APPENDICES

Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

Appendix A EMULATING BIT MANIPULATION OPCODES WITH MACROS

When porting code from the W65C02S to the W65C134, W65C816 and W65C265 the bit manipulation opcodes SMBn, RMBn, BBSn and BBRn are replaced by the following macros.

Two sets of macros are provided here: a long-form macro table and a short-form macro table. The long-form macros allow direct replacement of the bit manipulation instructions with no code rewrites; the cost is a few more bytes and a few extra W65C134S cycles. Using the short-form macros and making minor code changes, most of this overhead is eliminated.

W65C02S four (4) instructions:

SMBn (Set Memory Bit n)(where n is a bit from 0 to 7)	2 bytes, 5 cycles
RMBn (Reset Memory Bit n)	2 bytes, 5 cycles
BBSn (Branch on Bit n Set)	3 bytes, 5 cycles
BBRn (Branch on Bit n Reset)	3 bytes, 5 cycles

These are two or three byte instructions that operate only on memory bytes in page 0. In addition, each instruction can change or test only one bit at a time. Using TSB (Test and Set Bit) and TRB (Test and Reset Bit), programs can test and change up to 8 bits at once.

1. W65C02S to W65C134S Long-form Macro Table

These W65C134S macros will DIRECTLY replace W65C02S instructions without any code rewrites, although there is considerable overhead in extra cycles and bytes. At least two bytes of stack are required to use the macros; these are used to save the accumulator and processor status.

	Bytes	Cycles			Bytes	Cycles
RMBn PAGE0_BYTE	2	5		PHP	1	3
				PHA	1	3
				LDA #BIT_n	2	2
				TRB PAGE0_BYTE	2	5
				PLA	1	4
				PLP	1	4
Totals:	2	5		Totals:	8	21
SMBn PAGE0_BYTE	2	5		PHP	1	3
				PHA	1	3
				LDA #BIT_n	2	2
				TSB PAGE0_BYTE	2	5
				PLA	1	4
				PLP	1	4
Totals:	2	5		Totals:	8	21
BBRn PAGE0_BYTE, BITnCLR	3	5		PHP	1	3
				PHA	1	3
				LDA PAGE0_BYTE	2	3
				AND #BIT_n	2	3
				BNE L1	2	2/3
				PLA	1	4
				PLP	1	4
				BRA BITnCLR	2	3
			L1	PLA	1	4
				PLP	1	4
			L2			
Totals:	3	5		Totals:	14	25
BBSn PAGE0_BYTE, BITnSET	3	5		PHP	1	3
				PHA	1	3
				LDA PAGE0_BYTE	2	3
				AND #BIT_n	2	3
				BEQ L1	2	2/3
				PLA	1	4
				PLP	1	4
				BRA BITnSET	2	3
			L1	PLA	1	4
				PLP	1	4
			L2			
Totals:	3	5		Totals:	14	25

2. W65C02S to W65C134S Short-form Macro Table

These short-form macros will replace the W65C02S instructions in many cases. Unlike the actual bit manipulation instructions, these macros modify the accumulator and the status register after their execution. Therefore, minor code changes may be required in some cases.

```
Examples:   LDA  #$55
            SMB  $AA
            STA  $BB        ; assumes .A still has #$55 (rewrites required)
            SMB  $AA
            BNE  LOOP       ; assumes Z flag unchanged (rewrites required)
            SMB  $AA
            LDA  #$22       ; these macros will work because .A and Z flag
                            ; modified after SMB (no rewrites required)
```

W65C02S	Bytes	Cycles	WDC W65C134S Core Macro Equivalent	Bytes	Cycles
RMBn PAGE0_BYTE	2	5	LDA #BIT_n	2	2
			TRB PAGE0_BYTE	2	5
Totals:	2	5	Totals:	4	7
SMBn PAGE0_BYTE	2	5	LDA #BIT_n	2	2
			TSB PAGE0_BYTE	2	5
Totals:	2	5	Totals:	2	5
BBRn PAGE0_BYTE,BITnCLR	3	5	LDA PAGE0_BYTE	2	3
			BIT #BIT_n	2	2
			BEQ BITnCLR	2	3
Totals:	3	5	Totals:	6	8
(alternate, bit 6)					
BBR6 PAGE0_BYTE,BIT6CLR	3	5	BIT TEMP_PAGE0	2	3
			BVC BIT6CLR	2	3
Totals:	3	5	Totals:	4	6
(alternate, bit 7)					
BBR7 PAGE0_BYTE,BIT7CLR	3	5	BIT TEMP_PAGE0	2	3
			BPL BIT7CLR	2	3
Totals:	3	5	Totals:	4	6
BBSn PAGE0_BYTE,BITnSET	3	5	LDA PAGE0_BYTE	2	3
			BIT #BIT_n	2	2
			BNE BITnSET	2	3
Totals:	3	5	Totals:	6	8
(alternate, bit 6)					
BBS6 PAGE0_BYTE,BIT6SET	3	5	BIT TEMP_PAGE0	2	3
			BVS BIT6SET	2	3
Totals:	3	5	Totals:	4	6
(alternate, bit 7)					
BBS7 PAGE0_BYTE,BIT7SET	3	5	BIT TEMP_PAGE0	2	3
			BMI BIT7SET	2	3
Totals:	3	5	Totals:	4	6

THIS PAGE LEFT INTENTIONALLY BLANK

Appendix B CHANGES

Version 3.49 9/14/05

Optimizer:
Enhancement to reduce code size generation for certain addressing modes lda #$0, sta to stz - removal of some redundant loading of pseudo registers - removal of reloading for known conditions.
Modified factorization option to handle removal of unneeded stack frames.

65816 Compiler:
Increased Register Storage Allocation.
Modified Modulos operations for efficient computation where possible.
Removed extra code generated when a cast is not assigned to a variable.
Modified to handle redefined structures.
Modified to recognize underscores in function names (to 30 characters).

Debugger:
Modified for 6502 mode to wrap zero page address from $FF to $00

Version 3.48 3/4/05

65816 Compiler:
Added an error, so the compiler would not hang if a macro was split across two lines.
Added a check to exit the compiler if a while statement doesn't reduce to an expression.
Modified the symbol path for H_IBMPC on include files to \
Forced full path output.
Forced current working directory into include.

Linker:
Converted a warning to an error when the same function is defined in multiple objects
Modified # remarks to seek to next line in linker files for the –f option.

Library:
Modified CSUP.AS mul32 hardware multiply

Debugger:
Added a -m option for forcing a hardware multiply during simulation.
Added HWMultiply=0 to WIN.INI to force a hardware multiply.
Increased the line length of Module paths to 500 characters.

Optimizer:
Modified to fix post incrementing arrays.

Version 3.47 7/12/04

65816 Compiler:
Changed so that absolute references to memory in small model outside the first bank do generate long absolute addressing mode.
Removed error on casting near pointers to shorts or ints.

65816 Assembler and Optimizer:

Changed input line buffer to 2048 from 512.
Fixed bug in optimizer where ASMSTART line preceded by blank line confused optimizer state.

Version 3.46 5/10/04

65816 Optimizer:
Fixed bug with stack references optimization when RTS not at end of function.

Version 3.45 4/27/04

65816 Optimizer:
Added check for no stack references after optimization and eliminated unnecessary function startup and exit code.

Version 3.44 4/5/04

6502 Compiler:
Fixed bug with casts from int to float causing compiler crash.

65816 Compiler:
Fixed warnings generated by code generator to have the correct line number.
Added some more bit field cases where TRB/TSB are used.

65816 Optimizer:
Fixed bug where ASL A was considered 3 bytes long by factorization.

Misc:
Changed WDCLIB to have a bigger buffer size for redirected arguments.
Added WCTYPE.H to include files.

Version 3.43 3/25/04

Compiler:
Changed cast optimization to handle AND, OR and EOR more aggressively.
Added more cases where TSB/TRB are used.
Initialized const scalars are now treated as constants.

Optimizer:
Removed extraneous print messages.
Increased minimum levels for factorization.

Misc:
Fixed makefile again for W65C22S.DLL so that it adds in the BMP resources correctly.

Version 3.42 3/1/04

Compiler:
Added warning 170 to warn of inability to use TSB/TRB on volatile values.

Optimizer:
Fixed bugs in factorizing.

Version 3.41 2/17/04

Compiler:
Modified –md option to set bits on function exit.
Modified –md option to use STA IRE instead of STZ.
Added –sx option to enable factorizing in optimizer. –sp option must be set as well.
Fixed bug where global symbols referenced in #asm statements weren't generating and XREF.
Changed AND and OR of constants to use TSB and TRB if possible.

Optimizer:
Added –f option to enable factorizing.

Misc:
Fixed makefile for W65C22S.DLL so that it adds in the BMP resources correctly.
Fixed makefile for W65C22S.DLL to make three versions for 134 and 265 as well.

Version 3.40 1/12/04

Compiler:
New –md option for Micronas interrupt handling.

Optimizer:
Fixed bug where inline assembly wasn't being handled properly.

Version 3.39 11/03/03

Compiler:
Fixed bug where character bit fields are shifted using ROR instead of LSR.

Version 3.38 10/06/03

Compiler:
Fixed bug with TRB.
Changed so that absolute references to memory in small model don't generate long absolute addressing mode.
Fixed bug where pointer cast to long was being treated as a far pointer.
Changed compiler to allow long bit fields.
Fixed bug in long bit field handling.
Fixed bug in expression type reducer that was generating incorrect shifts.
Fixed bug internal error where registers weren't being freed.

Optimizer:
Fixed bug where tracking of X and Y was incorrect.

Version 3.37 9/15/03

Compiler:
Fixed bug with bit fields and large data model.
Removed optimization for multiplies and divides that removes unneeded casts.
Removed unnecessary AND after AND used for testing.
Changed to use TRB in cases where argument and source are a global byte.

Version 3.36 7/21/03

Compiler:
Fixed bug with volatile pointer in bit-field assignment.

Version 3.35 7/9/03

All:
Changed object format definition to use unsigned char for module name length.

Compiler:
Changed references to hardware multiply registers to use previous declaration if made.
Fixed bug where a multiply defined symbol caused a bad pointer reference in the compiler.

Version 3.34 6/26/03

Optimizer:
Fixed limitation for number of labels referenced.

Linker:
Increased size of buffer used to display object file information.

Version 3.33 5/27/03

Compiler:
Fixed bug with volatile bitfield handling and constant 0.
Changed compiler/optimizer to always generate sep/longa and rep/longa pairs.
Changed compiler to add volatile comment to volatile references.

Optimizer:
Changed to not remove references to volatile locations.

Linker:
Added option –Pxx to set fill character to hex value XX.
Fixed bug with long file names.

Version 3.32 5/20/03

Compiler:
Fixed bug with LONGA on/off when using addresses.

Optimizer:
Added listing options to 02 optimizer.

Version 3.31 5/12/03

Compiler:
Changed long multiplies to use lhwmul finction.
Improved character bitfield handling.
Changed bitfield handling of volatile references.
Updated 6502 version with changes generic to 65816 version.

Linker:

Fixed bug in binary output where holes were not being handled correctly.

Library:
Added 32 bit hardware multiply.

Debugger:
Added Delay parameter to WDCDB.INI to slow down parallel port access for fast machines. Default is zero for no delay.

Version 3.30 04/13/03

Compiler:
Changed MULCAND, MULPLIER and IRRET to all be "volatile unsigned char" type.
Changed MULPROD to be "volatile unsigned short" type.
Changed error message for symbol type changed from 157 to 90.
Fixed bug when casting from an integer type to a pointer.
Fixed bug where a '#' in a macro that was not a call was treated like a call.

Version 3.29 04/06/03

Compiler:
Changed support for Micronas HW Multiply.

Assembler:
Fixed bug that occurred if the last line of an included file was not terminated.

Debugger:
Fixed bug with structure size in watch window.

Library:
Added 16 bit hardware multiplier for Micronas.

Version 3.28 02/23/03

Optimizer:
Fixed bug with not tracking Acc size when holding a constant value.
Fixed bug where redundant load was removed incorrectly.

Linker:
References to _BEG_section or _END_section that are preceded by '__', '_~', '~_' or
'~~' are now mapped to the appropriate symbol.

Version 3.27 02/07/03

Compiler:
Fixed a bug where UDATA section directives were being put out twice.

Version 3.26 02/02/03

Optimizer:
Fixed internal bug.
Improved X and Y register tracking.

Version 3.25 02/01/03

Compiler:
Added -lt option to generate listing with embedded source statements.
Fixed store to HWMUL to be 16 bit.
Improved section handling.

Assembler:
Added CODE and DATA qualifiers to SECTION directives. Default is CODE.

Optimizer:
Added support for optimizing all sections of type CODE other than CODE section.
Added X register tracking.

Version 3.24 01/31/03

Compiler:
Fixed bug with copying bit fields.
Fixed bug with const chars in comparisons.
Multiplies by small integers are performed using shifts and adds.
Some additional shifts are now performed in-line without a function call.
Added option -mh to generate calls to ~hwmul instead of ~mul for hardware multiply.
Also added additional STA HWMUL if -mi option specified.

Optimizer:
Added Y register tracking.

Linker:
Fixed bug with spread option.

Version 3.23 12/17/02

Compiler:
Fixed bug with missing #endasm statement
Fixed '-mi' option to generate eight bit reference and to use memory model.

Optimizer:
Fixed -lw to work on optimizer.

Version 3.22 12/17/02

Compiler:
Changed function entry and exit code for interrupts not to adjust data bank register when in small or medium model.
Added '-mi' option to generate special Micronas interrupt controller support.
Added '-l' option to generate assembly listings and '-lw' to generate wide assembly listings.
Added 'asmstart' and 'asmend' directives before and after in-line assembly.
Added '-wq' option to generate WDC816CC.ERR error file in the following format:

file>line:col:type:errnum:errstr:sym

file - name of file

line - line number
col - column error occurred
type - E for error or W for warning
errnum - the actual error number
errstr - the description of the error
sym - an optional name of the symbol or function in the error

For example:
j.c>14:5:E:157:incompatible function declarations:u08
Fixed bug with long compares to small constants.

Peephole optimizer:
Added checks for 'asmstart' and 'asmend' and treats lines between as comments.
Removed redundant store if only modifying accumulator followed by store to same location.
Added '-l', '-lw', and '-k' options to support corresponding compiler options.

Assembler:
Added '-w' option to force 132 wide listings.
Added '-k path' option to set listing file name.
Added 'asmstart' and 'asmend' directives which are ignored.
Linker:
Improved section overlap error message.

Version 3.21 11/18/02

Compiler:
Fixed crash bug with improperly formed enum declaration.
Fixed bug with improper code generation.

65816 Optimizer:
Fixed a number of bugs introduced in the 3.20 version.

Linker:
Disabled checking for PAGE0 references in new overlap checking.

Version 3.20 11/11/02

65816 Optimizer:
Changed to work correctly with embedded source debug statements and comments.
Improved long branch calculation.
Function entry and exit removed for functions with no arguments and no autos.

65816 Compilers:
Fixed bug with compound errors causing an internal error.
Changed internal symbol length from 31 to 64.
Void functions no longer save the accumulator on return.
Improved code generation for single bit bit fields.

Linker:
Added more comprehensive checks for section overlap.

Version 3.19 09/18/02

Optimizer:
Fixed bug where a REP between two loads of the same value didn't kill the first load.

Version 3.18 09/11/02

Optimizer:
Fixed bug where output name was sometimes misformed.

Linker:
Fixed bug where large file names caused problems in debug output.

Version 3.17 08/17/02

Compiler and optimizer:
Changed execlp() call to a spawnlp() call to force a wait for the called program to finish.

Linker:
Changed static Aux array of 1024 to be dynamic.
Changed -V option to show full path name for files and to show just the file name for modules. Also reversed the order so that the module name is before the file name.
Added -J option to sort module info by name.

Examples:
Removed unnecessary references in C sample startup code.

Version 3.16 08/17/02

Linker:
Changed the size of the debug file reference array from 100 to 5000 which fixes a problem where too many file references would abort the link.

Debugger:
Changed display of negative values in watches and inspectors to not show 32 bits for non-32 bit types.

Version 3.15 08/01/02

Compiler:
Fixed a bug where const pointers weren't being optimized properly.

Optimizer:
Fixed a bug with a null pointer causing a crash.

Linker:
Re-fixed problem with Extended Intel Hex format output.

Debugger:
Changed error message for bad op codes in 6502 mode.

Library:
Fixed bug in 65C816 multiply.

Version 3.14 06/05/02

All:
Increased the size of arrays used to hold file paths to 256.

Compiler:
Fixed a bug with unterminated asm{} statements.

Assembler:
Increased the number of global equates and added a check for overflow.

Linker:
Fixed problem with Extended Intel Hex format output.

Version 3.13 5/17/02

Compiler:
Fixed to correctly generate debug info for unsigned longs, shorts and ints.
Now accepts long file names correctly.
Fixed bugs in 6502 floating point code generation.

Assembler:
Fixed to correctly generate debug info for unsigned longs, shorts and ints.

Linker:
Fixed to correctly generate debug info for unsigned longs, shorts and ints.

Optimizer:
Fixed bug in 6502 optimizer calling wrong assembler.

Debugger:
Long doubles are now displayed.
Fixed isassembly problem with d_st_p and f_st_p routine calls.
Fixed to correctly handle unsigned longs, shorts and ints.

WDCLIB and WDCOBJ:
Fixed to correctly handle unsigned longs, shorts and ints.

Library:
New 65c816 short circuit multiply and divide routines.
Numerous fixes to 6502 floating point library.

Version 3.12 3/21/02

Compiler:
Fixed a bug on the 816 with an incorrect addressing mode in certain situations.

Assembler:
Fixed bug with illegal 816 addresssing mode allowed for 02.

Linker:
Changed default fill character to 0xff from 0x00.
Fixed minor bugs.

Library:
New faster math routines.
Fixed bugs in 816 floating point.
Fixed some batch file typos.

Debugger:
Added support for running under WindowsNT/2000/XP.

Version 3.11 3/05/02

All:
Dongle-less operation for Windows 95, 98, ME, 2000, NT4.0, and XP SDS use.

DB:
ICD hardware breakpoint window in WDCDB.EXE to be used with the Xilinx 95108 CPLD with hardware breakpoint configuration. This can be used with W95, 98 & ME; however, W2000, NT4.0 and XP require additional changes due to parallel port configuration problems.

New folders for ICD and homework software examples

Updated user manual sections, Quick Start for SDS, Getting Started with DB's and ICD User Guide.

Version 3.11 3/07/00

Assembler:
Fixed bug with too large a debug record being written.

Version 3.10 10/5/99

Compiler:
Fixed bug with nested macros.
Fixed bug where compare with zero generated a FP check.
Constants being ANDed if less than a byte now only affect one byte.

Linker:
Fixed bug with too many symbols in debug output.

Version 3.09 04/05/99

Linker:
Changed to pass db and dw sizes to WDC symbol file.

Version 3.08 02/18/99

Assembler:
Changed to support floating point expressions.
Changed to support mult-character constants.
Other changes for 2500AD compatibility.

Version 3.07 12/14/98

Fixed INSTALL.EXE to detect errors using CCMOVE.EXE.

Compiler:
Fixed a bug with register floating point declarations.

Assembler:

Changed hex numbers to allow h and H at end of number started with '$'.
Changed space handling to always allow spaces after commas.
Fixed bug in printing long source lines.
Changed to assume zpage addressing on constants in range on 6502.
Changed to accept a '#' character on any constant data.
Changed DB to generate a zero on empty fields.
Changed to accept _x where x is a non-alphabetic character.

Linker:
The Microtek symbol option now takes an additional character to specify a two byte address. The default is a three byte address. The new options are -sm2 and -sn2.
The linker now deletes the error file when it is started.

6502 Library:
Fixed a bug in the labs() library routine.
Fixed a bug in the compiler support routines for shifts.
Examples:
Made changes to ASMSAMP/EXMPL1.ASM to support 65134 and 65265. Added two batch files to make them.

Debug:
Added support for 65134 and 65265 to WDCMON.

Version 3.06 08/10/98

Released 65C02 version and new 65C816 version.
Added COPYCONTROL copy protection to compiler and assembler.
Added INSTALL.EXE for installation.
Changed executable names, and environment variable names.

Version 3.05 03/31/98

Fixed a bug in the linker where filenames were indexed improperly and written to the debugger symbol file.

Version 3.04 02/25/98

Fixed a bug in the simulator where TXS was handled wrong in emulation mode.

Added the following equivalences to the assembler:

xor	eor
ret	rts
#.low.	#<
#.high.	#>

Fixed a bug in the assembler where the SPACES ON directive didn't work correctly.

Changed the assembler to handle absolute addresses in zero page with RMB and SMB instructions.

Fixed a bug in the assembler where STA LABEL,Y when LABEL was in zero page generated an error. It now generates an absolute address. It is STILL an error to say STA <LABEL,Y since this implies a different addressing mode which is not legal.

Added support for variable sized bit-fields in the compiler. It is now possible to declare byte sized bit-fields. An error will be generated if different sized types of bit-fields are mixed unless the new type starts at a boundary. Note that a boundary can be forced by specifying a bit size of zero.

Fixed a bug where a structure member named the same as a bit field would not be detected by the compiler.

Version 3.03 08/19/97

Fixed bug in assembler where 'ldx #imm' was considered illegal for 6502.

Version 3.02 08/19/97

Fixed bug in assembler with illegal addressing mode.
Fixed bug in assembler with not generating LINE records within reference-only sections.

Version 3.01 04/19/97

Functions declared with qualifier 'interrupt' will save and restore registers and return with 'rti'. Can be used directly from the interrupt vector table if located in Bank 0.
Fixed bug in farsbrk in compact memory model.
Added support for near/far structure copy.
Improved support for near keyword.

Version 3.00 03/11/97

Released new version with new documentation supporting source level debugging.
Pointers to single byte values that have a qualifier of type volatile will only load one byte. Useful for accessing I/O registers from C.

Version 2.0N 07/31/96

ZAS:
Changed BRK and COP and WDM opcodes to take an optional argument and to always generate two bytes.
Fixed BRL to correctly handle long displacements and to check for out of bank address errors.

Version 2.0M 10/23/95

LIB:
Fixed a bug in switch support routine in medium model.
Fixed a bug in floating subtraction in large memory model.
Fixed a bug in floating compare.

Version 2.0L 05/15/95

ZAS:
Added float and double directives for generating IEEE floating point values.

```
float   1.0,2,3.45e-23
double  2,3.4,-1.3e+12
```

Version 2.0K 02/07/95

ZAS:
Changed so that equates less than 256 can be used in stack addressing.

Version 2.0J 01/14/95

ZCC:
Fixed a bug where a statement like p = p->next will generate bad code if the pointer is a far pointer since the low word of the pointer is modified and then the pointer is used again to fetch the high word.
Changed floating point code generation to not promote to double on unary minus.
Fixed a bug where structure assignment of near and far structures wasn't handled correctly.
Fixed a bug in near to far pointer assignment.
Fixed a bug where a multiply of two shorts each cast to a long used the short multiply routine. This was an optimization that worked on a different processor. It now does the casts and uses the long routine.

ZAS:
Added IFMATCH directive which takes three comma delimited arguments, two strings and a count of characters to match.

ZLN:
Fixed a bug where a fatal error could cause a stack fault if the '-v' option was set.
Changed ISX output to strip '_' and '~' characters.

ZOPT:
Fixed a bug dealing with nested non-code sections.

LIB:
Fixed a bug when converting single precision floating point values to non-floating values.
Fixed a bug when converting double zero to float.
Fixed a bug when adding a fraction to a non-fraction.
Fixed some large code bugs.
Fixed some bugs in memcmp() and strstr().

Version 2.0I 10/04/94

ZCC:
Fixed two bugs involving loading the Y register with an index value.

Version 2.0H 3/13/94

ZCC:
Fixed a bug where "struct x far *" would generate a near pointer.

ZAS:
Changed INSERT to use the "-I" path when searching for files to insert.
Fixed DA directive to generate 3 bytes instead of 4 if no argument.
Added REPT cnt ... ENDREPT directives for repeating a sequence of assembly statements. Listing is controlled by the macro listing control.

ZLN:
Changed the fatal error routine to dump the bank information.

LIB:
Added farsbrk, farmalloc, farfree, farrealloc, farcalloc functions to the small and medium model libraries.

Version 2.0G 02/13/94

ZAS:

There is a new directive CHKIMMED which defaults to OFF. When turned on, it gives an error if an immediate load to a register is larger than will fit in the register. Legal values range from -127 to 255 when a short register is loaded and -32767 to 65536 when a long register is loaded.
Fixed a bug where a macro label with more than 4 digits would produce syntax errors.

Version 2.0F 12/20/93

ZAS:
Fixed a bug with undefined macro arguments.
A warning is now generated if an immediate value is truncated.
Fixed a bug where strings in macros weren't handled correctly.

Version 2.0E 12/08/93

ZAS:
Added four conditional directives IFLONGA, IFLONGI, IFSHORTA, and IFSHORTI.
Now quits with an error if macros are nested more than 256 deep.
Input and macro lines have been expanded from 128 to 512 characters.

ZLN:
Further improved Nintendo ISX format LONGA/LONGI ranges.
No longer maps symbols to upper case in ISX format.
Lines beginning with '#' in option input files ('-f') are ignored.

Version 2.0D 11/06/93

ZCC:
Fixed a bug in the compiler that caused an index register to be used twice without the proper value being loaded.

ZOPT:
Fixed a bug in the optimizer that occurred when multiple structure assignment was used.

ZLN:
Fixed a bug in the Nintendo ISX format dealing with the LONGA/LONGI ranges.

Version 2.0C 9/27/93

ZCC:
Added -SS option to generate better array indexing with far pointers. When -SS is enabled, the compiler assumes that arrays are <64K in size and generates more efficient code. The -SS option is automatically turned on if the -SO option is set. To use –SO without -SS, use -SO0S.

Fixed some bugs with the -MU, -MV, -MK and -MT options.
Changed the section pragma to accept 'section' as lower case or upper.
Fixed a bug with source debug information causing the optimizer to gag.
Fixed a bug with pointer addressing.

ZOPT:
Fixed a couple of bugs and added some new optimizations.

ZLN:
Fixed a bug with the code spreading logic.

Fixed bugs in the Nintendo ISX output file format.

ZAS:
Changed to allow more than 256 macros.

Appendix C Q&A

W65C02 C Compiler Questions

W65C816 C Compiler Questions

Assembler/Linker Questions

Assembler Questions

Linker Questions

Q1 - Large programs

When using the 65816 SDS the 65816 program code exceeds 65Kbytes the linker, WDCLN, gives the error message, "section, image exceeds bank 00 by <value>"

1) How do you solve this issue?

2) How do you use the WDCLN option -Z in detail?

A1 -

One way to solve it is to use the -Z option. This option causes the code from a particular section to be spread over multiple banks. The algorithm is to start at the ORG address for the section and then add modules until a particular bank has been filled and then move to the next bank. This continues until all modules have been located. The simplest case is to simply use '-Zcode=' which will spread the CODE section over the entire bank starting from 0 to $ffff and then automatically bump to the next bank. It is also possible to limit the area used by specifying a start and end address within the bank. Thus if you have ROM in each bank from $8000 to $bfff, you would use '-Zcode=8000,c000'.

The other way to solve it is to use multiple sections with less than 64K of code in each section and assign each section to a bank using the SECTION statement or using the linker -A option for each section. This gives you precise control of where code gets placed.

As a historical note, the -Z option was added for Super Nintendo development for developers of large programs that spanned a number of banks and who didn't want to have to constantly fiddle with SECTION assignments as sections changed sizes while developing. Instead, all functions were placed in one section that was spread over the banks by the linker. When the program was ready to ship, then the placement in memory was done by hand to insure the tightest fit. The linker just does a first fit search which is not optimal.

Spreading Code

A basic assumption for spreading code is that the program code has been broken into separate source files and that the code in each file does not care about the placement of the code in any other file. That is, the code in one file does not assume the code from another file will follow it in memory. This is easily accomplished by having functions within each source file that end with an RTS or more likely an RTL instruction.

Because the code from each source file or "module" is independent of other modules, the linker is free to place them in memory wherever it wants to without affecting the program's operation. In this case, the linker can spread the modules across several banks without the programmer having to explicitly declare where each module is to be located.

For example, consider a program with the following modules whose code size is shown in parentheses:

modA (8K)
modB (12K)
modC (4K)
modD (8K)
modE (4K)

If the 16K ROM area of the target machine is banked from $8000 to $BFFF, then if the linker command line is:

wdcln -Zcode=8000,c000 modA.obj modB.obj modC.obj modD.obj modE.obj

Then, the linker will do the following:

1) It will locate modA at 00:8000.
2) Since modB will not fit in the space remaining in bank 0, it will locate modB at 01:8000.
3) It will locate modC at 00:A000 since it will fit in the remaining space.
4) Since modD will not fit in the remaining space in banks 0 or 1, it will locate modD at 02:8000.
5) It will locate modE at 00:B000.

This example is somewhat simplified, but the advantage of using this option is that if modE's size changes from 4K to 5K, it will automatically be placed in bank 2 instead of bank 0 without having to change any options or any source code.

Simulator/Debugger Questions

Simulator Questions

Debugger Questions

Q1- Debugger Output File Formats

We are working on the W65C02/W65c816 ICE and we need the W65C02S/W65c816S' file output format for the high level debugger. Would you please provide this format?

A-1

These formats can be found in the WDCLN Chapter 7 and Simulator/Debugger Chapter 6 of the SDS Manual. Also, Obj816.h and Zsym.c located in the SDS Debug directory provide information.