十速科技股份有限公司
tenx technology inc.

Advance Information

TM89 Series MCU

Instructions Manual

Tenx reserves the right to change or discontinue this product without notice.

tenx technology inc.
TEL: 886-2-82571700 FAX: 886-2-82571701
886-3-5737682 886-3-5737632
URL: www.tenx.com.tw

CONTENTS

CHAPTER 1 INSTRUCTION SET QUICK REFERENCE TABLE 2

CHAPTER 2 INSTRUCTION SET EXPLANATION 9

2-1. INDEX REGISTER ACCESS INSTRUCTIONS 9
2-2. RAM DATA TRANSFER INSTRUCTIONS 22
2-3. LCD INSTRUCTIONS 31
2-4. REGISTER ACCESS INSTRUCTIONS 37
2-5. ARITHMETIC/LOGIC OPERATION INSTRUCTIONS 43
2-6. I/O PORT INSTRUCTIONS 68
2-7. TABLE ROM INSTRUCTIONS 76
2-8. RFC FUNCTION INSTRUCTIONS 79
2-9. JUMP/CALL INSTRUCTIONS 83
2-10. RAM PAGE/ROM BANK SETTING INSTRUCTIONS 92
2-11. TIMER INSTRUCTIONS 96
2-12. PERIPHERAL FUNCTION INSTRUCTIONS 107
2-13. SYSTEM FUNCTION INSTRUCTIONS 113

Chapter 1 Instruction Set Quick Reference Table

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
ADC	Arithmetic/ Logic Operation	1 word	4	4	44
ADCI	Arithmetic/ Logic Operation	1 word	4	4	57
ADCM	Arithmetic/ Logic Operation	1 word	4	4	45
ADD	Arithmetic/ Logic Operation	1 word	4	4	49
ADDI	Arithmetic/ Logic Operation	1 word	4	4	58
ADDM	Arithmetic/ Logic Operation	1 word	4	4	50
ADN	Arithmetic/ Logic Operation	1 word	4	4	54
ADNI	Arithmetic/ Logic Operation	1 word	4	4	59
ALM	Peripheral Function	1 word	4		111
AND	Arithmetic/ Logic Operation	1 word	4	4	54
ANDI	Arithmetic/ Logic Operation	1 word	4	4	62
CAC	Jump/ Call	multi-words	4 or 8		90
CALL	Jump/ Call	1 word	4		87
CLPG	RAM Page Setting	1 word	4		95
CPHL	Jump/ Call	1 word	4		88
CPHLH	Jump/ Call	2 words	8		89
CPZR	Jump/ Call	1 word	4		88
CPZRH	Jump/ Call	2 words	8		89
DAA	Arithmetic/ Logic Operation	1 word	4	4	60
DAS	Arithmetic/ Logic Operation	1 word	4	4	60
DEC*	Arithmetic/ Logic Operation	1 word	4	4	62
DISTM	Timer	1 word	4		106
ELC	Peripheral Function	1 word	4		110
ELZ	RAM Page	1 word	4		94

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
	Setting				
EOR	Arithmetic/ Logic Operation	1 word	4	4	55
EORI	Arithmetic/ Logic Operation	1 word	4	4	63
ERX	RAM Page Setting	1 word	4		93
ERY	RAM Page Setting	1 word	4		92
FAST	System Function	1 word	4		119
FRQ	System Function	1 word	4	4/8	114
FRQX					
HALT	System Function	1 word	4		113
IDC	Index Register Access	1 word	4		9
IDC8	Index Register Access	1 word	4		9
IDCH	Index Register Access	1 word	4		10
INC*	Arithmetic/ Logic Operation	1 word	4	4	61
IPA	I/O Port	1 word	4	4	68
IPB	I/O Port	1 word	4	4	70
IPC	I/O Port	1 word	4	4	72
IPD	I/O Port	1 word	4	4	73
IPE	I/O Port	1 word	4	4	75
JAC	Jump/ Call	multi-words	4 or 8		90
JB0	Jump/ Call	1 word	4		83
JB1	Jump/ Call	1 word	4		83
JB2	Jump/ Call	1 word	4		84
JB3	Jump/ Call	1 word	4		84
JC	Jump/ Call	1 word	4		86
JMP	Jump/ Call	1 word	4		87
JNC	Jump/ Call	1 word	4		86
JNZ	Jump/ Call	1 word	4		85
JZ	Jump/ Call	1 word	4		85
LCB	LCD	1 word	4	Read: 4 Write: 8	32
LCD	LCD	1 word	4	8	34
LCDH	Table ROM	1 word	4	16	77
LCE	LCD	1 word	4	8	35

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
LCP	LCD	1 word	4	Read: 4 Write: 8	33
LCT	LCD	1 word	4	Read: 4 Write: 8	31
LDA	RAM Data Transfer	1 word	4	4	30
LDH	Table ROM	1 word	4	4	76
LDL	Table ROM	1 word	4	4	76
LDS	RAM Data Transfer	1 word	4	4	26
LDS8	RAM Data Transfer	1 word	4	8	26
LDSH	RAM Data Transfer	2 words	8	16	27
LID	RAM Data Transfer	1 word	4	4	22
LID8	RAM Data Transfer	1 word	4	8	22
LIDH	RAM Data Transfer	1 word	4	16	24
LSP	Register Access	1 word	4	4	39
MAF	Register Access	1 word	4	4	39
MCX	Register Access	1 word	4	3	41
MDX	Register Access	1 word	4	4	42
MHL	Index Register Access	1 word	4	16	18
MKI	Register Access	1 word	4	4	42
MMH	Register Access	1 word	4	4	37
MMW	Register Access	1 word	4	4	38
MRA	Register Access	1 word	4	1	39
MRF1	RFC Function	1 word	4	4	81
MRF2	RFC Function	1 word	4	4	81
MRF3	RFC Function	1 word	4	4	82
MRF4	RFC Function	1 word	4	4	82
MRH	Index Register Access	1 word	4	4	14

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
MRL	Index Register Access	1 word	4	4	14
MRU	Index Register Access	1 word	4	4	15
MRV	Index Register Access	1 word	4	4	15
MRW	RAM Data Transfer	1 word	4	4	29
MSB	Register Access	1 word	4	4	40
MSC	Register Access	1 word	4	4	40
MSD	Register Access	1 word	4	4	41
MULD	Arithmetic/ Logic Operation	1 word	4	4	43
MULH	Arithmetic/ Logic Operation	1 word	4	4	43
MVL	Index Register Access	1 word	4	4	10
MVH	Index Register Access	1 word	4	4	10
MVU	Index Register Access	1 word	4	4	11
MVV	Index Register Access	1 word	4	4	11
MWM	Register Access	1 word	4	4	38
MWR	RAM Data Transfer	1 word	4	4	29
MZR	Index Register Access	1 word	4	13	20
NOP	System Function	1 word	4		113
OPA	I/O Port	1 word	4	4	69
OPAS	I/O Port	1 word	4	2	69
OPB	I/O Port	1 word	4	4	71
OPC	I/O Port	1 word	4	4	72
OPD	I/O Port	1 word	4	4	74
OPE	I/O Port	1 word	4	4	75
OR	Arithmetic/ Logic Operation	1 word	4	4	56
ORI	Arithmetic/ Logic Operation	1 word	4	4	64
PLC	System	1 word	4		123

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
	Function				
RF	System Function	1 word	4		121
RF2	System Function	1 word	4		122
RHL	Index Register Access	1 word	4	16	19
RLC	Arithmetic/ Logic Operation	1 word	4	4	66
RTS	Jump/ Call	1 word	4		91
RRC	Arithmetic/ Logic Operation	1 word	4	4	65
RRL	Index Register Access	1 word	4	4	15
RRH	Index Register Access	1 word	4	4	16
RRU	Index Register Access	1 word	4	4	16
RRV	Index Register Access	1 word	4	4	17
RTM2L RTM21 RTM1H RTM3L RTM31	Timer	1 word	4	4	104
RVL	Index Register Access	1 word	4	4	12
RVH	Index Register Access	1 word	4	4	12
RVU	Index Register Access	1 word	4	4	12
RVV	Index Register Access	1 word	4	4	13
RZR	Index Register Access	1 word	4	16	21
SBC	Arithmetic/ Logic Operation	1 word	4	4	46
SBCI	Arithmetic/ Logic Operation	1 word	4	4	58
SBCM	Arithmetic/ Logic Operation	1 word	4	4	48
SBZ	Peripheral Function	1 word	4		112
SCA	System Function	1 word	4		116

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
SCC	System Function	1 word	4		115
SCNT	RFC Function	1 word	4		80
SCX	System Function	1 word	4		117
SF	System Function	1 word	4		120
SF2	System Function	1 word	4		121
SHE	System Function	1 word	4		118
SHLX	Index Register Access	2 words	8		13
SIE*	System Function	1 word	4		117
SL0 SL1	Arithmetic/ Logic Operation	1 word	4	4	65
SLOW	System Function	1 word	4		120
SLZ	RAM Page Setting (Compiler only)	1 word	4		94
SMUI	Register Access	1 word	4	4	37
SPA	I/O Port	1 word	4		68
SPB	I/O Port	1 word	4		70
SPBK	ROM Bank Setting (Compiler only)	1 word	4		95
SPC	I/O Port	1 word	4		71
SPD	I/O Port	1 word	4		73
SPE	I/O Port	1 word	4		74
SPK SPKX	Peripheral Function	1 word	4	4/8	108
SPKTH	Peripheral Function	1 word	4	16	107
SPKXH		2 words	8		
SPKRH		2 words	8	16	
SR0, SR1	Arithmetic/ Logic Operation	1 word	4	4	64
SRE	System Function	1 word	4		119
SRF	RFC Function	1 word	4		79
SRX	RAM Page Setting (Compiler only)	1 word	4		93

OPCODE	Instruction Type	Instruction Length	Instruction Cycle (machine cycle)	Data width Transferred in memory (bits)	Page
SRY	RAM Page Setting (Compiler only)	1 word	4		92
STA	RAM Data Transfer	1 word	4	4	28
STM	Timer	1 word	4		105
STOP	System Function	1 word	4		114
SUB	Arithmetic/ Logic Operation	1 word	4	4	51
SUBI	Arithmetic/ Logic Operation	1 word	4	4	59
SUBM	Arithmetic/ Logic Operation	1 word	4	4	52
SZRX	Index Register Access	2 words	8		17
T1XH	Timer	2 words	8		97
T1RH		2 words	8	16	
T1TH		1 word	4	16	
T2M3X	Timer	2 words	8		104
T2XH	Timer	2 words	8		99
T2RH		2 words	8	16	
T2TH		1 word	4	16	
T3XH	Timer	2 words	8		102
T3RH		2 words	8	16	
T3TH		1 word	4	16	
TM2	Timer	1 word	4	4/8	98
TM2X					
TM3	Timer	1 word	4	4/8	101
TM3X					
TMS	Timer	1 word	4	4/8	96
TMSX					

Chapter 2 Instruction Set Explanation

2-1. Index Register Access Instructions

IDC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Increment the contents of HL or ZR by 1 at the same time or separately.

Note:
This instruction does not affect the content of AC during execution.

Instruction Syntax:

OP code	Operand	Instruction Operation
IDC&		HL ← HL+1
IDC%		ZR ← ZR+1
IDC$		HL ← HL+1 ZR ← ZR+1

IDC8

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Increment the contents of HL or ZR by 2 at the same time or separately.

Note:
This instruction does not affect the content of AC during execution.

Instruction Syntax:

OP code	Operand	Instruction Operation
IDC8&		HL ← HL+2
IDC8%		ZR ← ZR+2
IDC8$		HL ← HL+2 ZR ← ZR+2

IDCH

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Increment the contents of HL or ZR by 4 at the same time or separately.

Note:
This instruction does not affect the content of AC during execution.

Instruction Syntax:

OP code	Operand	Instruction Operation
IDCH&		HL ← HL+4
IDCH%		ZR ← ZR+4
IDCH$		HL ← HL+4 ZR ← ZR+4

MVL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to HL-L.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of HL-L	IDBF3	IDBF2	IDBF1	IDBF0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MVL	Rx	IDBF3 ~ IDBF0 ← (Rx)

MVH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to HL-H.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of HL-H	IDBF7	IDBF6	IDBF5	IDBF4

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MVH	Rx	IDBF7 ~ IDBF4 ← (Rx)

MVU

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to HL-U.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of HL-U	IDBF11	IDBF10	IDBF9	IDBF8

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MVU	Rx	IDBF11 ~ IDBF8 ← (Rx)

MVV

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to HL-V.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of HL-V	IDBF15	IDBF14	IDBF13	IDBF12

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MVV	Rx	IDBF15 ~ IDBF12 ← (Rx)

RVL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of HL-L to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of HL-L	IDBF3	IDBF2	IDBF1	IDBF0
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RVL	Rx	(Rx) ← IDBF3 ~ IDBF0, AC ← IDBF3 ~ IDBF0

RVH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of HL-H to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of HL-H	IDBF7	IDBF6	IDBF5	IDBF4
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RVH	Rx	(Rx) ← IDBF7 ~ IDBF4, AC ← IDBF7 ~ IDBF4

RVU

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of register HL-U to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of HL-U	IDBF11	IDBF10	IDBF9	IDBF8
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RVU	Rx	(Rx) ← IDBF11 ~ IDBF8, AC ← IDBF11 ~ IDBF8

RVV

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of HL-V to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of HL-V	IDBF15	IDBF14	IDBF13	IDBF12
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RVV	Rx	(Rx) ← IDBF15 ~ IDBF12, AC ← IDBF15 ~ IDBF12

SHLX

Instruction Characteristics:
Two-word instruction, 8 machine cycles.

Instruction Explanation:
Write the 16-bit Immediate data (D) to HL.

This instruction is 2-word in length and takes 8 machine cycles to complete. The 1st character in the instruction is the OPcode while the 16-bit long Immediate data is in the instruction's 2nd character.

The bit data correspondence is as follows:

Imm. Data D	D7	D6	D5	D4	D3	D2	D1	D0
HL register	IDBF7	IDBF6	IDBF5	IDBF4	IDBF3	IDBF2	IDBF1	IDBF0
Imm. Data D	D15	D14	D13	D12	D11	D10	D9	D8
HL register	IDBF15	IDBF14	IDBF13	IDBF12	IDBF11	IDBF10	IDBF9	IDBF8

Notes:
(1). D = 0h ~ FFFFh (maximum value will vary according to the number of bits in each MCU's HL register)
(2). MCU will temporarily suspend all interrupt requests during the 8 machine cycles used for executing this instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
SHLX SETDAT	D	IDBF15~0 ← D

MRL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of data RAM pointed to by Rx to @ZR-L.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of ZR-L	ZRBF3	ZRBF2	ZRBF1	ZRBF0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRL	Rx	ZRBF3 ~ ZRBF0 ← (Rx)

MRH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to ZR-H.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of ZR-H	ZRBF7	ZRBF6	ZRBF5	ZRBF4

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRH	Rx	ZRBF7 ~ ZRBF4 ← (Rx)

MRU

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to ZR-U.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of ZR-U	ZRBF11	ZRBF10	ZRBF9	ZRBF8

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRU	Rx	ZRBF11 ~ ZRBF8 ← (Rx)

MRV

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of data RAM pointed to by Rx to ZR-V.

The bit data correspondence is as follows:

Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of ZR-V	0	0	0	ZRBF12

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRV	Rx	ZRBF12 ← (Rx)

RRL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of ZR-L to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of ZR-L	ZRBF3	ZRBF2	ZRBF1	ZRBF0
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RRL	Rx	(Rx) ← ZRBF3 ~ ZRBF0, AC ← ZRBF3 ~ ZRBF0

RRH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of ZR-H to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of ZR-H	ZRBF7	ZRBF6	ZRBF5	ZRBF4
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RRH	Rx	(Rx) ← ZRBF7 ~ ZRBF4, AC ← ZRBF7 ~ ZRBF4

RRU

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of ZR-U to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of ZR-U	ZRBF11	ZRBF10	ZRBF9	ZRBF8
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RRU	Rx	(Rx) ← ZRBF11 ~ ZRBF8, AC ← ZRBF11 ~ ZRBF8

RRV

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of ZR-V to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of ZR-V	0	0	0	ZRBF12
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RRV	Rx	(Rx) ← 0,0,0,ZRBF12, AC ← 0,0,0,ZRBF12

SZRX

Instruction Characteristics:
Two-word instruction, 8 machine cycles.

Instruction Explanation:
Write the 16-bit Immediate data (D) to ZR.

This instruction uses an instruction 2-character in length and takes 8 machine cycles to complete. The 1st character in the instruction is the Opcode while the 13-bit long Immediate data is in the instruction's 2nd character.

The bit data correspondence is as follows:

Imm. Data D	D7	D6	D5	D4	D3	D2	D1	D0
ZR register	ZRBF7	ZRBF6	ZRBF5	ZRBF4	ZRBF3	ZRBF2	ZRBF1	ZRBF0
Imm. Data D				D12	D11	D10	D9	D8
ZR register				ZRBF12	ZRBF11	ZRBF10	ZRBF9	ZRBF8

Notes:
(1). D = 0h ~ 1FFFh (maximum value will vary according to the number of bits in each MCU's ZR register)
(2). MCU will temporarily suspend all interrupt requests during the 8 machine cycles used for executing this instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
SZRX SETDAT	D	ZRBF12~0 ← D

MHL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, up to 16 bits of data transferred.

Instruction Explanation:
Write the content of the 16-bit data backup register pointed to by operand X to the HL index register.

Each X value represents the data RAM at 4 contiguous addresses. Their correspondence is as follows:

X	Addr+3	Addr+2	Addr+1	Addr
0	$0083	$0082	$0081	$0080
1	$0087	$0086	$0085	$0084
2	$008B	$008A	$0089	$0088
3	$008F	$008E	$008D	$008C
4	$0093	$0092	$0091	$0090
5	$0097	$0096	$0095	$0094
6	$009B	$009A	$0099	$0098
7	$009F	$009E	$009D	$009C
8	$00A3	$00A2	$00A1	$00A0
9	$00A7	$00A6	$00A5	$00A4
A	$00AB	$00AA	$00A9	$00A8
B	$00AF	$00AE	$00AD	$00AC
C	$00B3	$00B2	$00B1	$00B0
D	$00B7	$00B6	$00B5	$00B4
E	$00BB	$00BA	$00B9	$00B8
F	$00BF	$00BE	$00BD	$00BC

The bit data correspondence is as follows:

Content of HL	IDBF7	IDBF6	IDBF5	IDBF4	IDBF3	IDBF2	IDBF1	IDBF0
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+1				Addr			
Content of HL	IDBF15	IDBF14	IDBF13	IDBF12	IDBF11	IDBF10	IDBF9	IDBF8
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+3				Addr+2			

Note:
X = 0 ~ Fh

Instruction Syntax:

OP code	Operand	Instruction Operation
MHL	X	IDBF3~0 ← (Addr), IDBF7~4 ← (Addr +1), IDBF11~8 ← (Addr +2), IDBF15~12 ← (Addr +3)

RHL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 16 bits of data transferred.

Instruction Explanation:
Store the 16-bit content of the HL index register to the 16-bit data backup register pointed to by operand X.

Each X value represents the data RAM at 4 contiguous addresses. Their correspondence is as follows:

X	Addr+3	Addr+2	Addr+1	Addr
0	$0083	$0082	$0081	$0080
1	$0087	$0086	$0085	$0084
2	$008B	$008A	$0089	$0088
3	$008F	$008E	$008D	$008C
4	$0093	$0092	$0091	$0090
5	$0097	$0096	$0095	$0094
6	$009B	$009A	$0099	$0098
7	$009F	$009E	$009D	$009C
8	$00A3	$00A2	$00A1	$00A0
9	$00A7	$00A6	$00A5	$00A4
A	$00AB	$00AA	$00A9	$00A8
B	$00AF	$00AE	$00AD	$00AC
C	$00B3	$00B2	$00B1	$00B0
D	$00B7	$00B6	$00B5	$00B4
E	$00BB	$00BA	$00B9	$00B8
F	$00BF	$00BE	$00BD	$00BC

The bit data correspondence is as follows:

Content of HL	IDBF7	IDBF6	IDBF5	IDBF4	IDBF3	IDBF2	IDBF1	IDBF0
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+1				Addr			
Content of HL	IDBF15	IDBF14	IDBF13	IDBF12	IDBF11	IDBF10	IDBF9	IDBF8
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+3				Addr+2			

Note:
X = 0 ~ Fh

Instruction Syntax:

OP code	Operand	Instruction Operation
RHL	X	(Addr) ← IDBF3~0, (Addr+1) ← IDBF7~4, (Addr+2) ← IDBF11~8, (Addr+3) ← IDBF15~12

MZR

Instruction Characteristics:
Single-word instruction, 4 machine cycles, up to 13 bits of data transferred.

Instruction Explanation:
Write the content of the 16-bit data backup register pointed to by operand X to the ZR index register.

Each X value represents the data RAM at 4 contiguous addresses. Their correspondence is as follows:

X	Addr+3	Addr+2	Addr+1	Addr
0	$00C3	$00C2	$00C1	$00C0
1	$00C7	$00C6	$00C5	$00C4
2	$00CB	$00CA	$00C9	$00C8
3	$00CF	$00CE	$00CD	$00CC
4	$00D3	$00D2	$00D1	$00D0
5	$00D7	$00D6	$00D5	$00D4
6	$00DB	$00DA	$00D9	$00D8
7	$00DF	$00DE	$00DD	$00DC
8	$00E3	$00E2	$00E1	$00E0
9	$00E7	$00E6	$00E5	$00E4
A	$00EB	$00EA	$00E9	$00E8
B	$00EF	$00EE	$00ED	$00EC
C	$00F3	$00F2	$00F1	$00F0
D	$00F7	$00F6	$00F5	$00F4
E	$00FB	$00FA	$00F9	$00F8
F	$00FF	$00FE	$00FD	$00FC

The bit data correspondence is as follows:

Content of ZR	ZRBF7	ZRBF6	ZRBF5	ZRBF4	ZRBF3	ZRBF2	ZRBF1	ZRBF0
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+1				Addr			
Content of ZR	NA	NA	NA	ZRBF12	ZRBF11	ZRBF10	ZRBF9	ZRBF8
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+3				Addr+2			

Note:
X = 0 ~ Fh

Instruction Syntax:

OP code	Operand	Instruction Operation
MZR	X	ZRBF3~0 ← (Addr), ZRBF7~4 ← (Addr+1), ZRBF11~8 ← (Addr+2), ZRBF12 ← (Addr+3)

RZR

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 16 bits of data transferred.

Instruction Explanation:
Store the 13-bit content of the ZR index register to the 16-bit data backup register pointed to by operand X.

Each X value represents the data RAM at 4 contiguous addresses. Their correspondence is as follows:

X	Addr+3	Addr+2	Addr+1	Addr
0	$00C3	$00C2	$00C1	$00C0
1	$00C7	$00C6	$00C5	$00C4
2	$00CB	$00CA	$00C9	$00C8
3	$00CF	$00CE	$00CD	$00CC
4	$00D3	$00D2	$00D1	$00D0
5	$00D7	$00D6	$00D5	$00D4
6	$00DB	$00DA	$00D9	$00D8
7	$00DF	$00DE	$00DD	$00DC
8	$00E3	$00E2	$00E1	$00E0
9	$00E7	$00E6	$00E5	$00E4
A	$00EB	$00EA	$00E9	$00E8
B	$00EF	$00EE	$00ED	$00EC
C	$00F3	$00F2	$00F1	$00F0
D	$00F7	$00F6	$00F5	$00F4
E	$00FB	$00FA	$00F9	$00F8
F	$00FF	$00FE	$00FD	$00FC

The bit data correspondence is as follows:

Content of ZR	ZRBF7	ZRBF6	ZRBF5	ZRBF4	ZRBF3	ZRBF2	ZRBF1	ZRBF0
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+1				Addr			
Content of ZR	NA	NA	NA	ZRBF12	ZRBF11	ZRBF10	ZRBF9	ZRBF8
RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
RAM addr.	Addr+3				Addr+2			

Note:
X = 0 ~ Fh

Instruction Syntax:

OP code	Operand	Instruction Operation
RZR	X	(Addr) ← ZRBF3~0, (Addr+1) ← ZRBF7~4, (Addr+2) ← ZRBF11~8, (Addr+3) ← ZRBF12

2-2. RAM Data Transfer Instructions

LID

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the data RAM pointed to by @HL or @ZR to the data RAM address pointed to by @ZR or @HL.

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.

Instruction Syntax:

OP code	Operand	Instruction Operation
LID	@ZR, @HL	(@ZR) ← (@HL)
LID&	@ZR, @HL	(@ZR) ← (@HL) HL ← HL+1
LID%	@ZR, @HL	(@ZR) ← (@HL) ZR ← ZR+1
LID$	@ZR, @HL	(@ZR) ← (@HL) HL ← HL+1 ZR ← ZR+1
LID	@HL, @ZR	(@HL) ← (@ZR)
LID&	@HL, @ZR	(@HL) ← (@ZR) HL ← HL+1
LID%	@HL, @ZR	(@HL) ← (@ZR) ZR ← ZR+1
LID$	@HL, @ZR	(@HL) ← (@ZR) HL ← HL+1, ZR ← ZR+1

LID8

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits of data transferred.

Instruction Explanation:
Store the 8-bit content from the two contiguous addresses at the data RAM pointed to by @HL or @ZR to the two contiguous data RAM addresses pointed to by @ZR or @HL.

Regardless of what the actual LSB value of @HL or @ZR may be, this instruction will ignore the actual LSB value during execution. The contiguous addresses accessed will be @HL/ZR (RAM address bit0=0) and @HL/ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Source RAM addr. & data mapping	RAM addr. = Ns*+1				RAM addr. = Ns*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Source 8bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Source RAM addr	@HL/ZR(RAM Address bit0=1)				@HL/ZR(RAM Address bit0=0)			
Destination RAM addr. & data mapping	RAM addr. = Nd*+1				RAM addr. = Nd*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Destination 8bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Destination RAM addr.	@ZR/HL(RAM Address bit0=1)				@ZR/HL(RAM Address bit0=0)			

***: Ns, Nd represents odd-numbered data RAM addresses.**

Notes:

(1). The contents of HL and ZR must be set before executing this instruction.

(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 2.

Instruction Syntax:

OP code	Operand	Instruction Operation
LID8	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL(RAM Address bit0=0)) (@ZR(RAM Address bit0=1)) ← (@HL(RAM Address bit0=1))
LID8&	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL(RAM Address bit0=0)) (@ZR(RAM Address bit0=1)) ← (@HL(RAM Address bit0=1)) HL ← HL+2
LID8%	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL(RAM Address bit0=0)) (@ZR(RAM Address bit0=1)) ← (@HL(RAM Address bit0=1)) ZR ← ZR+2
LID8$	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL(RAM Address bit0=0)) (@ZR(RAM Address bit0=1)) ← (@HL(RAM Address bit0=1)) HL ← HL+2 ZR ← ZR+2
LID8	@HL, @ZR	(@HL(RAM Address bit0=0)) ← (@ZR(RAM Address bit0=0)) (@HL(RAM Address bit0=1)) ← (@ZR(RAM Address bit0=1))
LID8&	@HL, @ZR	(@HL(RAM Address bit0=0)) ← (@ZR(RAM Address bit0=0)) (@HL(RAM Address bit0=1)) ← (@ZR(RAM Address bit0=1)) HL ← HL+2
LID8%	@HL, @ZR	(@HL(RAM Address bit0=0)) ← (@ZR(RAM Address bit0=0)) (@HL(RAM Address bit0=1)) ← (@ZR(RAM Address bit0=1)) ZR ← ZR+2
LID8$	@HL, @ZR	(@HL(RAM Address bit0=0)) ← (@ZR(RAM Address bit0=0)) (@HL(RAM Address bit0=1)) ← (@ZR(RAM Address bit0=1)) HL ← HL+2, ZR ← ZR+2

LIDH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 16 bits of data transferred.

Instruction Explanation:
Store the 16-bit content from the four contiguous addresses at the data RAM pointed to by @HL or @ZR to the four contiguous data RAM addresses pointed to by @ZR or @HL.

Regardless of what the actual value of the two least significant bits of @HL or @ZR may be, they will be ignored during execution of this instruction. The contiguous four addresses accessed will be @HL/ZR(RAM address bit1,0=00), @HL/ZR(RAM address bit1,0=01), @HL/ZR(RAM address bit1,0=10) and @HL/ZR(RAM address bit1,0=11)

The bit data correspondence is as follows:

Source RAM addr. & data mapping	RAM addr. = Ns*+1				RAM addr. = Ns*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Source 16bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Source RAM addr	@HL/ZR(RAM Address bit1,0=01)				@HL/ZR(RAM Address bit1,0=00)			
Destination RAM addr. & data mapping	RAM addr. = Nd*+1				RAM addr. = Nd*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Destination 16bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Destination RAM addr.	@ZR/HL(RAM Address bit1,0=01)				@ZR/HL(RAM Address bit1,0=00)			

Source RAM addr. & data mapping	RAM addr. = Ns*+3				RAM addr. = Ns*+2			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Source 16bits data	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Destination RAM addr	@HL/ZR(RAM Address bit1,0=11)				@HL/ZR(RAM Address bit1,0=10)			
Destination RAM addr. & data mapping	RAM addr. = Nd*+3				RAM addr. = Nd*+2			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Destination 16bits data	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Destination RAM addr.	@ZR/HL(RAM Address bit1,0=11)				@ZR/HL(RAM Address bit1,0=10)			

***: Ns Nd represents data RAM addresses that are multiples of four.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 4.

Instruction Syntax:

OP code	Operand	Instruction Operation
LIDH	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← (@HL(RAM Address bit1,0=00)) (@ZR(RAM Address bit1,0=01)) ← (@HL(RAM Address bit1,0=01)) (@ZR(RAM Address bit1,0=10)) ← (@HL(RAM Address bit1,0=10)) (@ZR(RAM Address bit1,0=11)) ← (@HL(RAM Address bit1,0=11))
LIDH&	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← (@HL(RAM Address bit1,0=00)) (@ZR(RAM Address bit1,0=01)) ← (@HL(RAM Address bit1,0=01)) (@ZR(RAM Address bit1,0=10)) ← (@HL(RAM Address bit1,0=10)) (@ZR(RAM Address bit1,0=11)) ← (@HL(RAM Address bit1,0=11)) HL ← HL+4
LIDH%	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← (@HL(RAM Address bit1,0=00)) (@ZR(RAM Address bit1,0=01)) ← (@HL(RAM Address bit1,0=01)) (@ZR(RAM Address bit1,0=10)) ← (@HL(RAM Address bit1,0=10)) (@ZR(RAM Address bit1,0=11)) ← (@HL(RAM Address bit1,0=11)) ZR ← ZR+4
LIDH$	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← (@HL(RAM Address bit1,0=00)) (@ZR(RAM Address bit1,0=01)) ← (@HL(RAM Address bit1,0=01)) (@ZR(RAM Address bit1,0=10)) ← (@HL(RAM Address bit1,0=10)) (@ZR(RAM Address bit1,0=11)) ← (@HL(RAM Address bit1,0=11)) HL ← HL+4 ZR ← ZR+4
LIDH	@HL, @ZR	(@HL(RAM Address bit1,0=00)) ← (@ZR(RAM Address bit1,0=00)) (@HL(RAM Address bit1,0=01)) ← (@ZR(RAM Address bit1,0=01)) (@HL(RAM Address bit1,0=10)) ← (@ZR(RAM Address bit1,0=10)) (@HL(RAM Address bit1,0=11)) ← (@ZR(RAM Address bit1,0=11))
LIDH&	@HL, @ZR	(@HL(RAM Address bit1,0=00)) ← (@ZR(RAM Address bit1,0=00)) (@HL(RAM Address bit1,0=01)) ← (@ZR(RAM Address bit1,0=01)) (@HL(RAM Address bit1,0=10)) ← (@ZR(RAM Address bit1,0=10)) (@HL(RAM Address bit1,0=11)) ← (@ZR(RAM Address bit1,0=11)) HL ← HL+4
LIDH%	@HL, @ZR	(@HL(RAM Address bit1,0=00)) ← (@ZR(RAM Address bit1,0=00)) (@HL(RAM Address bit1,0=01)) ← (@ZR(RAM Address bit1,0=01)) (@HL(RAM Address bit1,0=10)) ← (@ZR(RAM Address bit1,0=10)) (@HL(RAM Address bit1,0=11)) ← (@ZR(RAM Address bit1,0=11)) ZR ← ZR+4
LIDH$	@HL, @ZR	(@HL(RAM Address bit1,0=00)) ← (@ZR(RAM Address bit1,0=00)) (@HL(RAM Address bit1,0=01)) ← (@ZR(RAM Address bit1,0=01)) (@HL(RAM Address bit1,0=10)) ← (@ZR(RAM Address bit1,0=10)) (@HL(RAM Address bit1,0=11)) ← (@ZR(RAM Address bit1,0=11)) HL ← HL+4, ZR ← ZR+4

LDS

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the 4-bit Immediate data (D) to the data RAM pointed to by Rx, @HL or @ZR and AC. The bit data correspondence is as follows:

Immediate data	D3	D2	D1	D0
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.
(3). D = 0 ~ Fh.

Instruction Syntax:

OP code	Operand	Instruction Operation
LDS	Rx, D	(Rx) ← D, AC ← D
LDS	@HL, D	(@HL) ← D, AC ← D
LDS#	@HL, D	(@HL) ← D, AC ← D HL ← HL+1
LDS	@ZR, D	(@ZR) ← D, AC ← D
LDS#	@ZR, D	(@ZR) ← D, AC ← D ZR ← ZR+1

LDS8

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits of data transferred.

Instruction Explanation:
Store the 8-bit Immediate data (D) to the data RAM at the two contiguous addresses pointed to by @HL or @ZR.

Regardless of what the actual LSB value of HL or ZR may be, this instruction will ignore the actual LSB value during execution. The two contiguous addresses accessed will be @HL/ZR (RAM address bit0=0) and @HL/ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Imm. data D	D7	D6	D5	D4	D3	D2	D1	D0
Destination RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
Destination RAM addr.	@HL/ZR(RAM Address bit0=1)				@HL/ZR(RAM Address bit0=0)			

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 2.
(2). D = 0 ~ FFh

Instruction Syntax:

OP code	Operand	Instruction Operation
LDS8	@HL, D	(@HL(RAM Address bit0=0)) ← D3~D0, (@HL(RAM Address bit0=0)) ← D7~D4,
LDS8#	@HL, D	(@HL(RAM Address bit0=0)) ← D3~D0, (@HL(RAM Address bit0=0)) ← D7~D4, HL ← HL+2
LDS8	@ZR, D	(@ZR(RAM Address bit0=0)) ← D3~D0, (@ZR(RAM Address bit0=1)) ← D7~D4,
LDS8#	@ZR, D	(@ZR(RAM Address bit0=0)) ← D3~D0, (@ZR(RAM Address bit0=1)) ← D7~D4, ZR ← ZR+2

LDSH

Instruction Characteristics:
Two-word instruction, 8 machine cycles, 16 bits of data transferred.

Instruction Explanation:
Store the 16-bit Immediate data (D) to the data RAM at the four contiguous addresses pointed to by @HL or @ZR.

This instruction uses an instruction 2-character in length and takes 8 machine cycles to complete. The 1st character in the instruction is the OPcode and the 1st operand, while the 16-bit long Immediate data is in the instruction's 2nd character.

Regardless of what the actual value of the two least significant bits of HL or ZR may be, they will be ignored during execution of this instruction. The four contiguous addresses accessed will be @HL/ZR(RAM address bit1,0=10), @HL/ZR(RAM address bit1,0=01), @HL/ZR(RAM address bit 1,0=10) and @HL/ZR (RAM address bit1,0=11).

The bit data correspondence is as follows:

Imm. Data D	D7	D6	D5	D4	D3	D2	D1	D0
Destination RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
Destination RAM addr.	@HL/ZR(RAM Address bit1,0=01)				@HL/ZR(RAM Address bit1,0=00)			
Imm. Data D	D15	D14	D13	D12	D11	D10	D9	D8
Destination RAM data	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
Destination RAM addr.	@HL/ZR(RAM Address bit1,0=11)				@HL/ZR(RAM Address bit1,0=10)			

Notes:
(1). D = 0h ~ FFFFh.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 4.

(3). MCU will temporarily suspend all interrupt requests during the 8 machine cycles used for executing this instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
LDSH SETDAT	@HL, D	(@HL(RAM Address bit1,0=00)) ← D3~D0, (@HL(RAM Address bit1,0=01)) ← D7~D4, (@HL(RAM Address bit1,0=10)) ← D11~D8, (@HL(RAM Address bit1,0=11)) ← D15~D12,
LDSH# SETDAT	@HL, D	(@HL(RAM Address bit1,0=00)) ← D3~D0, (@HL(RAM Address bit1,0=01)) ← D7~D4, (@HL(RAM Address bit1,0=10)) ← D11~D8, (@HL(RAM Address bit1,0=11)) ← D15~D12, HL ← HL+4
LDSH SETDAT	@ZR, D	(@ZR(RAM Address bit1,0=00)) ← D3~D0, (@ZR(RAM Address bit1,0=01)) ← D7~D4, (@ZR(RAM Address bit1,0=10)) ← D11~D8, (@ZR(RAM Address bit1,0=11)) ← D15~D12,
LDSH# SETDAT	@ZR, D	(@ZR(RAM Address bit1,0=00)) ← D3~D0, (@ZR(RAM Address bit1,0=01)) ← D7~D4, (@ZR(RAM Address bit1,0=10)) ← D11~D8, (@ZR(RAM Address bit1,0=11)) ← D15~D12, ZR ← ZR+4

STA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of AC to the data RAM pointed to by Rx, @HL or @ZR.

The bit data correspondence is as follows:

AC	AC3	AC2	AC1	AC0
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
STA	Rx	(Rx) ← AC
STA	@HL	(@HL) ← AC
STA#	@HL	(@HL) ← AC HL ← HL+1
STA	@ZR	(@ZR) ← AC
STA#	@ZR	(@ZR) ← AC ZR ← ZR+1

MRW

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the data RAM pointed to by Rx to the data RAM pointed to by @HL or @ZR or Ry and AC.

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx and Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRW	Ry, Rx	(Ry) ← (Rx), AC ← (Rx)
MRW	@HL, Rx	(@HL) ← (Rx), AC ← (Rx)
MRW#	@HL, Rx	(@HL) ← (Rx), AC ← (Rx) HL ← HL+1
MRW	@ZR, Rx	(@ZR) ← (Rx), AC ← (Rx)
MRW#	@ZR, Rx	(@ZR) ← (Rx), AC ← (Rx) ZR ← ZR+1

MWR

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the data RAM pointed to by @HL or @ZR or Ry to the data RAM pointed to by Rx and AC.

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx and Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MWR	Rx, Ry	(Rx) ← (Ry), AC ← (Ry)
MWR	Rx, @HL	(Rx) ← (@HL), AC ← (@HL)
MWR#	Rx, @HL	(Rx) ← (@HL), AC ← (@HL)

OP code	Operand	Instruction Operation
		HL ← HL+1
MWR	Rx, @ZR	(Rx) ← (@ZR), AC ← (@ZR)
MWR#	Rx, @ZR	(Rx) ← (@ZR), AC ← (@ZR) ZR ← ZR+1

LDA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx, @HL or @ZR to AC.

The bit data correspondence is as follows:

Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
LDA	Rx	AC ← (Rx)
LDA	@HL	AC ← (@HL)
LDA#	@HL	AC ← (@HL), HL ← HL + 1
LDA	@ZR	AC ← (@ZR)
LDA#	@ZR	AC ← (@ZR), ZR ← ZR + 1

2-3. LCD Instructions

LCT

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits (write) and 4 bits(read) of data transferred.

Instruction Explanation:
Pass the content of the data RAM pointed to by Ry or @HL to the 7-segment decoder to generate 8 bits of data (DBUSH~DBUSA) then store these 8 bits of data to the two contiguous addresses pointed to by @ZR or the address of the LCD display memory pointed to by Lz.

Regardless of what the actual LSB value of @ZR may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @ZR (RAM address bit0=0) and @ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Destination RAM addr. & data mapping	RAM addr. = N*+1				RAM addr. = N*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
7-seg decoder output data	DBUSH	DBUSG	DBUSF	DBUSE	DBUSD	DBUSC	DBUSB	DBUSA
@ZR addr.	@ZR(RAM Address bit0=1)				@ZR(RAM Address bit0=0)			
LZ addr.	0100H + Lz x 2 + 1				0100H + Lz x 2 + 0			

***: N represents odd-numbered data RAM addresses.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). Absolute addressing must be used for Ry and Lz syntax.
(3). After executing this instruction, certain syntax automatically increments the content of ZR by 2 or the content of HL by 1.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCT	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (Ry) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (Ry)
LCT#	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (Ry) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (Ry) ZR ← ZR+2
LCT	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL)
LCT&	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) HL ← HL+1
LCT%	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) ZR ← ZR+2
LCT$	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) HL ← HL+1 ZR ← ZR+2
LCT	Lz, Ry	(Lz) ← 7-segment decoder ← (Ry)
LCT	Lz, @HL	(Lz) ← 7-segment decoder ← (@HL)
LCT#	Lz, @HL	*Lz) ← 7-segment decoder ← (@HL) HL ← HL+1

LCB

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits (write) and 4 bits(read) of data transferred.

Instruction Explanation:
Pass the content of the data RAM pointed to by Ry or @HL to the 7-segment decoder to generate 8 bits of data then store these 8 bits of data to the two contiguous addresses pointed to by @ZR or the address of the LCD display memory pointed to by Lz.

If the content of the data RAM pointed to by Ry or @HL is 0h, then the 8 bits of data (DBUSH~A) generated by the 7-segment decoder will be 00h. Regardless of what the actual LSB value of @ZR may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @ZR (RAM address bit0=0) and @ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Destination RAM addr. & data mapping	RAM addr. = N*+1				RAM addr. = N*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
7-seg decoder output data	DBUSH	DBUSG	DBUSF	DBUSE	DBUSD	DBUSC	DBUSB	DBUSA
@ZR addr.	@ZR(RAM Address bit0=1)				@ZR(RAM Address bit0=0)			
LZ addr.	0100H + Lz x 2 + 1				0100H + Lz x 2 + 0			

***: N represents odd-numbered data RAM addresses.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). Absolute addressing must be used for Ry and Lz syntax.
(3). After executing this instruction, certain syntax automatically increments the content of ZR by 2 or the content of HL by 1.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCB	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (Ry) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (Ry) 00 ← 7-segment decoder ← 0
LCB#	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (Ry) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (Ry) ZR ← ZR+2 00 ← 7-segment decoder ← 0
LCB	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) 00 ← 7-segment decoder ← 0
LCB&	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) HL ← HL+1 00 ← 7-segment decoder ← 0
LCB%	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL)

OP code	Operand	Instruction Operation
		ZR ← ZR+2 00 ← 7-segment decoder ← 0
LCB$	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← 7-segment decoder(DBUSD~A) ← (@HL) (@ZR(RAM Address bit0=1)) ← 7-segment decoder(DBUSH~E) ← (@HL) HL ← HL+1 ZR ← ZR+2 00 ← 7-segment decoder ← 0
LCB	Lz, Ry	(Lz) ← 7-segment decoder ← (Ry) 00 ← 7-segment decoder ← 0
LCB	Lz, @HL	(Lz) ← 7-segment decoder ← (@HL) 00 ← 7-segment decoder ← 0
LCB#	Lz, @HL	(Lz) ← 7-segment decoder ← (@HL) HL ← HL+1 00 ← 7-segment decoder ← 0

LCP

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits (write) and 4 bits(read) of data transferred.

Instruction Explanation:
Store the content of the data RAM pointed to by Ry or @HL and the content at AC to the two contiguous data RAM addresses pointed to by @ZR or the LCD display memory address pointed to by Lz.

Regardless of what the actual LSB value of @ZR may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @ZR (RAM address bit0=0) and @ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Destination RAM addr. & data mapping	RAM addr. = N*+1				RAM addr. = N*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Output 8bits Data	AC3	AC2	AC2	AC0	(Ry)3/ (@HL)3	(Ry2)/ (@HL)2	(Ry)1/ (@HL)1	(Ry)0/ (@HL)0
@ZR addr.	@ZR(RAM Address bit0=1)				@ZR(RAM Address bit0=0)			
LZ addr.	0100H + Lz x 2 + 1				0100H + Lz x 2 + 0			

***: N represents odd-numbered data RAM addresses.**

Notes:
(1). The content of ZR or HL must be set before executing this instruction.
(2). Absolute addressing must be used for Ry and Lz syntax.
(3). After executing this instruction, certain syntax automatically increments the content of ZR by 2 or the content of HL by 1.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCP	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← (Ry) (@ZR(RAM Address bit0=1)) ← AC

OP code	Operand	Instruction Operation
LCP#	@ZR, Ry	(@ZR(RAM Address bit0=0)) ← (Ry) (@ZR(RAM Address bit0=1)) ← AC ZR ← ZR+2
LCP	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL) (@ZR(RAM Address bit0=1)) ← AC
LCP&	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL) (@ZR(RAM Address bit0=1)) ← AC HL ← HL+1
LCP%	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL) (@ZR(RAM Address bit0=1)) ← AC ZR ← ZR+2
LCP$	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← (@HL) (@ZR(RAM Address bit0=1)) ← AC HL ← HL+1, ZR ← ZR+2
LCP	Lz, Ry	$(Lz)_{low\ nibble}$ ← (Ry), $(Lz)_{high\ nibble}$ ← AC
LCP	Lz, @HL	$(Lz)_{low\ nibble}$ ← (@HL), $(Lz)_{high\ nibble}$ ← AC
LCP#	Lz, @HL	$(Lz)_{low\ nibble}$ ← (@HL), $(Lz)_{high\ nibble}$ ← AC HL ← HL+1

LCD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits of data transferred.

Instruction Explanation:
This instruction stores the 8 bits content of the table ROM pointed to by @HL to the two contiguous data RAM addresses pointed to by @ZR or the LCD display memory address pointed to by Lz.

Regardless of what the actual LSB value of @ZR may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @ZR (RAM address bit0=0) and @ZR (RAM address bit0=1).

The bit data correspondence is as follows:

Destination RAM addr. & data mapping	RAM addr. = N*+1				RAM addr. = N*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Table ROM output 8bits data	T(@HL)7	T(@HL)6	T(@HL)5	T(@HL)4	T(@HL)3	T(@HL)2	T(@HL)1	T(@HL)0
@ZR addr.	@ZR(RAM Address bit0=1)				@ZR(RAM Address bit0=0)			
LZ addr.	0100H + Lz x 2 + 1				0100H + Lz x 2 + 0			

***: N represents odd-numbered data RAM addresses.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). Absolute addressing must be used for Lz syntax.
(3). After executing this instruction, certain syntax automatically increments the content of ZR by 2 or content L of H by 1.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCD	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← T(@HL)$_{low\ nibble}$, (@ZR(RAM Address bit0=1)) ← T(@HL)$_{high\ nibble}$,
LCD&	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← T(@HL)$_{low\ nibble}$, (@ZR(RAM Address bit0=1)) ← T(@HL)$_{high\ nibble}$, HL ← HL+1
LCD%	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← T(@HL)$_{low\ nibble}$, (@ZR(RAM Address bit0=1)) ← T(@HL)$_{high\ nibble}$, ZR ← ZR+2
LCD$	@ZR, @HL	(@ZR(RAM Address bit0=0)) ← T(@HL)$_{low\ nibble}$, (@ZR(RAM Address bit0=1)) ← T(@HL)$_{high\ nibble}$, HL ← HL+1 ZR ← ZR+2
LCD	Lz, @HL	(Lz) ← T(@HL)
LCD#	Lz, @HL	(Lz) ← T(@HL) HL ← HL+1

LCE

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 8 bits of data transferred.

Instruction Explanation:
Store the 8-bit content from the two contiguous addresses at the data RAM pointed to by @HL or @ZR to LCD display memory address pointed to by Lz.

Regardless of what the actual LSB value of @HL or @ZR may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @ZR (RAM address bit0=0), @ZR(RAM address bit0=1) or @HL(RAM Address bit0=0), @HL (RAM address bit0=1).

The bit data correspondence is as follows:

Source RAM addr. & data mapping	RAM addr. = Ns*+1				RAM addr. = Ns*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Source 8bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Source RAM addr	@HL/ZR(RAM Address bit0=1)				@HL/ZR(RAM Address bit0=0)			
Destination RAM addr. & data mapping	RAM addr. = Nd*+1				RAM addr. = Nd*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Destination 8bits data	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Destination RAM addr.	0100H + Lz x 2 + 1				0100H + Lz x 2 + 0			

***: Ns, Nd represents odd-numbered data RAM addresses.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 2.
(3). Absolute addressing must be used for Lz syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCE	Lz, @HL	$(Lz)_{low\ nibble}$ ← (@HL(RAM Address bit0=0)) $(Lz)_{high\ nibble}$ ← (@HL(RAM Address bit0=1))
LCE#	Lz, @HL	$(Lz)_{low\ nibble}$ ← (@HL(RAM Address bit0=0)) $(Lz)_{high\ nibble}$ ← (@HL(RAM Address bit0=1)) HL ← HL+2
LCE	Lz, @ZR	$(Lz)_{low\ nibble}$ ← (@ZR(RAM Address bit0=0)) $(Lz)_{high\ nibble}$ ← (@ZR(RAM Address bit0=1))
LCE#	Lz, @ZR	$(Lz)_{low\ nibble}$ ← (@ZR(RAM Address bit0=0)) $(Lz)_{high\ nibble}$ ← (@ZR(RAM Address bit0=1)) ZR ← ZR+2

2-4. Register Access Instructions

SMUI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx, @HL or @ZR to MUI for use as the multiplier during the multiplication operation.

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SMUI	Rx	MUI ← (Rx)
SMUI	@HL	MUI ← (@HL)
SMUI#	@HL	MUI ← (@HL) HL ← HL+1
SMUI	@ZR	MUI ← (@ZR)
SMUI#	@ZR	MUI ← (@ZR) ZR ← ZR+1

MMH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of MU (the 4 upper bits of the multiplication operation result) to the data RAM pointed to by Rx, @HL or @ZR and AC.

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MMH	Rx	(Rx) ← MU, AC ← MU
MMH	@HL	(@HL) ← MU, AC ← MU
MMH#	@HL	(@HL) ← MU, AC ← MU HL ← HL+1
MMH	@ZR	(@ZR) ← MU,

OP code	Operand	Instruction Operation
		AC ← MU
MMH#	@ZR	(@ZR) ← MU, AC ← MU ZR ← ZR+1

MWM

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Pass the content of the data RAM (work register) pointed to by Ry to the register pointed to by RM (on the EV chip (TM8999) this is an expansion IO port).

The bit data correspondence is as follows:

Source RAM data	(Ry)3	(Ry)2	(Ry)1	(Ry)0
Register Content of Rm	(Rm)3	(Rm)2	(Rm)1	(Rm)0

Notes:
(1). Rm = 0 ~ F
(2). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MWM	Rm, Ry	(Rm) ← (Ry)

MMW

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Pass the content of the register pointed to by Rm (on the EV chip (TM8999) this is an expansion IO port) to the data RAM pointed to by Ry and AC.

The bit data correspondence is as follows:

Content of Rm register	(Rm)3	(Rm)2	(Rm)1	(Rm)0
Destination RAM data	(Ry)3	(Ry)2	(Ry)1	(Ry)0
Destination AC register	AC3	AC2	AC1	AC0

Notes:
(1). Rm = 0 ~ F
(2). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MMW	Ry, Rm	(Ry) ← (Rm) AC ← (Rm)

LSP

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the Stack Pointer to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Stack Pointer	SP3	SP2	SP1	SP0
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
LSP	Rx	(Rx) ← STACK pointer, AC ← STACK pointer

MAF

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS1 register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS1	CF	Zero	SCF12	SCF11
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MAF	Rx	(Rx) ← STS1, AC ← STS1

MRA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 1 bit of data transferred.

Instruction Explanation:
Store bit3 from the content of data RAM pointed to by Rx to CF.

The bit data correspondence is as follows:

Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Content of CF	CF	-	-	-

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRA	Rx	CF ← (Rx)3

MSB

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS2 register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS2	SCF3(IOD)	SCF2(HRx)	SCF1(IOC)	BCF(PSF)
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MSB	Rx	(Rx) ← STS2, AC ← STS2

MSC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS3 register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content STS3	SCF7(Pre-divider)	PH15	SCF5(TMR1)	SCF4(INT)
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MSC	Rx	(Rx) ← STS3, AC ← STS3

MCX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 3 bits of data transferred.

Instruction Explanation:
Store the content of the STS3X register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS3X	–	SCF0(IOA)	SCF6(TMR2)	SCF8(SKI)
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MCX	Rx	(Rx) ← STS3X, AC ← STS3X

MSD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS4 register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS4	LBF	RFOVF	WDF	CSF
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MSD	Rx	(Rx) ← STS4, AC ← STS4

MDX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS4X register to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS4X	SCF10 (TMR3)	INT	CX2	CX
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MDX	Rx	(Rx) ← STS4X, AC ← STS4X

MKI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the content of the STS5 register (KI1~KI4) to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Content of STS5	KI4	KI3	KI2	KI1
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MKI	Rx	(Rx) ← STS5, AC ← STS5

2-5. Arithmetic/Logic Operation Instructions

MULH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary multiplication instruction. The 4-bit content of the data RAM pointed to by Rx, @HL or @ZR is set as the multiplicand then multiplied against the content of MUI (multiplier) in a multiplication operation. The 8-bit operation result is stored to MU and AC. The multiplication result and its correspondence with the MU and AC bits data are as follows:

Result	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Destination register	MU3	MU2	MU1	MU0	AC3	AC2	AC1	AC0

Notes:
(1). Before executing this instruction make sure that the multiplicand and multiplier are both in binary format. The result of the operation will be in the binary format as well.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MULH	Rx	MU, AC ← (Rx) * MUI
MULH	@HL	MU, AC ← (@HL) * MUI
MULH#	@HL	MU, AC ← (@HL) * MUI HL ← HL+1
MULH	@ZR	MU, AC ← (@ZR) * MUI
MULH#	@ZR	MU, AC ← (@ZR) * MUI ZR ← ZR+1

MULD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a decimal multiplication instruction. The 4-bit content of the data RAM pointed to by Rx, @HL or @ZR is set as the multiplicand then multiplied against the content of MUI (multiplier) in a multiplication operation. The 8-bit operation result is stored to MU and AC. The multiplication result and its correspondence with the MU and AC register bits are as follows:

Result	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Destination register	MU3	MU2	MU1	MU0	AC3	AC2	AC1	AC0

Notes:
(1). Before executing this instruction make sure that the multiplicand and multiplier are both in decimal format. The result of the operation will be in the decimal format as well.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MULD	Rx	MU_{10}, AC_{10} ← $(Rx)_{10}$ * MUI_{10}
MULD	@HL	MU_{10}, AC_{10} ← $(@HL)_{10}$ * MUI_{10}
MULD#	@HL	MU_{10}, AC_{10} ← $(@HL)_{10}$ * MUI_{10} HL ← HL+1
MULD	@ZR	MU_{10}, AC_{10} ← $(@ZR)_{10}$ * MUI_{10}
MULD#	@ZR	MU_{10}, AC_{10} ← $(@ZR)_{10}$ * MUI_{10} ZR ← ZR+1

ADC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal addition operation instruction. The content of the data RAM pointed to by Rx, @HL or @ZR is added to the content of AC and CF in an additional operation. The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

The instruction distinguishes between binary or decimal addition using the existence of “DA” in the operand. Use of DA indicates decimal addition; its absence means use of binary addition.

Notes:
(1). Before executing this instruction make sure that the summand and addend are both in the same number system. The existence or absence of the “DA” in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The carried number in the result of this addition operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADC	Rx	AC ← (Rx) + AC +CF
ADC*	Rx	AC ← (Rx) + AC +CF, (Rx) ← (Rx) + AC +CF
ADC	@HL	AC ← (@HL) + AC +CF
ADC#	@HL	AC ← (@HL) + AC +CF HL ← HL+1
ADC*	@HL	AC ← (@HL) + AC +CF, (@HL) ← (@HL) + AC +CF

OP code	Operand	Instruction Operation
ADC*#	@HL	AC ← (@HL) + AC +CF, (@HL) ← (@HL) + AC +CF HL ← HL+1
ADC	@HL, DA	AC_{10} ← $(@HL)_{10}$ + AC_{10} +CF
ADC#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + AC_{10} +CF HL ← HL+1
ADC*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + AC_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + AC_{10} +CF
ADC*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + AC_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + AC_{10} +CF HL ← HL+1
ADC	@ZR	AC ← (@ZR) + AC +CF
ADC#	@ZR	AC ← (@ZR) + AC +CF ZR ← ZR+1
ADC*	@ZR	AC ← (@ZR) + AC +CF, (@ZR) ← (@ZR) + AC +CF
ADC*#	@ZR	AC ← (@ZR) + AC +CF, (@ZR) ← (@ZR) + AC +CF ZR ← ZR+1
ADC	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + AC_{10} +CF
ADC#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + AC_{10} +CF ZR ← ZR+1
ADC*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + AC_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + AC_{10} +CF
ADC*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + AC_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + AC_{10} +CF ZR ← ZR+1

ADCM

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal addition operation instruction. The content of the data RAM pointed to by Rx, @HL or @ZR is added to the content of MU (upper register for multiplication operation result) and CF in an addition operation.

The instruction distinguishes between binary or decimal addition using the existence of “DA” in the operand. Use of DA indicates decimal addition; its absence means use of binary addition.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this instruction make sure that the summand and addend are both in the same number system. The existence or absence of the “DA” in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The carried number in the result of this addition operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADCM	Rx	AC ← (Rx) + MU +CF
ADCM*	Rx	AC ← (Rx) + MU +CF, (Rx) ← (Rx) + MU +CF
ADCM	@HL	AC ← (@HL) + MU +CF
ADCM#	@HL	AC ← (@HL) + MU +CF HL ← HL+1
ADCM*	@HL	AC ← (@HL) + MU +CF, (@HL) ← (@HL) + MU +CF
ADCM*#	@HL	AC ← (@HL) + MU +CF, (@HL) ← (@HL) + MU +CF HL ← HL+1
ADCM	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} +CF
ADCM#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} +CF HL ← HL+1
ADCM*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + MU_{10} +CF
ADCM*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + MU_{10} +CF HL ← HL+1
ADCM	@ZR	AC ← (@ZR) + MU +CF
ADCM#	@ZR	AC ← (@ZR) + MU +CF ZR ← ZR+1
ADCM*	@ZR	AC ← (@ZR) + MU +CF, (@ZR) ← (@ZR) + MU +CF
ADCM*#	@ZR	AC ← (@ZR) + MU +CF, (@ZR) ← (@ZR) + MU +CF ZR ← ZR+1
ADCM	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} +CF
ADCM#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} +CF ZR ← ZR+1
ADCM*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MU_{10} +CF
ADCM*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MU_{10} +CF ZR ← ZR+1

SBC

Instruction Characteristics:

Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:

This is a binary or decimal subtraction operation instruction. The content (subtrahend) of AC and CF (borrow) is subtracted from the data RAM pointed to by Rx, @HL or @ZR (minuend).

The instruction distinguishes between binary or decimal subtraction using the existence of “DA” in the operand. Use of DA indicates decimal subtraction; its absence means use of binary subtraction.

The operation result will be stored to the AC register or the data RAM pointed to by Rx, @HL or @ZR.

Notes:

(1). Before executing this instruction make sure that the minuend and subtrahend are both in the same number system. The existence or absence of the “DA” in the operand will determine whether the result of the operation will be in the binary or decimal format.

(2). The borrowed number in the result of this subtraction operation will change the value of CF.

(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.

(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SBC	Rx	AC ← (Rx) + ACB +CF
SBC*	Rx	AC ← (Rx) + ACB +CF, (Rx) ← (Rx) + ACB +CF
SBC	@HL	AC ← (@HL) + ACB +CF
SBC#	@HL	AC ← (@HL) + ACB +CF HL ← HL+1
SBC*	@HL	AC ← (@HL) + ACB +CF (@HL) ← (@HL) + ACB +CF
SBC*#	@HL	AC ← (@HL) + ACB +CF, (@HL) ← (@HL) + ACB +CF HL ← HL+1
SBC	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +CF
SBC#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +CF HL ← HL+1
SBC*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + ACB_{10} +CF
SBC*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + ACB_{10} +CF HL ← HL+1
SBC	@ZR	AC ← (@ZR) + ACB +CF
SBC#	@ZR	AC ← (@ZR) + ACB +CF ZR ← ZR+1
SBC*	@ZR	AC ← (@ZR) + ACB +CF, (@ZR) ← (@ZR) + ACB +CF
SBC*#	@ZR	AC ← (@ZR) + ACB +CF, (@ZR) ← (@ZR) + ACB +CF ZR ← ZR+1
SBC	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +CF
SBC#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +CF ZR ← ZR+1
SBC*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + ACB_{10} +CF
SBC*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +CF, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + ACB_{10} +CF ZR ← ZR+1

SBCM

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal subtraction operation instruction. The content (subtrahend) of MU (upper register for multiplication operation result) and CF (borrow) is subtracted from the data RAM pointed to by Rx, @HL or @ZR (minuend)

The instruction distinguishes between binary or decimal subtraction using the existence of "DA" in the operand. Use of DA indicates decimal subtraction; its absence means use of binary subtraction.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this instruction make sure that the summand and addend are both in the same number system. The existence or absence of the "DA" in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The borrowed numbered number in the result of this subtraction operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SBCM	Rx	AC ← (Rx) + MUB +CF
SBCM*	Rx	AC ← (Rx) + MUB +CF, (Rx) ← (Rx) + MUB +CF
SBCM	@HL	AC ← (@HL) + MUB +CF
SBCM#	@HL	AC ← (@HL) + MUB +CF HL ← HL+1
SBCM*	@HL	AC ← (@HL) + MUB +CF (@HL) ← (@HL) + MUB +CF
SBCM*#	@HL	AC ← (@HL) + MUB +CF, (@HL) ← (@HL) + MUB +CF HL ← HL+1
SBCM	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +CF
SBCM#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +CF HL ← HL+1
SBCM*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + MUB_{10} +CF
SBCM*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +CF, $(@HL)_{10}$ ← $(@HL)_{10}$ + MUB_{10} +CF HL ← HL+1
SBCM	@ZR	AC ← (@ZR) + MUB +CF
SBCM#	@ZR	AC ← (@ZR) + MUB +CF ZR ← ZR+1
SBCM*	@ZR	AC ← (@ZR) + MUB +CF, (@ZR) ← (@ZR) + MUB +CF
SBCM*#	@ZR	AC ← (@ZR) + MUB +CF, (@ZR) ← (@ZR) + MUB +CF ZR ← ZR+1

OP code	Operand	Instruction Operation
SBCM	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$
SBCM#	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$ $ZR \leftarrow ZR+1$
SBCM*	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$, $(@ZR)_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$
SBCM*#	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$, $(@ZR)_{10} \leftarrow (@ZR)_{10} + MUB_{10} + CF$ $ZR \leftarrow ZR+1$

ADD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal addition operation instruction. The content of the data RAM pointed to by Rx, @HL or @ZR is added to the content of AC in an addition operation.

The instruction distinguishes between binary or decimal addition using the existence of "DA" in the operand. Use of DA indicates decimal addition; its absence means use of binary addition.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this instruction make sure that the summand and addend are both in the same number system. The existence or absence of the "DA" in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The carried number in the result of this addition operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADD	Rx	$AC \leftarrow (Rx) + AC$
ADD*	Rx	$AC \leftarrow (Rx) + AC$, $(Rx) \leftarrow (Rx) + AC$
ADD	@HL	$AC \leftarrow (@HL) + AC$
ADD#	@HL	$AC \leftarrow (@HL) + AC$ $HL \leftarrow HL+1$
ADD*	@HL	$AC \leftarrow (@HL) + AC$, $(@HL) \leftarrow (@HL) + AC$
ADD*#	@HL	$AC \leftarrow (@HL) + AC$, $(@HL) \leftarrow (@HL) + AC$ $HL \leftarrow HL+1$
ADD	@HL, DA	$AC_{10} \leftarrow (@HL)_{10} + AC_{10}$
ADD#	@HL, DA	$AC_{10} \leftarrow (@HL)_{10} + AC_{10}$ $HL \leftarrow HL+1$
ADD*	@HL, DA	$AC_{10} \leftarrow (@HL)_{10} + AC_{10}$, $(@HL)_{10} \leftarrow (@HL)_{10} + AC_{10}$
ADD*#	@HL, DA	$AC_{10} \leftarrow (@HL)_{10} + AC_{10}$,

OP code	Operand	Instruction Operation
		$(@HL)_{10} \leftarrow (@HL)_{10} + AC_{10}$ HL ← HL+1
ADD	@ZR	AC ← (@ZR) + AC
ADD#	@ZR	AC ← (@ZR) + AC ZR ← ZR+1
ADD*	@ZR	AC ← (@ZR) + AC , (@ZR) ← (@ZR) + AC
ADD*#	@ZR	AC ← (@ZR) + AC , (@ZR) ← (@ZR) + AC ZR ← ZR+1
ADD	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + AC_{10}$
ADD#	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + AC_{10}$ ZR ← ZR+1
ADD*	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + AC_{10}$, $(@ZR)_{10} \leftarrow (@ZR)_{10} + AC_{10}$
ADD*#	@ZR, DA	$AC_{10} \leftarrow (@ZR)_{10} + AC_{10}$, $(@ZR)_{10} \leftarrow (@ZR)_{10} + AC_{10}$ ZR ← ZR+1

ADDM

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal addition operation instruction. The content of the data RAM pointed to by Rx, @HL or @ZR is added to the content of MU (upper register for multiplication operation result) in an addition operation.

The instruction distinguishes between binary or decimal addition using the existence of “DA” in the operand. Use of DA indicates decimal addition; its absence means use of binary addition.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this instruction make sure that the summand and addend are both in the same number system. The existence or absence of the “DA” in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The carried number in the result of this addition operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADDM	Rx	AC ← (Rx) + MU
ADDM*	Rx	AC ← (Rx) + MU , (Rx) ← (Rx) + MU
ADDM	@HL	AC ← (@HL) + MU
ADDM#	@HL	AC ← (@HL) + MU HL ← HL+1

OP code	Operand	Instruction Operation
ADDM*	@HL	AC ← (@HL) + MU , (@HL) ← (@HL) + MU
ADDM*#	@HL	AC ← (@HL) + MU , (@HL) ← (@HL) + MU HL ← HL+1
ADDM	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10}
ADDM#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} HL ← HL+1
ADDM*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} , $(@HL)_{10}$ ← $(@HL)_{10}$ + MU_{10}
ADDM*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MU_{10} , $(@HL)_{10}$ ← $(@HL)_{10}$ + MU_{10} HL ← HL+1
ADDM	@ZR	AC ← (@ZR) + MU
ADDM#	@ZR	AC ← (@ZR) + MU ZR ← ZR+1
ADDM*	@ZR	AC ← (@ZR) + MU , (@ZR) ← (@ZR) + MU
ADDM*#	@ZR	AC ← (@ZR) + MU , (@ZR) ← (@ZR) + MU ZR ← ZR+1
ADDM	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10}
ADDM#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} ZR ← ZR+1
ADDM*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} , $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MU_{10}
ADDM*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MU_{10} , $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MU_{10} ZR ← ZR+1

SUB

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal subtraction operation instruction. The content of AC (subtrahend) is subtracted from the data RAM pointed to by Rx, @HL or @ZR (minuend) with no borrowing. In this operation, the borrowed number value will be fixed as 1 (no borrowing).

The instruction distinguishes between binary or decimal subtraction using the existence of “DA” in the operand. Use of DA indicates decimal subtraction; its absence means use of binary subtraction.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this instruction make sure that the minuend and subtrahend are both in the same number system. The existence or absence of the “DA” in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The borrowed number in the result of this subtraction operation will change the value of CF.

(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SUB	Rx	AC ← (Rx) + ACB +1
SUB*	Rx	AC ← (Rx) + ACB +1, (Rx) ← (Rx) + ACB +1
SUB	@HL	AC ← (@HL) + ACB +1
SUB#	@HL	AC ← (@HL) + ACB +1 HL ← HL+1
SUB*	@HL	AC ← (@HL) + ACB +1, (@HL) ← (@HL) + ACB +1
SUB*#	@HL	AC ← (@HL) + ACB +1, (@HL) ← (@HL) + ACB +1 HL ← HL+1
SUB	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +1
SUB#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +1 HL ← HL+1
SUB*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +1, $(@HL)_{10}$ ← $(@HL)_{10}$ + ACB_{10} +1
SUB*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + ACB_{10} +1, $(@HL)_{10}$ ← $(@HL)_{10}$ + ACB_{10} +1 HL ← HL+1
SUB	@ZR	AC ← (@ZR) + ACB +1
SUB#	@ZR	AC ← (@ZR) + ACB +1 ZR ← ZR+1
SUB*	@ZR	AC ← (@ZR) + ACB +1, (@ZR) ← (@ZR) + ACB +1
SUB*#	@ZR	AC ← (@ZR) + ACB +1, (@ZR) ← (@ZR) + ACB +1 ZR ← ZR+1
SUB	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +1
SUB#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +1 ZR ← ZR+1
SUB*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +1, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + ACB_{10} +1
SUB*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + ACB_{10} +1, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + ACB_{10} +1 ZR ← ZR+1

SUBM

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary or decimal subtraction operation instruction. The content (subtrahend) of MU (upper register for multiplication operation result) is subtracted from the data RAM pointed to by Rx, @HL or @ZR (minuend) with no borrowing. In this operation, the borrowed number value will be fixed as 1 (no borrowing).

The instruction distinguishes between binary or decimal subtraction using the existence of "DA" in the operand. Use of DA indicates decimal subtraction; its absence means use of binary subtraction.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:

(1). Before executing this instruction make sure that the minuend and subtrahend are both in the same number system. The existence or absence of the "DA" in the operand will determine whether the result of the operation will be in the binary or decimal format.
(2). The borrowed number in the result of this subtraction operation will change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SUBM	Rx	AC ← (Rx) + MUB +1
SUBM*	Rx	AC ← (Rx) + MUB +1, (Rx) ← (Rx) + MUB +1
SUBM	@HL	AC ← (@HL) + MUB +1
SUBM#	@HL	AC ← (@HL) + MUB +1 HL ← HL+1
SUBM*	@HL	AC ← (@HL) + MUB +1, (@HL) ← (@HL) + MUB +1
SUBM*#	@HL	AC ← (@HL) + MUB +1, (@HL) ← (@HL) + MUB +1 HL ← HL+1
SUBM	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +1
SUBM#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +1 HL ← HL+1
SUBM*	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +1, $(@HL)_{10}$ ← $(@HL)_{10}$ + MUB_{10} +1
SUBM*#	@HL, DA	AC_{10} ← $(@HL)_{10}$ + MUB_{10} +1, $(@HL)_{10}$ ← $(@HL)_{10}$ + MUB_{10} +1 HL ← HL+1
SUBM	@ZR	AC ← (@ZR) + MUB +1
SUBM#	@ZR	AC ← (@ZR) + MUB +1 ZR ← ZR+1
SUBM*	@ZR	AC ← (@ZR) + MUB +1, (@ZR) ← (@ZR) + MUB +1
SUBM*#	@ZR	AC ← (@ZR) + MUB +1, (@ZR) ← (@ZR) + MUB +1 ZR ← ZR+1
SUBM	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MUB_{10} +1
SUBM#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MUB_{10} +1 ZR ← ZR+1
SUBM*	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MUB_{10} +1, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MUB_{10} +1
SUBM*#	@ZR, DA	AC_{10} ← $(@ZR)_{10}$ + MUB_{10} +1, $(@ZR)_{10}$ ← $(@ZR)_{10}$ + MUB_{10} +1 ZR ← ZR+1

ADN

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary addition operation instruction. The content of the data RAM pointed to by Rx, @HL or @ZR is added to the content of AC in an addition operation.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). Before executing this operation make sure that the summand and addend are both in the same number system.
(2). The carried number in the result of this addition operation will not change the value of CF.
(3). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(4). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADN	Rx	AC ← (Rx) + AC
ADN*	Rx	AC ← (Rx) + AC, (Rx) ← (Rx) + AC
ADN	@HL	AC ← (@HL) + AC
ADN#	@HL	AC ← (@HL) + AC, HL ← HL+1
ADN*	@HL	AC ← (@HL) + AC, (@HL) ← (@HL) + AC
ADN*#	@HL	AC ← (@HL) + AC, (@HL) ← (@HL) + AC HL ← HL+1
ADN	@ZR	AC ← (@ZR) + AC
ADN#	@ZR	AC ← (@ZR) + AC, ZR ← ZR+1
ADN*	@ZR	AC ← (@ZR) + AC, (@ZR) ← (@ZR) + AC
ADN*#	@ZR	AC ← (@ZR) + AC, (@ZR) ← (@ZR) + AC ZR ← ZR+1

AND

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is the AND logic operation instruction. The AND logic operation is performed on the content of the data RAM pointed to by Rx, @HL or @ZR and the content of AC register.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Truth Table:

A	B	A & B
0	0	0
0	1	0
1	1	1
1	0	0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
AND	Rx	AC ← (Rx) & AC
AND*	Rx	AC ← (Rx) & AC, (Rx) ← (Rx) & AC
AND	@HL	AC ← (@HL) & AC
AND#	@HL	AC ← (@HL) & AC, HL ← HL+1
AND*	@HL	AC ← (@HL) & AC, (@HL) ← (@HL)& AC
AND*#	@HL	AC ← (@HL) & AC, (@HL) ← (@HL) & AC, HL ← HL+1
AND	@ZR	AC ← (@ZR) & AC
AND#	@ZR	AC ← (@ZR) & AC, ZR ← ZR+1
AND*	@ZR	AC ← (@ZR) & AC, (@ZR) ← (@ZR) & AC
AND*#	@ZR	AC ← (@ZR) & AC, (@ZR) ← (@ZR) & AC, ZR ← ZR+1

EOR

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is the Exclusive-OR logic operation instruction. The Exclusive-OR logic operation is performed on the content of the data RAM pointed to by Rx, @HL or @ZR and the content of AC register.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Truth Table:

A	B	A xor B
0	0	0
0	1	1
1	1	0
1	0	1

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
EOR	Rx	AC ← (Rx) xor AC
EOR*	Rx	AC ← (Rx) xor AC, (Rx) ← (Rx) xor AC
EOR	@HL	AC ← (@HL) xor AC
EOR#	@HL	AC ← (@HL) xor AC, HL ← HL+1
EOR*	@HL	AC ← (@HL) xor AC, (@HL) ← (@HL) xor AC
EOR*#	@HL	AC ← (@HL) xor AC, (@HL) ← (@HL) xor AC, HL ← HL+1
EOR	@ZR	AC ← (@ZR) xor AC
EOR#	@ZR	AC ← (@ZR) xor AC, ZR ← ZR+1
EOR*	@ZR	AC ← (@ZR) xor AC, (@ZR) ← (@ZR) xor AC
EOR*#	@ZR	AC ← (@ZR) xor AC, (@ZR) ← (@ZR) xor AC, ZR ← ZR+1

OR

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is the OR logic operation instruction. The OR logic operation is performed on the content of the data RAM pointed to by Rx, @HL or @ZR and the content of AC register.

The operation result will be stored to AC or the data RAM pointed to by Rx, @HL or @ZR.

Truth Table:

A	B	A | B
0	0	0
0	1	1
1	1	1
1	0	1

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OR	Rx	AC ← (Rx) | AC

OP code	Operand	Instruction Operation
OR*	Rx	(Rx) ← (Rx) | AC, AC ← (Rx) | AC
OR	@HL	AC ← (@HL) | AC
OR#	@HL	AC ← (@HL) | AC, HL← HL+1
OR*	@HL	(@HL) ← (@HL) | AC, AC ← (@HL) | AC
OR*#	@HL	(@HL) ← (@HL) | AC, AC ← (@HL) | AC HL← HL+1
OR	@ZR	AC ← (@ZR) | AC
OR#	@ZR	AC ← (@ZR) | AC, ZR← ZR+1
OR*	@ZR	(@ZR) ← (@ZR) | AC, AC ← (@ZR) | AC
OR*#	@ZR	(@ZR) ← (@ZR) | AC, AC ← (@ZR) | AC ZR← ZR+1

ADCI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary addition operation instruction. The content of the data RAM pointed to by Ry is added to the content of Immediate data (D) and CF in an addition operation.

The operation result will be stored to AC or the data RAM pointed to by Ry.

Notes:
(1). The carried number in the result of this addition operation will change the value of CF.
(2). Before executing this operation make sure that the summand and addend are both in the same number system.
(3). D = 0 ~Fh
(4). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADCI	Ry, D	AC ← (Ry) + D + CF
ADCI*	Ry, D	(Ry) ← (Ry) + D + CF, AC ← (Ry) + D + CF

SBCI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary subtraction operation instruction. The content (subtrahend) of Immediate data (D) and CF (borrow) is subtracted from the data RAM pointed to by Ry (minuend).

The operation result will be stored to AC or the data RAM pointed to by Ry.

Notes:
(1). The borrowed number in the result of this subtraction operation will change the value of CF.
(2). Before executing this operation make sure that the minuend and subtrahend are both in the same number system.
(3). D = 0 ~Fh
(4). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SBCI	Ry, D	AC ← (Ry) + DB + CF
SBCI*	Ry, D	(Ry) ← (Ry) + DB + CF AC ← (Ry) + DB + CF

ADDI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary addition operation instruction. The content of the data RAM pointed to by Ry is added to the content of Immediate data (D) in an addition operation.

The operation result will be stored to AC or the data RAM pointed to by Ry.

Notes:
(1). The carried number in the result of this addition operation will change the value of CF.
(2). Before executing this operation make sure that the summand and addend are both in the same number system.
(3). D = 0 ~Fh
(4). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADDI	Ry, D	AC ← (Ry) + D
ADDI*	Ry, D	(Ry) ← (Ry) + D, AC ← (Ry) + D

SUBI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary subtraction operation instruction. The content (subtrahend) of Immediate data (D) where borrow = 1 is subtracted from the data RAM pointed to by Ry (minuend). In this operation, the borrowed number value will be fixed as 1 (no borrowing).

The operation result will be stored to AC or the data RAM pointed to by Ry.

Notes:
(1). The carried number in the result of this subtraction operation will change the value of CF.
(2). Before executing this operation make sure that the minuend and subtrahend are both in the same number system.
(3). D = 0 ~Fh
(4). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SUBI	Ry, D	AC ← (Ry) + DB + 1
SUBI*	Ry, D	(Ry) ← (Ry) + DB + 1, AC ← (Ry) + DB + 1

ADNI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is a binary addition operation instruction. The content of the data RAM pointed to by Ry is added to the content of Immediate data (D) in an addition operation.

The operation result will be stored to AC or the data RAM pointed to by Ry.

Notes:
(1). The carried number in the result of this addition operation will not change the value of CF.
(2). Before executing this operation make sure that the summand and addend are both in the same number system.
(3). D = 0 ~Fh
(4). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ADNI	Ry, D	AC ← (Ry) + D
ADNI*	Ry, D	(Ry) ← (Ry) + D, AC ← (Ry) + D

DAA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Convert the content of AC acquired after binary addition into decimal then restore the conversion result to AC or the data RAM pointed to by Rx, @HL or @ZR. When using this instruction must make sure that the content of AC is the result from a binary addition operation.

The converted data's correspondence is as follows:

AC data before DAA execution	CF data before DAA execution	AC data after DAA execution	CF data after DAA execution
0 ≤ AC ≤ 9	CF = 0	no change	no change
A ≤ AC ≤ F	CF = 0	AC= AC+ 6	CF = 1
0 ≤ AC ≤ 3	CF = 1	AC= AC+ 6	no change

Notes:
(1). The result from this operation will change the value of CF.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
DAA		AC_{10} ← DAA ← AC
DAA*	Rx	AC_{10} ← DAA ← AC, $(Rx)_{10}$ ← DAA ← AC
DAA*	@HL	AC_{10} ← DAA ← AC, $(@HL)_{10}$ ← DAA ← AC
DAA*#	@HL	AC_{10} ← DAA ← AC, $(@HL)_{10}$ ← DAA ← AC HL ← HL+1
DAA*	@ZR	AC_{10} ← DAA ← AC, $(@ZR)_{10}$ ← DAA ← AC
DAA*#	@ZR	AC_{10} ← DAA ← AC, $(@ZR)_{10}$ ← DAA ← AC ZR ← ZR+1

DAS

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Convert the content of AC acquired after binary subtraction into decimal then restore the conversion result to AC or the data RAM pointed to by Rx, @HL or @ZR. When using this instruction must make sure that the content of AC is the result from a binary subtraction operation.

The converted data's correspondence is as follows:

AC data before DAS execution	CF data before DAS execution	AC data after DAS execution	CF data after DAS execution
0 ≤ AC ≤ 9	CF = 1	No change	no change
6 ≤ AC ≤ F	CF = 0	AC= AC+A	no change
AC data before DAS execution	CF data before DAS execution	AC data after DAS execution	CF data after DAS execution

Notes:
(1). The result from this operation will change the value of CF.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
DAS		AC_{10} ← DAS ← AC
DAS*	Rx	AC_{10} ← DAS ← AC, Rx_{10} ← DAS ← AC
DAS*	@HL	AC_{10} ← DAS ← AC, $(@HL)_{10}$ ← DAS ← AC
DAS*#	@HL	AC_{10} ← DAS ← AC, $(@HL)_{10}$ ← DAS ← AC HL ← HL+1
DAS*	@ZR	AC_{10} ← DAS ← AC, $(@ZR)_{10}$ ← DAS ← AC
DAS*#	@ZR	AC_{10} ← DAS ← AC, $(@ZR)_{10}$ ← DAS ← AC ZR ← ZR+1

INC*

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Increment the content of the data RAM pointed to by Rx, @HL or @ZR by 1 then store the result to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). The carried number in the result of this addition operation will change the value of CF.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
INC*	Rx	(Rx) ← (Rx)+1, AC ← (Rx)+1
INC*	@HL	(@HL) ← (@HL)+1, AC ← (@HL)+1

OP code	Operand	Instruction Operation
INC*#	@HL	(@HL) ← (@HL)+1, AC ← (@HL)+1 HL ← HL+1
INC*	@ZR	(@ZR) ← (@ZR)+1, AC ← (@ZR)+1
INC*#	@ZR	(@ZR) ← (@ZR)+1, AC ← (@ZR)+1 ZR ← ZR+1

DEC*

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Decrement the content of the data RAM pointed to by Rx, @HL or @ZR by 1 then store the result to AC or the data RAM pointed to by Rx, @HL or @ZR.

Notes:
(1). The carried number in the result of this subtraction operation will change the value of CF.
(2). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
DEC*	Rx	(Rx) ← (Rx)-1, AC ← (Rx)-1
DEC*	@HL	(@HL) ← (@HL)-1, AC ← (@HL)-1
DEC*#	@HL	(@HL) ← (@HL)-1, AC ← (@HL)-1 HL ← HL+1
DEC*	@ZR	(@ZR) ← (@ZR)-1, AC ← (@ZR)-1
DEC*#	@ZR	(@ZR) ← (@ZR)-1, AC ← (@ZR)-1 ZR ← ZR+1

ANDI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is an AND logic operation instruction. The AND logic operation is performed on the content of the data RAM pointed to by Ry and the content of Immediate data (D).

The operation result will be stored to AC or the data RAM pointed to by Ry.

Truth Table:

A	B	A & B
0	0	0
0	1	0
1	1	1
1	0	0

Notes:
(1). D = 0 ~Fh
(2). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ANDI	Ry, D	AC ← (Ry) & D
ANDI*	Ry, D	AC ← (Ry) & D, (Ry) ← (Ry) & D

EORI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is an Exclusive-OR logic operation instruction. The Exclusive-OR logic operation is performed on the content of the data RAM pointed to by Ry and the content of Immediate data (D).

The operation result will be stored to AC or the data RAM pointed to by Ry.

Truth Table:

A	B	A xor B
0	0	0
0	1	1
1	1	0
1	0	1

Notes:
(1). D = 0 ~Fh
(2). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
EORI	Ry, D	AC ← (Ry) xor D
EORI*	Ry, D	AC ← (Ry) xor D, (Ry) ← (Ry) xor D

ORI

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
This is an OR logic operation instruction. The OR logic operation is performed on the content of the data RAM pointed to by Ry and the content of Immediate data (D).

The operation result will be stored to AC or the data RAM pointed to by Ry.

Truth Table:

A	B	A | B
0	0	0
0	1	1
1	1	1
1	0	1

Notes:
(1). D = 0 ~Fh
(2). Absolute addressing must be used for Ry syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
ORI	Ry, D	AC ← (Ry) | D
ORI*	Ry, D	AC ← (Ry) | D, (Ry) ← Ry | D

SR0, SR1

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Right shift the content of the data RAM pointed to by Rx in the direction of the LSB by one bit and write 0 (SR0) or 1 (SR1) at the MSB. Store the result of the shift back to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Original RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Shifted RAM data	0(1)	(Rx)3	(Rx)2	(Rx)1
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SR0	Rx	$(Rx)_n$, AC_n ← $(Rx)_{n+1}$, AC_{n+1}, $(Rx)_3$, AC_3 ← 0
SR1	Rx	$(Rx)_n$, AC_n ← $(Rx)_{n+1}$, AC_{n+1}, $(Rx)_3$, AC_3 ← 1

SL0, SL1

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Left shift the content of the data RAM pointed to by Rx in the direction of the MSB by one bit and write 0 (SL0) or 1 (SL1) at the LSB. Store the result of the shift back to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Original RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Shifted RAM data	(Rx)2	(Rx)1	(Rx)0	0(1)
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
SL0	Rx	$(Rx)_{n+1}$, $AC_{n+1} \leftarrow (Rx)_n$, AC_n, $(Rx)_0$, $AC_0 \leftarrow 0$
SL1	Rx	$(Rx)_{n+1}$, $AC_{n+1} \leftarrow (Rx)_n$, AC_n, $(Rx)_0$, $AC_0 \leftarrow 1$

RRC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Rotate clockwise the content of the data RAM pointed to by Rx, @HL or @ZR and CF in the direction of the LSB by one bit. After the rotation Store the result back to the data RAM pointed to by Rx, @HL or @ZR and AC.

The bit data correspondence is as follows:

Original RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0	-
Original CF	-	-	-	-	C
Rotated RAM data	C	(Rx)3	(Rx)2	(Rx)1	-
New CF					(Rx)0
Destination AC register	AC3	AC2	AC1	AC0	

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). The result from this rotation operation will change the value of CF.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RRC	Rx	Temp ← $(Rx)_0$, $(Rx)_n$, AC_n ← $(Rx)_{n+1}$, AC_{n+1}, $(Rx)_3$, AC_3 ← CF, CF ← Temp
RRC	@HL	Temp ← $(@HL)_0$, $(@HL)_n$, AC_n ← $(@HL)_{n+1}$, AC_{n+1}, $(@HL)_3$, AC_3 ← CF, CF ← Temp
RRC#	@HL	Temp ← $(@HL)_0$, $(@HL)_n$, AC_n ← $(@HL)_{n+1}$, AC_{n+1}, $(@HL)_3$, AC_3 ← CF, CF ← Temp HL ← HL+1
RRC	@ZR	Temp ← $(@ZR)_0$, $(@ZR)_n$, AC_n ← $(@ZR)_{n+1}$, AC_{n+1}, $(@ZR)_3$, AC_3 ← CF, CF ← Temp
RRC#	@ZR	Temp ← $(@ZR)_0$, $(@ZR)_n$, AC_n ← $(@ZR)_{n+1}$, AC_{n+1}, $(@ZR)_3$, AC_3 ← CF, CF ← Temp ZR ← ZR+1

RLC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Rotate anti-clockwise the content of the data RAM pointed to by Rx, @HL or @ZR and CF in the direction of the MSB by one bit. After the rotation Store the result back to the data RAM pointed to by Rx, @HL or @ZR and AC.

The bit data correspondence is as follows:

Original RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0	-
Original CF	-	-	-	-	C
Rotated RAM data	(Rx)2	(Rx)1	(Rx)0	C	-
New CF					(Rx)3
Destination AC register	AC3	AC2	AC1	AC0	

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL or ZR by 1.
(2). The result from this rotation operation will change the value of CF.
(3). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RLC	Rx	Temp ← $(Rx)_3$, $(Rx)_{n+1}$, AC_{n+1} ← $(Rx)_n$, AC_n, $(Rx)_0$, AC_0 ← CF, CF ← Temp
RLC	@HL	Temp ← $(@HL)_3$, $(@HL)_{n+1}$, AC_{n+1} ← $(@HL)_n$, AC_n, $(@HL)_0$, AC_0 ← CF, CF ← Temp
RLC#	@HL	Temp ← $(@HL)_3$, $(@HL)_{n+1}$, AC_{n+1} ← $(@HL)_n$, AC_n, $(@HL)_0$, AC_0 ← CF, CF ← Temp, HL ← HL+1
RLC	@ZR	Temp ← $(@ZR)_3$, $(@ZR)_{n+1}$, AC_{n+1} ← $(@ZR)_n$, AC_n, $(@ZR)_0$, AC_0 ← CF, CF ← Temp
RLC#	@ZR	Temp ← $(@ZR)_3$, $(@ZR)_{n+1}$, AC_{n+1} ← $(@ZR)_n$, AC_n, $(@ZR)_0$, AC_0 ← CF, CF ← Temp, ZR ← ZR+1

2-6. I/O Port Instructions

SPA

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the configuration of the IOA port.

X4 = 1, enable the pull-low components on the IOA1~4 pins.
= 0, disable the pull-low components on the IOA1~4 pins.

X3 = 1, set IOA4 to output mode
= 0, set IOA4 to input mode

X2 = 1, set IOA3 to output mode
= 0, set IOA3 to input mode

X1 = 1, set IOA2 to output mode
= 0, set IOA2 to input mode

X0 = 1, set IOA1 to output mode
= 0, set IOA1 to input mode

Note:
X4 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPA	X	Enable/disable Pull-low component ← X4 Set input/output mode for IOA4 ← X3 Set input/output mode for IOA3 ← X2 Set input/output mode for IOA2 ← X1 Set input/output mode for IOA1 ← X0

IPA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the state of IOA port to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Port IOA	IOA4	IOA3	IOA2	IOA1
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
IPA	Rx	(Rx) ← IOA port, AC ← IOA port

OPA

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx to the IOA port's output register. The content of output register will be output to the IOA port when the IOA port is set to output mode. The bit data correspondence is as follows:

Port IOA	IOA4	IOA3	IOA2	IOA1
Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OPA	Rx	Output register of IOA port ← (Rx) IOA port ← Output register of IOA port,

OPAS

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 2 bits of data transferred.

Instruction Explanation:
Output the bit01 and bit1 of the data RAM pointed to by Rx to the IOA port's IOA1 and IOA2 pins. Output the 1-bit data (0/1) set by D to the IOA port's IOA3 pin.

A High Pulse with a width of BCKL/2 will be output on the IOA4 pin before the end of the instruction cycle. The bit data correspondence is as follows:

Port IOA	IOA4	IOA3	IOA2	IOA1
Bit data	Pulse	D	(Rx)1	(Rx)0

Notes:
(1). Absolute addressing must be used for Rx syntax.
(2). D = 0 or 1

Instruction Syntax:

OP code	Operand	Instruction Operation
OPAS	Rx, D	IOA1, A2 ← (Rx)0, (Rx)1 IOA3 ← D IOA4 ← pulse

SPB

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the configuration of the IOB port.

X4 = 1, enable the pull-low components on the IOB1~4 pins.
= 0, disable the pull-low components on the IOA1~4 pins.

X3 = 1, set IOB4 to output mode
= 0, set IOB4 to input mode

X2 = 1, set IOB3 to output mode
= 0, set IOB3 to input mode

X1 = 1, set IOB2 to output mode
= 0, set IOB2 to input mode

X0 = 1, set IOB1 to output mode
= 0, set IOB1 to input mode

Note:
X4 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPB	X	Enable/disable Pull-low component ← X4 Set input/output mode for IOB4 ← X3 Set input/output mode for IOB3 ← X2 Set input/output mode for IOB2 ← X1 Set input/output mode for IOB1 ← X0

IPB

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the state of IOB port to the data RAM pointed to by Rx and AC. The bit data correspondence is as follows:

Port IOB	IOB4	IOB3	IOB2	IOB1
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
IPB	Rx	(Rx) ← IOB port, AC ← IOB port

OPB

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx to the IOB port's output register. The content of output register will be output to the IOB port when the IOB port is set to output mode. The bit data correspondence is as follows:

Port IOB	IOB4	IOB3	IOB2	IOB1
Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OPB	Rx	Output register of IOB port ← (Rx) IOB port ← Output register of IOB port,

SPC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the configuration of the IOC port.

X4 = 1, enable the pull-low components on the IOC1~4 pins.
=0, disable the pull-low components on the IOC1~4 pins.

X3 = 1, set IOC4 to output mode
= 0, set IOC4 to input mode

X2 = 1, set IOC3 to output mode
= 0, set IOC3 to input mode

X1 = 1, set IOC2 to output mode
= 0, set IOC2 to input mode

X0 = 1, set IOC1 to output mode
= 0, set IOC1 to input mode

Note:
X4 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPC	X	Enable/disable Pull-low component ← X4 Set input/output mode for IOC4 ← X3 Set input/output mode for IOC3 ← X2 Set input/output mode for IOC2 ← X1 Set input/output mode for IOC1 ← X0

IPC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the state of IOC port to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Port IOC	IOC4	IOC3	IOC2	IOC1
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
IPC	Rx	(Rx) ← IOC port, AC ← IOC port

OPC

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx to the IOC port's output register. The content of output register will be output to the IOC port when the IOC port is set to output mode.

The bit data correspondence is as follows:

Port IOC	IOC4	IOC3	IOC2	IOC1
Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OPC	Rx	Output register of IOC port ← (Rx) IOC port ← Output register of IOC port,

SPD

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the configuration of the IOD port.

X4 = 1, enable the pull-low components on the IOD1~4 pins.
= 0, disable the pull-low components on the IOD1~4 pins.

X3 = 1, set IOD4 to output mode
= 0, set IOD4 to input mode

X2 = 1, set IOD3 to output mode
= 0, set IOD3 to input mode

X1 = 1, set IOD2 to output mode
= 0, set IOD2 to input mode

X0 = 1, set IOD1 to output mode
= 0, set IOD1 to input mode

Note:
X4 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPD	X	Enable/disable Pull-low component ← X4 Set input/output mode for IOD4 ← X3 Set input/output mode for IOD3 ← X2 Set input/output mode for IOD2 ← X1 Set input/output mode for IOD1 ← X0

IPD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the state of IOD port to the data RAM pointed to by Rx and AC. The bit data correspondence is as follows:

Port IOD	IOD4	IOD3	IOD2	IOD1
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
IPD	Rx	(Rx) ← IOD port, AC ← IOD port

OPD

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx to the IOD port's output register. The content of output register will be output to the IOD port when the IOD port is set to output mode. The bit data correspondence is as follows:

Port IOD	IOD4	IOD3	IOD2	IOD1
Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OPD	Rx	Output register of IOD port ← (Rx) IOD port ← Output register of IOD port,

SPE

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the configuration of the IOE port.

X4 = 1, enable the pull-low components on the IOE1~4 pins.
= 0, disable the pull-low components on the IOE1~4 pins.

X3 = 1, set IOE4 to output mode
= 0, set IOE4 to input mode

X2 = 1, set IOE3 to output mode
= 0, set IOE3 to input mode

X1 = 1, set IOE2 to output mode
= 0, set IOE2 to input mode

X0 = 1, set IOE1 to output mode
= 0, set IOE1 to input mode

Note:
X4 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPE	X	Enable/disable Pull-low component ← X4 Set input/output mode for IOE4 ← X3 Set input/output mode for IOE3 ← X2 Set input/output mode for IOE2 ← X1 Set input/output mode for IOE1 ← X0

IPE

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the state of IOE port to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

Port IOE	IOE4	IOE3	IOE2	IOE1
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
IPE	Rx	(Rx) ← IOE port, AC ← IOE port

OPE

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Write the content of the data RAM pointed to by Rx to the IOE port's output register. The content of output register will be output to the IOE port when the IOE port is set to output mode.

The bit data correspondence is as follows:

Port IOE	IOE4	IOE3	IOE2	IOE1
Source RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
OPE	Rx	Output register of IOE port ← (Rx) IOE port ← Output register of IOE port,

2-7. Table ROM Instructions

LDH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the upper 4 bits of the Table ROM content pointed to by HL to AC and the data RAM pointed to by Rx.

The written data's correspondence is as follows:

Source table data	T(@HL)7	T(@HL)6	T(@HL)5	T(@HL)4
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
LDH	Rx, @HL	(Rx) ← T(@HL) high nibble, AC ← T(@HL) high nibble
LDH*	Rx, @HL	(Rx) ← T(@HL) high nibble, AC ← T(@HL) high nibble, HL ← HL+1

LDL

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store the lower 4 bits of the Table ROM content pointed to by HL to AC and the data RAM pointed to by Rx.

The written data's correspondence is as follows:

Source table data	T(@HL)3	T(@HL)2	T(@HL)1	T(@HL)0
Destination RAM data	(Rx)3	(Rx)2	(Rx)1	(Rx)0
Destination AC register	AC3	AC2	AC1	AC0

Notes:
(1). After executing this instruction, certain syntax automatically increments the content of HL by 1.
(2). Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
LDL	Rx, @HL	(Rx) ← T(@HL) low nibble, AC ← T(@HL) low nibble
LDL*	Rx, @HL	(Rx) ← T(@HL) low nibble, AC ← T(@HL) low nibble, HL ← HL+1

LCDH

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 16 bits of data transferred.

Instruction Explanation:
Store the 16-bit content from two contiguous addresses in the Table ROM pointed to by @HL to the four contiguous data RAM addresses pointed to by @ZR.

Regardless of what the actual LSB value of @HL may be, it will be ignored during the execution of this instruction. The two contiguous addresses accessed will be @HL (RAM address bit0=0) and @HL (RAM address bit0=1).

Regardless of what the actual value of the two least significant bits of @ZR may be, they will be ignored during execution of this instruction. The four contiguous addresses accessed will be @ZR (RAM address bit1,0=00), @ZR (RAM address bit1,0=01), @ZR (RAM address bit 1,0=10) and @ZR (RAM address bit1,0=11).

The bit data correspondence is as follows:

Destination RAM addr. & data mapping	RAM addr. = N*+1				RAM addr. = N*			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Table ROM output 16bits data	T(@HL)7	T(@HL)6	T(@HL)5	T(@HL)4	T(@HL)3	T(@HL)2	T(@HL)1	T(@HL)0
@ZR addr.	@ZR(RAM Address bit1,0=01)				@ZR(RAM Address bit1,0=00)			
Destination RAM addr. & data mapping	RAM addr. = N*+3				RAM addr. = N*+2			
	bit3	bit2	bit1	bit0	bit3	bit2	bit1	bit0
Table ROM output 16bits data	T(@HL) 15	T(@HL) 14	T(@HL) 13	T(@HL) 12	T(@HL) 11	T(@HL) 10	T(@HL) 9	T(@HL) 8
@ZR addr.	@ZR(RAM Address bit1,0=11)				@ZR(RAM Address bit1,0=10)			

***: N represents data RAM addresses that are multiples of four.**

Notes:
(1). The contents of HL and ZR must be set before executing this instruction.
(2). After executing this instruction, certain syntax automatically increments the content of ZR by 4 or the content of HL by 2.

Instruction Syntax:

OP code	Operand	Instruction Operation
LCDH	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← T(@HL)3~0, (@ZR(RAM Address bit1,0=01)) ← T(@HL)7~4, (@ZR(RAM Address bit1,0=10)) ← T(@HL)11~8, (@ZR(RAM Address bit1,0=11)) ← T(@HL)15~12,
LCDH&	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← T(@HL)3~0, (@ZR(RAM Address bit1,0=01)) ← T(@HL)7~4, (@ZR(RAM Address bit1,0=10)) ← T(@HL)11~8, (@ZR(RAM Address bit1,0=11)) ← T(@HL)15~12, HL ← HL+2
LCDH%	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← T(@HL)3~0, (@ZR(RAM Address bit1,0=01)) ← T(@HL)7~4, (@ZR(RAM Address bit1,0=10)) ← T(@HL)11~8, (@ZR(RAM Address bit1,0=11)) ← T(@HL)15~12, ZR ← ZR+4
LCDH$	@ZR, @HL	(@ZR(RAM Address bit1,0=00)) ← T(@HL)3~0, (@ZR(RAM Address bit1,0=01)) ← T(@HL)7~4, (@ZR(RAM Address bit1,0=10)) ← T(@HL)11~8, (@ZR(RAM Address bit1,0=11)) ← T(@HL)15~12, HL ← HL+2 ZR ← ZR+4

2-8. RFC Function Instructions

SRF

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the output function of RFC0~RFC5 pins.

X5 = 1, enable the output function of the RFC5 pin.
= 0, disable the output function of the RFC5 pin, outputs high impedance.

X4 = 1, enable the output function of the RFC4 pin.
= 0, disable the output function of the RFC4 pin, outputs high impedance.

X3 = 1, enable the output function of the RFC3 pin.
= 0, disable the output function of the RFC3 pin, outputs high impedance.

X2 = 1, enable the output function of the RFC2 pin.
= 0, disable the output function of the RFC2 pin, outputs high impedance.

X1 = 1, enable the output function of the RFC1 pin.
= 0, disable the output function of the RFC1 pin, outputs high impedance.

X0 = 1, enable the output function of the RFC1 pin.
= 0, disable the output function of the RFC1 pin, outputs high impedance.

Note:
X5 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SRF	X	Set output function of RFC5 ← X5 Set output function of RFC4 ← X4 Set output function of RFC3 ← X3 Set output function of RFC2 ← X2 Set output function of RFC1 ← X1 Set output function of RFC0 ← X0

SCNT

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of instruction operand X to set the operational mode, RFC counter type and clock source of the two RFC counters CX2/CX.

X5 = 1, setting RFC counter with CX2 pin.
Use X3, X2 to set the operational mode, X1, X0 to set counter type and X4 to set clock source.
= 0, setting RFC counter with CX pin.
Use X3, X2 to set the operational mode, X1, X0 to set counter type and X4 to set clock source.

X4 = 1, in an operational mode where CX2/CX pin isn't being used to control the RFC counter, set the output clock (FREQ) from the frequency generator as the clock source for the RFC counter selected by X5. This setting will only work when X3=0
= 0, in an operational mode where CX2/CX pin isn't being used to control the RFC counter, set the input signal applies on CX2/CX pin as the clock source for the RFC counter selected by X5. This setting will only work when X3=0.

X3, X2 = 00, the counting function of the CX2/CX RFC counter will be enabled after the execution of the instruction SF2 and stop after executing the instruction RF2 immediately.
= 01, after the execution of the instruction SF2, the counting function of the CX2/CX RFC counter is enabled by TMR2 starts counting and disabled by TMR2 stops.
TMR2 enabled duration is as follows:
Time duration = (TMR2's clock source cycle) x (TMR2 initial setting value).
In this mode, if one of the RFC function has enabled, the other RFC function can't be enabled by TMR2 either.
= 10, after the execution of the instruction SF2, the counting function of the CX2/CX RFC counter is enabled due to the first falling edge signal to appear on the CX2/CX pin. Stop when the next falling edge signal appears.
In this mode, the clock source of the RFC Counter always comes from the output of frequency generator (FREQ).
= 11, after the execution of the instruction SF2, the counting function of the CX2/CX RFC counter is enabled due to the first rising edge signal to appear on the CX2/CX pin. Stop when the next falling edge signal appears.
In this mode, the clock source of the RFC Counter always comes from the output of frequency generator (FREQ).

X1, X0 = 00, set 16-bit RFC counter to be the CX2/CX RFC counter.
This is the default setting of the MCU.
= 01, set TMR1 as the CX2/CX RFC counter
= 10, set TMR2 as the CX2/CX RFC counter
= 11, set TMR3 as the CX2/CX RFC counter

Notes:
(1). When TMR1~3 is set as the RFC Counter and enabled, do not execute any other Timer setting related instructions to avoid interfering with the RFC counter's counting function.
(2). X5 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SCNT	X	Choose to set CX2 or CX ← X5 Set the clock source of CX2/CX RFC when X3=0 ← X4 Set the start/stop control method for the CX2/CX RFC Counter ← X3, X2 Set the CX2/CX RFC Counter ← X1, X0

MRF1

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store bit3~bit0 of 16-bit RFC counter to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

RFC counter	RFC3	RFC2	RFC1	RFC0
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRF1	Rx	(Rx) ← RFC3 ~ RFC0, AC ← RFC3 ~ RFC0

MRF2

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store bit7~bit4 of 16-bit RFC counter to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

RFC counter	RFC7	RFC6	RFC5	RFC4
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRF2	Rx	(Rx) ← RFC7 ~ RFC4, AC ← RFC7 ~ RFC4

MRF3

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store bit11~bit8 of 16-bit RFC counter to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

RFC counter	RFC11	RFC10	RFC9	RFC8
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRF3	Rx	(Rx) ← RFC11 ~ RFC8, AC ← RFC11 ~ RFC8

MRF4

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Store bit15~bit12 of 16-bit RFC counter to the data RAM pointed to by Rx and AC.

The bit data correspondence is as follows:

RFC counter	RFC15	RFC14	RFC13	RFC12
Rx	(Rx)3	(Rx)2	(Rx)1	(Rx)0
AC	AC3	AC2	AC1	AC0

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
MRF4	Rx	(Rx) ← RFC15 ~ RFC12, AC ← RFC15 ~ RFC12

2-9. Jump/Call Instructions

JB0

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When bit0 of AC is equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If bit0 of AC is not equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JB0	X	If AC0 = 1, PC ← X; If AC0 != 1, PC ← PC+1

JB1

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When bit1 of AC is equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If bit1 of AC is not equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JB1	X	If AC1 = 1, PC ← X; If AC1 != 1, PC ← PC+1

JB2

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When bit2 of AC is equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If bit2 of AC is not equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JB2	X	If AC2 = 1, PC ← X; If AC2 != 1, PC ← PC+1

JB3

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When bit3 of AC is equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If bit3 of AC is not equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JB3	X	If AC3 = 1, PC ← X; If AC3 != 1, PC ← PC+1

JNZ

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the content of AC is not equal to 0, the program will jump to the PC = X address for the next instruction to execute.

If AC is equal to 0, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JNZ	X	If AC != 0, PC ← X; If AC = 0, PC ← PC+1

JZ

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the content of AC is equal to 0, the program will jump to the PC = X address for the next instruction to execute.

If AC is not equal to 0, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JZ	X	If AC = 0, PC ← X; If AC ! = 0, PC ← PC+1

JNC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the content of CF is not equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If CF is equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JNC	X	If CF != 1, PC ← X; If CF = 1, PC ← PC+1

JC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the content of CF is equal to 1, the program will jump to the PC = X address for the next instruction to execute.

If CF is not equal to 1, the program will execute the instruction from the next address.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the jump instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this jump instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JC	X	If CF = 1, PC ← X; If CF != 1, PC ← PC+1

CALL

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
This instruction is used to call a subroutine. Upon executing this instruction, the address of the next instruction will first be stored to the Stack. The program will jump to the PC = X address for the next instruction to execute in next instruction cycle.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the CALL instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this CALL instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
CALL	X	STACK ← PC+1, STACK pointer ← STACK pointer + 1, PC ← X

JMP

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
After executing this instruction, the program will jump to the PC = X address for the next instruction to execute in next instruction cycle.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). If the destination address of the CALL instruction belongs to a different bank from the address of the current instruction, the ICE compiler program will automatically insert a SPBK instruction in front of this CALL instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JMP	X	PC ← X

CPHL

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the lower byte data (IDBF7~IDBF0) of HL are not equal to the value defined for X, the program will execute the instruction from the next address as normal;

When the lower byte data (IDBF7~IDBF0) of HL are equal to the value defined for X, the program will force to execute the NOP instruction in the next instruction cycle.

Notes:
(1). X = 0h ~ FFh
(2). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
CPHL	X	If IDBF7~0 != X, execute next instruction as normal; If IDBF7~0 = X, execute NOP as the next instruction instead

CPZR

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
When the lower byte (ZRBF7~ZRBF0) of ZR are not equal to the value defined for X, the program will execute the instruction from the next address as normal.

When the lower byte (ZRBF7~ZRBF0) of ZR are equal to the value defined for X, the program will force to execute the NOP instruction in the next instruction cycle.

Notes:
(1). X = 0h ~ FFh
(2). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
CPZR	X	If ZRBF7~0 != X, execute next instruction as normal; If ZRBF7~0 = X, execute NOP as the next instruction instead

CPHLH

Instruction Characteristics:
Two-word instruction, 8 machine cycles.

Instruction Explanation:
When the 16-bit value of HL is not equal to the value defined for X, the program will execute the instruction from the next address as normal; if the 16-bit value of HL is equal to the value defined for X, the program will force to execute the NOP instruction in the next instruction cycle.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
CPHLH SETDAT	X	If IDBF != X, execute next instruction as normal; If IDBF = X, execute NOP as the next instruction instead

CPZRH

Instruction Characteristics:
Two-word instruction, 8 machine cycles.

Instruction Explanation:
When the 16-bit value of ZR is not equal to the value defined for X, the program will execute the instruction from the next address as normal; if the 16-bit value of ZR is equal to the value defined for X, the program will force to execute the NOP instruction in the next instruction cycle.

Notes:
(1). X = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(2). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
CPZRH SETDAT	X	If ZRBF != X, execute next instruction as normal; If ZRBF = X, execute NOP as the next instruction instead

CAC

Instruction Characteristics:
Multi-word instruction, 4 or 8 machine cycles.

Instruction Explanation:
This is a subroutine calling instruction with multiple choices. Up to 16 different subroutine addresses can be specified in the instruction and the value of AC used to determine the subroutine address to call.

The instruction operand X represents the total number of subroutines(X+1) that can be called in the instruction.

When AC<=X, the MCU will store the address of the next instruction (PC+X+2) to the STACK then load the subroutine address corresponding to the AC value to the program counter. It takes 8 machine cycles to complete whole instruction cycle.

When AC>X, the MCU will not call any subroutine and will only load the address of the next instruction (PC+X+2) into the program counter. It only takes 4 machine cycles to complete whole instruction cycle.

Notes:
(1). X = 0 ~ Fh.
(2). Addr(X) = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(3). MCU will temporarily suspend all interrupt requests during the 8 machine cycles used for executing this instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
CAC SETDAT SETDAT : SETDAT	X Addr(0) Addr(1) : Addr(X)	If AC <= X, PC=Addr(AC), STACK ← PC +X + 2, STACK pointer ← STACK pointer +1; If AC > X, PC ← PC+X+2;

JAC

Instruction Characteristics:
Multi-word instruction, 4 or 8 machine cycles.

Instruction Explanation:
This is a jump instruction with multiple choices. Up to 16 different destination addresses can be specified in the instruction and the value of AC used to determine the destination address to jump to.

The instruction operand X represents the total number of destination addresses (X+1) available to jump to in the instruction.

When AC<=X, the MCU will load the destination address corresponding to the current AC value into the program counter. It takes 8 machine cycles to complete whole instruction cycle.

When AC>X, the MCU will load the address of the next instruction into the program counter. It only takes 4 machine cycles to complete whole instruction cycle.

Notes:
(1). X = 0 ~ Fh.
(2). Addr(X) = 0h ~ FFFFh. (Range of X depends on the specifications for each MCU)
(3). MCU will temporarily suspend all interrupt requests during the 8 machine cycles used for executing this instruction.

Instruction Syntax:

OP code	Operand	Instruction Operation
JAC SETDAT SETDAT : SETDAT	X Addr(0) Addr(1) : Addr(X)	If AC <= X, PC ← Addr(AC); If AC > X, PC ← PC+X+2;

RTS

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Exit the subroutine.

Instruction Syntax:

OP code	Operand	Instruction Operation
RTS		STACK pointer ← STACK pointer – 1 PC ← STACK

2-10. RAM Page/ROM Bank Setting Instructions

SRY

Instruction Characteristics:
Single-word instruction, 4 machine cycles, dedicated for compiler.

Instruction Explanation:
The instruction operand X is used to set the Page number for Ry memory. In the program, the absolute address ($0h ~ $1FFFh) must be used to describe the Ry address. When the compiler discovers that the Ry address is not within the range of the initial page, it will automatically insert the SRY instruction before that instruction to ensure that the program can correctly access data from the Ry page.

As page 7 is Ry's initial page, if Ry is specified as 0~FH in operand, this means instruction will access the RAM data from Ry page 0. The compiler will therefore insert the instruction "SRY 0" in front of this instruction.

Notes:
(1). X = 0h ~ 6h, 8h ~ 1FFh. (Range of X depends on the specifications for each MCU)
(2). Ry memory's initial page is on page 7 (Ry=70~7Fh).
(3). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
SRY	X	Ry page number ← X

ERY

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
The instruction operand X is used to the set the page number of Ry memory. The Page number will keep constant until another ERY instruction is executed to change the page number or the CLPGX instruction (set X1 = 1) is executed to release this setting.

After executing any ERY, ERX or ELZ instruction, the MCU inhibits all interrupt service. After all Ry, Rx and Lz page settings have been released, MCU recovers the interrupt service again.

Notes:
(1). X = 0h ~ 6h, 8h ~ 1FFh. (Range of X depends on the specifications for each MCU)
(2). Don't access the RAM data of initial page when the program is within the memory page constrained segment.

Instruction Syntax:

OP code	Operand	Instruction Operation
ERY	X	Ry page number ← X Fix Ry page number Inhibit all interrupt services

SRX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, dedicated for compiler.

Instruction Explanation:
The instruction operand X is used to set the Page number for Rx memory. In the program, the absolute address ($0h ~ $1FFFh) must be used to describe the Rx address. When the compiler discovers that the Rx address is not within the range of the initial page, it will automatically insert the SRX instruction before that instruction to ensure that the program can correctly access data from the Rx page.

Notes:
(1). X = 1h ~ 3Fh. (Range of X depends on the specifications for each MCU)
(2). Rx memory's initial page is on page 0 (Rx = 00~7FH).
(3). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
SRX	X	Rx page number ← X

ERX

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
The instruction operand X is used to the set the page number of Rx memory. The Page number will keep constant until another ERX instruction is executed to change the page number or the CLPGX instruction (set X0 = 1) is executed to release this setting.

After executing any ERY, ERX or ELZ instruction, the MCU inhibits all interrupt services. After all Ry, Rx and Lz page settings have been released, MCU recovers the interrupt service again.

Notes:
(1). X = 1h ~ 3Fh. (Range of X depends on the specifications for each MCU)
(2). Don't access the RAM data of initial page when the program is within the memory page constrained segment.

Instruction Syntax:

OP code	Operand	Instruction Operation
ERX	X	Rx page number ← X Fix Rx page number Inhibit all interrupt services

SLZ

Instruction Characteristics:
Single-word instruction, 4 machine cycles, dedicated for compiler.

Instruction Explanation:
The instruction operand X is used to set the Page number for Lz memory. In the program, the absolute address ($00 ~ $7Fh) must be used to describe the Lz address. When the compiler discovers that the Lz address is not within the range of the initial page, it will automatically insert the SLZ instruction before that instruction to ensure that the program can correctly access data from the Lz page.

Notes:
(1). X = 1h ~ 3h. (Range of X depends on the specifications for each MCU)
(2). Lz memory's initial page is on page 0.
(3). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
SLZ	X	Lz page number ← X

ELZ

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
The instruction operand X is used to the set the page number of Lz memory. The Page number will keep constant until another ELZ instruction is executed to change the page number or the CLPGX instruction (set X2 = 1) is executed to release this setting.

After executing any ERY, ERX or ELZ instruction, the MCU inhibits all interrupt services. After all Ry, Rx and Lz page settings have been released, MCU recovers the interrupt service again.

Notes:
(1). X = 1h ~ 3h. (Range of X depends on the specifications for each MCU)
(2). Don't access the RAM data of initial page when the program is within the memory page constrained segment.

Instruction Syntax:

OP code	Operand	Instruction Operation
ELZ	X	Lz page number ← X Fix Lz page number Inhibit all interrupt services

CLPG

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
This instruction releases the page number setting by instructions like ERX, ERY and ELZ and recovers all interrupt services.

The bit data correspondence is as follows:

Operand X	X0 = 1	X1 = 1	X2 = 1
Action	Release ELZ setting	Release ERY setting	Release ERX setting

Note:
X2 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
CPLG	X	Release ELZ setting ← X0=1 Release ERY setting ← X1=1 Release ERX setting ← X2=1

SPBK

Instruction Characteristics:
Single-word instruction, 4 machine cycles, dedicated for compiler.

Instruction Explanation:
The instruction operand X is used to set the bank number of Program ROM. If the Compiler finds during compilation that jump instructions such as JMP and CALL have a destination address bank number different from the current bank number, it will automatically insert the SPBK instruction in front of that instruction to ensure that the program can correctly jump to the destination address.

Notes:
(1). X = 00h ~ 1Fh (Range of X depends on the specifications for each MCU)
(2). The MCU will temporarily suspend all interrupt requests during current and next instruction cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPBK	X	Program ROM bank number ← X

2-11. Timer Instructions

TMS, TMSX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4/8 bits of data transferred.

Instruction Explanation:
The instruction operand's Immediate data (X), the Table ROM content pointed to by HL, AC and the content of the data RAM pointed to by Rx can be used to set the 6-bit TMR1's clock source and the timer's pre-set data. TMR1 will start counting after this instruction cycle is completed. This instruction can only be used to set the 6-bit TMR1.

Each bit's settings and its corresponding function options are shown below:

OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT5~0)					
	CS3	CS2	CS1	CS0	CT5	CT4	CT3	CT2	CT1	CT0
TMSX X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
TMS Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
TMS @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

The table below explains how to set the clock source for 6-bit TMR1:

Bit setting in instruction				clock source of timer 1
CS3	CS2	CS1	CS0	
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13

With the TMSX instruction, Immediate data (X) can be used to set the clock source and pre-set data of TMR1.

With the TMS Rx instruction, the content of the data RAM pointed to by Rx and the content of AC is used to set the clock source and pre-set data of TMR1.

With the TMS @HL instruction, the content of the Table ROM pointed to by HL is used to set the clock source and pre-set data of TMR1.

Notes:
(1). The TMS# @HL syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). Absolute addressing must be used for Rx syntax.
(3). X = 0 ~ 1FFh.

Instruction Syntax:

OP code	Operand	Instruction Operation
TMS	Rx	Preset data ← (AC1, 0), ((Rx)3 ~ (Rx)0) Selection of clock source ← (AC3, 2)
TMS	@HL	Preset data ← T(@HL)5 ~ T(@HL)0 Selection of clock source ← T(@HL)7 ~ T(@HL)6
TMS#	@HL	Preset data ← T(@HL)5 ~ T(@HL)0

OP code	Operand	Instruction Operation
		Selection of clock source ← T(@HL)7 ~ T(@HL)6 HL ← HL + 1
TMSX	X	Preset data ← (X5 ~ X0) Selection of clock source ← (X8 ~ X6)

T1XH, T1RH, T1TH

Instruction Characteristics:

T1TH: Single-word instruction, 4 machine cycles, 16 bits of data transferred.
T1RH: Two-word instruction, 8 machine cycles, 16 bits of data transferred.
T1XH: Two-word instruction, 8 machine cycles.

Instruction Explanation:

The instruction uses the Immediate data (SD) set by SETDAT, the Table ROM content pointed to by HL or the data RAM pointed to by Rx to set the 6-bit TMR1's clock source and the timer's pre-set data. TM1 will start counting after this instruction cycle is completed.

This instruction can only be used to set the 6-bit TMR1.

Each bit's settings and its corresponding function options are shown below: (Any bit settings not listed in the table below can be ignored)

	Select clock(CS3~0)				Pre-set data of timer(CT5~0)					
OPCODE	CS3	CS2	CS1	CS0	CT5	CT4	CT3	CT2	CT1	CT0
T1XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6
T1TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6
	Select clock(CS3~0)				Pre-set data of timer(CT5~0)					
OPCODE	CS3	CS2	CS1	CS0	CT5	CT4	CT3	CT2	CT1	CT0
T1RH SETDAT RX	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2
	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)11	(Rx)10	(Rx)9	(Rx)8	(Rx)7	(Rx)6
Rx address	Rx(RAM Address bit1,0=11)				Rx(RAM Address bit1,0=10)				Rx(RAM Address bit1,0=01)	

When these instructions are used to set the 6-bit TMR1, the settings for SD5~SD0, TD5~TD0 and (Rx)5~(Rx)0 won't affect the operation of TMR1.

The table below explains how to set the clock source for 6-bit TMR1:

Bit setting in instruction				
CS3	CS2	CS1	CS0	clock source of timer 1
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13
1	0	0	0	CX
1	0	0	1	CX2
1	0	1	0	INT

With the T1XH instruction, the Immediate data (X) set by SETDAT can be used to directly set the clock source and pre-set data of TMR1.

With the T1RH instruction, the two least significant bits of the Rx address set by SETDAT will be ignored. 16-bit of data is read simultaneously from the contiguous addresses Rx (RAM address bit1,0=00), Rx (RAM address bit1,0=01), Rx (RAM address bit1,0=10) and Rx (RAM address bit1,0=11) in the Data RAM then used for setting the TMR1's clock source and initial value.

With the T1TH instruction, the LSB of @HL will be ignored. 16-bit of data is read simultaneously from the contiguous addresses @HL (RAM address bit0=0) and @HL (RAM address bit0=1) in the Table ROM then used for setting the TMR1's clock source and initial value.

Notes:
(1). The T1THS# X syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). SD = 0 ~ FFFFh
(3). Absolute addressing must be used for Rx syntax.
(4). MCU will temporarily suspend all interrupt requests during the instruction execution with 8 machine cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
T1XH SETDAT	SD	Preset data ← (SD11 ~ SD6) Selection of clock source ← (SD15 ~ SD12)
T1TH	@HL	Preset data ← T(@HL)11 ~ T(@HL)6 Selection of clock source ← T(@HL)15 ~ T(@HL)12
T1TH#	@HL	Preset data ← T(@HL)11 ~ T(@HL)6 Selection of clock source ← T(@HL)15 ~ T(@HL)12 HL ← HL + 2
T1RH SETDAT	Rx	Preset data ← ((Rx)11 ~ (Rx)6) Selection of clock source ← ((Rx)15 ~ (Rx)12)

TM2, TM2X

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4/8 bits of data transferred.

Instruction Explanation:
The instruction operand's Immediate data (X), the Table ROM content pointed to by @HL, AC and the content of the data RAM pointed to by Rx can be used to set the 6-bit TMR2's clock source and the timer's pre-set data. TMR2 will start counting after this instruction cycle is completed. This instruction can only be used to set the 6-bit TMR2.

Each bit's settings and its corresponding function options are shown below:

	Select clock(CS3~0)				Pre-set data of timer(CT5~0)					
OPCODE	CS3	CS2	CS1	CS0	CT5	CT4	CT3	CT2	CT1	CT0
TM2X X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
TM2 Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
TM2 @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

The table below explains how to set the clock source for 6-bit TMR2:

Bit setting in instruction				clock source of timer 2
CS3	CS2	CS1	CS0	
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13

With the TM2X X instruction, Immediate data (X) can be used to set the clock source and pre-set data of TMR2.

With the TM2 Rx instruction, the content of the data RAM pointed to by Rx and the content of AC is used to set the clock source and pre-set data of TMR2.

With the TM2 @HL instruction, the content of the Table ROM pointed to by HL is used to set the clock source and pre-set data of TMR2.

Notes:
(1). The TM2# @HL syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). Absolute addressing must be used for Rx syntax.
(3). X = 0 ~ 1FFh

Instruction Syntax:

OP code	Operand	Instruction Operation
TM2	Rx	Preset data ← (AC1, 0), ((Rx)3 ~ (Rx)0) Selection of clock source ← (AC3, 2)
TM2	@HL	Preset data ← T(@HL)5 ~ T(@HL)0 Selection of clock source ← T(@HL)7 ~ T(@HL)6
TM2#	@HL	Preset data ← T(@HL)5 ~ T(@HL)0 Selection of clock source ← T(@HL)7 ~ T(@HL)6 HL ← HL + 1
TM2X	X	Preset data ← (X5 ~ X0) Selection of clock source ← (X8 ~ X6)

T2XH, T2RH, T2TH

Instruction Characteristics:
T2TH : Single-word instruction, 4 machine cycles, 16 bits of data transferred.
T2RH : Two-word instruction, 8 machine cycles, 16 bits of data transferred.
T2XH : Two-word instruction, 8 machine cycles.

Instruction Explanation:
The instruction uses the Immediate data (SD) set by SETDAT, the Table ROM content pointed to by HL or the data RAM pointed to by Rx to set the 12-bit or 6-bit TMR2's clock

source and the timer's pre-set data. TMR2 will start counting after this instruction cycle is completed.

Each bit's settings and its corresponding function options are shown below:

OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT11~0)											
T2XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
T2TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT11~0)											
T2RH	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
SETDAT RX	(RX)15	(RX)14	(RX)13	(RX)12	(RX)11	(RX)10	(RX)9	(RX)8	(RX)7	(RX)6	(RX)5	(RX)4	(RX)3	(RX)2	(RX)1	(RX)0
Rx address	Rx(RAM Address bit1,0=11)				Rx(RAM Address bit1,0=10)				Rx(RAM Address bit1,0=01)				Rx(RAM Address bit1,0=00)			

When these instructions are used to set the 6-bit TMR2, the settings for SD11~SD6, TD11~TD6 and (Rx)11~(Rx)6 won't affect the operation of TMR2.

The table below explains how to set the clock source for 6-bit or 12-bit TMR2:

Bit setting in instruction				clock source of timer 2
CS3	CS2	CS1	CS0	
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13
1	0	0	0	CX
1	0	0	1	CX2
1	0	1	0	INT

With the T2XH instruction, the Immediate data (X) set by SETDAT can be used to directly set the clock source and pre-set data of TMR2.

With the T2RH instruction, the two least significant bits of the Rx address set by SETDAT will be ignored. 16-bit of data is read simultaneously from the contiguous addresses Rx (RAM address bit1,0=00), Rx (RAM address bit1,0=01), Rx (RAM address bit1,0=10) and Rx (RAM address bit1,0=11) in the Data RAM then used for setting the TMR2's clock source and initial value.

With the T2TH instruction, the LSB of @HL will be ignored. 16-bit of data is read simultaneously from the contiguous addresses @HL(RAM address bit0=0) and @HL(RAM address bit0=1) in the Table ROM then used for setting the TMR2's clock source and initial value.

Notes:

(1). The T2THS# syntax will automatically increment the content of HL by 1 after executing this instruction.

(2). Absolute addressing must be used for Rx syntax.

(3). SD = 0 ~ FFFFh.
(4). MCU will temporarily suspend all interrupt requests during the instruction execution with 8 machine cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
T2XH SETDAT	SD	Preset data ← (SD11 ~ SD0) Selection of clock source ← (SD15 ~ SD12)
T2TH	@HL	Preset data ← T(@HL)11 ~ T(@HL)0 Selection of clock source ← T(@HL)15 ~ T(@HL)12
T2TH#	@HL	Preset data ← T(@HL)11 ~ T(@HL)0 Selection of clock source ← T(@HL)15 ~ T(@HL)12 HL ← HL + 2
T2RH SETDAT	Rx	Preset data ← ((Rx)11 ~ (Rx)0) Selection of clock source ← ((Rx)15 ~ (Rx)12)

TM3, TM3X

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4/8 bits of data transferred.

Instruction Explanation:
The instruction operand's Immediate data (X), the Table ROM content pointed to by @HL, AC and the content of the data RAM pointed to by Rx can be used to set the 6-bit TMR3's clock source and the timer's pre-set data. TMR3 will start counting after this instruction cycle is completed. This instruction can only be used to set the 6-bit TMR3.

Each bit's settings and its corresponding function options are shown below:

	Select clock(CS3~0)				Pre-set data of timer(CT5~0)					
OPCODE	CS3	CS2	CS1	CS0	CT5	CT4	CT3	CT2	CT1	CT0
TM3X X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
TM3 Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
TM3 @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

The table below explains how to set the clock source for 6-bit TMR3:

Bit setting in instruction				
CS3	CS2	CS1	CS0	clock source of timer 3
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13

With the TM3X X instruction, Immediate data (X) can be used to set the clock source and pre-set data of TMR3.

With the TM3 Rx instruction, the content of the data RAM pointed to by Rx and the content of AC is used to set the clock source and pre-set data of TMR3.

With the TM3 @HL instruction, the content of the Table ROM pointed to by HL is used to set the clock source and pre-set data of TMR3.

Notes:

(1). The TM3# @HL syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). Absolute addressing must be used for Rx syntax.
(3). X = 0 ~ 1FFh

Instruction Syntax:

OP code	Operand	Instruction Operation
TM3	Rx	Preset data ← (AC1, 0), ((Rx)3 ~ (Rx)0) Selection of clock source ← (AC3, 2)
TM3	@HL	Preset data ← T(@HL)5 ~ T(@HL)0 Selection of clock source ← T(@HL)7 ~ T(@HL)6
TM3#	@HL	Preset data ← T(@HL)5 ~ T(@HL)0 Selection of clock source ← T(@HL)7 ~ T(@HL)6 HL ← HL + 1
TM3X	X	Preset data ← (X5 ~ X0) Selection of clock source ← (X8 ~ X6)

T3XH, T3RH, T3TH

Instruction Characteristics:

T3TH: Single-word instruction, 4 machine cycles, 16 bits of data transferred.
T3RH: Two-word instruction, 8 machine cycles, 16 bits of data transferred.
T3XH: Two-word instruction, 8 machine cycles.

Instruction Explanation:

The instruction uses the Immediate data (SD) set by SETDAT, the Table ROM content pointed to by HL or the data RAM pointed to by Rx to set the 12-bit or 6-bit TMR3's clock source and the timer's pre-set data. TMR3 will start counting after this instruction cycle is completed.

Each bit's settings and its corresponding function options are shown below:

OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT11~0)											
T3XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
T3TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT11~0)											
T3RH SETDAT RX	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
	(RX)15	(RX)14	(RX)13	(RX)12	(RX)11	(RX)10	(RX)9	(RX)8	(RX)7	(RX)6	(RX)5	(RX)4	(RX)3	(RX)2	(RX)1	(RX)0
Rx address	Rx(RAM Address bit1,0=11)				Rx(RAM Address bit1,0=10)				Rx(RAM Address bit1,0=01)				Rx(RAM Address bit1,0=00)			

When these instructions are used to set the 6-bit TMR3, the settings for SD11~SD6, TD11~TD6 and (Rx)11~(Rx)6 won't affect the operation of TMR3.

The table below explains how to set the clock source for 6-bit or 12-bit TMR3:

Bit setting in instruction				
CS3	CS2	CS1	CS0	clock source of timer 3
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13
1	0	0	0	CX
1	0	0	1	CX2
1	0	1	0	INT

With the T3XH instruction, the Immediate data (X) set by SETDAT can be used to directly set the clock source and pre-set data of TMR3.

With the T3RH instruction, the two least significant bits of the Rx address set by SETDAT will be ignored. 16-bit of data is read simultaneously from the contiguous addresses Rx (RAM address bit1,0=00), Rx (RAM address bit1,0=01), Rx (RAM address bit1,0=10) and Rx (RAM address bit1,0=11) in the Data RAM then used for setting the TMR3's clock source and initial value.

With the T3TH instruction, the LSB of @HL will be ignored. 16-bit of data is read simultaneously from the contiguous addresses @HL(RAM address bit0=0) and @HL(RAM address bit0=1) in the Table ROM then used for setting the TMR3's clock source and initial value.

Notes:

(1). The T3TH# syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). SD = 0 ~ FFFFh
(3). Absolute addressing must be used for Rx syntax.
(4). MCU will temporarily suspend all interrupt requests during the instruction execution with 8 machine cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
T3XH SETDAT	 SD	Preset data ← (SD11 ~ SD0) Selection of clock source ← (SD15 ~ SD12)
T3TH	@HL	Preset data ← T(@HL)11 ~ T(@HL)0 Selection of clock source ← T(@HL)15 ~ T(@HL)12
T3TH#	@HL	Preset data ← T(@HL)11 ~ T(@HL)0 Selection of clock source ← T(@HL)15 ~ T(@HL)12 HL ← HL + 2
T3RH SETDAT	 Rx	Preset data ← ((Rx)11 ~ (Rx)0) Selection of clock source ← ((Rx)15 ~ (Rx)12)

RTM2L, RTM21, RTM1H, RTM3L, RTM31

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4 bits of data transferred.

Instruction Explanation:
Read the current content from TMR1~TMR3 or a specific 18-bit register then store to the data RAM pointed to by Rx and AC.

The table bellows shows which timers can be read by each instruction:

timer read-out inst.	Content of TMR1						Content of TMR3						Content of TMR2					
	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RTM2L															(Rx)3	(Rx)2	(Rx)1	(Rx)0
RTM1H	(Rx)3	(Rx)2	(Rx)1	(Rx)0														
RTM3L									(Rx)3	(Rx)2	(Rx)1	(Rx)0						
RTM31					(Rx)3	(Rx)2	(Rx)1	(Rx)0										
RTM21					(Rx)3	(Rx)2							(Rx)1	(Rx)0				

Note:
Absolute addressing must be used for Rx syntax.

Instruction Syntax:

OP code	Operand	Instruction Operation
RTM2L	Rx	(Rx), AC ← TMR2(bit3 ~ bit0)
RTM21	Rx	(Rx), AC ← TMR1(bit1, bit0), TMR2(bit5, bit4)
RTM1H	Rx	(Rx), AC ← TMR1(bit5 ~ bit2)
RTM3L	Rx	(Rx), AC ← TMR3(bit3 ~ bit0)
RTM31	Rx	(Rx), AC ← TMR1(bit1, bit0), TMR3(bit5, bit4)

T2M3X

Instruction Characteristics:
Two-word instruction, 8 machine cycles.

Instruction Explanation:
The data of the instruction operand X and Immediate data (SD) can be used to set the 18-bit TMR2's clock source and timer's pre-set data.

TMR2 will start counting after the instruction cycle is completed.

Each bit's settings and its corresponding function options are shown below:

OPCODE	Select clock source(CS3~0)				Set timer's initial value (CT17~0)					
T2M3X X SETDAT SD	SD15	SD14	SD13	SD12	CT17	CT16	CT15	CT14	CT13	CT12
					X5	X4	X3	X2	X1	X0
					CT11	CT10	CT19	CT18	CT17	CT16
					SD11	SD10	SD9	SD8	SD7	SD6
					CT5	CT4	CT3	CT2	CT1	CT0
					SD5	SD4	SD3	SD2	SD1	SD0

The table below explains how to set the clock source for 18-bit TMR2:

Bit setting in instruction				clock source of TMR2
CS3	CS2	CS1	CS0	
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13

Notes:
(1). This instruction can only be used to set the 18-bit TMR2.
(2). Absolute addressing must be used for Rx syntax.
(3). X = 0 ~ 3Fh.
(4). SD = 0 ~ FFFFh
(5). MCU will temporarily suspend all interrupt requests during this instruction cycle with 8 machine cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
T2M3X SETDAT	X SD	Pre-set data of TMR2 ← (X5 ~ 0), (SD11 ~ SD0) Select clock source ← (SD15 ~ SD12) TMR2 starts counting

STM

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to combine different TMR as a 12 or 18-bit timer.

X1, X0 = 00, set TMR1, TMR2 and TMR3 as three independent 6-bit timers. This is the default setting of the MCU.

X1, X0 = 01, combine TMR1 and TMR2, with TMR2 becoming a 12-bit timer (TMR1 takes the role of bit11~bit6 in 12-bit TMR2). TMR3 remains as an individual 6-bit timer.

X1, X0 = 10, combine TMR1 and TMR3, with TMR3 becoming a 12-bit timer (TMR1 takes the role of bit11~bit6 in 12-bit TMR3). TMR2 remains as an individual 6-bit timer.

X1, X0 = 11, combine TMR1, TMR3 and TMR2, with TMR2 becoming a 18-bit timer. (TMR1 takes the role of bit11~bit6 and TMR3 takes the role of bit17~bit12, in 18-bit TMR2)

Notes:
(1). Executing the STM instruction will stop the counting of all Timers (including disabled the re-load function) so make sure that TMR1, 2 and 3 are not counting before executing STM.
(2). X1 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
STM	X	TMR1, 2, 3 are 6-bit ← X = 0 (default) TMR2 = TMR1 + TMR2 ← X = 1 TMR2 (12-bit), TMR3 (6-bit) TMR3 = TMR1 + TMR3 ← X = 2 TMR3 (12-bit), TMR2 (6-bit) TMR2 = TMR3 + TMR1 + TMR2 ← X = 3 TMR2 (18-bit)

DISTM

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to stop the counting of TMR1 ~ TMR3.
X2 = 1, stop TMR3 (includes disabled the re-load function)
X1 = 1, stop TMR2 (includes disabled the re-load function)
X0 = 1, stop TMR1 (includes disabled the re-load function)

Note:
X2 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
DISTM	X	Stop TMR3 ← X2 = 1 Stop TMR2 ← X2 = 1 Stop TMR1 ← X2 = 1

2-12. Peripheral Function Instructions

SPKTH, SPKXH, SPKRH

Instruction Characteristics:
SPKTH: Single-word instruction, 4 machine cycles, 16 bits of data transferred.
SPKRH: Two-word instruction, 8 machine cycles, 16 bits of data transferred.
SPKXH: Two-word instruction, 8 machine cycles.

Instruction Explanation:
The data of the instruction's operand can be used to choose whether the key matrix scanning function's output pins (KO) output the scanning signal and how a HALT release request is generated.

D = 0, once the key matrix scanning function detects any scan signal on input pins KI1~4, the halt release request flag 5 (HRF5) is set to 1.
= 1, at the end of each scanning cycle the key matrix scanning function sets the halt release request flag 5 (HRF5) to 1.

The table below shows the correspondence between instruction's operand bits and the KO pins.

Instruction	KO16	KO15	KO14	KO13	KO12	KO11	KO10	KO9	KO8	KO7	KO6	KO5	KO4	KO3	KO2	KO1
SPKXH D SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
SPKTH(#) D, @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
SPKRH D SETDAT Rx	(RX)15	(RX)14	(RX)13	(RX)12	(RX)11	(RX)10	(RX)9	(RX)8	(RX)7	(RX)6	(RX)5	(RX)4	(RX)3	(RX)2	(RX)1	(RX)0
Rx address	Rx(RAM Address bit1,0=11)				Rx(RAM Address bit1,0=10)				Rx(RAM Address bit1,0=01)				Rx(RAM Address bit1,0=00)			

With the SPKXH D instruction, the Immediate data (SD) set by SETDAT represents the corresponding KO pin could output the scan signal.

With the SPKTH D instruction, the LSB of @HL will be ignored. 16-bit of data is then read from the contiguous addresses @HL(RAM address bit0=0) and @HL(RAM address bit0=1) in the Table ROM and used to set the scan signal output of the corresponding KO pin.

With the SPKRH D instruction, the two least significant bits of the Rx address set by SETDAT will be ignored. 16-bit of data is then read from the contiguous addresses Rx (RAM address bit1,0=00), Rx (RAM address bit1,0=01), Rx (RAM address bit1,0=10) and Rx (RAM address bit1,0=11) in the data RAM and used to set the scan signal output of the corresponding KO pin.

Notes:
(1). The SPKTH# D syntax will automatically increment the content of HL by 2 after executing this instruction.
(2). D = 0 or 1
(3). SD = 0 ~ FFFFh
(4). Absolute addressing must be used for Rx syntax.

(5). MCU will temporarily suspend all interrupt requests during the instruction cycle with 8 machine cycles.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPKXH SETDAT	D SD	Select method of generating halt released ← D Set the scan signal for KO pin ← SD16 ~ SD0
SPKTH	D, @HL	Select method of generating halt released ← D Set the scan signal for KO pin ← T(@HL)15 ~ T(@HL)0
SPKTH#	D, @HL	Select method of generating halt released ← D Set the scan signal for KO pin ← T(@HL)15 ~ T(@HL)0 HL ← HL + 2
SPKRH SETDAT	D Rx	Select method of generating halt released ← D Set the scan signal for KO pin ← (Rx)15 ~ (Rx)0

SPK, SPKX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4/8 bits of data transferred.

Instruction Explanation:
The value of the instruction operand can set the scanning method used by the key matrix scanning function and how a HALT release request is generated.
The table below shows the correspondence between SPK instruction's operand bits and the KO pins.

Operand bit definition	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPKX X	X7	X6	X5	X4	X3	X2	X1	X0
SPK Rx	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
SPK(#) @HL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

X6 = 0, once the key matrix scanning function detects any scan signal on input pins KI1~4, the halt release request flag 5 (HRF5) is set to 1.
= 1, when a scan signal synchronized with the LCD waveform appears, the key matrix scanning function will set the halt release request flag 5 (HRF5) to 1.

X7, X5, X4 = 000, set it so that in each scanning cycle only one KO output pin will send a scan signal. The output pin is selected using X3 ~ X0.
X3 ~ X0 = 0000, choose to send scan signal with KO1
X3 ~ X0 = 0001, choose to send scan signal with KO2
……..
X3 ~ X0 = 1110, choose to send scan signal with KO15
X3 ~ X0 = 1111, choose to send scan signal with KO16

X7, X5, X4 = 001, set is so that in each scanning cycle all KO output pins (KO1 ~ KO16) will simultaneously send scan signals. X3 ~ X0 can be set to any value.

X7, X5, X4 = 010, disable the key matrix scanning function. X3 ~ X0 can be set to any value.

X7, X5, X4 = 10X, set it so that in each scanning cycle only 8 KO output pins will send scan signal simultaneously. The output pins are selected using X3.

X3 = 0, choose to send scan signal with KO1 ~ KO8
X3 = 1, choose to send scan signal with KO9 ~ KO16
Here X4, X2 ~ X0 can be set to any value.

X7, X5, X4 = 110, set it so that in each scanning cycle only 4 KO output pins will send scan signal simultaneously. The output pins are selected using X3 and X2.
X3X2 = 00, choose to send scan signal with KO1 ~ KO4
X3X2 = 01, choose to send scan signal with KO5 ~ KO8
X3X2 = 10, choose to send scan signal with KO9 ~ K12
X3X2 = 11, choose to send scan signal with K13 ~ K16
X1 and X0 can be set to any value.

X7, X5, X4 = 111, set it so that in each scanning cycle only 2 KO output pins will send scan signal simultaneously. The output pins are selected using X3, X2 and X1.
X3X2X1 = 000, choose to send scan signal with KO1 ~ KO2
X3X2X1 = 001, choose to send scan signal with KO3 ~ KO4
……
X3X2X1 = 110, choose to send scan signal with K13 ~ K14
X3X2X1 = 111, choose to send scan signal with K15 ~ K16
X0 can be set to any value.

Notes:
(1). The SPK# @HL syntax will automatically increment the value of HL by 1 after executing this instruction.
(2). Absolute addressing must be used for Rx syntax.
(3). X7 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SPK	Rx	Select method of generating halt released ← AC2 Select scanning method ← AC3, AC1, AC0, (Rx)3 ~ (Rx)0
SPK	@HL	Select method of generating halt released ← T(@HL)6 Select scanning method ← T(@HL)7, T(@HL)5 ~ T(@HL)0
SPK#	@HL	Select method of generating halt released ← T(@HL)6 Select scanning method ← T(@HL)7, T(@HL)5 ~ T(@HL)0 HL ← HL + 1
SPKX	X	Select method of generating halt released ← X6 Select scanning method ← X7, X5 ~ X0

ELC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to choose how to drive the EL panel.

Setting the ELP output pins' output frequency and duty cycle:

X8, 7, 6 = 111, choose BCKL/8 as the ELP output waveform.
= 110, choose BCKL/4 as the ELP output waveform.
= 101, choose BCKL/2 as the ELP output waveform.
= 100, choose BCKL as the ELP output waveform.
= 011, choose the inverted waveform of the frequency generator's output signal (FREQ) as the ELP output waveform.
= 000, choose PH0 as the ELP output waveform.

X9, 5, 4 = 101, choose 2/3 duty as the duty cycle of the ELP output waveform.
= 100, choose 3/4 duty as the duty cycle of the ELP output waveform.
= x11, choose 1/1 duty as the duty cycle of the ELP output waveform.
= x10, choose 1/2 duty as the duty cycle of the ELP output waveform.
= 001, choose 1/3 duty as the duty cycle of the ELP output waveform.
= 000, choose 1/4 duty as the duty cycle of the ELP output waveform.

Setting the ELC output pins' output frequency and duty cycle:

X3, 2 = 11, choose PH5 as the ELC output waveform.
= 10, choose PH6 as the ELC output waveform.
= 01, choose PH7 as the ELC output waveform.
= 00, choose PH8 as the ELC output waveform.

X1, 0 = 11, choose 1/1 as the duty cycle of the ELC output waveform.
= 10, choose 1/2 as the duty cycle of the ELC output waveform.
= 01, choose 1/3 as the duty cycle of the ELC output waveform.
= 00, choose 1/4 as the duty cycle of the ELC output waveform.

Notes:
(1). X9 represents the MSB of operand X, X0 represents the LSB of operand X.
(2). When the ELP output waveform's duty cycle is set to 1/1 or 1/2 duty, X9 can be any value.

Instruction Syntax:

OP code	Operand	Instruction Operation
ELC	X	Set ELP pin's output frequency ← X8, 7, 6 Set ELP pin's output duty cycle ← X9, 5, 4 Set ELC pin's output frequency ← X3, 2 Set ELC pin's output duty cycle ← X1, 0

ALM

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
The value of the instruction operand X can be used to synthesize a modulated waveform. The envelope of the modulated waveform and the carrier's signal source can all be set using the X value.

X8, 7, 6 = 111, choose the frequency generator output (FREQ) as the carrier signal
= 100, choose direct-current logic high (DC1) as the carrier signal
= 011, choose PH3 as the carrier signal
= 010, choose PH4 as the carrier signal
= 001, choose PH5 as the carrier signal
= 100, choose direct-current logic low (DC0) as the carrier signal

X5 = 1, choose to include PH15 frequency element in envelope signal.

X4 = 1, choose to include PH14 frequency element in envelope signal.

X3 = 1, choose to include PH13 frequency element in envelope signal.

X2 = 1, choose to include PH12 frequency element in envelope signal.

X1 = 1, choose to include PH11 frequency element in envelope signal.

X0 = 1, choose to include PH10 frequency element in envelope signal.

Notes:
(1). The Envelope signal is an output of AND logic with the input signals selected by X5~X0.
(2). The modulated waveform only outputs the carrier signal when the envelope signal is in GND state.
(3). X8 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
ALM	X	Set carrier signal in modulated waveform ← X8, 7, 6 Set envelope signal in modulated waveform ← X5 ~ 0

SBZ

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to select whether the output waveforms from the two output pins BZB and BZ comes from the output signal of the frequency generator or the modulated signal generated by the ALM instruction's settings.

X1 = 0, invert the modulated signal set by the ALM instruction then output it to the BZB pin. This is the default setting of the MCU.
= 1, invert the output signal from the frequency generator (FREQ) then output it to the BZB pin.

X0 = 0, directly output the modulated signal set by the ALM instruction to the BZ pin. This is the default setting of the MCU.
= 1, directly output the output signal from the frequency generator (FREQ) to the BZ pin.

Note:
X1 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SBZ	X	Select signal source for BZB pin ← X1 Select signal source for BZ pin ← X0

2-13. System Function Instructions

NOP

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
The MCU does not do anything after executing this instruction but it will take up 4 machine cycles. This can be used as a program delay.

Instruction Syntax:

OP code	Operand	Instruction Operation
NOP		No operation

HALT

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
MCU enters the HALT mode.
The MCU will release the HALT mode if one of the four following situations occurs:
(1). When any interrupt request occurs.
After the program completes the interrupt service routine and executes the RTS instruction, the MCU will return to HALT mode again.

(2). When the conditions set by the SCA instruction are met (when the MCU determines that the changes to the input signals for the three IO ports IOA, IOC and IOD are valid)

(3). When the condition set by the SCX instruction is met (when the RFC counter is controlled by CX2/CX pin and the control signal from CX2/CX pin becomes inactive)

(4). When the condition set by the SHE instruction is met.

Instruction Syntax:

OP code	Operand	Instruction Operation
HALT		MCU enters HALT mode

STOP

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
MCU enters STOP mode and stops all oscillator operations.

The MCU will release the STOP mode if one of the three following situations occurs:

(1). When a high level signal appears on any one of the IOA, IOC and IOD input pins and the condition set by the SRE instruction is met. Keep the high level signal on the IO pins until the MCU releases the HALT mode.

If "Chatter Prevention" function is not selected for the IOA, IOC and IOD ports then the SRE instruction does not need to be executed for setting the STOP release conditions, and the high level signal on the IO pins needn't keep till the MCU releases HALT mode.

(2). The input signal change occurred on INT pin.

(3). When the key matrix scanning function detects a low level input signal on pins KI1~4.

Instruction Syntax:

OP code	Operand	Instruction Operation
STOP		MCU enters STOP mode.

FRQ, FRQX

Instruction Characteristics:
Single-word instruction, 4 machine cycles, 4/8 bits of data transferred.

Instruction Explanation:
The Immediate data (D) in the instruction operand can be use to set the duty cycle of the output waveform from the frequency generator. At the same time, the contents of RX and AC, the content of Table ROM pointed to by @HL and Immediate data (X) can be used to set the initial value of programmable divider in frequency generator. The frequency generator starts to output the waveform after executing this instruction.

The table below shows the correspondence between the instruction operand value and the initial value of programmable divider in frequency generator:

Instruction\Operand	Bit correspondence table for the initial value of programmable divider							
	bit7	Bit6	bit 5	bit 4	bit 3	Bit 2	bit 1	bit 0
FRQ D,Rx	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
FRQ D,@HL	T7	T6	T5	T4	T3	T2	T1	T0
FRQX D,X	X7	X6	X5	X4	X3	X2	X1	X0

Notes: 1. T0 ~ T7 represents the data of table ROM.
2. X0 ~ X7 represents the Immediate data specified in operand X.

To set the duty cycle of the output waveform from the frequency generator, use the following method:

D = 0, output 1/4 duty waveform.
= 1, output 1/3 duty waveform.
= 2, output 1/2 duty waveform.
= 3, output 1/1 duty waveform.

Notes:
(1). The FRQ# syntax will automatically increment the content of HL by 1 after executing this instruction.
(2). Absolute addressing must be used for Rx syntax.
(3). D = 0 ~ 3h.
(4). X = 0 ~ FFh.

Instruction Syntax:

OP code	Operand	Instruction Operation
FRQ	D, Rx	Programmable divider ← AC, (Rx) Duty cycle generator ← D
FRQ	D, @HL	Programmable divider ← T(@HL) Duty cycle generator ← D
FRQ#	D, @HL	Programmable divider ← T(@HL) Duty cycle generator ← D HL = HL + 1
FRQX	D, X	Programmable divider ← X Duty cycle generator ← D

SCC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to select the clock source of the frequency generator and select the clock source for the chattering prevention function on the IOA, IOC and IOD ports.

X6 = 1, select system clock (BCLK) as the clock source for the frequency generator
= 0, select PH0 as the clock source for the frequency generator

X5 = 1, choose to set the clock source of chattering prevention function on the IOA port

X4 = 1, choose to set the clock source of chattering prevention function on the IOC port

X3 = 1, choose to set the clock source of chattering prevention function on the IOD port

X2, X1, X0 = 001, set PH10 as the clock source for the IO port chattering prevention function selected by X5~X3
X2, X1, X0 = 010, set PH18 as the clock source for the IO port chattering prevention function selected by X5~X3
X2, X1, X0 = 100, set PH6 as the clock source for the IO port chattering prevention function selected by X5~X3

Note:
X6 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SCC	X	Select clock source for frequency generator ← X6 Whether clock source can be set for IOA's chattering prevention ← X5 Whether clock source can be set for IOC's chattering prevention ← X4 Whether clock source can be set for IOD's chattering prevention ← X3 Select clock source for chattering prevention ← X2, 1, 0

SCA

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the 3 flags SEF5, 4, 3.

When there is any change in the input signal on the 3 IO ports IOA, IOC and IOD, a HALT request is sent to the MCU.

X5 = 1, set SEF5 flag to 1. When there is any change in the input signal on IOA port the MCU generates a HALT release.

X4 = 1, set SEF4 flag to 1. When there is any change in the input signal on IOC port the MCU generates a HALT release.

X3 = 1, set SEF3 flag to 1. When there is any change in the input signal on IOD port the MCU generates a HALT release.

Notes:
(1). X2~0 can be of any value.
(2). X5 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SCA	X	Set/clear SEF5 flag ← X5 Set/clear SEF4 flag ← X4 Set/clear SEF3 flag ← X3

SCX

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the two flags SEF1 and SEF0.

If the CX2/CX pin control function of the RFC counter is enabled, the RFC function will send a HALT release request when the control signal becomes inactive.

X1 = 1, set SEF1 flag to 1. When the CX2 pin control signal becomes inactive the MCU generates a HALT release.

X0 = 1, set SEF0 flag to 1. When the CX pin control signal becomes inactive the MCU generates a HALT release.

Note:
X1 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SCX	X	Set/clear SEF1 flag ← X1 Set/clear SEF0 flag ← X0

SIE*

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set each interrupt enable flags.

Setting these flags allows the MCU to decide whether to accept these interrupt requests and execute their interrupt service.

X7 = 1, set IEF7 to 1, MCU will accept interrupt request generated by underflow at TMR3.

X6 = 1, set IEF6 to 1, MCU will accept interrupt request generated by the cessation of the RFC counter control signal (under CX or CX2 pin control mode).

X5 = 1, set IEF5 to 1, MCU will accept interrupt request generated by the detection of scan signal by the key matrix scanning function.

X4 = 1, set IEF4 to 1, MCU will accept interrupt request generated by underflow at TMR2.

X3 = 1, set IEF3 to 1, MCU will accept interrupt request generated due to a change in the Pre-divider's PH15 output signal.

X2 = 1, set IEF2 to 1, MCU will accept interrupt request generated by a valid trigger on the INT pin.

X1 = 1, set IEF1 to 1, MCU will accept interrupt request generated by underflow at TMR1.

X0 = 1, set IEF0 to 1, MCU will accept interrupt request generated by a valid change in input signal on the IOA, IOC and IOD ports.

Note:
X7 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SIE*	X	Set/clear IEF7 flag ← X7 Set/clear IEF6 flag ← X6 Set/clear IEF5 flag ← X5 Set/clear IEF4 flag ← X4 Set/clear IEF3 flag ← X3 Set/clear IEF2 flag ← X2 Set/clear IEF1 flag ← X1 Set/clear IEF0 flag ← X0

SHE

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set each halt release enable flags.

X7 = 1, set HEF7 to 1, MCU allowed to generate a HALT release when there is an underflow at TMR3.

X5 = 1, set HEF5 to 1, MCU allowed to generate HALT release upon detection of scan signal by the key matrix scanning function.

X4 = 1, set HEF4 to 1, MCU allowed to generate a HALT release when there is an underflow at TMR2.

X3 = 1, set HEF3 to 1, MCU allowed to generate HALT release when there is a change in the Pre-divider's PH15 output signal.

X2 = 1, set HEF2 to 1, MCU allowed to generate a HALT release when there is a valid trigger on the INT.

X1 = 1, set HEF1 to 1, MCU allowed to generate a HALT release when there is an underflow at TMR1.

Notes:
(1). X7 represents the MSB of operand X, X0 represents the LSB of operand X.
(2). The two bits X6 and X0 must be fixed as 0.

Instruction Syntax:

OP code	Operand	Instruction Operation
SHE	X	Set/clear HEF7 flag ← X7 Set/clear HEF5 flag ← X5 Set/clear HEF4 flag ← X4 Set/clear HEF3 flag ← X3 Set/clear HEF2 flag ← X2 Set/clear HEF1 flag ← X1

SRE

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to set the stop release enable flags (SRF6, SRF4, SRF3).

This instruction is only needed when the chattering prevention function is enabled for ports IOA, IOC and IOD.

X6 = 1, set SRF6 to 1, MCU allowed to generate a STOP release when a high level signal appears on IOA port's input pin.

X4 = 1, set SRF4 to 1, MCU allowed to generate a STOP release when a high level signal appears on IOC port's input pin.

X3 = 1, set SRF3 to 1, MCU allowed to generate a STOP release when a high level signal appears on IOD port's input pin.

Notes:
(1). X6 represents the MSB of operand X, X0 represents the LSB of operand X.
(2). Bits X5, X2, X1 and X0 can be of any value.

Instruction Syntax:

OP code	Operand	Instruction Operation
SRE	X	Set/clear SRF6 flag ← X6 Set/clear SRF4 flag ← X4 Set/clear SRF3 flag ← X3

FAST

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Executing this instruction will first enable the CFOSC oscillator then switch the system clock (BLCK) for instruction execution to the high speed clock (CF clock) generated by the CFOSC oscillator.

Instruction Syntax:

OP code	Operand	Instruction Operation
FAST		Switch system clock to high-speed clock.

SLOW

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Switch the system clock (BCLK) used for instruction execution to the low-speed clock (XT clock) generated by the XTOSC oscillator then disable the CFOSC oscillator.

Instruction Syntax:

OP code	Operand	Instruction Operation
SLOW		Switch system clock to low-speed clock.

SF

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data defined for the instruction operand X to enable the corresponding functions.

X7 = 1, enable the re-load function for TMR1.

X6 = 1, enable the re-load function for TMR3.

X5 = 1, the content of timers can only be accessed through an 18-bit special register. The content of TMR1~3 are simultaneously written to the 18-bit special register only after this instruction is executed.

X4 = 1, restart the watchdog timer after resetting it to zero and set the WDF flag to 1.

X3 = 1, if both X2 and X3 are set to 1 at the same time, executing this function enables the EL panel driver function and puts the MCU into HALT mode.

X2 = 1, enable the EL panel driver function.

X1 = 1, MCU enters Power Back Up mode and sets BCF flag to 1.

X0 = 1, set CF to 1.

Note:
X7 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SF	X	Enable the re-load function for TMR1 ← X7 Enable the re-load function for TMR3 ← X6 Set write before read function for accessing timer value← X5 Restart watchdog timer after reset, set WDF flag to 1 ← X4 Enable EL panel driver then enter HALT mode ← X3 Enable EL panel driver ← X2 Enter power back up mode and set BCF flag to 1 ← X1 Set CF flag to 1 ← X0

RF

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data defined for the instruction operand X to disable the corresponding functions enabled by the SF instruction.

X7 = 1, disable the re-load function for TMR1

X6 = 1, disable the re-load function for TMR3

X5 = 1, the content of timers can be accessed directly and stored to data RAM

X4 = 1, stop the watchdog timer and clear WDF flag to 0

X2 = 1, disable the EL panel driver function

X1 = 1, MCU exits Power Back Up mode and clears BCF flag to 0

X0 = 1, clear CF to 0

Note:
X7 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
RF	X	Disable the re-load function for TMR1 ← X7 Disable the re-load function for TMR3 ← X6 Allow direct access of timer value ← X5 Stop watchdog timer, clear WDF flag to 0 ← X4 Disable EL panel driver ← X2 Exit backup mode and clear BCF flag to 0 ← X1 Clear CF flag to 0 ← X0

SF2

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data defined for the instruction operand X to enable the corresponding functions.

X6 = 1, enable the use of CX2 pin as RFC input signal including the start-up of the RC oscillator and the counting of the RFC counter

X5 = 1, enable the use of CX pin as RFC input signal including the start-up of the RC oscillator and the counting of the RFC counter

X4 = 1, enable low battery voltage detection function

X3 = 1, enable the built-in low impedance pull-low component on the INT input pin

X2 = 1, force all LCD COM and SEG pins output GND signal only

X1 = 1, set Dis_ENX flag to 1, so in re-load mode TMR2 doesn't halt the RFC counter when there is timer underflow

X0 = 1, enable the re-load function for TMR2

Note:
X6 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
SF2	X	Enable CX2 control for RFC counter ← X6 Enable CX control for RFC counter ← X5 Enable low battery voltage detection function ← X4 Enable low impedance pull-low component on INT pin ← X3 Force all LCD COM and SEG pins output GND ← X2 Set Dis_ENX flag to 0 ← X1 Enable the re-load function for TMR2 ← X0

RF2

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data defined for the instruction operand X to disable the corresponding functions enabled by the SF2 instruction.

X6 = 1, disable the use of CX2 pin as input signal for RFC function

X5 = 1, disable the use of CX pin as input signal for RFC function

X4 = 1, disable the low battery voltage detection function

X3 = 1, disable the built-in low impedance pull-low component on the INT input pin

X2 = 1, Recover normal output waveforms to all LCD COM and SEG output pins

X1 = 1, Clear Dis_ENX flag to 0, so in re-load mode TMR2 halts the RFC counter when there is timer underflow

X0 = 1, disable the re-load function for TMR2

Note:
X6 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
RF2	X	Cancel CX2 control of RFC counter ← X6 Cancel CX control of RFC counter ← X6 Disable low battery voltage detection ← X4 Disable low impedance pull-low component on INT pin ← X3 Recover normal output waveforms to all COM and SEG pins ← X2 Clear Dis_ENX flag to 0 ← X1 Disable the re-load function for TMR2 ← X0

PLC

Instruction Characteristics:
Single-word instruction, 4 machine cycles.

Instruction Explanation:
Use the data of the instruction operand X to clear the corresponding halt release request flags (HRFn).

X8 = 1, clear the last 5 bits (PH11~15) of the pre-divider to 0. To avoid the generation of a halt release request (HRF3) due to clearing PH14 to 0, this setting should be used concurrently with X3.

X7 = 1, clear the halt release request flag (HRF7) set by TMR3 underflow to 0.
If TMR3 is already counting and in single countdown mode, TMR3 will stop immediately; if in re-load mode, TMR3's operation will not be affected.

X6 = 1, clear the halt release request flag (HRF6) set by the cessation of control signal for RFC counter to 0.
This setting is used to enable or disable the CX2 or CX pin control mode for the RFC counter.

X5 = 1, clear the halt release request flag (HRF5) set by the detection of scan signal by the key matrix scanning function to 0.

X4 = 1, clear the halt release request flag (HRF4) set by TMR2 underflow to 0.
If TMR2 is already counting and in single countdown mode, TMR2 will stop immediately; if in re-load mode, TMR2's operation will not be affected.

X3 = 1, clear the halt release request flag (HRF3) set by pre-divider overflow to 0.

X2 = 1, clear the halt release request flag (HRF2) set by the detection of a valid change in the INT input in to 0.

X1 = 1, clear the halt release request flag (HRF1) set by TMR1 underflow to 0.
If TMR1 is already counting and in single countdown mode, TMR1 will stop immediately; if in re-load mode, TMR1's operation will not be affected.

X0 = 1, clear the halt release request flag (HRF2) set by the detection of a valid change in the input signal in the port IOA, IOC and IOD pins to 0.

Note:
X8 represents the MSB of operand X, X0 represents the LSB of operand X.

Instruction Syntax:

OP code	Operand	Instruction Operation
PLC	X	Clear the last 5 bits of the pre-divider register ← X8 Clear the HRF7 flag ← X7 Clear the HRF6 flag ← X6 Clear the HRF5 flag ← X5 Clear the HRF4 flag ← X4 Clear the HRF3 flag ← X3 Clear the HRF2 flag ← X2 Clear the HRF1 flag ← X1 Clear the HRF0 flag ← X0