MACRO CROSS ASSEMBLERS
Series I and II

USER'S MANUAL

for the following microprocessors:

Series I	Series II
8044	6502
8048	65C02
8051/52	6800
8080	6803
8085	6805
8096	6809
Z80	

MICEPT INSTRUMENTS INC.

MACRO CROSS ASSEMBLERS
Series I and II

USER'S MANUAL

Edition 3

MICEPT INSTRUMENTS INC.
377 Julien St.
Cap De La Madeleine, PQ
Canada G8T 6W6

www: http://www3.sympatico.ca/jveillet/micept
e-mail: mailto: jveillet@sympatico.ca

NOTICE

DISCLAIMER

Micept Instruments Inc. disclaims all warranties as to this software and the documentation, whether express or implied, including, without limitation, any implied warranties of merchantability and of fitness for any purpose. Micept Instruments Inc. assumes no liability for damages, direct or consequential, which may results from the use of this software or the documentation.

Information in this document is subject to change without notice.

TRADEMARKS

Intel is a registered trademark of Intel Corporation.
Zilog and Z80 are registered trademarks of Zilog Inc.
Rockwell 6502 is a registered trademark of Rockwell International Corporation.
MS-DOS is a trademark of Microsoft Corporation.
IBM, PC, XT, AT are trademarks of International Business Machines Corporation.
All brand and product names are trademarks or registered trademarks of their respective companies.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview

The Series I contains Macro Cross Assemblers for the Intel 8044, 8048, 8051/52, 8080, 8085, 8096, and the Zilog Z80 microprocessors. The Series II contains Macro Cross Assemblers for the Motorola 6800, 6803, 6805, 6809 and the Rockwell 6502 and 65C02. This software are basic two-pass assemblers which supports macros, conditional assembly and include files.

These powerful Macro Cross Assemblers for the MS-DOS operating system support the complete instruction set and a complete set of directives. They also support the 8051/52 and 8044 special function registers and their assigned bit names. These assemblers can produce formatted listing files and object code in both the Intel Hex and the Motorola Hex formats.

The supported assembler directives are the same as the ones used by the microprocessor manufacturers, but you can customised them to your liking using the REDEF directive.

1.2 Specifications

The Macro Cross Assemblers supports:

- up to 16 nested include files
- up to 16 nested conditional assembly sections
- up to 16 nested macros
- unlimited number of include files
- unlimited number of macros (only limited by memory)

1.3 About this manual

This manual provides the necessary technical information to install and use the Macro Cross Assemblers in the series I. It is written for the individual end user but a certain measure of experience in the operation of MS-DOS has been assumed. A certain knowledge of the assembly language of each microprocessor is also assumed.

1.4 Notational conventions

The following rules are used in this manual:

1. Words in capital letters are keywords and must be entered as shown. They may be entered in uppercase or lowercase characters.
2. Items in square brackets ([]) are optional.

3. Symbolic fields are enclosed in < >.

4. An ellipsis (...) indicates that an item may be repeated as many time as you wish.

5. All punctuation, except the one mentioned before, must be included where shown.

6. xxx is a two or three character string representing the microprocessor identification.
   This string is:

03	: for the Motorola 6803 microprocessor
05	: for the Motorola 6805 microprocessor
09	: for the Motorola 6809 microprocessor
44	: for the Intel 8044 microprocessor
48	: for the Intel 8048 microprocessor
51	: for the Intel 8051/52 microprocessor
65	: for the Rockwell 6502 microprocessor
65C	: for the Rockwell 65C02 microprocessor
68	: for the Motorola 6800 microprocessor
80	: for the Intel 8080 microprocessor
85	: for the Intel 8085 microprocessor
96	: for the Intel 8096 microprocessor
Z80	: for the Zilog Z80 microprocessor

1.5 Learning more about the microprocessors

You may want to familiarise yourself with the microprocessors by reading assembly language books, other handbooks and data books.

2. GETTING STARTED

2.1 Installation

Make a working copy of your disk, using the COPY or DISKCOPY utility supplied with MS-DOS. Save the original disk for backup.

No particular installation is required except that you have to copy the executable files to the disk you are going to use.

2.2 System requirements

- IBM PC/XT/AT or compatible computer
- DOS 2.0 or later
- 256K of RAM required (more for bigger symbol table or many macros)

2.3 Disk contents

Each disk (one for each microprocessor) will contain the following files:

MAxxx.EXE	The executable file of the corresponding Macro Cross Assembler
EXAMPLEx.xxx	An example of a source file
README	The file containing the latest information

2.4 Setting up the environment

Each assembler environment can be customised by the use of an environment variable. The variable, if set, will specify the default command line options. The options specified in that variable will not replace but will be added to the options invoked in the command line. Options are specified in the same manner as if they were invoked in the command line. The setting of this variable can be automatically done if the set command is placed in the AUTOEXEC.BAT file.

The variable is called **MAxxxOPT** and can be set by the following command:

```
SET MAxxxOPT=-l -m -r10h
```

That command would invoked the l, m and r options each time the assembler is invoked.

Note: Any error in this environment variable will be ignored.

3. ASSEMBLING

3.1 Running the assembler

The assembler can be invoked as follow:

MAxxx [<options>] <sourcefile> [<objfile> [<listfile>]]

Where <***options***> can be one or more of the following separated by spaces:

-l	Produce a listing file
-o	Produce NO object file
-t	Produce NO symbol table in the listing
-m	Produce object in Motorola Hex (Series I only)
-i	Produce object in Intel Hex (Series II only)
-q	Quiet mode
-s	Display software license
-p<*n*>	Set listing page length (in lines per page)
-w<*n*>	Set listing width (in characters per line)
-r<*n*>	Set object record size (in bytes)
-e	Expand macros in the listing
-f	Do not list false conditions
-x	Use extended special function register set for the 8052
-c	Produce complete object in listing for long constants

<***sourcefile***>	Source file name.	Default extension is ".xxx".
<***objfile***>	Object file name.	Default extension is ".OBJ".
<***listfile***>	Listing file name.	Default extension is ".LST".

If the assembler is invoked with an error in the command line, a help screen is displayed on the console.

For <***objfile***> and <***listfile***>, if no name is specified, the source file name with the default extensions will be used and the files will be created in the current directory.

3.2 Options in more details

c Complete object code in listing for long constants. This is used to see all object codes generated by long constants, like strings and series of data. A source line may then generate more than one line in the listing because only a few object codes can be written on one line of listing.

e Expand macros in the listing. This option lists the complete macro text when that macro is invoked. Even when this option is invoked, the expansion of macros in the listing file will still be controlled by the EXPAND and NOEXPAND directives.

f False conditional sections not listed. This option suppresses the listing of conditional sections that evaluate as false. Even when this option is invoked, the listing of conditional section that evaluate as false will still be controlled by the FLIST and FNOLIST directives.

i Intel Hex format. This option causes the object file to be produced in the Intel Hex format rather than the default Motorola Hex format. This option is only available for the Series II since Intel Hex is the default format for the Series I.

l Listing file. This option enables the output to a listing file. The listing produced has page headers and footers inserted at appropriate intervals. Even when the option is invoked, the output to the listing file will still be controlled by the LIST and NOLIST directives.

m Motorola Hex format. This option causes the object file to be produced in the Motorola Hex format rather than the default Intel Hex format. This option is only available for the Series I since Motorola Hex is the default format for the Series II.

o Object file disabled. This option suppresses the output to an object file.

p Page length. This option sets the number of lines per page in the listing file. When a listing file is produced, since page headers and footers are inserted, a page length is required. If this option, immediately followed by a numeric constant, is invoked, the page length will be set to that value. The value must be between 20 and 120 or it will be ignored. Even when this option is invoked, the page length may be set again anywhere in the file by the PLEN directive. The default page length is 66 lines.

q Quiet mode. This option turns off the display of line numbers during assembly. Normally a running count, incremented by 50s, of the numbers of lines being assembled is displayed on the console.

r Record size in the object file. This option sets the number of bytes in each object file record. The number selected is the number of counted bytes in the selected format. The number must be between 8 and 255 or it will be ignored. If the m option is invoked, the r option must appear after it in the command line to have any effect. The default record size in Intel Hex is 24. The default record size in Motorola Hex is 27.

s Software license. This option displays an information screen on the software license.

t Table of symbols suppressed in the listing file. This option suppresses the output of the symbol table in the listing file. Of course if output to the listing file is disabled at the end of the listing, the symbol table will not appear at the end of the listing file anyway.

w Width of the listing file. This option sets the number of characters per line in the listing file. If this option, immediately followed by a numeric constant, is invoked, the line length will be set to that value. The value must be between 60 and 132 or it will be ignored. Even when this option is invoked, the line length may be set again anywhere in the file by the LLEN directive. The default line length is 80 characters.

x Use the extended function register set for the 8052. (only for the 8051 assembler)

Note: Each option must be preceded by a dash (-) in the command line, to distinguish it from a file name.

3.3 Error codes returned to DOS

Condition	**Error code**
Normal completion, without errors	0
Normal completion, with assembly errors	1
Abnormal completion, insufficient memory	2
Abnormal completion, file access error	3

3.4 Object code file formats

This assembler supports two object file formats:

1. Intel Hex
2. Motorola Hex (S1-S9)

4. ASSEMBLER LANGUAGE

4.1 Source line format

Lines in the source file consist of up to four fields, as follows:

[Label] [Opcode] [Operand] [Comment]

Each line of the file may be up to 256 characters long. An arbitrary number of white space (spaces or tabs) must separate each field.

Label

Labels are symbols and follow the same rules. If a label starts in the first column, it must be terminated with at least one white space (space or tab) or the end of the line but it may follow the rules of a label starting in any other column. If a label starts in any other column, it must be terminated with a colon (:) followed with at least one white space (space or tab) or the end of the line. The ending colon is never part of the symbol.

A line may consist of a label alone. With a normal instruction mnemonic, the assembler assigns a value to the label equal to the current location counter.

Opcode

The operand field contains the instruction mnemonic or the assembler directive that is to be performed. This field must not start in the first column.

Operand

The operand field may or may not be required, depending on the opcode or directive being assembled. If present, this field will contain a certain number of operands, separated by commas. There must not be any white space (space or tab) between these operands.

Comment

The optional comment field is used for annotation purposes. This field is ignored by the assembler, but is included in the listing. No character is necessary to designate a comment which does not begin in the first column. For the Series I, a semicolon (;) is used by the assembler to designate the beginning of a comment and it can be used anywhere in the

line, including in the first column for a stand-alone comment. For the Series II, an asterisk (*) may be used in the first column to designate that the whole line is a comment and should be ignored by the assembler.

4.2 Expressions in the operand field

Expressions are combinations of constants, symbols, operators and parentheses that are evaluated as an integer value.

4.2.1 Symbols

Symbols are strings of characters that contain letters (A-Z, a-z), numbers (0-9), points(.) and underscores (_) but they must begin with a letter. A symbol may be of any length, but only the first 8 characters are significant and will be listed in the symbol table. Since all lowercase letter will be converted to uppercase, symbols which are capitalised differently will be the same. A symbol can not be a reserved word.

examples: LOOP
loop_2
key.buffer

The reserved words are the ones representing:

- the mnemonics and assembler directives.
- the microprocessor's registers.

4.2.2 Constants

Numeric constants for the Series I

Numeric constants must always begin with a decimal digit. The radix is determine by a letter immediately following the number according to the following table:

Format	**Suffix**
Binary	B
Octal	O or Q
Decimal	D or nothing
Hexadecimal	H

examples: 10010001b
7774Q

18d
255
6fh
0FEH

Numeric constants for the Series II

Numeric constants must always begin with a prefix followed by the number. The radix is determine by the prefix according to the following table:

Format	**Prefix**
Binary	%
Octal	@
Decimal	nothing
Hexadecimal	$

examples: %10010001
@7774
-18
255
$6fh
$FE

Character constants

A character constant consists of one ASCII character surrounded by single quote marks (').

Note: To put a single quote mark into a character constant, two consecutive single quotes may be used to represent one such character.

examples: 'a'
'*'
''''

The location counter

The current value of the location counter can be used in an expression. It is represented by the dollar sign ($) in the Series I and by an asterisk (*) in the Series II.

4.2.3 Operators

The operators allowed in expressions are shown, in the following table, in descending order of precedence:

Order	Operator	Type	Description
1.	()	expression	parentheses
2.	-	unary	unary negative
	~		one's complement
3.	&	binary	bitwise AND
	!		bitwise OR
	^		bitwise EXCLUSIVE OR
4.	*	multiplicative	multiplication
	/		division
	%		remainder
5.	+	additive	addition
	-		subtraction
6.	<<	shift	shift left
	>>		shift right
7.	=	relational	equal
	<		less than
	>		greater than
	<=		less than or equal
	>=		greater than or equal

4.2.4 How expressions are evaluated

In an expression parenthetical expressions are evaluated first, the innermost parentheses before the outer ones. Next, operations involving higher precedence operators are done first. Where several operators have the same precedence they are evaluated from left to right except for unary operators which are evaluated from right to left. All intermediate values are truncated to a 16-bit integer value. The result of the expression is also a 16-bit integer value.

4.3 Assembler directives

Assembly Control	
ORG	Set assembly origin
END	End of source
REDEF	Redefine an assembler directive
SETDP	Indicate the content of the DP register (only for 6809)
Symbol Definition	
EQU	Equate symbol value
SET	Set symbol value (temporary)
Memory Allocation	
DB	Define constant byte(s)
DW	Define constant word(s)
RSB	Reserve storage for byte(s)
RSW	Reserve storage for word(s)
Listing Control	
TITLE	Set title of page(s)
LLEN	Set line length (characters per line)
PLEN	Set page length (lines per page)
PAGE	Advance to top of next page
LIST	List the assembly
NOLIST	Do not list the assembly
ACLIST	List all object codes for long constants
NOACLIST	Do not list all object codes for long constants
FLIST	List sections that evaluate as false
FNOLIST	Do not list sections that evaluate as false
EXPAND	Expand macros
NOEXPAND	Do not expand macros
Conditional Assembly	
IF	Assemble if not equal to zero
IFN	Assemble if equal to zero
IFC	Assemble if string1 equals string2
IFNC	Assemble if string1 not equal to string2
ELSE	Reverse condition
ENDIF	End of conditional assembly
FAIL	Print error message
Include files	
INCLUDE	Include file
Macro Directives	
MACRO	Define a macro
ENDM	End of macro definition
EXITM	Skip to the end of a macro definition

Table 4.1 Assembler directives

ACLIST

Purpose: This directive restores the listing of the complete object code for long constants. If the object codes generated by the line do not fit on a single line of listing, they will be listed on the following lines.

Format: ACLIST

Remarks: NOACLIST is the complementary directive.

DB

Purpose: This directive allocates and initialises one or more bytes.

Format: [<*label*>] DB <*expression*>[,<*expression*>]...

Remarks: <*expression*> must be an 8-bit integer value or a string surrounded by single quote marks (').

DW

Purpose: This directive allocates and initialises one or more words.

Format: [<*label*>] DW <*expression*>[,<*expression*>]...

Remarks: <*expression*> must be a 16-bit integer value.

ELSE

Purpose: This directive allows the alternate section of code to be assembled when the opposite condition exists in a conditional section.

Format: ELSE

END

Purpose: This directive designates the end of the source file.

Format: END

Remarks: No line in the source file will be read after this directive.

ENDIF

Purpose: This directive designates the end of a conditional section in the code.

Format: ENDIF

Remarks: Each conditional assembly section must end with an ENDIF directive.

ENDM

Purpose: This directive designates the end of a macro definition.

Format: ENDM

Remarks: Each macro definition must end with an ENDM directive.

EQU

Purpose: This directive assigns the value of <*expression*> to <*label*>.

Format: <*label*> EQU <*expression*>

Remarks: The <*label*> entry can not be redefined.

EXITM

Purpose: This directive is used to skip to the end of a macro definition. When this directive is encountered, the expansion is exited immediately and any remaining statement in the macro is not generated.

Format: EXITM

Remarks: An ENDM directive is still required when an EXITM is encountered.

EXPAND

Purpose: This directive lists the complete macro text for all expansions.
Format: EXPAND

Remarks: NOEXPAND is the complementary directive.

FAIL

Purpose: This directive sends the <*message*> string to the console and writes it in the listing file.

Format: FAIL <*message*>

FLIST

Purpose: This directive restores the listing of conditional section that evaluate as false.

Format: FLIST

Remarks: FLIST is the default condition. FNOLIST is the complementary directive.

FNOLIST

Purpose: This directive suppresses the listing of conditional section that evaluate as false.

Format: FNOLIST

Remarks: FLIST is the complementary directive.

IF

Purpose: This directive designates the start of a conditional section. The conditional section will only be assembled if <*expression*> is not equal to zero.

Format: IF <*expression*>

IFC

Purpose: This directive designates the start of a conditional section. The conditional section will only be assembled if <*string1*> is identical to <*string2*>.

Format: IFC <*string1*>,<*string2*>

IFN

Purpose: This directive designates the start of a conditional section. The conditional section will only be assembled if <*expression*> is equal to zero.

Format: IFN <*expression*>

IFNC

Purpose: This directive designates the start of a conditional section. The conditional section will only be assembled if <*string1*> is not identical to <*string2*>.

Format: IFNC <*string1*>,<*string2*>

INCLUDE

Purpose: This directive assembles the source statements from the <*pathname*> file into the current source file.

Format: INCLUDE <*pathname*>

Remarks: The <*pathname*> entry can be any valid file specification as determined by the Disk Operating System. The file is opened and assembled into the current source file immediately following the INCLUDE statement. When the end-of-file is reached, or an END directive is encountered, assembly resumes with the statement following the INCLUDE. There may be up to 16 nested INCLUDE directives. The maximum length of <*pathname*> is 60 characters.

LIST

Purpose: This directive restores the output in the listing.

Format: LIST

Remarks: If a listing is not being made, this directive will start one. NOLIST is the complementary directive.

LLEN

Purpose: This directive sets the number of characters per line, in the listing, to the value of <*expression*>.

Format: LLEN <*expression*>

Remarks: The number of characters per line must be between 60 and 132. The default is 80 characters per line.

MACRO

Purpose: This directive defines a macro called <*label*>. A macro is a sequence of statements that can be generated in various places in the program.

Format: <*label*> MACRO

NOACLIST

Purpose: This directive suppresses the listing of the complete object code for long constants. If the object codes generated by the line do not fit on a single line of listing, they will not all be listed.

Format: NOACLIST

Remarks: NOACLIST is the default condition. ACLIST is the complementary directive.

NOEXPAND

Purpose: This directive suppresses listing of the complete macro text for all expansions.

Format: NOEXPAND

Remarks: NOEXPAND is the default condition. EXPAND is the complementary directive.

NOLIST

Purpose: This directive suppresses the output in the listing.

Format: NOLIST

Remarks: NOLIST is the default condition. LIST is the complementary directive.

ORG

Purpose: This directive sets the location counter to the value of <*expression*>.

Format: ORG <*expression*>

Remarks: <*expression*> must not involved any forward references.

PAGE

Purpose: This directive starts a new page in the listing.

Format: PAGE

PLEN

Purpose: This directive sets the number of lines per page, in the listing, to the value of <*expression*>.

Format: PLEN <*expression*>

Remarks: The number of lines per page must be between 20 and 120. The default is 66 characters per line. The first 3 lines and the last 3 lines of the page are left blank. The fourth and fifth line are used for the date, page number and page title.

REDEF

Purpose: This directive redefines the <*directive*> assembler directive to the <*directive2*> directive.

Format: REDEF <*directive*>,<*directive2*>

Remarks: This allows you to change the name of each and every directive to satisfy your particular source file needs. You can also use it to make your own set of universal directives.

RSB

Purpose: This directive reserves a block of <*expression*> byte(s) for storage.

Format: [<*label*>] RSB <*expression*>

Remarks: The number of bytes reserved must be between 1 and 32767.

RSW

Purpose: This directive reserves a block of <*expression*> word(s) for storage.

Format: [<*label*>] RSW <*expression*>

Remarks: The number of words reserved must be between 1 and 16383.

SET

Purpose: This directive temporarily assigns the value of <*expression*> to <*label*>.

Format: <*label*> SET <*expression*>

Remarks: The <*label*> entry can be redefined.

SETDP

Purpose: This directive indicates that the DP register of the Motorola 6809 is suppose to contain the value <*expression*>.

Format: SETDP <*expression*>

Remarks: This directive only applies to the 6809 Macro Cross Assembler. It is used to determine if the direct addressing mode can be used or not. At the beginning, the DP register content is assumed to be 0.

TITLE

Purpose: This directive sets the title of the current and following page(s) to the <*text*> string.

Format: TITLE <*text*>

Remarks: <*text*> is truncated to the first 60 characters.

4.4 Conditional assembly

All conditional section of code uses the following format:

```
IFxx argument
.
.
.
[ELSE]
.
.
.
ENDIF
```

Note: There may be up to 16 nested conditional sections. Any argument to a conditional must be known in pass 1.

4.5 Macro assembly

A macro is a collection of assembler statements that may appear several times in a program with some optional modifications each time it is used.

To properly use a macro you must:

1. Define the macro using the MACRO and ENDM assembler directives.

2. Invoke the macro by using the name of the macro just as if it were a microprocessor mnemonic. (in the opcode field)

The invocation must be done after the definition. When invoked, the statements in the macro are inserted, starting on the current line, and assembled.

A macro may contain up to 9 parameters. In the macro definition, parameters are called dummy parameters and are represented by two characters. The dummy parameter is formed of a backslash (\) followed by the parameter identification number (1-9).

In the macro invocation, actual parameters are specified in the operand field of the invoking line. These parameters are separated by commas and will replaced their corresponding dummy parameters when the macro is invoked.

Example:

macro definition

```
message   MACRO
          LDA       #\2       ;dummy parameter no 2
          JSR       SET_Y
          LDA       #\1       ;dummy parameter no 1
          JSR       SET_X
          LDX       #\3       ;dummy parameter no 3
          JSR       TEXT
          ENDM
```

macro invocation

```
          message   12,5,mssg1     ;print mssg1 at 12,5
```

In this example, the first actual parameter is 12 and would replace the first dummy parameter \1. The second actual parameter is 5 and would replace the second dummy parameter \2. The third actual parameter is mssg1 and would replace the dummy parameter \3. The actual parameters are all strings that will replace the dummy parameters.

The code assembled would become:

```
          LDA       #5
          JSR       SET_Y
          LDA       #12
          JSR       SET_X
          LDX       #mssg1
          JSR       TEXT
```

Since labels may be required in a macro and since macros, generally, will be invoked more than once, there must be a way to generate distinct labels each time the macro is invoked. This is required to eliminated multiply defined symbols. These symbols require a special syntax to be recognise by the assembler. This syntax is only recognised in macro definitions.

These symbols must end by an underscore (_) followed by a dollar sign ($). Only the first 4 characters of these symbols will be used and a three character number (in hexadecimal) preceded by an underscore will follow these 4 characters to make the symbol.

Example:

macro definition

```
count     MACRO
          DECA
wait_$:   BNE       wait_$
          DECB
          BNE       wait_$
          ENDM
```

code generated the first time the macro is invoked:

```
          DECA
wait_001: BNE       wait_001
          DECB
          BNE       wait_001
```

code generated the second time the macro is invoked:

```
          DECA
wait_002: BNE       wait_002
          DECB
          BNE       wait_002
```

5. ERROR MESSAGES

5.1 Common errors

These errors are reported with the name of the source file where they are detected, and the line number.

Bit number out of range

This occurs when the bit number specified is too big or negative.

Branch out of range

This occurs when the relative branch required is not within the range -128 +127 of the current instruction.

Branch out of the 2Kbytes page

This occurs when the branch required is out of the 2Kbyte page.

Data is not an 8-bit value

This occurs when the data is not an 8-bit integer value.

Different size registers

This occurs when two or more of the specified registers are of different sizes when they have to be of the same size.

Directive is reserved word

This occurs on a try to redefine a directive with a reserved word.

Division by zero attempted

This occurs when a division by zero is attempted in an expression.

Embedded macro not allowed

This occurs on an attempt to define a macro within the definition of another macro.

End of conditional assembly section missing

This occurs when a conditional assembly section does not end with an ENDIF directive.

End of macro missing

This occurs when a macro definition does not end with an ENDM directive.

File name too long

This occurs when the specified file path name is more than 60 characters long.

Forward reference is illegal

This occurs when the expression could not be evaluated in the first pass of the assembly.

Illegal addressing mode

This occurs when the addressing mode does not match any admissible addressing mode for the current opcode.

Illegal bit addressable location

This occurs when the bit addressed location is not a bit addressable special function register or a byte of the bit addressable segment of the 8051 microprocessor.

Illegal bit location

This occurs when the accessed location is not a direct addressed bit in the internal data RAM or a special function register bit.

Illegal block length

This occurs when the specified length for a reserved memory block is out of the specified range.

Illegal character in expression

This occurs when an illegal character is encountered in an expression constant.

Illegal direct location

This occurs when the accessed location is not part of the internal data RAM of the 8051 microprocessor.

Illegal forward macro call

This occurs on an attempt to invoke a macro that is forward defined. All macros must be backward defined.

Illegal interrupt mode

This occurs when an illegal interrupt mode is specified.

Illegal label

This occurs when an illegal character is encountered in a label.

Illegal listing line length

This occurs when the specified listing line length is out of the allowed range.

Illegal listing page length

This occurs when the specified listing page length is out of the allowed range.

Illegal opcode

This occurs when the opcode in the statement is not a valid mnemonic or a valid assembler directive.

Illegal port

This occurs when an illegal port name or number is specified.

Illegal register for indirect access

This occurs when the specified register number is not 0 or 1 in an indirect access.

Illegal shift count

This occurs when the shift count is not between 0 and 15.

Illegal starting address

This occurs when an illegal starting address is specified. It is either too big or negative.

Illegal word register

This occurs when the specified register is not a word register.

Immediate data is not a 3-bit value

This occurs when the specified immediate data is not a 3-bit value. It is probably too big.

Immediate data is not an 8-bit value

This occurs when an immediate data is not an 8-bit integer value.

Label is reserved word

This occurs when the specified label is a reserved word. The mnemonics, special function registers, special function register bits and registers of the microprocessor are all reserved words.

Label required with this directive

This occurs when the directive in the statement requires a label and none is present.

Line too long

This occurs if the line is more than 256 characters long.

Macro already defined

This occurs on an attempt to redefine a macro.

Macro name is reserved word

This occurs when the specified macro name is a reserved word. The mnemonics, special function registers, special function register bits and registers of the microprocessor are all reserved words.

Missing argument in invoked macro

This occurs when an argument required in the macro is not present in the invoking line.

Missing file or bad file

This occurs on an attempt to include a file that does not exist in the specified directory.

Missing operand

This occurs when the opcode or assembler directive requires an operand and none is present.

Not a byte register

This occurs when the specified register is not a byte register.

Not a long word register

This occurs when the specified register is not a long word register.

Not a word register

This occurs when the specified register is not a word register.

Not aligned at a multiple of 4 boundary

This occurs when the specified address is not aligned at a multiple of 4 boundary.

Not aligned at an even byte boundary

This occurs when the specified address is not aligned at an even byte boundary.

Not an even byte address for a word

This occurs when the specified address do not point to an even byte.

Not in conditional section

This occurs when an ELSE or ENDIF is encountered without a previous conditional assembly directive active.

Not in macro definition

This occurs when an EXITM or ENDM is encountered without a previous macro definition active.

Offset is not an 8-bit value

This occurs when the specified offset is not an 8-bit value.

Phase error

This occurs when a label value in the second pass of assembly differs from its value in the first pass.

Same register specified twice

This occurs when the same register is specified more than one in a list or registers.

Symbol multiply defined

This occurs on an attempt to redefine a symbol that can not be redefined.

Syntax error

This occurs when the syntax of the statement does not match any recognisable syntax.

Too many nested conditional assembly sections

This occurs on an attempt to insert a new conditional assembly section within one while the maximum number of nested conditional assembly sections has been reached.

Too many nested files

This occurs on an attempt to include a file within one while the maximum number of nested include has been reached.

Too many nested macros

This occurs on an attempt to invoked a macro within one while the maximum number of nested macros has been reached.

U register and S register together

This occurs when the two stack pointers (U and S) are specified in the push or pull operation.

Undefined symbol

This occurs when a symbol encountered is not defined.

Unknown directive

This occurs when a specified directive is unknown. An unknown directive can not be redefined.

5.2 Warnings

These warnings are generated with the name of the source file where they are detected, and the line number.

Ascii constant is not an 8-bit value

This occurs when an ascii constant is more than one character long.

End directive missing

This occurs when a source file ends without an END directive.

Label ignored

This occurs when the definition of a label is attempted with a directive which does not require (and allow) one.

Numeric constant is not an 16-bit value

This occurs when a numeric constant is not a 16-bit integer value.

5.3 Fatal errors

When one of these errors occur, the assembler sends a message to the console and aborts, returning and error code to DOS.

Error closing input file: "error message"
Error closing listing file: "error message"
Error closing object file: "error message"
Error opening input file: "error message"
Error opening listing file: "error message"
Error opening object file: "error message"
Error reading input file: "error message"
Error writing to listing file: "error message"
Error writing to object file: "error message"

Out of memory for the symbol table
Out of memory for the macro table

6. SPECIAL SECTION ON THE MICROS

6.1 Intel 8044

The reserved words in this assembler are:
A, AB, C, DPTR, R0, R1, R2, R3, R4, R5, R6, R7

Special Function Registers:

The 8044 has Special Function Registers. These constants are words representing the 8044 registers accessible through the use of their addresses in the on-chip RAM.

These are predefined in this assembler:
ACC, B, PSW, SP, DPL, DPH, P0, P1, P2, P3, IP, IE, TMOD, TCON, TH0, TL0, TH1, TL1, FIFO, TBS, TBL, TCB, NSNR, STAD, RFL, RBS, RBL, RCB, SMD, STS, SMD

Special Function Register Bits:

These Special Function Registers have bits which are directly accessible. These constants are words representing the 8044 microprocessor register bits accessible through the use of their addresses in the on-chip RAM.

These are predefined in the assembler:

SFR bit	**Description**
P	PSW bit 0
OV	PSW bit 2
RS0	PSW bit 3
RS1	PSW bit 4
F0	PSW bit 5
AC	PSW bit 6
CY	PSW bit 7
EX0	IE bit 0
ET0	IE bit 1
EX1	IE bit 2
ET1	IE bit 3
ES	IE bit 4
ET2	IE bit 5
EA	IE bit 7
PX0	IP bit 0
PT0	IP bit 1
PX1	IP bit 2
PT1	IP bit 3

PS	IP bit 4
PT2	IP bit 5
IT0	TCON bit 0
IE0	TCON bit 1
IT1	TCON bit 2
IE1	TCON bit 3
TR0	TCON bit 4
TF0	TCON bit 5
TR1	TCON bit 6
TF1	TCON bit 7
ACC.0	Accumulator bit 0
ACC.1	Accumulator bit 1
ACC.2	Accumulator bit 2
ACC.3	Accumulator bit 3
ACC.4	Accumulator bit 4
ACC.5	Accumulator bit 5
ACC.6	Accumulator bit 6
ACC.7	Accumulator bit 7
B.0	Register B bit 0
B.1	Register B bit 1
B.2	Register B bit 2
B.3	Register B bit 3
B.4	Register B bit 4
B.5	Register B bit 5
B.6	Register B bit 6
B.7	Register B bit 7
PSW.0	PSW bit 0
PSW.1	PSW bit 1
PSW.2	PSW bit 2
PSW.3	PSW bit 3
PSW.4	PSW bit 4
PSW.5	PSW bit 5
PSW.6	PSW bit 6
PSW.7	PSW bit 7
P0.0	Port 0 bit 0
P0.1	Port 0 bit 1
P0.2	Port 0 bit 2
P0.3	Port 0 bit 3
P0.4	Port 0 bit 4
P0.5	Port 0 bit 5
P0.6	Port 0 bit 6
P0.7	Port 0 bit 7
P1.0	Port 1 bit 0
P1.1	Port 1 bit 1

P1.2	Port 1 bit 2
P1.3	Port 1 bit 3
P1.4	Port 1 bit 4
P1.5	Port 1 bit 5
P1.6	Port 1 bit 6
P1.7	Port 1 bit 7
P2.0	Port 2 bit 0
P2.1	Port 2 bit 1
P2.2	Port 2 bit 2
P2.3	Port 2 bit 3
P2.4	Port 2 bit 4
P2.5	Port 2 bit 5
P2.6	Port 2 bit 6
P2.7	Port 2 bit 7
P3.0	Port 3 bit 0
P3.1	Port 3 bit 1
P3.2	Port 3 bit 2
P3.3	Port 3 bit 3
P3.4	Port 3 bit 4
P3.5	Port 3 bit 5
P3.6	Port 3 bit 6
P3.7	Port 3 bit 7
IP.0	IP bit 0
IP.1	IP bit 1
IP.2	IP bit 2
IP.3	IP bit 3
IP.4	IP bit 4
IP.5	IP bit 5
IP.6	IP bit 6
IP.7	IP bit 7
IE.0	IE bit 0
IE.1	IE bit 1
IE.2	IE bit 2
IE.3	IE bit 3
IE.4	IE bit 4
IE.5	IE bit 5
IE.6	IE bit 6
IE.7	IE bit 7
TCON.0	TCON bit 0
TCON.1	TCON bit 1
TCON.2	TCON bit 2
TCON.3	TCON bit 3
TCON.4	TCON bit 4
TCON.5	TCON bit 5

TCON.6	TCON bit 6
TCON.7	TCON bit 7
RBP	STS bit 0
AM	STS bit 1
CPB	STS bit 2
BV	STS bit 3
SI	STS bit 4
RTS	STS bit 5
RE	STS bit 6
TBF	STS bit 7
SER	NSNR bit 0
NR0	NSNR bit 1
NR1	NSNR bit 2
NR2	NSNR bit 3
SES	NSNR bit 4
NS0	NSNR bit 5
NS1	NSNR bit 6
NS2	NSNR bit 7
STS.0	STS bit 0
STS.1	STS bit 1
STS.2	STS bit 2
STS.3	STS bit 3
STS.4	STS bit 4
STS.5	STS bit 5
STS.6	STS bit 6
STS.7	STS bit 7
NSNR.0	NSNR bit 0
NSNR.1	NSNR bit 1
NSNR.2	NSNR bit 2
NSNR.3	NSNR bit 3
NSNR.4	NSNR bit 4
NSNR.5	NSNR bit 5
NSNR.6	NSNR bit 6
NSNR.7	NSNR bit 7

Note: Consult a data book on the 8044 microprocessor architecture for more details on the special function registers.

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

DATA	instead of EQU
ORG	
END	
DB	
DW	
SET	
DS	instead of RSB
DSW	instead of RSW

6.2 Intel 8048

The reserved words in this assembler are:
A, R0, R1, R2, R3, R4, R5, R6, R7, @R0, @R1, BUS, P1, P2, P4, P5, P6, P7, C, F0, F1, I, TCNTI, CLK, @A, PSW, T, MB0, MB1, RB0, RB1, TCNT, CNT

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

DATA	instead of EQU
ORG	
END	
DB	
DW	
SET	
DS	instead of RSB
RSW	is not supported, because it is not necessary

6.3 Intel 8051/52

The reserved words in this assembler are:
A, AB, C, DPTR, R0, R1, R2, R3, R4, R5, R6, R7

Special Function Registers:

The 8051 has Special Function Registers. These constants are words representing the 8051 microprocessor registers accessible through the use of their addresses in the on-chip RAM.

These are predefined in this assembler:
ACC, B, PSW, SP, DPL, DPH, P0, P1, P2, P3, IP, IE, TMOD, TCON, TH0, TL0, TH1, TL1, SCON, SBUF, PCON

If the extended special function register set option is invoked (x option), the 8052 microprocessor additional special function registers are also predefined in the assembler.

These are:
T2CON, TH2, TL2, RCAP2H, RCAP2L

Special Function Register Bits:

These Special Function Registers have bits which are directly accessible. These constants are words representing the 8051 microprocessor register bits accessible through the use of their addresses in the on-chip RAM.

These are predefined in the assembler:

SFR bit	**Description**
P	PSW bit 0
OV	PSW bit 2
RS0	PSW bit 3
RS1	PSW bit 4
F0	PSW bit 5
AC	PSW bit 6
CY	PSW bit 7
EX0	IE bit 0
ET0	IE bit 1
EX1	IE bit 2
ET1	IE bit 3
ES	IE bit 4
ET2	IE bit 5
EA	IE bit 7

PX0	IP bit 0
PT0	IP bit 1
PX1	IP bit 2
PT1	IP bit 3
PS	IP bit 4
PT2	IP bit 5
IT0	TCON bit 0
IE0	TCON bit 1
IT1	TCON bit 2
IE1	TCON bit 3
TR0	TCON bit 4
TF0	TCON bit 5
TR1	TCON bit 6
TF1	TCON bit 7
RI	SCON bit 0
TI	SCON bit 1
RB8	SCON bit 2
TB8	SCON bit 3
REN	SCON bit 4
SM2	SCON bit 5
SM1	SCON bit 6
SM0	SCON bit 7
RXD	Port 3 bit 0
TXD	Port 3 bit 1
INT0	Port 3 bit 2
INT1	Port 3 bit 3
T0	Port 3 bit 4
T1	Port 3 bit 5
WR	Port 3 bit 6
RD	Port 3 bit 7
ACC.0	Accumulator bit 0
ACC.1	Accumulator bit 1
ACC.2	Accumulator bit 2
ACC.3	Accumulator bit 3
ACC.4	Accumulator bit 4
ACC.5	Accumulator bit 5
ACC.6	Accumulator bit 6
ACC.7	Accumulator bit 7
B.0	Register B bit 0
B.1	Register B bit 1
B.2	Register B bit 2
B.3	Register B bit 3
B.4	Register B bit 4
B.5	Register B bit 5

B.6	Register B bit 6
B.7	Register B bit 7
PSW.0	PSW bit 0
PSW.1	PSW bit 1
PSW.2	PSW bit 2
PSW.3	PSW bit 3
PSW.4	PSW bit 4
PSW.5	PSW bit 5
PSW.6	PSW bit 6
PSW.7	PSW bit 7
P0.0	Port 0 bit 0
P0.1	Port 0 bit 1
P0.2	Port 0 bit 2
P0.3	Port 0 bit 3
P0.4	Port 0 bit 4
P0.5	Port 0 bit 5
P0.6	Port 0 bit 6
P0.7	Port 0 bit 7
P1.0	Port 1 bit 0
P1.1	Port 1 bit 1
P1.2	Port 1 bit 2
P1.3	Port 1 bit 3
P1.4	Port 1 bit 4
P1.5	Port 1 bit 5
P1.6	Port 1 bit 6
P1.7	Port 1 bit 7
P2.0	Port 2 bit 0
P2.1	Port 2 bit 1
P2.2	Port 2 bit 2
P2.3	Port 2 bit 3
P2.4	Port 2 bit 4
P2.5	Port 2 bit 5
P2.6	Port 2 bit 6
P2.7	Port 2 bit 7
P3.0	Port 3 bit 0
P3.1	Port 3 bit 1
P3.2	Port 3 bit 2
P3.3	Port 3 bit 3
P3.4	Port 3 bit 4
P3.5	Port 3 bit 5
P3.6	Port 3 bit 6
P3.7	Port 3 bit 7
IP.0	IP bit 0
IP.1	IP bit 1

IP.2	IP bit 2
IP.3	IP bit 3
IP.4	IP bit 4
IP.5	IP bit 5
IP.6	IP bit 6
IP.7	IP bit 7
IE.0	IE bit 0
IE.1	IE bit 1
IE.2	IE bit 2
IE.3	IE bit 3
IE.4	IE bit 4
IE.5	IE bit 5
IE.6	IE bit 6
IE.7	IE bit 7
TCON.0	TCON bit 0
TCON.1	TCON bit 1
TCON.2	TCON bit 2
TCON.3	TCON bit 3
TCON.4	TCON bit 4
TCON.5	TCON bit 5
TCON.6	TCON bit 6
TCON.7	TCON bit 7
SCON.0	SCON bit 0
SCON.1	SCON bit 1
SCON.2	SCON bit 2
SCON.3	SCON bit 3
SCON.4	SCON bit 4
SCON.5	SCON bit 5
SCON.6	SCON bit 6
SCON.7	SCON bit 7

If the extended special function register set option is invoked (x option), the 8052 microprocessor additional special function register bits are also predefined in the assembler.

These are:

SFR bit	**Description**
CPRL2	T2CON bit 0
CT2	T2CON bit 1
TR2	T2CON bit 2
EXEN2	T2CON bit 3
TCLK	T2CON bit 4
RCLK	T2CON bit 5

EXF2	T2CON bit 6
TF2	T2CON bit 7
T2CON.0	T2CON bit 0
T2CON.1	T2CON bit 1
T2CON.2	T2CON bit 2
T2CON.3	T2CON bit 3
T2CON.4	T2CON bit 4
T2CON.5	T2CON bit 5
T2CON.6	T2CON bit 6
T2CON.7	T2CON bit 7

Note: Consult a data book on the 8051, 8052 microprocessor architectures for more details on the special function registers.

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

DATA	instead of EQU
ORG	
END	
DB	
DW	
SET	
DS	instead of RSB
DSW	instead of RSW

6.4 Intel 8080

The reserved words in this assembler are:
A, B, C, D, E, H, L, M, BC, DE, HL, SP

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
DB	
DW	
SET	
DS	instead of RSB
RSW	is not supported, because it is not necessary

6.5 Intel 8085

The reserved words in this assembler are:
A, B, C, D, E, H, L, M, BC, DE, HL, SP

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:
EQU
ORG
END
DB
DW
SET
DS instead of RSB
RSW is not supported, because it is not necessary

6.6 Intel 8096

There are no reserved words in this assembler.

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
DCB	instead of DB
DCW	instead of DW
SET	
DSB	instead of RSB
DSW	instead of RSW

6.7 Motorola 6800

The reserved word in this assembler is:
X

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
FCB	instead of DB
FDB	instead of DW
FCC	a second definition of DB
SET	
RMB	instead of RSB
RSW	is not supported, because it is not necessary

6.8 Motorola 6803

The reserved word in this assembler is:
X

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
FCB	instead of DB
FDB	instead of DW
FCC	a second definition of DB
SET	
RMB	instead of RSB
RSW	is not supported, because it is not necessary

6.9 Motorola 6805

The reserved word in this assembler is:
X

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
FCB	instead of DB
FDB	instead of DW
FCC	a second definition of DB
SET	
RMB	instead of RSB
RSW	is not supported, because it is not necessary

6.10 Motorola 6809

The reserved words in this assembler are:
A, B, CC, D, DP, PC, S, U, X, Y

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
FCB	instead of DB
FDB	instead of DW
FCC	a second definition of DB
SET	
RMB	instead of RSB
RSW	is not supported, because it is not necessary
SETDP	(special and reserved to the 6809)

Note: If the operand is preceded by a '<' character, it forces direct addressing.

6.11 Rockwell 6502

The reserved words in this assembler are:
A, X, Y, P, PC, SP

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
BYTE	instead of DB
WORD	instead of DW
SET	
RESERVE	instead of RSB
RSW	is not supported, because it is not necessary

6.12 Rockwell 65C02

The reserved words in this assembler are:
A, X, Y, P, PC, SP

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
BYTE	instead of DB
WORD	instead of DW
SET	
RESERVE	instead of RSB
RSW	is not supported, because it is not necessary

6.13 Zilog Z80

The reserved words in this assembler are:
A, B, C, D, E, H, L, BC, DE, HL, SP, IX, IY, M, NC, NZ, Z, P, PE, PO, AF, I, R

Assembler directives:

For this microprocessor, the standard assembler directives have been predefined as following:

EQU	
ORG	
END	
DEFB	instead of DB
DEFW	instead of DW
DEFL	instead of SET
DEFS	instead of RSB
RSW	is not supported, because it is not necessary

7. BUG REPORTING PROCEDURE

Although each software is extensively tested prior to release, no software is perfect and may contain a few bugs. It is the intention of Micept Instruments Inc. to correct any genuine problem that will be considered as a problem by the company, although this is not a warranty that any problem will be corrected.

If you have any comments or suggestions, please send them to the following address or e-mail them to us.

If you think you have found a bug in this software, please take the time to report it. Any report will be helpful, but you should follow these directions:

1. Provide the name and version of the software.

2. Provide a brief description of the problem and provide a copy or listing of the problem source file. Please try to isolate the problem area in the source and only provide that part of the source file.

3. Send it to:

Micept Instruments Inc
377 Julien St.
Cap De La Madeleine, PQ
Canada G8T 6W6

www: http://www3.sympatico.ca/jveillet/micept
e-mail: mailto: jveillet@sympatico.ca

8. UPDATE POLICY

Micept Instruments Inc. will not notify users of new version releases as improvements are made constantly. If registered users would like to keep current they may order the latest version of the Macro Cross Assembler(s). No update policy is necessary since the software price is so low.

9. LICENSE AGREEMENT

xxxxx Macro Cross Assembler (the "Software") is copyright 1991-1996 by Micept Instruments Inc., and is protected by the Canadian Copyright Act and International Treaty provisions. All rights are reserved. You ("The Original Purchaser") is granted a license to use the Software only, subject to the following restrictions and limitations.

1. The license is to the Original Purchaser only, and is not transferable without the written permission of Micept Instruments Inc.

2. You may use the Software on a single computer. You may not use the Software on more than a single machine without the written consent of Micept Instruments Inc.

3. You may make back-up copies of the Software for your own use only, subject to the limitations of this license.

4. You may not engage in, nor engage third parties to engage in, any of the following:

   a) Providing or permitting use of or disclosing the Software to third parties.

   b) Providing use of the Software in a network, timesharing, multiple CPU or multiple user arrangement to users who are not individually licensed by Micept Instruments Inc.

   c) Making alterations or copies of any kind in the Software (except as specifically permitted above)

   d) Attempting to modify, decompile, disassemble or reverse engineer the Software in any way.

   e) Granting sub-licenses, lease or other rights in the Software to others.

   f) Making telecommunication data transmission of the Software.

Micept Instruments Inc. reserves the right to terminate this license if there is a violation of its term or default by the Original Purchaser. Upon termination for any reason, all copies of the Software must be immediately returned to Micept Instruments Inc. and the Original Purchaser shall be liable for any and all damages suffered as a result of the violation or default.

This agreement shall be construed, interpreted and governed by the laws of the Province of Quebec, Canada.

10. ORDERING INFORMATION

To register please use the order form.

The registration fee will licence one copy of the software that you will receive, for use on any one computer at any one time. See the LICENSE AGREEMENT section.

Site License:

If you plan on using this software in a corporation, government office, or for any business purpose, quantity discounts are available under a Site License agreement, described below:

Site license discount rates are as follows:

Total copies	**Discount**
11 - 20	5%
21 - 30	10%
31 - 40	15%
41 - 50	20%
51 - 100	25%
101+	30%

Note: All prices and discounts are subject to change without notice. Discounts are not cumulative; they apply to separate orders only.

Update:

To get the latest version, just order the software again. The price is so affordable.

Please use the third line of the order form to order the followings:

25 MA09. Site Registration @ $18.00 ea $ _____
(Example of a site registration, 10% discount)

Cat. No.	Description	License Fee
	Series I	
MA44	Macro Cross Assembler for the Intel 8044	$20.00
MA48	Macro Cross Assembler for the Intel 8048	$20.00
MA51	Macro Cross Assembler for the Intel 8051/52	$20.00
MA80	Macro Cross Assembler for the Intel 8080	$20.00
MA85	Macro Cross Assembler for the Intel 8085	$20.00
MA96	Macro Cross Assembler for the Intel 8096	$20.00
MAZ80	Macro Cross Assembler for the Zilog Z80	$20.00
SERIES-I	Complete series containing the Macro Cross Assemblers for the Intel 8044, 8048, 8051/52, 8080, 8085, 8096 and the Zilog Z80 (manual on disk)	$99.00
	Series II	
MA03	Macro Cross Assembler for the Motorola 6803	$20.00
MA05	Macro Cross Assembler for the Motorola 6805	$20.00
MA09	Macro Cross Assembler for the Motorola 6809	$20.00
MA65	Macro Cross Assembler for the Rockwell 6502	$20.00
MA65C	Macro Cross Assembler for the Rockwell 65C02	$20.00
MA68	Macro Cross Assembler for the Motorola 6800	$20.00
SERIES-II	Complete series containing the Macro Cross Assemblers for the Motorola 6800, 6803, 6805, 6809, and the Rockwell 6502 and 65C02 (manual on disk)	$79.00
	Complete package	
MICRO-8	Package containing all the Macro Cross Assemblers of the series I and II (manual on disk)	$139.00

* Prices are in U.S. dollars. We accept only U.S. and Canadian funds. All other will be refused. Prices and availability are subject to change without notice.

ORDER FORM

Date: _____ #99-MC225-MA2

Remit to: Micept Instruments Inc.
377 Julien St.
Cap De La Madeleine, PQ
Canada G8T 6W6

Qty	Cat. No.	Description	Unit Price	Price

Subtotal	
Add $6.00 for Shipping and Handling (outside North America Add $10.00)	
Canada residents Add GST	
Subtotal	
Quebec residents Add Provincial Sales Tax	
Total (U.S. Dollars)	

Payments in Canadian dollars: Exchange Rate _____ : Total Can. dollars _____

Payment by: ( ) Check ( ) Money Order

Name _____
Company _____
Address _____
_____
_____

Phone: (day) _____ (evening) _____
Signature _____ Date _____

Where did you get your demo copy of this software? _____

APPENDIX A - SAMPLE LISTING FILES

```
8051 Macro Cross Assembler  2.21                                        11/09/92   page    1
Example of a source file for the Intel 8051                                         14:33:57

                    00001    ;
                    00002    ; Example of a source file for the Intel 8051
                    00003    ; Copyright (C) 1992 by Micept Instruments Inc.
                    00004    ; Version 1.00
                    00005    ;
                    00006    ; This code as no purpose whatsoever except to
                    00007    ; show the use of Macros, conditional assembly,
                    00008    ; the assembler directives and the addressing
                    00009    ; modes of the microprocessor.
                    00010    ;
                    00011              PLEN     70
                    00012              LLEN     100
                    00013              TITLE   Example of a source file for the Intel 8051
                    00014              REDEF    SET,SETV ;Redefine the SET directive
                    00015    ;
                    00016    ; Definitions
                    00017    ;
   = 8000           00018    DATAMEM   DATA     8000h
   = C000           00019    ROM       DATA     0C000H
   = B800           00020    PORTA     DATA     0B800h
   = B801           00021    PORTB     DATA     PORTA+1
   = 000D           00022    CR        DATA     13
   = 000A           00023    LF        DATA     00001010b
   = 0004           00024    NKEY      SETV     8*2>>2
                    00025    ;
                    00026    ; Data
                    00027    ;
8000                00028              ORG      DATAMEM
8000                00029    KEYB      DS       16       ;16 bytes reserved for the key buffer
                    00030    ;
                    00031    ; Macro to display text - macro definition
                    00032    ;
                    00033    TEXT      MACRO
                    00034              IFNC     \3,NOTHING
                    00035              IF       \2       ;New row?
                    00036              LCALL    SETY     ;Position the cursor in the row
                    00037              ENDIF
                    00038              IF       \1       ;New column?
                    00039              LCALL    SETX     ;Position the cursor in the column
                    00040              ENDIF
                    00041              ELSE
                    00042              EXITM
                    00043              ENDIF
                    00044              MOV      DPTR,#\3
                    00045              MOVC     A,@A+DPTR
                    00046              JZ       ENDT_$
                    00047              LCALL    CHROUT
                    00048    ENDT_$:
                    00049              ENDM
```

```
8051 Macro Cross Assembler  2.21                                          11/09/92   page    2
Example of a source file for the Intel 8051                                          14:33:57

                    00050               PAGE
                    00051    ;
                    00052    ; Main routine
                    00053    ;
C000                00054               ORG       ROM
C000  00            00055    START      NOP                ;Demonstration of addressing modes
C001  74 31         00056               MOV       A,#'a'-'0'
C003  E9            00057               MOV       A,R1
C004  F6            00058               MOV       @R0,A
C005  04            00059               INC       A
C006  12 C020       00060               LCALL     CHROUT
C009  90 ABCD       00061               MOV       DPTR,#0ABCDH
                    00062    ;
C00C                00063               TEXT      10,5,HELLO ;Macro invocation
C01B  74 C0         00064               MOV       A,#0C0H
C01D  02 C000       00065               LJMP      START
                    00066    ;
                    00067    ; Subroutine to send a character
                    00068    ;
C020  E4            00069    CHROUT:    CLR       A
C021  90 B800       00070               MOV       DPTR,#PORTA
C024  93            00071               MOVC      A,@A+DPTR
C025  54 01         00072               ANL       A,#1
C027  70 F7         00073               JNZ       CHROUT
C029  22            00074               RET
                    00075    ;
                    00076    ; Subroutine to set the row on the terminal
                    00077    ;
C02A  C0 E0         00078    SETY       PUSH      ACC
C02C  75 F0 1B      00079               MOV       B,#1Bh
C02F  12 C020       00080               LCALL     CHROUT
C032  75 F0 00      00081               MOV       B,#0H
C035  12 C020       00082               LCALL     CHROUT
C038  D0 F0         00083               POP       B
C03A  12 C020       00084               LCALL     CHROUT
C03D  22            00085               RET
                    00086    ;
                    00087    ; Subroutine to set the column on the terminal
                    00088    ;
C03E  C0 E0         00089    SETX       PUSH      ACC
C040  75 F0 1B      00090               MOV       B,#1BH
C043  12 C020       00091               LCALL     CHROUT
C046  75 F0 01      00092               MOV       B,#1
C049  12 C020       00093               LCALL     CHROUT
C04C  D0 F0         00094               POP       B
C04E  12 C020       00095               LCALL     CHROUT
C051  22            00096               RET
                    00097    ;
C052  30 31 32 33   00098    DIGCODE    DB        '0','1','2','3','4','5','6','7','8','9',0
C05D  CC DD E0      00099    MDATA      DB        0CCH,0DDH,0E0H
C060  C03E C02A     00100    GOTODT     DW        SETX,SETY,0
C066  48 65 6C 6C   00101    HELLO      DB        'Hello,  this is only a test!',cr,lf,0
                    00102               END

Assembly completed
102 lines
Time elapsed: 0 seconds
No errors detected
No warnings generated
```

```
8051 Macro Cross Assembler  2.21                                        11/09/92   page    3
Example of a source file for the Intel 8051                                         14:33:57

Symbol table:

AC       00D6   ACC      00E0   ACC.0    00E0   ACC.1    00E1   ACC.2    00E2   ACC.3    00E3
ACC.4    00E4   ACC.5    00E5   ACC.6    00E6   ACC.7    00E7   B        00F0   B.0      00F0
B.1      00F1   B.2      00F2   B.3      00F3   B.4      00F4   B.5      00F5   B.6      00F6
B.7      00F7   CHROUT   C020   CR       000D   CY       00D7   DATAMEM  8000   DIGCODE  C052
DPH      0083   DPL      0082   EA       00AF   ENDT_001 C01B   ES       00AC   ET0      00A9
ET1      00AB   EX0      00A8   EX1      00AA   F0       00D5   GOTODT   C060   HELLO    C066
IE       00A8   IE.0     00A8   IE.1     00A9   IE.2     00AA   IE.3     00AB   IE.4     00AC
IE.5     00AD   IE.6     00AE   IE.7     00AF   IE0      0089   IE1      008B   INT0     00B2
INT1     00B3   IP       00B8   IP.0     00B8   IP.1     00B9   IP.2     00BA   IP.3     00BB
IP.4     00BC   IP.5     00BD   IP.6     00BE   IP.7     00BF   IT0      0088   IT1      008A
KEYB     8000   LF       000A   MDATA    C05D   NKEY     0004   OV       00D2   P        00D0
P0       0080   P0.0     0080   P0.1     0081   P0.2     0082   P0.3     0083   P0.4     0084
P0.5     0085   P0.6     0086   P0.7     0087   P1       0090   P1.0     0090   P1.1     0091
P1.2     0092   P1.3     0093   P1.4     0094   P1.5     0095   P1.6     0096   P1.7     0097
P2       00A0   P2.0     00A0   P2.1     00A1   P2.2     00A2   P2.3     00A3   P2.4     00A4
P2.5     00A5   P2.6     00A6   P2.7     00A7   P3       00B0   P3.0     00B0   P3.1     00B1
P3.2     00B2   P3.3     00B3   P3.4     00B4   P3.5     00B5   P3.6     00B6   P3.7     00B7
PCON     0087   PORTA    B800   PORTB    B801   PS       00BC   PSW      00D0   PSW.0    00D0
PSW.1    00D1   PSW.2    00D2   PSW.3    00D3   PSW.4    00D4   PSW.5    00D5   PSW.6    00D6
PSW.7    00D7   PT0      00B9   PT1      00BB   PX0      00B8   PX1      00BA   RB8      009A
RD       00B7   REN      009C   RI       0098   ROM      C000   RS0      00D3   RS1      00D4
RXD      00B0   SBUF     0099   SCON     0098   SCON.0   0098   SCON.1   0099   SCON.2   009A
SCON.3   009B   SCON.4   009C   SCON.5   009D   SCON.6   009E   SCON.7   009F   SETX     C03E
SETY     C02A   SM0      009F   SM1      009E   SM2      009D   SP       0081   START    C000
T0       00B4   T1       00B5   TB8      009B   TCON     0088   TCON.0   0088   TCON.1   0089
TCON.2   008A   TCON.3   008B   TCON.4   008C   TCON.5   008D   TCON.6   008E   TCON.7   008F
TF0      008D   TF1      008F   TH0      008C   TH1      008D   TI       0099   TL0      008A
TL1      008B   TMOD     0089   TR0      008C   TR1      008E   TXD      00B1   WR       00B6
```

```
6809 Macro Cross Assembler  2.20                                        02/15/92   page    1
Example of a source file for the Motorola 6809                                      13:36:39

                    00001    *
                    00002    * Example of a source file for the Motorola 6809
                    00003    * Copyright (C) 1992 by Micept Instruments Inc.
                    00004    * Version 1.00
                    00005    *
                    00006    * This code as no purpose whatsoever except to
                    00007    * show the use of Macros, conditional assembly,
                    00008    * the assembler directives and the addressing
                    00009    * modes of the microprocessor.
                    00010    *
                    00011              PLEN     70
                    00012              LLEN     100
                    00013              TITLE   Example of a source file for the Motorola 6809
                    00014              REDEF    SET,SETV ;Redefine the SET directive
                    00015    *
                    00016    * Definitions
                    00017    *
   = 8000           00018    DATAMEM   EQU      $8000
   = C000           00019    ROM       EQU      $C000
   = B800           00020    PORTA     EQU      $B800
   = B801           00021    PORTB     EQU      PORTA+1
   = 000D           00022    CR        EQU      13
   = 000A           00023    LF        EQU      %00001010
   = 0004           00024    NKEY      SETV     8*2>>2
                    00025    *
                    00026    * Data
                    00027    *
8000                00028              ORG      DATAMEM
8000                00029    KEYB      RMB      16       ;16 bytes reserved for the key buffer
                    00030    *
                    00031    * Macro to display text - macro definition
                    00032    *
                    00033    TEXT      MACRO
                    00034              IFNC     \3,NOTHING
                    00035              IF       \2       ;New row?
                    00036              JSR      SETY     ;Position the cursor in the row
                    00037              ENDIF
                    00038              IF       \1       ;New column?
                    00039              JSR      SETX     ;Position the cursor in the column
                    00040              ENDIF
                    00041              ELSE
                    00042              EXITM
                    00043              ENDIF
                    00044              LDX      \3
                    00045              LDB      ,X+
                    00046              BEQ      ENDT_$
                    00047              JSR      CHROUT
                    00048    ENDT_$:
                    00049              ENDM
```

```
6809 Macro Cross Assembler  2.20                                        02/15/92   page    2
Example of a source file for the Motorola 6809                                      13:36:39

                    00050              PAGE
                    00051    *
                    00052    * Main routine
                    00053    *
C000                00054              ORG       ROM
C000  12            00055    START     NOP                ;Demonstration of addressing modes
C001  86 31         00056              LDA       #'a'-'0'
C003  BE 123F       00057              LDX       $123F
C006  30 01         00058              LEAX      1,X
C008  8D 2E         00059              BSR       CHROUT
C00A  1F 89         00060              TFR       A,B
C00C  34 47         00061              PSHS      A,B,U,CC
C00E  A6 C4         00062              LDA       ,U
C010  E6 F4         00063              LDB       [,S]
C012  EC 98 F6      00064              LDD       [-10,X]
C015  10AE B6       00065              LDY       [A,Y]
C018  31 83         00066              LEAY      ,--X
C01A  E4 9F C075    00067              ANDB      [HELLO]
                    00068    *
C01E                00069              TEXT      10,5,HELLO ;Macro invocation
C0                  00070              SETDP     $C0       ;New DP - let's tell the assembler
C02E  86 C0         00071              LDA       #$C0
C030  1F B8         00072              TFR       DP,A
C032  B6 C017       00073              LDA       $C017
C035  16 FFC8       00074              LBRA      START
                    00075    *
                    00076    * Subroutine to send a character
                    00077    *
C038  B6 B801       00078    CHROUT:   LDA       PORTB
C03B  84 01         00079              ANDA      #1
C03D  26 F9         00080              BNE       CHROUT
C03F  F7 B800       00081              STB       PORTA
C042  39            00082              RTS
                    00083    *
                    00084    * Subroutine to set the row on the terminal
                    00085    *
C043  34 02         00086    SETY      PSHS      A
C045  C6 1B         00087              LDB       #$1B
C047  8D EF         00088              BSR       CHROUT
C049  C6 00         00089              LDB       #$0
C04B  8D EB         00090              BSR       CHROUT
C04D  35 04         00091              PULS      B
C04F  8D E7         00092              BSR       CHROUT
C051  39            00093              RTS
                    00094    *
                    00095    * Subroutine to set the column on the terminal
                    00096    *
C052  34 02         00097    SETX      PSHS      A
C054  C6 1B         00098              LDB       #$1B
C056  8D E0         00099              BSR       CHROUT
C058  C6 01         00100              LDB       #$1
C05A  8D DC         00101              BSR       CHROUT
C05C  35 04         00102              PULS      B
C05E  8D D8         00103              BSR       CHROUT
C060  39            00104              RTS
                    00105    *
C061  30 31 32 33   00106    DIGCODE   FCB       '0','1','2','3','4','5','6','7','8','9',0
C06C  CC DD E0      00107    MDATA     FCB       $CC,$DD,$E0
C06F  C052 C043     00108    GOTODT    FDB       SETX,SETY,0
C075  48 65 6C 6C   00109    HELLO     FCC       'Hello,  this is only a test!',cr,lf,0
                    00110              END
```

```
6809 Macro Cross Assembler  2.20                                02/15/92   page    3
Example of a source file for the Motorola 6809                               13:36:40


Assembly completed
110 lines
Time elapsed: 2 seconds
No errors detected
No warnings generated
```

```
6809 Macro Cross Assembler  2.20                                        02/15/92   page    4
Example of a source file for the Motorola 6809                                      13:36:40

Symbol table:

CHROUT   C038   CR       000D   DATAMEM  8000   DIGCODE  C061   ENDT_001 C02E   GOTODT   C06F
HELLO    C075   KEYB     8000   LF       000A   MDATA    C06C   NKEY     0004   PORTA    B800
PORTB    B801   ROM      C000   SETX     C052   SETY     C043   START    C000
```

INDEX

Notes

Notes