Order this document
as EB410/D

MOTOROLA
SEMICONDUCTOR
ENGINEERING BULLETIN

EB410

PASM05 to INTROL HC05 Assembler Conversion

By Dugald Campbell,
Motorola Ltd.,
East Kilbride, Scotland

ABSTRACT

The following engineering bulletin describes some of the differences between Motorola's PASM05 V0.05 Assembler and a third party assembler, the INTROL Assembler V3.06 for MS-DOS Hosts. This document describes some of the most common changes. This should help speed up the engineering time to convert the syntax from PASM05 to INTROL. With the use of an intelligent Editor that supports MACRO EDITING functions, multiple conversions can be done with one key press. The EDITOR used in the conversion described was "BRIEF V3.0" by UnderWare Inc.

ASSEMBLER CONTENTS

Both Assemblers come complete with 3 main programs

1) An Assembler

2) A Linker

3 An "Object to S-record" translator

The Assembler program will assemble ASCII source code files to relocatable object code files. The Linker program will bind the object code files into a single absolute binary file. The "Object to S-record" translator translates binary object code to an ASCII S-record file.

	PASM05	INTROL
Assembler	PASM05.EXE	AS05.EXE
Linker	PLD.EXE	ILD05.EXE
S-Record translator	UBUILDS.EXE	IHEX.EXE

MS-DOS ASSEMBLER COMMANDS

Both Assemblers have very similar MS-DOS Line Commands. Both have options to create LIST and OBJECT files. The main difference, at this level, is that the PASM05 looks for the source file at the end of the command line, while INTROL looks for the source file immediately after the command name.

PASM05

pasm05 -l file.lst file.asm
(-l option ensures a list file is generated)

INTROL

as05 file.asm -t
(A list file is generated by default called file.lst.
The -t option inhibits 'branch-size optimizations)

© MOTOROLA LTD., 1993

MS-DOS LINKER COMMANDS

Both assemblers produce relocatable object code, and provide a LINKER program that connects multiple modules together and assigns absolute addresses. In both cases an "OPTIONS" or "LOAD" file is used to determine desired addresses to locate input modules by explicitly determining these parameters. The INTROL Linker will only access an "OPTIONS" file that has the extension " .LD ". For example, the equivalent command lines to link files, would be:

PASM05

pld -o FILE.obj -d FILE.lnk
(-o creates an output file called FILE.obj
-d open OPTIONS file "FILE.lnk")

INTROL

ild05 -o FILE.obj -g FILE
(-o creates an output file called FILE.obj
-g open OPTIONS file "FILE.ld")

OPTION FILE CONTENTS

The PASM05 OPTION file is shown below:

```
SECTION
{
IO_REGS:   org = $0000, end =$001F
ROM:       org = $0020, end =$004F
RAM:       org = $0076, end =$00BF
ROMEXT:    org = $0800, end =$08FF
PROGRAM:   org = $0900, end =$1EFF
VECTORS:   org = $1FF0, end =$1FFF
}
SECTIONS
{
ASCT :     {} > IO_REGS
IDSCT :    {} > ROM
DSCT :     {} > ROMEXT
BSCT :     {} > RAM
PSCT :     {} > PROGRAM
VSCT :     {} >VECTORS
}
ram.o
table.o
program1.o
program2.o
program3.o
vectors.o
```

The "SECTION {}" directive defines output segments with a Start and End address. The "SECTION {}" directive defines input segments, which are defined in the SOURCE modules, to output segments. The FILES.o are binary object code files that were generated from the source code modules using the assembler, e.g. in the source file "program2.asm " the name DSCT was given to a block of code. When the Linker fetches the object file (program2.o) it recognises this label. It then assigns this block of code to the area defined by ROMEXT: which starts at $0800 and finishes at $08FF.

The input segments have reserved names as shown above which limits the number of output sections. See Assembler Syntax for more details.

The INTROL OPTIONS file is shown below:

```
section .ROM0       origin 0x0020      maxsize 0x001F = ROM0;
section .RAM        origin 0x0076      maxsize 0x0049 = RAM;
section .ROMEXT     origin 0x0800      maxsize 0x00FF = ROM1;
section .PROGRAM    origin 0x0900      maxsize 0x15FF = ROM;
section .VECTORS    origin 0x1FF0      maxsize 0x1FFF = VECTORS;

ram.o05
table.o05
program1.o05
program2.o05
program3.o05
vectors.o05
```

The INTROL OPTIONS file assigns output segments directly from sections named in the Source code modules, thus allowing the user to have an unlimited amount of sections. If the input sections are larger than the output segment specified then a linker error will occur. See Assembler Syntax for further details.

OBJECT CODE TO S-RECORD TRANSLATION

PASM05	INTROL
ubuilds FILE.obj FILE.mx	**ihex FILE.obj -I 32**

The IHEX program converts the input file FILE.obj to an S-record file called "FILE.0".

The "-I 32" ensures the S-record length is the same as the PASM05 file "FILE.MX".

ASSEMBLER SYNTAX

Below is a list of syntax changes required, to allow the INTROL Assembler to execute without error. These changes should be implemented to the Source, Macro, and Module files:

PASM05	INTROL
TTL	**TITLE**
OPT FMT,NOS	no equivalent syntax (remove from source)
XDEF	**EXPORT**
XREF	**IMPORT**

include	**lib**
PSCT	**SECTION ROM**
ASCT	no equivalent syntax (remove from source)
DSCT	**SECTION ROM1**
VSCT	**SECTION VECTORS**
IDSCT	**SECTION ROM0**
MACR	**MACRO**
loop MACR \0,\1,\2	**loop MACRO \0 ,\1 ,\2** ^ ^ space inserted before the comma
IFGT Variable	**IF Variable>0**
IFEQ Variable	**IF Variable==0**
ENDC	**ENDIF**
MEXIT	**EXITM**
label MEXIT	**label <CRLF>** **<TAB>EXITM**
var1 equ +32-offset	**var1 equ 32-offset** (INTROL produces an error with "+" prefix)
lda #2!^index (!^ = exponential)	**lda #1<<index** (no exponential in INTROL therefore a shift to right of #1 gives same result as #2(exp)index)
lda variable!+index (!+ = inclusive OR)	**lda variable I index** (I = inclusive OR)
<TAB><TAB> !Comment	***<TAB><TAB> !Comment** (The '*' must be on the 1st character of the line if no assembly code is present)
label<TAB><TAB> !Comment	**label<CRLF>** *<TAB><TAB>! Comment (INTROL will not accept comments following a label that has no assembly code.)

INTROL "LIB" DIRECTIVE LIMITATION

The LIB directive calls other files, which are then included with the file being assembled. A limitation exists in the INTROL assembler, when using the LIB directive. If the LIB directive is nested more than 3 times then an assembler error will occur, e.g.:

FILE1.ASM	FILE 2.ASM	FILE3.ASM	FILE4.ASM
lib FILE2.ASM	**lib FILE3.ASM**	**lib FILE4.ASM**	**lda #data**
sta location1	**stx location2**	**ldx #temp**	**END**
END	**END**	**END**	

Executing AS05 (INTROL) on FILE1.asm would cause an error due to the LIB directive being nested 3 times.

To enable the INTROL assembler to produce an output file the files would be written as shown below:

FILE1.ASM	FILE 2.ASM	FILE3.ASM	FILE4.ASM
lib FILE2.ASM	**lib FILE3.ASM**	**ldx #temp**	**lda #data**
sta location1	**lib FILE4.ASM**	**END**	**END**
END	**stx location2**		
	END		

This will produce an output list file

FILE1.LST

```
0100    AE 3F       ldx     #temp
0102    A6 55       lda     #data
0104    BF 51       stx     location2
0106    B7 50       sta     location1

                    END
```

USING AN INTELLIGENT EDITOR TO DO SYNTAX CHANGES

Existing software developed using PASM05 for the HC05, if done modularily, and following Good Software Quality Control,would produce many different files. Also if the Source code written is for a MCU with more than 6K of ROM/EPROM then the amount of syntax changes will be very large, and hand editing the file or files could be a time consuming process.

Today there are a wide range of Intelligent Editors that allow USERS to generate their own Editing functions. Using these new functions can help reduce the Editing time of doing a PASM05 to INTROL conversion.

The editor used, in this case, was " BRIEF V3.0" written by "UnderWare Inc.". "Brief" allows users to make Custom Editing Macros using a 'C' type language combined with special function calls. After writing these macros, the file can be compiled and recalled by some simple key presses. Below is a MACRO Source file that was used to do most of the Syntax changes needed for a PASM05 to INTROL conversion.

```
/* THIS MACRO IS EXECUTED ON A HC05 SOURCE CODE FILE USING THE TEXT EDITOR "BRIEF". THIS
MACRO WILL CONVERT PASM05 SYNTAXTO INTROL SYNTAX. WHEN THE MACRO FINISHES THEN THE
FILE SHOULD BE SAVED, AND IS READY TO BE ASSEMBLED USING THE INTROL ASSEMBLER */

void CHANGE()                          /* function name is "CHANGE" */
{
  top_of_buffer();                     /* move cursor to top of file */

/*
 The "WHILE" loop below looks for Comments that start in the middle of
a line, and have no LABELs or Assembly Code prior. The INTROL assembler
needs a "*" at the start_of_the_line if the comment has no Assembler code prior.
*/
 while ( search_fwd ("!") != 0)        /* search for "!" if found do the following below */
       {
          beginning_of_line();

          while ( (read(1) == " ") || (read(1) == "\t") )
       { right(); };                   /* If the Cursor position is either a space or TAB then move right
*/

        if   ( read(1) == "!" )        /* If this character is a "!" then go to */
             { beginning_of_line();
               insert("*");            /* and place a "*".                      */
             };                        /* If any other char. then it must be assembler */

        down();                        /* move down a line to ensure next Search goes forward */

        beginning_of_line();
       };                              /* end of while loop .. Placing "*" in Comment positions */

  top_of_buffer();                     /* Move cursor back up to the top of the file */

/*
The WHILE loop below looks for the PASM05 MACRO directive
  "     IFGT     variable     !comments     "
 and replaces it with the INTROL syntax
  "     IF variable>0      !comments   "
*/
 while ( search_fwd ("IFGT") != 0)     /* search for "IFGT" if found do the following below */
       {
         right();                      /* move Cursor 2 positions to the right */
         right();
         delete_char(2);               /* delete the characters 'G' and 'T' */
         insert("  ");                 /* insert two spaces to keep syntax tidy */

         while ( (read(1) == " ") || (read(1) == "\t" ) ) /* Move cursor till it reaches a char */
             { right(); };             /* that isn't a space or TAB             */

         while ( ( read(1) != " " ) && ( read(1) != "\t" ) && ( read(1) != "\n" ) )
             { right(); };             /* move cursor till it reaches a space or TAB character*/
```

```
    /* Cursor should be at end of "operand"  */

        insert(">0");                       /* Place the " >0" here */
        right();
      };                                    /* end of while loop */

  top_of_buffer();

/*
The WHILE loop below looks for the PASM05 MACRO directive
  "     IFEQ     variable     !comments  "
and replaces it with the INTROL syntax
  "     IF variable==0      !comments  "
*/
  while ( search_fwd ("IFEQ") != 0)
      {
        right();
        right();
        delete_char(2);
        insert("  ");

        while ( (read(1) == " ") || (read(1) == "\t" ) )
            { right(); };

      while ( ( read(1) != " " ) && ( read(1) != "\t" ) && ( read(1) != "\n" ) )
            { right(); };
        insert("==0");
        right();
      };                                    /* end of while loop */

  top_of_buffer();                          /* Move cursor back up to the top of the file */

/*
the functions below look for all occurrences of the 1st string and replace it with the 2nd string. The " ,1 "
ensures that the editor changes all occurrences without prompting.
*/

  translate ("include","lib",1);
  translate ("TTL","TITLE",1);
  translate ("XDEF","EXPORT",1);
  translate ("PSCT","SECTION ROM",1);
  translate ("BSCT","SECTION RAM",1);
  translate ("XREF","IMPORT",1);
  translate ("ASCT","    ",1);
  translate ("MACR","MACRO",1);
  translate ("ENDC","ENDIF",1);
  translate ("MEXIT","EXITM",1);
  translate ("\\0,\\1,\\2","\\0 , \\1 , \\2",1,0);  /*  0 => specified expression is not regular. */
  translate ("DSCT","SECTION ROM1",1);
  translate ("xdef","export",1);
  translate ("xref","import",1);
  translate ("2!^","1<<",1);
  translate ("NOCL,FMT,G,MC,NOMD,MEX,NOS","NOCL,G,MC,NOMD,MEX",1);


} /* END of function "CHANGE" . Source file has now been converted to INTROL syntax
     File should now be saved, and INTROL assembly can be executed on this file */
```

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447

MFAX: RMFAX0@email.sps.mot.com –TOUCHTONE (602) 244–6609
INTERNET: http://Design–NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, Toshikatsu Otsuki,
6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–3521–8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298