



**User Manual
Instruction Set
H08B**

目录

1	简介	4
2	指令列表	5
3	指令说明	8
3.1	MVL	8
3.2	MVF	8
3.3	ADDC.....	9
3.4	ADDF	9
3.5	ADDL	10
3.6	INF	11
3.7	SUBC.....	11
3.8	SUBF	12
3.9	SUBL	13
3.10	DCF	13
3.11	ANDL.....	14
3.12	ANDF	15
3.13	IORL	15
3.14	IORF	16
3.15	XORL.....	17
3.16	XORF	17
3.17	CLRF	18
3.18	COMF	18
3.19	SETF.....	19

3.20 RLF	20
3.21 RLFC	20
3.22 RRF	21
3.23 RRFC	22
3.24 SWPF	23
3.25 INSZ	24
3.26 INSUZ	24
3.27 DCSZ	25
3.28 DCSUZ	26
3.29 CPSE	27
3.30 CPSG	27
3.31 CPSL	28
3.32 TFSZ	28
3.33 JMP	29
3.34 CALL	30
3.35 RET	30
3.36 RETI	31
3.37 RETL	31
3.38 CWDT	32
3.39 IDLE	32
3.40 SLP	33
3.41 NOP	33
3.42 BCF	33
3.43 BSF	34

3.44 BTGF	34
3.45 BTSS	34
3.46 BTSZ.....	35
4 参考文献.....	37
5 修订记录.....	38

1 简介

本文主要介绍 H08B 指令集的应用说明，其中包括 H08B 指令快速索引表格，为了使用者尽快熟悉本文档的内容，需要对这几个别名进行相应的了解。

在本文档里‘w’表示工作寄存器；‘f’表示寄存器（可以包括用户自己定义的一般功能的寄存器或者特殊功能的寄存器）；‘b’表示寄存器的第 b 个位；‘n’表示内存位置或者程序记忆体的位置；‘k’表示 8 位常数；‘d’表示资料存放地方：d = 0 表示存放在 W 工作寄存器，d = 1 表示存放在 f 寄存器；‘a’表示资料存放内存的位置。

H08B 指令集区别于 H08A 指令集，关于它们的不同请参照 H08A 与 H08B 指令集比较表与补充说明。

‘f’寄存器中有专用于搭配指令的特殊功能寄存器，指令说明中也有一些简称，它们的名称与功能如下：

特殊功能寄存器 (SPR)	功能
工作寄存器 WREG	资料搬移，运算，判断
STATUS	进位或借位，半进位，溢位等的判断，旗标会根据判断改变
PSTATUS	芯片进入休眠或待机模式，看门狗计数器溢出等状态的判断，旗标会根据状态改变
程序计数器 PC	指向程序执行地址，包含 PCLATH 与 PCLATL
堆栈的叠顶寄存器 TOS	执行 CALL 指令或发生中断 (INT) 服务时存放程序计数器 PC 地址，子程序或中断返回时又会将地址返回给 PC
堆栈控制器 STKCN	包含堆栈满位、欠位、溢位旗标和堆栈指标 STKPRT
堆栈层寄存器 STKn	当被堆栈指标 STKPRT 指定时，会将寄存器中的资料传送到 TOS
关键词	含义
MSB	最高位
LSB	最低位
MSB4	高 4 位
LSB4	低 4 位

有关特殊功能寄存器的详细介绍请阅读 User's Guide。在 H08B 指令集目录与指令快速索引表中每一条指令都可以通过超链接的方式使读者快速进入指令详解部分。

3 指令说明

3.1 MVL MoVe Literal to w

Syntax: MVL k
Operands: $0 \leq k \leq 255$
Operation: $k \rightarrow W$
Status Affected: None
Description: 将常数 k 搬到 W 累加器中。
Words: 1
Cycles: 1

Example 1: MVL 0FFH

Before Instruction:

WREG(02CH)=000H

After Instruction:

WREG(02CH)=0FFH

3.2 MVF MoVe F to w or MoVe w to F

Syntax: MVF f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$
Operation: $(f) \rightarrow W$, or $(W) \rightarrow f$
Status Affected: None
Description: 将 f 寄存器的值搬到 W 累加器中；或是将 W 累加器的值搬到 f 寄存器中。
若 $d = 0$ ，则表示将 f 寄存器的值搬到 W 累加器中；
若 $d = 1$ ，则表示将 W 累加器的值搬到 f 寄存器中；
若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1
Cycles: 1

Example 1: MVF REG, 0, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=0AAH

REG1(170H)=0FFH

After Instruction:

WREG(02CH)=0AAH

REG(080H)=0AAH

REG1(170H)=0FF

Remark: $d=0$ ，表示将 REG 寄存器的值搬到 W 累加器中。

Example 2: MVF REG1, 1, 0

Before Instruction:

WREG(02CH)=055H

REG1(170H)=0FFH

REG(080H)=0AAH

After Instruction:

WREG(02CH)=055H

REG1(170H)=055H

REG(080H)=0AA

Remark: d=1, 表示将 W 累加器的值搬到 f 寄存器中。

3.3 ADDC ADD w and Carry bit to f

Syntax: ADDC f, d, a

Operands: $0 \leq f \leq 255; d \in (0, 1); a \in (0, 1)$

Operation: $(W) + (f) + (\text{Status}\langle C \rangle) \rightarrow \text{destination}$

Status Affected: C, Z

Description: W 累加器中的值与 f 寄存器中的值与进位旗标 C 三者相加, 并将运算结果放回 d 所指定的寄存器中。

若 d = 0, 则表示将 f 寄存器的值搬到 W 累加器中;

若 d = 1, 则表示将 W 累加器的值搬到 f 寄存器中;

若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;

若 a = 1, 则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ADDC REG, 0, 0

Before Instruction:

WREG=001H

REG(080H)=01FH

C=Z=0

After Instruction:

WREG=020H

REG(080H)=01FH

C=Z=0

Remark: d=0, 则执行结果放回 W 累加器中。

Example 2: ADDC REG, 1, 0

Before Instruction:

WREG=001H

REG(070H)=00EH

C=1, Z=0

After Instruction:

WREG=001H

REG(070H)=010H

C=Z=0

Remark: d=1, 则执行结果放回 f 寄存器中。

3.4 ADDF ADD w to F

Syntax: ADDF f, d, a

Operands: $0 \leq f \leq 255; d \in (0, 1); a \in (0, 1)$

Operation: $(W) + (f) \rightarrow \text{destination}$

Status Affected: C, Z

Description: W 累加器中的值与 f 寄存器中的值相加, 并将运算结果放回 d 所指定的寄存器中。

若 d = 0, 则运算后的结果放到 W 累加器中;

若 d = 1, 则运算后的结果放到 f 寄存器中;

若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;

若 a = 1, 则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ADDF REG, 0, 0

Before Instruction:

WREG=001H

REG(080H)=01FH

C=Z=0

After Instruction:

WREG=020H

REG(080H)=01FH

C=Z=0

Remark: d=0, 则执行结果放回 W 累加器中。

Example 2: ADDF REG, 1

Before Instruction:

WREG=001H

REG(080H)=01FH

C=Z=0

After Instruction:

WREG=001H

REG(080H)=020H

C=Z=0

Remark: d=1, 则执行结果放回 f 寄存器中。a=0 为默认值, 若程序中 a=0 时, 则程序中可以不加入此参数。

3.5 ADDL ADD Literal to w

Syntax: ADDL k

Operands: $0 \leq k \leq 255$

Operation: $(W) + K \rightarrow W$

Status Affected: C, Z

Description: W 累加器中的值与 k 值相加, 并将运算结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: ADDL 00FH

Before Instruction:

WREG=001H

C=Z=0

After Instruction:

WREG=010H

C=Z=0

Example 2: ADDL 00FH

Before Instruction:

WREG=071H

C=Z=0

After Instruction:

WREG=080H

C=Z=0

Example 3: ADDL 00FH

Before Instruction:

WREG=081H

C=Z=0

After Instruction:

WREG=090H

C=OV=Z=0

Example 4: ADDL 00FH

Before Instruction:

After Instruction:

WREG=0F1H

WREG=000H

C=Z=0

C=Z=1

Remark: 执行结果大于 0FFH，进位旗标 C=1。执行结果为 000H，则零位旗标 Z=1。

3.6 INF INcrement F

Syntax: INF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f) + 1 \rightarrow \text{destination}$

Status Affected: C, Z

Description: 将寄存器 f 的值加 1，并将运算结果放回 d 所指定的寄存器中。
 若 d = 0，则运算后的结果放到 W 累加器中；
 若 d = 1，则运算后的结果放到 f 寄存器中；
 若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
 若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: INF REG, 0, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=0FFH

C=Z=0

After Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

C=Z=1

Remark: C 进位，所以 C=1；执行后结果等于 0，所以 Z=1。

Example 2: INF REG, 1, 0

Before Instruction:

WREG(02CH)=055H

REG(070H)=00FH

C=Z=0

After Instruction:

WREG(02CH)=055H

REG(070H)=010H

C=Z=0

Example 3: INF REG, 1, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=07FH

C=Z=0

After Instruction:

WREG(02CH)=055H

REG(080H)=080H

C=Z=0

3.7 SUBC SUBtract w from f with Carry

Syntax: SUBC f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f) - (W) - \overline{(C)} \rightarrow \text{destination}$

Status Affected: C, Z

Description: f 寄存器的值减去 W 累加器与进位旗标 C 的反向值，并将结果放置 d。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 寄存器中；
若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SUBC REG, 0, 0

Before Instruction:

WREG=001H

REG(080H)=001H

C=1, Z=0

After Instruction:

WREG=000H

REG(07FH)=001H

C= Z=1

Remark: C 未借位，所以 C=1，执行结果为 0，所以 Z=1。

Example 2: SUBF REG, 1, 0

Before Instruction:

WREG=000H

REG(07FH)=080H

C=Z=0

After Instruction:

WREG=000H

REG(07FH)=07FH

C=1, Z=0

Remark: C 未借位，所以 C=1。

3.8 SUBF SUBtract w from F

Syntax: SUBF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (f) - (W) → destination

Status Affected: C, Z

Description: 将 f 寄存器的值减掉 W 累加器的值，并将结果放置 d。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 寄存器中；
若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SUBF REG, 0, 0

Before Instruction:

WREG=001H

REG(080H)=001H

C=Z=0

After Instruction:

WREG=000H

REG(080H)=001H

C=Z=1

Remark: C 未借位, 所以 C=1, 执行结果为 0, 所以 Z=1。

Example 2: SUBF REG, 1, 0

Before Instruction:

WREG=001H

REG(07FH)=080H

C=Z=0

After Instruction:

WREG=001H

REG(07FH)=07FH

C=1, Z=0

Remark: C 未借位, 所以 C=1。

3.9 SUBL SUBtract w from Literal

Syntax: SUBL k

Operands: $0 \leq k \leq 255$

Operation: $K - (W) \rightarrow W$

Status Affected: C, Z

Description: 将常数 k 与 W 累加器的值相减并将结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: SUBL 001H

Before Instruction:

WREG=001H

C=Z=0

After Instruction:

WREG=000H

C=Z=1

Remark: C 未借位, 所以 C=1, 执行结果为 0, 所以 Z=1。

Example 2: SUBL 080H

Before Instruction:

WREG=001H

C=Z=0

After Instruction:

WREG=07FH

C=1, Z=0

Remark: C 未借位, 所以 C=1。

Example 3: SUBL 07FH

Before Instruction:

WREG=0FFH

C=Z=0

After Instruction:

WREG=080H

C= Z=0

Remark: C 被借位, 所以 C=0。

Example 4: SUBL 000H

Before Instruction:

WREG=001H

C=Z=0

After Instruction:

WREG=0FFH

C=Z=0

Remark: C 被借位, 所以 C=0。

3.10 DCF DeCrement F

Syntax: DCF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f) - 1 \rightarrow \text{destination}$

Status Affected: C, Z

Description: 将寄存器 f 的值减 1，并将运算结果放回 d 所指定的寄存器中。
若 $d = 0$ ，则运算后的结果放到 W 累加器中；
若 $d = 1$ ，则运算后的结果放到 f 寄存器中；
若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: DCFREG, 0, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=0FFH

C=Z=0

After Instruction:

WREG(02CH)=0FEH

REG(080H)=0FFH

C=1, Z=0

Remark: C 未被借位，所以 C=1。

Example 2: DCFREG, 1, 0

Before Instruction:

WREG(02CH)=055H

REG(070H)=000H

C=Z=0

After Instruction:

WREG(02CH)=055H

REG(070H)=0FFH

C=Z=0

Remark: C 皆被借位，所以 C=0。

Example 3: DCFREG, 1, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=080H

C=Z=0

After Instruction:

WREG(02CH)=055H

REG(080H)=07FH

C=1, Z=0

Remark: C 未被借位，所以 C=1。

3.11 ANDL AND Literal with w

Syntax: ANDL k

Operands: $0 \leq k \leq 255$

Operation: $(W) \text{ AND } k \rightarrow W$

Status Affected: Z

Description: W 累加器中的值与 k 值做逻辑的 AND 运算，并将运算结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: ANDL 0A0H

Before Instruction:

WREG=055H

Z=0

After Instruction:

WREG=000H

Z=1

Remark: 执行结果为 000H，零位旗标 Z=1。

Example 2: ANDL 0FF0H

Before Instruction:

WREG=080H

Z=0

After Instruction:

WREG=080H

Z=0

3.12 ANDF AND w with F

Syntax: ANDF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (W) AND (f) \rightarrow destination

Status Affected: Z

Description: W 累加器中的值与 f 寄存器中的值做逻辑 AND 运算，并将运算结果放回 d 所指定的寄存器中。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 寄存器中；

若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ANDF REG, 0

Before Instruction:

WREG=055H

REG(080H)=0AAH

C=Z=0

After Instruction:

WREG=000H

REG(080H)=0AAH

Z=1, C=0

Remark: 执行结果为 000H，零位旗标 Z=1。a=0 为默认值，若程序中 a=0 时，则程序中可以不加入此参数。

Example 2: ANDF REG, 1, 0

Before Instruction:

WREG=080H

REG(070H)=0FFH

Z=0

After Instruction:

WREG=080H

REG(070H)=080H

Z=0

3.13 IORL Inclusive OR Literal with w

Syntax: IORL k
Operands: $0 \leq k \leq 255$
Operation: (W) OR k \rightarrow W
Status Affected: Z

Description: W 累加器中的值与 k 值作逻辑的 OR 运算，并将运算结果放回 W 累加器中。

Words: 1
Cycles: 1

Example 1: IORL 055H

Before Instruction:

WREG(02CH)=0AAH

Z=0

After Instruction:

WREG(02CH)=0FFH

Z=0

Example 2: IORL 000H

Before Instruction:

WREG(02CH)=000H

Z=0

After Instruction:

WREG(02CH)=000H

Z=1

Remark: 执行结果等于 0，所以 Z=1。

3.14 IORF Inclusive OR w with F

Syntax: IORF f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$
Operation: (W) OR (f) \rightarrow destination
Status Affected: Z

Description: W 累加器中的值与 f 寄存器中的值作逻辑的 OR 运算，并将运算结果放回 d 所指定的寄存器中。

若 $d = 0$ ，则运算后的结果放到 W 累加器中；

若 $d = 1$ ，则运算后的结果放到 f 寄存器中；

若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1
Cycles: 1

Example 1: IORF REG, 0, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=0AAH

Z=0

After Instruction:

WREG(02CH)=0FFH

REG(080H)=0AAH

Z=0

Example 2: IORF REG, 1, 0

Before Instruction:

After Instruction:

WREG(02CH)=00FH

WREG(02CH)=00FH

REG(070H)=0F0H

REG(070H)=0FFH

Z=0

Z=0

3.15 XORL Exclusive OR Literal with w

Syntax: XORL k

Operands: $0 \leq f \leq 255$

Operation: (W) XOR k \rightarrow W

Status Affected: Z

Description: 将常数 k 与 W 累加器的值做逻辑互斥或(XOR)的运算，并将结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: XORL 055H

Before Instruction:

WREG(02CH)=0AAH

Z=0

After Instruction:

WREG(02CH)=0FFH

Z=0

Remark: XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 2: XORL 0FFH

Before Instruction:

WREG(02CH)=0FFH

Z=0

After Instruction:

WREG(02CH)=000H

Z=1

Remark: 执行结果等于 0，所以 Z=1。XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 3: XORL 000H

Before Instruction:

WREG(02CH)=000H

Z=0

After Instruction:

WREG(02CH)=000H

Z=1

Remark: 执行结果等于 0，所以 Z=1。XOR: 两者同则结果为 0；两者不同则结果为 1。

3.16 XORF Exclusive OR w with F

Syntax: XORF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (W) XOR (f) \rightarrow destination

Status Affected: Z

Description: 将常数 k 与 W 累加器的值做逻辑互斥或(XOR)运算，并将结果放回 d 中。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 寄存器中；

若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: XORF REG, 0, 0

Before Instruction:

WREG(02CH)=0AAH

REG(080H)=055H

Z=0

After Instruction:

WREG(02CH)=0FFH

REG(080H)=055H

Z=0

Remark: XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 2: XORF REG, 1, 0

Before Instruction:

WREG(02CH)=0FFH

REG(070H)=0FFH

Z=0

After Instruction:

WREG(02CH)=0FFH

REG(070H)=000H

Z=1

Remark: 执行结果等于 0，所以 Z=1。XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 3: XORF REG, 0, 0

Before Instruction:

WREG(02CH)=000H

REG(080H)=000H

Z=0

After Instruction:

WREG(02CH)=000H

REG(080H)=000H

Z=1

Remark: 执行结果等于 0，所以 Z=1。XOR: 两者同则结果为 0；两者不同则结果为 1。

3.17 CLRF CLear F

Syntax: CLRF f, a

Operands: $0 \leq f \leq 255$; $a \in (0, 1)$

Operation: 000H \rightarrow f

Status Affected: None

Description: 将寄存器 f 的值全部清除为 0。

若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: CLRF REG, 0

Before Instruction:

REG(080H)=055H

After Instruction:

REG(080H)=000H

Remark: REG 寄存器中的数值被清为 0。

3.18 COMF COMplement F

Syntax: COMF f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$
Operation: $\overline{(f)} \rightarrow \text{destination}$

Status Affected: Z

Description: 将寄存器中的值取补码，并将运算结果放回 d 所指定的寄存器中。
若 $d = 0$ ，则运算后的结果放到 W 累加器中；
若 $d = 1$ ，则运算后的结果放到 f 寄存器中；
若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: COMF REG, 0, 0

Before Instruction:

WREG(02CH)=055H

REG(080H)=0FFH

Z=0

After Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

Z=1

Example 2: COMF REG, 1, 0

Before Instruction:

WREG(02CH)=055H

REG(070H)=055H

Z=0

After Instruction:

WREG(02CH)=055H

REG(070H)=0AAH

Z=0

3.19 SETF SET F

Syntax: SETF f, a
Operands: $0 \leq f \leq 255$; $a \in (0, 1)$
Operation: 0FFH \rightarrow f

Status Affected: None

Description: 将 f 寄存器内的值全部设定为 1。
若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SETF REG, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

After Instruction:

WREG(02CH)=00FH

REG(080H)=0FFH

3.20 RLF Rotate Left F (no carry)

Syntax: RLF f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$
Operation: (f<n>) → destination <n+1>,
 (f<7>) → destination <0>。

Status Affected: Z

Description: 将 f 寄存器内的值向左旋转。
 若 d = 0, 则运算后的结果放到 W 累加器中;
 若 d = 1, 则运算后的结果放到 f 寄存器中;
 若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;
 若 a = 1, 则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。



Words: 1
Cycles: 1
Example 1: RLF REG, 1, 0

Before Instruction:	After Instruction:
WREG(02CH)=00FH	WREG(02CH)=00FH
REG(080H)=0AAH	REG(080H)=055H
Z=0	Z=0

Example 2: RLF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=00FH	WREG(02CH)=000H
REG(07FH)=000H	REG(07FH)=000H
Z=0	Z=1

Example 3: RLF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=00FH	WREG(02CH)=0AAH
REG(080H)=055H	REG(080H)=055H
Z=0	Z=0

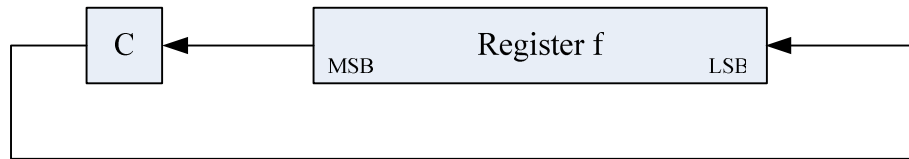
3.21 RLFC Rotate Left F through Carry

Syntax: RLFC f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (f<n>) → destination <n+1 >,
 (f<7>) → Status< C > ,
 Status< C > → destination < 0 >。

Status Affected: C, Z

Description: 将 f 寄存器内的值与进位旗标 C 一起向左旋转。
 若 d = 0, 则运算后的结果放到 W 累加器中;
 若 d = 1, 则运算后的结果放到 f 寄存器中;
 若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;
 若 a = 1, 则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RLFC REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

C=Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=054H

C=1, Z=0

Example 2: RLFC REG, 0, 0

Before Instruction:

WREG(02CH)=0FH

REG(070H)=0EAH

C=Z=0

After Instruction:

WREG(02CH)=0D4H

REG(070H)=0EAH

C=1, Z=0

Example 3: RLFC REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(070H)=080H

C=Z=0

After Instruction:

WREG(02CH)=00FH

REG(070H)=000H

C=Z=1

3.22 RRF Rotate Right F (no carry)

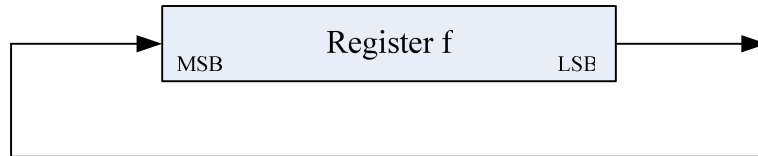
Syntax: RRF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (f<n>) → destination <n - 1 > ,
 (f<0>) → destination < 7 > 。

Status Affected: Z

Description: 将 f 寄存器内的值向右旋转。
若 d = 0, 则运算后的结果放到 W 累加器中;
若 d = 1, 则运算后的结果放到 f 寄存器中;
若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;
若 a = 1, 则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RRF REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=055H

Z=0

Example 2: RRF REG, 0, 0

Before Instruction:

WREG(02CH)=00FH

REG(07FH)=000H

Z=0

After Instruction:

WREG(02CH)=000H

REG(07FH)=000H

Z=1

Example 3: RRF REG, 0, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=055H

Z=0

After Instruction:

WREG(02CH)=0AAH

REG(080H)=055H

Z=0

3.23 RRFC Rotate Right F through Carry

Syntax: RRFC f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (f<n>) → destination <n-1 >,
(f<0>) → Status< C >,
Status< C > → destination < 7 >。

Status Affected: C, Z

Description: 将 f 寄存器内的值与进位旗标 C 一起向右旋转。
若 d = 0, 则运算后的结果放到 W 累加器中;
若 d = 1, 则运算后的结果放到 f 寄存器中;
若 a = 0, 则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中;

若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RRFC REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

C=Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=055H

C=Z=0

Example 2: RRFC REG, 0, 0

Before Instruction:

WREG(02CH)=00FH

REG(07FH)=055H

C=1, Z=0

After Instruction:

WREG(02CH)=0AAH

REG(07FH)=055H

C=1, Z=0

Example 3: RRFC REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(07FH)=001H

C=Z=0

After Instruction:

WREG(02CH)=00FH

REG(07FH)=000H

C=Z=1

3.24 SWPF SWAP F

Syntax: SWPF f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f<3:0>) \rightarrow \text{destination}<7:4>$,
 $(f<7:4>) \rightarrow \text{destination}<3:0>$.

Status Affected: None

Description: 将 f 寄存器内的高 4 位值与低 4 位值做交换。
 若 $d = 0$ ，则运算后的结果放到 W 累加器中；
 若 $d = 1$ ，则运算后的结果放到 f 寄存器中；
 若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；
 若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SWPF REG, 1, 0

Before Instruction:

After Instruction:

WREG=001H
REG(080H)=05AH

WREG=001H
REG(080H)=0A5H

3.25 INSZ INcrement f, Skip if Zero

Syntax: INSZ f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f) + 1 \rightarrow \text{destination}$, skip if destination=0

Status Affected: None

Description: 将寄存器的值加 1 后与 0 作比较，若是寄存器的值等于 0 则跳过下一个指令，若不等于 0 则往下执行，并将运算结果放回 d 所指定的寄存器中。

若 $d = 0$ ，则运算后的结果放到 W 累加器中；

若 $d = 1$ ，则运算后的结果放到 f 寄存器中；

若 $a = 0$ ，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 $a = 1$ ，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1: INSZ REG, 0, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0FFH

After Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

Remark: 执行结果为 0，所以跳过下一个指令。

Example 2: INSZ REG, 1, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=055H

REG(070H)=000H

After Instruction:

WREG(02CH)=00AH

REG(070H)=001H

Remark: 执行结果不为 0，所以继续往下执行程序段，执行结果被放回 REG 寄存器。

3.26 INSUZ INcrement f, Skip if Un-Zero

Syntax: INSUZ f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: $(f) + 1 \rightarrow \text{destination}$, skip if destination $\neq 0$

Status Affected: None

Description: 将寄存器的值加 1 后与 0 作比较，若是寄存器的值不等于 0 则跳过下一个指令，若等于 0

则往下执行，并将运算结果放回 d 所指定的寄存器中。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 寄存器中；

若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1
Cycles: 1(2)(3)
Example 1: INSUZ REG, 1, 0
 MVL 00AH
 NOP

Before Instruction:	After Instruction:
WREG(02CH)=00FH	WREG(02CH)=00AH
REG(080H)=0FFH	REG(080H)=000H

Remark: 执行结果为 0，所以继续往下执行程序段，执行结果被放回 REG 寄存器。

Example 2: INSUZ REG, 0, 0
 MVL 00AH
 NOP

Before Instruction:	After Instruction:
WREG(02CH)=055H	WREG(02CH)=001H
REG(070H)=000H	REG(070H)=000H

Remark: 执行结果不为 0，所以跳过下一个指令，执行结果被放回 W 累加器。

3.27 DCSZ DeCrement f, Skip if Zero

Syntax: DCSZ f, d, a
Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$
Operation: (f) - 1 → destination, skip if destination=0
Status Affected: None

Description: 将寄存器的值减 1 后与 0 作比较，若是寄存器的值等于 0 则跳过下一个指令，若不等于 0 则往下执行，并将运算结果放回 d 所指定的寄存器中。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 寄存器中；

若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1
Cycles: 1(2)(3)
Example 1: DCSZ REG, 0, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=00FH

REG(080H)=001H

After Instruction:

WREG(02CH)=000H

REG(080H)=001H

Remark: 执行结果为 0，所以跳过下一个指令。

Example 2: DCSZ REG, 1, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=055H

REG(070H)=000H

After Instruction:

WREG(02CH)=00AH

REG(070H)=0FFH

Remark: 执行结果不为 0，所以继续往下执行程序段，执行结果被放回 REG 寄存器。

3.28 DCSUZ DeCrement f, Skip if Un-Zero

Syntax: DCSUZ f, d, a

Operands: $0 \leq f \leq 255$; $d \in (0, 1)$; $a \in (0, 1)$

Operation: (f) - 1 → destination, skip if destination ≠ 0

Status Affected: None

Description: 将寄存器的值减 1 后与 0 作比较，若是寄存器的值不等于 0 则跳过下一个指令，若等于 0 则往下执行，并将运算结果放回 d 所指定的寄存器中。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 寄存器中；

若 a = 0，则表示 f 寄存器存在于 000H 到 0FFH 所指定的 RAM 地址中；

若 a = 1，则表示 f 寄存器存在于 100H 到 1FFH 所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1: DCSUZ REG, 1, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=00FH

REG(080H)=001H

After Instruction:

WREG(02CH)=00AH

REG(080H)=000H

Remark: 执行结果为 0，所以继续往下执行程序段，执行结果被放回 REG 寄存器。

Example 2: DCSUZ REG, 0, 0
 MVL 00AH
 NOP

Before Instruction:

WREG(02CH)=055H

REG(070H)=000H

After Instruction:

WREG(02CH)=0FFH

REG(070H)=000H

Remark: 执行结果不为 0，所以跳过下一个指令，执行结果被放回 W 累加器。

3.29 CPSE ComPare f with w, Skip if f Equal w

Syntax: CPSE f, a

Operands: $0 \leq f \leq 255$; $a \in (0,1)$

Operation: skip if (f) = (W)

Status Affected: None

Description: 将寄存器的值与 W 累加器的值作比较，若是两个值相等则跳过下一个指令，若大于或是小于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:
CPSE REG, 0
MVL 001H
NOP

Before Instruction:

WREG(02CH)=000H

REG(080H)=000H

After Instruction:

WREG(02CH)=000H

REG(080H)=000H

Remark: f 寄存器和 W 累加器数值相等，所以跳过下一个指令。

Example 2:
CPSE REG, 0
MVL 001H
NOP

Before Instruction:

WREG(02CH)=000H

REG(070H)=07FH

After Instruction:

WREG(02CH)=001H

REG(070H)=07FH

Remark: f 寄存器和 W 累加器数值不相等，所以程序继续往下执行。

3.30 CPSG ComPare f with w, Skip if f Greater than w

Syntax: CPSG f, a

Operands: $0 \leq f \leq 255$; $a \in (0,1)$

Operation: skip if (f) > (W)

Status Affected: None

Description: 将寄存器的值与 W 累加器的值作比较，若是寄存器的值大于 W 累加器的值则跳过下一个指令，若小于或是等于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:
CPSG REG, 0
MVL 00FH

NOP

Before Instruction:

WREG(02CH)=005H

REG(080H)=006H

After Instruction:

WREG(02CH)=005H

REG(080H)=006H

Remark: f 寄存器的数值大于 W 累加器数值，所以跳过下一个指令。

Example 2:
CPSG REG, 0
MVL 00FH
NOP

Before Instruction:

WREG(02CH)=005H

REG(070H)=005H

After Instruction:

WREG(02CH)=00FH

REG(070H)=005H

Remark: f 寄存器数值等于 W 累加器数值，所以程序继续往下执行。

3.31 CPSL ComPare f with w, Skip if f Less than w

Syntax: CPSL f, a

Operands: $0 \leq f \leq 255$; $a \in (0,1)$

Operation: skip if $(f) < (W)$

Status Affected: None

Description: 将寄存器的值与 W 累加器的值作比较，若是寄存器的值小于 W 累加器的值则跳过下一个指令，若大于或是等于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:
CPSL REG, 0
MVL 00FH
NOP

Before Instruction:

WREG(02CH)=005H

REG(080H)=004H

After Instruction:

WREG(02CH)=005H

REG(080H)=004H

Remark: f 寄存器的数值小于 W 累加器数值，所以跳过下一个指令。

Example 2:
CPSL REG, 0
MVL 00FH
NOP

Before Instruction:

WREG(02CH)=005H

REG(070H)=005H

After Instruction:

WREG(02CH)=00FH

REG(070H)=005H

Remark: f 寄存器数值等于 W 累加器数值，所以程序继续往下执行。

3.32 TFSZ Test F, Skip if Zero

Syntax: TFSZ f, a

Operands: $0 \leq f \leq 255$; $a \in (0,1)$

Operation: skip if $f = 0$

Status Affected: None

Description: 假如 f 寄存器内的值等于 0 则跳过下一个指令；若 f 寄存器内的值不等于 0，则执行下一个指令。

Words: 1

Cycles: 1(2)(3)

Example 1:
TFSZ REG, 0
MVL 00FH
NOP

Before Instruction:

WREG(02CH)=005H

REG(080H)=000H

After Instruction:

WREG(02CH)=005H

REG(080H)=000H

Remark: f 寄存器的数值等于 0，所以跳过下一个指令。

Example 2:
TFSZ REG, 0
MVL 00FH
NOP

Before Instruction:

WREG(02CH)=005H

REG(070H)=001H

After Instruction:

WREG(02CH)=00FH

REG(070H)=001H

Remark: f 寄存器数值不等于 0，所以程序继续往下执行。

3.33 JMP Unconditional JUMP

Syntax: JMP n

Operands: $0 \leq n \leq 2047(07FFH)$

Operation: $n \rightarrow PC$

Status Affected: None

Description: 无条件跳跃至指定的地址 n。

Words: 2

Cycles: 2

Example 1:
LABEL: JMP NEXT
.
.
.
NEXT: NOP

Before Instruction:

After Instruction:

PC = address (LABEL)

PC = address (NEXT)

3.34 CALL Subroutine CALL

Syntax: CALL n

Operands: $0 \leq n \leq 2047(07FFH)$

Operation: (PC) + 1 → TOS, n → PC

Status Affected: STKPTR<STKFL>, STKPTR<STKOV>, Pstatus<SKERR>.

Description: 呼叫子程序，呼叫的范围最大到 2Kbytes 的内存范围。
若呼叫子程序之后堆栈层为该产品最后一层堆栈，则 STKFL 旗标会被设定为 1。
在 SBMSET1<7>=0 的条件下，堆栈层满之后再进行 CALL 指令则 STKOV 旗标会被设定为 1。
SKERR 也会被设定为 1。PC 正常执行。
在 SBMSET1<7>=1 的条件下，堆栈层满之后再进行 CALL 指令则 STKOV 旗标会被设定为 1。
SKERR 也会被设定为 1，芯片重置，PC 回到 000H。
STKFL 或 STKOV 发生时，只要其中一个旗标被清除时，两者同时都会被清除。
TOS 无法被使用者读取使用。

Words: 2

Cycles: 2

Example 1: LABEL: CALL NEXT

.
. .
. .

NEXT: NOP

Before Instruction:

PC = address (LABEL)

After Instruction:

PC= address (NEXT)

TOS= address (LABEL + 2)

3.35 RET RETurn from subroutine

Syntax: RET

Operands: None

Operation: (TOS) → PC

Status Affected: STKPTR<STKUN>, Pstatus<SKERR>

Description: 离开子程序，并将推送指针寄存器的数值存到 PC 中。
当未呼叫子程序且 STKPTR=000H 时，执行 RET 指令时会造成芯片重置，STKUN 旗标会被设定为 1，SKERR 旗标会被设定为 1。
TOS 无法被使用者读取。

Words: 1

Cycles: 2

Example 1: RET

Before Instruction:

None

After Instruction:

PC=TOS

3.36 RETI RETurn from Interrupt

Syntax: RETI

Operands: None

Operation: (TOS) → PC, 1 → GIE

Status Affected: GIE, STKPTR<STKUN>, Pstatus<SKERR>

Description: 离开中断程序子程序，并将推送指针寄存器的值存到 PC 中，中断致能接脚再度被设定为 1。

当未呼叫子程序且 STKPTR=000H 时，执行 RETI 指令时会造成芯片重置，STKUN 旗标会被设定为 1，SKERR 旗标会被设定为 1。

TOS 无法被使用者读取。

Words: 1

Cycles: 2

Example 1: RETI

Before Instruction:

None

After Instruction:

PC=TOS

GIE=1

3.37 RETL RETurn Literal to w

Syntax: RETL k

Operands: $0 \leq k \leq 255$

Operation: k → W, (TOS) → PC

Status Affected: STKPTR<STKUN>, Pstatus<SKERR>

Description: 从子程序返回主程序的指令，但本指令在返回的时候，还会顺便将常数 k 载入到 W 累加器中。

此指令多半在查表法时会用到。

当未呼叫子程序且 STKPTR=000H 时，执行 RETL 指令时会造成芯片重置，STKUN 旗标会被设定为 1，SKERR 旗标会被设定为 1。

Words: 1

Cycles: 2

Example 1: LABEL: MVL 001H
CALL TABLE

```
TABLE:  ADDF  PCLATL, 1, 0
RETL    055H
RETL    0AAH
```

Before Instruction:

WREG(02CH)=001H

After Instruction:

WREG(02CH)=0AAH

Remark: 返回主程序时，顺便将常数 k 加载到 W 累加器中。此范例为藉由写入 PCLATL 的 Offset 数值来决定取得 TABLE 表中第几笔数值。

3.38 CWDT Clear WatchDog Timer

Syntax: CWDT

Operands: None

Operation: 000H → Watch dog counter

Status Affected: TO

Description: 将看门狗定时器的值全部清除为 0。

Words: 1

Cycles: 1

Example 1: CWDT

Before Instruction:

WDT counter = ???

After Instruction:

WDT counter = 000H

Pstatus<TO> = 0

3.39 IDLE IDLE mode

Syntax: IDLE

Operands: None

Operation: CPU Halt

Status Affected: Pstatus<IdleB>

Description: CPU 进入暂停模式，程序空指令执行动作。
IDLE 指令后，建议加上 NOP 指令。

Words: 1

Cycles: 1

Example 1: IDLE
NOP

Before Instruction:

Pstatus<IdleB>=0

After Instruction:

Pstatus<IdleB>=1

IDLE

NOP.....program break here

3.40 SLP Enter SLeep mode

Syntax: SLP
Operands: None
Operation: 1 → PD
Status Affected: PD
Description: CPU 进入睡眠状态，震荡器停止动作。
Words: 1
Cycles: 1
Example 1: SLP
 NOP

Before Instruction:

PD=0

After Instruction:

PD=1

3.41 NOP No Operation

Syntax: NOP
Operands: None
Operation: No operation
Status Affected: None
Description: 不做任何运算，只延迟一个指令时间。
Words: 1
Cycles: 1
Example 1: NOP

Remark: 空指令，只做一个指令周期的延迟时间。

3.42 BCF Bit Clear F

Syntax: BCF f, b, a
Operands: $0 \leq f \leq 255$; $0 \leq b \leq 7$; $a \in (0,1)$
Operation: $0 \rightarrow f < b >$
Status Affected: None
Description: 将 f 寄存器中的设定的位清除为 0。
Words: 1
Cycles: 1
Example 1: BCF REG,2

Before Instruction:

After Instruction:

REG(080H)=1111 1111B

REG(080H)=1111 1011B

Remark: 将 REG 寄存器中 BIT2 清除为 0，其它位值不变。

3.43 BSF Bit Set F

Syntax: BSF f, b, a

Operands: $0 \leq f \leq 255$; $0 \leq b \leq 7$; $a \in (0,1)$

Operation: $1 \rightarrow f < b >$

Status Affected: None

Description: 将 f 寄存器中的设定的位设定为 1。

Words: 1

Cycles: 1

Example 1: BSF REG,2

Before Instruction:

REG(080H)=00000000B

After Instruction:

REG(080H)=00000100B

Remark: 将 REG 寄存器中 BIT2 设定为 1，其它位值不变。

3.44 BTGF Bit ToGgle F

Syntax: BTGF f, b, a

Operands: $0 \leq f \leq 255$; $0 \leq b \leq 7$; $a \in (0,1)$

Operation: $\overline{(f < b >)} \rightarrow f < b >$

Status Affected: None

Description: 将寄存器中的某一个位取补码。

Words: 1

Cycles: 1

Example 1: BTGF REG, 7, 0

Before Instruction:

REG(080H)=0111 1111B

After Instruction:

REG(080H)=1111 1111B

Remark: 针对 REG 寄存器中 BIT7 取补码。

3.45 BTSS Bit Test and Skip if Set

Syntax: BTSS f, b, a

Operands: $0 \leq f \leq 255$; $0 \leq b \leq 7$; $a \in (0,1)$

Operation: skip if $(f < b >)=1$

Status Affected: None

Description: 比较寄存器中的某一个位是否为 1。若为 1 则跳过下一个指令，若不为 1 则往下执行。

Words: 1
Cycles: 1(2)(3)
Example 1: BTSS REG, 7, 0
MVL 001H
NOP

Before Instruction: WREG(02CH)=000H
REG(080H)=0FFH
After Instruction: WREG(02CH)=000H
REG(080H)=0FFH

Remark: REG 寄存器中 BIT7 为 1，所以跳过下一个指令。

Example 2: BTSS REG, 7, 0
MVL 001H
NOP

Before Instruction: WREG(02CH)=000H
REG(080H)=07FH
After Instruction: WREG(02CH)=001H
REG(080H)=07FH

Remark: REG 寄存器中 BIT7 为 0，所以程序直接往下执行。

3.46 BTSZ Bit Test and Skip if Zero

Syntax: BTSZ f, b, a
Operands: $0 \leq f \leq 255$; $0 \leq b \leq 7$; $a \in (0,1)$
Operation: skip if $(f < b) = 0$
Status Affected: None

Description: 比较寄存器中的某一个位是否为 0。若为 0 则跳过下一个指令，若不为 0 则往下执行。

Words: 1
Cycles: 1(2)(3)
Example 1: BTSZ REG, 7, 0
MVL 001H
NOP

Before Instruction: WREG(02CH)=000H
REG(080H)=0FFH
After Instruction: WREG(02CH)=001H
REG(080H)=0FFH

Remark: REG 寄存器中 BIT7 不为 0，所以程序直接往下执行。

Example 2: BTSZ REG, 7, 0
MVL 001H
NOP

Before Instruction: WREG(02CH)=000H
REG(080H)=07FH
After Instruction: WREG(02CH)=000H
REG(080H)=07FH

Remark: REG 寄存器中 BIT7 为 0，所以跳过下一个指令。

4 参考文献

- a. APD-CORE001-Vxx **Compare Instruction Set H08A and H08B**
- b. APD-CORE003-Vxx **Instructions set H08B**
- c. UG-HY11S14-Vxx **HY11Pxx Family User's Guide**

5 修订记录

以下描述本文件差异较大的地方，而标点符号与字形的改变不在此描述范围。

日期	页次	变更摘要
2008/12/24	Page4	将 RET 与 RETI 指令后的“s”去掉。
2008/12/24	Page5	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/24	Page6	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/24	Page8	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。Example 中涉及旗标 C 的描述全部删除，只留下关于对旗标 Z 的影响的结果。
2008/12/24	Page9	Example 中涉及旗标 C 的描述全部删除，只留下关于对旗标 Z 的影响的结果。
2008/12/24	Page10	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/24	Page11	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page12	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page13	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page14	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page16	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page17	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page18	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page19	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page20	将 $a \in (0)$ 改为 $a \in (0, 1)$ 。
2008/12/25	Page21	Examp 中加入指令执行后的描述: WDT counter = 000H 和 Pstatus<TO> = 0
2008/12/25	Page22	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/25	Page24	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page25	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page27	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page28	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page29	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page30	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page33	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page39	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page40	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page41	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page42	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page43	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page45	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/26	Page46	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/27	Page48	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。
2008/12/27	Page49	将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。

2008/12/27 Page50 将 $a \in (0)$ 改为 $a \in (0, 1)$ ，加入 $a \in (1)$ 情况的描述。

2008/12/27 Page51 将 Example 2 中，After Instruction:N=0,改为 Z=1。

2009/1/6 All 文件编号由：APD-CORE003-V01 改为：APD-CORE003-V02_SC

为了便于读者的理解，将指令集做了重新排序，修订记录中不包括此记录，即修订记录中的 Page 全部基于以前版本。