USER'S MANUAL

NEC

78K/0 SERIES

8-BIT SINGLE-CHIP MICROCONTROLLERS

INSTRUCTIONS

FOR ALL 78K/0 SERIES

Document No. U12326EJ3V0UM00 (3rd edition)
(O. D. No. IEU-849)
Date Published June 1997 N
Printed in Japan

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note: Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note: No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS device behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note: Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

FIP and IEBus are trademarks of NEC Corporation.

Caution: The following products are provided with an I²C bus interface circuit:
μPD78002Y Subseries, μPD78014Y Subseries, μPD78018FY Subseries
μPD78054Y Subseries, μPD78058FY Subseries, μPD78064Y Subseries
μPD78075BY Subseries, μPD78078Y Subseries, μPD780018Y Subseries
μPD780024Y Subseries, μPD780034Y Subseries, μPD780058Y Subseries
μPD780308Y Subseries, μPD78070AY

Purchase of NEC I²C components conveys a license under the Philips I²C Patent Rights to use these components in an I²C system, provided that the system conforms to the I²C Standard Specification as defined by Philips.

The export of these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

M7 96.5

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

Major Revisions in This Edition

Page	Contents
Throughout	Added products, deleted products Added products : μPD78014H, 78018FY, 78044F, 78044H, 78058F, 78058FY, 78064Y, 78064B, 78075B, 78075BY, 78078Y, 78098B, 780018Y, 780024, 780024Y, 780034, 780034Y, 780058, 780058Y, 780228, 780308, 780308Y, 780924, 780964 Subseries and μPD78011F, 78012F, 78070A, 78070AY, 780001, 78P0914, 780206, 780208 Deleted products: μPD78024, 78044, 78044A Subseries
p.11	Add **Table 1-13 Internal RAM Space of 78K/0 Series Products**
p.17	Change **Table 1-14 External Memory Space of 78K/0 Series Products**

The mark ★ shows major revised points.

INTRODUCTION

Intended Readership

This manual has been prepared for user engineers who want to understand the functions of the 78K/0 Series products and design and develop its application systems and programs.

78K/0 Series products

- μPD78002 Subseries : μPD78001B, 78002B
- μPD78002Y Subseries : μPD78001BY, 78002BY
- μPD78014 Subseries : μPD78011B, 78012B, 78013, 78014, 78P014
- μPD78014Y Subseries : μPD78011BY, 78012BY, 78013Y, 78014Y, 78P014Y
- μPD78014H Subseries : μPD78011H, 78012H, 78013H, 78014H
- μPD78018F Subseries : μPD78011F, 78012F, 78013F, 78014F, 78015F, 78016F, 78018F, 78P018F
- μPD78018FY Subseries : μPD78011FY, 78012FY, 78013FY, 78014FY, 78015FY, 78016FY, 78018FY, 78P018FY
- μPD78044F Subseries : μPD78042F, 78043F, 78044F, 78045F, 78P048A
- μPD78044H Subseries : μPD78044H, 78045H, 78046H, 78P048B
- μPD78054 Subseries : μPD78052, 78053, 78054, 78P054, 78055, 78056, 78058, 78P058
- μPD78054Y Subseries : μPD78052Y, 78053Y, 78054Y, 78P054Y, 78055Y, 78056Y, 78058Y, 78P058Y
- μPD78058F Subseries : μPD78056F, 78058F, 78P058F
- μPD78058FY Subseries : μPD78056FY, 78058FY, 78P058FY
- μPD78064 Subseries : μPD78062, 78063, 78064, 78P064
- μPD78064Y Subseries : μPD78062Y, 78063Y, 78064Y, 78P064Y
- μPD78064B Subseries : μPD78064B, 78P064B
- μPD78070A : μPD78070A
- μPD78070AY : μPD78070AY
- μPD78075B Subseries : μPD78074B, 78075B
- μPD78075BY Subseries : μPD78074BY, 78075BY
- μPD78078 Subseries : μPD78076, 78078, 78P078
- μPD78078Y Subseries : μPD78076Y, 78078Y, 78P078Y
- μPD78083 Subseries : μPD78081, 78082, 78P083
- μPD78098 Subseries : μPD78094, 78095, 78096, 78098A, 78P098A
- μPD78098B Subseries[Note] : μPD78095B, 78096B, 78098B, 78P098B
- μPD780001 : μPD78001
- μPD780018Y Subseries[Note] : μPD780016Y, 780018Y, 78P0018Y
- μPD780024 Subseries[Note] : μPD780021, 780022, 780023, 780024
- μPD780024Y Subseries[Note] : μPD780021Y, 780022Y, 780023Y, 780024Y
- μPD780034 Subseries[Note] : μPD780031, 780032, 780033, 780034, 78F0034
- μPD780034Y Subseries[Note] : μPD780031Y, 780032Y, 780033Y, 780034Y, 78F0034Y
- μPD780058 Subseries[Note] : μPD780053, 780054, 780055, 780056, 780058, 78F0058

- μPD780058Y Subseries[Note] : μPD780053Y, 780054Y, 780055Y, 780056Y, 780058Y, 78F0058Y
- μPD780208 Subseries[Note] : μPD780204, 780205, 780206, 780208, 78P0208
- μPD780228 Subseries[Note] : μPD780226, 780228, 78F0228
- μPD780308 Subseries[Note] : μPD780306, 780308, 78P0308
- μPD780308Y Subseries[Note] : μPD780306Y, 780308Y, 78P0308Y
- μPD78P0914[Note] : μPD78P0914
- μPD780924 Subseries[Note] : μPD780921, 780922, 780923, 780924, 78F0924
- μPD780964 Subseries[Note] : μPD780961, 780962, 780963, 780964, 78F0964

Note Under development

Purpose

This manual is intended for the users to understand the various kinds of instruction functions of 78K/0 Series products.

Organization

This manual consists of the following contents.

- CPU functions
- Instruction set
- Explanation of instructions

How to Read This Manual

Before reading this manual, you must have general knowledge of electric and logic circuits and microcontroller.

- To check the details of the functions of an instruction whose mnemonic is known:
  → Refer to **APPENDICES B** and **C INSTRUCTION INDEX**.

- To check an instruction whose mnemonic is not known and whose general function is known:
  → Check the mnemonic in **CHAPTER 4 INSTRUCTION SET** and then the functions in **CHAPTER 5 EXPLANATION OF INSTRUCTIONS**.

- To learn about various kinds of 78K/0 Series products instructions in general:
  → Read this manual in the order of contents.

- To learn about the hardware functions of 78K/0 Series products:
  → See the separate User's Manual (refer to **Related Documents**).

Legend

Data representation weight	: High digits on the left and low digits on the right
Note	: Description of Note in the text
Caution	: Information requiring particular attention
Remark	: Additional explanatory material
Numeral representations	: Binary ..... XXXX or XXXXB
	Decimal XXXX
	Hexadecimal ... XXXXH

★ **Related Documents**

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

• Documents Common for 78K/0 Series

Document Name		Japanese Document Number	English Document Number
User's Manual Instructions		U12326J	This manual
Application Note[Note]	Basic I	IEA-715	IEA-1288
	Basic II	U10121J	U10121E
	Basic III	IEA-767	U10182E
	Floating-point Operation Program	IEA-718	IEA-1289
Selection Guide		U11126J	U11126E
Instruction Table		C10903J	—
Instruction Set		C10904J	—

Note Some Subseries may not be covered.

• Individual Documents

μPD78002, 78002Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD78001B, 78002B Data Sheet	U10674J	U10674E
μPD78001BY, 78002BY Data Sheet	IC-8571	IC-3173
User's Manual	U10039J	U10039E
Special Function Register Table	IEM-5556	—

μPD78014 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78011B, 78012B, 78013, 78014 Data Sheet	IC-8201	IC-3179
μPD78P014 Data Sheet	IC-8111	IC-3098
User's Manual	U10085J	U10085E
Special Function Register Table	IEM-5527	—

μPD78014Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD78011BY, 78012BY, 78013Y, 78014Y Data Sheet	IC-8573	IC-3405
μPD78P014Y Data Sheet	IC-8572	IC-3180
User's Manual	U10085J	U10085E
Special Function Register Table	IEM-5527	—

μPD78014H Subseries

Document Name	Japanese Document Number	English Document Number
μPD78011H, 78012H, 78013H, 78014H Data Sheet	U11898J	U11898E
User's Manual	U12220J	U12220E
Special Function Register Table	To be prepared	–

μPD78018F Subseries

Document Name	Japanese Document Number	English Document Number
μPD78011F, 78012F, 78013F, 78014F, 78015F, 78016F Data Sheet	U10280J	U10280E
μPD78P018F Data Sheet	U10955J	U10955E
User's Manual	U10659J	U10659E
Special Function Register Table	IEM-5594	—

μPD78018FY Subseries

Document Name	Japanese Document Number	English Document Number
μPD78011FY, 78012FY, 78013FY, 78014FY, 78015FY, 78016FY Data Sheet	U10280J	U10280E
μPD78P018FY Data Sheet	U10989J	U10989E
User's Manual	U10659J	U10659E
Special Function Register Table	U10287J	—

μPD78044F Subseries

Document Name	Japanese Document Number	English Document Number
μPD78042F, 78043F, 78044F, 78045F Data Sheet	U10700J	U10700E
μPD78P048A Data Sheet	U10611J	U10611E
User's Manual	U10908J	U10908E
Special Function Register Table	U10701J	—

μPD78044H Subseries

Document Name	Japanese Document Number	English Document Number
μPD78044H, 78045H, 78046H Data Sheet	U10865J	U10865E
μPD78P048B Data Sheet	To be prepared	To be prepared
User's Manual	U11756J	U11756E

μPD78054 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78052, 78053, 78054, 78055, 78056, 78058 Data Sheet	IC-8631	IC-3403
μPD78P054 Data Sheet	IC-8635	IC-3216
μPD78P058 Data Sheet	IC-8884	U10417E
User's Manual	U11747J	U11747E
Special Function Register Table	U10102J	—

μPD78054Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD78052Y, 78053Y, 78054Y, 78055Y, 78056Y, 78058Y Data Sheet	U10906J	U10906E
μPD78P054Y Preliminary Product Information	IP-8719	IP-3205
μPD78P058Y Data Sheet	U10907J	U10907E
User's Manual	U11747J	U11747E
Special Function Register Table	U10087J	—

μPD78058F Subseries

Document Name	Japanese Document Number	English Document Number
μPD78056F, 78058F Data Sheet	U11795J	U11795E
μPD78P058F Data Sheet	U11796J	U11796E
User's Manual	U12068J	U12068E
Special Function Register Table	To be prepared	—

μPD78058FY Subseries

Document Name	Japanese Document Number	English Document Number
μPD78056FY, 78058FY Data Sheet	U12325J	U12325E
μPD78P058FY Data Sheet	U12076J	U12076E
User's Manual	U12068J	U12068E
Special Function Register Table	To be prepared	—

μPD78064 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78062, 78063, 78064 Data Sheet	IC-8632	IC-3244
μPD78P064 Data Sheet	IC-8636	IC-3224
User's Manual	U10105J	U10105E
Special Function Register Table	IEM-5568	—

μPD78064Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD78062Y, 78063Y, 78064Y Data Sheet	IC-8704	U10337E
μPD78P064Y Data Sheet	U10321J	IP-3236
User's Manual	U10105J	U10105E
Special Function Register Table	IEM-5583	—

μPD78064B Subseries

Document Name	Japanese Document Number	English Document Number
μPD78064B Data Sheet	U11590J	U11590E
μPD78P064B Data Sheet	U11598J	U11598E
User's Manual	U10785J	U10785E
Special Function Register Table	To be prepared	—

μPD78070A

Document Name	Japanese Document Number	English Document Number
μPD78070A Data Sheet	U10326J	U10326E
User's Manual	IEU-907	U10200E
Special Function Register Table	U10133J	—

μPD78070AY

Document Name	Japanese Document Number	English Document Number
μPD78070AY Data Sheet	U10542J	U10542E
User's Manual	IEU-907	U10200E
Special Function Register Table	U10134J	—

μPD78075B Subseries

Document Name	Japanese Document Number	English Document Number
μPD78074B, 78075B Data Sheet	U12017J	U12017E
User's Manual	To be prepared	To be prepared
Special Function Register Table	To be prepared	—

μPD78075BY Subseries

Document Name	Japanese Document Number	English Document Number
μPD78074BY, 78075BY Data Sheet	To be prepared	To be prepared
User's Manual	To be prepared	To be prepared
Special Function Register Table	To be prepared	—

μPD78078 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78076, 78078 Data Sheet	U10167J	U10167E
μPD78P078 Data Sheet	U10168J	U10168E
User's Manual	U10641J	U10641E
Special Function Register Table	IEM-5607	—

μPD78078Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD78076Y, 78078Y Data Sheet	U10605J	U10605E
μPD78P078Y Data Sheet	U10606J	U10606E
User's Manual	U10641J	U10641E
Special Function Register Table	U10257J	—

μPD78083 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78081, 78082 Data Sheet	U11415J	U11415E
μPD78P083 Data Sheet	U11006J	U11006E
User's Manual	U12176J	U12176E
Special Function Register Table	IEM-5599	—

μPD78098 Subseries

Document Name	Japanese Document Number	English Document Number
μPD78094, 78095, 78096, 78098A Data Sheet	U10146J	U10146E
μPD78P098A Data Sheet	U10203J	U10203E
User's Manual	IEU-854	IEU-1381
Special Function Register Table	IEM-5591	—

μPD780208 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780204, 780205, 780206, 780208 Data Sheet	U10436J	IP-3540
μPD78P0208 Data Sheet	U11295J	IP-3475
User's Manual	U11302J	U11302E
Special Function Register Table	U10997J	—

μPD780001

Document Name	Japanese Document Number	English Document Number
μPD780001 Data Sheet	U10324J	U10324E
User's Manual	U10885J	U10885E
Special Function Register Table	To be prepared	—

μPD780018Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD780016Y, 780018Y Preliminary Product Information	U11810J	U11810E
μPD78P0018Y Preliminary Product Information	U11606J	U11606E
User's Manual	U11754J	U11754E
Special Function Register Table	To be prepared	—

μPD780024 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780021, 780022, 780023, 780024 Preliminary Product Information	U12299J	U12299E
User's Manual	U12022J	U12022E
Special Function Register Table	To be prepared	—

μPD780024Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD780021Y, 780022Y, 780023Y, 780024Y Preliminary Product Information	U12165J	U12165E
User's Manual	U12022J	U12022E
Special Function Register Table	To be prepared	—

μPD780034 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780031, 780032, 780033, 780034 Preliminary Product Information	U12300J	U12300E
μPD78F0034 Preliminary Product Information	U11993J	U11993E
User's Manual	U12022J	U12022E
Special Function Register Table	To be prepared	—

μPD780034Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD780031Y, 780032Y, 780033Y, 780034Y Preliminary Product Information	U12166J	U12166E
μPD78F0034Y Preliminary Product Information	U11994J	U11994E
User's Manual	U12022J	U12022E
Special Function Register Table	To be prepared	—

μPD780058 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780053, 780054, 780055, 780056, 780058 Preliminary Product Information	U12182J	U12182E
μPD78F0058 Preliminary Product Information	U12092J	U12092E
User's Manual	U12013J	U12013E
Special Function Register Table	To be prepared	—

μPD780058Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD780053Y, 780054Y, 780055Y, 780056Y, 780058Y Preliminary Product Information	To be prepared	To be prepared
μPD78F0058Y Preliminary Product Information	U12324J	U12324E
User's Manual	U12013J	U12013E
Special Function Register Table	To be prepared	—

μPD780228 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780226, 780228 Preliminary Product Information	U11797J	U11797E
μPD78F0228 Preliminary Product Information	U11971J	U11971E
User's Manual	U12012J	U12012E
Special Function Register Table	To be prepared	—

μPD780308 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780306, 780308 Preliminary Product Information	U12183J	U12183E
μPD78P0308 Preliminary Product Information	U11776J	U11776E
User's Manual	U11377J	U11377E
Special Function Register Table	To be prepared	—

μPD780308Y Subseries

Document Name	Japanese Document Number	English Document Number
μPD780306Y, 780308Y Preliminary Product Information	To be prepared	To be prepared
μPD78P0308Y Preliminary Product Information	U11832J	U11832E
User's Manual	U11377J	U11377E
Special Function Register Table	To be prepared	—

μPD78P0914

Document Name	Japanese Document Number	English Document Number
μPD78P0914 Preliminary Product Information	U10058J	U10058E
User's Manual	To be prepared	To be prepared
Special Function Register Table	U10866J	—

μPD780924 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780921, 780922, 780923, 780924 Preliminary Product Information	U11804J	U11804E
μPD78F0924 Preliminary Product Information	U11930J	U11930E
User's Manual	U12071J	U12071E
Special Function Register Table	U12230J	—

μPD780964 Subseries

Document Name	Japanese Document Number	English Document Number
μPD780961, 780962, 780963, 780964 Preliminary Product Information	U11879J	U11879E
μPD78F0964 Preliminary Product Information	U11956J	U11956E
User's Manual	U12071J	U12071E
Special Function Register Table	U12230J	—

Caution The above related documents are subject to change without notice. Be sure to use the latest version when designing your system.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

[MEMO]

CHAPTER 1 MEMORY SPACE

1.1 Memory Spaces

The 78K/0 Series product program memory map varies depending on the internal memory capacity. For details of memory mapped address area, refer to **each product User's Manual**.

1.2 Internal Program Memory (Internal ROM) Space

Each 78K/0 Series product has internal ROM in the address space shown below. Program and table data, etc. are stored in ROM. Normally, this memory space is addressed by the program counter (PC).

★ **Table 1-1. Internal ROM Space of 78K/0 Series Products (1/3)**

Capacity / Address space / Subseries name	8 Kbytes	16 Kbytes	24 Kbytes	32 Kbytes	40 Kbytes	48 Kbytes	60 Kbytes
	0000H-1FFFH	0000H-3FFFH	0000H-5FFFH	0000H-7FFFH	0000H-9FFFH	0000H-BFFFH	0000H-EFFFH
μPD78002 Subseries	μPD78001B	μPD78002B					
μPD78002Y Subseries	μPD78001BY	μPD78002BY					
μPD78014 Subseries	μPD78011B	μPD78012B	μPD78013	μPD78014, μPD78P014			
μPD78014Y Subseries	μPD78011BY	μPD78012BY	μPD78013Y	μPD78014Y, μPD78P014Y			
μPD78014H Subseries	μPD78011H	μPD78012H	μPD78013H	μPD78014H			
μPD78018F Subseries	μPD78011F	μPD78012F	μPD78013F	μPD78014F	μPD78015F	μPD78016F	μPD78018F, μPD78P018F
μPD78018FY Subseries	μPD78011FY	μPD78012FY	μPD78013FY	μPD78014FY	μPD78015FY	μPD78016FY	μPD78018FY, μPD78P018FY
μPD78044F Subseries		μPD78042F	μPD78043F	μPD78044F	μPD78045F		μPD78P048A
μPD78044H Subseries				μPD78044H	μPD78045H	μPD78046H	μPD78P048B
μPD78054 Subseries		μPD78052	μPD78053	μPD78054, μPD78P054	μPD78055	μPD78056	μPD78058, μPD78P058
μPD78054Y Subseries		μPD78052Y	μPD78053Y	μPD78054Y, μPD78P054Y	μPD78055Y	μPD78056Y	μPD78058Y, μPD78P058Y
μPD78058F Subseries						μPD78056F	μPD78058F, μPD78P058F
μPD78058FY Subseries						μPD78056FY	μPD78058FY, μPD78P058FY
μPD78064 Subseries		μPD78062	μPD78063	μPD78064, μPD78P064			
μPD78064Y Subseries		μPD78062Y	μPD78063Y	μPD78064Y, μPD78P064Y			
μPD78064B Subseries				μPD78064B, μPD78P064B			

Remarks 1. The μPD78070A and 78070AY do not incorporate ROMs.

2. The internal ROM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS).

★ **Table 1-1. Internal ROM Space of 78K/0 Series Products (2/3)**

Capacity / Address space / Subseries name	8 Kbytes	16 Kbytes	24 Kbytes	32 Kbytes	40 Kbytes	48 Kbytes	60 Kbytes
	0000H-1FFFH	0000H-3FFFH	0000H-5FFFH	0000H-7FFFH	0000H-9FFFH	0000H-BFFFH	0000H-EFFFH
μPD78075B Subseries				μPD78074B	μPD78075B		
μPD78075BY Subseries				μPD78074BY	μPD78075BY		
μPD78078 Subseries						μPD78076	μPD78078, μPD78P078
μPD78078Y Subseries						μPD78076Y	μPD78078Y, μPD78P078Y
μPD78083 Subseries	μPD78081	μPD78082	μPD78P083				
μPD78098 Subseries				μPD78094	μPD78095	μPD78096	μPD78098A, μPD78P098A
μPD78098B Subseries					μPD78095B	μPD78096B	μPD78098B, μPD78P098B
μPD780001	μPD780001						
μPD780018Y Subseries						μPD780016Y	μPD780018Y, μPD78P0018Y
μPD780024 Subseries	μPD780021	μPD780022	μPD780023	μPD780024			
μPD780024Y Subseries	μPD780021Y	μPD780022Y	μPD780023Y	μPD780024Y			
μPD780034 Subseries	μPD780031	μPD780032	μPD780033	μPD780034, μPD78F0034			
μPD780034Y Subseries	μPD780031Y	μPD780032Y	μPD780033Y	μPD780034Y, μPD78F0034Y			
μPD780058 Subseries			μPD780053	μPD780054	μPD780055	μPD780056	μPD780058, μPD78F0058
μPD780058Y Subseries			μPD780053Y	μPD780054Y	μPD780055Y	μPD780056Y	μPD780058Y, μPD78F0058Y
μPD780208 Subseries				μPD780204	μPD780205	μPD780206	μPD780208, μPD78P0208

Remark The internal ROM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS).

★ **Table 1-1. Internal ROM Space of 78K/0 Series Products (3/3)**

Capacity / Address space / Subseries name	8 Kbytes	16 Kbytes	24 Kbytes	32 Kbytes	40 Kbytes	48 Kbytes	60 Kbytes
	0000H-1FFFH	0000H-3FFFH	0000H-5FFFH	0000H-7FFFH	0000H-9FFFH	0000H-BFFFH	0000H-EFFFH
μPD780228 Subseries						μPD780226	μPD780228, μPD78F0228
μPD780308 Subseries						μPD780306	μPD780308, μPD78P0308
μPD780308Y Subseries						μPD780306Y	μPD780308Y, μPD78P0308Y
μPD78P0914				μPD78P0914			
μPD780924 Subseries	μPD780921	μPD780922	μPD780923	μPD780924, μPD78F0924			
μPD780964 Subseries	μPD780961	μPD780962	μPD780963	μPD780964, μPD78F0964			

Remark The internal ROM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS).

1.3 Vector Table Area

The 64-byte area 0000H to 003FH is reserved as a vector table area. The $\overline{\text{RESET}}$ input and program start addresses for branch upon generation of each interrupt request are stored in the vector table area. Of the 16-bit address, low-order 8 bits are stored at even addresses and high-order 8 bits are stored at odd addresses.

★ **Table 1-2. Vector Table (μPD78002, 78002Y, 78014, 78014Y, 78014H, 78018F, 78018FY Subseries and μPD780001)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0010H	INTCSI1 Note 2
0004H	INTWDT	0012H	INTTM3
0006H	INTP0 Note 1	0014H	INTTM0 Note 1, 2
0008H	INTP1	0016H	INTTM1
000AH	INTP2	0018H	INTTM2
000CH	INTP3	001AH	INTAD Note 2
000EH	INTCSI0 Note 1	003EH	BRK instruction

Notes 1. Excluding μPD780001

2. Excluding μPD78002 and 78002Y Subseries

★ **Table 1-3. Vector Table (μPD78044F, 78044H, 780208 Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0010H	INTCSI1
0004H	INTWDT	0012H	INTTM3
0006H	INTP0	0014H	INTTM0
0008H	INTP1	0016H	INTTM1
000AH	INTP2	0018H	INTTM2
000CH	INTP3/INTUD Note	001AH	INTAD
000EH	INTCSI0 Note	001CH	INTKS
		003EH	BRK instruction

Note The μPD78044H Subseries contain neither INTUD nor INTCSI0.

★ **Table 1-4. Vector Table (μPD78054, 78054Y, 78058F, 78058FY, 78075B, 78075BY, 78078, 78078Y Subseries and μPD78070A, 78070AY)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	001AH	INTSR/INTCSI2
0004H	INTWDT	001CH	INTST
0006H	INTP0	001EH	INTTM3
0008H	INTP1	0020H	INTTM00
000AH	INTP2	0022H	INTTM01
000CH	INTP3	0024H	INTTM1
000EH	INTP4	0026H	INTTM2
0010H	INTP5	0028H	INTAD
0012H	INTP6	002AH	INTTM5[Note]
0014H	INTCSI0	002CH	INTTM6[Note]
0016H	INTCSI1	003EH	BRK instruction
0018H	INTSER		

Note Only the μPD78075B, 78075BY, 78078, and 78078Y Subseries

★ **Table 1-5. Vector Table (μPD78064, 78064Y, 78064B, 780308, 780308Y, 780058, 780058Y Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	001AH	INTSR/INTCSI2
0004H	INTWDT	001CH	INTST
0006H	INTP0	001EH	INTTM3
0008H	INTP1	0020H	INTTM00
000AH	INTP2	0022H	INTTM01
000CH	INTP3	0024H	INTTM1
000EH	INTP4	0026H	INTTM2
0010H	INTP5	0028H	INTAD
0014H	INTCSI0	002AH	INTCSI1[Note 2]
0016H	INTCSI1[Note 1]	003EH	BRK instruction
0018H	INTSER		

Notes 1. Only the μPD780058 and 780058Y Subseries

2. Only the μPD780308 and 780308Y Subseries

Table 1-6. Vector Table (μPD78083 Subseries)

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	001AH	INTSR/INTCSI2
0004H	INTWDT	001CH	INTST
0008H	INTP1	0028H	INTAD
000AH	INTP2	002AH	INTTM5
000CH	INTP3	002CH	INTTM6
0018H	INTSER	003EH	BRK instruction

★ **Table 1-7. Vector Table (μPD78098, 78098B Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0018H	INTSER
0004H	INTWDT	001AH	INTSR/INTCSI2
0006H	INTP0	001CH	INTST
0008H	INTP1	001EH	INTTM3
000AH	INTP2	0020H	INTTM00
000CH	INTP3	0022H	INTTM01
000EH	INTP4	0024H	INTTM1
0010H	INTP5	0026H	INTTM2
0012H	INTP6	0028H	INTAD
0014H	INTCSI0	002AH	INTIE
0016H	INTCSI1	003EH	BRK instruction

★ **Table 1-8. Vector Table (μPD780018Y Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0020H	INTTM00
0004H	INTWDT	0022H	INTTM01
0006H	INTP0	0024H	INTTM1
0008H	INTP1	0026H	INTTM2
000AH	INTP2	0028H	INTAD
000CH	INTP3	002AH	INTTM5
000EH	INTP4	002CH	INTTM6
0010H	INTP5	002EH	INTCSI4
0012H	INTP6	0030H	INTIIC
0016H	INTCSI1	003EH	BRK instruction
001EH	INTTM3		

★ **Table 1-9. Vector Table (μPD780024, 780024Y, 780034, 780034Y Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0018H	INTIIC0 Note2
0004H	INTWDT	001AH	INTWTI
0006H	INTP0	001CH	INTTM00
0008H	INTP1	001EH	INTTM01
000AH	INTP2	0020H	INTTM50
000CH	INTP3	0022H	INTTM51
000EH	INTSER0	0024H	INTAD0
0010H	INTSR0	0026H	INTWT
0012H	INTST0	0028H	INTKR
0014H	INTCSI30	003EH	BRK instruction
0016H	INTCSI31 Note 1		

Notes 1. Only the μPD780024 and 780034 Subseries

2. Only the μPD780024Y and 780034Y Subseries

★ **Table 1-10. Vector Table (μPD780228 Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	000EH	INTKS
0004H	INTWDT	0010H	INTCSI3
0006H	INTP0	0012H	INTTM50
0008H	INTP1	0014H	INTTM51
000AH	INTTM10	0016H	INTAD
000CH	INTTM11	003EH	BRK instruction

★ **Table 1-11. Vector Table (μPD78P0914)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0012H	INTHS
0004H	INTWDT	0014H	INTTM6
0006H	INTVS	0016H	INTTM7
0008H	INTP1	0018H	INTTM8
000AH	INTP2	001AH	INTCSI0
000CH	INTTM5	001CH	INTCSI1
000EH	INTTM1	001EH	INTAD
0010H	INTTM2	003EH	BRK instruction

★ **Table 1-12. Vector Table (μPD780924, 780964 Subseries)**

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	0014H	INTST0
0004H	INTWDT	0016H	INTSER1
0006H	INTP0	0018H	INTSR1
0008H	INTP1	001AH	INTST1
000AH	INTP2	001CH	INTTM50
000CH	INTP3	001EH	INTTM51
000EH	INTTM7	0020H	INTTM52
0010H	INTSER0	0022H	INTAD0
0012H	INTSR0	003EH	BRK instruction

1.4 CALLT Instruction Table Area

The 64-byte area 0040H to 007FH can store the subroutine entry address of a 1-byte call instruction (CALLT).

1.5 CALLF Instruction Entry Area

The 2048-byte area 0800H to 0FFFH can perform a direct subroutine call with a 2-byte call instruction (CALLF).

1.6 Internal Data Memory (Internal RAM) Space

The 78K/0 Series products incorporate the following RAMs. For the RAMs incorporated in each product, see **Table 1-13 Internal RAM Space of 78K/0 Series Products**.

(1) Internal high-speed RAM

Each 78K/0 Series product incorporates an internal high-speed RAM in the address space shown in Table 1-13. In the 32-byte area FEE0H to FEFFH of these areas, 4 banks of general registers, each bank consisting of eight 8-bit registers, are allocated.

The internal high-speed RAM can also be used as a stack memory.

(2) Buffer RAM

A buffer RAM is allocated to each area shown in Table 1-13. This RAM is used to store the transfer/receive data of the serial interface channel 1 (3-wired serial I/O mode with automatic transfer/receive function). If not used in this mode, the buffer RAM can also be used as an ordinary RAM area.

(3) RAM for FIP™ display

A RAM for FIP display is allocated to each area shown in Table 1-13. This RAM can also be used as a normal RAM.

(4) Internal expansion RAM

An internal expansion RAM is allocated to each area shown in Table 1-13.

(5) RAM for LCD display

A RAM for LCD display is allocated to each area shown in Table 1-13. This RAM can also be used as a normal RAM.

★ **Table 1-13. Internal RAM Space of 78K/0 Series Products (1/5)**

Subseries Name	Products	High-speed RAM	Buffer RAM	Extended RAM	RAM for FIP Display	RAM for LCD Display
μPD78002 Subseries	μPD78001B	FE00H to FEFFH (256 bytes)	–	–	–	–
	μPD78002B	FD80H to FEFFH (384 bytes)				
μPD78002Y Subseries	μPD78001BY	FE00H to FEFFH (256 bytes)	–	–	–	–
	μPD78002BY	FD80H to FEFFH (384 bytes)				
μPD78014 Subseries	μPD78011B	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78012B					
	μPD78013	FB00H to FEFFH (1024 bytes)				
	μPD78014					
	μPD78P014					
μPD78014Y Subseries	μPD78011BY	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78012BY					
	μPD78013Y	FB00H to FEFFH (1024 bytes)				
	μPD78014Y					
	μPD78P014Y					
μPD78014H Subseries	μPD78011H	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78012H					
	μPD78013H	FB00H to FEFFH (1024 bytes)				
	μPD78014H					
μPD78018F Subseries	μPD78011F	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78012F					
	μPD78013F	FB00H to FEFFH (1024 bytes)				
	μPD78014F					
	μPD78015F			F600H to F7FFH (512 bytes)		
	μPD78016F					
	μPD78018F			F400H to F7FFH (1024 bytes)		
	μPD78P018F					
μPD78018FY Subseries	μPD78011FY	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78012FY					
	μPD78013FY	FB00H to FEFFH (1024 bytes)				
	μPD78014FY					
	μPD78015FY			F600H to F7FFH (512 bytes)		
	μPD78016FY					
	μPD78018FY			F400H to F7FFH (1024 bytes)		
	μPD78P018FY					

Remark The internal high-speed RAM capacity and expansion RAM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS) and internal expansion RAM size switching register (IXS).

★ **Table 1-13. Internal RAM Space of 78K/0 Series Products (2/5)**

Subseries Name	Products	High-speed RAM	Buffer RAM	Extended RAM	RAM for FIP Display	RAM for LCD Display
μPD78044F Subseries	μPD78042F	FD00H to FEFFH (512 bytes)	FAC0H to FAFFH (64 bytes)	–	FA50H to FA7FH (48 bytes)	–
	μPD78043F					
	μPD78044F	FB00H to FEFFH (1024 bytes)				
	μPD78045F					
	μPD78P048A			F400H to F7FFH (1024 bytes)		
μPD78044H Subseries	μPD78044H	FB00H to FEFFH (1024 bytes)	–	–	FA50H to FA7FH (48 bytes)	–
	μPD78045H					
	μPD78046H					
	μPD78P048B			F400H to F7FFH (1024 bytes)		
μPD78054 Subseries	μPD78052	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78053	FB00H to FEFFH (1024 bytes)				
	μPD78054					
	μPD78P054					
	μPD78055					
	μPD78056					
	μPD78058			F400H to F7FFH (1024 bytes)		
	μPD78P058					
μPD78054Y Subseries	μPD78052Y	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78053Y					
	μPD78054Y	FB00H to FEFFH (1024 bytes)				
	μPD78055Y					
	μPD78056Y					
	μPD78058Y			F400H to F7FFH (1024 bytes)		
	μPD78P058Y					
μPD78058F Subseries	μPD78056F	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78058F					
	μPD78P058F			F400H to F7FFH (1024 bytes)		
μPD78058FY Subseries	μPD78056FY	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78058FY					
	μPD78P058FY			F400H to F7FFH (1024 bytes)		

Remark The internal high-speed RAM capacity and expansion RAM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS) and internal expansion RAM size switching register (IXS).

★ **Table 1-13. Internal RAM Space of 78K/0 Series Products (3/5)**

Subseries Name	Products	High-speed RAM	Buffer RAM	Extended RAM	RAM for FIP Display	RAM for LCD Display
μPD78064 Subseries	μPD78062	FD00H to FEFFH (512 bytes)	–	–	–	FA58H to FA7FH (40 x 4 bits)
	μPD78063	FB00H to FEFFH (1024 bytes)				
	μPD78064					
	μPD78P064					
μPD78064Y Subseries	μPD78062Y	FD00H to FEFFH (512 bytes)	–	–	–	FA58H to FA7FH (40 x 4 bits)
	μPD78063Y	FB00H to FEFFH (1024 bytes)				
	μPD78064Y					
	μPD78P064Y					
μPD78064B Subseries	μPD78064B	FB00H to FEFFH (1024 bytes)	–	–	–	FA58H to FA7FH (40 x 4 bits)
	μPD78P064B					
μPD78070A	μPD78070A	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
μPD78070AY	μPD78070AY					
μPD78075B Subseries	μPD78074B	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytres)	–	–	–
	μPD78075B					
μPD78075BY Subseries	μPD78074BY	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78075BY					
μPD78078 Subseries	μPD78076	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	F400H to F7FFH (1024 bytes)	–	–
	μPD78078					
	μPD78P078					
μPD78078Y Subseries	μPD78076Y	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	F400H to F7FFH (1024 bytes)	–	–
	μPD78078Y					
	μPD78P078Y					
μPD78083 Subseries	μPD78081	FE00H to FEFFH (256 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78082	FD80H to FEFFH (384 bytes)				
	μPD78P083	FD00H to FEFFH (512 bytes)				
μPD78098 Subseries	μPD78094	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78095					
	μPD78096					
	μPD78098A			F000H to F7FFH (2048 bytes)		
	μPD78P098A					

Remark The internal high-speed RAM capacity and expansion RAM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS) and internal expansion RAM size switching register (IXS).

★ **Table 1-13. Internal RAM Space of 78K/0 Series Products (4/5)**

Subseries Name	Products	High-speed RAM	Buffer RAM	Extended RAM	RAM for FIP Display	RAM for LCD Display
μPD78098B Subseries	μPD78095B	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD78096B					
	μPD78098B			F000H to F7FFH (2048 bytes)		
	μPD78P098B					
μPD780001	μPD780001	FE40H to FEFFH (192 bytes)	–	–	–	–
μPD780018Y Subseries	μPD780016Y	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	F400H to F7FFH (1024 bytes)	–	–
	μPD780018Y					
	μPD78P0018Y					
μPD780024 Subseries	μPD780021	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780022					
	μPD780023	FB00H to FEFFH (1024 bytes)				
	μPD780024					
μPD780024Y Subseries	μPD780021Y	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780022Y					
	μPD780023Y	FB00H to FEFFH (1024 bytes)				
	μPD780024Y					
μPD780034 Subseries	μPD780031	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780032					
	μPD780033	FB00H to FEFFH (1024 bytes)				
	μPD780034					
	μPD78F0034					
μPD780034Y Subseries	μPD780031Y	FD00H to FEFFH (512 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780032Y					
	μPD780033Y	FB00H to FEFFH (1024 bytes)				
	μPD780034Y					
	μPD78F0034Y					
μPD780058 Subseries	μPD780053	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780054					
	μPD780055					
	μPD780056					
	μPD780058			F400H to F7FFH (1024 bytes)		
	μPD78F0058					

Remark The internal high-speed RAM capacity and expansion RAM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS) and internal expansion RAM size switching register (IXS).

★ **Table 1-13. Internal RAM Space of 78K/0 Series Products (5/5)**

Subseries Name	Products	High-speed RAM	Buffer RAM	Extended RAM	RAM for FIP Display	RAM for LCD Display
μPD780058Y Subseries	μPD780053Y	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (32 bytes)	–	–	–
	μPD780054Y					
	μPD780055Y					
	μPD780056Y					
	μPD780058Y			F400H to F7FFH (1024 bytes)		
	μPD78F0058Y					
μPD780208 Subseries	μPD780204	FB00H to FEFFH (1024 bytes)	FAC0H to FADFH (64 bytes)	–	FA30H to FA7FH (80 bytes)	–
	μPD780205					
	μPD780206					
	μPD780208			F000H to F7FFH (2048 bytes)		
	μPD78F0208					
μPD780228 Subseries	μPD780226	FB00H to FEFFH (1024 bytes)	–	F600H to F7FFH (512 bytes)	FA00H to FA5FH (96 bytes)	–
	μPD780228					
	μPD78F0228					
μPD780308 Subseries	μPD780306	FB00H to FEFFH (1024 bytes)	–	F400H to F7FFH (1024 bytes)	–	FA58H to FA7FH (40 × 4 bits)
	μPD780308					
	μPD78P0308					
μPD780308Y Subseries	μPD780306Y	FB00H to FEFFH (1024 bytes)	–	F400H to F7FFH (1024 bytes)	–	FA58H to FA7FH (40 × 4 bits)
	μPD780308Y					
	μPD78P0308Y					
μPD78P0914	μPD78P0914	FD00H to FEFFH (512 bytes)	–	–	–	–
μPD780924 Subseries	μPD780921	FD00H to FEFFH (512 bytes)	–	–	–	–
	μPD780922					
	μPD780923	FB00H to FEFFH (1024 bytes)				
	μPD780924					
	μPD78F0924					
μPD780964 Subseries	μPD780961	FD00H to FEFFH (512 bytes)	–	–	–	–
	μPD780962					
	μPD780963	FB00H to FEFFH (1024 bytes)				
	μPD780964					
	μPD78F0964					

Remark The internal high-speed RAM capacity and expansion RAM capacity of PROM versions can be changed by manipulating the value of the memory size switching register (IMS) and internal expansion RAM size switching register (IXS).

1.7 Special Function Register (SFR) Area

An on-chip peripheral hardware special-function register (SFR) is allocated in the area FF00H to FFFFH. (Refer to **each product User's Manual**).

Caution Do not access addresses where the SFR is not assigned. If the address is carelessly accessed, the CPU may be deadlocked.

1.8 External Memory Space

This is an external memory space which can be accessed by setting the memory extension mode register. This space allows program and table data storage, and peripheral device assignment.

For products for which an external memory space can be used, refer to **Table 1-14 External Memory Space of 78K/0 Series Products**.

★ **Table 1-14. External Memory Space of 78K/0 Series Products (1/3)**

Subseries Name	Products	Address (capacity)
μPD78002, 78002Y Subseries	μPD78001B, 78001BY	2000H to FA7FH (55936 bytes)
	μPD78002B, 78002BY	4000H to FA7FH (47744 bytes)
	μPD78P014, 78P014Y Note 1	F000H to F3FFH (1024 bytes)
μPD78014, 78014Y Subseries	μPD78011B, 78011BY	2000H to FA7FH (55936 bytes)
	μPD78012B, 78012BY	4000H to FA7FH (47744 bytes)
	μPD78013, 78013Y	6000H to FA7FH (39552 bytes)
	μPD78014, 78014Y	8000H to FA7FH (31360 bytes)
	μPD78P014, 78P014Y Note 1	F000H to F3FFH (1024 bytes)
μPD78014H Subseries	μPD78011H	2000H to FA7FH (55936 bytes)
	μPD78012H	4000H to FA7FH (47744 bytes)
	μPD78013H	6000H to FA7FH (39552 bytes)
	μPD78014H	8000H to FA7FH (31360 bytes)
	μPD78P018F Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78018F, 78018FY Subseries	μPD78011F, 78011FY	2000H to FA7FH (55936 bytes)
	μPD78012F, 78012FY	4000H to FA7FH (47744 bytes)
	μPD78013F, 78013FY	6000H to FA7FH (39552 bytes)
	μPD78014F, 78014FY	8000H to FA7FH (31360 bytes)
	μPD78015F, 78015FY	A000H to F5FFH (22016 bytes)
	μPD78016F, 78016FY	C000H to F5FFH (13824 bytes)
	μPD78018F, 78018FY Note 2	F000H to F3FFH (1024 bytes)
	μPD78P018F, 78P018FY Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78054, 78054Y Subseries	μPD78052, 78052Y	4000H to FA7FH (47744 bytes)
	μPD78053, 78053Y	6000H to FA7FH (39552 bytes)
	μPD78054, 78054Y	8000H to FA7FH (31360 bytes)
	μPD78P054, 78P054Y Note 1	8000H to FA7FH (31360 bytes)
	μPD78055, 78055Y	A000H to FA7FH (23168 bytes)
	μPD78056, 78056Y	C000H to FA7FH (14976 bytes)
	μPD78058, 78058Y Note 2	F000H to F3FFH (1024 bytes)
	μPD78P058, 78P058Y Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78058F, 78058FY Subseries	μPD78056F, 78056FY	C000H to FA7FH (14976 bytes)
	μPD78058F, 78058FY Note 2	F000H to F3FFH (1024 bytes)
	μPD78P058F, 78P058FY Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78070A, 78070AY	μPD78070A, 78070AY	0000H to FA7FH (64128 bytes)

Notes 1. The external memory capacity of PROM and flash memory versions can be changed by using the memory size switching register (IMS).

2. When an internal ROM is set to 60 Kbytes, the area of F000H to F3FFH is not available. Setting the internal ROM to 56 Kbytes or less enables the area of F000H to F3FFH to be used as an external memory.

★ **Table 1-14. External Memory Space of 78K/0 Series Products (2/3)**

Subseries Name	Products	Address (capacity)
μPD78075B, 78075BY Subseries	μPD78074B, 78074BY	8000H to FA7FH (31360 bytes)
	μPD78075B, 78075BY	A000H to FA7FH (23168 bytes)
	μPD78P078, 78P078Y Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78078, 78078Y Subseries	μPD78076, 78076Y	C000H to FA7FH (13312 bytes)
	μPD78078, 78078Y Note 2	F000H to F3FFH (1024 bytes)
	μPD78P078, 78P078Y Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD78098 Subseries	μPD78094	8000H to EFFFH (28672 bytes)
	μPD78095	A000H to EFFFH (20480 bytes)
	μPD78096	C000H to EFFFH (12288 bytes)
	μPD78098A Note 3	E000H to EFFFH (4096 bytes)
	μPD78P098A Notes 1, 3	E000H to EFFFH (4096 bytes)
μPD78098B Subseries	μPD78095B	A000H to EFFFH (20480 bytes)
	μPD78096B	C000H to EFFFH (12288 bytes)
	μPD78098B Note 3	E000H to EFFFH (4096 bytes)
	μPD78P098B Notes 1, 3	E000H to EFFFH (4096 bytes)
μPD780018Y Subseries	μPD780016Y	C000H to FA7FH (13312 bytes)
	μPD780018Y Note 2	F000H to F3FFH (1024 bytes)
	μPD78P0018Y Notes 1, 2	F000H to F3FFH (1024 bytes)
μPD780024, 780024Y Subseries	μPD780021, 780021Y	2000H to F7FFH (55296 bytes)
	μPD780022, 780022Y	4000H to F7FFH (47104 bytes)
	μPD780023, 780023Y	6000H to F7FFH (38912 bytes)
	μPD780024, 780024Y	8000H to F7FFH (30720 bytes)
	μPD78F0034, 78F0034Y Note 1	8000H to F7FFH (30720 bytes)
μPD780034, 780034Y Subseries	μPD780031, 780031Y	2000H to F7FFH (55296 bytes)
	μPD780032, 780032Y	4000H to F7FFH (47104 bytes)
	μPD780033, 780033Y	6000H to F7FFH (38912 bytes)
	μPD780034, 780034Y	8000H to F7FFH (30720 bytes)
	μPD78F0034, 78F0034Y Note 1	8000H to F7FFH (30720 bytes)

Notes 1. The external memory capacity of PROM and flash memory versions can be changed by using the memory size switching register (IMS).

2. When an internal ROM is set to 60 Kbytes, the area of F000H to F3FFH is not available. Setting the internal ROM to 56 Kbytes or less (or to 48 Kbytes only for the μPD78P0018Y) enables the area of F000H to F3FFH to be used as an external memory.

3. When an internal ROM is set to 60 Kbytes, the area of E000H to EFFFH is not available. By setting the internal ROM to 56 Kbytes or less, the internal ROM can be used as an external memory within the range of the last address to EFFFH.

★ **Table 1-14. External Memory Space of 78K/0 Series Products (3/3)**

Subseries Name	Products	Address (capacity)
μPD780058, 780058Y Subseries	μPD780053, 780053Y	6000H to FA7FH (39552 bytes)
	μPD780054, 780054Y	8000H to FA7FH (31360 bytes)
	μPD780055, 780055Y	A000H to FA7FH (23168 bytes)
	μPD780056, 780056Y	C000H to FA7FH (14976 bytes)
	μPD780058, 780058Y Note 2	F000H to F3FFH (1024 bytes)
	μPD78F0058, 78F0058Y Note 1	F000H to F3FFH (1024 bytes)
μPD78P0914	μPD78P0914 Note 1	8000H to F7FFH (30720 bytes)
μPD780924 Subseries	μPD780921	6000H to FA7FH (55296 bytes)
	μPD780922	8000H to FA7FH (47104 bytes)
	μPD780923	A000H to FA7FH (38912 bytes)
	μPD780924	C000H to FA7FH (30720 bytes)
	μPD78F0924 Note 1	F000H to F3FFH (30720 bytes)
μPD780964 Subseries	μPD780961	6000H to FA7FH (55296 bytes)
	μPD780962	8000H to FA7FH (47104 bytes)
	μPD780963	A000H to FA7FH (38912 bytes)
	μPD780964	C000H to FA7FH (30720 bytes)
	μPD78F0964 Note 1	F000H to F3FFH (30720 bytes)

Notes 1. The external memory capacity of PROM and flash memory versions can be changed by using the memory size switching register (IMS).

2. When an internal ROM is set to 60 Kbytes, the area of F000H to F3FFH is not available. Setting the internal ROM to 56 Kbytes or less enables the area of F000H to F3FFH to be used as an external memory.

★ 1.9 IEBus™ Register Area (μPD78098 and 78098B Subseries Only)

IEBus registers that are used to control the IEBus controller are allocated to the area of F8E0H to F8FFH.

[MEMO]

CHAPTER 2 REGISTER

2.1 Control Registers

The control registers control the program sequence, statuses and stack memory. A program counter, a program status word and a stack pointer are control registers.

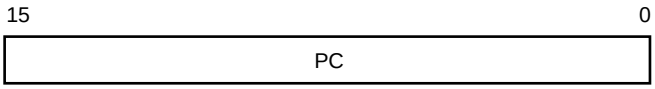
2.1.1 Program counter (PC)

The program counter is a 16-bit register which holds the address information of the next program to be executed.

In normal operation, the PC is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data and register contents are set.

$\overline{\text{RESET}}$ input sets the reset vector table values at addresses 0000H and 0001H to the program counter.

Figure 2-1. Program Counter Configuration



2.1.2 Program status word (PSW)

The program status word is an 8-bit register consisting of various flags to be set/reset by instruction execution.

Program status word contents are automatically stacked upon interrupt request generation or PUSH PSW instruction execution and are automatically reset upon execution of the RETB, RETI and POP PSW instructions.

$\overline{\text{RESET}}$ input sets the PSW to 02H.

Figure 2-2. Program Status Word Configuration

7							0
IE	Z	RBS1	AC	RBS0	0	ISP	CY

(1) Interrupt enable flag (IE)

This flag controls interrupt request acknowledge operations of CPU.

When IE = 0, the IE is set to interrupt disable (DI), and interrupts other than non-maskable interrupts are all disabled.

When IE = 1, the IE is set to interrupt enable (EI), and an interrupt request acknowledge is controlled with an in-service priority flag (ISP), an interrupt mask flag for various interrupt sources, and a priority specification flag.

This flag is reset (0) upon the DI instruction execution or interrupt request acknowledgment and is set (1) upon execution of the EI instruction.

(2) Zero flag (Z)

When the operation result is zero, this flag is set to (1). It is reset to (0) in all other cases.

(3) Register bank select flags (RBS0 and RBS1)

These are 2-bit flags to select one of the four register banks.

In these flags, the 2-bit information which indicates the register bank selected by SBL RBn instruction execution is stored.

(4) Auxiliary carry flag (AC)

If the operation result has a carry from bit 3 or a borrow at bit 3, this flag is set to (1). It is reset to (0) in all other cases.

(5) In-service priority flag (ISP)

This flag manages the priority of acknowledgeable, maskable vectored interrupts. When ISP = 0, vectored interrupt requests specified to the low level with the priority specification flag register (PR) are disabled for acknowledgment. Actual acknowledgment for interrupt requests is controlled by the state of the interrupt enable flag (IE).

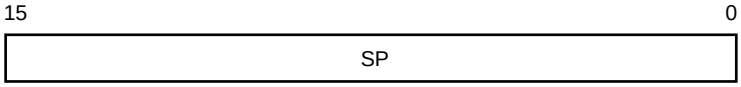
(6) Carry flag (CY)

This flag stores overflow and underflow upon add/subtract instruction execution. It stores the shift-out value upon rotate instruction execution and functions as a bit accumulator during bit manipulation instruction execution.

2.1.3 Stack pointer (SP)

This is a 16-bit register to hold the start address of the memory stack area. Only the internal high-speed RAM area can be set as the stack area.

Figure 2-3. Stack Pointer Configuration



The SP is decremented ahead of write (save) to the stack memory and is incremented after read (reset) from the stack memory.

Each stack operation saves/resets data as shown in Figures 2-4 and 2-5.

Caution Since $\overline{\text{RESET}}$ input makes SP contents undefined, be sure to initialize the SP before instruction execution.

Figure 2-4. Data to be Saved to Stack Memory

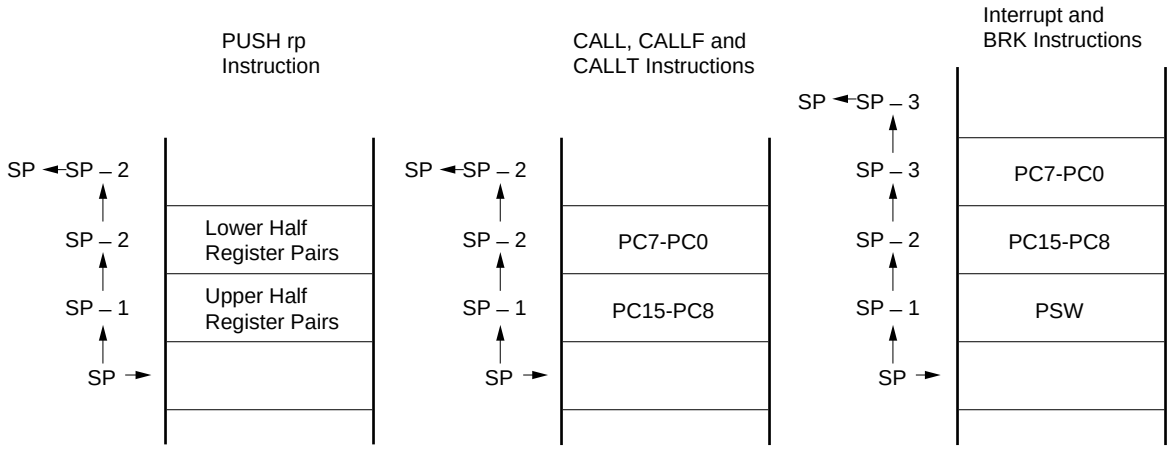
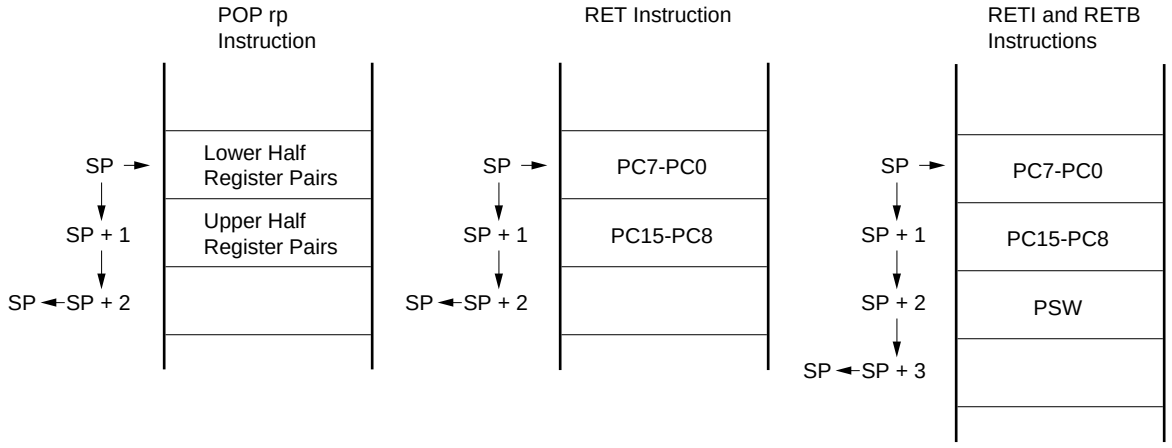


Figure 2-5. Data to be Reset from Stack Memory



2.2 General Registers

A general register is mapped at particular addresses (FEE0H to FEFFH) of the data memory. It consists of 4 banks, each bank consisting of eight 8-bit registers (X, A, C, B, E, D, L and H).

In addition that each register can be used as an 8-bit register, two 8-bit registers in pairs can be used as a 16-bit register (AX, BC, DE and HL).

They can be described in terms of functional names (X, A, C, B, E, D, L, H, AX, BC, DE and HL) and absolute names (R0 to R7 and RP0 to RP3).

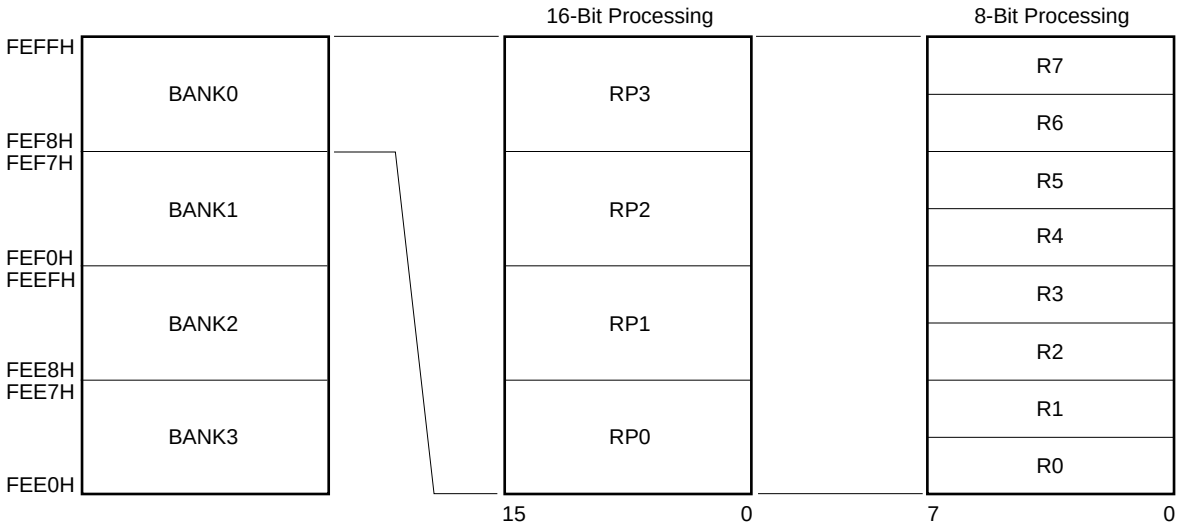
Register banks to be used for instruction execution are set with the CPU control instruction (SEL RBn). Because of the 4-register bank configuration, an efficient program can be created by switching between a register for normal processing and a register for interruption for each bank.

Table 2-1. General Register Absolute Address Correspondence Table

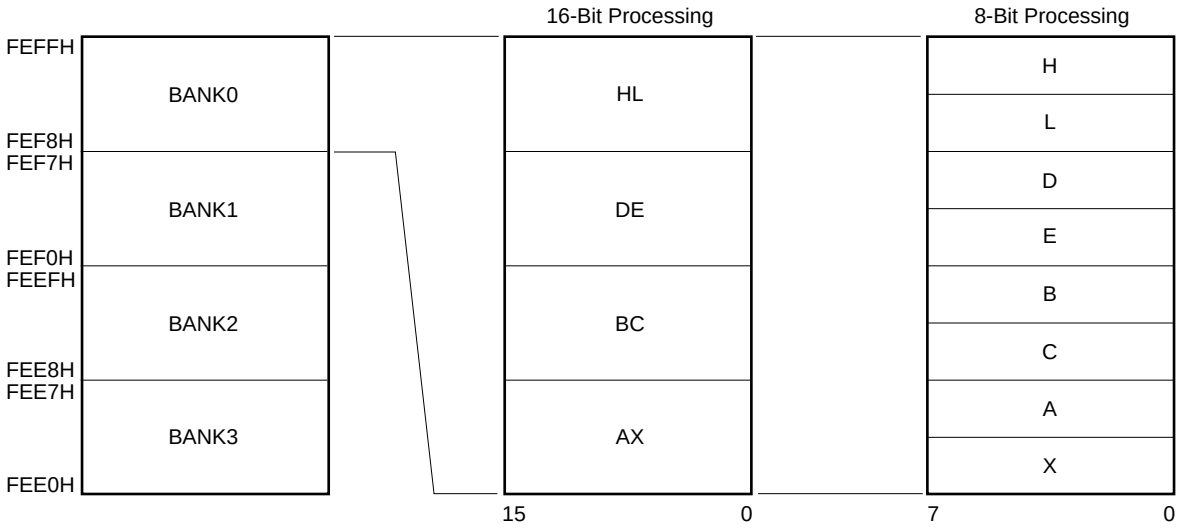
Bank Name	Register		Absolute Address	Bank Name	Register		Absolute Address
	Functional Name	Absolute Name			Functional Name	Absolute Name	
BANK0	H	R7	FEFFH	BANK2	H	R7	FEEFH
	L	R6	FEFEH		L	R6	FEEEH
	D	R5	FEFDH		D	R5	FEEDH
	E	R4	FEFCH		E	R4	FEECH
	B	R3	FEFBH		B	R3	FEEBH
	C	R2	FEFAH		C	R2	FEEAH
	A	R1	FEF9H		A	R1	FEE9H
	X	R0	FEF8H		X	R0	FEE8H
BANK1	H	R7	FEF7H	BANK3	H	R7	FEE7H
	L	R6	FEF6H		L	R6	FEE6H
	D	R5	FEF5H		D	R5	FEE5H
	E	R4	FEF4H		E	R4	FEE4H
	B	R3	FEF3H		B	R3	FEE3H
	C	R2	FEF2H		C	R2	FEE2H
	A	R1	FEF1H		A	R1	FEE1H
	X	R0	FEF0H		X	R0	FEE0H

Figure 2-6. General Register Configuration

(a) Absolute Names



(b) Functional Names



2.3 Special-Function Register (SFR)

Unlike a general register, each special-function register has a special function.

It is allocated in the 256-byte area FF00H to FFFFH.

The special-function register can be manipulated, like the general register, with the operation, transfer and bit manipulation instructions. Manipulatable bit units (1, 8, and 16) differ depending on the special-function register type.

Each manipulation bit unit can be specified as follows.

- 1-bit manipulation
  Describes a symbol reserved with assembler for the 1-bit manipulation instruction operand (sfr.bit). This manipulation can also be specified with an address.

- 8-bit manipulation
  Describes a symbol reserved with assembler for the 8-bit manipulation instruction operand (sfr). This manipulation can also be specified with an address.

- 16-bit manipulation
  Describes a symbol reserved with assembler for the 16-bit manipulation instruction operand (sfrp). When addressing an address, describe an even address.

With the special function register, refer to **each product User's Manual**.

Caution Do not access addresses where the SFR is not assigned. If the address is carelessly accessed, the CPU may be deadlocked.

CHAPTER 3 ADDRESSING

3.1 Instruction Address Addressing

An instruction address is determined by program counter (PC) contents. The PC contents are normally incremented (+1 for each byte) automatically according to the number of bytes of an instruction to be fetched each time another instruction is executed. When a branch instruction is executed, the branch destination information is set to the PC and branched by the following addressing (For details of each instruction, refer to **CHAPTER 5 EXPLANATION OF INSTRUCTIONS**).

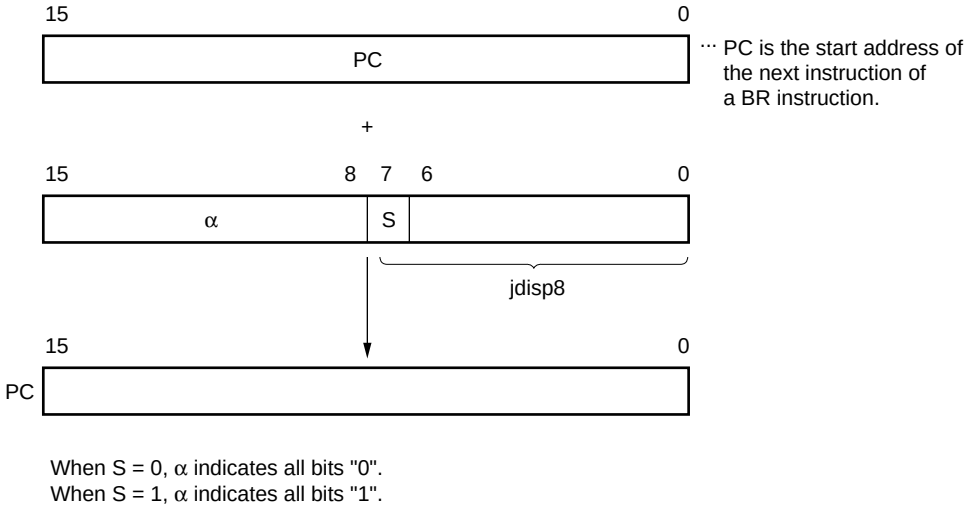
3.1.1 Relative addressing

[Function]

The value obtained by adding 8-bit immediate data (displacement value: jdisp8) of an instruction code to the start address of the following instruction is transferred to the program counter (PC) and branched. The displacement value is treated as signed two's complement data (–128 to +127) and bit 7 becomes a sign bit. In other words, in relative addressing, the value is relatively transferred to the range between –128 and +127 from the start address of the following instruction.

This function is carried out when the "BR $addr16" instruction or a conditional branch instruction is executed.

[Illustration]



When S = 0, α indicates all bits "0".
When S = 1, α indicates all bits "1".

3.1.2 Immediate addressing

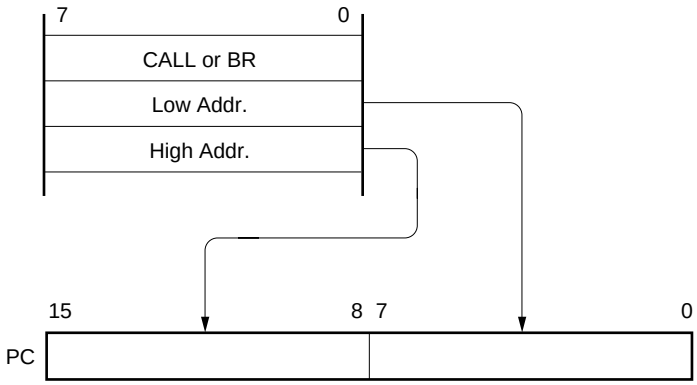
[Function]

Immediate data in the instruction word is transferred to the program counter (PC) and branched.

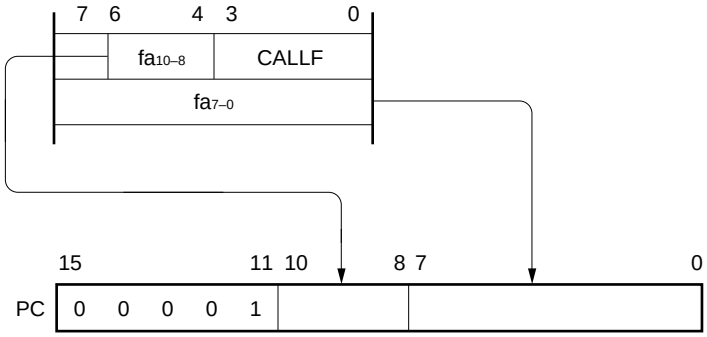
This function is carried out when the “CALL !addr16” or “BR !addr16” or “CALLF !addr11” instruction is executed. The CALL !addr16 and BR !addr16 instructions can be branched to all memory spaces. The CALLF !addr11 instruction is branched to the area of 0800H to 0FFFH.

[Illustration]

In case of CALL !addr16, BR !addr16 instruction



In case of CALLF !addr11 instruction



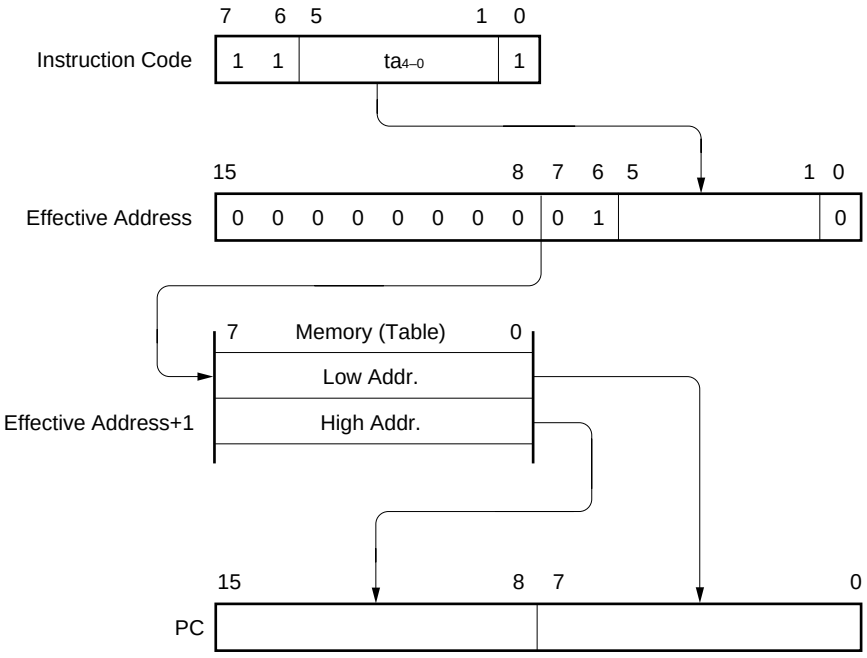
3.1.3 Table indirect addressing

[Function]

Table contents (branch destination address) of the particular location to be addressed by the low-order-5-bit immediate data of an instruction code from bit 1 to bit 5 are transferred to the program counter (PC) and branched.

When the “CALLT [addr5]” instruction is executed, a table indirect addressing is performed. Executing this instruction enables the value to be branched to all memory spaces referencing the address stored to the memory table of 40H to 7FH.

[Illustration]



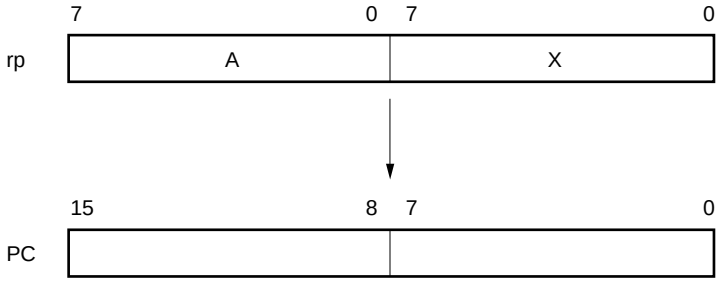
3.1.4 Register addressing

[Function]

Register pair (AX) contents to be specified with an instruction word are transferred to the program counter (PC) and branched.

This function is carried out when the “BR AX” instruction is executed.

[Illustration]



3.2 Operand Address Addressing

The following various methods are available to specify the register and memory (addressing) which undergo manipulation during instruction execution.

3.2.1 Implied addressing

[Function]

This addressing automatically specifies to an address the register which functions as an accumulator (A and AX) in the general register.

Of the 78K/0 Series instruction words, the following instructions employ implied addressings.

Instruction	Register to be Specified by Implied Addressing
MULU[Note]	A register for multiplicand and AX register for product storage
DIVUW[Note]	AX register for dividend and quotient storage
ADJBA/ADJBS	A register for storage of numeric values which become decimal correction
ROR4/ROL4	A register for storage of digit data which undergoes digit rotation

Note The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

[Operand format]

Because implied addressing can be automatically employed with an instruction, no particular operand format is necessary.

[Description example]

In the case of MULU X

With an 8-bit x 8-bit multiply instruction, the product of A register and X register is stored in AX. In this example, the A and AX registers are specified by implied addressing.

3.2.2 Register addressing

[Function]

Register addressing accesses the general-purpose register as an operand. The general-purpose register to be accessed is specified by the register bank selection flags (RBS0 and RBS1) and the register specification codes (Rn and RPn) among instruction codes.

Register addressing is carried out when an instruction with the following operand format is executed. When an 8-bit register is specified, one of the eight registers is specified with 3 bits in the instruction code.

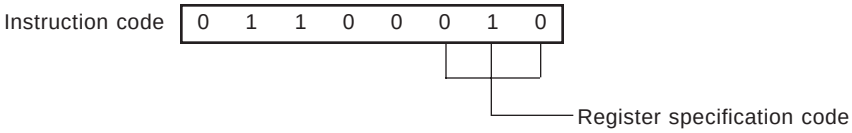
[Operand format]

Identifier	Description
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

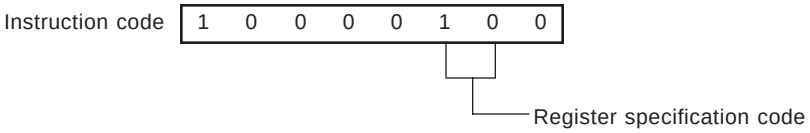
'r' and 'rp' can be described with absolute names (R0 to R7 and RP0 to RP3) as well as function names (X, A, C, B, E, D, L, H, AX, BC, DE and HL).

[Description example]

MOV A, C; When selecting the C register for r



INCW DE; When selecting the DE register pair for rp



3.2.3 Direct addressing

[Function]

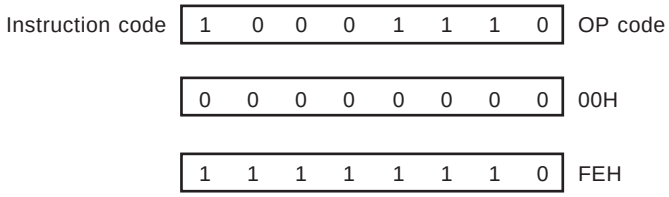
Direct addressing directly addresses the memory indicated by the immediate data in the instruction word.

[Operand format]

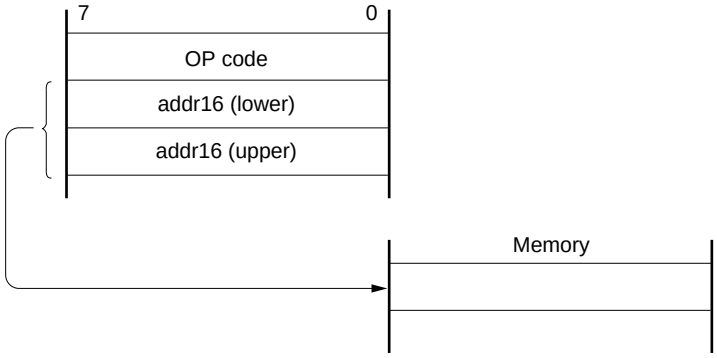
Identifier	Description
addr16	Label or 16-bit immediate data

[Description example]

MOV A, !FE00H; When setting !addr16 to FE00H



[Illustration]



3.2.4 Short direct addressing

[Function]

The memory to be manipulated in the fixed space is directly addressed with 8-bit data in an instruction word. This addressing is applied to the 256-byte fixed space FE20H to FF1FH. An internal high-speed RAM and a special-function register (SFR) are mapped at FE20H to FEFFH and FF00H to FF1FH, respectively.

The SFR area (FF00H to FF1FH) where short direct addressing is applied is a part of the entire SFR area. Ports which are frequently accessed in a program, a compare register of the timer/event counter and a capture register of the timer/event counter are mapped in the area FF00H through FF1FH, and these SFRs can be manipulated with a small number of bytes and clocks.

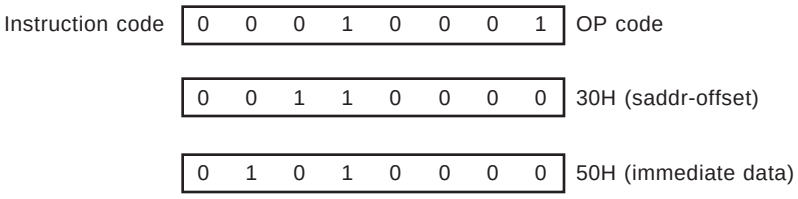
When 8-bit immediate data is at 20H to FFH, bit 8 of an effective address is set to 0. When it is at 00H to 1FH, bit 8 is set to 1. See **[Illustration]** below.

[Operand format]

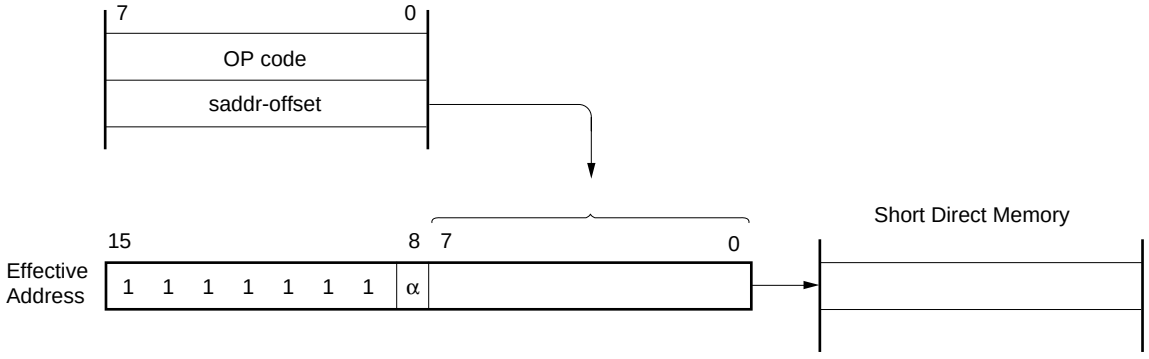
Identifier	Description
saddr	Label or FE20H to FF1FH immediate data
saddrp	Label or FE20H to FF1FH immediate data (even address only)

[Description example]

MOV FE30H, #50H; When setting saddr to FE30H and the immediate data to 50H



[Illustration]



When 8-bit immediate data is 20H to FFH, α = 0.
When 8-bit immediate data is 00H to 1FH, α = 1.

3.2.5 Special-function register (SFR) addressing

[Function]

The memory-mapped special-function register (SFR) is addressed with 8-bit immediate data in an instruction word.

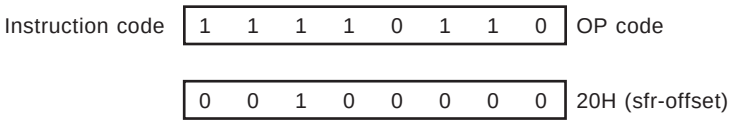
This addressing is applied to the 240-byte spaces FF00H to FFCFH and FFE0H to FFFFH. However, the SFR mapped at FF00H to FF1FH can be accessed with short direct addressing.

[Operand format]

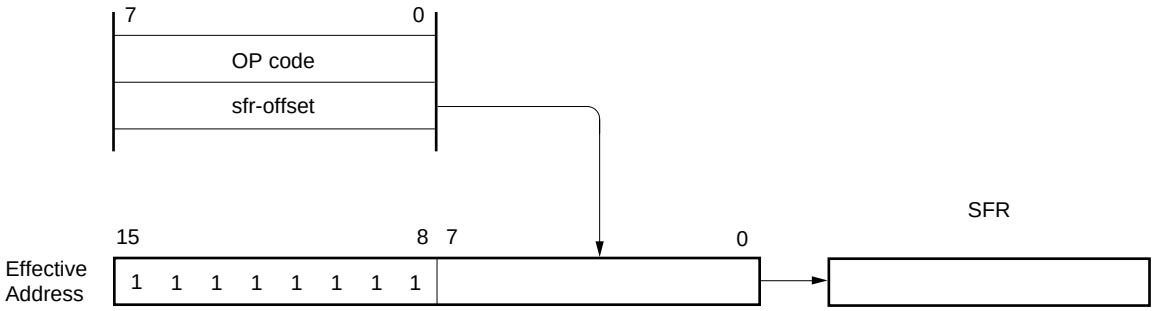
Identifier	Description
sfr	Special-function register name
sfrp	16-bit manipulatable special-function register name (even address only)

[Description example]

MOV PM0, A; When selecting PM0 for sfr



[Illustration]



3.2.6 Register indirect addressing

[Function]

The register indirect addressing addresses memory with register pair contents specified as an operand. The register pair to be accessed is specified by the register bank selection flags (RBS0 and RBS1) and the register pair specification in instruction codes.

[Operand format]

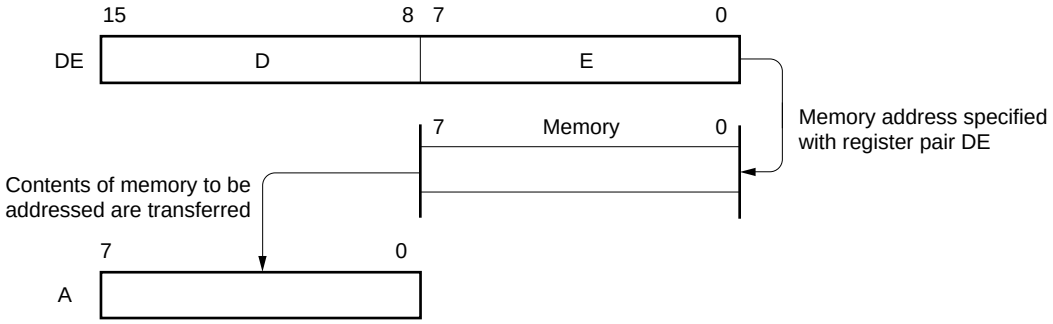
Identifier	Description
–	[DE], [HL]

[Description example]

MOV A, [DE]; When selecting register pair [DE]

Instruction code	1	0	0	0	0	1	0	1

[Illustration]



3.2.7 Based addressing

[Function]

8-bit immediate data is added to the contents of the HL register pair as a base register and the sum is used to address the memory. The HL register pair to be accessed is in the register bank specified with the register bank select flag (RBS0 and RBS1). Addition is performed by expanding the offset data as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

[Operand format]

Identifier	Description
—	[HL+byte]

[Description example]

MOV A, [HL+10H]; When setting byte to 10H

Instruction code	1	0	1	0	1	1	1	0
	0	0	0	1	0	0	0	0

3.2.8 Based indexed addressing

[Function]

The B or C register contents specified in an instruction word are added to the contents of the HL register pair as a base register and the sum is used to address the memory. The HL, B, and C registers to be accessed are registers in the register bank specified with the register bank select flag (RBS0 to RBS1). Addition is performed by expanding the B or C register as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

[Operand format]

Identifier	Description
—	[HL+B], [HL+C]

[Description example]

In the case of MOV A, [HL+B]

Instruction code	1	0	1	0	1	0	1	1

3.2.9 Stack addressing

[Function]

The stack area is indirectly addressed with the stack pointer (SP) contents.

This addressing method is automatically employed when the PUSH, POP, subroutine call and RETURN instructions are executed or the register is saved/reset upon generation of an interrupt request.

Stack addressing enables to address the internal high-speed RAM area only.

[Description example]

In the case of PUSH DE

Instruction code	1	0	1	1	0	1	0	1

[MEMO]

CHAPTER 4 INSTRUCTION SET

This chapter covers the list of 78K/0 Series instruction set. The instructions are common to all the 78K/0 Series products.

4.1 Operation

4.1.1 Operand identifiers and description methods

Operands are described in “Operand” column of each instruction in accordance with the description method of the instruction operand identifier (refer to the assembler specifications for detail). When there are two or more description methods, select one of them. Alphabetic letters in capitals and symbols, #, !, $ and [ ] are key words and are described as they are. Each symbol has the following meaning.

- # : Immediate data specification
- ! : Absolute address specification
- $: Relative address specification
- [ ] : Indirect address specification

In the case of immediate data, describe an appropriate numeric value or a label. When using a label, be sure to describe the #, !, $ and [ ] symbols.

For operand register identifiers, r and rp, either function names (X, A, C, etc.) or absolute names (names in parentheses in the table below, R0, R1, R2, etc.) can be used for description.

Table 4-1. Operand Identifiers and Description Methods

Identifier	Description Method
r	X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7)
rp	AX (RP0), BC (RP1), DE (RP2), HL (RP3)
sfr	Special-function register symbol[Note]
sfrp	Special-function register symbols (16-bit manipulatable register even addresses only)[Note]
saddr	FE20H to FF1FH Immediate data or labels
saddrp	FE20H to FF1FH Immediate data or labels (even addresses only)
addr16	0000H to FFFFH Immediate data or labels (Only even addresses for 16-bit data transfer instructions)
addr11	0800H to 0FFFH Immediate data or labels
addr5	0040H to 007FH Immediate data or labels (even addresses only)
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
RBn	RB0 to RB3

Note FFD0H to FFDFH are not addressable.

Remark Refer to **each product User’s Manual** for symbols of special function registers.

4.1.2 Description of “operation” column

A : A register; 8-bit accumulator
X : X register
B : B register
C : C register
D : D register
E : E register
H : H register
L : L register
AX : AX register pair; 16-bit accumulator
BC : BC register pair
DE : DE register pair
HL : HL register pair
PC : Program counter
SP : Stack pointer
PSW : Program status word
CY : Carry flag
AC : Auxiliary carry flag
Z : Zero flag
RBS : Register bank select flag
IE : Interrupt request enable flag
NMIS : Flag indicating non-maskable interrupt servicing in progress
( ) : Memory contents indicated by address or register contents in parentheses
X_H, X_L : Higher 8 bits and lower 8 bits of 16-bit register
∧ : Logical product (AND)
∨ : Logical sum (OR)
$\veebar$: Exclusive logical sum (exclusive OR)
— : Inverted data
addr16 : 16-bit immediate data or label
jdisp8 : Signed 8-bit data (displacement value)

4.1.3 Description of “flag operation” column

(Blank) : Unchanged
0 : Cleared to 0
1 : Set to 1
× : Set/cleared according to the result
R : Previously saved value is restored

4.1.4 Description of “clock” column

The number of clock cycles during instruction execution is outlined as follows.
1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

(1) Classification of “clock” column

The number of instruction clocks differs in the following two cases.

<1> When the internal high-speed RAM area is accessed, or in the instruction with no data access

<2> When an area except the internal high-speed RAM area is accessed

(2) When “n” or “m” is written in “clock” column

n indicates the number of waits when external memory expansion area is read.
m indicates the number of waits when external memory expansion area is written to.

(3) Number of clocks if program is stored in external ROM

<1> When no wait cycle is inserted

The number of clocks is the same as when the program is stored in the internal ROM area.

<2> When wait cycle is inserted

★ The number of clocks differs depending on the product. There are the following three product groups classified according to the number of clocks.

- With μPD78002, 78002Y, 78014, 78014Y, 78014H, 78018F, 78018FY, 78P0914 Subseries
  The number of clocks increases by **“the number of bytes × number of wait cycles × 2”**, as compared with when the program is stored in the internal ROM area.
- μPD78054, 78054Y, 78058F, 78058FY, 78075B, 78075BY, 78078, 78078Y, 78098, 78098B, 780018Y, 780024, 780024Y, 780034, 780034Y, 780058, 780058Y, 780308, 780308Y, 780924, 780964 Subseries
  The number of clocks increases by **“the number of bytes × number of wait cycles”**, as compared with when the program is stored in the internal ROM area.
- μPD78070A, 78070AY
  The μPD78070A and 78070AY do not incorporate an internal ROM area.
  The number of clocks depends on the instruction (See **4.1.5 Operation list (4)**).

The number of clocks during instruction execution differs in the product. These are divided into the following four groups.

- μPD78002, 78002Y, 78014, 78014Y, 78014H, 78018F, 78018FY Subseries and μPD780001, 78P0914
- μPD78044F, 78044H, 78064, 78064Y, 78064B, 780208, 780228 Subseries
- μPD78054, 78054Y, 78058F, 78058FY, 78075B, 78075BY, 78078, 78078Y, 78083, 78098, 78098B, 780018Y, 780024, 780024Y, 780034, 780034Y, 780058, 780058Y, 780308, 780308Y, 780924, 780964 Subseries
- μPD78070A, 78070AY

The operation list for each product is shown below.

4.1.5 Operation list

★ **(1) μPD78002/78002Y/78014/78014Y/78014H/78018F/78018FY Subseries and μPD780001, 78P0914**

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**MOV**	r,#byte	2	8	–	r ← byte	
		saddr,#byte	3	12	14	(saddr) ← byte	
		sfr,#byte	3	–	14	sfr ← byte	
		A,r Note 3	1	4	–	A ← r	
		r,A Note 3	1	4	–	r ← A	
		A,saddr	2	8	10	A ← (saddr)	
		saddr,A	2	8	10	(saddr) ← A	
		A,sfr	2	–	10	A ← sfr	
		sfr,A	2	–	10	sfr ← A	
		A,!addr16	3	16	18+2n	A ← (addr16)	
		!addr16,A	3	16	18+2m	(addr16) ← A	
		PSW,#byte	3	–	14	PSW ← byte	× × ×
		A,PSW	2	–	10	A ← PSW	
		PSW,A	2	–	10	PSW ← A	× × ×
		A,[DE]	1	8	10+2n	A ← (DE)	
		[DE],A	1	8	10+2m	(DE) ← A	
		A,[HL]	1	8	10+2n	A ← (HL)	
		[HL],A	1	8	10+2m	(HL) ← A	
		A,[HL+byte]	2	16	18+2n	A ← (HL+byte)	
		[HL+byte],A	2	16	18+2m	(HL+byte) ← A	
		A,[HL+B]	1	12	14+2n	A ← (HL+B)	
		[HL+B],A	1	12	14+2m	(HL+B) ← A	
		A,[HL+C]	1	12	14+2n	A ← (HL+C)	
		[HL+C],A	1	12	14+2m	(HL+C) ← A	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**XCH**	A,r Note 3	1	4	–	A ↔ r	
		A,saddr	2	8	12	A ↔ (saddr)	
		A,sfr	2	–	12	A ↔ (sfr)	
		A,!addr16	3	16	20+2n+2m	A ↔ (addr16)	
		A,[DE]	1	8	12+2n+2m	A ↔ (DE)	
		A,[HL]	1	8	12+2n+2m	A ↔ (HL)	
		A,[HL+byte]	2	16	20+2n+2m	A ↔ (HL+byte)	
		A,[HL+B]	2	16	20+2n+2m	A ↔ (HL+B)	
		A,[HL+C]	2	16	20+2n+2m	A ↔ (HL+C)	
16-Bit Data Transfer	**MOVW**	rp,#word	3	12	–	rp ← word	
		saddrp,#word	4	16	20	(saddrp) ← word	
		sfrp,#word	4	–	20	sfrp ← word	
		AX,saddrp	2	12	16	AX ← (saddrp)	
		saddrp,AX	2	12	16	(saddrp) ← AX	
		AX,sfrp	2	–	16	AX ← sfrp	
		sfrp,AX	2	–	16	sfrp ← AX	
		AX,rp Note 4	1	8	–	A X ← r p	
		rp,AX Note 4	1	8	–	rp ← AX	
		AX,!addr16	3	20	24+4n	AX ← (addr16)	
		!addr16,AX	3	20	24+4m	(addr16) ← AX	
	XCHW	AX,rp Note 4	1	8	–	AX ↔ rp	
8-Bit Operation	**ADD**	A,#byte	2	8	–	A, CY ← A+byte	× × ×
		saddr,#byte	3	12	16	(saddr), CY ← (saddr) + byte	× × ×
		A,r Note 3	2	8	–	A,CY ← A+r	× × ×
		r,A	2	8	–	r,CY ← r+A	× × ×
		A,saddr	2	8	10	A,CY ← A+(saddr)	× × ×
		A,!addr16	3	16	18+2n	A,CY ← A+(addr16)	× × ×
		A,[HL]	1	8	10+2n	A,CY ← A+(HL)	× × ×
		A,[HL+byte]	2	16	18+2n	A,CY ← A+(HL+byte)	× × ×
		A,[HL+B]	2	16	18+2n	A, CY ← A+(HL+B)	× × ×
		A,[HL+C]	2	16	18+2n	A,CY ← A+(HL+C)	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

4. Only when rp = BC, DE or HL.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**ADDC**	A,#byte	2	8	–	A,CY ← A+byte+CY	× × ×
		saddr,#byte	3	12	16	(saddr), CY ← (saddr)+byte+CY	× × ×
		A,r Note 3	2	8	–	A,CY ← A+r+CY	× × ×
		r,A	2	8	–	r,CY ← r+A+CY	× × ×
		A,saddr	2	8	10	A,CY ← A+(saddr)+CY	× × ×
		A,!addr16	3	16	18+2n	A,CY ← A+(addr16)+CY	× × ×
		A,[HL]	1	8	10+2n	A,CY ← A+(HL)+CY	× × ×
		A,[HL+byte]	2	16	18+2n	A,CY ← A+(HL+byte)+CY	× × ×
		A,[HL+B]	2	16	18+2n	A,CY ← A+(HL+B)+CY	× × ×
		A,[HL+C]	2	16	18+2n	A,CY ← A+(HL+C)+CY	× × ×
	SUB	A,#byte	2	8	–	A,CY ← A–byte	× × ×
		saddr,#byte	3	12	16	(saddr),CY ← (saddr)– byte	× × ×
		A,r Note 3	2	8	–	A,CY ← A–r	× × ×
		r,A	2	8	–	r,CY ← r–A	× × ×
		A,saddr	2	8	10	A,CY ← A–(saddr)	× × ×
		A,!addr16	3	16	18+2n	A,CY ← A–(addr16)	× × ×
		A,[HL]	1	8	10+2n	A,CY ← A–(HL)	× × ×
		A,[HL+byte]	2	16	18+2n	A,CY ← A–(HL+byte)	× × ×
		A,[HL+B]	2	16	18+2n	A,CY ← A–(HL+B)	× × ×
		A,[HL+C]	2	16	18+2n	A,CY ← A–(HL+C)	× × ×
	SUBC	A,#byte	2	8	–	A,CY ← A–byte–CY	× × ×
		saddr,#byte	3	12	16	(saddr),CY ← (saddr)–byte–CY	× × ×
		A,r Note 3	2	8	–	A,CY ← A–r–CY	× × ×
		r,A	2	8	–	r,CY ← r–A–CY	× × ×
		A,saddr	2	8	10	A,CY ← A–(saddr)–CY	× × ×
		A,!addr16	3	16	18+2n	A,CY ← A–(addr16)–CY	× × ×
		A,[HL]	1	8	10+2n	A,CY ← A–(HL)–CY	× × ×
		A,[HL+byte]	2	16	18+2n	A,CY ← A–(HL+byte)–CY	× × ×
		A,[HL+B]	2	16	18+2n	A,CY ← A–(HL+B)–CY	× × ×
		A,[HL+C]	2	16	18+2n	A,CY ← A–(HL+C)–CY	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**AND**	A,#byte	2	8	–	A ← A∧byte	×
		saddr,#byte	3	12	16	(saddr) ← (saddr)∧byte	×
		A,r Note 3	2	8	–	A ← A∧r	×
		r,A	2	8	–	r ← r∧A	×
		A,saddr	2	8	10	A ← A∧(saddr)	×
		A,!addr16	3	16	18+2n	A ← A∧(addr16)	×
		A,[HL]	1	8	10+2n	A ← A∧(HL)	×
		A,[HL+byte]	2	16	18+2n	A ← A∧(HL+byte)	×
		A,[HL+B]	2	16	18+2n	A ← A∧(HL+B)	×
		A,[HL+C]	2	16	18+2n	A ← A∧(HL+C)	×
	OR	A,#byte	2	8	–	A ← A∨byte	×
		saddr,#byte	3	12	16	(saddr) ← (saddr)∨byte	×
		A,r Note 3	2	8	–	A ← A∨r	×
		r,A	2	8	–	r ← r∨A	×
		A,saddr	2	8	10	A ← A∨(saddr)	×
		A,!addr16	3	16	18+2n	A ← A∨(addr16)	×
		A,[HL]	1	8	10+2n	A ← A∨(HL)	×
		A,[HL+byte]	2	16	18+2n	A ← A∨(HL+byte)	×
		A,[HL+B]	2	16	18+2n	A ← A∨(HL+B)	×
		A,[HL+C]	2	16	18+2n	A ← A∨(HL+C)	×
	XOR	A,#byte	2	8	–	A ← A∨byte	×
		saddr,#byte	3	12	16	(saddr) ← (saddr)∨byte	×
		A,r Note 3	2	8	–	A ← A∨r	×
		r,A	2	8	–	r ← r∨A	×
		A,saddr	2	8	10	A ← A∨(saddr)	×
		A,!addr16	3	16	18+2n	A ← A∨(addr16)	×
		A,[HL]	1	8	10+2n	A ← A∨(HL)	×
		A,[HL+byte]	2	16	18+2n	A ← A∨(HL+byte)	×
		A,[HL+B]	2	16	18+2n	A ← A∨(HL+B)	×
		A,[HL+C]	2	16	18+2n	A ← A∨(HL+C)	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**CMP**	A,#byte	2	8	–	A–byte	× × ×
		saddr,#byte	3	12	16	(saddr)–byte	× × ×
		A,r Note 3	2	8	–	A–r	× × ×
		r,A	2	8	–	r–A	× × ×
		A,saddr	2	8	10	A–(saddr)	× × ×
		A,!addr16	3	16	18+2n	A–(addr16)	× × ×
		A,[HL]	1	8	10+2n	A–(HL)	× × ×
		A,[HL+byte]	2	16	18+2n	A–(HL+byte)	× × ×
		A,[HL+B]	2	16	18+2n	A–(HL+B)	× × ×
		A,[HL+C]	2	16	18+2n	A–(HL+C)	× × ×
16-Bit Operation	**ADDW**	AX,#word	3	12	–	AX,CY ← AX+word	× × ×
	SUBW	AX,#word	3	12	–	AX,CY ← AX–word	× × ×
	CMPW	AX,#word	3	12	–	AX–word	× × ×
Multiply/ divide	**MULU** Note 4	X	2	32	–	AX ← A×X	
	DIVUW Note 4	C	2	50	–	AX (quotient), C (remainder) ← AX ÷ C	
Increment/ decrement	**INC**	r	1	4	–	r ← r+1	× ×
		saddr	2	8	12	(saddr) ← (saddr)+1	× ×
	DEC	r	1	4	–	r ← r–1	× ×
		saddr	2	8	12	(saddr) ← (saddr)–1	× ×
	INCW	rp	1	8	–	rp ← rp+1	
	DECW	rp	1	8	–	rp ← rp–1	
Rotate	**ROR**	A,1	1	4	–	(CY, $A_7 \leftarrow A_0$, $A_{m-1} \leftarrow A_m$)×1	×
	ROL	A,1	1	4	–	(CY, $A_0 \leftarrow A_7$, $A_{m+1} \leftarrow A_m$)×1	×
	RORC	A,1	1	4	–	(CY ← A_0, $A_7 \leftarrow$ CY, $A_{m-1} \leftarrow A_m$)×1	×
	ROLC	A,1	1	4	–	(CY ← A_7, $A_0 \leftarrow$ CY, $A_{m+1} \leftarrow A_m$)×1	×
	ROR4	[HL]	2	20	24+2n+2m	$A_{3-0} \leftarrow (HL)_{3-0}$, $(HL)_{7-4} \leftarrow A_{3-0}$, $(HL)_{3-0} \leftarrow (HL)_{7-4}$	
	ROL4	[HL]	2	20	24+2n+2m	$A_{3-0} \leftarrow (HL)_{7-4}$, $(HL)_{3-0} \leftarrow A_{3-0}$, $(HL)_{7-4} \leftarrow (HL)_{3-0}$	
BCD Adjust	**ADJBA**		2	8	–	Decimal Adjust Accumulator after Addition	× × ×
	ADJBS		2	8	–	Decimal Adjust Accumulator after Subtract	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

4. The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	**Note 2**		Z AC CY
Bit Manipulation	**MOV1**	CY,saddr.bit	3	12	14	CY ← (saddr.bit)	×
		CY,sfr.bit	3	–	14	CY ← sfr.bit	×
		CY,A.bit	2	8	–	CY ← A.bit	×
		CY,PSW.bit	3	–	14	CY ← PSW.bit	×
		CY,[HL].bit	2	12	14+2n	CY ← (HL).bit	×
		saddr.bit,CY	3	12	16	(saddr.bit) ← CY	
		sfr.bit,CY	3	–	16	sfr.bit ← CY	
		A.bit,CY	2	8	–	A.bit ← CY	
		PSW.bit,CY	3	–	16	PSW.bit ← CY	× ×
		[HL].bit,CY	2	12	16+2n+2m	(HL).bit ← CY	
	AND1	CY,saddr.bit	3	12	14	CY ← CY∧(saddr.bit)	×
		CY,sfr.bit	3	–	14	CY ← CY∧sfr.bit	×
		CY,A.bit	2	8	–	CY ← CY∧A.bit	×
		CY,PSW.bit	3	–	14	CY ← CY∧PSW.bit	×
		CY,[HL].bit	2	12	14+2n	CY ← CY∧(HL).bit	×
	OR1	CY,saddr.bit	3	12	14	CY ← CY∨(saddr.bit)	×
		CY,sfr.bit	3	–	14	CY ← CY∨sfr.bit	×
		CY,A.bit	2	8	–	CY ← CY∨A.bit	×
		CY,PSW.bit	3	–	14	CY ← CY∨PSW.bit	×
		CY,[HL].bit	2	12	14+2n	CY ← CY∨(HL).bit	×
	XOR1	CY,saddr.bit	3	12	14	CY ← CY⩛(saddr.bit)	×
		CY,sfr.bit	3	–	14	CY ← CY⩛sfr.bit	×
		CY,A.bit	2	8	–	CY ← CY⩛A.bit	×
		CY,PSW.bit	3	–	14	CY ← CY⩛PSW.bit	×
		CY,[HL].bit	2	12	14+2n	CY ← CY⩛(HL).bit	×
	SET1	saddr.bit	2	8	12	(saddr.bit) ← 1	
		sfr.bit	3	–	16	sfr.bit ← 1	
		A.bit	2	8	–	A.bit ← 1	
		PSW.bit	2	–	12	PSW.bit ← 1	× × ×
		[HL].bit	2	12	16+2n+2m	(HL).bit ← 1	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Bit Manipulation	**CLR1**	saddr.bit	2	8	12	(saddr.bit) ← 0	
		sfr.bit	3	–	16	sfr.bit ← 0	
		A.bit	2	8	–	A.bit ← 0	
		PSW.bit	2	–	12	PSW.bit ← 0	× × ×
		[HL].bit	2	12	16+2n+2m	(HL).bit ← 0	
	SET1	CY	1	4	–	CY ← 1	1
	CLR1	CY	1	4	–	CY ← 0	0
	NOT1	CY	1	4	–	CY ← $\overline{CY}$	×
Call Return	**CALL**	!addr16	3	14	–	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, PC ← addr16, SP ← SP–2	
	CALLF	!addr11	2	10	–	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_{15-11} \leftarrow 00001$, $PC_{10-0} \leftarrow$ addr11, SP ← SP–2	
	CALLT	[addr5]	1	12	–	$(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (00000000, addr5+1), $PC_L \leftarrow$ (00000000, addr5), SP ← SP–2	
	BRK		1	12	–	$(SP-1) \leftarrow$ PSW, $(SP-2) \leftarrow (PC+1)_H$, $(SP-3) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (003FH), $PC_L \leftarrow$ (003EH), SP ← SP–3, IE ← 0	
	RET		1	12	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), SP ← SP+2	
	RETI		1	12	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), PSW ← (SP+2), SP ← SP+3, NMIS ← 0	R R R
	RETB		1	12	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), PSW ← (SP+2), SP ← SP+3	R R R
Stack Manipulation	**PUSH**	PSW	1	4	–	$(SP-1) \leftarrow$ PSW, SP ← SP–1	
		rp	1	8	–	$(SP-1) \leftarrow rp_H$, $(SP-2) \leftarrow rp_L$, SP ← SP–2	
	POP	PSW	1	4	–	PSW ← (SP), SP ← SP+1	R R R
		rp	1	8	–	$rp_H \leftarrow (SP+1)$, $rp_L \leftarrow$ (SP), SP ← SP+2	
	MOVW	SP,#word	4	–	20	SP ← word	
		SP, AX	2	–	16	SP ← AX	
		AX, SP	2	–	16	AX ← SP	
Unconditional Branch	**BR**	!addr16	3	12	–	PC ← addr16	
		$addr16	2	12	–	PC ← PC+2+jdisp8	
		AX	2	16	–	$PC_H \leftarrow$ A, $PC_L \leftarrow$ X	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Conditional Branch	**BC**	$addr16	2	12	–	PC ← PC+2+jdisp8 if CY=1	
	BNC	$addr16	2	12	–	PC ← PC+2+jdisp8 if CY=0	
	BZ	$addr16	2	12	–	PC ← PC+2+jdisp8 if Z=1	
	BNZ	$addr16	2	12	–	PC ← PC+2+jdisp8 if Z=0	
	BT	saddr.bit,$addr16	3	16	18	PC ← PC+3+jdisp8 if (saddr.bit)=1	
		sfr.bit,$addr16	4	–	22	PC ← PC+4+jdisp8 if sfr.bit=1	
		A.bit,$addr16	3	16	–	PC ← PC+3+jdisp8 if A.bit=1	
		PSW.bit,$addr16	3	–	18	PC ← PC+3+jdisp8 if PSW.bit=1	
		[HL].bit,$addr16	3	20	22+2n	PC ← PC+3+jdisp8 if (HL).bit=1	
	BF	saddr.bit,$addr16	4	20	22	PC ← PC+4+jdisp8 if (saddr.bit)=0	
		sfr.bit,$addr16	4	–	22	PC ← PC+4+jdisp8 if sfr.bit=0	
		A.bit,$addr16	3	16	–	PC ← PC+3+jdisp8 if A.bit=0	
		PSW.bit,$addr16	4	–	22	PC ← PC+4+jdisp8 if PSW.bit=0	
		[HL].bit,$addr16	3	20	22+2n	PC ← PC+3+jdisp8 if (HL).bit=0	
	BTCLR	saddr.bit,$addr16	4	20	24	PC ← PC+4+jdisp8 if (saddr.bit)=1 then reset (saddr.bit)	
		sfr.bit,$addr16	4	–	24	PC ← PC+4+jdisp8 if sfr.bit=1 then reset sfr.bit	
		A.bit,$addr16	3	16	–	PC ← PC+3+jdisp8 if A.bit=1 then reset A.bit	
		PSW.bit,$addr16	4	–	24	PC ← PC+4+jdisp8 if PSW.bit=1 then reset PSW.bit	× × ×
		[HL].bit,$addr16	3	20	24+2n+2m	PC ← PC+3+jdisp8 if (HL).bit=1 then reset (HL).bit	
	DBNZ	B,$addr16	2	12	–	B ← B–1, then PC ← PC+2+jdisp8 if B≠0	
		C,$addr16	2	12	–	C ← C–1, then PC ← PC+2+jdisp8 if C≠0	
		saddr,$addr16	3	16	20	(saddr) ← (saddr)–1, then PC ← PC+3+jdisp8 if (saddr)≠0	
CPU Control	**SEL**	RBn	2	8	–	RBS1,0 ← n	
	NOP		1	4	–	No Operation	
	EI		2	–	12	IE ← 1 (Enable Interrupt)	
	DI		2	–	12	IE ← 0 (Disable Interrupt)	
	HALT		2	12	–	Set HALT Mode	
	STOP		2	12	–	Set STOP Mode	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

★ **(2) μPD78044F/78044H/78064/78064Y/78064B/780208/780228 Subseries**

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**MOV**	r,#byte	2	4	–	r ← byte	
		saddr,#byte	3	6	7	(saddr) ← byte	
		sfr,#byte	3	–	7	sfr ← byte	
		A,r Note 3	1	2	–	A ← r	
		r,A Note 3	1	2	–	r ← A	
		A,saddr	2	4	5	A ← (saddr)	
		saddr,A	2	4	5	(saddr) ← A	
		A,sfr	2	–	5	A ← sfr	
		sfr,A	2	–	5	sfr ← A	
		A,!addr16	3	8	9	A ← (addr16)	
		!addr16,A	3	8	9	(addr16) ← A	
		PSW,#byte	3	–	7	PSW ← byte	× × ×
		A,PSW	2	–	5	A ← PSW	
		PSW,A	2	–	5	PSW ← A	× × ×
		A,[DE]	1	4	5	A ← (DE)	
		[DE],A	1	4	5	(DE) ← A	
		A,[HL]	1	4	5	A ← (HL)	
		[HL],A	1	4	5	(HL) ← A	
		A,[HL+byte]	2	8	9	A ← (HL+byte)	
		[HL+byte],A	2	8	9	(HL+byte) ← A	
		A,[HL+B]	1	6	7	A ← (HL+B)	
		[HL+B],A	1	6	7	(HL+B) ← A	
		A,[HL+C]	1	6	7	A ← (HL+C)	
		[HL+C],A	1	6	7	(HL+C) ← A	
	XCH	A,r Note 3	1	2	–	A ↔ r	
		A,saddr	2	4	6	A ↔ (saddr)	
		A,sfr	2	–	6	A ↔ (sfr)	
		A,!addr16	3	8	10	A ↔ (addr16)	
		A,[DE]	1	4	6	A ↔ (DE)	
		A,[HL]	1	4	6	A ↔ (HL)	
		A,[HL+byte]	2	8	10	A ↔ (HL+byte)	
		A,[HL+B]	2	8	10	A ↔ (HL+B)	
		A,[HL+C]	2	8	10	A ↔ (HL+C)	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
16-Bit Data Transfer	**MOVW**	rp,#word	3	6	–	rp ← word	
		saddrp,#word	4	8	10	(saddrp) ← word	
		sfrp,#word	4	–	10	sfrp ← word	
		AX,saddrp	2	6	8	AX ← (saddrp)	
		saddrp,AX	2	6	8	(saddrp) ← AX	
		AX,sfrp	2	–	8	AX ← sfrp	
		sfrp,AX	2	–	8	sfrp ← AX	
		AX,rp Note 3	1	4	–	AX ← rp	
		rp,AX Note 3	1	4	–	rp ← AX	
		AX,!addr16	3	10	12	AX ← (addr16)	
		!addr16,AX	3	10	12	(addr16) ← AX	
	XCHW	AX,rp Note 3	1	4	–	AX ↔ rp	
8-Bit Operation	**ADD**	A,#byte	2	4	–	A,CY ← A+byte	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)+byte	× × ×
		A,r Note 4	2	4	–	A,CY ← A+r	× × ×
		r,A	2	4	–	r,CY ← r+A	× × ×
		A,saddr	2	4	5	A,CY ← A+(saddr)	× × ×
		A,!addr16	3	8	9	A,CY ← A+(addr16)	× × ×
		A,[HL]	1	4	5	A,CY ← A+(HL)	× × ×
		A,[HL+byte]	2	8	9	A,CY ← A+(HL+byte)	× × ×
		A,[HL+B]	2	8	9	A,CY ← A+(HL+B)	× × ×
		A,[HL+C]	2	8	9	A,CY ← A+(HL+C)	× × ×
	ADDC	A,#byte	2	4	–	A,CY ← A+byte+CY	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)+byte+CY	× × ×
		A,r Note 4	2	4	–	A,CY ← A+r+CY	× × ×
		r,A	2	4	–	r,CY ← r+A+CY	× × ×
		A,saddr	2	4	5	A,CY ← A+(saddr)+CY	× × ×
		A,!addr16	3	8	9	A,CY ← A+(addr16)+CY	× × ×
		A,[HL]	1	4	5	A,CY ← A+(HL)+CY	× × ×
		A,[HL+byte]	2	8	9	A,CY ← A+(HL+byte)+CY	× × ×
		A,[HL+B]	2	8	9	A,CY ← A+(HL+B)+CY	× × ×
		A,[HL+C]	2	8	9	A,CY ← A+(HL+C)+CY	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Only when rp = BC, DE or HL.

4. Except r = A.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**SUB**	A,#byte	2	4	–	A,CY ← A–byte	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)–byte	× × ×
		A,r Note 3	2	4	–	A,CY ← A–r	× × ×
		r,A	2	4	–	r,CY ← r–A	× × ×
		A,saddr	2	4	5	A,CY ← A–(saddr)	× × ×
		A,!addr16	3	8	9	A,CY ← A–(addr16)	× × ×
		A,[HL]	1	4	5	A,CY ← A–(HL)	× × ×
		A,[HL+byte]	2	8	9	A,CY ← A–(HL+byte)	× × ×
		A,[HL+B]	2	8	9	A,CY ← A–(HL+B)	× × ×
		A,[HL+C]	2	8	9	A,CY ← A–(HL+C)	× × ×
	SUBC	A,#byte	2	4	–	A,CY ← A–byte–CY	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)–byte–CY	× × ×
		A,r Note 3	2	4	–	A,CY ← A–r–CY	× × ×
		r,A	2	4	–	r,CY ← r–A–CY	× × ×
		A,saddr	2	4	5	A,CY ← A–(saddr)–CY	× × ×
		A,!addr16	3	8	9	A,CY ← A–(addr16)–CY	× × ×
		A,[HL]	1	4	5	A,CY ← A–(HL)–CY	× × ×
		A,[HL+byte]	2	8	9	A,CY ← A–(HL+byte)–CY	× × ×
		A,[HL+B]	2	8	9	A,CY ← A–(HL+B)–CY	× × ×
		A,[HL+C]	2	8	9	A,CY ← A–(HL+C)–CY	× × ×
	AND	A,#byte	2	4	–	A ← A ∧ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ∧ byte	×
		A,r Note 3	2	4	–	A ← A ∧ r	×
		r,A	2	4	–	r ← r ∧ A	×
		A,saddr	2	4	5	A ← A ∧ (saddr)	×
		A,!addr16	3	8	9	A ← A ∧ (addr16)	×
		A,[HL]	1	4	5	A ← A ∧ (HL)	×
		A,[HL+byte]	2	8	9	A ← A ∧ (HL+byte)	×
		A,[HL+B]	2	8	9	A ← A ∧ (HL+B)	×
		A,[HL+C]	2	8	9	A ← A ∧ (HL+C)	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**OR**	A,#byte	2	4	–	A ← A ∨ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ∨ byte	×
		A,r Note 3	2	4	–	A ← A ∨ r	×
		r,A	2	4	–	r ← r ∨ A	×
		A,saddr	2	4	5	A ← A ∨ (saddr)	×
		A,!addr16	3	8	9	A ← A ∨ (addr16)	×
		A,[HL]	1	4	5	A ← A ∨ (HL)	×
		A,[HL+byte]	2	8	9	A ← A ∨ (HL+byte)	×
		A,[HL+B]	2	8	9	A ← A ∨ (HL+B)	×
		A,[HL+C]	2	8	9	A ← A ∨ (HL+C)	×
	XOR	A,#byte	2	4	–	A ← A ⊻ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ⊻ byte	×
		A,r Note 3	2	4	–	A ← A ⊻ r	×
		r,A	2	4	–	r ← r ⊻ A	×
		A,saddr	2	4	5	A ← A ⊻ (saddr)	×
		A,!addr16	3	8	9	A ← A ⊻ (addr16)	×
		A,[HL]	1	4	5	A ← A ⊻ (HL)	×
		A,[HL+byte]	2	8	9	A ← A ⊻ (HL+byte)	×
		A,[HL+B]	2	8	9	A ← A ⊻ (HL+B)	×
		A,[HL+C]	2	8	9	A ← A ⊻ (HL+C)	×
	CMP	A,#byte	2	4	–	A–byte	× × ×
		saddr,#byte	3	6	8	(saddr)–byte	× × ×
		A,r Note 3	2	4	–	A–r	× × ×
		r,A	2	4	–	r–A	× × ×
		A,saddr	2	4	5	A–(saddr)	× × ×
		A,!addr16	3	8	9	A–(addr16)	× × ×
		A,[HL]	1	4	5	A–(HL)	× × ×
		A,[HL+byte]	2	8	9	A–(HL+byte)	× × ×
		A,[HL+B]	2	8	9	A–(HL+B)	× × ×
		A,[HL+C]	2	8	9	A–(HL+C)	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

3. Except r = A.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
16-Bit Operation	**ADDW**	AX,#word	3	6	–	AX,CY ← AX+word	× × ×
	SUBW	AX,#word	3	6	–	AX,CY ← AX–word	× × ×
	CMPW	AX,#word	3	6	–	AX–word	× × ×
Multiply/ divide	**MULU**	X	2	16	–	AX ← A × X	
	DIVUW	C	2	25	–	AX (quotient), C (remainder) ← AX ÷ C	
Increment/ decrement	**INC**	r	1	2	–	r ← r+1	× ×
		saddr	2	4	6	(saddr) ← (saddr)+1	× ×
	DEC	r	1	2	–	r ← r–1	× ×
		saddr	2	4	6	(saddr) ← (saddr)–1	× ×
	INCW	rp	1	4	–	rp ← rp+1	
	DECW	rp	1	4	–	rp ← rp–1	
Rotate	**ROR**	A,1	1	2	–	$(CY, A_7 \leftarrow A_0, A_{m-1} \leftarrow A_m) \times 1$	×
	ROL	A,1	1	2	–	$(CY, A_0 \leftarrow A_7, A_{m+1} \leftarrow A_m) \times 1$	×
	RORC	A,1	1	2	–	$(CY \leftarrow A_0, A_7 \leftarrow CY, A_{m-1} \leftarrow A_m) \times 1$	×
	ROLC	A,1	1	2	–	$(CY \leftarrow A_7, A_0 \leftarrow CY, A_{m+1} \leftarrow A_m) \times 1$	×
	ROR4	[HL]	2	10	12	$A_{3-0} \leftarrow (HL)_{3-0}$, $(HL)_{7-4} \leftarrow A_{3-0}$, $(HL)_{3-0} \leftarrow (HL)_{7-4}$	
	ROL4	[HL]	2	10	12	$A_{3-0} \leftarrow (HL)_{7-4}$, $(HL)_{3-0} \leftarrow A_{3-0}$, $(HL)_{7-4} \leftarrow (HL)_{3-0}$	
BCD Adjust	**ADJBA**		2	4	–	Decimal Adjust Accumulator after Addition	× × ×
	ADJBS		2	4	–	Decimal Adjust Accumulator after Subtract	× × ×
Bit Manipulation	**MOV1**	CY,saddr.bit	3	6	7	CY ← (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← sfr.bit	×
		CY,A.bit	2	4	–	CY ← A.bit	×
		CY,PSW.bit	3	–	7	CY ← PSW.bit	×
		CY,[HL].bit	2	6	7	CY ← (HL).bit	×
		saddr.bit,CY	3	6	8	(saddr.bit) ← CY	
		sfr.bit,CY	3	–	8	sfr.bit ← CY	
		A.bit,CY	2	4	–	A.bit ← CY	
		PSW.bit,CY	3	–	8	PSW.bit ← CY	× ×
		[HL].bit,CY	2	6	8	(HL).bit ← CY	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	**Note 2**		Z AC CY
Bit Manipulation	**AND1**	CY,saddr.bit	3	6	7	CY ← CY ∧ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ∧ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ∧ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ∧ PSW.bit	×
		CY,[HL].bit	2	6	7	CY ← CY ∧ (HL).bit	×
	OR1	CY,saddr.bit	3	6	7	CY ← CY ∨ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ∨ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ∨ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ∨ PSW.bit	×
		CY,[HL].bit	2	6	7	CY ← CY ∨ (HL).bit	×
	XOR1	CY,saddr.bit	3	6	7	CY ← CY ⩛ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ⩛ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ⩛ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ⩛ PSW.bit	×
		CY,[HL].bit	2	6	7	CY ← CY ⩛ (HL).bit	×
	SET1	saddr.bit	2	4	6	(saddr.bit) ← 1	
		sfr.bit	3	–	8	sfr.bit ← 1	
		A.bit	2	4	–	A.bit ← 1	
		PSW.bit	2	–	6	PSW.bit ← 1	× × ×
		[HL].bit	2	6	8	(HL).bit ← 1	
	CLR1	saddr.bit	2	4	6	(saddr.bit) ← 0	
		sfr.bit	3	–	8	sfr.bit ← 0	
		A.bit	2	4	–	A.bit ← 0	
		PSW.bit	2	–	6	PSW.bit ← 0	× × ×
		[HL].bit	2	6	8	(HL).bit ← 0	
	SET1	CY	1	2	–	CY ← 1	1
	CLR1	CY	1	2	–	CY ← 0	0
	NOT1	CY	1	2	–	CY ← $\overline{CY}$	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Call Return	**CALL**	!addr16	3	7	–	(SP–1) ← (PC+3)H, (SP–2) ← (PC+3)L, PC ← addr16, SP ← SP–2	
	CALLF	!addr11	2	5	–	(SP–1) ← (PC+2)H, (SP–2) ← (PC+2)L, PC15–11 ← 00001, PC10–0 ← addr11, SP ← SP–2	
	CALLT	[addr5]	1	6	–	(SP–1) ← (PC+1)H, (SP–2) ← (PC+1)L, PCH ← (00000000,addr5+1), PCL ← (00000000,addr5), SP ← SP–2	
	BRK		1	6	–	(SP–1) ← PSW, (SP–2) ← (PC+1)H, (SP–3) ← (PC+1)L, PCH ← (003FH), PCL ← (003EH), SP ← SP–3, IE ← 0	
	RET		1	6	–	PCH ← (SP+1), PCL ← (SP), SP ← SP+2	
	RETI		1	6	–	PCH ← (SP+1), PCL ← (SP), PSW ← (SP+2), SP ← SP+3, NMIS ← 0	R R R
	RETB		1	6	–	PCH ← (SP+1), PCL ← (SP), PSW ← (SP+2), SP ← SP+3	R R R
Stack Manipulation	**PUSH**	PSW	1	2	–	(SP–1) ← PSW, SP ← SP–1	
		rp	1	4	–	(SP–1) ← rpH, (SP–2) ← rpL, SP ← SP–2	
	POP	PSW	1	2	–	PSW ← (SP), SP ← SP+1	R R R
		rp	1	4	–	rpH ← (SP+1), rpL ← (SP), SP ← SP+2	
	MOVW	SP,#word	4	–	10	SP ← word	
		SP, AX	2	–	8	SP ← AX	
		AX, SP	2	–	8	AX ← SP	
Unconditional Branch	**BR**	!addr16	3	6	–	PC ← addr16	
		$addr16	2	6	–	PC ← PC+2+jdisp8	
		AX	2	8	–	PCH ← A, PCL ← X	
Conditional Branch	**BC**	$addr16	2	6	–	PC ← PC+2+jdisp8 if CY = 1	
	BNC	$addr16	2	6	–	PC ← PC+2+jdisp8 if CY = 0	
	BZ	$addr16	2	6	–	PC ← PC+2+jdisp8 if Z = 1	
	BNZ	$addr16	2	6	–	PC ← PC+2+jdisp8 if Z = 0	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	**Note 2**		Z AC CY
Conditional Branch	**BT**	saddr.bit,$addr16	3	8	9	PC ← PC+3+jdisp8 if (saddr.bit) = 1	
		sfr.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if sfr.bit = 1	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit = 1	
		PSW.bit,$addr16	3	–	9	PC ← PC+3+jdisp8 if PSW.bit = 1	
		[HL].bit,$addr16	3	10	11	PC ← PC+3+jdisp8 if (HL).bit = 1	
	BF	saddr.bit,$addr16	4	10	11	PC ← PC+4+jdisp8 if (saddr.bit) = 0	
		sfr.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if sfr.bit = 0	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit = 0	
		PSW.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if PSW.bit = 0	
		[HL].bit,$addr16	3	10	11	PC ← PC+3+jdisp8 if (HL).bit = 0	
	BTCLR	saddr.bit,$addr16	4	10	12	PC ← PC+4+jdisp8 if (saddr.bit) = 1 then reset (saddr.bit)	
		sfr.bit,$addr16	4	–	12	PC ← PC+4+jdisp8 if sfr.bit = 1 then reset sfr.bit	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit = 1 then reset A.bit	
		PSW.bit,$addr16	4	–	12	PC ← PC+4+jdisp8 if PSW.bit = 1 then reset PSW.bit	× × ×
		[HL].bit,$addr16	3	10	12	PC ← PC+3+jdisp8 if (HL).bit = 1 then reset (HL).bit	
	DBNZ	B,$addr16	2	6	–	B ← B–1, then PC ← PC+2+jdisp8 if B ≠ 0	
		C,$addr16	2	6	–	C ← C–1, then PC ← PC+2+jdisp8 if C ≠ 0	
		saddr,$addr16	3	8	10	(saddr) ← (saddr)–1, then PC ← PC+3+jdisp8 if (saddr) ≠ 0	
CPU Control	**SEL**	RBn	2	4	–	RBS1, 0 ← n	
	NOP		1	2	–	No Operation	
	EI		2	–	6	IE ← 1 (Enable Interrupt)	
	DI		2	–	6	IE ← 0 (Disable Interrupt)	
	HALT		2	6	–	Set HALT Mode	
	STOP		2	6	–	Set STOP Mode	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remark 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

★ **(3) μPD78054, 78054Y, 78058F, 78058FY, 78075B, 78075BY, 78078, 78078Y, 78083, 78098, 78098B, 780018Y, 780024, 780024Y, 780034, 780034Y, 780058, 780058Y, 780308, 780308Y, 780924, 780964 Subseries**

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**MOV**	r,#byte	2	4	–	r ← byte	
		saddr,#byte	3	6	7	(saddr) ← byte	
		sfr,#byte	3	–	7	sfr ← byte	
		A,r Note 3	1	2	–	A ← r	
		r,A Note 3	1	2	–	r ← A	
		A,saddr	2	4	5	A ← (saddr)	
		saddr,A	2	4	5	(saddr) ← A	
		A,sfr	2	–	5	A ← sfr	
		sfr,A	2	–	5	sfr ← A	
		A,!addr16	3	8	9+n	A ← (addr16)	
		!addr16,A	3	8	9+m	(addr16) ← A	
		PSW,#byte	3	–	7	PSW ← byte	× × ×
		A,PSW	2	–	5	A ← PSW	
		PSW,A	2	–	5	PSW ← A	× × ×
		A,[DE]	1	4	5+n	A ← (DE)	
		[DE],A	1	4	5+m	(DE) ← A	
		A,[HL]	1	4	5+n	A ← (HL)	
		[HL],A	1	4	5+m	(HL) ← A	
		A,[HL+byte]	2	8	9+n	A ← (HL+byte)	
		[HL+byte],A	2	8	9+m	(HL+byte) ← A	
		A,[HL+B]	1	6	7+n	A ← (HL+B)	
		[HL+B],A	1	6	7+m	(HL+B) ← A	
		A,[HL+C]	1	6	7+n	A ← (HL+C)	
		[HL+C],A	1	6	7+m	(HL+C) ← A	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.
4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**XCH**	A,r Note 4	1	2	–	A ↔ r	
		A,saddr	2	4	6	A ↔ (saddr)	
		A,sfr	2	–	6	A ↔ sfr	
		A,!addr16	3	8	10+n+m	A ↔ (addr16)	
		A,[DE]	1	4	6+n+m	A ↔ (DE)	
		A,[HL]	1	4	6+n+m	A ↔ (HL)	
		A,[HL+byte]	2	8	10+n+m	A ↔ (HL+byte)	
		A,[HL+B]	2	8	10+n+m	A ↔ (HL+B)	
		A,[HL+C]	2	8	10+n+m	A ↔ (HL+C)	
16-Bit Data Transfer	**MOVW**	rp,#word	3	6	–	rp ← word	
		saddrp,#word	4	8	10	(saddrp) ← word	
		sfrp,#word	4	–	10	sfrp ← word	
		AX,saddrp	2	6	8	AX ← (saddrp)	
		saddrp,AX	2	6	8	(saddrp) ← AX	
		AX,sfrp	2	–	8	AX ← sfrp	
		sfrp,AX	2	–	8	sfrp ← AX	
		AX,rp Note 3	1	4	–	AX ← rp	
		rp,AX Note 3	1	4	–	rp ← AX	
		AX,!addr16	3	10	12+2n	AX ← (addr16)	
		!addr16,AX	3	10	12+2m	(addr16) ← AX	
	XCHW	AX,rp Note 3	1	4	–	AX ↔ rp	
8-Bit Operation	**ADD**	A,#byte	2	4	–	A,CY ← A+byte	× × ×
		saddr,#byte	3	6	8	(saddr), CY ← (saddr)+byte	× × ×
		A,r Note 4	2	4	–	A,CY ← A+r	× × ×
		r,A	2	4	–	r,CY ← r+A	× × ×
		A,saddr	2	4	5	A,CY ← A+(saddr)	× × ×
		A,!addr16	3	8	9+n	A,CY ← A+(addr16)	× × ×
		A,[HL]	1	4	5+n	A,CY ← A+(HL)	× × ×
		A,[HL+byte]	2	8	9+n	A,CY ← A+(HL+byte)	× × ×
		A,[HL+B]	2	8	9+n	A,CY ← A+(HL+B)	× × ×
		A,[HL+C]	2	8	9+n	A,CY ← A+(HL+C)	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Only when rp = BC, DE or HL.
4. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.
4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**ADDC**	A,#byte	2	4	–	A,CY ← A+byte+CY	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)+byte+CY	× × ×
		A,r Note 3	2	4	–	A,CY ← A+r+CY	× × ×
		r,A	2	4	–	r,CY ← r+A+CY	× × ×
		A,saddr	2	4	5	A,CY ← A+(saddr)+CY	× × ×
		A,!addr16	3	8	9+n	A,CY ← A+(addr16)+CY	× × ×
		A,[HL]	1	4	5+n	A,CY ← A+(HL)+CY	× × ×
		A,[HL+byte]	2	8	9+n	A,CY ← A+(HL+byte)+CY	× × ×
		A,[HL+B]	2	8	9+n	A,CY ← A+(HL+B)+CY	× × ×
		A,[HL+C]	2	8	9+n	A,CY ← A+(HL+C)+CY	× × ×
	SUB	A,#byte	2	4	–	A,CY ← A–byte	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)–byte	× × ×
		A,r Note 3	2	4	–	A,CY ← A–r	× × ×
		r,A	2	4	–	r,CY ← r–A	× × ×
		A,saddr	2	4	5	A,CY ← A–(saddr)	× × ×
		A,!addr16	3	8	9+n	A,CY ← A–(addr16)	× × ×
		A,[HL]	1	4	5+n	A,CY ← A–(HL)	× × ×
		A,[HL+byte]	2	8	9+n	A,CY ← A–(HL+byte)	× × ×
		A,[HL+B]	2	8	9+n	A,CY ← A–(HL+B)	× × ×
		A,[HL+C]	2	8	9+n	A,CY ← A–(HL+C)	× × ×
	SUBC	A,#byte	2	4	–	A,CY ← A–byte–CY	× × ×
		saddr,#byte	3	6	8	(saddr),CY ← (saddr)–byte–CY	× × ×
		A,r Note 3	2	4	–	A,CY ← A–r–CY	× × ×
		r,A	2	4	–	r,CY ← r–A–CY	× × ×
		A,saddr	2	4	5	A,CY ← A–(saddr)–CY	× × ×
		A,!addr16	3	8	9+n	A,CY ← A–(addr16)–CY	× × ×
		A,[HL]	1	4	5+n	A,CY ← A–(HL)–CY	× × ×
		A,[HL+byte]	2	8	9+n	A,CY ← A–(HL+byte)–CY	× × ×
		A,[HL+B]	2	8	9+n	A,CY ← A–(HL+B)–CY	× × ×
		A,[HL+C]	2	8	9+n	A,CY ← A–(HL+C)–CY	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**AND**	A,#byte	2	4	–	A ← A ∧ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ∧ byte	×
		A,r Note 3	2	4	–	A ← A ∧ r	×
		r,A	2	4	–	r ← r ∧ A	×
		A,saddr	2	4	5	A ← A ∧ (saddr)	×
		A,!addr16	3	8	9+n	A ← A ∧ (addr16)	×
		A,[HL]	1	4	5+n	A ← A ∧ (HL)	×
		A,[HL+byte]	2	8	9+n	A ← A ∧ (HL+byte)	×
		A,[HL+B]	2	8	9+n	A ← A ∧ (HL+B)	×
		A,[HL+C]	2	8	9+n	A ← A ∧ (HL+C)	×
	OR	A,#byte	2	4	–	A ← A ∨ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ∨ byte	×
		A,r Note 3	2	4	–	A ← A ∨ r	×
		r,A	2	4	–	r ← r ∨ A	×
		A,saddr	2	4	5	A ← A ∨ (saddr)	×
		A,!addr16	3	8	9+n	A ← A ∨ (addr16)	×
		A,[HL]	1	4	5+n	A ← A ∨ (HL)	×
		A,[HL+byte]	2	8	9+n	A ← A ∨ (HL+byte)	×
		A,[HL+B]	2	8	9+n	A ← A ∨ (HL+B)	×
		A,[HL+C]	2	8	9+n	A ← A ∨ (HL+C)	×
	XOR	A,#byte	2	4	–	A ← A ⩛ byte	×
		saddr,#byte	3	6	8	(saddr) ← (saddr) ⩛ byte	×
		A,r Note 3	2	4	–	A ← A ⩛ r	×
		r,A	2	4	–	r ← r ⩛ A	×
		A,saddr	2	4	5	A ← A ⩛ (saddr)	×
		A,!addr16	3	8	9+n	A ← A ⩛ (addr16)	×
		A,[HL]	1	4	5+n	A ← A ⩛ (HL)	×
		A,[HL+byte]	2	8	9+n	A ← A ⩛ (HL+byte)	×
		A,[HL+B]	2	8	9+n	A ← A ⩛ (HL+B)	×
		A,[HL+C]	2	8	9+n	A ← A ⩛ (HL+C)	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**CMP**	A,#byte	2	4	–	A–byte	× × ×
		saddr,#byte	3	6	8	(saddr)–byte	× × ×
		A,r Note 3	2	4	–	A–r	× × ×
		r,A	2	4	–	r–A	× × ×
		A,saddr	2	4	5	A–(saddr)	× × ×
		A,!addr16	3	8	9+n	A–(addr16)	× × ×
		A,[HL]	1	4	5+n	A–(HL)	× × ×
		A,[HL+byte]	2	8	9+n	A–(HL+byte)	× × ×
		A,[HL+B]	2	8	9+n	A–(HL+B)	× × ×
		A,[HL+C]	2	8	9+n	A–(HL+C)	× × ×
16-Bit Operation	**ADDW**	AX,#word	3	6	–	AX,CY ← AX+word	× × ×
	SUBW	AX,#word	3	6	–	AX,CY ← AX–word	× × ×
	CMPW	AX,#word	3	6	–	AX–word	× × ×
Multiply/ divide	**MULU**	X	2	16	–	AX ← A × X	
	DIVUW	C	2	25	–	AX (quotient), C (remainder) ← AX ÷ C	
Increment/ decrement	**INC**	r	1	2	–	r ← r+1	× ×
		saddr	2	4	6	(saddr) ← (saddr)+1	× ×
	DEC	r	1	2	–	r ← r–1	× ×
		saddr	2	4	6	(saddr) ← (saddr)–1	× ×
	INCW	rp	1	4	–	rp ← rp+1	
	DECW	rp	1	4	–	rp ← rp–1	
Rotate	**ROR**	A,1	1	2	–	(CY, $A_7 \leftarrow A_0$, $A_{m-1} \leftarrow A_m$) × 1	×
	ROL	A,1	1	2	–	(CY, $A_0 \leftarrow A_7$, $A_{m+1} \leftarrow A_m$) × 1	×
	RORC	A,1	1	2	–	(CY ← A_0, $A_7 \leftarrow$ CY, $A_{m-1} \leftarrow A_m$) × 1	×
	ROLC	A,1	1	2	–	(CY ← A_7, $A_0 \leftarrow$ CY, $A_{m+1} \leftarrow A_m$) × 1	×
	ROR4	[HL]	2	10	12+n+m	$A_{3-0} \leftarrow (HL)_{3-0}$, $(HL)_{7-4} \leftarrow A_{3-0}$, $(HL)_{3-0} \leftarrow (HL)_{7-4}$	
	ROL4	[HL]	2	10	12+n+m	$A_{3-0} \leftarrow (HL)_{7-4}$, $(HL)_{3-0} \leftarrow A_{3-0}$, $(HL)_{7-4} \leftarrow (HL)_{3-0}$	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.
4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	**Note 2**		Z AC CY
BCD	**ADJBA**		2	4	–	Decimal Adjust Accumulator after Addition	× × ×
Adjust	**ADJBS**		2	4	–	Decimal Adjust Accumulator after Subtract	× × ×
Bit	**MOV1**	CY,saddr.bit	3	6	7	CY ← (saddr.bit)	×
Manipulation		CY,sfr.bit	3	–	7	CY ← sfr.bit	×
		CY,A.bit	2	4	–	CY ← A.bit	×
		CY,PSW.bit	3	–	7	CY ← PSW.bit	×
		CY,[HL].bit	2	6	7+n	CY ← (HL).bit	×
		saddr.bit,CY	3	6	8	(saddr.bit) ← CY	
		sfr.bit,CY	3	–	8	sfr.bit ← CY	
		A.bit,CY	2	4	–	A.bit ← CY	
		PSW.bit,CY	3	–	8	PSW.bit ← CY	× ×
		[HL].bit,CY	2	6	8+n+m	(HL).bit ← CY	
	AND1	CY,saddr.bit	3	6	7	CY ← CY ∧ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ∧ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ∧ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ∧ PSW.bit	×
		CY,[HL].bit	2	6	7+n	CY ← CY ∧ (HL).bit	×
	OR1	CY,saddr.bit	3	6	7	CY ← CY ∨ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ∨ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ∨ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ∨ PSW.bit	×
		CY,[HL].bit	2	6	7+n	CY ← CY ∨ (HL).bit	×
	XOR1	CY,saddr.bit	3	6	7	CY ← CY ⩛ (saddr.bit)	×
		CY,sfr.bit	3	–	7	CY ← CY ⩛ sfr.bit	×
		CY,A.bit	2	4	–	CY ← CY ⩛ A.bit	×
		CY,PSW.bit	3	–	7	CY ← CY ⩛ PSW.bit	×
		CY,[HL].bit	2	6	7+n	CY ← CY ⩛ (HL).bit	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Bit Manipulation	**SET1**	saddr.bit	2	4	6	(saddr.bit) ← 1	
		sfr.bit	3	–	8	sfr.bit ← 1	
		A.bit	2	4	–	A.bit ← 1	
		PSW.bit	2	–	6	PSW.bit ← 1	× × ×
		[HL].bit	2	6	8+n+m	(HL).bit ← 1	
	CLR1	saddr.bit	2	4	6	(saddr.bit) ← 0	
		sfr.bit	3	–	8	sfr.bit ← 0	
		A.bit	2	4	–	A.bit ← 0	
		PSW.bit	2	–	6	PSW.bit ← 0	× × ×
		[HL].bit	2	6	8+n+m	(HL).bit ← 0	
	SET1	CY	1	2	–	CY ← 1	1
	CLR1	CY	1	2	–	CY ← 0	0
	NOT1	CY	1	2	–	CY ← CY	×
Call Return	**CALL**	!addr16	3	7	–	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, PC ← addr16, SP ← SP–2	
	CALLF	!addr11	2	5	–	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_{15-11} \leftarrow 00001$, $PC_{10-0} \leftarrow$ addr11, SP ← SP–2	
	CALLT	[addr5]	1	6	–	$(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (00000000,addr5+1), $PC_L \leftarrow$ (00000000, addr5), SP ← SP–2	
	BRK		1	6	–	$(SP-1) \leftarrow$ PSW, $(SP-2) \leftarrow (PC+1)_H$, $(SP-3) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (003FH), $PC_L \leftarrow$ (003EH), SP ← SP–3, IE ← 0	
	RET		1	6	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), SP ← SP+2	
	RETI		1	6	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), PSW ← (SP+2), SP ← SP+3, NMIS ← 0	R R R
	RETB		1	6	–	$PC_H \leftarrow (SP+1)$, $PC_L \leftarrow$ (SP), PSW ← (SP+2), SP ← SP+3	R R R

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.

2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

2. Number of clock cycles is when there is a program in the internal ROM area.

3. n indicates the number of waits when the external memory expansion area is read.

4. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Stack Manipulation	**PUSH**	PSW	1	2	–	(SP–1) ← PSW, SP ← SP–1	
		rp	1	4	–	(SP–1) ← rp_H, (SP–2) ← rp_L, SP ← SP–2	
	POP	PSW	1	2	–	PSW ← (SP), SP ← SP+1	R R R
		rp	1	4	–	rp_H ← (SP+1), rp_L ← (SP), SP ← SP+2	
	MOVW	SP,#word	4	–	10	SP ← word	
		SP,AX	2	–	8	SP ← AX	
		AX,SP	2	–	8	AX ← SP	
Unconditional Branch	**BR**	!addr16	3	6	–	PC ← addr16	
		$addr16	2	6	–	PC ← PC+2+jdisp8	
		AX	2	8	–	PC_H ← A, PC_L ← X	
Conditional Branch	**BC**	$addr16	2	6	–	PC ← PC+2+jdisp8 if CY=1	
	BNC	$addr16	2	6	–	PC ← PC+2+jdisp8 if CY=0	
	BZ	$addr16	2	6	–	PC ← PC+2+jdisp8 if Z=1	
	BNZ	$addr16	2	6	–	PC ← PC+2+jdisp8 if Z=0	
	BT	saddr.bit,$addr16	3	8	9	PC ← PC+3+jdisp8 if (saddr.bit)=1	
		sfr.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if sfr.bit=1	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit=1	
		PSW.bit,$addr16	3	–	9	PC ← PC+3+jdisp8 if PSW.bit=1	
		[HL].bit,$addr16	3	10	11+n	PC ← PC+3+jdisp8 if (HL).bit=1	
	BF	saddr.bit,$addr16	4	10	11	PC ← PC+4+jdisp8 if (saddr.bit)=0	
		sfr.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if sfr.bit=0	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit=0	
		PSW.bit,$addr16	4	–	11	PC ← PC+4+jdisp8 if PSW.bit=0	
		[HL].bit,$addr16	3	10	11+n	PC ← PC+3+jdisp8 if (HL).bit=0	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Conditional Branch	**BTCLR**	saddr.bit,$addr16	4	10	12	PC ← PC+4+jdisp8 if (saddr.bit)=1 then reset (saddr.bit)	
		sfr.bit,$addr16	4	–	12	PC ← PC+4+jdisp8 if sfr.bit=1 then reset sfr.bit	
		A.bit,$addr16	3	8	–	PC ← PC+3+jdisp8 if A.bit=1 then reset A.bit	
		PSW.bit,$addr16	4	–	12	PC ← PC+4+jdisp8 if PSW.bit=1 then reset PSW.bit	
		[HL].bit,$addr16	3	10	12+n+m	PC ← PC+3+jdisp8 if (HL).bit=1 then reset (HL).bit	
	DBNZ	B,$addr16	2	6	–	B ← B–1, then PC ← PC+2+jdisp8 if B≠0	
		C,$addr16	2	6	–	C ← C–1, then PC ← PC+2+jdisp8 if C≠0	
		saddr,$addr16	3	8	10	(saddr) ← (saddr)–1, then PC ← PC+3+jdisp8 if (saddr)≠0	
CPU Control	**SEL**	RBn	2	4	–	RBS1,0 ← n	× × ×
	NOP		1	2	–	No Operation	
	EI		2	–	6	IE ← 1 (Enable Interrupt)	
	DI		2	–	6	IE ← 0 (Disable Interrupt)	
	HALT		2	6	–	Set HALT Mode	
	STOP		2	6	–	Set STOP Mode	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. Number of clock cycles is when there is a program in the internal ROM area.
3. n indicates the number of waits when the external memory expansion area is read.
4. m indicates the number of waits when the external memory expansion area is written to.

★ **(4) μPD78070A, 78070AY**

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Data Transfer	**MOV**	r,#byte	2	4+2n	–	r ← byte	
		saddr,#byte	3	6+3n	7+3n	(saddr) ← byte	
		sfr,#byte	3	–	7+3n	sfr ← byte	
		A,r Note 3	1	2+n	–	A ← r	
		r,A Note 3	1	2+n	–	r ← A	
		A,saddr	2	4+2n	5+2n	A ← (saddr)	
		saddr,A	2	4+2n	5+2n	(saddr) ← A	
		A,sfr	2	–	5+2n	A ← sfr	
		sfr,A	2	–	5+2n	sfr ← A	
		A,!addr16	3	8+3n	9+4n	A ← (addr16)	
		!addr16,A	3	8+3n	9+3n+m	(addr16) ← A	
		PSW,#byte	3	–	7+3n	PSW ← byte	× × ×
		A,PSW	2	–	5+2n	A ← PSW	
		PSW,A	2	–	5+2n	PSW ← A	× × ×
		A,[DE]	1	4+n	5+2n	A ← (DE)	
		[DE],A	1	4+n	5+n+m	(DE) ← A	
		A,[HL]	1	4+n	5+2n	A ← (HL)	
		[HL],A	1	4+n	5+n+m	(HL) ← A	
		A,[HL+byte]	2	8+2n	9+3n	A ← (HL+byte)	
		[HL+byte],A	2	8+2n	9+2n+m	(HL+byte) ← A	
		A,[HL+B]	1	6+n	7+2n	A ← (HL+B)	
		[HL+B],A	1	6+n	7+n+m	(HL+B) ← A	
		A,[HL+C]	1	6+n	7+2n	A ← (HL+C)	
		[HL+C],A	1	6+n	7+n+m	(HL+C) ← A	
	XCH	A,r Note 3	1	2+n	–	A ↔ r	
		A,saddr	2	4+2n	6+2n	A ↔ (saddr)	
		A,sfr	2	–	6+2n	A ↔ (sfr)	
		A,!addr16	3	8+3n	10+4n+m	A ↔ (addr16)	
		A,[DE]	1	4+n	6+2n+m	A ↔ (DE)	
		A,[HL]	1	4+n	6+2n+m	A ↔ (HL)	
		A,[HL+byte]	2	8+2n	10+3n+m	A ↔ (HL+byte)	
		A,[HL+B]	2	8+2n	10+3n+m	A ↔ (HL+B)	
		A,[HL+C]	2	8+2n	10+3n+m	A ↔ (HL+C)	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.
3. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
16-Bit Data Transfer	**MOVW**	rp,#word	3	6+3n	–	rp ← word	
		saddrp,#word	4	8+4n	10+4n	(saddrp) ← word	
		sfrp,#word	4	–	10+4n	sfrp ← word	
		AX,saddrp	2	6+2n	8+2n	AX ← (saddrp)	
		saddrp,AX	2	6+2n	8+2n	(saddrp) ← AX	
		AX,sfrp	2	–	8+2n	AX ← sfrp	
		sfrp,AX	2	–	8+2n	sfrp ← AX	
		AX,rp Note 3	1	4+n	–	AX ← rp	
		rp,AX Note 3	1	4+n	–	rp ← AX	
		AX,!addr16	3	10+3n	12+5n	AX ← (addr16)	
		!addr16,AX	3	10+3n	12+2m+3n	(addr16) ← AX	
	XCHW	AX,rp Note 3	1	4+n	–	AX ↔ rp	
8-Bit Operation	**ADD**	A,#byte	2	4+2n	–	A,CY ← A+byte	× × ×
		saddr,#byte	3	6+3n	8+3n	(saddr), CY ← (saddr)+byte	× × ×
		A,r Note 4	2	4+2n	–	A,CY ← A+r	× × ×
		r,A	2	4+2n	–	r,CY ← r+A	× × ×
		A,saddr	2	4+2n	5+2n	A,CY ← A+(saddr)	× × ×
		A,!addr16	3	8+3n	9+4n	A,CY ← A+(addr16)	× × ×
		A,[HL]	1	4+n	5+2n	A,CY ← A+(HL)	× × ×
		A,[HL+byte]	2	8+2n	9+3n	A,CY ← A+(HL+byte)	× × ×
		A,[HL+B]	2	8+2n	9+3n	A,CY ← A+(HL+B)	× × ×
		A,[HL+C]	2	8+2n	9+3n	A,CY ← A+(HL+C)	× × ×
	ADDC	A,#byte	2	4+2n	–	A,CY ← A+byte+CY	× × ×
		saddr,#byte	3	6+3n	8+3n	(saddr),CY ← (saddr)+byte+CY	× × ×
		A,r Note 4	2	4+2n	–	A,CY ← A+r+CY	× × ×
		r,A	2	4+2n	–	r,CY ← r+A+CY	× × ×
		A,saddr	2	4+2n	5+2n	A,CY ← A+(saddr)+CY	× × ×
		A,!addr16	3	8+3n	9+4n	A,CY ← A+(addr16)+CY	× × ×
		A,[HL]	1	4+n	5+2n	A,CY ← A+(HL)+CY	× × ×
		A,[HL+byte]	2	8+2n	9+3n	A,CY ← A+(HL+byte)+CY	× × ×
		A,[HL+B]	2	8+2n	9+3n	A,CY ← A+(HL+B)+CY	× × ×
		A,[HL+C]	2	8+2n	9+3n	A,CY ← A+(HL+C)+CY	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Only when rp = BC, DE, or HL
4. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.
3. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**SUB**	A,#byte	2	4+2n	–	A,CY ← A–byte	× × ×
		saddr,#byte	3	6+3n	8+3n	(saddr),CY ← (saddr)–byte	× × ×
		A,r Note 3	2	4+2n	–	A,CY ← A–r	× × ×
		r,A	2	4+2n	–	r,CY ← r–A	× × ×
		A,saddr	2	4+2n	5+2n	A,CY ← A–(saddr)	× × ×
		A,!addr16	3	8+3n	9+4n	A,CY ← A–(addr16)	× × ×
		A,[HL]	1	4+n	5+2n	A,CY ← A–(HL)	× × ×
		A,[HL+byte]	2	8+2n	9+3n	A,CY ← A–(HL+byte)	× × ×
		A,[HL+B]	2	8+2n	9+3n	A,CY ← A–(HL+B)	× × ×
		A,[HL+C]	2	8+2n	9+3n	A,CY ← A–(HL+C)	× × ×
	SUBC	A,#byte	2	4+2n	–	A,CY ← A–byte–CY	× × ×
		saddr,#byte	3	6+3n	8+3n	(saddr),CY ← (saddr)–byte–CY	× × ×
		A,r Note 3	2	4+2n	–	A,CY ← A–r–CY	× × ×
		r,A	2	4+2n	–	r,CY ← r–A–CY	× × ×
		A,saddr	2	4+2n	5+2n	A,CY ← A–(saddr)–CY	× × ×
		A,!addr16	3	8+3n	9+4n	A,CY ← A–(addr16)–CY	× × ×
		A,[HL]	1	4+n	5+2n	A,CY ← A–(HL)–CY	× × ×
		A,[HL+byte]	2	8+2n	9+3n	A,CY ← A–(HL+byte)–CY	× × ×
		A,[HL+B]	2	8+2n	9+3n	A,CY ← A–(HL+B)–CY	× × ×
		A,[HL+C]	2	8+2n	9+3n	A,CY ← A–(HL+C)–CY	× × ×
	AND	A,#byte	2	4+2n	–	A ← A ∧ byte	×
		saddr,#byte	3	6+3n	8+3n	(saddr) ← (saddr) ∧ byte	×
		A,r Note 3	2	4+2n	–	A ← A ∧ r	×
		r,A	2	4+2n	–	r ← r ∧ A	×
		A,saddr	2	4+2n	5+2n	A ← A ∧ (saddr)	×
		A,!addr16	3	8+3n	9+4n	A ← A ∧ (addr16)	×
		A,[HL]	1	4+n	5+2n	A ← A ∧ (HL)	×
		A,[HL+byte]	2	8+2n	9+3n	A ← A ∧ (HL+byte)	×
		A,[HL+B]	2	8+2n	9+3n	A ← A ∧ (HL+B)	×
		A,[HL+C]	2	8+2n	9+3n	A ← A ∧ (HL+C)	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
8-Bit Operation	**OR**	A,#byte	2	4+2n	–	A ← A ∨ byte	×
		saddr,#byte	3	6+3n	8+3n	(saddr) ← (saddr) ∨ byte	×
		A,r Note 3	2	4+2n	–	A ← A ∨ r	×
		r,A	2	4+2n	–	r ← r ∨ A	×
		A,saddr	2	4+2n	5+2n	A ← A ∨ (saddr)	×
		A,!addr16	3	8+3n	9+4n	A ← A ∨ (addr16)	×
		A,[HL]	1	4+n	5+2n	A ← A ∨ (HL)	×
		A,[HL+byte]	2	8+2n	9+3n	A ← A ∨ (HL+byte)	×
		A,[HL+B]	2	8+2n	9+3n	A ← A ∨ (HL+B)	×
		A,[HL+C]	2	8+2n	9+3n	A ← A ∨ (HL+C)	×
	XOR	A,#byte	2	4+2n	–	A ← A ⩛ byte	×
		saddr,#byte	3	6+3n	8+3n	(saddr) ← (saddr) ⩛ byte	×
		A,r Note 3	2	4+2n	–	A ← A ⩛ r	×
		r,A	2	4+2n	–	r ← r ⩛ A	×
		A,saddr	2	4+2n	5+2n	A ← A ⩛ (saddr)	×
		A,!addr16	3	8+3n	9+4n	A ← A ⩛ (addr16)	×
		A,[HL]	1	4+n	5+2n	A ← A ⩛ (HL)	×
		A,[HL+byte]	2	8+2n	9+3n	A ← A ⩛ (HL+byte)	×
		A,[HL+B]	2	8+2n	9+3n	A ← A ⩛ (HL+B)	×
		A,[HL+C]	2	8+2n	9+3n	A ← A ⩛ (HL+C)	×
	CMP	A,#byte	2	4+2n	–	A–byte	× × ×
		saddr,#byte	3	6+3n	8+3n	(saddr)–byte	× × ×
		A,r Note 3	2	4+2n	–	A–r	× × ×
		r,A	2	4+2n	–	r–A	× × ×
		A,saddr	2	4+2n	5+2n	A–(saddr)	× × ×
		A,!addr16	3	8+3n	9+4n	A–(addr16)	× × ×
		A,[HL]	1	4+n	5+2n	A–(HL)	× × ×
		A,[HL+byte]	2	8+2n	9+3n	A–(HL+byte)	× × ×
		A,[HL+B]	2	8+2n	9+3n	A–(HL+B)	× × ×
		A,[HL+C]	2	8+2n	9+3n	A–(HL+C)	× × ×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.
3. Except r = A.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	**Note 2**		Z AC CY
16-Bit Operation	**ADDW**	AX,#word	3	6+3n	–	AX,CY ← AX+word	× × ×
	SUBW	AX,#word	3	6+3n	–	AX,CY ← AX–word	× × ×
	CMPW	AX,#word	3	6+3n	–	AX–word	× × ×
Multiply/ divide	**MULU**	X	2	16+2n	–	AX ← A × X	
	DIVUW	C	2	25+2n	–	AX (quotient), C (remainder) ← AX ÷ C	
Increment/ decrement	**INC**	r	1	2+n	–	r ← r+1	× ×
		saddr	2	4+2n	6+2n	(saddr) ← (saddr)+1	× ×
	DEC	r	1	2+n	–	r ← r–1	× ×
		saddr	2	4+2n	6+2n	(saddr) ← (saddr)–1	× ×
	INCW	rp	1	4+n	–	rp ← rp+1	
	DECW	rp	1	4+n	–	rp ← rp–1	
Rotate	**ROR**	A,1	1	2+n	–	$(CY, A_7 \leftarrow A_0, A_{m-1} \leftarrow A_m) \times 1$	×
	ROL	A,1	1	2+n	–	$(CY, A_0 \leftarrow A_7, A_{m+1} \leftarrow A_m) \times 1$	×
	RORC	A,1	1	2+n	–	$(CY \leftarrow A_0, A_7 \leftarrow CY, A_{m-1} \leftarrow A_m) \times 1$	×
	ROLC	A,1	1	2+n	–	$(CY \leftarrow A_7, A_0 \leftarrow CY, A_{m+1} \leftarrow A_m) \times 1$	×
	ROR4	[HL]	2	10+2n	12+3n+m	$A_{3-0} \leftarrow (HL)_{3-0}$, $(HL)_{7-4} \leftarrow A_{3-0}$, $(HL)_{3-0} \leftarrow (HL)_{7-4}$	
	ROL4	[HL]	2	10+2n	12+3n+m	$A_{3-0} \leftarrow (HL)_{7-4}$, $(HL)_{3-0} \leftarrow A_{3-0}$, $(HL)_{7-4} \leftarrow (HL)_{3-0}$	
BCD Adjust	**ADJBA**		2	4+2n	–	Decimal Adjust Accumulator after Addition	× × ×
	ADJBS		2	4+2n	–	Decimal Adjust Accumulator after Subtract	× × ×
Bit Manipulation	**MOV1**	CY,saddr.bit	3	6+3n	7+3n	CY ← (saddr.bit)	×
		CY,sfr.bit	3	–	7+3n	CY ← sfr.bit	×
		CY,A.bit	2	4+2n	–	CY ← A.bit	×
		CY,PSW.bit	3	–	7+3n	CY ← PSW.bit	×
		CY,[HL].bit	2	6+2n	7+3n	CY ← (HL).bit	×
		saddr.bit,CY	3	6+3n	8+3n	(saddr.bit) ← CY	
		sfr.bit,CY	3	–	8+3n	sfr.bit ← CY	
		A.bit,CY	2	4+2n	–	A.bit ← CY	
		PSW.bit,CY	3	–	8+3n	PSW.bit ← CY	× ×
		[HL].bit,CY	2	6+2n	8+3n+m	(HL).bit ← CY	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.
3. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Bit Manipulation	**AND1**	CY,saddr.bit	3	6+3n	7+3n	CY ← CY ∧ (saddr.bit)	×
		CY,sfr.bit	3	–	7+3n	CY ← CY ∧ sfr.bit	×
		CY,A.bit	2	4+2n	–	CY ← CY ∧ A.bit	×
		CY,PSW.bit	3	–	7+3n	CY ← CY ∧ PSW.bit	×
		CY,[HL].bit	2	6+2n	7+3n	CY ← CY ∧ (HL).bit	×
	OR1	CY,saddr.bit	3	6+3n	7+3n	CY ← CY ∨ (saddr.bit)	×
		CY,sfr.bit	3	–	7+3n	CY ← CY ∨ sfr.bit	×
		CY,A.bit	2	4+2n	–	CY ← CY ∨ A.bit	×
		CY,PSW.bit	3	–	7+3n	CY ← CY ∨ PSW.bit	×
		CY,[HL].bit	2	6+2n	7+3n	CY ← CY ∨ (HL).bit	×
	XOR1	CY,saddr.bit	3	6+3n	7+3n	CY ← CY ⩛ (saddr.bit)	×
		CY,sfr.bit	3	–	7+3n	CY ← CY ⩛ sfr.bit	×
		CY,A.bit	2	4+2n	–	CY ← CY ⩛ A.bit	×
		CY,PSW.bit	3	–	7+3n	CY ← CY ⩛ PSW.bit	×
		CY,[HL].bit	2	6+2n	7+3n	CY ← CY ⩛ (HL).bit	×
	SET1	saddr.bit	2	4+2n	6+2n	(saddr.bit) ← 1	
		sfr.bit	3	–	8+3n	sfr.bit ← 1	
		A.bit	2	4+2n	–	A.bit ← 1	
		PSW.bit	2	–	6+2n	PSW.bit ← 1	× × ×
		[HL].bit	2	6+2n	8+3n+m	(HL).bit ← 1	
	CLR1	saddr.bit	2	4+2n	6+2n	(saddr.bit) ← 0	
		sfr.bit	3	–	8+3n	sfr.bit ← 0	
		A.bit	2	4+2n	–	A.bit ← 0	
		PSW.bit	2	–	6+2n	PSW.bit ← 0	× × ×
		[HL].bit	2	6+2n	8+3n+m	(HL).bit ← 0	
	SET1	CY	1	2+n	–	CY ← 1	1
	CLR1	CY	1	2+n	–	CY ← 0	0
	NOT1	CY	1	2+n	–	CY ← $\overline{CY}$	×

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.
3. m indicates the number of waits when the external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Call Return	**CALL**	!addr16	3	7+3n	–	(SP–1) ← (PC+3)$_H$, (SP–2) ← (PC+3)$_L$, PC ← addr16, SP ← SP–2	
	CALLF	!addr11	2	5+2n	–	(SP–1) ← (PC+2)$_H$, (SP–2) ← (PC+2)$_L$, PC$_{15–11}$ ← 00001, PC$_{10–0}$ ← addr11, SP ← SP–2	
	CALLT	[addr5]	1	6+n	–	(SP–1) ← (PC+1)$_H$, (SP–2) ← (PC+1)$_L$, PC$_H$ ← (00000000,addr5+1), PC$_L$ ← (00000000, addr5), SP ← SP–2	
	BRK		1	6+n	–	(SP–1) ← PSW, (SP–2) ← (PC+1)$_H$, (SP–3) ← (PC+1)$_L$, PC$_H$ ← (003FH), PC$_L$ ← (003EH), SP ← SP–3, IE ← 0	
	RET		1	6+n	–	PC$_H$ ← (SP+1), PC$_L$ ← (SP), SP ← SP+2	
	RETI		1	6+n	–	PC$_H$ ← (SP+1), PC$_L$ ← (SP), PSW ← (SP+2), SP ← SP+3, NMIS ← 0	R R R
	RETB		1	6+n	–	PC$_H$ ← (SP+1), PC$_L$ ← (SP), PSW ← (SP+2), SP ← SP+3	R R R
Stack Manipulation	**PUSH**	PSW	1	2+n	–	(SP–1) ← PSW, SP ← SP–1	
		rp	1	4+n	–	(SP–1) ← rp$_H$, (SP–2) ← rp$_L$, SP ← SP–2	
	POP	PSW	1	2+n	–	PSW ← (SP), SP ← SP+1	R R R
		rp	1	4+n	–	rp$_H$ ← (SP+1), rp$_L$ ← (SP), SP ← SP+2	
	MOVW	SP,#word	4	–	10+4n	SP ← word	
		SP,AX	2	–	8+2n	SP ← AX	
		AX,SP	2	–	8+2n	AX ← SP	
Unconditional Branch	**BR**	!addr16	3	6+3n	–	PC ← addr16	
		$addr16	2	6+2n	–	PC ← PC+2+jdisp8	
		AX	2	8+2n	–	PC$_H$ ← A, PC$_L$ ← X	
Conditional Branch	**BC**	$addr16	2	6+2n	–	PC ← PC+2+jdisp8 if CY=1	
	BNC	$addr16	2	6+2n	–	PC ← PC+2+jdisp8 if CY=0	
	BZ	$addr16	2	6+2n	–	PC ← PC+2+jdisp8 if Z=1	
	BNZ	$addr16	2	6+2n	–	PC ← PC+2+jdisp8 if Z=0	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag
				Note 1	Note 2		Z AC CY
Conditional Branch	**BT**	saddr.bit,$addr16	3	8+3n	9+3n	PC ← PC+3+jdisp8 if (saddr.bit)=1	
		sfr.bit,$addr16	4	–	11+4n	PC ← PC+4+jdisp8 if sfr.bit=1	
		A.bit,$addr16	3	8+3n	–	PC ← PC+3+jdisp8 if A.bit=1	
		PSW.bit,$addr16	3	–	9+3n	PC ← PC+3+jdisp8 if PSW.bit=1	
		[HL].bit,$addr16	3	10+3n	11+4n	PC ← PC+3+jdisp8 if (HL).bit=1	
	BF	saddr.bit,$addr16	4	10+4n	11+4n	PC ← PC+4+jdisp8 if (saddr.bit)=0	
		sfr.bit,$addr16	4	–	11+4n	PC ← PC+4+jdisp8 if sfr.bit=0	
		A.bit,$addr16	3	8+3n	–	PC ← PC+3+jdisp8 if A.bit=0	
		PSW.bit,$addr16	4	–	11+4n	PC ← PC+4+jdisp8 if PSW.bit=0	
		[HL].bit,$addr16	3	10+3n	11+4n	PC ← PC+3+jdisp8 if (HL).bit=0	
	BTCLR	saddr.bit,$addr16	4	10+4n	12+4n	PC ← PC+4+jdisp8 if (saddr.bit)=1 then reset (saddr.bit)	
		sfr.bit,$addr16	4	–	12+4n	PC ← PC+4+jdisp8 if sfr.bit=1 then reset sfr.bit	
		A.bit,$addr16	3	8+3n	–	PC ← PC+3+jdisp8 if A.bit=1 then reset A.bit	
		PSW.bit,$addr16	4	–	12+4n	PC ← PC+4+jdisp8 if PSW.bit=1 then reset PSW.bit	× × ×
		[HL].bit,$addr16	3	10+3n	12+4n+m	PC ← PC+3+jdisp8 if (HL).bit=1 then reset (HL).bit	
	DBNZ	B,$addr16	2	6+2n	–	B ← B–1, then PC ← PC+2+jdisp8 if B≠0	
		C,$addr16	2	6+2n	–	C ← C–1, then PC ← PC+2+jdisp8 if C≠0	
		saddr,$addr16	3	8+3n	10+3n	(saddr) ← (saddr)–1, then PC ← PC+3+jdisp8 if (saddr)≠0	
CPU Control	**SEL**	RBn	2	4+2n	–	RBS1,0 ← n	
	NOP		1	2+n	–	No Operation	
	EI		2	–	6+2n	IE ← 1 (Enable Interrupt)	
	DI		2	–	6+2n	IE ← 0 (Disable Interrupt)	
	HALT		2	6+2n	–	Set HALT Mode	
	STOP		2	6+2n	–	Set STOP Mode	

Notes 1. When the internal high-speed RAM area is accessed or in the instruction with no data access.
2. When an area except the internal high-speed RAM area is accessed.

Remarks 1. 1 instruction clock cycle is 1 CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).
2. n indicates the number of waits per byte when the external memory expansion area is read or fetched.
3. m indicates the number of waits when the external memory expansion area is written to.

4.1.6 Instructions listed by addressing type

(1) 8-bit instructions

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, MULU[Note], DIVUW[Note], INC, DEC, ROR, ROL, RORC, ROLC, ROR4, ROL4, PUSH, POP, DBNZ

Note The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

1st Operand \ 2nd Operand	#byte	A	r[Note 1]	sfr	saddr	!addr16	PSW	[DE]	[HL]	[HL+byte] [HL+B] [HL+C]	$addr16	1	None
A	ADD ADDC SUB SUBC AND OR XOR CMP		MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP		ROR ROL RORC ROLC	
r	MOV	MOV ADD ADDC SUB SUBC AND OR XOR CMP											INC DEC
B, C											DBNZ		
sfr	MOV	MOV											
saddr	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV									DBNZ		INC DEC
!addr16		MOV											
PSW	MOV	MOV											PUSH POP
[DE]		MOV											
[HL]		MOV											ROR4 ROL4
[HL+byte] [HL+B] [HL+C]		MOV											
X													MULU[Note2]
C													DIVUW[Note2]

Notes 1. Except r = A.

2. The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

(2) 16-bit instructions

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

1st Operand \ 2nd Operand	#word	AX	rp[Note]	sfrp	saddrp	!addr16	SP	None
AX	ADDW SUBW CMPW		MOVW XCHW	MOVW	MOVW	MOVW	MOVW	
rp	MOVW	MOVW[Note]						INCW DECW PUSH POP
sfrp	MOVW	MOVW						
saddrp	MOVW	MOVW						
!addr16		MOVW						
SP	MOVW	MOVW						

Note Only when rp = BC, DE or HL.

(3) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR

1st Operand \ 2nd Operand	A.bit	sfr.bit	saddr.bit	PSW.bit	[HL].bit	CY	$addr16	None
A.bit						MOV1	BT BF BTCLR	SET1 CLR1
sfr.bit						MOV1	BT BF BTCLR	SET1 CLR1
saddr.bit						MOV1	BT BF BTCLR	SET1 CLR1
PSW.bit						MOV1	BT BF BTCLR	SET1 CLR1`
[HL].bit						MOV1	BT BF BTCLR	SET1 CLR1
CY	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1			SET1 CLR1 NOT1

(4) Call instructions/branch instructions

CALL, CALLF, CALLT, BR, BC, BNC, BZ, BNZ, BT, BF, BTCLR, DBNZ

2nd Operand / 1st Operand	AX	!addr16	!addr11	[addr5]	$addr16
Basic Instructions	BR	CALL BR	CALLF	CALLT	BR BC BNC BZ BNZ
Compound Instructions					BT BF BTCLR DBNZ

(5) Other instructions

ADJBA, ADJBS, BRK, RET, RETI, RETB, SEL, NOP, EI, DI, HALT, STOP

4.2 Instruction Codes

4.2.1 Description of instruction code table

r

R_2	R_1	R_0	reg	
0	0	0	R0	X
0	0	1	R1	A
0	1	0	R2	C
0	1	1	R3	B
1	0	0	R4	E
1	0	1	R5	D
1	1	0	R6	L
1	1	1	R7	H

rp

P_1	P_0	reg-pair	
0	0	RP0	AX
0	1	RP1	BC
1	0	RP2	DE
1	1	RP3	HL

RB

RB_1	RB_0	reg-bank
0	0	RB0
0	1	RB1
1	0	RB2
1	1	RB3

Bn : Immediate data corresponding to bit
Data : 8-bit immediate data corresponding to byte
Low/High byte : 16-bit immediate data corresponding to word
Saddr-offset : 16-bit address lower 8-bit offset data corresponding to saddr
Sfr-offset : sfr 16-bit address lower 8-bit offset data
Low/High addr : 16-bit immediate data corresponding to addr16
jdisp : Signed two's complement data (8 bits) of relative address distance between the start and branch addresses of the next instruction
fa_{10-0} : 11 bits of immediate data corresponding to addr11
ta_{4-0} : 5 bits of immediate data corresponding to addr5

4.2.2 Instruction code list

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
8-Bit Data Transfer	**MOV**	r,#byte	1 0 1 0 0 R_2 R_1 R_0	Data		
		saddr,#byte	0 0 0 1 0 0 0 1	Saddr-offset	Data	
		sfr,#byte	0 0 0 1 0 0 1 1	Sfr-offset	Data	
		A,r Note	0 1 1 0 0 R_2 R_1 R_0			
		r,A Note	0 1 1 1 0 R_2 R_1 R_0			
		A,saddr	1 1 1 1 0 0 0 0	Saddr-offset		
		saddr,A	1 1 1 1 0 0 1 0	Saddr-offset		
		A,sfr	1 1 1 1 0 1 0 0	Sfr-offset		
		sfr,A	1 1 1 1 0 1 1 0	Sfr-offset		
		A,!addr16	1 0 0 0 1 1 1 0	Low addr	High addr	
		!addr16,A	1 0 0 1 1 1 1 0	Low addr	High addr	
		PSW,#byte	0 0 0 1 0 0 0 1	0 0 0 1 1 1 1 0	Data	
		A,PSW	1 1 1 1 0 0 0 0	0 0 0 1 1 1 1 0		
		PSW,A	1 1 1 1 0 0 1 0	0 0 0 1 1 1 1 0		
		A,[DE]	1 0 0 0 0 1 0 1			
		[DE],A	1 0 0 1 0 1 0 1			
		A,[HL]	1 0 0 0 0 1 1 1			
		[HL],A	1 0 0 1 0 1 1 1			
		A,[HL+byte]	1 0 1 0 1 1 1 0	Data		
		[HL+byte],A	1 0 1 1 1 1 1 0	Data		
		A,[HL+B]	1 0 1 0 1 0 1 1			
		[HL+B],A	1 0 1 1 1 0 1 1			
		A,[HL+C]	1 0 1 0 1 0 1 0			
		[HL+C],A	1 0 1 1 1 0 1 0			
	XCH	A,r Note	0 0 1 1 0 R_2 R_1 R_0			
		A,saddr	1 0 0 0 0 0 1 1	Saddr-offset		
		A,sfr	1 0 0 1 0 0 1 1	Sfr-offset		
		A,!addr16	1 1 0 0 1 1 1 0	Low addr	High addr	
		A,[DE]	0 0 0 0 0 1 0 1			
		A,[HL]	0 0 0 0 0 1 1 1			
		A,[HL+byte]	1 1 0 1 1 1 1 0	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	1 0 0 0 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	1 0 0 0 1 0 1 0		

Note Except r = A.

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
16-Bit Data Transfer	**MOVW**	rp,#word	0 0 0 1 0 P_1 P_0 0	Low byte	High byte	
		saddrp,#word	1 1 1 0 1 1 1 0	Saddr-offset	Low byte	High byte
		sfrp,#word	1 1 1 1 1 1 1 0	Sfr-offset	Low byte	High byte
		AX,saddrp	1 0 0 0 1 0 0 1	Saddr-offset		
		saddrp,AX	1 0 0 1 1 0 0 1	Saddr-offset		
		AX,sfrp	1 0 1 0 1 0 0 1	Sfr-offset		
		sfrp,AX	1 0 1 1 1 0 0 1	Sfr-offset		
		AX,rp Note 1	1 1 0 0 0 P_1 P_0 0			
		rp,AX Note 1	1 1 0 1 0 P_1 P_0 0			
		AX,!addr16	0 0 0 0 0 0 1 0	Low addr	High addr	
		!addr16,AX	0 0 0 0 0 0 1 1	Low addr	High addr	
	XCHW	AX,rp Note 1	1 1 1 0 0 P_1 P_0 0			
8-Bit Operation	**ADD**	A,#byte	0 0 0 0 1 1 0 1	Data		
		saddr,#byte	1 0 0 0 1 0 0 0	Saddr-offset	Data	
		A,r Note 2	0 1 1 0 0 0 0 1	0 0 0 0 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 0 0 0 0 R_2 R_1 R_0		
		A,saddr	0 0 0 0 1 1 1 0	Saddr-offset		
		A,!addr16	0 0 0 0 1 0 0 0	Low addr	High addr	
		A,[HL]	0 0 0 0 1 1 1 1			
		A,[HL+byte]	0 0 0 0 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 0 0 0 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 0 0 0 1 0 1 0		
	ADDC	A,#byte	0 0 1 0 1 1 0 1	Data		
		saddr,#byte	1 0 1 0 1 0 0 0	Saddr-offset	Data	
		A,r Note 2	0 1 1 0 0 0 0 1	0 0 1 0 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 0 1 0 0 R_2 R_1 R_0		
		A,saddr	0 0 1 0 1 1 1 0	Saddr-offset		
		A,!addr16	0 0 1 0 1 0 0 0	Low addr	High addr	
		A,[HL]	0 0 1 0 1 1 1 1			
		A,[HL+byte]	0 0 1 0 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 0 1 0 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 0 1 0 1 0 1 0		

Notes 1. Only when rp = BC, DE or HL.

2. Except r = A.

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
8-Bit Operation	**SUB**	A,#byte	0 0 0 1 1 1 0 1	Data		
		saddr,#byte	1 0 0 1 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 0 0 1 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 0 0 1 0 R_2 R_1 R_0		
		A,saddr	0 0 0 1 1 1 1 0	Saddr-offset		
		A,!addr16	0 0 0 1 1 0 0 0	Low addr	High addr	
		A,[HL]	0 0 0 1 1 1 1 1			
		A,[HL+byte]	0 0 0 1 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 0 0 1 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 0 0 1 1 0 1 0		
	SUBC	A,#byte	0 0 1 1 1 1 0 1	Data		
		saddr,#byte	1 0 1 1 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 0 1 1 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 0 1 1 0 R_2 R_1 R_0		
		A,saddr	0 0 1 1 1 1 1 0	Saddr-offset		
		A,!addr16	0 0 1 1 1 0 0 0	Low addr	High addr	
		A,[HL]	0 0 1 1 1 1 1 1			
		A,[HL+byte]	0 0 1 1 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 0 1 1 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 0 1 1 1 0 1 0		
	AND	A,#byte	0 1 0 1 1 1 0 1	Data		
		saddr,#byte	1 1 0 1 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 1 0 1 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 1 0 1 0 R_2 R_1 R_0		
		A,saddr	0 1 0 1 1 1 1 0	Saddr-offset		
		A,!addr16	0 1 0 1 1 0 0 0	Low addr	High addr	
		A,[HL]	0 1 0 1 1 1 1 1			
		A,[HL+byte]	0 1 0 1 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 1 0 1 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 1 0 1 1 0 1 0		

Note Except r = A.

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
8-Bit Operation	**OR**	A,#byte	0 1 1 0 1 1 0 1	Data		
		saddr,#byte	1 1 1 0 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 1 1 0 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 1 1 0 0 R_2 R_1 R_0		
		A,saddr	0 1 1 0 1 1 1 0	Saddr-offset		
		A,!addr16	0 1 1 0 1 0 0 0	Low addr	High addr	
		A,[HL]	0 1 1 0 1 1 1 1			
		A,[HL+byte]	0 1 1 0 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 1 1 0 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 1 1 0 1 0 1 0		
	XOR	A,#byte	0 1 1 1 1 1 0 1	Data		
		saddr,#byte	1 1 1 1 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 1 1 1 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 1 1 1 0 R_2 R_1 R_0		
		A,saddr	0 1 1 1 1 1 1 0	Saddr-offset		
		A,!addr16	0 1 1 1 1 0 0 0	Low addr	High addr	
		A,[HL]	0 1 1 1 1 1 1 1			
		A,[HL+byte]	0 1 1 1 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 1 1 1 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 1 1 1 1 0 1 0		
	CMP	A,#byte	0 1 0 0 1 1 0 1	Data		
		saddr,#byte	1 1 0 0 1 0 0 0	Saddr-offset	Data	
		A,r Note	0 1 1 0 0 0 0 1	0 1 0 0 1 R_2 R_1 R_0		
		r,A	0 1 1 0 0 0 0 1	0 1 0 0 0 R_2 R_1 R_0		
		A,saddr	0 1 0 0 1 1 1 0	Saddr-offset		
		A,!addr16	0 1 0 0 1 0 0 0	Low addr	High addr	
		A,[HL]	0 1 0 0 1 1 1 1			
		A,[HL+byte]	0 1 0 0 1 0 0 1	Data		
		A,[HL+B]	0 0 1 1 0 0 0 1	0 1 0 0 1 0 1 1		
		A,[HL+C]	0 0 1 1 0 0 0 1	0 1 0 0 1 0 1 0		

Note Except r = A.

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
16-Bit Operation	**ADDW**	AX,#word	1 1 0 0 1 0 1 0	Low byte	High byte	
	SUBW	AX,#word	1 1 0 1 1 0 1 0	Low byte	High byte	
	CMPW	AX,#word	1 1 1 0 1 0 1 0	Low byte	High byte	
Multiply/ divide	**MULU**Note	X	0 0 1 1 0 0 0 1	1 0 0 0 1 0 0 0		
	DIVUWNote	C	0 0 1 1 0 0 0 1	1 0 0 0 0 0 1 0		
Increment/ decrement	**INC**	r	0 1 0 0 0 R_2 R_1 R_0			
		saddr	1 0 0 0 0 0 0 1	Saddr-offset		
	DEC	r	0 1 0 1 0 R_2 R_1 R_0			
		saddr	1 0 0 1 0 0 0 1	Saddr-offset		
	INCW	rp	1 0 0 0 0 P_1 P_0 0			
	DECW	rp	1 0 0 1 0 P_1 P_0 0			
Rotate	**ROR**	A,1	0 0 1 0 0 1 0 0			
	ROL	A,1	0 0 1 0 0 1 1 0			
	RORC	A,1	0 0 1 0 0 1 0 1			
	ROLC	A,1	0 0 1 0 0 1 1 1			
	ROR4	[HL]	0 0 1 1 0 0 0 1	1 0 0 1 0 0 0 0		
	ROL4	[HL]	0 0 1 1 0 0 0 1	1 0 0 0 0 0 0 0		
BCD Adjust	**ADJBA**		0 1 1 0 0 0 0 1	1 0 0 0 0 0 0 0		
	ADJBS		0 1 1 0 0 0 0 1	1 0 0 1 0 0 0 0		
Bit Manipulation	**MOV1**	CY,saddr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 0 0	Saddr-offset	
		CY,sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 0 0	Sfr-offset	
		CY,A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 1 0 0		
		CY,PSW.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 0 0	0 0 0 1 1 1 1 0	
		CY,[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 0 0		
		saddr.bit,CY	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 0 1	Saddr-offset	
		sfr.bit,CY	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 0 0 1	Sfr-offset	
		A.bit,CY	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 0 0 1		
		PSW.bit,CY	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 0 1	0 0 0 1 1 1 1 0	
		[HL].bit,CY	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 0 0 1		
	AND1	CY,saddr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 0 1	Saddr-offset	
		CY,sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 0 1	Sfr-offset	
		CY,A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 1 0 1		
		CY,PSW.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 0 1	0 0 0 1 1 1 1 0	
		CY,[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 0 1		

Note The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
Bit Manipulation	**OR1**	CY,saddr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 0	Saddr-offset	
		CY,sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 1 0	Sfr-offset	
		CY,A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 1 1 0		
		CY,PSW.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 0	0 0 0 1 1 1 1 0	
		CY,[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 1 0		
	XOR1	CY,saddr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 1	Saddr-offset	
		CY,sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 1 1	Sfr-offset	
		CY,A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 1 1 1		
		CY,PSW.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 1	0 0 0 1 1 1 1 0	
		CY,[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 1 1		
	SET1	saddr.bit	0 B_2 B_1 B_0 1 0 1 0	Saddr-offset		
		sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 0 1 0	Sfr-offset	
		A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 0 1 0		
		PSW.bit	0 B_2 B_1 B_0 1 0 1 0	0 0 0 1 1 1 1 0		
		[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 0 1 0		
	CLR1	saddr.bit	0 B_2 B_1 B_0 1 0 1 1	Saddr-offset		
		sfr.bit	0 1 1 1 0 0 0 1	0 B_2 B_1 B_0 1 0 1 1	Sfr-offset	
		A.bit	0 1 1 0 0 0 0 1	1 B_2 B_1 B_0 1 0 1 1		
		PSW.bit	0 B_2 B_1 B_0 1 0 1 1	0 0 0 1 1 1 1 0		
		[HL].bit	0 1 1 1 0 0 0 1	1 B_2 B_1 B_0 0 0 1 1		
	SET1	CY	0 0 1 0 0 0 0 0			
	CLR1	CY	0 0 1 0 0 0 0 1			
	NOT1	CY	0 0 0 0 0 0 0 1			
Call Return	**CALL**	!addr16	1 0 0 1 1 0 1 0	Low addr	High addr	
	CALLF	!addr11	0 fa_{10-8} 1 1 0 0	fa_{7-0}		
	CALLT	[addr5]	1 1 ta_{4-0} 1			
	BRK		1 0 1 1 1 1 1 1			
	RET		1 0 1 0 1 1 1 1			
	RETB		1 0 0 1 1 1 1 1			
	RETI		1 0 0 0 1 1 1 1			
Stack Manipulation	**PUSH**	PSW	0 0 1 0 0 0 1 0			
		rp	1 0 1 1 0 P_1 P_0 1			
	POP	PSW	0 0 1 0 0 0 1 1			
		rp	1 0 1 1 0 P_1 P_0 0			
	MOVW	SP,#word	1 1 1 0 1 1 1 0	0 0 0 1 1 1 0 0	Low byte	High byte
		SP,AX	1 0 0 1 1 0 0 1	0 0 0 1 1 1 0 0		
		AX,SP	1 0 0 0 1 0 0 1	0 0 0 1 1 1 0 0		

Instruction Group	Mnemonic	Operands	Operation Code			
			B1	B2	B3	B4
Unconditional Branch	**BR**	!addr16	1 0 0 1 1 0 1 1	Low addr	High addr	
		$addr16	1 1 1 1 1 0 1 0	jdisp		
		AX	0 0 1 1 0 0 0 1	1 0 0 1 1 0 0 0		
Conditional Branch	**BC**	$addr16	1 0 0 0 1 1 0 1	jdisp		
	BNC	$addr16	1 0 0 1 1 1 0 1	jdisp		
	BZ	$addr16	1 0 1 0 1 1 0 1	jdisp		
	BNZ	$addr16	1 0 1 1 1 1 0 1	jdisp		
	BT	saddr.bit,$addr16	1 B_2 B_1 B_0 1 1 0 0	Saddr-offset	jdisp	
		sfr.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 0	Sfr-offset	jdisp
		A.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 1 0	jdisp	
		PSW.bit,$addr16	1 B_2 B_1 B_0 1 1 0 0	0 0 0 1 1 1 1 0	jdisp	
		[HL].bit,$addr16	0 0 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 1 0	jdisp	
	BF	saddr.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 1 1	Saddr-offset	jdisp
		sfr.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 1 1	Sfr-offset	jdisp
		A.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 1 1	jdisp	
		PSW.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 1 1	0 0 0 1 1 1 1 0	jdisp
		[HL].bit,$addr16	0 0 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 1 1	jdisp	
	BTCLR	saddr.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 0 1	Saddr-offset	jdisp
		sfr.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 1 0 1	Sfr-offset	jdisp
		A.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 1 1 0 1	jdisp	
		PSW.bit,$addr16	0 0 1 1 0 0 0 1	0 B_2 B_1 B_0 0 0 0 1	0 0 0 1 1 1 1 0	jdisp
		[HL].bit,$addr16	0 0 1 1 0 0 0 1	1 B_2 B_1 B_0 0 1 0 1	jdisp	
	DBNZ	B,$addr16	1 0 0 0 1 0 1 1	jdisp		
		C,$addr16	1 0 0 0 1 0 1 0	jdisp		
		saddr,$addr16	0 0 0 0 0 1 0 0	Saddr-offset	jdisp	
CPU control	**SEL**	RBn	0 1 1 0 0 0 0 1	1 1 RB_1 1 RB_0 0 0 0		
	NOP		0 0 0 0 0 0 0 0			
	EI		0 1 1 1 1 0 1 0	0 0 0 1 1 1 1 0		
	DI		0 1 1 1 1 0 1 1	0 0 0 1 1 1 1 0		
	HALT		0 1 1 1 0 0 0 1	0 0 0 1 0 0 0 0		
	STOP		0 1 1 1 0 0 0 1	0 0 0 0 0 0 0 0		

[MEMO]

CHAPTER 5 EXPLANATION OF INSTRUCTIONS

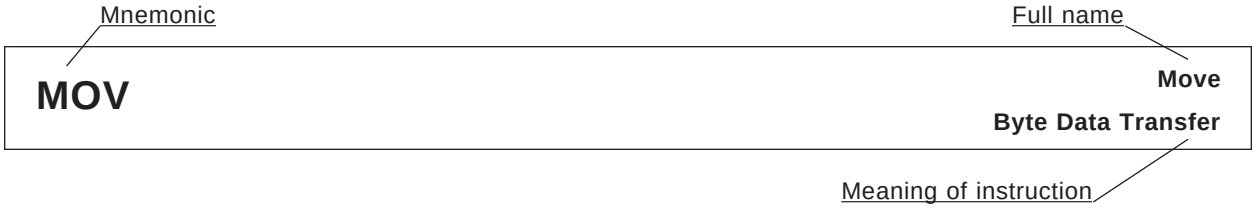
This chapter covers the explanation of the 78K/0 Series products' instructions. Each instruction is described with a mnemonic, including description of multiple operands.

The basic configuration of instruction description is shown on the next page.

For the number of instruction bytes and the operation code, refer to **CHAPTER 4 INSTRUCTION SET**.

All the instructions are common to the 78K/0 Series products.

DESCRIPTION EXAMPLE



[Instruction format] **MOV dst, src:** Indicates the basic description format of the instruction.

[Operation] **dst ← src:** Indicates instruction operation using symbols.

[Operand] Indicates operands which can be specified with this instruction. Refer to **4.1 Operation** for the description of each operand symbol.

Mnemonic	Operand(dst,src)
MOV	r, #byte
	A, saddr
	saddr, A
	PSW, #byte

Mnemonic	Operand(dst,src)
MOV	A, PSW
	[HL], A
	A, [HL+byte]
	[HL+C], A

[Flag] Indicates the flag operation which changes by instruction execution.
Each flag operation symbol is shown in the legend.

Z	AC	CY

Legend

Symbol	Description
Blank	Unchanged
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
R	Previously saved value is restored

[Description] : Describes the instruction operation in detail.

- The contents of the source operand (src) specified by the 2nd operand are transferred to the destination operand (dst) specified by the 1st operand.

[Description example]

MOV A, #4DH; 4DH is transferred to A register.

5.1 8-Bit Data Transfer Instructions

The following instructions are 8-bit data transfer instructions.

MOV

Move

Byte Data Transfer

[Instruction format] **MOV dst, src**

[Operation] **dst ← src**

[Operand]

Mnemonic	Operand(dst,src)
MOV	r, #byte
	saddr, #byte
	sfr, #byte
	A, r Note
	r, A Note
	A, saddr
	saddr, A
	A, sfr
	sfr, A
	A, !addr16
	!addr16, A
	PSW, #byte

Mnemonic	Operand(dst,src)
MOV	A, PSW
	PSW, A
	A, [DE]
	[DE], A
	A, [HL]
	[HL], A
	A, [HL+byte]
	[HL+byte], A
	A, [HL+B]
	[HL+B], A
	A, [HL+C]
	[HL+C], A

Note Except r = A

[Flag]

PSW, #byte and PSW, A operands

Z	AC	CY
×	×	×

All other operand combinations

Z	AC	CY

[Description]

- The contents of the source operand (src) specified by the 2nd operand are transferred to the destination operand (dst) specified by the 1st operand.
- None of the interrupts is acknowledged between the MOV PSW, #byte instruction/the MOV PSW, A instruction and the next one instruction.

[Description example]

MOV A, #4DH; 4DH is transferred to A register.

XCH

Exchange

Byte Data Exchange

[Instruction format] **XCH dst, src**

[Operation] **dst ↔ src**

[Operand]

Mnemonic	Operand(dst,src)
XCH	A, r Note
	A, saddr
	A, sfr
	A, !addr16
	A, [DE]

Mnemonic	Operand(dst,src)
XCH	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY

[Description]

- The 1st and 2nd operand contents are exchanged.

[Description example]

XCH A, FEBCH; The A register contents and address FEBCH contents are exchanged.

5.2 16-Bit Data Transfer Instructions

The following instructions are 16-bit data transfer instructions.

MOVW

Move Word

Word Data Transfer

[Instruction format] **MOVW dst, src**

[Operation] **dst ← src**

[Operand]

Mnemonic	Operand(dst,src)
MOVW	rp, #word
	saddrp, #word
	sfrp, #word
	AX, saddrp
	saddrp, AX
	AX, sfrp

Mnemonic	Operand(dst,src)
MOVW	sfrp, AX
	AX, rp **Note**
	rp, AX **Note**
	AX, !addr16
	!addr16, AX

Note Only when rp = BC, DE or HL

[Flag]

Z	AC	CY

[Description]

- The contents of the source operand (src) specified by the 2nd operand are transferred to the destination operand (dst) specified by the 1st operand.

[Description example]

MOVW AX, HL; The HL register contents are transferred to the AX register.

[Caution]

Only an even address can be specified. An odd address cannot be specified.

XCHW

Exchange Word
Word Data Exchange

[Instruction format] **XCHW dst, src**

[Operation] **dst ↔ src**

[Operand]

Mnemonic	Operand(dst,src)
XCHW	AX, rp Note

Note Only when rp = BC, DE or HL

[Flag]

Z	AC	CY

[Description]

- The 1st and 2nd operand contents are exchanged.

[Description example]

XCHW AX, BC; The memory contents of AX register are exchanged with those of the BC register.

5.3 8-Bit Operation Instructions

The following are 8-bit operation instructions.

ADD

Add

Byte Data Addition

[Instruction format] **ADD dst, src**

[Operation] **dst, CY ← dst + src**

[Operand]

Mnemonic	Operand(dst,src)
ADD	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
ADD	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×	×	×

[Description]

- The destination operand (dst) specified with the 1st operand is added to the source operand (src) specified with the 2nd operand and the result is stored in the CY flag and the destination operand (dst).
- If the addition result shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the addition generates a carry out of bit 7, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- If the addition generates a carry for bit 4 out of bit 3, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).

[Description example]

ADD CR10, #56H; 56H is added to the CR10 register and the result is stored in the CR10 register.

ADDC

Add with Carry

Addition of Byte Data with Carry

[Instruction format] **ADDC dst, src**

[Operation] **dst, CY ← dst + src + CY**

[Operand]

Mnemonic	Operand(dst,src)
ADDC	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
ADDC	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×	×	×

[Description]

- The destination operand (dst) specified with the 1st operand, the source operand (src) specified with the 2nd operand and the CY flag are added and the result is stored in the destination operand (dst) and the CY flag.
  The CY flag is added to the least significant bit. This instruction is mainly used to add two or more bytes.
- If the addition result shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the addition generates a carry out of bit 7, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- If the addition generates a carry for bit 4 out of bit 3, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).

[Description example]

ADDC A, [HL+B]; The A register contents and the contents at address (HL register + (B register)) and the CY flag are added and the result is stored in the A register.

SUB

Subtract

Byte Data Subtraction

[Instruction format] **SUB dst, src**

[Operation] **dst, CY ← dst – src**

[Operand]

Mnemonic	Operand(dst,src)
SUB	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
SUB	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×	×	×

[Description]

- The source operand (src) specified with the 2nd operand is subtracted from the destination operand (dst) specified with the 1st operand and the result is stored in the destination operand (dst) and the CY flag. The destination operand can be cleared to 0 by equalizing the source operand (src) and the destination operand (dst).
- If the subtraction shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the subtraction generates a borrow out of bit 7, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- If the subtraction generates a borrow for bit 3 out of bit 4, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).

[Description example]

SUB D, A; The A register is subtracted from the D register and the result is stored in the D register.

SUBC

Subtract with Carry

Subtraction of Byte Data with Carry

[Instruction format] **SUBC dst, src**

[Operation] **dst, CY ← dst – src – CY**

[Operand]

Mnemonic	Operand(dst,src)
SUBC	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
SUBC	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×	×	×

[Description]

- The source operand (src) specified with the 2nd operand and the CY flag are subtracted from the destination operand (dst) specified with the 1st operand and the result is stored in the destination operand (dst). The CY flag is subtracted from the least significant bit. This instruction is mainly used for subtraction of two or more bytes.
- If the subtraction shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the subtraction generates a borrow out of bit 7, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- If the subtraction generates a borrow for bit 3 out of bit 4, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).

[Description example]

SUBC A, [HL]; The (HL register) address contents and the CY flag are subtracted from the A register and the result is stored in the A register.

AND

And

Logical Product of Byte Data

[Instruction format] **AND dst, src**

[Operation] **dst ← dst ∧ src**

[Operand]

Mnemonic	Operand(dst,src)
AND	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
AND	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×		

[Description]

- Bit-wise logical product is obtained from the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand and the result is stored in the destination operand (dst).
- If the logical product shows that all bits are 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).

[Description example]

AND FEBAH, #11011100B; Bit-wise logical product of FEBAH contents and 11011100B is obtained and the result is stored at FEBAH.

OR

Or

Logical Sum of Byte Data

[Instruction format] **OR dst, src**

[Operation] **dst ← dst ∨ src**

[Operand]

Mnemonic	Operand(dst,src)
OR	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
OR	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×		

[Description]

- Bit-wise logical sum is obtained from the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand and the result is stored in the destination operand (dst).
- If the logical sum shows that all bits are 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).

[Description example]

OR A, FE98H; Bit-wise logical sum of the A register and FE98H is obtained and the result is stored in the A register.

XOR

Exclusive Or

Exclusive Logical Sum of Byte Data

[Instruction format] **XOR dst, src**

[Operation] **dst ← dst ⩛ src**

[Operand]

Mnemonic	Operand(dst,src)
XOR	A, #byte
	saddr, #byte
	A, r **Note**
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
XOR	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×		

[Description]

- Bit-wise exclusive logical sum is obtained from the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand and the result is stored in the destination operand (dst).
  Logical negation of all bits of the destination operand (dst) is possible by selecting #0FFH for the source operand (src) with this instruction.
- If the exclusive logical sum shows that all bits are 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).

[Description example]

XOR A, L; Bit-wise exclusive logical sum of the A and L registers is obtained and the result is stored in the A register.

CMP

Compare

Byte Data Comparison

[Instruction format] **CMP dst, src**

[Operation] **dst – src**

[Operand]

Mnemonic	Operand(dst,src)
CMP	A, #byte
	saddr, #byte
	A, r Note
	r, A
	A, saddr

Mnemonic	Operand(dst,src)
CMP	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]

Note Except r = A

[Flag]

Z	AC	CY
×	×	×

[Description]

- The source operand (src) specified with the 2nd operand is subtracted from the destination operand (dst) specified with the 1st operand.
  The subtraction result is not stored anywhere and only the Z, AC and CY flags are changed.
- If the subtraction result is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the subtraction generates a borrow out of bit 7, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- If the subtraction generates a borrow for bit 3 out of bit 4, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).

[Description example]

CMP FE38H, #38H; 38H is subtracted from the contents at address FE38H and only the flags are changed (comparison of contents at address FE38H and the immediate data).

5.4 16-Bit Operation Instructions

The following are 16-bit operation instructions.

ADDW

Add Word
Word Data Addition

[Instruction format] **ADDW dst, src**

[Operation] **dst, CY ← dst + src**

[Operand]

Mnemonic	Operand(dst,src)
ADDW	AX, #word

[Flag]

Z	AC	CY
×	×	×

[Description]

- The destination operand (dst) specified with the 1st operand is added to the source operand (src) specified with the 2nd operand and the result is stored in the destination operand (dst).
- If the addition result shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the addition generates a carry out of bit 15, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- As a result of addition, the AC flag becomes undefined.

[Description example]

ADDW AX, #ABCDH; ABCDH is added to the AX register and the result is stored in the AX register.

SUBW

Subtract Word
Word Data Subtraction

[Instruction format] **SUBW dst, src**

[Operation] **dst, CY ← dst – src**

[Operand]

Mnemonic	Operand(dst,src)
SUBW	AX, #word

[Flag]

Z	AC	CY
×	×	×

[Description]

- The source operand (src) specified with the 2nd operand is subtracted from the destination operand (dst) specified with the 1st operand and the result is stored in the destination operand (dst) and the CY flag. The destination operand can be cleared to 0 by equalizing the source operand (src) and the destination operand (dst).
- If the subtraction shows that dst is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the subtraction generates a borrow out of bit 15, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- As a result of subtraction, the AC flag becomes undefined.

[Description example]

SUBW AX, #ABCDH; ABCDH is subtracted from the AX register contents and the result is stored in the AX register.

CMPW

Compare Word
Word Data Comparison

[Instruction format] **CMPW dst, src**

[Operation] **dst – src**

[Operand]

Mnemonic	Operand(dst,src)
CMPW	AX, #word

[Flag]

Z	AC	CY
×	×	×

[Description]

- The source operand (src) specified with the 2nd operand is subtracted from the destination operand (dst) specified with the 1st operand.
  The subtraction result is not stored anywhere and only the Z, AC and CY flags are changed.
- If the subtraction result is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the subtraction generates a borrow out of bit 15, the CY flag is set to (1). In all other cases, the CY flag is cleared to (0).
- As a result of subtraction, the AC flag becomes undefined.

[Description example]

CMPW AX, #ABCDH; ABCDH is subtracted from the AX register and only the flags are changed (comparison of the AX register and the immediate data).

5.5 Multiply/Divide Instructions

The following are multiply/divide instructions.

Caution The μPD78002/78002Y Subseries have no MULU/DIVUW instructions.

MULU

Multiply Unsigned
Unsigned Multiplication of Data

[Instruction format] **MULU src**

[Operation] **AX ← A × src**

[Operand]

Mnemonic	Operand(src)
MULU	X

[Flag]

Z	AC	CY

[Description]

- The A register contents and the source operand (src) data are multiplied as unsigned data and the result is stored in the AX register.

[Description example]

MULU X; The A register contents and the X register contents are multiplied and the result is stored in the AX register.

DIVUW

Divide Unsigned Word
Unsigned Division of Word Data

[Instruction format] **DIVUW dst**

[Operation] **AX (quotient), dst (remainder) ← AX ÷ dst**

[Operand]

Mnemonic	Operand(dst)
DIVUW	C

[Flag]

Z	AC	CY

[Description]

- The AX register contents are divided with the destination operand (dst) contents and the quotient and the remainder are stored in the AX register and the destination operand (dst), respectively.
  Division is executed using the AX register and destination operand (dst) contents as unsigned data.
  However, when the destination operand (dst) is 0, the X register contents are stored in the C register and AX becomes 0FFFFH.

[Description example]

DIVUW C; The AX register contents are divided by the C register contents and the quotient and the remainder are stored in the AX register and the C register, respectively.

5.6 Increment/Decrement Instructions

The following are increment/decrement instructions.

INC

Increment

Byte Data Increment

[Instruction format] **INC dst**

[Operation] **dst ← dst + 1**

[Operand]

Mnemonic	Operand(dst)
INC	r
	saddr

[Flag]

Z	AC	CY
×	×	

[Description]

- The destination operand (dst) contents are incremented by only one.
- If the increment result is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the increment generates a carry for bit 4 out of bit 3, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).
- Because this instruction is frequently used for increment of a counter for repeated operations and an indexed addressing offset register, the CY flag contents are not changed (to hold the CY flag contents in multiple-byte operation).

[Description example]

INC B; The B register is incremented.

DEC

Decrement

Byte Data Decrement

[Instruction format] **DEC dst**

[Operation] **dst ← dst – 1**

[Operand]

Mnemonic	Operand(dst)
DEC	r
	saddr

[Flag]

Z	AC	CY
×	×	

[Description]

- The destination operand (dst) contents are decremented by only one.
- If the decrement result is 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).
- If the decrement generates a carry for bit 3 out of bit 4, the AC flag is set to (1). In all other cases, the AC flag is cleared to (0).
- Because this instruction is frequently used for decrement of a counter for repeated operations and an indexed addressing offset register, the CY flag contents are not changed (to hold the CY flag contents in multiple-byte operation).
- If dst is the B or C register or saddr, and it is not desired to change the AC and CY flag contents, the DBNZ instruction can be used.

[Description example]

DEC FE92H; The contents at address FE92H are decremented.

INCW

Increment Word

Word Data Increment

[Instruction format] **INCW dst**

[Operation] **dst ← dst + 1**

[Operand]

Mnemonic	Operand(dst)
INCW	rp

[Flag]

Z	AC	CY

[Description]

- The destination operand (dst) contents are incremented by only one.
- Because this instruction is frequently used for increment of a register (pointer) used for addressing, the Z, AC and CY flag contents are not changed.

[Description example]

INCW HL; The HL register is incremented.

DECW

Decrement Word
Word Data Decrement

[Instruction format] **DECW dst**

[Operation] **dst ← dst – 1**

[Operand]

Mnemonic	Operand (dst)
DECW	rp

[Flag]

Z	AC	CY

[Description]

- The destination operand (dst) contents are decremented by only one.
- Because this instruction is frequently used for decrement of a register (pointer) used for addressing, the Z, AC and CY flag contents are not changed.

[Description example]

DECW DE; The DE register is decremented.

5.7 Rotate Instructions

The following are rotate instructions.

ROR

Rotate Right

Byte Data Rotation to the Right

[Instruction format] **ROR dst, cnt**

[Operation] **(CY, $dst_7 \leftarrow dst_0$, $dst_{m-1} \leftarrow dst_m$) × one time**

[Operand]

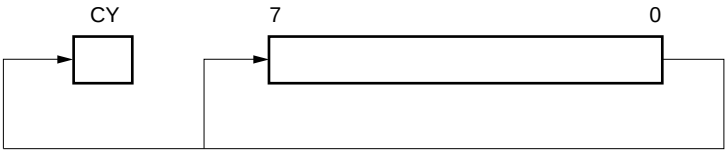
Mnemonic	Operand(dst,cnt)
ROR	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) contents specified with the 1st operand are rotated to the right just once.
- The LSB (bit 0) contents are simultaneously rotated to MSB (bit 7) and transferred to the CY flag.



[Description example]

ROR A, 1; The A register contents are rotated one bit to the right.

ROL

Rotate Left

Byte Data Rotation to the Left

[Instruction format] **ROL dst, cnt**

[Operation] **(CY, $dst_0 \leftarrow dst_7$, $dst_{m+1} \leftarrow dst_m$) × one time**

[Operand]

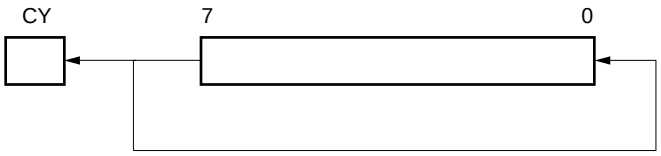
Mnemonic	Operand(dst,cnt)
ROL	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) contents specified with the 1st operand are rotated to the left just once.
- The MSB (bit 7) contents are simultaneously rotated to LSB (bit 0) and transferred to the CY flag.



[Description example]

ROL A, 1; The A register contents are rotated to the left by one bit.

RORC

Rotate Right with Carry

Byte Data Rotation to the Right with Carry

[Instruction format] **RORC dst, cnt**

[Operation] **(CY ← dst_0, dst_7 ← CY, dst_{m-1} ← dst_m) × one time**

[Operand]

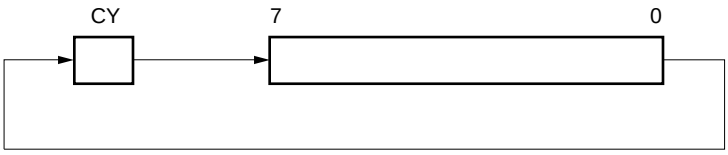
Mnemonic	Operand(dst,cnt)
RORC	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) contents specified with the 1st operand are rotated just once to the right with carry.



[Description example]

RORC A, 1; The A register contents are rotated to the right by one bit including the CY flag.

ROLC

Rotate Left with Carry

Byte Data Rotation to the Left with Carry

[Instruction format] **ROLC dst, cnt**

[Operation] **(CY ← dst_7, dst_0 ← CY, dst_{m+1} ← dst_m) × one time**

[Operand]

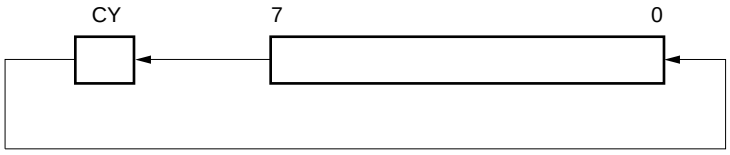
Mnemonic	Operand(dst,cnt)
ROLC	A, 1

[Flag]

Z	AC	CY
		×

[Description]

- The destination operand (dst) contents specified with the 1st operand are rotated just once to the left with carry.



[Description example]

ROLC A, 1; The A register contents are rotated to the left by one bit including the CY flag.

ROR4

Rotate Right Digit
Digit Rotation to the Right

[Instruction format] **ROR4 dst**

[Operation] $A_{3-0} \leftarrow (dst)_{3-0}$, $(dst)_{7-4} \leftarrow A_{3-0}$, $(dst)_{3-0} \leftarrow (dst)_{7-4}$

[Operand]

Mnemonic	Operand(dst)
ROR4	[HL] Note

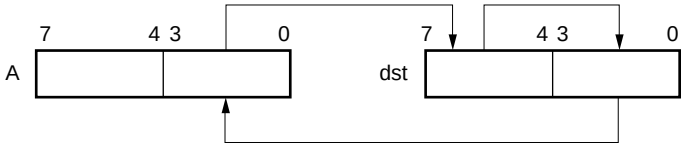
Note Specify an area other than the SFR area as operand [HL].

[Flag]

Z	AC	CY

[Description]

- The lower 4 bits of the A register and the 2-digit data (4-bit data) of the destination operand (dst) are rotated to the right.
  The higher 4 bits of the A register remain unchanged.



[Description example]

ROR4 [HL]; Rightward digit rotation is executed with the memory contents specified with the A and HL registers.

	A		(HL)	
	7 4 3 0		7 4 3 0	
Before Execution	1010	0011	1100	0101
After Execution	1010	0101	0011	1100

ROL4

Rotate Left Digit
Digit Rotation to the Left

[Instruction format] **ROL4 dst**

[Operation] $A_{3-0} \leftarrow (dst)_{7-4}$, $(dst)_{3-0} \leftarrow A_{3-0}$, $(dst)_{7-4} \leftarrow (dst)_{3-0}$

[Operand]

Mnemonic	Operand(dst)
ROL4	[HL] Note

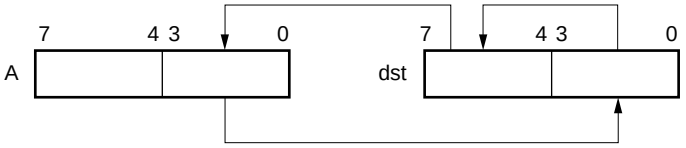
Note Specify an area other than the SFR area as operand [HL].

[Flag]

Z	AC	CY

[Description]

- The lower 4 bits of the A register and the 2-digit data (4-bit data) of the destination operand (dst) are rotated to the left.
  The higher 4 bits of the A register remain unchanged.



[Description example]

ROL4 [HL]; Leftward digit rotation is executed with the memory contents specified with the A and HL registers.

	A		(HL)	
	7 4	3 0	7 4	3 0
Before Execution	0001	0010	0100	1000
After Execution	0001	0100	1000	0010

5.8 BCD Adjust Instructions

The following are BCD adjust instructions.

ADJBA

Decimal Adjust Register for Addition
Decimal Adjustment of Addition Result

[Instruction format] **ADJBA**

[Operation] **Decimal Adjust Accumulator for Addition**

[Operand]

None

[Flag]

Z	AC	CY
×	×	×

[Description]

- The A register, CY flag and AC flag are decimally adjusted from their contents. This instruction carries out an operation having meaning only when the BCD (binary coded decimal) data is added and the addition result is stored in the A register (in all other cases, the instruction carries out an operation having no meaning). See the table below for the adjustment method.
- If the adjustment result shows that the A register contents are 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).

Condition		Operation
A_3 to $A_0 \leq 9$	A_7 to $A_4 \leq 9$ and CY = 0	A ← A, CY ← 0, AC ← 0
AC = 0	A_7 to $A_4 \geq 10$ or CY = 1	A ← A+01100000B, CY ← 1, AC ← 0
A_3 to $A_0 \geq 10$	A_7 to $A_4 < 9$ and CY = 0	A ← A+00000110B, CY ← 0, AC ← 1
AC = 0	A_7 to $A_4 \geq 9$ or CY = 1	A ← A+01100110B, CY ← 1, AC ← 1
AC = 1	A_7 to $A_4 \leq 9$ and CY = 0	A ← A+00000110B, CY ← 0, AC ← 0
	A_7 to $A_4 \geq 10$ or CY = 1	A ← A+01100110B, CY ← 1, AC ← 0

ADJBS

Decimal Adjust Register for Subtraction
Decimal Adjustment of Subtraction Result

[Instruction format] **ADJBS**

[Operation] **Decimal Adjust Accumulator for Subtraction**

[Operand]

None

[Flag]

Z	AC	CY
×	×	×

[Description]

- The A register, CY flag and AC flag are decimally adjusted from their contents. This instruction carries out an operation having meaning only when the BCD (binary coded decimal) data is subtracted and the subtraction result is stored in the A register (in all other cases, the instruction carries out an operation having no meaning).
  See the table below for the adjustment method.
- If the adjustment result shows that the A register contents are 0, the Z flag is set to (1). In all other cases, the Z flag is cleared to (0).

Condition		Operation
AC = 0	CY = 0	A ← A, CY ← 0, AC ← 0
	CY = 1	A ← A–01100000B, CY ← 1, AC ← 0
AC = 1	CY = 0	A ← A–00000110B, CY ← 0, AC ← 0
	CY = 1	A ← A–01100110B, CY ← 1, AC ← 0

5.9 Bit Manipulation Instructions

The following are bit manipulation instructions.

MOV1

Move Single Bit

1 Bit Data Transfer

[Instruction format] **MOV1 dst, src**

[Operation] **dst ← src**

[Operand]

Mnemonic	Operand(dst,src)
MOV1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit

Mnemonic	Operand(dst,src)
MOV1	saddr.bit, CY
	sfr.bit, CY
	A.bit, CY
	PSW.bit, CY
	[HL].bit, CY

[Flag]

dst = CY

Z	AC	CY
		×

PSW.bit

Z	AC	CY
×	×	

In all other cases

Z	AC	CY

[Description]

- Bit data of the source operand (src) specified with the 2nd operand is transferred to the destination operand (dst) specified with the 1st operand.
- When the destination operand (dst) is CY or PSW.bit, only the corresponding flag is changed.

[Description example]

MOV1 P3.4, CY; The CY flag contents are transferred to bit 4 of port 3.

AND1

And Single Bit
1 Bit Data Logical Product

[Instruction format] **AND1 dst, src**

[Operation] **dst ← dst ∧ src**

[Operand]

Mnemonic	Operand(dst,src)
AND1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit

[Flag]

Z	AC	CY
		×

[Description]

- Logical product of bit data of the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand is obtained and the result is stored in the destination operand (dst).
- The operation result is stored in the CY flag (because of the destination operand (dst)).

[Description example]

AND1 CY, FE7FH.3; Logical product of FE7FH bit 3 and the CY flag is obtained and the result is stored in the CY flag.

OR1

Or Single Bit
1 Bit Data Logical Sum

[Instruction format] **OR1 dst, src**

[Operation] **dst ← dst ∨ src**

[Operand]

Mnemonic	Operand(dst,src)
OR1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit

[Flag]

Z	AC	CY
		×

[Description]

- Logical sum of bit data of the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand is obtained and the result is stored in the destination operand (dst).
- The operation result is stored in the CY flag (because of the destination operand (dst)).

[Description example]

OR1 CY, P2.5; Logical sum or port 2 bit 5 and the CY flag is obtained and the result is stored in the CY flag.

XOR1

Exclusive Or Single Bit
1 Bit Data Exclusive Logical Sum

[Instruction format] **XOR1 dst, src**

[Operation] **dst ← dst ⩛ src**

[Operand]

Mnemonic	Operand(dst,src)
XOR1	CY, saddr.bit
	CY, sfr.bit
	CY, A.bit
	CY, PSW.bit
	CY, [HL].bit

[Flag]

Z	AC	CY
		×

[Description]

- Exclusive logical sum of bit data of the destination operand (dst) specified with the 1st operand and the source operand (src) specified with the 2nd operand is obtained and the result is stored in the destination operand (dst).
- The operation result is stored in the CY flag (because of the destination operand (dst)).

[Description example]

XOR1 CY, A.7; Exclusive logical sum of A register bit 7 and the CY flag is obtained and the result is stored in the CY flag.

SET1

Set Single Bit (Carry Flag)
1 Bit Data Set

[Instruction format] **SET1 dst**

[Operation] **dst ←1**

[Operand]

Mnemonic	Operand(dst)
SET1	saddr.bit
	sfr.bit
	A.bit
	PSW.bit
	[HL].bit
	CY

[Flag]

dst = PSW.bit

Z	AC	CY
×	×	×

dst = CY

Z	AC	CY
		1

In all other cases

Z	AC	CY

[Description]

- The destination operand (dst) is set to (1).
- When the destination operand (dst) is CY or PSW.bit, only the corresponding flag is set to (1).

[Description example]

SET1 FE55H.1; Bit 1 of FE55H is set to (1).

CLR1

Clear Single Bit (Carry Flag)
1 Bit Data Clear

[Instruction format] **CLR1 dst**

[Operation] **dst ← 0**

[Operand]

Mnemonic	Operand(dst)
CLR1	saddr.bit
	sfr.bit
	A.bit
	PSW.bit
	[HL].bit
	CY

[Flag]

dst = PSW.bit

Z	AC	CY
×	×	×

dst = CY

Z	AC	CY
		0

In all other cases

Z	AC	CY

[Description]

- The destination operand (dst) is cleared to (0).
- When the destination operand (dst) is CY or PSW.bit, only the corresponding flag is cleared to (0).

[Description example]

CLR1 P3.7; Bit 7 of port 3 is cleared to (0).

NOT1

Not Single Bit (Carry Flag)
1 Bit Data Logical Negation

[Instruction format] **NOT1 dst**

[Operation] **dst ← $\overline{\text{dst}}$**

[Operand]

Mnemonic	Operand(dst)
NOT1	CY

[Flag]

Z	AC	CY
		×

[Description]

- The CY flag is inverted.

[Description example]

NOT1 CY; The CY flag is inverted.

5.10 Call Return Instructions

The following are call return instructions.

CALL

Call

Subroutine Call (16 Bit Direct)

[Instruction format] **CALL target**

[Operation]

(SP–1) ← (PC+3)$_H$,
(SP–2) ← (PC+3)$_L$,
SP ← SP–2,
PC ← target

[Operand]

Mnemonic	Operand(target)
CALL	!addr16

[Flag]

Z	AC	CY

[Description]

- This is a subroutine call with a 16-bit absolute address or a register indirect address.
- The start address (PC+3) of the next instruction is saved in the stack and is branched to the address specified with the target operand (target).

[Description example]

CALL !3059H; Subroutine call to 3059H

CALLF

Call Flag

Subroutine Call (11 Bit Direct Specification)

[Instruction format] **CALLF Target**

[Operation]

(SP–1) ← (PC+2)H,
(SP–2) ← (PC+2)L,
SP ← SP–2,
PC ← target

[Operand]

Mnemonic	Operand(target)
CALLF	!addr11

[Flag]

Z	AC	CY

[Description]

- This is a subroutine call which can only be branched to addresses 0800H to 0FFFH.
- The start address (PC+2) of the next instruction is saved in the stack and is branched in the range of addresses 0800H to 0FFFH.
- Only the lower 11 bits of an address are specified (with the higher 5 bits fixed to 00001B).
- The program size can be compressed by locating the subroutine at 0800H to 0FFFH and using this instruction. If the program is in the external memory, the execution time can be decreased.

[Description example]

CALLF !0C2AH; Subroutine call to 0C2AH

CALLT

Call Table

Subroutine Call (Refer to the Call Table)

[Instruction format] **CALLT [addr5]**

[Operation]

(SP–1) ← **(PC+1)$_H$,**
(SP–2) ← **(PC+1)$_L$,**
SP ← **SP–2,**
PC$_H$ ← **(00000000, addr5+1)**
PC$_L$ ← **(00000000, addr5)**

[Operand]

Mnemonic	Operand([addr5])
CALLT	[addr5]

[Flag]

Z	AC	CY

[Description]

- This is a subroutine call for call table reference.
- The start address (PC+1) of the next instruction is saved in the stack and is branched to the address indicated with the word data of a call table (the higher 8 bits of address are fixed to 00000000B and the next 5 bits are specified with addr5).

[Description example]

CALLT [40H]; Subroutine call to the word data addresses 0040H and 0041H.

BRK

Break
Software Vectored Interrupt

[Instruction format] **BRK**

[Operation]

(SP–1) ← PSW,
(SP–2) ← (PC+1)$_H$,
(SP–3) ← (PC+1)$_L$,
IE ← 0,
SP ← SP–3,
PC$_H$ ← (3FH),
PC$_L$ ← (3EH)

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- This is a software interrupt instruction.
- PSW and the next instruction address (PC+1) are saved in the stack. After that, the IE flag is cleared to (0) and the saved data is branched to the address indicated with the word data at the vector address (003EH). Because the IE flag is cleared to (0), the subsequent maskable vectored interrupts are disabled.
- The RETB instruction is used to return from the software vectored interrupt generated with this instruction.

RET

Return

Return from Subroutine

[Instruction format] **RET**

[Operation] **$PC_L \leftarrow (SP)$,**
$PC_H \leftarrow (SP+1)$,
$SP \leftarrow SP+2$

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- This is a return instruction from the subroutine call made with the CALL, CALLF and CALLT instructions.
- The word data saved in the stack returns to the PC, and the program returns from the subroutine.

RETI

Return from Interrupt
Return from Hardware Vectored Interrupt

[Instruction format] **RETI**

[Operation]

$PC_L \leftarrow (SP)$,
$PC_H \leftarrow (SP+1)$,
PSW $\leftarrow$ (SP+2),
SP $\leftarrow$ SP+3,
NMIS $\leftarrow$ 0

[Operand]

None

[Flag]

Z	AC	CY
R	R	R

[Description]

- This is a return instruction from the vectored interrupt.
- The data saved in the stack returns to the PC and the PSW, and the program returns from the interrupt service routine.
- This instruction cannot be used for return from the software interrupt with the BRK instruction.
- None of interrupts are acknowledged between this instruction and the next instruction to be executed.
- The NMIS flag is set to 1 by acknowledgment of a non-maskable interrupt, and cleared to 0 by the RETI instruction.

[Caution]

When the return from non-maskable interrupt servicing is performed by an instruction other than the RETI instruction, the NMIS flag is not cleared to 0, and therefore no interrupts (including non-maskable interrupts) except software interrupts can be acknowledged.

RETB

Return from Break
Return from Software Vectored Interrupt

[Instruction format] **RETB**

[Operation]

$PC_L \leftarrow (SP)$,
$PC_H \leftarrow (SP+1)$,
$PSW \leftarrow (SP+2)$,
$SP \leftarrow SP+3$

[Operand]

None

[Flag]

Z	AC	CY
R	R	R

[Description]

- This is a return instruction from the software interrupt generated with the BRK instruction.
- The data saved in the stack returns to the PC and the PSW, and the program returns from the interrupt service routine.
- None of interrupts are acknowledged between this instruction and the next instruction to be executed.

5.11 Stack Manipulation Instructions

The following are stack manipulation instructions.

PUSH

Push
Push

[Instruction format] **PUSH src**

[Operation]

When src = rp	When src = PSW
(SP–1) ← src_H,	(SP–1) ← src
(SP–2) ← src_L,	SP ← SP–1
SP ← SP–2	

[Operand]

Mnemonic	Operand(src)
PUSH	PSW
	rp

[Flag]

Z	AC	CY

[Description]

- The data of the register specified with the source operand (src) is saved in the stack.

[Description example]

PUSH AX; AX register contents are saved in the stack.

POP

Pop
Pop

[Instruction format] **POP dst**

[Operation]

When dst = rp	When dst = PSW
$dst_L \leftarrow (SP)$,	**$dst \leftarrow (SP)$**
$dst_H \leftarrow (SP+1)$,	**$SP \leftarrow SP+1$**
$SP \leftarrow SP+2$	

[Operand]

Mnemonic	Operand(dst)
POP	PSW
	rp

[Flag]

dst =rp

Z	AC	CY

PSW

Z	AC	CY
R	R	R

[Description]

- Data is returned from the stack to the register specified with the destination operand (dst).
- When the operand is PSW, each flag is replaced with stack data.
- None of interrupts are acknowledged between the POP PSW instruction and the subsequent instruction.

[Description example]

POP AX; The stack data is returned to the AX register.

MOVW SP, src
MOVW AX, SP

Move Word

Word Data Transfer with Stack Pointer

[Instruction format] **MOVW dst, src**

[Operation] **dst ← src**

[Operand]

Mnemonic	Operand(dst,src)
MOVW	SP, #word
	SP, AX
	AX, SP

[Flag]

Z	AC	CY

[Description]

- This is an instruction to manipulate the stack pointer contents.
- The source operand (src) specified with the 2nd operand is stored in the destination operand (dst) specified with the 1st operand.

[Description example]

MOVW SP, #FE1FH; FE1FH is stored in the stack pointer.

5.12 Unconditional Branch Instruction

Unconditional branch instruction is shown below.

```
BR ... 151
```

BR

Branch
Unconditional Branch

[Instruction format] **BR target**

[Operation] **PC ← target**

[Operand]

Mnemonic	Operand(target)
BR	!addr16
	AX
	$addr16

[Flag]

Z	AC	CY

[Description]

- This is an instruction to branch unconditionally.
- The word data of the target address operand (target) is transferred to PC and branched.

[Description example]

BR AX; The AX register contents are branched as address.

5.13 Conditional Branch Instructions

Conditional branch instructions are shown below.

BC

Branch if Carry

Conditional Branch with Carry Flag (CY = 1)

[Instruction format] **BC $addr16**

[Operation] **PC ← PC+2+jdisp8 if CY = 1**

[Operand]

Mnemonic	Operand($addr16)
BC	$addr16

[Flag]

Z	AC	CY

[Description]

- When CY = 1, data is branched to the address specified with the operand.
  When CY = 0, no processing is carried out and the subsequent instruction is executed.

[Description example]

BC $300H; When CY = 1, data is branched to 0300H (with the start of this instruction set in the range of addresses 027FH to 037EH).

BNC

Branch if Not Carry

Conditional Branch with Carry Flag (CY = 0)

[Instruction format] **BNC $addr16**

[Operation] **PC ← PC+2+jdisp8 if CY = 0**

[Operand]

Mnemonic	Operand($addr16)
BNC	$addr16

[Flag]

Z	AC	CY

[Description]

- When CY = 0, data is branched to the address specified with the operand.
  When CY = 1, no processing is carried out and the subsequent instruction is executed.

[Description example]

BNC $300H; When CY = 0, data is branched to 0300H (with the start of this instruction set in the range of addresses 027FH to 037EH).

BZ

Branch if Zero

Conditional Branch with Zero Flag (Z = 1)

[Instruction format] **BZ $addr16**

[Operation] **PC ← PC+2+jdisp8 if Z = 1**

[Operand]

Mnemonic	Operand($addr16)
BZ	$addr16

[Flag]

Z	AC	CY

[Description]

- When Z = 1, data is branched to the address specified with the operand.
  When Z = 0, no processing is carried out and the subsequent instruction is executed.

[Description example]

DEC B

BZ $3C5H; When the B register is 0, data is branched to 03C5H (with the start of this instruction set in the range of addresses 0344H to 0443H).

BNZ

Branch if Not Zero
Conditional Branch with Zero Flag (Z = 0)

[Instruction format] **BNZ $addr16**

[Operation] **PC ← PC+2+jdisp8 if Z = 0**

[Operand]

Mnemonic	Operand($addr16)
BNZ	$addr16

[Flag]

Z	AC	CY

[Description]

- When Z = 0, data is branched to the address specified with the operand.
  When Z = 1, no processing is carried out and the subsequent instruction is executed.

[Description example]

```
CMP A, #55H
BNZ $0A39H;
```

If the A register is not 0055H, data is branched to 0A39H (with the start of this instruction set in the range of addresses 09B8H to 0AB7H).

BT

Branch if True

Conditional Branch by Bit Test (Byte Data Bit = 1)

[Instruction format] **BT bit, $addr16**

[Operation] **PC ← PC+b+jdisp8 if bit = 1**

[Operand]

Mnemonic	Operand(bit,$addr16)	b(Number of bytes)
BT	saddr.bit, $addr16	3
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	3
	[HL].bit, $addr16	3

[Flag]

Z	AC	CY

[Description]

- If the 1st operand (bit) contents have been set to (1), data is branched to the address specified with the 2nd operand ($addr16).
  If the 1st operand (bit) contents have not been set to (1), no processing is carried out and the subsequent instruction is executed.

[Description example]

BT FE47H.3, $55CH; When bit 3 at address FE47H is 1, data is branched to 055CH (with the start of this instruction set in the range of addresses 04DAH to 05D9H).

BF

Branch if False

Conditional Branch by Bit Test (Byte Data Bit = 0)

[Instruction format] **BF bit, $addr16**

[Operation] **PC ← PC+b+jdisp8 if bit = 0**

[Operand]

Mnemonic	Operand(bit,$addr16)	b(Number of bytes)
BF	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	4
	[HL].bit, $addr16	3

[Flag]

Z	AC	CY

[Description]

- If the 1st operand (bit) contents have been cleared to (0), data is branched to the address specified with the 2nd operand ($addr16).
  If the 1st operand (bit) contents have not been cleared to (0), no processing is carried out and the subsequent instruction is executed.

[Description example]

BF P2.2, $1549H; When bit 2 of port 2 is 0, data is branched to address 1549H (with the start of this instruction set in the range of addresses 14C6H to 15C5H).

BTCLR

Branch if True and Clear

Conditional Branch and Clear by Bit Test (Byte Data Bit = 1)

[Instruction format] **BTCLR bit, $addr16**

[Operation] **PC ← PC+b+jdisp8 if bit = 1, then bit ← 0**

[Operand]

Mnemonic	Operand(bit,$addr16)	b(Number of bytes)
BTCLR	saddr.bit, $addr16	4
	sfr.bit, $addr16	4
	A.bit, $addr16	3
	PSW.bit, $addr16	4
	[HL].bit, $addr16	3

[Flag]

bit =PSW.bit

Z	AC	CY
×	×	×

In all other cases

Z	AC	CY

[Description]

- If the 1st operand (bit) contents have been set to (1), they are cleared to (0) and branched to the address specified with the 2nd operand.
  If the 1st operand (bit) contents have not been set to (1), no processing is carried out and the subsequent instruction is executed.
- When the 1st operand (bit) is PSW.bit, the corresponding flag contents are cleared to (0).

[Description example]

BTCLR PSW.0, $356H; When bit 0 (CY flag) of PSW is 1, the CY flag is cleared to 0 and branched to address 0356H (with the start of this instruction set in the range of addresses 02D4H to 03D3H).

DBNZ

Decrement and Branch if Not Zero
Conditional Loop (R1 ≠ 0)

[Instruction format] **DBNZ dst, $addr16**

[Operation] **dst ← dst–1,**
then PC ← PC+b+jdisp16 if dst R1 ≠ 0

[Operand]

Mnemonic	Operand(dst,$addr16)	b(Number of bytes)
DBNZ	B, $addr16	2
	C, $addr16	2
	saddr, $addr16	3

[Flag]

Z	AC	CY

[Description]

- One is subtracted from the destination operand (dst) contents specified with the 1st operand and the subtraction result is stored in the destination operand (dst).
- If the subtraction result is not 0, data is branched to the address indicated with the 2nd operand ($addr16). When the subtraction result is 0, no processing is carried out and the subsequent instruction is executed.
- The flag remains unchanged.

[Description example]

DBNZ B, $1215H; The B register contents are decremented. If the result is not 0, data is branched to 1215H (with the start of this instruction set in the range of addresses 1194H to 1293H).

5.14 CPU Control Instructions

The following are CPU control instructions.

SEL RBn

Select Register Bank
Register Bank Selection

[Instruction format] **SEL RBn**

[Operation] **RBS0, RBS1 ← n; (n = 0-3)**

[Operand]

Mnemonic	Operand(RBn)
SEL	RBn

[Flag]

Z	AC	CY

[Description]

- The register bank specified with the operand (RBn) is made a register bank for use with the next instruction onward.
- RBn ranges from RB0 to RB3.

[Description example]

SEL RB2; Register bank 2 is selected as one for used with the next instruction onward.

NOP

No Operation

No Operation

[Instruction format] **NOP**

[Operation] **no operation**

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- Only the time is consumed without processing.

EI

Enable Interrupt
Interrupt Enabled

[Instruction format] **EI**

[Operation] **IE ← 1**

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- The maskable interrupt acknowledgeable status is set (by setting the interrupt enable flag (IE) to (1)).
- None of interrupts are acknowledged between this instruction and the next one instruction.
- If this instruction is executed, vectored interrupt acknowledgment with another source can be disabled. For details, refer to **"Interrupt Functions"** in **each product User's Manual**.

DI

Disable Interrupt

Interrupt Disabled

[Instruction format] **DI**

[Operation] **IE ← 0**

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- Maskable interrupt acknowledgment with vectored interrupt is disabled (with the interrupt enable flag (IE) cleared to (0)).
- None of interrupts are acknowledged between this instruction and the next one instruction.
- For details of interrupt service, refer to **"Interrupt Functions"** in **each product User's Manual**.

HALT

Halt
HALT Mode Set

[Instruction format] **HALT**

[Operation] **Set HALT Mode**

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- This instruction is used to set the HALT mode to stop the CPU operation clock. Total power consumption of the system can be decreased with intermittent operations through a combination with the normal operation mode.

STOP

Stop
Stop Mode Set

[Instruction format] **STOP**

[Operation] **Set STOP Mode**

[Operand]

None

[Flag]

Z	AC	CY

[Description]

- This instruction is used to set the STOP mode to stop the main system clock oscillator and to stop the whole system. Power dissipation can be minimized to an ultra-low level with only leakage current.

[MEMO]

APPENDIX A REVISION HISTORY

The following table shows the revision history of the previous editions. The Applied to: column describes the chapters of each edition.

Edition	Major Revision from the Previous Edition	Applied to:
2nd	Added the following versions: μPD78055 and 78P058, and μPD78018F, 78044A, 78054Y, 78078, 78083, 78098, and 780208 Subseries	Throughout
	Added the English documentation No. to the related documents	Introduction
	Added the IEBus register area (the μPD78098 Subseries only)	**CHAPTER 1 MEMORY SPACE**
	Added the description of the number of clocks when the external ROM contains the program to the clock column.	**CHAPTER 4 INSTRUCTION SET**
	Added Notes to the description of the ROR4 and ROL4 instructions in the rotate instruction.	**CHAPTER 5 EXPLANATION OF INSTRUCTIONS**
	Change the operation of the ADJBA and ADJBS instructions in the BCD Adjust instruction.	
3rd	Added the following versions: μPD78014H, 78018FY, 78044F, 78044H, 78058F, 78058FY, 78064Y, 78064B, 78075B, 78075BY, 78078Y, 78098B, 780018Y, 780024, 780024Y, 780034, 780034Y, 780058, 780058Y, 780228, 780308, 780308Y, 780924, and 780964 Subseries, and μPD78011F, 78012F, 78070A, 78070AY, 780001, 78P0914, 780206, and 780208	Throughout
	Deleted the following versions μPD78024, 78044, and 78044A Subseries	
	Added Table of all internal RAM spaces of each model	**CHAPTER 1 MEMORY SPACE**
	Change the format of external memory space table	

[MEMO]

APPENDIX B INSTRUCTION INDEX (MNEMONIC: BY FUNCTION)

[Unconditional branch instruction]

[Conditional branch instructions]

[CPU control instructions]

APPENDIX C INSTRUCTION INDEX (MNEMONIC: IN ALPHABETICAL ORDER)

[S]

[X]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑