2500AD Macro Cross Assemblers

The Macro Cross Assembler

- The 2500AD Macro Cross Assembler can assemble programs into relocatable object code which can then be linked to the desired execution address using the 25OOAD Linker.
- Each Assembler includes the Macro Cross Assembler, Linker and Librarian.

Inside The Assembler

- The Assembler supports pathnames for input files, output files and include files. The Assembler has built-in cross referencing which includes an alphabetized symbol table. The symbol table indicates the line number on which the label is defined, the value of the label and all line numbers where the label is referenced.
- With Series 5, there is no limit to the program size that can be assembled and linked. There is no limit to the number of symbols and macros.
- A macro may be defined by using the Macro directive. The Assembler will store the macro definition, and upon encountering the macro name, will substitute the previously defined source lines. Arguments may be included in the Macro definition.
- The Assembler and Linker may be run in either prompt mode or command line mode.
- Labels, Macro Names, Register Names and User Defined Section Names may be up to 32 characters in length.
- The Conditional Assembly directives enable the user to direct the Assembler to process different sections in the source file. Conditionals may be nested 248 levels. The Assembler aids in detection of conditional nesting errors by checking for unbalanced conditional levels and displaying the current active conditional level in the listing's object code field.
- Calculations with up to 16 pending operands may be processed. The Assembler does 80 bit arithmetic on all operands. The algebraic hierarchy of calculations may be changed with the use of parentheses.
- The Linker creates an output code file format from one or many object files made by the Assembler. The object files can be put in the output file, used for reference purposes only or linked indirectly to run at an address other than the load address.
- The Linker outputs Executable Binary, Intel Hex, Extended Intel Hex, Tektronix Hex, Motorola S19, S28 and S37 file formats. 25OOAD Global, Abbreviated Global, High Level, Rockwell, Microtek, Extended Microtek and Zax symbol tables are also supported.
- Sections may be linked to a specific address or stacked on top of each previous identically named section. Any section in any file may be used for reference only.
- The Librarian is used to create a library of object modules. The libraries can be linked with any program created by the Assembler. Useful routines can be archived for reuse.

Special Features

Assembly Language Syntax

Number Bases

The Assembler's default number base is 10. The default prefix "$" designates a hex numeral for Motorola processors and the default prefix "%" designates a hex numeral for Zilog processors.

Number Base	Designator
Binary	**B**
Octal	O or Q
Decimal	D or none
Hexadecimal	H
Ascii String	Double or single quotes " " or ' '

Program Comments

Comments may have their own line or follow source code on the same line. ***Comments*** can start with a ";" or "*".

Labels

All label names are case-sensitive.

Non-local labels can have any number of characters, but only 32 are significant. A label can start in any column if its name ends with a colon.

Local Labels

Local Labels are used much the same as ***non-local*** *labels,* but the definition of a local label is only valid between ***non-local labels.*** A "local area" is one bounded by labels which keep their definition throughout the entire program.

For compatibility with manufacturers' assemblers, ***local labels can*** be identified with a leading or trailing *local* ***label character.*** Additionally, local ***label characters*** may be changed to provide compatibility with existing source.

High And Low Byte

The high byte of a 16 bit value,bits 8 through 15, can be saved and used as a relocatable byte value.

The low ***byte*** of a 16 bit value, bits 0 through 7, can be saved and used as a relocatable byte value.

Upper And Lower Case

Assembler directives are not case-sensitive. They can be typed in upper or lower case, or in a combination of both.

All labels, including macro and section names, are case-sensitive.

Assembler directives can start with an optional "." to differentiate them from instructions.

Assembly Time Calculations

The Assembler permits several assembly time operators for doing calculations and comparisons with each being prioritized. Parenthesis can be used to change priority.

All calculations use 80 bit integer arithmetic except exponentiation which uses an 8 bit exponent. There may be 16 pending operations.

Assembler Directives

Storage Control

ASCII

Stores a string in memory up to, but not including, either a carriage return or a broken bar character ("I", Hex 7C).

BLKB

Reserves a specified number of bytes and stores a value in each byte.

BLKL

Reserves a specified number of 32 bit long words and stores a value in each long word.

BLKW

Reserves a specified number of 16 bit words and stores a value in each word.

BYTE, DB, DEFB, FCB, STRING

Stores a value in consecutive memory locations. The value may be any mix of operand types.

DATE

Stores the current date in Ascii.

DC

Stores a value in consecutive memory locations. However, DC also sets the high bit of the last data byte. DC is useful for storing strings since the setting of bit 7 flags the end of the string.

DDW

Stores a 32-bit linear numeric value in a 32-bit storage location.

DEFS, DS, RMB

Reserves a specified number of bytes, but stores nothing in the reserved area. If the storage locations are at the end of a program section and the Linker output is executable and the Linker has not been instructed to stack another module on top of this section, DEFS does not include the reserved bytes in the output file.

DLONG

Stores a value in a 64-bit storage location.

DOUBLE

Converts a value to double-precision floating-point format. If the fractional part of the value is larger than 52 bits, DOUBLE truncates the value.

END

Defines the end of a program or an included file. A value can be given specifying the program starting address which is encoded in the output file if the output format definition has a record type for a program starting address.

EXTENDED

Converts a value to extended-precision floating-point format. If the mantissa is larger than 63 bits, EXTENDED truncates the value.

FCC

Stores a string in memory until a character is reached that matches the first character.

FLOAT

Converts a value to single-precision floating point format. If the fractional part of the value is larger than 24 bits, FLOAT truncates the value.

LONG, LONGW, LWORD

Stores a value in a 32-bit storage location.

MASK

Defines a mask for Ascii character literals and strings. The Ascii characters are logically anded with the first value and ored with the second value.

MESSAGE

Outputs a user-defined message to the screen.

ORIGIN, ORG

Sets the program assembly address equal to a specified value.

PACKED

Converts a value to packed bed format. If the converted value is larger than 18 bits, PACKED truncates the value.

SQUOTE

Instructs the Assembler to treat an Ascii string with only a leading apostrophe.

WORD, DEFW, DW, FDB

Stores a value in a 16-bit storage location.

Assembly Mode

ABSOLUTE

Gives a symbol an absolute address; useful for defming templates.

BIT7

Instructs the Assembler to set the high bit of each character in an Ascii string. This modifies characters stored with the ASCII and BYTE directives.

CHIP

Changes processor mode to assemble for the specified CHIP.

COMMENT

Allows blocks of comments to be written. This is a useful way to stop a portion of code temporarily from being executed.

DCMODE

The DC directive stores values in consecutive memory locations setting the high bit of the last data byte. DCMODE determines whether DC sets the last byte of each string or the last byte of each line.

ENDMOD

Specifies the end of a module in a file.

ENDS

Terminates nested sections in a file.

EVEN

Code addresses always start on an even boundary. EVEN forces data addresses also to start on an even boundary.

INCLUDE

Instructs the Assembler to include the named file in the Assembly.

LEADZERO

Executed at the start of a program, LEADZERO ON instructs the Assembler to recognize alphabetic characters in the operand field as hex numerals only if they are immediately preceded by a zero, and otherwise to treat them as label names.

MODULE

Use with ENDMOD and the Library Manager. Normally, libraries consist of many individual modules. The Assembler produces an output file containing individual modules which the Librarian can group into library files. When the Linker cannot find a Global symbol in a file to be linked, it searches the libraries and includes only the module of the specified symbol name.

OUTPUT

Allows the user to choose between 2500AD and Microsoft object output formats.

RADIX

Specifies the number base for all numerical operations within a program. The base can be octal, binary, decimal or hexadecimal.

RELATIVE

Instructs the Assembler that the address of the following code is relocatable.

SECTION

Useful for locating a block of code or data at a specific address. The user may switch to and from the defined section using the name as a mnemonic.

SPACES

Instructs the Assembler to allow instructions containing spaces between operands or to disallow spaces between operands.

SYNTAX

Zilog and Rockwell processors only.
Instructs the Assembler to process standard Zilog addressing mode syntax,standard Rockwell 6502 addressing mode syntax,or 25OOAD addressing mode syntax.

Allows use of the Ascii two character abbreviations or disallows the use of these abbreviations.

Conditional Assemblv

ELSE

In the event an IF result is false, assemble subsequent statements.

ENDIF ENDC

Defines the end of a conditional assembly block. Unmatched IF & ENDI F pairs will produce an error message.

EXIT

Executed inside a conditional, EXIT terminates assembly.

IF, COND, IFN, IFNZ

If a value does not equal zero, assemble subsequent statements.

IFABS , IFNREL

If a label is non-relocatable, assemble subsequent statements.

IFCLEAR

In a recursive macro, IFCLEA Rkeeps IF & ENDI Fpairs balanced.

IFDEF

If a label is found, assemble subsequent statements.

IFDIFF, IFNSAME

If two strings are not identical, assemble subsequent statements.

IFEXT

If a label is external, assemble subsequent statements.

IFFALSE, IFNTRUE

If the condition is false, assemble subsequent statements.

IFMA

Used inside a macro. If the argument number specified by a value is found in the macro call line, assemble subsequent statements.

IFNDEF

If a label is not found, assemble subsequent statements.

IFNEXT

If a label is not external, assemble subsequent statements.

IFNMA

If the macro argument number specified by a value is not found, assemble subsequent statements.

IFPAGEO

If a label is not defined as a PAGE0 variable, assemble subsequent statements.

IFNPAGEO

If a label is defined as a PAGE0 variable, assemble subsequent statements.

IFREL IFNABS

If a label is relocatable, assemble subsequent statements.

IFSAME , IFNDIFF

If two strings are identical, assemble subsequent statements.

IFSSEQ

Compares a sub-string with a compariso string starting at a specified character number in the comparison string. If the strings are the same, assemble subsequent statements.

IFSSNEQ

Compares a sub-string with a string starting at a specified character number in the comparison string. If the strings are different, assemble subsequent statements.

IFTRUE

If the condition is true, assemble subsequent statements.

IFZ , IFE

If a value equals zero, assemble subsequent statements.

Listing Control

ASCLIST

Turns on listing of Ascii strings that require more than 1 line of listed source code.

CONDLIST

Turns on listing of false conditional assembly blocks.

LIST ON/OFF

In conjunction with the listing control assembly option, use ON to start listing a program section and use OFF to stop it.

MACLIST ON/OFF

MACLIST ON turns on listing of macro expansions .MACLIST OFF turns the listing of macro expansions off.

PAGE, EJECT, PAG

Outputs a form feed to the listing device.

PASS1

Turns on the listing of assembly pass 1.

PL

Sets the printer page length.

Sets the printer page width.

SUBTITLE, STTL, SUBTTL

prints a string at the top of every page If a title is defined, the subtitle will appear below.

TITLE, HEADING, NAM, TTL

prints a string at the top of every page.

TOP

Sets the number of lines from the top of the page to the page number.

Definition Control

ARGCHK

Use to invoke a macro with fewer parameters than were used in defining it.

ASK

Use to receive a one character response from the terminal. The purpose of this is usually to introduce a O/l flag into the program, for such things as conditional assembly.

EQUAL, EQU

Equates a label to a value. The value may be another symbol or any legal arithmetic expression.

EXTERNAL, EXTERN, XREF

States that a LABEL is defined in another file or module.

GLOBAL, PUBLIC, XDEF

Defines a LABEL as a global label that other files or modules can reference.

GLOBALS

All labels defined after GLOBALS are global labels which other files can reference. This does not affect Local Labels.

LLCHAR

Changes the character that identifies a Local Label.

MESSAGE, MESSG

Outputs a user defined message to the screen.

REGISTER, REG

For Zilog and Intel processors only.
Allows a customized register name to be specified. The user may define as many as desired. The Assembler treats register names like any other label, recognizing up to 32 characters and performing normal error checks and stores them in a separate buffer so that it does not need a symbol-table search to check for a register name.

UPC ON

UPC ON enables the decoding of Memory Registers 128 to 223 which exist in the Zilog Universal Peripheral Controller.

VAR, DEFL, SET

Equates a label to a value. The assignment with VAR can be changed as often as desired throughout the program.

Linker Control

COMREC

Allows a comment record to be inserted in a Motorola formatted output file.

FILLCHAR

Instructs the Linker, when creating an executable output file, to fill between memory areas with a specified value.

OPTIONS

Selects the options for linking, such as symbol file type, code file output type and creation of optional map file.

RECSIZE

Allows the default lengths for Intel Hex and Motorola S records to be changed.

LINKLIST

Instructs the Linker to relocate the Assembler listings so that the execution address, hex code field addresses and cross-reference table values are the actual run-time values.

SYMBOLS

Instructs the Linker to send Microtek, Extended Microtek and Zax symbol table formatted output to the output symbol file.

Macros

A macro is a sequence of source lines that may be substituted for a single source line. A macro must be defined before it can be used. The Assembler stores the defmition and when it reaches the macro name, substitutes the defined source lines. A macro definition may include arguments substituted into any field except the comment field.

For actual macro calls, arguments may be of ***any*** type; direct, indirect, character string or register. No argument except an Ascii string bracketed by apostrophes may include spaces. As long as the dummy argument names are identical, arguments can be passed through to nested macros. Memory space is the only limit on macro nesting.

In the macro call line, arguments must be separated by commas. If there is no argument, a single comma acts as a place holder. Values or strings can be concatenated inside of a macro.

Labels

Macro definitions can contain labels which may be defined as explicit or implicit. Explicit labels in the macro definition will not be altered by the Assembler. Implicit labels are followed by a #, for which the Assembler substitutes a 4digit macro expansion number. An implicit label and its macro expansion number must not exceed 32 characters. An argument can specify a *label.*

MACDELIM

Allows an argument containing a comma to be passed into a macro. Normally commas default to being argument separators.

MACEND, ENDM

Specifies the end of a Macro Definition.

MACEXIT

Exits from a macro immediately and unconditionally. Unlike MACEND, MACEXIT does not terminate a macro definition. It simply exits from wherever it is placed within a macro leaving the rest unexecuted and restoring all conditional values to what they were before the macro was invoked. MACEXIT is necessary in recursive macros.

MACFIRST

To redefine a mnemonic, MACFIRST must precede the new macro definition.

MACRO

Specifies the start of a Macro Definition.

MACSEP

Allows the use of addressing modes that require commas.

The Linker

The 25OOAD Linker allows the user to write modular Assembly language programs. The Linker can resolve external references and relocate addresses. It generates all the frequently used code file formats so that a separate format-conversion utility is not needed. Files of any size may be linked.

Link Address

A ***link address*** may be specified in a file and relocated at link time. A section's link controls can be set as the section is defined in the source file. For example, a section may be defined as reference only. Though the Linker includes its link information, it does not appear in the output file. Or, a section can be linked at different run-time and load addresses generating ROM-able code that moves to read/write memory at run time.

Operating Mode

The Linker may be invoked in Prompt (answer questions when prompted), Data File (create a data file for the Linker to read) or Command line mode (put entire link command, including file names and address offsets, on a single line for the Linker to read). The load map, alphabetic global symbol list and all link errors may be saved on disk.

Output Formats

The Linker will output Executable Binary, Intel Hex, Extended Intel Hex, Tektronix Hex, Motorola S19, S28 and S37 file formats. The Linker supports 25OOAD Global, Abbreviated Global, High Level, Rockwell, Microtek, Extended Microtek and Zax symbol tables.

Indirect Linking

Indirect Zinking is the linking of a file section to run at an address other than its load address. This is comparable to phase and dephase; but whereas phase and dephase change the address at assembly time, ***indirect linking*** changes it at link time.

Reference Only

Sections linked ***reference*** only are used by the Linker for information only. After relocating a ***reference only*** section to the specified address, the Linker will use in the link process any globals defined in the ***reference only*** section but will not include the section in the output file.

Reference only linking is useful when there is a program section resident in ROM and a data section resident in RAM but only the program section is wanted in the output file. The output file can then be loaded into ROM.

The Librarian

The 25OOAD Librarian allows the creation of a library of object modules for linking with a program generated by the Assembler. The Linker searches all libraries that are specified and links only referenced modules.

The Librarian is full color, menu driven and can be run interactively using a keyboard or a mouse to select and execute commands and modules.

Commands

Different commands will:

- Add modules to a library.
- Delete modules from a library.
- Save the current library and exit from the Librarian.
- Find and select a specified module.
- List modules' global symbols.
- Create or load a different library.
- Exit from the Librarian after an optional save.
- Delete a current library module and load a new version.
- Display details of the current library.

8080 To Z-80 Source Code Converter

This converter is available for Z-80, Z-280 and 64 180 processors only.

The 8080 to Z-80 Source Code Converter converts standard Intel 8080 source code to Zilog Z-80 source code. The Assembler can then assemble the Z-80 source code.

8080/Z-80 To Z-8000 Source Code Translator

This translator is available for the Z-8000 processor only.

The 8080/Z-80 to Z-8000 Source Code Translator will convert Assembly language written in either 8080 or Z-80 syntax into Z-8001 or Z-8001 source code. The Assembler can then assemble the Z-8000 source code.

8080/Z-80 To 8086 Source Code Translator

This translator is available for the 8086, 80186 and 80386 processors only.

The source code translator converts Assembly language written in either 8080 or Z-80 syntax and converts it into 8086 source code. The Assembler can then assemble the 8086 source code.

Additional Features

Series 4

- Accepts 256 Section Names
- Linker Accepts 256 Input Files

OS/2

- 1025 Externals Per File
- Accepts 400 Section Names
- Linker Accepts 1000 Input Files
- Includes May Be Nested 50 Deep
- Virtually Unlimited Symbol Table Size
- Librarian Accepts 500 Modules
- Section Size Checking
- Section Address Summary

DOS** 32, **Xenix,** 386 **Unix, Ultrix and Sun 4

- ***2500*** Externals Per File
- Accepts 1000 Section Names
- Linker Accepts 2000 Input Files
- Includes May Be Nested 50 Deep
- Virtually Unlimited Symbol Table Size
- Librarian Accepts 500 Modules
- Section Size Checking
- Section Address Summary

System Requirements

Series 4 requires 640k of memory. Series 4 products are only available for the MSDOS operating system. Series 5 requires a minimum 4 megabytes of extended memory. All new feature enhancements will only be added to the Series 5. Versions are available for MSDOS, OS/2, Xenix, 386 Unix, HP 700 and Sun 4 operating systems.