

MSP430 Addressing Modes

| As | Ad | d/s | Register | Syntax | Description |
|--|----|-----|------------------|-------------------|--|
| 00 | 0 | ds | $n \neq 3$ | Rn | Register direct. The operand is the contents of Rn. $A_d=0$ |
| 01 | 1 | ds | $n \neq 0, 2, 3$ | x(Rn) | Indexed. The operand is in memory at address $Rn+x$. |
| 10 | - | s | $n \neq 0, 2, 3$ | @Rn | Register indirect. The operand is in memory at the address held in Rn. |
| 11 | - | s | $n \neq 0, 2, 3$ | @Rn+ | Indirect auto-increment. As above, then the register is incremented by 1 or 2. |
| Addressing modes using R0 (PC) | | | | | |
| 01 | 1 | ds | 0 (PC) | LABEL | Symbolic. x(PC) The operand is in memory at address $PC+x$. |
| 11 | - | s | 0 (PC) | #x | Immediate. @PC+ The operand is the next word in the instruction stream. |
| Addressing modes using R2 (SR) and R3 (CG), special-case decoding | | | | | |
| 01 | 1 | ds | 2 (SR) | &LABEL | Absolute. The operand is in memory at address x. |
| 10 | - | s | 2 (SR) | #4 | Constant. The operand is the constant 4. |
| 11 | - | s | 2 (SR) | #8 | Constant. The operand is the constant 8. |
| 00 | - | s | 3 (CG) | #0 | Constant. The operand is the constant 0. |
| 01 | - | s | 3 (CG) | #1 | Constant. The operand is the constant 1. There is no index word. |
| 10 | - | s | 3 (CG) | #2 | Constant. The operand is the constant 2. |
| 11 | - | s | 3 (CG) | #-1 | Constant. The operand is the constant -1. |

MSP430 Instruction Set

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Instruction |
|----|----|----|----|----|----|--------|---|---|-----|----|----------|---|---|---|---|--|
| 0 | 0 | 0 | 1 | 0 | 0 | opcode | | | B/W | As | register | | | | | Single-operand arithmetic |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | B/W | As | register | | | | | RRC Rotate right through carry |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | As | register | | | | | SWPB Swap bytes |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | B/W | As | register | | | | | RRA Rotate right arithmetic |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | As | register | | | | | SXT Sign extend byte to word |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | B/W | As | register | | | | | PUSH Push value onto stack |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | As | register | | | | | CALL Subroutine call; push PC and move source to PC |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RETI Return from interrupt; pop SR then pop PC |

| | | | | | | | | | | | | | | | | |
|---|---|---|-----------|---|---|----------------------|--|--|--|--|--|--|--|--|--|--|
| 0 | 0 | 1 | condition | | | 10-bit signed offset | | | | Conditional jump; $PC = PC + 2 \times \text{offset}$ | | | | | | |
| 0 | 0 | 1 | 0 | 0 | 0 | 10-bit signed offset | | | | JNE/JNZ Jump if not equal/zero | | | | | | |
| 0 | 0 | 1 | 0 | 0 | 1 | 10-bit signed offset | | | | JEQ/JZ Jump if equal/zero | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 10-bit signed offset | | | | JNC/JLO Jump if no carry/lower | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 1 | 10-bit signed offset | | | | JC/JHS Jump if carry/higher or same | | | | | | |
| 0 | 0 | 1 | 1 | 0 | 0 | 10-bit signed offset | | | | JN Jump if negative | | | | | | |
| 0 | 0 | 1 | 1 | 0 | 1 | 10-bit signed offset | | | | JGE Jump if greater or equal | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 0 | 10-bit signed offset | | | | JL Jump if less | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 10-bit signed offset | | | | JMP Jump (unconditionally) | | | | | | |

| | | | | | | | | | |
|--------|---|---|---|--------|----|-----|----|-------------|---|
| opcode | | | | source | Ad | B/W | As | destination | Two-operand arithmetic |
| 0 | 1 | 0 | 0 | source | Ad | B/W | As | destination | MOV Move source to destination |
| 0 | 1 | 0 | 1 | source | Ad | B/W | As | destination | ADD Add source to destination |
| 0 | 1 | 1 | 0 | source | Ad | B/W | As | destination | ADDC Add source and carry to destination |
| 0 | 1 | 1 | 1 | source | Ad | B/W | As | destination | SUBC Subtract source from destination (with carry) |
| 1 | 0 | 0 | 0 | source | Ad | B/W | As | destination | SUB Subtract source from destination |
| 1 | 0 | 0 | 1 | source | Ad | B/W | As | destination | CMP Compare (pretend to subtract) source from destination |
| 1 | 0 | 1 | 0 | source | Ad | B/W | As | destination | DADD <i>Decimal</i> add source to destination (with carry) |
| 1 | 0 | 1 | 1 | source | Ad | B/W | As | destination | BIT Test bits of source AND destination |
| 1 | 1 | 0 | 0 | source | Ad | B/W | As | destination | BIC Bit clear (dest &= ~src) |
| 1 | 1 | 0 | 1 | source | Ad | B/W | As | destination | BIS Bit set (logical OR) |
| 1 | 1 | 1 | 0 | source | Ad | B/W | As | destination | XOR <i>Exclusive or</i> source with destination |
| 1 | 1 | 1 | 1 | source | Ad | B/W | As | destination | AND <i>Logical AND</i> source with destination (dest &= src) |

MSP430 Emulated Instructions

| Mnemonic | Operation | Emulation | Description |
|--------------------------------|---------------------------------|-----------------------|--|
| Arithmetic Instructions | | | |
| ADC(.B or .W) dst | dst+C→dst | ADDC(.B or .W) #0,dst | Add carry to destination |
| DADC(.B or .W) dst | dst+C→dst (decimally) | DADD(.B or .W) #0,dst | Decimal add carry to destination |
| DEC(.B or .W) dst | dst-1→dst | SUB(.B or .W) #1,dst | Decrement destination |
| DECD(.B or .W) dst | dst-2→dst | SUB(.B or .W) #2,dst | Decrement destination twice |
| INC(.B or .W) dst | dst+1→dst | ADD(.B or .W) #1,dst | Increment destination |
| INCD(.B or .W) dst | dst+2→dst | ADD(.B or .W) #2,dst | Increment destination twice |
| SBC(.B or .W) dst | dst+0FFFh+C→dst dst+0FFh→dst | SUBC(.B or .W) #0,dst | Subtract source and borrow /.NOT. carry from dest. |

| Mnemonic | Operation | Emulation | Description |
|--|----------------------------|-----------------------------|----------------------------|
| Logical and Register Control Instructions | | | |
| INV(.B or .W) dst | .NOT.dst→dst | XOR(.B or .W) #0(FF)FFh,dst | Invert bits in destination |
| RLA(.B or .W) dst | C←MSB←MSB-1 LSB+1←LSB←0 | ADD(.B or .W) dst,dst | Rotate left arithmetically |
| RLC(.B or .W) dst | C←MSB←MSB-1 LSB+1←LSB←C | ADDC(.B or .W) dst,dst | Rotate left through carry |

| | | | |
|-----------------------------|-------------------|-------------|------------------------------|
| Program Flow Control | | | |
| BR dst | dst→PC | MOV dst,PC | Branch to destination |
| DINT | 0→GIE | BIC #8,SR | Disable (general) interrupts |
| EINT | 1→GIE | BIS #8,SR | Enable (general) interrupts |
| NOP | None | MOV #0,R3 | No operation |
| RET | @SP→PC SP+2→SP | MOV @SP+,PC | Return from subroutine |

| Mnemonic | Operation | Emulation | Description |
|--------------------------|-----------------------------------|------------------------|---|
| Data Instructions | | | |
| CLR(.B or .W) dst | 0→dst | MOV(.B or .W) #0,dst | Clear destination |
| CLRC | 0→C | BIC #1,SR | Clear carry flag |
| CLRN | 0→N | BIC #4,SR | Clear negative flag |
| CLRZ | 0→Z | BIC #2,SR | Clear zero flag |
| POP(.B or .W) dst | @SP→temp SP+2→SP temp→dst | MOV(.B or .W) @SP+,dst | Pop byte/word from stack to destination |
| SETC | 1→C | BIS #1,SR | Set carry flag |
| SETN | 1→N | BIS #4,SR | Set negative flag |
| SETZ | 1→Z | BIS #2,SR | Set zero flag |
| TST(.B or .W) dst | dst + 0FFFh + 1 dst + 0FFh + 1 | CMP(.B or .W) #0,dst | Test destination |

