Totally Logical

CHAPTER 3
INSTRUCTION SET

FUNCTIONAL SUMMARY

Z8^{PLUS} instructions can be divided into the following eight functional groups:

- Load
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Block Transfer
- Rotate and Shift
- CPU Control

Table 3-1 through Table 3-8 show the instructions belonging to each group and the number of operands required for each. The source operand is `src`, the destination operand is `dst`, and a condition code is `cc`.

When instructions are executed, registers defined as sources are read only. All General-Purpose Registers function as:

- accumulators
- address pointers
- index registers
- stack areas
- scratch pad memory

Table 3-1. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant
POP	dst	Pop
PUSH	src	Push

Table 3-2. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
SBC	dst, src	Subtract with Carry
SUB	dst, src	Subtract

Table 3-3. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical Exclusive OR

Table 3-4. Program Control Instructions

Mnemonic	Operands	Instruction
CALL	dst	Call Procedure
DJNZ	dst, src	Decrement and Jump Non-Zero
IRET		Interrupt Return
JP	cc, dst	Jump
JR	cc, dst	Jump Relative
RET		Return

Table 3-5. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
TCM	dst, src	Test Complement Under Mask
TM	dst, src	Test Under Mask
AND	dst, src	Bit Clear
OR	dst, src	Bit Set
XOR	dst, src	Bit Complement

Table 3-6. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant Auto Increment

Table 3-7. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
RL	dst	Rotate Left
RLC	dst	Rotate Left Through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right Through Carry
SRA	dst	Shift Right Arithmetic
SWAP	dst	Swap Nibbles

Table 3-8. CPU Control Instructions

Mnemonic	Operands	Instruction
CCF		Complement Carry Flag
DI		Disable Interrupts
EI		Enable Interrupts
HALT		Halt
NOP		No Operation
RCF		Reset Carry Flag
SCF		Set Carry Flag
SRP	src	Set Register Pointer
STOP		Stop
WDT		Refresh WDT

PROCESSOR FLAGS

The Flag Register (FCH) informs the user of the processor'sbcurrent status. The flags and their bit positions in the Flag Register are shown in Figure 3-1.

The Flag Register contains eight bits of status information which are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested for use with conditional Jump instructions. Two flags (H and D) are used for BCD arithmetic. The two remaining bits in the Flag Register are the watch-dog timer reset flag and the stop mode recovery flag. Both of these flag bits may be tested and must be explicitly cleared by software.

As with bits in the other control registers, the Flag Register bits can be set or reset by instructions; however, only those instructions that do not affect the flags as an outcome of the execution should be assigned a value.

Figure 3-1. Flag Register

Flag Register (FCH: Read/Write) R252 Flags

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	U	U	U	U	U	U	*	*
R = Read W = Write X = Indeterminate U = Unchanged								

Bit/Field	Bit Position	R/W	Value	Description
Carry Flag (C)	7	R/W		The Carry Flag is set to 1 whenever the result of an arithmetic operation generates a carry out of or a borrow into the high order bit 7. Otherwise, the Carry Flag is cleared to 0. Following Rotate and Shift instructions, the Carry Flag contains the last value shifted out of the specified register. An instruction can set (I), reset(O), or complement the Carry Flag. The carry flag is not effected by RESET.
Zero Flag (Z)	6	R/W		For arithmetic and logical operations, the Zero Flag is set to 1 if the result is 0. Otherwise, the Zero Flag is cleared to 0. If the result of testing bits in a register is 00H, the Zero Flag is set to 1. Otherwise the Zero Flag is cleared to 0. If the result of a Rotate or Shift operation is 00H, the Zero Flag is set to 1. The Zero Flag is not effected by a RESET command.

Sign Flag (S)	5	R/W	The Sign Flag stores the value of the most significant bit of a result following an arithmetic, logical, rotate, or shift operation. When performing arithmetic operations on signed numbers, binary two's-complement notation is used to represent and process information. A positive number is identified by a 0 in the most significant bit position (bit 7); therefore, the Sign Flag is also 0. A negative number is identified by a 1 in the most significant bit position (bit 7); therefore, the Sign Flag is also 1. The Sign Flag is not effected by RESET.
Overflow (V)	4	R/W	For signed arithmetic, rotate, and shift operations, the Overflow Flag is set to 1 when the result is greater than the maximum possible number (>127) or less than the minimum possible number (<–128) that can be represented in two's-complement form . The Overflow Flag is cleared to 0 if no overflow occurs. Following logical operations the Overflow Flag is cleared to 0. The Overflow Flag is not effected by RESET.
Decimal Adjust Flag (D)	3	R/W	The Decimal Adjust Flag is used for BCD arithmetic. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag specifies what type of instruction was last executed so that the subsequent Decimal Adjust (DA) operation can function properly. Normally, the Decimal Adjust Flag cannot be used as a test condition. After a subtraction, the Decimal Adjust Flag is set to 1. Following an addition it is cleared to 0. The Decimal Adjust Flag is not effected by RESET.
Half-Carry Flag (H)	2	R/W	The Half Carry Flag is set to 1 whenever an addition generates a carry out of bit 3 (Overflow) or a subtraction generates a "borrow into" bit 3. The Half Carry Flag is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. As in the case of the Decimal Adjust Flag, the user does not normally access this flag. The Half Carry flag is not effected by RESET.

Watch-Dog Timer (WDT)	1	R/W	The Watch-Dog Timer reset flag is set by a watchdog timer timeout. This permits software to determine if a timeout of the watchdog timer has occurred. The WDT flag is cleared by the $\overline{\text{RESET}}$ pin. The WDT and SMR flags are the only flags effected by RESET. This behavior permits software to determine if a RESET occurred, if a WDT timeout occurred, or if a return from STOP mode occurred. Software must explicitly clear this flag after detecting the timeout condition. Failure to clear this flag may result in undefined behavior.
Stop Mode Recovery Flag (SMR)	0	R/W	The Stop Mode Recovery (SMR) flag is set upon the execution of a STOP instruction. This permits software to determine if a return from stop mode has occurred upon returning to active status. The SMR flag is cleared by the $\overline{\text{RESET}}$ pin. The WDT and SMR flags are the only flags effected by RESET. This behavior permits software to determine if a RESET occurred, if a WDT timeout occurred, or if a return from STOP mode occurred. Software must explicitly clear this flag after detecting the SMR condition. Failure to clear this flag may result in undefined behavior.

CONDITION CODES

The C, Z, S, and V Flags control the operation of the conditional JUMP instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 4-7 of the conditional instructions.

Flag Definitions, Flag Settings and Condition Codes are summarized in Table 3-9, Table 3-10, and Table 3-11.

Table 3-9. Flag Definitions

Flag	Description
C	Carry Flag
Z	Zero Flag
S	Sign Flag
V	Overflow Flag

Table 3-10. Flag Settings Definitions

Symbol	Definition
0	Cleared to 0
1	Set to 1
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Table 3-11. Condition Codes

Binary	HEX	Mnemonic	Definition	Flag Settings
0000	0	F	Always False	–
1000	8	(blank)	Always True	–
0111	7	C	Carry	C = 1
1111	F	NC	No Carry	C = 0
0110	6	Z	Zero	Z = 1
1110	E	NZ	Non-Zero	Z = 0
1101	D	PL	Plus	S = 0
0101	5	Ml	Minus	S = 1
0100	4	OV	Overflow	V = 1
1100	C	NOV	No Overflow	V = 0

Table 3-11. Condition Codes (Continued)

Binary	HEX	Mnemonic	Definition	Flag Settings
0110	6	EQ	Equal	Z = 1
1110	E	NE	Not Equal	Z = 0
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
0001	1	LT	Less Than	(S XOR V) = 1
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
1111	F	UGE	Unsigned Greater Than or Equal	C = 0
0111	7	ULT	Unsigned Less Than	C = 1
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1

NOTATION AND BINARY ENCODING

The operands and status flags use a notational shorthand. Operands, condition codes, address modes, and their notations are described in Table 3-12.

Table 3-12. Notational Shorthand

Notation	Address Mode	Operand	Range*
cc	Condition Code		See Table 3-11, condition codes
r	Working Register	Rn	n = 0 – 15
R	Register or Working Register	Reg Rn	`Reg`. represents a number in the range of 00H to FFH n = 0 – 15
RR	Indirect Register Pair or Working Register Pair	Reg RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
Ir	Indirect Working Register	@Rn	n = 0 –15
IR	Indirect Register or Indirect Working Register	@Reg @Rn	`Reg`. represents a number in the range of 00H to FFH n = 0– 15
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair or Working Register Pair	@Reg @RRp	`Reg`. represents an even number in the range 00H to FFH p=0, 2, 4, 6, 8, 10, 12, or 14
X	Indexed	Reg (Rn)	`Reg`. represents a number in the range of 00H to FFH n = 0 – 15
DA	Direct Address	Addrs	`Addrs`. represents a number in the range of 0000H to FFFFH
RA	Relative Address	Addrs	`Addrs`. represents a number in the range of +127 to –128 which is an offset relative to the address of the next instruction
IM	Immediate	#Data	Data is a number between 00H to FFH

*See the device product specification to determine the exact register file range available. The register file size varies by the device type.

Table 3-13, which follows, describes additional symbols used.

Table 3-13. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag Register (FCH)
RP	Register Pointer (FDH)
IMR	Interrupt Mask Register (FBH)
#	Immediate Operand Prefix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix
B	Binary Number Suffix
OPC	op code

Assignment of a value is indicated by the symbol ←, for example:

```
dst ← dst + src
```

indicates the source data is added to the destination data and the result is stored in the destination location.

The notation `addr(n)` is used to refer to bit 'n' of a given location. The following example refers to bit `7` of the destination operand.

```
dst (7)
```

Some instructions operate with several addressing modes. This situation is indicated by an op code number written like `x[ ]`. The brackets are filled by a nibble indicating the addressing mode in use. For example, `ADD 0[ ]` indicates that the `ADD` instruction works identically for more than one addressing mode.

Assembly Language Syntax

For proper instruction execution, assembly language syntax requires that the destination and source be specified as `dst, src` (in that order). The following instruction descriptions show the format of the object code produced by the assembler. This binary format should be followed by users who prefer manual program coding or who intend to implement their own assembler. Other third party assemblers can differ. Please consult the software user's manual for detailed information.

Example: The contents of registers `43H` and `08H` are added, and the result is stored in `43H`. The assembly syntax and resulting object code are:

```
ASM:      ADD    43H,     08H    (ADD dst, src)
OBJ:     04      08       43     (OPC src, dst)
```

In general, whenever an instruction format requires an 8-bit register address, that address can specify any register location in the range 0 - 255. When using working registers (R0-R15), a 4-bit address is used. If a working register is used and an 8-bit address is required by the assembler, an E is pre-pended to the 4-bit working register address. If, in the above example, the source register is a working register, the assembly syntax and resulting object code are:

```
ASM:      ADD    43H,     R8     (ADD dst, src)
OBJ:     04      E8       43     (OPC src, dst)
```

NOTES:

1. Note that the 4-bit address R8 was expanded to 8-bits by pre-pending `EH`. This expansion occurs any time a 4-bit address isspecified for an instruction that takes 8-bit operands.

2. See the device product specification to determine the exact register file range available. The register file size varies by device type

Z8^{PLUS} INSTRUCTION SUMMARY

The instructions marked with this symbol (†) have an identical set of addressing modes, which are encoded for brevity. The upper nibble is described in Table 3-14, and the lower nibble is represented by `[].` The second nibble's value is described in Table 3-15, and is found beside the applicable addressing mode pair. For example, the op code of an ADC instruction using the addressing modes `r` (destination) and `Ir` (source) is `13H.`

Table 3-14. Instruction Summary

Instruction and Operation	Address Mode		op code Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
ADC dst, src dst ← dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst ← dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst ← dst AND src	†		5[ ]	–	*	*	0	–	–
CALL src SP ← SP – 2 PC ← src		DA	D6	–	–	–	–	–	–
CALL src SP ← SP – 2 PC ← @src		IRR	D4	–	–	–	–	–	–
CCF C ←NOT C			EF	*	–	–	–	–	–
CLR dst dst ← 0	R IR		B0 B1	–	–	–	–	–	–
COM dst dst ← NOT dst	R IR		60 61	–	*	*	0	–	–
CP dst, src dst – src	†		A[ ]	*	*	*	*	–	–
DA dst dst ← DA dst	R IR		40 41	*	*	*	–	–	–
DEC dst dst ← dst – 1	R IR		00 01	–	*	*	*	–	–
DECW dst dst ← dst – 1	RR IR		80 81	–	*	*	*	–	–

Table 3-14. Instruction Summary (Continued)

Instruction and Operation	Address Mode		op code Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
DI IMR(7) ← 0			8F	–	–	–	–	–	–
DJNZ, dst, src r RA dst ← dst – 1 if dst ≠ 0 then PC ← PC + src Range: -128 ≤ src ≤ 127	RA		rA (r = 0 – F)	–	–	–	–	–	–
EI IMR(7) ← 1			9F	–	–	–	–	–	–
HALT			7F	–	–	–	–	–	–
INC dst dst ← dst + 1	 r R IR		rE (r = 0 – F) 20 21	–	*	*	*	–	–
INCW dst dst ← dst + 1	RR IR		A0 A1	–	*	*	*	–	–

IRET
FLAGS←@SP;
SP ← SP + 1
PC ← @SP;
SP ← SP + 2;
IMR(7) ← 1
			BF	*	*	*	*	*	*

JP cc, src if cc is true, then PC ← src		DA	ccD (cc = 0 – F)	–	–	–	–	–	–
JP src PC ← @src		IRR	30	–	–	–	–	–	–
JR cc, src if cc is true, then PC ← PC + src Range: -128 ≤ src ≤ 127		RA	ccB c = 0 – F	–	–	–	–	–	–

Table 3-14. Instruction Summary (Continued)

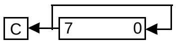
Instruction and Operation	Address Mode		op code Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
LD dst, src dst ← src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	r C r 8 r 9 (r = 0 – F) C7 D7 E3 F3 E4 E5 E6 E7 F5	–	–	–	–	–	–
LDC dst, src dst ← src	r lrr	Irr r	C2 D2	–	–	–	–	–	–
LDCI dst, src @dst ← @src dst ← dst + 1 src ←src + 1	Ir lrr	Irr r	C3 D3	–	–	–	–	–	–
NOP			FF	–	–	–	–	–	–
OR dst, src dst ← dst OR src	†		4[ ]	–	*	*	0	–	–
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	–	–	–	–	–	–
PUSH src SP ← SP – 1 @SP ← src	R IR		70 71	–	–	–	–	–	–
RCF C ← 0			CF	0	–	–	–	–	–
RET PC ← @SP; SP ← SP + 2			AF	–	–	–	–	–	–
RL dst 	R IR		90 91	*	*	*	*	–	–

Table 3-14. Instruction Summary (Continued)

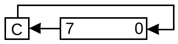
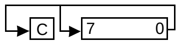
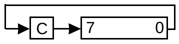
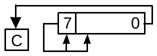
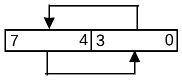
Instruction and Operation	Address Mode		op code	Flags Affected					
	dst	src	Byte (Hex)	C	Z	S	V	D	H
RLC dst 	R IR		10 11	*	*	*	*	–	–
RR dst 	R IR		E0 E1	*	*	*	*	–	–
RRC dst 	R IR		C0 C1	*	*	*	*	–	–
SBC dst, src dst ← dst – src – C	†		3[ ]	*	*	*	*	1	*
SCF C ← 1			DF	1	–	–	–	–	–
SRA dst 	R IR		D0 D1	*	*	*	0	–	–
SRP src RP ← src	Im		31	–	–	–	–	–	–
STOP			6F	–	–	–	–	–	–
SUB dst, src dst ← dst – src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	–	*	*	–	–	–
TCM dst, src (NOT dst) AND src	†		6[ ]	–	*	*	0	–	–
TM dst, src dst AND src	†		7[ ]	–	*	*	0	–	–
WDT			5F	–	–	–	–	–	–
XOR dst, src dst ← dst XOR src	†		7[ ]	–	*	*	0	–	–

Table 3-15. Lower Nibble Values

Address Mode dst	src	Lower op code Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

Figure 3-2, which follows, illustrates the Op Code map.

OP CODE MAP

LOWER NIBBLE (HEX) / UPPER NIBBLE (HEX)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**DEC** R1	**DEC** IR1	**ADD** r1, r2	**ADD** r1, Ir2	**ADD** R2, R1	**ADD** IR2, R1	**ADD** R1, IM	**ADD** IR1, IM	**LD** r1, R2	**LD** r2, R1	**DJNZ** r1, RA	**JR** cc, RA	**LD** r1, IM	**JP** cc, DA	**INC** r1	
1	**RLC** R1	**RLC** IR1	**ADC** r1, r2	**ADC** r1, Ir2	**ADC** R2, R1	**ADC** IR2, R1	**ADC** R1, IM	**ADC** IR1, IM	↓	↓	↓	↓	↓	↓	↓	
2	**INC** R1	**INC** IR1	**SUB** r1, r2	**SUB** r1, Ir2	**SUB** R2, R1	**SUB** IR2, R1	**SUB** R1, IM	**SUB** IR1, IM	↓	↓	↓	↓	↓	↓	↓	
3	**JP** IRR1	**SRP** IM	**SBC** r1, r2	**SBC** r1, Ir2	**SBC** R2, R1	**SBC** IR2, R1	**SBC** R1, IM	**SBC** IR1, IM	↓	↓	↓	↓	↓	↓	↓	
4	**DA** R1	**DA** IR1	**OR** r1, r2	**OR** r1, Ir2	**OR** R2, R1	**OR** IR2, R1	**OR** R1, IM	**OR** IR1, IM	↓	↓	↓	↓	↓	↓	↓	
5	**POP** R1	**POP** IR1	**AND** r1, r2	**AND** r1, Ir2	**AND** R2, R1	**AND** IR2, R1	**AND** R1, IM	**AND** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**WDT**
6	**COM** R1	**COM** IR1	**TCM** r1, r2	**TCM** r1, Ir2	**TCM** R2, R1	**TCM** IR2, R1	**TCM** R1, IM	**TCM** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**STOP**
7	**PUSH** R2	**PUSH** IR2	**TM** r1, r2	**TM** r1, Ir2	**TM** R2, R1	**TM** IR2, R1	**TM** R1, IM	**TM** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**HALT**
8	**DECW** RR1	**DECW** IR1							↓	↓	↓	↓	↓	↓	↓	**DI**
9	**RL** R1	**RL** IR1							↓	↓	↓	↓	↓	↓	↓	**EI**
A	**INCW** RR1	**INCW** IR1	**CP** r1, r2	**CP** r1, Ir2	**CP** R2, R1	**CP** IR2, R1	**CP** R1, IM	**CP** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**RET**
B	**CLR** R1	**CLR** IR1	**XOR** r1, r2	**XOR** r1, Ir2	**XOR** R2, R1	**XOR** IR2, R1	**XOR** R1, IM	**XOR** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**IRET**
C	**RRC** R1	**RRC** IR1	**LDC** r1, Irr2	**LDCI** Ir1, Irr2				**LD** r1,x,R2	↓	↓	↓	↓	↓	↓	↓	**RCF**
D	**SRA** R1	**SRA** IR1	**LDC** Irr1, r2	**LDCI** Irr1, Ir2	**CALL*** IRR1		**CALL** DA	**LD** r2,x,R1	↓	↓	↓	↓	↓	↓	↓	**SCF**
E	**RR** R1	**RR** IR1		**LD** r1, IR2	**LD** R2, R1	**LD** IR2, R1	**LD** R1, IM	**LD** IR1, IM	↓	↓	↓	↓	↓	↓	↓	**CCF**
F	**SWAP** R1	**SWAP** IR1		**LD** Ir1, r2		**LD** R2, IR1			↓	↓	↓	↓	↓	↓	↓	**NOP**
BYTES PER INSTRUCTION	2				3				2					3	1	

Notes:

All **Z8^{PLUS}** instructions execute in ten XTAL clock cycles, (1 µS at 10 MHz).

Blank areas are reserved and execute as NOP.

***** 2-byte instruction appears as a 3-byte instruction.

Legend:
R = 8-bit Addr
r = 4-bit Addr
R1 or r1 = Dst Addr
R2 or r2 = Src Addr

Sequence:
op code**,**
First Operand,
Second Operand

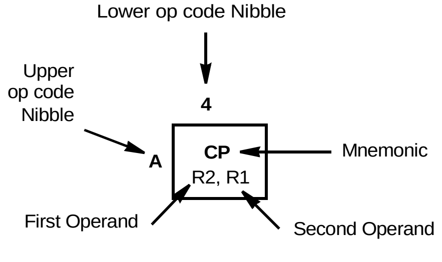


Figure 3-2. Op Code Map