

To all our customers

---

## **Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.**

---

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.  
Customer Support Dept.  
April 1, 2003

MITSUBISHI 8-BIT SINGLE-CHIP MICROCOMPUTER  
740 FAMILY

**740**  
**Family**

Software Manual

keep safety first in your circuit designs !

- Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation or a third party.
- Mitsubishi Electric Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams and charts, represent information on products at the time of publication of these materials, and are subject to change by Mitsubishi Electric Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein.
- Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Mitsubishi Electric Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of JAPAN and/or the country of destination is prohibited.
- Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for further details on these materials or the products contained therein.

REVISION DESCRIPTION LIST

740 Family Software Manual

Rev. No.	Revision Description	Rev. date
1.0	First Edition	970829

## **Preface**

This software manual is for users of the 740 Family. Register structures, addressing modes and instructions are introduced in each section.

The enhanced instruction set with enhanced data and memory operations enable efficient programming. Please refer to the "USER'S MANUAL" appropriate for the hardware device or the development support tools used.

---

# Table of contents

<b>CHAPTER 1. OVERVIEW .....</b>	<b>1</b>
<b>CHAPTER 2. CENTRAL PROCESSING UNIT (CPU).....</b>	<b>2</b>
2.1 Accumulator (A) .....	2
2.2 Index Register X (X), Index Register Y (Y) .....	2
2.3 Stack Pointer (S) .....	3
2.4 Program Counter (PC) .....	4
2.5 Processor Status Register (PS).....	4
<b>CHAPTER 3. INSTRUCTIONS .....</b>	<b>6</b>
3.1 Addressing Mode .....	6
3.2 Instruction Set .....	26
3.2.1 Data transfer instructions.....	26
3.2.2 Operating instruction .....	27
3.2.3 Bit managing instructions .....	28
3.2.4 Flag setting instructions .....	28
3.2.5 Jump, Branch and Return instructions.....	28
3.2.6 Interrupt instruction (Break instruction) .....	29
3.2.7 Special instructions.....	29
3.2.8 Other instruction .....	29
3.3 Description of instructions .....	30
<b>CHAPTER 4. NOTES ON USE .....</b>	<b>102</b>
4.1 Notes on interrupts .....	102
4.1.1 Setting for interrupt request bit and interrupt enable bit.....	102
4.1.2 Switching of detection edge .....	102
4.1.3 Distinction of interrupt request bit .....	103
4.2 Notes on programming .....	104
4.2.1 Processor Status Register .....	104
4.2.2 BRK instruction .....	105
4.2.3 Decimal calculations .....	105
4.2.4 JMP instruction .....	106
<b>APPENDIX 1. Instruction Cycles in each Addressing Mode .....</b>	<b>107</b>
<b>APPENDIX 2. 740 Family Machine Language Instruction Table .....</b>	<b>173</b>
<b>APPENDIX 3. 740 Family list of Instruction Codes .....</b>	<b>179</b>

**<Addressing Mode>**

Immediate .....	7	Special Page .....	21
Accumulator .....	8	Zero Page Bit .....	22
Zero Page .....	9	Accumulator Bit .....	23
Zero Page X .....	10	Accumulator Bit Relative .....	24
Zero Page Y .....	11	Zero Page Bit Relative .....	25
Absolute .....	12		
Absolute X .....	13		
Absolute Y .....	14		
Implied .....	15		
Relative .....	16		
Indirect X .....	17		
Indirect Y .....	18		
Indirect Absolute .....	19		
Zero Page Indirect .....	20		

**<Instructions>**

ADC .....	31	CLI .....	50	LDX .....	69	SEI .....	88
AND .....	32	CLT .....	51	LDY .....	70	SET .....	89
ASL .....	33	CLV .....	52	LSR .....	71	STA .....	90
BBC .....	34	CMP .....	53	MUL .....	72	STP .....	91
BBS .....	35	COM .....	54	NOP .....	73	STX .....	92
BCC .....	36	CPX .....	55	ORA .....	74	STY .....	93
BCS .....	37	CPY .....	56	PHA .....	75	TAX .....	94
BEQ .....	38	DEC .....	57	PHP .....	76	TAY .....	95
BIT .....	39	DEX .....	58	PLA .....	77	TST .....	96
BMI .....	40	DEY .....	59	PLP .....	78	TSX .....	97
BNE .....	41	DIV .....	60	ROL .....	79	TXA .....	98
BPL .....	42	EOR .....	61	ROR .....	80	TXS .....	99
BRA .....	43	INC .....	62	RRF .....	81	TYA .....	100
BRK .....	44	INX .....	63	RTI .....	82	WIT .....	101
BVC .....	45	INY .....	64	RTS .....	83		
BVS .....	46	JMP .....	65	SBC .....	84		
CLB .....	47	JSR .....	66	SEB .....	85		
CLC .....	48	LDA .....	67	SEC .....	86		
CLD .....	49	LDM .....	68	SED .....	87		

# OVERVIEW

---

## 1. OVERVIEW

The distinctive features of the CMOS 8-bit microcomputers 740 Family's software are described below:

- 1) An efficient instruction set and many addressing modes allow the effective use of ROM.
- 2) The same bit management, test, and branch instructions can be performed on the Accumulator, memory, or I/O area.
- 3) Multiple interrupts with separate interrupt vectors allow servicing of different non-periodic events.
- 4) Byte processing and table referencing can be easily performed using the index addressing mode.
- 5) Decimal mode needs no software correction for proper decimal operation.
- 6) The Accumulator does not need to be used in operations using memory and/or I/O.



# CENTRAL PROCESSING UNIT

Accumulator (A)  
Index Register X (X), Index Register Y (Y)

## 2. CENTRAL PROCESSING UNIT (CPU)

Six main registers are built into the CPU of the 740 Family. The Program Counter (PC) is a sixteen-bit register; however, the Accumulator (A), Index Register X (X), Index Register Y (Y), Stack Pointer (S) and Processor Status Register (PS) are eight-bit registers.

☞ Except for the I flag, the contents of these registers are indeterminate after a hardware reset; therefore, initialization is required with some programs (immediately after reset the I flag is set to "1").

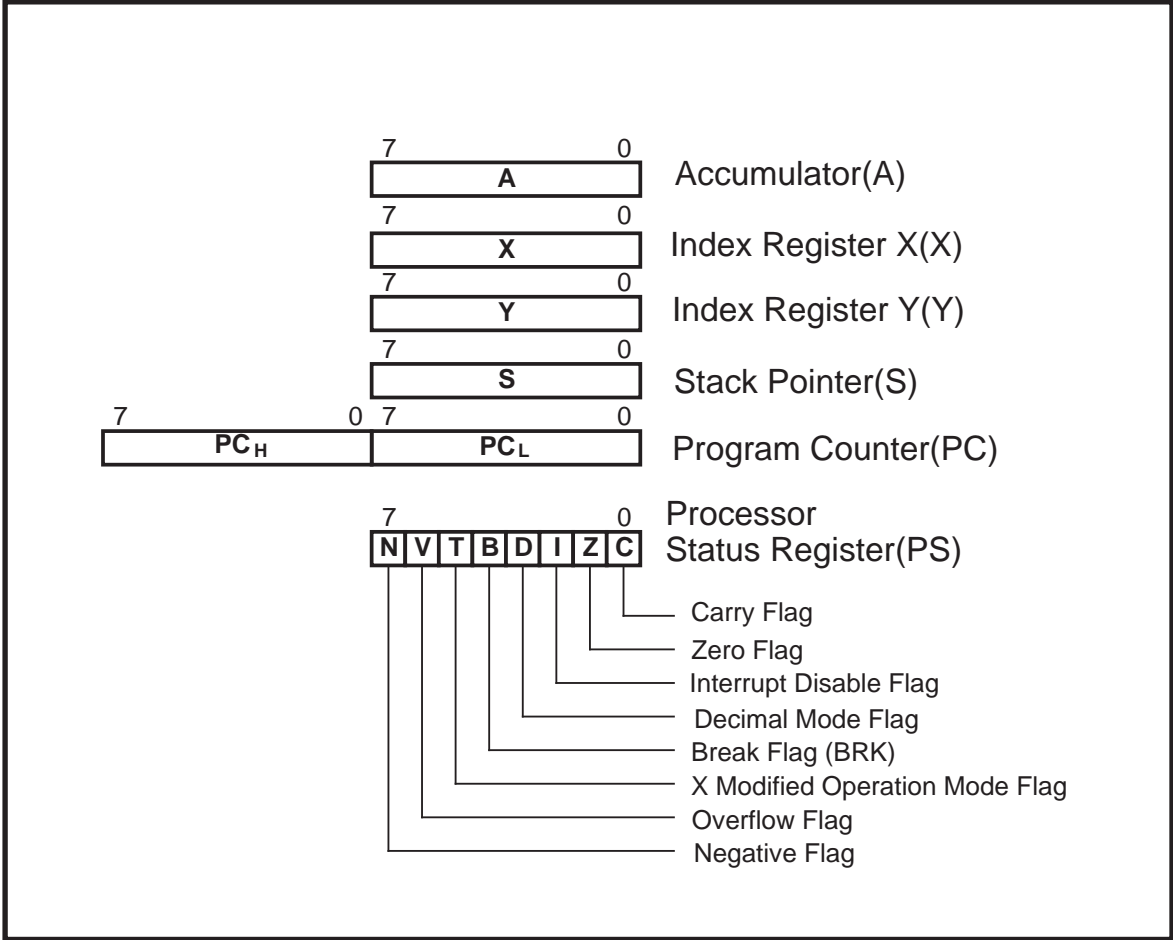


Fig.2.1.1 Register Configuration

### 2.1 Accumulator (A)

The Accumulator, an eight-bit register, is the main register of the microcomputer. This general-purpose register is used most frequently for arithmetic operations, data transfer, temporary memory, conditional judgments, etc.

### 2.2 Index Register X (X), Index Register Y (Y)

The 740 Family has an Index Register X and an Index Register Y, both of which are eight-bit registers.

When using addressing modes which use these index registers, the address, which is added the contents of Index Register to the address specified with operand, is accessed. These modes are extremely effective for referencing subroutine and memory tables.

The index registers also have increment, decrement, compare, and data transfer functions; therefore, these registers can be used as simple accumulators.

# CENTRAL PROCESSING UNIT

## Stack Pointer (S)

---

### 2.3 Stack Pointer (S)

The Stack Pointer is an eight-bit register used for generating interrupts and calling subroutines. When an interrupt is received, the following procedure is performed automatically in the indicated sequence:

- (1) The contents of the high-order eight bits of the Program Counter (PCH) are saved to an address using the Stack Pointer contents for the low-order eight bits of the address.
- (2) The Stack Pointer contents are decremented by 1.
- (3) The contents of the low-order eight bits of the Program Counter (PCL) are saved to an address using the Stack Pointer Contents for the low-order eight bits of the address.
- (4) The Stack Pointer contents are decremented by 1.
- (5) The contents of the Processor Status Register (PS) are saved to an address using the Stack Pointer contents for the low-order eight bits of the address.
- (6) The Stack Pointer contents are decremented by 1.

The Processor Status Register is not saved when calling subroutines (items (5) and (6) above are not executed). The Processor Status Register is saved by executing the PHP instruction in software.

To prevent data loss when generating interrupts and calling subroutines, it is necessary to save other registers as well. This is done by executing the proper instruction in software while in the interrupt service routine or subroutine.

The high-order eight bits of the address are determined by the Stack Page Selection Bit.

For example, the PHA instruction is executed to save the contents of the Accumulator. Executing the PHA instruction saves the Accumulator contents to an address using the Stack Pointer contents as the low-order eight bits of the address.

The RTI instruction is executed to return from an interrupt routine.

When the RTI instruction is executed, the following procedure is performed automatically in sequence.

- (1) The Stack Pointer contents are incremented by 1.
- (2) The contents of an address using the Stack Pointer contents as the low-order eight bits of the address is returned to the Processor Status Register (PS).
- (3) The Stack Pointer contents are incremented by 1.
- (4) The contents of an address using the Stack Pointer as the low-order eight bits of the address is returned to the low-order eight bits of the Program Counter (PCL).
- (5) The Stack Pointer contents are incremented by 1.
- (6) The contents of an address using the Stack Pointer as the low-order eight bits of the address is returned to the high-order eight bits of the Program Counter (PCH).

Steps (1) and (2) are not performed when returning from a subroutine using the RTS instruction. The Processor Status Register should be restored before returning from a subroutine by using the PLP instruction. The Accumulator should be restored before returning from a subroutine or an interrupt servicing routine by using the PLA instruction.

The PLA and PLP instructions increment the Stack Pointer by 1 and return the contents of an address stored in the Stack Pointer to the Accumulator or Processor Status Register, respectively.

☞ Saving data in the stack area gradually fills the RAM area with saved data; therefore, caution must be exercised concerning the depth of interrupt levels and subroutine nesting.

# CENTRAL PROCESSING UNIT

## Program Counter (PC) Processor Status Register (PS)

---

### 2.4 Program Counter (PC)

The Program Counter is a sixteen-bit counter consisting of PCH and PCL, which are each eight-bit registers. The contents of the Program Counter indicates the address which an instruction to be executed next is stored.

The 740 Family uses a stored program system; to start a new operation it is necessary to transfer the instruction and relevant data from memory to the CPU.

Normally the Program Counter is used to indicate the next memory address. After each instruction is executed, the next instruction required is read. This cycle is repeated until the program is finished.

☞ The control of the Program Counter of the 740 Family is almost fully automatic. However, caution must be exercised to avoid differences between program flow and Program Counter contents when using the Stack Pointer or directly altering the contents of the Program Counter.

### 2.5 Processor Status Register (PS)

The Processor Status Register is an eight-bit register consisting of 5 flags which indicate the status of arithmetic operations and 3 flags which determine operation.

Each of these flags is described below. Table 2.5.1 lists the instructions to set/clear each flag. Refer to the section "Appendix 2 MACHINE LANGUAGE INSTRUCTION TABLE" or "3.3 INSTRUCTIONS" for details on when these flags are altered.

[ Carry flag C ]----- Bit 0

This flag stores any carry or borrow from the Arithmetic Logic Unit (ALU) after an arithmetic operation and is also changed by the Shift or Rotate instruction.

This flag is set by the SEC instruction and is cleared by the CLC instruction.

[ Zero flag Z ]----- Bit 1

This flag is set when the result of an arithmetic operation or data transfer is "0" and is cleared by any other result.

[ Interrupt disable flag I ]----- Bit 2

This flag disables interrupts when it is set to "1." This flag immediately becomes "1" when an interrupt is received.

This flag is set by the SEI instruction and is cleared by the CLI instruction.

[ Decimal mode flag D ]----- Bit 3

This flag determines whether addition and subtraction are performed in binary or decimal notation. Addition and subtraction are performed in binary notation when this flag is set to "0" and as a 2-digit, 1-word decimal numeral when set to "1." Decimal notation correction is performed automatically at this time.

This flag is set by the SED instruction and is cleared by the CLD instruction.

Only the ADC and SBC instructions are used for decimal arithmetic operations.

Note that the flags N, V and Z are invalid when decimal arithmetic operations are performed by these instructions.

[ Break flag B ]----- Bit 4

This flag determines whether an interrupt was generated with the BRK instruction. When a BRK instruction interrupt occurs, the flag B is set to "1" and saved to the stack; for all other interrupts the flag is set to "0" and saved to the stack.

# CENTRAL PROCESSING UNIT

## Processor Status Register (PS)

[ X modified operation mode flag T ] ----- Bit 5

This flag determines whether arithmetic operations are performed via the Accumulator or directly on a memory location. When the flag is set to "0", arithmetic operations are performed between the Accumulator and memory. When "1", arithmetic operations are performed directly on a memory location.

This flag is set by the SET instruction and is cleared by the CLT instruction.

(1) When the T flag = 0

$$A \leftarrow A * M2$$

\* : indicates an arithmetic operation

A: accumulator contents

M2: contents of a memory location specified by the addressing mode of the arithmetic operation

(2) When the T flag = 1

$$M1 \leftarrow M1 * M2$$

\* : indicates arithmetic operation

M1: contents of a memory location, designated by the contents of Index Register X.

M2: contents of a memory location specified by the addressing mode of arithmetic operation.

[ Overflow flag V ] ----- Bit 6

This flag is set to "1" when an overflow occurs as a result of a signed arithmetic operation. An overflow occurs when the result of an addition or subtraction exceeds +127 (7F<sub>16</sub>) or -128 (80<sub>16</sub>) respectively.

The CLV instruction clears the Overflow Flag. There is no set instruction.

The overflow flag is also set during the BIT instruction when bit 6 of the value being tested is "1."

☞ Overflows do not occur when the result of an addition or subtraction is equal to or smaller than the above numerical values, or for additions involving values with different signs.

[ Negative flag N ] ----- Bit 7

This flag is set to match the sign bit (bit 7) of the result of a data or arithmetic operation.

This flag can be used to determine whether the results of arithmetic operations are positive or negative, and also to perform a simple bit test.

**Table 2.5.1** Instructions to set/clear each flag of processor status register

	Flag C	Flag Z	Flag I	Flag D	Flag B	Flag T	Flag V	Flag N
Set instruction	SEC	---	SEI	SED	---	SET	---	---
Clear instruction	CLC	---	CLI	CLD	---	CLT	CLV	---

# INSTRUCTIONS

## Addressing mode

### 3. INSTRUCTIONS

#### 3.1 Addressing Mode

The 740 Family has 19 addressing modes and a powerful memory access capability. When extracting data required for arithmetic and logic operations from memory or when storing the results of such operations in memory, a memory address must be specified. The specification of the memory address is called addressing. The data required for addressing and the registers involved are described below. The 740 Family instructions can be classified into three kinds, by the number of bytes required in program memory for the instruction: 1-byte, 2-byte and 3-byte instructions. In each case, the first byte is known as the “Op-Code (operation code)” which forms the basis of the instruction. The second or third byte is called the “operand” which affects the addressing. The contents of index registers X and Y can also effect the addressing.

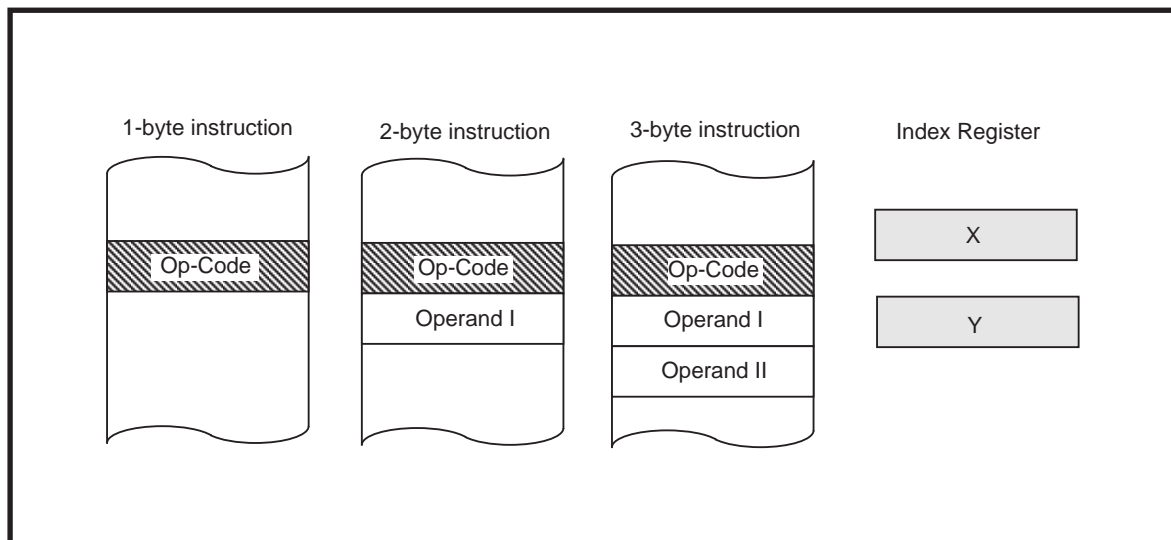


Fig.3.1.1 Byte Structure of Instructions

Although there are many addressing modes, there is always a particular memory location specified. What differs is whether the operand, or the index register contents, or a combination of both should be used to specify the memory or jump destination. Based on these 3 types of instructions, the range of variation is increased and operation is enhanced by combinations of the bit operation instructions, jump instruction, and arithmetic instructions.

As for 1-byte instruction, an accumulator or a register is specified, so that the instruction does not have “operand,” which specify memory.

## INSTRUCTIONS

# Immediate

Addressing mode

Addressing mode : **Immediate**

Function : **Specifies the Operand as the data for the instruction.**

Instructions : **ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, SBC**

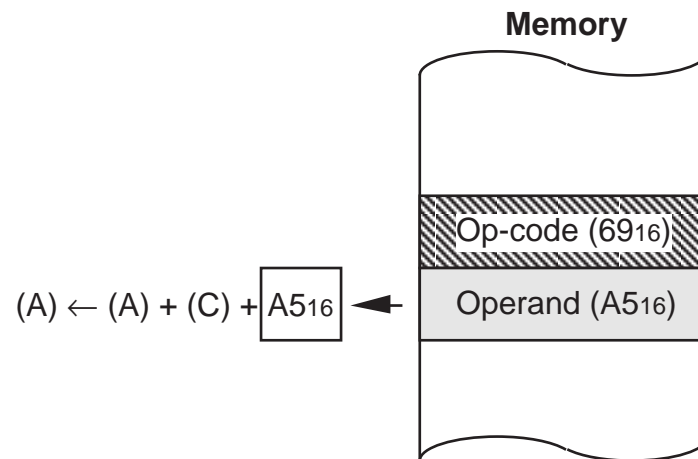
Example : Mnemonic

$\Delta$ ADC $\Delta$ # $\$A5$

Machine code

$69_{16}$   $A5_{16}$

┆  
This symbol(#) indicates the Immediate addressing mode.



## INSTRUCTIONS

# Accumulator

Addressing mode

---

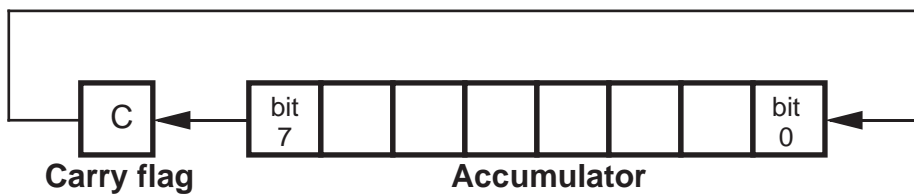
Addressing mode : **Accumulator**

Function : **Specifies the contents of the Accumulator as the data for the instruction.**

Instructions : **ASL, DEC, INC, LSR, ROL, ROR**

Example : Mnemonic  
**ΔROLΔA**

Machine code  
**2A<sub>16</sub>**



# Zero Page

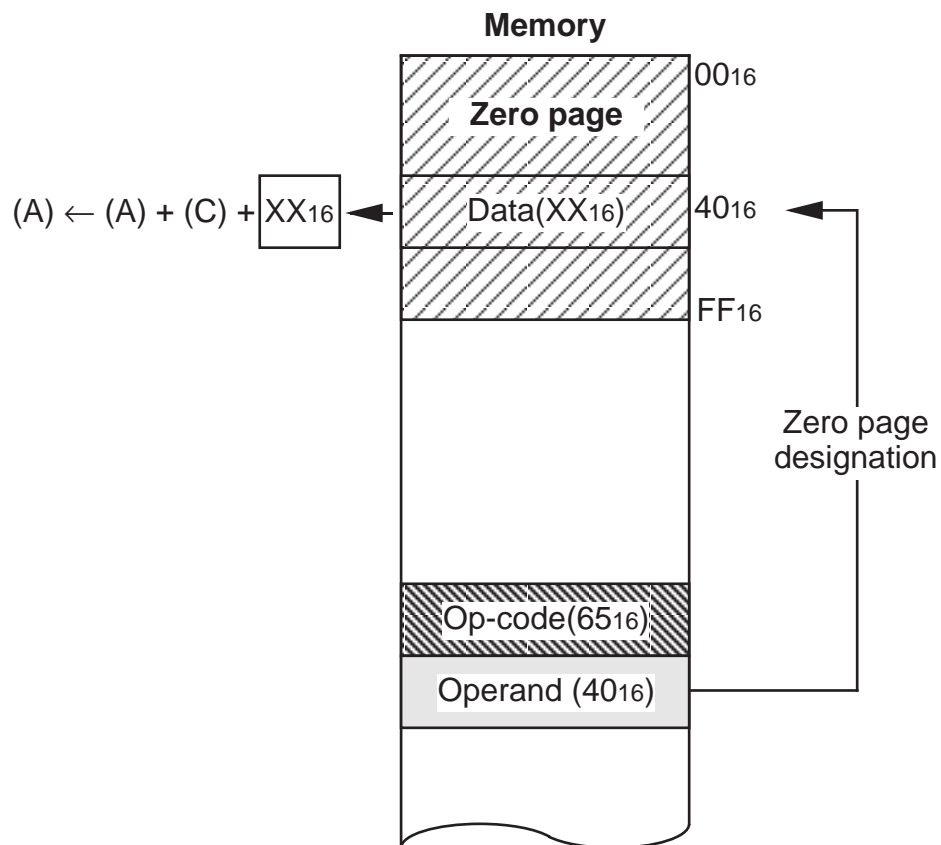
Addressing mode

Addressing mode : **Zero Page**

Function : Specifies the contents in a Zero Page memory location as the data for the instruction. The address in the Zero Page memory location is determined by using Operand as the low-order byte of the address and  $00_{16}$  as the high-order byte.

Instructions : **ADC, AND, ASL, BIT, CMP, COM, CPX, CPY, DEC, EOR, INC, LDA, LDM, LDX, LDY, LSR, ORA, ROL, ROR, RRF, SBC, STA, STX, STY, TST**

Example : Mnemonic	Machine code
$\Delta\text{ADC}\Delta\$40$	$65_{16} 40_{16}$





# Zero Page X

Addressing mode : Zero Page X

Function : Specified the contents in a Zero Page memory location as the data for the instruction. The address in the Zero Page memory location is determined by the following:

- (a) Operand and the Index Register X are added. (If as a result of this addition a carry occurs, it is ignored.)
- (b) The result of the addition is used as the low-order byte of the address and 00<sub>16</sub> as the high-order byte.

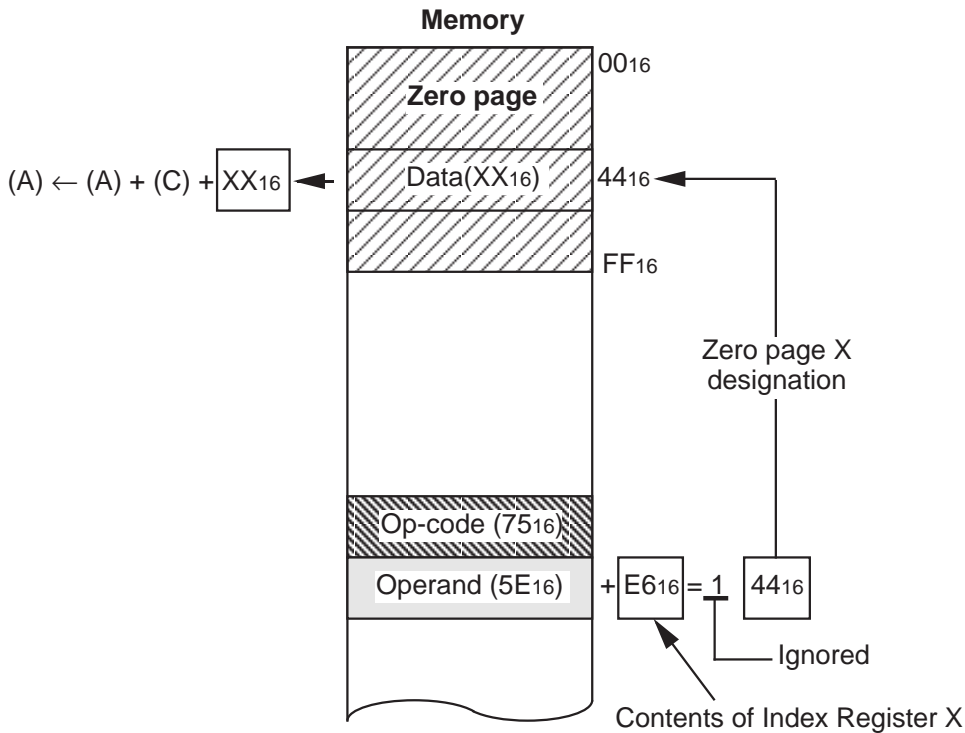
Instructions : ADC, AND, ASL, CMP, DEC, DIV, EOR, INC, LDA, LDY, LSR, MUL, ORA, ROL, ROR, SBC, STA, STY

Example : Mnemonic

$\Delta$ ADC $\Delta$ \$5E,X

Machine code

75<sub>16</sub> 5E<sub>16</sub>



# Zero Page Y

Addressing mode : **Zero Page Y**

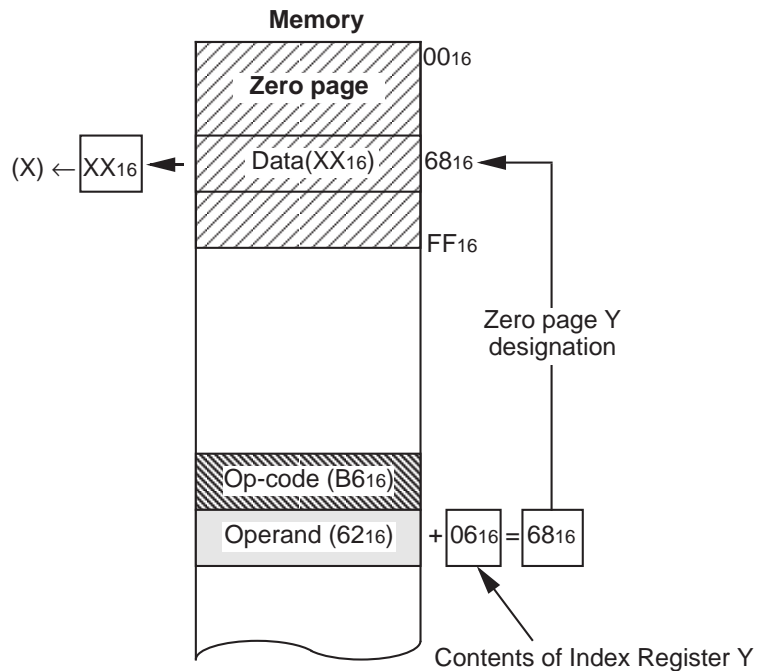
Function : **Specifies the contents in a Zero Page memory location as the data for the instruction. The address in the Zero Page memory location is determined by the following:**

- (a) **Operand and the Index Register Y are added (if as a result of this addition a carry occurs, it is ignored).**
- (b) **The result of the addition is used as the low-order byte of the address and 00<sub>16</sub> as the high-order byte.**

Instructions : **LDX, STX**

Example : Mnemonic  
 $\Delta$ **LDX** $\Delta$ **\$62,Y**

Machine code  
**B6<sub>16</sub> 62<sub>16</sub>**



# Absolute

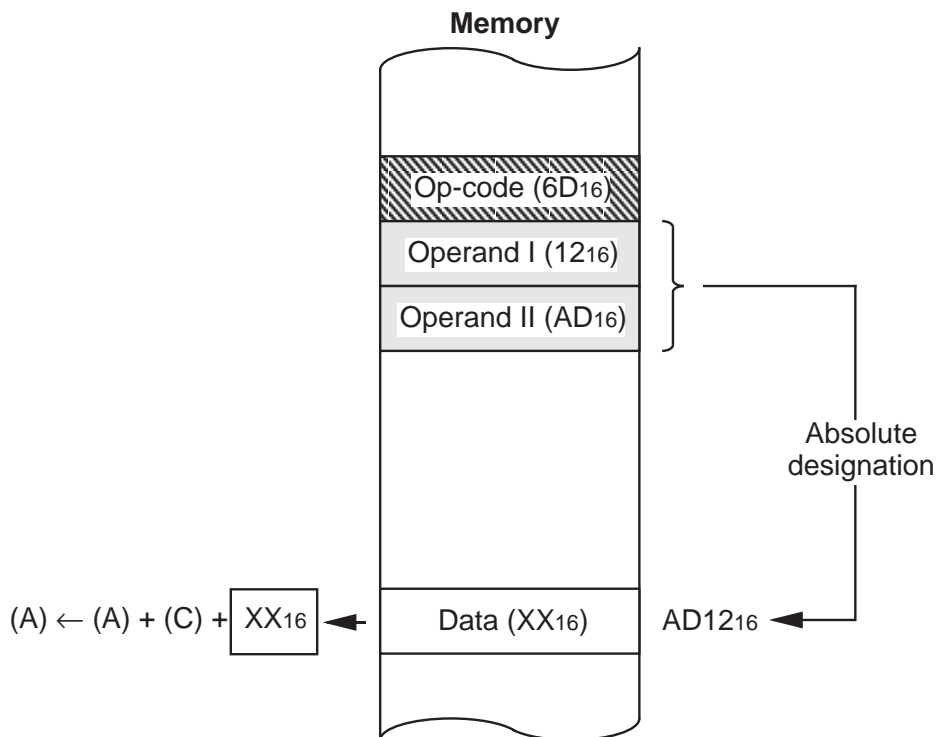
Addressing mode : **Absolute**

Function : **Specifies the contents in a memory location as the data for the instruction. The address in the memory location is determined by using Operand I as the low-order byte of the address and Operand II as the high-order byte.**

Instructions : **ADC, AND, ASL, BIT, CMP, CPX, CPY, DEC, EOR, INC, JMP, JSR, LDA, LDX, LDY, LSR, ORA, ROL, ROR, SBC, STA, STX, STY**

Example : Mnemonic  
 $\Delta$ ADC $\Delta$ \$AD12

Machine code  
 6D<sub>16</sub> 12<sub>16</sub> AD<sub>16</sub>



# Absolute X

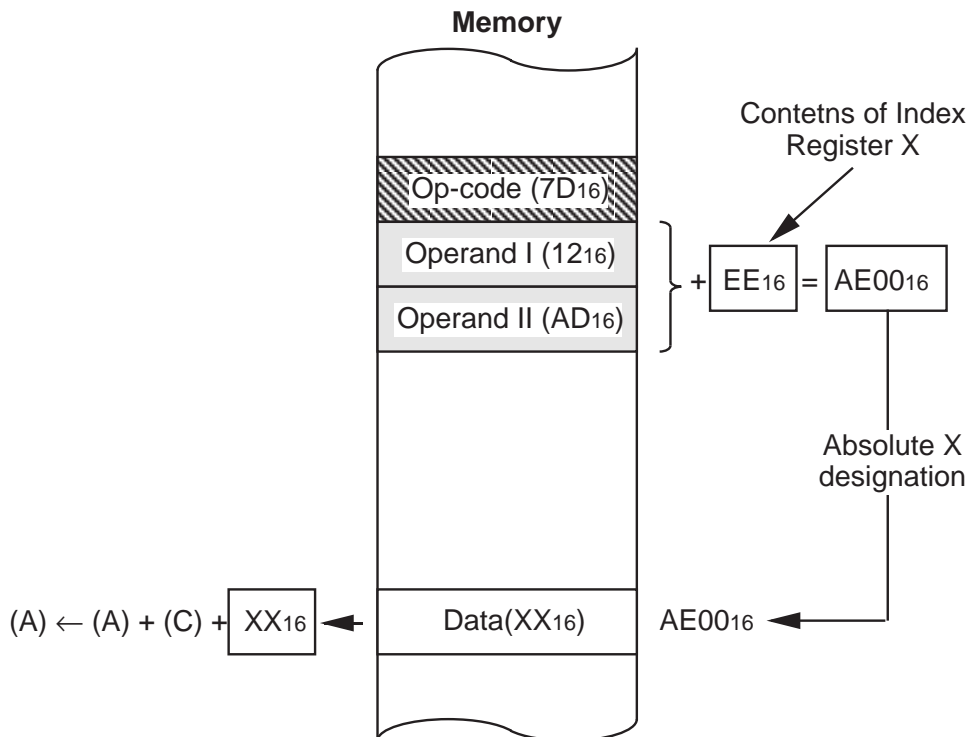
Addressing mode : **Absolute X**

Function : **Specifies the contents in a memory location as the data for the instruction. The address in the memory location is determined by the following:**

- (a) **Operand I is used as the low-order byte of an address, Operand II as the high-order byte.**
- (b) **Index Register X is added to the address above. The result is the address in the memory location.**

Instructions : **ADC, AND, ASL, CMP, DEC, EOR, INC, LDA, LDY, LSR, ORA, ROL, ROR, SBC, STA**

Example : Mnemonic	Machine code
$\Delta$ ADC $\Delta$ \$AD12, X	7D <sub>16</sub> 12 <sub>16</sub> AD <sub>16</sub>



# Absolute Y

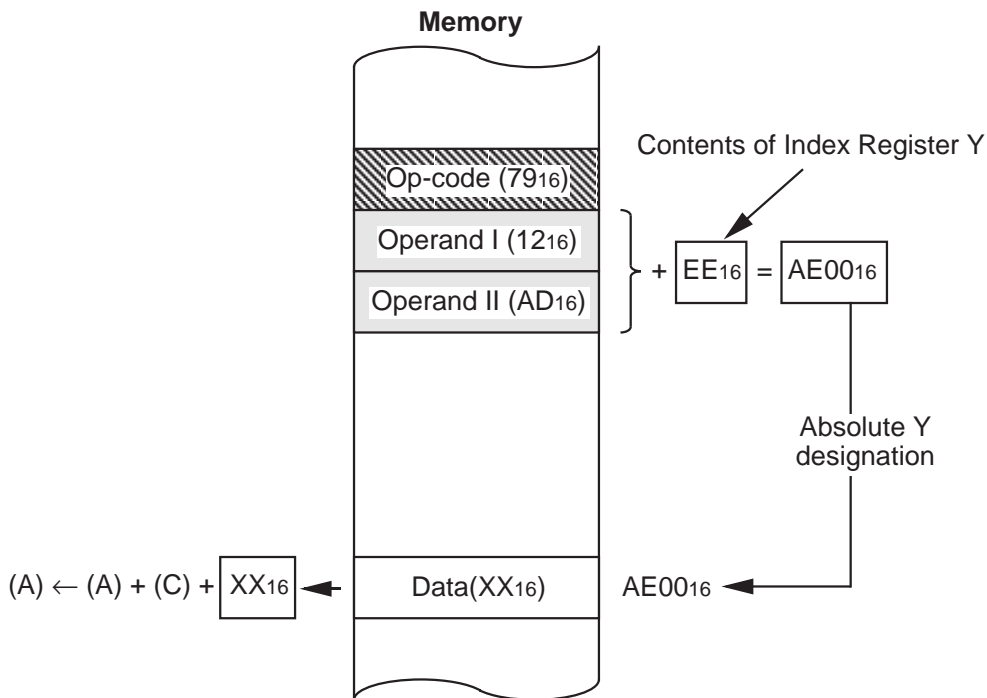
Addressing mode : **Absolute Y**

Function : **Specifies the contents in a memory location as the data for the instruction. The address in the memory location is determined by the following:**

- (a) **Operand I is used as the low-order byte of an address, Operand II as the high-order byte.**
- (b) **Index Register Y is added to the address above. The result is the address in the memory location.**

Instructions : **ADC, AND, CMP, EOR, LDA, LDX, ORA, SBC, STA**

Example : Mnemonics	Machine code
$\Delta$ ADC $\Delta$ \$AD12, Y	79 <sub>16</sub> 12 <sub>16</sub> AD <sub>16</sub>



# Implied

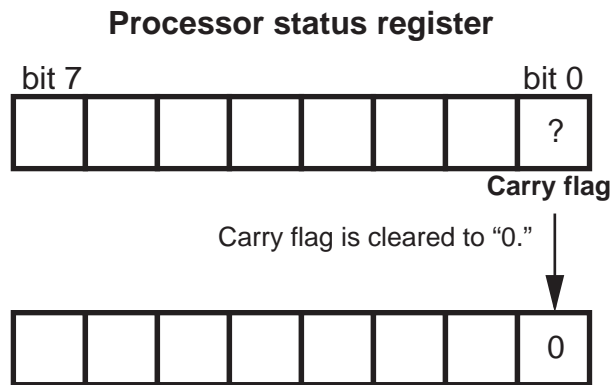
Addressing mode : **Implied**

Function : **Operates on a given register or the Accumulator, but the address is always inherent in the instruction.**

Instructions : **BRK, CLC, CLD, CLI, CLT, CLV, DEX, DEY, INX, INY, NOP, PHA, PHP, PLA, PLP, RTI, RTS, SEC, SED, SEI, SET, STP, TAX, TAY, TSX, TXA, TXS, TYA, WIT**

Example : Mnemonic  
**ΔCLC**

Machine code  
**1816**



# Relative

Addressing mode : **Relative**

Function : Specifies the address in a memory location where the next Op-Code is located.

When the branch condition is satisfied, Operand and the Program Counter are added. The result of this addition is the address in the memory location.

When the branch condition is not satisfied, the next instruction is executed.

Instructions : **BCC, BCS, BEQ, BMI, BNE, BPL, BRA, BVC, BVS**

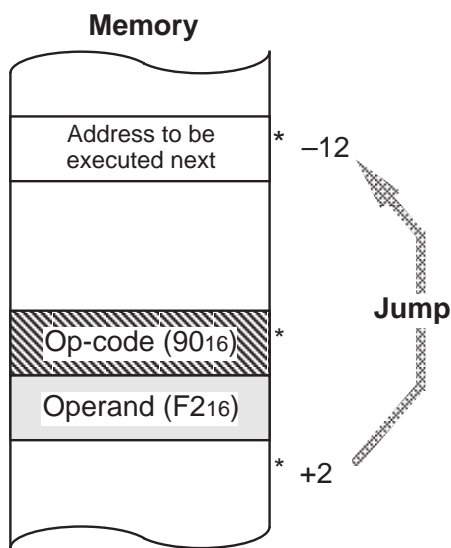
Example : Mnemonic

$\Delta$ **BCC** $\Delta$  $\ast$ -12  
 └─ Decimal

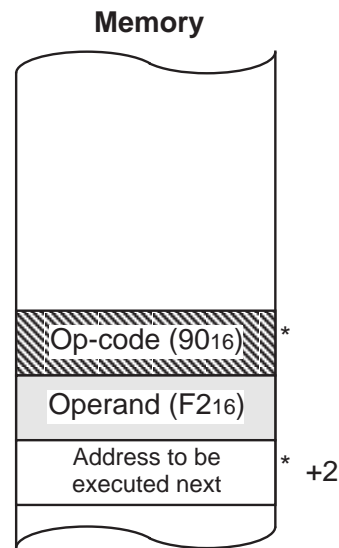
Machine code

**90<sub>16</sub> F2<sub>16</sub>**

*When the carry flag is cleared, jumps to address  $\ast$ -12.*



*When the carry flag is set, goes to address  $\ast$ +2.*



# Indirect X

Addressing mode : Indirect X

Function : Specifies the contents in a memory location as the data for the instruction. The address in the memory location is determined by the following:

- (a) A Zero Page memory location is determined by the adding the Operand and Index Register X (if as a result of this addition a carry occurs, it is ignored).
- (b) The result of the addition is used as the low-order byte of an address in the Zero Page memory location and 00<sub>16</sub> as the high-order byte.
- (c) The contents of the address in the Zero Page memory location is used as the low-order byte of the address in the memory location.
- (d) The next Zero Page memory location is used as the high-order byte of the address in the memory location.

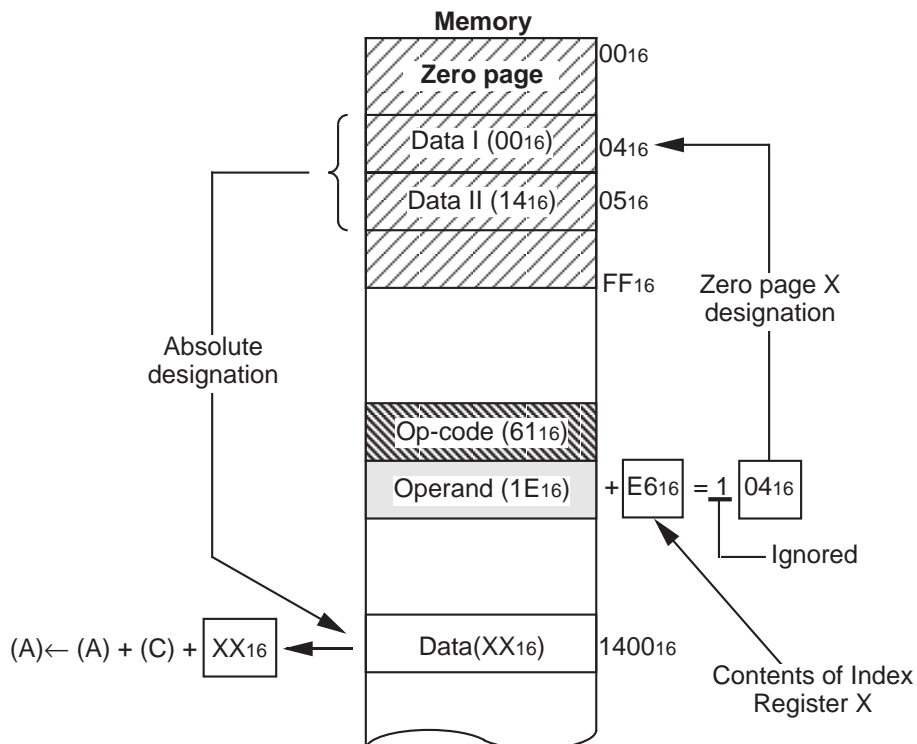
Instructions : **ADC, AND, CMP, EOR, LDA, ORA, SBC, STA**

Example : Mnemonic

$\Delta$ ADC $\Delta$ (\$1E,X)

Machine code

61<sub>16</sub> 1E<sub>16</sub>



Assuming that "00<sub>16</sub>" for Data I, and "14<sub>16</sub>" for Data II are stored in advance.



# Indirect Y

Addressing mode : Indirect Y

Function : Specifies the contents in a memory location as the data for the instruction. The address in the memory location is determined by the following:

- (a) The Operand is used the low-order byte of an address in the Zero Page memory location and  $00_{16}$  of the high-order byte.
- (b) The contents of the address in the Zero Page memory location is used as the low-order byte of an address. The next Zero Page memory location is used as the high-order byte.
- (c) The Index Register Y is added to the address in Step b. The result of this addition is the address in the memory location.

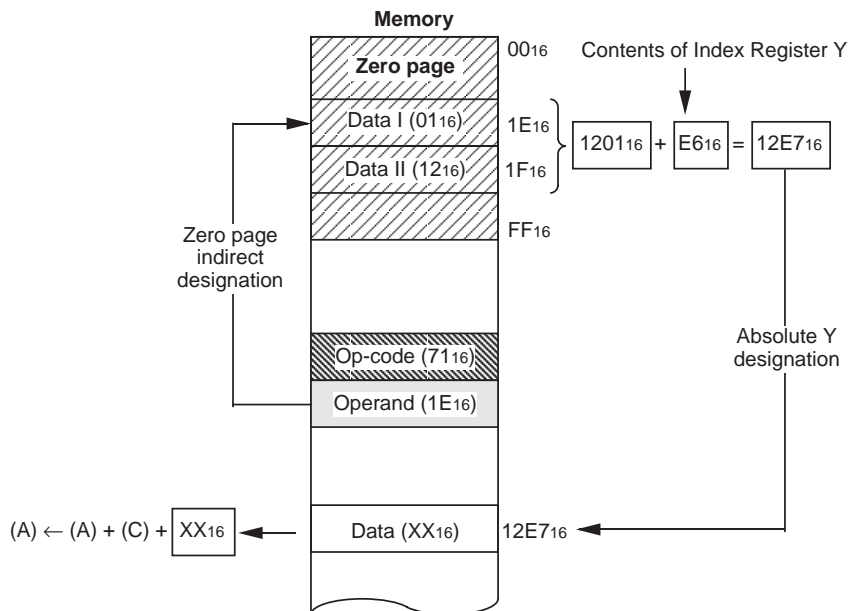
Instructions : ADC, AND, CMP, EOR, LDA, ORA, SBC, STA

Example : Mnemonic

$\Delta$ ADC $\Delta$ (\$1E),Y

Machine code

$71_{16} 1E_{16}$



Assuming that "01<sub>16</sub>" for Data I, and "12<sub>16</sub>" for Data II are stored in advance.

## INSTRUCTIONS

# Indirect Absolute

Addressing mode

Addressing mode : Indirect Absolute

Function : Specifies the address in a memory location as the jump destination address.

The address in the memory location is determined by the following:

- Operand I is used as the low-order byte of an address and Operand II as the high-order byte.
- The contents of the address above is used as the low-order byte and the contents of the next address as the high-order byte.
- The high-order and low-order bytes in step b together form the address in the memory location.

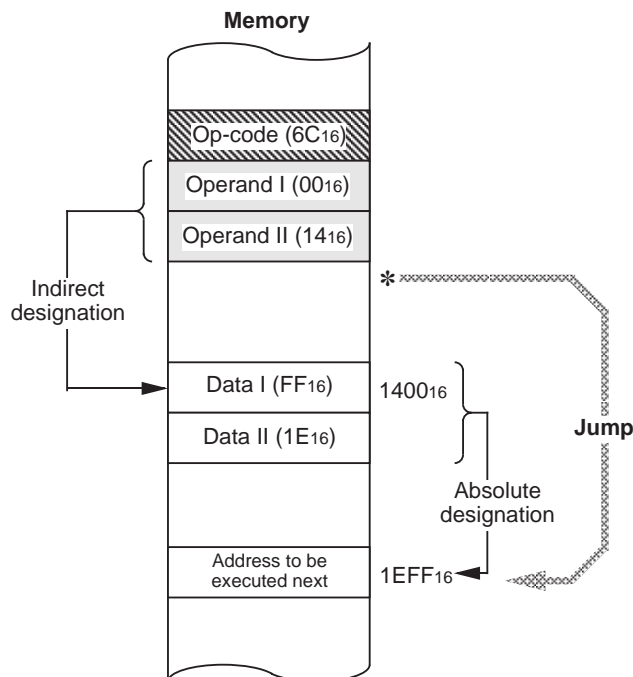
Instructions : JMP

Example : Mnemonic

$\Delta$ JMP $\Delta$ (\$1400)

Machine code

6C<sub>16</sub> 00<sub>16</sub> 14<sub>16</sub>



Assuming that "FF<sub>16</sub>" for Data I, and "1E<sub>16</sub>" for Data II are stored in advance.

**Note:** The page's last address (address XXFF<sub>16</sub>) can not be specified for the indirect designation address; in other words, JMP (\$XXFF) can not be executed.

## INSTRUCTIONS

# Zero Page Indirect Addressing mode

Addressing mode : Zero Page Indirect Absolute

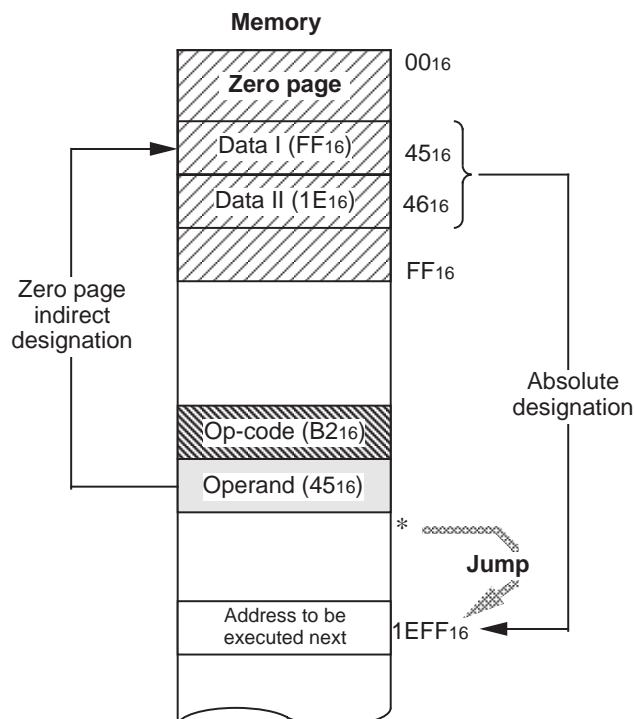
Function : Specifies the address in a memory location as the jump destination address. The address in the memory location is determined by the following:

- Operand is used as the low-order byte of an address in the Zero Page memory location and  $00_{16}$  as the high-order byte.
- The contents of the address in the Zero Page memory location is used as the low-order byte and the contents of the next Zero Page memory location as high-order byte.
- The high-order and low-order bytes in step b together form the address of the memory location.

Instructions : **JMP, JSR**

Example : Mnemonic  
 **$\Delta$ JMP $\Delta$ (\$45)**

Machine code  
 **$B2_{16}$   $45_{16}$**



Assuming that " $FF_{16}$ " for Data I, and " $1E_{16}$ " for Data II are stored in advance.

# Special Page

Addressing mode : **Special Page**

Function : Specifies the address in a Special Page memory location as the jump destination address. The address in the Special Page memory location is determined by using Operand as the low-order byte of the address and FF<sub>16</sub> as the high-order byte.


Instructions : **JSR**

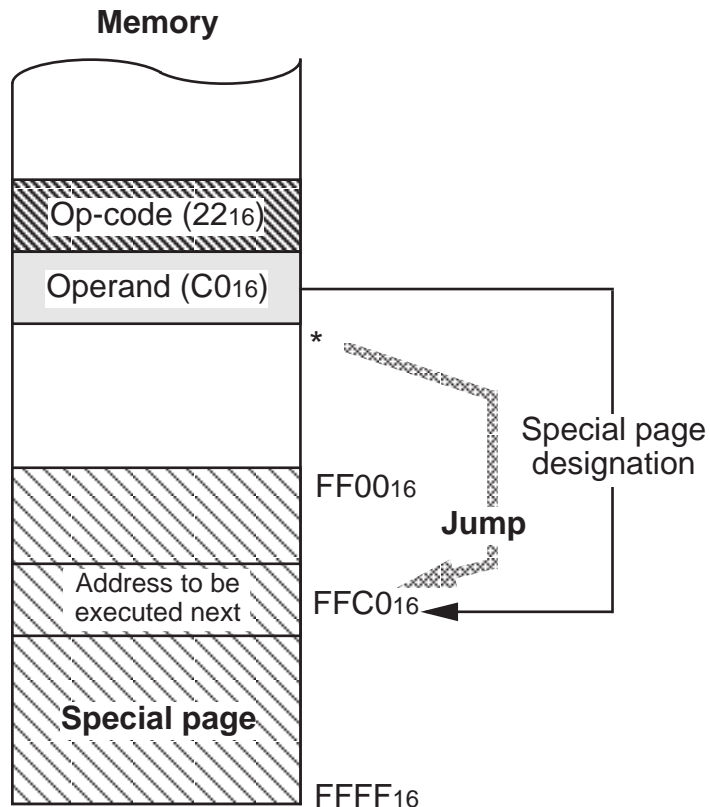
Example : Mnemonic

$\Delta$ JSR $\Delta$ \$FFC0

Machine code

22<sub>16</sub> C0<sub>16</sub>

 This symbol indicates the Special page mode.



# Zero Page Bit

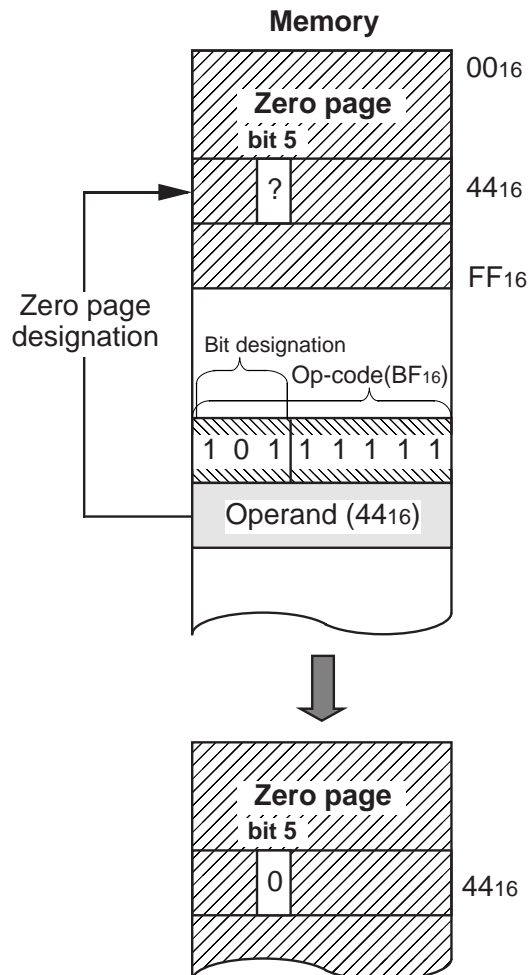
Addressing mode : Zero Page Bit

Function : Specifies one bit of the contents in a Zero Page memory location as the data for the instruction. Operand is used as the low-order byte of the address in the Zero Page memory location and 00<sub>16</sub> as the high-order byte. The bit position is designated by the high-order three bits of the Op-code.

Instructions : CLB, SEB

Example : Mnemonic  
 $\Delta$ CLB $\Delta$ 5,\$44

Machine code  
 BF<sub>16</sub> 44<sub>16</sub>



## INSTRUCTIONS

# Accumulator Bit

Addressing mode

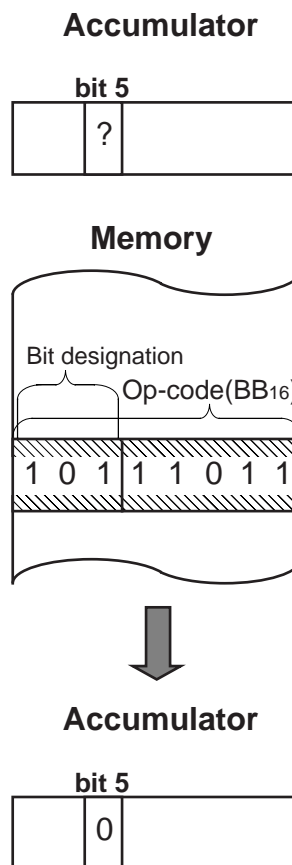
Addressing mode : **Accumulator Bit**

Function : **Specifies one bit of the Accumulator as the data for the instruction. The bit position is designated by the high-order three bits of the Op-Code.**

Instruction: **CLB, SEB**

Example : Mnemonic  
**ΔCLBΔ5,A**

Machine code  
**BB<sub>16</sub>**



**INSTRUCTIONS**

# Accumulator Bit Relative

Addressing mode

Addressing mode : Accumulator Bit Relative

**Function :** Specifies the address in a memory location where the next Op-Code is located. The bit position is designated by the high-order three bits of the Op-Code. If the branch condition is satisfied, Operand and the Program Counter are added. The result of this addition is the address in the memory location. When the branch condition is not satisfied, the next instruction is executed.

Instructions : **BBC, BBS**

Example : Mnemonic

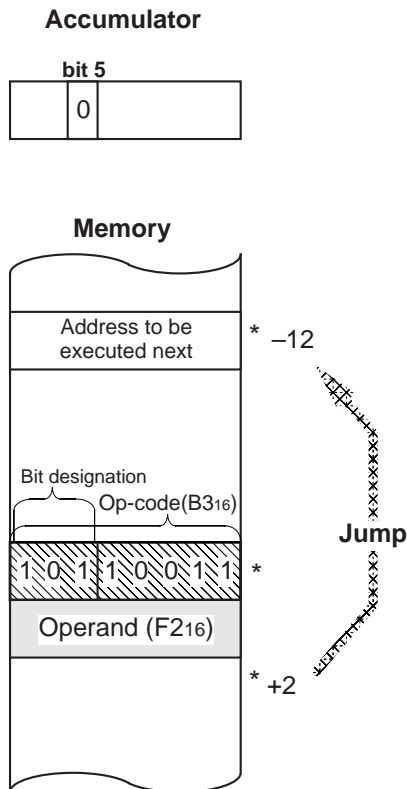
$\Delta$ **BBC** $\Delta$ 5,A,\*-12

Machine code

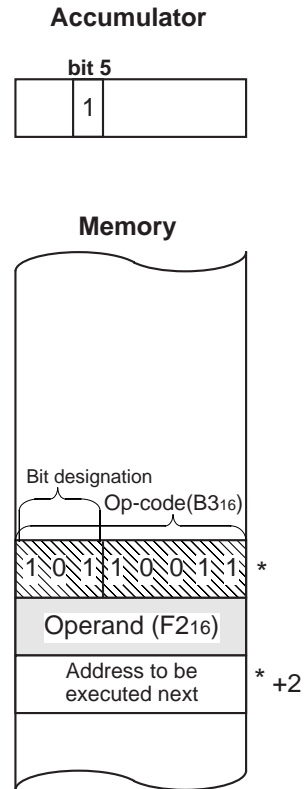
**B3<sub>16</sub> F2<sub>16</sub>**

Decimal

When the bit 5 of the Accumulator is cleared



When the bit 5 of the Accumulator is set



# Zero Page Bit Relative

Addressing mode : Zero Page Bit Relative

Function : Specifies the address of a memory location where the next Op-Code is located.

The bit position is designated by the high-order three bits of the Op-Code. The address in the Zero Page memory location is determined by using Operand I as low-order byte of the address and 00<sub>16</sub> as the high-order byte. If the branch condition is satisfied, Operand II and the Program Counter are added. The result of this addition is the address in the memory location. When the branch condition is not satisfied, the next instruction is executed.

Instructions : **BBC, BBS**

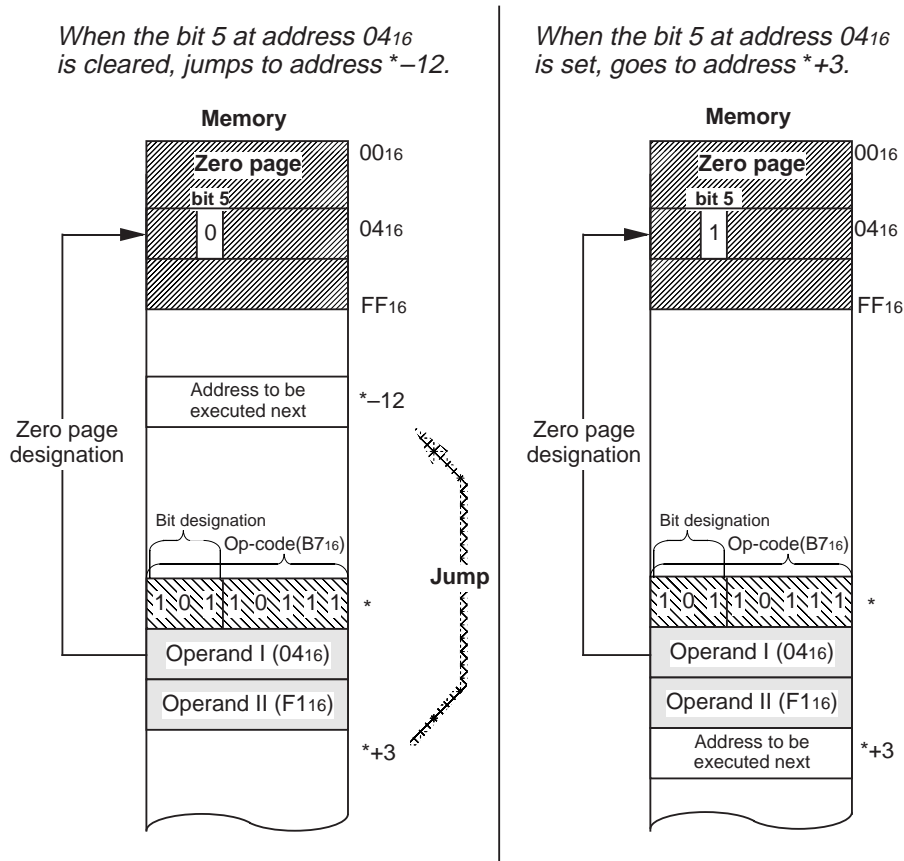
Example : Mnemonic

**ΔBBCΔ5,\$04,\*-12**

Decimal

Machine language

**B7<sub>16</sub> 04<sub>16</sub> F1<sub>16</sub>**





# INSTRUCTIONS

## Instruction Set

---

### 3.2 Instruction Set

The 740 Family has 71 types of instructions. The detailed explanation of the instructions is presented in §3.3. Note that some instructions cannot be used for any products.

#### 3.2.1 Data transfer instructions

These instructions transfer the data between registers, register and memory, and memories. The following are data transfer instructions.

	Instruction	Function
<b>Load</b>	LDA	Load memory value into Accumulator, or memory where is indicated by Index Register X
	LDM	Load immediate value into memory
	LDX	Load memory contents into Index Register X
	LDY	Load memory contents into Index Register Y
<b>Store</b>	STA	Store Accumulator into memory
	STX	Store Index Register X into memory
	STY	Store Index Register Y into memory
<b>Transfer</b>	TAX	Transfer Accumulator to the Index Register X
	TXA	Transfer Index Register X into the Accumulator
	TAY	Transfer Accumulator into the Index Register Y
	TYA	Transfer Index Register Y into the Accumulator
	TSX	Transfer Stack Pointer into the Index Register X
	TXS	Transfer Index Register X into the Stack Pointer
<b>Stack Operation</b>	PHA	Push Accumulator onto the Stack
	PHP	Push Processor Status onto the Stack
	PLA	Pull Accumulator from the Stack
	PLP	Pull Processor Status from the Stack

# INSTRUCTIONS

## 3.2.2 Operating instruction

The operating instructions include the operations of addition and subtraction, logic, comparison, rotation, and shift.

The operating instructions are as follows:

	Instructions	Contents
<b>Addition &amp; Subtraction</b>	ADC	Add memory contents and C flag to Accumulator or memory where is indicated by Index Register X
	SBC	Subtracts memory contents and C flag's complement from Accumulator or memory where is indicated by Index Register X
	INC	Increment Accumulator or memory contents by 1
	DEC	Decrement Accumulator or memory contents by 1
	INX	Increment Index Register X by 1
	DEX	Decrement Index Register X by 1
	INY	Increment Index Register Y by 1
	DEY	Decrement Index Register Y by 1
<b>Multiplication &amp; Division</b>	MUL (Note)	Multiply Accumulator with memory specified by Zero Page X addressing mode and store high-order byte of result on Stack and low-order byte in Accumulator
	DIV (Note)	Quotient is stored in Accumulator and one's complement of remainder is pushed onto stack
<b>Logical Operation</b>	AND	"AND" memory with Accumulator or memory where is indicated by Index Register X
	ORA	"OR" memory with Accumulator or memory where is indicated by Index Register X
	EOR	"Exclusive-OR" memory with Accumulator or memory where is indicated by Index Register X
	COM	Store one's complement of memory contents to memory
	BIT	"AND" memory with Accumulator (The result is not stored into anywhere.)
<b>Comparison</b>	TST	Test whether memory content is "0" or not
	CMP	Compare memory contents and Accumulator or memory where is indicated by Index Register X
	CPX CPY	Compare memory contents and Index Register X Compare memory contents and Index Register Y
<b>Shift &amp; Rotate</b>	ASL	Shift left one bit (memory contents or Accumulator)
	LSR	Shift right one bit (memory contents or Accumulator)
	ROL	Rotate one bit left with carry (memory contents or Accumulator)
	ROR	Rotate one bit right with carry (memory contents or Accumulator)
	RRF	Rotate four bits right without carry (memory)

**Note:** For some products, multiplication and division instructions cannot be used.

# INSTRUCTIONS

## Instruction Set

### 3.2.3 Bit managing instructions

The bit managing instructions clear “0” or set “1” designated bits of the Accumulator or memory.

	Instructions	Contents
Bit Managing	CLB	Clear designated bit in the Accumulator or memory
	SEB	Set designated bit in the Accumulator or memory

### 3.2.4 Flag setting instructions

The flag setting instructions clear “0” or set “1” C, D, I, T and V flags.

	Instructions	Contents
Flag Setting	CLC	Clear C flag
	SEC	Set C flag
	CLD	Clear D flag
	SED	Set D flag
	CLI	Clear I flag
	SEI	Set I flag
	CLT	Clear T flag
	SET	Set T flag
	CLV	Clear V flag
		C flag : Carry Flag
		D flag : Decimal Mode Flag
		I flag : Interrupt Disable Flag
		T flag : X Modified Operation Mode Flag
		V flag : Overflow Flag

### 3.2.5 Jump, Branch and Return instructions

The jump, branch and return instructions as following are used to change program flow.

	Instructions	Contents
Jump	JMP	Jump to new location
	BRA	Jump to new location
	JSR	Jump to new location saving the current address
Branch	BBC	Branch when the designated bit in the Accumulator or memory is “0”
	BBS	Branch when the designated bit in the Accumulator or memory is “1”
	BCC	Branch when the C Flag is “0”
	BCS	Branch when the C Flag is “1”
	BNE	Branch when the Z Flag is “0”
	BEQ	Branch when the Z Flag is “1”
	BPL	Branch when the N Flag is “0”
	BMI	Branch when the N Flag is “1”
	BVC	Branch when the V Flag is “0”
BVS	Branch when the V Flag is “1”	
Return	RTI	Return from interrupt
	RTS	Return from subroutine

# INSTRUCTIONS

## Instruction Set

---

### 3.2.6 Interrupt instruction (Break instruction)

This instruction causes a software interrupt.

	Instruction	Contents
Interrupt	BRK	Executes a software interrupt.

### 3.2.7 Special instructions

These special instructions control the oscillation and the internal clock.

	Instructions	Contents
Special	WIT	Stops the internal clock.
	STP	Stops the oscillation of oscillator.

### 3.2.8 Other instruction

	Instruction	Contents
Other	NOP	Only advances the program counter.

# INSTRUCTIONS

## 3.3 Description of instructions

This section presents in detail the 740 Family instructions by arranging mnemonics of instructions alphabetically and dividing each instruction essentially into one page.

The heading of each page is a mnemonic. Operation, explanation and changes of status flags are indicated for each instruction. In addition, assembler coding format, machine code, byte number, and list of cycle numbers for each addressing mode are indicated.

The following are symbols used in this manual:

Symbol	Description	Symbol	Description
A	Accumulator	hh	Address high-order byte data in 0 to 255
Ai	Bit i of Accumulator	ll	Address low-order byte data in 0 to 255
PC	Program Counter	zz	Zero page address data in 0 to 255
PCL	Low-order byte of Program Counter	nn	Data in 0 to 255
PCH	High-order byte of Program Counter	i	Data in 0 to 7
PS	Processor Status Register	*	Contents of the Program Counter
S	Stack Pointer	Δ	Tab or space
X	Index Register X	#	Immediate mode
Y	Index Register Y	\	Special page mode
M	Memory	\$	Hexadecimal symbol
Mi	Bit i of memory	+	Addition
C	Carry Flag	-	Subtraction
Z	Zero Flag	X	Multiplication
I	Interrupt Disable Flag	/	Division
D	Decimal Operation Mode Flag	^	Logical AND
B	Break Flag	∨	Logical OR
T	X Modified Operations Mode Flag	∇	Logical exclusive OR
V	Overflow Flag	( )	Contents of register, memory, etc.
N	Negative Flag	←	Direction of data transfer
REL	Relative address		
BADRS	Break address		

## ADD WITH CARRY

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (A) + (M) + (C)$   
 When  $(T) = 1$ ,  $(M(X)) \leftarrow (M(X)) + (M) + (C)$

**Function :** When  $T = 0$ , this instruction adds the contents  $M$ ,  $C$ , and  $A$ ; and stores the results in  $A$  and  $C$ .  
 When  $T = 1$ , this instruction adds the contents of  $M(X)$ ,  $M$  and  $C$ ; and stores the results in  $M(X)$  and  $C$ . When  $T=1$ , the contents of  $A$  remain unchanged, but the contents of status flags are changed.  
 $M(X)$  represents the contents of memory where is indicated by  $X$ .

**Status flag:** **N** :  $N$  is 1 when bit 7 is 1 after the operation; otherwise it is 0.  
**V** :  $V$  is 1 when the operation result exceeds +127 or  $-128$ ; otherwise it is 0.  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** :  $Z$  is 1 when the operation result is 0; otherwise it is 0.  
**C** :  $C$  is 1 when the result of a binary addition exceeds 255 or when the result of a decimal addition exceeds 99; otherwise it is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta ADC \Delta \# \$nn$	$69_{16}, nn_{16}$	2	2
Zero page	$\Delta ADC \Delta \$zz$	$65_{16}, zz_{16}$	2	3
Zero page X	$\Delta ADC \Delta \$zz, X$	$75_{16}, zz_{16}$	2	4
Absolute	$\Delta ADC \Delta \$hhll$	$6D_{16}, ll_{16}, hh_{16}$	3	4
Absolute X	$\Delta ADC \Delta \$hhll, X$	$7D_{16}, ll_{16}, hh_{16}$	3	5
Absolute Y	$\Delta ADC \Delta \$hhll, Y$	$79_{16}, ll_{16}, hh_{16}$	3	5
(Indirect X)	$\Delta ADC \Delta (\$zz, X)$	$61_{16}, zz_{16}$	2	6
(Indirect Y)	$\Delta ADC \Delta (\$zz), Y$	$71_{16}, zz_{16}$	2	6

Notes 1: When  $T=1$ , add 3 to the cycle number.

2: When ADC instruction is executed in the decimal operation mode ( $D = 1$ ), decision of  $C$  is delayed. Accordingly, do not execute the instruction which operates  $C$  such as SEC, CLC, etc.

# AND

# AND

## LOGICAL AND

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (A) \wedge (M)$   
 When  $(T) = 1$ ,  $(M(X)) \leftarrow (M(X)) \wedge (M)$

**Function :** When  $T = 0$ , this instruction transfers the contents of A and M to the ALU which performs a bit-wise AND operation and stores the result back in A.

When  $T = 1$ , this instruction transfers the contents  $M(X)$  and M to the ALU which performs a bit-wise AND operation and stores the results back in  $M(X)$ . When  $T = 1$  the contents of A remain unchanged, but status flags are changed.

$M(X)$  represents the contents of memory where is indicated by X.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise it is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise it is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{AND}\Delta\#\$\text{nn}$	$29_{16}, \text{nn}_{16}$	2	2
Zero page	$\Delta\text{AND}\Delta\$\text{zz}$	$25_{16}, \text{zz}_{16}$	2	3
Zero page X	$\Delta\text{AND}\Delta\$\text{zz},\text{X}$	$35_{16}, \text{zz}_{16}$	2	4
Absolute	$\Delta\text{AND}\Delta\$\text{hhll}$	$2\text{D}_{16}, \text{ll}_{16}, \text{hh}_{16}$	3	4
Absolute X	$\Delta\text{AND}\Delta\$\text{hhll},\text{X}$	$3\text{D}_{16}, \text{ll}_{16}, \text{hh}_{16}$	3	5
Absolute Y	$\Delta\text{AND}\Delta\$\text{hhll},\text{Y}$	$39_{16}, \text{ll}_{16}, \text{hh}_{16}$	3	5
(Indirect X)	$\Delta\text{AND}\Delta(\$\text{zz},\text{X})$	$21_{16}, \text{zz}_{16}$	2	6
(Indirect Y)	$\Delta\text{AND}\Delta(\$\text{zz}),\text{Y}$	$31_{16}, \text{zz}_{16}$	2	6

Note: When  $T = 1$ , add 3 to a cycle number.

# ASL

# ASL

## ARITHMETIC SHIFT LEFT

**Operation :**

C	←	b7						b0	←	0
---	---	----	--	--	--	--	--	----	---	---

**Function :** This instruction shifts the content of A or M by one bit to the left, with bit 0 always being set to 0 and bit 7 of A or M always being contained in C.

**Status flag:** **N** : N is 1 when bit 7 of A or M is 1 after the operation; otherwise it is 0.  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : Z is 1 when the operation result is 0; otherwise it is 0.  
**C** : C is 1 when bit 7 of A or M is 1, before this operation; otherwise it is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta$ ASL $\Delta$ A	0A <sub>16</sub>	1	2
Zero page	$\Delta$ ASL $\Delta$ \$zz	06 <sub>16</sub> , zz <sub>16</sub>	2	5
Zero page X	$\Delta$ ASL $\Delta$ \$zz,X	16 <sub>16</sub> , zz <sub>16</sub>	2	6
Absolute	$\Delta$ ASL $\Delta$ \$hhll	0E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
Absolute X	$\Delta$ ASL $\Delta$ \$hhll,X	1E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	7



## BRANCH ON BIT CLEAR

---

**Operation :** When  $(M_i)$  or  $(A_i) = 0$ ,  $(PC) \leftarrow (PC) + n + REL$   
 $(M_i)$  or  $(A_i) = 1$ ,  $(PC) \leftarrow (PC) + n$   
n: If addressing mode is Zero Page Bit Relative,  $n=3$ . And if addressing mode is Accumulator Bit Relative,  $n=2$ .

**Function :** This instruction tests the designated bit  $i$  of  $M$  or  $A$  and takes a branch if the bit is 0. The branch address is specified by a relative address. If the bit is 1, next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit Relative	$\Delta BBC \Delta i, A, \$hhll$	$(20i+13)_{16}, rr_{16}$	2	4
Zero page bit Relative	$\Delta BBC \Delta i, \$zz, \$hhll$	$(20i+17)_{16}, ZZ_{16}, rr_{16}$	3	5

- Notes 1:  $rr_{16} = \$hhll - (*+n)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.  
3: When executing the BBC instruction after the contents of the interrupt request bit is changed, one instruction or more must be passed before the BBC instruction is executed.

## BRANCH ON BIT SET

---

**Operation :** When  $(Mi)$  or  $(Ai) = 1$ ,  $(PC) \leftarrow (PC) + n + REL$   
                   $(Mi)$  or  $(Ai) = 0$ ,  $(PC) \leftarrow (PC) + n$   
                   $n$  : If addressing mode is Zero Page Bit Relative,  $n=3$ . And if  
                  addressing mode is Accumulator Bit Relative,  $n=2$ .

**Function :** This instruction tests the designated bit  $i$  of the M or A and takes a branch if the bit is 1. The branch address is specified by a relative address. If the bit is 0, next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit Relative	$\Delta BBS\Delta i, A, \$hhll$	$(20i+3)_{16}, rr_{16}$	2	4
Zero page bit Relative	$\Delta BBS\Delta i, \$zz, \$hhll$	$(20i+7)_{16}, zz_{16}, rr_{16}$	3	5

Notes 1:  $rr_{16} = \$hhll - (*+n)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .

2: When a branch is executed, add 2 to the cycle number.

3: When executing the BBS instruction after the contents of the interrupt request bit is changed, one instruction or more must be passed before the BBS instruction is executed.

# BCC

# BCC

## BRANCH ON CARRY CLEAR

---

**Operation :** When  $(C) = 0$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
 $(C) = 1$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address if C is 0. The branch address is specified by a relative address. If C is 1, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BCC \Delta \$hhll$	90 <sub>16</sub> , rr <sub>16</sub>	2	2

Notes 1:  $rr_{16} = \$hhll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.

## BRANCH ON CARRY SET

---

**Operation :** When  $(C) = 1$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
 $(C) = 0$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address if C is 1. The branch address is specified by a relative address. If C is 0, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BCS \Delta \$hll$	B016, rr16	2	2

Notes 1:  $rr16 = \$hll - (*+2)$ . The rr16 is a value in a range of -128 to +127.  
2: When a branch is executed, add 2 to the cycle number.

# BEQ

## BRANCH ON EQUAL

# BEQ

**Operation :** When  $(Z) = 1$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
                   $(Z) = 0$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address when Z is 1. The branch address is specified by a relative address. If Z is 0, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BEQ \Delta \$hhll$	$F0_{16}, rr_{16}$	2	2

Notes 1:  $rr_{16} = \$hhll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.

# BIT

## TEST BIT IN MEMORY WITH ACCUMULATOR

# BIT

**Operation :**  $(A) \wedge (M)$

**Function :** This instruction takes a bit-wise logical AND of A and M contents; however, the contents of A and M are not modified. The contents of N, V, Z are changed, but the contents of A, M remain unchanged.

**Status flag:** **N** : N is 1 when bit 7 of M is 1; otherwise it is 0.  
**V** : V is 1 when bit 6 of M is 1; otherwise it is 0.  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : Z is 1 when the result of the operation is 0; otherwise Z is 0.  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta\text{BIT}\Delta\$zz$	$24_{16}, zz_{16}$	2	3
Absolute	$\Delta\text{BIT}\Delta\$hhl$	$2C_{16}, ll_{16}, hh_{16}$	3	4

## BRANCH ON RESULT MINUS

---

**Operation :** When  $(N) = 1$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
                   $(N) = 0$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address when N is 1. The branch address is specified by a relative address. If N is 0, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BMI \Delta \$hhll$	$30_{16}, rr_{16}$	2	2

Notes 1:  $rr_{16} = \$hhll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.

# BNE

# BNE

## BRANCH ON NOT EQUAL

---

**Operation :** When  $(Z) = 0$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
 $(Z) = 1$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address if Z is 0. The branch address is specified by a relative address. If Z is 1, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BNE \Delta \$hll$	D0 <sub>16</sub> , rr <sub>16</sub>	2	2

Notes 1:  $rr_{16} = \$hll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.



# BPL

# BPL

## BRANCH ON RESULT PLUS

---

**Operation :** When  $(N) = 0$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
 $(N) = 1$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address if N is 0. The branch address is specified by a relative address. If N is 1, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BPL \Delta \$hhll$	1016, rr16	2	2

Notes 1:  $rr16 = \$hhll - (*+2)$ . The rr16 is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.

# BRA

## BRANCH ALWAYS

# BRA

**Operation :**  $(PC) \leftarrow (PC) + 2 + REL$

**Function :** This instruction branches to the appointed address. The branch address is specified by a relative address.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BRA \Delta \$hhll$	80 <sub>16</sub> , rr <sub>16</sub>	2	4

Note: rr<sub>16</sub>=\$hhll-( $\ast+2$ ). The rr<sub>16</sub> is a value in a range of -128 to +127.

# BRK

## FORCE BREAK

# BRK

**Operation :** (B)  $\leftarrow$  1  
(PC)  $\leftarrow$  (PC) + 2  
(M(S))  $\leftarrow$  (PCH)  
(S)  $\leftarrow$  (S) - 1  
(M(S))  $\leftarrow$  (PCL)  
(S)  $\leftarrow$  (S) - 1  
(M(S))  $\leftarrow$  (PS)  
(S)  $\leftarrow$  (S) - 1  
(I)  $\leftarrow$  1  
(PC)  $\leftarrow$  BADRS (Note 1)

**Function :** When the BRK instruction is executed, the CPU pushes the current PC contents onto the stack. The BADRS designated in the interrupt vector table is stored into the PC.

**Status flag:** N : No change  
V : No change  
T : No change  
B : 1  
I : 1  
D : No change  
Z : No change  
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ BRK $\Delta$	0016	1	7

Notes 1: "BADRS" means a break address.

- 2: The value of the PC pushed onto the stack by the execution of the BRK instruction is the BRK instruction address plus two. Therefore, the byte following the BRK will not be executed when the value of the PC is returned from the BRK routine.
- 3: Both after the BRK instruction is executed and after INT is input, the program is branched to the address where is specified by the interrupt vector table. By testing the value of the B Flag in the PS (pushed on the Stack) in the interrupt service routine, the user can determine if the interrupt was caused by the BRK instruction.

# BVC

# BVC

## BRANCH ON OVERFLOW CLEAR

---

**Operation :** When  $(V) = 0$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
 $(V) = 1$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address if V is 0. The branch address is specified by a relative address. If V is 1, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BVC \Delta \$hll$	$50_{16}, rr_{16}$	2	2

Notes 1:  $rr_{16} = \$hll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .

2: When a branch is executed, add 2 to the cycle number.

# BVS

# BVS

## BRANCH ON OVERFLOW SET

---

**Operation :** When  $(V) = 1$ ,  $(PC) \leftarrow (PC) + 2 + REL$   
                   $(V) = 0$ ,  $(PC) \leftarrow (PC) + 2$

**Function :** This instruction takes a branch to the appointed address when  $V$  is 1. The branch address is specified by a relative address. When  $V$  is 0, the next instruction is executed.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	$\Delta BVS \Delta \$hhll$	$70_{16}, rr_{16}$	2	2

Notes 1:  $rr_{16} = \$hhll - (*+2)$ . The  $rr_{16}$  is a value in a range of  $-128$  to  $+127$ .  
2: When a branch is executed, add 2 to the cycle number.

# CLB

# CLB

## CLEAR BIT

---

**Operation :**  $(A_i) \leftarrow 0$ , or  
 $(M_i) \leftarrow 0$

**Function :** This instruction clears the designated bit  $i$  of  $A$  or  $M$ .

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit	$\Delta CLB \Delta i, A$	$(20i+1B)_{16}$	1	2
Zero page bit	$\Delta CLB \Delta i, \$zz$	$(20i+1F)_{16},$ $ZZ_{16}$	2	5

# CLC

## CLEAR CARRY FLAG

# CLC

**Operation :**  $(C) \leftarrow 0$

**Function :** This instruction clears C.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : No change  
**C** : 0

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ CLC	1816	1	2

# CLD

## CLEAR DECIMAL MODE

# CLD

**Operation :** (D) ← 0

**Function :** This instruction clears D.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : 0  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ CLD	D8 <sub>16</sub>	1	2



# CLI

## CLEAR INTERRUPT DISABLE STATUS

---

# CLI

**Operation :**  $(I) \leftarrow 0$

**Function :** This instruction clears I.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : 0  
**D** : No change  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ CLI	58 <sub>16</sub>	1	2

# CLT

## CLEAR TRANSFER FLAG

# CLT

**Operation :**  $(T) \leftarrow 0$

**Function :** This instruction clears T.

**Status flag:** **N** : No change  
**V** : No change  
**T** : 0  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ CLT	12 <sub>16</sub>	1	2

# CLV

## CLEAR OVERFLOW FLAG

# CLV

**Operation :**  $(V) \leftarrow 0$

**Function :** This instruction clears V.

**Status flag**

- N :** No change
- V :** 0
- T :** No change
- B :** No change
- I :** No change
- D :** No change
- Z :** No change
- C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ CLV	B816	1	2

**Operation :** When (T) = 0, (A) – (M)  
 (T) = 1, (M(X)) – (M)

**Function :** When T = 0, this instruction subtracts the contents of M from the contents of A. The result is not stored and the contents of A or M are not modified.  
 When T = 1, the CMP subtracts the contents of M from the contents of M(X). The result is not stored and the contents of X, M, and A are not modified.  
 M(X) represents the contents of memory where is indicated by X.

**Status flag:** **N :** N is 1 when bit 7 of the operation result is 1 after the operation; otherwise N is 0.  
**V :** No change  
**T :** No change  
**B :** No change  
**I :** No change  
**D :** No change  
**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.  
**C :** C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{CMP}\Delta\#\$\text{nn}$	C9 <sub>16</sub> , nn <sub>16</sub>	2	2
Zero page	$\Delta\text{CMP}\Delta\$\text{zz}$	C5 <sub>16</sub> , ZZ <sub>16</sub>	2	3
Zero page X	$\Delta\text{CMP}\Delta\$\text{zz},\text{X}$	D5 <sub>16</sub> , ZZ <sub>16</sub>	2	4
Absolute	$\Delta\text{CMP}\Delta\$\text{hhll}$	CD <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	4
Absolute X	$\Delta\text{CMP}\Delta\$\text{hhll},\text{X}$	DD <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
Absolute Y	$\Delta\text{CMP}\Delta\$\text{hhll},\text{Y}$	D9 <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
(Indirect X)	$\Delta\text{CMP}\Delta(\$\text{zz},\text{X})$	C1 <sub>16</sub> , ZZ <sub>16</sub>	2	6
(Indirect Y)	$\Delta\text{CMP}\Delta(\$\text{zz}),\text{Y}$	D1 <sub>16</sub> , ZZ <sub>16</sub>	2	6

Note: When T=1, add 1 to the cycle number.

# COM

## COMPLEMENT

# COM

**Operation :**  $(M) \leftarrow \overline{(M)}$

**Function :** This instruction takes the one's complement of the contents of M and stores the result in M.

**Status flag:** **N :** N is 1 when bit 7 of the M is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta\text{COM}\Delta\$zz$	$44_{16}, zz_{16}$	2	5

**Operation :** (X) – (M)

**Function :** This instruction subtracts the contents of M from the contents of X. The result is not stored and the contents of X and M are not modified.

**Status flag: N :** N is 1 when bit 7 of the operation result is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{CPX}\Delta\#\$\text{nn}$	$\text{E0}_{16}, \text{nn}_{16}$	2	2
Zero page	$\Delta\text{CPX}\Delta\$\text{zz}$	$\text{E4}_{16}, \text{zz}_{16}$	2	3
Absolute	$\Delta\text{CPX}\Delta\$\text{hhll}$	$\text{EC}_{16}, \text{ll}_{16}, \text{hh}_{16}$	3	4

# CPY

## COMPARE MEMORY AND INDEX REGISTER Y

# CPY

**Operation :** (Y) – (M)

**Function :** This instruction subtracts the contents of M from the contents of Y. The result is not stored and the contents of Y and M are not modified.

**Status flag:** **N** : N is 1 when bit 7 of the operation result is 1 after the operation; otherwise N is 0.

**V** : No change

**T** : No change

**B** : No change

**I** : No change

**D** : No change

**Z** : Z is 1 when the operation result is 0; otherwise Z is 0.

**C** : C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{CPY}\Delta\#\$\text{nn}$	$\text{C0}_{16}, \text{nn}_{16}$	2	2
Zero page	$\Delta\text{CPY}\Delta\$\text{zz}$	$\text{C4}_{16}, \text{zz}_{16}$	2	3
Absolute	$\Delta\text{CPY}\Delta\$\text{hhll}$	$\text{CC}_{16}, \text{ll}_{16}, \text{hh}_{16}$	3	4

# DEC

## DECREMENT BY ONE

# DEC

**Operation :**  $(A) \leftarrow (A) - 1$ , or  
 $(M) \leftarrow (M) - 1$

**Function :** This instruction subtracts 1 from the contents of A or M.

**Status flag:** **N** : N is 1 when bit 7 is 1 after the addition; otherwise N is 0.  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : Z is 1 when the operation result is 0; otherwise Z is 0.  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta\text{DEC}\Delta A$	$1A_{16}$	1	2
Zero page	$\Delta\text{DEC}\Delta \$zz$	$C6_{16}, zz_{16}$	2	5
Zero page X	$\Delta\text{DEC}\Delta \$zz,X$	$D6_{16}, zz_{16}$	2	6
Absolute	$\Delta\text{DEC}\Delta \$hhl$	$CE_{16}, ll_{16}, hh_{16}$	3	6
Absolute X	$\Delta\text{DEC}\Delta \$hhl,X$	$DE_{16}, ll_{16}, hh_{16}$	3	7



# DEX

DECREMENT INDEX REGISTER X BY ONE

# DEX

**Operation :**  $(X) \leftarrow (X) - 1$

**Function :** This instruction subtracts one from the current contents of X.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ DEX	CA <sub>16</sub>	1	2

# DEY

## DECREMENT INDEX REGISTER Y BY ONE

# DEY

**Operation :**  $(Y) \leftarrow (Y) - 1$

**Function :** This instruction subtracts one from the current contents of Y.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

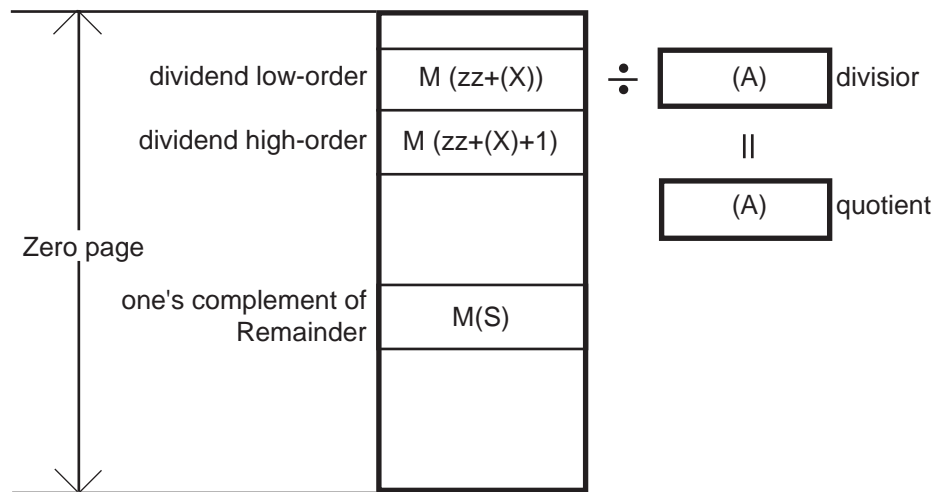
**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ DEY	88 <sub>16</sub>	1	2

## DIVIDE MEMORY BY ACCUMULATOR

**Operation :**  $(A) \leftarrow (M(zz+(X)+1), M(zz+(X))) / (A)$   
 $M(S) \leftarrow$  one's complement of Remainder  
 $(S) \leftarrow (S) - 1$

**Function :** Divides the 16-bit data in  $M(zz+(X))$  (low-order byte) and  $M(zz+(X)+1)$  (high-order byte) by the contents of A. The quotient is stored in A and the one's complement of the remainder is pushed onto the stack.



**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page X	$\Delta DIV \Delta \$zz, X$	E2 <sub>16</sub> , ZZ <sub>16</sub>	2	16

Notes 1: The quotient's overflow and zero division can not be detected. Check the quotient's overflow and zero division by software before DIV instruction is executed. This instruction changes the Stack Pointer and the contents of the Accumulator.

2: The DIV instruction can not be used for any products.

# EOR

# EOR

## EXCLUSIVE OR MEMORY WITH ACCUMULATOR

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (A) \vee (M)$   
 $(T) = 1$ ,  $(M(X)) \leftarrow (M(X)) \vee (M)$

**Function :** When  $T = 0$ , this instruction transfers the contents of the M and A to the ALU which performs a bit-wise Exclusive OR, and stores the result in A.

When  $T = 1$ , the contents of  $M(X)$  and M are transferred to the ALU, which performs a bit-wise Exclusive OR and stores the results in  $M(X)$ . The contents of A remain unchanged, but status flags are changed.

$M(X)$  represents the contents of memory where is indicated by X.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta EOR \Delta \# \$nn$	$49_{16}, nn_{16}$	2	2
Zero page	$\Delta EOR \Delta \$zz$	$45_{16}, zz_{16}$	2	3
Zero page X	$\Delta EOR \Delta \$zz, X$	$55_{16}, zz_{16}$	2	4
Absolute	$\Delta EOR \Delta \$hhll$	$4D_{16}, ll_{16}, hh_{16}$	3	4
Absolute X	$\Delta EOR \Delta \$hhll, X$	$5D_{16}, ll_{16}, hh_{16}$	3	5
Absolute Y	$\Delta EOR \Delta \$hhll, Y$	$59_{16}, ll_{16}, hh_{16}$	3	5
(Indirect X)	$\Delta EOR \Delta (\$zz, X)$	$41_{16}, zz_{16}$	2	6
(Indirect Y)	$\Delta EOR \Delta (\$zz), Y$	$51_{16}, zz_{16}$	2	6

Note: When  $T=1$ , add 3 to the cycle number.

# INC

## INCREMENT BY ONE

# INC

**Operation :** (A)  $\leftarrow$  (A) + 1, or  
(M)  $\leftarrow$  (M) + 1

**Function :** This instruction adds one to the contents of A or M.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta$ INC $\Delta$ A	3A <sub>16</sub>	1	2
Zero page	$\Delta$ INC $\Delta$ \$zz	E6 <sub>16</sub> , zz <sub>16</sub>	2	5
Zero page X	$\Delta$ INC $\Delta$ \$zz,X	F6 <sub>16</sub> , zz <sub>16</sub>	2	6
Absolute	$\Delta$ INC $\Delta$ \$hhll	EE <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
Absolute X	$\Delta$ INC $\Delta$ \$hhll,X	FE <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	7

**Operation :**  $(X) \leftarrow (X) + 1$

**Function :** This instruction adds one to the contents of X.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ INX	E8 <sub>16</sub>	1	2

## INCREMENT INDEX REGISTER Y BY ONE

---

**Operation :**  $(Y) \leftarrow (Y) + 1$

**Function :** This instruction adds one to the contents of Y.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ INY	C8 <sub>16</sub>	1	2

# JMP

## JUMP

# JMP

- Operation :** When addressing mode is
- (a) Absolute, then  
 $(PC) \leftarrow hhll$
  - (b) Indirect Absolute, then  
 $(PCL) \leftarrow (hhll)$   
 $(PCH) \leftarrow (hhll+1)$
  - (c) Zero page Indirect Absolute, then  
 $(PCL) \leftarrow (zz)$   
 $(PCH) \leftarrow (zz+1)$

**Function :** This instruction jumps to the address designated by the following three addressing modes:

Absolute  
Indirect Absolute  
Zero Page Indirect Absolute

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Absolute	$\Delta JMP \Delta \$hhll$	$4C_{16}, ll_{16}, hh_{16}$	3	3
Indirect Absolute	$\Delta JMP \Delta (\$hhll)$	$6C_{16}, ll_{16}, hh_{16}$	3	5
Zero Page Indirect	$\Delta JMP \Delta (\$zz)$	$B2_{16}, zz_{16}$	2	4



**Operation :**  $(M(S)) \leftarrow (PCH)$

$(S) \leftarrow (S) - 1$

$(M(S)) \leftarrow (PCL)$

$(S) \leftarrow (S) - 1$

After the above operations, if the addressing mode is

(a) Absolute, then

$(PC) \leftarrow hhll$

(b) Special page, then

$(PCL) \leftarrow ll$

$(PCH) \leftarrow FF_{16}$

(c) Zero page Indirect, then

$(PCL) \leftarrow (zz)$

$(PCH) \leftarrow (zz+1)$

**Function :** This instruction stores the contents of the PC in the stack, then jumps to the address designated by the following addressing modes:

Absolute

Special Page

Zero Page Indirect Absolute

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Absolute	$\Delta JSR \Delta \$hhll$	$20_{16}, ll_{16}, hh_{16}$	3	6
Special page	$\Delta JSR \Delta \$hhll$ (Note)	$22_{16}, ll_{16}$	2	5
Zero page Indirect	$\Delta JSR \Delta (\$zz)$	$02_{16}, zz_{16}$	2	7

(Note) “\” ( $5C_{16}$  of the ASCII code) denotes special page.  $hh_{16}$  must be  $FF_{16}$  in the special page addressing mode.

# LDA

## LOAD ACCUMULATOR WITH MEMORY

# LDA

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (M)$   
                  When  $(T) = 1$ ,  $(M(X)) \leftarrow (M)$

**Function :** When  $T = 0$ , this instruction transfers the contents of  $M$  to  $A$ .  
When  $T = 1$ , this instruction transfers the contents of  $M$  to  $(M(X))$ . The contents of  $A$  remain unchanged, but status flags are changed.  
 $M(X)$  represents the contents of memory where is indicated by  $X$ .

**Status flag:** **N :**  $N$  is 1 when bit 7 is 1 after the operation; otherwise  $N$  is 0.  
**V :** No change  
**T :** No change  
**B :** No change  
**I :** No change  
**D :** No change  
**Z :**  $Z$  is 1 when the operation result is 0; otherwise  $Z$  is 0.  
**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta LDA \Delta \# \$nn$	$A9_{16}, nn_{16}$	2	2
Zero page	$\Delta LDA \Delta \$zz$	$A5_{16}, zz_{16}$	2	3
Zero page X	$\Delta LDA \Delta \$zz, X$	$B5_{16}, zz_{16}$	2	4
Absolute	$\Delta LDA \Delta \$hhll$	$AD_{16}, ll_{16}, hh_{16}$	3	4
Absolute X	$\Delta LDA \Delta \$hhll, X$	$BD_{16}, ll_{16}, hh_{16}$	3	5
Absolute Y	$\Delta LDA \Delta \$hhll, Y$	$B9_{16}, ll_{16}, hh_{16}$	3	5
(Indirect X)	$\Delta LDA \Delta (\$zz, X)$	$A1_{16}, zz_{16}$	2	6
(Indirect Y)	$\Delta LDA \Delta (\$zz), Y$	$B1_{16}, zz_{16}$	2	6

Note: When  $T = 1$ , add 2 to the cycle number.

# LDM

LOAD IMMEDIATE DATA TO MEMORY

# LDM

**Operation :** (M) ← nn

**Function :** This instruction loads the immediate value in M.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta$ LDM $\Delta$ #\$nn,\$zz	3C <sub>16</sub> , nn <sub>16</sub> , zz <sub>16</sub>	3	4

# LDX

## LOAD INDEX REGISTER X FROM MEMORY

# LDX

**Operation :**  $(X) \leftarrow (M)$

**Function :** This instruction loads the contents of M in X.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{LDX}\Delta\#\$\text{nn}$	A2 <sub>16</sub> , nn <sub>16</sub>	2	2
Zero page	$\Delta\text{LDX}\Delta\$\text{zz}$	A6 <sub>16</sub> , ZZ <sub>16</sub>	2	3
Zero page Y	$\Delta\text{LDX}\Delta\$\text{zz},\text{Y}$	B6 <sub>16</sub> , ZZ <sub>16</sub>	2	4
Absolute	$\Delta\text{LDX}\Delta\$\text{hhll}$	AE <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	4
Absolute Y	$\Delta\text{LDX}\Delta\$\text{hhll},\text{Y}$	BE <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5

# LDY

LOAD INDEX REGISTER Y FROM MEMORY

# LDY

**Operation :**  $(Y) \leftarrow (M)$

**Function :** This instruction loads the contents of M in Y.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta LDY \Delta \# \$nn$	$A0_{16}, nn_{16}$	2	2
Zero page	$\Delta LDY \Delta \$zz$	$A4_{16}, zz_{16}$	2	3
Zero page X	$\Delta LDY \Delta \$zz, X$	$B4_{16}, zz_{16}$	2	4
Absolute	$\Delta LDY \Delta \$hhll$	$AC_{16}, ll_{16}, hh_{16}$	3	4
Absolute X	$\Delta LDY \Delta \$hhll, X$	$BC_{16}, ll_{16}, hh_{16}$	3	5

# LSR

# LSR

## LOGICAL SHIFT RIGHT

**Operation :** 0 → 

b7							b0
----	--	--	--	--	--	--	----

 → 

C
---

**Function :** This instruction shifts either A or M one bit to the right such that bit 7 of the result always is set to 0, and the bit 0 is stored in C.

**Status flag:** **N :** 0  
**V :** No change  
**T :** No change  
**B :** No change  
**I :** No change  
**D :** No change  
**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.  
**C :** C is 1 when the bit 0 of either the A or the M before the operation is 1; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta$ LSR $\Delta$ A	4A <sub>16</sub>	1	2
Zero page	$\Delta$ LSR $\Delta$ \$zz	46 <sub>16</sub> , zz <sub>16</sub>	2	5
Zero page X	$\Delta$ LSR $\Delta$ \$zz,X	56 <sub>16</sub> , zz <sub>16</sub>	2	6
Absolute	$\Delta$ LSR $\Delta$ \$hhll	4E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
Absolute X	$\Delta$ LSR $\Delta$ \$hhll,X	5E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	7

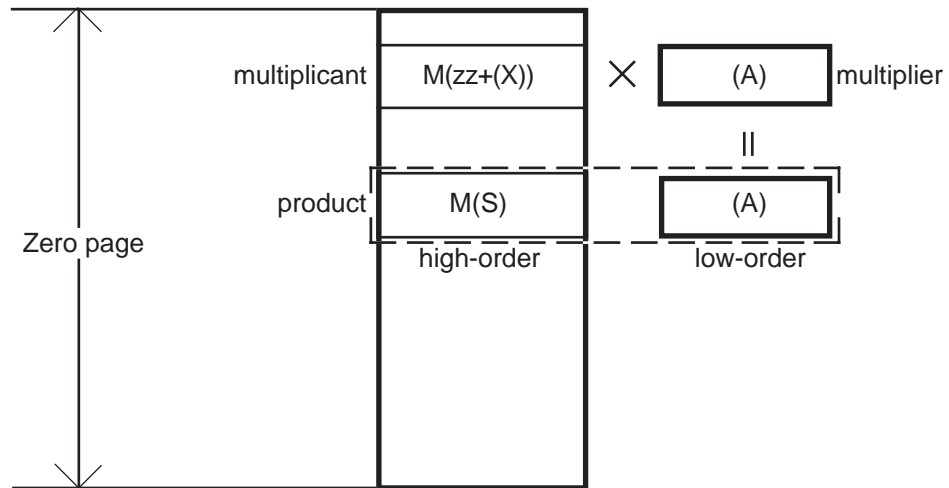
# MUL

## MULTIPLY ACCUMULATOR AND MEMORY

# MUL

**Operation :**  $M(S) \cdot (A) \leftarrow (A) \times M(zz+(X))$   
 $(S) \leftarrow (S) - 1$

**Function :** Multiplies Accumulator with the memory specified by the Zero Page X addressing mode and stores the high-order byte of the result on the Stack and the low-order byte in A.



**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page X	$\Delta$ MUL $\Delta$ \$zz,X	6216, ZZ16	2	15

Notes 1: This instruction changes the contents of S and A.

2: The MUL instruction can not be used for some products.

# NOP

NO OPERATION

# NOP

**Operation :**  $(PC) \leftarrow (PC) + 1$

**Function :** This instruction adds one to the PC but does no other operation.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle Number
Implied	$\Delta$ NOP	EA <sub>16</sub>	1	2



# ORA

## OR MEMORY WITH ACCUMULATOR

# ORA

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (A) \vee (M)$   
 When  $(T) = 1$ ,  $(M(X)) \leftarrow (M(X)) \vee (M)$

**Function :** When  $T = 0$ , this instruction transfers the contents of A and M to the ALU which performs a bit-wise “OR”, and stores the result in A.

When  $T = 1$ , this instruction transfers the contents of M(X) and the M to the ALU which performs a bit-wise OR, and stores the result in M(X). The contents of A remain unchanged, but status flags are changed.

M(X) represents the contents of memory where is indicated by X.

**Status flag:** **N** : N is 1 when bit 7 is 1 after the operation; otherwise N is 0.  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : Z is 1 when the execution result is 0; otherwise Z is 0.  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta\text{ORA}\Delta\$\text{nn}$	09 <sub>16</sub> , nn <sub>16</sub>	2	2
Zero page	$\Delta\text{ORA}\Delta\$\text{zz}$	05 <sub>16</sub> , zz <sub>16</sub>	2	3
Zero page X	$\Delta\text{ORA}\Delta\$\text{zz},\text{X}$	15 <sub>16</sub> , zz <sub>16</sub>	2	4
Absolute	$\Delta\text{ORA}\Delta\$\text{hhll}$	0D <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	4
Absolute X	$\Delta\text{ORA}\Delta\$\text{hhll},\text{X}$	1D <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
Absolute Y	$\Delta\text{ORA}\Delta\$\text{hhll},\text{Y}$	19 <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
(Indirect X)	$\Delta\text{ORA}\Delta(\$\text{zz},\text{X})$	01 <sub>16</sub> , zz <sub>16</sub>	2	6
(Indirect Y)	$\Delta\text{ORA}\Delta(\$\text{zz}),\text{Y}$	11 <sub>16</sub> , zz <sub>16</sub>	2	6

Note: When  $T=1$ , add 3 to the cycle number.

# PHA

## PUSH ACCUMULATOR ON STACK

---

# PHA

**Operation :**  $(M(S)) \leftarrow (A)$   
 $(S) \leftarrow (S) - 1$

**Function :** This instruction pushes the contents of A to the memory location designated by S, and decrements the contents of S by one.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ PHA	48 <sub>16</sub>	1	3

**Operation :**  $(M(S)) \leftarrow (PS)$   
 $(S) \leftarrow (S) - 1$

**Function :** This instruction pushes the contents of PS to the memory location designated by S and decrements the contents of S by one.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ PHP	0816	1	3

## PULL ACCUMULATOR FROM STACK

---

**Operation :**  $(S) \leftarrow (S) + 1$   
 $(A) \leftarrow (M(S))$

**Function :** This instruction increments S by one and stores the contents of the memory designated by S in A.

**Status flag:** **N** : N is 1 when bit 7 is 1 after the operation ; otherwise N is 0.  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : Z is 1 when the operation result is 0; otherwise Z is 0.  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ PLA	68 <sub>16</sub>	1	4

# PLP

## PULL PROCESSOR STATUS FROM STACK

---

# PLP

**Operation :**  $(S) \leftarrow (S) + 1$   
 $(PS) \leftarrow (M(S))$

**Function :** This instruction increments S by one and stores the contents of the memory location designated by S in PS.

**Status flag :** Value returns to the original one that was pushed in the stack.

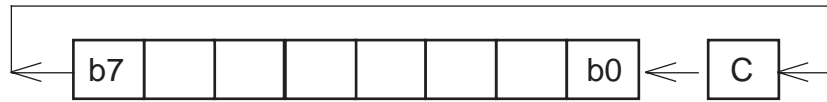
Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ PLP	28 <sub>16</sub>	1	4

# ROL

## ROTATE ONE BIT LEFT

# ROL

**Operation :**



**Function :** This instruction shifts either A or M one bit left through C. C is stored in bit 0 and bit 7 is stored in C.

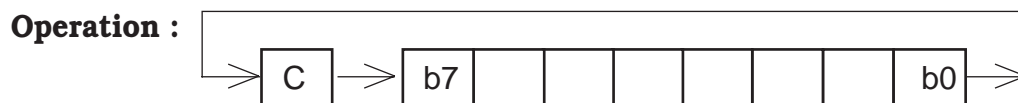
**Status flag:** **N :** N is 1 when bit 6 is 1 before the operation; otherwise N is 0.  
**V:** No change  
**T:** No change  
**B:** No change  
**I:** No change  
**D:** No change  
**Z:** Z is 1 when the operation result is 0; otherwise Z is 0.  
**C:** C is 1 when bit 7 is 1 before the operation; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta\text{ROL}\Delta A$	$2A_{16}$	1	2
Zero page	$\Delta\text{ROL}\Delta \$zz$	$26_{16}, zz_{16}$	2	5
Zero page X	$\Delta\text{ROL}\Delta \$zz,X$	$36_{16}, zz_{16}$	2	6
Absolute	$\Delta\text{ROL}\Delta \$hhll$	$2E_{16}, ll_{16}, hh_{16}$	3	6
Absolute X	$\Delta\text{ROL}\Delta \$hhll,X$	$3E_{16}, ll_{16}, hh_{16}$	3	7

# ROR

# ROR

## ROTATE ONE BIT RIGHT



**Function :** This instruction shifts either A or M one bit right through C. C is stored in bit 7 and bit 0 is stored in C.

**Status flag:**

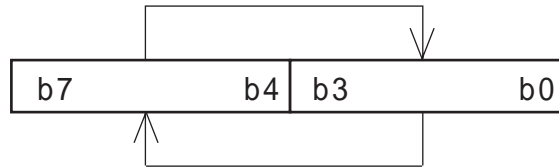
- N :** N is 1 when C is 1 before the operation; otherwise N is 0.
- V:** No change
- T:** No change
- B:** No change
- I:** No change
- D:** No change
- Z:** Z is 1 when the operation result is 0; otherwise Z is 0.
- C:** C is 1 when bit 0 is 1 before the operation; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	$\Delta$ ROR $\Delta$ A	6A <sub>16</sub>	1	2
Zero page	$\Delta$ ROR $\Delta$ \$zz	66 <sub>16</sub> , zz <sub>16</sub>	2	5
Zero page X	$\Delta$ ROR $\Delta$ \$zz,X	76 <sub>16</sub> , zz <sub>16</sub>	2	6
Absolute	$\Delta$ ROR $\Delta$ \$hhl	6E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
Absolute X	$\Delta$ ROR $\Delta$ \$hhl,X	7E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	7

## ROTATE RIGHT OF FOUR BITS

---

**Operation :**



**Function :** This instruction rotates 4 bits of the M content to the right.

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta$ RRF $\Delta$ \$zz	82 <sub>16</sub> , ZZ <sub>16</sub>	2	8



**Operation :**  $(S) \leftarrow (S) + 1$   
 $(PS) \leftarrow (M(S))$   
 $(S) \leftarrow (S) + 1$   
 $(PCL) \leftarrow (M(S))$   
 $(S) \leftarrow (S) + 1$   
 $(PCH) \leftarrow (M(S))$

**Function :** This instruction increments S by one, and stores the contents of the memory location designated by S in PS. S is again incremented by one and stores the contents of the memory location designated by S in PCL. S is again incremented by one and stores the contents of memory location designated by S in PCH.

Value returns to the original one that was pushed in the stack.

**Status flag :**  $(S) \leftarrow (S) + 1$

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta RTI$	$40_{16}$	1	6

**Operation :**  $(PCL) \leftarrow (M(S))$   
 $(S) \leftarrow (S) + 1$   
 $(PCH) \leftarrow (M(S))$   
 $(PC) \leftarrow (PC) + 1$

**Function :** This instruction increments S by one and stores the contents of the memory location designated by S in PCL. S is again incremented by one and the contents of the memory location is stored in PCH. PC is incremented by 1.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ RTS	60 <sub>16</sub>	1	6

## SUBTRACT WITH CARRY

**Operation :** When  $(T) = 0$ ,  $(A) \leftarrow (A) - (M) - \overline{(C)}$   
 When  $(T) = 1$ ,  $(M(X)) \leftarrow (M(X)) - (M) - \overline{(C)}$

**Function :** When  $T = 0$ , this instruction subtracts the value of M and the complement of C from A, and stores the results in A and C. When  $T = 1$ , the instruction subtracts the contents of M and the complement of C from the contents of M(X), and stores the results in M(X) and C.  
 A remain unchanged, but status flag are changed.  
 M(X) represents the contents of memory where is indicated by X.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.  
**V :** V is 1 when the operation result exceeds +127 or  $-128$ ; otherwise V is 0.  
**T :** No change  
**B :** No change  
**I :** No change  
**D :** No change  
**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.  
**C :** C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	$\Delta SBC \Delta \# \$nn$	E9 <sub>16</sub> , nn <sub>16</sub>	2	2
Zero page	$\Delta SBC \Delta \$zz$	E5 <sub>16</sub> , zz <sub>16</sub>	2	3
Zero page X	$\Delta SBC \Delta \$zz, X$	F5 <sub>16</sub> , zz <sub>16</sub>	2	4
Absolute	$\Delta SBC \Delta \$hhll$	ED <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	4
Absolute X	$\Delta SBC \Delta \$hhll, X$	FD <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
Absolute Y	$\Delta SBC \Delta \$hhll, Y$	F9 <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
(Indirect X)	$\Delta SBC \Delta (\$zz, X)$	E1 <sub>16</sub> , zz <sub>16</sub>	2	6
(Indirect Y)	$\Delta SBC \Delta (\$zz), Y$	F1 <sub>16</sub> , zz <sub>16</sub>	2	6

Notes 1: When  $T=1$ , add 3 to the cycle number.  
 2: When SBC instruction is executed in the decimal operation mode ( $D = 1$ ), decision of C is delayed. Accordingly, do not execute the instruction which operates C such as SEC, CLC, etc.

# SEB

# SEB

## SET BIT

---

**Operation :**  $(Ai) \leftarrow 1$ , or  
 $(Mi) \leftarrow 1$

**Function :** This instruction sets the designated bit  $i$  of  $A$  or  $M$ .

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit	$\Delta\text{SEB}\Delta i, A$	$(20i+B)_{16}$	1	2
Zero page bit	$\Delta\text{SEB}\Delta i, \$zz$	$(20i+F)_{16}, ZZ_{16}$	2	5

**Operation :** (C) ← 1

**Function :** This instruction sets C.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : No change  
**C** : 1

Addressing mode	Statement	Machine code	Byte number	Cycle number
Implied	ΔSEC	3816	1	2

**Operation :** (D) ← 1

**Function :** This instruction set D.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : No change  
**D** : 1  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔSED	F8 <sub>16</sub>	1	2

# SEI

## SET INTERRUPT DISABLE FLAG

# SEI

**Operation :**  $(I) \leftarrow 1$

**Function :** This instruction sets I.

**Status flag:** **N** : No change  
**V** : No change  
**T** : No change  
**B** : No change  
**I** : 1  
**D** : No change  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ SEI	7816	1	2

# SET

## SET TRANSFER FLAG

# SET

**Operation :**  $(T) \leftarrow 1$

**Function :** This instruction sets T.

**Status flag:** **N** : No change  
**V** : No change  
**T** : 1  
**B** : No change  
**I** : No change  
**D** : No change  
**Z** : No change  
**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ SET	3216	1	2



# STA

## STORE ACCUMULATOR IN MEMORY

# STA

**Operation :**  $(M) \leftarrow (A)$

**Function :** This instruction stores the contents of A in M.  
The contents of A does not change.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta STA \Delta \$zz$	85 <sub>16</sub> , ZZ <sub>16</sub>	2	4
Zero page X	$\Delta STA \Delta \$zz, X$	95 <sub>16</sub> , ZZ <sub>16</sub>	2	5
Absolute	$\Delta STA \Delta \$hhll$	8D <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5
Absolute X	$\Delta STA \Delta \$hhll, X$	9D <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
Absolute Y	$\Delta STA \Delta \$hhll, Y$	99 <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	6
(Indirect X )	$\Delta STA \Delta (\$zz, X)$	81 <sub>16</sub> , ZZ <sub>16</sub>	2	7
(Indirect Y)	$\Delta STA \Delta (\$zz), Y$	91 <sub>16</sub> , ZZ <sub>16</sub>	2	7

# STP

## STOP

# STP

**Operation :** CPU ← Stand-by state (Oscillation stopped)

**Function :** This instruction resets the oscillation control F/F and the oscillation stops. Reset or interrupt input is needed to wake up from this mode.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ STP	42 <sub>16</sub>	1	2

Note: If the STP instruction is disabled the cycle number will be 2 (same in operation as NOP). However, disabling this instruction is an optional feature; therefore, consult the specifications for the particular chip in question.

# STX

## STORE INDEX REGISTER X IN MEMORY

# STX

**Operation :**  $(M) \leftarrow (X)$

**Function :** This instruction stores the contents of X in M. The contents of X does not change.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta\text{STX}\Delta\$zz$	86 <sub>16</sub> , zz <sub>16</sub>	2	4
Zero page Y	$\Delta\text{STX}\Delta\$zz,Y$	96 <sub>16</sub> , zz <sub>16</sub>	2	5
Absolute	$\Delta\text{STX}\Delta\$hhll$	8E <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5

# STY

## STORE INDEX REGISTER Y IN MEMORY

# STY

**Operation :**  $(M) \leftarrow (Y)$

**Function :** This instruction stores the contents of Y in M.  
The contents of Y does not change.

**Status flag:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta\text{STY}\Delta\$zz$	84 <sub>16</sub> , ZZ <sub>16</sub>	2	4
Zero page X	$\Delta\text{STY}\Delta\$zz,X$	94 <sub>16</sub> , ZZ <sub>16</sub>	2	5
Absolute	$\Delta\text{STY}\Delta\$hhll$	8C <sub>16</sub> , ll <sub>16</sub> , hh <sub>16</sub>	3	5

# TAX

# TAX

## TRANSFER ACCUMULATOR TO INDEX REGISTER X

---

**Operation :**  $(X) \leftarrow (A)$

**Function :** This instruction stores the contents of A in X. The contents of A does not change.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TAX	AA <sub>16</sub>	1	2

# TAY

# TAY

## TRANSFER ACCUMULATOR TO INDEX REGISTER Y

---

**Operation :**  $(Y) \leftarrow (A)$

**Function :** This instruction stores the contents of A in Y. The contents of A does not change.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the operation result is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TAY	A816	1	2

**Operation :**  $(M) = 0 ?$

**Function :** This instruction tests whether the contents of M are “0” or not and modifies the N and Z.

**Status flag:** **N :** N is 1 when bit 7 of M is 1; otherwise N is 0.

**V :** No change

**T :** No change

**B :** No change

**I :** No change

**D :** No change

**Z :** Z is 1 when the M content is 0; otherwise Z is 0.

**C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	$\Delta TST \Delta \$zz$	64 <sub>16</sub> , ZZ <sub>16</sub>	2	3

**Operation :**  $(X) \leftarrow (S)$

**Function :** This instruction transfers the contents of S in X.

**Status flag:** **N** : N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V** : No change

**T** : No change

**B** : No change

**I** : No change

**D** : No change

**Z** : Z is 1 when the operation result is 0; otherwise Z is 0.

**C** : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TSX	BA <sub>16</sub>	1	2



# TXA

# TXA

## TRANSFER INDEX REGISTER X TO ACCUMULATOR

---

**Operation :** (A) ← (X)

**Function :** This instruction stores the contents of X in A.

**Status flag: N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V:** No change

**T:** No change

**B:** No change

**I:** No change

**D:** No change

**Z:** Z is 1 when the operation result is 0; otherwise Z is 0.

**C:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TXA	8A <sub>16</sub>	1	2

# TXS

# TXS

## TRANSFER INDEX REGISTER **X** TO **S** STACK POINTER

---

**Operation :** (S)  $\leftarrow$  (X)

**Function :** This instruction stores the contents of X in S.

**Status flag** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TXS	9A <sub>16</sub>	1	2

# TYA

# TYA

## TRANSFER INDEX REGISTER Y TO ACCUMULATOR

---

**Operation :**  $(A) \leftarrow (Y)$

**Function :** This instruction stores the contents of Y in A.

**Status flag:** **N :** N is 1 when bit 7 is 1 after the operation; otherwise N is 0.

**V:** No change

**T:** No change

**B:** No change

**I:** No change

**D:** No change

**Z:** Z is 1 when the operation result is 0; otherwise Z is 0.

**C:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ TYA	98 <sub>16</sub>	1	2

**Operation :** CPU ← Wait state

**Function :** The WIT instruction stops the internal clock but not the oscillation of the oscillation circuit is not stopped. CPU starts its function after the Timer X over flows (comes to the terminal count). All registers or internal memory contents except Timer X will not change during this mode. (Of course needs  $V_{DD}$ ).

**Status flag :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	$\Delta$ WIT	C2 <sub>16</sub>	1	2

# NOTES ON USE

---

## 4. NOTES ON USE

### 4.1 Notes on interrupts

#### 4.1.1 Setting for interrupt request bit and interrupt enable bit

To set an interrupt request bit and an interrupt enable bit for interrupts, execute as the following sequence:

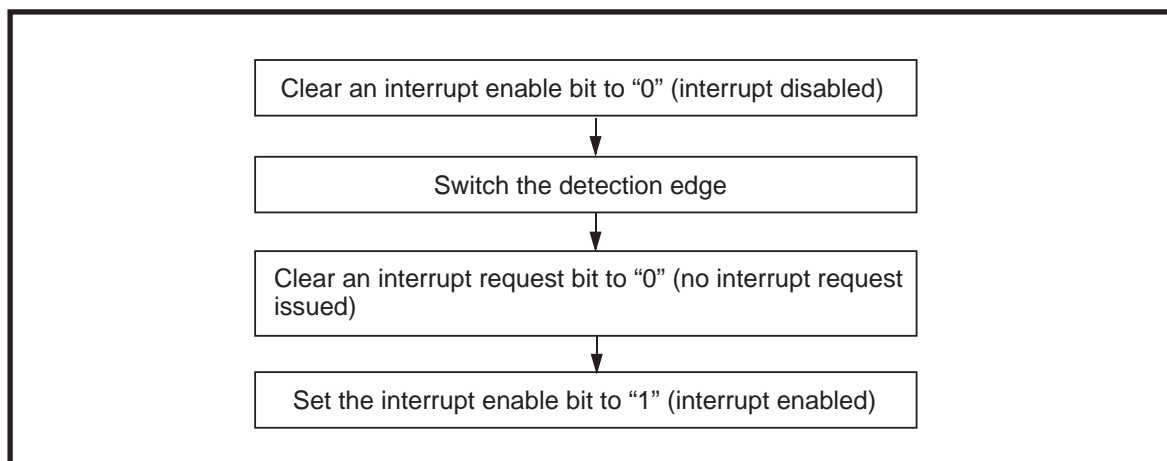
- ① Clear an interrupt request bit to “0” (no interrupt request issued).
- ② Set an interrupt enable bit to “1” (interrupts enabled).

#### ●Reason

If the above setting are performed simultaneously with one instruction, an unnecessary interrupt processing routine is executed. Because an interrupt enable bit is set to “1” (interrupts enabled) before an interrupt request bit is cleared to “0.”

#### 4.1.2 Switching of detection edge

For the products able to switch the external interrupt detection edge, switch it as the following sequence.



**Fig. 4.1.1 Switching sequence of detection edge**

#### ●Reason

The interrupt circuit recognizes the switching of the detection edge as the change of external input signals. This may cause to execute an unnecessary interrupt processing routine.

### 4.1.3 Distinction of interrupt request bit

When executing the BBC or BBS instruction to an interrupt request (request distinguish) bit of an interrupt request register (interrupt request distinguish register) immediately after this bit is set to "0" by using a data transfer instruction\*, execute one or more instructions before executing the BBC or BBS instruction.

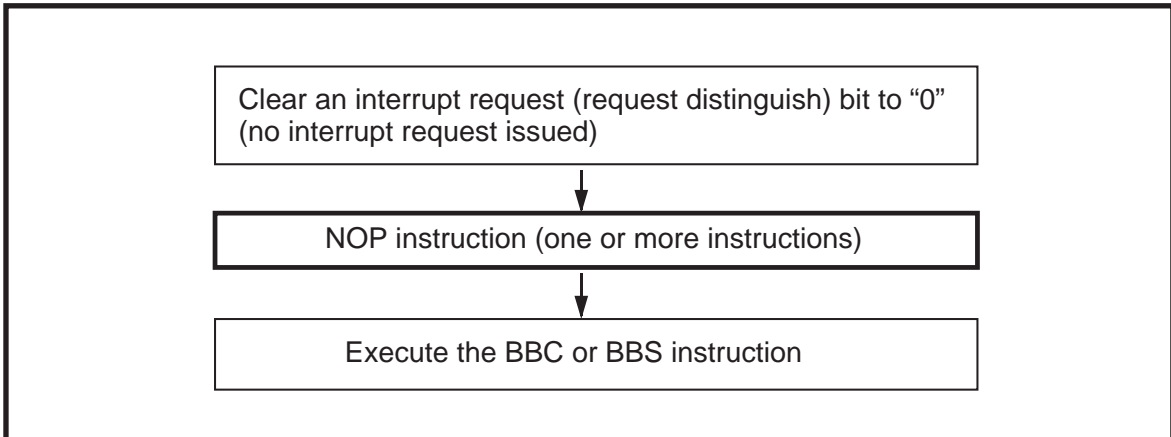


Fig. 4.1.2 Distinction sequence of interrupt request bit

\* Data transfer instruction : LDM, LDA, STA, STX, STY

#### ●Reason

If the BBC or BBS instruction is executed immediately after an interrupt request (request distinguish) bit of an interrupt request register (interrupt request distinguish register) is cleared to "0," the value of the interrupt request (request distinguish) bit before being cleared to "0" is read.

# NOTES ON USE

## 4.2 Notes on programming

### 4.2.1 Processor Status Register

#### (1) Initialization of Processor Status Register

Flags which affect program execution must be initialized after a reset. In particular, it is essential to initialize the T and D flags because they have an important effect on calculations.

●Reason

After a reset, the contents of processor status register (PS) are undefined except for the I flag which is "1."

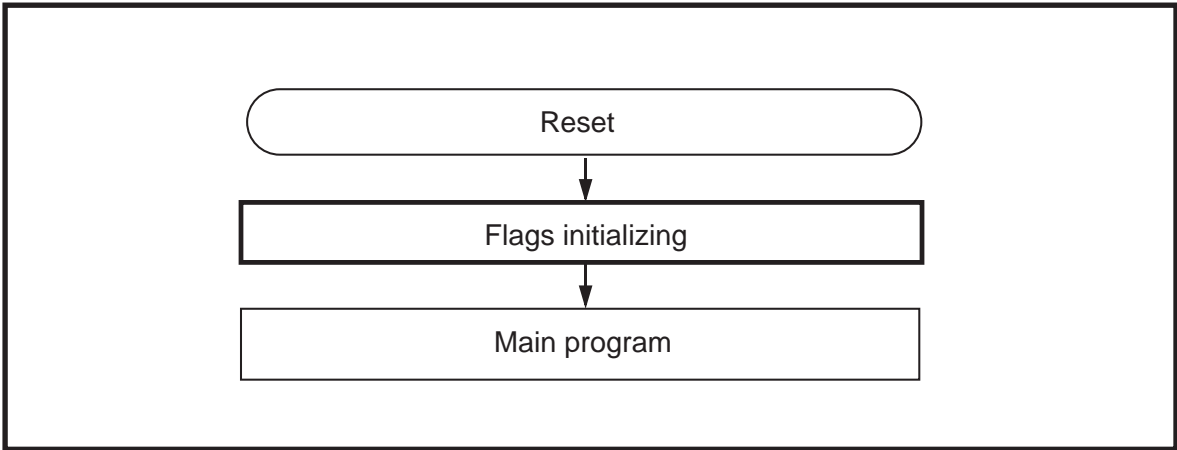


Fig. 4.2.1 Initialization of flags in Processor Status Register

#### (2) How to reference Processor Status Register

To reference the contents of the processor status register (PS), execute the PHP instruction once then read the contents of (S + 1). If necessary, execute the PLP instruction to return the PS to its original status.

A NOP instruction should be executed after every PLP instruction.

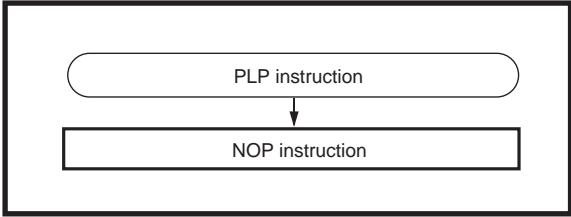


Fig. 4.2.2 PLP instruction execution sequence

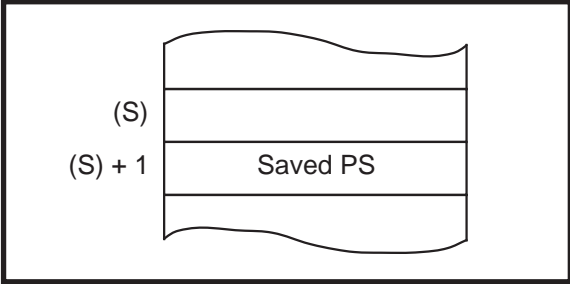


Fig. 4.2.3 Stack memory contents after PHP instruction execution

## 4.2.2 BRK instruction

### (1) Method detecting interrupt source

It can be detected that the BRK instruction interrupt event or the least priority interrupt event by referring the stored B flag state. Refer the stored B flag state in the interrupt routine, in this case.

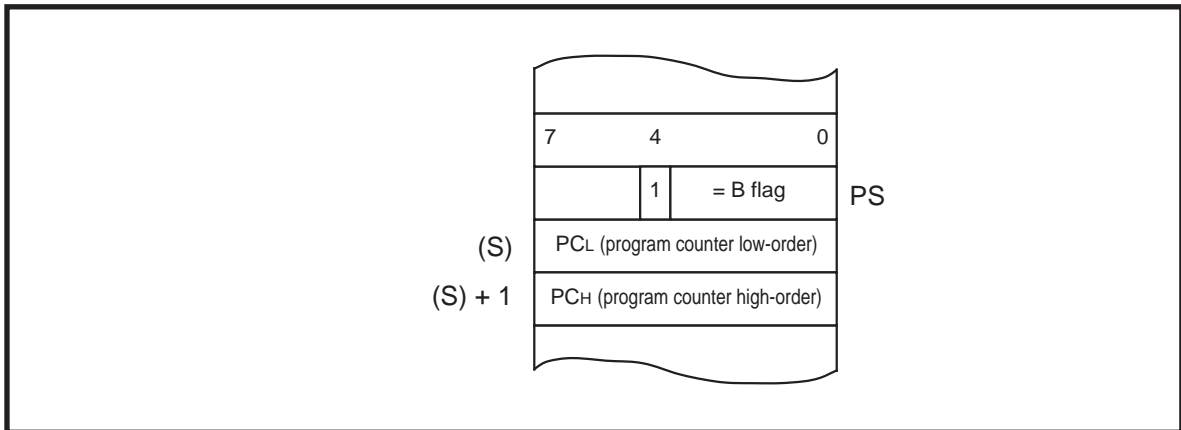


Fig. 4.2.4 Contents of stack memory in interrupt processing routine

### (2) Interrupt priority level

At the following status,

- ① the interrupt request bit has set to "1."
- ② the interrupt enable bit has set to "1."
- ③ the interrupt disable flag (I) has set to "1."

If the BRK instruction is executed, the interrupt disable state is cancelled and it becomes in the interrupt enable state. So that the requested interrupts (the interrupts that corresponding to their request bits have set to "1") are accepted.

## 4.2.3 Decimal calculations

### (1) Execution of Decimal calculations

The ADC and SBC are the only instructions which will yield proper decimal results in decimal mode. To calculate in decimal notation, set the decimal mode flag (D) to "1" with the SED instruction. After executing the ADC or SBC instruction, execute another instruction before executing the SEC, CLC, or CLD instruction.



# NOTES ON USE

---

## (2) Status flags in decimal mode

When decimal mode is selected ( $D = 1$ ), the values of three of the flags in the status register (the flags N, V, and Z) are invalid after a ADC or SBC instruction is executed. The carry flag (C) is set to "1" if a carry is generated as a result of the calculation, or is cleared to "0" if a borrow is generated. To determine whether a calculation has generated a carry, the C flag must be initialized to "0" before each calculation. To check for a borrow, the C flag must be initialized to "1" before each calculation.

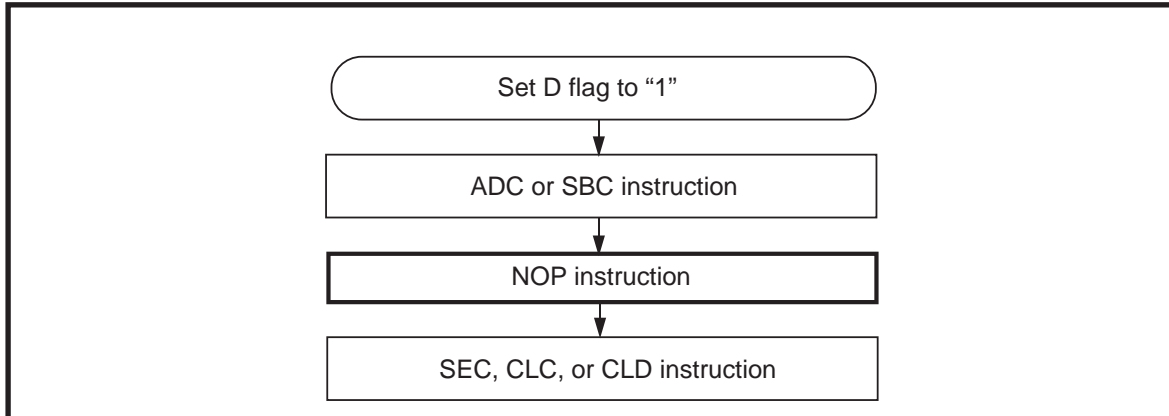


Fig. 4.2.5 Status flags in decimal mode

## 4.2.4 JMP instruction

When using the JMP instruction in indirect addressing mode, do not specify the last address on a page as an indirect address.

# APPENDIX 1

## Instruction Cycles in each Addressing Mode

### APPENDIX 1. Instruction Cycles in each Addressing Mode

Clock  $\phi$  controls the system timing of 740 Family. The SYNC signal and the value of PC (Program Counter) are output in every instruction fetch cycle. The Op-Code is fetched during the next half-period of  $\phi$ . The instruction decoder of CPU decodes this Op-Code and determines the following how to execute the instruction. The instruction timings of all addressing modes are described on the following pages.

In these figures,  $\phi$ , SYNC, R/W (RD, WR), ADDR (ADDR<sub>L</sub>, ADDR<sub>H</sub>), and DATA are internal signals of the single-chip microcomputer; therefore, these signals can be investigated only in the microprocessor mode.

The combination of these signals differs according to the microcomputer's type. The following table lists the valid signal for each product.

Valid signal for each product

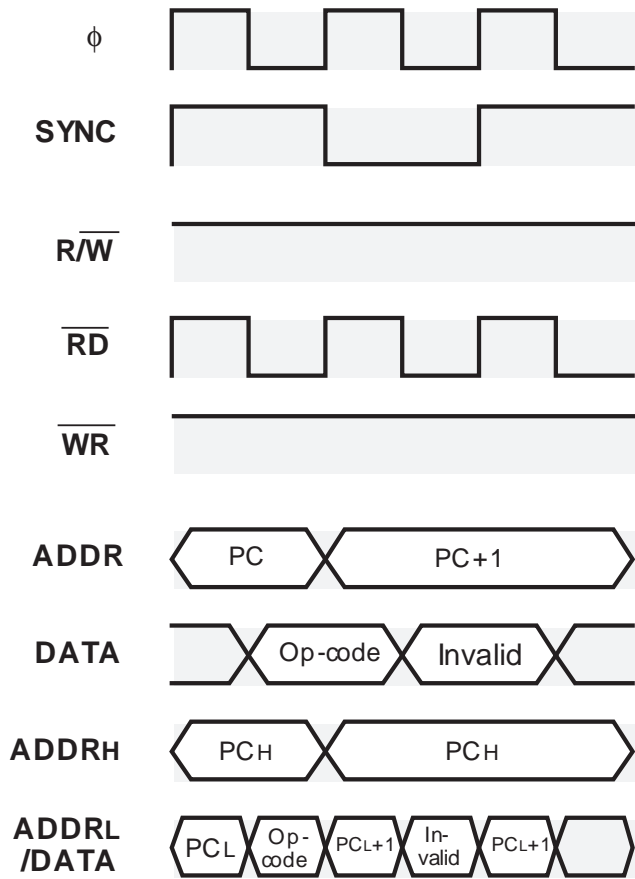
Type	$\phi$	SYNC	R/W	$\bar{R}$	$\bar{W}$	ADDR	DATA	ADDR <sub>H</sub>	ADDR <sub>L</sub> /DATA
M507XX M509XX M374XX (Except M37451)	○	○	○			○	○		
M38XXX M375XX M37451	○	○		○	○	○	○		
M50734	○	○		○	○			○	○

# IMPLIED

Instructions	: $\Delta$ CLC	$\Delta$ SEC
	$\Delta$ CLD	$\Delta$ SED
	$\Delta$ CLI	$\Delta$ SEI
	$\Delta$ CLT	$\Delta$ SET
	$\Delta$ CLV	$\Delta$ TAX
	$\Delta$ DEX	$\Delta$ TAY
	$\Delta$ DEY	$\Delta$ TSX
	$\Delta$ INX	$\Delta$ TXA
	$\Delta$ INY	$\Delta$ TXS
	$\Delta$ NOP	$\Delta$ TYA

Byte length : 1  
 Cycle number : 2

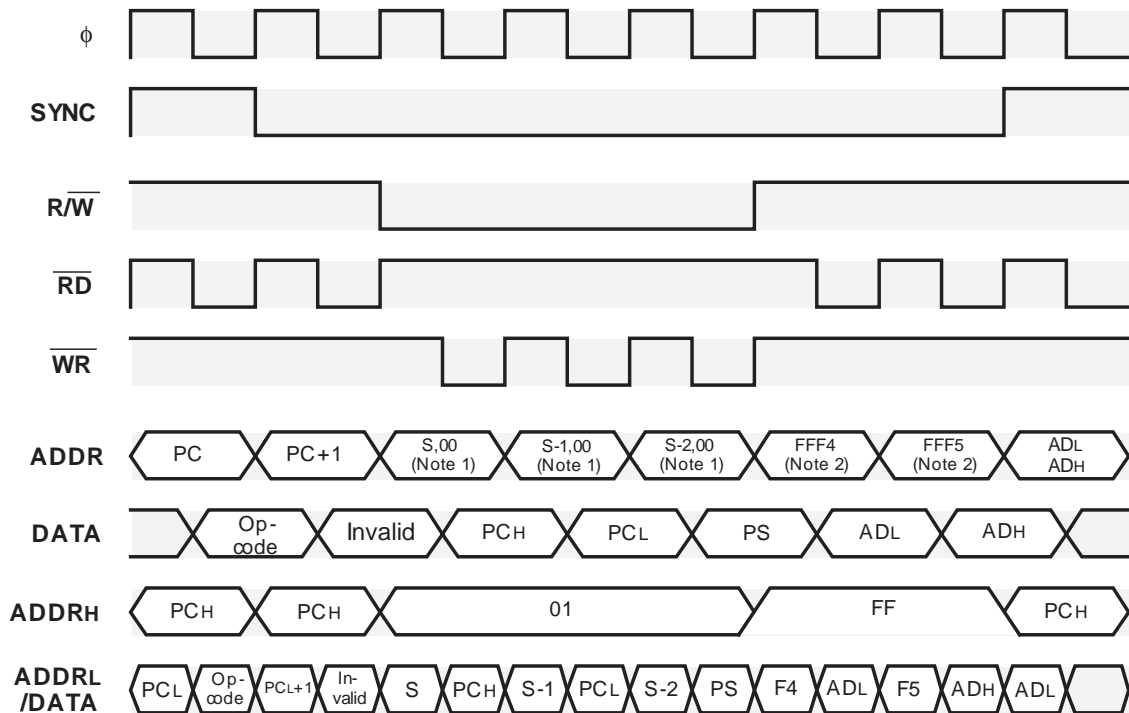
Timing :



# IMPLIED

Instruction :  $\Delta$ BRK  
 Byte length : 1  
 Cycle number : 7

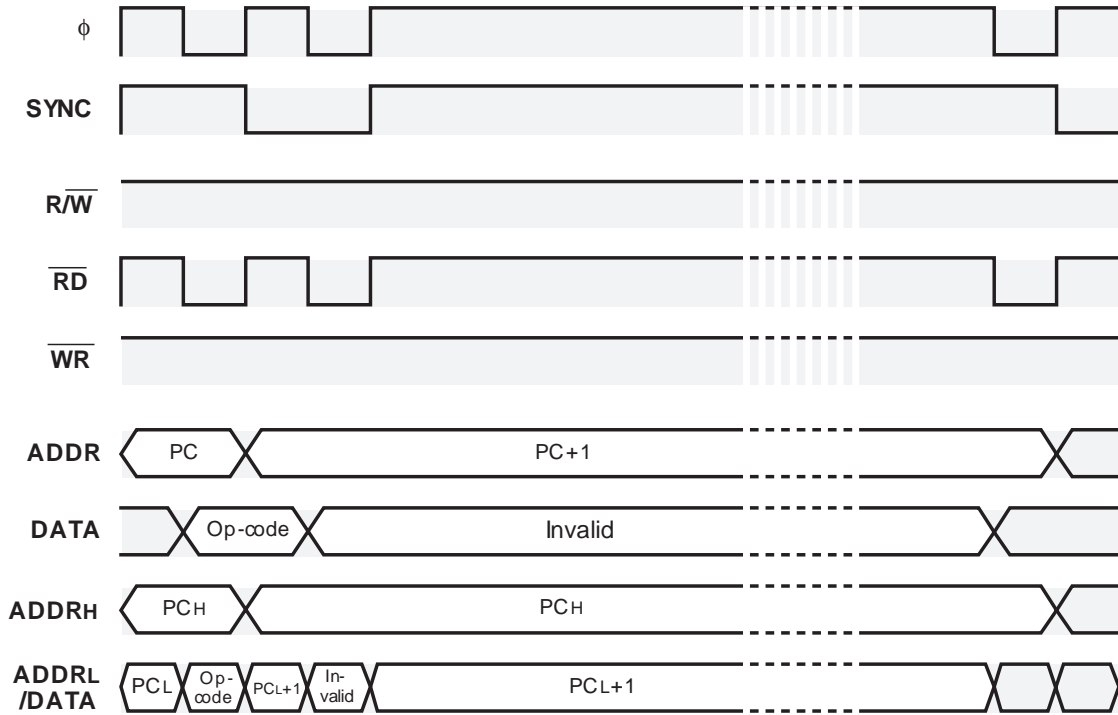
Timing :



Notes 1 : Some products are "01" or content of SPS flag.  
 2 : Some products differ the address.

# IMPLIED

Instructions :  $\Delta$ STP  
                    $\Delta$ WIT  
 Byte length : 1  
 Timing :

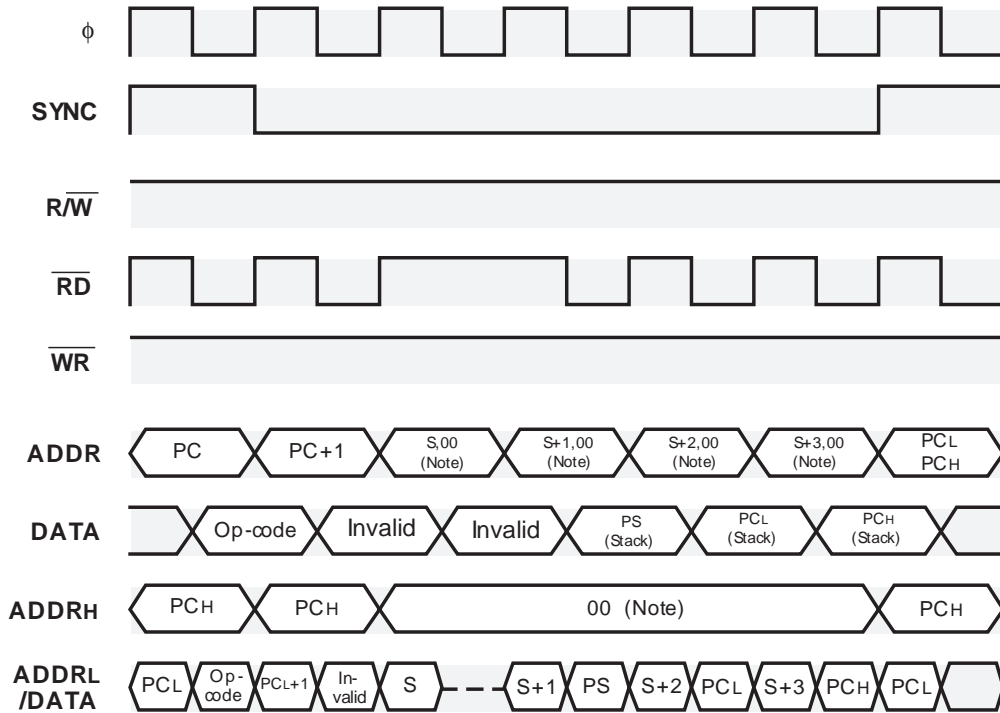


Return from standby state is executed by external interrupt.  
 Return from wait state is executed by internal or external interrupt.

# IMPLIED

Instruction :  $\Delta$ RTI  
 Byte length : 1  
 Cycle number : 6

Timing :

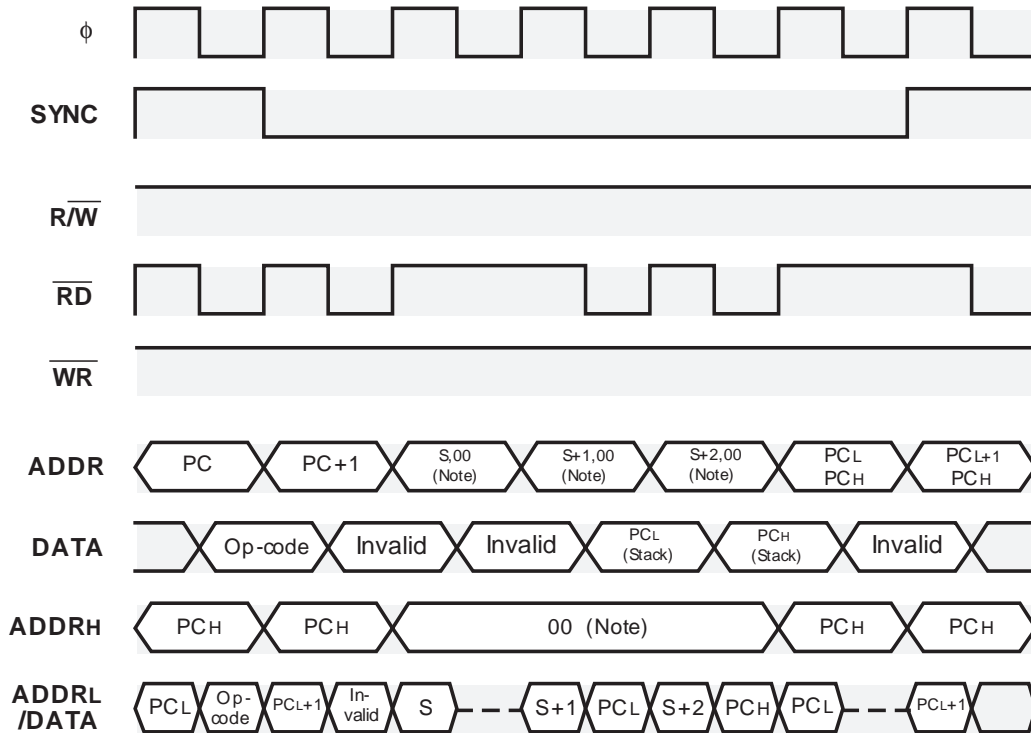


Note: Some products are "01" or content of SPS flag.

# IMPLIED

Instruction :  $\Delta$ RTS  
 Byte length : 1  
 Cycle number : 6

Timing :

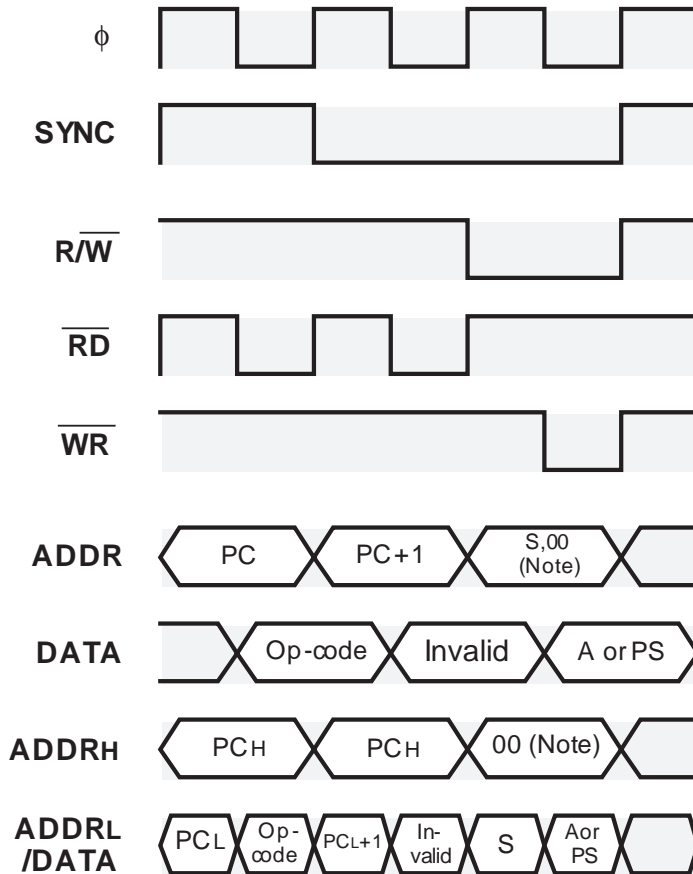


Note: Some products are "01" or content of SPS flag.

# IMPLIED

---

Instructions :  $\Delta$ PHA  
                    $\Delta$ PHP  
 Byte length : 1  
 Cycle number : 3  
 Timing :

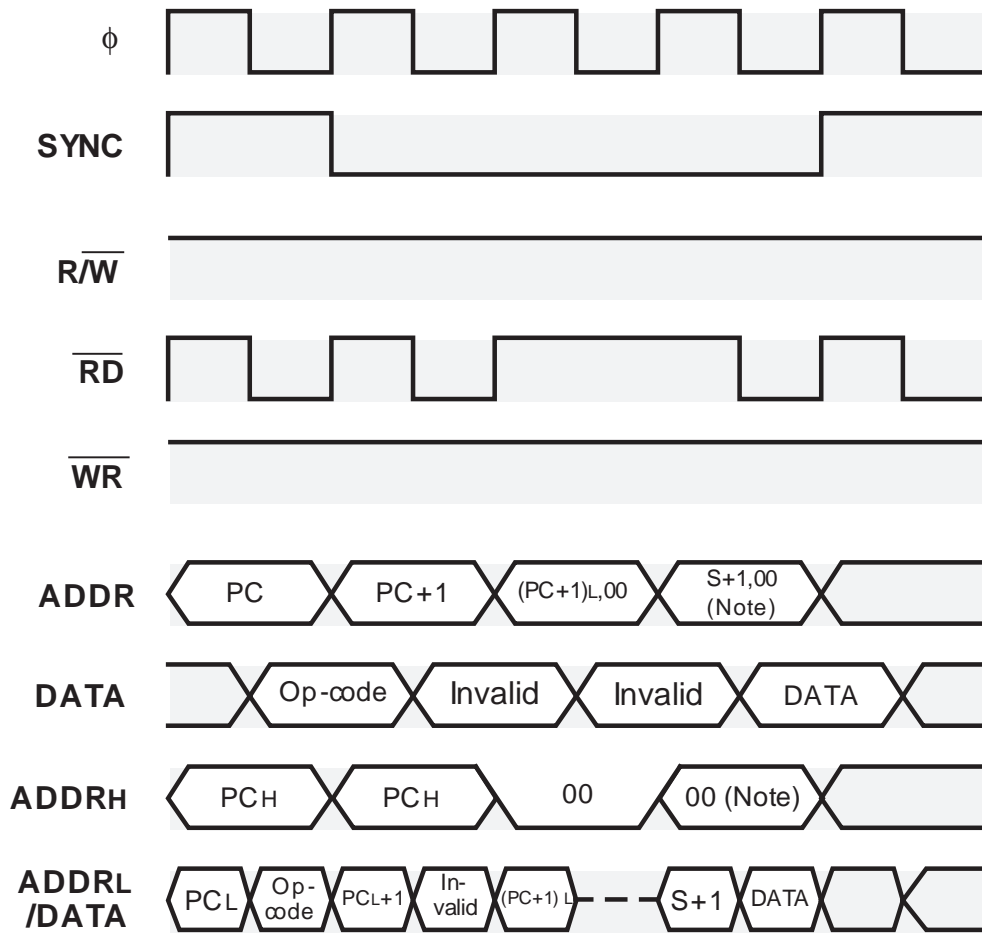


Note: Some products are "01" or content of SPS flag.



# IMPLIED

Instructions :  $\Delta$ PLA  
                    $\Delta$ PLP  
 Byte length : 1  
 Cycle number : 4  
 Timing :



Note: Some products are "01" or content of SPS flag.

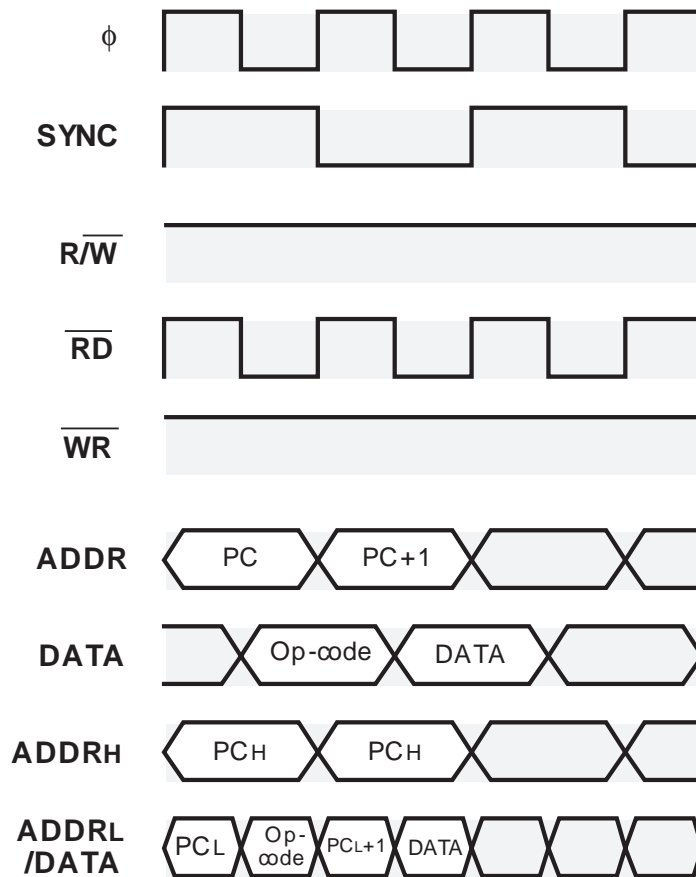
[T=0]

## IMMEDIATE

Instructions :  $\Delta$ ADC $\Delta$ #\$nn (T=0)  
 $\Delta$ AND $\Delta$ #\$nn (T=0)  
 $\Delta$ CMP $\Delta$ #\$nn (T=0)  
 $\Delta$ CPX $\Delta$ #\$nn  
 $\Delta$ CPY $\Delta$ #\$nn  
 $\Delta$ EOR $\Delta$ #\$nn (T=0)  
 $\Delta$ LDA $\Delta$ #\$nn (T=0)  
 $\Delta$ LDX $\Delta$ #\$nn  
 $\Delta$ LDY $\Delta$ #\$nn  
 $\Delta$ ORA $\Delta$ #\$nn (T=0)  
 $\Delta$ SBC $\Delta$ #\$nn (T=0)

Byte length : 2  
 Cycle number : 2

Timing :



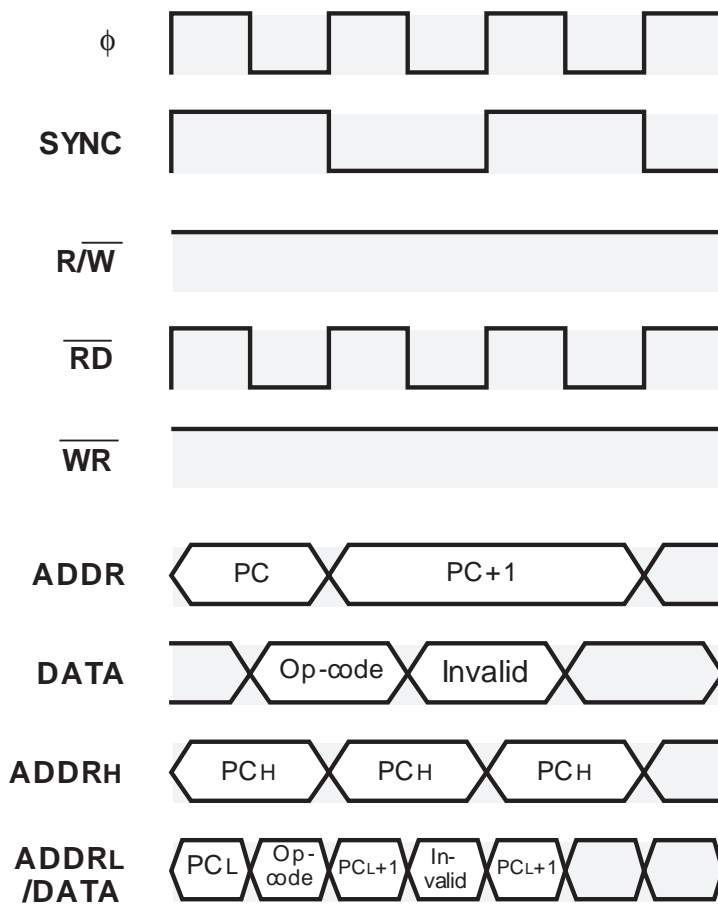
# ACCUMULATOR

---

Instructions :  $\Delta$ ASL  $\Delta$ A  
                    $\Delta$ DEC  $\Delta$ A  
                    $\Delta$ INC  $\Delta$ A  
                    $\Delta$ LSR  $\Delta$ A  
                    $\Delta$ ROL  $\Delta$ A  
                    $\Delta$ ROR  $\Delta$ A

Byte length : 1  
 Cycle number : 2

Timing :



## ACCUMULATOR BIT RELATIVE

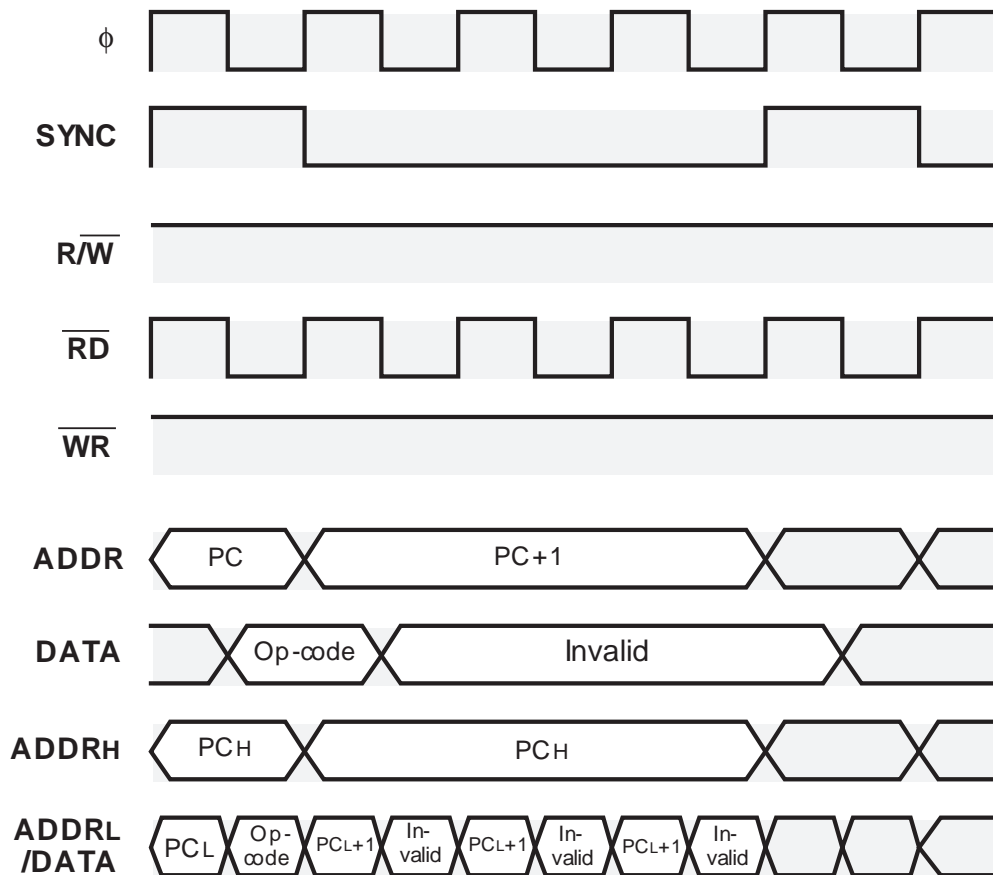
---

Instructions :  $\Delta\text{BBC}\Delta i, A, \$hhll$   
 $\Delta\text{BBS}\Delta i, A, \$hhll$

Byte length : 2

(1) With no branch  
 Cycle number : 4

Timing :



## ACCUMULATOR BIT RELATIVE

Instructions :  $\Delta\text{BBC}\Delta i, A, \$hhll$

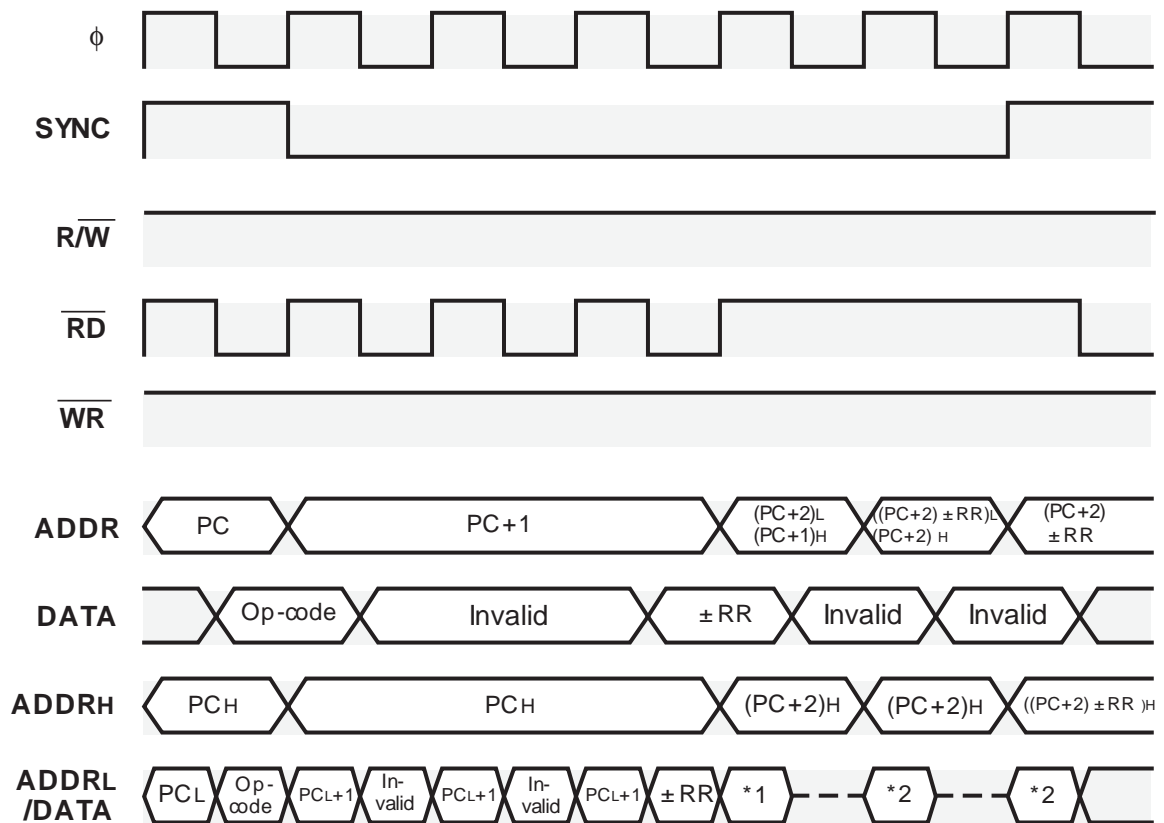
$\Delta\text{BBS}\Delta i, A, \$hhll$

Byte length : 2

(2) With branch

Cycle number : 6

Timing :



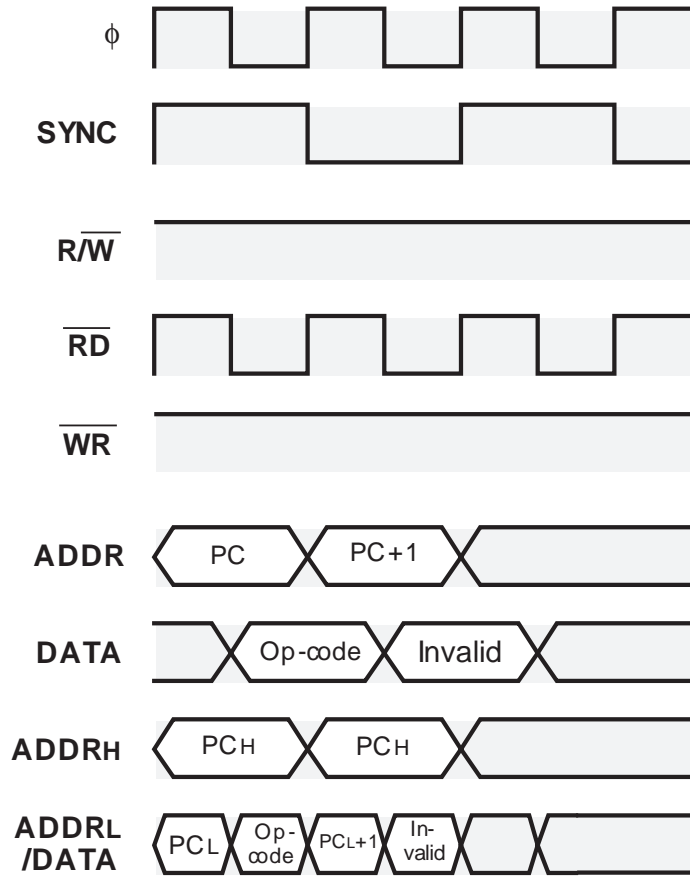
RR : Offset address

\*1 :  $(PC+1)_L$

\*2 :  $((PC+2) \pm RR)_L$

# ACCUMULATOR BIT

Instructions :  $\Delta\text{CLB}\Delta i, A$   
                $\Delta\text{SEB}\Delta i, A$   
 Byte length : 1  
 Cycle number : 2  
 Timing :



# BIT RELATIVE

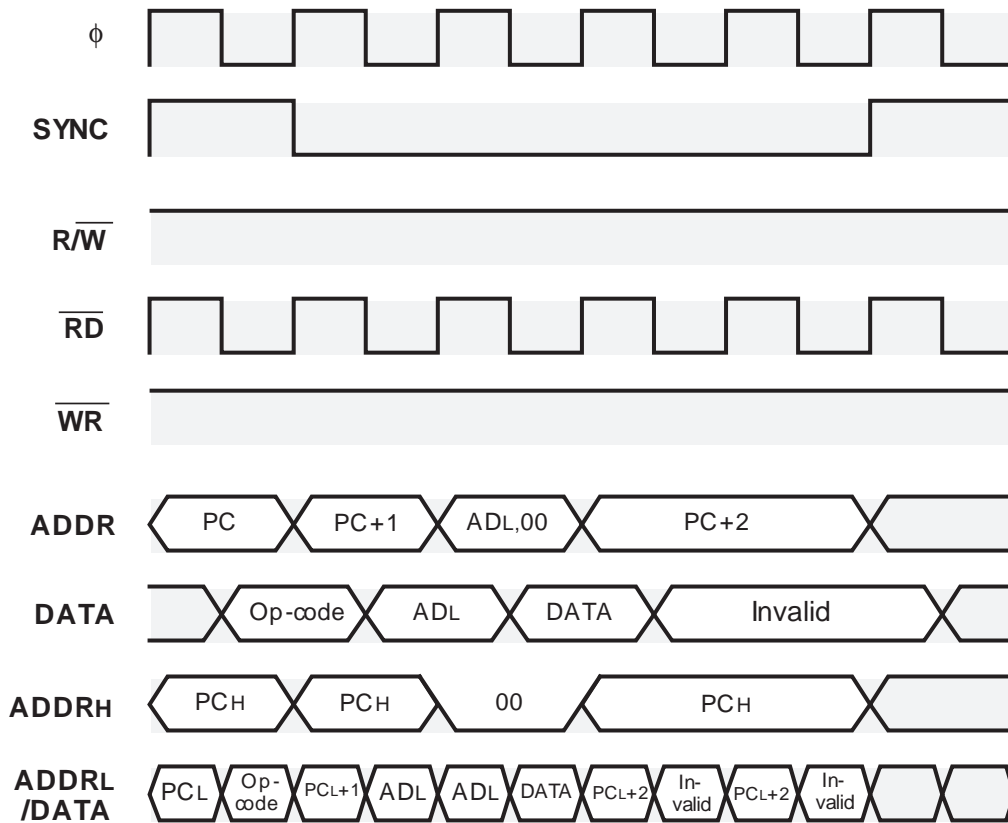
---

Instructions :  $\Delta BBC \Delta i, \$zz, \$hll$   
 $\Delta BBS \Delta i, \$zz, \$hll$

Byte length : 3

(1) With no branch  
 Cycle number : 5

Timing :



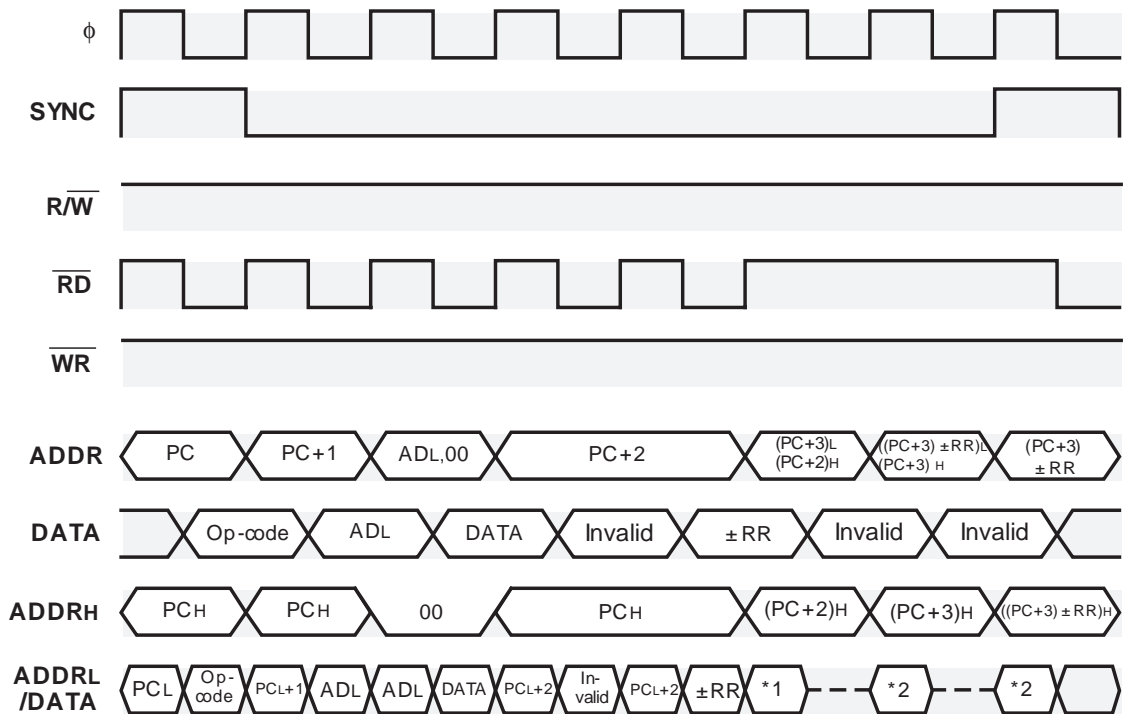
# BIT RELATIVE

Instructions :  $\Delta BBC\Delta i, \$zz, \$hhll$   
 $\Delta BBS\Delta i, \$zz, \$hhll$

Byte length : 3

(2) With branch  
 Cycle number : 7

Timing :



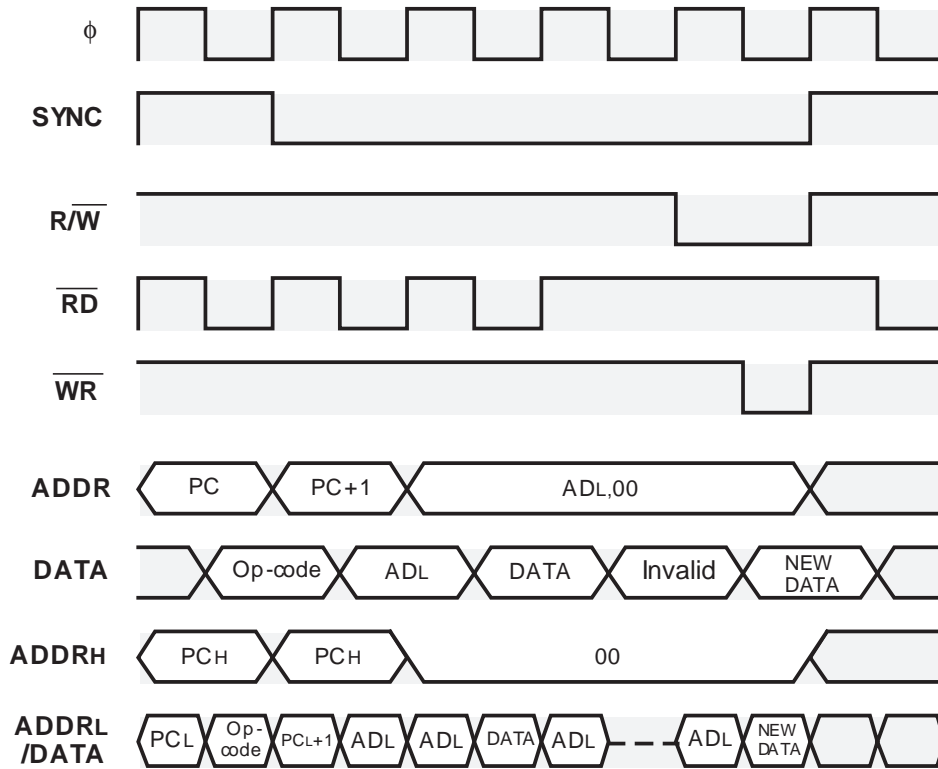
RR : Offset address  
 \*1 : (PC+3)L  
 \*2 : ((PC+3) ± RR)L



# ZERO PAGE BIT

---

Instructions :  $\Delta\text{CLB}\Delta i, \$zz$   
                    $\Delta\text{SEB}\Delta i, \$zz$   
 Byte length : 2  
 Cycle number : 5  
 Timing :



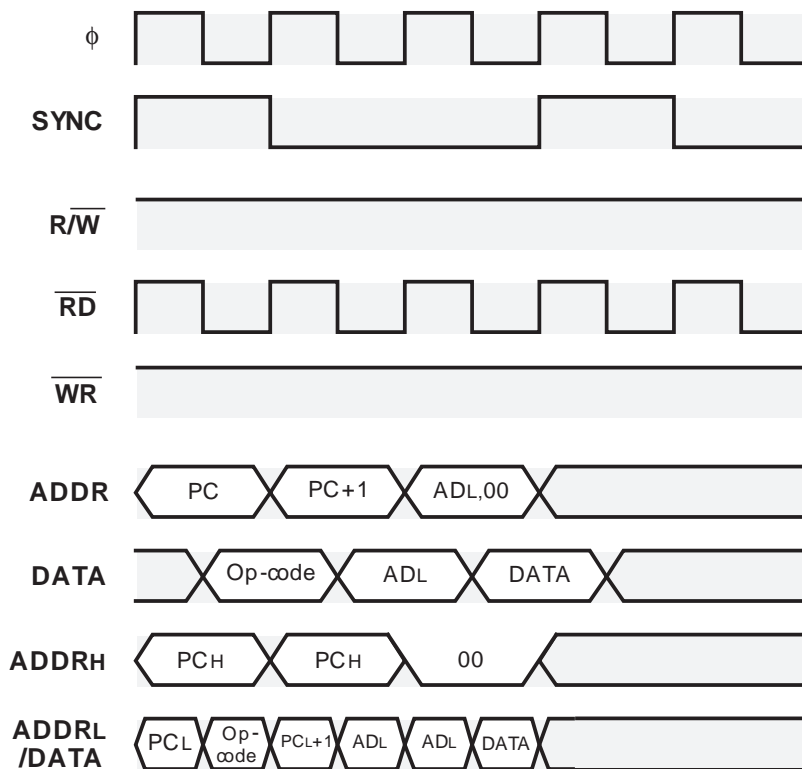
[T=0]

## ZERO PAGE

Instructions :  $\Delta$ ADC  $\Delta$ \$zz (T=0)  
 $\Delta$ AND  $\Delta$ \$zz (T=0)  
 $\Delta$ BIT  $\Delta$ \$zz  
 $\Delta$ CMP  $\Delta$ \$zz (T=0)  
 $\Delta$ CPX  $\Delta$ \$zz  
 $\Delta$ CPY  $\Delta$ \$zz  
 $\Delta$ EOR  $\Delta$ \$zz (T=0)  
 $\Delta$ LDA  $\Delta$ \$zz (T=0)  
 $\Delta$ LDX  $\Delta$ \$zz  
 $\Delta$ LDY  $\Delta$ \$zz  
 $\Delta$ ORA  $\Delta$ \$zz (T=0)  
 $\Delta$ SBC  $\Delta$ \$zz (T=0)  
 $\Delta$ TST  $\Delta$ \$zz

Byte length : 2  
 Cycle number : 3

Timing :



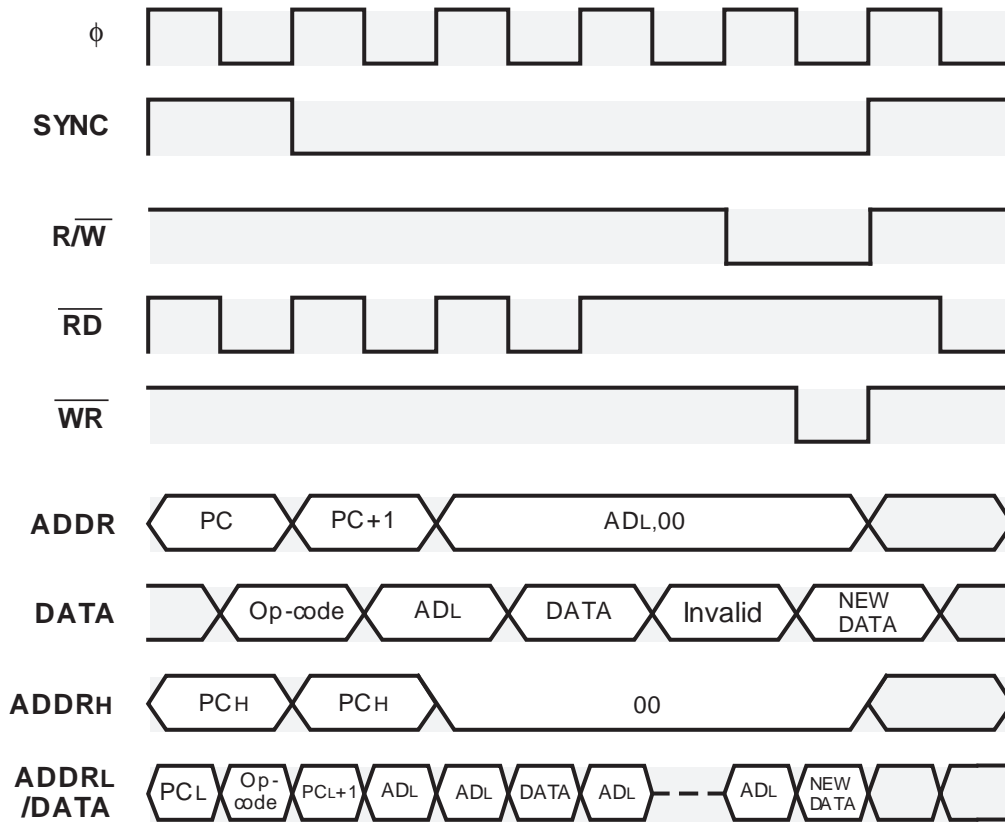
# ZERO PAGE

---

Instructions :  $\Delta$ ASL  $\Delta$ \$zz  
 $\Delta$ COM  $\Delta$ \$zz  
 $\Delta$ DEC  $\Delta$ \$zz  
 $\Delta$ INC  $\Delta$ \$zz  
 $\Delta$ LSR  $\Delta$ \$zz  
 $\Delta$ ROL  $\Delta$ \$zz  
 $\Delta$ ROR  $\Delta$ \$zz

Byte length : 2  
 Cycle number : 5

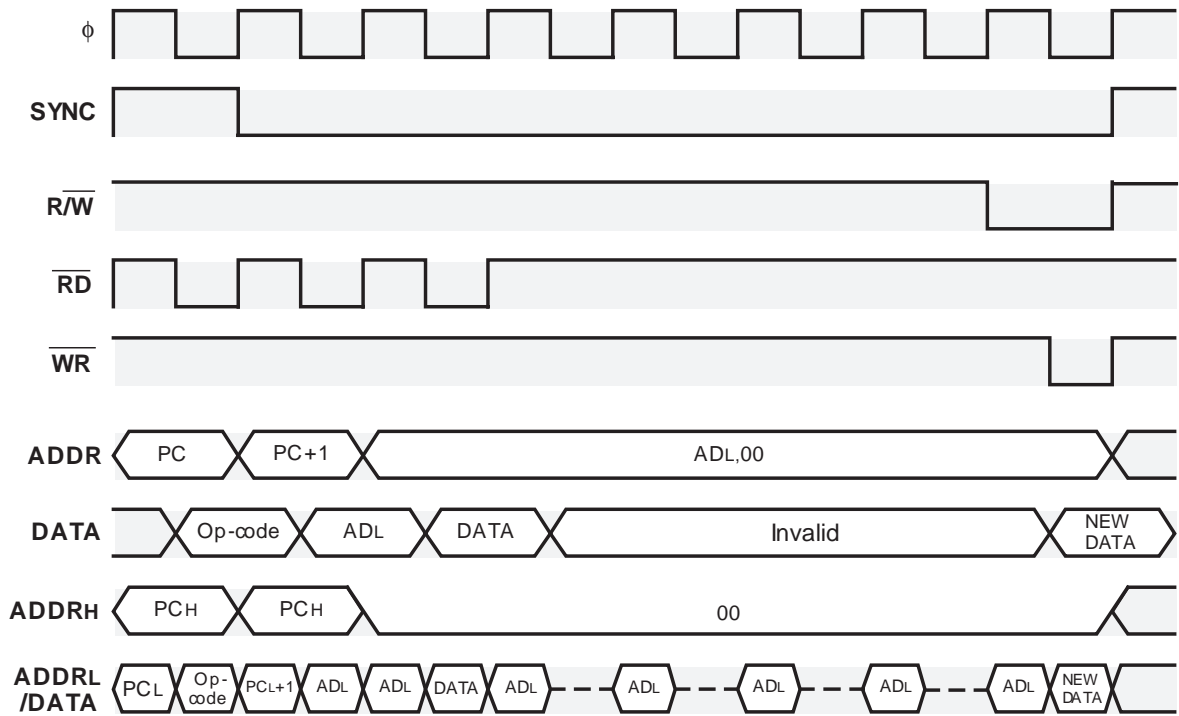
Timing :



# ZERO PAGE

Instruction :  $\Delta RRF \Delta \$zz$   
 Byte length : 2  
 Cycle number : 8

Timing :

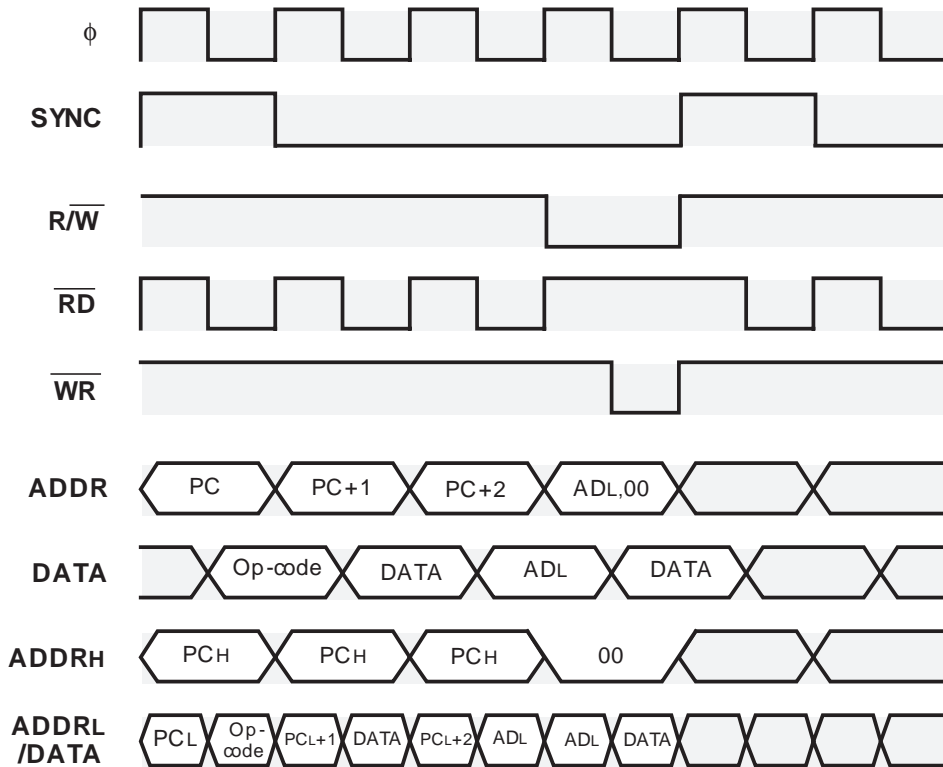


# ZERO PAGE

---

Instruction :  $\Delta$ LDM $\Delta$ # $\$$ nn, $\$$ zz  
 Byte length : 3  
 Cycle number : 4

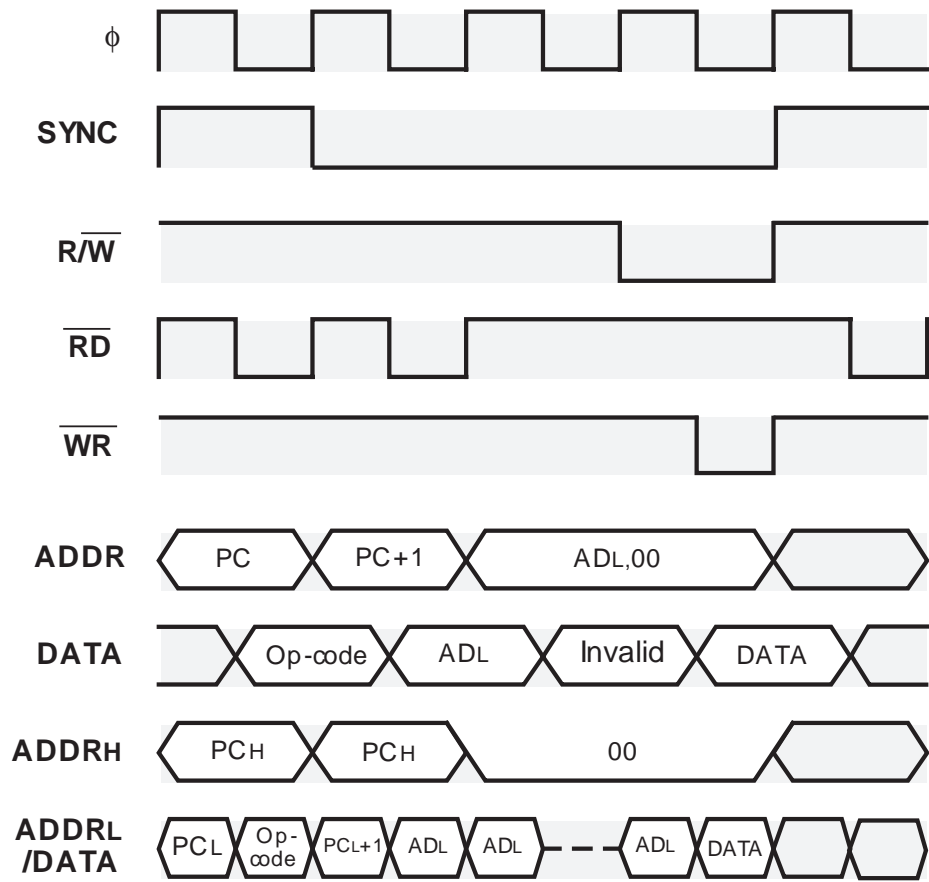
Timing :



# ZERO PAGE

---

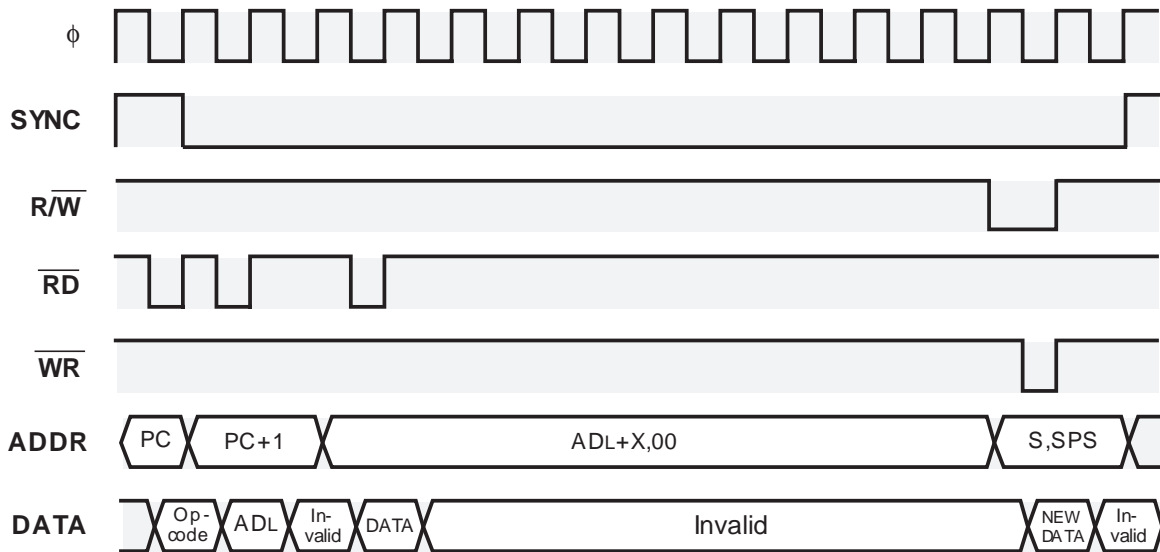
Instructions :  $\Delta$ STA $\Delta$ \$zz  
                    $\Delta$ STX $\Delta$ \$zz  
                    $\Delta$ STY $\Delta$ \$zz  
 Byte length : 2  
 Cycle number : 4  
 Timing :



# Zero Page X

Instruction :  $\Delta$ MUL $\Delta$ \$zz,X (Note)  
 Byte length : 2  
 Cycle number : 15

Timing :



SPS: A selected page by stack page selection bit of the CPU mode register.

Note: This instruction cannot be used for any products.



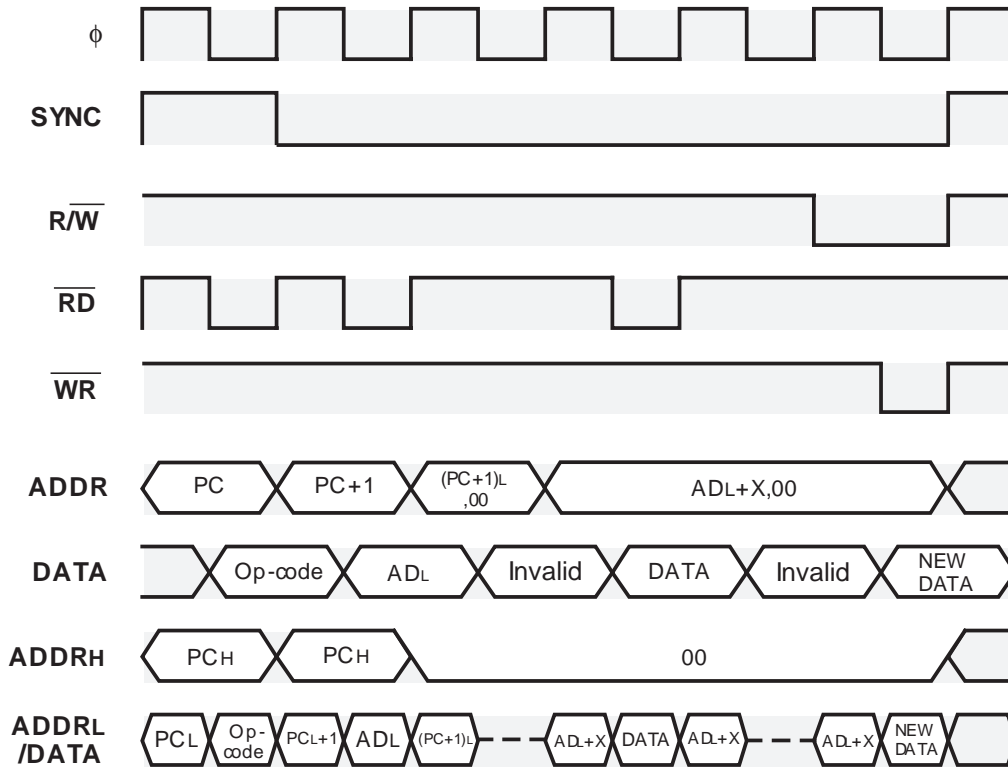


## Zero Page X

Instructions :  $\Delta$ ASL  $\Delta$ \$zz,X  
 $\Delta$ DEC  $\Delta$ \$zz,X  
 $\Delta$ INC  $\Delta$ \$zz,X  
 $\Delta$ LSR  $\Delta$ \$zz,X  
 $\Delta$ ROL  $\Delta$ \$zz,X  
 $\Delta$ ROR  $\Delta$ \$zz,X

Byte length : 2  
 Cycle number : 6

Timing :



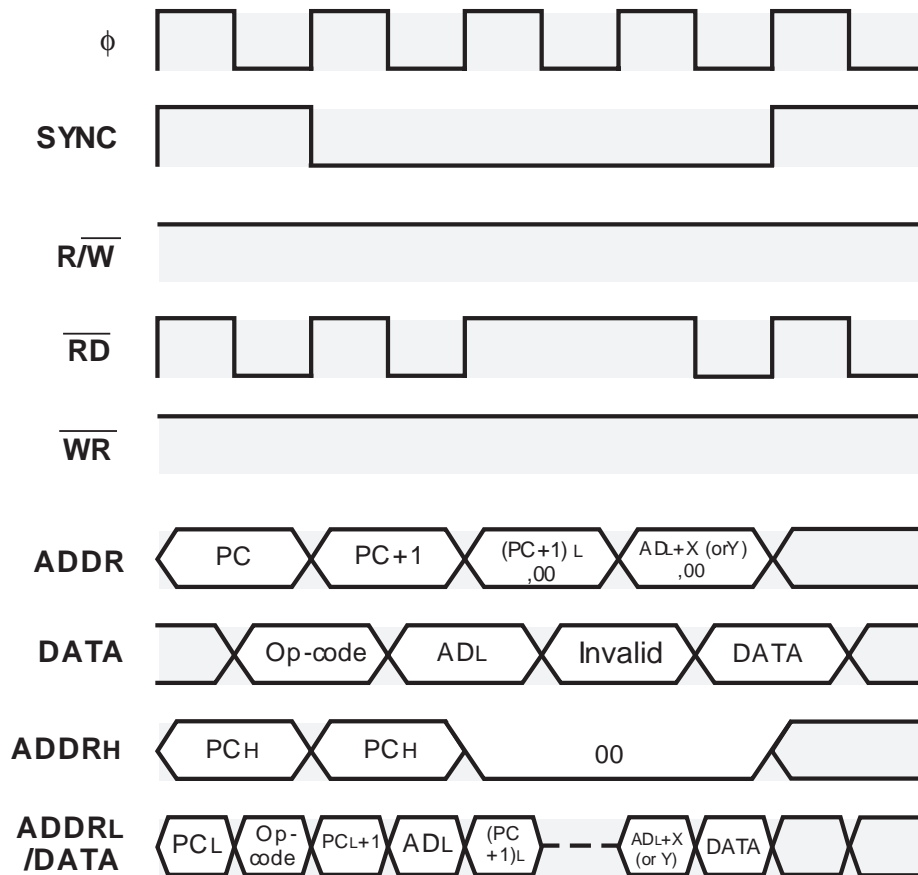
[T=0]

## ZERO PAGE X, ZERO PAGE Y

Instructions :  $\Delta$ ADC  $\Delta$ \$zz,X (T=0)  
 $\Delta$ AND  $\Delta$ \$zz,X (T=0)  
 $\Delta$ CMP  $\Delta$ \$zz,X (T=0)  
 $\Delta$ EOR  $\Delta$ \$zz,X (T=0)  
 $\Delta$ LDA  $\Delta$ \$zz,X (T=0)  
 $\Delta$ LDX  $\Delta$ \$zz,Y  
 $\Delta$ LDY  $\Delta$ \$zz,X  
 $\Delta$ ORA  $\Delta$ \$zz,X (T=0)  
 $\Delta$ SBC  $\Delta$ \$zz,X (T=0)

Byte length : 2  
 Cycle number : 4

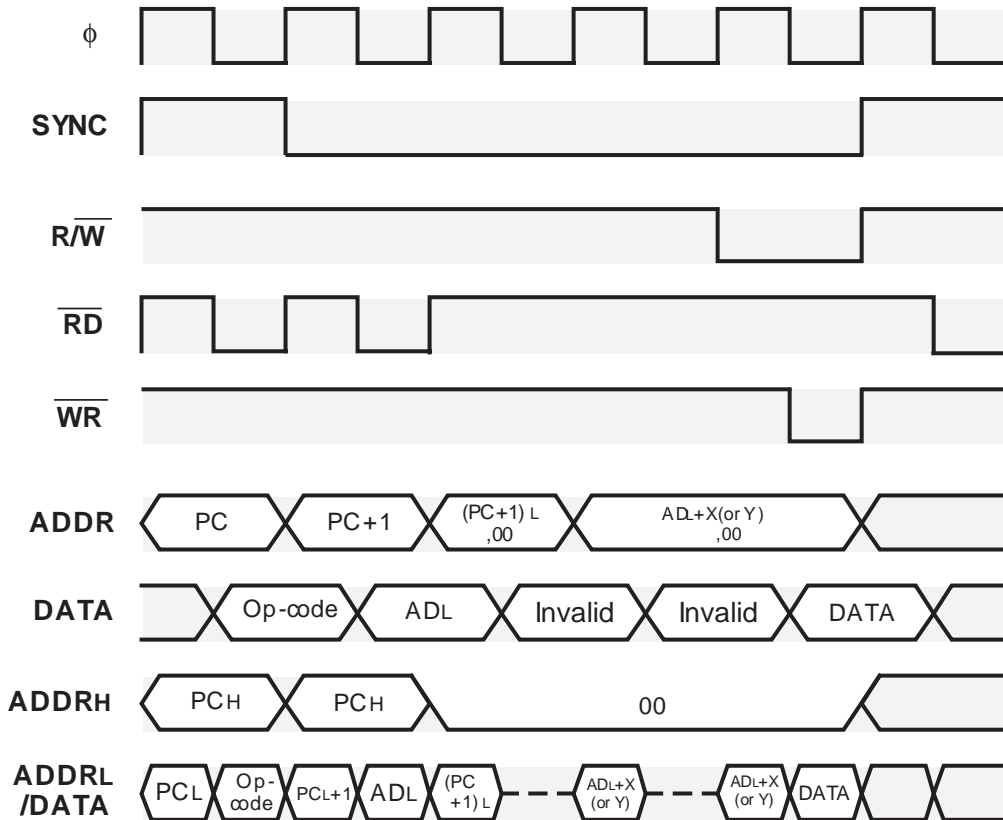
Timing :



# ZERO PAGE X, ZERO PAGE Y

---

Instructions :  $\Delta$ STA $\Delta$ \$zz,X  
                    $\Delta$ STX $\Delta$ \$zz,Y  
                    $\Delta$ STY $\Delta$ \$zz,X  
 Byte length : 2  
 Cycle number : 5  
 Timing :



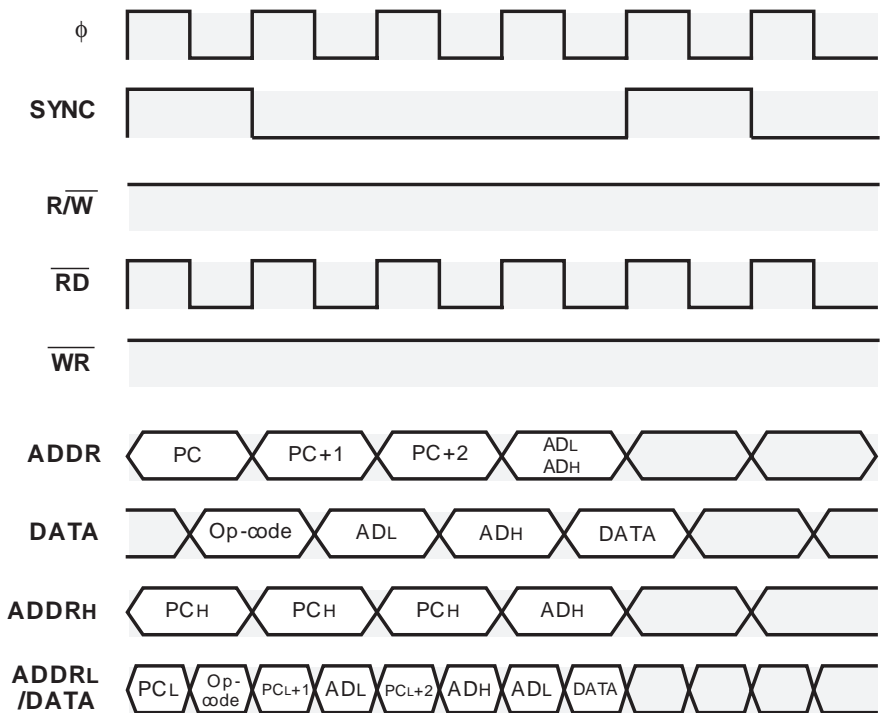
[T=0]

# ABSOLUTE

Instructions :  $\Delta$ ADC  $\Delta$ \$hhl (T=0)  
 $\Delta$ AND  $\Delta$ \$hhl (T=0)  
 $\Delta$ BIT  $\Delta$ \$hhl  
 $\Delta$ CMP  $\Delta$ \$hhl (T=0)  
 $\Delta$ CPX  $\Delta$ \$hhl  
 $\Delta$ CPY  $\Delta$ \$hhl  
 $\Delta$ EOR  $\Delta$ \$hhl (T=0)  
 $\Delta$ LDA  $\Delta$ \$hhl (T=0)  
 $\Delta$ LDX  $\Delta$ \$hhl  
 $\Delta$ LDY  $\Delta$ \$hhl  
 $\Delta$ ORA  $\Delta$ \$hhl (T=0)  
 $\Delta$ SBC  $\Delta$ \$hhl (T=0)

Byte length : 3  
 Cycle number : 4

Timing :

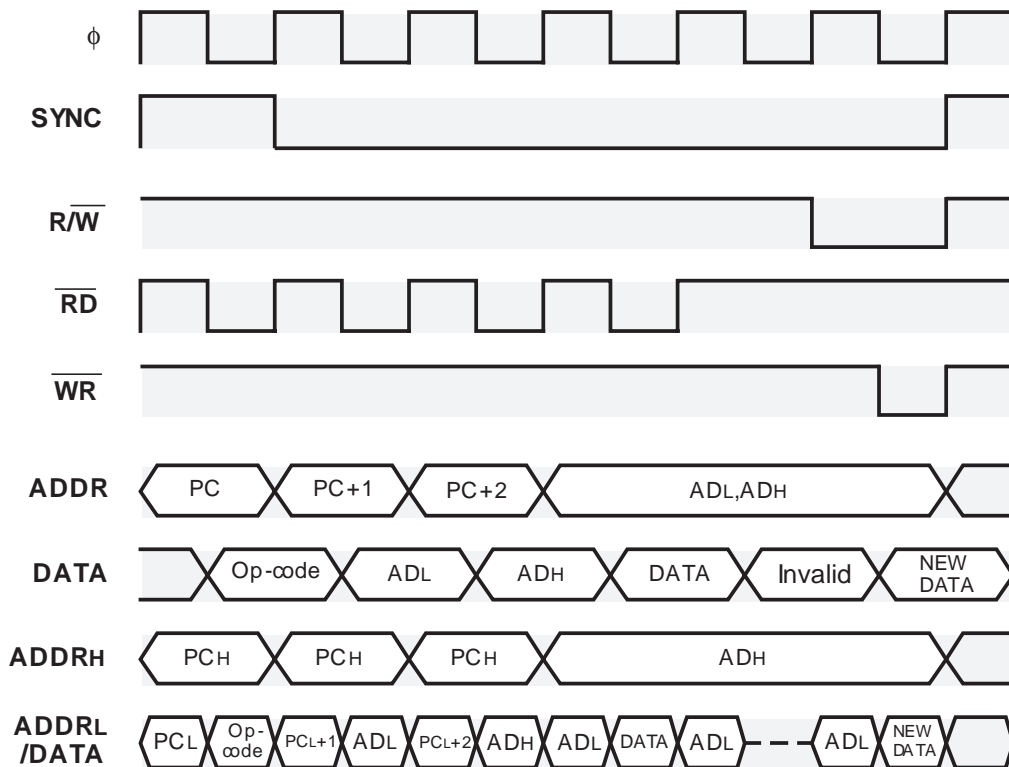


# ABSOLUTE

Instructions :  $\Delta$ ASL  $\Delta$ \$hhll  
                    $\Delta$ DEC  $\Delta$ \$hhll  
                    $\Delta$ INC  $\Delta$ \$hhll  
                    $\Delta$ LSR  $\Delta$ \$hhll  
                    $\Delta$ ROL  $\Delta$ \$hhll  
                    $\Delta$ ROR  $\Delta$ \$hhll

Byte length : 3  
 Cycle number : 6

Timing :

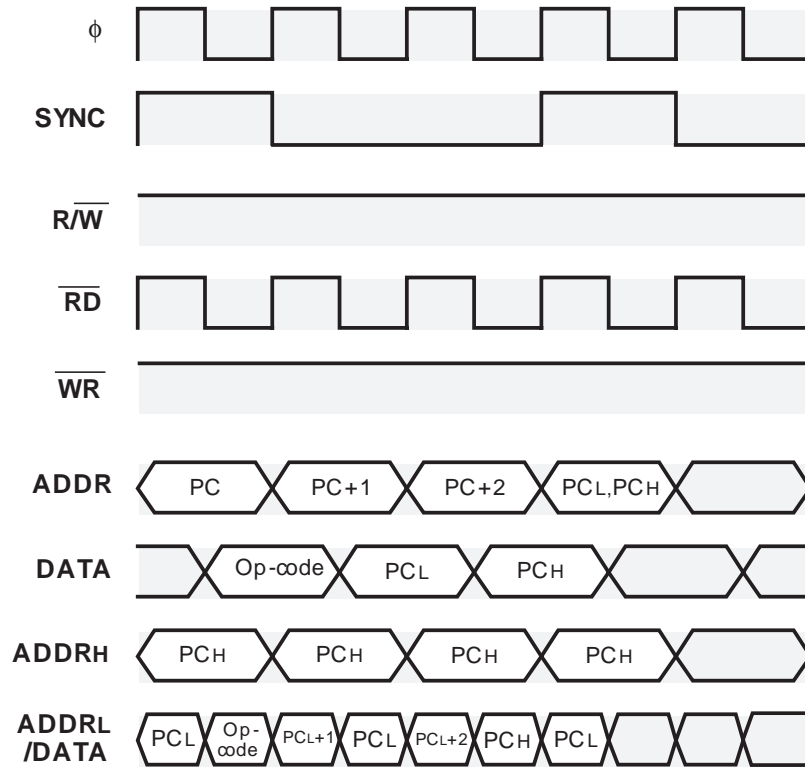


# ABSOLUTE

---

Instruction :  $\Delta$ JMP $\Delta$ \$hhll  
 Byte length : 3  
 Cycle number : 3

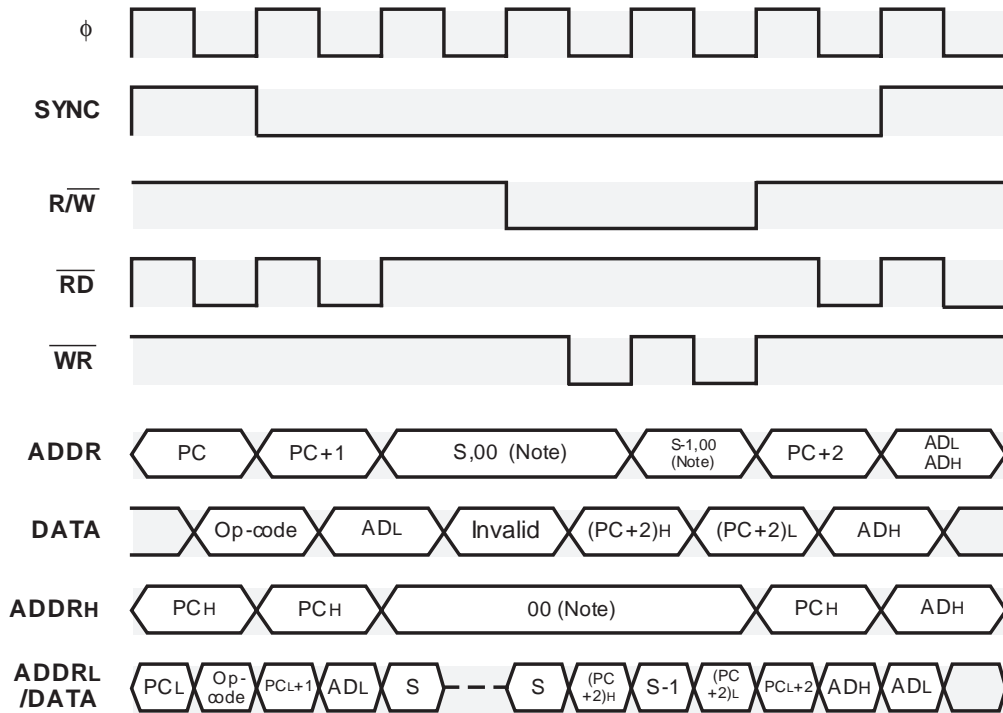
Timing :



# ABSOLUTE

Instruction :  $\Delta$ JSR $\Delta$ \$hhll  
 Byte length : 3  
 Cycle number : 6

Timing :



Note: Some products are "01" or content of SPS flag.

# ABSOLUTE

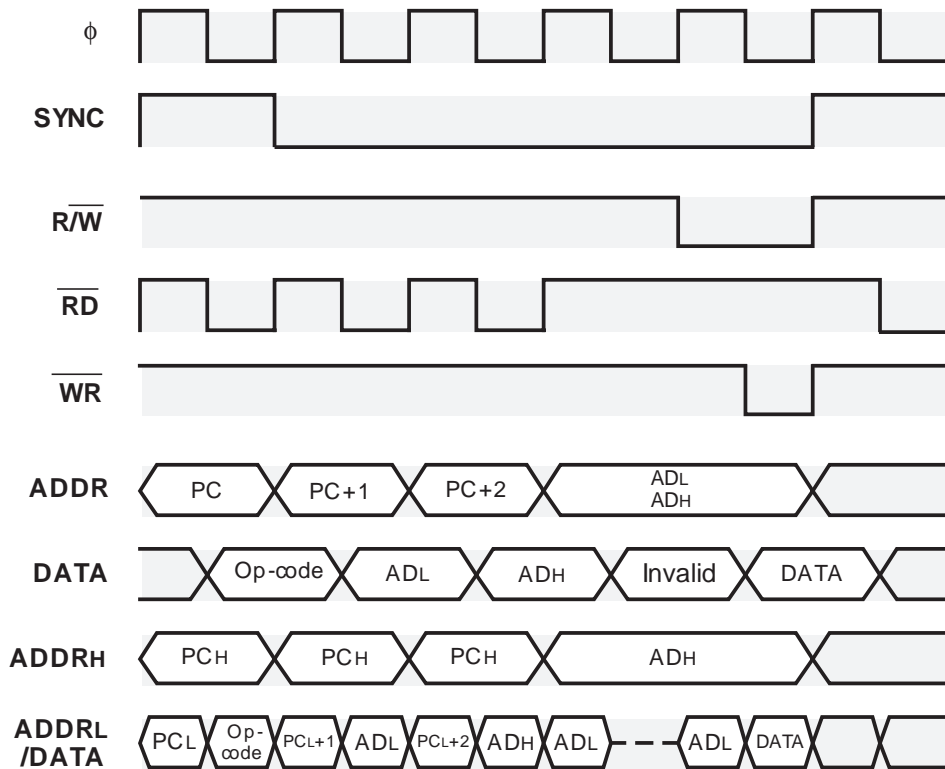
---

Instructions :  $\Delta$ STA $\Delta$ \$hhl  
 $\Delta$ STX $\Delta$ \$hhl  
 $\Delta$ STY $\Delta$ \$hhl

Byte length : 3

Cycle number : 5

Timing :





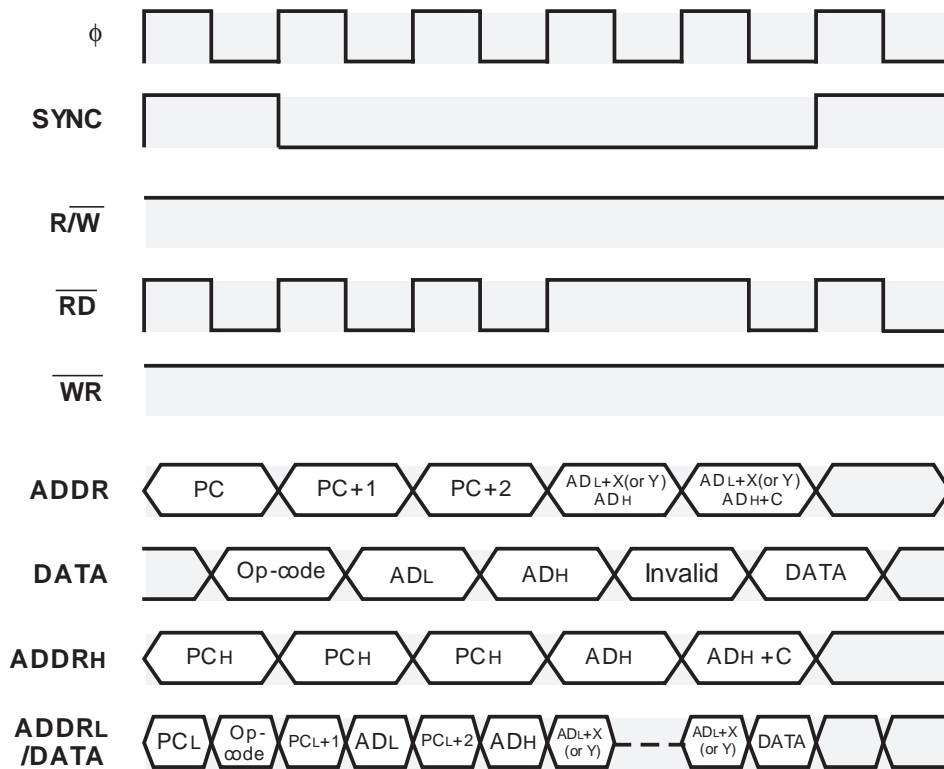
[T=0]

## ABSOLUTE X, ABSOLUTE Y

Instructions :  $\Delta$ ADC  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ AND  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ CMP  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ EOR  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ LDA  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ LDX  $\Delta$ \$hhl,Y  
 $\Delta$ LDY  $\Delta$ \$hhl,X  
 $\Delta$ ORA  $\Delta$ \$hhl,X or Y (T=0)  
 $\Delta$ SBC  $\Delta$ \$hhl,X or Y (T=0)

Byte length : 3  
 Cycle number : 5

Timing :



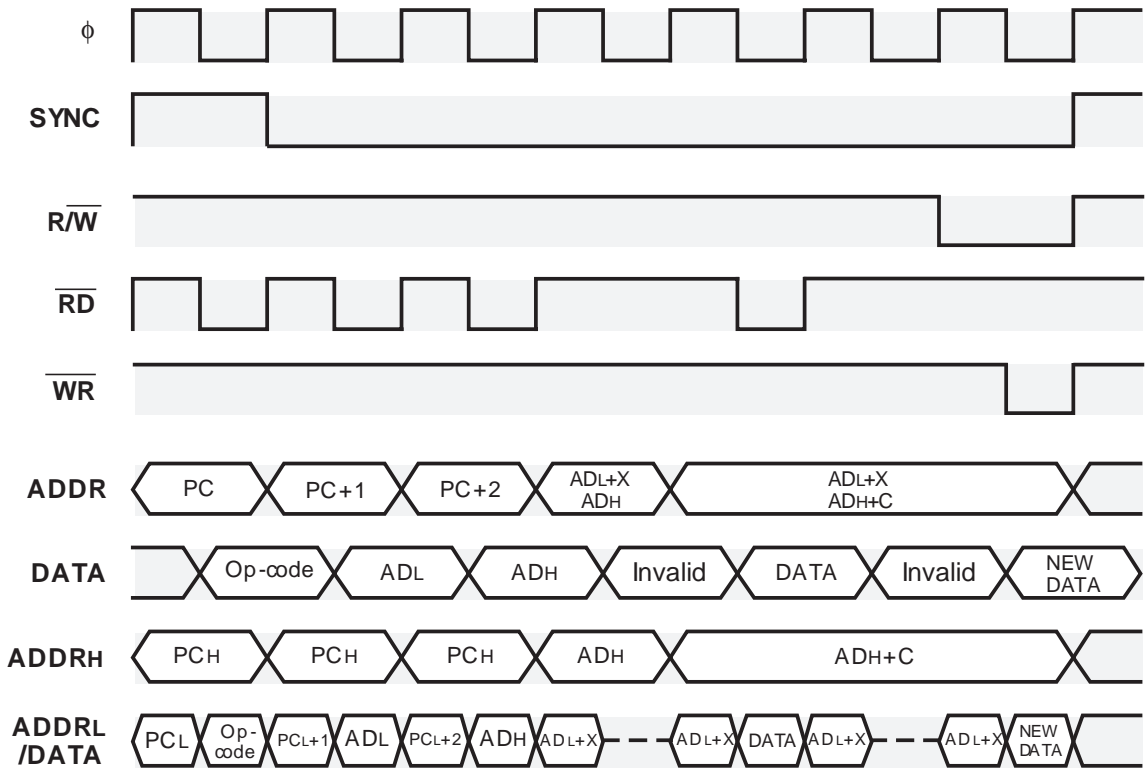
C : Carry of ADL+X or Y

# ABSOLUTE X

Instructions :  $\Delta$ ASL  $\Delta$ \$hhll,X  
                $\Delta$ DEC  $\Delta$ \$hhll,X  
                $\Delta$ INC  $\Delta$ \$hhll,X  
                $\Delta$ LSR  $\Delta$ \$hhll,X  
                $\Delta$ ROL  $\Delta$ \$hhll,X  
                $\Delta$ ROR  $\Delta$ \$hhll,X

Byte length : 3  
 Cycle number : 7

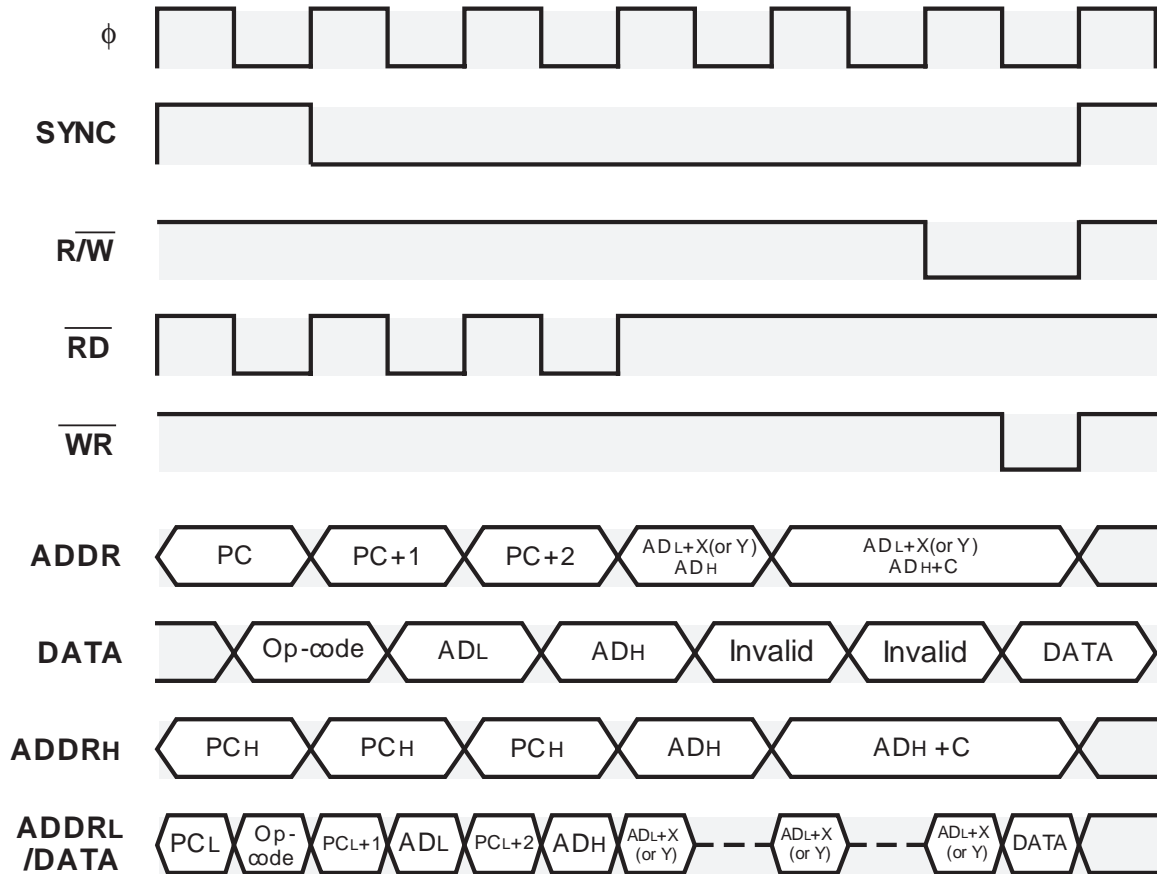
Timing :



C : Carry of ADL+X

# ABSOLUTE X, ABSOLUTE Y

Instruction :  $\Delta$ STA $\Delta$ \$hhll,X or Y  
 Byte length : 3  
 Cycle number : 6  
 Timing :

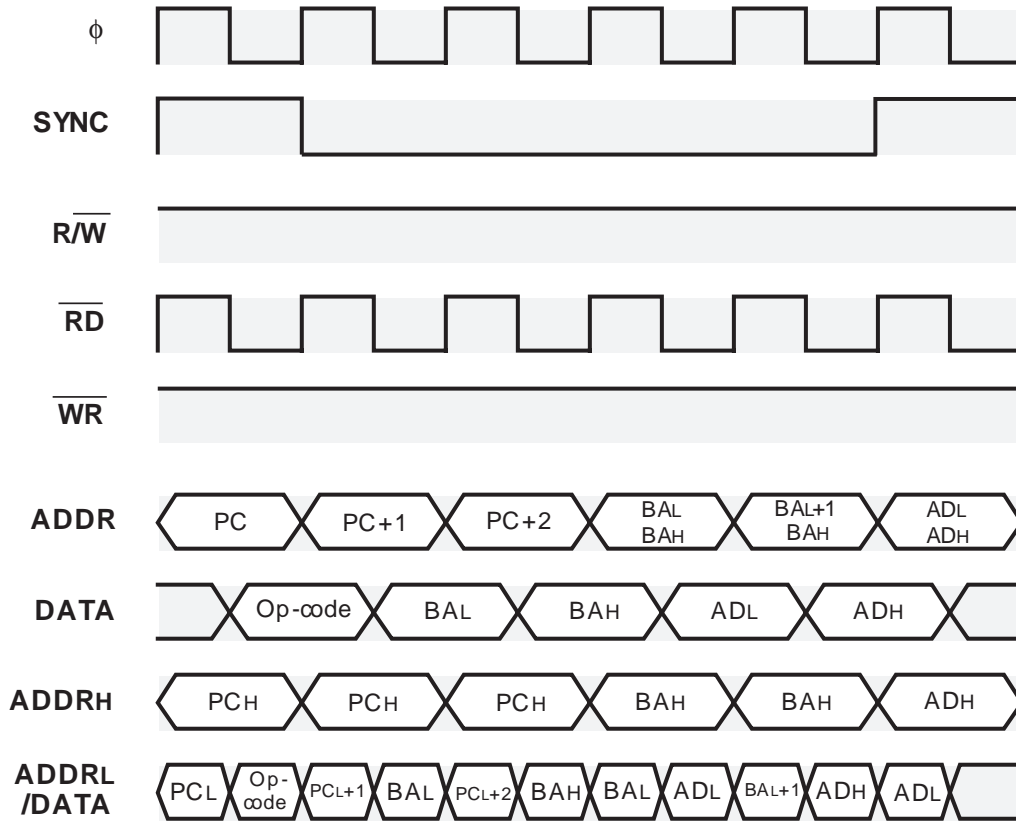


C : Carry of ADL+X or Y

# INDIRECT

Instruction :  $\Delta$ JMP $\Delta$ (\$hhll)  
 Byte length : 3  
 Cycle number : 5

Timing :



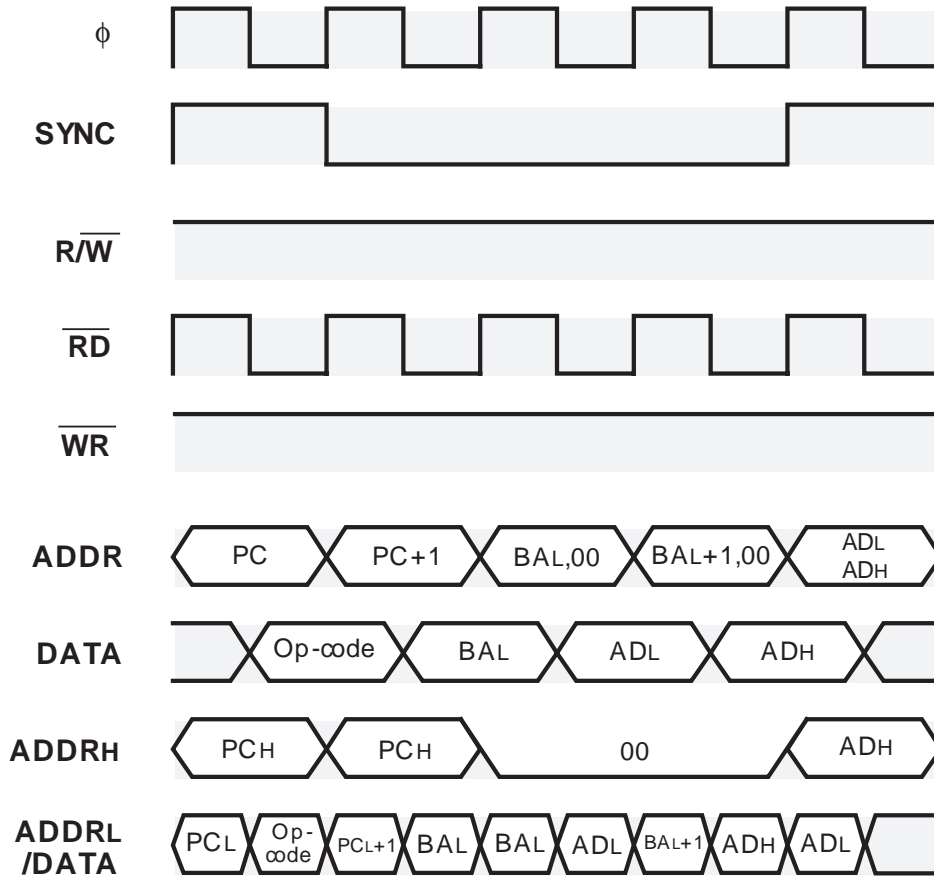
BA : Basic address

# ZERO PAGE INDIRECT

---

Instruction :  $\Delta$ JMP $\Delta$ (\$zz)  
 Byte length : 2  
 Cycle number : 4

Timing :

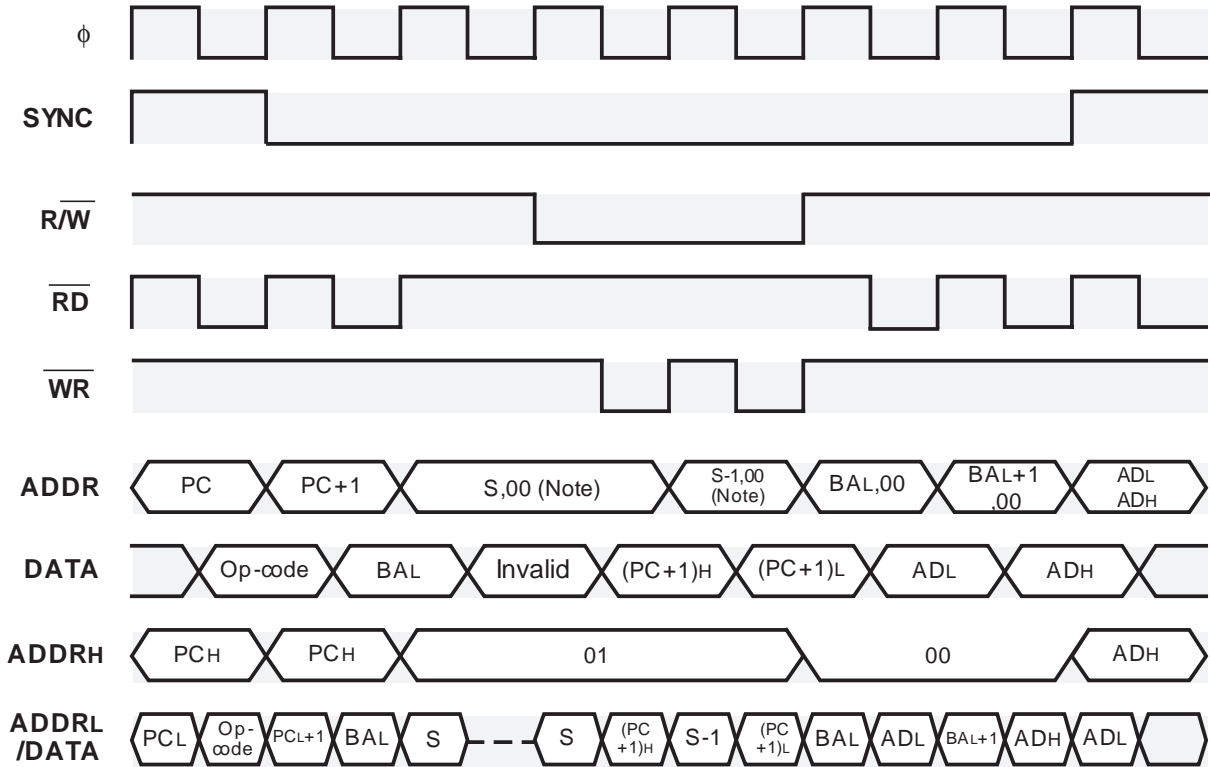


BA : Basic address

# ZERO PAGE INDIRECT

Instruction :  $\Delta$ JSR $\Delta$ (\$zz)  
 Byte length : 2  
 Cycle number : 7

Timing :



BA : Basic address

Note: Some kind types are "01" or content of SPS flag.

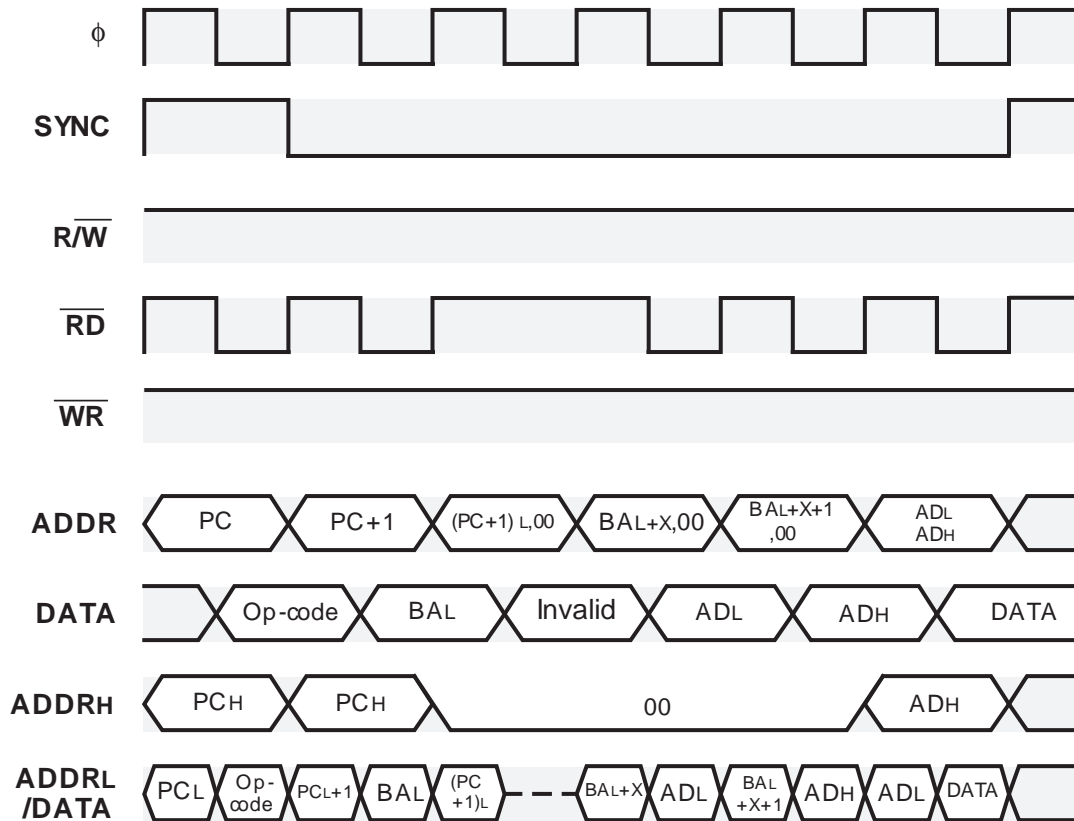
[T=0]

## INDIRECT X

Instructions :  $\Delta$ ADC  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ AND  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ CMP  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ EOR  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ LDA  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ ORA  $\Delta$ (\$zz,X) (T=0)  
 $\Delta$ SBC  $\Delta$ (\$zz,X) (T=0)

Byte length : 2  
 Cycle number : 6

Timing :

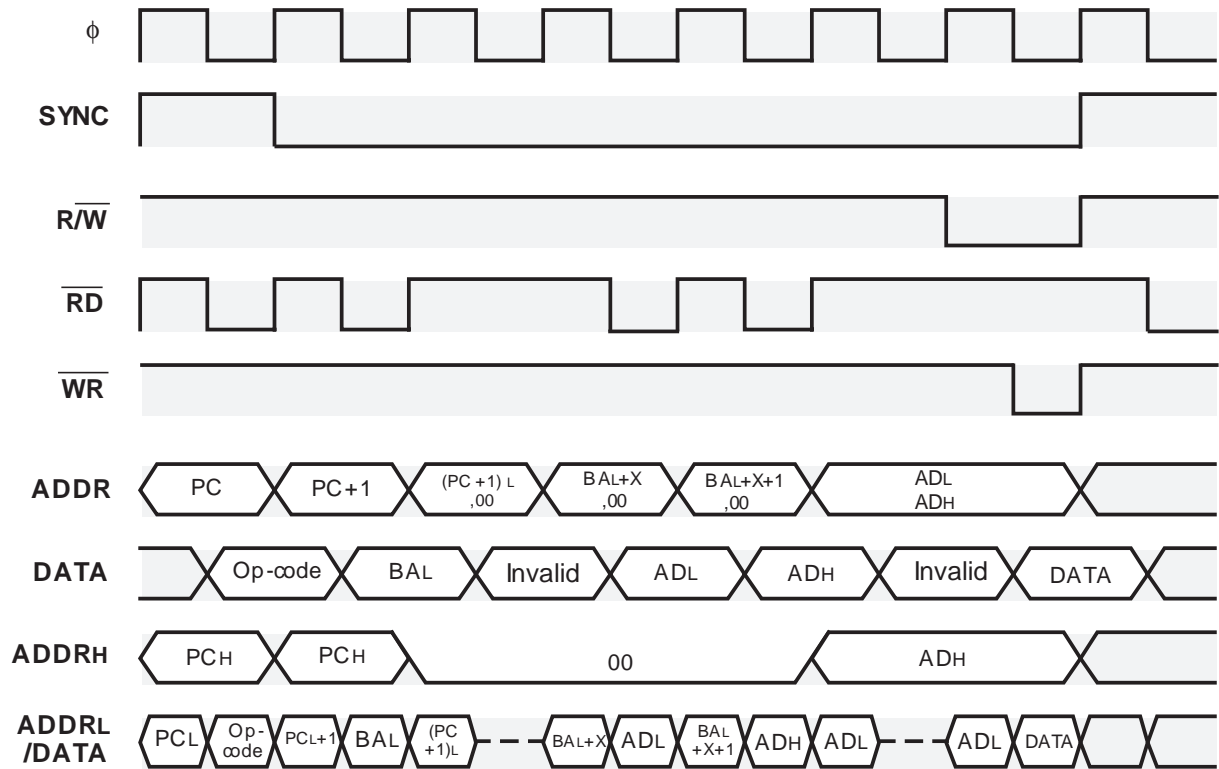


BA : Basic address

# INDIRECT X

Instruction :  $\Delta STA(\$zz,X)$   
 Byte length : 2  
 Cycle number : 7

Timing :



BA : Basic address



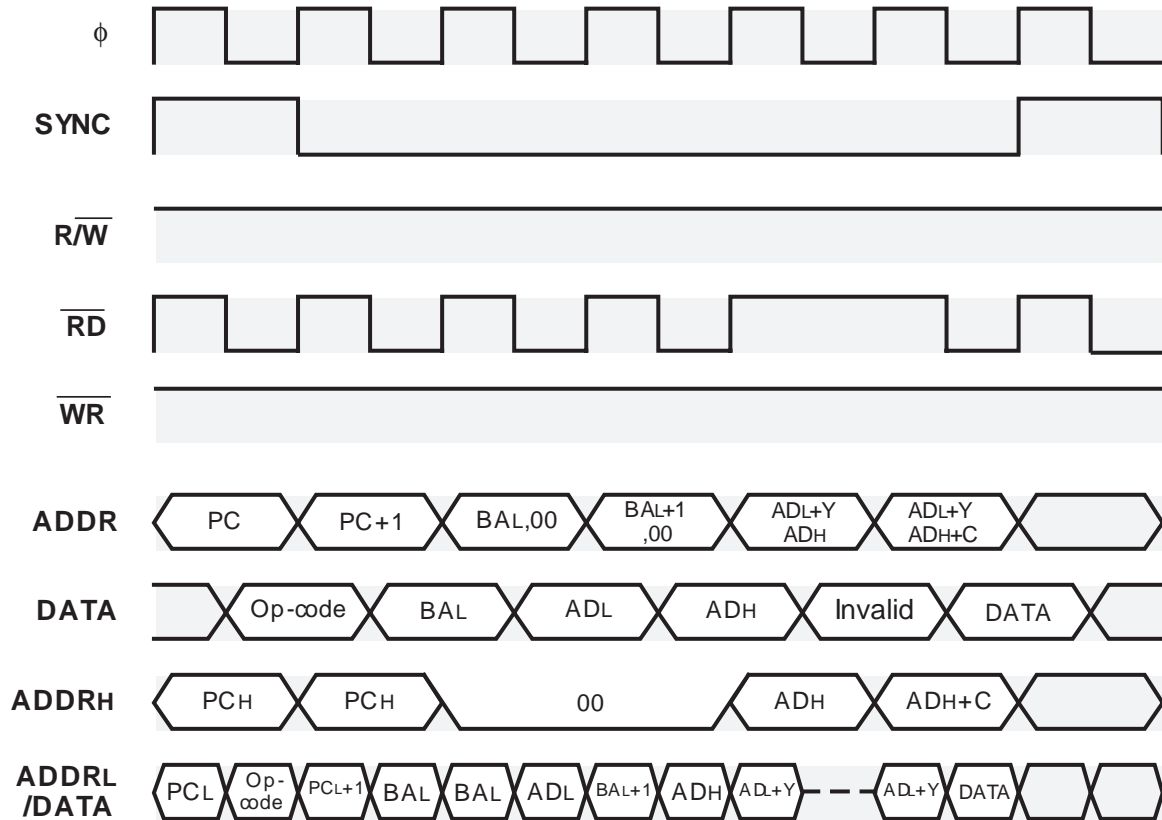
[T=0]

## INDIRECT Y

Instructions :  $\Delta$ ADC  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ AND  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ CMP  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ EOR  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ LDA  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ ORA  $\Delta$ (\$zz),Y (T=0)  
 $\Delta$ SBC  $\Delta$ (\$zz),Y (T=0)

Byte length : 2  
 Cycle number : 6

Timing :



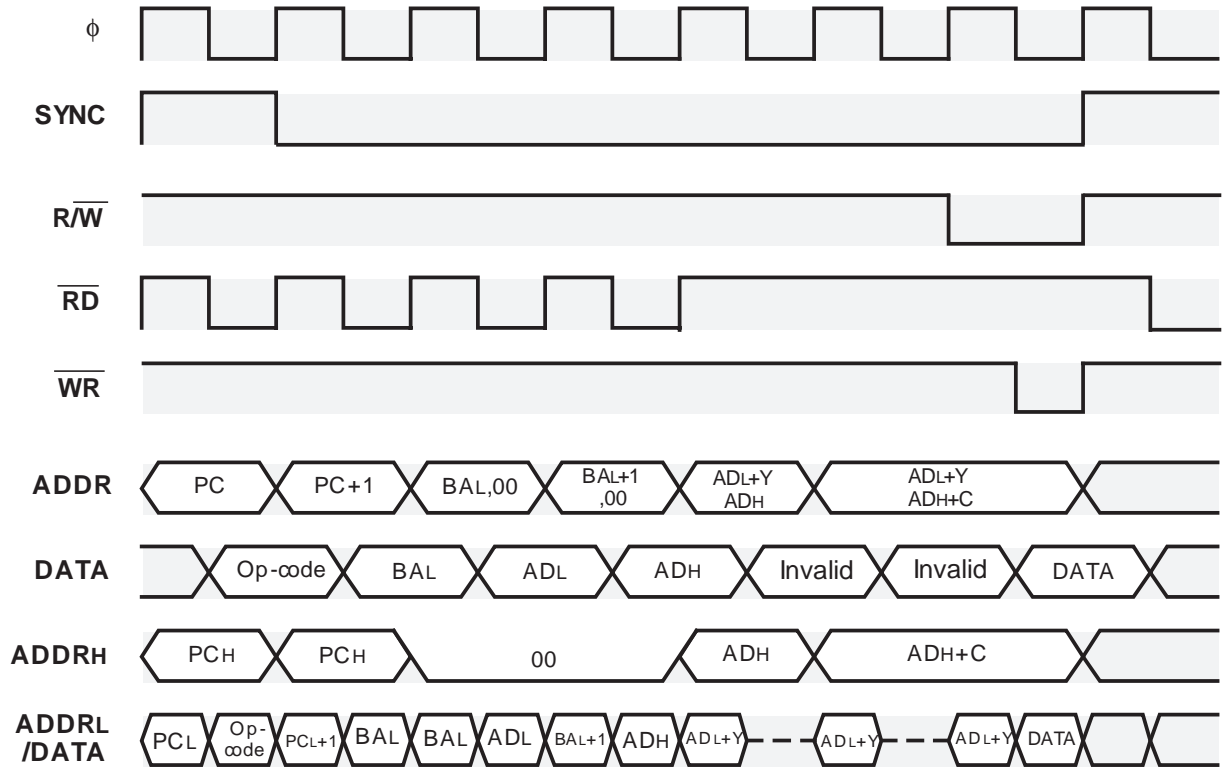
BA : Basic address

C : Carry of ADL+Y

# INDIRECT Y

Instruction :  $\Delta$ STAA( $\$zz$ ),Y  
 Byte length : 2  
 Cycle number : 7

Timing :



BA : Basic address

C : Carry of ADL+Y

# RELATIVE

---

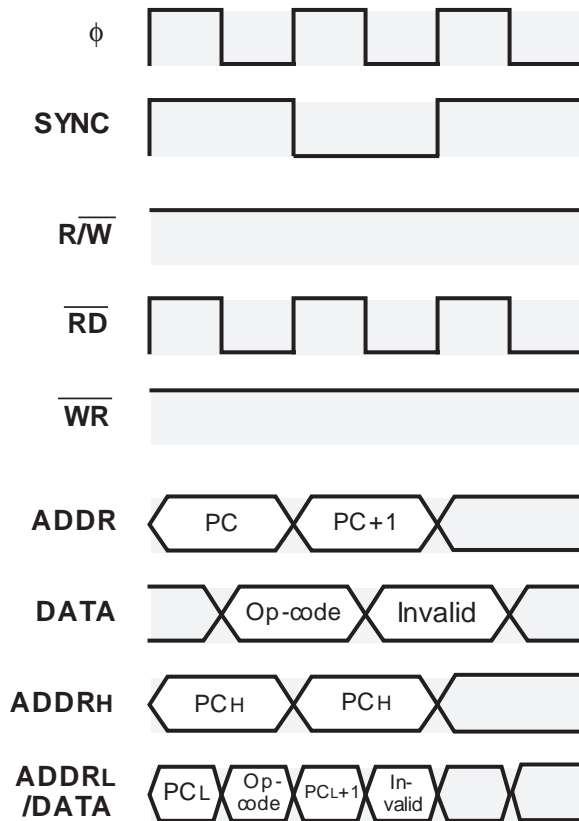
Instructions :  $\Delta$ BCC  $\Delta$ \$hhll  
                    $\Delta$ BCS  $\Delta$ \$hhll  
                    $\Delta$ BEQ  $\Delta$ \$hhll  
                    $\Delta$ BMI  $\Delta$ \$hhll  
                    $\Delta$ BNE  $\Delta$ \$hhll  
                    $\Delta$ BPL  $\Delta$ \$hhll  
                    $\Delta$ BVC  $\Delta$ \$hhll  
                    $\Delta$ BVS  $\Delta$ \$hhll

Byte length : 2

(1)With no branch

Cycle number : 2

Timing :



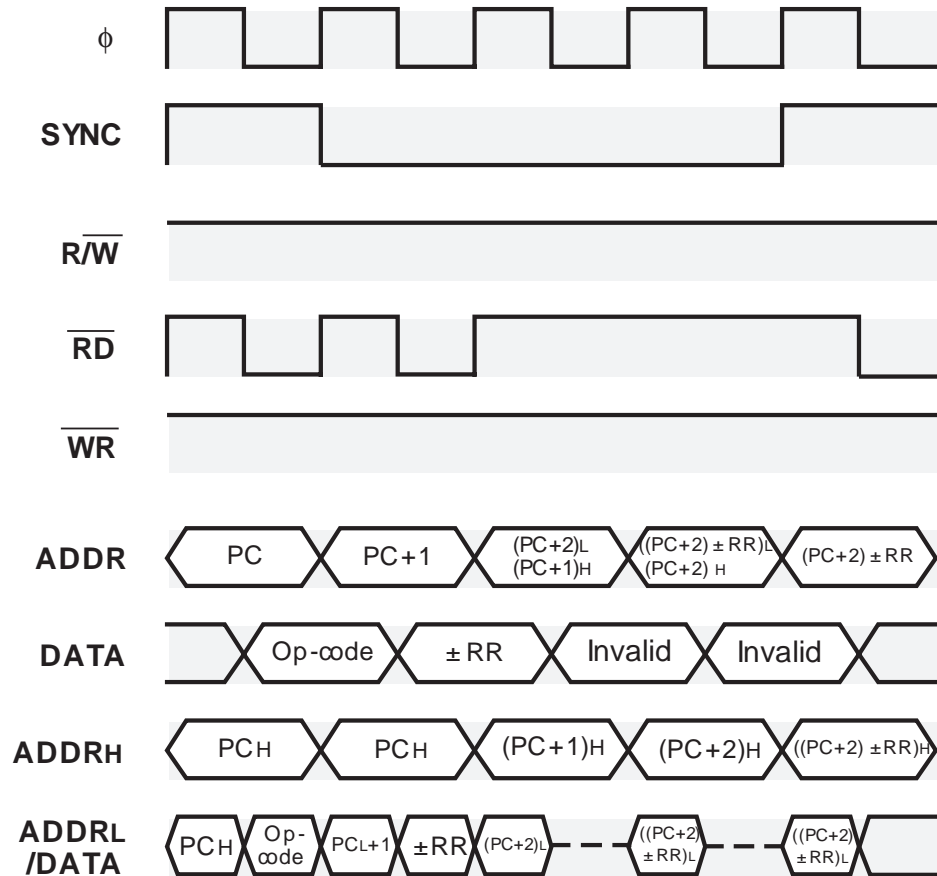
# RELATIVE

Instructions :  $\Delta$ BCC  $\Delta$ \$hhl  
 $\Delta$ BCS  $\Delta$ \$hhl  
 $\Delta$ BEQ  $\Delta$ \$hhl  
 $\Delta$ BMI  $\Delta$ \$hhl  
 $\Delta$ BNE  $\Delta$ \$hhl  
 $\Delta$ BPL  $\Delta$ \$hhl  
 $\Delta$ BVC  $\Delta$ \$hhl  
 $\Delta$ BVS  $\Delta$ \$hhl

Byte length : 2

(2)With branch  
 Cycle number : 4

Timing :

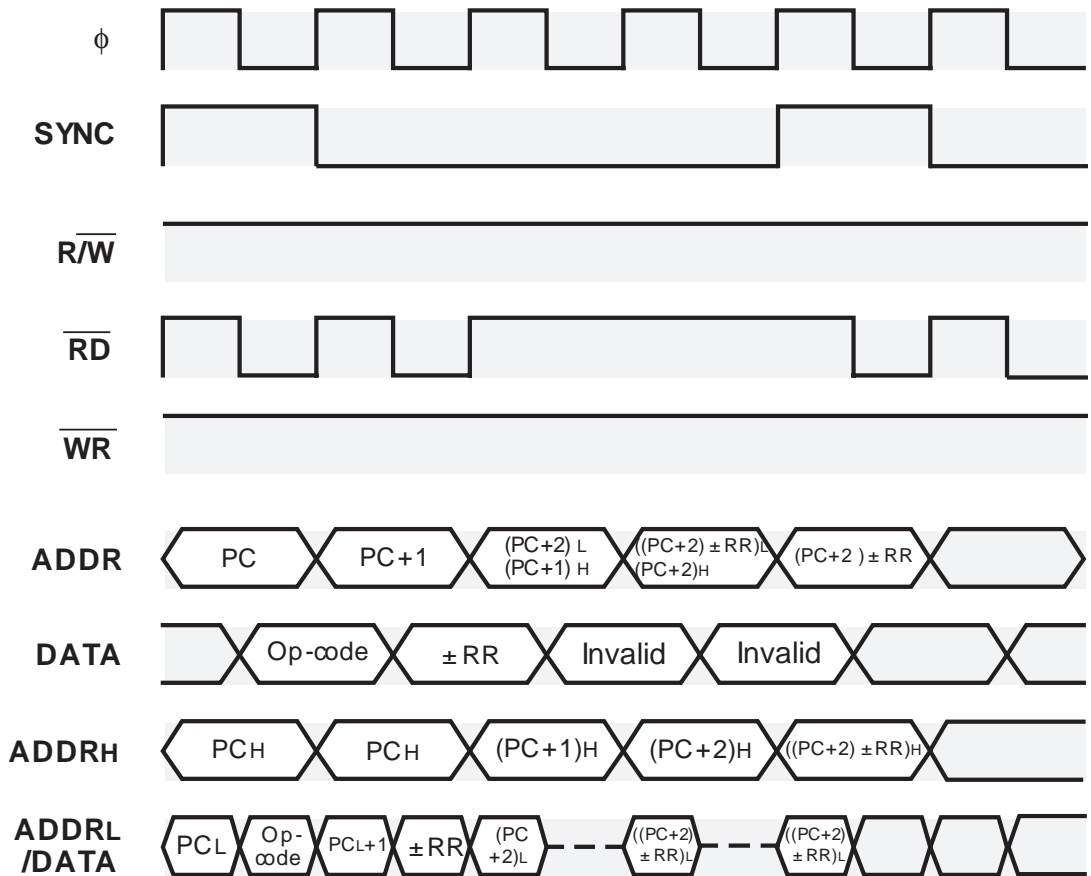


RR : Offset value

# RELATIVE

Instruction :  $\Delta$ BRA $\Delta$ \$hhll  
 Byte length : 2  
 Cycle number : 4

Timing :



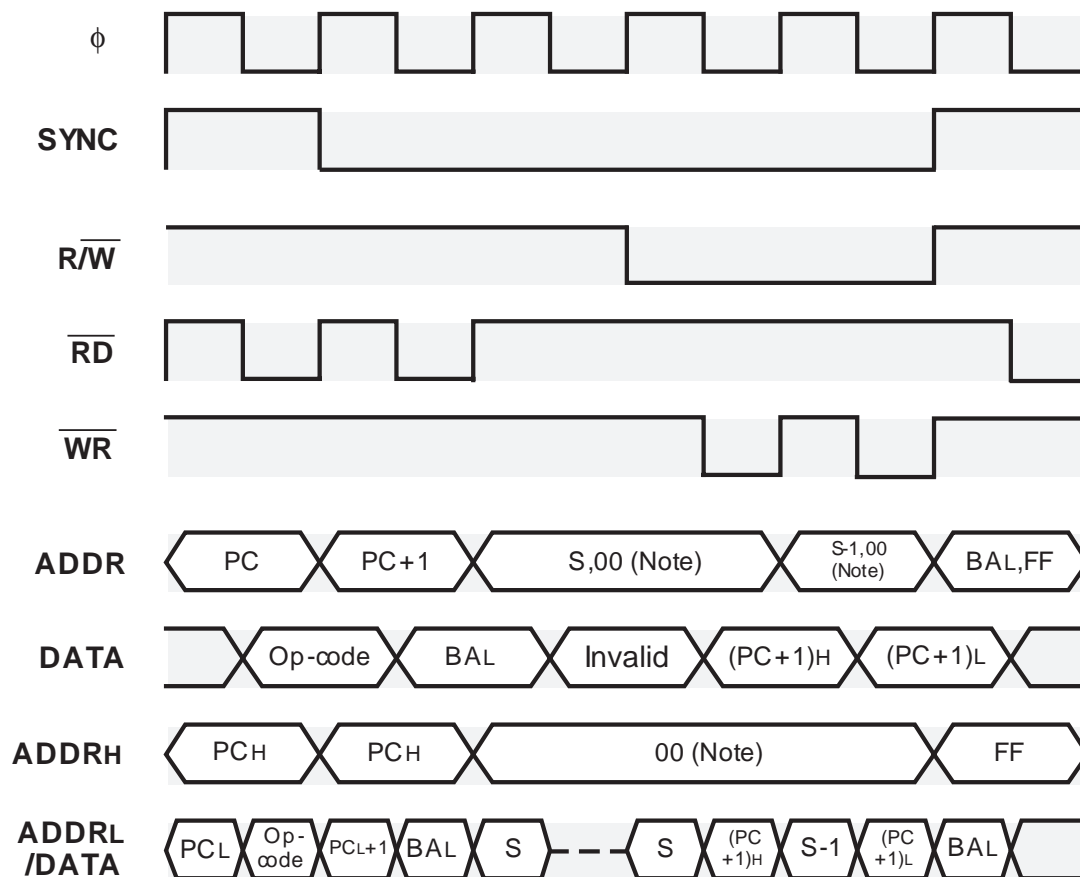
RR : Offset value

## SPECIAL PAGE

---

Instruction :  $\Delta$ JSR $\Delta$ \$hhll  
 Byte length : 2  
 Cycle number : 5

Timing :



BA : Basic address

Note : Some products are "01" or content of SPS flag.

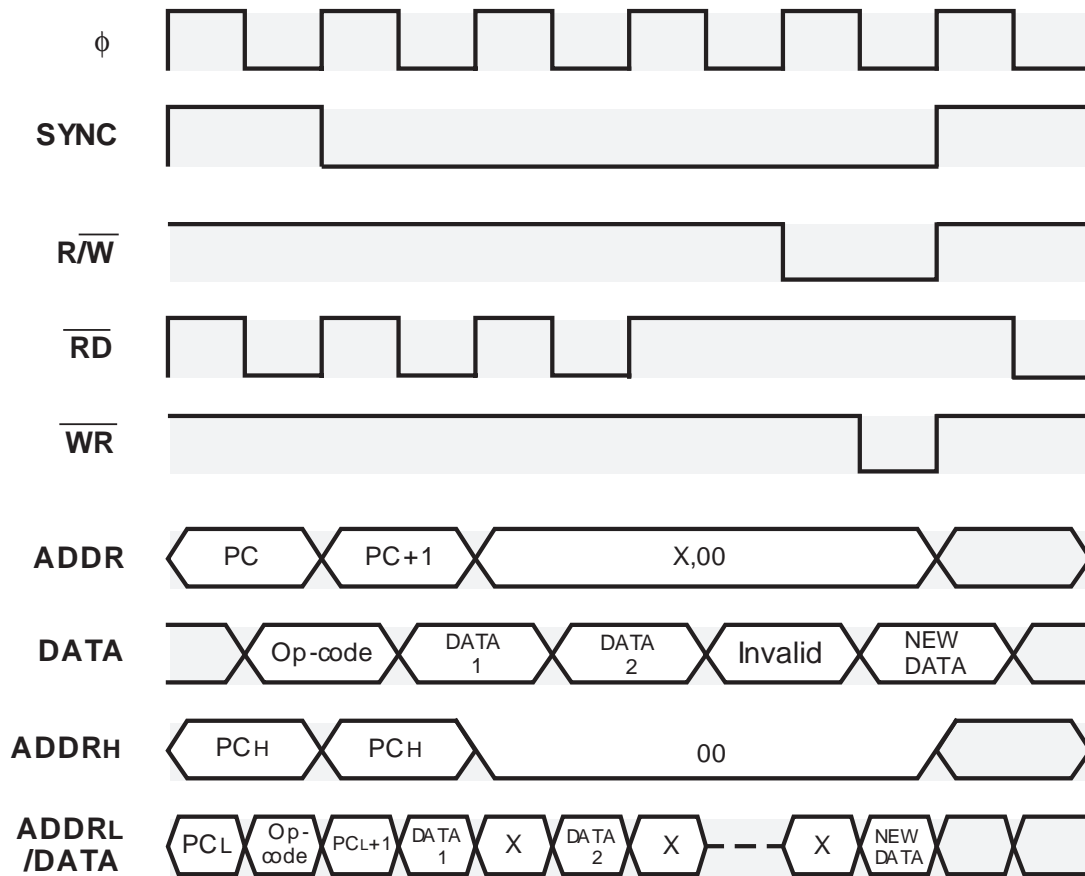
[T=1]

## IMMEDIATE

Instructions :  $\Delta$ ADC $\Delta$ #\$nn (T=1)  
 $\Delta$ AND $\Delta$ #\$nn (T=1)  
 $\Delta$ EOR $\Delta$ #\$nn (T=1)  
 $\Delta$ ORA $\Delta$ #\$nn (T=1)  
 $\Delta$ SBC $\Delta$ #\$nn (T=1)

Byte length : 2  
 Cycle number : 5

Timing :

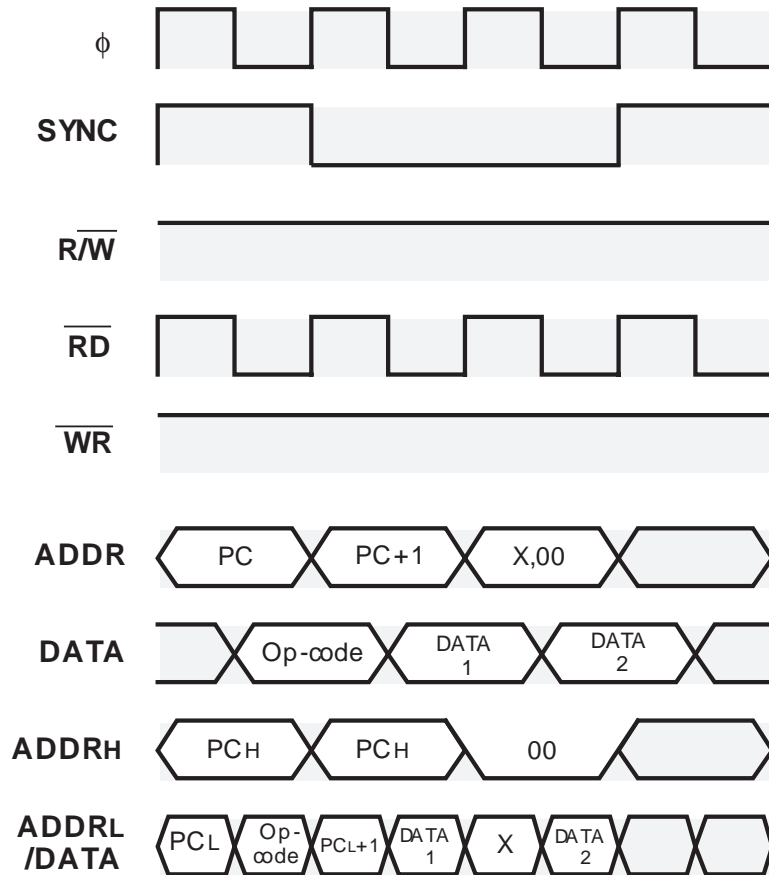


[T=1]

## IMMEDIATE

Instruction :  $\Delta$ CMP $\Delta$ # $\$$ nn (T=1)  
 Byte length : 2  
 Cycle number : 3

Timing :



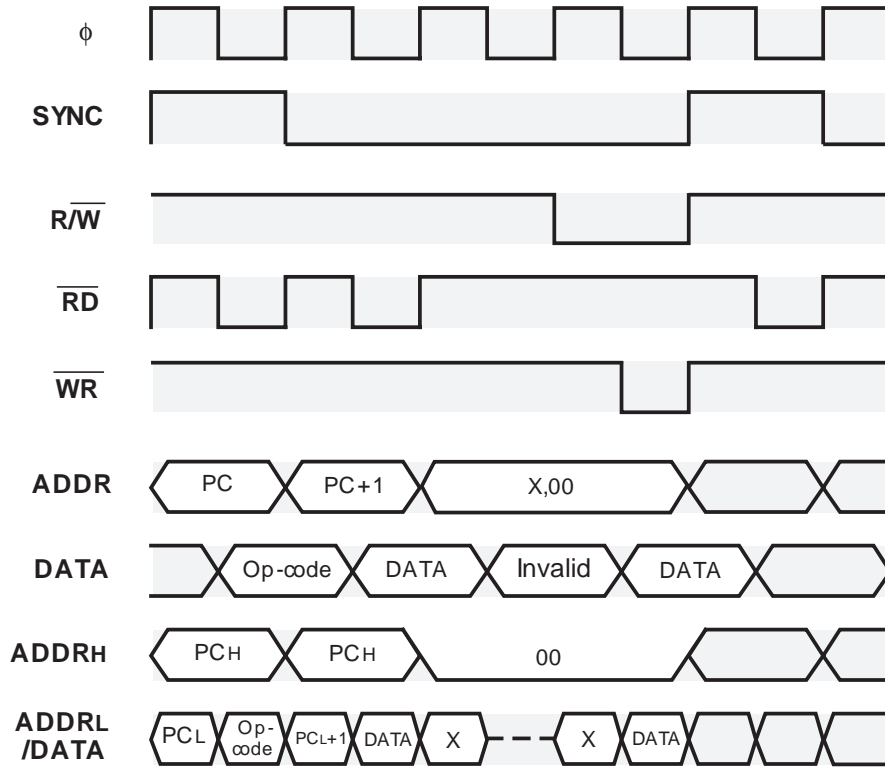


[T=1]

# IMMEDIATE

Instruction :  $\Delta$ LDA $\Delta$ # $\$$ nn (T=1)  
 Byte length : 2  
 Cycle number : 4

Timing :



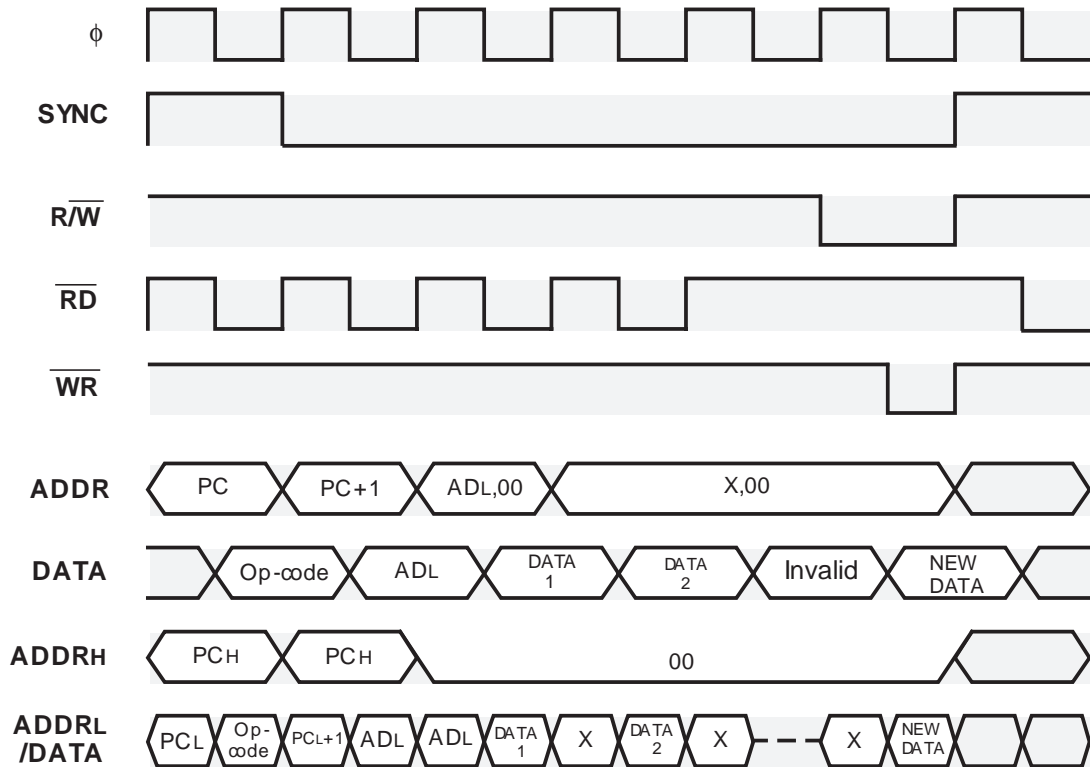
[T=1]

## ZERO PAGE

Instructions :  $\Delta$ ADC $\Delta$ \$zz (T=1)  
 $\Delta$ AND $\Delta$ \$zz (T=1)  
 $\Delta$ EOR $\Delta$ \$zz (T=1)  
 $\Delta$ ORA $\Delta$ \$zz (T=1)  
 $\Delta$ SBC $\Delta$ \$zz (T=1)

Byte length : 2  
 Cycle number : 6

Timing :

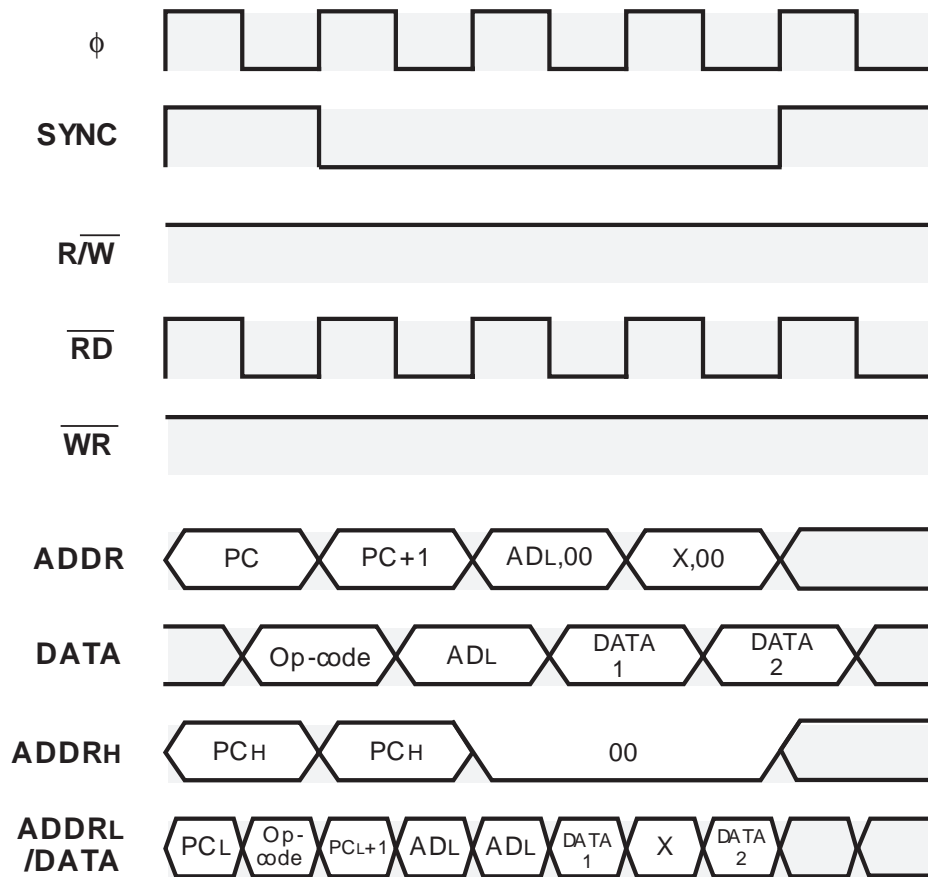


[T=1]

## ZERO PAGE

Instruction :  $\Delta$ CMP $\Delta$ \$zz (T=1)  
 Byte length : 2  
 Cycle number : 4

Timing :

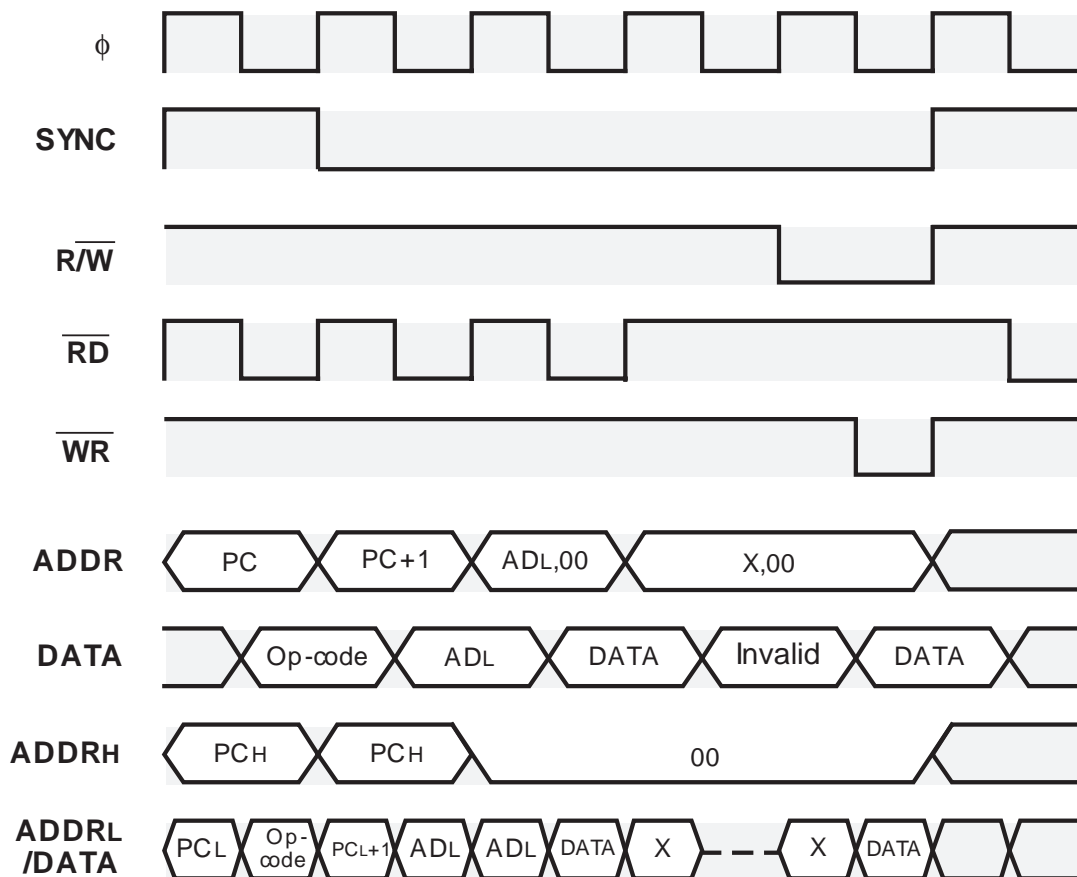


[T=1]

## ZERO PAGE

Instruction :  $\Delta$ LDA $\Delta$ \$zz (T=1)  
 Byte length : 2  
 Cycle number : 5

Timing :



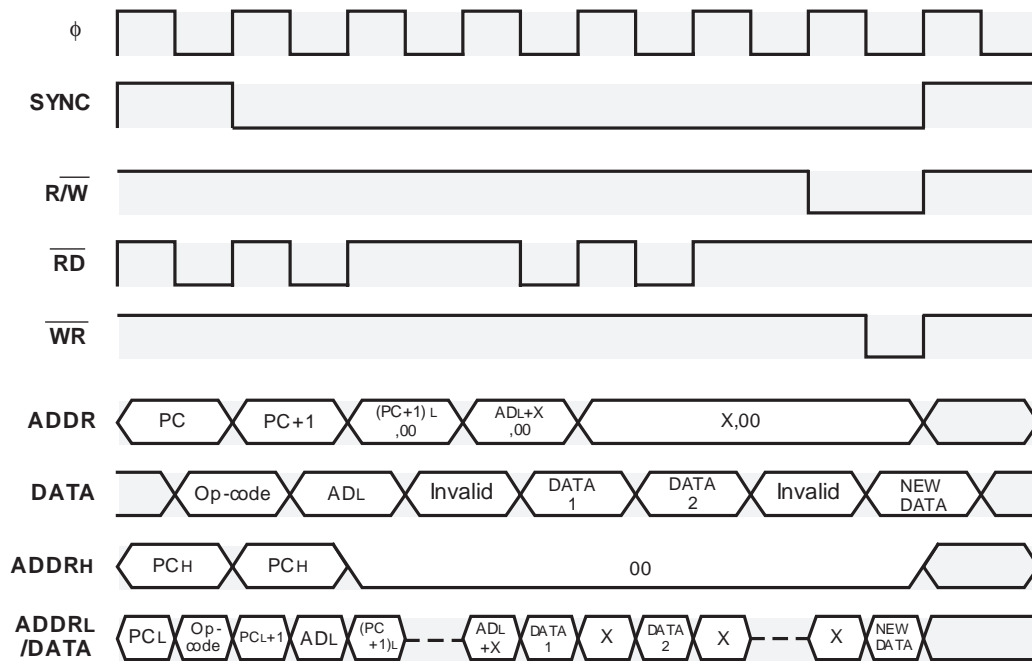
[T=1]

## ZERO PAGE X

Instructions :  $\Delta$ ADC $\Delta$ \$zz,X (T=1)  
 $\Delta$ AND $\Delta$ \$zz,X (T=1)  
 $\Delta$ EOR $\Delta$ \$zz,X (T=1)  
 $\Delta$ ORA $\Delta$ \$zz,X (T=1)  
 $\Delta$ SBC $\Delta$ \$zz,X (T=1)

Byte length : 2  
 Cycle number : 7

Timing :

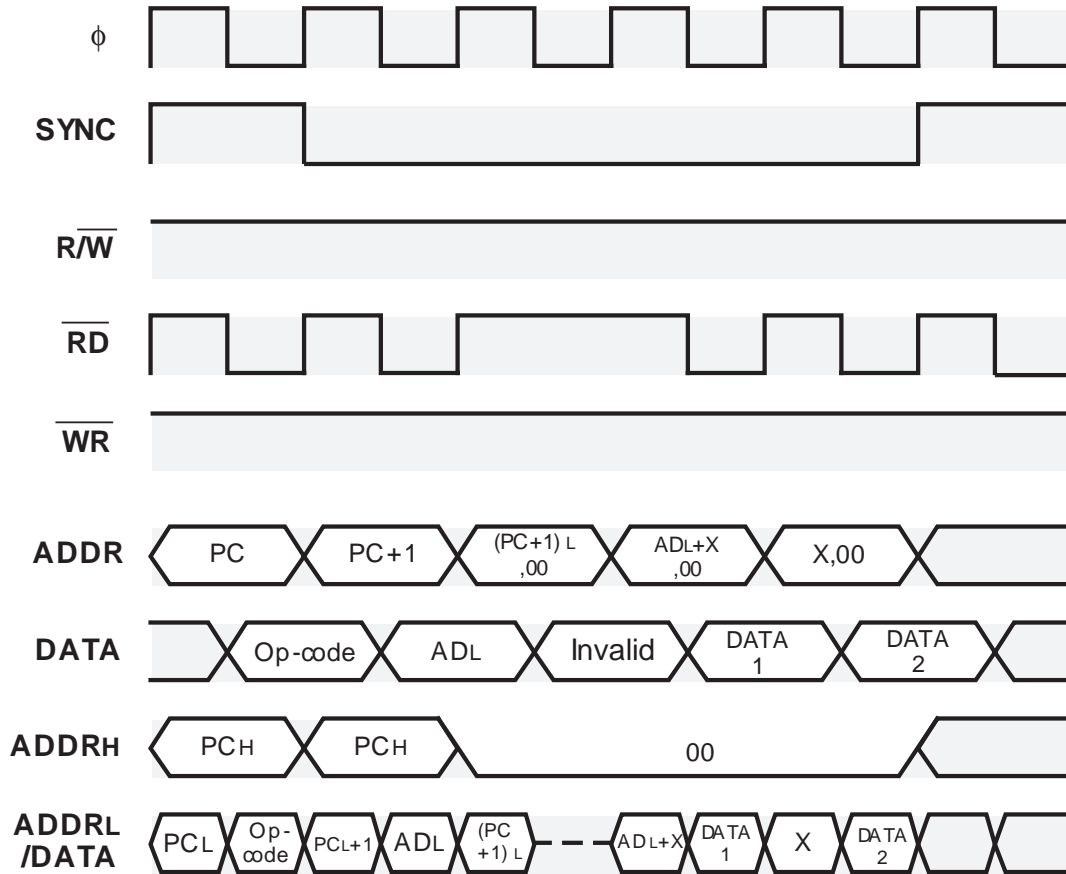


[T=1]

## ZERO PAGE X

Instruction :  $\Delta$ CMP $\Delta$ \$zz,X (T=1)  
 Byte length : 2  
 Cycle number : 5

Timing :

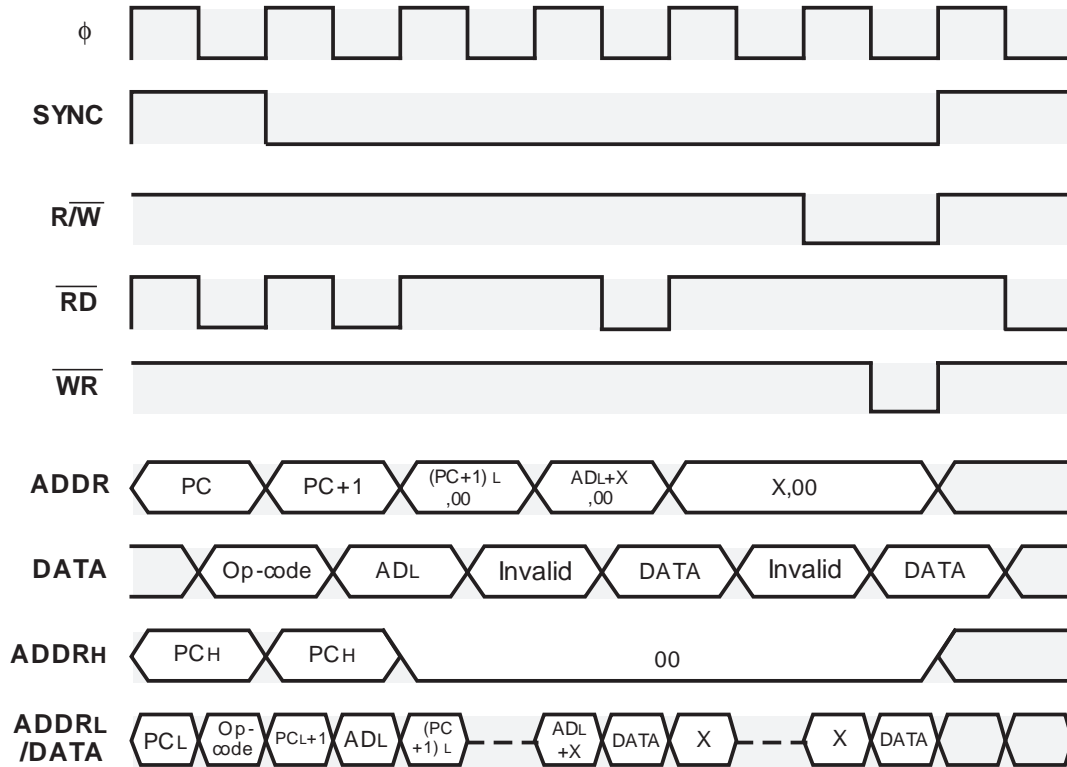


[T=1]

## ZERO PAGE X

Instruction :  $\Delta LDA \Delta \$zz, X$  (T=1)  
 Byte length : 2  
 Cycle number : 6

Timing :



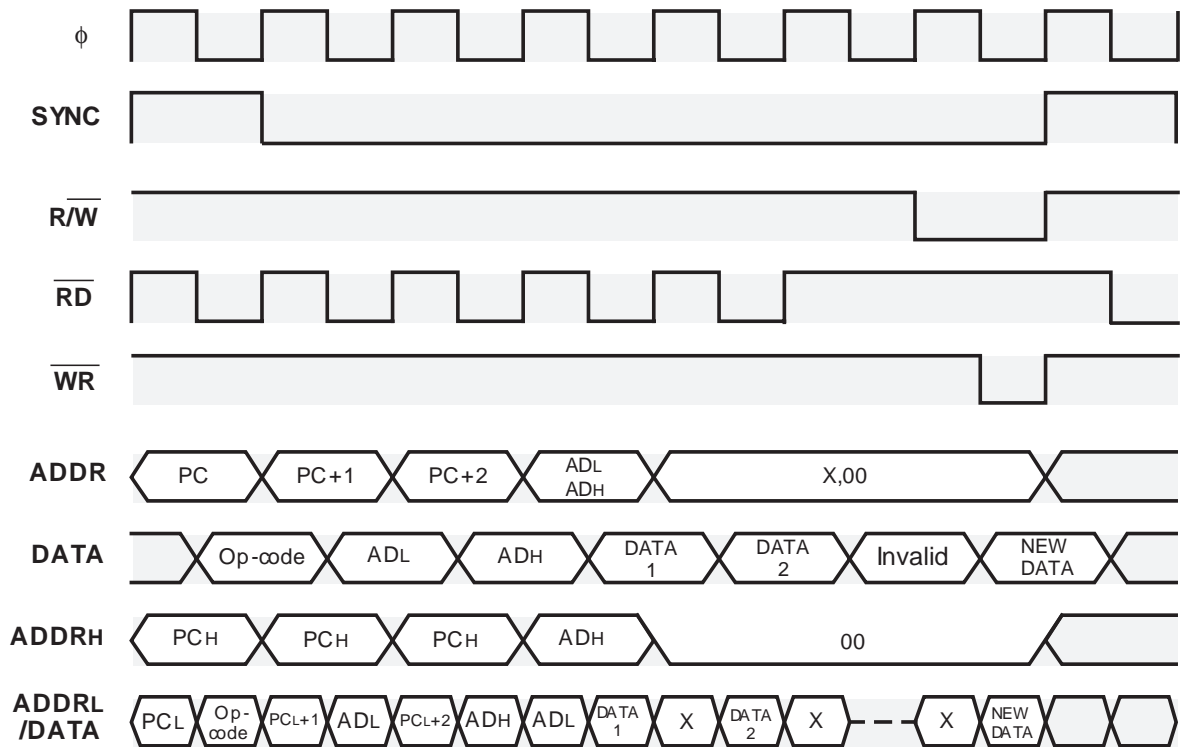
[T=1]

# ABSOLUTE

Instructions :  $\Delta$ ADC $\Delta$ \$hhll (T=1)  
 $\Delta$ AND $\Delta$ \$hhll (T=1)  
 $\Delta$ EOR $\Delta$ \$hhll (T=1)  
 $\Delta$ ORA $\Delta$ \$hhll (T=1)  
 $\Delta$ SBC $\Delta$ \$hhll (T=1)

Byte length : 3  
 Cycle number : 7

Timing :



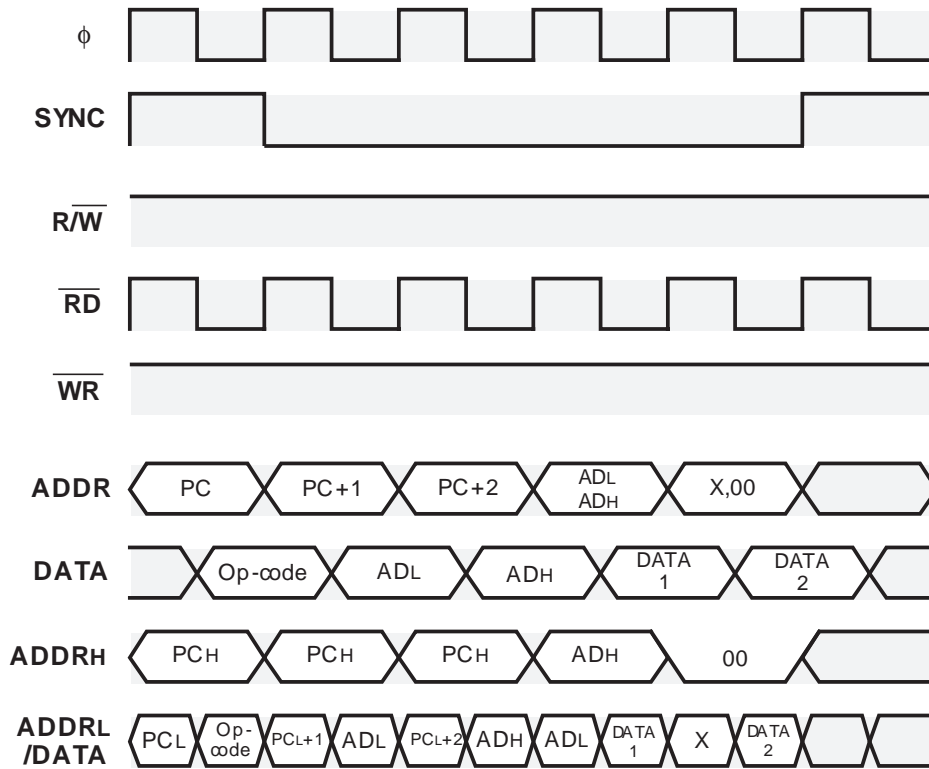


[T=1]

# ABSOLUTE

Instruction :  $\Delta$ CMP $\Delta$ \$hhll (T=1)  
 Byte length : 3  
 Cycle number : 5

Timing :

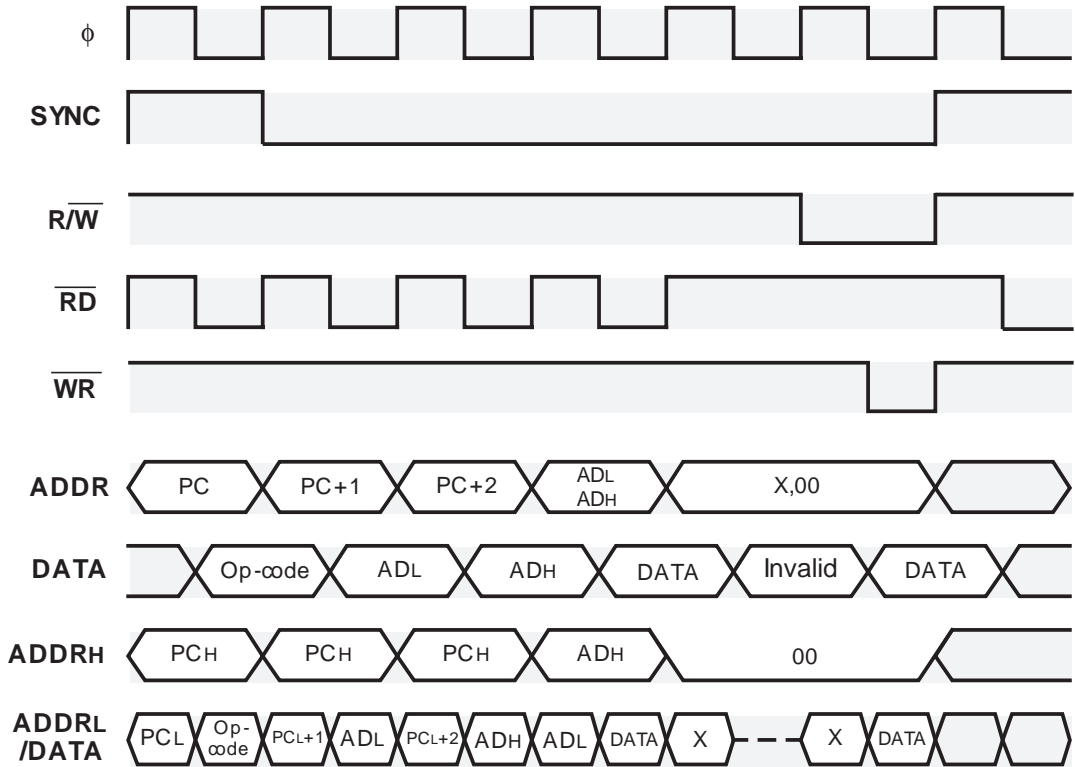


[T=1]

# ABSOLUTE

Instruction :  $\Delta$ LDA $\Delta$ \$hhll (T=1)  
 Byte length : 3  
 Cycle number : 6

Timing :



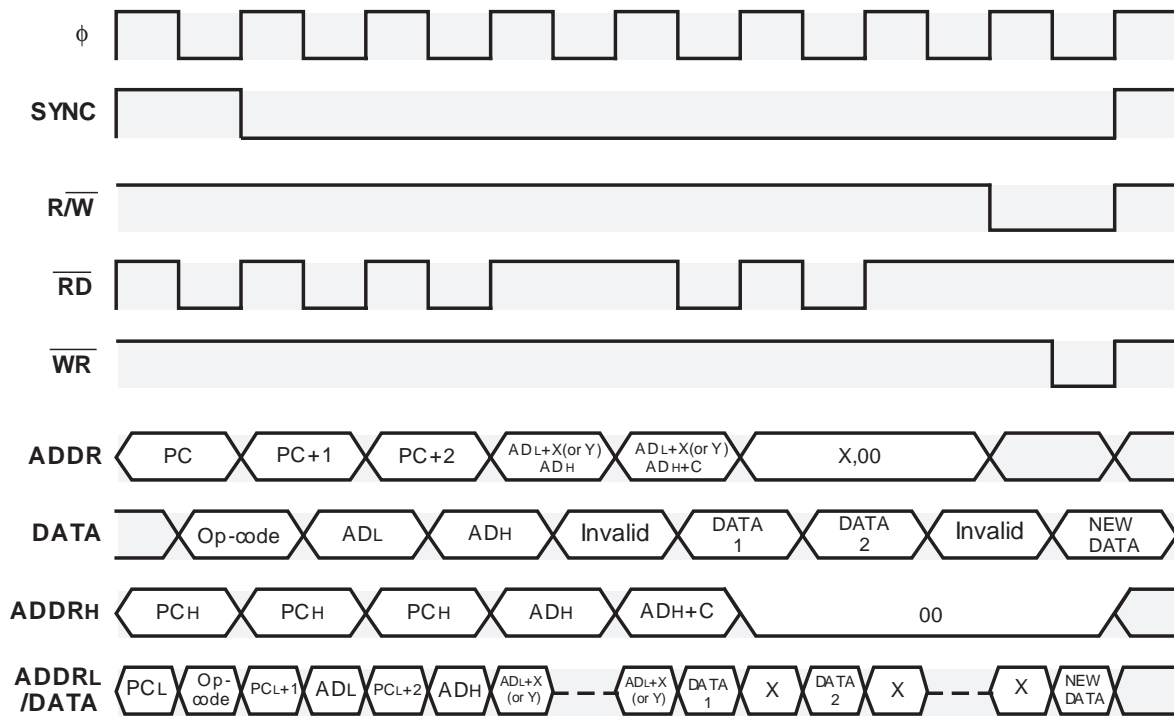
[T=1]

## ABSOLUTE X, ABSOLUTE Y

Instructions :  $\Delta$ ADC  $\Delta$ \$hhll, X or Y (T=1)  
 $\Delta$ AND  $\Delta$ \$hhll, X or Y (T=1)  
 $\Delta$ EOR  $\Delta$ \$hhll, X or Y (T=1)  
 $\Delta$ ORA  $\Delta$ \$hhll, X or Y (T=1)  
 $\Delta$ SBC  $\Delta$ \$hhll, X or Y (T=1)

Byte length : 3  
 Cycle number : 8

Timing :



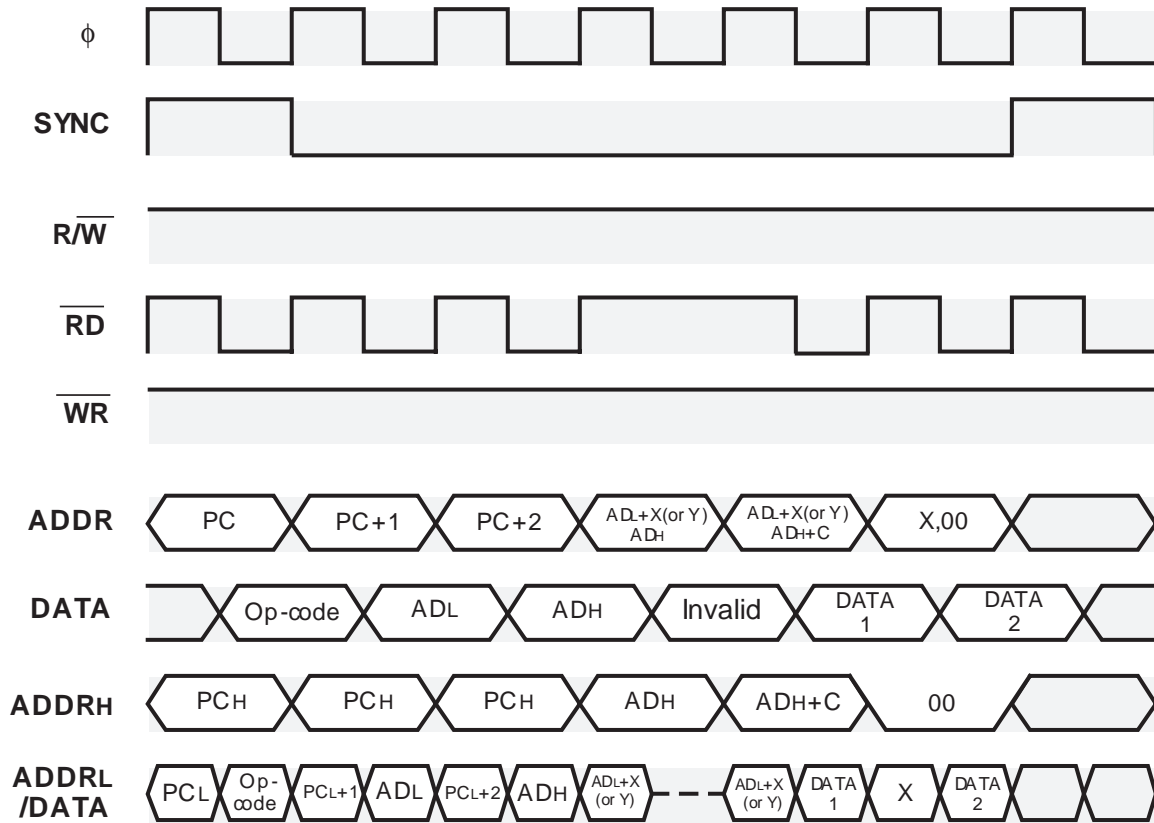
C : Carry of ADL+X or Y

[T=1]

## ABSOLUTE X, ABSOLUTE Y

Instruction :  $\Delta$ CMP $\Delta$ \$hhl,X or Y (T=1)  
 Byte length : 3  
 Cycle number : 6

Timing :



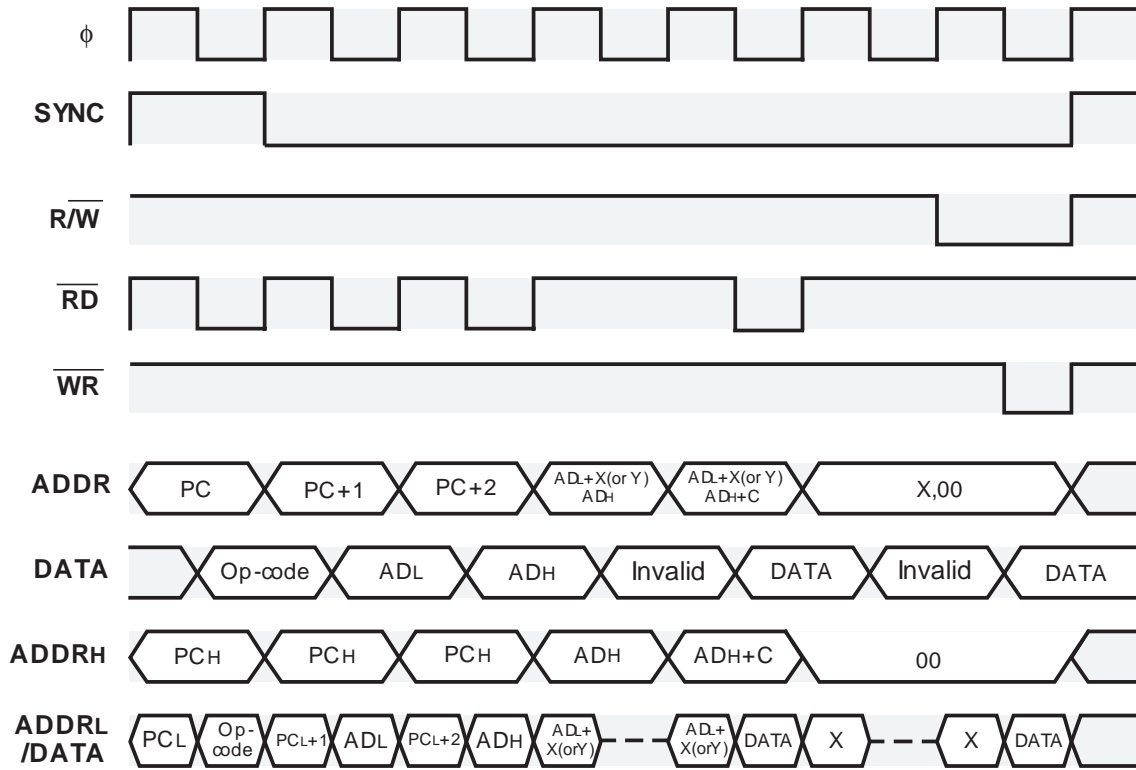
C : Carry of ADL+X or Y

[T=1]

## ABSOLUTE X, ABSOLUTE Y

Instruction :  $\Delta$ LDA $\Delta$ \$hhll,X or Y (T=1)  
 Byte length : 3  
 Cycle number : 7

Timing :



C : Carry of ADL+X or Y

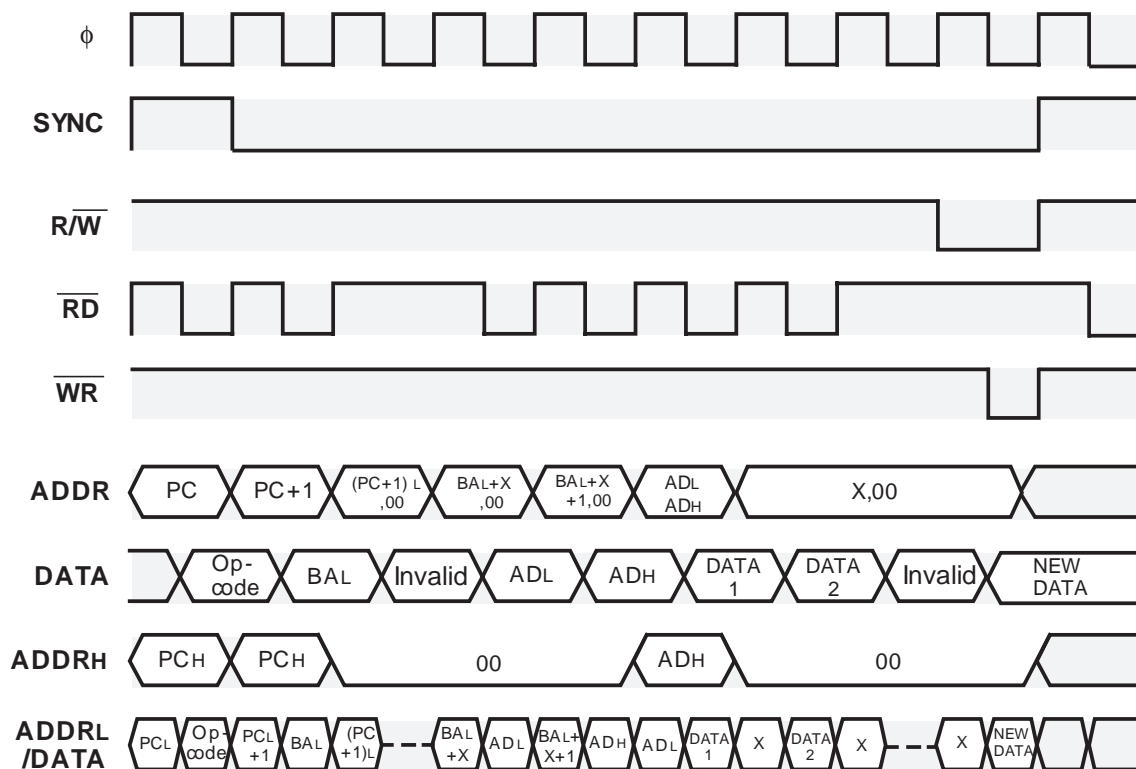
[T=1]

## INDIRECT X

Instructions :  $\Delta\text{ADC}(\$zz,X)$  (T=1)  
 $\Delta\text{AND}(\$zz,X)$  (T=1)  
 $\Delta\text{EOR}(\$zz,X)$  (T=1)  
 $\Delta\text{ORA}(\$zz,X)$  (T=1)  
 $\Delta\text{SBC}(\$zz,X)$  (T=1)

Byte length : 2  
 Cycle number : 9

Timing :



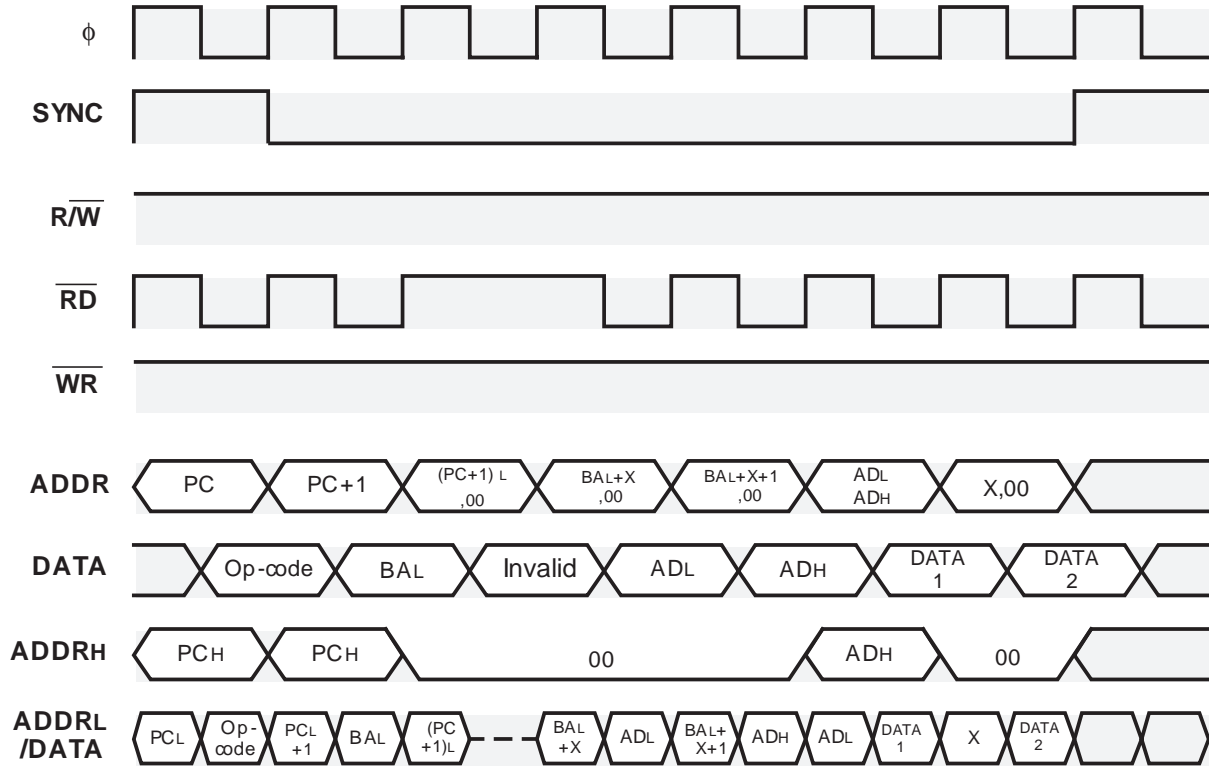
BA : Basic address

[T=1]

## INDIRECT X

Instruction :  $\Delta\text{CMP}\Delta(\$zz,X)$  (T=1)  
 Byte length : 2  
 Cycle number : 7

Timing :



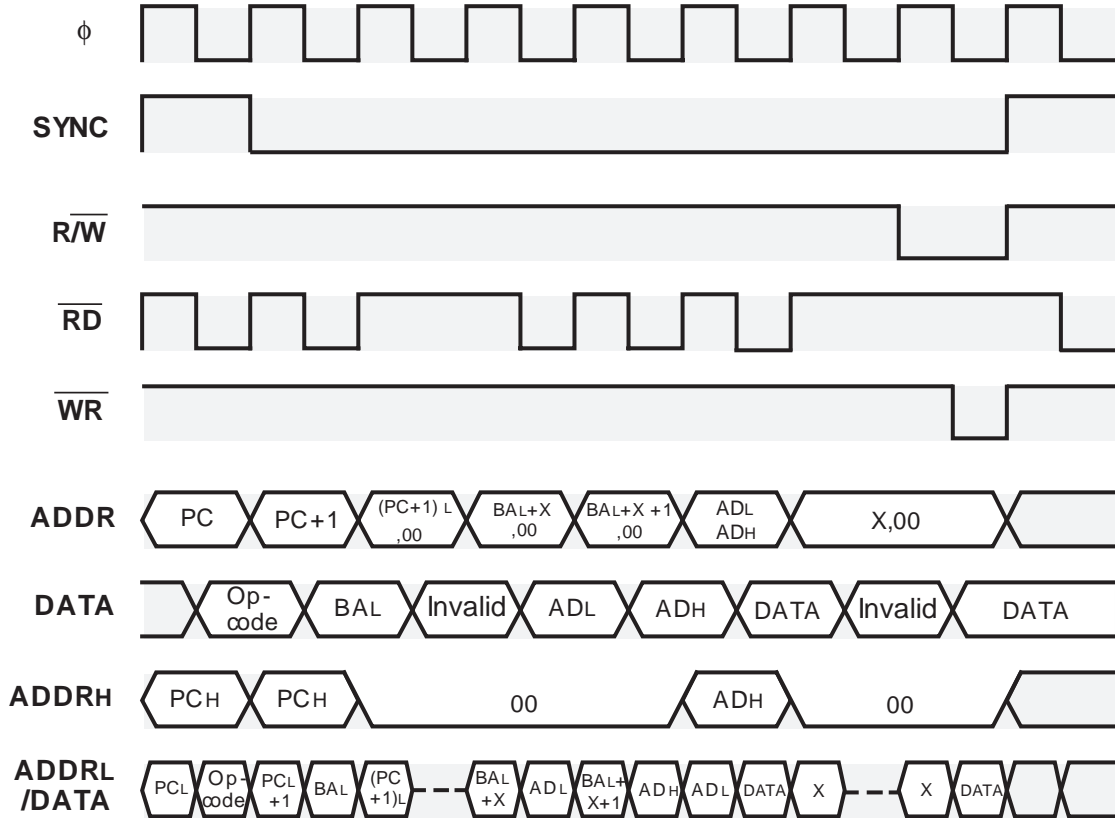
BA : Basic address

[T=1]

## INDIRECT X

Instruction :  $\Delta$ LDA $\Delta$ (\$zz,X) (T=1)  
 Byte length : 2  
 Cycle number : 8

Timing :



BA : Basic address



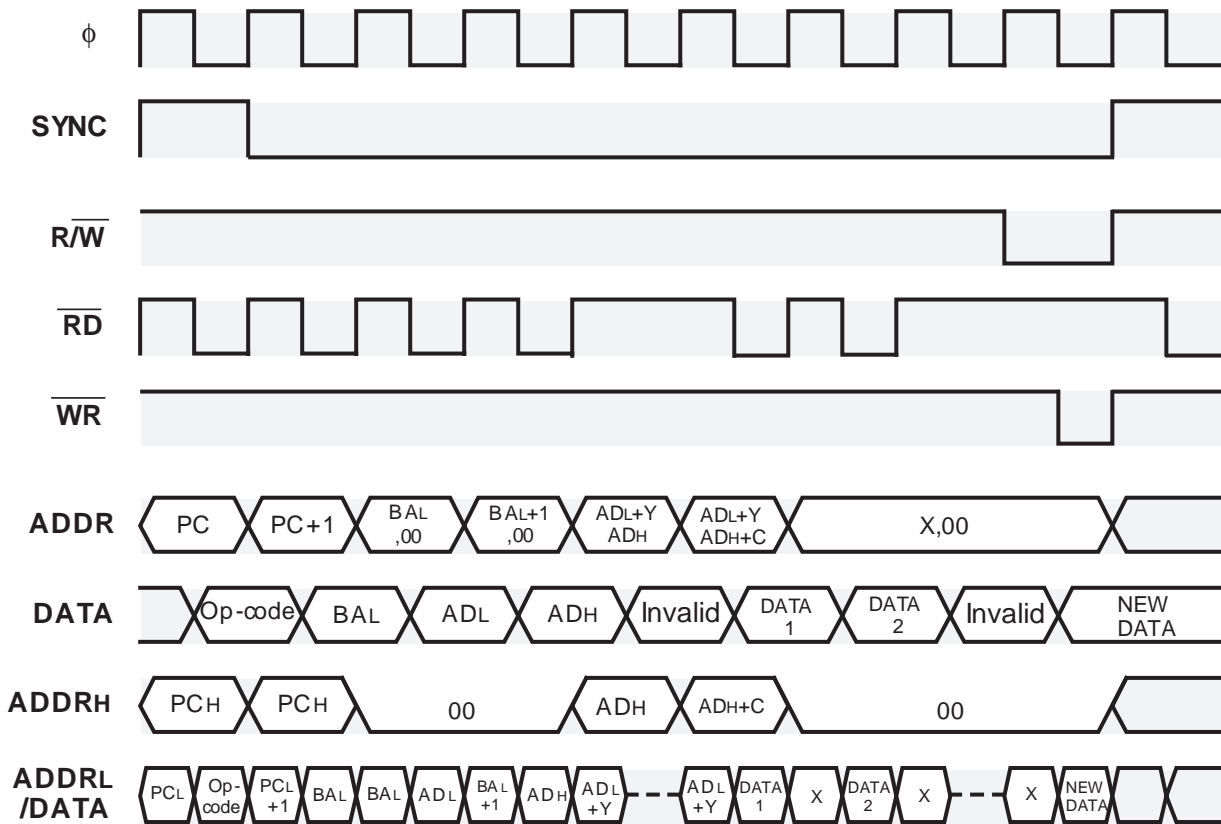
[T=1]

## INDIRECT Y

Instructions :  $\Delta$ ADC( $\$zz$ ),Y (T=1)  
 $\Delta$ AND( $\$zz$ ),Y (T=1)  
 $\Delta$ EOR( $\$zz$ ),Y (T=1)  
 $\Delta$ ORA( $\$zz$ ),Y (T=1)  
 $\Delta$ SBC( $\$zz$ ),Y (T=1)

Byte length : 2  
 Cycle number : 9

Timing :



BA : Basic address

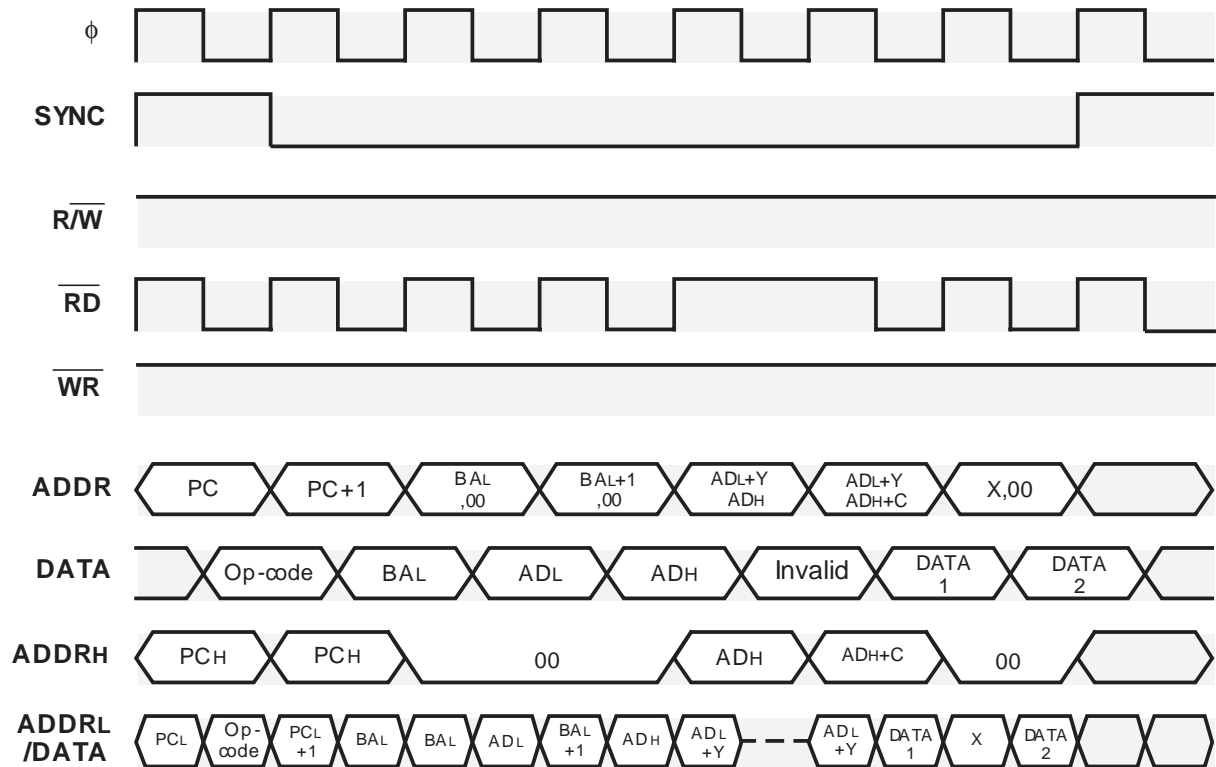
C : Carry of ADL+Y

[T=1]

## INDIRECT Y

Instruction :  $\Delta$ CMP $\Delta$ (\$zz),Y (T=1)  
 Byte length : 2  
 Cycle number : 7

Timing :



BA : Basic address

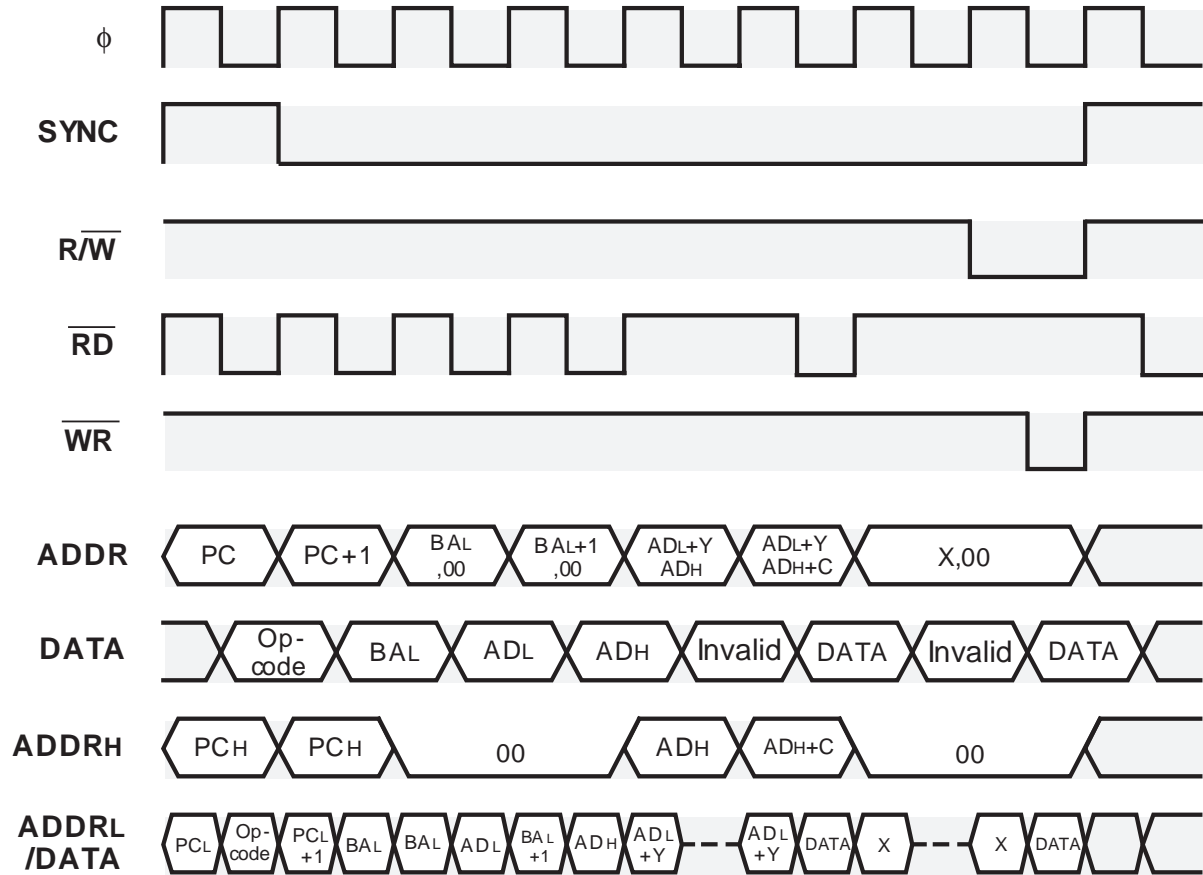
C : Carry of ADL+Y

[T=1]

## INDIRECT Y

Instruction :  $\Delta$ LDA $\Delta$ (\$zz),Y (T=1)  
 Byte length : 2  
 Cycle number : 8

Timing :



BA : Basic address

C : Carry of ADL+Y

# APPENDIX 2

## 740 Family Machine Language Instruction Table

### APPENDIX 2. 740 Family Machine Language Instruction Table

Parameter Classification	SYMBOL	FUNCTION	FLAG	INSTRUCTION CODE		BYTE NUMBER	CYCLE NUMBER	NOTE		
			N V T B D I Z C	D7D6D5D4	D3D2D1D0				HEX	
Data Transfer	Load	LDA # \$ nn	(A)←nn	○×××××○×	1 0 1 0	1 0 0 1	A9	2	2	2
		LDA \$ zz	(A)←(M) where M=(zz)	○×××××○×	1 0 1 0	0 1 0 1	A5	2	3	2
		LDA \$ zz, X	(A)←(M) where M=(zz+(X))	○×××××○×	1 0 1 1	0 1 0 1	B5	2	4	2
		LDA \$ hhll	(A)←(M) where M=(hhll)	○×××××○×	1 0 1 0	1 1 0 1	AD	3	4	2
		LDA \$ hhll, X	(A)←(M) where M=(hhll+(X))	○×××××○×	1 0 1 1	1 1 0 1	BD	3	5	2
		LDA \$ hhll, Y	(A)←(M) where M=(hhll+(Y))	○×××××○×	1 0 1 1	1 0 0 1	B9	3	5	2
		LDA (\$ zz, X)	(A)←(M) where M=((zz+(X)+1)(zz+(X)))	○×××××○×	1 0 1 0	0 0 0 1	A1	2	6	2
		LDA (\$ zz), Y	(A)←(M) where M=((zz+1)(zz)+(Y))	○×××××○×	1 0 1 1	0 0 0 1	B1	2	6	2
		LDX # \$ nn	(X)←nn	○×××××○×	1 0 1 0	0 0 1 0	A2	2	2	
		LDX \$ zz	(X)←(M) where M=(zz)	○×××××○×	1 0 1 0	0 1 1 0	A6	2	3	
		LDX \$ zz, Y	(X)←(M) where M=(zz+(Y))	○×××××○×	1 0 1 1	0 1 1 0	B6	2	4	
		LDX \$ hhll	(X)←(M) where M=(hhll)	○×××××○×	1 0 1 0	1 1 1 0	AE	3	4	
		LDX \$ hhll, Y	(X)←(M) where M=(hhll+(Y))	○×××××○×	1 0 1 1	1 1 1 0	BE	3	5	
		LDY # \$ nn	(Y)←nn	○×××××○×	1 0 1 0	0 0 0 0	A0	2	2	
		LDY \$ zz	(Y)←(M) where M=(zz)	○×××××○×	1 0 1 0	0 1 0 0	A4	2	3	
	LDY \$ zz, X	(Y)←(M) where M=(zz+(X))	○×××××○×	1 0 1 1	0 1 0 0	B4	2	4		
	LDY \$ hhll	(Y)←(M) where M=(hhll)	○×××××○×	1 0 1 0	1 1 0 0	AC	3	4		
	LDY \$ hhll, X	(Y)←(M) where M=(hhll+(X))	○×××××○×	1 0 1 1	1 1 0 0	BC	3	5		
	LDM # \$ nn, \$ zz	(M)←nn where M=(zz)	××××××××	0 0 1 1	1 1 0 0	3C	3	4		
	Store	STA \$ zz	(M)←(A) where M=(zz)	××××××××	1 0 0 0	0 1 0 1	85	2	4	
		STA \$ zz, X	(M)←(A) where M=(zz+(X))	××××××××	1 0 0 1	0 1 0 1	95	2	5	
		STA \$ hhll	(M)←(A) where M=(hhll)	××××××××	1 0 0 0	1 1 0 1	8D	3	5	
		STA \$ hhll, X	(M)←(A) where M=(hhll+(X))	××××××××	1 0 0 1	1 1 0 1	9D	3	6	
		STA \$ hhll, Y	(M)←(A) where M=(hhll+(Y))	××××××××	1 0 0 1	1 0 0 1	99	3	6	
		STA (\$ zz, X)	(M)←(A) where M=((zz+(X)+1)(zz+(X)))	××××××××	1 0 0 0	0 0 0 1	81	2	7	
		STA (\$ zz), Y	(M)←(A) where M=((zz+1)(zz)+(Y))	××××××××	1 0 0 1	0 0 0 1	91	2	7	
		STX \$ zz	(M)←(X) where M=(zz)	××××××××	1 0 0 0	0 1 1 0	86	2	4	
		STX \$ zz, Y	(M)←(X) where M=(zz+(Y))	××××××××	1 0 0 1	0 1 1 0	96	2	5	
		STX \$ hhll	(M)←(X) where M=(hhll)	××××××××	1 0 0 0	1 1 1 0	8E	3	5	
		STY \$ zz	(M)←(Y) where M=(zz)	××××××××	1 0 0 0	0 1 0 0	84	2	4	
STY \$ zz, X		(M)←(Y) where M=(zz+(X))	××××××××	1 0 0 1	0 1 0 0	94	2	5		
STY \$ hhll		(M)←(Y) where M=(hhll)	××××××××	1 0 0 0	1 1 0 0	8C	3	6		
Transfer		TAX	(X)←(A)	○×××××○×	1 0 1 0	1 0 1 0	AA	1	2	
		TXA	(A)←(X)	○×××××○×	1 0 0 0	1 0 1 0	8A	1	2	
	TAY	(Y)←(A)	○×××××○×	1 0 1 0	1 0 0 0	A8	1	2		
	TYA	(A)←(Y)	○×××××○×	1 0 0 1	1 0 0 0	98	1	2		
	TSX	(X)←(S)	○×××××○×	1 0 1 1	1 0 1 0	BA	1	2		
TXS	(S)←(X)	××××××××	1 0 0 1	1 0 1 0	9A	1	2			
Stack Operation	PHA	(M(S))←(A), (S)←(S)-1	××××××××	0 1 0 0	1 0 0 0	48	1	3		
	PHP	(M(S))←(PS), (S)←(S)-1	××××××××	0 0 0 0	1 0 0 0	08	1	3		
	PLA	(S)←(S)+1, (A)←(M(S))	○×××××○×	0 1 1 0	1 0 0 0	68	1	4		
PLP	(S)←(S)+1, (PS)←(M(S))	Previous status in stack	0 0 1 0	1 0 0 0	28	1	4			

# 740 Family Machine Language Instruction Table

Parameter Classification	SYMBOL	FUNCTION	FLAG					INSTRUCTION CODE			BYTE NUMBER	CYCLE NUMBER	NOTE									
			N	V	T	B	D	I	Z	C				D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub>	D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	HEX						
Operation Add and Subtract	ADC # \$ nn	$(A) \leftarrow (A) + nn + (C)$	0	0	x	x	x	x	0	0	0	1	1	0	1	0	0	1	69	2	2	1
	ADC \$ zz	$(A) \leftarrow (A) + (M) + (C)$ where $M = (zz)$	0	0	x	x	x	x	0	0	0	1	1	0	0	1	0	1	65	2	3	1
	ADC \$ zz, X	$(A) \leftarrow (A) + (M) + (C)$ where $M = (zz + (X))$	0	0	x	x	x	x	0	0	0	1	1	1	0	1	0	1	75	2	4	1
	ADC \$ hhll	$(A) \leftarrow (A) + (M) + (C)$ where $M = (hhll)$	0	0	x	x	x	x	0	0	0	1	1	0	1	1	1	0	6D	3	4	1
	ADC \$ hhll, X	$(A) \leftarrow (A) + (M) + (C)$ where $M = (hhll + (X))$	0	0	x	x	x	x	0	0	0	1	1	1	1	1	1	0	7D	3	5	1
	ADC \$ hhll, Y	$(A) \leftarrow (A) + (M) + (C)$ where $M = (hhll + (Y))$	0	0	x	x	x	x	0	0	0	1	1	1	1	1	0	0	79	3	5	1
	ADC (\$ zz, X)	$(A) \leftarrow (A) + (M) + (C)$ where $M = ((zz + (X) + 1)(zz + (X)))$	0	0	x	x	x	x	0	0	0	1	1	0	0	0	0	0	61	2	6	1
	ADC (\$ zz, Y)	$(A) \leftarrow (A) + (M) + (C)$ where $M = ((zz + 1)(zz + (Y)))$	0	0	x	x	x	x	0	0	0	1	1	1	0	0	0	0	71	2	6	1
	SBC # \$ nn	$(A) \leftarrow (A) - nn - (\bar{C})$	0	0	x	x	x	x	0	0	1	1	1	0	1	1	0	0	E9	2	2	1
	SBC \$ zz	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = (zz)$	0	0	x	x	x	x	0	0	1	1	1	0	0	1	0	1	E5	2	3	1
	SBC \$ zz, X	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = (zz + (X))$	0	0	x	x	x	x	0	0	1	1	1	1	0	1	0	1	F5	2	4	1
	SBC \$ hhll	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = (hhll)$	0	0	x	x	x	x	0	0	1	1	1	0	1	1	1	0	ED	3	4	1
	SBC \$ hhll, X	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = (hhll + (X))$	0	0	x	x	x	x	0	0	1	1	1	0	1	1	1	0	FD	3	5	1
	SBC \$ hhll, Y	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = (hhll + (Y))$	0	0	x	x	x	x	0	0	1	1	1	1	1	1	0	0	F9	3	5	1
	SBC (\$ zz, X)	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = ((zz + (X) + 1)(zz + (X)))$	0	0	x	x	x	x	0	0	1	1	1	0	0	0	0	0	E1	2	6	1
	SBC (\$ zz, Y)	$(A) \leftarrow (A) - (M) - (\bar{C})$ where $M = ((zz + 1)(zz + (Y)))$	0	0	x	x	x	x	0	0	1	1	1	1	0	0	0	0	F1	2	6	1
	INC A	$(A) \leftarrow (A) + 1$	0	x	x	x	x	x	0	x	0	0	1	1	1	0	1	0	3A	1	2	
	INC \$ zz	$(M) \leftarrow (M) + 1$ where $M = (zz)$	0	x	x	x	x	x	0	x	1	1	1	0	0	1	1	0	E6	2	5	
	INC \$ zz, X	$(M) \leftarrow (M) + 1$ where $M = (zz + (X))$	0	x	x	x	x	x	0	x	1	1	1	1	0	1	1	0	F6	2	6	
	INC \$ hhll	$(M) \leftarrow (M) + 1$ where $M = (hhll)$	0	x	x	x	x	x	0	x	1	1	1	0	1	1	1	0	EE	3	6	
	INC \$ hhll, X	$(M) \leftarrow (M) + 1$ where $M = (hhll + (X))$	0	x	x	x	x	x	0	x	1	1	1	1	1	1	1	0	FE	3	7	
	DEC A	$(A) \leftarrow (A) - 1$	0	x	x	x	x	x	0	x	0	0	0	1	1	0	1	0	1A	1	2	
	DEC \$ zz	$(M) \leftarrow (M) - 1$ where $M = (zz)$	0	x	x	x	x	x	0	x	1	1	0	0	0	1	1	0	C6	2	5	
	DEC \$ zz, X	$(M) \leftarrow (M) - 1$ where $M = (zz + (X))$	0	x	x	x	x	x	0	x	1	1	0	1	0	1	1	0	D6	2	6	
	DEC \$ hhll	$(M) \leftarrow (M) - 1$ where $M = (hhll)$	0	x	x	x	x	x	0	x	1	1	0	0	1	1	1	0	CE	3	6	
	DEC \$ hhll, X	$(M) \leftarrow (M) - 1$ where $M = (hhll + (X))$	0	x	x	x	x	x	0	x	1	1	0	1	1	1	1	0	DE	3	7	
	INX	$(X) \leftarrow (X) + 1$	0	x	x	x	x	x	0	x	1	1	1	0	1	0	0	0	E8	1	2	
	DEX	$(X) \leftarrow (X) - 1$	0	x	x	x	x	x	0	x	1	1	0	0	1	0	1	0	CA	1	2	
INY	$(Y) \leftarrow (Y) + 1$	0	x	x	x	x	x	0	x	1	1	0	0	1	0	0	0	C8	1	2		
DEY	$(Y) \leftarrow (Y) - 1$	0	x	x	x	x	x	0	x	1	0	0	0	1	0	0	0	88	1	2		
Multiply / Divide	MUL \$ zz, X	$M(S), (A) \leftarrow (A) \times M(zz + (X))$ $(S) \leftarrow (S) - 1$	x	x	x	x	x	x	x	0	1	1	0	0	0	1	0	62	2	15		
	DIV \$ zz, X	$(A) \leftarrow (M(zz + (X) + 1), M(zz + (X)) \div (A))$ $M(S) \leftarrow$ One's complement of remainder $(S) \leftarrow (S) - 1$	x	x	x	x	x	x	x	1	1	1	0	0	0	1	0	E2	2	16		

# 740 Family Machine Language Instruction Table

Parameter Classification	SYMBOL	FUNCTION	FLAG		INSTRUCTION CODE		BYTE NUMBER	CYCLE NUMBER	NOTE									
			N	V	T	B				D	I	Z	C	D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub>	D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	HEX		
Operation	AND # \$ nn	(A)←(A)∧nn										0 0 1 0	1 0 0 1	29	2	2	1	
	AND \$ zz	(A)←(A)∧(M) where M=(zz)										0 0 1 0	0 1 0 1	25	2	3	1	
	AND \$ zz, X	(A)←(A)∧(M) where M=(zz+(X))										0 0 1 1	0 1 0 1	35	2	4	1	
	AND \$ hhll	(A)←(A)∧(M) where M=(hhll)										0 0 1 0	1 1 0 1	2D	3	4	1	
	AND \$ hhll, X	(A)←(A)∧(M) where M=(hhll+(X))										0 0 1 1	1 1 0 1	3D	3	5	1	
	AND \$ hhll, Y	(A)←(A)∧(M) where M=(hhll+(Y))										0 0 1 1	1 0 0 1	39	3	5	1	
	AND (\$ zz, X)	(A)←(A)∧(M) where M=((zz+(X)+1)(zz+(X)))										0 0 1 0	0 0 0 1	21	2	6	1	
	AND (\$ zz, Y)	(A)←(A)∧(M) where M=((zz+1)(zz)+(Y))										0 0 1 1	0 0 0 1	31	2	6	1	
	ORA # \$ nn	(A)←(A)∨nn										0 0 0 0	1 0 0 1	09	2	2	1	
	ORA \$ zz	(A)←(A)∨(M) where M=(zz)										0 0 0 0	0 1 0 1	05	2	3	1	
	ORA \$ zz, X	(A)←(A)∨(M) where M=(zz+(X))										0 0 0 1	0 1 0 1	15	2	4	1	
	ORA \$ hhll	(A)←(A)∨(M) where M=(hhll)										0 0 0 0	1 1 0 1	0D	3	4	1	
	ORA \$ hhll, X	(A)←(A)∨(M) where M=(hhll+(X))										0 0 0 1	1 1 0 1	1D	3	5	1	
	ORA \$ hhll, Y	(A)←(A)∨(M) where M=(hhll+(Y))										0 0 0 1	1 0 0 1	19	3	5	1	
	ORA (\$ zz, X)	(A)←(A)∨(M) where M=((zz+(X)+1)(zz+(X)))										0 0 0 0	0 0 0 1	01	2	6	1	
	ORA (\$ zz, Y)	(A)←(A)∨(M) where M=((zz+1)(zz)+(Y))										0 0 0 1	0 0 0 1	11	2	6	1	
	EOR # \$ nn	(A)←(A)∨nn										0 1 0 0	1 0 0 1	49	2	2	1	
	EOR \$ zz	(A)←(A)∨(M) where M=(zz)										0 1 0 0	0 1 0 1	45	2	3	1	
	EOR \$ zz, X	(A)←(A)∨(M) where M=(zz+(X))										0 1 0 1	0 1 0 1	55	2	4	1	
	EOR \$ hhll	(A)←(A)∨(M) where M=(hhll)										0 1 0 0	1 1 0 1	4D	3	4	1	
	EOR \$ hhll, X	(A)←(A)∨(M) where M=(hhll+(X))										0 1 0 1	1 1 0 1	5D	3	5	1	
	EOR \$ hhll, Y	(A)←(A)∨(M) where M=(hhll+(Y))										0 1 0 1	1 0 0 1	59	3	5	1	
	EOR (\$ zz, X)	(A)←(A)∨(M) where M=((zz+(X)+1)(zz+(X)))										0 1 0 0	0 0 0 1	41	2	6	1	
	EOR (\$ zz, Y)	(A)←(A)∨(M) where M=((zz+1)(zz)+(Y))										0 1 0 1	0 0 0 1	51	2	6	1	
	COM \$ zz	(M)←(M̄) where M=(zz)										0 1 0 0	0 1 0 0	44	2	5		
	BIT \$ zz	(A)∧(M) where M=(zz)									M7M6	0 0 1 0	0 1 0 0	24	2	3		
	BIT \$ hhll	(A)∧(M) where M=(hhll)									M7M6	0 0 1 0	1 1 0 0	2C	3	4		
	TST \$ zz	(M)=0? where M=(zz)										0 1 1 0	0 1 0 0	64	2	3		
	Comparison	CMP # \$ nn	(A)-nn	Comparison in size									1 1 1 0	1 0 0 1	C9	2	2	3
		CMP \$ zz	(A)-(M) where M=(zz)										1 1 1 0	0 1 0 1	C5	2	3	3
		CMP \$ zz, X	(A)-(M) where M=(zz+(X))										1 1 1 1	0 1 0 1	D5	2	4	3
CMP \$ hhll		(A)-(M) where M=(hhll)										1 1 1 0	1 1 0 1	CD	3	4	3	
CMP \$ hhll, X		(A)-(M) where M=(hhll+(X))										1 1 1 1	1 1 0 1	DD	3	5	3	
CMP \$ hhll, Y		(A)-(M) where M=(hhll+(Y))										1 1 1 1	1 0 0 1	D9	3	5	3	
CMP (\$ zz, X)		(A)-(M) where M=((zz+(X)+1)(zz+(X)))										1 1 1 0	0 0 0 1	C1	2	6	3	
CMP (\$ zz, Y)		(A)-(M) where M=((zz+1)(zz)+(Y))										1 1 1 1	0 0 0 1	D1	2	6	3	
CPX # \$ nn		(X)-nn	Comparison in size									1 1 1 0	0 0 0 0	E0	2	2		
CPX \$ zz		(X)-(M) where M=(zz)										1 1 1 0	0 1 0 0	E4	2	3		
CPX \$ hhll		(X)-(M) where M=(hhll)										1 1 1 0	1 1 0 0	EC	3	4		
CPY # \$ nn		(Y)-nn	Comparison in size									1 1 1 0	0 0 0 0	C0	2	2		
CPY \$ zz		(Y)-(M) where M=(zz)										1 1 1 0	0 1 0 0	C4	2	3		
CPY \$ hhll		(Y)-(M) where M=(hhll)										1 1 1 0	1 1 0 0	CC	3	4		

# 740 Family Machine Language Instruction Table

Parameter Classification	SYMBOL	FUNCTION	FLAG		INSTRUCTION CODE			BYTE NUMBER	CYCLE NUMBER	NOTE												
			N	V	T	B	D				I	Z	C	D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub>	D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	HEX						
Operation	Rotate and Shift	ASL A ASL \$zz	Left Shift $C \leftarrow [A7A6 \quad A1A0] \leftarrow 0$ where $M=(zz)$	0	x	x	x	x	0	0	0	0	0	0	1	0	1	0	0A	1	2	
		0		x	x	x	x	0	0	0	0	0	0	0	1	1	0	06	2	5		
		ASL \$zz, X	Left Shift $C \leftarrow [M7M6 \quad M1M0] \leftarrow 0$ where $M=(zz+(X))$	0	x	x	x	x	0	0	0	0	0	1	<B2>	0	1	1	0	16	2	6
		0		x	x	x	x	0	0	0	0	0	0	<B2>	1	1	1	0	0E	3	6	
		ASL \$hhll, X	Left Shift $C \leftarrow [M7M6 \quad M1M0] \leftarrow 0$ where $M=(hhll+(X))$	0	x	x	x	x	0	0	0	0	0	1	<B2>	1	1	1	0	1E	3	7
		0		x	x	x	x	0	0	0	0	0	1	<B2>	1	1	1	0	1E	3	7	
		LSR A LSR \$zz	Right Shift $0 \rightarrow [A7A6 \quad A1A0] \rightarrow C$ where $M=(zz)$	0	x	x	x	x	0	0	0	1	0	0	1	0	1	0	4A	1	2	
		0		x	x	x	x	0	0	0	1	0	0	0	1	1	0	46	2	5		
	LSR \$zz, X	Right Shift $0 \rightarrow [M7M6 \quad M1M0] \rightarrow C$ where $M=(zz+(X))$	0	x	x	x	x	0	0	0	1	0	1	<B2>	0	1	1	0	56	2	6	
	0		x	x	x	x	0	0	0	1	0	0	<B2>	1	1	1	0	4E	3	6		
	LSR \$hhll, X	Right Shift $0 \rightarrow [M7M6 \quad M1M0] \rightarrow C$ where $M=(hhll+(X))$	0	x	x	x	x	0	0	0	1	0	0	<B2>	1	1	1	0	5E	3	7	
	0		x	x	x	x	0	0	0	1	0	1	<B2>	1	1	1	0	5E	3	7		
	Rotate and Shift	ROL A ROL \$zz	Left Shift $[A7A6 \quad A1A0] \leftarrow C \leftarrow$ where $M=(zz)$	0	x	x	x	x	0	0	0	0	1	0	1	0	1	0	2A	1	2	
		0		x	x	x	x	0	0	0	0	1	0	0	1	1	0	26	2	5		
		ROL \$zz, X	Left Shift $[M7M6 \quad M1M0] \leftarrow C \leftarrow$ where $M=(zz+(X))$	0	x	x	x	x	0	0	0	0	1	1	<B2>	0	1	1	0	36	2	6
		0		x	x	x	x	0	0	0	0	1	0	<B2>	1	1	1	0	2E	3	6	
		ROL \$hhll, X	Left Shift $[M7M6 \quad M1M0] \leftarrow C \leftarrow$ where $M=(hhll+(X))$	0	x	x	x	x	0	0	0	0	1	0	<B2>	1	1	1	0	3E	3	7
		0		x	x	x	x	0	0	0	0	1	1	<B2>	1	1	1	0	3E	3	7	
	Rotate and Shift	ROR A ROR \$zz	Right Shift $C \rightarrow [A7A6 \quad A1A0] \rightarrow$ where $M=(zz)$	0	x	x	x	x	0	0	1	1	0	0	1	0	1	0	6A	1	2	
0		x		x	x	x	0	0	0	1	1	0	0	1	1	0	66	2	5			
ROR \$zz, X		Right Shift $C \rightarrow [M7M6 \quad M1M0] \rightarrow$ where $M=(zz+(X))$	0	x	x	x	x	0	0	1	1	1	0	<B2>	0	1	1	0	76	2	6	
0			x	x	x	x	0	0	0	1	1	0	<B2>	1	1	1	0	6E	3	6		
ROR \$hhll, X		Right Shift $C \rightarrow [M7M6 \quad M1M0] \rightarrow$ where $M=(hhll+(X))$	0	x	x	x	x	0	0	0	1	1	0	<B2>	1	1	1	0	7E	3	7	
0			x	x	x	x	0	0	0	1	1	1	<B2>	1	1	1	0	7E	3	7		
Bit Management	CLB i, A CLB i, \$zz	(Ai) ← 0 (Mi) ← 0	where i=0-7 where i=0-7, M=(zz)	x	x	x	x	x	x	x	1	0	0	0	0	0	1	0	82	2	8	
	SEB i, A SEB i, \$zz	(Ai) ← 1 (Mi) ← 1	where i=0-7 where i=0-7, M=(zz)	x	x	x	x	x	x	x	1	0	0	0	0	0	1	0	82	2	8	
	SEB i, \$zz	(Mi) ← 1	where i=0-7, M=(zz)	x	x	x	x	x	x	x	1	0	0	0	0	0	1	0	82	2	8	
Flag setting	CLC	(C) ← 0		x	x	x	x	x	x	0	0	0	0	1	1	0	0	0	18	1	2	
	SEC	(C) ← 1		x	x	x	x	x	x	1	0	0	1	1	1	0	0	0	38	1	2	
	CLD	(D) ← 0		x	x	x	x	0	x	x	1	1	0	1	1	0	0	0	D8	1	2	
	SED	(D) ← 1		x	x	x	x	1	x	x	1	1	1	1	1	0	0	0	F8	1	2	
	CLI	(I) ← 0		x	x	x	x	0	x	x	0	1	0	1	1	0	0	0	58	1	2	
	SEI	(I) ← 1		x	x	x	x	1	x	x	0	1	1	1	1	0	0	0	78	1	2	
	CLT	(T) ← 0		x	x	0	x	x	x	x	0	0	0	1	0	0	1	0	12	1	2	
	SET	(T) ← 1		x	x	1	x	x	x	x	0	0	1	1	0	0	1	0	32	1	2	
	CLV	(V) ← 0		x	0	x	x	x	x	x	1	0	1	1	1	0	0	0	B8	1	2	

# 740 Family Machine Language Instruction Table

Parameter Classification	SYMBOL	FUNCTION	FLAG					INSTRUCTION CODE			BYTE NUMBER	CYCLE NUMBER	NOTE			
			N	V	T	B	D	I	Z	C				D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub>	D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	HEX
Branch and Return	Jump	BRA \$ hhl	(PC) ← (PC)+2+Rel	X	X	X	X	X	X	X	1 0 0 0	0 0 0 0	80	2	4	4
		JMP \$ hhl	(PC) ← hhl	X	X	X	X	X	X	0 1 0 0	1 1 0 0	4C	3	3		
		JMP (\$ hhl)	(PCL) ← (hhl), (PCH) ← (hhl+1)	X	X	X	X	X	X	0 1 1 0	1 1 0 0	6C	3	5		
		JMP (\$ zz)	(PCL) ← (zz), (PCH) ← (zz+1)	X	X	X	X	X	X	1 0 1 1	0 0 1 0	B2	2	4		
		JSR \$ hhl	(M(S))←(PCH), (S)←(S)-1, (M(S))←(PCL), (S)←(S)-1, and (PC)←hhl	X	X	X	X	X	X	0 0 1 0	0 0 0 0	20	3	6		
		JSR (\$ zz)	(M(S))←(PCH), (S)←(S)-1, (M(S))←(PCL), (S)←(S)-1, (PCL)←(zz), and (PCH)←(zz+1)	X	X	X	X	X	X	0 0 0 0	0 0 1 0	02	2	7		
		JSR \ \$ hhl	(M(S))←(PCH), (S)←(S)-1, (M(S))←(PCL), (S)←(S)-1, (PCL)←ll, and (PCH)←FF	X	X	X	X	X	X	0 0 1 0	0 0 1 0	22	2	5		
	Branch	BBC i, A, \$ hhl	When (Ai)=0 (PC) ← (PC)+2+Rel Where i=0-7 When (Ai)=1 (PC) ← (PC)+2	X	X	X	X	X	X	i i i 1	0 0 1 1	(1+2)x10+3	2	4	4	
		BBC i, \$ zz, \$ hhl	When (Mi)=0 (PC) ← (PC)+3+Rel Where i=0-7 When (Mi)=1 (PC) ← (PC)+3	X	X	X	X	X	X	i i i 1	0 1 1 1	(1+2)x10+7	3	5	4	
		BBS i, A, \$ hhl	When (Ai)=1 (PC) ← (PC)+2+Rel Where i=0-7 When (Ai)=0 (PC) ← (PC)+2	X	X	X	X	X	X	i i i 0	0 0 1 1	2ix10+3	2	4	4	
		BBS i, \$ zz, \$ hhl	When (Mi)=1 (PC) ← (PC)+3+Rel Where i=0-7 When (Mi)=0 (PC) ← (PC)+3	X	X	X	X	X	X	i i i 0	0 1 1 1	2ix10+7	3	5	4	
		BCC \$ hhl	When (C)=0 (PC) ← (PC)+2+Rel When (C)=1 (PC) ← (PC)+2	X	X	X	X	X	X	1 0 0 1	0 0 0 0	<B2>	90	2	2	4
		BCS \$ hhl	When (C)=1 (PC) ← (PC)+2+Rel When (C)=0 (PC) ← (PC)+2	X	X	X	X	X	X	1 0 1 1	0 0 0 0	<B2>	B0	2	2	4
		BNE \$ hhl	When (Z)=0 (PC) ← (PC)+2+Rel When (Z)=1 (PC) ← (PC)+2	X	X	X	X	X	X	1 1 0 1	0 0 0 0	<B2>	D0	2	2	4
		BEQ \$ hhl	When (Z)=1 (PC) ← (PC)+2+Rel When (Z)=0 (PC) ← (PC)+2	X	X	X	X	X	X	1 1 1 1	0 0 0 0	<B2>	F0	2	2	4
		BPL \$ hhl	When (N)=0 (PC) ← (PC)+2+Rel When (N)=1 (PC) ← (PC)+2	X	X	X	X	X	X	0 0 0 1	0 0 0 0	<B2>	10	2	2	4
		BMI \$ hhl	When (N)=1 (PC) ← (PC)+2+Rel When (N)=0 (PC) ← (PC)+2	X	X	X	X	X	X	0 0 1 1	0 0 0 0	<B2>	30	2	2	4
	Return	BVC \$ hhl	When (V)=0 (PC) ← (PC)+2+Rel When (V)=1 (PC) ← (PC)+2	X	X	X	X	X	X	0 1 0 1	0 0 0 0	<B2>	50	2	2	4
		BVS \$ hhl	When (V)=1 (PC) ← (PC)+2+Rel When (V)=0 (PC) ← (PC)+2	X	X	X	X	X	X	0 1 1 1	0 0 0 0	<B2>	70	2	2	4
	Interrupt	RTI	(S)←(S)+1, (PS)←(M(S)), (S)←(S)+1, (PCL)←(M(S)), (S)←(S)+1, and (PCH)←(M(S))	Previous status in stack					0 1 0 0	0 0 0 0	40	1	6			
		RTS	(S)←(S)+1, (PCL)←(M(S)), (S)←(S)+1, (PCH)←(M(S)), and (PC)←(PC)+1						0 1 1 0	0 0 0 0	60	1	6			
	Other	BRK	(B)←1, (PC)←(PC)+2, (M(S))←(PCH), (S)←(S)-1, (M(S))←(PCL), (S)←(S)-1, (M(S))←(PS), (S)←(S)-1, (I)←1, (PC)←BADRS	X	X	X	1	X	X	0 0 0 0	0 0 0 0	00	1	7		
		NOP	(PC) ← (PC)+1	X	X	X	X	X	X	1 1 1 0	1 0 1 0	EA	1	2		
	Special	WIT	Internal clock source is stopped.	X	X	X	X	X	X	1 1 0 0	0 0 1 0	C2	1	2		
		STP	Oscillation is stopped.	X	X	X	X	X	X	0 1 0 0	0 0 1 0	42	1	2	5	



## 740 Family Machine Language Instruction Table

Symbol	Means	Symbol	Means
A	Accumulator	hh	High-order byte of address (0—255)
Ai	Bit i of accumulator	ll	Low-order byte of address (0—255)
X	Index register X	zz	Zero page address (0—255)
Y	Index register Y	nn	Data at (0—255)
M	Memory	i	Data at (0—7)
Mi	Bit i of memory	iii	Data at (0—7)
PS	Processor status register	<B2>	Second byte of instruction
S	Stack Pointer	<B3>	Third byte of instruction
PC	Program counter	Rel	Relative address
PCL	Low-order byte of program counter	BADRS	Break address
PCH	High-order byte of program counter	←	Direction of data transfer
N	Negative flag	( )	Contents of register of memory
V	Overflow flag	+	Add
T	X modified operation mode flag	—	Subtract
B	Break flag	*	Multiplication
D	Decimal mode flag	÷	Division
I	Interrupt disable flag	∨	Logical OR
Z	Zero flag	^	Logical AND
C	Carry flag	∩	Logical Exclusive OR
#	Immediate mode	—	Negative
\$	Hexadecimal	X	Stable flag after execution
\	Special page mode	O	Variable flag after execution

Notes 1: Listed function is when (T) = 0.

When (T) = 1, (M(X)) is entered instead of (A) and the cycle number is increased by 3.

2: Ditto. The cycle number is increased by 2.

3: Ditto. The cycle number is increased by 1.

4: The cycle number is increased by 2 when a branch is occurred.

5: If the STP instruction is disabled the cycle number will be 2 (same in operation as two NOPs).

# APPENDIX 3

## 740 Family list of Instruction Codes

### APPENDIX 3. 740 Family list of Instruction Codes

D7 – D4	D3 – D0	Hexadecimal notation															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK	ORA IND, X	JSR ZP, IND	BBS 0, A	—	ORA ZP	ASL ZP	BBS 0, ZP	PHP	ORA IMM	ASL A	SEB 0, A	—	ORA ABS	ASL ABS	SEB 0, ZP
0001	1	BPL	ORA IND, Y	CLT	BBC 0, A	—	ORA ZP, X	ASL ZP, X	BBC 0, ZP	CLC	ORA ABS, Y	DEC A	CLB 0, A	—	ORA ABS, X	ASL ABS, X	CLB 0, ZP
0010	2	JSR ABS	AND IND, X	JSR SP	BBS 1, A	BIT ZP	AND ZP	ROL ZP	BBS 1, ZP	PLP	AND IMM	ROL A	SEB 1, A	BIT ABS	AND ABS	ROL ABS	SEB 1, ZP
0011	3	BMI	AND IND, Y	SET	BBC 1, A	—	AND ZP, X	ROL ZP, X	BBC 1, ZP	SEC	AND ABS, Y	INC A	CLB 1, A	LDM ZP	AND ABS, X	ROL ABS, X	CLB 1, ZP
0100	4	RTI	EOR IND, X	STP (Note)	BBS 2, A	COM ZP	EOR ZP	LSR ZP	BBS 2, ZP	PHA	EOR IMM	LSR A	SEB 2, A	JMP ABS	EOR ABS	LSR ABS	SEB 2, ZP
0101	5	BVC	EOR IND, Y	—	BBC 2, A	—	EOR ZP, X	LSR ZP, X	BBC 2, ZP	CLI	EOR ABS, Y	—	CLB 2, A	—	EOR ABS, X	LSR ABS, X	CLB 2, ZP
0110	6	RTS	ADC IND, X	MUL (Note)	BBS 3, A	TST ZP	ADC ZP	ROR ZP	BBS 3, ZP	PLA	ADC IMM	ROR A	SEB 3, A	JMP IND	ADC ABS	ROR ABS	SEB 3, ZP
0111	7	BVS	ADC IND, Y	—	BBC 3, A	—	ADC ZP, X	ROR ZP, X	BBC 3, ZP	SEI	ADC ABS, Y	—	CLB 3, A	—	ADC ABS, X	ROR ABS, X	CLB 3, ZP
1000	8	BRA	STA IND, X	RRF ZP	BBS 4, A	STY ZP	STA ZP	STX ZP	BBS 4, ZP	DEY	—	TXA	SEB 4, A	STY ABS	STA ABS	STX ABS	SEB 4, ZP
1001	9	BCC	STA IND, Y	—	BBC 4, A	STY ZP, X	STA ZP, X	STX ZP, Y	BBC 4, ZP	TYA	STA ABS, Y	TXS	CLB 4, A	—	STA ABS, X	—	CLB 4, ZP
1010	A	LDY IMM	LDA IND, X	LDX IMM	BBS 5, A	LDY ZP	LDA ZP	LDX ZP	BBS 5, ZP	TAY	LDA IMM	TAX	SEB 5, A	LDY ABS	LDA ABS	LDX ABS	SEB 5, ZP
1011	B	BCS	LDA IND, Y	JMP ZP, IND	BBC 5, A	LDY ZP, X	LDA ZP, X	LDX ZP, Y	BBC 5, ZP	CLV	LDA ABS, Y	TSX	CLB 5, A	LDY ABS, X	LDA ABS, X	LDX ABS, Y	CLB 5, ZP
1100	C	CPY IMM	CMP IND, X	WIT	BBS 6, A	CPY ZP	CMP ZP	DEC ZP	BBS 6, ZP	INY	CMP IMM	DEX	SEB 6, A	CPY ABS	CMP ABS	DEC ABS	SEB 6, ZP
1101	D	BNE	CMP IND, Y	—	BBC 6, A	—	CMP ZP, X	DEC ZP, X	BBC 6, ZP	CLD	CMP ABS, Y	—	CLB 6, A	—	CMP ABS, X	DEC ABS, X	CLB 6, ZP
1110	E	CPX IMM	SBC IND, X	DIV (Note)	BBS 7, A	CPX ZP	SBC ZP	INC ZP	BBS 7, ZP	INX	SBC IMM	NOP	SEB 7, A	CPX ABS	SBC ABS	INC ABS	SEB 7, ZP
1111	F	BEQ	SBC IND, Y	—	BBC 7, A	—	SBC ZP, X	INC ZP, X	BBC 7, ZP	SED	SBC ABS, Y	—	CLB 7, A	—	SBC ABS, X	INC ABS, X	CLB 7, ZP

Note: Some products unuse these instructions.

Instruction	Products which unuse these instructions
STP	M37424, M37524
MUL DIV	M507XX, M509XX, M37408, M37409, M37410, M37412, M37413, M37414, M37415, M37416, M37417, M37418, M37420, M37421

- 3-byte instruction
- 2-byte instruction
- 1-byte instruction

Refer to the related section because the clock control instruction and multiplication and division instruction depend on products.

*MEMORANDUM*

**mitsubishi semiconductors**  
**software manual**  
**740 Family**

---

Sep. First Edition 1997

Edited by  
Committee of editing of Mitsubishi Semiconductor USER'S MANUAL

Published by  
Mitsubishi Electric Corp., Semiconductor Marketing Division

---

This book, or parts thereof, may not be reproduced in any form without permission of Mitsubishi Electric Corporation.

**©1997 MITSUBISHI ELECTRIC CORPORATION**

# Software Manual

## 740 Family

**Renesas Technology Corp.**

Nippon Bldg., 6-2, Otemachi 2-chome, Chiyoda-ku, Tokyo, 100-0004 Japan