

PICAsm

Dirk Düsterberg

COLLABORATORS

	<i>TITLE :</i> PICAsm		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Dirk Düsterberg	February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	PICAsm	1
1.1	PICAsm Manual	1
1.2	overview	2
1.3	processors	2
1.4	support	3
1.5	installation	3
1.6	requirements	3
1.7	registration	4
1.8	legalmush	4
1.9	bugreports	5
1.10	author	5
1.11	history	5
1.12	Expression evaluation	6
1.13	assembler directives	8
1.14	device	11
1.15	list	11
1.16	nolist	13
1.17	radix	13
1.18	processor	13
1.19	org	13
1.20	end	14
1.21	ds	14
1.22	res	14
1.23	cblock	14
1.24	macro	15
1.25	include	15
1.26	incdir	16
1.27	define	16
1.28	undefine	17
1.29	ifdef	17

1.30	ifndef	17
1.31	else	18
1.32	endif	18
1.33	dt	18
1.34	if	18
1.35	messg	20
1.36	error	20
1.37	space	20
1.38	source Code format	20
1.39	label	21
1.40	command	22
1.41	operands	22
1.42	comments	22
1.43	Using PICasm with CygnusEd	22
1.44	about PIC's	24
1.45	p16c5x.inc	27
1.46	p16cxx.inc	28

Chapter 1

PICAsm

1.1 PICAsm Manual

PICAsm guide
Version 0.9.4

A Pic Cross Assembler
for Amiga Computers

by Joannis Petroglou

18.05.97

Overview
What is PICAsm?

Support
Call me!

Installation
How to install PICSim

Requirements
System Requirement

Registration
limitations ?

Using PICAsm

format
Source Code format

directives

assembler directives

expression
Expression evaluation

Processors
supported PIC's

Pic_Info
about PIC's

editor
using with CygnusEd

Copyright
Legal mush

Bug reports
How to report bugs

The author
Programmer's address

History
Revision history of PICSim

1.2 overview

PICAsm is an Amiga cross assembler for 12, 14 and 16 bit \leftrightarrow microcontroller from microchip. PICAsm uses the microchip commands and many features of microchip's mpasm assembler.

Questions, criticism, suggestions and bug reports are always welcome.

1.3 processors

12 bit types:

- PIC16C54
- PIC16C55
- PIC16C56
- PIC16C57
- PIC16C58

14 bit types:

- PIC16C61
- PIC16C62
- PIC16C64
- PIC16C65
- PIC16C620
- PIC16C621
- PIC16C622
- PIC16C71
- PIC16C73
- PIC16C74
- PIC16C84

16 bit types:

- PIC17C42
- PIC17C43
- PIC17C44

1.4 support

The official PICAsm homepage in the WWW has always the latest version and other information related to PIC Tools:

<http://linux.rz.fh-hannover.de/~duesterb>

1.5 installation

just copy the files in a directory on your harddisk

1.6 requirements

Requirements:

- The Amiga must have at least a 68020 processor. PICAsm will run on every Amiga with at least OS 2.0
- some free mem

1.7 registration

THIS PICAsm VERSION HAS NO LIMITATIONS !

please let us know if you like this program.

1.8 legalmush

The programs "PICSim", "PicProgger", "PicASM" may be freely distributed as long as they remain unchanged (archiving and packing are allowed).

No profit must be made by distributing PICSim, especially the price of a disk containing PICSim may not exceed US\$ 5,- (or equivalent amounts in other currencies).

Please feel free to distribute PICSim over bulletin board systems and networks as part of shareware/freeware CD-ROMs. All rights for commercial use remain at the author

.

The Program that registered users will receive, must only be installed one one computer and in no case passed on to others. Offences will result in penal prosecution by me. With your signature on the order form, you accept these conditions.

The program is presented to the users as it is, without any warranty of any kind, be it expressed or implicit. Anyone using this program agrees to incur the risk of using it for himself. In no way can the author be made responsible for any damage directly or indirectly caused by the use or misuse of the program.

Names of other hardware and software items mentioned in this manual and in program texts are in most cases registered trade marks of the respective

companies and not marked as such. So the lack of such a note may not be used as an indication that these names are free.

1.9 bugreports

If you find a bug or a misfeature in PICAsm, or have an idea how to make some things better, then please drop me a note so I'll be able to improve PICAsm in the future. My address can be found here

.

Important for a bug report is the following information:

- Version of PICAsm
- Used AmigaOS version (e.g. 3.1, 2.0 etc.)
- Used sourcefile
- Information about installed startup programs on the Amiga
- Detailed description what program produces the bug and how it can be reproduced

1.10 author

There was no Pic Simulator on Amiga.
So I had to do it. :-)

My address is:

Mail:

petroglo@unixserv.rz.fh-hannover.de

WWW:

<http://linux.rz.fh-hannover.de/~duesterb/>

Questions, criticism, suggestions and
bug reports
are always welcome.

1.11 history

xx.04.97 -several beta releases

06.05.97 -first release v0.9

18.05.97 -second release v0.9.4

1.12 Expression evaluation

Expresion evalution

Numeric Constnts and Radix

10 ;binary, dezimal or hexadezimal term,
;use RADIX to choose, hexadezimal is default

.10 ;dezimal term

0ch or 0xc ;hexadezimal term, must begin with number
;end with h or begins with "0x"

1100b ;binary term, end with "b"
;this term could make probs if hex is default

362o ;octal term, end with "o"

number ;symbol

"A" ;ascii, case sensitive (A = 41h, a = 61h),
;could be a "string" at RETLW command

or:

D'123' ;dezimal
H'1f' ;hexadezimal
O'123' ;octal
B'10001010' ;binary
A'W' or 'W' ;ascii, only single character

symbol declaration:

number = .12

or:

count equ .13

where ever a symbol is found in a term it will be substituted by declared value.

Arithmetic operators:

These operators perform their normal functions

```
+      add      ; eg 4 + 5
-      subtract ; eg 156 - 38
*      multiply  ; eg 7 * 101b
/      divide   ; eg 20h / 5
%      modulus  ; eg 134%2

(      left parenthesis
)      right parenthesis

{      every kind of paranthesis needs same kind to end
}      expression

[
]
```

Boolean operators

Boolean operators perform bit-wise logical operations on 8 bit data. The following Boolean operators are supported:

```
&      and      ; eg 1001101 & value
|      or       ; eg 0fh | 12h
^      exclusive-or ; eg 25 ^ number
>> term  shift right term times ; eg 89 >> 4
<< term  shift left term times  ; eg (4 * 8) << 4

!<expr> complement of expression
```

misc operators

```
low <expr> low byte of expression
high <expr> high byte of expression
```

```
$      value of program counter
```

comparation operators

```
== equal
!= not equal
> Greater
< Less
>= Greater or equal
<= Less or equal
&& Logical AND
|| Logical OR
```

substraction and addition have the lowest priority, division, multiplikation and boolean operation have the same priority. expressions are resolved from left to right.

eg:

```
4h * -5 + 30    = 10      ; -20 + 30
```

```
4h * (-5 + 30) = 100     ; 4 * 25
```

```
(4h * (-5 + 30)) >> 1 = 50      ; (4 * 25) shifted one time right
```

```
clockspeed = 4000000      ;clockspeed is 4 Mhz
baudrate   = 19200        ;enter baudrate here
delay      = (clockspeed/4/baudrate-12)/4 ;value for delay
```

digits after point are truncated. eg: 10/4 => 2

1.13 assembler directives

assembler directives

assembler directives must be found in PASS1, location in source doesn't matter.

```
device
    -setup a device

list
    -switches listfile outputon

nolist
    -switches listfile output off

radix
    -set value radix

processor
    -setup a processor

org
    -define adress in memory

end
    -stop assembling

ds
    -define space

res
    -reserve space

cblock
    -constant block

dt
    -define table

macro
    -macro

include
    -include file

incdir
    -set include directory
```

```
#define          -define symbol

#undefine       -undefine symbol

ifdef         -conditional assembling

ifndef        -conditional assembling

else          -conditional assembling

endif         -end of conditional assembling

if           -end of conditional assembling

messg        -Create User Defined Message

error        -Issue an Error Message

space        -Insert Blank Listing Lines

expand       -future

noexpand     -future

title        -future

subtitle     -future
```

1.14 device

```
DEVICE DEVICE sets pictype
```

eg:

```
DEVICE PIC16c54 ;defines pic type  
or:
```

```
DEVICE 16c54
```

for mpasm compatibility use

```
LIST  
or  
PROCESSOR  
command
```

1.15 list

```
LIST switches the listfile on, following source is written to ←  
the  
listfile until NOLIST command, optionally following options  
can be used:
```

default

```
radix  
(can changed several in source):
```

```
r=dez ;radix is dezimal (default for values)  
r=hex ;radix is hexadezimal  
r=bin ;radix is binary
```

eg:

```
LIST r=dec ;radix is decimal  
movlw 32 ;move d'32' (h'20') to w
```

```
LIST r=hex ;radix is hexa decimal  
movlw 32 ;move d'50' (h'32') to w
```

```
        processor
        type:
(last device in source used if several devices choosed)
```

```
p=pic16c58
```

```
or:
```

```
p=16c58
```

```
case sensitivity (can changed several in source):
```

```
s=on      ;case sensitive on (default)
s=off     ;case sensitive off
```

```
eg:
```

```
LIST p=16c84, s=on, r=dec
```

```
key = 4      ;key is 4 (4)
KEY = 5*key  ;KEY is 5*key (20)
number = KEY-2 ;number is KEY-2 (18)
```

```
LIST s=off
```

```
movlw Number ;Number (number) to w (18)
```

```
look also:
```

```
RADIX
```

```
PROCESSOR
```

```
DEVICE
```

```
NOLIST
```

1.16 nolist

NOLIST switches the listfile off, following source is not ↵
written to
listfile until
LIST
command.

1.17 radix

RADIX specify default radix
(can changed several in source)

see also

LIST
command

RADIX dec ;radix is decimal
RADIX hex ;radix is hexadecimal (default)
RADIX bin ;radix is binary

1.18 processor

PROCESSOR set processor type

see also

LIST
command

PROCESSOR 16c84
or:

PROCESSOR pic16c84

1.19 org

ORG define address in memory

ORG 1ffh sets program counter to 1ffh, following code
begins at this address.

Reset vektor of 5x PIC types is last memory address
(1ffh, 3ffh or 7ffh)

eg:

```
ORG 1ffh    ;this is the reset vektor
goto start ;start label
ORG 0      ;beginning of code
```

1.20 end

END this causes the assembler to end assembling, following source
doesn't matter.

1.21 ds

```
DS    define space in program memory, equal to
res
command
```

eg: DS 2 defines space of two bytes

1.22 res

```
RES   reserve memory, equal to
ds
command
```

eg: RES 2 defines space of two bytes

1.23 cblock

CBLOCK define a block of constants

Define a list of defined constants. Each is assigned
a value of one higher than the last one.
The end of block is defined by ENDC

eg:

```

cblock [expr] ;optionaly expression sets
    ;start
name_1      ;assigned to expr
name_2      ;assigned to expr + 1
name_3      ;assigned to expr + 2
name_4, name_5, name_6 ;assigned to +3, +4, +5
endc

```

<expr> indicates the starting value for the first name in the block. If no expression is found, the first name will receive a value one higher than the final name in the previous CBLOCK or the current program counter

1.24 macro

macro defines a macro

eg:

```

macro pagel ;this is a macro named pagel
    bcf 03h,5 ;clear pageselect bit 0
    bsf 03h,6 ;set page select bit 1
endm ;end of macro

```

or:

```

macro wait arg1,arg2 ;this is a macro with two arguments
    movlw arg1 ;arg1 is a value
    movwf arg2 ;arg2 is a fileregister
loop decfsz arg2 ;do loop arg1 times
    goto loop ;labels in macros are local loops
endm

```

using macros:

```

pagel ;set page select bits to page 1
goto adress ;jump to adress on page 1
wait 100, 0x1f ;wait 100 loops, use register 0x1f

```

1.25 include

#include includes source file, includes could be nested without limits ↔

eg:

```
#include "includes:p16c5x.inc" ;include file
```

```
p16c5x.inc  
-include file for 16c5x types
```

```
p16cxx.inc  
-include file for 16cxx types
```

see also

```
includir  
command
```

1.26 includir

```
#includir sets include directory.
```

eg:

```
#includir "pictools:includes"
```

see also

```
include  
command
```

1.27 define

```
#define defines text substitution or symbols (ifdef, etc)
```

eg:

```
#define demo
```

```
#define pagel bsf 03h,5 ;if pagel occurs it will  
;be exchanged with with 'bsf 03h'
```

defined names can be used in command or operand field.

see also

```
#undefine
```

command

1.28 undefine

#undefine defined labels can be undefined

eg:

```
#undefine demo
```

see also

```
#define  
command
```

1.29 ifdef

ifdef assemble following source if label is defined, several ifdef's can be nested

eg:

```
ifdef demo  
call demo  
endif
```

see also

```
#define  
command
```

1.30 ifndef

ifndef assemble following source if label is not defined

eg:

```
ifndef fullversion  
call demo
```

```
endif
```

see also

```
#define  
command
```

1.31 else

```
else    end of case in conditional assembly
```

eg:

```
ifdef  demo  
  call demo  
else  
  call nodemo  
endif  
goto  loop
```

see also

```
#define  
command
```

1.32 endif

```
endif    end of conditional assembly
```

1.33 dt

```
same as          define table  
                retlw
```

1.34 if

```
if <expr>
```

see also:

```
else
```

```
endif
```

```
eg:
```

```
num1 = 5
```

```
num2 = 5
```

```
; == equal  
; != not equal  
; >= Greater or equal  
; <= Less or equal  
; && Logical AND  
; || Logical OR  
;
```

```
if num1+num2 == 7  
  error "num1+num2 = 7"  
else  
  error "num1+num2 != 7"  
endif
```

```
if (num1 != 3) && (num2 != 5)  
  error "num1 is not 3 and num2 not 5"  
endif
```

```
if num1 < num2  
  error "num1 is less than num2"  
else  
  error "num2 is not less than num1"  
endif
```

```
if num1 <= num2  
  error "num1 is less or equal num2"  
endif
```

```
if num1 >= num2  
  error "num1 is greater or equal num2"  
endif
```

```
if num1 != num2
```

```
error "num1 is not equal num2"  
endif
```

```
if (num1==5) || (num2 == 5)  
error "num1 or num2 is 5"  
endif
```

1.35 messg

MESSG "<message_text>" - Create User Defined Message

Causes an informational message to be printed in the listing file. The message text can be up to 255 characters. Issuing a MESSG directive does not set any error return codes.

1.36 error

ERROR "<text_string>" - Issue an Error Message

The <text_string> is printed in a format identical to any PICAsm error message. <text_string> may be from oneto eighty characters.

1.37 space

SPACE <expr> - Insert Blank Listing Lines

example: space 3
Insert <expr> number of blank lines into the listing file.

1.38 source Code format

source code format

format:

```
[label]
[command]
[operands]
;
[comments]
eg:
```

```
loop    incfsz    0fh    ;increase register file 0fh
        goto     loop    ;goto loop
        movlw    01100b*4 ;move literal to working register
```

1.39 label

[label]
the label may begin with A...Z, a...z or underscore ("_"). case is sensitive as default, you can toggle sensitivity in source with the list parameter. The label may optionally be followed by a colon, which is ignored, but is useful in searching for the line where a label is defined. Labels have to begin at the beginning of the line.

eg:

```
loop    incfsz    09h
        goto     loop
```

"loop:" is handled same as "loop" without colon behind. A colon before the label defines a local label.

:[label]
this is an local label, same local labels can be used more than one time in the same program. An local label is only between two normal labels valid. There must be different root labels for same local labels.

eg:

```
start  swapf  0eh,f

:loop  incfsz  0eh,f
      goto   :loop

next   bsf    0fh,6

:loop  incfsz  0fh
      goto   :loop
```

1.40 command

[command] PIC mnemonic, macro or assembler directive

eg:

```
list    p=pic16c54, r=dec, s=off
movlw   89      ;move literal to w
```

1.41 operands

[operand] this could be a literal, expression or label

eg:

```
movlw   2*(9*(0x8-5)+10011b) ;move literal (expr) to w

movlw   h'ff'^"k"      ;complement of "k"
          ;(xor`ed with h'ff')
```

1.42 comments

[comments] comments to source

eg:

```
          ;this is a comment
```

1.43 Using PICasm with CygnusEd

use with CygnusEd

you can start PICAsm automatic with Cygnus Ed. If you opened a source file you only need to press choosed function key to save and assemble the actual source file (when ARexx is running). Here is an ARexx example: ↵

-Save this ARexx file to s:PICAsm.ced

-Choose "Spezial / Dos/ARexx interface / Install dos Arexx command" in Cygnus Ed

-Choose FKey number

-Enter REXX commando
(s:PICAsm.ced)

-Choose "Spezial / Dos/ARexx interface / Save dos Arexx command" in Cygnus Ed

-----cut here ↵

/* automtic assembling of Pic source code with CygnusEd
*/

```
PARSE SOURCE com res called resolved ext host . /* Get host */
IF host="REXX" THEN /* From command line */
  ADDRESS "rexx_ced" /* Talk to default ced */
```

```
OPTIONS RESULTS /* Enable results from commands */
ARG windowType . /* How to show? */
```

```
ADDRESS COMMAND "relabel ram: Ram_Disk"
```

```
'DM "assembling source..."'
```

```
'Status NUMCHANGES' /* Get number of changes made to file */
IF RESULT~=0 THEN /* Changes were made */
  'Save' /* Save current file */
```

```
'Status FILENAME' /* Fet file name (with path) */
FileName=RESULT
```

```
Comm="c:PICAsm" FileName      /* Set up command string */
                          /* Change status bar again */
ADDRESS COMMAND Comm          /* do the command */
'CEdToFront'                  /* Jump back to CEd's screen */

'DM'                           /* Restore status line */
```

EXIT

-----cut here ←

1.44 about PIC`s

about PIC`s

PIC16c5x register:

```
00h indirect addressing
01h real time clock counter (RTTC)
02h program counter
03h Status register bit0 => carry bit (c)
    bit1 => digit carry bit (dc)
    bit2 => zero bit (z)
    bit3 => power down bit (pd)
    bit4 => time out bit (to)
    bit5 => page select bit (pa0)
    bit6 => page select bit (pa1)
    bit7 => unused bit (pa2)

04h file select register, memory page select
05h I/O portA 4bit
06h I/O portB 8bit
07h I/O portC 8bit

08h user Ram
.
.
.
```

Pic16C5x commands:

```

fr => fileregister
lit => literal
    the RETLW literal can be a string, eg:

    retlw "hello World", h'0a', "good morning", h'0a'

    retlw 'S','C','E','E'

    retlw 0x1, 100011b, h'ef'

W => Workingregister
addr8 => 8bit address (goto)
addr9 => 9bit address (call)
d => d=0 -> result in W, d=1 -> result in fr (microchip)
    or: d=w -> result in w, d=f -> result in fr

    default is d=1

```

Microchip Description:

```

command:  operand:  change:  description:

ADDWF    fr,d      c,dc,z    add W and fr

ANDLW    lit      z      and W with literal
ANDWF    fr,d      z      and W with fr

BCF      fr,b      -      clear bit
BSF      fr,b      -      set bit

BTFSC    fr,b      -      bit test f and skip if bit clear
BTFSS    fr,b      -      bit test f and skip if bit set

CALL     addr8     -      call address

CLRF     fr        z      clear fileregister
CLRW     z         z      clear Workingregister
CLRWDT   to,pd     -      clear Watchdogtimer

COMF     fr,d      z      complement from fr

```

DECF	fr,d	z	decrease fr
DECFSC	fr,d	-	decrease fr and skip if zero
GOTO	addr9	-	goto address
INCF	fr,d	z	increase fr
INCFSC	fr,d	-	increase fr and skip if zero
IORLW	lit	z	inclusive or literal with W
IORWF	fr,d	z	inclusive or fr with W
MOVF	fr,d	z	move fr
MOVLW	lit	-	move literal to W
MOVWF	fr	-	move W to fr
NOP	-	-	no operation
OPTION	-	-	move W to Optionregister
RETLW	lit	-	return from subroutine with literal in W
RLF	fr,d	c	rotate fr left
RRF	fr,d	c	rotate fr right
SLEEP	to,pd	-	sleep mode
SUBWF	fr,d	c,dc,z	sub W from fr
SWAPF	fr,d	-	swap nibbles from fr
TRIS	port_fr	-	move W to Tristate register
XORLW	lit	z	literal xor W
XORWF	fr,d	z	exclusive or w with fr

additional 16Cxx commands:

ADDLW	lit	c,dc,z	add literal and fr
RETFIE	addr	-	return from interrupt
RETURN	addr	-	return
SUBLW	lit	c,dc,z	sub literal from w

Dirk Düsterberg, duesterb@unixserv.rz.fh-hannover.de

<http://linux.rz.fh-hannover.de/~duesterb>

1.45 p16c5x.inc

```
LIST
; P16C5X.INC Standard PICAsm Header File
NOLIST

; 16C54
; 16C55
; 16C56
; 16C57

;=====

W          EQU      H'0000'
F          EQU      H'0001'

;----- Register Files -----

    CBLOCK          H'0'
        INDF
        TMR0
        PCL
        STATUS
        FSR
        PORTA
        PORTB
    ENDC

;----- STATUS Bits -----

    CBLOCK          H'0'
        C
        DC
        Z
        NOT_PD
        NOT_TO
        PA0
        PA1
        PA2
    ENDC

;----- OPTION Bits -----
```

```

CBLOCK          H'0'
  PS0
  PS1
  PS2
  PSA
  TOSE
  TOCS
ENDC

```

```
LIST
```

1.46 p16cxx.inc

```

LIST
; P16CXX.INC Standard PICAsm Header File
NOLIST

```

```

;=====

W          EQU      H'0000'
F          EQU      H'0001'

;----- Register Files-----

INDF          EQU      H'0000'
TMR0         EQU      H'0001'
PCL          EQU      H'0002'
STATUS       EQU      H'0003'
FSR          EQU      H'0004'
PORTA        EQU      H'0005'
PORTB        EQU      H'0006'

PCLATH       EQU      H'000A'
INTCON       EQU      H'000B'

OPTION_REG   EQU      H'0081'
TRISA        EQU      H'0085'
TRISB        EQU      H'0086'

;----- INTCON Bits (except ADC/Periph) -----

GIE          EQU      H'0007'
TOIE         EQU      H'0005'
INTE         EQU      H'0004'
RBIE         EQU      H'0003'
TOIF         EQU      H'0002'
INTF         EQU      H'0001'
RBIF         EQU      H'0000'

;----- OPTION Bits -----

```



```
NOT_RBPU          EQU      H'0007'  
INTEEDG          EQU      H'0006'  
TOCS             EQU      H'0005'  
TOSE             EQU      H'0004'  
PSA              EQU      H'0003'  
PS2              EQU      H'0002'  
PS1              EQU      H'0001'  
PS0              EQU      H'0000'
```

```
;----- STATUS Bits -----
```

```
IRP              EQU      H'0007'  
RP1              EQU      H'0006'  
RP0              EQU      H'0005'  
NOT_TO          EQU      H'0004'  
NOT_PD          EQU      H'0003'  
Z                EQU      H'0002'  
DC              EQU      H'0001'  
C                EQU      H'0000'
```

```
;----- Register Files -----
```

```
EEDATA          EQU      H'0008'  
EEADR           EQU      H'0009'  
  
EECON1          EQU      H'0088'  
EECON2          EQU      H'0089'
```

```
LIST
```
