W65C816S
Microprocessor
DATA SHEET

© The Western Design Center, Inc., 2003. All rights reserved

WDC reserves the right to make changes at any time without notice in order to improve design and supply the best possible product. Information contained herein is provided gratuitously and without liability, to any user. Reasonable efforts have been made to verify accuracy of the information but no guarantee whatsoever is given as to the accuracy or as to its applicability to particular uses. In every instance, it must be the responsibility of the user to determine the suitability of the products for each application. WDC products are not authorized for use as critical components in life support devices or systems. Nothing contained herein shall be construed as a recommendation to use any product in violation of existing patents or other rights of third parties. The sale of any WDC product is subject to all WDC Terms and Conditions of Sales and Sales Policies, copies of which are available upon request.

Copyright (C) 1981-2003 by The Western Design Center, Inc. All rights reserved, including the right of reproduction in whole or in part in any form.

TABLE OF CONTENTS

Table of Tables

Table of Figures

1 INTRODUCTION

The W65C816S is a low power cost sensitive 16-bit microprocessor. The variable length instruction set and manually optimized core size makes the W65C816S an excellent choice for low power System-on-Chip (SoC) designs. The Verilog RTL model is available for ASIC design flows. WDC, a Fabless Semiconductor Company, provides packaged chips for evaluation or volume production. To aid in system development, WDC provides a Development System that includes a W65C816DB Developer Board, an In-Circuit Emulator (ICE) and the W65cSDS Software Development System, see www.westerndesigncenter.com for more information.

The WDC W65C816S is a fully static CMOS 16-bit microprocessor featuring software compatibility* with the 8-bit NMOS and CMOS 6500-series predecessors. The W65C816S extends addressing to a full 16 megabytes. These devices offer the many advantages of CMOS technology, including increased noise immunity, higher reliability, and greatly reduced power requirements. A software switch determines whether the processor is in the 8-bit "emulation" mode, or in the native mode, thus allowing existing systems to use the expanded features.

As shown in the W65C816S Processor Programming Model, Figure 2-2, the Accumulator, ALU, X and Y Index registers, and Stack Pointer register have all been extended to 16 bits. A new 16-bit Direct Page register augments the Direct Page addressing mode (formerly Zero Page addressing). Separate Program Bank and Data Bank registers provide 24-bit memory addressing with segmented or linear addressing.

Four new signals provide the system designer with many options. The ABORTB input can interrupt the currently executing instruction without modifying internal register, thus allowing virtual memory system design. Valid Data Address (VDA) and Valid Program Address (VPA) outputs facilitate dual cache memory by indicating whether a data segment or program segment is accessed. Modifying a vector is made easy by monitoring the Vector Pull (VPB) output.

KEY FEATURES OF THE W65C816S

- Advanced fully static CMOS design for low power consumption and increased noise immunity
- Wide operating voltage range, 1.8+/- 5%, 2.5+/- 5%, 3.0+/- 5%, 3.3+/- 10%, 5.0+/- 5% specified for use with advanced low voltage peripherals
- Emulation mode allows complete hardware and software compatibility with 6502 designs
- 24-bit address bus provides access to 16 MBytes of memory space
- Full 16-bit ALU, Accumulator, Stack Pointer and Index Registers
- Valid Data Address (VDA) and Valid Program Address (VPA) output for dual cache and cycle steal DMA implementation
- Vector Pull (VPB) output indicates when interrupt vectors are being addressed
- Abort (ABORTB) input and associated vector supports processor repairs of bus error conditions
- Low power consumption (300uA@1MHz)
- Separate program and data bank registers allow program segmentation or full 16 MByte linear addressing
- New Direct Register and stack relative addressing provides capability for re-entrant, re-cursive and re-locatable programming
- 24 addressing modes - 13 original 6502 modes with 92 instructions using 256 OpCodes
- Wait-for-Interrupt (WAI) and Stop-the-Clock (STP) instructions further reduce power consumption, decrease interrupt latency and allows synchronization with external events
- Co-Processor (COP) instruction with associated vector supports co-processor configurations, i.e., floating point processors
- Block move ability

*Except for the BBRx, BBSx, RMBx, and SMBx bit manipulation instructions which do not exist for the W65C816S

2 W65C816S FUNCTIONAL DESCRIPTION

The W65C816S provides the design engineer with upward software compatibility from 8-bit W65C02 in applications to 16-bit system application. In Emulation mode, the W65C816S offers many advantages, including full software compatibility with 6502 coding.

Internal organization of the W65C816S can be divided into two parts: 1) The Register Section and 2) The Control Section. Instructions obtained from program memory are executed by implementing a series of data transfers within the Register Section. Signals that cause data transfers to be executed are generated within the Control Section. The W65C816S has a 16-bit internal bus architecture with an 8-bit external data bus and 24-bit external address bus.

2.1 Instruction Register (IR)

An Operation Code enters the processor on the Data Bus, and is latched into the Instruction Register during the OpCode fetch cycle. This OpCode is then decoded, along with timing and interrupt signals, to generate various Instruction Register control signals for use during instruction operations.

2.2 Timing Control Unit (TCU)

The Timing Control Unit keeps track of each instruction cycle as it is executed. The TCU is set to zero each time an instruction fetch is executed, and is advanced at the beginning of each cycle for as many cycles as is required to complete the instruction. Each data transfer between registers depends upon decoding the contents of both the Instruction Register and the Timing Control Unit.

2.3 Arithmetic and Logic Unit (ALU)

All arithmetic and logic operations take place within the 16-bit ALU. In addition to data operations, the ALU also calculates the effective address for relative and indexed addressing modes. The result of a data operation is stored in either memory or an internal register. Carry, Negative, Overflow and Zero flags may be updated following the ALU data operation.

2.4 Internal Registers (Refer to Programming Model Table 2-2)

2.5 Accumulator (A)

The Accumulator (A) is a general purpose register which contains one of the operands and the result of most arithmetic and logical operations. In the Native mode (E=0), when the Accumulator Select Bit (M) equals zero, the Accumulator is established as 16 bits wide (A, B=C). When the Accumulator Select Bit (M) equals one, the Accumulator is 8 bits wide (A). In this case, the upper 8 bits (B) may be used for temporary storage in conjunction with the Exchange Accumulator (XBA) instruction.

2.6 Data Bank Register (DBR)

During modes of operation, the 8-bit Data Bank Register (DBR) holds the bank address for memory transfers. The 24-bit address is composed of the 16-bit instruction effective address and the 8-bit Data Bank address. The register value is multiplexed with the data value and is present on the Data/Address lines during the first half of a data transfer memory cycle for the W65C816S. The Data Bank Register is initialized to zero during Reset.

2.7 Direct (D)

The 16-bit Direct Register (D) provides an address offset for all instructions using direct addressing. The effective Direct Address is formed by adding the 8-bit instruction Direct Address field to the Direct Register. The Direct Register is initialized to zero during Reset. The bank address for Direct Addressing is always zero

2.8 Index (X and Y)

There are two general purpose registers that are commonly referred to as Index Registers (X and Y) and are frequently used as an index value for calculation of the effective address. When executing an instruction with indexed addressing, the microprocessor fetches the OpCode and the base address, and then modifies the address by adding an Index Register contents to the address prior to performing the desired operation. Pre-indexing or post-indexing of indirect addresses may be selected. In the Native mode (E=0), both Index Registers are 16 bits wide where the Index Select Bit (X) of the Processor Status (P) register equals zero. If the Index Select Bit (X) equals one, both registers will be 8 bits wide, and the high byte is forced to zero.

2.9 Processor Status Register (P)

The 8-bit Processor Status Register (P) contains status flags and mode select bits. The Carry (C), Negative (N), Overflow (V), and Zero (Z) status flags serve to report the status of most ALU operations. These status flags are tested by use of Conditional Branch instructions. The Decimal (D), IRQ Disable (I), Memory/Accumulator (M), and Index (X) bits are used as mode select flags. These flags are set by the program to change microprocessor operations.

The Emulation (E) select and the Break (B) flags are accessible only through the Processor Status Register. The Emulation mode select flag is selected by the Exchange Carry and Emulation Bits (XCE) instruction. Table 8-1, W65C816S Compatibility Information, illustrates the features of the Native (E=0) and Emulation (E=1) modes. The M and X flags are always equal to one in Emulation mode. When an interrupt occurs during Emulation mode, the Break flag is written to stack memory as bit 4 of the Processor Status Register.

2.10 Program Bank Register (PBR)

The 8-bit Program Bank Register (PBR) holds the bank address for all instruction fetches. The 24-bit address consists of the 16-bit instruction effective address and the 8-bit Program Bank address. The register value is multiplexed with the data bus and presented on the Data bus lines during the first half of a program memory cycle. The Program Bank Register is initialized to zero during Reset. The PHK instruction pushes the PBR register onto the Stack.

2.11 Program Counter (PC)

The 16-bit Program Counter (PC) Register provides the addresses which are used to step the microprocessor through sequential 8-bit program instruction fields. The PC is incremented for each 8-bit instruction field that is fetched from program memory.

2.12 Stack Pointer (S)

The Stack Pointer (S) is a 16-bit register which is used to indicate the next available location in the stack memory area. It serves as the effective address in stack addressing modes as well as subroutine and interrupt processing. The Stack Pointer provides simple implementation of nested subroutines and multiple-level interrupts. During Emulation mode, the S High-order byte (SH) is always equal to one. The bank address for all stack operations is Bank zero.

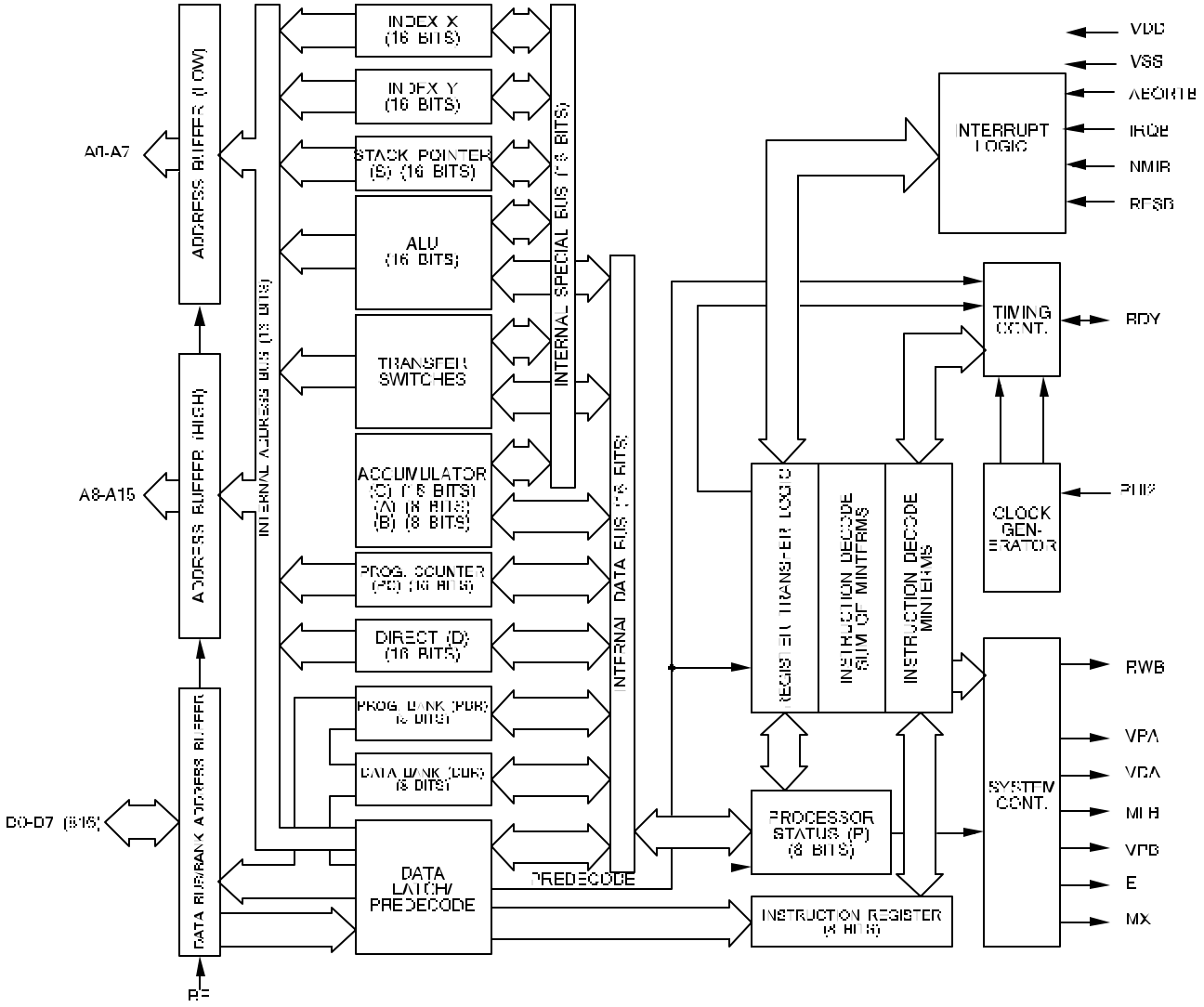
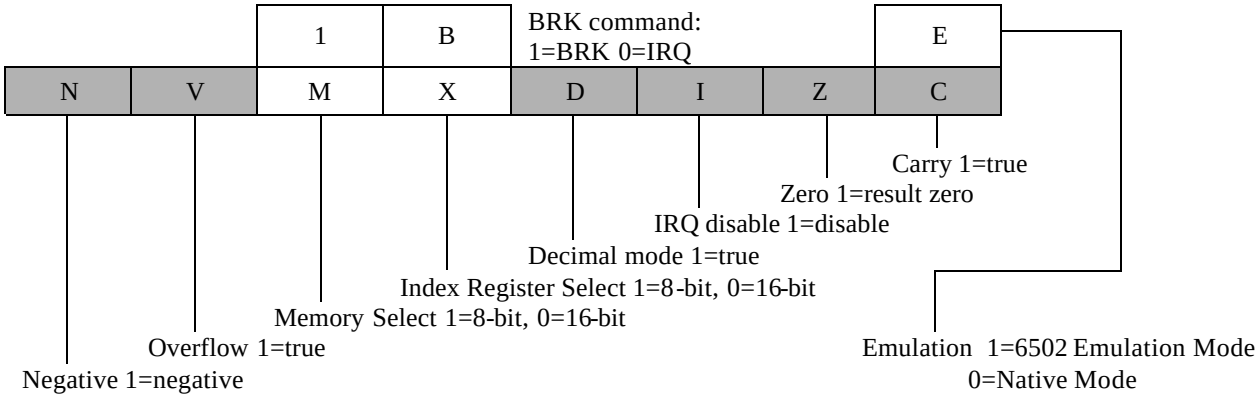


Figure 2-1W65C816S Internal Architecture Simplified Block Diagram

Table 2-1 W65C816S Microprocessor Programming Model

8 BITS	8 BITS	8 BITS
Data Bank Register (DBR)	X Register (XH)	X Register (XL)
Data Bank Register (DBR)	Y Register (YH)	Y Register (YL)
00	Stack Register (SH)	Stack Register (SL)
	Accumulator (B)	Accumulator (A)
Program Bank Register (PBR)	Program (PCH)	Counter (PCL)
00	Direct Register (DH)	Direct Register (DL)

Shaded blocks = 6502 registers



3 PIN FUNCTION DESCRIPTION

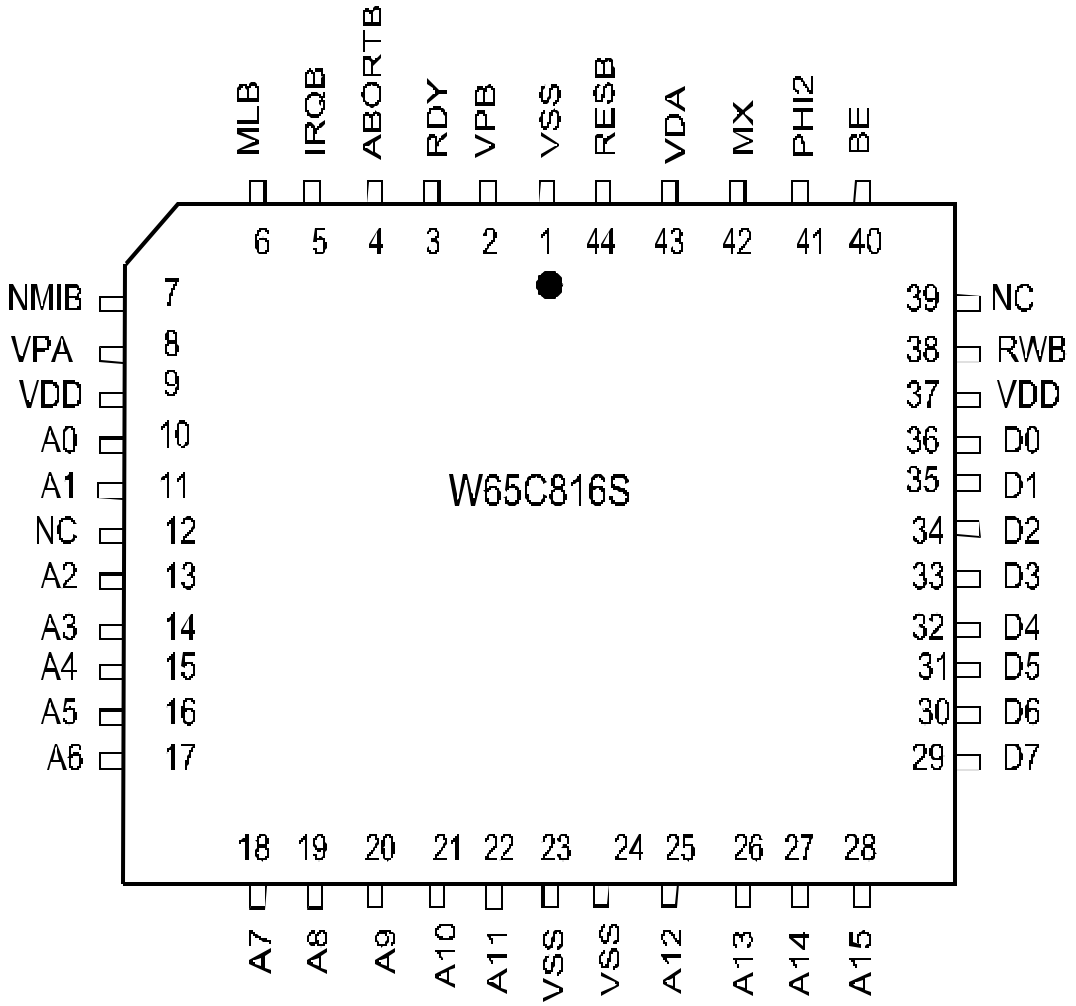


Figure 3-1 W65C816S 44 Pin PLCC Pinout

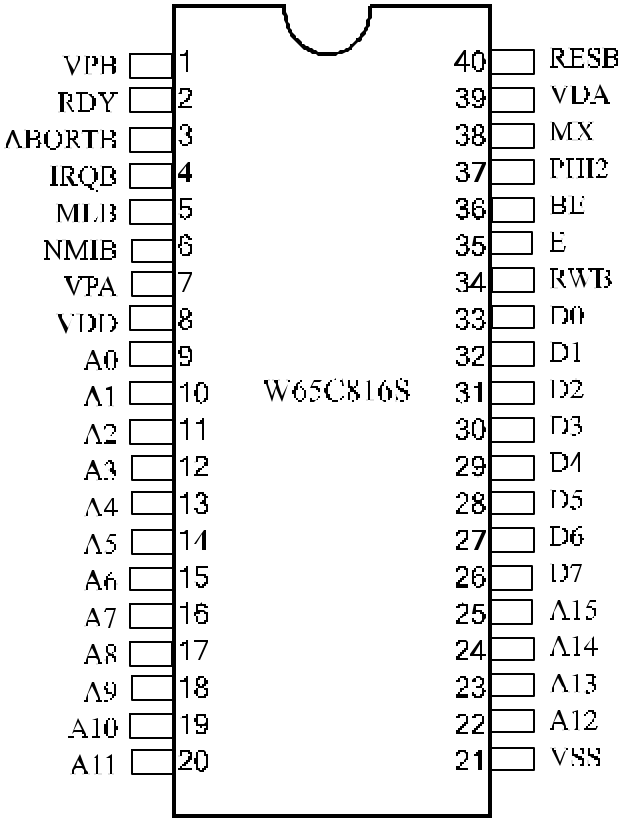


Figure 3-2 W65C816S 40 Pin PDIP Pinout

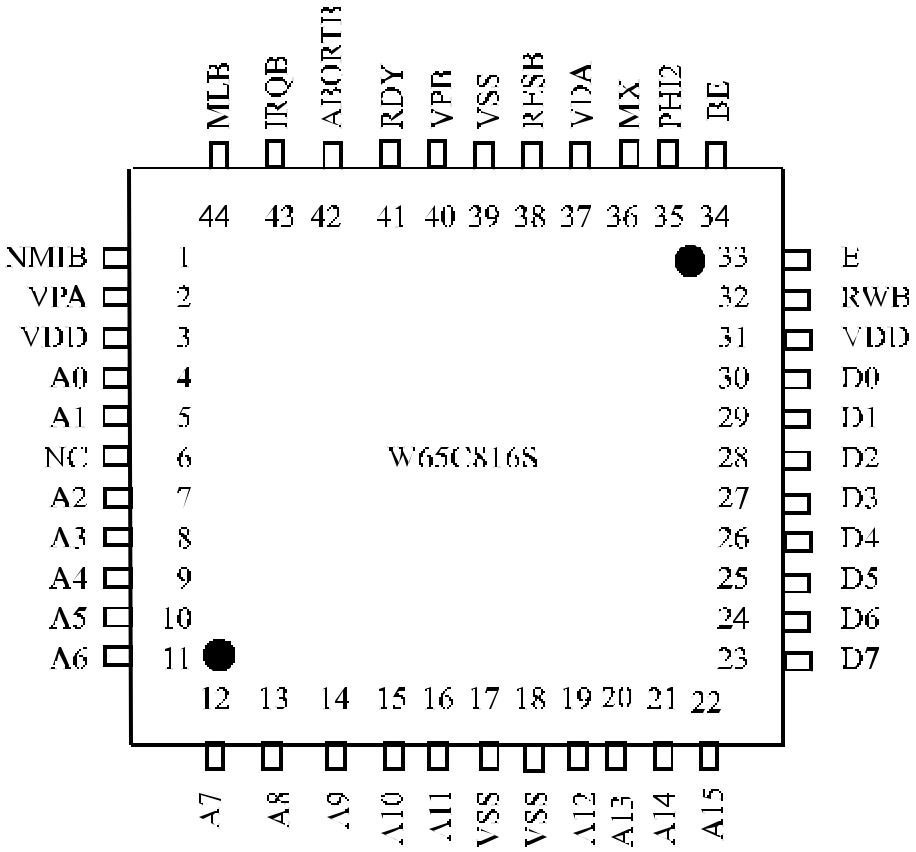


Figure 3-3 W65C816S 44 PIN QFP Pinout

Table 3-1 Pin Function Table

Pin	Description
A0-A15	Address Bus
ABORTB	Abort Input
BE	Bus Enable
PHI2	Phase 2 In Clock
D0-D7	Data Bus/Bank Address Bus
E	Emulation OR Native Mode Select
IRQB	Interrupt Request
MLB	Memory Lock
MX	Memory and Index Register Mode Select
NC	No Connect
NMIB	Non-Maskable Interrupt
RDY	Ready
RESB	Reset
RWB	Read/Write
VDA	Valid Data Address
VPB	Vector Pull
VPA	Valid Program Address
VDD	Positive Power Supply
VSS	Internal Logic Ground

3.1 Abort (ABORTB)

The Abort (ABORTB) negative pulse active input is used to abort instructions (usually due to an Address Bus condition). A negative transition will inhibit modification of any internal register during the current instruction. Upon completion of this instruction, an interrupt sequence is initiated. The location of the aborted OpCode is stored as the return address in stack memory. The Abort vector address is 00FFF8,9 (Emulation mode) or 00FFE8,9 (Native mode). Note that ABORTB is a pulse-sensitive signal; i.e., an abort will occur whenever there is a negative pulse (or level) on the ABORTB pin during a PHI2 clock.

3.2 Address Bus (A0-A15)

The sixteen Address Bus (A0-A15) output lines along with the bank address (multiplexed on the first half cycle of the Data Bus (D0-D7) pins) form the 24-bit Address Bus for memory and I/O exchange on the Data Bus. When using the W65C816S, the address lines may be set to the high impedance state by the Bus Enable (BE) signal.

3.3 Bus Enable (BE)

The Bus Enable (BE) input signal allows external control of the Address and Data Buffers, as well as the RWB signal. With Bus Enable high, the RWB and Address Buffers are active. The Data/Address Buffers are active during the first half of every cycle and the second half of a write cycle. When BE is low, these buffers are disabled. Bus Enable is an asynchronous signal.

3.4 Data/Bank Address Bus (D0-D7)

The Data/Bank Address Bus (D0-D7) pins provide both the Bank Address and Data. The bank address is present during the first half of a memory cycle, and the data value is read or written during the second half of the memory cycle. Two memory cycles are required to transfer 16-bit values. These lines may be set to the high impedance state by the Bus Enable (BE) signal.

3.5 Emulation Status (E)

The Emulation Status (E) output reflects the state of the Emulation (E) mode flag in the Processor Status (P) Register. This signal may be thought of as an OpCode extension and used for memory and system management.

3.6 Interrupt Request (IRQB)

The Interrupt Request (IRQB) negative level active input signal is used to request that an interrupt sequence be initiated. When the IRQB Disable (I) flag is cleared, a low input logic level initiates an interrupt sequence after the current instruction is completed. The Wait-for-Interrupt (WAI) instruction may be executed to ensure the interrupt will be recognized immediately. The Interrupt Request vector address is 00FFFE,F (Emulation mode) or 00FFEE,F (Native mode). Since IRQB is a level-sensitive input, an interrupt will occur if the interrupt source was not cleared since the last interrupt. Also, no interrupt will occur if the interrupt source is cleared prior to interrupt recognition. The IRQB signal going low causes 4 bytes of information to be pushed onto the stack before jumping to the interrupt handler. The first byte is PBR followed by PCH, PCL and P (Processor Status Register). These register values are used by the RTI instruction to return the processor to its original state prior to handling the IRQ interrupt (see Table 6-1)

3.7 Memory Lock (MLB)

The Memory Lock (MLB) active low output may be used to ensure the integrity of Read-Modify-Write instructions in a multiprocessor system. Memory Lock indicates the need to defer arbitration of the next bus cycle. Memory Lock is low during the last three or five cycles of ASL, DEC, INC, LSR, ROL, ROR, TRB, and TSB memory referencing instructions, depending on the state of the M flag.

3.8 Memory/Index Select Status (MX)

The Memory/Index Select Status (MX) multiplexed output reflects the state of the Accumulator (M) and Index (X) elect flags (bits 5 and 4 of the Processor Status (P) Register. Flag M is valid during PHI2 negative transition and Flag X is valid during PHI2 positive transition. These bits may be thought of as OpCode extensions and may be used for memory and system management.

3.9 Non-Maskable Interrupt (NMIB)

A negative transition on the Non-Maskable Interrupt (NMIB) input initiates an interrupt sequence. A high-to-low transition initiates an interrupt sequence after the current instruction is completed. The Wait for Interrupt (WAI) instruction may be executed to ensure that the interrupt will be recognized immediately. The Non-Maskable Interrupt vector address is 00FFFA,B (Emulation mode) or 00FFEA,B (Native mode). Since NMIB is an edge-sensitive input, an interrupt will occur if there is a negative transition while servicing a previous interrupt. No interrupt will occur if NMIB remains low after the negative transition was processed. The NMIB signal going low causes 4 bytes of information to be pushed onto the stack before jumping to the interrupt handler. The first byte on the stack is the PBR followed by the PCH, PCL and P, these register values are used by the RTI instruction to return the processor to its original state prior to the NMI interrupt.

3.10 Phase 2 In (PHI2)

Phase 2 In (PHI2) is the system clock input to the microprocessor. PHI2 can be held in either state to preserve the contents of internal registers and reduce power as a Standby mode.

3.11 Read/Write (RWB)

The Read/Write (RWB) output signal is used to control whether the microprocessor is "Reading" or "Writing" to memory. When the RWB is in the high state, the microprocessor is reading data from memory or I/O. When RBW is low the Data Bus contains valid data from the microprocessor which is to written to the addressed memory location. The RWB signal is set to the high impedance state when Bus Enable (BE) is low.

3.12 Ready (RDY)

The Ready (RDY) is a bi-directional signal. When it is an output it indicates that a Wait for Interrupt (WAI) instruction has been executed halting operation of the microprocessor. A low input logic level will halt the microprocessor in its current state. Returning RDY to the active high state releases the microprocessor to continue processing following the next PHI2 negative transition. The RDY signal is internally pulled low following the execution of a Wait for Interrupt (WAI) instruction, and then returned to the high state when a RESB, ABORTB, NMIB, or IRQB external interrupt is active. This feature may be used to reduce interrupt latency by executing the WAI instruction and waiting for an interrupt to begin processing. If the IRQB Disable flag has been set, the next instruction will be executed when the IRQB occurs. The processor will not stop after a WAI instruction if RDY has been forced to a high state. The Stop (STP) instruction has no effect on RDY. The RDY pin has an active pull-up and when outputting a low level, the pull-up is turned off to reduce power. The RDY pin can be wired ORed.

3.13 Reset (RESB)

The Reset (RESB) active low input is used to initialize the microprocessor and start program execution. The Reset input buffer has hysteresis such that a simple R-C timing circuit may be used with the internal pull-up device. The RESB signal must be held low for at least two clock cycles after VDD reaches operating voltage. Ready (RDY) has no effect while RESB is being held low. The stack pointer must be initialized by the user's software. During the Reset conditioning period the following processor initialization takes place:

Registers

D=0000	**SH=01, SL=***
DBR=00	**XH=00, XL=***
PBR=00	**YH=00, YL=***
	A=*

Signals

E=1	**VDA=0**
MX=1	**VPB=1**
RWB=1	**VPA=0**

P Register

N	V	M	X	D	I	Z	C/E
*	*	1	1	0	1	*	*/1

***=not initialized**

When Reset is brought high, an interrupt sequence is initiated

- STP and WAI instructions are cleared
- RWB remains in the high state during the stack address cycles.
- The Reset vector address is 00FFFC,D.(see Table 6-1 for Vectors)
- PC is loaded with the contents of 00FFFC,D

3.14 Valid Data Address (VDA) and Valid Program Address (VPA)

The Valid Data Address (VDA) and Valid Program Address (VDA) output signals indicate valid memory addresses when high and are used for memory or I/O address qualification.

VDA	VPA	
0	0	Internal Operation Address and Data Bus available. The Address Bus may be invalid.
0	1	Valid program address-may be used for program cache control.
1	0	Valid data address-may be used for data cache control.
1	1	OpCode fetch-may be used for program cache control and single step control

3.15 VDD and VSS

VDD is the positive supply voltage and VSS is system logic ground.

3.16 Vector Pull (VPB)

The Vector Pull (VPB) active low output indicates that a vector location is being addressed during an interrupt sequence. VPB is low during the last two interrupt sequence cycles, during which time the processor loads the PC with the interrupt handler vector location. The VPB signal may be used to select and prioritize interrupts from several sources by modifying the vector addresses.

4 ADDRESSING MODES

The W65C816S is capable of directly addressing 16 MBytes of memory. This address space has special significance within certain addressing modes, as follows:

4.1 Reset and Interrupt Vectors

The Reset and Interrupt Vectors use the majority of the fixed addresses between 00FFE0 and 00FFFF.

4.2 Stack

The Stack may be use memory from 000000 to 00FFFF. The effective address of Stack and Stack Relative addressing modes will be always be within this range.

4.3 Direct

The Direct addressing modes are usually used to store memory registers and pointers. The effective address generated by Direct, Direct,X and Direct,Y addressing modes is always in Bank 0 (000000-00FFFF).

4.4 Program Address Space

The Program Bank register is not affected by the Relative, Relative Long, Absolute, Absolute Indirect, and Absolute Indexed Indirect addressing modes or by incrementing the Program Counter from FFFF. The only instructions that affect the Program Bank register are: RTI, RTL, JML, JSL, and JMP Absolute Long. Program code may exceed 64K bytes although code segments may not span bank boundaries.

4.5 Data Address Space

The Data Address space is contiguous throughout the 16 MByte address space. Words, arrays, records, or any data structures may span 64 KByte bank boundaries with no compromise in code efficiency. The following addressing modes generate 24-bit effective addresses:

- Absolute a
- Absolute a,x
- Absolute a,y
- Absolute Long al
- Absolute Long Indexed al,x
- Direct Indexed Indirect (d,x)
- Direct Indirect (d)
- Direct Indirect Indexed (d),y
- Direct Indirect Long [d]
- Direct Indirect Long Indexed [d],y
- Stack Relative Indirect Indexed (d,x),y

The following addressing mode descriptions provide additional detail as to how effective addresses are calculated. Twenty-four addressing modes are available for the W65C816S.

4.5.1 Absolute - a

With Absolute (a) addressing the second and third bytes of the instruction form the low-order 16 bits of the effective address. The Data Bank Register contains the high-order 8 bits of the operand address.

Instruction:	OpCode	addrl	addrh
Operand	DBR	addrh	addrl

4.5.2 Absolute Indexed Indirect-(a,x)

With Absolute Indexed Indirect ((a,x)) addressing the second and third bytes of the instruction are added to the X Index Register to form a 16-bit pointer in Bank 0. The contents of this pointer are loaded in the Program Counter for the JMP instruction. The Program Bank Register is not changed.

Instruction:	OpCode	addrl	addrh
		addrh	addrl
			X Reg
	PBR	address	

then:
PC = (address)

4.5.3 Absolute Indexed with X - a,x

With Absolute Indexed with X (a,x) addressing the second and third bytes of the instruction are added to the X Index Register to form the low-order 16-bits of the effective address. The Data Bank Register contains the high-order 8 bits of the effective address.

Instruction:	OpCode	addrl	addrh
	DBR	addrh	addrl
	+		X Reg
Operand Address:	effective address		

4.5.4 Absolute Indexed with Y-a,y

With Absolute Indexed with Y (a,y) addressing the second and third bytes of the instruction are added to the Y Index Register to form the low-order 16-bits of the effective address. The Data Bank Register contains the high-order 8 bits of the effective address.

Instruction:	OpCode	addrl	addrh
	DBR	addrh	addrl
	+		Y Reg
Operand Address:	effective address		

4.5.5 Absolute Indirect-(a)

With Absolute Indirect ((a)) addressing the second and third bytes of the instruction form an address to a pointer in Bank 0. The Program Counter is loaded with the first and second bytes at this pointer. With the Jump Long (JML) instruction, the Program Bank Register is loaded with the third byte of the pointer.

Instruction:	OpCode	addrl	addrh	
Indirect		00	addrh	addrl

4.5.6 Absolute Long Indexed With X-al,x

With Absolute Long Indexed with X (al,x) addressing the second, third and fourth bytes of the instruction form a 24-bit base address. The effective address is the sum of this 24-bit address and the X Index Register.

Instruction:	OpCode	addrl	addrh	baddr
	baddr	addrh	addrl	
	+		X Reg	
Operand Address:	effective address			

4.5.7 Absolute Long-al

With Absolute Long (al) addressing the second, third and fourth byte of the instruction form the 24-bit effective address.

Instruction:	OpCode	addrl	addrh	baddr
Operand Address:	baddr	addrh	addrl	

4.5.8 Accumulator-A

With Accumulator (A) addressing the operand is the Accumulator.

4.5.9 Block Move-xyc

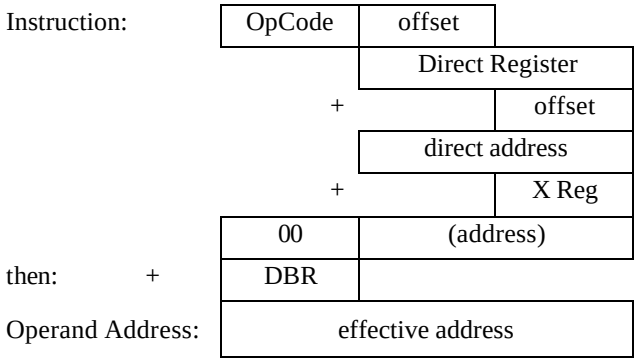
Block Move (xyc) addressing is used by the Block Move instructions. The second byte of the instruction contains the high-order 8 bits of the destination address and the Y Index Register contains the low-order 16 bits of the destination address. The third byte of the instruction contains the high-order 8 bits of the source address and the X Index Register contains the low-order bits of the source address. The C Accumulator contains one less than the number of bytes to move. The second byte of the block move instructions is also loaded into the Data Bank Register.

Instruction:	OpCode	dstbnk	srcbnk
		dstbnk Y DBR	
Source Address:		srcbnk	X Reg
Dest. Address;		DBR	Y Reg

Increment X and Y (MVN) or decrement X and Y (MVP) and decrement C (if greater than zero), then PC=PC+3.

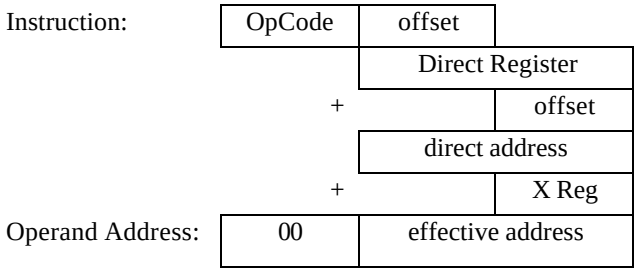
4.5.10 Direct Indexed Indirect-(d,x)

Direct Indexed Indirect ((d,x)) addressing is often referred to as Indirect X addressing. The second byte of the instruction is added to the sum of the Direct Register and the X Index Register. The result points to the X low-order 16 bits of the effective address. The Data Bank Register contains the high-order 8 bits of the effective address.



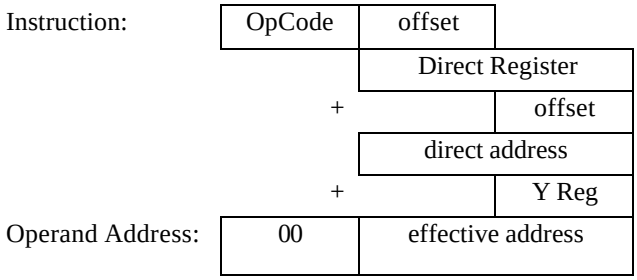
4.5.11 Direct Indexed with X-d,x

With Direct Indexed with X (d,x) addressing the second byte of the instruction is added to the sum of the Direct Register and the X Index Register to form the 16-bit effective address. The operand is always in Bank 0.



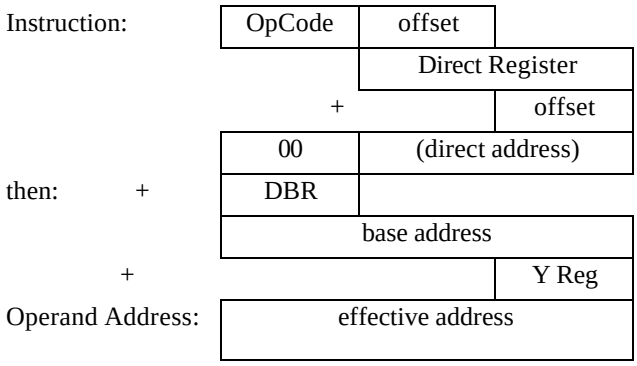
4.5.12 Direct Indexed with Y-d,y

With Direct Indexed with Y (d,y) addressing the second byte of the instruction is added to the sum of the Direct Register and the Y Index Register to form the 16-bit effective address. The operand is always in Bank 0.



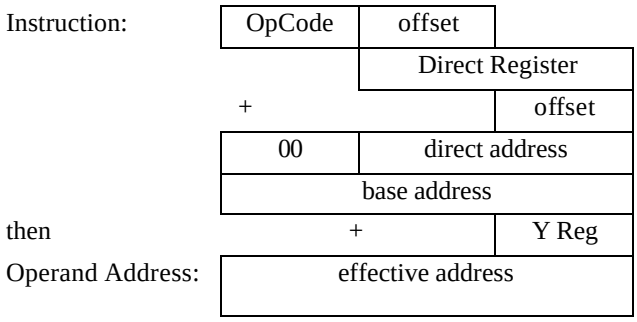
4.5.13 Direct Indirect Indexed-(d),y

Direct Indirect Indexed ((d),y) addressing is often referred to as Indirect Y addressing. The second byte of the instruction is added to the Direct Register (D). The 16-bit content of this memory location is then combined with the Data Bank register to form a 24-bit base address. The Y Index Register is added to the base address to form the effective address.



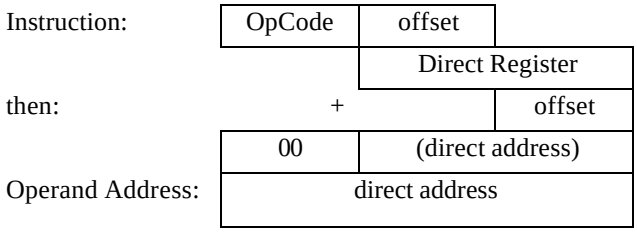
4.5.14 Direct Indirect Long Indexed-[d],y

With Direct Indirect Long Indexed ([d],y) addressing the 24-bit base address is pointed to by the sum of the second byte of the instruction and the Direct Register. The effective address is this 24-bit base address plus the Y Index Register.



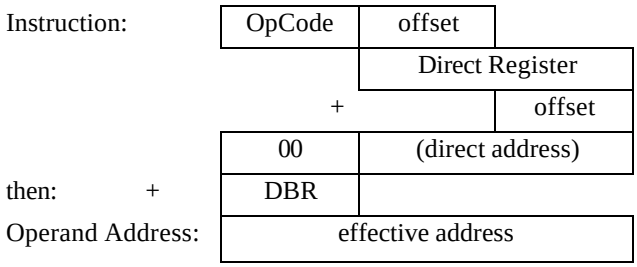
4.5.15 Direct Indirect Long-[d]

With Direct Indirect Long ([d]) addressing the second byte of the instruction is added to the Direct Register to form a pointer to the 24-bit effective address.



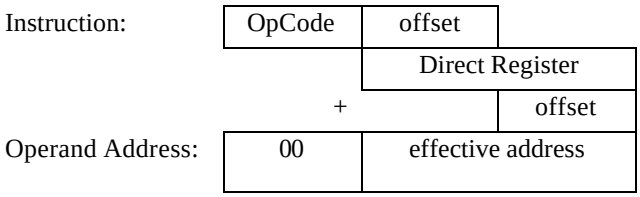
4.5.16 Direct Indirect-(d)

With Direct Indirect ((d)) addressing the second byte of the instruction is added to the Direct Register to form a pointer to the low-order 16 bits of the effective address. The Data Bank Register contains the high-order 8 bits of the effective address.



4.5.17 Direct-d

With Direct (d) addressing the second byte of the instruction is added to the Direct Register (D) to form the effective address. An additional cycle is required when the Direct Register is not page aligned (DL not equal 0). The Bank register is always 0.



4.5.18 Immediate-#

With Immediate (#) addressing the operand is the second byte (second and third bytes when in the 16-bit mode) of the instruction.

4.5.19 Implied-i

Implied (i) addressing uses a single byte instruction. The operand is implicitly defined by the instruction.

4.5.20 Program Counter Relative Long-rl

The Program Counter Relative Long (rl) addressing mode is used with only with the unconditional Branch Long instruction (BRL) and the Push Effective Relative instruction (PER). The second and third bytes of the instruction are added to the Program Counter, which has been updated to point to the OpCode of the next instruction. With the branch instruction, the Program Counter is loaded with the result. With the Push Effective Relative instruction, the result is stored on the stack. The offset is a signed 16-bit quantity in the range from -32768 to 32767. The Program Bank Register is not affected.

4.5.21 Program Counter Relative -r

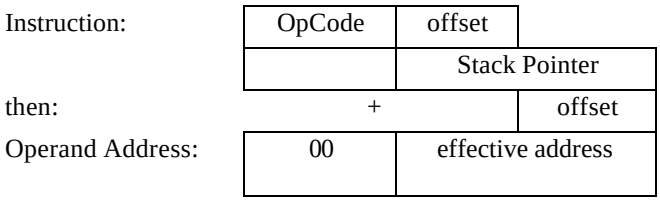
The Program Counter Relative (r) addressing is referred to as Relative Addressing and is used only with the Branch instructions. If the condition being tested is met, the second byte of the instruction is added to the Program Counter, which has been updated to point to the OpCode of the next instruction. The offset is a signed 8-bit quantity in the range from -128 to 127. The Program Bank Register is not affected.

4.5.22 Stack -s

Stack (s) addressing refers to all instructions that push or pull data from the stack, such as Push, Pull, Jump to Subroutine, Return from Subroutine, Interrupts, and Return from Interrupt. The bank address is always 0. Interrupt Vectors are always fetched from Bank 0.

4.5.23 Stack Relative - d,s

With Stack Relative (d,s) addressing the low-order 16 bits of the effective address is formed from the sum of the second byte of the instruction and the stack pointer. The high-order 8 bits of the effective address are always zero. The relative offset is an unsigned 8-bit quantity in the range of 0 to 255.



4.5.24 Stack Relative Indirect Indexed-(d,s),y

With Stack Relative Indirect Indexed ((d,s),y) addressing the second byte of the instruction is added to the Stack Pointer to form a pointer to the low-order 16-bit base address in Bank 0. The Data Bank Register contains the high-order 8 bits of the base address. The effective address is the sum of the 24-bit base address and the Y Index Register.

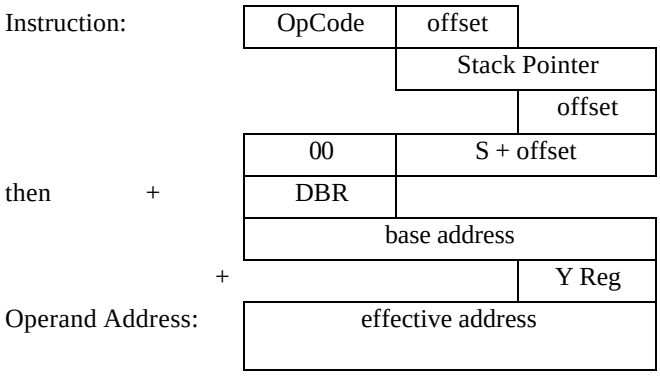


Table 4-1 Addressing Mode Summary

Address Mode	Instruction Times in Memory Cycle		Memory Utilization in Number of Program Sequence Bytes	
	Original 8-bit NMOS 6502	New W65C816S	Original 8-bit NMOS 6502	New W65C816S
Absolute	4 (5)	4 (3,5)	3	3
Absolute Indexed Indirect (Jump)	-	6	-	3
Absolute Indirect (Jump)	5	5	3	3
Absolute Long	-	5 (3)	-	4
Absolute Long, X	-	5 (3)	-	4
Absolute, X	4 (1,5)	4 (1,3,5)	3	3
Absolute, Y	4 (1)	4 (1,3)	3	3
Accumulator	2	2	1	1
Block Move (xyc)	-	7	-	3
Direct	3 (5)	3 (3,4,5)	2	2
Direct Indexed Indirect (d,x)	6	6 (3,4)	2	2
Direct Indirect	-	5 (3,4)	-	2
Direct Indirect Indexed (d),y	5 (1)	5 (1,3,4)	2	2
Direct Indirect Indexed Long [d],y	-	6 (3,4)	-	2
Direct Indirect Long	-	6 (3,4)	-	2
Direct, X	4 (5)	4 (3,4,5)	2	2
Direct, Y	4	4 (3,4)	2	2
Immediate	2	2 (3)	2	2 (3)
Implied	2	2	1	1
Relative	2 (1,2)	2 (2)	2	2
Relative Long	-	3 (2)	-	3
Stack	3-7	3-8	1-3	1-4
Stack Relative	-	4 (3)	-	2
Stack Relative Indirect Indexed	-	7 (3)	-	2

Notes (these are indicated in parentheses):

1. Page boundary, add 1 cycle if page boundary is crossed when forming address.
2. Branch taken, add 1 cycle if branch is taken.
3. M = 0 or X = 0, 16 bit operation, add 1 cycle, add 1 byte for immediate.
4. Direct register low (DL) not equal zero, add 1 cycle.
5. Read-Modify-Write, add 2 cycles for M = 1, add 3 cycles for M = 0.

5 TIMING, AC AND DC CHARACTERISTICS

5.1 Absolute Maximum Ratings

Table 5-1 Absolute Maximum Ratings

Rating	Symbol	Value
Supply Voltage	VDD	-0.3 to +7.0V
Input Voltage	VIN	-0.3 to VDD +0.3V
Storage Temperature	TS	-55°C to +150°C

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

Note: Exceeding these ratings may result in permanent damage. Functional operation under the conditions is not implied.

5.2 DC Characteristics TA = -40°C to +85°C

Table 5-2 DC Characteristics

Symbol		5.0 +/ - 5%		3.3 +/ - 10%		3.0 +/ - 5%		2.5 +/ - 5%		1.8 +/ - 5%		Units
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
VDO		4.25	5.25	3.0	3.6	2.85	3.15	2.375	2.625	1.71	1.89	V
Vih	Input High Voltage ABORTB, BE, Data, IRQB, RDY, NMIB, PHI2, RESB	VDDx0.8	VDD+0.3	VDDx0.8	VDD+0.3	VDDx0.8	VDD+0.3	VDDx0.8	VDD+0.3	VDDx0.8	VDD+0.3	V
Vil	Input Low Voltage ABORTB, BE, Data, IRQB, RDY, NMIB, PHI2, RESB	VSS-0.3	VDDx0.2	VSS-0.3	VDDx0.2	VSS-0.3	VDDx0.2	VSS-0.3	VDDx0.2	VSS-0.3	VDDx0.2	V
Ipup	RDY Input Pullup-Current (VIN=VDDx0.8)	5	20	5	20	5	20	2	10	2	10	µA
Iin	Input Leakage Current (Vin=0.4 to 2.4) PHI2, Address, Data, RWB, (Off state, BE=0), All other inputs	-0.2	0.2	-0.2	0.2	-0.2	0.2	-0.2	0.2	-0.2	0.2	µA
Ioh	Output High Voltage (Vol=VDD-0.4V) Address, Data, E, MLB, MX, RWB, VDA, VPA, VPB	700	-	300	-	300	-	200	-	100	-	µA
Iol	Output Low Voltage (Vol=VSS+0.4V) Address, Data, E, MLB, MX, RWB, VDA, VPA, VPB	1.6	-	1.6	-	1.6	-	1.0	-	.5	-	mA
Idd	Supply Current (no load) Supply Current (core)	- -	2.0 1.0	-	1.5 0.6	-	1.5 0.5	-	1.0 0.4	-	0.75 0.30	mA/ MHz
Isby	Standby Current (No Load, Data Bus = VSS or VDD) ABORTB, BE, IRQB, NMIB, RESB, PHI2=VDD			-	1	-	1	-	1	-	1	µA
Cin Cts	Capacitance (Vin=0V, TA=25°C, f=1MHz) ABORTB, BE, IRQB, NMIB, PHI2, RBW, RESB, RDY, Address, Data, R/W - (Off state) * Not inspected during production test; verified on a sample basis.	-	5	-	5	-	5	-	5	-	5	pF pF

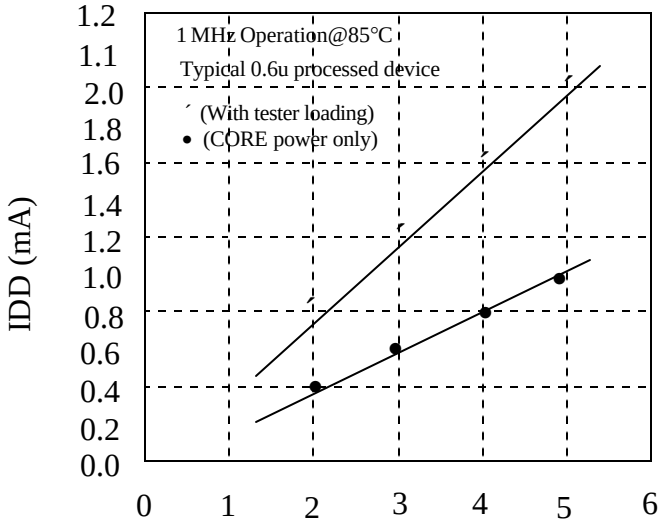


Table 5-3 IDD vs. VDD

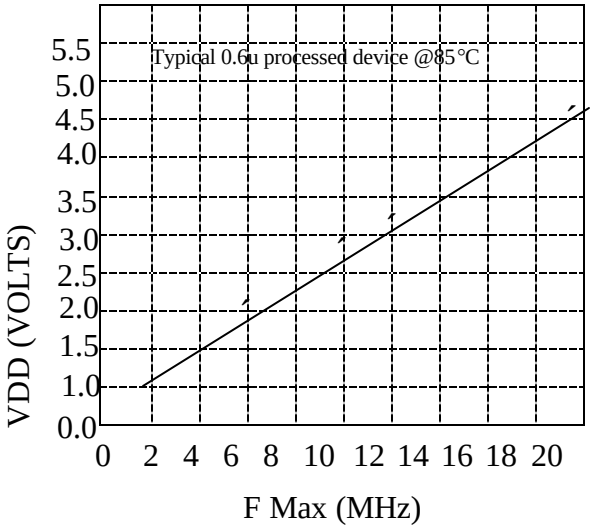


Table 5- 4 F Max vs. VDD

6.0

Table 5-4 W65C816S AC Characteristics

Symbol	Parameter	5.0 +/- 5%		3.3 +/- 10%		3.0 +/- 5%		2.5 +/- 5%		1.8 +/- 5%		Units
		14MHz		8MHz		8MHz		4MHz		2MHz		
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
VDD		4.75	5.25	3.0	3.6	2.85	3.15	2.375	2.675	1.71	1.89	V
tCYC	Cycle Time	70	DC	125	DC	125	DC	250	DC	500	DC	nS
tPWL	Clock Pulse Width Low	35	-	63	-	63	-	125	-	250	-	nS
tPWH	Clock Pulse Width High	35	-	62	-	62	-	125	-	250	-	nS
tF,tR	Fall Time, Rise Time	-	5	-	5	-	5	-	5	-	5	nS
tAH	A0-A15 Hold Time	10	-	10	-	10	-	20	-	40	-	nS
tADS	A0-A15 Setup Time	-	30	-	40	-	40	-	75	-	150	nS
tBH	BA0-BA7 Hold Time	10	-	10	-	10	-	20	-	40	-	nS
tBAS	BA0-BA7 Setup Time	-	33	-	40	-	40	-	75	-	150	nS
tACC	Access Time	30	-	70	-	70	-	130	-	365	-	nS
tDHR	Read Data Hold Time	10	-	10	-	10	-	20	-	40	-	nS
tMDS	Write Data Delay Time	-	30	-	40	-	40	-	70	-	140	nS
tDHW	Write Data Hold Time	10	-	10	-	10	-	20	-	40	-	nS
tPCS	Processor Control Setup Time	10	-	15	-	15	-	30	-	60	-	nS
tPCH	Processor Control Hold Time	10	-	10	-	10	-	20	-	40	-	nS
tEH	E, MX Output Hold Time	-	5	-	5	-	5	-	5	-	5	nS
tES	E, MX Output Setup Time	10	-	15	-	15	-	30	-	60	-	nS
CEXT	Capacitive Load (1)	-	35	-	35	-	35	-	35	-	35	Pf
tBVD	be TO Valid Data (2)	-	25	-	30	-	30	-	60	-	120	nS

1. Test or loading on all outputs.
2. BE to High Impedance State is not testable but should be the same amount of time as BE to Valid Data.

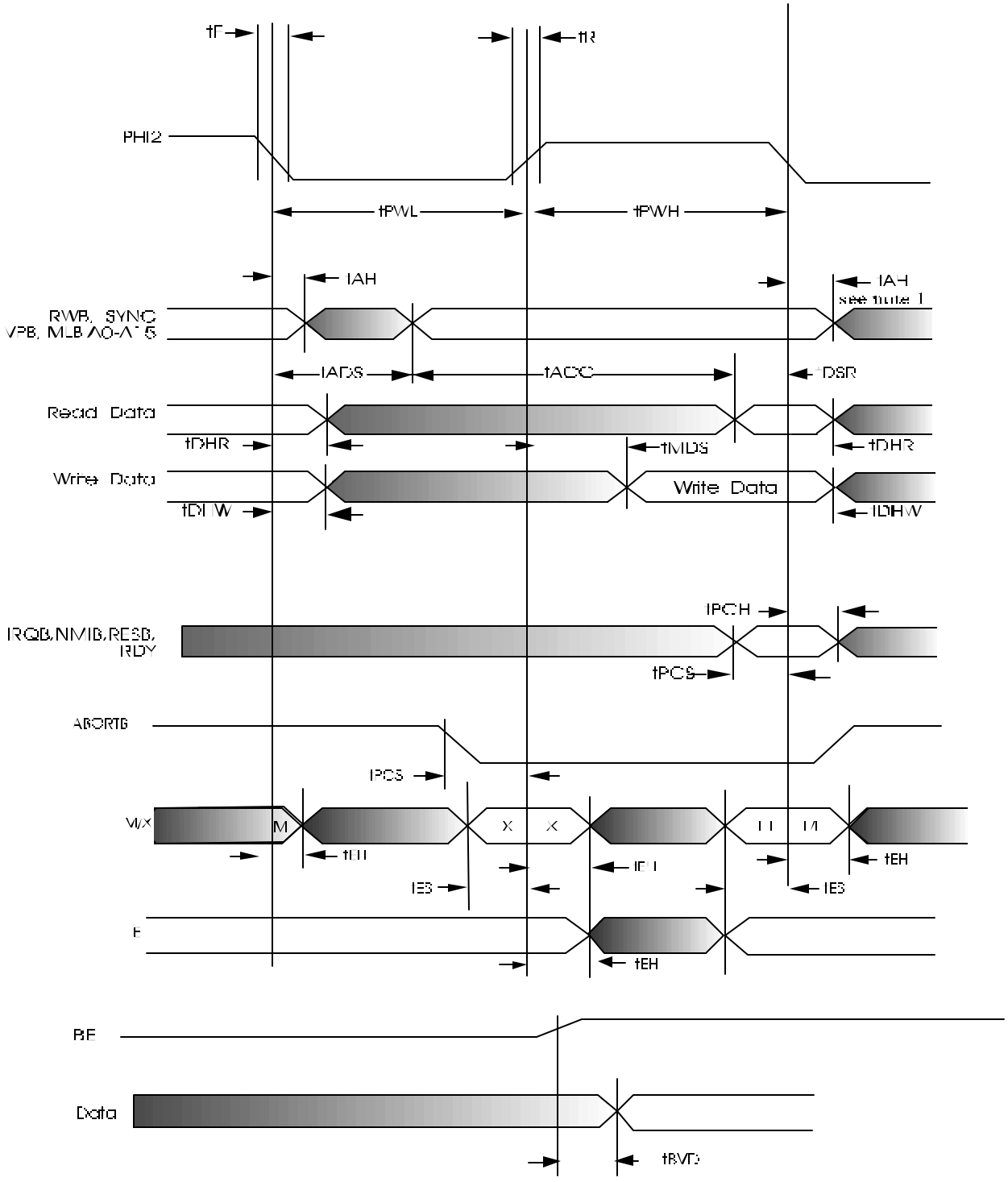


Figure 5-1 General Timing Diagram

1. Timing measurement points are 50% VDD.

6 OPERATION TABLES

Table 6-1 W65C816S Instruction Set-Alphabetical Sequence

(continued on following page)

1.	ADC	Add Memory to Accumulator with Carry	30.	INY	Increment Index Y by One
2.	AND	"AND" Memory with Accumulator	31.	JML	Jump Long
3.	ASL	Shift One Bit Left, Memory or Accumulator	32.	JMP	Jump to New Location
4.	BCC	Branch on Carry Clear (C=0)	33.	JSL	Jump Subroutine Long
5.	BCS	Branch on Carry Set (C=1)	34.	JSR	Jump to News Location Saving Return
6.	BEQ	Branch if Equal (Z=1)	35.	LDA	Load Accumulator with Memory
7.	BIT	Bit Test	36.	LDX	Load Index X with Memory
8.	BMI	Branch if Result Minus (N=1)	37.	LDY	Load Index Y with Memory
9.	BNE	Branch if Not Equal (Z=0)	38.	LSR	Shift One Bit Right (Memory or Accumulator)
10	BPL	Branch if Result Plus (N=0)	39.	MVN	Block Move Negative
11	BRA	Branch Always	40.	MVP	Block Move Positive
12	BRK	Force Break	41.	NOP	No Operation
13	BRL	Branch Always Long	42.	ORA	"OR" Memory with Accumulator
14	BVC	Branch on Overflow Clear (V=0)	43.	PEA	Push Effective Absolute Address on Stack (or Push Immediate Data on Stack)
15	BVS	Branch on Overflow Set (V=1)	44.	PEI	Push Effective Absolute Address on Stack (Or Push Direct Data on Stack)
16	CLC	Clear Carry Flag	45.	PER	Push Effective Program Counter Relative Address on Stack
17	CLD	Clear Decimal Mode	46.	PHA	Push Accumulator on Stack
18	CLI	Clear Interrupt Disable Bit	47.	PHB	Push Data Bank Register on Stack
19	CLV	Clear Overflow Flag	48.	PHD	Push Direct Register on Stack
20	CMP	Compare Memory and Accumulator	49.	PHK	Push Program Bank Register on Stack
21	COP	Coprocessor	50.	PHP	Push Processor Status on Stack
22	CPX	Compare Memory and Index X	51.	PHX	Push Index X on Stack
23	CPY	Compare Memory and Index Y	52.	PHY	Push Index Y on Stack
24	DEC	Decrement Memory or Accumulator by One	53.	PLA	Pull Accumulator from Stack
25	DEX	Decrement Index X by One	54.	PLB	Pull Data Bank Register from Stack
26	DEY	Decrement Index Y by One	55.	PLD	Pull Direct Register from Stack
27	EOR	"Exclusive OR" Memory with Accumulator	56.	PLP	Pull Processor Status from Stack
28	INC	Increment Memory or Accumulator by One	57.	PLX	Pull Index X from Stack
29	INX	Increment Index X by One	58.	PLY	Pull Index Y from Stack

59.	REP	Reset Status Bits	76.	TAY	Transfer Accumulator to Index Y
60.	ROL	Rotate One Bit Left (Memory or Accumulator)	77.	TCD	Transfer C Accumulator to Direct Register
61.	ROR	Rotate One Bit Right (Memory or Accumulator)	78.	TCS	Transfer C Accumulator to Stack Pointer Register
62.	RTI	Return from Interrupt	79.	TDC	Transfer Direct Register to C Accumulator
63.	RTL	Return from Subroutine Long	80.	TRB	Test and Reset Bit
64.	RTS	Return from Subroutine	81.	TSB	Test and Set Bit
65.	SBC	Subtract Memory from Accumulator with Borrow	82.	TSC	Transfer Stack Pointer Register to C Accumulator
66.	SEP	Set Processor Status Bit	83.	TSX	Transfer Stack Pointer Register to Index X
67.	SEC	Set Carry Flag	84.	TXA	Transfer Index X to Accumulator
68.	SED	Set Decimal Mode	85.	TXS	Transfer Index X to Stack Pointer Register
69.	SEI	Set Interrupt Disable Status	86.	TXY	Transfer Index X to Index Y
70.	STA	Store Accumulator in Memory	87.	TYA	Transfer Index Y to Accumulator
71.	STP	Stop the Clock	88.	TYX	Transfer Index Y to Index X
72.	STX	Store Index X in Memory	89.	WAI	Wait for Interrupt
73.	STY	Store Index Y in Memory	90.	WDM	Reserved for future use
74.	STZ	Store Zero in Memory	91.	XBA	Exchange B and A Accumulator
75.	TAX	Transfer Accumulator in Index X	92.	XCE	Exchange Carry and Emulation Bits

Table 6-2 Emulation Mode Vector Locations (8-bit Mode)

Address	Label	Function
00FFFE,F	IRQB/BRK	Hardware/Software
00FFFC,D	RESETB	Hardware
00FFFA,B	NMIB	Hardware
00FFF8,9	ABORTB	Hardware
00FFF6,7	(Reserved)	Hardware
00FFF4,5	COP	Software
00FFF2,3	(Reserved)	
00FFF0,1	(Reserved)	

Table 6-3 Native Mode Vector Locations (16-bit Mode)

Address	Label	Function
00FFFE,F	IRQB	Hardware
00FFFC,D	(Reserved)	
00FFFA,B	NMIB	Hardware
00FFF8,9	ABORTB	
00FFF6,7	BRK	Software
00FFF4,5	COP	Software
00FFF2,3	(Reserved)	
00FFF0,1	(Reserved)	

The VP output is low during the two cycles used for vector location access. When an interrupt is executed, D=0 and I=1 in Status Register P.
Page Left Blank Intentionally

Table 6-4 OpCode Matrix

MSD	LSD																MSD
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK s 7,2	ORA(d,x) 6,2	COP s 7,2 *	ORA d,s 4,2 *	TBS d 5,2 •	ORA d 3,2	ASL d 5,2 *	ORA [d] 6,2	PHP s 3,1	ORA# 2,2	ASL A 2,1	PHD s 4,1 *	TSB a 6,3 •	ORA a 4,3	ASL a 6,3	ORA al 5,4 *	0
1	BPL r 2,2	ORA(d),y 5,2	ORA(d) 5,2 •	ORA(d,s),y 7,2 *	TRB d 5,2 •	ORA d,x 4,2	ASL d,x 6,2 *	ORA[d],y 6,2	CLC i 2,1	ORA a,y 4,3	INC A 2,1 •	TCS i 2,1 *	TRB a 6,3 •	ORA a,x 4,3	ASL a,x 7,3	ORA al,x 5,4 *	1
2	JSR a 6,3	AND(d,x) 6,2	JSL al 8,4 *	AND d,s 4,2 *	BIT d 3,2	AND d 3,2	ROL d 5,2 *	AND [d] 6,2	PLP s 4,1	AND# 2,2	ROL A 2,1	PLD s 5,1 *	BIT a 4,3	AND a 4,3	ROL a 6,3	AND al 5,4 *	2
3	BMI r 2,2	AND(d),y 5,2	AND(d) 5,2 •	AND(d,s),y 7,2 *	BIT d,x 4,2 •	AND d,x 4,2	ROL d,x 6,2 *	AND[d],y 6,2	SEC i 2,1	AND a,y 4,3	DEC A 2,1 •	TSC i 2,1 *	BIT a,x 4,3 •	AND a,x 4,3	ROL a,x 7,3	AND al,x 5,4 *	3
4	RTI s 7,1	EOR(d,x) 6,2	WDM 2,2 *	EOR d,s 4,2 *	MVP xyc 7,3 *	EOR d 3,2	LSR d 5,2 *	EOR [d] 6,2	PHA s 3,1	EOR # 2,2	LSR A 2,1	PHK s 3,1 *	JMP a 3,3	EOR a 4,3	LSR a 6,3	EOR al 5,4 *	4
5	BVC r 2,2	EOR (d),y 5,2	EOR (d) 5,2 •	EOR(d,s),y 7,2 *	MVN xyc 7,3 *	EOR d,x 4,2	LSR d,x 6,2 *	EOR[d].y 6,2	CLI i 2,1	EOR a,y 4,3	PHY s 3,1 •	TCD i 2,1 *	JMP al 4,4 *	EOR a,x 4,3	LSR a,x 7,3	EOR al,x 5,4 *	5
6	RTS s 6,1	ADC(d,x) 6,2	PER s 6,3 *	ADC d,s 4,2 *	STZ d 3,2 •	ADC d 3,2	ROR d 5,2 *	ADC[d] 6,2	PLA s 4,1	ADC# 2,2	ROR A 2,1	RTL s 6,1 *	JMP (a) 5,3	ADC a 4,3	ROR a 6,3	ADC al 5,4 *	6
7	BVS r 2,2	ADC(d),y 5,2	ADC(d) 5,2 •	ADC(d,s),y 7,2 *	STZ d,x 4,2 •	ADC d,x 4,2	ROR d,x 6,2 *	ADC[d],y 6,2	SEI i 2,1	ADC a,y 4,3	PLY s 4,1	TDC i 2,1 *	JMP(a,x) 6,3 •	ADC a,x 4,3	ROR a,x 7,3	ADC al,x 5,4 *	7
8	BRA r 2,2 •	STA(d,x) 6,2	BRL rl 3,3 *	STA d,s 4,2 *	STY d 3,2	STA d 3,2	STX d 3,2 *	STA[d] 6,2	DEY i 2,1	BIT # 2,2 •	TXA i 2,1	PHB s 3,1 *	STY a 4,3	STA a 4,3	STX a 4,3	STA al 5,4 *	8
9	BCC r 2,2	STA(d),y 6,2	STA (d) 5,2 •	STA(d,s),y 7,2 *	STY d,x 4,2	STA d,x 4,2	STX d,y 4,2 *	STA[d],y 6,2	TYA i 2,1	STA a,y 5,3	TXS i 2,1	TXY i 2,1 *	STZ a 3,4 •	STA a,x 5,3	STZ a,x 5,3 •	STA al,x 5,4 *	9
A	LDY # 2,2	LDA (d,x) 6,2	LDX # 2,2	LDA d,s 4,2 *	LDY d 3,2	LDA d 3,2	LDX d 3,2 *	LDA [d] 6,2	TAY i 2,1	LDA# 2,2	TAX i 2,1	PLB s 4,1 *	LDY a 4,3	LDA a 4,3	LDX a 4,3	LDA al 5,4 *	A
B	BCS r 2,2	LDA (d),y 5,2	LDA (d) 5,2 •	LDA(d,s),y 7,2 *	LDY d,x 4,2	LDA d,x 4,2	LDX d,y 4,2 *	LDA[d],y 6,2	CLV i 2,1	LDA a,y 4,3	TSX i 2,1	TYX i 2,1 *	LDY a,x 4,3	LDA a,x 4,3	LDX a,y 4,3	LDA al,x 5,4 *	B
C	CPY # 2,2	CMP (d,x) 6,2	REP # 3,2 *	CMP d,s 4,2 *	CPY d 3,2	CMP d 3,2	DEC d 5,2 *	CMP [d] 6,2	INY i 2,1	CMP # 2,2	DEX i 2,1	WAI i 3,1 •	CPY a 4,3	CMP a 4,3	DEC a 6,3	CMP al 5,4 *	C
D	BNE r 2,2	CMP (d),y 5,2	CMP (d) 5,2 •	CMP(d,s),y 7,2 *	PEI s 6,2 *	CMP d,x 4,2	DEC d,x 6,2 *	CMP[d],y 6,2	CLD i 2,1	CMP a,y 4,3	PHX s 3,1 •	STP i 3,1 •	JML (a) 6,3 *	CMP a,x 4,3	DEC a,x 7,3	CMP al,x 5,4 *	D
E	CPX # 2,2	SBC (d,x) 6,2	SEP # 3,2 *	SBC d,s 4,2 *	CPX d 3,2	SBC d 3,2	INC d 5,2 *	SBC [d] 6,2	INX i 2,1	SBC # 2,2	NOP i 2,1	XBA i 3,1 *	CPX a 4,3	SBC a 4,3	INC a 6,3	SBC al 5,4 *	E
F	BEQ r 2,2	SBC (d),y 5,2	SBC (d) 5,2 •	SBC(d,s),y 7,2 *	PEA s 5,3 *	SBC d,x 4,2	INC d,x 6,2 *	SBC[d],y 6,2	SED i 2,1	SBC a,y 4,3	PLX s 4,1 •	XCE i 2,1 *	JSR (a,x) 6,3 *	SBC a,x 4,3	INC a,x 7,3	SBC al,x 5,4 *	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

* = Old instruction with new addressing modes

• = New Instruction

Table 6-5 Operation, Operation Codes, and Status Register (continued on following 4 pages)

Mnemonic	Operation + Add - Subtract ^ AND v OR ⊻ Exclusive OR	a 1	A 2	a,x 3	a,y 4	al 5	al,x 6	(a) 7	(a,x) 8	d 9	d,s 10	d,x 11	d,y 12	(d) 13	[d] 14	(d,s),y 15	(d,x) 16	(d),y 17	[d],y 18	i 19	r 20	rl 21	s 22	xyc 23	# 24	Processor Status Code 7 N	6 V	5 1	4 B	3 D	2 1	1 Z	0 C
ADC	A+M+C→A	6D		7D	79	6F	7F	17		65	63	75	11	72	67	73	61	71	77						69	N	V	.	.	.	.	Z	C
AND	A^M→A	2D		3E	39	2F	3F			25	23	36		32	27	33	91	31	37						29	N	.	.	.	.	.	Z	.
ASL	C←15/7 6 5 4 3 2 10 ←0	0E	0A	1E						06		16														N	.	.	.	.	.	Z	C
BCC	Branch if C = 0																				90					.	.	.	.	.	.	.	.
BCS	Branch if C = 1																				B0					.	.	.	.	.	.	.	.
BEQ	Branch if Z = 1																				F0					.	.	.	.	.	.	.	.
BIT	A ^ M (Note 1)	2C		3C						24		34													89	M_7	M_6	.	.	.	.	Z	.
BMI	Branch if N = 0																				30					.	.	.	.	.	.	.	.
BNE	Branch if Z = 0																				D0					.	.	.	.	.	.	.	.
BPL	Branch if N = 0																				10					.	.	.	.	.	.	.	.
BRA	Branch Always																				80					.	.	.	.	.	.	.	.
BRK	Break (Note 2)																						00			.	.	.	•	0	1	.	.
BRL*	Branch Long Always																					82				.	.	.	.	.	.	.	.
BVC	Branch if V = 0																				50					.	.	.	.	.	.	.	.
BVS	Branch if V = 1																				70					.	.	.	.	.	.	.	.
CLC	C → 0																			18						.	.	.	.	.	.	.	0
CLD	0 → D																			D8						.	.	.	.	0	.	.	.
CLI	0 → 1																			58						.	.	.	.	.	0	.	.
CLV	0 → V																			B8						.	0	.	.	.	.	.	.
CMP	A-M	CD		DD	D9	CF	DF			C5	C3	D5		D2	C7	D3	C1	D1	D7						C9	N	.	.	.	.	.	Z	C
COP*	Co-Processor																						02			.	.	.	.	0	1	.	.

Table 6-5 (continued)

Mnemonic	Operation + Add - Subtract ^ AND v OR ⊻ Exclusive OR	a	A	a,x	a,y	al	al,x	(a)	(a,x)	d	d,s	d,x	d,y	(d)	[d]	(d,s),y	(d,x)	(d),y	[d],y	i	r	rl	s	xyc	#	Processor Status Code							
																										7	6	5	4	3	2	1	0
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N	V	1	B	D	I	Z	C
CPX	X-M	EC								E4															E0	N	.	.	.	.	.	Z	C
CPY	Y-M	CC								C4															C0	N	.	.	.	.	.	Z	C
DEC	Decrement	CE	3A	DE						C6		D6														N	.	.	.	.	.	Z	.
DEX	X-1 → A																			CA						N	.	.	.	.	.	Z	.
DEY	Y-1 → Y																			88						N	.	.	.	.	.	Z	.
EOR	A ⊻ M → A	4D		5D	59	4F	5F		5D	45	43	55		52	47	53	41	51	57						49	N	.	.	.	.	.	Z	.
INC	Increments	EE	TA	FE						E6		F6														N	.	.	.	.	.	Z	.
INX	X+1 → X																			E8						N	.	.	.	.	.	Z	.
INY	Y+1 → Y																			C8						N	.	.	.	.	.	Z	.
JML*	Jump Long to new location							DC																		.	.	.	.	.	.	.	.
JMP	Jump to new location	4C				5C		6C	7C																	.	.	.	.	.	.	.	.
JSL	Jump long to Subroutine					22																				.	.	.	.	.	.	.	.
JSR	Jump to Subroutine	20							FC																	.	.	.	.	.	.	.	.
LDA	M → A	AD		BD	B9	AF	BF			A5	A3	B5		B2	A7	B3	A1	B1	B7						A9	N	.	.	.	.	.	Z	.
LDX	M → X	AE			BE					A6			B6												A2	N	.	.	.	.	.	Z	.
LDY	M → Y	AC		BC						A4		B4													A0	N	.	.	.	.	.	Z	.
LSR	0 → 15/7 6 5 4 3 2 1 0 → C	4E	4A	5E						46		56														0	.	.	.	.	.	Z	C
MVN*	M→M NEGATIVE																							54		.	.	.	.	.	.	.	.
MVP*	M→M POSITIVE																							44		.	.	.	.	.	.	.	.
NOP	No Operation																			EA						.	.	.	.	.	.	.	.
ORA	A V M → A	0D		1D	19	0F	1F			05	03	15		12	07	13	01	11	17						09	N	.	.	.	.	.	Z	.

Table 6-5 (continued)

Mnemonic	Operation + Add - Subtract ^ AND v OR ⊻ Exclusive OR	a	A	a,x	a,y	al	al,x	(a)	(a,x)	d	d,s	d,x	d,y	(d)	[d]	(d,s),y	(d,x)	(d),y	[d],y	i	r	rl	s	xyc	#	Processor Status Code 7	6	5	4	3	2	1	0
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N	V	1	B	D	I	Z	C
PEA*	Mpc+1, Mpc+2 ? Ms-1, Ms S-2 ? S																						F4			.	.	.	.	.	.	.	.
PEI*	M(d), M(d+1) ? Ms-1, Ms S-2 ? S																						D4			.	.	.	.	.	.	.	.
PER*	Mpc+rl, Mpc+rl+1 ? Ms-1, Ms S-2 ? S																						62			.	.	.	.	.	.	.	.
PHA	A → Ms, S-1 → S																						48			.	.	.	.	.	.	.	.
PHB	DBR ? Ms, S-1 ? S																						8B			.	.	.	.	.	.	.	.
PHD*	D ? Ms, Ms-1, S-2 ? S																						0B			.	.	.	.	.	.	.	.
PHK*	PBR ? Ms, S-1 ? S																						4B			.	.	.	.	.	.	.	.
PHP	P → Ms, S-1 → S																						08			.	.	.	.	.	.	.	.
PHX	X → Ms, S-1 → S																						DA			.	.	.	.	.	.	.	.
PHY	Y → Ms, S-1 → S																						5A			.	.	.	.	.	.	.	.
PLA	S + 1 → S, Ms→ A																						68			N	.	.	.	.	.	Z	.
PLB*	S + 1 → S, Ms→ DBR																						AB			N	.	.	.	.	.	Z	.
PLD*	S+ 2→ S, Ms – 1, Ms ? D																						2B			N	.	.	.	.	.	Z	.
PLP	S + 1 → S, Ms→ P																						28			N	V	M	X	D	1	Z	C
PLX	S + 1 → S, Ms→ X																						FA			N	.	.	.	.	.	Z	.
PLY	S + 1 → S, Ms→ Y																						7A			N	.	.	.	.	.	Z	
REP*	M^P →P																								C2	N	V	M	X	D	1	Z	C
ROL	C ← 15/7 6 5 4 3 2 1 0 ← C	2E	2A	3E						26		36														N	.	.	.	.	.	Z	C
ROR	C →15/7 6 5 4 3 2 1 0 → C	6E	6A	7E						66		76														N	.	.	.	.	.	Z	C
RTI	Return from Interrupt																						40			N	V	M	X	D	1	Z	C
RTL*	Return from Sub. Long																						6B			.	.	.	.	.	.	.	.

Table 6-5 (continued)

Mnemonic	Operation + Add - Subtract ^ AND v OR ⊻ Exclusive OR	a	A	a,x	a,y	al	al,x	(a)	(a,x)	d	d,s	d,x	d,y	(d)	[d]	(d,s),y	(d,x)	(d),y	[d],y	i	r	rl	s	xyc	#	Processor Status Code							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	7 N	6 V	5 M	4 X	3 D	2 1	1 Z	0 C
RTS	Return from Subroutine																						60			.	.	.	.	.	.	.	.
SBC	A - M - $(\overline{C}) \rightarrow$ A	ED		FD	F9		FF			E5	E3	F5		F2	E7	F3	E1	F1	F7						E9	N	.	.	.	.	.	Z	C
SEC	1 → C																			38						.	.	.	.	.	.	.	1
SED	1 → D																			F8						.	.	.	.	1	.	.	.
SEI	1 ? 1																			78						.	.	.	.	.	1	.	.
SEP*	MVP → P																								E2	N	V	M	X	D	1	Z	C
STA	A → M	8D		9D		8F	9F			85	83	95		92	87	93	81	91	97							.	.	.	.	.	.	.	.
STP	STOP (1→ PHI2)																			DB						.	.	.	.	.	.	.	.
STX	X → M	8E								86			96													.	.	.	.	.	.	.	.
STY	Y → M	8C								84		94														.	.	.	.	.	.	.	.
STZ	00 → M	9C		9E						64		74														.	.	.	.	.	.	.	.
TAX	A → X																			AA						N	.	.	.	.	.	Z	.
TAY	A → Y																			AB						N	.	.	.	.	.	Z	.
TCD*	C → D																			5B						N	.	.	.	.	.	Z	.
TCS*	C → S																			1B						.	.	.	.	.	.	.	.
TDC*	D → C																			7B						N	.	.	.	.	.	Z	.
TRB		1C								14																.	.	.	.	.	.	Z	.
TSB	AVM → M	0C								04																.	.	.	.	.	.	Z	.
TSC*	S ? C		3B																							N	.	.	.	.	.	Z	.
TSX	S → X		BA																							N	.	.	.	.	.	Z	.
TXA	X → A		8A																							N	.	.	.	.	.	Z	.

Table 6-5 (continued)

Mnemonic	Operation + Add - Subtract ^ AND v OR ⊻ Exclusive OR	a	A	a,x	a,y	al	al,x	(a)	(a,x)	d	d,s	d,x	d,y	(d)	[d]	(d,s),y	(d,x)	(d),y	[d],y	i	r	rl	s	xyc	#	Processor Status Code 7	6	5	4	3	2	1	0
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N	V	M	X	D	1	Z	C
TXS	X → S		9A																							.	.	.	.	.	.	.	.
TXY*	X → Y		9B																							N	.	.	.	.	.	Z	.
TYA	Y → A		98																							N	.	.	.	.	.	Z	.
TYX*	Y → X		BB																							N	.	.	.	.	.	Z	.
WAI	0 → RDY		CB																							.	.	.	.	.	.	.	.
WDM*	No Operation (Reserved)		42																							.	.	.	.	.	.	.	.
XBA*	B ? A		EB																							N	.	.	.	.	.	Z	.
XCE*	C ? E		FB																							.	.	.	.	.	.	.	E

Notes:

1. Bit immediate N and V flags not affected. When M=0, $M_{15} \rightarrow N$ and $M_{14} \rightarrow V$.
2. Break Bit (B) in Status register indicates hardware or software break.

* = New W65C816 instructions

Table 6-6 Addressing Mode Symbol Table

Symbol	Addressing Mode	Symbol	Addressing Mode
#	immediate	[d]	direct indirect long
A	accumulator	[d],y	direct indirect long indexed
r	program counter relative	a	absolute
rl	program counter relative long	a,x	absolute indexed with x
I	implied	a,y	absolute indexed with y
s	stack	al	absolute long
d	direct	al,x	absolute long indexed
d,x	direct indexed with x	d,s	stack relative
d,y	direct indexed with y	(d,s),y	stack relative indirect indexed
(d)	direct indirect	(a)	absolute indirect
(d,x)	direct indexed indirect	(a,x)	absolute indexed indirect
(d),y	direct indirect indexed	xyc	block move

Table 6-7 Instruction Operation (continued on following 6 pages)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
1a. Absolute a		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, BIT, CMP, CPX, CPY, EOR,		2	1	1	0	1	PBR,PC+1	AAL	1
LDA, LDX LDY ORA, SBC, STA, STX,		3	1	1	0	1	PBR,PC+2	AAH	1
STY, STZ,		4	1	1	1	0	DBR,AA	Data Low	1/0
18 OpCodes, 3 bytes, 4 & 5 cycles	(1)	4a	1	1	1	0	DBR,AA+1	Data High	1/0
1b. Absolute a		1	1	1	1	1	PBR,PC	OpCode	1
JMP (4C)		2	1	1	0	1	PBR,PC+1	New PCL	1
1 OpCode, 3 bytes, 3 cycles		3	1	1	0	1	PBR,PC+2	New PCH	1
		1	1	1	1	1	PBR,New PC	OpCode	1
1c. Absolute a		1	1	1	1	1	PBR,PC	OpCode	1
JSR		2	1	1	0	1	PBR,PC+1	New PCL	1
1 OpCode, 3 bytes, 6 cycles		3	1	1	0	1	PBR,PC+2	New PCH	1
(different order from N6502)		4	1	1	0	0	PBR,PC+2	IO	1
		5	1	1	1	0	0,S	PCH	0
		6	1	1	1	0	0,S-1	PCL	0
		1	1	1	1	1	PBR,NEWPC	Next OpCode	1
1d. Absolute (R-M-W) a		1	1	1	1	1	PBR,PC	OpCode	1
ASL, DEC, INC, LSR, ROL, ROR, TRB,		2	1	1	0	1	PBR,PC+1	AAL	1
TSB		3	1	1	0	1	PBR,PC+2	AAH	1
6 OpCodes, 3 bytes, 6 & 8 cycles		4	1	0	1	0	DBR,AA	Data Low	1
	(1)	4a	1	0	1	0	DBR,AA+1	Data High	1
	(3)(17)	5	1	0	0	0	DBR,AA+1	IO	1
	(1)	6a	1	0	1	0	DBR,AA+1	Data High	0
		6	1	0	1	0	DBR,AA	Data Low	0
2a. Absolute Indexed Indirect (a,x)		1	1	1	1	1	PBR,PC	OpCode	1
JMP		2	1	1	0	1	PBR-PC+1	AAL	1
1 OpCode, 3 bytes, 6 cycles		3	1	1	0	1	PBR-PC+2	AAH	1
		4	1	1	0	0	PBR,PC+2	IO	1
		5	1	1	0	1	PBR,AA+X	New PCL	1
		6	1	1	0	1	PBR,AA+X+1	New PCH	1
		1	1	1	1	1	PBR,NEW PC	OpCode	1
2b. Absolute Indexed Indirect (a,x)		1	1	1	1	1	PBR,PC	OpCode	1
JSR		2	1	1	0	1	PBR,PC+1	AAL	1
1 OpCode, 3 bytes, 8 cycles		3	1	1	1	0	0,S	PCH	0
		4	1	1	1	0	0,S-1	PCL	0
		5	1	1	0	1	PBR,PC+2	AAH	1
		6	1	1	0	0	PBR,PC+2	IO	1
		7	1	1	0	1	PBR,AA+X	New PCL	1
		8	1	1	0	1	PBR,AA+X+1	New PCH	1
		1	1	1	1	1	PBR,NEW PC	Next OpCode	1
3a. Absolute Indirect (a)		1	1	1	1	1	PBR,PC	OpCode	1
JML		2	1	1	0	1	PBR,PC+1	AAL	1
1 OpCode, 3 bytes, 6 cycles		3	1	1	0	1	PBR,PC+2	AAH	1
		4	1	1	1	0	0,AA	New PCL	1
		5	1	1	1	0	0,AA+1	New PCH	1
		6	1	1	1	0	0,AA+2	New PBR	1
		1	1	1	1	1	NEW PBR,PC	OpCode	1
3b. Absolute Indirect (a)		1	1	1	1	1	PBR,PC	OpCode	1
JMP		2	1	1	0	1	PBR,PC+1	AAL	1
1 OpCode, 3 bytes, 5 cycles		3	1	1	0	1	PBR,PC+2	AAH	1
		4	1	1	1	0	0,AA	New PCL	1
		5	1	1	1	0	0,AA+1	New PCH	1
		1	1	1	1	1	PBR,NEW PC	OpCode	1

(See Table 6.8 for abbreviations.)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
4a. Absolute Long al		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	AAL	1
STA, 8 OpCodes, 4 bytes,		3	1	1	0	1	PBR,PC+2	AAH	1
5 & 6 cycles		4	1	1	0	1	PBR,PC+3	AAB	1
		5	1	1	1	0	AAB,AA	Data Low	1/0
	(1)	5a	1	1	1	0	AAB,AA+1	Data High	1/0
4b. Absolute Long (JUMP) al		1	1	1	1	1	PBR,PC	OpCode	1
JMP		2	1	1	0	1	PBR,PC+1	New PCL	1
1 OpCode, 4 bytes, 4 cycles		3	1	1	0	1	PBR,PC+2	New PCH	1
		4	1	1	0	1	PBR,PC+3	New BR	1
		1	1	1	1	1	New PBR,PC	OpCode	1
4c. Absolute Long (JUMP to Subroutine		1	1	1	1	1	PBR,PC	OpCode	1
Long) al		2	1	1	0	1	PBR,PC+1	New PCL	1
JSL		3	1	1	0	1	PBR,PC+2	New PCH	1
1 OpCode, 4 bytes, 7 cycles		4	1	1	1	0	0,S	PBR	0
		5	1	1	0	0	0,S	IO	1
		6	1	1	0	1	PBR,PC+3	New PBR	1
		7	1	1	1	0	0,S-1	PCH	0
		8	1	1	1	0	0,S-2	PCL	0
		1	1	1	1	1	New PBR,PC	Next OpCode	1
5. Absolute Long,X al,x		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	AAL	1
STA		3	1	1	0	1	PBR,PC+2	AAH	1
8 OpCodes, 4 bytes, 5 and 6 cycles		4	1	1	0	1	PBR,PC+3	AAB	1
		5	1	1	1	0	AAB,AA+X	Data Low	1/0
	(1)	5a	1	1	1	0	AAB,AA+X+1	Data High	1/0
6a Absolute, X a, x		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, BIT, CMP, EOR, LDA, LDY,		2	1	1	0	1	PBR,PC+1	AAL	1
ORA, SBC, STA, STA, STZ		3	1	1	0	1	PBR,PC+2	AAH	1
12 OpCodes, 3 bytes, 4,5 and 6 cycles	(4)	3a	1	1	0	0	DBR,AAH,AAL+XL	IO	1
		4	1	1	1	0	DBR,AA+X	Data Low	1/0
	(1)	4a	1	1	1	0	DBR,AA+X+1	Data High	1/0
6b Absolute, X(R-M-W) a,x		1	1	1	1	1	PBR,PC	OpCode	1
ASL, DEC INC LSR ROL, ROR		2	1	1	0	1	PBR,PC+1	AAL	1
6 OpCodes, 3 bytes, 7 and 9 cycles		3	1	1	0	1	PBR,PC+2	AAH	1
		4	1	1	0	0	DBR,AAH,AAL+XL	IO	1
		5	1	0	1	0	DBR,AA+X	Data Low	1
	(1)	5a	1	0	1	0	DBR,AA+X+1	Date High	1
	(3)(17)	6	1	0	0	0	DBR,AA+X+1	IO	1
	(1)	7a	1	0	1	0	DBR,AA+X+1	Data High	0
		7	1	0	1	0	DBR,AA+X	Data Low	0
7. Absolute, Y a,y		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, LDX, ORA,		2	1	1	0	1	PBR,PC+1	AAL	1
SBC, STA		3	1	1	0	1	PBR,PC+2	AAH	1
9 OpCodes, 3 bytes, 4,5 and 6 cycles	(4)	3a	1	1	0	0	DBR,AAH,AAL+YL	IO	1
		4	1	1	1	0	DBR,AA+Y	Data Low	1/0
	(1)	4a	1	1	1	0	DBR,AA+Y+1	Data High	1/0
8. Accumulator A		1	1	1	1	1	PBR,PC	OpCode	1
ASL, DEC, INC, LSR, ROL, ROR		2	1	1	0	0	PBR,PC+1	IO	1
6 OpCodes, 1 byte, 2 cycles									

(see Table 6-8 for abbreviations)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
9a. Block Move Negative (backward) xyc MVN 1 Op Code N-2 3 bytes Byte 7 cycles C=2 x=Source Address y=Destination c=# of bytes to move-1 x,y Increment FFFFFF Source End Dest. End Source Start Dest. Start N Byte 000000 C=0		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X	SRC Data	1
		5	1	1	1	0	DBA,Y	Dest Data	0
		6	1	1	0	0	DBA,Y	IO	1
		7	1	1	0	0	DBA,Y	IO	1
		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X+1	SRC Data	1
		5	1	1	1	0	DBA,Y+1	Dest Data	0
		6	1	1	0	0	DBA,Y+1	IO	1
		7	1	1	0	0	DBA,Y+1	IO	1
		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X+2	SRC Data	1
		5	1	1	1	0	DBA,Y+2	Dest Data	0
		6	1	1	0	0	DBA,Y+2	IO	1
		7	1	1	0	0	DBA,Y+2	IO	1
		1	1	1	1	1	PBR,PC+3	Next OpCode	1
9b. Block Move Positive (forward) xyc (MVP) 1 Op Code N-2 3 bytes Byte 7 cycles C=2 x=Source Address y=Destination c=# of bytes to move-1 x,y Decrement MVP is used when the destination start address is higher (more positive) than the source start address. FFFFFF Destination Start Source Start Destination N Byte Last C=0 000000		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X	SRC Data	1
		5	1	1	1	0	DBA,Y	Dest Data	0
		6	1	1	0	0	DBA,Y	IO	1
		7	1	1	0	0	DBA,Y	IO	1
		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X-1	SRC Data	1
		5	1	1	1	0	DBA,Y-1	Dest Data	0
		6	1	1	0	0	DBA,Y-1	IO	1
		7	1	1	0	0	DBA,Y-1	IO	1
		1	1	1	1	1	PBR,PC	OP Code	1
		2	1	1	0	1	PBR,PC+1	DBA	1
		3	1	1	0	1	PBR,PC+2	SBA	1
		4	1	1	1	0	SBA,X-2	SRC Data	1
		5	1	1	1	0	DBA,Y-2	Dest Data	0
		6	1	1	0	0	DBA,Y-2	IO	1
		7	1	1	0	0	DBA,Y-2	IO	1
		1	1	1	1	1	PBR,PC+3	Next OpCode	1

(see Table 6-7 for abbreviations)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
10a. Direct d		1	1	1	1	1	PBR,PC	OpCode	1
ADC AND BIT, CMP, CPX, CPY ,EOR,		2	1	1	0	1	PBR,PC+1	DO	1
LDA, LDX, LDY, ORA, SBC, STA, STX,	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
STY, STZ		3	1	1	1	0	0,D+DO	Data Low	1/0
16 OpCodes, 2 bytes, 3, 4 & 5 cycles	(1)	3a	1	1	1	0	0,D+DO+1	Data High	1/0
10b. Direct (R-M-W)d		1	1	1	1	1	PBR, PC	OpCode	1
ASL, DEC, INC, LSR, ROL, ROR, TRB,		2	1	1	0	1	PBR,PC+1	DO	1
TSB,	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes, 5,6,7 and 8 cycles		3	1	0	1	0	0,D+DO	Data Low	1
	(1)	3a	1	0	1	0	0,D+DO+1	Data High	1
	(3),(17)	4	1	0	0	0	0,D+DO+1	IO	1
	(1)	5a	1	0	1	0	0,D+DO+1	Data High	0
		5	1	0	1	0	0,D+DO	Data Low	0
11. Direct Indexed Indirect (d,x)		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	DO	1
STA,	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes, 6,7 and 8 cycles		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	1	1	0	0,D+DO+X	AAL	1
		5	1	1	1	0	0,D+DO+X+1	AAH	1
		6	1	1	1	0	DBR,AA	Data Low	1/0
	(1)	6a	1	1	1	0	DBR,AA+1	Data High	1/0
12. Direct Indirect (d)		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC.		2	1	1	0	1	PBR,PC+1	DO	1
STA,	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes 2 bytes, 5,6 and 7 cycles		3	1	1	1	0	0,D+DO	AAL	1
		4	1	1	1	0	0,D+DO+1	AAH	1
		5	1	1	1	0	DBR,AA	Data Low	1/0
	(1)	5a	1	1	1	0	DBR,AA+1	Data High	1/0
13. Direct Indirect Indexed (d),y		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	DO	1
STA	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes, 5,6,7 and 8 cycles		3	1	1	1	0	0,D+DO	AAL	1
		4	1	1	1	0	0,D+DO+1	AAH	1
	(4)	4a	1	1	0	0	DBR,AAH.AAL+YL	IO	1
		5	1	1	1	0	DBR,AA+Y	Data Low	1/0
	(1)	5a	1	1	1	0	DBR,AA+Y+1	Data High	1/0
14. Direct Indirect Indexed Long [d],y		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	DO	1
STA	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes, 6,7 and 8 cycles		3	1	1	1	0	0,D+DO	AAL	1
		4	1	1	1	0	0,D+DO+1	AAH	1
		5	1	1	1	0	0,D+DO+2	AAB	1
		6	1	1	1	0	AAB,AA+Y	Data Low	1/0
	(1)	6a	1	1	1	0	AAB,AA+Y+1	Data High	1/0
15. Direct Indirect Long [d]		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	DO	1
STA	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes, 6,7 and 8 cycles		3	1	1	1	0	0,D+DO	AAL	1
		4	1	1	1	0	0,D+D0+1	AAH	1
		5	1	1	1	0	0,D+DO+2	AAB	1
		6	1	1	1	0	AAB,AA	Data Low	1/0
	(1)	6a	1	1	1	0	AAB,AA+1	Data High	1/0

(see Table 6-8 for abbreviations)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
16a. Direct, X d,x		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, BIT, CMP, EOR, LDA		2	1	1	0	1	PBR,PC+1	DO	1
LDY, ORA, SBC, STA, STY, STZ,	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
12 OpCodes,2 bytes, 4,5,and 6 cycles		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	1	1	0	0,D+DO+X	Data Low	1/0
	(1)	4a	1	1	1	0	0,D+DO+X+1	Data High	1/0
16b. Direct, X (R-M-W) d,x		1	1	1	1	1	PBR,PC	OpCode	1
ASL, DEC, INC, LSR, ROL, ROR,		2	1	1	0	1	PBR,PC+1	DO	1
6 OpCodes, 2 bytes, 6,7,8 and 9 cycles	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	0	1	0	0,D+DO+X	Data Low	1
	(1)	4a	1	0	1	0	0,D+DO+X+1	Data High	1
	(3),(17)	5	1	0	0	0	0,D+DO+X+1	IO	1
	(1)	6a	1	0	1	0	0,D+DO+X+1	Data High	0
		6	1	0	1	0	0,D+DO+X	Data Low	0
17. Direct, Y d,y		1	1	1	1	1	PBR,PC	OpCode	1
LDX, STX		2	1	1	0	1	PBR,PC+1	DO	1
2 OpCodes, 2 bytes, 4,5 and 6 cycles	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	1	1	0	0,D+DO+Y	Data Low	1/0
	(1)	4a	1	1	1	0	0,D+DO+Y+1	Data High	1/0
18.Immediate #		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, BIT, CMP, CPX, CPY,		2	1	1	0	1	PBR,PC+1	IDL	1
EOR, LDA, LDX, LDY, ORA, REP, SEC, SEP	(1)(6)	2a	1	1	0	1	PBR,PC+2	IDH	1
14 OpCodes, 2 and 3 bytes, 2 and 3 cycles									
19a. Implied i		1	1	1	1	1	PBR,PC	OpCode	1
CLC, CLD, CLI, CLV, DEX, DEY, INX, INY, NOP, SEC, SED, SEI, TAX, TAY, TCD, TCS, TDC, TSC, TSX, TXA, TXS, TXY, TYA, TYX, XCE		2	1	1	0	0	PBR,PC+1	IO	1
25 OpCodes, 1 byte, 2 cycles									
19b. Implied i		1	1	1	1	1	PBR,PC	OpCode	1
XBA		2	1	1	0	0	PBR,PC+1	IO	1
1 OpCode, 1 byte, 3 cycles		3	1	1	0	0	PBR,PC+1	IO	1
19c. Stop the Clock		1	1	1	1	1	PBR,PC	OpCode	1
STP		2	1	1	0	0	PBR,PC+1	IO	1
1 OpCode		3	1	1	0	0	PBR,PC+1	IO	1
1 byte RESB=1		1c	1	1	0	0	PBR,PC+1	RES (BRK)	1
3 cycles RESB=0		1b	1	1	0	0	PBR,PC+1	RES (BRK)	1
RESB=0		1a	1	1	0	0	PBR,PC+1	RES (BRK)	1
RESB=1		1	1	1	1	1	PBR,PC+1	BEGIN	1
(See 22a. Stack Hardware Interrupt)									
19d. Wait for Interrupt	RDY=1	1	1	1	1	1	PBR,PC	OpCode	1
WAI	(9)RDY=1	2	1	1	0	0	PBR,PC+1	IO	1
1 OpCode,1 byte	RDY=0	3	1	1	0	0	PBR,PC+1	IO	1
3 cycles IRQB, NMIB	RDY=1	1	1	1	1	1	PBR,PC+1	IRQ(BRK)	1

(see Table 6-8 for abbreviations)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
20. Relative r		1	1	1	1	1	PBR,PC	OpCode	1
BCC, BCS, BEQ, BMI, BNE, BPL, BRA,		2	1	1	0	1	PBR,PC+1	Offset	1
BVC,BVS	(5)	2a	1	1	0	0	PBR,PC+1	IO	1
9 OpCodes, 2 bytes, 2,3 and 4 cycles	(6)	2b	1	1	0	0	PBR,PC+1	IO	1
		1	1	1	1	1	PBR,PC+Offset	OpCode	1
21. Relative Long rl		1	1	1	1	1	PBR,PC	OpCode	1
BRL		2	1	1	0	1	PBR,PC+1	Offset Low	1
1 OpCode, 3 bytes, 4 cycles		3	1	1	0	1	PBR,PC+2	Offset High	1
		4	1	1	0	0	PBR,PC+2	IO	1
		1	1	1	1	1	PBR,PC+Offset	OpCode	1
22a. Stack s	(3)	1	1	1	1	1	PBR,PC	IO	1
ABORT, IRQ, NMI, RES	(7)	2	1	1	0	0	PBR,PC	IO	1
4 hardware interrupts	(10)	3	1	1	1	0	0,S	PBR	0
0 bytes, 7 and 8 cycles	(10)	4	1	1	1	0	0,S-1	PCH	0
	(10)	5	1	1	1	0	0,S-2	PCL	0
	(11)	6	1	1	1	0	0.S-3	P	0
		7	0	1	1	0	0,VA	AAVL	1
		8	0	1	1	0	0,VA+1	AAVH	1
		1	1	1	1	1	0,AAV	Next OpCode	1
22b. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
PLA, PLB, PLD, PLP, PLX, PLY		2	1	1	0	0	PBR,PC+1	IO	1
Different than N6502		3	1	1	0	0	PBR,PC+1	IO	1
6 Op Codes,1 byte, 4 and 5 cycles		4	1	1	1	0	0,S+1	REG Low	1
	(1)	4a	1	1	1	0	0,S+2	REG High	1
22c. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
PHA, PHB PHP, PHD, PHK, PHX, PHY	(1)	2	1	1	0	0	PBR,PC+1	IO	1
7 Op Codes, 1 byte, 3 and 4 cycles	(12)	3a	1	1	1	0	0,S	REG High	1
		3	1	1	1	0	0,S-1	REG Low	1
22d. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
PEA		2	1	1	0	1	PBR,PC+1	AAL	1
1 Op Code, 3 bytes, 5 cycles		3	1	1	0	1	PBR,PC+2	AAH	1
		4	1	1	1	0	0,S	AAH	0
		5	1	1	1	0	0,S-1	AAL	0
22e. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
PEI		2	1	1	0	1	PBR,PC+1	DO	1
1 Op Code, 2 bytes, 6 and 7 cycles	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
		3	1	1	1	0	0,D+DO	AAL	1
		4	1	1	1	0	0,D+DO+1	AAH	1
		5	1	1	1	0	0,S	AAH	0
		6	1	1	1	0	0,S-1	AAL	0
22f. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
PER		2	1	1	0	1	PBR,PC+1	Offset Low	1
1 Op Code, 3 bytes, 6 cycles		3	1	1	0	1	PBR,PC+2	Offset High	1
		4	1	1	0	0	PBR,PC+2	IO	1
		5	1	1	1	0	0,S	PCH+Offset+Carry	0
		6	1	1	1	0	0,S-1	PCL+Offset	0

(see Table 6-8 for abbreviations)

Table 6-7 (continued)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
22g. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
RTI		2	1	1	0	0	PBR,PC+1	IO	1
1 Op Code, 1 byte, 6 and 7 cycles	(3)	3	1	1	0	0	PBR,PC+1	IO	1
(different order fromN6502)		4	1	1	1	0	0,S+1	P	1
		5	1	1	1	0	0,S+2	New PCL	1
		6	1	1	1	0	0,S+3	New PCH	1
	(7)	7	1	1	1	0	0,S+4	PBR	1
		1	1	1	1	1	PBR,New PC	Next OpCode	1
22h. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
RTS		2	1	1	0	0	PBR,PC+1	IO	1
1 OpCode, 1 byte, 6 cycles		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	1	1	0	0,S+1	PCL	1
		5	1	1	1	0	0,S+2	PCH	1
		6	1	1	0	0	0,S+2	IO	1
		1	1	1	1	1	PBR,PC	OpCode	1
22i. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
RTL		2	1	1	0	0	PBR,PC+1	IO	1
1 Op Code, 1 byte, 6 cycles		3	1	1	0	0	PBR,PC+1	IO	1
		4	1	1	1	0	0,S+1	New PCL	1
		5	1	1	1	0	0,S+2	New PCH	1
		6	1	1	1	0	0,S+3	New PBR	1
		1	1	1	1	1	NEW PBR,PC	Next OpCode	1
22j. Stack s		1	1	1	1	1	PBR,PC	OpCode	1
BRK,COP	(3)	2	1	1	0	1	PBR,PC+1	Signature	1
2 OpCodes, 2 bytes	(7)	3	1	1	1	0	0,S	PBR	0
7 and 8 cycles	(10)	4	1	1	1	0	0,S-1	PCH	0
	(10)	5	1	1	1	0	0,S-2	PCL	0
	(10)	6	1	1	1	0	0,S-3 (16)	P	0
		7	0	1	1	0	0,VA	AAVL	1
		8	0	1	1	0	0,VA+1	AAVH	1
		1	1	1	1	1	0,AAV	Next OpCode	1
23. Stack Relative d,s		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR,		2	1	1	0	1	PBR,PC+1	SO	1
LDA, ORA, SBC, STA		3	1	1	0	0	PBR,PC+1	IO	1
8 Op Codes, 2 bytes, 4 and 5 cycles		4	1	1	1	0	0,S+SO	Data Low	1/0
	(1)	4a	1	1	1	0	0,S+SO+1	Data High	1/0
24. Stack Relative Indirect Indexed (d,s),y		1	1	1	1	1	PBR,PC	OpCode	1
ADC, AND, CMP, EOR, LDA, ORA, SBC,		2	1	1	0	1	PBR,PC+1	SO	1
STA		3	1	1	0	0	PBR,PC+1	IO	1
8 Op Codes, 2 bytes, 7 and 8 cycles		4	1	1	1	0	0,S+SO	AAL	1
		5	1	1	1	0	0,S+SO+1	AAH	1
		6	1	1	0	0	0,S+SO+1	IO	1
		7	1	1	1	0	DBR,AA+Y	Data Low	1/0
	(1)	7a	1	1	1	0	DBR,AA+Y+1	Data High	1/0

(see Table 6-8 for abbreviations)

Notes: Be aware that notes #4-7, 9 and 10 apply to the W65C02S and W65C816S. All other notes apply to the W65C816S only.

1. Add 1 byte (for immediate only) for M=0 or X=0 (i.e. 16-bit data), add 1 cycle for M=0 or X=0. REP, SEP are always 3 cycle instructions and VPA is low during the third cycle. The address bus is PC+1 during the third cycle.
2. Add 1 cycle for direct register low (DL) not equal 0.
3. Special case for aborting instruction. This is the last cycle which may be aborted or the Status, PBR or DBRregisters will be updated.
4. Add 1 cycle for indexing across page boundaries, or write, or X=0. When X=1 or in the emulation mode, this cycle contains invalid addresses.
5. Add 1 cycle if branch is taken.
6. Add 1 cycle if branch is taken across page boundaries in 6502 emulation mode (E=1).
7. Subtract 1 cycle for 6502 emulation mode (E=1).
8. Add 1 cycle for REP, SEP.
9. Wait at cycle 2 for 2 cycles after NMIB or IRQB active input.
10. RWB remains high during Reset.
11. BRK bit 4 equals "0" in Emulation mode.
12. PHP and PLP.
13. Some OpCodes shown are compatible only with the W65C816S.
14. VDA and VPA are not valid outputs on the W65C02S but are valid on the W65C816S. The two signals, VDA and VPA, are included to point out the upward compatibility to the W65C816S. When VDA and VPA are both a one level, this is equivalent to SYNC being a one level.
15. The PBR is only applicable to the W65C816S.
16. COP Latches.
17. In the emulation mode, during a R-M-W instruction the RWB is low during both write and modify cycles.

Table 6-8 **Abbreviations**

Abbreviation	Explanation
AAB	Absolute Address Bank
AAH	Absolute Address High
AAL	Absolute Address Low
AAVH	Absolute Address Vector High
AAVL	Absolute Address Vector Low
C	Accumulator
D	Direct Register
DBA	Destination Bank Address
DBR	Data Bank Address
DEST	Destination
DO	Direct Offset
IDH	Immediate Data High
IDL	Immediate Data Low
IO	Internal Operation
OFF	Offset
P	Status Register
PBR	Program Bank Register
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
R-M-W	Read-Modify-Write
REG	Register
S	Stack Address
SBA	Source Bank Address
SRC	Source
SO	Stack Offset
VA	Vector Address
x,y	Index Register

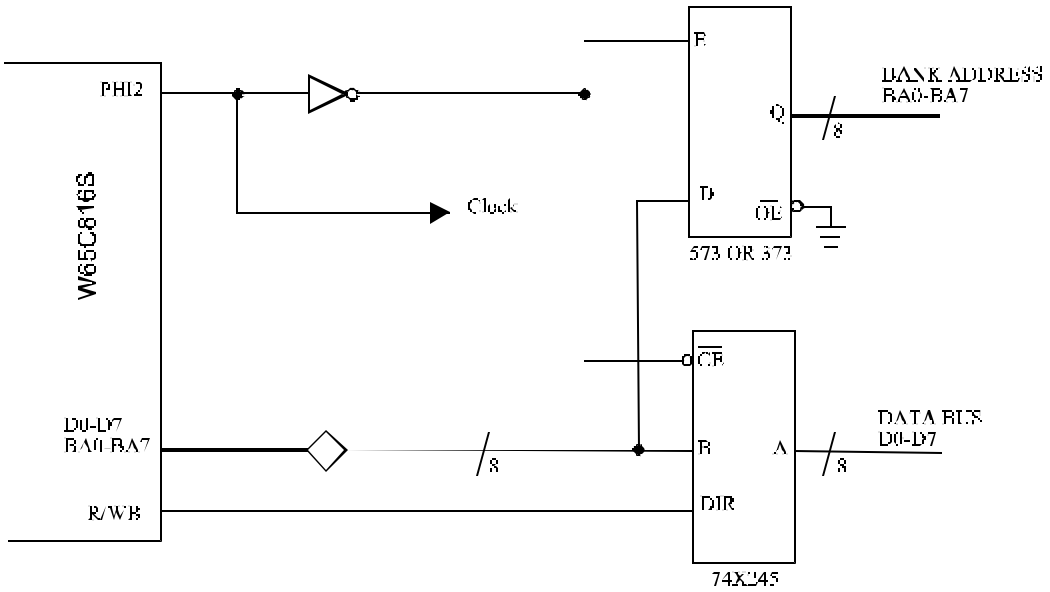


Figure 6-1 Bank Address Latching Circuit

7 RECOMMENDED W65C816S ASSEMBLER SYNTAX STANDARDS

7.1 Directives

Assembler directives are those parts of the assembly language source program which give directions to the assembler; this includes the definition of data area and constants within a program. This standard excludes any definitions of assembler directives.

7.2 Comments

An assembler should provide a way to use any line of the source program as a comment. The recommended way of doing this is to treat any blank line, or any line that starts with a semi-colon or an asterisk as a comment. Other special characters may be used as well.

7.3 The Source Line

Any line which causes the generation of a single W65C816S machine language instruction should be divided into four fields: a label field, the operation code, the operand, the comment field.

7.3.1 The Label Field

The label field begins in column one of the line. A label must start with an alphabetic character, and may be followed by zero or more alphanumeric characters. An assembler may define an upper limit on the number of characters that can be in a label, so long as that upper limit is greater than or equal to six characters. An assembler may limit the alphabetic characters to upper-case characters if desired. If lower-case characters are allowed, they should be treated as identical to their upper-case equivalents. Other characters may be allowed in the label, so long as their use does not conflict with the coding of operand fields.

7.3.2 The Operation Code Field

The operation code shall consist of a three character sequence (mnemonic) from Table 7-1. It shall start no sooner than column 2 of the line, or one space after the label if a label is coded.

7.3.2.1 Many of the operation codes in Table 6-1 have duplicate mnemonics; when two or more machine language instruction has the same mnemonic, the assembler resolves the difference based on the operand.

7.3.2.2 If an assembler allows lower-case letters in labels, it must also allow lower-case letters in the mnemonic. When lower-case letters are used in the mnemonic, they shall be treated as equivalent to the upper-case counterpart. Thus, the mnemonics LDA, lda and LdA must all be recognized, and are equivalent.

7.3.2.3 In addition to the mnemonics shown in Table 7-1, an assembler may provide the alternate mnemonics shown in Table 7-1.

Table 7-1 Alternate Mnemonics

WDC Standard	Alias
BCC	BLT
BCS	BGE
CMP A	CMA
DEC A	DEA
INC A	INA
JSL	JSR
JML	JMP
TCD	TAD
TCS	TAS
TDC	TDA
TSC	TSA
XBA	SWA

7.3.2.4 JSL should be recognized as equivalent to JSR when it is specified with a long absolute address forced. JML is equivalent to JMP with long addressing forced.

7.3.3 The Operand Field

The operand field may start no sooner than one space after the operation code field. The assembler must be capable of at least twenty-four bit address calculations. The assembler should be capable of specifying addresses as labels, integer constants, and hexadecimal constants. The assembler must allow addition and subtraction in the operand field. Labels shall be recognized by the fact they start with alphabetic characters. Decimal numbers shall be recognized as containing only the decimal digits 0...9. Hexadecimal constants shall be recognized by prefixing the constant with a "$" character, followed by zero or more of either the decimal digits or the hexadecimal digits "A"..."F". If lower-case letters are allowed in the label field, then they shall also be allowed as hexadecimal digits.

7.3.3.1 All constants, no matter what their format, shall provide at least enough precision to specify all values that can be represented by a twenty-four bit signed or unsigned integer represented in two's complement notation.

7.3.3.2 Table 7-2 shows the operand formats that shall be recognized by the assembler. bol **d** is a label or value which the assembler can recognize as being less than $100. The symbol **a** is a label or value which the assembler can recognize as greater than $FF but less than $10000; the symbol **al** is a label or value that the assembler can recognize as being greater than $FFF. The symbol EXT is a label which cannot be located by the assembler at the time the instruction is assembled. Unless instructed otherwise, an assembler shall assume that EXT labels are two bytes long. The symbols **r** and **rl** are 8 and 16 bit signed displacements calculated by the assembler.

Table 7-2 Address Mode Formats

Addressing Mode	Format
Immediate	#d
	#a
	#al
	#EXT
	#<d
	#<a
	#<al
	#<EXT
	#>d
	#>a
	#>al
	#>EXT
	#^d
	#^a
	#^al
	#^EXT
Absolute	!d
	!a
	a
	!al
	!EXT
	EXT
Absolute Long	>d
	>a
	>al
	al
	>EXT
Direct Page	d
	<d
	<a
	<al
	<EXT
Accumulator	A
Implied Addressing	(no operand)
Direct Indirect Indexed	(d),y
	(<d,y)
	(<a),y
	(<al),y
	(<EXT),y
Direct Indirect Indexed Long	[d],y
	[<d],y
	[<a],y
	[<al],y
	[<EXT],y
Direct Indexed Indirect	(d,x)
	(<d,x)
	(<a,x)
	(<al,x)
	(<EXT,x)
Direct Indexed by X	d,x
	<d,x
	<a,x
	<al,x
	<EXT,x
Direct Indexed by Y	d,y
	<d,y
	<a,y
	<al,y
	<EXT,y
Absolute Indexed by X	d,x
	!d,x
	a,x
	!a,x
	!al,x
	!EXT,x
	EXT,x
Absolute Indexed by Y	!d,y
	d,y
	a,y
	!a,y
	!al,y
	!EXT,y
	EXT,y
Absolute Long Indexed by X	>d,x
	>a,x
	>al,x
	al,x
	>EXT,x
Program Counter Relative and Program Counter Relative Long	d
	a
	al
	(EXT)
Absolute Indirect	(d)
	(!d)
	(a)
	(!a)
	(!al)
	EXT
Direct Indirect	(d)
	(<a)
	(<al)
	(<EXT)
Direct Indirect Long	[d]
	[>a]
	[>al]
	[>EXT]
Absolute Indexed	(d,x)
	(!d,x)
	(a,x)
	(!a,x)
	(!al,x)
	(EXT,x)
	(!EXT,x)
Stack Addressing	(no operand)
Stack Relative Indirect Indexed	(d,s),y
	(<d,s),y
	(<a,s),y
	(<al,s),y
	(<EXT,s),y
Block Move	d,d
	d,a
	d,al
	d,EXT
	a,d
	a,a
	a,al
	a,EXT
	al,d
	al,a
	al,al
	al,EXT
	EXT,d
	EXT,a
	EXT,al
	EXT,EXT

Note: The alternate ! (exclamation point) is used in place of the | (vertical bar).

7.3.3.3 Note that the operand does not determine whether or not immediate address loads one or two bytes, this is determined by the setting of the status register. This forces the requirement for a directive or directives that tell the assembler to generate one or two bytes of space for immediate loads. The directives provided shall allow separate settings for the accumulator and index registers.

7.3.3.4 The assembler shall use the <, >, and ^ characters after the # character in immediate address to specify which byte or bytes will be selected from the value of the operand. Any calculations in the operand must be performed before the byte selection takes place. Table 7-3 defines the action taken by each operand by showing the effect of the operator on an address. The column that shows a two byte immediate value show the bytes in the order in which they appear in memory. The coding of the operand is for an assembler which uses 32-bit address calculations, showing the way that the address should be reduced to a 24-bit value.

Table 7-3 Byte Selection Operator

Operand	One Byte Result	Two Byte Result	
#$01020304	04	04	03
#<$01020304	04	04	03
#>$01020304	03	03	02
#^$01020304	02	02	01

7.3.3.5 In any location in an operand where an address, or expression resulting in an address, can be coded, the assembler shall recognize the prefix characters <, |, and >, which force one byte (direct page), two byte (absolute) or three byte (long absolute) addressing. In cases where the addressing modes is not forced, the assembler shall assume that the address is two bytes unless the assembler is able to determine the type of addressing required by context, in which case that addressing mode will be used. Addresses shall be truncated without error in an addressing mode is forced which does not require the entire value of the address. For example, LDA $0203 and LDA |$010203 are completely equivalent. If the addressing mode is not forced, and the type of addressing cannot be determined from context, the assembler shall assume that a two byte address is to be used. If an instruction does not have a short addressing mode (as in LDA< which has no direct page indexed by Y) and a short address is used in the operand, the assembler shall automatically extend the address by padding the most significant bytes with zeroes in order to extend the address to the length needed. As with immediate address, any expression evaluation shall take place before the address is selected; thus, the address selection character is only used once, before the address of expression.

7.3.3.6 The (!) exclamation point character should be supported as an alternative to the | (vertical bar).

7.3.3.7 A long indirect address is indicated in the operand field of an instruction field of an instruction by surrounding the direct page address where the indirect address is found by square brackets; direct page addresses which contain sixteen-bit addresses are indicated by being surrounded by parentheses.

7.3.4 Comment Field

The comment field may start no sooner than one space after the operation code field or operand field depending on instruction type.

8 Caveats

Table 8-1 Caveats

Compatibility Issue	NMOS 6502	W65C02	W65C02S	W65C816S
S (Stack)	Always Page 1, 8 bits	Always Page 1, 8 bits	Always page 1, 8 bits	Always page 1 8 bits when(E=1), 16 bits when E=0
X (X Index Reg)	Always Page 0 Always less than 256 ie 8 Bits	Always Page 0 Always less than 256 ie 8 Bits	Always Page 0 Always less than 256 ie 8 Bits	Indexed page zero always in page 0 (E=1), Cross page (E=0)
Y (Y Index Reg)	Always Page 0 Always less than 256 ie 8 Bits	Always Page 0 Always less than 256 ie 8 Bits	Always Page 0 Always less than 256 ie 8 Bits	Indexed page zero always in page 0 (E=1), Cross page (E=0)
A (Accumulator)	8 bits	8 bits	8 bits	8 bits (M=1), 16 bits (M=0)
(Flag Reg)	N, V and Z flags invalid in decimal mode. D=unknown after reset. D not modified after interrupt	N,V and Z flags valid in decimal mode. D=0 after reset/interrupt	N,V and Z flags valid in decimal mode. D=0 after reset/interrupt	N,V and Z flags valid in decimal mode. D=0 after reset/interrupt
Timing A.ABS,X,ASL,LSR, ROL with no Page Crossing B. Jump Indirect Operand =XXFF C. Branch Across Page D. Decimal Mode	7 cycles 5 cycles and invalid page crossing 4 cycles No add. cycles	6 cycles 6 cycles 4 cycles Add 1 cycle	6 cycles 6 cycles 4 cycles Add 1 Cycle	7 cycles 5 cycles 4 cycles No add. cycles
BRK Vector	FFFE,F BRK bit=0 on stack if IRQ, NMI	FFFE,F BRK bit=0 on stack if IRQ, NMI	FFFE,F BRK bit=0 on stack if IRQ, NMI	00FFFE,F(E=1) BRK bit=0 on stack if IRQ- NMIB, ABORT B 000FFE6,7 (E=0), X=X on stack always
Interrupt or Break Bank Address	Not available	Not available	Not available	PBR not pushed (E=1) RTI, PBR, not pulled (E-1) PRB pushed (E=0) RTI, PBR pulled (E=0)
Memory Lock (ML)	Not available	MLB=0 during Modify and Write cycles	MB=0 during Modify and Write cycles	MLB=0 during Read Modify and Write cycles
Indexed Across Page Boundary (d),y a,x a,y	Extra read of invalid address	Extra read of last instruction fetch	Extra read of last instruction fetch	Extra read of invalid address
RDY Pulled during Write Cycle	Ignored	Processor stops	Processor stops	Processor Stops

8.1 Stack Addressing

When in the Native mode, the Stack may use memory locations 000000 to 00FFFF. The effective address of Stack, Stack Relative, and Stack Relative Indirect Indexed addressing modes will always be within this range. In the Emulation mode, the Stack address range is 000100 to 0001FF. The following OpCodes and addressing modes will increment or decrement beyond this range when accessing two or three bytes: JSL, JSR (a,x), PEA, PEI, PER, PHD, PLD, RTL

8.2 Direct Addressing

8.2.1 The Direct Addressing modes are often used to access memory registers and pointers. The effective address generated by Direct; Direct,X and Direct,Y addressing modes will always be in the Native mode range 000000 to 00FFFF. When in the Emulation mode, the direct addressing range is 000000 to 0000FF, except for [Direct] and [Direct],Y addressing modes and the PEI instruction which will increment from 0000FE or 0000FF into the Stack area.

8.2.2 When in the Emulation mode and DH is not equal to zero, the direct addressing range is 00DH00 to 00DHFF, except for [Direct] and [Direct],Y addressing modes and the PEI instruction which will increment from 00DHFE or 00DHFF into the next higher page.

8.2.3 When in the Emulation mode and DL in not equal to zero, the direct addressing range is 000000 to 00FFFF.

8.3 Absolute Indexed Addressing

The Absolute Indexed addressing modes are used to address data outside the direct addressing range. The W65C02S addressing range is 0000 to FFFF. Indexing from page FFXX may result in a 00YY data fetch when using the W65C02S. In contrast, indexing from page ZZFFXX may result in ZZ+1,00YY when using the W65C816S.

8.4 ABORTB Input

8.4.1 ABORTB should be held low for a period not to exceed one cycle. Also, if ABORTB is held low during the Abort Interrupt sequence, the Abort Interrupt will be aborted. It is not recommended to abort the Abort Interrupt. The ABORTB internal latch is cleared during the second cycle of the Abort Interrupt. Asserting the ABORTB input after the following instruction cycles will cause registers to be modified:

8.4.1.1 Read-Modify-Write: Processor status modified if ABORTB is asserted after a modify cycle.

8.4.1.2 RTI: Processor status modified if ABORTB is asserted after cycle 3.

8.4.1.3 IRQB, NMIB, ABORTB BRK, COP: When ABORTB is asserted after cycle 2, PBR and DBR will become 00 (Emulation mode) or PBR will become 00 (Native mode).

8.4.2 The ABORT Interrupt has been designed for virtual memory systems. For this reason, asynchronous ABORTB's may cause undesirable results due to the above conditions

8.5 VDA and VPA Valid Memory Address Output Signals

When VDA or VPA are high and during all write cycles, the Address Bus is always valid. VDA and VPA should be used to qualify all memory cycles. Note that when VDA and VPA are both low, invalid addresses may be generated. The Page and Bank addresses could also be invalid. This will be due to low byte addition only. The cycle when only low byte addition occurs is an optional cycle for instructions which read memory when the Index Register consists of 8 bits. This optional cycle becomes a standard cycle for the Store instruction, all instructions using the 16-bit Index Register mode, and the Read-Modify-Write instruction when using 8- or 16-bit Index Register modes.

8.6 DB/BA operation when RDY is Pulled Low

When RDY is low, the Data Bus is held in the data transfer state (i.e. PHI2 high). The Bank address external transparent latch should be latched on the rising edge of the PHI2 clock.

8.7 MX Output

The MX output reflects the value of the M and X bits of the processor Status Register. The REP, SEP and PLP instructions may change the state of the M and X bits. Note that the MX output is invalid during the instruction cycle following REP, SEP and PLP instruction execution. This cycle is used as the OpCode fetch cycle of the next instruction.

8.8 All OpCodes Function in All Modes of Operation

8.7.1 It should be noted that all OpCodes function in all modes of operation. However, some instructions and addressing modes are intended for W65C816S 24-bit addressing, and are therefore less useful for the emulation mode. The JSL, RTL, JMP al and JML instructions and addressing modes are primarily intended for W65C816S native mode use.

8.7.2 The following instructions may be used with the emulation mode even though a Bank Address is not multiplexed on the Data Bus: PHK, PHB and PLB

8.7.3 The following instructions have "limited" use in the Emulation mode:

8.7.3.1 The REP and SEP instructions cannot modify the M and X bits when in the Emulation mode. In this mode the M and X bits will always be high (logic 1).

8.7.3.2 When in the Emulation mode, the MVP and MVN instructions use the X and Y Index Registers for the memory address. Also, the MVP and MVN instructions can only move data within the memory range 0000 (Source Bank) to 00FF (Destination Bank) for the W65C816S, and 0000 to 00FF for the emulation mode.

8.9 Indirect Jumps

The JMP (a) and JML (a) instructions use the direct Bank for indirect addressing, while JMP (a,x) and JSR (a,x) use the Program Bank for indirect address tables.

8.10 Switching Modes

When switching from the Native mode to the Emulation mode, the X and M bits of the Status Register are set high (logic 1), the high byte of the Stack is set to 01, and the high bytes of the X and Y Index Registers are set to 00. To save previous values, these bytes must always be stored before changing modes. Note that the low byte of the S, X and Y Registers and the low and high byte of the Accumulator (A and B) are not affected by a mode change.

8.11 How Interrupts Affect the Program Bank and the Data Bank Registers

8.11.1 When in the Native mode, the Program Bank register (PBR) is cleared to 00 when a hardware interrupt, BRK or COP is executed. In the Native mode, previous PBR contents are automatically saved on Stack.

8.11.2 In the Emulation mode, the PBR and DBR registers are cleared to 00 when a hardware interrupt, BRK or COP is executed. In this case, previous contents of the PBR are not automatically saved.

8.11.3 Note that a Return from Interrupt (RTI) should always be executed from the same "mode" which originally generated the interrupt.

8.12 Binary Mode

The Binary Mode is set whenever a hardware or software interrupt is executed. The D flag within the Status Register is cleared to zero.

8.13 WAI Instruction

The WAI instruction pulls RDY low and places the processor in the WAI "low power" mode. NMIB, IRQB or RESB will terminate the WAI condition and transfer control to the interrupt handler routine. Note that an ABORTB input will abort the WAI instruction, but will not restart the processor. When the Status Register I flag is set (IRQB disabled) the IRQB interrupt will cause the next instruction (following the WAI instruction) to be executed without going to the IRQB interrupt handler. This method results in the highest speed response to an IRQB input. When an interrupt is received after an ABORTB which occurs during the WAI instruction, the processor will return to the WAI instruction. Other than RESB (highest priority), ABORTB is the next highest priority, followed by NMIB or IRQB interrupts.

8.14 The STP Instruction

The STP instruction disables the PHI2 clock to all internal circuitry. When disabled, the PHI2 clock is held in the high state. In this case, the Data Bus will remain in the data transfer state and the Bank address will not be multiplexed onto the Data Bus. Upon executing the STP instruction, the RESB signal is the only input which can restart the processor. The processor is restarted by enabling the PHI2 clock, which occurs on the falling edge of the RESB input. Note that the external oscillator must be stable and operating properly before RESB goes high.

8.15 COP Signatures

Signatures 00-7F may be user defined, while signatures 80-FF are reserved for instructions on future microprocessors. Contact WDC for software emulation of future microprocessor hardware functions.

8.16 WDM OpCode Use

The WDM OpCode may be used on future microprocessors. It performs no operation. WDM are the initials of William D. Mensch, Jr., the founder of WDC.

8.17 RDY Pulled During Write

The NMOS 6502 does not stop during a write operation. In contrast, both the W65C02S and the W65C816S do stop during write operations

8.18 MVN and MVP Affects on the Data Bank Register

The MVN and MVP instructions change the Data Bank Register to the value of the second byte of the instruction (destination bank address).

8.19 Interrupt Priorities

The following interrupt priorities will be in effect should more than one interrupt occur at the same time:

Highest Priority	Lowest Priority
RESB	ABORTB, NMIB, IRQB

8.20 Transfers from 8-Bit to 16-Bit, or 16-Bit to 8-Bit Registers

All transfers from one register to another will result in a full 16-bit output from the source register. The destination register size will determine the number of bits actually stored in the destination register and the values stored in the processor Status Register. The following are always 16-bit transfers, regardless of the accumulator size: TCS, TSC, TCD and TDC
Note: PHP and PLP are always 8 bit operations.

8.21 Stack Transfers

When in the Emulation mode, a 01 is forced into SH. In this case, the B Accumulator will not be loaded into SH during a TCS instruction. When in the Native mode, the B Accumulator is transferred to SH. Note that in both the Emulation and Native modes, the full 16 bits of the Stack Register are transferred to the A, B and C Accumulators, regardless of the state of the M bit in the Status Register.

8.22 BRK Instruction

The BRK instruction for the NMOS 6502, 65C02 and 65C816 is actually a 2 byte instruction. The NMOS device simply skips the second byte (i.e. doesn't care about the second byte) by incrementing the program counter twice. The 65C02 and 65C816 does the same thing except the assembler is looking for the second byte as a "signature byte". With either device (NMOS or CMOS), the second byte is not used. It is important to realize that if a return from interrupt is used it will return to the location after the second or signature byte.

8.23 Accumulator switching from 8 bit to 16 bit

Care must be taken when switching from 16 bit mode to 8 bit mode then to 16 bit mode. The B register is restored so that the following code shows a potential problem:

```
LONGA     ON
REP       #$20
LDA       #$2345
STA       MIKE
LONGA     PFF
SEP       #$20
LDA       #$01
STA       SAM
LONGA     ON
REP       #$20
STA       BOB
```

Here BOB = $2301 and NOT $000V

9 W65C816DB Developer Board and W65C816ICE In-Circuit Emulator (ICE) Board

The W65C816DB is used for W65C816 core microprocessor System-Chip Development, W65C816S (chip) System Development, or Embedded W65C816DB (board) Development

9.1 Features:

W65C816S 16-bit MPU, total access to all control lines, Memory Bus, Programmable I/O Bus, PC Interface, 20 I/O lines, easy oscillator change, 32K SRAM, 32K EPROM, W65C22S Versatile Interface Adapter VIA peripheral chip, on-board matrix, PLD for Memory map decoding and ASIC design.

The PLD chip is a XILINX XC95108 for changing the chip select and I/O functions if required. To change the PLD chip to suit your own setup, you need XILINX Data Manager for the XC95108 CPLD chip. The W65C816DB includes an on-board programming header for JTAG configuration. For more details refer to the circuit diagram. The on-board W65C816S and the W65C22S devices have measurement points for core power consumption. Power input is provided by an optional power board which plugs into the 10 pin power header.

An EPROM programmer or an EPROM emulator is required to use the board. WDC's Software Development System includes a W65C816S Assembler and Linker, W65C816S C-Compiler and Optimizer, and W65C816S Simulator/Debugger. WDC's PC IO daughter board can be used to connect the Developer Board to the parallel port of a PC.

9.2 Memory map:

CS1B: 8000-FFFF	⇒	EPROM (27C256)
CS3B: 0000-00EF & 0100-7FFF	⇒	SRAM (62C256)
CS2B: 00F0-00FF	⇒	VIA (W65C22S)

10 HARD CORE MODEL

10.1 W65C816 Core Information

- The W65C816S core uses the same instruction set as the W65C816S
- The only functional difference between the W65C816S and W65C816S core is the RDY pin. The W65C816S RDY pin is bi-directional utilizing an active pull-up. The W65C816S core RDY function is split into 2 pins, RDYIN and WAITN. The WAITN output goes low when a WAI instruction is executed.
- The W65C816S core will be a smaller die since the I/O buffers have been removed.
- The outputs are the N-channel and P-channel output transistors drivers.
- The following inputs, if not used, must be held in the high state: RDY input, IRQB, MIB, BE and ABORTB.
- The timing of the W65C816S core is the same as the W65C816S.

11 SOFT CORE RTL MODEL

11.1 W65C816 Synthesizable RTL-Code in Verilog HDL

The RTL-Code (**R**egister **T**ransfer **L**evel) in Verilog is a synthesizable model. The behavior of this model is equivalent to the original W65C816S hard core. The W65C816S RTL-Code is available as the core model and the W65C816S standard chip model. The standard chip model includes the soft core and the buffer ring in RTL-Code. Synthesizable cores are useful in ASIC design.

12 ORDERING INFORMATION

W65C816S6PL-14	
Description W65C = standard product	W65C
Product Identification Number	816S
Foundry Process Blank = 1.2u 8 = .8u 6 = .6u	6
Package P = Plastic Dual-In-Line, 40 pins PL = Plastic Leaded Chip Carrier, 44 pins Q = Quad Flat Pack, 44 pins	PL
Temperature/Processing DIP = 0°C to + 70°C PLCC and QFP = -40°C to + 85°C	
Speed Designator -14 = 14MHz	-14

To receive general sales or technical support on standard product or information about our module library licenses, contact us at:

The Western Design Center, Inc.
2166 East Brown Road
Mesa, Arizona 85213 USA
Phone: 480-962-4545 Fax: 480-835-6442
information@westerndesigncenter.com
www.westerndesigncenter.com

WARNING: MOS CIRCUITS ARE SUBJECT TO DAMAGE FROM STATIC DISCHARGE

Internal static discharge circuits are provided to minimize part damage due to environmental static electrical charge build-ups. Industry established recommendations for handling MOS circuits include:

1. Ship and store product in conductive shipping tubes or conductive foam plastic. Never ship or store product in non-conductive plastic containers or non-conductive plastic foam material.
2. Handle MOS parts only at conductive work stations.
3. Ground all assembly and repair tools.