

# INSTRUCCIONES

Las instrucciones del microcontrolador Z8<sup>PLUS</sup> se pueden clasificar en grupos de acuerdo a su función como:

- Instrucciones de carga
- Manipulación de bit
- Aritméticas
- Transferencias
- Lógicas
- Rotación y desplazamiento
- Control de programa
- Control de CPU

Cuando las instrucciones son ejecutadas, los registros definidos como fuentes son de solo lectura. Todos los registros de propósito general funcionan como:

- Acumuladores
- Apuntadores de dirección
- Registros índice
- Áreas de stack

Las siguientes tablas nos muestran las instrucciones que pertenecen a cada grupo y el número de operandos que se requiere para cada una. El operando fuente es “scr”, el operando destino “dst” y las condiciones de código, “cc”.

Tabla 9-1 Instrucciones de carga

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

Tabla 9-2 Instrucciones de operaciones aritméticas

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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

Tabla 9-3 Instrucciones de operaciones lógicas

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical Exclusive OR

Tabla 9-4 Instrucciones de control de programa

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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

Tabla 9-5 Instrucciones de manipulación de bits

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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

Tabla 9-6 Instrucciones de carga

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
LDCI	dst, src	Load Constant Auto Increment

Tabla 9-7 Instrucciones de rotación y desplazamiento

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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

Tabla 9-8 Instrucciones de control del CPU

<b>Mnemonic</b>	<b>Operands</b>	<b>Instruction</b>
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

## BANDERAS

El Registro de banderas es el que informa al usuario el estado actual del microcontrolador. Las banderas y la posición de sus bits en el registro de banderas (Flag Register) se muestran en la figura.

El Registro de banderas contiene 8 bits que nos dicen el estado del microcontrolador y son manipuladas por las operaciones del CPU. Cuatro de estos bits (C, V, Z y S) se pueden examinar para usar las condiciones de salto condicional. Dos banderas (H y D) son usadas para aritmética BCD. Los otros dos bits en el registro de banderas son para las banderas del reinicio por termino de conteo del WDT y para la bandera del Stop Mode Recovery.

Como en cualquier otro registro de instrucciones, los bits de este registro se pueden poner en estado alto o bajo por medio de instrucciones; sin embargo, solo se pueden usar instrucciones que no afecten el registro de banderas como consecuencia de su ejecución.

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	*	*

Fig. 9.1 Registro de banderas

### BANDERA DE ACARREO (C)

La bandera de acarreo se pone en estado alto cuando el resultado de una operación aritmética genera un acarreo o un préstamo del bit 7. De lo contrario, la bandera de acarreo se pone en estado bajo. En las instrucciones de rotación y desplazamiento, la bandera de acarreo contiene el ultimo valor del bit que sale del registro en el que se hace la operación.

Una instrucción puede poner en estado alto, en estado bajo o complementar la bandera de acarreo.

La bandera de acarreo no se ve afectada por el RESET.

## **BANDERA DE CERO (Z)**

Para operaciones lógicas y aritméticas, la bandera de cero se pone en estado alto si el resultado es cero. En caso contrario la bandera se mantiene en estado bajo.

Si el resultado de examinar un bit de un registro es cero entonces la bandera de cero se pone en estado alto. En caso contrario la bandera se mantiene en estado bajo. Si el resultado de la operación de rotación o desplazamiento es cero entonces la bandera de cero se pone en estado alto.

La bandera de cero no se ve afectada por el comando de RESET.

## **BANDERA DE SIGNO (S)**

La bandera de signo guarda el bit mas significativo del resultado de una operación lógica, aritmética, de rotación o de desplazamiento.

Cuando se llevan a cabo operaciones aritméticas con números que tienen signos, la notación binaria de complemento a dos es utilizada para representar y procesar información. Un número positivo se identifica con un cero en el bit más significativo, por lo tanto, en este caso la bandera de signo también es cero.

Un número negativo es identificado con un 1 en el bit mas significativo, por lo tanto la bandera de signo también es 1.

La bandera de cero no se ve afectada por el RESET.

## **BANDERA DE SOBREFLUJO (V)**

Para operaciones aritméticas de rotación o de desplazamiento, la bandera de sobreflujo se pone en estado alto cuando el resultado es más grande que el máximo número posible (mayor de 127) o menor que el número mínimo posible (menor que -128) que puede ser representado en la forma de complemento a dos. La bandera de sobreflujo se mantiene en estado bajo si no ocurren sobreflujos.

La bandera de sobreflujo no se ve afectada por el RESET.

## **BANDERA DE AJUSTE DECIMAL (D)**

La bandera de ajuste decimal es usada en aritmética decimal. Desde que el algoritmo para operaciones BCD es diferente para la suma y la resta, esta bandera especifica que tipo de instrucción fue la última que se ejecuto para que la siguiente operación pueda funcionar apropiadamente. El usuario no puede tener acceso a esta bandera.

Después de una resta, esta bandera se pone en estado alto y después de una suma se pone en estado bajo.

Esta bandera no se ve afectada por el RESET.

## **BANDERA DE MEDIO ACARREO (H)**

La bandera de medio acarreo se pone en estado alto cuando una suma genera un acarreo en el bit 3 o cuando una resta genera un “préstamo “ del bit 3. Esta bandera de medio acarreo es usada en las operaciones de ajuste decimal para convertir un resultado binario de una suma o una resta en un resultado decimal (BCD). Al igual que en la bandera de ajuste decimal el usuario no puede tener acceso a esta bandera.

Esta bandera no se ve afectada por el RESET.

## **BANDERA DEL WATCH DOG TIMER (WDT)**

La bandera de Watch Dog Timer se activa cuando el tiempo del Watch Dog Timer llega a cero y ocurre un reset. Esto le permite al software determinar cuando ha terminado el conteo del WDT.

Esta bandera se vuelve a poner en estado bajo con el pin del reset. Las banderas WDT y SMR son las únicas que se ven afectadas por el RESET. Este comportamiento le permite al software determinar cuando ha ocurrido un RESET, si el tiempo de conteo del WDT ha terminado, o si ha ocurrido un regreso del modo STOP.

El software debe de “limpiar” esta bandera después de detectar esta condición. En caso contrario podría haber un comportamiento inesperado.

## **BANDERA DE RECUPERACIÓN DE MODO STOP (SMR)**

La bandera de Recuperación del Modo Stop (Stop Mode Recovery, SMR) se pone en estado alto con la ejecución del modo STOP. Esto permite al software

determinar si un retorno del modo STOP ha ocurrido para regresar al estado activo.

Esta bandera se limpia con el pin de RESET. Las banderas WDT y SMR son las únicas banderas que se ven afectadas por el RESET. Este comportamiento permite al software determinar cuando ha ocurrido un RESET, si el tiempo del WDT ha terminado, o si un regreso del modo STOP ha ocurrido.

El programa debe de limpiar esta bandera después de detectar esta condición. En caso contrario podría haber un comportamiento inesperado.

Tabla 9-9 Definiciones de configuración de banderas

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

Tabla 9-10 Códigos de condición

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	MI	Minus	S = 1
0100	4	OV	Overflow	V = 1
1100	C	NOV	No Overflow	V = 0

Las banderas C, Z, S y V controlan la operación de las instrucciones de salto condicional. Las condiciones de código se ven resumidas en la tabla 8-10.



## NOTACIÓN Y CODIFICACIÓN BINARIA

Los operandos y los estados de banderas usan una notación. Las notaciones de los operandos, condiciones de código y modos de direccionamiento se describen en la tabla 8-11.

Tabla 9-11 Abreviaciones

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	Addr	Addr. represents a number in the range of 0000H to FFFFH
RA	Relative Address	Addr	Addr. 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.			

La siguiente tabla nos muestra símbolos adicionales:

Tabla 9-12 símbolos adicionales

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

La asignación de un valor se indica con el símbolo  $\leftarrow$ , por ejemplo:

$$\text{dst} \leftarrow \text{dst} + \text{src}$$

indica que el dato destino se suma al dato fuente y el resultado se guarda en la localidad del destino.

La notación  $\text{addr}(n)$  se usa para hacer referencia a un bit "n" de una localidad dada. El siguiente ejemplo hace referencia al bit 7 del operando destino:

$$\text{dst}(7)$$

Tabla 9-13 Resumen de instrucciones

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

Tabla 9-13 (continuación) resumen de instrucciones

Instruction and Operation	Address Mode		op code Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>DI</b> IMR(7) ← 0			8F	-	-	-	-	-	-
<b>DJNZ</b> , dst, src r RA dst ← dst - 1 if dst ≠ 0 then PC ← PC + src Range: -128 ≤ src ≤ 127	RA		rA (r = 0 - F)	-	-	-	-	-	-
<b>EI</b> IMR(7) ← 1			9F	-	-	-	-	-	-
<b>HALT</b>			7F	-	-	-	-	-	-
<b>INC</b> dst dst ← dst + 1	r R IR		rE (r = 0 - F) 20 21	-	*	*	*	-	-
<b>INCW</b> dst dst ← dst + 1	RR IR		A0 A1	-	*	*	*	-	-
<b>IRET</b> FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	*	*	*	*	*	*
<b>JP</b> cc, src if cc is true, then PC ← src		DA	ccD (cc = 0 - F)	-	-	-	-	-	-
<b>JP</b> src PC ← @src		IRR	30	-	-	-	-	-	-
<b>JR</b> cc, src if cc is true, then PC ← PC + src Range: -128 ≤ src ≤ 127		RA	ccB c = 0 - F	-	-	-	-	-	-

Tabla 9-13 (continuación) resumen de instrucciones

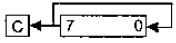
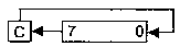
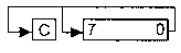
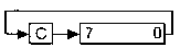
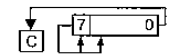
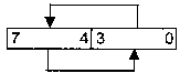
Instruction and Operation	Address Mode		op code Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>LD</b> dst, src dst ← src	r r R  r X r lr R R R IR IR	Im R r  X r r R R IM IM R	r C r 8 r 9 (r = 0 – F) C7 D7 E3 F3 E4 E5 E6 E7 F5	–	–	–	–	–	–
<b>LDC</b> dst, src dst ← src	r lrr	lrr r	C2 D2	–	–	–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

Tabla 9-13 (continuación) resumen de instrucciones

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

		LOWER NIBBLE (HEX)																		
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
UPPER NIBBLE (HEX)	0	DEC R1	DEC IR1	ADD r1, r2	ADD r1, r2	ADD R2, R1	ADD R2, 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, r2	ADC R2, R1	ADC R2, R1	ADC R1, IM	ADC IR1, IM											
	2	INC R1	INC IR1	SUB r1, r2	SUB r1, r2	SUB R2, R1	SUB R2, R1	SUB R1, IM	SUB IR1, IM											
	3	JP IRR1	SRP IM	SBC r1, r2	SBC r1, r2	SBC R2, R1	SBC R2, R1	SBC R1, IM	SBC IR1, IM											
	4	DA R1	DA IR1	OR r1, r2	OR r1, r2	OR R2, R1	OR R2, R1	OR R1, IM	OR IR1, IM											
	5	POP R1	POP IR1	AND r1, r2	AND r1, r2	AND R2, R1	AND R2, R1	AND R1, IM	AND IR1, IM											WDT
	6	COM R1	COM IR1	TCM r1, r2	TCM r1, r2	TCM R2, R1	TCM R2, R1	TCM R1, IM	TCM IR1, IM											STOP
	7	PUSH R2	PUSH IR2	TM r1, r2	TM r1, r2	TM R2, R1	TM R2, 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, r2	CP R2, R1	CP R2, R1	CP R1, IM	CP IR1, IM											RET
	B	CLR R1	CLR IR1	XOR r1, r2	XOR r1, r2	XOR R2, R1	XOR R2, R1	XOR R1, IM	XOR IR1, IM											IRET
	C	RRC R1	RRC IR1	LDC r1, r2	LDCI r1, r2															RCF
	D	SRA R1	SRA IR1	LDC IR1, r2	LDCI IR1, r2	CALL* IRR1		CALL DA	LD r2, R1											SCF
	E	RR R1	RR IR1		LD r1, R2	LD R2, R1	LD R2, R1	LD R1, IM	LD IR1, IM											CCF
	F	SWAP R1	SWAP IR1		LD R1, r2		LD R2, R1													NOP
		2		3		2		3		1										
		BYTES PER INSTRUCTION																		

Fig. 9.2 Mapa de códigos