

Magutz Microchip *MPLab Programming Instructions

A collection of macros to facilitate assembly language programming.

Along with simplifying the use of words, 16 bit registers.

A collection of 158 instruction that also include the standard *Microchip commands. 110 for working with single register and 48 for working with double registers or words.

Once you get use to them; you'll wonder how you managed without them!

License:

This information and macros are free, you can redistribute modify or do with them what ever makes you happy, just at least give me some of the credit. I spent many hours writing and testing them. Also you may want to check with *Microchip, the language it self is theirs.

Disclaimer:

And of course there is no guarantee what so ever. By using this information and/or macros you agree that all risks involved are yours.

Any collaboration from your part will be appreciated, like bugs or additional material that could be added.

software@magutz.com

Let's Keep Life Simple, Why Complicate It?
www.magutz.com

* Microchip & MPLab are register trade marks of Microchip Technology Inc.

Copyright © 2005, MaGutz, inc. All rights reserved.

Version 1.1

MG Instruction Set

Step	Instruction	Format	Description	Step	Instruction	Format	Description
2	ADD	fr, lit	Add literal to fr	5	INVF	fr1, fr2	Exchanges fr1 with fr2
2	ADDF	fr1, fr2	Add 2 frs	1	INCFSZ	fr, w	Inc fr result in W - Skp
1	ADDWF	fr, f	Add W to fr	2	JB	bit, label	Jump if bit = 1
1	ADDWF	fr, w	Add fr to W	2	JC	label	Jump to label if Carry
1	ADDLW	lit	Add literal to W	1	GOTO	label	Jump to label
2	ADDB	fr, bit	Add bit to fr	2	JNB	bit, label	Jump if no bit
2	AND	fr, lit	AND fr with literal	2	JNC	label	Jump if no Carry
2	ANDF	fr1, fr2	AND 2 frs	2	JNZ	label	Jump if not Zero
1	ANDWF	fr, f	AND fr with W	2	JZ	label	Jump if Zero
1	ANDWF	fr, w	AND W with fr	1-3	LCALL	label	Long Call
1	ANDLW	lit	AND W with literal	1-3	LGOTO	label	Long Jump
1	CALL	label	Call function	2	MOV	fr, lit	Move literal to fr
4	CJA	fr, lit, label	CJA fr to literal	2	MOVX	fr1, fr2	Copy fr2 to fr1
4	CJAF	fr1, fr2, label	CJA 2 frs	1	MOVWF	fr	Copy W to fr
4	CJAE	fr, lit, label	CJAE fr to literal	1	MOVLW	lit	Copy literal to W
4	CJAEF	fr1, fr2, label	CJAE 2 frs	4	MOVB	bit1, bit2	Copy bit2 to bit1
4	CJB	fr, lit, label	CJB fr to literal	4	MOVXB	bit1, bit2	Copy not-bit2 to bit1
4	CJBF	fr1, fr2, label	CJB 2 frs	1	NOP		No operation
4	CJBE	fr, lit, label	CJBE fr to literal	2	OR	fr, lit	OR fr with literal
4	CJBEF	fr1, fr2, label	CJBE 2 frs	2	ORF	fr1, fr2	OR 2 frs
4	CJE	fr, lit, label	CJE fr to literal	1	IORWF	fr, f	OR fr with W
4	CJEF	fr1, fr2, label	CJE 2 frs	1	IORWF	fr, w	OR W with fr
4	CJNE	fr, lit, label	CJNE fr to literal	1	IORLW	lit	OR W with literal
4	CJNEF	fr1, fr2, label	CJNE 2 frs	1	RETURN		Return 1 stack level
1	CLRB	bit	Clear bit	?	RETLW	lit	Return literal in W
1	CLRC		Clear Carry	1	RL	fr	Rotate Left fr
1	CLRDC		Clear Digit Carry	1	RLF	fr, w	RL fr result in W
1	CLR	fr	Clear register	1	RR	fr	Rotate Right fr
1	CLRW		Clear W	1	RRF	fr, w	RR fr result in W
1	CLRWD		Clear Watch Dog Tmr	1	SB	bit	Skip if bit
1	CLRZ		Clear Zero	1	SC		Skip if Carry
1	COMF	fr, f	Complement fr	1	SETB	bit	Set bit
1	COMF	fr, w	Complement fr to W	1	SKIP		Skip following intruct
3	CSA	fr, lit	CSA fr to literal	1	SLEEP		Sleep Mode set
3	CSAF	fr1, fr2	CSA 2 frs	1	SNB	bit	Skip if no bit
3	CSAE	fr, lit	CSAE fr to literal	1	SNC		Skip if No Carry
3	CSAEF	fr1, fr2	CSAE 2 frs	1	SNZ		Skip if Not Zero
3	CSB	fr, lit	CSB fr to literal	1	SETC		Set Carry
3	CSBF	fr1, fr2	CSB 2 frs	1	SETZ		Set Zerro
3	CSBE	fr, lit	CSBE fr to literal	2	SUB	fr, lit	Sub literal from fr
3	CSBEF	fr1, fr2	CSBE 2 frs	2	SUBF	fr1, fr2	Sub fr2 from fr1
3	CSE	fr, lit	CSE fr to literal	1	SUBWF	fr, f	Sub W from fr
3	CSEF	fr1, fr2	CSNE 2 frs	1	SUBWF	fr, w	Sub W from fr result W
3	CSNE	fr, lit	CSNE fr to literal	2	SUBB	fr, bit	Sub bit from fr
3	CSNEF	fr1,fr2	CSNE 2 frs	1	SWAP	fr	Swap nibbles if fr
1	DEC	fr	Decrement fr	1	SWAPF	fr, w	Swap nibbles fr to W
1	DECF	fr, w	Dec fr result in W	1	SZ		Skip if Zero
1	DECFSZ	fr, f	Dec fr result in fr - Skp	1	TEST	fr	Test fr for zero
1	DECFSZ	fr, w	Dec fr result in W - Sk	1	TESTW		Test W for zero
2	DJNZ	fr, label	Dec fr Jump if not Z	2	XOR	fr, lit	XOR fr with literal
2	IJNZ	fr, label	Inc fr Jump if not Zero	5	XORB	fr, lit	Invert bit
1	INC	fr	Increment register	2	XORF	fr1, fr2	XOR 2 frs
1	INCF	fr, w	Inc fr result in W	1	XORWF	fr, f	XOR fr with W
1	INCFSZ	fr, f	Inc fr result in fr - Skip	1	XORWF	fr, w	XOR W with fr
3	INW	fr	Exchanges W with fr1	1	XORLW	lit	XOR W with literal

MG Word Instruction Set

Step	Instruction	Format	Description
6	WADD	wd, litH, litL	Adds a 2-byte literal value to word.
6	WADDF	wd1, wd2	Adds two words together and puts the result in wd1.
6	WSUB	wd, litH, litL	Subtracts a 2-byte literal value from word wd1.
6	WSUBF	wd1, wd2	Subtracts word wd2 from word wd1, result in wd1.
3	WINC	wd	Increment word.
4	WDEC	wd	Decrement word.
30	WDIV	Rslt, wd, litH, litL	Divide wd(fr1H/L) by 2-byte literal, result in Rslt(16bit)
30	WDIVF	Rslt, wd1, wd2	Divide word wd1 by wd2, result goes to RsltH/L(16bit)
10	WMULT	Result, fr	Multiplies fr x W, result goes to Result_H/L.(16bit)
12	WCJA	wd, litH, litL, Label	Compare word jump if above 2-byte literal value.
12	WCJAF	wd1, wd2, Label	Compare two words jump if wd1 is above wd2.
12	WCJAE	wd, litH, litL, Label	Compare word jump if above or equal to 2-byte literal value.
12	WCJAEF	wd1, wd2, Label	Compare two words jump if wd1 is above or equal to wd2.
12	WCJB	wd, litH, litL, Label	Compare word jump if below 2-byte literal value.
12	WCJBF	wd1, wd2, Label	Compare two words jump if wd1 is below wd2.
12	WCJBE	wd, litH, litL, Label	Compare word jump if below or equal to 2-byte literal value.
12	WCJBEF	wd1, wd2, Label	Compare two words jump if wd1 is below or equal to wd2.
8	WCJE	wd, litH, litL, Label	Compare word jump if equal to 2-byte literal value.
8	WCJEF	wd1, wd2, Label	Compare two words jump if equal.
8	WCJNE	wd, litH, litL, Label	Compare word jump if not equal to 2-byte literal value.
8	WCJNEF	wd1, wd2, Label	Compare two words and jump if not equal.
11	WCSA	wd, litH, litL	Compare word skip if above 2-byte literal value.
11	WCSAF	wd1, wd2	Compare two words skip if wd1 is above wd2.
11	WCSAE	wd, litH, litL	Compare word skip if above or equal 2-byte literal value.
11	WCSAEF	wd1, wd2	Compare two words skip if wd1 is above or equal to wd2.
11	WCSB	wd, litH, litL	Compare word skip if below the 2-byte literal value.
11	WCSBF	wd1, wd2	Compare two words skip if wd1 is below wd2.
11	WCSBE	wd, litH, litL	Compare word skip if below or equal 2-byte literal value.
11	WCSBEF	wd1, wd2	Compare two words skip if wd1 is below or equal to wd2.
7	WCSE	wd, litH, litL	Compare word skip if equal to 2-byte literal value.
7	WCSEF	wd1, wd2	Compare two words skip if equal.
7	WCSNE	wd, litH, litL	Compare word skip if not equal to 2-byte literal value.
7	WCSNEF	wd1, wd2	Compare two words skip if not equal.
4	WAND	wd, litH, litL	AND word with a 2-byte literal value.
4	WANDF	wd1, wd2	AND word wd1 with value in word wd2.
4	WOR	wd, litH, litL	OR word wd1 with a 2-byte literal value.
4	WORF	wd1, wd2	OR word wd1 with value in word wd2.
4	WXOR	wd, litH, litL	XOR word wd1 with a 2-byte literal value.
4	WXORF	wd1, wd2	XOR word wd1 with value in word wd2.
2	WCLR	wd	Clears the contents in word.
4	WTEST	wd	Check word Z status.
2	WRL	wd	Rotate word bits left wd1.
2	WRR	wd	Rotate word bits right wd1.
10	WDJNZ	wd, Label	Decrement word jump if not zero, try using wijnz.
5	WIJNZ	wd, Label	Increment word jump if not zero, more efficient than wdjnz.
4	WMOV	wd, litH, litL	Copy 2-byte Literal value to word wd1.
4	WMOVX	wd1, wd2	Copy word wd2 to word wd1.
5	WSWAP	wd	Swap the contents between H and L bytes of word wd1.

Defining words:

```
#define volts volts_H, volts_L
#define timer timer_H, timer_L
```

NOTE: It is very important to always remember that there is a difference between working with registers and literal values, the little "f" makes a huge difference.

A word is composed of frH and frL.

Notes:

To use this macros, they must be included in the header of the program as shown below. (also see below for using word macros)

```
#include <MG Macros.inc> ; List of Macro definitions
```

For use with Microchip MPLab MPASM.

Microchip & MPLab are register trade marks of Microchip Technology Inc.

When using values in registers, just add an f to the end of the instruction, all except mov, use movx because movf is already used by Mpsasm. The other exception is movbx = move NOT bit.

example: add = for literal value, addf = for value in register.

Use lower case z & c, Upper case only points to a bit number.

Word Macros, working with two registers. This are the same as the basic instructions, just with a "w" on front of them.

To use the word macros, the words must be defined in the program.

Defining a word:

```
#define volts      volts_H, volts_L  
#define timer     timer_H, timer_L
```

NOTE: It is very important to always remember that their is a difference between working with registers and literal values, the little "f" makes a huge difference.

A word is composed of frH and frL.

Other Instructions:

```
Return      Return form call function
```

This Bank set instructions are the same as "banksel" which also use two instructions even if they are not needed.

```
Bank0      Move to Bank0  
Bank1      Move to Bank1  
Bank2      Move to Bank2  
Bank3      Move to Bank3
```