# *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

# Register

**STL Instructions**

**Mathematic Routines**

**Special Instructions**

# 1. SYNTAX EXPLANATION

## 1.1. CPU TYPE

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

| Processor | CPU TYPE | Module |
|---|---|---|
| **6303** | **A** | CP40, CP41, NTCP33, NTCP34, PSCP35, PP40, CP30, CP31, CP32 |
| **6809** | **B** | 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Motorola** | | **Function** | | | | | | | | | | | | | | | | | | | **CPU type A / 6303** | | | | | |
| **B&R** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Short** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000** | | | | |
| | | | | | | | | | | | | | | | | | | | | | | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Motorola** | | **Function** | | | | | | | | | | | | | | | | | | | **CPU type B / 6809** | | | | | |
| **B&R** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Short** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **CP 80** | | | | |
| | | | | | | | | | | | | | | | | | | | | | | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

## 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:

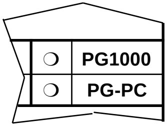
| | |
|---|---|
| A | Accumulator A (8 Bit) |
| B | Accumulator B (8 Bit) |
| D | Accumulator  D (16 Bit; A = HOB, B = LOB) |
| CCR | Condition code register (8 Bit) |
| DP | Register for direct addressing (8 Bit)<br>(Direct Page Register) |
| X | Index register X (16 Bit) |
| Y | Index register Y (16 Bit) |
| SP! | System stack pointer (16 Bit) |
| SPU | User stack pointer (16 Bit) |
| PC | Program counter |
| M | Address of a memory location |
| (M) | Contents of the memory location adressed by M (8 Bit) |
| (M:M+1) | Contents of both memory locations addressed by M and M+1 (16 Bit) |

| | |
|---|---|
| $r_8$ | 8 bit register (A, B, CCR, DP) |
| $r_{16}$ | 16 bit register (D, X, Y, SP!, SPU) |
| I | IRQ Mask |
| ∧ | Logical AND |
| ∨ | Logical OR |
| ⊕ | Logical exclusive-or (EXOR) |
| HOB | High order byte |
| LOB | Low order byte |
| EA | Effective address |
| IMM | Immediate value (constant) |
| dn | Data bit n of a register or position in memory |

# 1.3.4. Operating Mode

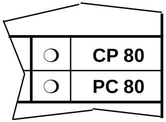
The symbol “❍” shows that the displayed command can be used in the PROgramming SYStem in this operating mode.

**Example:**



| | |
|---|---|
| ❍ | **PG1000** |
| ❍ | **PG-PC** |

or



| | |
|---|---|
| ❍ | **CP 80** |
| ❍ | **PC 80** |

# 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:

**IMPL.** **Implied**
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[1], SP![1], SPU[1] ) 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[1] byte offset.

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

[2] **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 | E | Input | |
| O | A | Output | |
| F | M | Flag | |
| S | F | Start timer | |
| T | Z | Timer (time elapsed flag) | |
| # | # | Constant | |
| P | P | Peripheral address | |
| R | C | Register | |
| X | I | Index register X | Indexed addressing |
| Y | Y | Index register Y[1] | |
| U | U | User stack pointer[1] | |
| ! | ! | System stack pointer[1] | |
| D | D | Direct addressing[1] | |
| G | G | Global RAM in PP60 (extended dual port RAM)[1] | |
| B | B | Block memory in PP60[1] | |

The following symbols will be used to explain:

❍ can be used in the specified operating mode

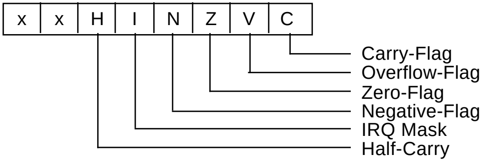
● can be used with PROgramming SYStem Version 5.00 in PC 80 operating mode

[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:** 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.

**Bit 6 and 7:** 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 **6809**!

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

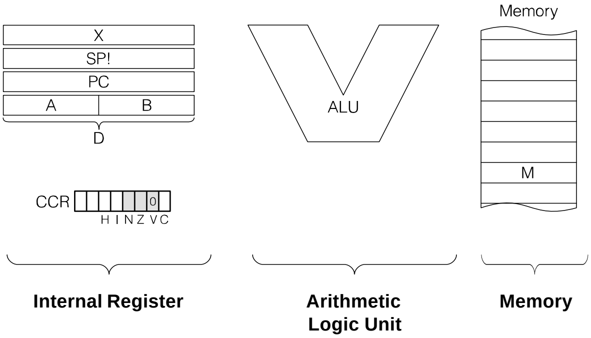
# 1.4. Detailed Description

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

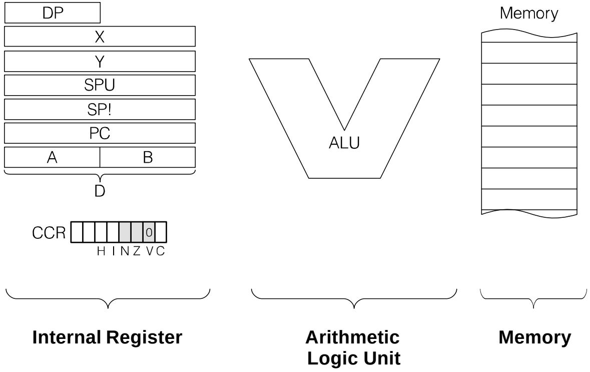
1) A detailed description of the command
2) A diagram showing the data flow

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

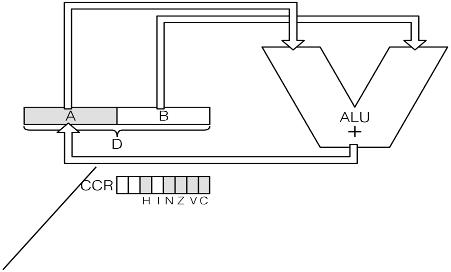
## CPU type A



## CPU type B



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.

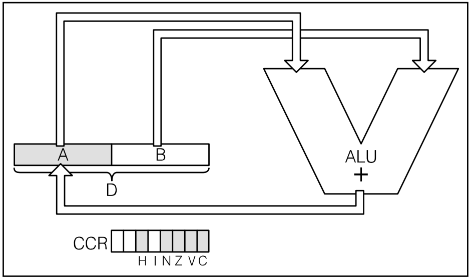
**Small arrow** represents a pointer (the contents of a register point to a memory location).

**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

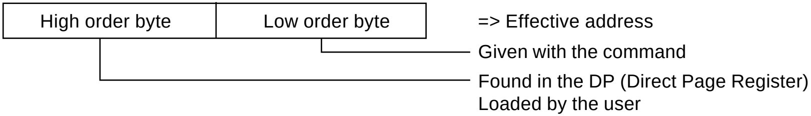
| MOTOROLA | B&R | MOTOROLA | B&R | MOTOROLA | B&R |
|---|---|---|---|---|---|
| ABA | A+B | DEX | DR | PULX | RVS |
| ABX | B+R | EXG[1] | EXG[1] | ROLA | RLA |
| ASLA | SLA | INCA | IA | ROLB | RLB |
| ASLB | SLB | INCB | IB | RORA | RRA |
| ASLD | SLD | INS | IS | RORB | RRB |
| CBA | AVB | INX |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

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

[2] 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.

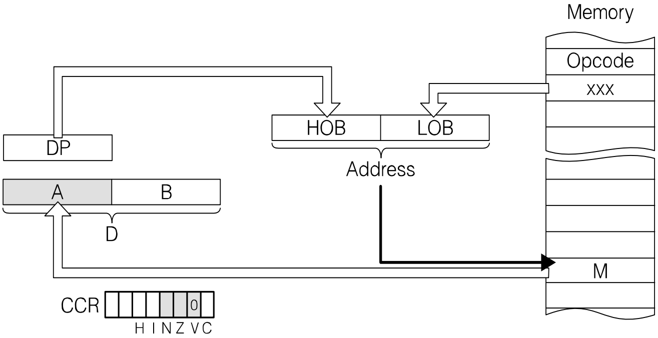


The memory range is broken down into pages of 256 bytes each. A page can be addressed directly with the preselection "D". This is written as follows. E.g.:

```
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.

**Example:** **`LDAA    D xxx`**



# Instruction Overview / Direct Addressing

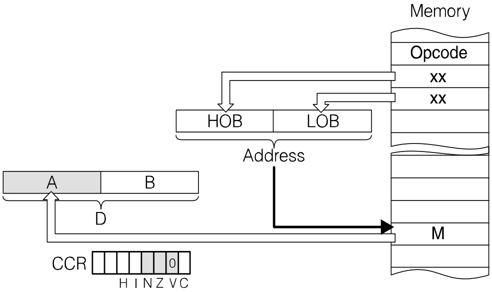
This addressing mode can only be used with the PROgramming SYStem Version 5.00 or higher in PC 80 operating mode.

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

**Example:** `LDAA R 0000`



# Instruction Overview / Extended Addressing

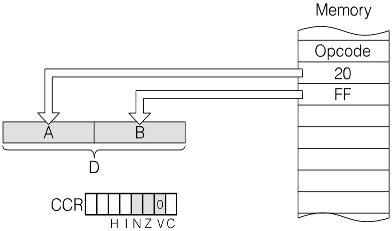
| MOTOROLA | B&R |
|---|---|
| ADCA | ADD |
| ADCB | ++B |
| ADDA | + |
| ADDB | +B |
| ADDD | +D |
| ANDA | UND |
| ANDB | UB |
| ASL | SL |
| BITA | B |
| BITB | BB |
| CLR | CLR |
| CMPA | CMP |
| CMPB | VB |
| COM | K |
| CPX | VR |
| CPY[1] | VY[1] |
| DEC | DEC |
| EORA | EXO |
| EORB | EB |
|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

[1] 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.

**Example:** **`LDD # $20FF`**



# Instruction Overview / Immediate Addressing

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

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

[2] can only be used with PROgramming SYStem Version 5.00 or higher in PG-PC operating mode (CPU type A only)

[3] 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<br>Y -32768 to Y 32767<br>U -32768 to U 32767<br>! -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<br>+16 to +127 | -32768 to -129<br>+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.

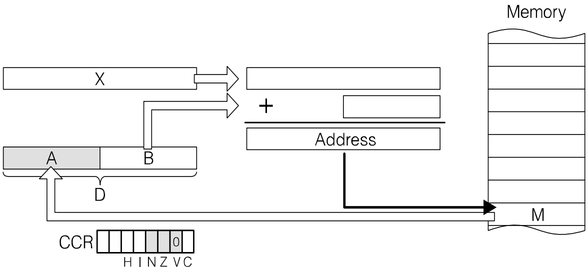
**Example:** Calculating the effective instruction length for the LDAA instruction in bytes. The description for this instruction shows the instruction length for indexed addressing (IND.) given with 2 +:

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

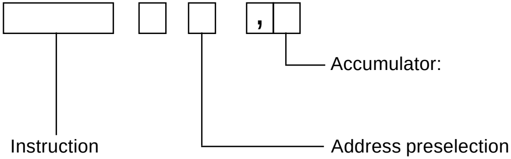
### 2.5.2. ACCUMULATOR - OFFSET INDEXED

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.

**Example:** **`LDAA      X ,B`**



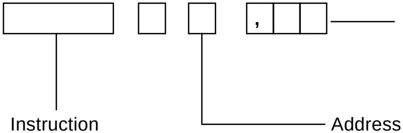
The following can be used for an accumulator offset:



| | | |
|---|---|---|
| A | => | Accumulator A |
| B | => | Accumulator B |
| D | => | Accumulator D |
| I | => | Index register X |
| Y | => | Index register Y |
| U | => | User stack pointer SPU |
| ! | => | System stack pointer SP! |

### 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.



| | |
|---|---|
| + | **Increment** the pointer register by **1 after** executing the instruction |
| ++ | **Increment** the pointer register by **2 after** executing the instruction |
| - | **Decrement** the pointer register by **1 before** executing the instruction |
| -- | **Decrement** the pointer register by **2 before** executing the instruction |

Instruction

Address preselection

| | | |
|---|---|---|
| X | => | Index register X |
| Y | => | Index register Y |
| U | => | User stack pointer SPU |
| ! | => | System stack pointer SP! |

**Example:** `LDAA    X ,++`

# Instruction Overview / Indexed Addressing

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

[1] 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

| SHORT | | LONG[1] | |
|---|---|---|---|
| **MOTOROLA** | **B&R** | **MOTOROLA** | **B&R** |
| BCC | JC0 | BCCL | JC0L |
| BEQ | SP0 | BEQL | SP0L |
| BHI | SP> | BHIL | SP>L |
| BCS | SP< | BCSL | SP<L |
| BLS | J<= | BLSL | J<=L |
| BMI | J- | BMIL | J-L |
| BNE | SN0 | BNEL | SN0L |
| BPL | J+ | BPLL | J+L |

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

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

[2] 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[1] | 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 | B | LDD | LD | SET | SET |
| BITB | BB | LDK | LDK | STAA | = |
| CLR | CLR | LDX# | LRK | STAB | =B |
| CMPA | CMP | LDY'[1] | LYK[1] | STD | =D |
| CMPB | VB | LSR | SR | SUBA | - |
| COM | K | | | SUBB | -B |

[1] 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 | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| LDAA | LAD | ❍ | ❍ | ❍ | ❍ |
| LDAB | LB | ❍ | ❍ | ❍ | ❍ |
| LDD | LD | ❍ | ❍ | ❍ | ❍ |
| LDK | LDK | | ❍ | | ❍ |
| LDL | LDL | | ❍ | | ❍ |
| LDX | LR | ❍ | ❍ | ❍ | ❍ |
| LDX# | LRK | ❍ | ❍ | ❍ | ❍ |
| LDXL | LRL | ❍ | ❍ | ❍ | ❍ |
| LDY | LY | | | | ❍ |
| LDY# | LYK | | | | ❍ |
| LDYL | LYL | | | | ❍ |
| LDS | LS | ❍ | ❍ | ❍ | ❍ |
| LEA! | LE! | | | | ❍ |
| LEAU | LEU | | | | ❍ |
| LEAX | LER | | | ❍ | ❍ |
| LEAY | LEY | | | | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | **LDAA** | **Function** | **CPU type A / 6303** |
| **B&R** | **LAD** | **(M) ⇨ A** | |
| **Short** | **L** | | |

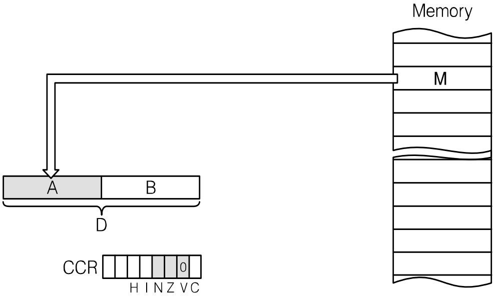
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 96 | B6 | 86 | A6 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | **LDAA** | **Function** | **CPU type B / 6809** |
| **B&R** | **LAD** | **(M) ⇨ A** | |
| **Short** | **L** | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 96 | B6 | 86 | A6 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



| | | | |
|---|---|---|---|
| Motorola | LDAB | Function | CPU type A / 6303 |
| B&R | LB | (M) ⇨ B | |
| Short | | | |

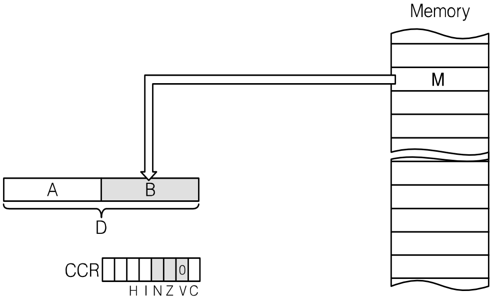
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D6 | F6 | C6 | E6 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| | | | |
|---|---|---|---|
| Motorola | LDAB | Function | CPU type B / 6809 |
| B&R | LB | (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D6 | F6 | C6 | E6 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



| | | | |
|---|---|---|---|
| Motorola | LDD | Function | CPU type A / 6303 |
| B&R | LD | (M:M+1) ⇨ D | |
| Short | | | |

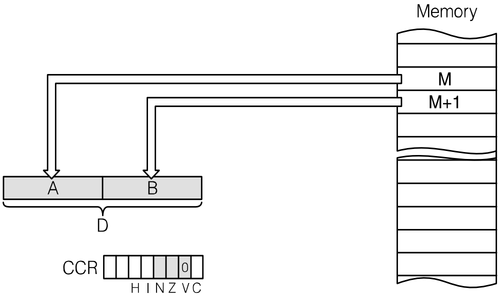
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 3/3 | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DC | FC | CC | EC | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| | | | |
|---|---|---|---|
| Motorola | LDD | Function | CPU type B / 6809 |
| B&R | LD | (M:M+1) ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 5/2 | 6/3 | 3/3 | 5+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DC | FC | CC | EC | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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.



| Motorola | LDK | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | LDK | M ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | |
|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. |
| | | | 3/3 | | |
| | | | CC | | |

| Address preselection | | | | | | | | | | | | | | | | PG1000 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | ○ | PG-PC | | | | | |
| I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | | |
| E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C | |
| ○ | ○ | ○ | ○ | ○ | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ | |

| Motorola | LDK | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | LDK | M ⇨ D | |
| Short | | | |

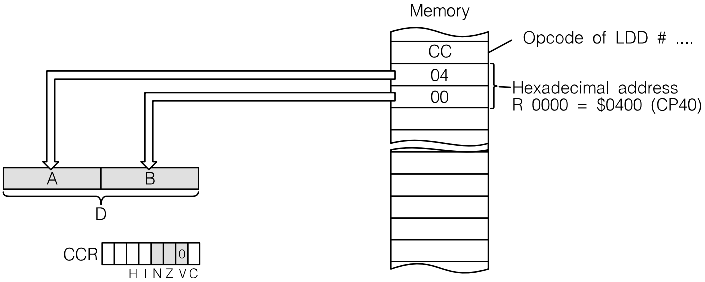
| Addressing mode / Opcode | | | | | |
|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. |
| | | | 3/3 | | |
| | | | CC | | |

| Address preselection | | | | | | | | | | | | | | | | CP 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | ○ | PC 80 | | | | | |
| I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | | |
| E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C | |
| ○ | ○ | ○ | ○ | ○ | | ○ | ○ | ○ | | | | | ● | ● | ○ | ○ | ● | ● | ▼ | ○ | |

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)



| Motorola | LDL | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | LDL | M ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | PG1000 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ❍ | PG-PC | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | CC | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

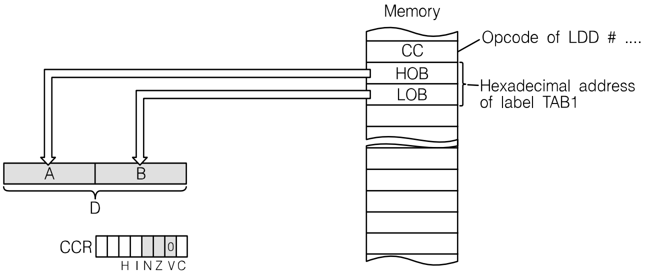
| Motorola | LDL | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | LDL | M ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | CP 80 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ❍ | PC 80 | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | CC | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

A label with address M is loaded to accumulator D.

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.

**Example:** `LDL TAB1`



**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).

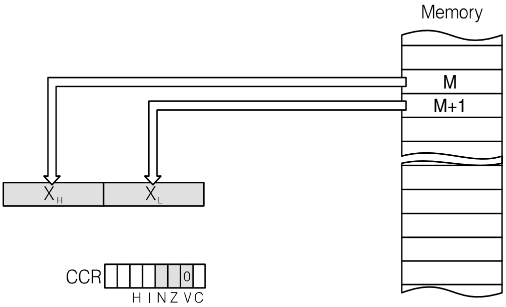
|  |  |  |  |
|---|---|---|---|
| Motorola | LDX | Function | CPU type A / 6303 |
| B&R | LR | (M:M+1) ⇨ X |  |
| Short |  |  |  |

| Addressing mode / Opcode |  |  |  |  |  | Address preselection |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ PG1000<br>❍ PG-PC |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR |  |  |  |  |  |
|  | 4/2 | 5/3 | 3/3 | 5/2 |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
|  | DE | FE | CE | EE |  |  |  |  |  |  |  | ❍ | ❍ | ❍ |  |  |  |  |  |  | ❍ | ❍ | ● | ● | ▼ | ❍ |

|  |  |  |  |
|---|---|---|---|
| Motorola | LDX | Function | CPU type B / 6809 |
| B&R | LR | (M:M+1) ⇨ X |  |
| Short |  |  |  |

| Addressing mode / Opcode |  |  |  |  |  | Address preselection |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ CP 80<br>❍ PC 80 |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR |  |  |  |  |  |
|  | 5/2 | 6/3 | 3/3 | 5+/2+ |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
|  | 9E | BE | 8E | AE |  |  |  |  |  |  |  | ❍ | ❍ | ❍ | ● |  | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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



| Motorola | LDX# | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | LRK | M ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | CE | | | ❍ | ❍ | ❍ | ❍ | ❍ | | ❍ | ❍ | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

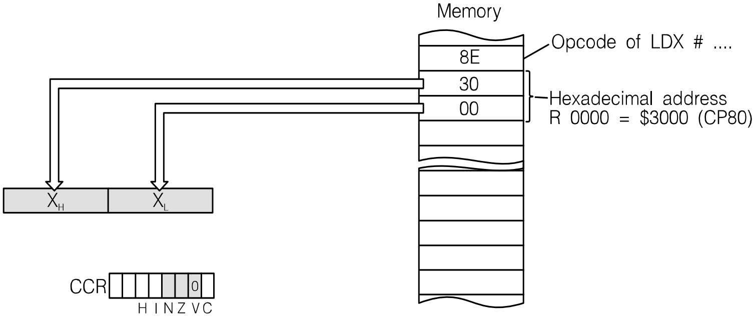
| Motorola | LDX# | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | LRK | M ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 8E | | | ❍ | ❍ | ❍ | ❍ | ❍ | | ❍ | ❍ | | | | | | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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)



| Motorola | LDXL | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | LRL | M ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | CE | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

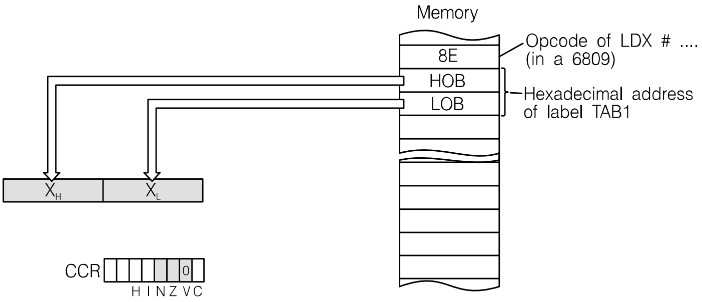
| Motorola | LDXL | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | LRL | M ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 8E | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

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).

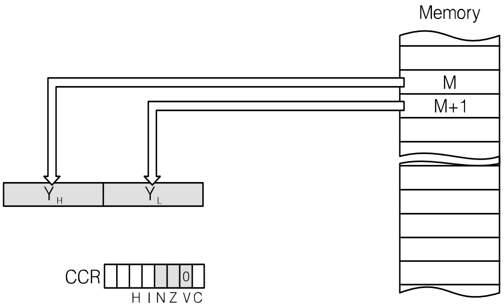
| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | □ | **PG1000**<br>**PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **LDY** | **Function** | **CPU type B / 6809** |
| **B&R** | **LY** | **(M:M+1) ⇨ Y** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | □<br>○ | **CP 80**<br>**PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 6/3 | 7/4 | 4/4 | 6+/3+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 10 9E | 10 BE | 10 8E | 10 AE | | | | | | | | ● | ● | ● | ● | | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | | | | **PG1000**<br>**PG-PC** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

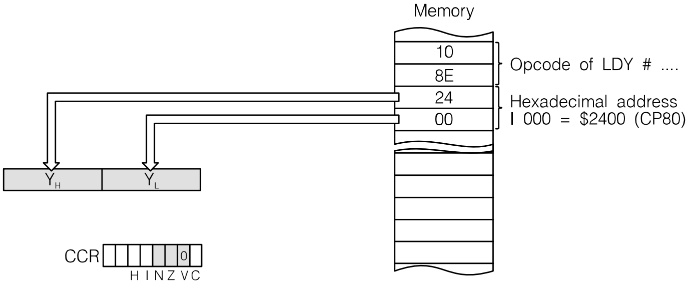
| | | | |
|---|---|---|---|
| **Motorola** | **LDY#** | **Function** | **CPU type B / 6809** |
| **B&R** | **LYK** | **M ⇨ Y** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | | | | **CP 80**<br>❍ **PC 80** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | 4/4 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 10 8E | | | ● | ● | ● | ● | ● | | ● | ● | | | | | | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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)



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **PG1000**<br>**PG-PC** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

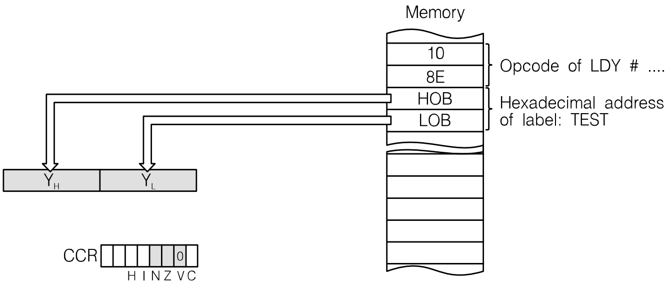
| | | | |
|---|---|---|---|
| **Motorola** | **LDYL** | **Function** | **CPU type B / 6809** |
| **B&R** | **LYL** | **M ⇨ Y** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **CP 80**<br>❍ **PC 80** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | 4/4 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 10 8E | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

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

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.

**Example:** **LYL** **TEST**



**Note:** 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).

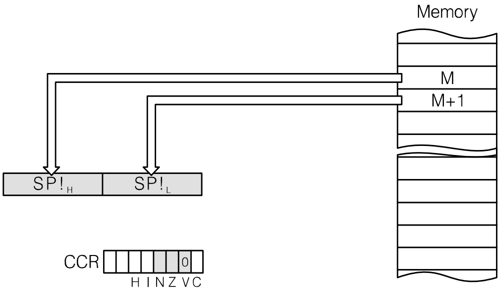
| | | | |
|---|---|---|---|
| **Motorola** | **LDS** | **Function** | **CPU type A / 6303** |
| **B&R** | **LS** | **(M:M+1) ⇨ SP!** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 4/2 | 5/3 | 3/3 | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9E | BE | 8E | AE | | | | | | | | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | **LDS** | **Function** | **CPU type B / 6809** |
| **B&R** | **LS** | **(M:M+1) ⇨ SP!** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 6/3 | 7/4 | 4/4 | 6+/3+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 10 DE | 10 FE | 10 CE | 10 EE | | | | | | | | ❍ | ❍ | ❍ | ● | | ● | ● | | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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



| | | | |
|---|---|---|---|
| Motorola | | Function | CPU type A / 6303 |
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | PG1000<br>PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| Motorola | LEA! | Function | CPU type B / 6809 |
| B&R | LE! | EA ⇨ SP! | |
| Short | | | |

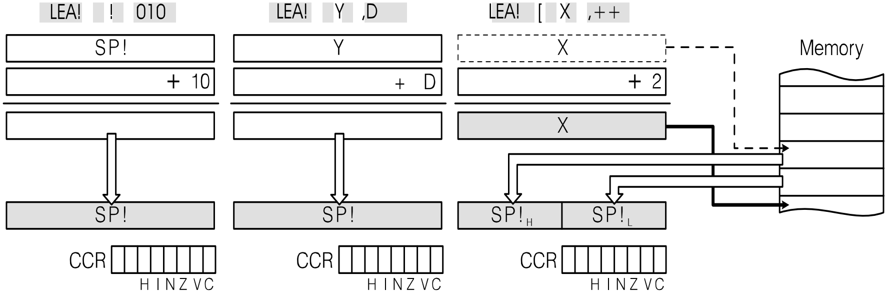
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 4+/2+ | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 32 | | | | | | | | | | | ● | ● | | ● | ● | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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 ⇨ SP!

`LEA! Y ,D` Y + D ⇨ SP!

`LEA! [ X ,++` (X:X+1) ⇨ SP!; X + 2 ⇨ X



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **PG1000<br>PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **LEAU** | **Function** | **CPU type B / 6809** |
| **B&R** | **LEU** | **EA ⇨ SPU** | |
| **Short** | | | |

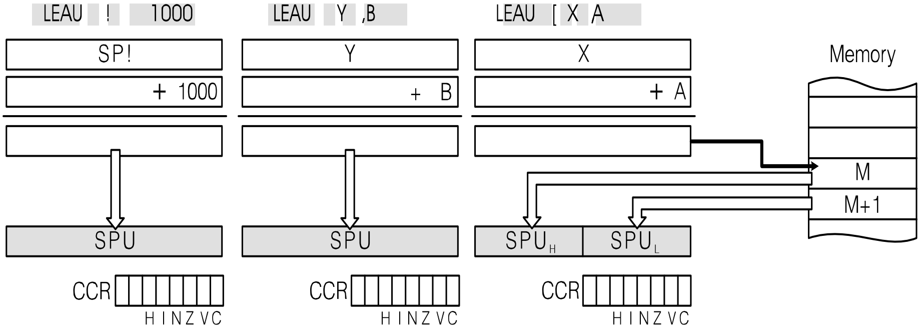
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **CP 80<br>❍ PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | 4+/2+ | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 33 | | | | | | | | | | | ● | ● | | ● | ● | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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 ⇨ SPU

`LEAU [ X ,A` (X+A:X+A+1) ⇨ SPU



| Motorola | | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | □ PG1000<br>□ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Motorola | LEAX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | LER | EA ⇨ X | |
| Short | | | |

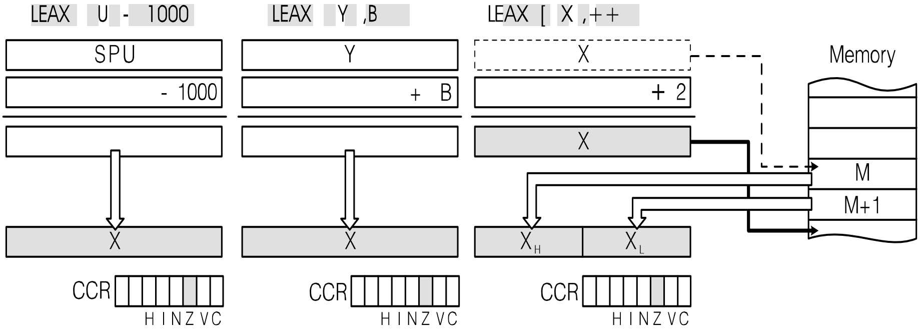
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 4+/2+ | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 30 | | | | | | | | | | | ○ | ● | | ● | ● | | | ○ | ○ | ○ | ● | ○ | ○ |

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

**Example:** **LEAX U - 1000** SPU - 1000 ⇨ X

**LEAX Y ,B** Y + B ⇨ X

**LEAX [ X ,++** (X:X+1) ⇨ X; X + 2 X



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

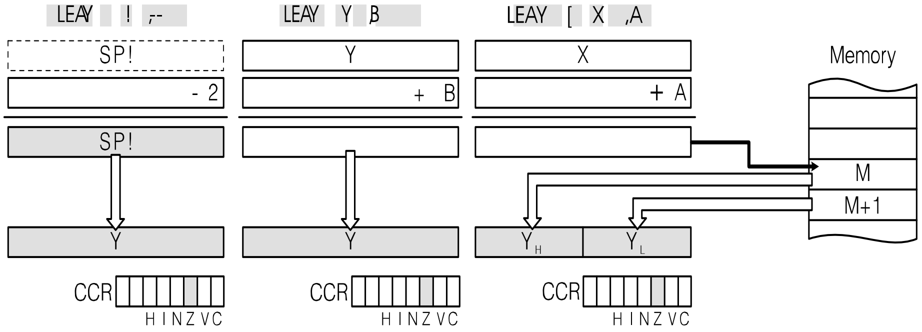
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **PG1000**<br>**PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **LEAY** | **Function** | **CPU type B / 6809** |
| **B&R** | **LEY** | **EA ⇨ Y** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | 4+/2+ | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 31 | | | | | | | | | | | ● | ● | | ● | ● | | | ❍ | ❍ | ❍ | ● | ❍ | ❍ |

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

| | | |
|---|---|---|
| **Example:** | **LEAY ! ,--** | SP! -2 ⇨ SP!; SP! ⇨ Y |
| | **LEAY Y ,B** | Y + B ⇨ Y |
| | **LEAY [ X ,A** | (X+A:X+A+1) ⇨ Y |



# 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.

| Motorola | B&R | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| STAA | = | ❍ | ❍ | ❍ | ❍ |
| STAB | =B | ❍ | ❍ | ❍ | ❍ |
| STD | =D | ❍ | ❍ | ❍ | ❍ |
| STX | =R | ❍ | ❍ | ❍ | ❍ |
| STY | =Y | | | | ❍ |
| STS | =S | ❍ | ❍ | ❍ | ❍ |

| | | | |
|---|---|---|---|
| Motorola | STAA | Function | CPU type A / 6303 |
| B&R | = | A ⇨ (M) | |
| Short | I | | |

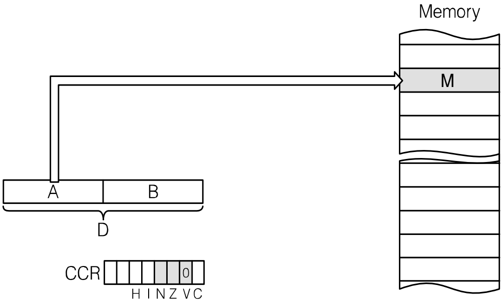
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 97 | B7 | | A7 | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| | | | |
|---|---|---|---|
| Motorola | STAA | Function | CPU type B / 6809 |
| B&R | = | A ⇨ (M) | |
| Short | I | | |

| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 97 | B7 | | A7 | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | ● | ● | ● | ● | | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



| Motorola | STAB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | =B | B ⇨ (M) | |
| Short | | | |

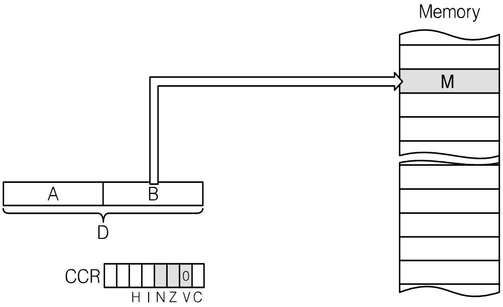
| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D7 | F7 | | E7 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | STAB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | =B | B ⇨ (M) | |
| Short | | | |

| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D7 | F7 | | E7 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | ○ | ○ | ● | ● | ▼ | ○ |

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.



| Motorola | STAD | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | =D | D ⇨ (M:M+1) | |
| Short | | | |

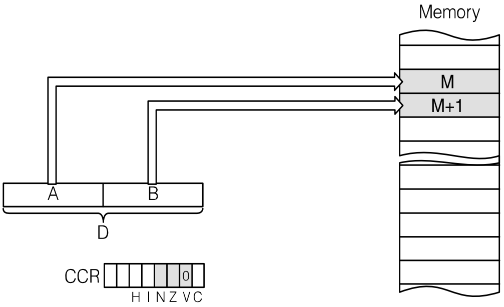
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DD | FD | | ED | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| Motorola | STAD | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | =D | D ⇨ (M:M+1) | |
| Short | | | |

| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 5/2 | 6/3 | | 5+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DD | FD | | ED | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | ● | ● | ● | ● | | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



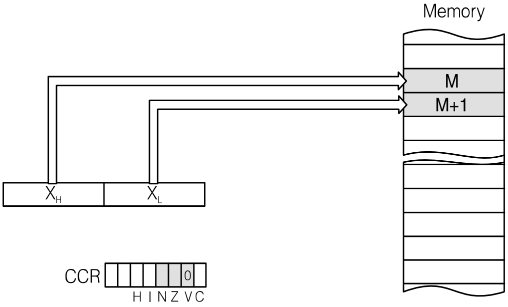
| | | | |
|---|---|---|---|
| **Motorola** | **STX** | **Function** | **CPU type A / 6303** |
| **B&R** | **=R** | **X ⇨ (M:M+1)** | |
| **Short** | | | |

| **Addressing modes / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 4/2 | 5/3 | | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DF | FF | | EF | | | | | | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| | | | |
|---|---|---|---|
| **Motorola** | **STX** | **Function** | **CPU type B / 6809** |
| **B&R** | **=R** | **X ⇨ (M:M+1)** | |
| **Short** | | | |

| **Addressing modes / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 5/2 | 6/3 | | 5+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9F | BF | | AF | | | | | | | | ○ | ○ | ○ | ● | | ● | ● | | ● | ○ | ○ | ● | ● | ▼ | ○ |

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



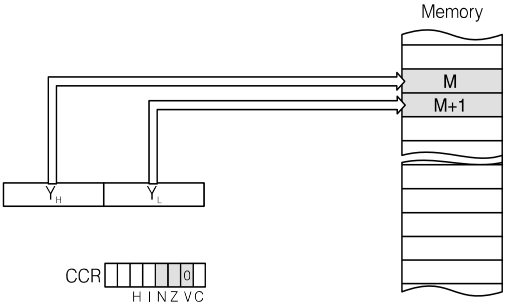
| Motorola | | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | PG1000 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | PG-PC | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Motorola | STY | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | =Y | Y ⇨ (M:M+1) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | CP 80 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ❍ | PC 80 | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/3 | 7/4 | | 6+/3+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 10 9F | 10 BF | | 10 AF | | | | | | | | ❍ | ❍ | ❍ | ● | | ● | ● | | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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



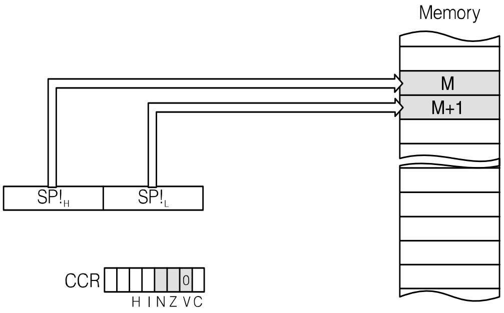
| Motorola | STS | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | =S | SP! ⇨ (M:M+1) | |
| Short | | | |

| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9F | BF | | AF | | | | | | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | STS | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | =S | SP! ⇨ (M:M+1) | |
| Short | | | |

| Addressing modes / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/3 | 7/4 | | 6+/3+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 10 DF | 10 FF | | 10 EF | | | | | | | | ○ | ○ | ○ | ● | | ● | ● | | ● | ○ | ○ | ● | ● | ▼ | ○ |

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

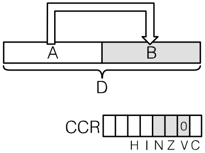
All instructions which store data (1 or 2 bytes) from a register to another or exchange data between two registers, are described in this section.

| Motorola | B&R | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| TAB | MAB | ❍ | ❍ | ❍ | ❍ |
| TBA | MBA | ❍ | ❍ | ❍ | ❍ |
| TAP | MAC | ❍ | ❍ | ❍ | ❍ |
| TPA | MCA | ❍ | ❍ | ❍ | ❍ |
| TSX | MSR | ❍ | ❍ | ❍ | ❍ |
| TXS | MRS | ❍ | ❍ | ❍ | ❍ |
| XGDX | DXR | ❍ | ❍ | ❍ | ❍ |
| TFR | TFR | | | | ❍ |
| EXG | EXG | | | | ❍ |

| Motorola | TAB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | MAB | A ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 16 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

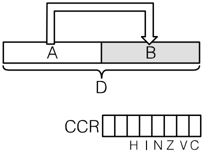
The contents of accumulator A will be copied to accumulator B.



| Motorola | TAB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | MAB | A ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F 89 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

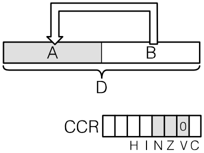
The contents of accumulator A will be copied to accumulator B.



| Motorola | TBA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | MBA | B ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 17 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

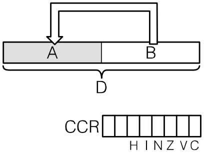
The contents of accumulator B will be copied to accumulator A.



| Motorola | TBA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | MBA | B ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F 98 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

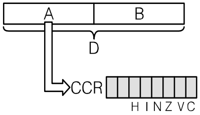
The contents of accumulator B will be copied to accumulator A.



| Motorola | TAP | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | MAC | A ⇨ CCR | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 06 | | | | | | | | | | | | | | | | | | | | | ● | ● | ● | ● | ● | ● |

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



**Note:**

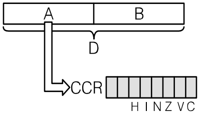
If the Interrupt Inhibit Bit I (Bit 4 of the condition code register) is set to logic 1 by the user all interrupts are locked. This technique is used if parts of the user program are not to be broken by an interrupt in any case.

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

| Motorola | TAP | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | MAC | A ⇨ CCR | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 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.**

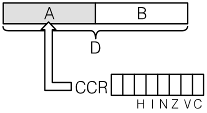
| | | | |
|---|---|---|---|
| Motorola | TPA | Function | CPU type A / 6303 |
| B&R | MCA | CCR ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 07 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

| | | | |
|---|---|---|---|
| Motorola | TPA | Function | CPU type B / 6809 |
| B&R | MCA | CCR ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F A8 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

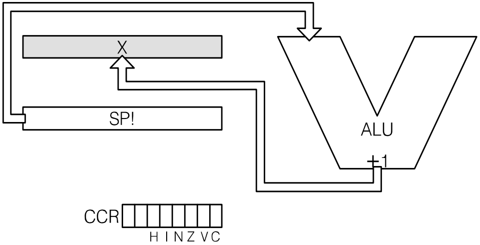
The contents of condition code register CCR will be copied to accumulator A.



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

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 30 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

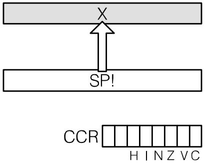
Index register X will be loaded with the address of the memory location which holds the last valid value of the stack. Since in 6303 processors the stack pointer points to the next free location in the stack a value of 1 is added to the system stack pointer SP! and the accumulator is stored in index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



| | | | |
|---|---|---|---|
| Motorola | TSX | Function | CPU type B / 6809 |
| B&R | MSR | SP! ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F 41 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

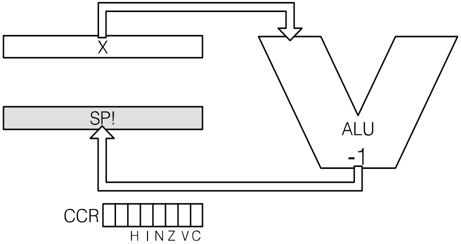
Index register X will be loaded with the address of the memory location which holds the last valid value of the stack Since in 6809 processors the stack pointer points to the last valid value in the stack no corrections which are required in the 6303 processor must be done. Therefore SP! is copied to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



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

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 35 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

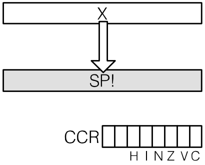
This instruction is the opposite of the TSX instruction. The contents of index register X are reduced by 1 and loaded to the system stack pointer.



| | | | |
|---|---|---|---|
| **Motorola** | **TXS** | **Function** | **CPU type B / 6809** |
| **B&R** | **MRS** | **X ⇨ SP!** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F 14 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.

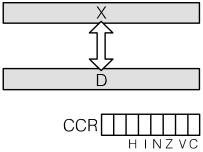
| Motorola | XGDX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | DXR | D ⟺ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 18 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

| Motorola | XGDX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | DXR | D ⟺ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 7/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1E 10 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | **PG1000**<br>**PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **TFR** | **Function** | **CPU type B / 6809** |
| **B&R** | **TFR** | $r_{Source} \Rightarrow r_{Destination}$ | |
| **Short** | | | |

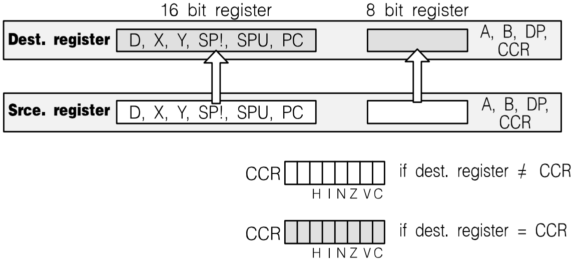
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1F | | | | | | | | | | | | | | | | | | | | | ● | ● | ● | ● | ● | ● |

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** `r,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.

| **Dest. / Srce. register** | **r** | **Postbyte**<br>Srce. | <br>Dest. |
|---|---|---|---|
| D | D | 0000 | 0000 |
| X | X | 0001 | 0001 |
| Y | Y | 0010 | 0010 |
| SPU | U | 0011 | 0011 |
| SP! | ! | 0100 | 0100 |
| PC | $ | 0101 | 0101 |
| A | A | 1000 | 1000 |
| B | B | 1001 | 1001 |
| DP | P | 1011 | 1011 |
| CCR | C | 1010 | 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.

| Motorola | | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Motorola | EXG | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | EXG | $r_1 \Leftrightarrow r_2$ | |
| Short | | | |

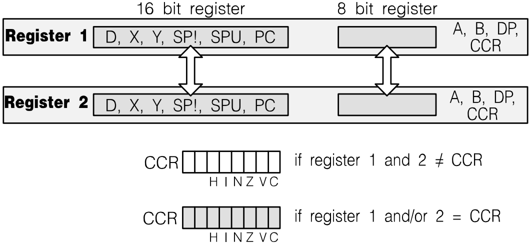
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 7/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 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
Register 1

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.

| Register 1 / 2 | r | Postbyte<br>Register 1 | Postbyte<br>Register 2 |
|---|---|---|---|
| D | D | 0000 | 0000 |
| X | X | 0001 | 0001 |
| Y | Y | 0010 | 0010 |
| SPU | U | 0011 | 0011 |
| SP! | ! | 0100 | 0100 |
| PC | $ | 0101 | 0101 |
| A | A | 1000 | 1000 |
| B | B | 1001 | 1001 |
| DP | P | 1011 | 1011 |
| CCR | C | 1010 | 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 | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| PSH | PSH | | | | ❍ |
| PUL | PUL | | | | ❍ |
| PSHA | ANS | ❍ | ❍ | ❍ | ❍ |
| PULA | AVS | ❍ | ❍ | ❍ | ❍ |
| PSHB | BNS | ❍ | ❍ | ❍ | ❍ |
| PULB | BVS | ❍ | ❍ | ❍ | ❍ |
| PSHX | RNS | ❍ | ❍ | ❍ | ❍ |
| PULX | RVS | ❍ | ❍ | ❍ | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000** | | | | |
| | | | | | | | | | | | | | | | | | | | | | | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **PSH** | **Function** | **CPU type B / 6809** |
| **B&R** | **PSH** | Saves contents of the indicated register to the system or user stack. | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **CP 80** | | | | |
| | | | | | | | | | | | | | | | | | | | | | ○ | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 5+/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 34 / 36 | | | | | | | | | | | | | | | | | ● | ● | | | ○ | ○ | ○ | ○ | ○ | ○ |

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

s — 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.

**s = ! (Opcode = $34)**
**Register order on a SP!**

| Register | r | Postbyte |
|---|---|---|
| CCR | **C** | xxxxxxx1 |
| A | **A** | xxxxxx1x |
| B | **B** | xxxxx1xx |
| DP | **P** | xxxx1xxx |
| X | **X** | xxx1xxxx |
| Y | **Y** | xx1xxxxx |
| SPU | **U** | x1xxxxxx |
| PC | **$** | 1xxxxxxx |
| All except PC | ***** | 01111111 |

**s = U (Opcode = $36)**
**Register order on a SPU**

| Register | r | Postbyte |
|---|---|---|
| CCR | **C** | xxxxxxx1 |
| A | **A** | xxxxxx1x |
| B | **B** | xxxxx1xx |
| DP | **P** | xxxx1xxx |
| X | **X** | xxx1xxxx |
| Y | **Y** | xx1xxxxx |
| SP! | **!** | x1xxxxxx |
| PC | **$** | 1xxxxxxx |
| All except PC | ***** | 01111111 |

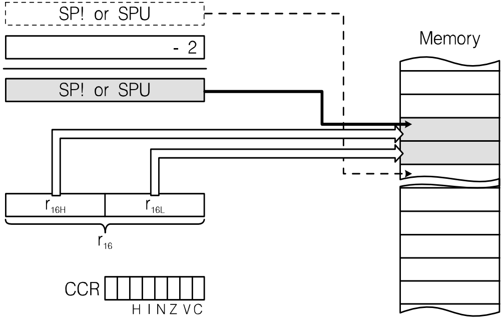
Different registers can be combined in the **postbyte**.

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

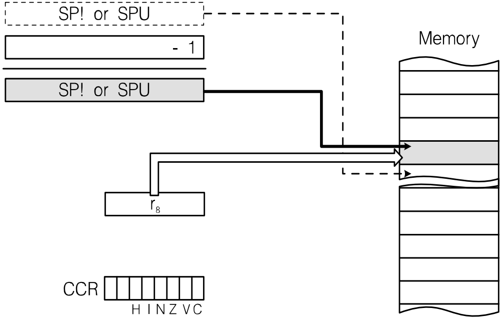
| **An entry of...** | **is changed by the PROgramming SYStem to...** |
|---|---|
| `PSH ! ,*` | `PSH ! ,CABPXYU` |
| `PSH U ,*` | `PSH U ,CABPXY!` |
| `PSH U ,XBC` | `PSH U ,CBX` |

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

For 16 bit registers X, Y, SPU/SP!, PC



For 8 bit registers CCR, A, B, DP



| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000**<br>**PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| **Motorola** | **PUL** | **Function** | **CPU type B / 6809** |
| **B&R** | **PUL** | Load indicated register with data from the system or user stack. | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | <br>○ | **CP 80**<br>**PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 5+/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 35 / 37 | | | | | | | | | | | | | | | | | ● | ● | | | ● | ● | ● | ● | ● | ● |

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

`s` —— 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.

**s = ! (Opcode = $35)**
**Load register from SP!**

| **Register** | **r** | **Postbyte** |
|---|---|---|
| CCR | C | xxxxxxx1 |
| A | A | xxxxxx1x |
| B | B | xxxxx1xx |
| DP | P | xxxx1xxx |
| X | X | xxx1xxxx |
| Y | Y | xx1xxxxx |
| SPU | U | x1xxxxxx |
| PC | $ | 1xxxxxxx |
| All except PC | * | 01111111 |

**s = U (Opcode = $37)**
**Load register from SPU**

| **Register** | **r** | **Postbyte** |
|---|---|---|
| CCR | C | xxxxxxx1 |
| A | A | xxxxxx1x |
| B | B | xxxxx1xx |
| DP | P | xxxx1xxx |
| X | X | xxx1xxxx |
| Y | Y | xx1xxxxx |
| SP! | ! | x1xxxxxx |
| PC | $ | 1xxxxxxx |
| All except PC | * | 01111111 |

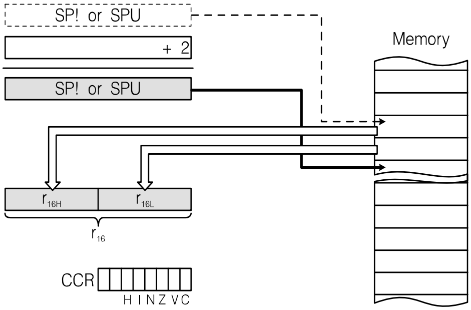
Different registers can be combined in the **postbyte.**

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

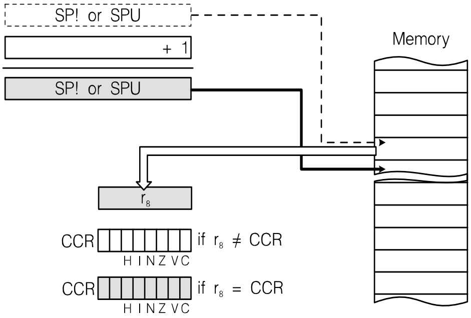
| **An entry of...** | **is changed by the PROgramming SYStem to...** |
|---|---|
| `PUL ! ,*` | `PSH ! ,CABPXYU` |
| `PUL U ,*` | `PSH U ,CABPXY!` |
| `PUL U ,XBC` | `PSH U ,CBX` |

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

For 16 bit registers X, Y, SPU/SP!, PC



For 8 bit registers CCR, A, B, DP

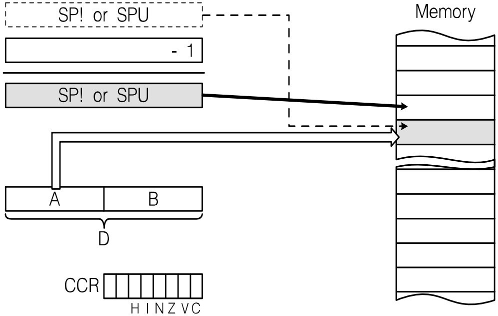


| | | | |
|---|---|---|---|
| Motorola | PSHA | Function | CPU type A / 6303 |
| B&R | ANS | A ⇨ (SP!); SP! - 1 ⇨ SP! | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 4/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 36 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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.

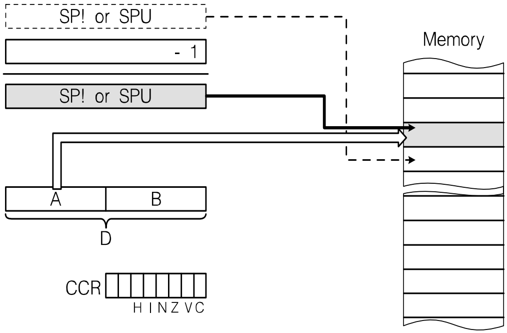


| | | | |
|---|---|---|---|
| Motorola | PSHA | Function | CPU type B / 6809 |
| B&R | ANS | SP! - 1 ⇨ SP!; A ⇨ (SP!) | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 34 02 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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.

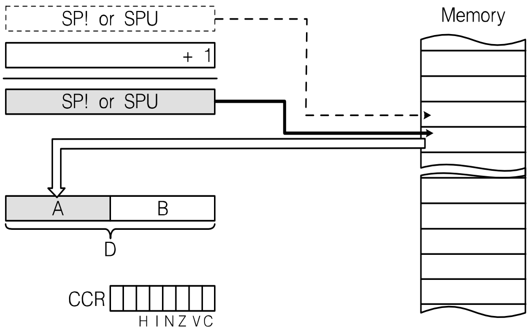


| Motorola | PULA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | AVS | SP! + 1 ⇨ SP!; (SP!) ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 32 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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!.

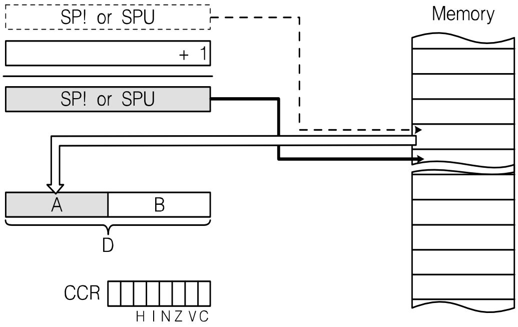


| Motorola | PULA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | AVS | (SP!) ⇨ A; SP! + 1 ⇨ SP! | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 35 02 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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.

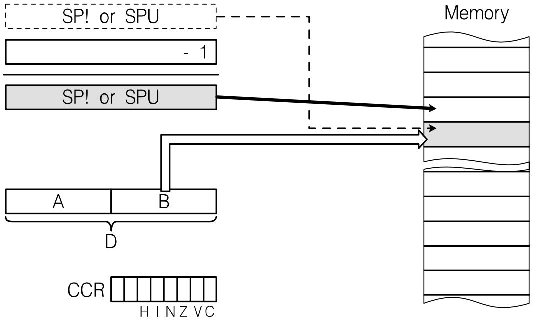


| | | | |
|---|---|---|---|
| Motorola | PSHB | Function | CPU type A / 6303 |
| B&R | BNS | B ⇨ (SP!); SP! - 1 ⇨ SP! | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 4/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 37 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.

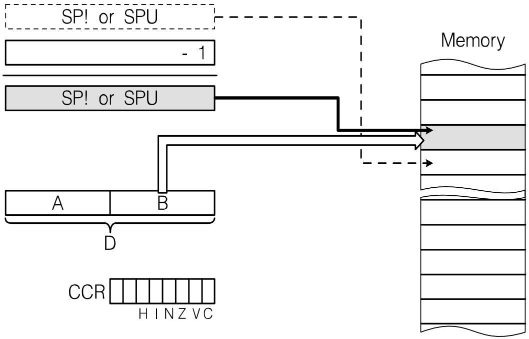


| | | | |
|---|---|---|---|
| Motorola | PSHB | Function | CPU type B / 6809 |
| B&R | BNS | SP! - 1 ⇨ SP!; B ⇨ (SP!) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 34 04 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.

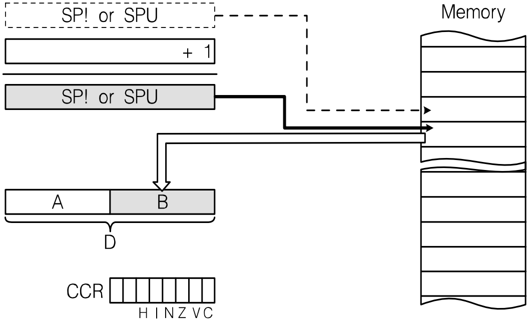


| Motorola | PULB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | BVS | SP! + 1 ⇨ SP!; (SP!) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 33 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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!.

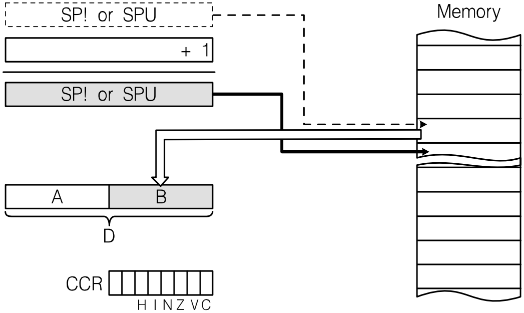


| Motorola | PULB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | BVS | (SP!) ⇨ B; SP! + 1 ⇨ SP! | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 6/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 35 04 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.



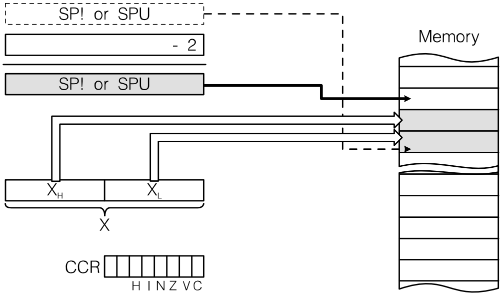
| Motorola | PSHX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RNS | $X_L$ ⇨ (SP!); SP! - 1 ⇨ SP! | |
| Short | | $X_H$ ⇨ (SP!); SP! - 1 ⇨ SP! | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 5/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 3C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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) $X_L$ ⇨ (SP!)
2) SP! - 1 ⇨ SP!
3) $X_H$ ⇨ (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.



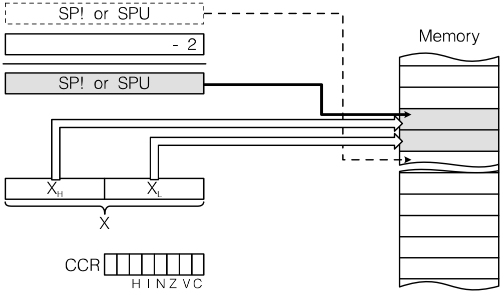
| Motorola | PSHX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RNS | SP! - 1 ⇨ SP!; $X_L$ ⇨ (SP!) | |
| Short | | SP! - 1 ⇨ SP!; $X_H$ ⇨ (SP!) | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 7/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 34 10 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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) $X_L$ ⇨ (SP!)
3) SP! - 1 ⇨ SP!
4) $X_H$ ⇨ (SP!)

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



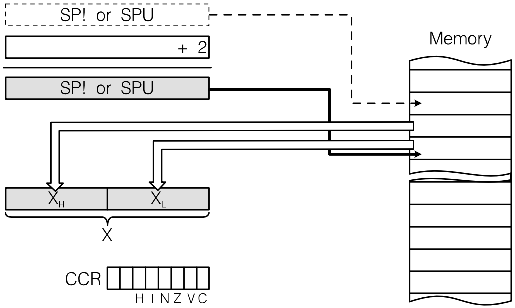
| Motorola | PULX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RVS | SP! + 1 ⇨ SP!; (SP!) ⇨ $X_H$ | |
| Short | | SP! + 1 ⇨ SP!; (SP!) ⇨ $X_L$ | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 4/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 38 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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:

1) SP! + 1 ⇨ SP!
2) (SP!) ⇨ $X_H$
3) SP! + 1 ⇨ SP!
4) (SP!) ⇨ $X_L$

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



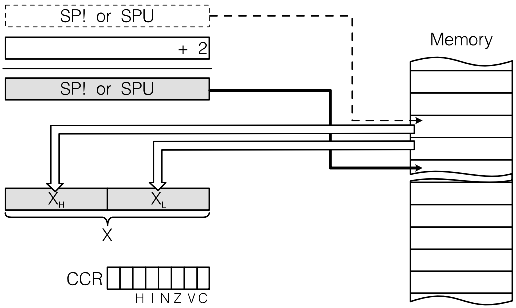
| Motorola | PULX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RVS | (SP!) ⇨ $X_H$; SP! + 1 ⇨ SP! | |
| Short | | (SP!) ⇨ $X_L$; SP! + 1 ⇨ SP! | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 7/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 35 10 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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:

1) (SP!) ⇨ $X_H$
2) SP! + 1 ⇨ SP!
3) (SP!) ⇨ $X_L$
4) SP! + 1 ⇨ SP!

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 | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| ANDA | UND | ❍ | ❍ | ❍ | ❍ |
| ANDB | UB | ❍ | ❍ | ❍ | ❍ |
| AIM | AIM | | ❍ | | |
| ORAA | OD | ❍ | ❍ | ❍ | ❍ |
| ORAB | OB | ❍ | ❍ | ❍ | ❍ |
| OIM | OIM | | ❍ | | |
| EORA | EXO | ❍ | ❍ | ❍ | ❍ |
| EORB | EB | ❍ | ❍ | ❍ | ❍ |
| EIM | EIM | | ❍ | | |

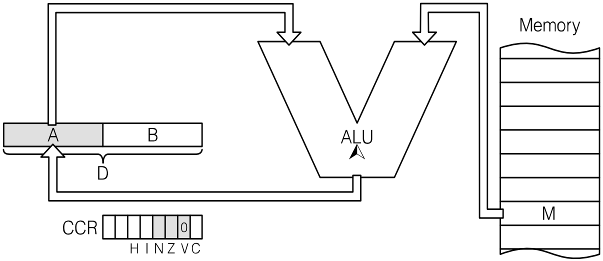
| Motorola | ANDA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | UND | A ∧ (M) ⇨ A | |
| Short | U | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 94 | B4 | 84 | A4 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | ANDA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | UND | A ∧ (M) ⇨ A | |
| Short | U | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 94 | B4 | 84 | A4 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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.



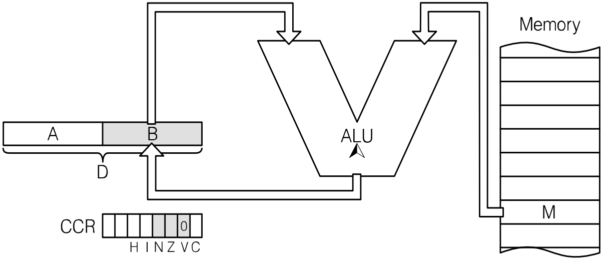
| Motorola | ANDB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | UB | B ∧ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D4 | F4 | C4 | E4 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| Motorola | ANDB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | UB | B ∧ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D4 | F4 | C4 | E4 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



| Motorola | AIM | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | AIM | (X + Offset) ∧ IMM ⇨ (X + Offset) | |
| Short | | | |

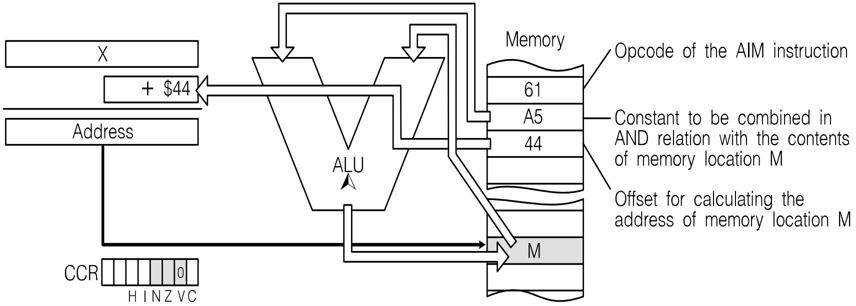
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 7/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 61 | | | | | | | ○ | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | CP 80<br>PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

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.

```
Syntax:     AIM    # $xxyy      xx => 1 byte constant
                                yy => 1 byte offset

Example:    AIM    # $A544
```



X
+ $44
Address
ALU
CCR
0
H I N Z V C
Memory
61
A5
44
M
Opcode of the AIM instruction
Constant to be combined in AND relation with the contents of memory location M
Offset for calculating the address of memory location M

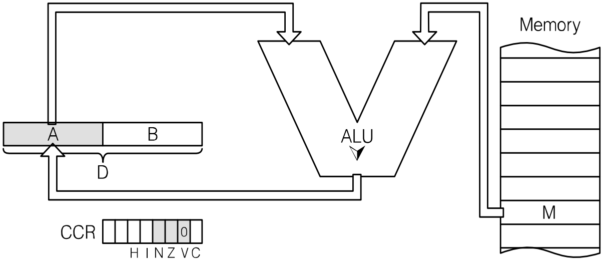
| Motorola | ORAA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | OD | A ∨ (M) ⇨ A | |
| Short | O | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9A | BA | 8A | AA | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| Motorola | ORAA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | OD | A ∨ (M) ⇨ A | |
| Short | O | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9A | BA | 8A | AA | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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



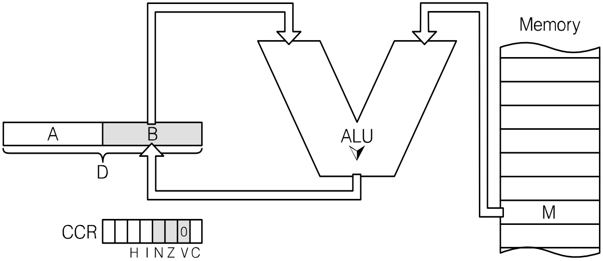
| Motorola | ORAB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | OB | B ∨ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DA | FA | CA | EA | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | ORAB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | OB | B ∨ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DA | FA | CA | EA | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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



| Motorola | OIM | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | OIM | (X + Offset) ∨ IMM ⇨ (X + Offset) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | PG1000 | | | | |
| | | | | | | | | | | | | | | | | | | | | | ○ | PG-PC | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 7/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 62 | | | | | | | ○ | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

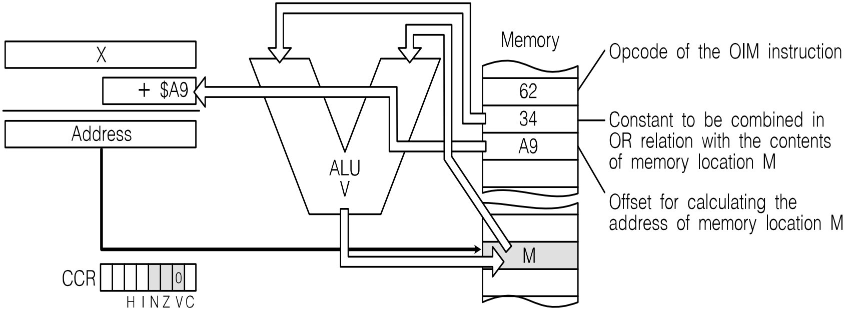
| Motorola | | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | CP 80 | | | | |
| | | | | | | | | | | | | | | | | | | | | | | PC 80 | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

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.

**Syntax:** `OIM # $xxyy` xx => 1 byte constant
yy => 1 byte offset

**Example:** `OIM # $34A9`


X
+ $A9
Address
ALU
V
CCR
0
H I N Z V C
Memory
62
34
A9
M
Opcode of the OIM instruction
Constant to be combined in OR relation with the contents of memory location M
Offset for calculating the address of memory location M

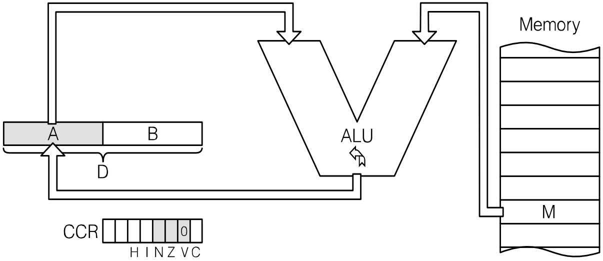
| Motorola | EORA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | EXO | A ⊕ (M) ⇨ A | |
| Short | E | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 98 | B8 | 88 | A8 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| Motorola | EORA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | EXO | A ⊕ (M) ⇨ A | |
| Short | E | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 98 | B8 | 88 | A8 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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



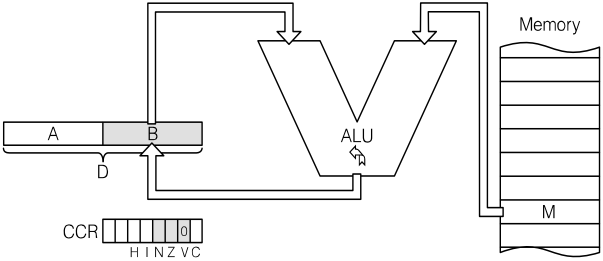
| Motorola | EORB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | EB | B ⊕ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D8 | F8 | C8 | E8 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | EORB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | EB | B ⊕ (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D8 | F8 | C8 | E8 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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



| Motorola | EIM | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | EIM | (X + Offset) ⊕ IMM ⇨ (X + Offset) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | PG1000 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ○ | PG-PC | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 7/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 65 | | | | | | | ○ | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

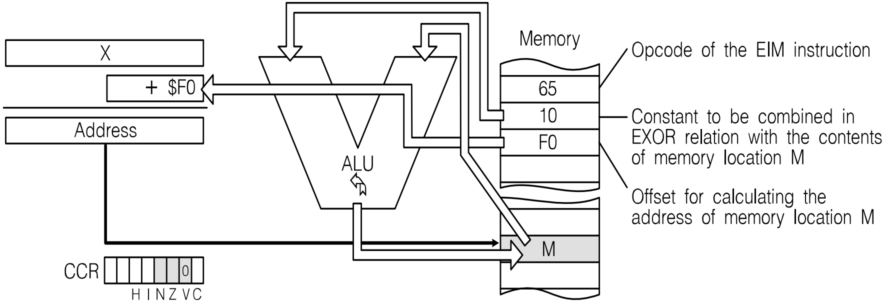
| Motorola | | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | | CP 80 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | PC 80 | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

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.

**Syntax:** `EIM # $xxyy` xx => 1 byte constant
yy => 1 byte offset

**Example:** `EIM # $10F0`


X
+ $F0
Address
ALU
CCR
0
H I N Z V C
Memory
65
10
F0
M
Opcode of the EIM instruction
Constant to be combined in EXOR relation with the contents of memory location M
Offset for calculating the address of memory location M

# 3.6. ARITHMETIC INSTRUCTIONS

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

| Motorola | B&R | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| ADDA | + | ❍ | ❍ | ❍ | ❍ |
| ADCA | ADD | ❍ | ❍ | ❍ | ❍ |
| ADDB | +B | ❍ | ❍ | ❍ | ❍ |
| ADCB | ++B | ❍ | ❍ | ❍ | ❍ |
| ADDD | +D | ❍ | ❍ | ❍ | ❍ |
| ABA | A+B | ❍ | ❍ | ❍ | ❍ |
| SUBA | - | ❍ | ❍ | ❍ | ❍ |
| SBCA | SUB | ❍ | ❍ | ❍ | ❍ |
| SUBB | -B | ❍ | ❍ | ❍ | ❍ |
| SBCB | - -B | ❍ | ❍ | ❍ | ❍ |
| SUBD | -D | ❍ | ❍ | ❍ | ❍ |
| SBA | A-B | ❍ | ❍ | ❍ | ❍ |
| ABX | B+R | ❍ | ❍ | ❍ | ❍ |
| MUL | A*B | ❍ | ❍ | ❍ | ❍ |

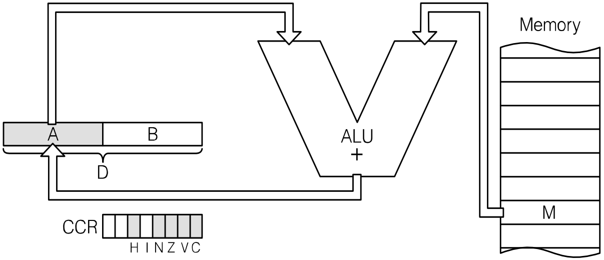
| Motorola | ADDA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | + | A + (M) ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9B | BB | 8B | AB | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ● | ○ | ● | ● | ● | ● |

| Motorola | ADDA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | + | A + (M) ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9B | BB | 8B | AB | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● |

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



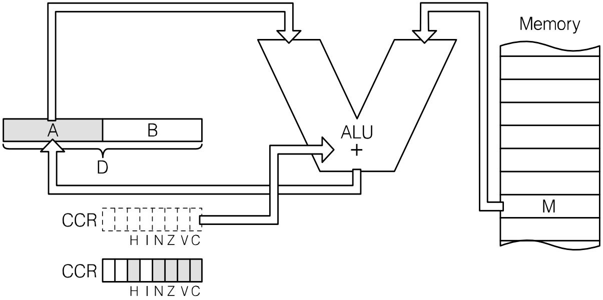
| Motorola | ADCA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | ADD | A + (M) + C ⇨ A | |
| Short | A, ++ | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 99 | B9 | 89 | A9 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ● | ○ | ● | ● | ● | ● |

| Motorola | ADCA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | ADD | A + (M) + C ⇨ A | |
| Short | A, ++ | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 99 | B9 | 89 | A9 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● |

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.



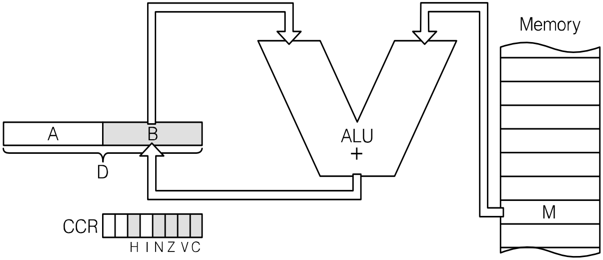
| Motorola | ADDB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | +B | B + (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DB | FB | CB | EB | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ● | ○ | ● | ● | ● | ● |

| Motorola | ADDB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | +B | B + (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | DB | FB | CB | EB | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● |

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



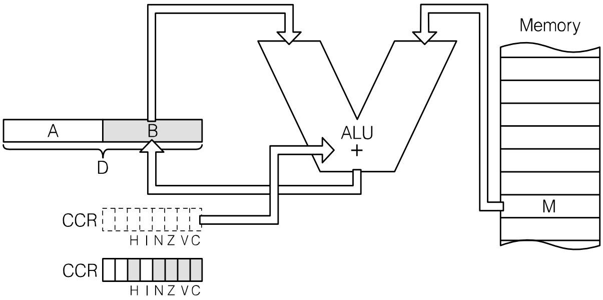
| Motorola | ADCB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | ++B | B + (M) + C ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D9 | F9 | C9 | E9 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ● | ❍ | ● | ● | ● | ● |

| Motorola | ADCB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | ++B | B + (M) + C ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D9 | F9 | C9 | E9 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ● | ❍ | ● | ● | ● | ● |

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.



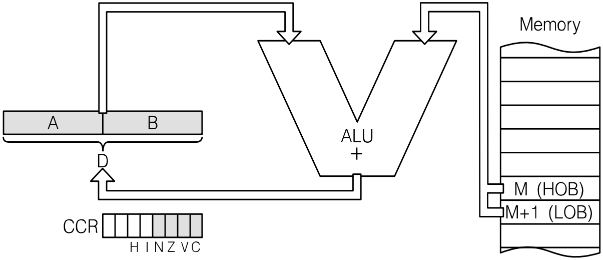
| Motorola | ADDD | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | +D | D + (M:M+1) ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 3/3 | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D3 | F3 | C3 | E3 | | | | | | | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | ADDD | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | +D | D + (M:M+1) ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | 4/3 | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D3 | F3 | C3 | E3 | | | | | | | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ● | ● |

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



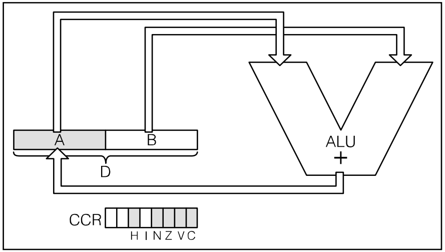
|  |  |  |  |
|---|---|---|---|
| **Motorola** | **ABA** | **Function** | **CPU type A / 6303** |
| **B&R** | **A+B** | **A + B ⇨ A** |  |
| **Short** |  |  |  |

| **Addressing mode / Opcode** |  |  |  |  |  | **Address preselection** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ **PG1000**<br>❍ **PG-PC** |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** |  |  |  |  |  |
| 1/1 |  |  |  |  |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ● | ❍ | ● | ● | ● | ● |

|  |  |  |  |
|---|---|---|---|
| **Motorola** | **ABA** | **Function** | **CPU type B / 6809** |
| **B&R** | **A+B** | **A + B ⇨ A** |  |
| **Short** |  |  |  |

| **Addressing mode / Opcode** |  |  |  |  |  | **Address preselection** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ **CP 80**<br>❍ **PC 80** |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** |  |  |  |  |  |
|  |  | 25/3 |  |  |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
|  |  | BD FF 90 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ● | ❍ | ● | ● | ● | ● |

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:<br>SP! - 1 ⇨ SP!<br>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):<br>A + (SP!) ⇨ A<br>SP! + 1 ⇨ SP! |
| **RTS** |  | Leaves the sub-program. |

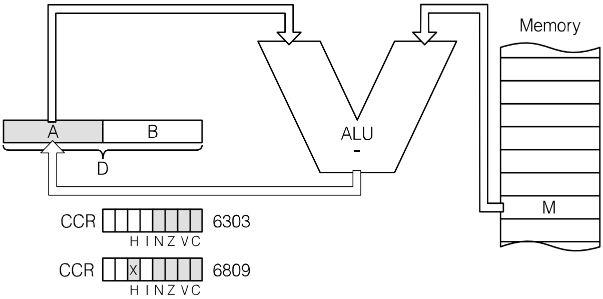
| Motorola | SUBA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | - | A - (M) ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 90 | B0 | 80 | A0 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | SUBA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | - | A - (M) ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 90 | B0 | 80 | A0 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | x | ○ | ● | ● | ● | ● |

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



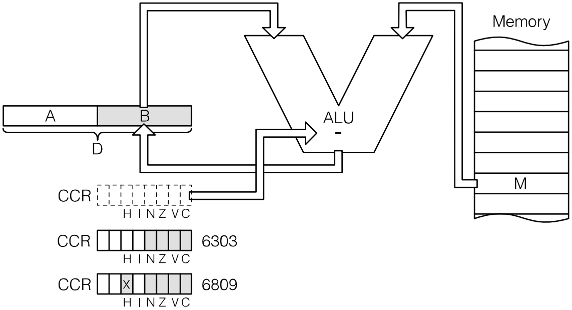
| Motorola | SBCA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SUB | A - (M) - C ⇨ A | |
| Short | - - | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 92 | B2 | 82 | A2 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ● | ● |

| Motorola | SBCA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SUB | A - (M) - C ⇨ A | |
| Short | - - | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 92 | B2 | 82 | A2 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | x | ❍ | ● | ● | ● | ● |

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



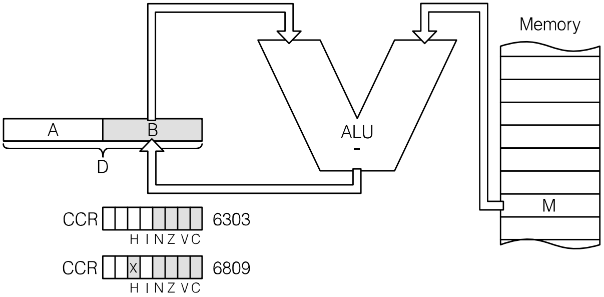
| | | | |
|---|---|---|---|
| Motorola | SUBB | Function | CPU type A / 6303 |
| B&R | -B | B - (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D0 | F0 | C0 | E0 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| Motorola | SUBB | Function | CPU type B / 6809 |
| B&R | -B | B - (M) ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D0 | F0 | C0 | E0 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | x | ○ | ● | ● | ● | ● |

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



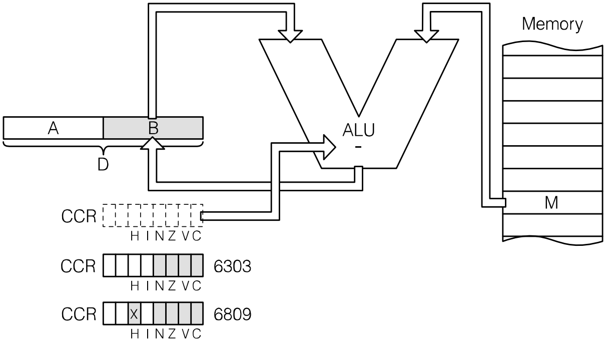
| Motorola | SBCB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | - -B | B - (M) - C ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D2 | F2 | C2 | E2 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | SBCB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | - -B | B - (M) - C ⇨ B | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D2 | F2 | C2 | E2 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | X | ○ | ● | ● | ● | ● |

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



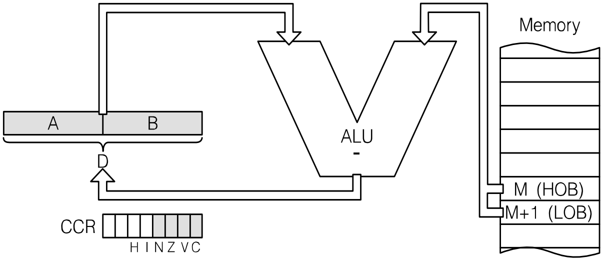
| Motorola | SUBD | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | -D | D - (M:M+1) ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 3/3 | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 93 | B3 | 83 | A3 | | | | | | | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | SUBD | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | -D | D - (M:M+1) ⇨ D | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | 4/3 | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 93 | B3 | 83 | A3 | | | | | | | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ● | ● |

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



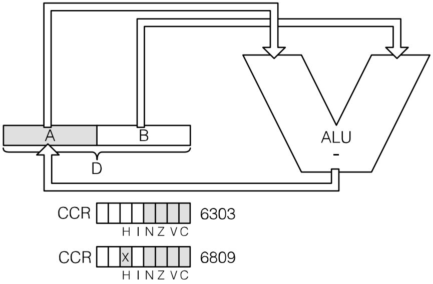
| Motorola | SBA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | A-B | A - B ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 10 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | SBA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | A-B | A - B ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 25/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | BD FF 9A | | | | | | | | | | | | | | | | | | | x | ○ | ● | ● | ● | ● |

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:<br>SP! - 1 ⇨ SP!<br>B ⇨ (SP!) |
| **SUBA** | **! ,+** | 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):<br>A - (SP!) ⇨ A<br>SP! + 1 ⇨ SP! |
| **RTS** | | Leaves the sub-program. |

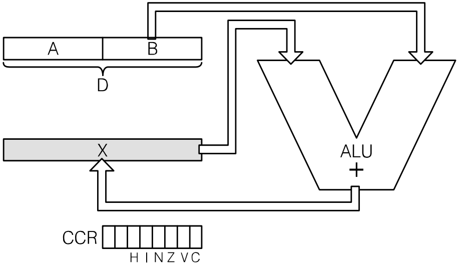
| Motorola | ABX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | B+R | B + X ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 3A | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

| Motorola | ABX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | B+R | B + X ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 3A | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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



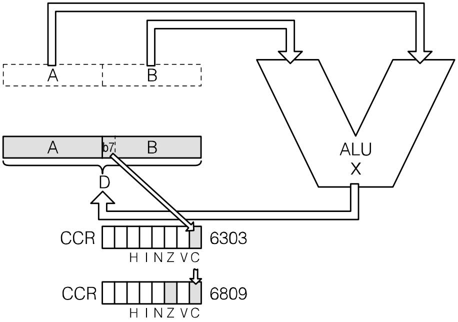
| | | | |
|---|---|---|---|
| **Motorola** | **MUL** | **Function** | **CPU type A / 6303** |
| **B&R** | **A*B** | **A ✕ B ⇨ D** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 7/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 3D | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ● |

| | | | |
|---|---|---|---|
| **Motorola** | **MUL** | **Function** | **CPU type B / 6809** |
| **B&R** | **A*B** | **A ✕ B ⇨ D** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 11/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 3D | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ● | ❍ | ● |

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 | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| CBA | AVB | ❍ | ❍ | ❍ | ❍ |
| CMPA | CMP | ❍ | ❍ | ❍ | ❍ |
| CMPB | VB | ❍ | ❍ | ❍ | ❍ |
| CPX | VR | ❍ | ❍ | ❍ | ❍ |
| CPX# | VRK | ❍ | ❍ | ❍ | ❍ |
| CPY | VY | | | | ❍ |
| CPY# | VYK | | | | ❍ |
| BITA | B | ❍ | ❍ | ❍ | ❍ |
| BITB | BB | ❍ | ❍ | ❍ | ❍ |
| TIM | TIM | | ❍ | | |

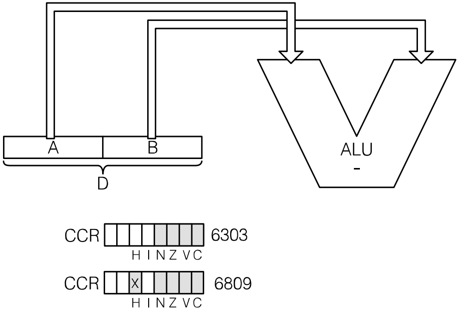
| | | | |
|---|---|---|---|
| **Motorola** | **CBA** | **Function** | **CPU type A / 6303** |
| **B&R** | **AVB** | **A - B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 11 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Motorola** | **CBA** | **Function** | **CPU type B / 6809** |
| **B&R** | **AVB** | **A - B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | 25/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | BD FF 95 | | | | | | | | | | | | | | | | | | | × | ○ | ● | ● | ● | ● |

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.



**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:

| | | | |
|---|---|---|---|
| `PSH` | `!` | `,B` | The contents of accumulator B are placed on the stack:<br>SP! - 1 ⇨ SP!<br>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):<br>A - (SP!)<br>SP! + 1 ⇨ SP! |
| `RTS` | | | Leaves the sub-program. |

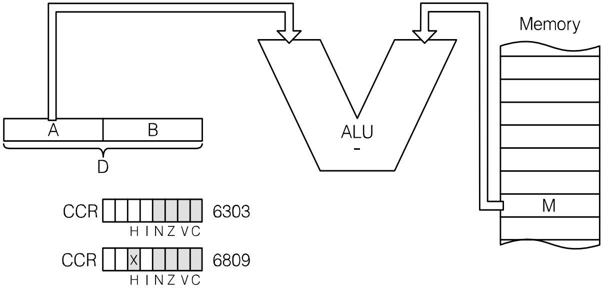
| Motorola | CMPA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | CMP | A - (M) | |
| Short | V | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 91 | B1 | 81 | A1 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | CMPA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | CMP | A - (M) | |
| Short | V | | |

| Addressing mode/ Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 91 | B1 | 81 | A1 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | x | ○ | ● | ● | ● | ● |

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.



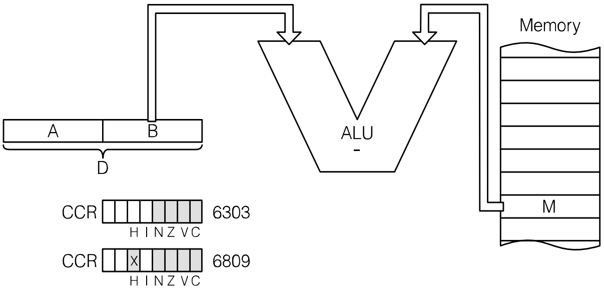
| | | | |
|---|---|---|---|
| Motorola | CMPB | Function | CPU type A / 6303 |
| B&R | VB | B - (M) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D1 | F1 | C1 | E1 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| Motorola | CMPB | Function | CPU type B / 6809 |
| B&R | VB | B - (M) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D1 | F1 | C1 | E1 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | 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.



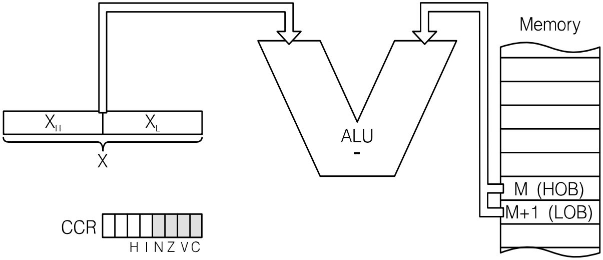
| Motorola | CPX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | VR | X - (M:M+1) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 3/3 | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9C | BC | 8C | AC | | | | | | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | CPX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | VR | X - (M:M+1) | |
| Short | | | |

| Addressing mode/ Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | 4/3 | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9C | BC | 8C | AC | | | | | | | | ○ | ○ | ○ | ● | | ● | ● | ● | ● | ○ | ○ | ● | ● | ● | ● |

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.



| Motorola | CPX# | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | VRK | X - M | |
| Short | | | |

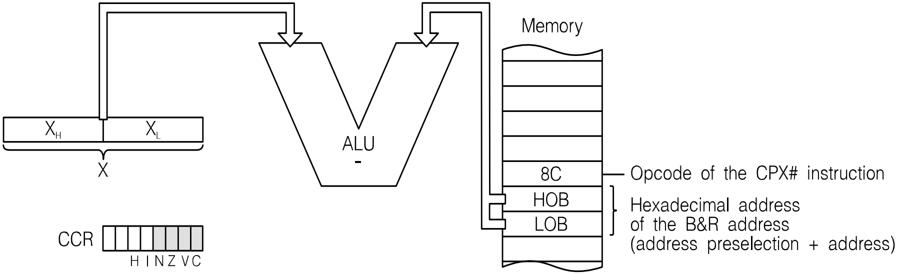
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 3/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 8C | | | ○ | ○ | ○ | ○ | ○ | | ○ | ○ | | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | CPX# | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | VRK | X - M | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 4/3 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 8C | | | ○ | ○ | ○ | ○ | ○ | | ○ | ○ | | | | | | ● | ● | ○ | ○ | ● | ● | ● | ● |

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.



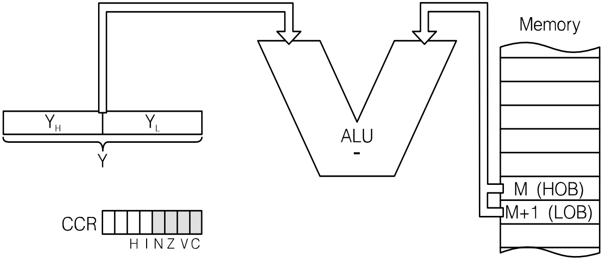
|  |  |  |  |
|---|---|---|---|
| **Motorola** |  | **Function** | **CPU type A / 6303** |
| **B&R** |  |  |  |
| **Short** |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** |  |  |  |  |  | **Address preselection** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **PG1000**<br>**PG-PC** |  |  |  |  |  |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** |  |  |  |  |  |  |
|  |  |  |  |  |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |
|---|---|---|---|
| **Motorola** | **CPY** | **Function** | **CPU type B / 6809** |
| **B&R** | **VY** | **Y - (M:M+1)** |  |
| **Short** |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** |  |  |  |  |  | **Address preselection** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **CP 80**<br>❍ **PC 80** |  |  |  |  |  |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** |  |  |  |  |  |
|  | 7/3 | 8/4 | 5/4 | 7+/3+ |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
|  | 10 9C | 10 BC | 10 8C | 10 AC |  |  |  |  |  |  |  | ● | ● | ● | ● |  | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ● | ● |

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.



| Motorola | | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

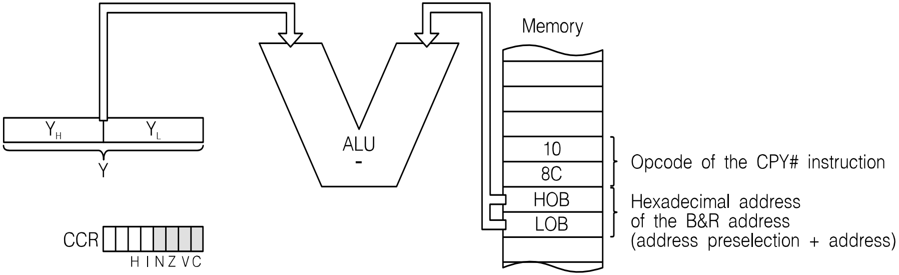
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | PG1000<br>PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Motorola | CPY# | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | VYK | Y - M | |
| Short | | | |

| Addressing mode/ Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | 5/4 | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 10 8C | | | ● | ● | ● | ● | ● | | ● | ● | | | | | | ● | ● | ❍ | ❍ | ● | ● | ● | ● |

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.



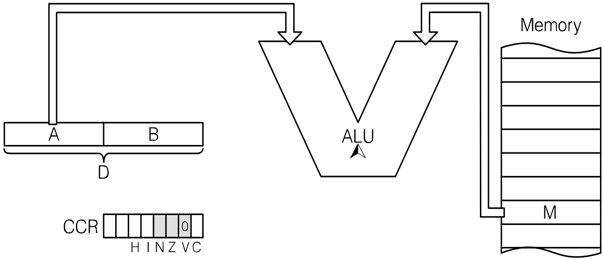
| Motorola | BITA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | B | A ∧ (M) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 95 | B5 | 85 | A5 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | BITA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | B | A ∧ (M) | |
| Short | | | |

| Addressing mode/ Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 95 | B5 | 85 | A5 | | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ○ | ○ | ● | ● | ▼ | ○ |

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.



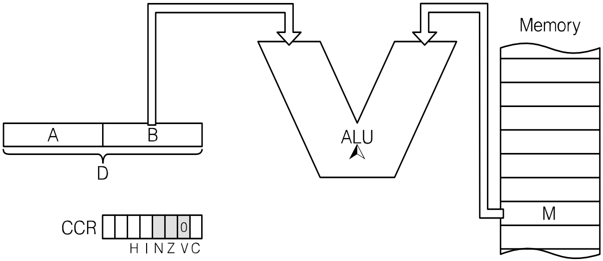
| | | | |
|---|---|---|---|
| **Motorola** | **BITB** | **Function** | **CPU type A / 6303** |
| **B&R** | **BB** | **B ∧ (M)** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 3/2 | 4/3 | 2/2 | 4/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D5 | F5 | C5 | E5 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | **BITB** | **Function** | **CPU type B / 6809** |
| **B&R** | **BB** | **B ∧ (M)** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 4/2 | 5/3 | 2/2 | 4+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | D5 | F5 | C5 | E5 | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ● | ● | ● | ● | ● | ❍ | ❍ | ● | ● | ▼ | ❍ |

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.



| Motorola | TIM | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | TIM | (X + Offset) ∧ IMM | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | 5/3 | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 6B | | | | | | | | ❍ | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

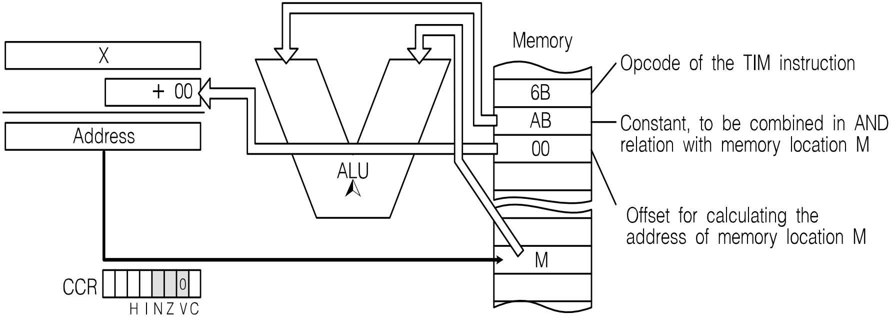
| Motorola | | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | | | |
| Short | | | |

| Addressing mode/ Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | CP 80<br>PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

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`


X
+ 00
Address
ALU
CCR
0
H I N Z V C
Memory
6B
AB
00
M
Opcode of the TIM instruction
Constant, to be combined in AND relation with memory location M
Offset for calculating the address of memory location M

# 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.

| Motorola | B&R | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| INC | INC | ❍ | ❍ | ❍ | ❍ |
| INCA | IA | ❍ | ❍ | ❍ | ❍ |
| INCB | IB | ❍ | ❍ | ❍ | ❍ |
| INX | IR | ❍ | ❍ | ❍ | ❍ |
| INS | IS | ❍ | ❍ | ❍ | ❍ |
| DEC | DEC | ❍ | ❍ | ❍ | ❍ |
| DECA | DA | ❍ | ❍ | ❍ | ❍ |
| DECB | DB | ❍ | ❍ | ❍ | ❍ |
| DEX | DR | ❍ | ❍ | ❍ | ❍ |
| DES | DS | ❍ | ❍ | ❍ | ❍ |

| | | | |
|---|---|---|---|
| **Motorola** | **INC** | **Function** | **CPU type A / 6303** |
| **B&R** | **INC** | **(M) + 1 ⇨ (M)** | |
| **Short** | | | |

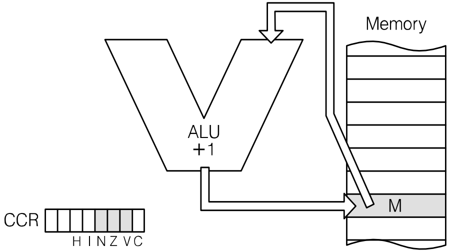
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | 6/3 | | 6/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 7C | | 6C | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ○ |

| | | | |
|---|---|---|---|
| **Motorola** | **INC** | **Function** | **CPU type B / 6809** |
| **B&R** | **INC** | **(M) + 1 ⇨ (M)** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | 6/2 | 7/3 | | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 0C | 7C | | 6C | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | ○ | ○ | ● | ● | ● | ○ |

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.



| Motorola | INCA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | IA | A + 1 ⇨ A | |
| Short | | | |

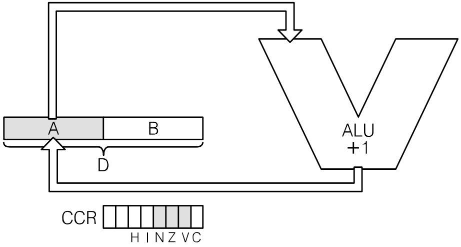
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

| Motorola | INCA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | IA | A + 1 ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

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.



| | | | |
|---|---|---|---|
| **Motorola** | **INCB** | **Function** | **CPU type A / 6303** |
| **B&R** | **IB** | **B + 1 ⇨ B** | |
| **Short** | | | |

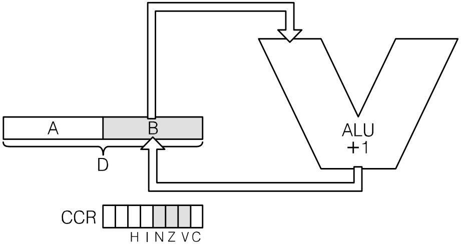
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ | **PG1000** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ○ | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

| | | | |
|---|---|---|---|
| **Motorola** | **INCB** | **Function** | **CPU type B / 6809** |
| **B&R** | **IB** | **B + 1 ⇨ B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ | **CP 80** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ○ | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

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.



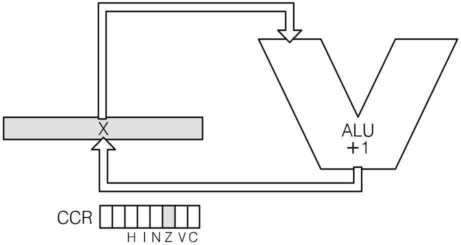
| Motorola | INX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | IR | X + 1 ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 08 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ● | ○ | ○ |

| Motorola | INX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | IR | X + 1 ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 5/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 30 88 01 | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ● | ○ | ○ |

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.

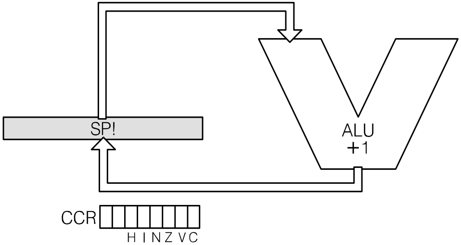
| Motorola | INS | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | IS | S + 1 ⇨ S | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 31 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

| Motorola | INS | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | IS | S + 1 ⇨ S | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 5/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 32 E8 01 | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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



**6809:**

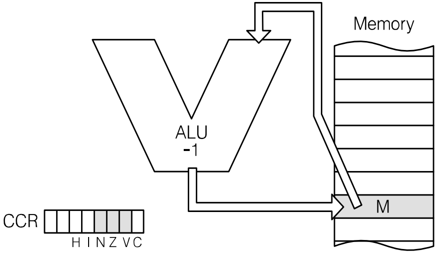
The instruction "INS" does not exist for the 6809. It is simulated by the "LEA!   !  001" command.

<table>
<tr><td colspan="2">Motorola</td><td colspan="2">DEC</td><td colspan="14">Function</td><td colspan="9">CPU type A / 6303</td></tr>
<tr><td colspan="2">B&R</td><td colspan="2">DEC</td><td colspan="23">(M) - 1 ⇨ (M)</td></tr>
<tr><td colspan="2">Short</td><td colspan="2"></td><td colspan="23"></td></tr>
<tr><td colspan="6" rowspan="2">Addressing mode / Opcode</td><td colspan="15" rowspan="2">Address preselection</td><td>○</td><td colspan="5">PG1000</td></tr>
<tr><td>○</td><td colspan="5">PG-PC</td></tr>
<tr><td>IMPL.</td><td>DIR.</td><td>EXT.</td><td>IMMED.</td><td>IND.</td><td>REL.</td><td>I</td><td>O</td><td>F</td><td>S</td><td>T</td><td>#</td><td>P</td><td>R</td><td>X</td><td>Y</td><td>D</td><td>U</td><td>!</td><td>B</td><td>G</td><td colspan="6">CCR</td></tr>
<tr><td></td><td></td><td>6/3</td><td></td><td>6/2</td><td></td><td>E</td><td>A</td><td>M</td><td>F</td><td>Z</td><td>#</td><td>P</td><td>C</td><td>I</td><td>Y</td><td>D</td><td>U</td><td>!</td><td>B</td><td>G</td><td>H</td><td>I</td><td>N</td><td>Z</td><td>V</td><td>C</td></tr>
<tr><td></td><td></td><td>7A</td><td></td><td>6A</td><td></td><td></td><td>○</td><td>○</td><td>○</td><td></td><td></td><td>○</td><td>○</td><td>○</td><td></td><td></td><td></td><td></td><td></td><td></td><td>○</td><td>○</td><td>●</td><td>●</td><td>●</td><td>○</td></tr>
</table>

<table>
<tr><td colspan="2">Motorola</td><td colspan="2">DEC</td><td colspan="14">Function</td><td colspan="9">CPU type B / 6809</td></tr>
<tr><td colspan="2">B&R</td><td colspan="2">DEC</td><td colspan="23">(M) - 1 ⇨ (M)</td></tr>
<tr><td colspan="2">Short</td><td colspan="2"></td><td colspan="23"></td></tr>
<tr><td colspan="6" rowspan="2">Addressing mode / Opcode</td><td colspan="15" rowspan="2">Address preselection</td><td>○</td><td colspan="5">CP 80</td></tr>
<tr><td>○</td><td colspan="5">PC 80</td></tr>
<tr><td>IMPL.</td><td>DIR.</td><td>EXT.</td><td>IMMED.</td><td>IND.</td><td>REL.</td><td>I</td><td>O</td><td>F</td><td>S</td><td>T</td><td>#</td><td>P</td><td>R</td><td>X</td><td>Y</td><td>D</td><td>U</td><td>!</td><td>B</td><td>G</td><td colspan="6">CCR</td></tr>
<tr><td></td><td>6/2</td><td>7/3</td><td></td><td>6+/2+</td><td></td><td>E</td><td>A</td><td>M</td><td>F</td><td>Z</td><td>#</td><td>P</td><td>C</td><td>I</td><td>Y</td><td>D</td><td>U</td><td>!</td><td>B</td><td>G</td><td>H</td><td>I</td><td>N</td><td>Z</td><td>V</td><td>C</td></tr>
<tr><td></td><td>0A</td><td>7A</td><td></td><td>6A</td><td></td><td></td><td>○</td><td>○</td><td>○</td><td></td><td></td><td>○</td><td>○</td><td>○</td><td>●</td><td>●</td><td>●</td><td>●</td><td></td><td>●</td><td>○</td><td>○</td><td>●</td><td>●</td><td>●</td><td>○</td></tr>
</table>

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.



| Motorola | DECA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | DA | A - 1 ⇨ A | |
| Short | | | |

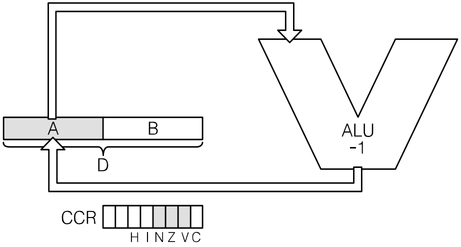
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4A | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ● | ❍ |

| Motorola | DECA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | DA | A - 1 ⇨ A | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4A | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ● | ❍ |

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.



| | | | |
|---|---|---|---|
| **Motorola** | **DECB** | **Function** | **CPU type A / 6303** |
| **B&R** | **DB** | **B - 1 ⇨ B** | |
| **Short** | | | |

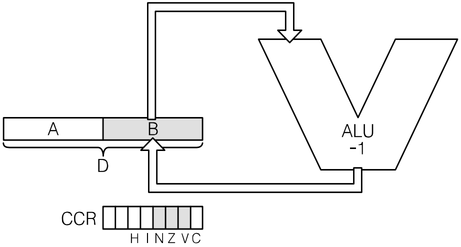
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5A | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

| | | | |
|---|---|---|---|
| **Motorola** | **DECB** | **Function** | **CPU type B / 6809** |
| **B&R** | **DB** | **B - 1 ⇨ B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5A | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ○ |

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.



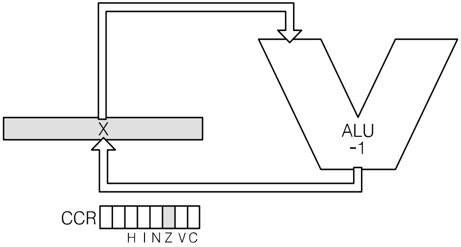
| Motorola | DEX | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | DR | X - 1 ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 09 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ● | ○ | ○ |

| Motorola | DEX | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | DR | X - 1 ⇨ X | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 5/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 30 88 FF | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ● | ○ | ○ |

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.

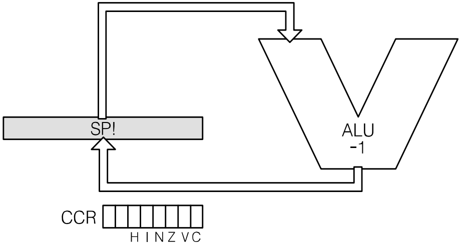
| Motorola | DES | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | DS | S - 1 ⇨ S | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 34 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

| Motorola | DES | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | DS | S - 1 ⇨ S | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | 5/3 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | 32 E8 FF | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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



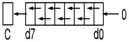
**6809:**

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

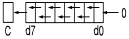
# 3.9. SHIFT AND ROTATE INSTRUCTIONS

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.

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

| Motorola | ASL | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SL | Shift contents of M |  |
| Short | | 1 bit to the left | |

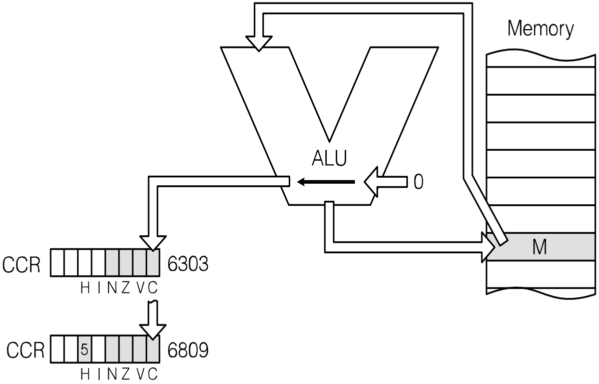
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 6/3 | | 6/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 78 | | 68 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

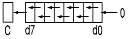
| Motorola | ASL | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SL | Shift contents of M |  |
| Short | | 1 bit to the left | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 08 | 78 | | 68 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | x | ○ | ● | ● | ● | ● |

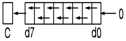
The contents of memory location M will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



| Motorola | ASLA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SLA | Shift contents of A |  |
| Short | | 1 bit to the left | |

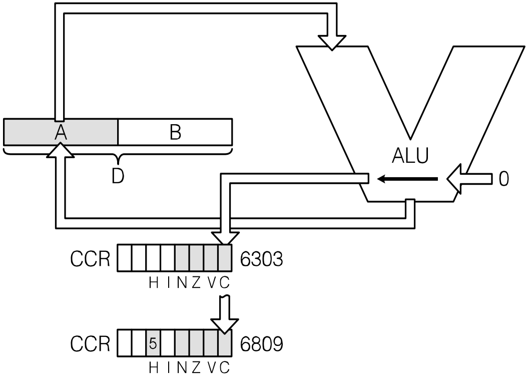
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 48 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

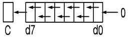
| Motorola | ASLA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SLA | Shift contents of A |  |
| Short | | 1 bit to the left | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 48 | | | | | | | | | | | | | | | | | | | | | × | ○ | ● | ● | ● | ● |

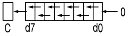
The contents of accumulator A will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



| Motorola | ASLB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SLB | Shift contents of B<br>1 bit to the left |  |
| Short | | | |

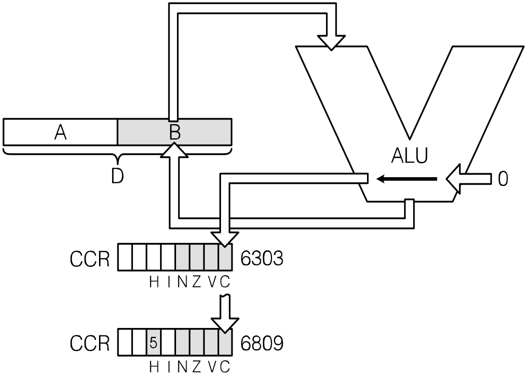
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 58 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

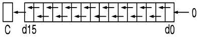
| Motorola | ASLB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SLB | Shift contents of B<br>1 bit to the left |  |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 58 | | | | | | | | | | | | | | | | | | | | | ✕ | ○ | ● | ● | ● | ● |

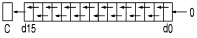
The contents of accumulator B will be moved 1 bit to the left. Bit 0 is replaced by logic 0 and bit 7 is moved to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



| Motorola | ASLD | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SLD | Shift contents of D<br>1 bit to the left |  |
| Short | | | |

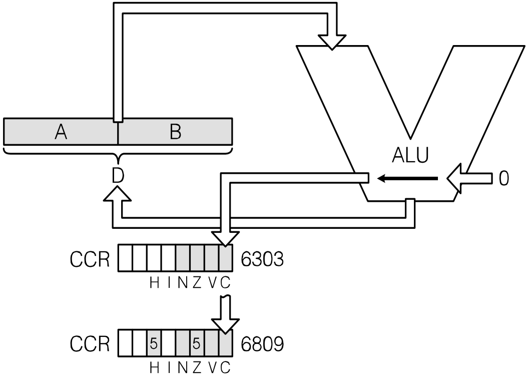
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 05 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

| Motorola | ASLD | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SLD | Shift contents of D<br>1 bit to the left |  |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 4/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 58 49 | | | | | | | | | | | | | | | | | | | | | × | ○ | ● | × | ● | ● |

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 ⊕ C (V = N ⊕ 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).

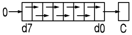


**6809:**

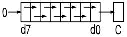
Since this instruction is not available in the 6809 it is simulated by the use of two other instructions.

| | |
|---|---|
| **ASLB** | 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.

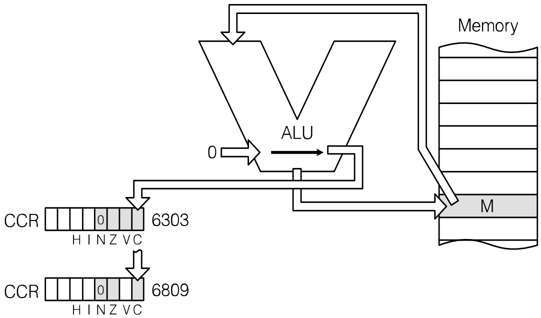
| Motorola | LSR | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SR | Shift contents of M<br>1 bit to the right |  |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 6/3 | | 6/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 74 | | 64 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ▼ | ● | ● | ● |

| Motorola | LSR | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SR | Shift contents of M<br>1 bit to the right |  |
| Short | | | |

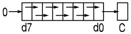
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 04 | 74 | | 64 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | ○ | ○ | ▼ | ● | ○ | ● |

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.

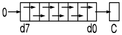


**6303:**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

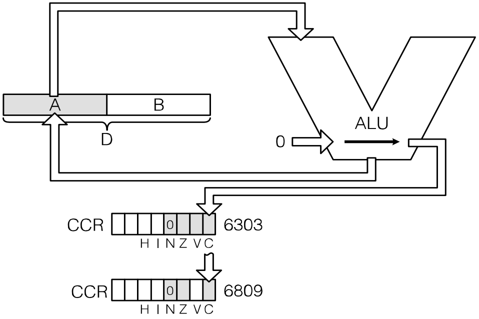
| | | | |
|---|---|---|---|
| **Motorola** | **LSRA** | **Function** | **CPU type A / 6303** |
| **B&R** | **SRA** | **Shift contents of A**<br>**1 bit to the right** |  |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 44 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Motorola** | **LSRA** | **Function** | **CPU type B / 6809** |
| **B&R** | **SRA** | **Shift contents of A**<br>**1 bit to the right** |  |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 44 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ● | ❍ | ● |

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.

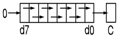


**6303:**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

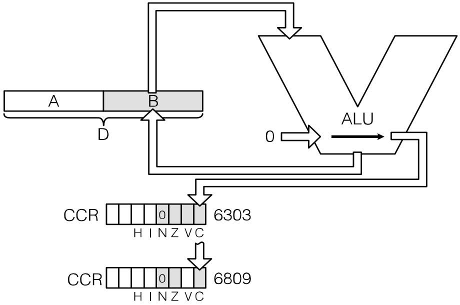
| Motorola | LSRB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SRB | Shift contents of B | |
| Short | | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 54 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ▼ | ● | ● | ● |

| Motorola | LSRB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SRB | Shift contents of B |  |
| Short | | 1 bit to the right | |

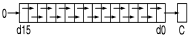
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 54 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ▼ | ● | ○ | ● |

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.

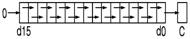


**6303:**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

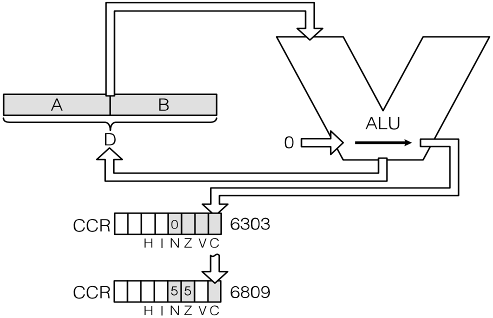
| Motorola | LSRD | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SRD | Shift contents of D |  |
| Short | | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 04 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ▼ | ● | ● | ● |

| Motorola | LSRD | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SRD | Shift contents of D |  |
| Short | | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 4/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 44 56 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | × | × | ○ | ● |

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:**

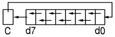
When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ C). The V flag shows whether bit 15 has been changed by the operation or not (V = 1 => bit 15 was changed; V = 0 => bit 15 was not changed).

**6809:**

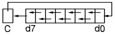
Since this instruction is not available in the 6809 it is simulated by the use of two other instructions.

| | |
|---|---|
| **LSRA** | The contents of A are moved 1 bit to the right. Bit 0 is moved to the carry flag and bit 7 is set to logic 0. |
| **RORB** | The contents of B are moved 1 bit to the right. The 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.

| | | | |
|---|---|---|---|
| Motorola | ROL | Function | CPU type A / 6303 |
| B&R | SLI | Rotate contents of M |  |
| Short | RL | 1 bit to the left | |

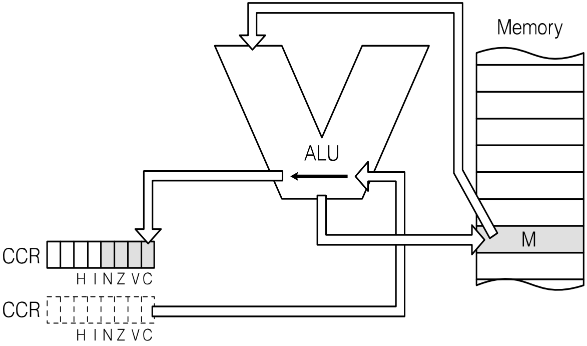
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 6/3 | | 6/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 79 | | 69 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ● | ● |

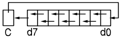
| | | | |
|---|---|---|---|
| Motorola | ROL | Function | CPU type B / 6809 |
| B&R | SLI | Rotate contents of M |  |
| Short | RL | 1 bit to the left | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 09 | 79 | | 69 | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | ○ | ○ | ● | ● | ● | ● |

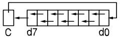
The contents of memory location M will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



| Motorola | ROLA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RLA | Rotate contents of A<br>1 bit to the left |  |
| Short | | | |

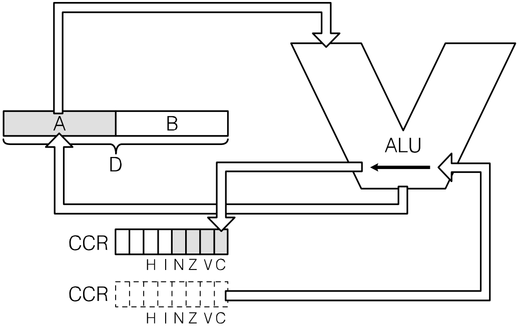
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 49 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

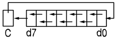
| Motorola | ROLA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RLA | Rotate contents of A<br>1 bit to the left |  |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 49 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

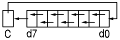
The contents of accumulator A will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



| Motorola | ROLB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RLB | Rotate contents of B |  C d7 d0 |
| Short |  | 1 bit to the left |  |

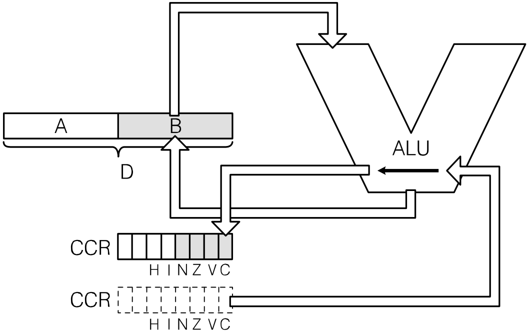
| Addressing mode / Opcode |  |  |  |  |  | Address preselection |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ PG1000<br>❍ PG-PC |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR |  |  |  |  |  |
| 1/1 |  |  |  |  |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ | ❍ | ● | ● | ● | ● |

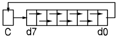
| Motorola | ROLB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RLB | Rotate contents of B |  C d7 d0 |
| Short |  | 1 bit to the left |  |

| Addressing mode / Opcode |  |  |  |  |  | Address preselection |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ CP 80<br>❍ PC 80 |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR |  |  |  |  |  |
| 2/1 |  |  |  |  |  | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ❍ | ❍ | ● | ● | ● | ● |

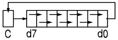
The contents of accumulator B will be moved 1 bit to the left. The carry flag is moved to bit 0 and bit 7 to the carry flag.

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).



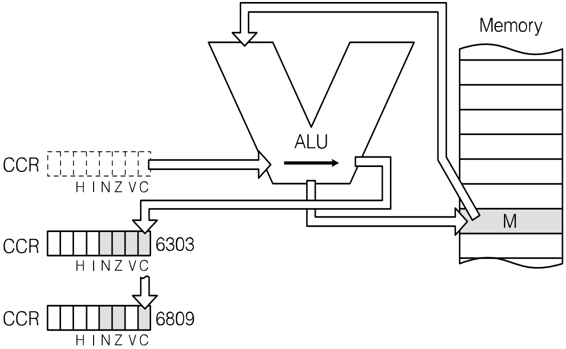
| | | | |
|---|---|---|---|
| Motorola | ROR | Function | CPU type A / 6303 |
| B&R | SRE | Rotate contents of M |  |
| Short | RR | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 6/3 | | 6/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 76 | | 66 | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| Motorola | ROR | Function | CPU type B / 6809 |
| B&R | SRE | Rotate contents of M |  |
| Short | RR | 1 bit to the right | |

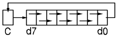
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 6/2 | 7/3 | | 6+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 06 | 76 | | 66 | | | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | ● | ● | ● | ● | | ● | ❍ | ❍ | ● | ● | ❍ | ● |

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.

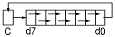


**6303:**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

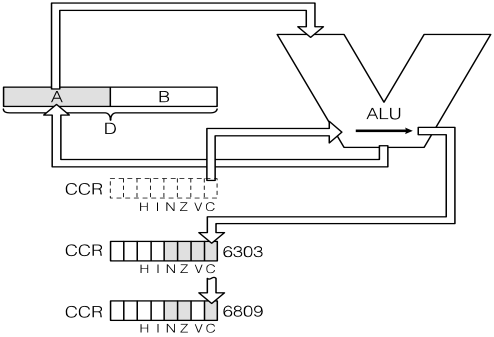
| | | | |
|---|---|---|---|
| **Motorola** | **RORA** | **Function** | **CPU type A / 6303** |
| **B&R** | **RRA** | **Rotate contents of A**<br>**1 bit to the right** |  |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 46 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Motorola** | **RORA** | **Function** | **CPU type B / 6809** |
| **B&R** | **RRA** | **Rotate contents of A**<br>**1 bit to the right** |  |
| **Short** | | | |

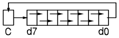
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 46 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | ○ | ● |

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.

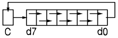


**6303:**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

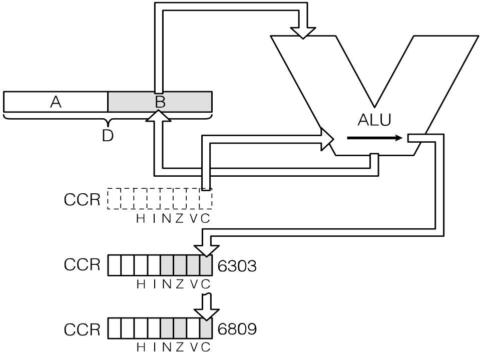
| Motorola | RORB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RRB | Rotate contents of B |  |
| Short | | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 56 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ● | ● |

| Motorola | RORB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RRB | Rotate contents of B |  |
| Short | | 1 bit to the right | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 56 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ❍ | ● |

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.



**<u>6303:</u>**

When the instruction is executed the V flag is assigned with the result of N ⊕ C (V = N ⊕ 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).

# 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 | Operating mode | | | |
|---|---|---|---|---|---|
| | | **PG1000** | **PG-PC** | **CP 80** | **PC 80** |
| BCC[1] | JC0[1] | ❍ | ❍ | ❍ | ❍ |
| BCS[1] | SP<[1] | ❍ | ❍ | ❍ | ❍ |
| BPL[1] | J+[1] | ❍ | ❍ | ❍ | ❍ |
| BMI[1] | J-[1] | ❍ | ❍ | ❍ | ❍ |
| BNE[1] | SN0[1] | ❍ | ❍ | ❍ | ❍ |
| BEQ[1] | SP0[1] | ❍ | ❍ | ❍ | ❍ |
| BHI[1] | SP>[1] | ❍ | ❍ | ❍ | ❍ |
| BLS[1] | J<=[1] | ❍ | ❍ | ❍ | ❍ |
| SK0 | SK0 | | ❍ | | ❍ |
| SK1 | SK1 | | ❍ | | ❍ |
| JSR | SPU | ❍ | ❍ | ❍ | ❍ |
| RTS | RET | ❍ | ❍ | ❍ | ❍ |
| JMP | SPI | ❍ | ❍ | ❍ | ❍ |
| NOP | NOP | ❍ | ❍ | ❍ | ❍ |
| END | END | ❍ | ❍ | ❍ | ❍ |

[1] **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 xxx is true** | |
| **Short** | **xxx** | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | xxx | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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

| **Mnemonics** | | | **Condition** | **Opcode** |
|---|---|---|---|---|
| **Motorola** | **B&R** | **Short** | | |
| BCC | JC0 | | C = 0 | 24 |
| BCS | SP< | J< | C = 1 | 25 |
| BEQ | SP0 | J0 | Z = 1 | 27 |
| BHI | SP> | J> | C ∨ 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 |

| | | | |
|---|---|---|---|
| **Motorola** | **xxx** | **Function** | **CPU type B / 6809** |
| **B&R** | **xxx** | **Branch, if condition xxx is true** | |
| **Short** | **xxx** | | |

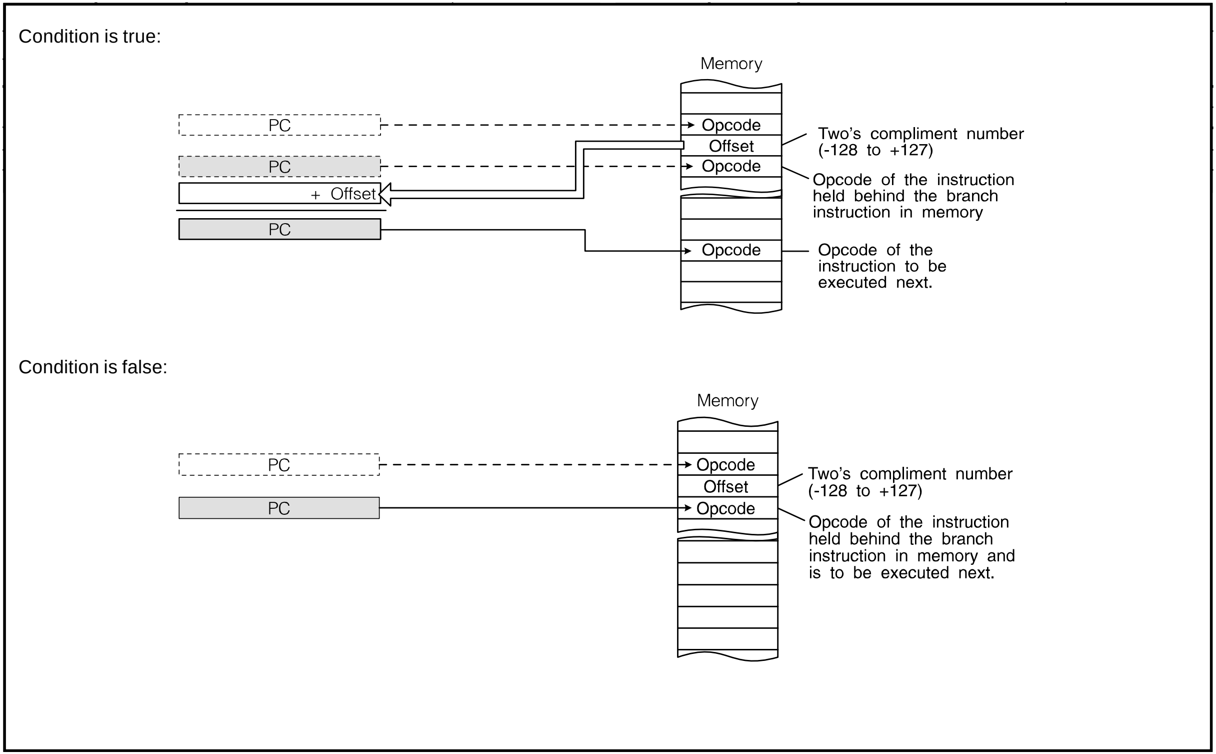
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | xxx | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

**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 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
```



Condition is true:

Memory
PC
Opcode
Offset
Two’s compliment number (-128 to +127)
PC
Opcode
Opcode of the instruction held behind the branch instruction in memory
+ Offset
PC
Opcode
Opcode of the instruction to be executed next.

Condition is false:

Memory
PC
Opcode
Offset
Two’s compliment number (-128 to +127)
PC
Opcode
Opcode of the instruction held behind the branch instruction in memory and is to be executed next.

| | | | |
|---|---|---|---|
| **Motorola** | | **Function** | **CPU type A / 6303** |
| **B&R** | | | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000** | | | | |
| | | | | | | | | | | | | | | | | | | | | | | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

“xxx” in the table above stands for the conditions defined below:

| **Mnemonics** | | | **Condition** | **Opcode** |
|---|---|---|---|---|
| **Motorola** | **B&R** | **Short** | | |
| BCCL | JC0L | | C = 0 | 10 24 |
| BCSL | SP<L | | C = 1 | 10 25 |
| BEQL | SP0L | | Z = 1 | 10 27 |
| BHIL | SP>L | | C ∨ 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 |

| | | | |
|---|---|---|---|
| **Motorola** | **xxx** | **Function** | **CPU type B / 6809** |
| **B&R** | **xxx** | **Branch, if condition xxx is true** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **CP 80** | | | | |
| | | | | | | | | | | | | | | | | | | | | | ○ | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 5(6)/4 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | xxx | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

**Long** relative branch consists of four bytes:

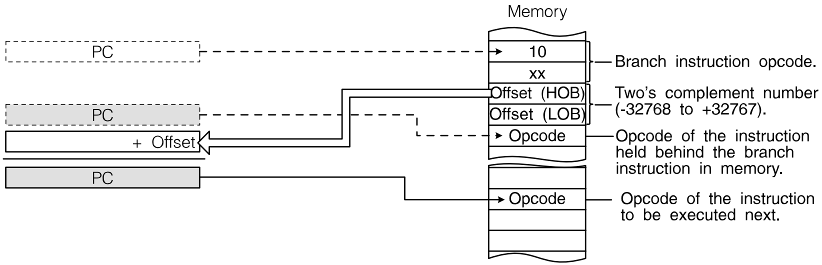
1) Bytes 1 and 2 are the opcode of the instruction.

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.

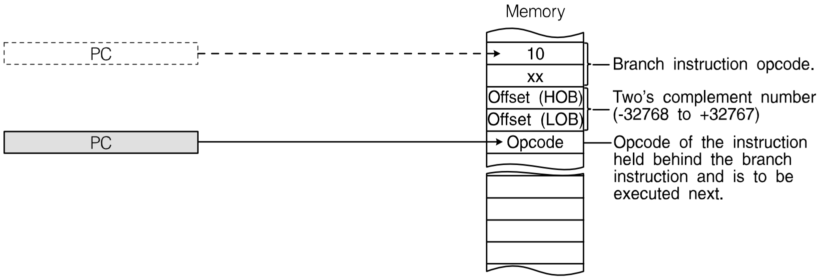
**Execution time:** 5(6) Clock cycles

```
                     5(6) Clock cycles
                     │ └── Condition true, branch executed.
                     └──── Condition false, branch not executed.
```

Condition is true:


Memory
PC
PC
+ Offset
PC
10
xx
Offset (HOB)
Offset (LOB)
Opcode
Opcode
Branch instruction opcode.
Two's complement number (-32768 to +32767).
Opcode of the instruction held behind the branch instruction in memory.
Opcode of the instruction to be executed next.

Condition is false:


Memory
PC
PC
10
xx
Offset (HOB)
Offset (LOB)
Opcode
Branch instruction opcode.
Two's complement number (-32768 to +32767)
Opcode of the instruction held behind the branch instruction and is to be executed next.

| | | | |
|---|---|---|---|
| **Motorola** | **SK0** | **Function** | **CPU type A / 6303** |
| **B&R** | **SK0** | **Skip next instruction, if Z = 1** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ❍ | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | 27 | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

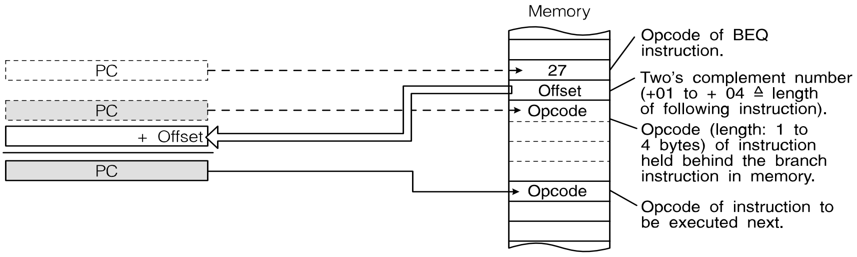
| | | | |
|---|---|---|---|
| **Motorola** | **SK0** | **Function** | **CPU type B / 6809** |
| **B&R** | **SK0** | **Skip next instruction, if Z = 1** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **CP 80** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ❍ | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | 27 | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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.

Condition is true:



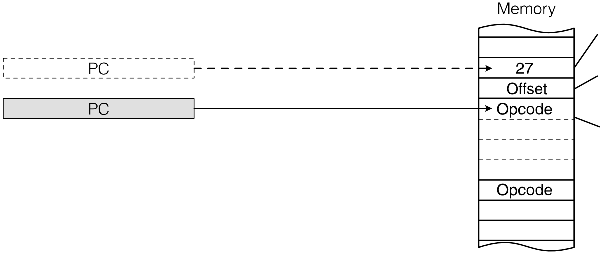
Opcode of BEQ instruction.

Two's complement number (+01 to + 04 ≙ length of following instruction).

Opcode (length: 1 to 4 bytes) of instruction held behind the branch instruction in memory.

Opcode of instruction to be executed next.

Condition is false:



Opcode of the BEQ instruction.

Two's complement number (+01 to + 04 ≙ length of the following instruction).

Opcode (length: 1 to 4 bytes) of the instruction held behind the branch instruction to be executed next.

| | | | |
|---|---|---|---|
| **Motorola** | **SK1** | **Function** | **CPU type A / 6303** |
| **B&R** | **SK1** | **Skip next instruction, if Z = 0** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **PG1000** | | | | |
| | | | | | | | | | | | | | | | | | | | | | ○ | **PG-PC** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | 26 | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

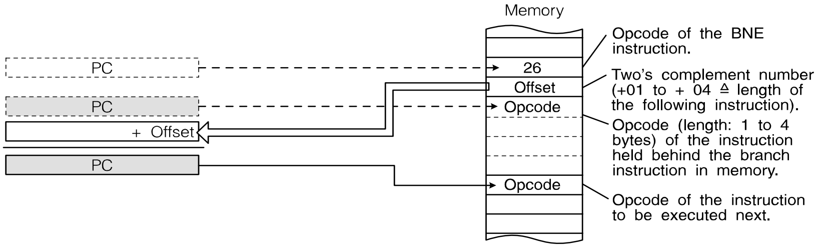
| | | | |
|---|---|---|---|
| **Motorola** | **SK1** | **Function** | **CPU type B / 6809** |
| **B&R** | **SK1** | **Skip next instruction, if Z = 0** | |
| **Short** | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | | **CP 80** | | | | |
| | | | | | | | | | | | | | | | | | | | | | ○ | **PC 80** | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | 3/2 | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | | | | 26 | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

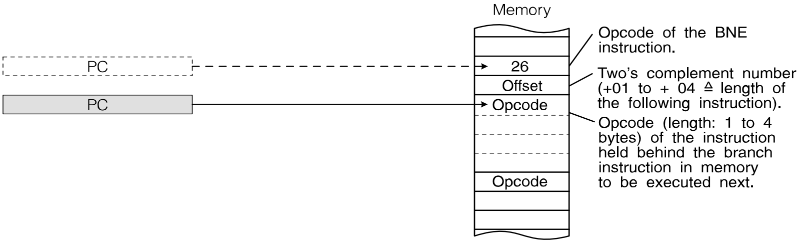
The following instruction is skipped if the zero flag is set to logic 0 (previous result of operation is not zero).

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.

Condition is true:


Memory
PC
26
Offset
PC
Opcode
+ Offset
PC
Opcode
Opcode of the BNE instruction.
Two's complement number (+01 to + 04 ≙ length of the following instruction).
Opcode (length: 1 to 4 bytes) of the instruction held behind the branch instruction in memory.
Opcode of the instruction to be executed next.

Condition is false:



Memory
PC
PC
26
Offset
Opcode
Opcode
Opcode of the BNE instruction.
Two's complement number (+01 to + 04 ≙ length of the following instruction).
Opcode (length: 1 to 4 bytes) of the instruction held behind the branch instruction in memory to be executed next.

| Motorola | JSR | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SPU | Unconditional branch in a sub-program | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 5/2 | 6/3 | | 5/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9D | BD | | AD | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

| Motorola | JSR | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SPU | Unconditional branch in a sub-program | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 7/2 | 8/3 | | 7+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 9D | BD | | AD | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

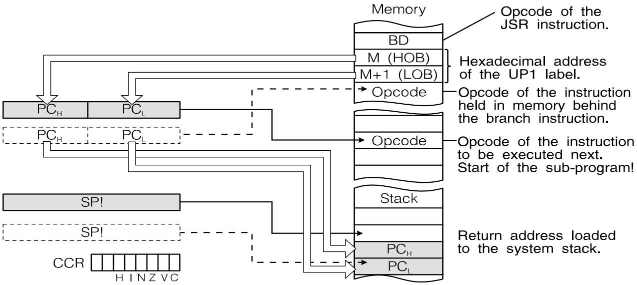
This instruction causes a jump to the desired location in the program. Before the jump the instruction address (=> return address) which is held behind the branch instruction is placed on the system stack.

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.

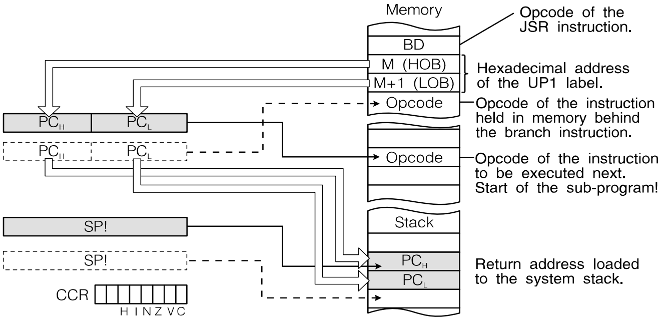
Stack processing varies between the two types of processors:

**Example:**
```
JSR     UP1
 :
```

**<u>6303:</u>**


Memory
Opcode of the JSR instruction.
BD
M (HOB)
M+1 (LOB)
Hexadecimal address of the UP1 label.
Opcode
Opcode of the instruction held in memory behind the branch instruction.
PCH
PCL
PCH
PCL
Opcode
Opcode of the instruction to be executed next. Start of the sub-program!
SP!
SP!
Stack
PCH
PCL
Return address loaded to the system stack.
CCR
H I N Z V C

**6809:**



Opcode of the JSR instruction.

Hexadecimal address of the UP1 label.

Opcode of the instruction held in memory behind the branch instruction.

Opcode of the instruction to be executed next. Start of the sub-program!

Return address loaded to the system stack.

| | | | |
|---|---|---|---|
| Motorola | RTS | Function | CPU type A / 6303 |
| B&R | RET | Return jump from sub-program | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 5/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 39 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

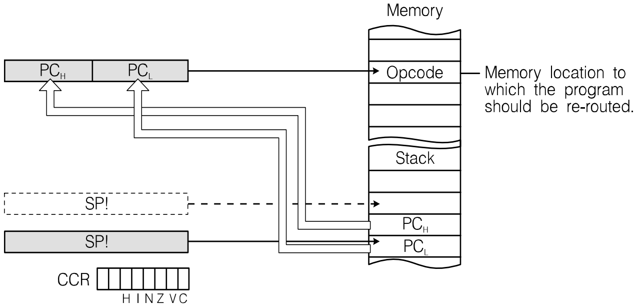
| | | | |
|---|---|---|---|
| Motorola | RTS | Function | CPU type B / 6809 |
| B&R | RET | Return jump from sub-program | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 5/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 39 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

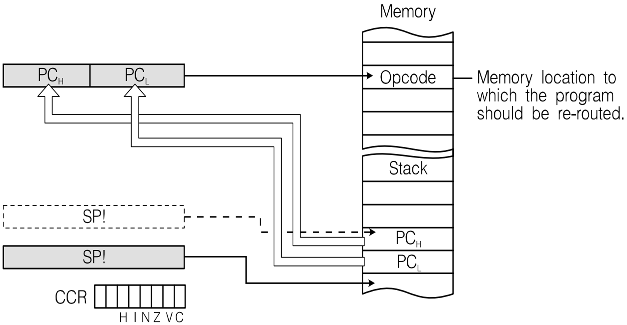
This instruction causes a return jump to the address located on the system stack. If a so called sub-program is called with the JSR instruction it must be quit with RTS, otherwise a "STACK FAILURE" can occur.

Stack processing varies between the two types of processors:

**6303:**



**6809:**


Memory
PC$_H$
PC$_L$
Opcode
Memory location to which the program should be re-routed.
Stack
SP!
SP!
PC$_H$
PC$_L$
CCR
H I N Z V C

| Motorola | JMP | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SPI | Unconditional jump | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 3/3 | | 3/2 | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 7E | | 6E | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

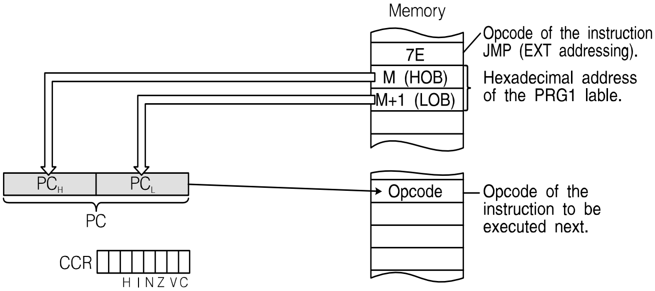
| Motorola | JMP | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SPI | Unconditional jump | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | 3/2 | 4/3 | | 3+/2+ | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 0E | 7E | | 6E | | | | | | | | | | | | | | | | | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

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`


Memory
Opcode of the instruction JMP (EXT addressing).
7E
M (HOB)
M+1 (LOB)
Hexadecimal address of the PRG1 lable.
PC_H
PC_L
PC
Opcode
Opcode of the instruction to be executed next.
CCR
H I N Z V C

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

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ○ **PG-PC** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 01 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.

| | | | |
|---|---|---|---|
| **Motorola** | **NOP** | **Function** | **CPU type B / 6809** |
| **B&R** | **NOP** | **No operation** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | ○ **PC 80** | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 12 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ |

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.

| | | | |
|---|---|---|---|
| **Motorola** | **END** | **Function** | **CPU type A / 6303** |
| **B&R** | **END** | **Program end!** | |
| **Short** | | **Start in program line 0** | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 7E C0 00 | | | | | | | | | | | | | | | | | | | ○ | ○ | ▼ | ▲ | ▼ | ▼ |

| | | | |
|---|---|---|---|
| **Motorola** | **END** | **Function** | **CPU type B / 6809** |
| **B&R** | **END** | **Program end!** | |
| **Short** | | **Start in program line 0** | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 11 3F | | | | | | | | | | | | | | | | | | | | | ○ | ▼ | ○ | ○ | ○ | ○ |

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.

**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

| Motorola | B&R | Operating mode | | | |
|---|---|---|---|---|---|
| | | PG1000 | PG-PC | CP 80 | PC 80 |
| PRS | PRS | ❍ | ❍ | ❍ | ❍ |
| RST | RST | ❍ | ❍ | ❍ | ❍ |
| CLR | CLR | ❍ | ❍ | ❍ | ❍ |
| CLRA | CLA | | ❍ | | ❍ |
| CLRB | CLB | | ❍ | | ❍ |
| SET | SET | ❍ | ❍ | ❍ | ❍ |
| CLC | CLC | ❍ | ❍ | ❍ | ❍ |
| SEC | SEC | ❍ | ❍ | ❍ | ❍ |
| CLI | CLI | ❍ | ❍ | ❍ | ❍ |
| SEI | SEI | ❍ | ❍ | ❍ | ❍ |
| COM | K | ❍ | ❍ | ❍ | ❍ |
| COMA | COA | | ❍ | | ❍ |
| COMB | COB | | ❍ | | ❍ |
| DAA | DK | ❍ | ❍ | ❍ | ❍ |

| Motorola | PRS | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | PRS | If d0 from A = 1: 1 ⇨ (M) | |
| Short | P | If d0 from A = 0: (M) remains unchanged | |

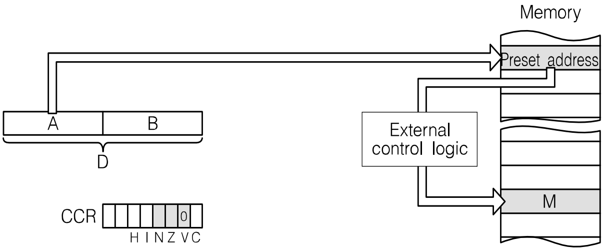
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 4/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | B7 | | | | | ○ | ○ | ○ | | | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

| Motorola | PRS | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | PRS | If d0 from A = 1: 1 ⇨ (M) | |
| Short | P | If d0 from A = 0: (M) remains unchanged | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 5/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | B7 | | | | | ○ | ○ | ○ | | | | | | | | | | | | ○ | ○ | ● | ● | ▼ | ○ |

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!**

| Motorola | RST | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | RST | If d0 from A = 1: 0 ⇨ (M) | |
| Short | R | If d0 from A = 0: (M) remains unchanged | |

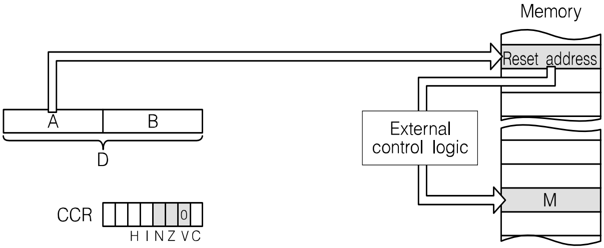
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ | PG1000 | | | | |
| | | | | | | | | | | | | | | | | | | | | | ❍ | PG-PC | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 4/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | B7 | | | | | ❍ | ❍ | ❍ | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

| Motorola | RST | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | RST | If d0 from A = 1: 0 ⇨ (M) | |
| Short | R | If d0 from A = 0: (M) remains unchanged | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ | CP 80 | | | | |
| | | | | | | | | | | | | | | | | | | | | | ❍ | PC 80 | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 5/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | B7 | | | | | ❍ | ❍ | ❍ | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ❍ |

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 | C | | |

| Addressing mode / Opcode | | | | | |
|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. |
| | | 5/3 | | 5/2 | |
| | | 7F | | 6F | |

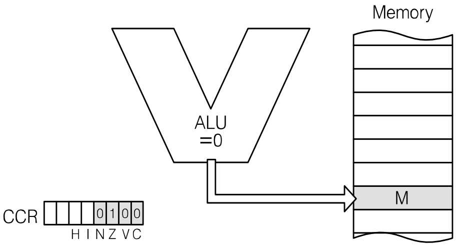
| Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | | | | | | | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

| Motorola | CLR | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | CLR | 0 ⇨ (M) | |
| Short | C | | |

| Addressing mode / Opcode | | | | | |
|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. |
| | 6/2 | 7/3 | | 6+/2+ | |
| | 0F | 7F | | 6F | |

| Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | ❍ | ❍ | ❍ | | | ❍ | ❍ | ❍ | ● | ● | ● | ● | | ● | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

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



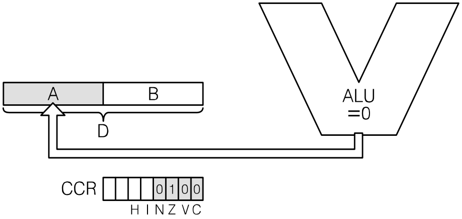
| Motorola | CLRA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | CLA | 0 ⇨ A | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4F | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

| Motorola | CLRA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | CLA | 0 ⇨ A | |
| Short | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 4F | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

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



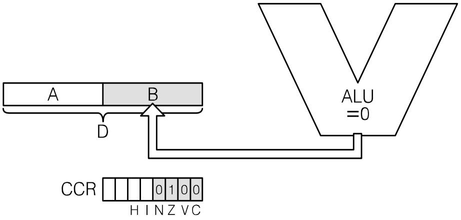
| | | | |
|---|---|---|---|
| **Motorola** | **CLRB** | **Function** | **CPU type A / 6303** |
| **B&R** | **CLB** | **0 ⇨ B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **PG1000**<br>❍ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5F | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

| | | | |
|---|---|---|---|
| **Motorola** | **CLRB** | **Function** | **CPU type B / 6809** |
| **B&R** | **CLB** | **0 ⇨ B** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ❍ **CP 80**<br>❍ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 5F | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ▼ | ▲ | ▼ | ▼ |

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 | | | |

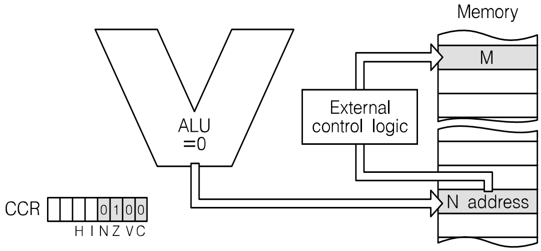
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 5/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 7F | | | | | ○ | ○ | ○ | | | | | | | | | | | | ○ | ○ | ▼ | ▲ | ▼ | ▼ |

| Motorola | SET | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SET | 1 ⇨ (M) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | 7/3 | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 7F | | | | | ○ | ○ | ○ | | | | | | | | | | | | ○ | ○ | ▼ | ▲ | ▼ | ▼ |

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!**

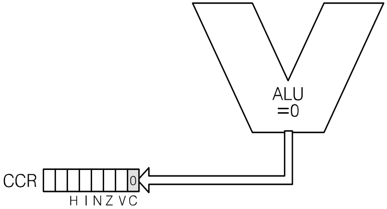
| | | | |
|---|---|---|---|
| **Motorola** | **CLC** | **Function** | **CPU type A / 6303** |
| **B&R** | **CLC** | **0 ⇨ C** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **PG1000**<br>○ **PG-PC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 0C | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ▼ |

| | | | |
|---|---|---|---|
| **Motorola** | **CLC** | **Function** | **CPU type B / 6809** |
| **B&R** | **CLC** | **0 ⇨ C** | |
| **Short** | | | |

| **Addressing mode / Opcode** | | | | | | **Address preselection** | | | | | | | | | | | | | | | ○ **CP 80**<br>○ **PC 80** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | **CCR** | | | | | |
| 3/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1C FE | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ▼ |

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



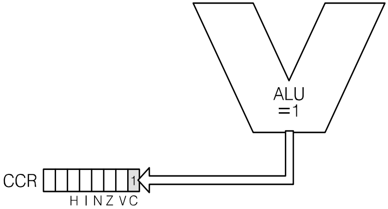
| Motorola | SEC | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SEC | 1 ⇨ C | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 0D | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ▲ |

| Motorola | SEC | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SEC | 1 ⇨ C | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1A 01 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ▲ |

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



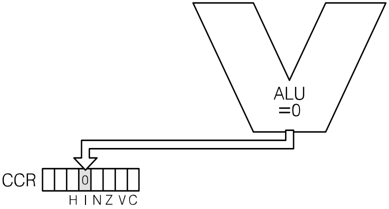
| Motorola | CLI | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | CLI | 0 ⇨ I | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 0E | | | | | | | | | | | | | | | | | | | | | ○ | ▼ | ○ | ○ | ○ | ○ |

| Motorola | CLI | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | CLI | 0 ⇨ I | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1C EF | | | | | | | | | | | | | | | | | | | | | ○ | ▼ | ○ | ○ | ○ | ○ |

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.



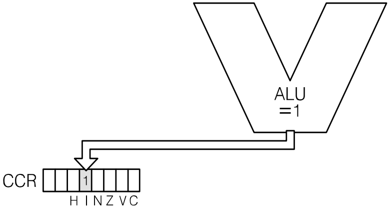
| Motorola | SEI | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | SEI | 1 ⇨ I | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 0F | | | | | | | | | | | | | | | | | | | | | ❍ | ▲ | ❍ | ❍ | ❍ | ❍ |

| Motorola | SEI | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | SEI | 1 ⇨ I | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 3/2 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 1A 10 | | | | | | | | | | | | | | | | | | | | | ❍ | ▲ | ❍ | ❍ | ❍ | ❍ |

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:**

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!**

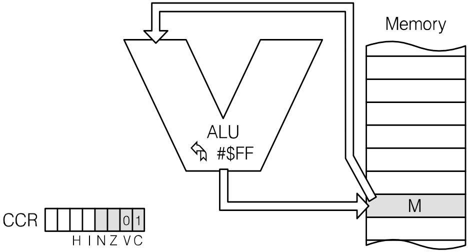
| Motorola | COM | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | K | (M) ⊕ #$FF ⇨ (M)   (=> Negation) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | | 6/3 | 6/2 | | | | | | | | | | | | | | | | | | | | | | | |
| | | 73 | 63 | | | | ○ | ○ | ○ | | | ○ | ○ | ○ | | | | | | | ○ | ○ | ● | ● | ▼ | ▲ |

| Motorola | COM | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | K | (M) ⊕ #$FF ⇨ (M)   (=> Negation) | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| | 6/2 | 7/3 | 6+/2+ | | | | | | | | | | | | | | | | | | | | | | | |
| | 03 | 73 | 63 | | | | ○ | ○ | ○ | | | ○ | ○ | ○ | ● | ● | ● | ● | | ● | ○ | ○ | ● | ● | ▼ | ▲ |

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



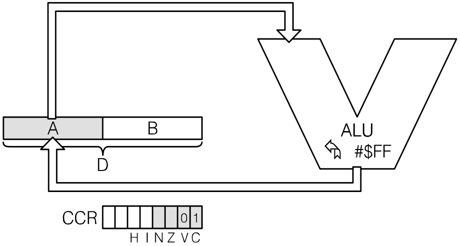
| Motorola | COMA | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | COA | A ⊕ #$FF ⇨ A (=> Negation) | |
| Short | KA | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 43 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ▲ |

| Motorola | COMA | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | COA | A ⊕ #$FF ⇨ A (=> Negation) | |
| Short | KA | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 43 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ▲ |

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



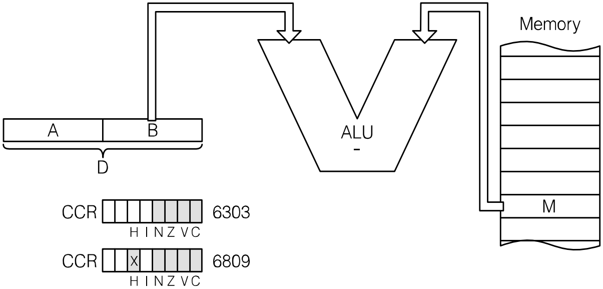
| Motorola | COMB | Function | CPU type A / 6303 |
|---|---|---|---|
| B&R | COB | B ⊕ #$FF ⇨ B (=> Negation) | |
| Short | KB | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ PG1000<br>❍ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 1/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 53 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ▲ |

| Motorola | COMB | Function | CPU type B / 6809 |
|---|---|---|---|
| B&R | COB | B ⊕ #$FF ⇨ B (=> Negation) | |
| Short | KB | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ❍ CP 80<br>❍ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 53 | | | | | | | | | | | | | | | | | | | | | ❍ | ❍ | ● | ● | ▼ | ▲ |

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



| | | | |
|---|---|---|---|
| Motorola | DAA | Function | CPU type A / 6303 |
| B&R | DK | Decimal adjustment of accumulator A after an addition of BCD numbers | |
| Short | | | |

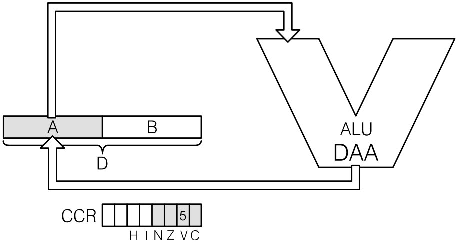
| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ PG1000<br>○ PG-PC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 19 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | × | ● |

| | | | |
|---|---|---|---|
| Motorola | DAA | Function | CPU type B / 6809 |
| B&R | DK | Decimal adjustment of accumulator A after an addition of BCD numbers | |
| Short | | | |

| Addressing mode / Opcode | | | | | | Address preselection | | | | | | | | | | | | | | | ○ CP 80<br>○ PC 80 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMPL. | DIR. | EXT. | IMMED. | IND. | REL. | I | O | F | S | T | # | P | R | X | Y | D | U | ! | B | G | CCR | | | | | |
| 2/1 | | | | | | E | A | M | F | Z | # | P | C | I | Y | D | U | ! | B | G | H | I | N | Z | V | C |
| 19 | | | | | | | | | | | | | | | | | | | | | ○ | ○ | ● | ● | × | ● |

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

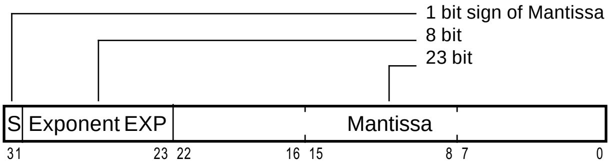
# ERROR MESSAGES

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 |

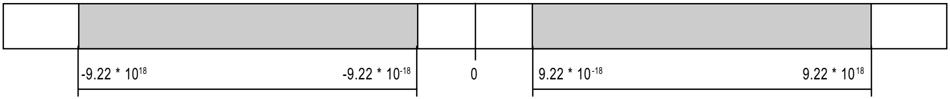
# DATA FORMAT

## Single Precision Floating Point Format



**Conversion:** $(-1)^{S} \bullet 2^{(EXP-127)} \bullet 1.\text{mantissa}$

**Number range display:**

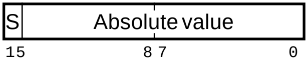


All numbers from $-9.22 * 10^{-18}$ to $+9.22 * 10^{-18}$ with the exception of 0 can be displayed and are used the same as 0.

# Absolute With Sign

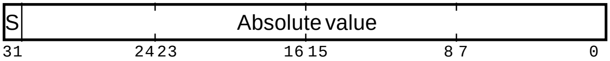
S ... Sign

## Absolute word



± 32767
±($2^{15}$ - 1)

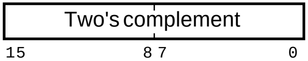
## Absolute long



± 2147483647
±($2^{31}$ - 1)

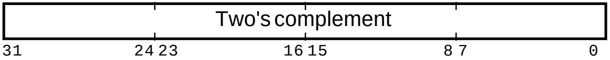
# Integer

## Integer word



-32768 (-$2^{15}$)
to
32767 ($2^{15}$-1)

## Integer long



-2147483648 (-$2^{31}$)
to
2147483647 ($2^{31}$-1)
(±2,15 * $10^9$)

## 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** | ❍ | ❍ | | | | ❍ | ❍ | | | | | ❍ | | |

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

**Parameters:** None

**Result:** D Changed
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** | ❍ | ❍ | | | | ❍ | ❍ | | | | | ❍ | | |

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

**Parameters:** None

**Result:** D Changed
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** | ❍ | ❍ | | | | ❍ | ❍ | | | | | ❍ | | |

**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

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **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** | ❍ | ❍ | ❍ | | | ❍ | ❍ | | | | | ❍ | | |

**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** | ❍ | | | | | ❍ | ❍ | ❍ | | | | ❍ | | |

**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 | 0 |
| N, Z | Corresponds with routine result |

## MCOP Copy Operand OP1 to Operand OP2

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

**Function:** Operand OP1 is copied to OP2.

**Parameters:** None

**Result:** D Changed

## MEXG Exchange Operands OP1 and OP2 with each other

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

**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 |  |  |  |  | ❍ |  |  |  |  |  |  |  |  |  |

**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.

**Parameters:** X Source address of binary number

**Result:** D Changed
N, Z Correspond with routine result

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

```
LDX#  R 0320  Binary # source address
LAL1
```

## 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 |  |  |  |  | ❍ |  |  |  |  |  |  |  |  |  |

**Function:** 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.

**Parameters:** X Source address of binary number

**Result:** D Changed
N, Z 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 | 13 | 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 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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** | | | | | ❍ | | | | | | | | | |

**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.

**Parameters:** X Source address of binary number

**Result:** D Changed
N, Z Correspond with routine result

## LIL2 Load Operand OP2 with Number (Integer long)

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 194 - 378 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 198 - 315 | **error message** | | | | | ❍ | | | | | | | | | |

**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.

**Parameters:** X Source address of binary number

**Result:** D Changed
N, Z 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 | 13 | 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 | ❍ |  |  |  |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

**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 | ❍ |  |  |  |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

**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 | Correspond with routine result |

## SAL

## Store Operand OP1 in Absolute Long Format

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 169 - 408 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 189 - 339 | **error message** | | | | ❍ | | | | | | | | ❍ | | |

**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** | | | | ❍ | | | | | | | | ❍ | | |

**Function:** 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 in µs | 6303 | 172 - 424 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 6809 | 190 - 347 |  |  |  |  | ❍ |  |  |  |  |  |  |  | ❍ |  |  |

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

**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 OP1 in IEEE Format

| **Execution time in µs** | **6303** | 43 | **Possible error message** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **6809** | 112 | | | | | | | | | | | | | | | |

**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 | 13 | 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

| Execution time in µs | 6303 | 56 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 116 | | | | | | | | | | | | | | | |

**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 in µs | 6303 | 56 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 116 | | | | | | | | | | | | | | | |

**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

| Execution time in µs | 6303 | 56 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 116 | | | | | | | | | | | | | | | |

**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 in µs | | | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **6303** | 115 - 191 | | | | | | | | | | | | | | | |
| | **6809** | 125 - 175 | | | | | | | | | | | | | | | |

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

**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 × 2 Byte Multiplication

| Execution time in µs | 6303 | 192 - 344 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 172 - 272 | | | | | | | | | | | | | | | |

**Function:** A number in integer long format (4 byte) is multiplied with a number in integer word format (2 byte). The result is a 6 byte integer number.

| **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 |
| | X | Unchanged |

# CAF — Convert ASCII String to Floating Point (IEEE) Format

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 280 - 2140 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 251 - 1460 | **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**

1. ***Signs*** permitted are <+>, <-> and <blank spaces>. The sign is optional.
2. 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 llead 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> ($OD) or <0> ($00)***.

**Examples of valid ASCII strings:**

```
" 12.45<CR>"         "-12.45<CR>"        "+.23<CR>"          ".1234567<CR>"      "0.0022<CR>"
"-1.234567 E 09"     "+.34E-8<CR>"       "0.022E1<CR>"       "-12.45 E+01"       ".3E+14"
```

# CFA

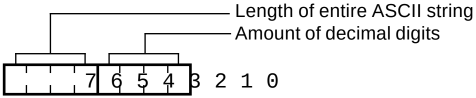
## Convert OP1 to ASCII (without leading zeros)

| Execution time | 6303 | 352 - 7310 | Possible | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| in µs | 6809 | 352 - 4440 | error message | ❍ |  |  | ❍ |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

**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 | |
| A | Format of ASCII string: | |



**Result:**

| | |
|---|---|
| A | Changed |
| B | Length 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.

```
  :
  :
LDX#   R  0250          Load destination address for ASCII string in index register X
                        Length of ASCII string (5 significant positions + sign + decimal point = 7)
                        Amount of decimal digits (2)
LDAA   #  $72           Format of ASCII string transferred to A
CFA
  :
  :
```

```
OP1 = 32.123        =>    "  32.12"
OP1 = -0.1          =>    "-  0.10"
OP1 = 4.87          =>    "   4.87"
OP1 = 2300.25       =>    " >>>.>>"
OP1 = -1000.25      =>    "-<<<.<<"
```

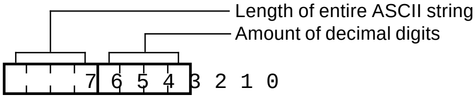
# CFA0 Convert OP1 to ASCII (with leading zeros)

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 310 - 7190 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 310 - 4320 | **error message** | ❍ | | | ❍ | | ❍ | ❍ | | | | | ❍ | | |

**Function:** The contents of OP1 are converted to an ASCII string and stored at the address indicated by index register X. Operands OP1 and OP2 remain unchanged.
The amount of significant digits is limited to a maximum of seven. The ASCII string can therefore be maximum nine characters long (including sign and decimal point). Leading zeros are not suppressed. If OP1 can not be converted to the desired ASCII format the string is filled with “>” characters.

**Parameters:** X Destination address of ASCII string

A Format of ASCII string:



**Result:**

A Changed
B Length of ASCII string
N, Z 0

**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.

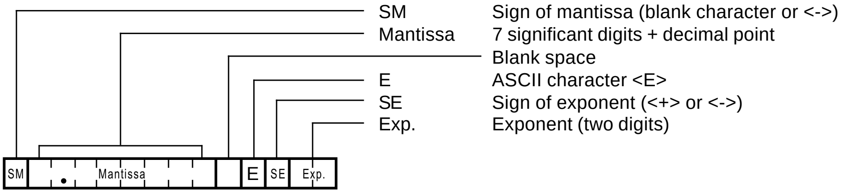
```
  :
  :
LDX#    R  0100         Load destination address for the ASCII string in index register X

                        Length of ASCII string (4 significant positions + sign + decimal  point = 6)
                        Amount of decimal digits (1)
LDAA    #  $61          Format of ASCII string transferred to A
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 in µs | 6303 | 570 - 7140 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 267 - 4140 | | ❍ | | | | | ❍ | ❍ | | | | | ❍ | | |

**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).



| | |
|---|---|
| SM | Sign of mantissa (blank character or <->) |
| Mantissa | 7 significant digits + decimal point |
| | Blank space |
| E | ASCII character <E> |
| SE | Sign of exponent (<+> or <->) |
| Exp. | Exponent (two digits) |

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
     :
     :
```

```
OP1 = 32111          =>     " 3.211100 E+04"
OP1 = -487           =>     "-4.870000 E+02"
OP1 = 0.0456         =>     " 4.560000 E-02"
OP1 = 0              =>     " 0.000000 E+00"
```

# CIA Convert Integer to ASCII (without leading zeros)

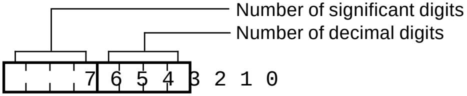
| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 380 - 2020 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 357 - 1370 | **error message** | | | | ❍ | | | | | | | | | | |

**Function:** A binary number is converted to an ASCII string and stored to the memory location which is addressed by index register X. Operands OP1 and OP2 remain unchanged.
The number of significant digits is limited to a maximum of nine. The ASCII string can therefore be up to eleven characters (including sign and decimal point). Leading zeros are replaced by blank spaces. If the binary number cannot be converted to the desired ASCII format the string is filled with “>” or “<” characters.

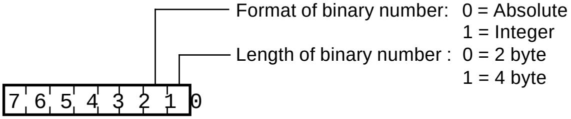
**Parameters:**

R1046& Source address of binary number
X Destination address of ASCII string

A Format of ASCII string:



B Format of binary number:


Format of binary number: 0 = Absolute
1 = Integer
Length of binary number : 0 = 2 byte
1 = 4 byte
7 6 5 4 3 2 1 0

**Result:**

A Changed
B Length of entire ASCII string
N, Z 0

**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.

```
 :
 :
LDX#   R  0100          Source address of binary number
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
LDAA   #  $62           Transfer format of ASCII string to accumulator A
                        Format of binary number: 1 => Integer
                        Length of binary number: 1 => 4 Byte
LDAB   #  %00000011     Transfer format of binary number to accumulator B
CIA                     Routine call
 :
 :
```

```
Integer-Long = 56499        =>    "  564.99""
Integer-Long = -23          =>    "-   0.23"
Integer-Long = 1000000      =>    " >>>>.>>"
```

# CIA0 Convert Integer to ASCII (with leading zeros)

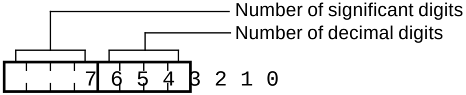
| Execution time in µs | 6303 | 310 - 1960 | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | 312 - 1320 | | | | | ❍ | | | | | | | | | | |

**Function:** A binary number is converted to an ASCII string and stored to the address given by index register X. Operands OP1 and OP2 remain unchanged.
The number of significant digits is limited to a maximum of nine. The ASCII string can therefore be maximum eleven characters long (including sign and decimal point). Leading zeros are not suppressed. If the binary number cannot be converted to the desired ASCII format the string is filled with “>” or “<” characters.

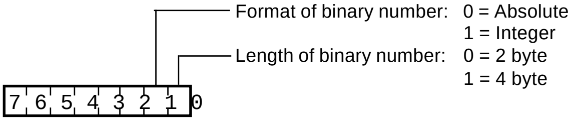
**Parameters:**

| | |
|---|---|
| R1046& | Source address of binary number |
| X | Destination address ASCII string |

A Format of ASCII string:



B Format of binary number:


Format of binary number: 0 = Absolute
1 = Integer
Length of binary number: 0 = 2 byte
1 = 4 byte
7 6 5 4 3 2 1 0

**Result:**

| | |
|---|---|
| A | Changed |
| B | Length of entire ASCII string |
| N, Z | 0 |

**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.

```
 :
 :
LDX#   R 0100        Source address of binary number
STD    R 1046        Store source address in register R1046&
LDX#   R 0200        Destination for ASCII string
                     Number of significant digits
                     Number of decimal digits
LDAA   # $83         Transfer format of ASCII string to accumulator A
                     Format of binary number: 0 => absolute
                     Length of binary number: 1 => 4 byte
LDAB   # %00000001   Transfer format of binary number to accumulator B
CIA                  Routine call
 :
 :
```

```
Absolute long = 56499          =>     " 00056.499"
Absolute long = -23            =>     "-00000.023"
Absolute long = 1000000        =>     " 01000.000"
Absolute long = 100000000             =>     " >>>>>.>>>"
```

# CBCD

## Convert Binary to BCD

| | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 192 - 1180 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 227 - 852 | **error message** | | | | ❍ | | | | | | | | | | |

**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 |
| X | Destination address BCD number |

**Result:**

| | |
|---|---|
| D | Least significant two bytes of the BCD number |
| N, Z | Correspond to the contents of D |

**Example:**

Binary number = 450 => BCD No. = $ 0 0 0 4 5 0

Binary number = 1956 => BCD No. = $ 0 0 1 9 5 6

Binary number = 1000000 => BCD No. = $ 9 9 9 9 9 9

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

```
 :
LDX#    R 0100          Source address of binary number
XGDX                    Load source address after D
LDX#    R 0290          Destination address for BCD number
CBCD
 :
```

# CBIN Convert BCD to Binary

| Execution time in µs | | | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6303 | 112 - 223 | | | | | | | | | | | | | | | |
| | 6809 | 166 258 | | | | | | | | | | | | | | | |

**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. = $ 0 0 0 4 5 0 => binary number = 450

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

BCD No. = $ 9 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#   R 0200        Source address of BCD number
XGDX                 Load source address after D
LDX#   R 3000        Destination address of binary number
CBIN
 :
```

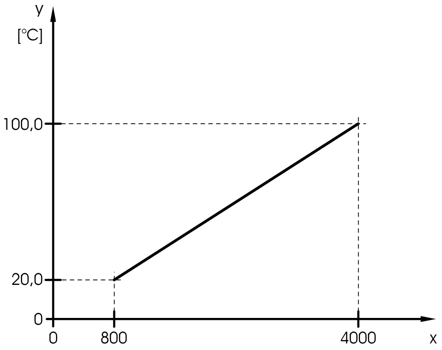
# Conversion: Binary <=> Physical (Scaling)

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:

| | | |
|---|---|---|
| 20.0 °C (4 mA) | => | 800 |
| 100.0 °C (20 mA) | => | 4000 |

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



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

I. $200 = 800k + d \quad => \quad d = 200 - 800k$

II. $1000 = 4000k + d$

II. $1000 = 4000k + 200 - 800k$

$800 = 3200k$

$k = \frac{800}{3200} = 0.25 \quad => \quad d = 200 - 800k = 0$

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 *$y = kx + d$*. Factors **k** and **d** must first be in IEEE format (e.g. calculated with CBPP routine).

**CBP** 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 ($x_1/y_1$) and ($x_2/y_2$). 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:

$$y = kx + d \qquad \Rightarrow \qquad x = \frac{y - d}{k}$$

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 ($x_1/y_1$) and ($x_2/y_2$). Number format (floating point IEEE format or integer word) for the factors can be selected by the user.

## CBPP Calculating Factors *k* and *d* from two Straight Line Points

| **Execution time** | **6303** | 2500 - 6700 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **in µs** | **6809** | 1200 - 4200 | **error messsage** | ❍ |  |  |  |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

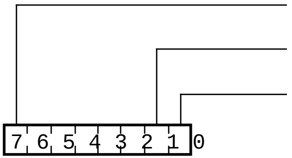
**Function:** This routine has two functions:

1. Calculating factors **k** and **d** (in IEEE format) from two straight line points ($x_1/y_1$) and ($x_2/y_2$). These straight line points can be either in integer word format or in IEEE format.

2. Factors **k** and **d** are transferred to the routine in integer word format or in IEEE format. Factors which are sent in integer word format are converted to IEEE format.

Number format is optional (see diagram on next page).

**Parameters:** R1048& Destination address for factors **k** and **d** (IEEE Format, => 8 Byte)



Function 0 = 1. Function
1 = 2. Function

Data format of x or k: 0 = Integer word
1 = IEEE format

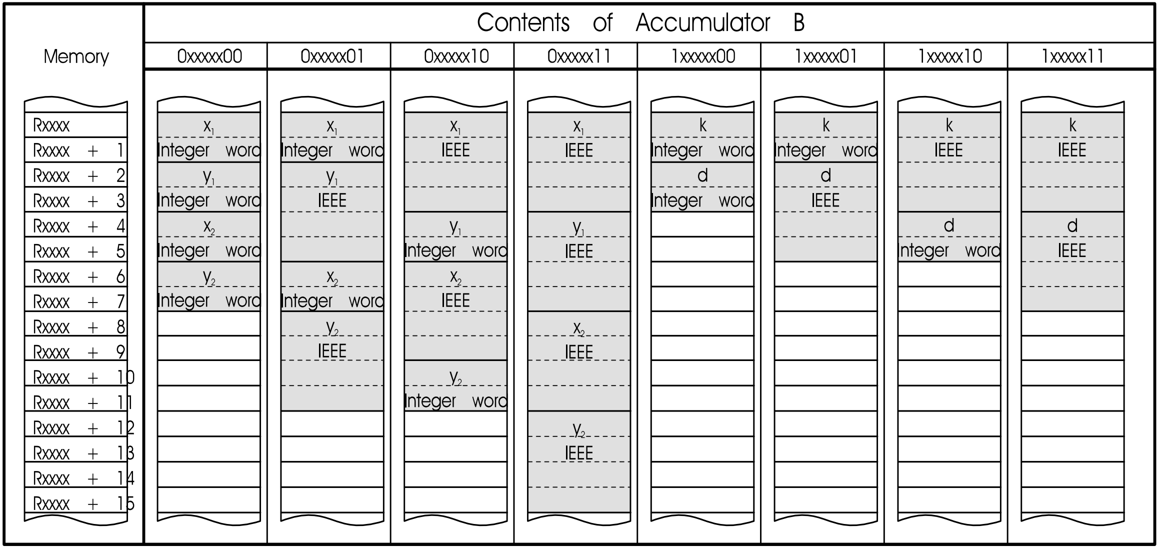
Data format of y or d: 0 = Integer word
1 = IEEE format

B Number format and function selection

X Source address of parameter (x/y or k/d)

**Result:** X, D Changed
N, Z Invalid

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



| Memory | Contents of Accumulator B | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0xxxxx00 | 0xxxxx01 | 0xxxxx10 | 0xxxxx11 | 1xxxxx00 | 1xxxxx01 | 1xxxxx10 | 1xxxxx11 |
| Rxxxx | $x_1$ | $x_1$ | $x_1$ | $x_1$ | k | k | k | k |
| Rxxxx + 1 | Integer word | Integer word | IEEE | IEEE | Integer word | Integer word | IEEE | IEEE |
| Rxxxx + 2 | $y_1$ | $y_1$ | | | d | d | | |
| Rxxxx + 3 | Integer word | IEEE | | | Integer word | IEEE | | |
| Rxxxx + 4 | $x_2$ | | $y_1$ | $y_1$ | | | d | d |
| Rxxxx + 5 | Integer word | | Integer word | IEEE | | | Integer word | IEEE |
| Rxxxx + 6 | $y_2$ | $x_2$ | $x_2$ | | | | | |
| Rxxxx + 7 | Integer word | Integer word | IEEE | | | | | |
| Rxxxx + 8 | | $y_2$ | | $x_2$ | | | | |
| Rxxxx + 9 | | IEEE | | IEEE | | | | |
| Rxxxx + 10 | | | $y_2$ | | | | | |
| Rxxxx + 11 | | | Integer word | | | | | |
| Rxxxx + 12 | | | | $y_2$ | | | | |
| Rxxxx + 13 | | | | IEEE | | | | |
| Rxxxx + 14 | | | | | | | | |
| Rxxxx + 15 | | | | | | | | |

# CBPQ Convert Binary => Physical, Quick

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution 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** | ❍ | | | | | ❍ | ❍ | | | | | ❍ | | |

**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 |

# CPBQ Convert Physical => Binary, Quick

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 780 - 1500 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 580 - 950 | **error message** | ❍ | | | | | ❍ | ❍ | | | | | ❍ | | |

**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 |

# CBP Convert Binary => Physical

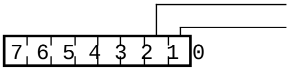
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 3400 - 8300 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | ca. 4000 | **error messsage** | ❍ |  |  |  |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

**Function:** 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 **k** and **d**.
As the calculation of **k** and **d** takes several msec. it is recommended to calculate these factors just once, using the CBPP routine in the initialization program and store them temporarily in registers. By calling the **CBP** routine the calculated factors are put into use.

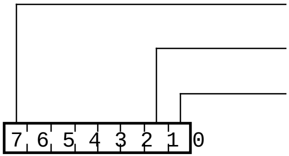
**Parameters:**

R1046& Source address of binary number *x*
R1048& Destination address for physical value *y*
X Source address of straight line points ($x_1/y_1$) and ($x_2/y_2$) or factors **k** and **d**

A



Number format for binary value *x*
Number format for physical value *y*
Number format of *x* and *y*: 0 = Integer word
1 = IEEE format



Straight line definition: 0 = with two straight line points
1 = with factors **k** and **d**
Number format for x or k: 0 = Integer word
1 = IEEE format
Number format for y or d: 0 = Integer word
1 = IEEE format

B Selection for straight line definition and number format

**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.

```
 :
 :
LDX#   R 0100          Load source address of binary number x in index register X
STX    R 1046          Store source address in R1046&
LDX#   R 0120          Load destination address for physical value y in index register X
STX    R 1048          Store source address in R1048&

                ┌──────── Format of x (0 = Integer word)
                │┌─────── Format of y (1 = IEEE)
LDAA   # %00000001     Number formats of x and y

          ┌────────────── Straight line defined by factors k and d
          │     ┌──────── Format of d (1 = IEEE)
          │     │┌─────── Format of k (1 = IEEE)
LDAB   # %10000011     Straight line definition and number format

LDX#   R 0300          Source address for straight line factors k and d
CBP                    Conversion routine call
 :
 :
```

# CPB Convert Physical => Binary

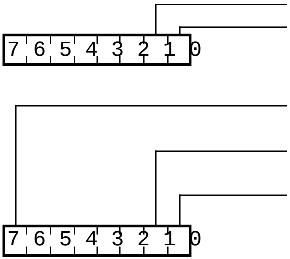
| Execution time | 6303 | 3400 - 8300 | Possible | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| in µs | 6809 | ca. 4000 | error messsage | ❍ |  |  |  |  | ❍ | ❍ |  |  |  |  | ❍ |  |  |

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

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 **CPB** routine the factors previously calculated are set for use.

**Parameters:**

R1046& Source address of physical value *y*
R1048& Destination address for binary number *x*
X Source address of straight line points ($x_1$/$y_1$) and ($x_2$/$y_2$) or factors **k** and **d**



Number format of binary value *x*
Number format of physical value *y*

A: Number format of *x* and *y*: 0 = Integer word
1 = IEEE format

Straight line definition: 0 = With two straight line points
1 = With factors **k** and **d**
Number format of x or k: 0 = Integer word
1 = IEEE format
Number format of y or d: 0 = Integer word
1 = IEEE format

B: Selection for straight line definition and for the number format

| | | |
|---|---|---|
| **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.

```
  :
  :
LDX#    R  0100         Load source address of physical value y in index register X
STX     R  1046         Store source address in R1046&
LDX#    R  0120         Load destination address for the binary value x in index register X
STX     R  1048         Store source address in R1048&
                          Format of x (1 = IEEE)
                          Format of y (1 = IEEE)
LDAA    #  %00000011    Number format of x and y
                          Line defined by factors k and d
                          Format of k (1 = IEEE)
                          Format of d (1 = IEEE)
LDAB    #  %10000011    Line definition and number format
LDX#    R  0300         Source address of factors k and d
CPB                     Call for conversion routine
  :
  :
```

# CIM Convert Inch => Millimeter

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 307 - 472 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 269 - 368 | **error messsage** | | | | | | | | | | | | | ❍ | ❍ |

**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& | Source address for inch number (Integer long) |
| X | Destination address for mm number (Integer long) |
| A | Exponent of inch number (0 through 5) |
| B | Exponent of mm number (0 through 3) |

**Result:**

| | |
|---|---|
| D | The two lower bytes of the mm number |
| N, Z | Correspond 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** | **mm 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

## 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** | | | | | | | | | | | | | ❍ | ❍ |

**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:**

| | | |
|---|---|---|
| **Parameters:** | R1046& | Source address for mm number (Integer long) |
| | X | Destination address for inch number (Integer long) |
| | A | Exponent of mm number (0 through 3) |
| | B | 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).

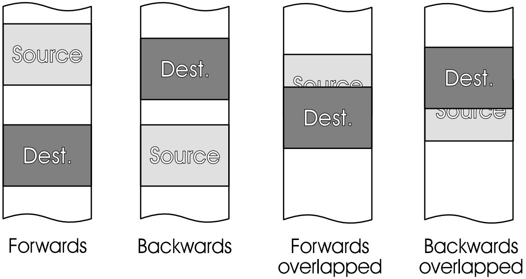
```
  :
  :
LDX#    R  0100         Source address of mm number
STX     R  1046
LDAA    #  002          Exponent of mm number (accuracy = 0.01; => 2 decimal digits)
LDAB    #  004          Exponent of inch number (accuracy = 0.0001; => 4 decimal digits)
LDX#    R  0200         Destination address for inch number
CMI                     Routine call
  :
  :
```

| **Binary number** | **mm number** | **Inch number** | **Binary number** |
|---|---|---|---|
| **in R0100 through R0103** | | | **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

| Execution time in µs | 6303 | see table | Possible error message | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6809 | see table | | | | | | | | | | | | | | | |

**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.



**Example:** The TEXT table will be copied to the area starting from R0100.

```
  :
LDX#    R  0100           Destination address
STX     R  1048
LDAA    #  000
JSR        TEXT           D contains the length of
                          the table and X the start
                          address of the table

FCOP                      Copy the table
```

**Parameters:**

| | |
|---|---|
| R1048& | Destination address for the block |
| D | Length of the block in bytes |
| X | Source address of block |

**Result:**

| | |
|---|---|
| D | Changed |
| N, Z, C | Invalid |

| | 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. | $58 + (\frac{L}{256} + 1)*54 + L*25$ | $112 + L*10,5$ |

# FSMB

## Function Set Memory Byte

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 48 + L*12 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 94 + L*7,5 | **error message** | | | | | | | | | | | | | | |

**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 Memory Word

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **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.

| | |
|---|---|
| **Parameters:** R1052& | 2 byte value |
| D | Length of memory area in words |
| X | 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    #  1000          Load 2 byte value (word) in A
STX    R  1052          Store 2 byte value (word) in R1052
LDD    #  00150         Length of memory range in words (150 words => 300 byte)
LDX#   R  3000          Start address of memory range
FSMW
  :
  :
```

# FCLR

## Function Clear Memory

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time in µs** | **6303** | 48 + L*12 | **Possible error message** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| | **6809** | 94 + L*7,5 | | | | | | | | | | | | | | | |

**Function:** All registers within a defined range (Start address, length of memory range in bytes) are deleted (overwritten with null value).

| | | |
|---|---|---|
| **Parameters:** | D | Length of memory range in bytes |
| | X | Start address of memory range |
| **Result:** | D | Changed |
| | N, Z, C | Invalid |

**Example:** Registers R0100 to R0199 should be deleted

```
 :
 :
LDD     #  00100          Length of memory range
LDX#    R  0100           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 | | | | | | | | | | | | ❍ | | |

**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 — Limitation to High Limit; If OP1 > OP2, then OP2 ⇨ OP1

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 215 - 271 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 205 - 237 | **error message** | | | | | | | | | | | | ❍ | | |

**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

## Limitation to Low Limit; If OP1 < OP2, then OP2 ⇨ OP1

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Execution time** | **6303** | 215 - 271 | **Possible** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **in µs** | **6809** | 205 - 237 | **error message** | | | | | | | | | | | | ❍ | | |

**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

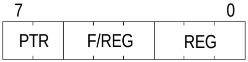
The arithmetic processor is equipped with 8 internal arithmetic registers. These are described as "internal operands". Others are "external operand" numbers which sit in the central processor's data range (R0000 to R7167). It is advisable to differentiate between the following APU instruction types:

- 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**



**PTR** Pointer to external operands

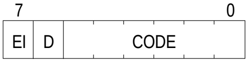
00 = index register X
01 = index register Y [1]
10 = user stack pointer [1]

**F/REG** If the instruction includes two internal operands:
F/REG = Register No. of second operand (OP2 [2])

If the instruction contains an external operand:
F/REG = Format code (see table in section “Number format”)

**REG** Register No. of the first operand (OP1 [2])

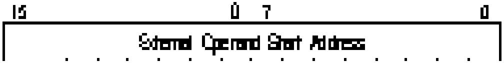
**B**



**EI** Instruction type:
0 = two internal operands
1 = one external operand

**D** Loading / saving internal operands:
0 = loading
1 = saving

**CODE** Instruction code (see “68881 instruction set”)



Index register X, Y or user stack pointer [1]

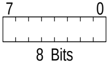
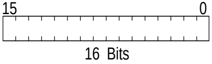
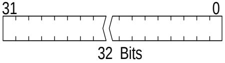
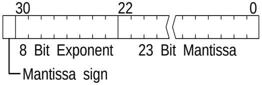
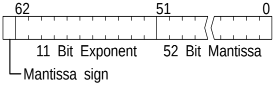
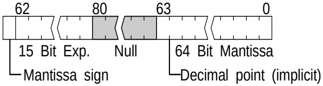
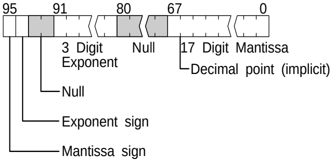
[1] Index register Y and the user stack pointer can be used with the B&R PROgramming SYStem version 5.0 or higher.

[2] OP1 and OP2 are not identical to the operands in mathematic routines.

**APU instruction example:**

```
LDX#    R  2000          Pointer to external operands
LDAA    #  %00100011     Data format "Word Integer"
LDAB    #  %10xxxxxx     An external operand, xxxxxx = Instruction
MAT
```

## 5.1.3. Number Formats

| FORMAT | Description | Length | Format code[1] |
|---|---|---|---|
|  | BYTE INTEGER | 8 Bit / 1 Byte | 110 |
|  | WORD INTEGER | 16 Bit / 2 Byte | 100 |
|  | LONG INTEGER | 32 Bit / 4 Byte | 000 |
|  30<br>22<br>0<br>8 Bit Exponent<br>23 Bit Mantissa<br>Mantissa sign | SINGLE REAL | 32 Bit / 4 Byte | 001 |
|  62<br>51<br>0<br>11 Bit Exponent<br>52 Bit Mantissa<br>Mantissa sign | DOUBLE REAL | 64 Bit / 8 Byte | 101 |
|  62<br>80<br>63<br>0<br>15 Bit Exp.<br>Null<br>64 Bit Mantissa<br>Mantissa sign<br>Decimal point (implicit) | EXTENDED REAL | 96 Bit / 12 Byte | 010 |
|  95<br>91<br>80<br>67<br>0<br>3 Digit Exponent<br>Null<br>17 Digit Mantissa<br>Decimal point (implicit)<br>Null<br>Exponent sign<br>Mantissa sign | PACKED DECIMAL REAL | 96 Bit / 12 Byte | 011 |

# 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 | $e^x$ | OP1 := $e^{OP1}$ | $10 | %010000 |
| ETOXM1 | $e^{x-1}$ | OP1 := $e^{OP1-1}$ | $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 := $\log_{10}$(OP1) | $15 | %010101 |
| LOG2 | logarithm base 2 | OP1 := $\log_2$(OP1) | $16 | %010110 |
| LOGN | logarithm base e | OP1 := ln (OP1) | $14 | %010100 |

| | | | | |
|---|---|---|---|---|
| LOGNP1 | | OP1 := ln (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 ($2^{OP1}$) | $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 := $10^{OP1}$ | $12 | %010010 |
| TWOTOX | | OP1 := $2^{OP1}$ | $11 | %010001 |

The average execution time of APU instructions is approx. 330 µsec.

# 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 | P | $30 | ln(2) | $36 | $10^{8}$ | $3C | $10^{512}$ |
| $0B | $\log_{10}(2)$ | $31 | ln(10) | $37 | $10^{16}$ | $3D | $10^{1024}$ |
| $0C | e | $32 | $10^{0}$ | $38 | $10^{32}$ | $3E | $10^{2048}$ |
| $0D | $\log_{2}(e)$ | $33 | $10^{1}$ | $39 | $10^{64}$ | $3F | $10^{4096}$ |
| $0E | $\log_{10}(e)$ | $34 | $10^{2}$ | $3A | $10^{128}$ | | |
| $0F | 0 | $35 | $10^{4}$ | $3B | $10^{256}$ | | |

**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:

```
LDAA    # %00100001     Register numbers of the two internal operands
LDAB    # %00100011     Instruction code for multiplication = $23
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
LDAB    # %10000000     Instruction code for loading / storage = $00
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:

<table>
<tr><th rowspan="3">C = 0</th><th colspan="2">C = 1</th></tr>
<tr><td colspan="2">no valid character received</td></tr>
<tr><th>Z = 1</th><th>Z = 0</th></tr>
<tr><td>valid character received and stored in accumulator A</td><td>input buffer empty</td><td>Transmission error; Accumulator A contains and error code:<br>7 0<br>RI FE PE OE ▨ ▨ ▨ ▨<br><b>OE</b> 1 = overrun error<br><b>PE</b> 1 = parity error<br><b>FE</b> 1 = framing error<br><b>RI</b> 1 = receiver interrupted</td></tr>
</table>

# 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.

**A**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RI | FE | PE | OE | | | | |

| | |
|---|---|
| **OE** | 1 = overrun error |
| **PE** | 1 = parity error |
| **FE** | 1 = framing error |
| **RI** | 1 = receiver interrupted |

| 15 | 8 | 7 | 0 |
|---|---|---|---|
| free characters in output buffer | | # of characters in input buffer | |

Index register X

**B**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RT | DC | DT | DS | TE | TR | TF | RR |

| | |
|---|---|
| **RR** | 1 = receiver ready |
| **TF** | 1 = transmitter shift register full |
| **TR** | 1 = transmitter ready |
| **TE** | 1 = transmitter empty |
| **DS** | Status of DSR input |
| **DT** | Status of DTR output |
| **DC** | Status of DCD input |
| **RT** | Status of RTS output |

# 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

b. 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:

**A**

| 7 | | | | | 0 |
|---|---|---|---|---|---|
| 0 | 0 | $RS_{PP}$ | RS | BAUD | |

**$RS_{PP}$** [1] 1 = RS485 receiver deactivated for operation with the PP60 with RS232 hardware on an interface convertor with RS485 characteristics.

**RS** 1 = RS485 mode [2]
0 = RS422/RS232 mode

| **Baud** | **Baud rate** | **Baud** | **Baud rate** |
|---|---|---|---|
| 0000 | ---- | 1000 | 1200 Baud |
| 0001 | 50 Baud | 1001 | 1800 Baud |
| 0010 | 75 Baud | 1010 | 2400 Baud |
| 0011 | 110 Baud | 1011 | 3600 Baud |
| 0100 | 135 Baud | 1100 | 4800 Baud |
| 0101 | 150 Baud | 1101 | 7200 Baud |
| 0110 | 300 Baud | 1110 | 9600 Baud |
| 0111 | 600 Baud | 1111 | 19200 Baud |

[1] can be used with system module version 3.8 or higher

[2] In RS485 mode the transmitter is automatically switched off the bus when no character has been transmitted for 300 msec.

**B**

| 7 | | | | | | 0 |
|---|---|---|---|---|---|---|
| RI | TI | SB | P | DA | OE | DB |

| | | | |
|---|---|---|---|
| **DB** | Data bits: | 00 = 5 bit<br>01 = 6 bit | 10 = 7 bit<br>11 = 8 bit |
| **OE** | Parity | 1 = odd | 0 = even |
| **DA** [3] | DSR active | 1 = inactive | 0 = active |
| **P** | Parity on / off | 1 = off | 0 = on |
| **SB** | Stop bits | 1 = 2 stop bits | 0 = 1 stop bit |
| **TI** | Transmitter interrupt | 1 = enabled | 0 = disabled |
| **RI** | Receiver interrupt | 1 = enabled | 0 = disabled |

[3] only used with point to point connections (RS232 <=> RS232)

| 15 8 | 7 0 |
|---|---|
| length of output buffer (1 to 255) | length of input buffer (1 to 255) |

Index register X

# SF - Manual Operation of Handshake Lines and Input / Output Buffers

| | | |
|---|---|---|
| 7 0<br>1 0 0 0 0 0 0 0 | LDAA # $80<br>SF | Set RTS line to low.[1] |
| 7 0<br>1 0 0 0 0 0 0 1 | LDAA # $81<br>SF | Set RTS line to high.[1] |
| 7 0<br>1 0 0 0 0 0 1 0 | LDAA # $82<br>SF | Automatic DTR handling on (after Power ON this is always switched off!). |
| 7 0<br>1 0 0 0 0 0 1 1 | LDAA # $83<br>SF | 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<br>1 0 0 0 0 1 0 0 | LDAA # $84<br>SF | Clear input / output buffer (reset pointer). Busy status is reset unless it was locked with the instruction $83. |
| 7 0<br>1 0 0 0 0 1 0 1 | LDAA # $85<br>SF | Clear input buffer (reset pointer). Busy status is reset unless it was locked with the $83 instruction. |
| 7 0<br>1 0 0 0 0 1 1 0 | LDAA # $86<br>SF | Clear output buffer (reset pointer). |

[1] 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 character which indicates the start of a frame ($02)
LEN ............. Length of the entire frame
NODE ......... Node number 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.
CHK ............ Checksum over the frame. This is computed automatically by the operating system and must not be sent by the user.
ACK ............ Confirmation that a frame was received without errors ($06)

*B&R Standard protocol (MININET)*

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

**A**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | MODE | |

| | | autom.<br>RTS handling | receiver<br>check |
|---|---|---|---|
| MODE | 00 | no | no |
| | 01 | no | yes |
| | 10 | yes | no |
| | 11 | yes | yes |

**B**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| PM | TIME | | | | | | |

**PM** protocol mode on / off (1 = on, 0 = off)
In protocol mode, the RS485 transmitter is automatically switched off after a transmission block (bus capability)

**TIME** transmission pause after every character (in msec.)

| 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start address of user output buffer | | | | | | | | | | | | | | | |

Index register X

# b. Start User Timer 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.

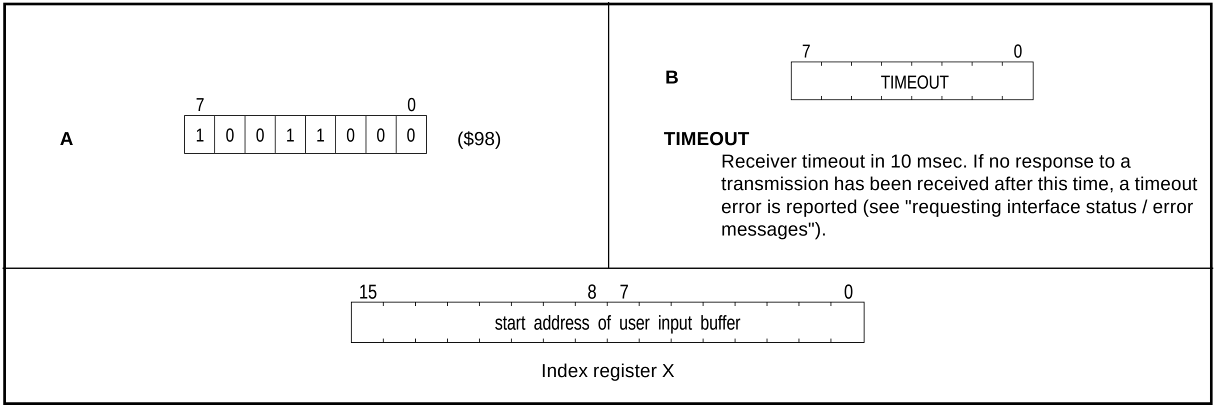
**Example:** The user interface is initialized for block mode without B&R protocol. The user buffer begins at R 3000. Since the receiver does not avail any handshake lines, the automatic RTS handling is turned off and a two msec. pause is inserted after each character. Baudrate, stop bits and parity 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.



**TIMEOUT**
Receiver timeout in 10 msec. If no response to a transmission has been received after this time, a timeout error is reported (see "requesting interface status / error messages").

# SF - Transmit Data in Block Mode

**A**

| 7 | | | | | | | 0 | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | ($99) |

**B**

| 7 | | 0 |
|---|---|---|
| ST | #TA | |

**ST** Start receiver timeout

**#TA** Number of transmission attempts before error message

| 15 | 8 | 7 | 0 |
|---|---|---|---|
| number of data bytes to be sent [1] | | | |

Index register X

[1] 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

|   | 7 |   |   |   |   |   |   | 0 |   |
|---|---|---|---|---|---|---|---|---|---|
| **A** | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | ($99) |

| 15 | 8 | 7 | 0 |
|---|---|---|---|
| size of input buffer | | | |

Index register X

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"
SF
```

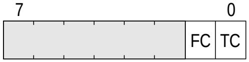
# SF - Request Interface Status ($9F)

This instruction requests information about the state of the interface (only relevant in block or protocol mode).

```
LDAA    #  $9F
SF
```

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:

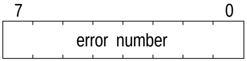
**B**



**FC** 1 = Frame complete (only in protocol mode)
**TC** 1 = Transmission complete

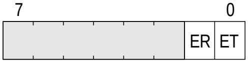
In case of error the carry flag is set after the SF invocation and accumulator A contains an error code:

**A**

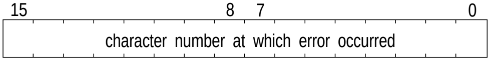


| | | |
|---|---|---|
| **10000001** | $81 | Bus collision (RS485) |
| **10000010** | $82 | Receiver timeout |
| **10000011** | $83 | Checksum error in frame reception |
| **10000100** | $84 | Incorrect frame length (<5) in protocol mode |
| **10000101** | $85 | Input buffer full (RS232) |
| **10000111** | $87 | Input buffer too small (< 32) |
| **10001000** | $88 | Transmission not possible (Busy) for RS232 |

**B**



**ER** 1 = Error in receiving
**ET** 1 = Error in transmitting

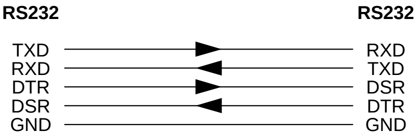


Index register X

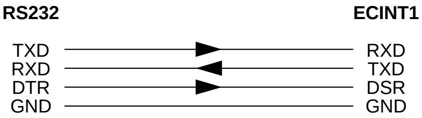
# 5.2.5. HandshakeTechniques

During initialization, the respective interrupts must be set if using transmitter and/or receiver. The following handshake line representations should be differentiated between.

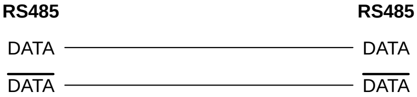
## Point to Point RS232 <=> RS232 (SIB/SOB, Block mode)



## Point to Point RS232 <=> ECINT1 (SIB/SOB, Block mode)



## RS485 (SIB/SOB, Block mode)



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[1], DTR [2] and RTS [2].

# 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.

[1] see “SF - Interface Initialization“

[2] 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, RS485 and RS232 do not affect one another.

|   | 7 |   |   |   |   |   |   | 0 |
|---|---|---|---|---|---|---|---|---|
| **A** | 0 | 0 | $RS_{PP}$ | RS | BAUD |   |   |   |

**$RS_{PP}$** 1 = RS485 receiver deactivated for operation with PP60 and RS232 hardware on an interface convertor with RS485 characteristics (system module version 3.8. or higher)

**RS** 1 = RS485 mode

## Block mode without protocol:

DSR: Not used

DTR: 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 $RS_{PP}$ 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 $RS_{PP}$ bit must be 0.

# 6. APPENDIX

## 6.1. Alphabetical Overview of B&R Mnemonics

| | |
|---|---|
| + | 92 |
| ++B | 95 |
| +B | 94 |
| +D | 96 |
| - | 98 |
| SUB | 99 |
| - -B | 101 |
| -B | 100 |
| -D | 102 |
| = | 52 |
| =B | 53 |
| =D | 54 |
| =R | 55 |
| =S | 57 |
| =Y | 56 |
| A*B | 105 |
| A+B | 97 |
| A-B | 103 |
| ADD | 93 |
| AIM | 84 |
| ANS | 74 |
| AVB | 108 |
| AVS | 75 |
| B | 115 |
| B+R | 104 |
| BB | 116 |
| BNS | 76 |
| BVS | 77 |
| CLA | 167 |
| CLB | 168 |
| CLC | 170 |
| CLI | 172 |
| CLR | 166 |
| CMP | 109 |
| COA | 175 |
| COB | 176 |
| DA | 126 |
| DB | 127 |
| DEC | 125 |
| DK | 177 |
| DR | 128 |
| DS | 129 |
| DXR | 66 |
| EB | 89 |
| EIM | 90 |
| END | 162 |
| EXG | 68 |
| EXO | 88 |
| IA | 121 |
| IB | 122 |
| INC | 120 |
| IR | 123 |
| IS | 124 |
| J+ | 148 |
| J+L | 150 |
| J- | 148 |
| J-L | 150 |
| J<= | 148 |
| J<=L | 150 |
| JC0 | 148 |
| JC0L | 150 |
| K | 174 |
| LAD | 34 |
| LB | 35 |
| LD | 36 |
| LDK | 37 |
| LDL | 38 |
| LE! | 46 |
| LER | 48 |
| LEU | 47 |
| LEY | 49 |
| LR | 39 |
| LRK | 40 |
| LRL | 41 |
| LS | 45 |
| LY | 42 |
| LYK | 43 |
| LYL | 44 |
| MAB | 60 |
| MAC | 62 |
| MBA | 61 |
| MCA | 63 |
| MRS | 65 |
| MSR | 64 |
| NOP | 161 |
| OB | 86 |
| OD | 85 |
| OIM | 87 |
| PRS | 164 |
| PSH | 70 |
| PUL | 72 |
| RET | 158 |
| RLA | 141 |
| RLB | 142 |
| RNS | 78 |
| RRA | 144 |
| RRB | 145 |
| RST | 165 |
| RVS | 79 |
| SEC | 171 |
| SEI | 173 |
| SET | 169 |
| SK0 | 152 |
| SK1 | 154 |
| SL | 132 |
| SLA | 133 |
| SLB | 134 |
| SLD | 135 |
| SLI | 140 |
| SN0 | 148 |
| SN0L | 150 |
| SP< | 148 |
| SP<L | 150 |
| SP> | 148 |
| SP>L | 150 |
| SP0 | 148 |
| SP0L | 150 |
| SPI | 160 |
| SPU | 156 |
| SR | 136 |
| SRA | 137 |
| SRB | 138 |
| SRD | 139 |
| SRE | 143 |
| TFR | 67 |
| TIM | 117 |
| UB | 83 |
| UND | 82 |
| VB | 110 |
| VR | 111 |
| VRK | 112 |
| VY | 113 |
| VYK | 114 |

# 6.2. Alphabetical Overview of Motorola Mnemonics

# 6.3. Alphabetical Overview of Mathematic Routines

# 6.4. Alphabetical Overview of B&R Short Mnemonics

# 6.5. Index