MSP430 Instruction Set

Double Operand Instructions

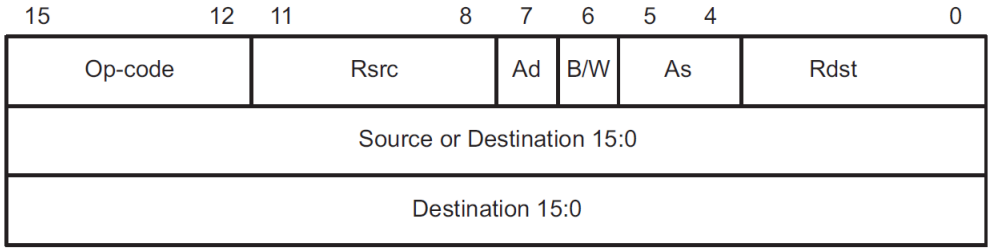


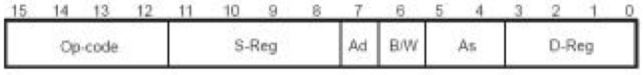
Figure 6-22. MSP430 Double-Operand Instruction Format

Table 6-4. MSP430 Double-Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits(1)			
			V	N	Z	C
MOV(.B)	src,dst	src → dst	–	–	–	–
ADD(.B)	src,dst	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(.B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(.B)	src,dst	dst - src	*	*	*	*
DADD(.B)	src,dst	src + dst + C → dst (decimally)	*	*	*	*
BIT(.B)	src,dst	src .and. dst	0	*	*	Z̄
BIC(.B)	src,dst	.not.src .and. dst → dst	–	–	–	–
BIS(.B)	src,dst	src .or. dst → dst	–	–	–	–
XOR(.B)	src,dst	src .xor. dst → dst	*	*	*	Z̄
AND(.B)	src,dst	src .and. dst → dst	0	*	*	Z̄

(1) * = Status bit is affected.
– = Status bit is not affected
0 = Status bit is cleared.
1 = Status bit is set.

MSP430: 16, 16-bit registers
R0 - Program counter
R1 - Stack pointer (SP) Last full location on TOS
R2 - Status register (SR)
R3 - Constant generator



As	Ad	Addressing Mode	Syntax	Description
00	0	Register Mode	Rn	Register contents are operand
01	1	Indexed Mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word
01	1	Symbolic Mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed Mode X(PC) is used
01	1	Absolute Mode	&ADDR	The word following the instruction contains the absolute address.
10	-	Indirect Register Mode	@Rn	Rn is used as a pointer to the operand
11	-	Indirect Autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards
11	-	Immediate Mode	#N	The word following the instruction contains the immediate constant N. Indirect Autoincrement Mode @PC+ is used

Single Operand Instructions

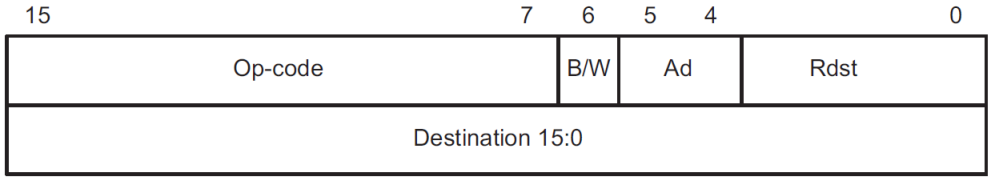


Figure 6-23. MSP430 Single-Operand Instructions

Table 6-5. MSP430 Single-Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits[1]			
			V	N	Z	C
RRC(.B)	dst	C → MSB →.....LSB → C	0	*	*	*
RRA(.B)	dst	MSB → MSB →....LSB → C	0	*	*	*
PUSH(.B)	src	SP - 2 → SP, src → SP	–	–	–	–
SWPB	dst	bit 15...bit 8 ↔ bit 7...bit 0	–	–	–	–
CALL	dst	Call subroutine in lower 64KB	–	–	–	–
RETI		TOS → SR, SP + 2 → SP	*	*	*	*
		TOS → PC,SP + 2 → SP				
SXT	dst	Register mode: bit 7 → bit 8...bit 19 Other modes: bit 7 → bit 8...bit 15	0	*	*	$\overline{Z}$

(1) * = Status bit is affected.
– = Status bit is not affected.
0 = Status bit is cleared.
1 = Status bit is set.

Jump Instructions

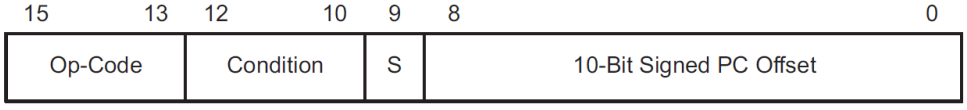


Figure 6-24. Format of Conditional Jump Instructions

Table 6-6. Conditional Jump Instructions

Mnemonic	S-Reg, D-Reg	Operation
JEQ, JZ	Label	Jump to label if zero bit is set
JNE, JNZ	Label	Jump to label if zero bit is reset
JC	Label	Jump to label if carry bit is set
JNC	Label	Jump to label if carry bit is reset
JN	Label	Jump to label if negative bit is set
JGE	Label	Jump to label if (N .XOR. V) = 0
JL	Label	Jump to label if (N .XOR. V) = 1
JMP	Label	Jump to label unconditionally

decimal	hexadecimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

.bss uninitialized
.data initialized
.text executable
.long 32 bit
.word 16 bit
.string 16 bit
.int 16 bit
words: .word 2, -1, 3, 4 (4 16 bit words)
Instruction length in words is +1 for all that are not indirect or autoincrement

Emulated Instructions

Table 6-7. Emulated Instructions

Instruction	Explanation	Emulation	Status Bits[1]			
			V	N	Z	C
ADC(.B) dst	Add Carry to dst	ADDC(.B) #0,dst	*	*	*	*
BR dst	Branch indirectly dst	MOV dst,PC	–	–	–	–
CLR(.B) dst	Clear dst	MOV(.B) #0,dst	–	–	–	–
CLRC	Clear Carry bit	BIC #1,SR	–	–	–	0
CLRN	Clear Negative bit	BIC #4,SR	–	0	–	–
CLRZ	Clear Zero bit	BIC #2,SR	–	–	0	–
DADC(.B) dst	Add Carry to dst decimally	DADD(.B) #0,dst	*	*	*	*
DEC(.B) dst	Decrement dst by 1	SUB(.B) #1,dst	*	*	*	*
DECD(.B) dst	Decrement dst by 2	SUB(.B) #2,dst	*	*	*	*
DINT	Disable interrupt	BIC #8,SR	–	–	–	–
EINT	Enable interrupt	BIS #8,SR	–	–	–	–
INC(.B) dst	Increment dst by 1	ADD(.B) #1,dst	*	*	*	*
INCD(.B) dst	Increment dst by 2	ADD(.B) #2,dst	*	*	*	*

(1) * = Status bit is affected.
– = Status bit is not affected.
0 = Status bit is cleared.
1 = Status bit is set.

Mnemonic and Syntax	Description	See
.bss *symbol, size in bytes*[, *alignment*]	Reserves *size* bytes in the .bss (uninitialized data) section	.bss topic
.data	Assembles into the .data (initialized data) section	.data topic
.intvec	Creates an interrupt vector entry in a named section that points to an interrupt routine name.	.intvec topic
.sect "*section name*"	Assembles into a named (initialized) section	.sect topic
.text	Assembles into the .text (executable code) section	.text topic
symbol **.usect** "*section name*", *size in bytes* [, *alignment*]	Reserves *size* bytes in a named (uninitialized) section	.usect topic

Table 6-7. Emulated Instructions (continued)

Instruction	Explanation	Emulation	Status Bits[1]			
			V	N	Z	C
INV(.B) dst	Invert dst	XOR(.B) #-1,dst	*	*	*	*
NOP	No operation	MOV R3,R3	–	–	–	–
POP dst	Pop operand from stack	MOV @SP+,dst	–	–	–	–
RET	Return from subroutine	MOV @SP+,PC	–	–	–	–
RLA(.B) dst	Shift left dst arithmetically	ADD(.B) dst,dst	*	*	*	*
RLC(.B) dst	Shift left dst logically through Carry	ADDC(.B) dst,dst	*	*	*	*
SBC(.B) dst	Subtract Carry from dst	SUBC(.B) #0,dst	*	*	*	*
SETC	Set Carry bit	BIS #1,SR	–	–	–	1
SETN	Set Negative bit	BIS #4,SR	–	1	–	–
SETZ	Set Zero bit	BIS #2,SR	–	–	1	–
TST(.B) dst	Test dst (compare with 0)	CMP(.B) #0,dst	0	*	*	1

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com