MPW QR4 Assembler
Release Notes

The following features have been added to the Assembler for version 3.2.

- New optimizations.
- Longer identifiers.
- New parameter and syntax for module directives.
- MC68040 support.
- “32-Bit Everything” support.
- Improved SADE support.
- A new macro function : &SYSINMOD

New Optimizations

The 3.2 Assembler supports optimization of instructions which were previously not optimized. The following additional optimizations are available:

```
ADD   #<Data>,An   [-8 <= data <= -1]  =====>   SUBQ   #-<Data>,An
ADDA  #<Data>,An   [-8 <= data <= -1]  =====>   SUBQ   #-<Data>,An
ADDI  #<Data>,An   [-8 <= data <= -1]  =====>   SUBQ   #-<Data>,An

SUB   #<Data>,An   [-8 <= data <= -1]  =====>   ADDQ   #-<Data>,An
SUBA  #<Data>,An   [-8 <= data <= -1]  =====>   ADDQ   #-<Data>,An
SUBI  #<Data>,An   [-8 <= data <= -1]  =====>   ADDQ   #-<Data>,An
```

The above optimizations only occur when the Assembler understands the operand to be a negative number. Since the Assembler operates on 32-bit arithmetic, the value of the operand must be written as a negative number in order to activate the optimization. This is necessary regardless of the size (B, W, L) of the instruction. Thus, in a byte operation, the operand must specify "-1" instead of "$FF", even though, un-optimized, they would both compile to the same instruction.

```
ADDI  #<Data>,<EA3>   [+1 <= data <= +8]  =====>   ADDQ   #<Data>,<EA3>
SUBI  #<Data>,<EA3>   [+1 <= data <= +8]  =====>   SUBQ   #<Data>,<EA3>
ADD.I #<data>,An      [-32767 <= data <= +32767]
                                          =====>   LEA   <Data>(An),An
```

Longer Identifiers

The Assembler now supports identifiers of lengths up to 251 characters (this was done primarily to support C++ name mangling). On detecting a name that exceeds this maximum, the Assembler issues the warning:

```
ID longer than maximum length...truncating to maximum length.
```

New Parameter and Syntax for Module Directives

A new parameter, `ForceActive`, exists for the code module and data module directives. A new syntax has been provided and error checking for conflicting parameters is in effect.

ForceActive

`ForceActive` causes a code or data module to be included in the final linked output even if the Linker would otherwise strip that code or data module because there were no references to it.

Syntax

For code module directives:

```
Line ::= [label] CodeDir [KeywordList]
CodeDir ::= PROC | FUNC | MAIN
KeywordList ::= Keyword | KeywordList, Keyword
Keyword ::= ENTRY | EXPORT | FORCEACTIVE
```

For data module directives:

```
Line ::= [label] RECORD [KeywordList]
```

```
KeywordList ::= Keyword | KeywordList, Keyword
```

```
Keyword ::= ENTRY | EXPORT | INCR[EMENT] | DECR[EMENT]
            MAIN | FORCEACTIVE
```

◆ Note: The MPW 3.0 Assembler Manual describes MAIN as a parameter for the RECORD directive, but appears to have inadvertently omitted MAIN from the syntax.

△ **Important**: Parameters following the directive RECORD determine whether a data module or a template is being defined. A template definition is assumed if the first token following a RECORD directive is any of the following: a number, a left bracket, a numeric expression, or an undefined label. △

MC68040 Support

The support for the MC68040 is initiated by the directive `MACHINE MC68040`. This enables the processing of mnemonics and addressing modes that are specific to the MC68040. New MC68040 registers are URP, ITT0, ITT1, DTT0, DTT1. Use of the `MACHINE MC68040` directive disallows all MC68030 registers except for TC, MMUSR, and SRP. It also disallows the MC68851, it and the MC68040 being mutually exclusive.

"32-Bit Everything" Support

A mechanism is now in place to support 32-bit references in assembly language. The requirements are: the option `-model far` must be used; the reference must be written using the absolute long address syntax `(xxxx).L`; and the symbol being referenced must be imported. (For further details and an example, see MPW QR4 Run-Time Architecture Release Notes.)

Independent of the `-model far` mechanisms, it is now possible to define records and templates which are larger than 32K.

◆ *Note:* The use of "32-Bit Everything" in the Assembler is demonstrated in the modified version of the *count* example to be found in the folder:

{MPW}Examples:32bitAExamples:

Improved SADE Support

The Assembler now outputs record names as SADE symbols. It also now outputs the correct creation date of the source file.

New Macro Function: &SYSINMOD

The &SYSINMOD macro function is a superset of the &SYSMOD macro function. In addition to the functions provided by &SYSMOD, the new function returns `'@GLOBAL'` if you are not in a module and `'@TEMPLATE'` if you are in a template.

Known Outstanding Bugs

Assembler Bugs

- In a decrementing data module, where the last field contains an odd number of bytes: a) in a main data module (record or "data" section in code) when the user has selected Align Off, exported fields will not evaluate to the same address as fully qualified field references; b) in Typed Data (data defined with a template) used within a "data" section in code, the alignment of previous fields is not guaranteed—this means that longword and word fields could appear on odd addresses without warning.
- When Typed Data is defined within the "data" section of a code module, and its label is exported, the references to it from other code modules are not consistent with those in the defining code module.
- A couple of problems arise when referring to the label of a decrementing typed data module: a) The label points to different fields in the record, depending upon whether the data module was defined i) within a "data" section of a code module, or ii) at a global level; b) The label, when the module is defined at a global level, does not point into the module at all, but rather to the next piece of data (or, if there is no next piece of data, to location `0(A5)`.) To be safe, **always** used qualified references when using decrementing typed data modules.

- Integer Too Large: 2147483648. Macro language accepts an incorrect value of 2147483648 as an integer. It is internally represented as -2147483648. For example the following is accepted:

```
                GBLA        &intVar
      &intVar   SETA        2147483648
                WRITELN     &intVar        ; writes out -2147483648
```

- If the &LIST function is used, divides a list into four items and is called again with four items, one of which is a null-string, the &LIST function will fill it with the value from the previous call, instead of a blank as suggested in the manual.
- String Truncation: The assembler truncates macro default parameter strings to a number less that 255 characters depending upon the length of the macro line and the setting of the -l option.
- There are references in the manual to ranges of -32768 .. +65535. These should be changed to be -32768 .. +32767.
- An error is not reported in the case of forward reference to a local label when using an absolute addressing mode.
- If either Floating Point branch instructions of the form FBcc, or PMMU branch instructions of the form PBcc are optimized from Long to Word, the optimization is incorrect. The amount of space allocated for the instruction is shortened, but the opword still represents the long form. This problem does not occur when “OPT NONE” is used.

Documentation Errors

- The description of the C calling conventions in the MPW 3.0 Assembler Manual is incorrect. Please use the documentation in the MPW 3.0 C manual.
- The Condition Codes given in Appendix F of the MPW 3.0 Assembler Manual are incorrect. Please use the Motorola User Manuals for an accurate description.

Fixed Bugs

The items listed below are bugs that existed in MPW 3.0 or MPW 3.1. The list does not include bugs that first appeared in Alpha or Beta versions of MPW 3.2.

- The Assembler no longer crashes if a very large number of macro array variables are set to very large default strings.

- The Assembler no longer crashes if a listing is to be generated in which there are lines which contain more than 230 characters.
- The &LEX function now properly handle semi-colon and backslash characters in the input.
- The Assembler now requires imported labels to be declared before use, generating an error otherwise; thus conforming to documentation.
- The Symbol Table entry mechanism was altered to correct potential problems with forward-referencing of local variables.
- The Object Assembly Macros now works for method overrides that call inherited routines.
- Structured Assembly Macros (FlowCtlMacs.a)

  A major bug fix was done to handle the case of reverse operands, which were handled incorrectly. The bug was that for reversed operands the reverse of a condition code is not the same as the negate of a condition code. Thus the reverse of GE is LE, but the negate of GE is LT.

  A second bug was fixed for the handling of the negation of constructs like "A rel B OR C rel D". These were processed incorrectly in IF# macros.
- The ACTR loop counter now exits when its value exceeds the ACTR limit value (as specified in the documentation), instead of when it reaches the ACTR limit value.
- Forward references to labels defined as registers or registers lists are now flagged as errors.
- Global equates to relocatable values are now flagged as errors.
- Warnings are now issued when local equates are redefined to the same value as a global equate.
- SADE records are no longer output for @-labels.
- A warning is now issued when the last instruction in a code module is a BRA.S to an imported label.
- An error is now reported when substitution of macro parameters generates a line longer than 254 characters.
- Expansion of macro variables on continued lines is now supported.
- Load/Dump now allows templates to be larger than 32K and template origins to be greater than 32K. (Be aware that the fix required a format change to Load/Dump files.)
- It is no longer possible to use a macro name which had previously been defined as an opword.

- An error is now reported when a label in a record module is imported and subsequently redefined.
- An error is now reported in the case of backward reference to a local label when using an absolute addressing mode.
- The value 32 can now be used as the width designation of a bit field. It was formerly necessary to represent it by using 0. (0 is still valid.)
- Globally imported data labels are no longer permitted for relative branch instructions, e.g. FBEq, DBcc.
- Floating-point constants, when defined with DC.D, DC.S, or DC.X are now correctly converted. (This was a bug in the 3.1 final version only.)
- The assembler now allows PC-relative addressing for CMPI instructions when the machine designation is either MC68020, MC68030, or MC68040.