag will be cleared (the contents are set to logic 0).

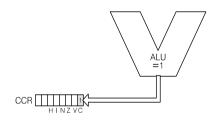


Motorola	SEC	Function		CPU type	A / 6303
B&R	SEC	1 ⇒ C			
Short					
Address	ing mode /	Oncode	Address preselecti	on O	PG1000
Auditos	ing mode /	Opcode	Address preselective	··· In	PG-PC

	ddroes	ina m	ode /	Oncor	10	Г		۸,	44	ro					٠.	ect		'n			О	)	P	310	000	
Α.	uui ess	sing ii	ioue /	Орсос	16			~	u		33	, ,	,, e	.5	-10			,,,			О	7	Р	G-F	'n	П
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	-	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		٦
1/1						Е	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
0D																					0	0	0	0	0	•

Mot	orola	SEC	;	Fun	ction												C	P	U	ty	γpe	e E	3 /	68	В0	9
B&F	₹	SEC	;	1 ⇒	С																					
Sho	rt			<u> </u>																						
Δ,	ddraes	ina m	node / (	Oncor	40	Г		Δ,	44	ro		s p	re	-	عاد	c	Hin	n			О	1	С	Pε	30	
_ ^`	Jui 633	mg m	oue / v	Specie	10	Ĺ		_				, p	-	,30	,,,			···			O	•	Ρ	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.		0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		Π
3/2						E	Α	М	F	Z	#	Р	С	Τ	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
1A 01						Г	Г	Г	П	П	П	П	П						П		0	0	0	0	0	Ā

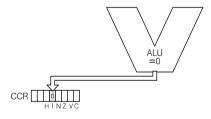
The carry flag will be set (contents set to logic 1).



Mot	orola	CLI		Fun	ction												C	٦,	'U	ty	/p	e A	A /	6	30	3
B&F	₹	CLI		0 ⇒	I																					
Sho	rt																									
	ddroes	ina n	node /	Oncor	10	Γ		۸,	44	rn			re		٠		4ic	'n			О	7	P	<b>G1</b> (	000	)
^'	uuies	sing ii	loue /	Орсос	16			~'	uu		33	۱ د	,, e	-31	-10	-		"			O	7	Р	G-F	S	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
1/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Η	I	Ν	Z	٧	С
0E						Γ	П					П									0	•	0	0	0	0

Mot	orola	CLI		Fun	ction												þ	P	U	ty	р	e E	3 /	6	В0	9
B&F	t	CLI		0 ⇨	I																					
Sho	rt																									
$\Box$	droce	ina m	ode /	Oncor	10	Γ		۸,	44						N/		tio	'n			С	<u> </u>	С	Pδ	30	_
_ ^	Jui 633	ilig ii	loue /	орсос	16			^'	uu	16	38	, h	,, ,	.5	216			""			C	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G			CC	R		_
3/2						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
1C EF																					0	•	0	0	0	0

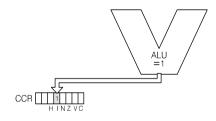
The IRQ mask bit will be cleared (contents set to logic 0). This instruction clears the SEI instruction. Interrupts which are already in process are not hindered.



Motorola	SEI	Function		CPU type	A / 6303
B&R	SEI	1 ⊅ I			
Short					
Address	ing mode /	Oncode	Address preselecti	on 0	PG1000
Audiess	ing mode /	Opcode	Address preselecti	o"  o	PG-PC

Mot	orola	SEI		Fun	ction												¢	P	U	ty	/p	e I	3 /	6	В0	9
B&F	t	SEI		1 ⇨	I																					
Sho	rt																									
۸,	ddress	ina m	odo /	Oncor	40	Γ		۸,	14	ra					٠		tio	'n			C	7	С	P	30	
^(	uui ess	iliy ii	ioue /	Орсос	ue			^'	Ju	16	3	۱ د	,, e	.5	516						O	<b>)</b>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	ļ	В	G	Г		C	CR		
3/2						Ε	Α	М	F	Z	#	Р	С	T	Υ	D	U	!	В	G	Н	ı	N	Z	٧	С
1A 10						Г	Г	Г	П		Г	Г	Г	П		Г			П		0	•	0	0	0	0

The IRQ mask bit will be set (contents set to logic 1). If an interrupt is started it is blocked and the interrupt program is not entered.



#### Note:

DIR

1/1 0F EXT. IMMED. IND.

If the IRQ mask bit I (bit 4 of the condition code register) is set to logic 1 by the user all interrupts are blocked. This technique is used if parts of a program must not be interrupted in any case.

Before the END instruction is processed bit I must again be set to 0 using the CLI instruction!

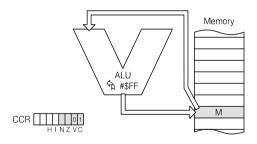
WIOL	oroia	001	"	i uii	CHOIL													<u>''</u>	_	٠,	Р		~ /	, 0.	<del></del>	ച
B&F	₹	к		(M)	⊕ #\$FI	F	⇨	(1	A)	(	=>	۱ -	۱e	ga	ıti	or	1)									7
Sho	rt																									
Δ,	ddroes	ina m	ode /	Oncor	40	Γ		Ad	44	rΔ		: r	ırc		al e	201	Hio	'n			$\circ$	ī	P	G10	000	ʲ
_^\	uui 638	ing ii	ioue /	Орсос	46			^	<i>a</i> u		30	, ,	,, ,	.30	210			···			U	,	Р	G-F	C	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Ρ	R	х	Υ	D	U	!	В	G			C	CR		
		6/3	6/2			Ε	Α	М	F	Z	#	Р	С	_	Υ	D	U	!	В	G	Н	1	Ν	Z	٧	С

Function

Motorola COM

T	Mot	orola	CON	И	Fun	ction												þ	P	U	ty	/p	e E	3 /	6	80	9
	B&F	ł	к		(M)	⊕ #\$F	F	₽	(1	A)	(	=>	۰ ۱	۱e	ga	ıti	or	1)									
	Sho	rt																									
I	Δι	dress	ina m	ode /	Oncor	łe	Γ		Δι	44	re		s p	re	20	عاد	20	tio	'n			O	1		P		
ı			g		орос.		L		•			-	- 1		-	•	-		•••			C	)	Ρ	C	30	
I	IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G	Г		CC	R	Т	_
ı		6/2	7/3	6+/2+			Е	Α	М	F	Z	#	Ρ	С	T	Υ	D	U	!	В	G	Н	I	N	z	٧	С
Ī		03	73	63			Γ	0	0	0			0	0	0	•	•	•	•		•	0	0	•	•	•	•

The contents of memory location M will be inverted (= Exclusive Or combination with #\$FF).



CPLI type A / 6303

Motorola	COMA	Function		CPU type	A / 6303
B&R	COA	A ⊕ #\$FF			
Short	KA				
Addross	ing mode /	Oncodo	Address preselecti	<u></u> O	PG1000
Address	ing mode /	Opcode	Address preselecti	°'' [Э	PG-PC

REL.

EXT. IMMED. IND.

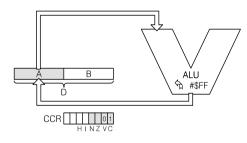
IMPL DIR

1/1 43

Mot	orola	CON	MΑ	Fun	ction												þ	CP	U	ty	γp	e E	3 /	6	В0	9
B&F	ł	COA	4	ΑФ	#\$FF	₽	Α		(=	>	Ne	eg	at	io	n)											
Sho	rt	KA																								
A	dress	ina m	ode /	Opcod	de	Γ		A	bb	re	SS	s n	re	s	ele	c	tic	n			С	-		P		
				-р		L															U	<u> </u>	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	A ⊕ #\$FF   A (=> Negation  Opcode  Address prese  IND. REL.														!	В	G			CC	R		
2/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	1	N	Z	٧	С
43						Г	Г	Г	П	Г	Г	Г	Г	П	Г	П		Г			$\circ$	0	•	•	v	•

The contents of accumulator A will be inverted (= Exclusive Or combination with #\$FF).

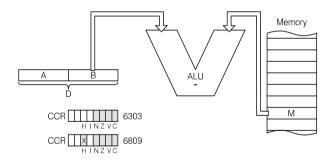
EAMFZ#PCIYDU!BGHINZVC



Mot	orola	COI	ИB	Fun	ction												C	CP	U	ty	/p	e.	A /	6	30	3
B&F	₹	COI	В	В⊕	#\$FF	$\Rightarrow$	В	•	(=	>	Ne	eg	at	io	n)											
Sho	rt	КВ																								
Λ.	ddroes	ina n	odo /	Oncor	40	Γ		۸,	44	rn					٦l/	~	410	'n			С	7	P	<b>G1</b> (	000	П
Α.	uuies	sing ii	loue /	Орсос	ode Address preselection O PC														G-F	c						
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ī	0	F	s	Т	#	Р	R	Х	Υ	D	U	!	В	G	Г		CC	R		П
1/1						Ε	Α	М	F	Z	#	Р	С	I	Υ	D	U	!	В	G	Н	ī	N	z	٧	С
53																					0	0	•	•	•	٨

																	_									
Mot	orola	COI	ИB	Fun	ction												¢	P	U	ty	p	e I	3 /	6	В0	9
B&F	₹	CO	В	В⊕	#\$FF		В		(=	>	Ne	eg	at	io	n)											
Sho	rt	КВ																								
Δ,	ddraee	ina m	node /	Onco	40	Γ		Δ,	44	ro		s p	rc		عاد		Hio	'n			O	)	С	Pδ	30	
ť	uui 633	ing ii	ioue /	Орсо	46				uu		30	, 1	,, ,	.31	510			···			U	)	Р	C	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
2/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Ι	I	N	Z	٧	С
53						Γ															0	0	•	•	•	•

The contents of accumulator B will be inverted (= Exclusive Or combination with #\$FF).

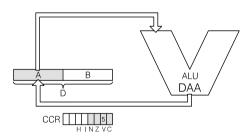


Mot	Addressing mode / Opcode Address preselection O PGPC											303	3																	
addition of BCD numbers														7																
Sho	ort																													
Addressing made / Opende Address pressleation O PG1000														٦																
Α.	uui es	sing ii	loue /	Орсос	16	L		^'	u		33	, 1	,, e	.5	316	-		•••		I	$\overline{\mathbf{o}}$	Г	PC	-P	C	7				
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	т	#	Ρ	R	Х	Υ	D	U	!	В	G		-	CCR							
2/1						Е	Α	М	F	z	#	Р	С	Т	Υ	D	υ	!	в	G	4	П	N	zΙ	V	╗				

Mot	orola	ction												¢	P	U	ty	р	e E	3 /	68	В0	9			
B&F	ł	DK			imal a										ım	ıu	lat	0	r <i>A</i>	\ a	ft	er	ar	1		
Sho	rt		ition c	f	ВС	C	n	uı	ml	be	rs															
	ddress	10	Γ		۸,	44	ra					٠		tio	'n			C	7	С	Pε	30				
_^`	Jui 638	oning in	loue /	орсос	16			^'	uu	16	33	۰,	,, e	.51	-16	-		""			C	)	Р	CE	30	
IMPL.	DIR.	EXT.	IMMED.	IND.	REL.	ı	0	F	s	Т	#	Р	R	х	Υ	D	U	!	В	G			CC	R		
2/1						Ε	Α	М	F	Z	#	Р	С	1	Υ	D	U	!	В	G	Н	I	Ν	Z	٧	С
19						Γ															0	0	•	•	×	•

The binary result (in accumulator A) produced by adding BCD numbers is converted back to a BCD number and is again stored in accumulator A. This instruction only works correctly if it is executed immediately after an addition (ADD, + or A+B).

The carry flag is then set to logic 1 if the BCD result is greater than 99.



**Example:** Addition of BCD numbers \$27 and \$96: A = \$BD

Decimal adjustment: A = \$23

C = 1

(=> carry hundreds position)

# 4. MATHEMATIC ROUTINES 4.1. GENERAL

All central and peripheral processors are equipped with mathematic routines as a standard. These routines are components of the operating system and are called from the assignment list by *instruction mnemonics*. Besides basic addition, subtractions, multiplication, division and square root there are conversion and utilities to be utilized (e.g. for comparisons and copying). Standard function blocks can use these routine. For floating point-numbers the 4 byte *IEEE format* is used.

MATHEMATIC ROUTINES MAY NOT BE OPERATED WITHIN INTERRUPT ROUTINES.

### **Operands and Memory**

Mathematic routines occupy the following locations in the user data range:

R1024 R1025 R1026 to R1029 R1030 to R1033 R1034 to R1037 R1038 to R1041 R1042 to R1045 R1046 to R1047 R1048 to R1049 R1050 to R1051 R1052 to R1053	Error number Reserved Operand 1 (OP1) Operand 2 (OP2) Temporary memory 1 (MEM1) Temporary memory 2 (MEM2) Temporary memory 3 (MEM3) Source address Destination address Length Data	IEEE format
--	--	-------------

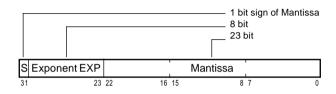
### **ERROR MESSAGES**

If an error occurs during a mathematic routine the carry flag is set and an error number is placed in R1024:

Error number	Short form	Description
1	MATH_OVERFLOW	Data format overflow during calculation
2	MATH_UNDERFLOW	Data format underflow during calculation
3	DIV_BY_ZERO	Division by 0
4	CONV_OVERFLOW	Data format overflow during number conversion
5	CUT_LSB	Low order byte (LOB) last when loading 4 byte mantissa
6	LOAD_OVERFLOW	Data format overflow during loading operands
7	LOAD_UNDERFLOW	Data format underflow during loading operands
8	NEG_SQRT	Negative operand in square root calculation
9	INVAL_CHAR	Invalid character in string conversion routine
10	NO_FPC	No floating point coprocessor installed (causes TRAP error)
11	INVAL_COMMAND	Invalid command (causes TRAP error)
12	NOT_A_NUMBER	Not a valid IEEE format number
13	INCH_EXP	Exponent error in inch-metric or metric-inch conversion
14	INCH_OVERFLOW	Data format overflow in inch-metric or metric-inch conversion

### **DATA FORMAT**

### **Single Precision Floating Point Format**



Conversion:

 $(-1)^S \cdot 2^{(EXP-127)} \cdot 1.mantissa$ 

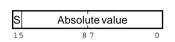
#### Number range display:



All numbers from  $-9.22*10^{-18}$  to  $+9.22*10^{-18}$  with the exception of 0 can be displayed and are used the same as 0.

### **AbsoluteWith Sign**





S ... Sign







± 2147483647 ±(2<sup>31</sup> - 1)

### Integer

### Integer word



-32768 (-2<sup>15</sup>) to 32767 (2<sup>15</sup>-1)

### Integer long



-2147483648 (-2<sup>31</sup>) to 2147483647 (2<sup>31</sup>-1) (±2,15 \* 10<sup>9</sup>)

### MADD Addition in Floating Point Format

Execution time	6303	209 - 690	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	207 - 503	error message	0	0				0	0					0		

Function: Operands OP1 and OP2 are added. The result is stored in OP1. OP2 remains unchanged.

Parameters: None

Result: Changed D

N. Z Corresponds with routine result

### MSUB

#### **Subtraction in Floating Point Format**

Execution time	6303	219 - 700	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	213 - 509	error message	0	0				0	0					0		

Function: Operand OP2 is subtracted from OP1. The result is stored in OP1. OP2 remains unchanged.

Parameters: None

Result: Changed D

N. Z Corresponds with routine result

### MMUL Multiplication in Floating Point Format

Execution time	6303	209 - 803	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	209 - 563	error message	0	0				0	0					0		

Function: Operands OP1 and OP2 are multiplied. The result is stored in OP1. OP2 remains unchanged.

Parameters: None

Result: D Changed

N, Z Corresponds with routine result

### MDIV Division in Floating Point Format

<b>Execution time</b>	6303	190 - 1980	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	227 - 1390	error message	0	0	0			0	0					0		

Function: Operand OP1 is divided by OP2. The result is stored in OP1. OP2 remains unchanged.

Parameters: None

Result: D Changed

N, Z Corresponds with routine result

#### MSQR

#### **Square Root in Floating Point Format**

Execution time	6303	71 - 8065	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	123 - 5100	error message	0					0	0	0				О		

Function: The square root of operand OP1 is calculated and the result is stored in OP1. OP2 remains unchanged. Calculation

accuracy of this function is limited to four decimal places.

Parameters: None

Result: D Changed

N, Z Corresponds with routine result

#### **MSGN**

#### Change Sign of Operand 1

Execution time	6303	85	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	126	error message														

Function: The sign of OP1 is inverted. This operation corresponds to a multiplication with -1. OP2 remains unchanged.

Parameters: None

Result: D Changed

C N. Z

Corresponds with routine result

# MCOP Copy Operand OP1 to Operand OP2

Execution time	6303	46	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	115	error message														

Function: Operand OP1 is copied to OP2.

Parameters: None

Result: D Changed

# MEXG Exchange Operands OP1 and OP2 with each other

Execution time	6303	76	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	125	error message														

Function: Operands OP1 and OP2 are exchanged with each other.

Parameters: None

Result: D Changed

#### LAL1

#### Load Operand OP1 with Number (Absolute long)

Execution time	6303	190 - 339	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	193 - 293	error message					0									

Function:

The binary number (format; absolute long), to which index register X points, is converted to IEEE format and stored in OP1. If more than 24 bits are used for a binary number only the most significant bits are converted and stored in OP1: error 5 (CUT\_LSB) occurs.

Source address of binary number Parameters:

Example:

Load OP1 with binary number from registers R 320 through R 323:

Result:

Changed D N 7

Correspond with routine result

LAL1

LDX# R 0320 Binary # source address

LAL2

#### Load Operand OP2 with Number (Absolute long)

Execution time	6303	190 - 339	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	193 - 293	error message					0									

Function:

Parameters:

The binary number (format: absolute long), to which index register X points, is converted to IEEE format and stored in OP2. If more than 24 bits are used for a binary number only the most significant bits are converted and stored in OP2; error 5

(CUT\_LSB) occurs.

Source address of binary number Result:

Changed D

N 7 Correspond with routine result

# LAW1 Load Operand OP1 with Number (Absolute word)

Execution time	6303	83 - 250	Possible	1	2	3	4	5	6	7	8	9	10	11	12	14
in µs	6809	125 - 241	error message													

Function: The binary number (format: absolute word) in accumulator D is converted to IEEE format and stored in operand OP1.

Parameters: D Binary number (format: absolute word)

Result: D Changed

N, Z Correspond with routine result

## LAW2 Load Operand OP2 with Number (Absolute word)

Execution time	6303	83 - 247	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	
in µs	6809	125 - 240	error message														

Function: The binary number (format: absolute word) in accumulator D is converted to IEEE format and stored in operand OP2.

Parameters: D Binary number (format: absolute word)

Result: D Changed

N, Z Correspond with routine result

#### LIL1 Load Operand OP1 with Number (Integer long)

Execution time	6303	197 - 381	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	201 - 315	error message					0									

Function: The binary number (format; integer long) to which index register X points is converted to IEEE format and stored in operand

OP1. If the binary number uses more than 24 bits only the most significant bits are converted and stored and error 5 (CUT\_LSB) occurs.

Result: Changed D

Parameters:

Parameters:

N 7 Correspond with routine result

Source address of binary number

#### LIL<sub>2</sub> Load Operand OP2 with Number (Integer long)

<b>Execution time</b>	6303	194 - 378	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	
in µs	6809	198 - 315	error message					О									

Function: The binary number (format: integer long) to which index register X points is converted to IEEE format and stored in operand

OP2. If the binary number uses more than 24 bits only the most significant bits are converted and stored and error 5 (CUT\_LSB) occurs.

Source address of binary number

Result: D Changed

> N 7 Correspond with routine result

### LIW1 Load Operand OP1 with Number (Integer word)

Execution time	6303	87 - 260	Possible	1	2	3	4	5	6	7	8	9	10	11	12	14
in µs	6809	128 - 249	error message													

Function: The binary number (format: integer word) in accumulator D is converted to IEEE format and stored in operand OP1.

Parameters: D Binary number

Result: D Changed

N, Z Correspond with routine result

# LIW2 | Load Operand OP2 with Number (Integer word)

Execution time	6303	84 - 257	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	126 - 247	error message														

Function: The binary number (format: integer word) in accumulator D is converted to IEEE format and stored in operand OP2.

Parameters: D Binary number

Result: D Changed

N, Z Correspond with routine result

# LF1 Load Operand OP1 with Number (IEEE format)

Execution time	6303	88 - 125	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	129 - 146	error message	0					0	0					0		

Function: The floating point (IEEE format) number to which index register X points is tested (for whether it is in the allowed range or

not) and stored in operand OP1.

Parameters: X Source address of floating point number

Result: D Changed

N, Z Correspond with routine result

#### **LF2** Load Operand OP2 with Number (IEEE format)

Execution time	6303	88 - 125	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	127 - 144	error message	O					0	0					0		

Function: The floating point (IEEE format) number to which index register X points is tested (for whether it is in the allowed range or

not) and stored in operand OP2.

Parameters: X Source address of floating point number

Result: D Changed N. Z Correspo

Correspond with routine result

# SAL Store Operand OP1 in Absolute Long Format Execution time in µs 6303 169 - 408 Possible error message 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Function: Operand OP1 is converted to a binary number (format: absolute long) and saved. Index register X holds the destination

address to which the binary number is stored. Operands OP1 and OP2 remain unchanged. If the value is too large for a 32 bit absolute number, error 4 (CONV\_OVERFLOW) occurs and the maximum negative or positive amount (+2147483647 or

-2147483647) is stored.

Parameters: X Destination address of binary number

Result: D Changed

N, Z Correspond with routine result

# **SAW** Store Operand OP1 in Absolute Word Format

Execution time	6303	158 - 373	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
in µs	6809	183 - 316	error messages				0								0			

Function: Ope

Operand OP1 is converted to a binary number (format: absolute word) and saved. Index register X holds the destination address to which the binary number is stored. Operands OP1 and OP2 remain unchanged. If the value is too large for a 16 bit absolute number, error 4 (CONV\_OVERFLOW) occurs and the maximum negative or positive amount (+32767 or - 32767) is stored.

Parameters: None

**Result:** D Binary number

N, Z Correspond with changed contents of D

# SIL Store Operand OP1 in Integer Long Format

Execution time	6303	172 - 424	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	190 - 347	error message				0								О		

Function: Operand OP1 is converted to a binary number (format: Integer long) and saved. Index register X holds the destination

address to which the binary number is stored. Operands OP1 and OP2 remain unchanged. If the value is loo large for a 32 bit integer number, error 4 (CONV\_OVERFLOW) occurs and the maximum negative or positive amount (-2147483647 or

+2147483647) is stored.

Parameters: X Destination address of binary number

Result: D Changed

N, Z Correspond with routine result

### SIW Store Operand OP1 in Integer Word Format

Execution time	6303	158 - 380	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	183 - 321	error messages				0								0		

Function: Operand OP1 is converted to a binary number (format: Integer word) and saved. Operands OP1 and OP2 remain

unchanged. If the value is too large for a 16 bit integer number, error 4 (CONV\_OVERFLOW) occurs and the maximum

negative

or positive amount (-32767 or +32767) is stored.

Parameters: None

Result: D Binary number

N. Z Correspond with changed contents of D

SFX	Store (	Operand O	P1 in IEEE Format															
Executio	n time	6303	43	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	112	error message														

Function: Operand OP1 is stored in floating point format (IEEE). Index register X holds the destination address to which the number in

floating point format is stored. OP1 and OP2 remain unchanged.

Parameters: X Destination address of number in floating point format

Result: D Changed

C, N, Z 0

#### SFM1 Store Operand OP1 in Memory 1

Execution time	6303	60	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	116	error message														

Function: Operand OP1 is stored in floating point format (IEEE) in temporary memory 1 (R1034 to R1037).

Parameters: None

Result: D Changed

#### **SFM2** Store Operand OP1 in Memory 2

Execution time	6303	60	Possible	1	2	3	4	5	6	7	8	9	10	11	12	14
in µs	6809	116	error message													

Function: Operand OP1 is stored in floating point format (IEEE) in temporary memory 2 (R1038 to R1041).

Parameters: None

Result: D Changed

## **SFM3** Store Operand OP1 in Memory 3

Execution time	6303	60	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	116	error message														
			•								_						_

Function: Operand OP1 is stored in floating point format (IEEE) in temporary memory temporary 3 (R1042 to R1045).

Parameters: None

Result: D Changed

#### RFM1 Load Operand OP2 from Memory 1 2 3 5 8 9 | 10 | 11 | 12 | 13 | 14 **Execution time** 6303 56 Possible 6 4 6809 116 in µs error message

Function: Operand OP2 with number in floating point format (IEEE) is loaded from temporary memory 1 (R1034 to R1037).

Parameters: None

Result: D Changed

# RFM2 Load Operand OP2 from Memory 2

Execution time	6303	56	Possible	1	2	3	4	5	6	7	8	9	10	12	13	14
in µs	6809	116	error message													

Function: Operand OP2 with number in floating point format (IEEE) is loaded from temporary memory 2 (R1038 to R1041).

Parameters: None

Result: D Changed

### RFM3 Load Operand OP2 from Memory 3

<b>Execution time</b>	6303	56	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	
in µs	6809	116	error message														

Function: Operand OP2 with number in floating point format (IEEE) is loaded from temporary memory 3 (R1042 to R1045).

Parameters: None

Result: D Changed

# FM2B 2 Byte X 2 Byte Multiplication Execution time 6303 115 - 191

Execution time in μs 6303 115 - 191 Possible 1 2 3 4 5 6 7 8 9 10 11 12 13 14 error message

Function: Two binary numbers (format: Integer word) are multiplied. The result is a number in integer long format.

Parameters: X Source address for multiplication

D Multiplier

R1048& Destination address for result

Result: D Changed

C, N, Z Invalid X Unchanged

#### FM3B

#### 3 Byte X 2 Byte Multiplication

Execution time	6303	156 - 270	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	150 - 225	error message														

Function: A 3 byte integer number is multiplied with a number in integer word format. The result is a 5 byte integer number.

Parameters: X Source address for multiplication (3 byte Integer number)

D Multiplier (Integer word)

R1048& Destination address for result (5 byte Integer number)

Result: D Changed

C, N, Z Invalid

X Unchanged

FM4B	4 Byte	X 2 Byte	Multiplication															
Executio	n time	6303	192 - 344	Possible	1	2	3	4	5	6	7	8	9	10	11	12 1	3 1	4
in μ	s	6809	172 - 272	error message														1
Function:	A numbe		g format (4 byte) is mu	ultiplied with a number in integ	ger w	orc	l for	mat	(2 b	oyte	e). T	he i	resi	ult is	s a 6	byte	)	

Parameters: X Source address for multiplication (Integer long)

D Multiplier (Integer word)

R1048& Destination address for result (6 byte Integer number)

Result: D Changed

C, N, Z Invalid Unchanged

#### CAF Convert ASCII String to Floating Point (IEEE) Format 9 | 10 | 11 | 12 | 13 | 14 **Execution time** 6303 280 - 2140 Possible 2 3 5 6 8 6809 251 - 1460 in µs error message

Function: The ASCII string, to which index register X points is converted to internal IEEE format and stored in OP1. OP2 remains

unchanged.

Parameters: X Source address of ASCII string

Result: D Changed

N, Z Invalid

#### Syntax rules for ASCII strings:5

Signs permitted are <+>, <-> and <blank spaces>. The sign is optional.

- The mantissa can begin with leading zeros or blank spaces and can include a decimal point plus up to seven significant digits.
   Separating the mantissa and exponent is done with an <E> or a blank space.
- 3. The *exponent* has two digits and a sign. It begins with an <E> after the last mantissa or after the separation character (blank space).
- 4. **Valid characters** are <0> to <9>, <space>, <->, <+>, <decimal point> and <E>. Invalid characters in an ASCII string Ilead to a break in the routine and error 9 (INVAL\_CHAR) occurs. The value held in operand OP1 after the break is invalid.
- 5. After an <E> maximum 3 characters are read automatically, otherwise the string must be terminated with <CR> (\$0D) or <0> (\$00).

#### Examples of valid ASCII strings:

CFA	Conve	rt OP1 to A	ASCII (without leadir	ng zeros)														
Executio	n time	6303	352 - 7310	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	352 - 4440	error message	0			0		О	0					0		

**Function:** The contents of OP1 are converted to an ASCII string and stored at the address indicated in index register X. Operands OP1 and OP2 remain unchanged.

The amount of significant digits is limited to a maximum of seven. An ASCII string can therefore be maximum nine characters long (including sign and decimal point). Leading zeros are replaced by blank spaces. If OP1 can not be converted to the desired ASCII format the string is filled with ">" characters.

Parameters: X Destination of ASCII string

Length of entire ASCII string
Amount of decimal digits

Result: A Changed

B Length of ASCII string

Format of ASCII string:

N. Z 0

Example:

OP1 should be converted to an ASCII string with a maximum of five significant positions (including two decimal digits). The ASCII string should be stored from the 9 bit memory location R0250.

```
:
                                    Load destination address for ASCII string in index register X
                R 0250
        LDX#
                                    Length of ASCII string (5 significant positions + sign + decimal point = 7)
                                    Amount of decimal digits (2)
                                    Format of ASCII string transferred to A
        LDAA
                 # $72
        CFA
         :
OP1 = 32.123
                                " 32.12"
OP1 = -0.1
                                "- 0.10"
OP1 = 4.87
                                     4.87"
OP1 = 2300.25
                                " >>>.>>"
                        =>
OP1 = -1000.25
                                "-<<<.<
                        =>
```

CFA0	Conve	rt OP1 to A	ASCII (with leading a	zeros)														
Executio	n time	6303	310 - 7190	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	310 - 4320	error message	О			0	(	0	0					0		
Function:	OP1 and The amo characte	OP2 remain u unt of significa rs long (includi	e converted to an ASCII str nchanged. nt digits is limited to a maxi ng sign and decimal point). e string is filled with ">" cha	mum of seven. The ASC Leading zeros are not s	II str	ing	can	the	refo	re b	oe n	nax	imu	m n	ine			



Amount of decimal digits

Length of entire ASCII string

A Format of ASCII string:

A Changed

B Length of ASCII string

N. Z 0

Result:

#### Example:

OP1 should be converted to an ASCII string with a maximum of four significant positions (including one decimal digit): The ASCII string should be stored from the 8 bit memory location R0100.

```
:
                                     Load destination address for the ASCII string in index register X
                 R 0100
        LDX#
                                     Length of ASCII string (4 significant positions + sign + decimal point = 6)
                                     Amount of decimal digits (1)
                                     Format of ASCII string transferred to A
        LDAA
                 # $61
        CFA
          :
          :
OP1 = 32.123
                                 " 032.1"
OP1 = -0.1
                                 "-000.1"
OP1 = 4.87
                        =>
                                 " 004.8"
OP1 = 2300.25
                                 " >>>.>"
                        =>
OP1 = -1000.25
                                 "-<<<.<"
                        =>
```

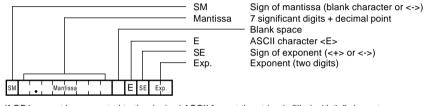
# CFEA Convert OP1 to ASCII String with Exponent Format

Execution time	6303	570 - 7140	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	267 - 4140	error message	0					О	0					О		

Function:

The contents of OP1 are converted to an ASCII string with floating point exponent format and stored to the memory location which is addressed by index register X. Operands OP1 and OP2 remain unchanged.

The string always has the same format (length: 14 characters).



If OP1 cannot be converted to the desired ASCII format the string is filled with ">" characters.

Parameters: X Destination address of ASCII string 3 2 1 0

Result:

A Changed

B Length of ASCII string

Z Corresponds with routine result

N Invalid

**Example:** OP1 should be converted to ASCII string with exponent format and stored to R0500.

```
:
:LDX# R 0500 Destination address for ASCII string
CFEA
:
```

CIA	Conver	t Integer t	o ASCII (with	out lea	ding zeros)											
Executio	n time	6303	380 - 2020		Possible		1	2 3	4 5	6	7	8	9 10	11 1	2 13	3 14
in µs	s	6809	357 - 1370		error messag	ge			0							
Function:	Operands The numb (including	OP1 and OP2 per of significations sign and deci	verted to an ASCII 2 remain unchange nt digits is limited t mal point). Leadin mat the string is fil	ed. to a maxin g zeros ar	num of nine. The A	ASCII nk spa	string	, can	therefo	re b	e up	to e	leven	chara	cter	s
Parameters:	R1046& X	Destination	dress of binary nu n address of ASCI ASCII string:		7 6 5 4 8		mber		nificar cimal c							
	В	Format of	binary number:	7 6 5	5, 4, 3 2 1 0				ary nur ary nur		1 : 0	= Int	eger byte	)		
Result:	A B N, Z	Changed Length of 6 0	entire ASCII string	l												

**Example:** The number in registers R0100 to R0103 (format: integer long) should be converted to an ASCII string with a maximum of six significant digits of which two are decimal digits. The string should be stored to R0250.

```
Source address of binary number
        L'DX#
                 R 0100
        STX
                 R 1046
                                     Store source address in register R1046&
        LDX#
                 R 3000
                                     Destination address for ASCII string
                                     Number of significant digits
                                     Number of decimal digits
                                     Transfer format of ASCII string to accumulator A
        LDAA
                 # $62
                                     Format of binary number: 1 => Integer
                                     Length of binary number: 1 => 4 Byte
                                     Transfer format of binary number to accumulator B
                 # %00000011
        LDAB
                                     Routine call
        CIA
Integer-Long = 56499
                                            564.99""
Integer-Long = -23
                                             0.23"
Integer-Long = 1000000
                                         " >>>>.
```

Execution	time	6303	310 - 1960	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13
in µs	,	6809	312 - 1320	error message				0									
Function:	OP2 rema The numb characters	in unchanged er of significa s long (includ	d. ant digits is limited t ing sign and decima	string and stored to the address of the address of a maximum of nine. The ASCII all point). Leading zeros are not so string is filled with ">" or "<" chara	strir uppı	ig ca	an tl	here	efor	e be	ma	axim	num	ele	even		d
Parameters:	R1046& X	Destination	ddress of binary nu on address ASCII st ASCII string:	ring	nbe					digit: gits	s						
	В	Format of	binary number:					•		ber: ber:	1 : 0 :	= Al = In = 2 = 4	teg byte	er e			
Result:	A B N, Z	Changed Length of 0	entire ASCII string														

#### Example:

The number in registers R0100 to R0103 (format: absolute long) should be converted to an ASCII string with a maximum of eight significant digits of which two are decimal digits. The string should be stored to R0200.

```
Source address of binary number
        L'DX#
                 R 0100
                R 1046
                                    Store source address in register R1046&
        STD
                R 0200
                                    Destination for ASCII string
        LDX#
                                    Number of significant digits
                                    Number of decimal digits
                                    Transfer format of ASCII string to accumulator A
        LDAA
                 # $83
                                     Format of binary number: 0 => absolute
                                     Length of binary number: 1 => 4 byte
                                    Transfer format of binary number to accumulator B
                 # %00000001
        LDAB
                                    Routine call
        CIA
Absolute long = 56499
                                         " 00056.499"
Absolute Iona = -23
                                         "-00000.023"
Absolute long = 1000000
                                         " 01000.000"
Absolute long = 100000000
                                                 " >>>> "
```

CBCD	Conve	rt Binary to	BCD															
Executio	n time	6303	192 - 1180	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	227 - 852	error message				0										

Function: A three byte binary number is converted to a three byte BCD number. This conversion to three byte BCD numbers is only

possible within the range of numbers from 0 to 999999. If the binary number is out of this range error 4

(CONV\_OVERFLOW) occurs and all BCD digits are set to nine (BCD number:; 999999). Operands OP1 and OP2 remain unchanged.

Parameters: D Source address of binary number

Χ Destination address BCD number

Result: D Least significant two bytes of the BCD number

Correspond to the contents of D N. Z

Example: Binary number = 450 BCD No. = \$ 0.004BCD No. = \$ 0 0 1 9

BCD No. = \$ 9 9 9 9

Binary number = 1956

Binary number = 1000000 =>

The binary number stored in registers R0100 to R0102 should be converted to a BCD number. The result should be saved to R0290.

LDX#

Source address of binary number R 0100 Load source address after D

XGDX R 0290 LDX# Destination address for BCD number

CBCD

CBIN	Conve	rt BCD to E	Binary															
Executio	n time	6303	112 - 223	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	166 258	error message														

Function: A three byte BCD number is converted to a three byte binary number. Source and destination addresses must be different.

Parameters: D Source address for BCD number

X Destination address for binary number

**Result:** D Least significant two bytes of the binary number

N, Z Correspond with the contents of D

**Example:** BCD No. =  $\begin{pmatrix} 0 & 0 & 4 & 5 \end{pmatrix}$  => binary number = 450

BCD No. = \$ 0, 0 1 9 5 6 => binary number = 1956

BCD No. = \$ 9 9 9 9 9 => binary number = 999999

The BCD number stored in registers R0200 to R0202 should be converted to a binary number. The result should be stored to R3000.

. LDX $\sharp$  R 0200 Source address of BCD number XGDX Load source address after D

LDX# R 3000 Destination address of binary number

CBIN

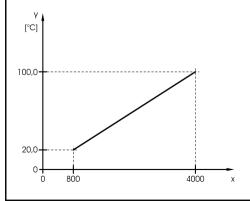
### Conversion: Binary <=> Physical (Scaling)

In PLC programs numbers are not usually stored in their physical units or with their physical values. Normally a binary value is used which corresponds to a certain physical size. To display a physical value (e.g. on operator terminals) the binary value must be converted to the physical unit.

Example:

A temperature in the range from 20.0 °C to 100.0 °C will be measured. The temperature sensor sends an analog signal between 4mA and 20mA which is converted to a PLC internal number value between 800 and 4000 by an AD converter. The internal representation is:

The conversion is done according to the straight line formula: v = kx + d



The straight line is defined by factors **k** and **d** which are calculated from the two points (800/200) and (4000/1000):

II. 
$$1000 = 4000k + d$$

$$=>$$
  $a = 200 - 800k$ 

$$k = \frac{800}{3200} = 0.25$$

$$> d = 200 - 800k = 0$$

The following routines are available for converting binary values to physical sizes:

**CBPP** Calculates k and d from two straight line points or from integer word format to IEEE format.

CBPQ Calculation according to formula v = kx + d. Factors **k** and **d** must first be in IEEE format (e.g. calculated with CBPP routine).

CRP Calculation according to formula y = kx + d. The straight line can be defined either from factors **k** and **d** or by two straight line points  $(x_n/v_n)$  and  $(x_n/v_n)$ . Number format (floating point IEEE format or integer word) for the factors can be selected by the user.

In many applications numbers are entered via operator interface terminals. These numbers are entered in their physical value and must be converted to the PLC internal value. This conversion is done by reversing the straight line equation:

$$y = kx + d$$
 =>  $x = \frac{k}{k}$ 

The following routines are available for converting physical sizes to binary values:

**CBPP** Calculates k and d from two straight line points or from integer word format to IEEE format.

**CPBQ** Calculates x according to the formula above. Factors k and d must first be in IEEE format (e.g. calculated with CBPP

routine).

**CPB** Calculates x according to the formula above. The straight line can be defined either from factors k and d or by two straight line points  $(x_x/y_x)$  and  $(x_x/y_x)$ . Number format (floating point IEEE format or integer word) for the factors can be selected by

the user

СВРР	Calcula	ating Facto	ors <i>k</i> and <i>d</i> f	rom two	Straight Line F	Point	s									
Execution	ntime	6303	2500 - 6700		Possible	1	2	3	4	5	6 7	8	9	10	11 12	2 13 14
in µs	•	6809	1200 - 4200		error messsage	0					00				С	
Function:	This routi	ine has two fur	ictions:													
			<b>k</b> and <b>d</b> (in IEEE eger word format		m two straight line po format.	ints (x	<sub>1</sub> /y <sub>1</sub> )	and	(x <sub>2</sub> /	/y <sub>2</sub> )	. Thes	se s	traig	ht lir	ne poi	nts
			e transferred to the tare converted to		n integer word format nat.	or in I	EEE	for	mat.	. Fa	ctors	whi	ch a	re se	nt in	
	Number f	ormat is option	nal (see diagram	on next pa	ge).											
Parameters:	R1048&	Destinatio	n address for fac	tors <b>k</b> and	d (IEEE Format, => 8	Byte)										
				—— Fun	ction		-	= 1.								
				—— Data	a format of x or k:		0	= 2. = In = IE	tege	er w	ord					
				—— Data	a format of y or d:			= In								
	В	7, 6, 5, 4	3 2 1 0		Number format and	d funct	ion	sele	ctior	n						
	X	Source ad	dress of parame	ter (x/y or k	z/d)											
Result:	X, D N, Z	Changed Invalid														

Depending on the function selected and the number format of the parameter the amount of memory used varies.

			Cor	ntents of	Accumulato	or B		
Memory	0xxxxx00	0xxxxx01	0xxxxx10	0xxxxx11	1xxxxx00	1xxxxx01	1xxxx10	1xxxx11
Rxxxx	Integer word  Yinteger word  Xinteger word  Integer word  Integer word	x, Integer word y, IEEE  X, Integer word  Y, IEEE	X, IEEE  Integer word  X, IEEE  Integer word	X, IEEE  JEEE  JEEE  JEEE  JEEE	k Integer word d Integer word	k Integer word d IEEE	k IEEE d nteger word	

CBPQ	Conve	rt Binary =	> Physical, Quick															
Execution	n time	6303	780 - 1700	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	;	6809	580 - 1100	error messsage	0					0	О					0		

Function: A binary number (integer word) is converted to a physical value according to the formula y = kx + d. Factors **k** and **d** are

calculated separately with CBPP and are stored in floating point format.

**Parameters:** D Binary number *x* 

X Source address for straight line factors **k** and **d** (IEEE format)

**Result:** D Physical value y (Integer word)

OP1 Physical value y (IEEE format)

X Changed

N, Z Correspond with the contents of D

Execution time 6303 780 - 1500 Possible 1 2 3 4 5 6 7 8 9 10 11 12 13	CPBQ	Conve	rt Physical	=> Binary, Quick															
	Executi	on time	6303	780 - 1500	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13 1	4
in µs 6809 580 - 950 error message O O O O O	in	μs	6809	580 - 950	error message	0					0	0					0		]

**Function:** A physical value is converted to a binary value according to the formula  $x = \frac{y-d}{k}$ .

Factors **k** and **d** are calculated separately with the **CBPP** routine and stored in floating point format.

**Parameters:** D Physical value *y* (Integer word)

X Source address for straight line factors **k** and **d** (IEEE format)

Result: D Binary value x (Integer word)

OP1 Binary value x (IEEE format)

X Changed

N, Z Correspond with the contents of D

СВР	Convert Binary => Physical																
Execution time		6303	3400 - 8300	Possible	1	2	3	4 5	6	7	8	9	10	11	12	13	14
in µs		6809	ca. 4000	error messsage	0				0	0					0		
Function:	The straig As the ca routine in	A binary number (integer word) is converted to a physical display value according to the straight line equation $y = kx + d$ . The straight line is either defined by two points $(x_1/y_1)$ and $(x_2/y_2)$ or by factors <b>k</b> and <b>d</b> . As the calculation of <b>k</b> and <b>d</b> takes several msec. it is recommended to calculate these factors just once, using the CBP routine in the initialization program and store them temporarily in registers. By calling the <b>CBP</b> routine the calculated factors are put into use.															
Parameters:	R1046& R1048& X	Destinatio	dress of binary number <i>x</i> n address for physical valudress of straight line points		tors	<b>k</b> aı	nd <b>d</b>	I									
	Α	7, 6, 5, 4		nber format for binary va nber format for physical Number format of <i>x</i> a	valu	lue y											
			Nun	ight line definition:  hber format for x or k:  hber format for y or d:		0 = with two straight line points 1 = with factors <b>k</b> and <b>d</b> 0 = Integer word 1 = IEEE format 0 = Integer word 1 = IEEE format											
	В	7, 6, 5, 4	3 2 1 0	Selection for straight	line	-					r for	mat					

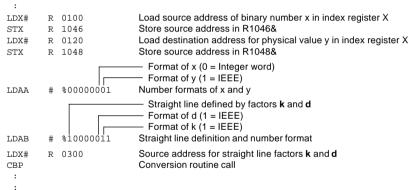
Result:

D Physical value y (Integer word), if for y integer word is selected, otherwise D is undefined.

OP1 Physical value y (IEEE format)
X Changed
N, Z Correspond with the contents of D

Example:

Floating point factors k and d have been calculated with the CBPP routine and are stored in registers R0300 through R0307. The binary number x (format: integer word) from R0100& will be converted and the result y should be stored from R0120 in IEEE format.

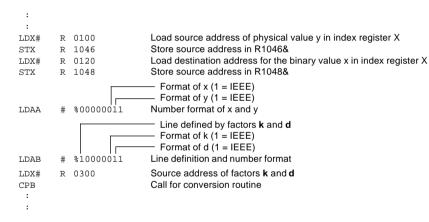


СРВ	Conve	Convert Physical => Binary															
Execution time		6303	3400 - 8300	Possible	1	2	3	4 5	6	7	8	9	10	11	12	13	14
in µs		6809	ca. 4000	error messsage	0				0	0					0		
Function:	A physica	al value is conv	according to the reverse stra	aigh	t line	e ed	quatio	n <i>x</i>	- d	—.							
	Since the CBPP rou	The straight line is defined either by two straight line points $(x_1/y_1)$ and $(x_2/y_2)$ or by factors $k$ and $d$ . Since the calculation of factors $k$ and $d$ takes several msec. it is recommended to make the calculation only once using the CBPP routine in an INIT program and store it in temporary memory (registers). By calling the <b>CPB</b> routine the factors previously calculated are set for use.															Э
Parameters:	R1046& R1048& X	Destinatio	ldress of physical value n address for binary nu dress of straight line po		tors	<b>k</b> ar	nd <b>c</b>	d .									
	Α	7 6 5 4		Number format of binary value <i>x</i> Number format of physical value <i>y</i> Number format of <i>x</i> and <i>y</i> :  1 = IEEE format													
				Straight line definition: Number format of x or k:		0 = With two straight line points 1 = With factors <b>k</b> and <b>d</b> 0 = Integer word 1 = IEEE format											
	В	7'6'5'4		Number format of y or d:  Selection for straight	lina	1 :	= IE	teger EEE fo	rmat	t	nun	nha	r for	mat			
	D	<u>[', 0, 5, 5</u>	<u>'                                    </u>	Colocilor for straight		uen		on and	101		iiuii	ibe	1 101	mat			

Result:

D
Binary value x (integer word), if for x integer word is selected, otherwise D is undefined
OP1
Binary value x (IEEE format)
X
Changed
N, Z
Corresponds to the contents of D

**Example:** Factors **k** and **d** have been converted in the **CBPP** routine and are now stored in registers R0300 through R0307. Physical value *y* (IEEE format) from R0100 through R0103 is converted to binary value *x* and stored to R0120.



CIM	Convert Inch => Millimeter												
Executio	n time	6303	307 - 472	Possible	1 2	3 4	5	6 7	8	9	10	11 1:	2 13 14
in µ:	s	6809	269 - 368	error messsage									00
Function:	An inch number in integer long format will be converted to millimeters (1 inch = 25.4mm). Accuracy can be defined by using powers of 10. To ensure measuring accuracy within a program the same power of 10 must be used throughout. Operands OP1 and OP2 remain unchanged.												
Parameters:	R1046& X A B	X Destination address for mm number (Integer long) A Exponent of inch number (0 through 5)											
Result:	D N, Z		wer bytes of the mm r nd to the contents of D										

#### Example:

Set point values are entered in inches via a keyboard. The internal representation of all live and set point values is stored in mm with a resolution of 0.01. For the inch to metric conversion with the least inaccuracy inch numbers must be entered with four decimal digits.

:			
:			
LDX#	R	0100	Source address of inch number
STX	R	1046	
LDAA	#	004	Exponent of inch number (accuracy = 0.0001; => 4 decimal digits)
LDAB	#	002	Exponent of mm number (accuracy = 0.01; => 2 decimal digits)
LDX#	R	0200	Destination address of mm number
CIM			Routine call
:			

Binary number in R0100 through R0103	Inch number	Binary number in R0200 through R0203	
3460	0.3460"	8.79mm	879
234500	23.4500"	595.62mm	59562
3937	0.3937"	10.00mm	1000
10000	1.0000"	25.40mm	2540

CMI	Conve	Convert Millimeter => Inch																
Execution time		6303	307 - 472	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs		6809	267 - 369	error messsage													0	О

Function: A millimeter number in integer long format will be converted to inches (1 millimeter = 0.03937 inches). Accuracy can be

defined by using powers of 10. To ensure measuring accuracy within a program the same power of 10 must be used

throughout.

Operands OP1 and OP2 remain unchanged.

Parameters: R1046& Source address for mm number (Integer long)

Destination address for inch number (Integer long) Χ

Exponent of mm number (0 through 3)

Exponent of inch number (0 through 5)

Result: D The two lower bytes of the inch number N. Z

Correspond to the contents of D

#### Example:

The live value in a programming system is in metric but must be displayed on an operator panel in inches. The internal representation of live and setpoint values is in mm with a resolution of 0.01. The number is displayed in inches to four decimal digits (=> accuracy = 0.0001 inch).

:			
LDX#	R	0100	Source address of mm number
STX	R	1046	
LDAA	#	002	Exponent of mm number (accuracy = 0.01; => 2 decimal digits)
LDAB	#	004	Exponent of inch number (accuracy = 0.0001; => 4 decimal digits)
LDX#	R	0200	Destination address for inch number
CMI			Routine call
:			

Binary number in R0100 through R0103	mm number	Inch number	Binary number in R0200 through R0203
346	3.46 mm	0.1362 "	1362
20045	200.45 mm	7.8927 "	78927
2540	25.40 mm	1.0001 "	10001
10000	100.00 mm	393.75 "	39375

#### **FCOP Function Copy** 2 3 5 6 8 9 10 11 12 13 14 **Execution time** 6303 see table Possible 4 6809 in µs see table error message Function: A data block with length L is copied from a source address to a destination address. The routine recognizes from the entered source address, destination address and the length of the block whether the source and destination should overlap. Correspondingly the routine copies either forwards or backwards. The source is not changed in any way other than when it is overlapped by the destination.

T.DX#

STX

LDAA

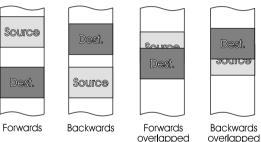
JSR

R 0100

R 1048

TEXT

# 000



Parameters:	R1048&	Destination address for the block Length of the block in bytes
	X	Source address of block

Result:	D	Changed
	N, Z, C	Invalid

		table and X the start dress of the table					
FCOP	Copy the table						
	Execution	time in µs					
	6303	6809					
Dest. addr. > Srce. addr.	82 + (\frac{L}{256} + 1)*55 + L*23	121 + L*10,5					
Dest. addr. < Srce. addr.	58 + (\frac{L}{256} + 1)*54 + L*25	112 + L*10,5					

Destination address

D contains the length of

**Example:** The TEXT table will be copied to the area starting from R0100.

FSMB	Functi	on Set Mer	mory Byte
Execution	n time	6303	48 + L*12

in μs 6809 94 + L\*7,5 error message | | | | | | |

Possible

2 3

5

6

8

9 | 10 | 11 | 12 | 13 | 14

Function: All registers within a defined memory range (start address, length of memory range in bytes) are loaded with a 1 byte value.

Parameters: R1052 1 byte value

D Length of memory range in bytes X Start address of memory range

Result: D Changed

N, Z, C Invalid

**Example:** Registers R3000 to R3299 should be loaded with the value 255 (\$FF):

:

 LDAA
 #
 255
 Load 1 byte value in A

 STAA
 R
 1052
 Store 1 byte value in R1052

LDD # 00300 Length of memory range in bytes (300)
LDX# R 3000 Start address of memory range

FSMB

:

FSMW Function Set Mem	nory Word
-----------------------	-----------

<b>Execution time</b>	6303	40 + L*14	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	94 + L*8,5	error message														

Function: All registers within a defined memory range (start address, length of memory range in words) are loaded with a 2 byte

value.

Х

Parameters: R1052& 2 byte value
D Length of me

Length of memory area in words Start address of memory range

Result: D Changed

N, Z, C Invalid

**Example:** The double register in the memory range from R3000 to R3299 should be loaded with the value 1000 (\$03E8):

:

LDD # 00150 Length of memory range in words (150 words => 300 byte)

FSMW:

FCLR	Functi	Function Clear Memory																
Execution	n time	6303	48 + L*12	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	94 + L*7,5	error message														

Function: All registers within a defined range (Start address, length of memory range in bytes) are deleted (overwritten with null

value).

Parameters: Length of memory range in bytes Χ

Start address of memory range

Result: Changed

N. Z. C Invalid

Example: Registers R0100 to R0199 should be deleted

# 00100 LDD LDX# R 0100

Length of memory range Start address of memory range

FCLR

# MCMP Compare OP1 and OP2

Execution time	6303	201 - 223	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µs	6809	195 - 207	error message												0		

Function: Operands OP1 and OP2 are compared with each other and depending on the result flags N, Z, and C are set. the operands

remain unchanged.

Parameters: None

Result: D Changed

R1024 Error number (0 => no error) N, Z, C Correspond with routine result

After the comparison is made the following conditional branches are possible:

Branch, if	Branch instruction
OP1 = OP2	BEQ (SP0)
OP1 ≠ OP2	BNE (SN0)
OP1 < OP2	BCS (SP<)
OP1 ≤ OP2	BLS (J<=)
OP1 > OP2	BHI (SP>)
OP1 ≥ OP2	BCC (JC0)

MHIL	Limitat	Limitation to High Limit; If OP1 > OP2, then OP2 ⇒ OP1																
Executio	n time	6303	215 - 271	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13	14
in µ	s	6809	205 - 237	error message												0		

Function: Operands OP1 and OP2 are compared with each other. If OP1 is larger than OP2, then OP1 is loaded with the contents of

OP2.

Parameters: OP1 Value to be limited

OP2 High limit

Result: D Changed

R1024 Error number (0 => no error)

N, Z Invalid

C Set, if OP1 is loaded with OP2

MLOL  Li	L   Limitation to Low Limit; If OP1 < OP2, then OP2 ⇒ OP1																
Execution tim	e 6303	215 - 271	Possible	1	2	3	4	5	6	7	8	9	10	11	12	13 1	4
in µs	6809	205 - 237	error message												0		1

Function: Operands OP1 and OP2 are compared with each other. If OP1 is smaller than OP2, then OP1 is loaded with the contents of

OP2.

Parameters: OP1 Value to be limited

OP2 Low limit

Result: D Changed

R1024 Error number (0 => no error)

N, Z Invalid

C Set if OP1 is loaded with OP2.

### 5. SPECIAL INSTRUCTIONS

## 5.1. ARITHMETIC PROCESSOR (ONLY CP80)

All central and peripheral processors are equipped with floating point mathematic routines as a standard (see section "4. Mathematic Routines"). The CP80 processor is enhanced by an additional fast floating point mathematic processor (MC68881, Motorola). This math processor (hereafter abreviated as APU) is utilized, if:

- the precision of floating point mathematic routines is not sufficient
- the calculating speed of floating point mathematic routines is not sufficient
- a calculation is required, which is not included in the instructions set of the floating point mathematic routines (e.g. Goniometric functions, Logarithms, ...)

## **5.1.1. Operands**

The arithmetic processor is equipped with 8 internal arithmetic registers. These are described as "internal operands". Others are "external operand" numbers which sit in the central processor's data range (R0000 to R7167). It is advisable to differentiate between the following APU instruction types:

- Instructions used for loading or reading APU operand registers
- Instructions used to couple two APU operand registers (two internal operands)
- Instructions used to couple an internal operand with an external operand

## 5.1.2. APU Instruction Calls

Instructions are transferred to the APU with the STL command "MAT". Index registers and accumulators include the following parameters:

A	7 0 PTR F/REG REG	В	7 0 EI D CODE
PTR	Pointer to external operands  00 = index register X  01 = index register Y 1)  10 = user stack pointer 1)	EI D	Instruction type: 0 = two internal operands 1 = one external operand Loading / saving internal operands:
F/REG	If the instruction includes two internal operands: F/REG = Register No. of second operand (OP2 <sup>2</sup> )  If the instruction contains an external operand:	CODE	0 = loading 1 = saving Instruction code (see "68881 instruction set")
REG	F/REG = Format code (see table in section "Number format")  Register No. of the first operand (OP1 <sup>2</sup> )	IS.	0 7 0 Stend Operard Stat Motes  Index register X, Y or user stack pointer 1)

Index register Y and the user stack pointer can be used with the B&R PROgramming SYStem version 5.0 or higher.

#### APU instruction example:

LDX#	R	2000	Pointer to external operands
LDAA	#	%00100011	Data format "Word Integer"
LDAB	#	%10xxxxxx	An external operand, xxxxxx = Instruction
MAT			

<sup>&</sup>lt;sup>2)</sup> OP1 and OP2 are not identical to the operands in mathematic routines.

# 5.1.3. Number Formats

FORMAT	Description	Length	Format code 1)
7 0 8 Bits	BYTE INTEGER	8 Bit / 1 Byte	110
15 0 16 Bits	WORD INTEGER	16 Bit / 2 Byte	100
31 0 32 Bits	LONG INTEGER	32 Bit / 4 Byte	000
30 22 0 8 Bit Exponent 23 Bit Mantissa Mantissa sign	SINGLE REAL	32 Bit / 4 Byte	001
62 51 0 11 Bit Exponent 52 Bit Mantissa Mantissa sign	DOUBLE REAL	64 Bit / 8 Byte	101
62 80 63 0 15 Bit Exp. Null 64 Bit Mantissa Mantissa sign Decimal point (implicit)	EXTENDED REAL	96 Bit / 12 Byte	010
95 91 80 67 0 3 Digit Null 17 Digit Mantissa Exponent Decimal point (implicit)	PACKED DECIMAL REAL	96 Bit / 12 Byte	011
Exponent sign  Mantissa sign			

# 5.1.4. 68881 APU Instruction Set

Instruction	Designation	Description	Code (hex.)	Code (binary)
ABS	absolute value	OP1 := abs (OP1)	\$18	%011000
ACOS	arccosine	OP1 := arccos (OP1)	\$1C	%011100
ADD	addition	OP1 := OP1 + OP2	\$22	%100010
ASIN	arcsine	OP1 := arcsin (OP1)	\$0C	%001100
ATAN	arctangent	OP1 := arctan (OP1)	\$0A	%001010
ATANH	hyperbolic arctangent	OP1 := atanh (OP1)	\$0D	%001101
cos	cosine	OP1 := cos (OP1)	\$1D	%011101
COSH	hyperbolic cosine	OP1 := cosh (OP1)	\$19	%011001
DIV	division	OP1 := OP1 / OP2	\$20	%100000
ETOX	e <sup>x</sup>	OP1 := e <sup>OP1</sup>	\$10	%010000
ETOXM1	e <sup>x-1</sup>	OP1 := e <sup>OP1-1</sup>	\$08	%001000
GETEXP		OP1 := exponent (OP1)	\$1E	%011110
GETMAN		OP1 := mantissa (OP1)	\$1F	%011111
INT	integer function	OP1 := int (OP1)	\$01	%000001
INTRZ	integer with rounding	OP1 := int (OP1)	\$03	%000011
LOG10	logarithm base 10	OP1 := log <sub>10</sub> (OP1)	\$15	%010101
LOG2	logarithm base 2	OP1 := log <sub>2</sub> (OP1)	\$16	%010110
LOGN	logarithm base e	OP1 := In (OP1)	\$14	%010100

LOGNP1		OP1 := In (OP1 + 1)	\$06	%000110
MOD	modulo function	OP1 := mod (OP1)	\$21	%100001
MOVE	load or store		\$00	%000000
MOVECR	load with constant	OP1 := const.	\$3B	%111011
MUL	multiplication	OP1 := OP1 * OP2	\$23	%100011
NEG	negation	OP1 := 0 - OP1	\$1A	%011010
SCALE		OP1 := OP1 * int (2 <sup>OP1</sup> )	\$26	%100110
SGLDIV	single precision division	OP1 := OP1 / OP2	\$24	%100100
SGLMUL	single precision multiplication	OP1 := OP1 * OP2	\$27	%100111
SIN	sine	OP1 := sin (OP1)	\$0E	%001110
SINCOS	sine and cosine	OP1 := sin (OP1); reg. n := cos (OP1)	\$3n	%110nnn
SINH	hyperbolic sine	OP1 := sinh (OP1)	\$02	%000010
SQRT	square root	OP1 := sqr (OP1)	\$04	%000100
SUB	subtraction	OP1 := OP1 - OP2	\$28	%101000
TAN	tangent	OP1 := tan (OP1)	\$0F	%001111
TANH	hyperbolic tangent	OP1 := tanh (OP1)	\$09	%001001
TENTOX		OP1 := 10 <sup>OP1</sup>	\$12	%010010
TWOTOX		OP1 := 2 <sup>OP1</sup>	\$11	%010001

The average execution time of APU instructions is approx. 330  $\mu sec.$ 

## **5.1.5. CONSTANTS**

The most important technical constants are already stored in the 68881 math coprocessor. The instruction \$3B (%111011) loads an operand register with one of these constants. Before the invocation of this instruction, the pointer defined in A (index register X, Y or user stack pointer) points to a storage address that contains the number of the desired constant from the following table:

No.	Constant	No.	Constant	No.	Constant	No.	Constant
\$00	Р	\$30	In(2)	\$36	108	\$3C	10512
\$0B	log <sub>10</sub> (2)	\$31	In(10)	\$37	10 <sup>16</sup>	\$3D	101024
\$0C	е	\$32	100	\$38	1032	\$3E	102048
\$0D	log <sub>2</sub> (e)	\$33	10¹	\$39	10 <sup>64</sup>	\$3F	10 <sup>4096</sup>
\$0E	log <sub>10</sub> (e)	\$34	10 <sup>2</sup>	\$3A	10 <sup>128</sup>		
\$0F	0	\$35	10⁴	\$3B	10 <sup>256</sup>		

**Example:** Load operand register 2 with the constant e (No. \$0C).

LDAA	#	\$0C	Number of constant
STAA	R	0100	Interim storage
LDX#	R	0100	Index register X set to constant number
LDAA	#	%00000010	Register number 2
LDAB	#	%00111011	Instruction code "load constant"
MAT			

**Example:** Multiply contents of APU operand registers 1 and 4:

MAT

**Example:** Load APU register 2 with 2 byte integer in R 0100, 0101:

LDX# R 0100 Data source address

LDAA # %00100010 Number format and register number

# %10000000 Instruction code for loading / storage = \$00

LDAB MAT

**Example:** Store result in APU register 6 in integer long format (4 bytes) in registers R 0200 to R 0203:

LDX# R 0200 Destination address
LDAA # %00000110 Number format and register number

LDAB # %11000000 Instruction code for loading / storage = \$00

MAT

## 5.2. INTERFACE INSTRUCTIONS (ONLY CP80, PP60 AND NTCP6#)

The following STL instructions support the software operation of interfaces used with the CP80, PP60 and NTCP6#:

SOB Output character SIB Read character

SC Request interface status

SF Interface functions (e.g.: initialization)

## 5.2.1. SOB - Output Character

The SOB instruction causes a single character to be output to the serial interface. The interface must have been initialized beforehand (see SF instruction). The character to be output is placed in accumulator A. Accumulator A is not changed by the SOB instruction. After the execution of SOB, the carry flag indicates whether the transmission was successful:

Carry flag = 0 Character output

Carry flag = 1 Output not possible (transmission buffer full)

**Example:** output of ASCII character "A"

LDAA # 'A Transfer the "A" character to accumulator A in the interface routine

SOB Transmit

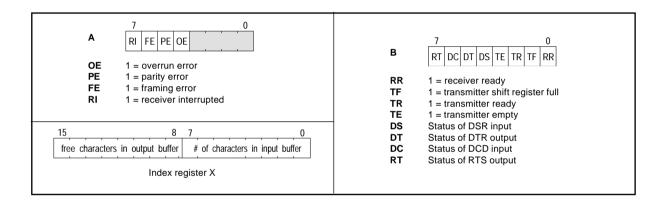
### 5.2.2. SIB - Read Character

The SIB instruction reads a single character from the input buffer. The interface must have been initialized beforehand (see SF instruction). After the SIB instruction has been executed the character read is held in accumulator A. If no character is received, the accumulator A is not changed by the SIB instruction. The carry and zero flags indicate whether a character was received or whether an interface error occurred:

C = 0		C=1	
	no valid character received		
	Z = 1	Z = 0	
valid character received and stored in accumulator A	input buffer empty	Transmission error; Accumulator A contains and error code:  7 0  RI FE PE OE  OE 1 = overrun error PE 1 = parity error FE 1 = framing error RI 1 = receiver interrupted	

## 5.2.3. SC - Request Interface Status

The SC instruction provides information on the status of the interface and the input / output buffers. The interface must be initialized before the SC function is executed. After the execution of SC the registers (A, B) and the index register contain the following information.



# 5.2.4. SF - Interface Functions (e.g. Initialization)

The SF instruction is used to:

- initialize an interface
- manually control handshake lines (DTR, RTS)
- clear input or output buffer
- define parameters for transmitting / receiving in block mode
- transmit and receive in block mode
- request block mode status

In addition to operation with the SOB (transmit single character) and SIB (receive single character) instructions, the interface can be operated in block mode. This allows the user to transmit and receive entire data blocks (frames). This function is also necessary to communicate with operator interface panels that have network capability (e.g. BRRT28) or with the mass storage device BRMEC.

The block mode function can only be used if:

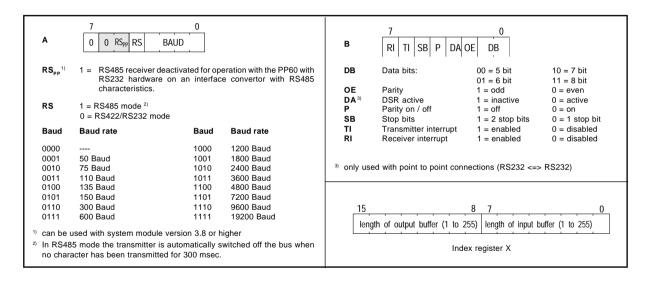
a. The B&R PROgramming SYStem Version 5.0 or a later version is used

or

 The system module of an older B&R PROgramming SYStem version was exchanged for a system module version 3.1 or later.

#### SF - Initialize Interface

The interface must be initialized before its first use regardless of whether the SOB / SIB (transmit / receive single character) instructions or block mode is to be used. The parameters are:



# SF - Manual Operation of Handshake Lines and Input / Output Buffers

7 0 1 0 0 0 0 0 0 0	LDAA SF	# \$80	Set RTS line to low. 1)
7 0 1 0 0 0 0 0 1	LDAA SF	# \$81	Set RTS line to high. <sup>1)</sup>
7 0 1 0 0 0 0 1 0	LDAA SF	# \$82	Automatic DTR handling on (after Power ON this is always switched off!).
7 0 1 0 0 0 0 1 1	LDAA SF	# \$83	Set DTR line to low. DTR stays on low until the \$82 instruction switches the automatic DTR handling on again and there is no busy status.
7 0 1 0 0 0 0 1 0 0	LDAA SF	# \$84	Clear input / output buffer (reset pointer). Busy status is reset unless it was locked with the instruction \$83.
7 0 1 0 0 0 1 0 1	LDAA SF	# \$85	Clear input buffer (reset pointer). Busy status is reset unless it was locked with the \$83 instruction.
7 0 1 0 0 0 0 1 1 0	LDAA SF	# \$86	Clear output buffer (reset pointer).

<sup>&</sup>lt;sup>1)</sup> With an RS232 connection to a bus (ECINT1) the DTR line is set with this command.

### **SF - Initialize Block Mode**

In block mode the user defines an input buffer and an output buffer. For transmission, the respective data block is written into this buffer and the transmission process is begun with the SF instruction. The transmission of the individual characters then occurs automatically, controlled by a timer interrupt routine activated with the user timer interrupt handler \$US2.

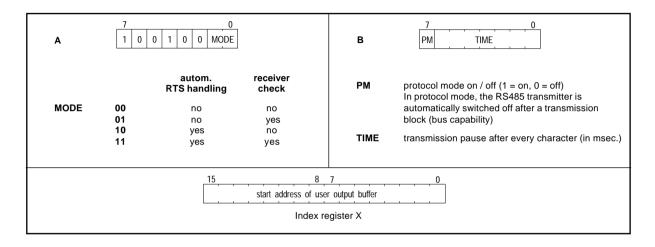
Block transmission and receiving can also occur with B&R standard protocol (MININET protocol). This protocol is required, e.g., for communication with a BRRT28 operator interface panel or a BRMEC mass storage device. In this case the entries in the output buffer (the frames) must match the B&R protocol.

Command from Master	STX LEN NODE INDEX DATA CHK	
Response from slave without data (short response)	ACK	
Response from slave with data (long response)	STX LEN NODE INDEX DATA CHK	
STX Start charac	cter which indicates the start of a frame (\$02)	
LENLength of th	ne entire frame	
NODE Node numb	per of the intended receiver	
INDEX Index number for frame identification		
DATA Data bytes. After every \$02 data byte a fill byte is inserted and transmitted, to enable \$02 (=> STX) as a data byte.		
	over the frame. This is computed automatically by the operating system and e sent by the user.	
ACK Confirmatio	on that a frame was received without errors (\$06)	

The following procedure is to be followed for the initialization:

- a. Initialize the block mode transmission with the SF instructions \$90 to \$93. This establishes the following:
  - the start address of the user output buffer
  - a transmission pause after every character
  - whether the RTS line is to be used for the handshake
  - whether a receiver check is to be carried out
  - whether the blocks are to be transmitted in protocol mode
- b. Start the timer interrupt routine with the timer interrupt handler \$US2. The required parameters are automatically determined by the SF invocation above.
- c. Initialize block mode reception with the SF instruction \$98. This establishes:
  - the start address of the user input buffer
  - a timeout for reception

## a. Initialize for Transmission in Block Mode



## b. Start UserTimer Interrupt Routine

The user timer interrupt handler \$US2 completes the transmission block mode initialization. The SF invocation described in a. automatically defines the parameters for \$US2.

#### Example:

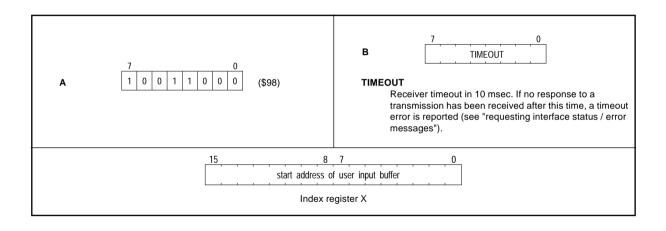
The user interface is initialized for block mode without B&R protocol. The user buffer begins at R 3000. Since the receiver does not avail any handshake lines, the automatic RTS handling is turned off and a two msec. pause is inserted after each character. Baudrate, stop bits and parity need to have been initialized beforehand.

LDX#	R	3000	Start address of user output buffer
LDAB	#	002	2 msec. transmission pause after each character
LDAA	#	%10010000	Autom. RTS handling off, check off
SF			
JSR		\$US2	Start timer interrupt routine

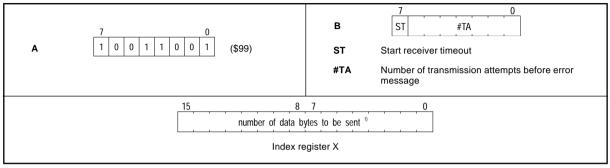
There must be no instruction that changes the contents of index register X or accumulator A between the SF invocation and the start of the timer interrupt routine with JSR \$US2.

## c. Initialize for Receiving in Block Mode

This instruction must be invoked after the initialization of transmission in block mode. It defines the start address of the user input buffer. Repeated invocation of this instruction with various buffer start addresses allows various input buffers to be used alternately.



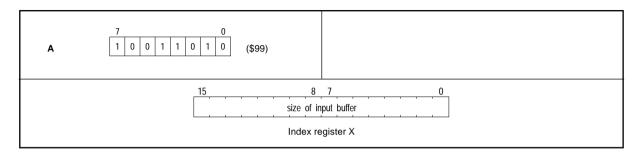
## SF - Transmit Data in Block Mode



<sup>&</sup>lt;sup>1)</sup> In protocol mode the number of characters to be sent is specified in the frame and the specification in index register X is ignored.

If the protocol mode is on and the first character in the output buffer is not STX (\$02), then the data block is not sent in protocol mode. This function can be used in order to transmit special characters that are control characters in protocol mode (e.g. \$06).

### SF - Receive Data in Block Mode



The size of the input buffer must be at least 32 characters. It must also be 2 characters larger than the actual data block length. In protocol mode the size of the input buffer is predefined to 255 + 2 characters; i.e., the specification in index register X is ignored and the user must have a buffer region of 255 + 2 characters available.

**Example:** Receive a data block; input buffer size = 128 bytes:

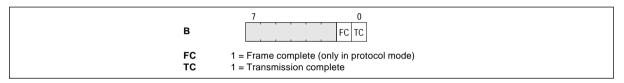
LDD # 00128 size of input buffer
XGDX
LDAA # \$9A instruction "receive data"

LDAA # \$9A instruction "receive data"
SF

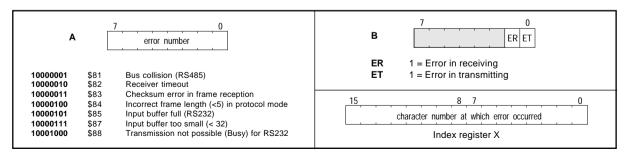
## SF - Request Interface Status (\$9F)

This instruction requests information about the state of the interface (only relevant in block or protocol mode).

The response depends on whether an error has occurred. If the carry flag = 0 after the invocation of SF, then no error occurred and accumulator B contains the following:



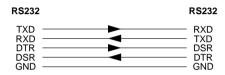
In case of error the carry flag is set after the SF invocation and accumulator A contains an error code:



## 5.2.5. Handshake Techniques

During initialization, the respective interrupts must be set if using transmitter and/or receiver. The following handshake line representations should be differentiated between.

#### Point to Point RS232 <=> RS232 (SIB/SOB, Block mode)



#### Point to Point RS232 <=> ECINT1 (SIB/SOB, Block mode)

RS232	ECINT1
TXD RXD DTR GND	RXD TXD DSR GND

#### RS485 (SIB/SOB, Block mode)

RS485	RS485
DATA —	——— DATA
DATA —	DATA

Following is a description of all possible connections and different transmitting and receiving operation types. For each type of operation an explanation of how the interface and data lines (hardware) are influenced by data bits DSR<sup>1)</sup>, DTR<sup>2)</sup> and RTS<sup>2)</sup>.

# a) RS232 (RS422) Point to Point

Interface initialization: RS Bit = 0 (RS232/422)

#### SIB/SOB:

DSR: DSR: inactive: The "busy" signal from the opposite station (DTR set low) is ignored.

DSR active: If the opposite station sends a "busy" signal, up to two more characters can be sent (characters

which are already in the interface circuit).

DTR: DTR handling is automatically turned off at power on. DTR handling is switched on with the \$82 command (when the buffer becomes 80% to 90% full the "busy" signal is sent and DTR is set low again). To manually override DTR handling a "busy" signal to the opposite station can be sent with the \$83 command.

RTS: Not used.

#### Block mode without protocol:

DSR: Same as SIB/SOB

DTR: Same as SIB/SOB

RTS: RTS is set to high with the command "send data in BM".

The RTS line can be set back to low by the \$80 command.

The user must pay attention that when all characters have entered the interface buffer and the transfer time is expired the RTS line should be switched back to low. After 300 msec. of no transmission the line is set to low automatically.

#### Block mode with protocol:

DSR: Same as SIB/SOB

DTR: Is not maintained (string must fit in the buffer)

RTS: Same as block mode without protocol.

Data format in transmit buffer:

STX LEN NODE INDEX DATA ......DATA

The checksum is calculated by the operating system. Fill bytes are also generated after every \$02 in the data stream.

<sup>1)</sup> see "SF - Interface Initialization"

<sup>2)</sup> see "SF - Manual Operation of Handshake Lines"

## b) RS232 through a Bus Convertor

Interface Initialization: RS Bit = 1 (RS485)

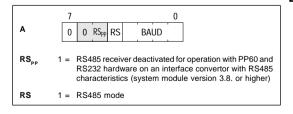
#### SIB/SOB:

DSR: Not used

DTR: Is not maintained

RTS: If RS485 mode is used with RS232 hardware using a bus convertor, switching between active and passive is done with DTS and the \$80 (DTR low) and \$81 (DTR high) commands. Any external device used must be able to generate an echo byte. The echo byte returned from a bus convertor or an external device must be evaluated by the program (during reading or clearing the receive buffer).

In order to facilitate this function also in the PP60, the RS485 receiver must be deactivated ensuring that TTY, RS232 and RS485 do not affect one another.



#### Block mode without protocol:

DSR: Not used

RTS: Same principle as SIB/SOB, otherwise the echo byte is handled by the operating system. The bus is active with the "send data in BM" command and when the last echo byte is received is switched back to passive.

!! NO MANUAL OPERATION !!

#### Block mode with protocol:

Data format in transmit buffer:

STX LEN NODE INDEX DATA.....DATA

otherwise the same as RS232 - Point to point connection

# c) RS485 (Networkable)

Interface Initialization: RS Bit = 1 (RS485)

#### SIB/SOB:

DSR: Not used

DTR: Is not maintained

RTS: Bus is switched between active and passive with \$80 and

\$81 commands. The echo byte must otherwise be evaluated. If no character is sent within 300 msec. the transmitter is switched automatically from the bus.

#### Block mode without protocol:

DSR: Not used

DTR: Is not maintained

RTS: The echo byte is handled by the operating system. The bus

is switched to active with the "send data in BM" command

and to passive by receiving the last echo byte.

!! NO MANUAL OPERATION !!

The RS<sub>PP</sub> bit must be 0.

#### Block mode with protocol:

DSR, DTR, RTS: Same as block mode without protocol

Data format in transmit buffer

STX LEN NODE INDEX DATA.....DATA

The checksum is calculated by the operating system. Fill bytes are also generated after every \$02 in the data stream.

The RS<sub>PP</sub> bit must be 0.

## 6. APPENDIX

# 6.1. Alphabetical Overview of B&R Mnemonics

+	92	COA	175	LE!	46	SEI	173
++B	95	COB	176	LER	48	SET	169
+B	94	DA	126	LEU	47	SK0	152
+D	96	DB	127	LEY	49	SK1	154
-	98	DEC	125	LR	39	SL	132
SUB	99	DK	177	LRK	40	SLA	133
B	101	DR	128	LRL	41	SLB	134
-B	100	DS	129	LS	45	SLD	135
-D	102	DXR	66	LY	42	SLI	140
=	52	EB	89	LYK	43	SN0	148
=B	53	EIM	90	LYL	44	SN0L	150
=D	54	END	162	MAB	60	SP<	148
=R	55	EXG	68	MAC	62	SP <l< td=""><td>150</td></l<>	150
=S	57	EXO	88	MBA	61	SP>	148
=Y	56	IA	121	MCA	63	SP>L	150
A*B	105	IB	122	MRS	65	SP0	148
A+B	97	INC	120	MSR	64	SP0L	150
A-B	103	IR	123	NOP	161	SPI	160
ADD	93	IS	124	OB	86	SPU	156
AIM	84	J+	148	OD	85	SR	136
ANS	74	J+L	150	OIM	87	SRA	137
AVB	108	J-	148	PRS	164	SRB	138
AVS	75	J-L	150	PSH	70	SRD	139
В	115	J<=	148	PUL	72	SRE	143
B+R	104	J<=L	150	RET	158	TFR	67
BB	116	JC0	148	RLA	141	TIM	117
BNS	76	JC0L	150	RLB	142	UB	83
BVS	77	K	174	RNS	78	UND	82
CLA	167	LAD	34	RRA	144	VB	110
CLB	168	LB	35	RRB	145	VR	111
CLC	170	LD	36	RST	165	VRK	112
CLI	172	LDK	37	RVS	79	VY	113
CLR	166	LDL	38	SEC	171	VYK	114
CMP	109						

# **6.2. Alphabetical Overview of Motorola Mnemonics**

ABA ABX ADCA ADCB ADDA ADDB ADDD AIM ANDA	97 104 93 95 92 94 96 84	CLI CLR CLRA CLRB CMPA CMPE COM COMA
ANDB	83	CPX
ASL	132	CPX#
ASLA	133	CPY
ASLB	134	CPY#
ASLD	135	DAA
BCC	148	DEC
BCCL	150	DECA
BCS	148	DECB
BCSL	150	DES
BEQ	148	DEX
BEQL	150	EIM
BHI	148	END
BHIL BITA	150	EORA
BITB	115 116	EORB EXG
BLS	148	INC
BLSL	150	INCA
BMI	148	INCB
BMIL	150	INS
BNE	148	INX
BNEL	150	JMP
BPL	148	JSR
BPLL	150	LDAA
CBA	108	LDAB
CLC	170	LDD

.l	172	LDK	
.R	166	LDL	
.RA	167	LDS	
.RB	168	LDX	
/IPA	109	LDX#	
/IPB	110	LDXL	
M	174	LDY	
AMC	175	LDY#	
OMB	176	LDYL	
PΧ	111	LEA!	
X#	112	LEAU	
Υ	113	LEAX	
PY#	114	LEAY	
λA	177	LSR	1:
С	125	LSRA	1:
CA	126	LSRB	1
CB	127	LSRD	1
S	129	MUL	1
X	128	NOP	1
М	90	OIM	
ID	162	ORAA	
RA	88	ORAB	
RB	89	PRS	1
(G	68	PSH	
С	120	PSHA	
CA	121	PSHB	
CB	122	PSHX	
S	124	PUL	
X	123	PULA	
IP	160	PULB	
R	156	PULX	
AA	34	ROL	1
AB	35	ROLA	1
D	36	ROLB	1.

ROR	143
RORA	144
RORB	145
RST	165
RTS	158
SBA	103
SBCA	99
SBCB	101
SEC	171
SEI	173
SET	169
SK0	152
SK1	154
STAA	52
STAB	53
STAD	54
STS	57
STX	55
STY	56
SUBA	98
SUBB	100
SUBD	102
TAB	60
TAP	62
TBA	61
TFR	67
TIM	117
TPA	63
TSX	64
TXS	65
XGDX	66

# 6.3. Alphabetical Overview of Mathematic Routines

# 6.4. Alphabetical Overview of B&R Short Mnemonics

CAF CBCD CBIN CBP CBPP CBPQ CFA CFA0 CFEA CIA	199 210 211 218 214 216 200 202 204 204 206	LF2 LIL1 LIL2 LIW1 LIW2 MADD MCMP MCOP MDIV MEXG
CIA0 CIM	208 222	MHIL MLOL
CMI	224	MMUL
CPB	220	MSGN
CPBQ	217	MSQR
FCLR	229	MSUB
FCOP	226	RFM1
FM2B	197	RFM2
FM3B	197	RFM3
FM4B FSMB	198	SAL
FSMW	227 228	SAW SFM1
LAL1	187	SFM1 SFM2
LAL1	187	SFM3
LAU1	188	SFX
LAW2	188	SIL
LF1	191	SIW

	99	KA	175
++	93	KB	176
A	93	L	34
С	166	0	85
E	88	Р	164
l	52	R	165
J<	148	RL	140
J>	148	RR	143
J0	148	U	82
J1	148	V	109

# 6.5. Index

Α		Н		N	
Absolute addressing	21	Half carry	14	Negation	32
Address preselection	13	_		Negative flag	14
Addressing modes	12, 17			Number formats (APU)	235
Direct addressing	19			Number formats	181
Extended addressing	21	IEEE-Format	181	Absolute with sign	182
Immediate addressing	23	Immediate addressing	23	Single precision floating point	181
Implied addressing	17	Implied addressing	17	IEEE-Format	181
Indexed addressing	25	Indexed addressing	25	Integer	182
Indirect addressing	30	Indirect addressing	30		
Negation	32	Instruction set 68881		0	
Relative addressing	29	Arithmetic processor	236		
Arithmetic instructions	233	Interface instructions	241	Opcode	12
Arithmetic processor	233	SC - Request interface status	243	Overflow flag	14
		SF - Interface functions	244	_	
С		Block mode initialization	247	R	
		Interface initialization	245		
Carry flag	14	Manual operation of handshake lines		Relative addressing	29
Command abbreviation	10	and input / output buffers	246		
Condition code register	14	Receive data in block mode	253	S	
CPU types	9	Requesting interface status (\$9F)	254	Cinale president fleating point format	18
CPU type A	15	Transmit data in block mode	252	Single precision floating point format	18
CPU type B	15	SIB - Read character	242	Syntax	,
		SOB - Output character	241	7	
D		IRQ mask	14	Z	
Direct addressing	19	M		Zero flag	14
		Mathematic routines	179		
		Mnemonics	10		