W65C816S DATA SHEET

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.

TABLE OF CONTENTS

INTRODUCTION

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 processor programming model, Figure 1-1, 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 allow 24-bit memory addressing with segmented or linear addressing.

Four new signals provide the system designer with many options. The ABORT 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 (VP) output. Future Microprocessors will support all current W65C816S operating modes for both index and offset address generation.

The information included in this data sheet reflects a standard 5 volt power supply specification. The testing process can be modified to meet most custom needs for power supply voltage, temperature or timing.

KEY FEATURES OF THE W65C816S

- Advanced fully static CMOS design for low power consumption and increased noise immunity
- Single 1.2-6.0 volt power supply, as specified
- Emulation mode allows complete hardware and software compatibility with 6502 designs
- 24-bit address bus allows 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 allows dual cache and cycle steal DMA implementation
- Vector Pull (VP) output indicates when interrupt vectors are being addressed; may be used to implement vectored interrupt design
- Abort (ABORT) input and associated vector supports virtual memory system design
- Low power consumption (2mA @ 1MHz) allows battery-powered operation (1FA) standby current.
- 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 bit manipulation instructions which do not exist for the W65C816S

SECTION 1

W65C816S FUNCTIONAL DESCRIPTION

The W65C816S provides the design engineer with upward mobility and software compatibility in applications where a 16-bit system configuration is desired. The W65C816S's 16-bit hardware configuration, coupled with current software, allows a wide selection of system applications. In the Emulation mode, the W65C816S offers many advantages, including full software compatibility with 6502 coding. In addition, the W65C816S's powerful instruction set and addressing modes make it an excellent choice for new 16-bit designs.

Internal organization of the W65C816S can be divided into two parts: 1) The Register Section and 2) The Control Section. Instructions (or OpCodes) 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 architecture with an 8-bit external data bus.

1.1 Instruction Register and Decode (IR)

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

1.2 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.

1.3 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.

1.4 Internal Registers (Refer to Programming Model)

1.5 Accumulators (A,B,C)

The Accumulator is a general purpose register which stores one of the operands, or 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.

1.6 Data Bank Register (DBR)

During modes of operation, the 8-bit Data Bank Register holds the default 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.

1.7 Direct (D)

The 16-bit Direct Register provides an address offset for all instructions using direct addressing. The effective bank zero address is formed by adding the 8-bit instruction operand address to the Direct Register. The Direct Register is initialized to zero during Reset.

1.8 Index (X and Y)

There are two Index Registers (X and Y) which may be used as general purpose registers or to provide 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 the 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 (providing the Index Select Bit (X) 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.

1.9 Processor Status (P)

The 8-bit Processor Status Register 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 the Emulation mode. When an interrupt occurs during the Emulation mode, the Break flag is written to stack memory as bit 4 of the Processor Status Register.

1.10 Program Bank Register (PBR)

The 8-bit Program Bank Register 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.

1.11 Program Counter (PC)

The 16-bit Program Counter Register provides the addresses which are used to step the microprocessor through sequential program instructions. The register is incremented each time an instruction or operand is fetched from program memory.

1.12 Stack Pointer (S)

The Stack Pointer 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 allows simple implementation of nested subroutines and multiple-level interrupts. During the Emulation mode, the Stack Pointer high-order byte (SH) is always equal to one. The bank address for all stack operations is Bank zero.

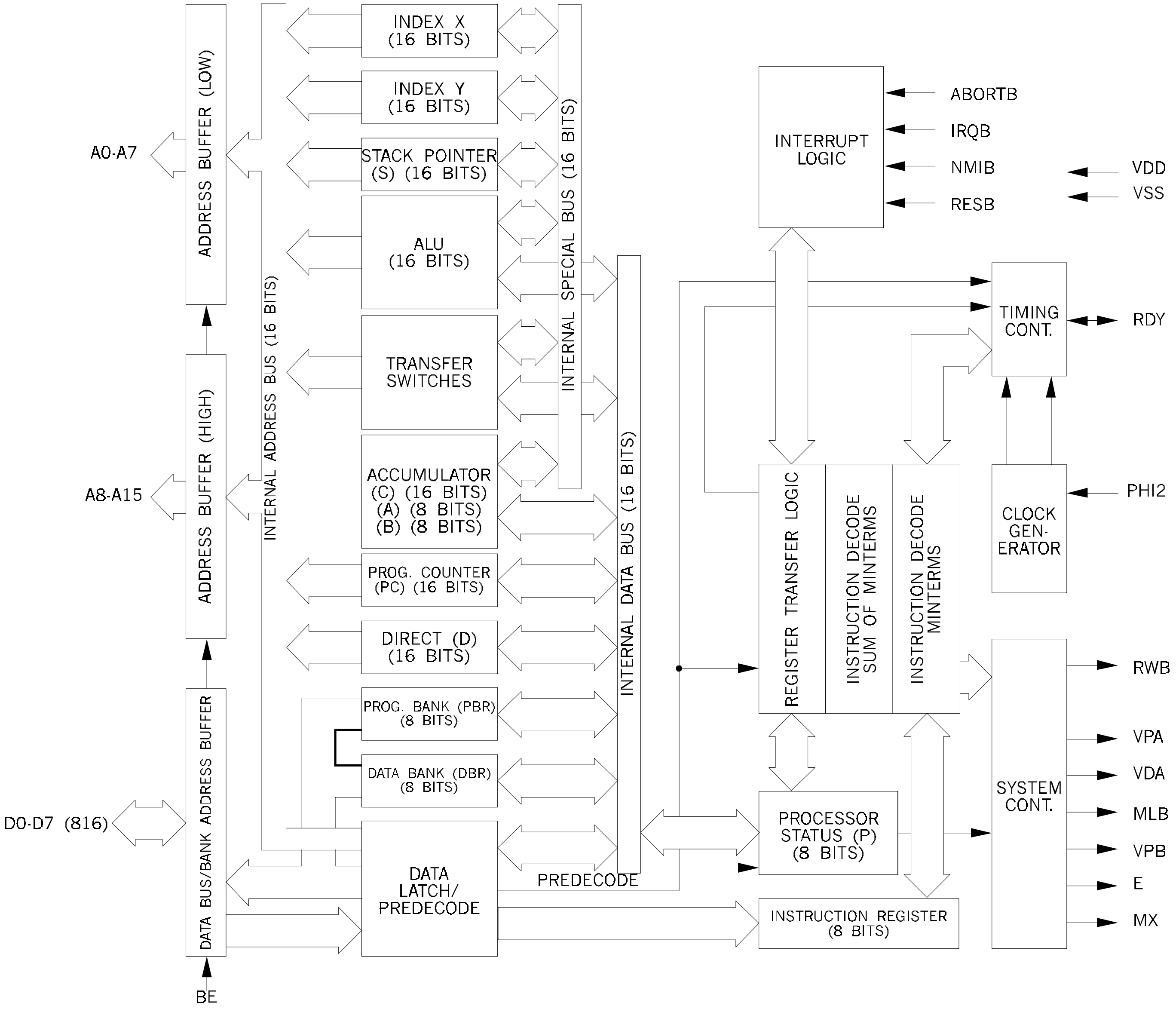

A0-A7
ADDRESS BUFFER (LOW)
A8-A15
ADDRESS BUFFER (HIGH)
D0-D7 (816)
DATA BUS/BANK ADDRESS BUFFER
BE
INTERNAL ADDRESS BUS (16 BITS)
INDEX X (16 BITS)
INDEX Y (16 BITS)
STACK POINTER (S) (16 BITS)
ALU (16 BITS)
TRANSFER SWITCHES
ACCUMULATOR (C) (16 BITS) (A) (8 BITS) (B) (8 BITS)
PROG. COUNTER (PC) (16 BITS)
DIRECT (D) (16 BITS)
PROG. BANK (PBR) (8 BITS)
DATA BANK (DBR) (8 BITS)
DATA LATCH/ PREDECODE
PREDECODE
INTERNAL SPECIAL BUS (16 BITS)
INTERNAL DATA BUS (16 BITS)
INTERRUPT LOGIC
ABORTB
IRQB
NMIB
RESB
VDD
VSS
TIMING CONT.
RDY
REGISTER TRANSFER LOGIC
INSTRUCTION DECODE SUM OF MINTERMS
INSTRUCTION DECODE MINTERMS
CLOCK GEN-ERATOR
PHI2
PROCESSOR STATUS (P) (8 BITS)
INSTRUCTION REGISTER (8 BITS)
SYSTEM CONT.
RWB
VPA
VDA
MLB
VPB
E
MX

Figure 1-1 W65C816S Internal Architecture Simplified Block Diagram

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

Figure 1-2 W65C816S Microprocessor Programming Model

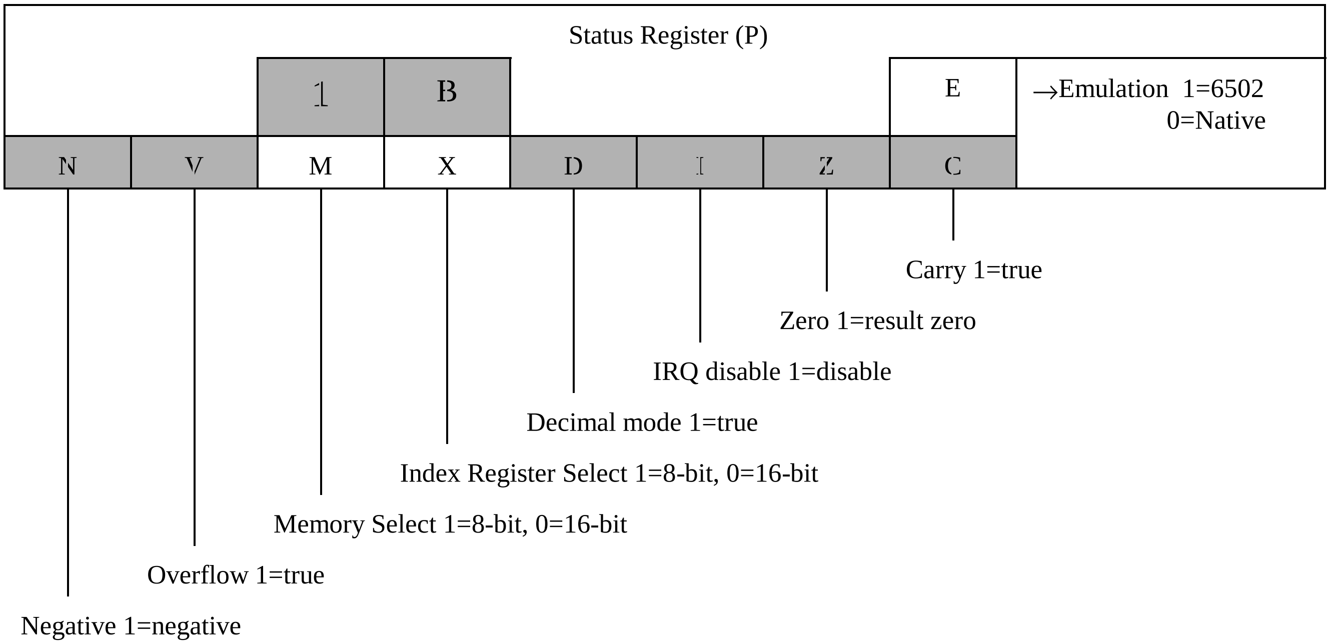
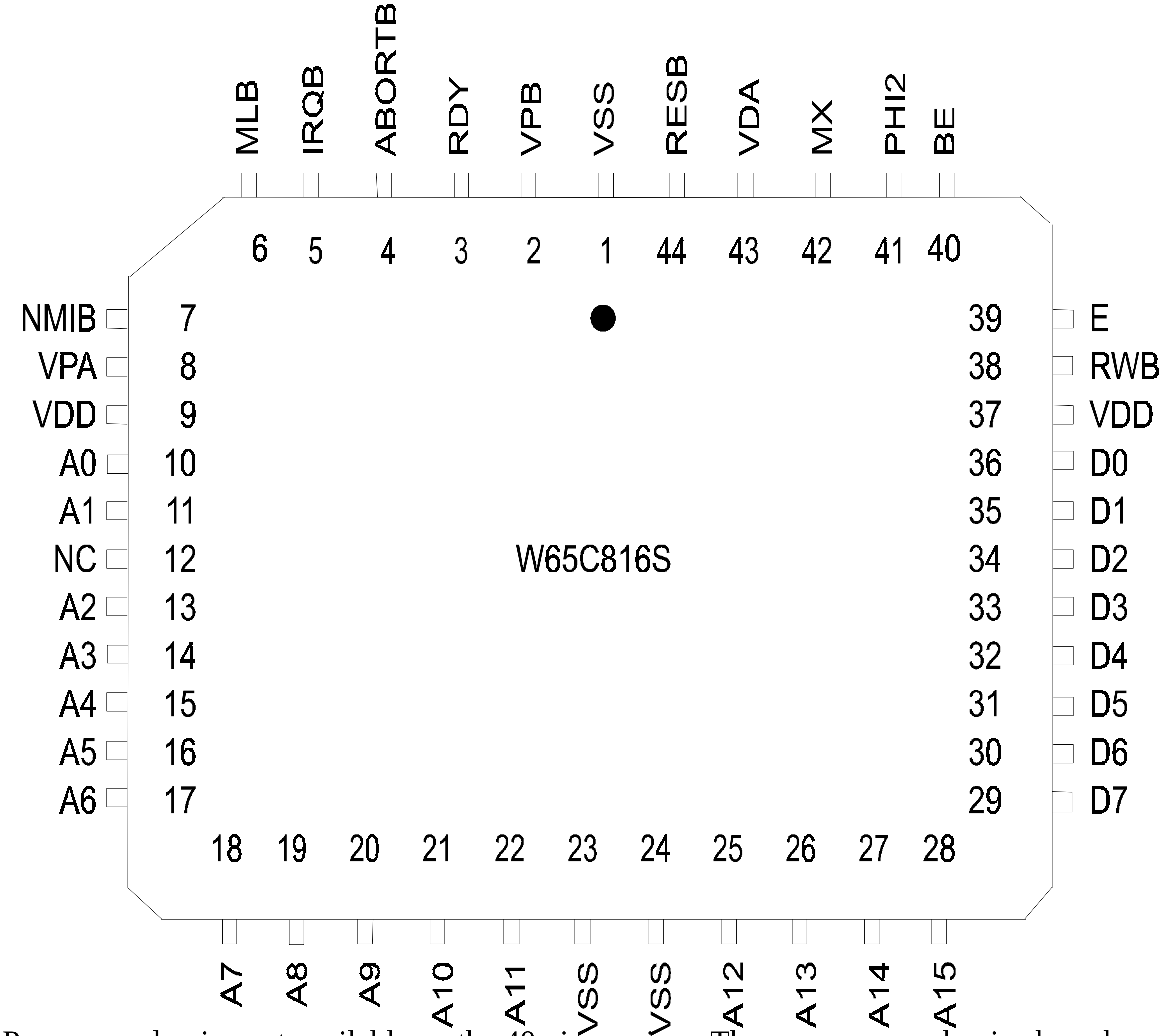


Figure 1-3 W65C816S Status Register Coding

SECTION 2

PIN FUNCTION DESCRIPTION



(1) Power supply pins not available on the 40 pin version. These power supply pins have been added for improved performance.

Figure 2-1 W65C816S 44 Pin PLCC Pinout

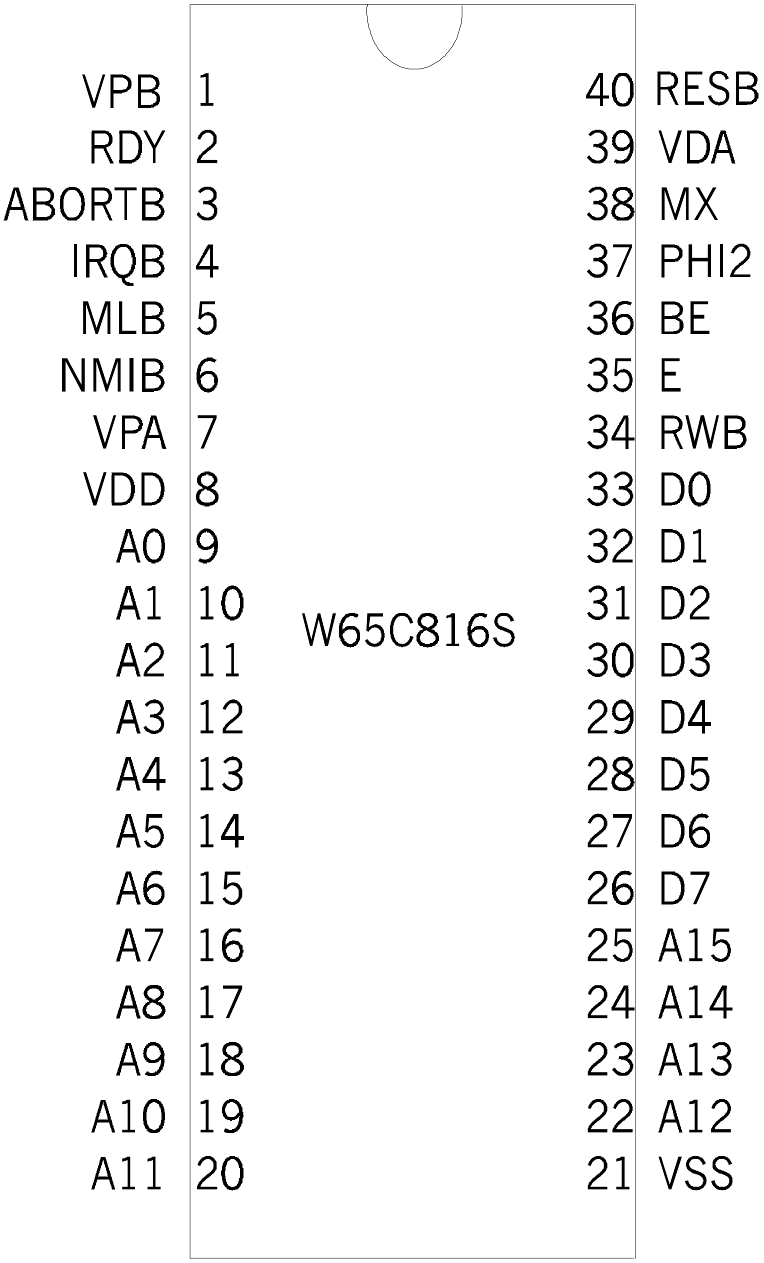
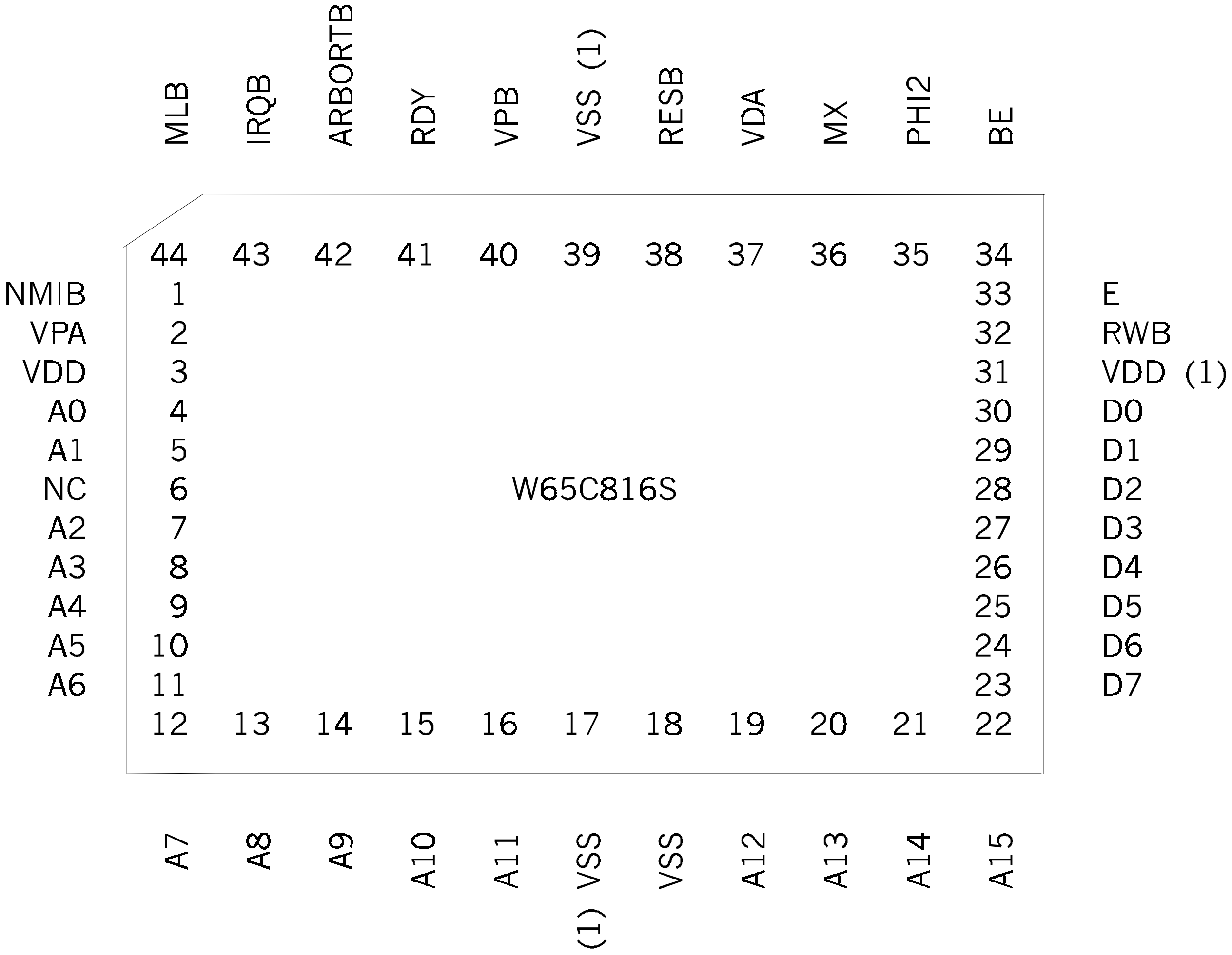


Figure 2-2 W65C816S 40 Pin PDIP Pinout



(1) Power supply pins not available on the 40 pin version. These power supply pins have been added for improved performance.

Figure 2-3 W65C816S 44 PIN QFP

Table 2-1 Pin Function Table

Pin	Description
A0-A15	Address Bus
ABORTB	Abort Input
BE	Bus Enable
PHI2	Phase 2 In Clock
D0-D7/BA7	Data Bus/Bank Address Bus
E	Emulation Select
IRQB	Interrupt Request
MLB	Memory Lock
MX	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 (+5 volts)
VSS	Internal Logic Ground

2.1 Abort (ABORTB)

The Abort 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.

2.2 Address Bus (A0-A15)

These sixteen output lines form the 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.

2.3 Bus Enable (BE)

The Bus Enable 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.

2.4 Data/Address Bus (D0-D7/BA7)

These eight lines multiplex address bits BA0-BA7 with the data value. The 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.

2.5 Emulation Status (E)

The Emulation Status 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.

2.6 Interrupt Request (IRQB)

The Interrupt Request 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 3 bytes of information to be pushed onto the stack before jumping to the interrupt handler. The first byte is the high byte in the Program Counter. The second byte is the Program Counter low byte. The third byte is the status register valve. These valves are used to return the processor to it's original state prior to the IRQ interrupt.

2.7 Memory Lock (MLB)

The Memory Lock 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.

2.8 Memory/Index Select Status (MX)

This 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 the Phase 2 clock negative transition and Flag X is valid during he Phase 2 clock positive transition. These bits may be thought of as OpCode extensions and may be used for memory and system management.

2.9 Non-Maskable Interrupt (NMIB)

A negative transition on the 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. Also, no interrupt will occur if NMIB remains low. The NMIB signal going low causes 3 bytes of information to be pushed onto the stack before jumping to the interrupt handler. The first byte is the high byte in the Program Counter. The second byte is the Program Counter low byte. The third byte is the status register valve. These valves are used to return the processor to it's original state prior to the NMI interrupt.

2.10 Phase 2 In (PHI2)

This is the system clock input to the microprocessor internal clock generator (equivalent to PHI0(IN) on the 6502). During the low power Standby Mode, PHI2 can be held in either state to preserve the contents of internal registers.

2.11 Read/Write (RWB)

When the RWB output signal is in the high state, the microprocessor is reading data from memory or I/O. When in the low state, the Data Bus contains valid data from the microprocessor which is to be stored at the addressed memory location. When using the W65C816S, the RWB signal may be set to the high impedance state by Bus Enable (BE).

2.12 Ready (RDY)

This bi-directional signal indicates that a Wait for Interrupt (WAI) instruction has been executed allowing the user to halt operation of the microprocessor. A low input logic level will halt the microprocessor in its current state. Returning RDY to the active high state allows the microprocessor to continue following the next Phase 2 In Clock 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 provided. This feature may be used to eliminate interrupt latency by placing the WAI instruction at the beginning of the IRQB servicing routine. 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. When outputting a low level, the pull-up is disabled. The RDY pin can still be wired ORed.

2.13 Reset (RESB)

The Reset 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 pullup 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 period, the following processor initialization takes place:

Registers

D=0000	SH=01
DBR=00	XH=00
PRB=00	YH=00

	N	V	M	X	D	I	Z	C/E	
P =	*	*	1	1	0	1	*	*/1	*=not initialized

STP and WAI instructions are cleared.

Signals

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

When Reset is brought high, an interrupt sequence is initiated:

- RWB remains in the high state during the stack address cycles.
- The Reset vector address is 00FFFC,D.

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

These two output signals indicate valid memory addresses when high and must be 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

2.15 VDD and VSS

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

2.16 Vector Pull (VPB)

The Vector Pull 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 reads the interrupt vector. The VPB signal may be used to select and prioritize interrupts from several sources by modifying the vector addresses.

SECTION 3

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:

3.1 Reset and Interrupt Vectors

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

3.2 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.

3.3 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).

3.4 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.

3.5 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:

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

3.5.1 Immediate Addressing-#

The operand is the second byte (second and third bytes when in the 16-bit mode) of the instruction.

3.5.2 Absolute-a

With Absolute 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

3.5.3 Absolute Long-al

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	

3.5.4 Direct-d

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.

Instruction:	OpCode	offset	
		Direct Register	
	+		offset
Operand Address:	00	effective address	

3.5.5 Accumulator-A

This form of addressing always uses a single byte instruction. The operand is the Accumulator.

3.5.6 Implied-i

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

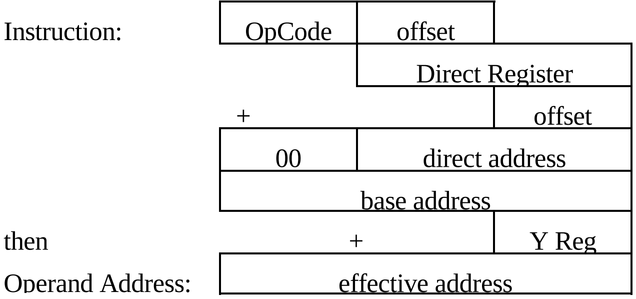
3.5.7 Direct Indirect Indexed-(d),y

This address mode is often referred to as Indirect,Y. The second byte of the instruction is added to the Direct Register (D). The 16-bit contents 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.

Instruction:		OpCode	offset	
			Direct Register	
		+		offset
		00	(direct address)	
then:	+	DBR		
		base address		
	+			Y Reg
Operand Address:		effective address		

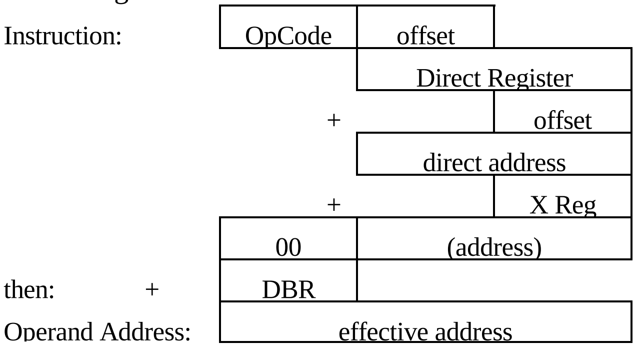
3.5.8 Direct Indirect Long Indexed-[d],y

With this addressing mode, 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.



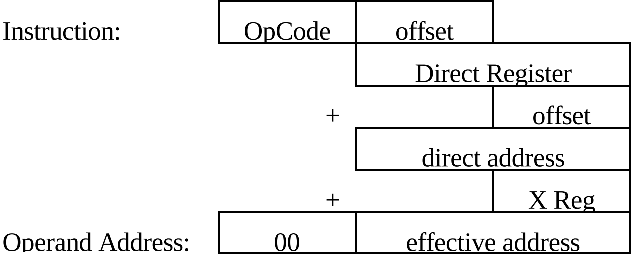
3.5.9 Direct Indexed Indirect-(d,x)

This address mode is often referred to as Indirect,X. 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.



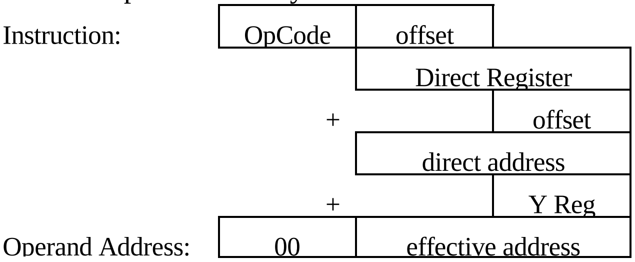
3.5.10 Direct Indexed With X-d,x

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.



3.5.11 Direct Indexed With Y-d,y

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.



3.5.12 Absolute Indexed With X-a,x

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:	OpCo	addrl	addrh
	DBR	addrh	addrl
	+		X Reg
Operand	effective address		

3.5.13 Absolute Long Indexed With X-al,x

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:	OpCo	addrl	addrh	baddr
	baddr	addrh	addrl	
	+		X Reg	
Operand	effective address			

3.5.14 Absolute Indexed With Y-a,y

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:	OpCo	addrl	addrh
	DBR	addrh	addrl
	+		Y Reg
Operand	effective address		

3.5.15 Program Counter Relative-r

This address mode, referred to as Relative Addressing, 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.

3.5.16 Program Counter Relative Long-rl

This address mode, referred to as Relative Long Addressing, is used 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.

3.5.17 Absolute Indirect-(a)

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 Address:		00	addrh	addrl

New PC = (indirect address)
with JML:
New PC = (indirect address)
New PBR = (indirect address +2)

3.5.18 Direct Indirect-(d)

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.

Instruction:	OpCode	offset	
		Direct Register	
	+		offset
	00	(direct address)	
then: +	DBR		
Operand Address:	effective address		

3.5.19 Direct Indirect Long-[d]

The second byte of the instruction is added to the Direct Register to form a pointer to the 24-bit effective address.

Instruction:	OpCode	offset	
		Direct Register	
then:	+		offset
	00	(direct address)	
Operand Address:	direct address		

3.5.20 Absolute Indexed Indirect-(a,x)

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. The Program Bank Register is not changed.

Instruction:	OpCode	addrl	addrh
		addrh	addrl
			X Reg
	PBR	address	

then:
PC = (address)

3.5.21 Stack-s

Stack 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.

3.5.22 Stack Relative-d,s

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 is always zero. The relative offset is an unsigned 8-bit quantity in the range of 0 to 255.

Instruction:	OpCode	offset	
		Stack Pointer	
then:	+		offset
Operand Address:	00	effective address	

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

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.

```
Instruction:            | OpCode | offset |
                                 |   Stack Pointer  |
                                          | offset  |
                        |   00   |    S + offset    |
then         +          |  DBR   |
                        |       base address        |
                +                         |  Y Reg  |
Operand Address:        |     effective address     |
```

3.5.24 Block Source Bank, Destination Bank-xyc

This addressing mode is used by the Block Move instructions. The second byte of the instruction contains the high-order 8 bits of the destination address. 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. 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 (MVN) or decrement (MVP) X and Y.
Decrement C (if greater than zero), then PC+3 6 PC.

Table 3-2 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 W65C816 S
1. Immediate	2	2 (3)	2	2 (3)
2. Absolute	4 (5)	4 (3,5)	3	3
3. Absolute Long	-	5 (3)	-	4
4. Direct	3 (5)	3 (3,4,5)	2	2
5. Accumulator	2	2	1	1
6. Implied	2	2	1	1
7. Direct Indirect Indexed (d),y	5 (1)	5 (1,3,4)	2	2
8. Direct Indirect Indexed Long [d],y	-	6 (3,4)	-	2
9. Direct Indexed Indirect (d,x)	6	6 (3,4)	2	2
10. Direct, X	4 (5)	4 (3,4,5)	2	2
11. Direct, Y	4	4 (3,4)	2	2
12. Absolute, X	4 (1,5)	4 (1,3,5)	3	3
13. Absolute Long, X	-	5 (3)	-	4
14. Absolute, Y	4 (1)	4 (1,3)	3	3
15. Relative	2 (1,2)	2 (2)	2	2
16. Relative Long	-	3 (2)	-	3
17. Absolute Indirect (Jump)	5	5	3	3
18. Direct Indirect	-	5 (3,4)	-	2
19. Direct Indirect Long	-	6 (3,4)	-	2
20. Absolute Indexed Indirect (Jump)	-	6	-	3
21. Stack	3-7	3-8	1-3	1-4
22. Stack Relative	-	4 (3)	-	2
23. Stack Relative Indirect Indexed	-	7 (3)	-	2
24. Block Move X,Y,C (Source, Destination, Block Length)	-	7	-	3

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.

SECTION 4

TIMING, AC AND DC CHARACTERISTICS

4.1 Absolute Maximum Ratings

Table 4-1 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 these conditions is not implied.

4.2 DC Characteristics VDD = 5.0V +/- 5%, VSS = 0V, TA = 0°C to +70°C

Table 4-2 DC Characteristics

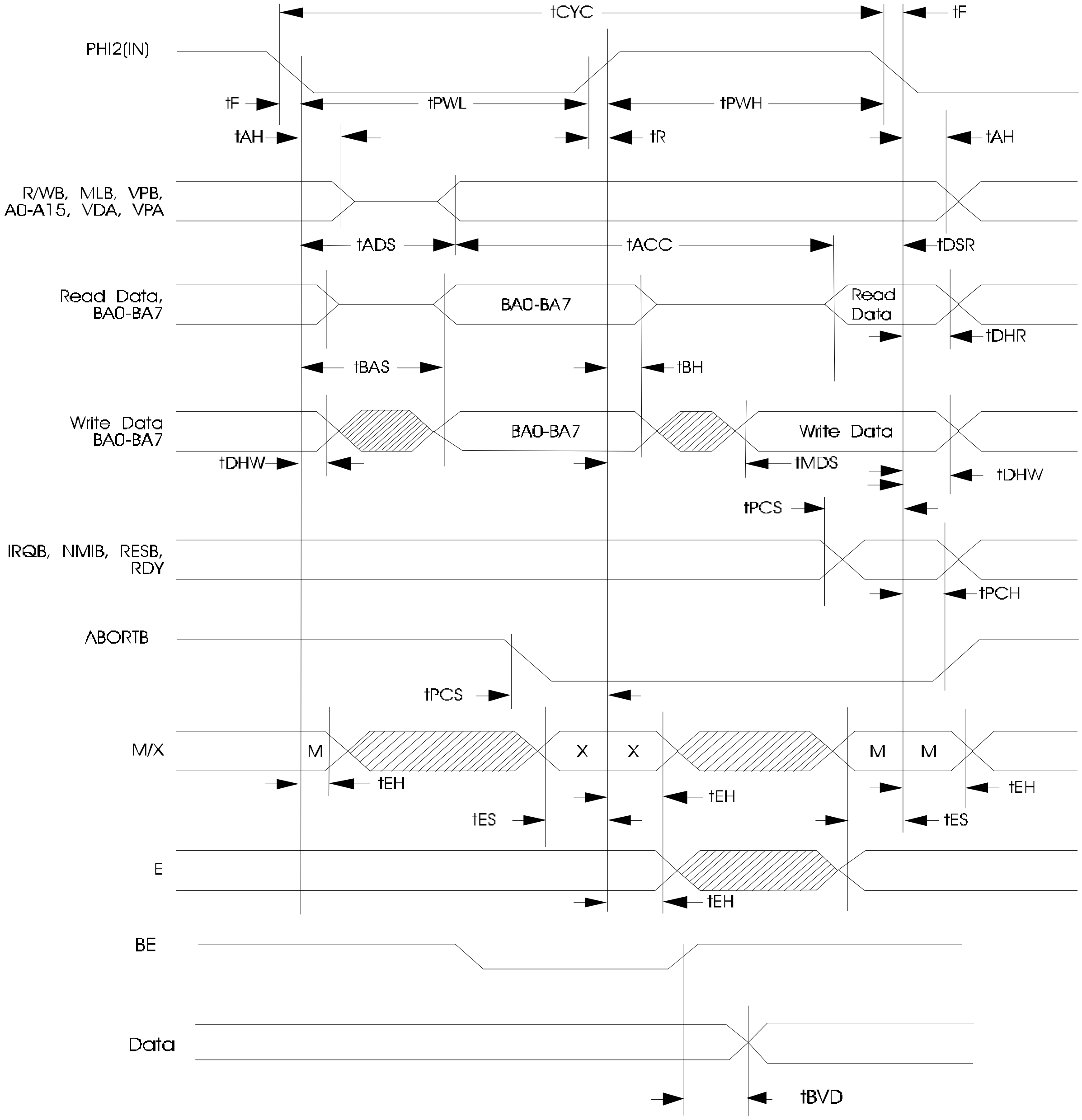
Parameter	Symbol	Min	Max	Unit
Input High Voltage RDY, IRQB, Data, PHI2, NMIB, ABORTB, BE, RESB	Vih	VDD-0.2V VDD-0.2V	VDD+0.3 VDD+0.3	V V
Input Low Voltage RDY, IRQB, Data, PHI2, NMIB, ABORTB, BE, RESB	Vil	VSS-0.3 VSS-0.3	VSS+0.2V VSS+0.2V	V V
Input Leakage Current (Vin=0.4 to 2.4) RDY, (Active Pullup) PHI2 Address, Data, RWB, (Off state, BE=0), All other inputs	Iin	-100 -1 -10	10 1 10	µA µA µA
Output High Voltage (Ioh=-100FA) Data, Address, RWB, MLB, VPB, MX, E, VDA, VPA	Voh	0.7VDD	-	V
Output Low Voltage (Iol=1.6mA) Data, Address, RWB, MLB, VPB, MX, E, VDA, VPA	Vol	-	0.4	V
Supply Current (no load)	Idd		3	mA/MHz
Standby Current (No Load, Data Bus = VSS or VDD) RESB, NMIB, IRQB, BE, ABORTB, PHI2=VDD	Isby	-	5	µA
Capacitance (Vin=0V, TA=25EC, f=2MHz) PHI2, MX, VDA, RESB, VPB, RDY, ABORTB, IRQB, MLB, NMIB, VPA, E, BE Address, Data, R/W- (Off state) * Not inspected during production test; verified on a sample basis.	Cin Cts	- -	10 15	pF pF

4.3 AC Characteristics VDD= 5.0V +/- 5%, VSS= 0V, TA= 0°C to +70°C (1)

Table 4-3 W65C816S AC Characteristics - 14 MHz

Parameter	Symbol	14 MHz		Unit
		Min	Max	
Cycle Time	tCYC	70	DC	nS
Clock Pulse Width Low	tPWL	35	-	nS
Clock Pulse Width High	tPWH	35	-	nS
Fall Time, Rise Time	tF,tR	-	5	nS
A0-A15 Hold Time	tAH	10	-	nS
A0-A15 Setup Time	tADS	-	30	nS
BA0-BA7 Hold Time	tBH	10	-	nS
BA0-BA7 Setup Time	tBAS	-	33	nS
Access Time	tACC	30	-	nS
Read Data Hold Time	tDHR	10	-	nS
Read Data Setup Time	tDSR	10	-	nS
Write Data Delay Time	tMDS	-	30	nS
Write Data Hold Time	tDHW	10	-	nS
Processor Control Setup Time	tPCS	10	-	nS
Processor Control Hold Time	tPCH	10	-	nS
E, MX Output Hold Time	tEH	5	-	nS
E, MX Output Setup Time	tES	10	-	nS
Capacitive Load (2)	CEXT	-	35	pF
BE to Valid Data (3)	tBVD	-	25	nS

1. Custom testing available covering full, voltage range 1.2-6.0 volts, temperature and timing
2. Applied to Address, Data, RWB
3. BE to High Impedance State is not testable but should be the same amount of time as BE to Valid Data.



Timing Notes:
1. Timing measurement points are 1.5V and 1.5V for VDD = 5V.

Figure 4-1 General Timing Diagram

SECTION 5

OPERATION TABLES

Table 5-1 W65C816S Instruction Set-Alphabetical Sequence (continued on following page)

ADC	Add Memory to Accumulator with Carry	INY	Increment Index Y by One
AND	"AND" Memory with Accumulator	JML	Jump Long
ASL	Shift One Bit Left, Memory or Accumulator	JMP	Jump to New Location
BCC	Branch on Carry Clear (Pc=0)	JSL	Jump Subroutine Long
BCS	Branch on Carry Set (Pc=1)	JSR	Jump to News Location Saving Return
BEQ	Branch if Equal (Pz=1)	LDA	Load Accumulator with Memory
BIT	Bit Test	LDX	Load Index X with Memory
BMI	Branch if Result Minus (Pn=1)	LDY	Load Index Y with Memory
BNE	Branch if Not Equal (Pz=0)	LSR	Shift One Bit Right (Memory or Accumulator)
BPL	Branch if Result Plus (Pn=0)	MVN	Block Move Negative
BRA	Branch Always	MVP	Block Move Positive
BRK	Force Break	NOP	No Operation
BRL	Branch Always Long	ORA	"OR" Memory with Accumulator
BVC	Branch on Overflow Clear (Pv=0)	PEA	Push Effective Absolute Address on Stack (or Push Immediate Data on Stack)
BVS	Branch on Overflow Set (Pv=1)	PEI	Push Effective Absolute Address on Stack (Or Push Direct Data on Stack)
CLC	Clear Carry Flag	PER	Push Effective Program Counter Relative Address on Stack
CLD	Clear Decimal Mode	PHA	Push Accumulator on Stack
CLI	Clear Interrupt Disable Bit	PHB	Push Data Bank Register on Stack
CLV	Clear Overflow Flag	PHD	Push Direct Register on Stack
CMP	Compare Memory and Accumulator	PHK	Push Program Bank Register on Stack
COP	Coprocessor	PHP	Push Processor Status on Stack
CPX	Compare Memory and Index X	PHX	Push Index X on Stack
CPY	Compare Memory and Index Y	PHY	Push Index Y on Stack
DEC	Decrement Memory or Accumulator by One	PLA	Pull Accumulator from Stack
DEX	Decrement Index X by One	PLB	Pull Data Bank Register from Stack
DEY	Decrement Index Y by One	PLD	Pull Direct Register from Stack
EOR	"Exclusive OR" Memory with Accumulator	PLP	Pull Processor Status from Stack
INC	Increment Memory or Accumulator by One	PLX	Pull Index X from Stack
INX	Increment Index X by One	PLY	Pull Index Y from Stack

REP	Reset Status Bits	TAY	Transfer Accumulator to Index Y
ROL	Rotate One Bit Left (Memory or Accumulator)	TCD	Transfer C Accumulator to Direct Register
ROR	Rotate One Bit Right (Memory or Accumulator)	TCS	Transfer C Accumulator to Stack Pointer Register
RTI	Return from Interrupt	TDC	Transfer Direct Register to C Accumulator
RTL	Return from Subroutine Long	TRB	Test and Reset Bit
RTS	Return from Subroutine	TSB	Test and Set Bit
SBC	Subtract Memory from Accumulator with Borrow	TSC	Transfer Stack Pointer Register to C Accumulator
SEP	Set Processor Status Bit	TSX	Transfer Stack Pointer Register to Index X
SEC	Set Carry Flag	TXA	Transfer Index X to Accumulator
SED	Set Decimal Mode	TXS	Transfer Index X to Stack Pointer Register
SEI	Set Interrupt Disable Status	TXY	Transfer Index X to Index Y
STA	Store Accumulator in Memory	TYA	Transfer Index Y to Accumulator
STP	Stop the Clock	TYX	Transfer Index Y to Index X
STX	Store Index X in Memory	WAI	Wait for Interrupt
STY	Store Index Y in Memory	WDM	Reserved for future use
STZ	Store Zero in Memory	XBA	Exchange B and A Accumulator
TAX	Transfer Accumulator in Index X	XCE	Exchange Carry and Emulation Bits

Table 5-2 Vector Locations

E = 1			E = 0		
00FFFE,F-	IRQB/BRK	Hardware/Software	00FFEE,F-	IRQB	Hardware
00FFFC,D-	RESETB	Hardware	00FFEC,D-	(Reserved)	
00FFFA,B-	NMIB	Hardware	00FFEA,B-	NMIB	Hardware
00FFF8,9-	ABORTB	Hardware	00FFE8,9-	ABORTB	Hardware
00FFF6,7-	(Reserved)		00FFE6,7-	BRK	Software
00FFF4,5-	COP	Software	00FFE4,5-	COP	Software

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.

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

5-3 OpCode Matrix (continued on following page)

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

Op Code Matrix Legend

INSTRUCTION MNEMONIC		ADDRESSING MODE
	∗= New W65C816S OpCodes ● = New W65C02 OpCodes Blank = NMOS 6502 OpCodes	
BASE NO. BYTES		BASE NO. CYCLES

Table 5-4 Operation, Operation Codes and Status Register (continued on next 3 pages)

Mnemonic	Operation													Processor Status Code							
														7	6	5	4	3	2	1	0
	∧ AND ∨ OR ⊻ Exclusive OR	#	a	al	d	A	i	(d),y	[d],y	(d,x)	d,x	d,y	a,x	N	V	M	X	D	I	Z	C
		1	2	3	4	5	6	7	8	9	10	11	12	N	V	I	B	D	I	Z	C
ADC	A +M + C → A	69	6D	6F	65			71	77	61	75		7D	N	V	.	.	.	.	Z	C
AND	A ∧ M → A	29	2D	2F	25			31	37	21	35		3D	N	.	.	.	.	.	Z	.
ASL	C ← $\boxed{15/7 \quad 0}$ ← 0		0E		06	0A					16		1E	N	.	.	.	.	.	Z	C
BCC	Branch if C=0													.	.	.	.	.	.	.	.
BCS	Branch if C=1													.	.	.	.	.	.	.	.
BEQ	Branch if Z=1													.	.	.	.	.	.	.	.
BIT	A ∧M (Note 1)	89	2C		24						34		3C	M_7	M_6	.	.	.	.	Z	.
BMI	Branch if N=1													.	.	.	.	.	.	.	.
BNE	Branch if Z=0													.	.	.	.	.	.	.	.
BPL	Branch if N=0													.	.	.	.	.	.	.	.
BRA	Branch Always													.	.	.	.	.	.	.	.
BRK	Break (Note 2)													.	.	.	.	0	1	.	.
BRL	Branch Long Always													.	.	.	.	.	.	.	.
BVC	Branch if V=0													.	.	.	.	.	.	.	.
BVS	Branch if V=1													.	.	.	.	.	.	.	.
CLC	0-C						18							.	.	.	.	.	.	.	0
CLD	0→D						D8							.	.	.	.	0	.	.	.
CLI	0→1						58							.	.	.	.	.	0	.	.
CLV	0→V						B8							.	0	.	.	.	.	.	.
CMP	A-M	C9	CD	CF	C5			D1	D7	C1	D5		DD	N	.	.	.	.	.	Z	C
COP	Co-Processor													.	.	.	.	0	1	.	.
CPX	X-M	E0	EC		E4									N	.	.	.	.	.	Z	C
CPY	Y-M	C0	CC		C4									N	.	.	.	.	.	Z	C
DEC	Decrement		CE		C6	3A					D6		DE	N	.	.	.	.	.	Z	.
DEX	X-1→X						CA							N	.	.	.	.	.	Z	.
DEY	Y-1→Y						88							N	.	.	.	.	.	Z	.
EOR	A⊻M→A	49	4D	4F	45			51	57	41	55		5D	N	.	.	.	.	.	Z	.
INC	Increments		EE		E6	1A					F6		FE	N	.	.	.	.	.	Z	.
INX	X+1→X						E8							N	.	.	.	.	.	Z	.
INY	Y+1→Y						C8							N	.	.	.	.	.	Z	.
JML	Jump Long to new Location													.	.	.	.	.	.	.	.
JMP	Jump to New Location		4C	5C										.	.	.	.	.	.	.	.
JSL	Jump Long to Subroutine			22										.	.	.	.	.	.	.	.
JSR	Jump to Subroutine		20											.	.	.	.	.	.	.	.
LDA	M→A	A9	AD	AF	A5			B1	B7	A1	B5		BD	N	.	.	.	.	.	Z	.
LDX	M→X	A2	AE		A6							B6		N	.	.	.	.	.	Z	.
LDY	M→Y	A0	AC		A4						B4		BC	N	.	.	.	.	.	Z	.
LSR	0 → $\boxed{15/7 \quad 0}$ → C		4E		46	4A					56		5E	0	.	.	.	.	.	Z	C
MVM	M→M Negative													.	.	.	.	.	.	.	.
MVP	M→M Positive													.	.	.	.	.	.	.	.
NOP	No Operation						EA							.	.	.	.	.	.	.	.
ORA	A∨M→A	09	0D	0F	05			11	17	01	15		1D	N	.	.	.	.	.	Z	.
PEA	Mpc+1,Mpc+2→Ms-1, Ms S-2→S													.	.	.	.	.	.	.	.
PEI	M(d), M(d+1)→Ms-1, Ms S-2→S													.	.	.	.	.	.	.	.
PER	Mpc+rl, Mpc+rl+1→Ms-1 MsS-2→S													.	.	.	.	.	.	.	.

Mnemonic	Operation													Processor Status Code							
														7	6	5	4	3	2	1	0
	∧ AND ∨ OR ⊻ Exclusive OR	#	a	al	d	A	i	(d),y	(d),y	(d),x	d,x	d,y	a,x	N	V	M	X	D	I	Z	C
		1	2	3	4	5	6	7	8	9	10	11	12	N	V	I	B	D	I	Z	C
PHA	A→Ms, S-1→S													.	.	.	.	.	.	.	.
PHB	DBR→Ms,S-1→S													.	.	.	.	.	.	.	.
PHD	D→Ms,Ms-1,S-2→S													.	.	.	.	.	.	.	.
PHK	PBR→Ms,S-1→S													.	.	.	.	.	.	.	.
PHP	P→Ms,S-1→S													.	.	.	.	.	.	.	.
PHX	X→Ms,S-1→S													.	.	.	.	.	.	.	.
PHY	Y→Ms,S-1→S													.	.	.	.	.	.	.	.
PLA	S + 1→S,Ms→A													N	.	.	.	.	.	Z	.
PLB	S + 1→S,Ms→DBR													N	.	.	.	.	.	Z	.
PLDL	S + 2 →S,Ms-1,Ms→D													N	.	.	.	.	.	Z	.
PLP	S + 1→S,MS→P													N	V	M	X	D	I	Z	C
PLX	S + 1→S,MS→X													N	.	.	.	.	.	Z	.
PLY	S + 1→S,MS→Y													N	.	.	.	.	.	Z	.
REP	M-∧P→P	C2												N	V	M	X	D	I	Z	C
ROL	15/7 0 ← C		2E		26	2A					36		3E	N	.	.	.	.	.	Z	C
ROR	→ C → 15/7 0													N	.	.	.	.	.	Z	C
RTI	Return from Interrupt		6E		66	6A					76		7E	N	V	M	X	D	I	Z	C
RTL	Return from Subroutine Long													.	.	.	.	.	.	.	.
RTS	Return Subroutine													.	.	.	.	.	.	.	.
SBC	A-M-C→A	E9	ED	EF	E5			F1	F6	E1	F5		FD	N	V	.	.	.	.	Z	C
SEC	1→C						38							.	.	.	.	.	.	.	1
SED	1→D						F8							.	.	.	.	1	.	.	.
SEI	1→I						78							.	.	.	.	.	1	.	.
SEP	M∨P→P	E2												N	V	M	X	D	I	Z	C
STA	A→M		8D	8F	85			91	97	81	95		9D	.	.	.	.	.	.	.	.
STP	STOP(1→PHI2)						DB							.	.	.	.	.	.	.	.
STX	X→M		8E		86							96		.	.	.	.	.	.	.	.
STY	Y→M		8C		84						94			.	.	.	.	.	.	.	.
STZ	00→M		9C		64						74		9E	.	.	.	.	.	.	.	.
TAX	A→X													N	.	.	.	.	.	Z	.
TAY	A→Y					A8								N	.	.	.	.	.	Z	.
TCD	C→D					5B								N	.	.	.	.	.	Z	.
TCS	C→S					1B								.	.	.	.	.	.	.	.
TDC	D→C					7B								N	.	.	.	.	.	Z	.
TRB	A∧M→M		1C		14									.	.	.	.	.	.	Z	.
TSB	A∨M→M		0C		04									.	.	.	.	.	.	Z	.
TSC	S→C						3B							N	.	.	.	.	.	Z	.
TSX	S→X						BA							N	.	.	.	.	.	Z	.
TSA	X→A						SA							N	.	.	.	.	.	Z	.
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

Mnemonic	Operation													Processor Status Code							
														7	6	5	4	3	2	1	0
	∧ AND ∨ OR ⊻ Exclusive OR	al,x	a,y	rl	rl	(a)	(d)	(d)	(a,x)	s	d,s	(d,s),y	a,x	N	V	M	X	D	I	Z	C
		13	14	15	16	17	18	19	20	21	22	23	24	N	V	I	B	D	I	Z	C
ADC	A +M + C → A	7F	79				72	67			63	73		N	V	.	.	.	.	Z	C
AND	A ∧ M → A	3F	39				32	27			23	33		N	.	.	.	.	.	Z	.
ASL	C ← $\boxed{15/7\quad 0}$ ← 0													N	V	.	.	.	.	Z	C
BCC	Branch if C=0			90										.	.	.	.	.	.	.	.
BCS	Branch if C=1			80										.	.	.	.	.	.	.	.
BEQ	Branch if Z=1			F0										.	.	.	.	.	.	.	.
BIT	A ∧M (Note 1)													M_7	M_6	.	.	.	.	Z	.
BMI	Branch if N=1			30										.	.	.	.	.	.	.	.
BNE	Branch if Z=0			D0										.	.	.	.	.	.	.	.
BPL	Branch if N=0			10										.	.	.	.	.	.	.	.
BRA	Branch Always			80										.	.	.	.	.	.	.	.
BRK	Break (Note 2)													.	.	.	.	0	I	.	.
BRL	Branch Long Always				82									.	.	.	.	.	.	.	.
BVC	Branch if V=0			50										.	.	.	.	.	.	.	.
BVS	Branch if V=1			70										.	.	.	.	.	.	.	.
CLC	0→C													.	.	.	.	.	.	.	0
CLD	0→D													.	.	.	.	0	.	.	.
CLI	0→1													.	.	.	.	.	0	.	.
CLV	0→V													.	0	.	.	.	.	.	.
CMP	A-M	DF	D9				D2	C7			C3	D3		N	.	.	.	.	.	Z	C
COP	Co-Processor										02			.	.	.	.	0	I	.	.
CPX	X-M													N	.	.	.	.	.	Z	C
CPY	Y-M													N	.	.	.	.	.	Z	C
DEC	Decrement													N	.	.	.	.	.	Z	.
DEX	X-1→X													N	.	.	.	.	.	Z	.
DEY	Y-1→Y													N	.	.	.	.	.	Z	.
EOR	A⊻M→A	5F	59				52	47			43	53		N	.	.	.	.	.	Z	.
INC	Increments													N	.	.	.	.	.	Z	.
INX	X+1→X													N	.	.	.	.	.	Z	.
INY	Y+1→Y													N	.	.	.	.	.	Z	.
JML	Jump Long to new Location					DC								.	.	.	.	.	.	.	.
JMP	Jump to New Location					6C			7C					.	.	.	.	.	.	.	.
JSL	Jump Long to Subroutine													.	.	.	.	.	.	.	.
JSR	Jump to Subroutine								FC					.	.	.	.	.	.	.	.
LDA	M→A	BF	B9				B2	A7			A3	B3		N	.	.	.	.	.	Z	.
LDX	M→X		BE											N	.	.	.	.	.	Z	.
LDY	M→Y													N	.	.	.	.	.	Z	.
LSR	0 → $\boxed{15/7\ \ 0}$ → C													0	.	.	.	.	.	Z	C
MVM	M→M Negative												54	.	.	.	.	.	.	.	.
MVP	M→M Positive												44	.	.	.	.	.	.	.	.
NOP	No Operation													.	.	.	.	.	.	.	.
ORA	A∨M→A	1F	19				12	07			03			N	.	.	.	.	.	Z	.
PEA	Mpc+1,Mpc+2→Ms-1,													.	.	.	.	.	.	.	.
	Ms S-2→S													.	.	.	.	.	.	.	.
PEI	M(d), M(d+1)→Ms-1,																				
	Ms S-2→S													.	.	.	.	.	.	.	.
PER	Mpc+rl, Mpc+rl+1→Ms-1																				
	Ms S-2→S																				

Mnemonic	Operation													Processor Status Code							
														7	6	5	4	3	2	1	0
	∧ AND ∨ OR	al,x	a,y	rl	rl	(a)	(d)	(d)	(a,x)	s	d,s	(d,s),y	a,x	N	V	M	X	D	I	Z	C
	⊻ Exclusive OR	13	14	15	16	17	18	19	20	21	22	23	24	N	V	I	B	D	I	Z	C
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	.
PLDL	S + 2→S, Ms-1, Ms→D									2B				N	.	.	.	.	.	Z	.
PLP	S + 1→S, MS→P									28				N	V	M	X	D	I	Z	C
PLX	S + 1→S, MS→X									FA				N	.	.	.	.	.	Z	.
PLY	S + 1→S,MS→Y									7A				N	.	.	.	.	.	Z	.
REP	M-∧ P→P													N	V	M	X	D	I	Z	C
ROL	15/7 0 ← C ←													N	.	.	.	.	.	Z	C
ROR	→ C → 15/7 0													N	.	.	.	.	.	Z	C
RTI	Return from Interrupt									40				N	V	M	X	D	I	Z	C
RTL	Return from Subroutine Long									6B				.	.	.	.	.	.	.	.
RTS	Return Subroutine									60				.	.	.	.	.	.	.	.
SBC	A-M-C→A	FF	F9				F2	E7			E3	F3		N	V	.	.	.	.	Z	C
SEC	1→C													.	.	.	.	.	.	.	1
SED	1→D													.	.	.	.	1	.	.	.
SEI	1→I													.	.	.	.	.	1	.	.
SEP	M∨P→P													N	V	M	X	D	I	Z	C
STA	A→M	9F	99				92	87			83	93		.	.	.	.	.	.	.	.
STP	STOP(1→PHI2)													.	.	.	.	.	.	.	.
STX	X→M													.	.	.	.	.	.	.	.
STY	Y→M													.	.	.	.	.	.	.	.
STZ	00→M													.	.	.	.	.	.	.	.
TAX	A→X													N	.	.	.	.	.	Z	.
TAY	A→Y													N	.	.	.	.	.	Z	.
TCD	C→D													N	.	.	.	.	.	Z	.
TCS	C→S													.	.	.	.	.	.	.	.
TDC	D→C													N	.	.	.	.	.	Z	.
TRB	A-∧ M→M													N	.	.	.	.	.	Z	.
TSB	A∨M→M													.	.	.	.	.	.	Z	.
TSC	S→C													N	.	.	.	.	.	Z	.
TSX	S→X													N	.	.	.	.	.	Z	.
TSA	X→A													N	.	.	.	.	.	Z	.
TXS	X→S													.	.	.	.	.	.	.	.
TXY	X→Y													N	.	.	.	.	.	Z	.
TYA	Y→A													N	.	.	.	.	.	Z	.
TYX	Y→X													N	.	.	.	.	.	Z	.
WAI	0→RDY													.	.	.	.	.	.	.	.
WDM	No Operation (Reserved)													.	.	.	.	.	.	.	.
XBA	B↔A													N	.	.	.	.	.	Z	.
XCE	C↔E													.	.	.	.	.	.	.	E

Table 5-5 Instruction Operation (13)

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
1.Immediate # LDY CPY CPX, LDX, ORA, AND EOR ADC, BIT, LDA, CMP, SEC, REP, SEP 14 OpCodes, 2 & 3 bytes 2 & 3 cycles		1	1	1	1	1	PBR,PC	OpCode	1
	(1)	2	1	1	0	1	PBR,PC+1	IDL	1
		2a	1	1	0	1	PBR,PC+2	IDH	1
2a. Absolute a BIT,STY,STZ,LDY,CPY,CPX,STX, LDX,ORA,AND,EOR,ABC,STA,LDA, CMP,SBC 18 OpCodes, 3 bytes, 4 & 5 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	AAL	1
		3	1	1	0	1	PBR,PC+2	AAH	1
		4	1	1	1	0	DBR,AA	Data Low	1/0
	(1)	4a	1	1	1	0	DBR,AA+1	Data High	1/0
2b. Absolute (R-M-W) a ASL,ROL,LSR,ROR,DEC,INC,TSB,TRB 6 OpCodes, 3 bytes, 6 & 8 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	AAL	1
		3	1	1	0	1	PBR,PC+2	AAH	1
		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
2c. Absolute (JUMP) a JMP (4C) 1 OpCode, 3 bytes, 3 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	New PCL	1
		3	1	1	0	1	PBR,PC+2	New PCH	1
		1	1	1	1	1	PBR,New PC	New OpCode	1
2d. Absolute (JUMP to subroutine) a JSR 1 OpCode, 3 bytes, 6 cycles (different order from N6502)		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	New PCL	1
		3	1	1	0	1	PBR,PC+2	New PCH	1
		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,New PC	New OpCode	1
3a. Absolute Long al ORA,AND,EOR,ABC,STA,LDA,CMP,SBC 8 OpCodes, 4 bytes, 5 & 6 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	AAL	1
		3	1	1	0	1	PBR,PC+2	AAH	1
		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
3b. Absolute Long (JUMP) al JMP 1 OpCode, 4 bytes, 4 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	New PCL	1
		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
3c. Absolute Long (JUMP to Subroutine Long) al JSL 1 OpCode, 4 bytes, 7 cycles		1	1	1	1	1	PBR,PC	OpCode	1
		2	1	1	0	1	PBR,PC+1	New PCL	1
		3	1	1	0	1	PBR,PC+2	New PCH	1
		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	New OpCode	1

Address Mode	Note	Cycle	VPB	MLB	VDA	VPA	Address Bus	Data Bus	RW
					(14)	(14)	(15)		B
4a. Direct d		1	1	1	1	1	PBR,PC	OpCode	1
BIT,STZ,STY,LDY,CPY,CPX,STX,LDX,ORA,		2	1	1	0	1	PBR,PC+1	DO	1
AND,EOR,ADC,STA,LDA,CMP,SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
18 OpCodes, 2 bytes, 3, 4 & 5 cycles		3	1	1	1	0	0,D+DO	Data Low	1/0
	(1)	3a	1	1	1	0	0,D+DO+1	Data High	1/0
4b. Direct (R-M-W) d		1	1	1	1	1	PBR, PC	OpCode	1
ASL,ROL,LSR,ROR,DEC,INC,TSB,TRB		2	1	1	0	1	PBR,PC+1	DO	1
6 OpCodes, 2 bytes, 5,6,7 and 8 cycles	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
		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
5. Accumulator A		1	1	1	1	1	PBR,PC	OpCode	1
ASL,INC,ROL,DEC,LSR,ROR		2	1	1	0	0	PBR,PC+1	IO	1
6 OpCodes, 1 byte, 2 cycles									
6a. Implied I		1	1	1	1	1	PBR,PC	OpCode	1
DEY,INY,INX,DEX,NOP,XCE,TYA,TAY,TXA		2	1	1	0	0	PBR,PC+1	IO	1
TXS,TAX,TSX,TCS,TSC,TCD,TDC,TXY,TYX									
CLC,SEC,CLI,SEI,CLV,CLD,SED									
25 OpCodes, 1 byte, 2 cycles									
6b. 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
6c. Wait for Interrupt							RDY		
WAI		1	1	1	1	1	1 PBR,PC	OpCode	1
1 OpCode	(9)	2	1	1	0	0	0 PBR,PC+1	IO	1
1 byte IRQB, NMIB		3	1	1	0	0	0 PBR,PC+1	IO	1
3 cycles,		1	1	1	1	1	1 PBR,PC+1	IRQ(BRK)	1
6d. Stop the Clock							RDY		
STP		1	1	1	1	1	1 PBR,PC	OpCode	1
1 OpCode, 1 byte, 3 cycles		2	1	1	0	0	1 PBR,PC+1	IO	1
RESB=1		3	1	1	0	0	1 PBR,PC+1	IO	1
RESB=0		1c	1	1	0	0	1 PBR,PC+1	RES (BRK)	1
RESB=0		1b	1	1	0	0	1 PBR,PC+1	RES (BRK)	1
RESB=1		1a	1	1	0	0	1 PBR,PC+1	RES (BRK)	1
(See 21a. Stack Hardware Interrupt)		1	1	1	1	1	1 PBR,PC+1	BEGIN	1
7. Direct Indirect		1	1	1	1	1	PBR,PC	OpCode	1
Indexed-(d),y		2	1	1	0	1	PBR,PC+1	DO	1
ORA,AND,EOR,ADC,STA,LDA,CMP,SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes		3	1	1	1	0	0,D+DO	AAL	1
5,6,7 and 8 cycles		4	1	1	1	0	0,D+DO+1	AAH	1
	(4)	4a	1	1	0	0	DBR,AAH	IO	1
							AAL+YL		
		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
8. Direct Indirect		1	1	1	1	1	PBR,PC	OpCode	1
Indexed Long [d],y		2	1	1	0	1	PBR,PC+1	DO	1
ORA,AND,EOR,ADC,STA,LDA,CMP,SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes,		3	1	1	1	0	0,D+DO	AAL	1
6,7 and 8 cycles		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

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
9. Direct Indexed		1	1	1	1	1	PBR,PC	OpCode	1
Indirect (d,x)		2	1	1	0	1	PBR,PC+1	DO	1
ORA,AND,EOR,ADC,STA,LDA,CMP,SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes, 2 bytes,		3	1	1	0	0	PBR,PC+1	IO	1
6,7 and 8 cycles		4	1	1	1	0	0,D+DO+X	AAL	1
		5	1	1	1	0	0,D+DO+X+1	AAH	1
							DBR,AA		
		6	1	1	1	0	DBR,AA+1	Data Low	1/0
	(1)	6a	1	1	1	0		Data High	1/0
10a. Direct,X d,x		1	1	1	1	1	PBR,PC	OpCode	1
BIT,STZ,STY,LDY,ORA,AND,EOR,ADC		2	1	1	0	1	PBR,PC+1	DO	1
STA,LDA,CMP,SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
11 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
10b. Direct, X (R-M-W) d,x		1	1	1	1	1	PBR,PC	OpCode	1
ASL,ROL,LSR,ROR,DEC,INC		2	1	1	0	1	PBR,PC+1	DO	1
6 OpCodes, 2 bytes	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
6,7,8 and 9 cycles		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
11. Direct, Y d,y		1	1	1	1	1	PBR,PC	OpCode	1
STX,LDX		2	1	1	0	1	PBR,PC+1	DO	1
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
12a Absolute, X a,x		1	1	1	1	1	PBR,PC	OpCode	1
BIT,LDY,STZ,ORA,AND,EOR,ADC		2	1	1	0	1	PBR,PC+1	AAL	1
STA,LDA,CMP,SBC		3	1	1	0	1	PBR,PC+2	AAH	1
11 OpCodes	(4)	3a	1	1	0	0	DBR,AAH,	IO	1
3 bytes, 4,5 and 6 cycles							AAL+XL		
		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
12b Absolute, X(R-M-W) a,x		1	1	1	1	1	PBR,PC	OpCode	1
ASL,ROL,LSR,ROR,DEC,INC		2	1	1	0	1	PBR,PC+1	AAL	1
6 OpCodes, 3 bytes		3	1	1	0	1	PBR,PC+2	AAH	1
7 and 9 cycles		4	1	1	0	0	DBR,AAH	IO	1
	(1)	5	1	0	1	0	AAL+XL	Data Low	1
		5a	1	0	1	0	DBR,AA+X	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+1	Data Low	0
							DBR,AA+X		

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
13. Absolute Long,X al,x		1	1	1	1	1	PBR,PC	OpCode	1
ORA,AND,EOR,ADC,STA,LDA,CMP,SB		2	1	1	0	1	PBR,PC+1	AAL	1
C		3	1	1	0	1	PBR,PC+2	AAH	1
8 OpCodes, 4 bytes		4	1	1	0	1	PBR,PC+3	AAB	1
5 and 6 cycles		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
14. Absolute, Y a,y		1	1	1	1	1	PBR,PC	OpCode	1
LDX,ORA,AND,EOR,ADC,STA,LDA,		2	1	1	0	1	PBR,PC+1	AAL	1
CMP,SBC		3	1	1	0	1	PBR,PC+2	AAH	1
9 OpCodes, 3 bytes	(4)	3a	1	1	0	0	DBR,AAH, AAL+YL	IO	1
4,5 and 6 cycles									
		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
15. Relative r		1	1	1	1	1	PBR,PC	OpCode	1
BPL,BMI,BVC,BVS,BCC		2	1	1	0	1	PBR,PC+1	Offset	1
BCS,BNE,BEQ,BRA	(5)	2a	1	1	0	0	PBR,PC+1	IO	1
9 OpCodes, 2 bytes	(6)	2b	1	1	0	0	PBR,PC+1	IO	1
2,3 and 4 cycles		1	1	1	1	1	PBR,PC+ Offset	OpCode	1
16. Relative Long rl		1	1	1	1	1	PBR,PC	OpCode	1
BRL		2	1	1	0	1	PBR,PC+1	Offset L	1
1 OpCode, 3 bytes		3	1	1	0	1	PBR,PC+2	Offset H	1
4 cycles		4	1	1	0	0	PBR,PC+2	IO	1
		1	1	1	1	1	PBR,PC+ Offset	OpCode	1
17a. 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		3	1	1	0	1	PBR,PC+2	AAH	1
5 cycles		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
17b. Absolute Indirect (a)		1	1	1	1	1	PBR,PC	OpCode	1
JML		2	1	1	0	1	PBR,PC+1	AAL	1
1 OpCodes, 3 bytes		3	1	1	0	1	PBR,PC+2	AAH	1
6 cycles		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
18. Direct Indirect (d)		1	1	1	1	1	PBR,PC	OpCode	1
ORA,AND,EOR,ADC,STA,LDA,CMP,SB		2	1	1	0	1	PBR,PC+1	DO	1
C	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes		3	1	1	1	0	0,D+DO	AAL	1
2 bytes		4	1	1	1	0	0,D+DO+1	AAH	1
5,6 and 7 cycles		5	1	1	1	0	DBR,AA	Data Low	1/0
	(1)	5a	1	1	1	0	DBR,AA+1	Data High	1/0

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RWB
19. Direct Indirect Long [d]		1	1	1	1	1	PBR,PC	OpCode	1
ORA,AND,EOR,ADC		2	1	1	0	1	PBR,PC+1	DO	1
STA,LDA,CMP, SBC	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
8 OpCodes		3	1	1	1	0	O,D+DO	AAL	1
2 bytes		4	1	1	1	0	O,D+DD+1	AAH	1
6,7 and 8 cycles		5	1	1	1	0	O,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
20a. 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	1	1	0	1	PBR-PC+2	AAH	1
3 bytes		4	1	1	0	0	PBR,PC+2	IO	1
6 cycles		5	1	1	0	1	PBR,AA+X	New PCL	1
		6	1	1	0	1	PBR,AA+X+1	New PCH	1
							PBR,NEW PC		
		1	1	1	1	1		OpCode	1
20b. Absolute Indexed Indirect		1	1	1	1	1	PBR,PC	OpCode	1
(a,x)		2	1	1	0	1	PBR,PC+1	AAL	1
JSR		3	1	1	1	0	O,S	PCH	0
1 OpCode		4	1	1	1	0	O,S-1	PCL	0
3 bytes		5	1	1	0	1	PBR,PC+2	AAH	1
8 cycles		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
							PBR,NEW PC		
		1	1	1	1	1		New OpCode	1
21a. Stack (Hardware		1	1	1	1	1	PBR,PC	IO	1
Interrupts) s	(3)	2	1	1	0	0	PBR,PC	IO	1
IRQ,NMI,ABORT,RES	(7)	3	1	1	1	0	O,S	PBR	0
4 hardware interrupts	(10)	4	1	1	1	0	O,S-1	PCH	0
0 bytes	(10)	5	1	1	1	0	O,S-2	PCL	0
7 and 8 cycles	(10)	6	1	1	1	0	O,S-3	P	0
(11)		7	0	1	1	0	O,VA	AAVL	1
		8	0	1	1	0	O,VA+1	AAVH	1
		1	1	1	1	1	O,AAV	New OpCode	1
21b. Stack (Software		1	1	1	1	1	PBR,PC	OpCode	1
Interrupts) s	(3)	2	1	1	0	1	PBR,PC+1	Signature	1
BRK,COP	(7)	3	1	1	1	0	O,S	PBR	0
2 OpCodes		4	1	1	1	0	O,S-1	PCH	0
2 bytes		5	1	1	1	0	O,S-2	PCL	0
7 and 8 cycles		6	1	1	1	0	O,S-3 (16)	P	0
		7	0	1	1	0	O,VA	AAVL	1
		8	0	1	1	0	O,VA+1	AAVH	1
		1	1	1	1	1	O,AAV	New OpCode	1
21c. Stack (Return from		1	1	1	1	1	PBR,PC	OpCode	1
Interrupt) s		2	1	1	0	0	PBR,PC+1	IO	1
RTI	(3)	3	1	1	0	0	PBR,PC+1	IO	1
1 Op Code		4	1	1	1	0	O,S+1	P	1
1 byte		5	1	1	1	0	O,S+2	New PCL	1
6 and 7 cycles		6	1	1	1	0	O,S+3	New PCH	1
(different order from	(7)	7	1	1	1	0	O,S+4	PBR	1
N6502)		1	1	1	1	1	PBR,New PC	New OpCode	1

Address Mode	Note	Cycle	VPB	MLB	VDA	VPA	Address Bus	Data Bus	RWB
					(14)	(14)	(15)		
21d. Stack (Return from		1	1	1	1	1	PBR,PC	OpCode	1
Subroutine) s		2	1	1	0	0	PBR,PC+1	IO	1
RTS		3	1	1	0	0	PBR,PC+1	IO	1
1 OpCode		4	1	1	1	0	O,S+1	PCL	1
1 byte		5	1	1	1	0	O,S+2	PCH	1
6 cycles		6	1	1	0	0	O,S+2	IO	1
		1	1	1	1	1	PBR,PC	OpCode	1
21e. Stack (Return from		1	1	1	1	1	PBR,PC	OpCode	1
Subroutine Long) s		2	1	1	0	0	PBR,PC+1	IO	1
RTL		3	1	1	0	0	PBR,PC+1	IO	1
1 Op Code		4	1	1	1	0	O,S+1	New PCL	1
1 byte		5	1	1	1	0	O,S+2	New PCH	1
6 cycles		6	1	1	1	0	O,S+3	New PBR	1
		1	1	1	1	1	NEW PBR,PC	New OpCode	1
21f. Stack (Push) s		1	1	1	1	1	PBR,PC	OpCode	1
PHP,PHA,PHY,PHX		2	1	1	0	0	PBR,PC+1	IO	1
PHD,PHK,PHB (1)	(12)	3a	1	1	1	0	O,S	REG High	0
7 Op Codes		3	1	1	1	0	O,S-1	REG Low	0
1 byte									
3 and 4 cycles									
21g. Stack (Pull) s		1	1	1	1	1	PBR,PC	OpCode	1
PLP,PLA,PLY,PLX,PLD,PLB		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		4	1	1	1	0	O,S+1	REG Low	1
1 byte	(1)	4a	1	1	1	0	O,S+2	REG High	1
4 and 5 cycles									
21h. Stack (Push Effective		1	1	1	1	1	PBR,PC	OpCode	1
Indirect Address) s		2	1	1	0	1	PBR,PC+1	DO	1
PEI	(2)	2a	1	1	0	0	PBR,PC+1	IO	1
1 Op Code		3	1	1	1	0	O,D+DO	AAL	1
2 bytes		4	1	1	1	0	O,D+DO+1	AAH	1
6 and 7 cycles		5	1	1	1	0	O,S	AAH	0
		6	1	1	1	0	O,S-1	AAL	0
21i.Stack (Push Effective		1	1	1	1	1	PBR,PC	OpCode	1
Absolute Address) s		2	1	1	0	1	PBR,PC+1	AAL	1
PEA		3	1	1	0	1	PBR,PC+2	AAH	1
1 Op Code		4	1	1	1	0	O,S	AAH	0
3 bytes		5	1	1	1	0	O,S-1	AAL	0
5 cycles									
21j. Stack (Push Effective		1	1	1	1	1	PBR,PC	OpCode	1
Program Counter Relative		2	1	1	0	1	PBR,PC+1	Offset Low	1
Address) s		3	1	1	0	1	PBR,PC+2	Offset High	1
PER		4	1	1	0	0	PBR,PC+2	IO	1
1 Op Code		5	1	1	1	0	O,S	PC+3+ Offset H	0
3 bytes									
6 cycles		6	1	1	1	0	O,S-1	PC+3+ Offset L	0
22. Stack Relative d,s		1	1	1	1	1	PBR,PC	OpCode	1
ORA,AND,EOR,ADC		2	1	1	0	1	PBR,PC+1	SO	1
STA,LDA,CMP,SBC		3	1	1	0	0	PBR,PC+1	IO	1
8 Op Codes		4	1	1	1	0	O,S+SO	Data Low	1/0
2 bytes	(1)	4a	1	1	1	0	O,S+SO+1	Data High	1/0
4 and 5 cycles									

Address Mode	Note	Cycle	VPB	MLB	VDA (14)	VPA (14)	Address Bus (15)	Data Bus	RW B
23. Stack Relative Indirect		1	1	1	1	1	PBR,PC	OpCode	1
Indexed (d,s),y		2	1	1	0	1	PBR,PC+1	SO	1
ORA,AND,EOR,AD ,STA		3	1	1	0	0	PBR,PC+1	IO	1
LDA,CMP,SBC		4	1	1	1	0	O,S+SO	AAL	1
8 Op Codes		5	1	1	1	0	O,S+SO+1	AAH	1
2 bytes		6	1	1	0	0	O,S+SO+1	IO	1
7 and 8 cycles		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
24a. Block Move Positive		1	1	1	1	1	PBR,PC	OpCode	1
(forward) xyc		2	1	1	0	1	PBR,PC+1	DBA	1
(MVP)		3	1	1	0	1	PBR,PC+2	SBA	1
1 Op Code N-2		4	1	1	1	0	SBA,X	SRC Data	1
3 bytes Byte		5	1	1	1	0	DBA,Y	DEST Data	0
7 cycles C=2		6	1	1	0	0	DBA,Y	IO	1
		7	1	1	0	0	DBA,Y	IO	1
x=Source Address									
y=Destination									
c=# of bytes to move-1		1	1	1	1	1	PBR,PC	OpCode	1
x,y Decrement		2	1	1	0	1	PBR,PC+1	DBA	1
MVP is used when the N-1		3	1	1	0	1	PBR,PC+2	SBA	1
dest. start address Byte		4	1	1	1	0	SBA,X-1	SRC Data	1
is higher (more C=1		5	1	1	1	0	DBA,Y-1	DEST Data	0
positive) than the source		6	1	1	0	0	DBA,Y-1	IO	1
start address.		7	1	1	0	0	DBA,Y-1	IO	1
		1	1	1	1	1	PBR,PC	OP Code	1
FFFFFF		2	1	1	0	1	PBR,PC+1	DBA	1
▲ Destination Start N Byte		3	1	1	0	1	PBR,PC+2	SBA	1
Source Start Last		4	1	1	1	0	SBA,X-2	SRC Data	1
Destination End C=0		5	1	1	1	0	DBA,Y-2	DEST Data	0
Source End		6	1	1	0	0	DBA,Y-2	IO	1
000000		7	1	1	0	0	DBA,Y-2	IO	1
		1	1	1	1	1	PBR,PC+3	New OpCode	1
24b. Block Move Negative		1	1	1	1	1	PBR,PC	OpCode	1
(backward) xyc		2	1	1	0	1	PBR,PC+1	DBA	1
MVN		3	1	1	0	1	PBR,PC+2	SBA	1
1 Op Code N-2		4	1	1	1	0	SBA,X	SRC Data	1
3 bytes Byte		5	1	1	1	0	DBA,Y	DEST Data	0
7 cycles C=2		6	1	1	0	0	DBA,Y	IO	1
		7	1	1	0	0	DBA,Y	IO	1
x=Source Address									
y=Destination		1	1	1	1	1	PBR,PC	OpCode	1
c=# of bytes to move-1		2	1	1	0	1	PBR,PC+1	DBA	1
x,y Increment		3	1	1	0	1	PBR,PC+2	SBA	1
MVN is used when the N-1		4	1	1	1	0	SBA,X+1	SRC Data	1
dest. start address Byte		5	1	1	1	0	DBA,Y+1	DEST Data	0
is lower (more C=1		6	1	1	0	0	DBA,Y+1	IO	1
negative) than the source		7	1	1	0	0	DBA,Y+1	IO	1
start address.									
		1	1	1	1	1	BR,PC	OpCode	1
FFFFFF		2	1	1	0	1	PBR,PC+1	DBA	1
Source End		3	1	1	0	1	PBR,PC+2	SBA	1
N Byte		4	1	1	1	0	SBA,X+2	SRC Data	1
Destination End Last		5	1	1	1	0	DBA,Y+2	DEST Data	0
Source Start C=0		6	1	1	0	0	DBA,Y+2	IO	1
Destination Start		7	1	1	0	0	DBA,Y+2	IO	1
▼		1	1	1	1	1	PBR,PC+3	New OpCode	1
000000									

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.

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 Register
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

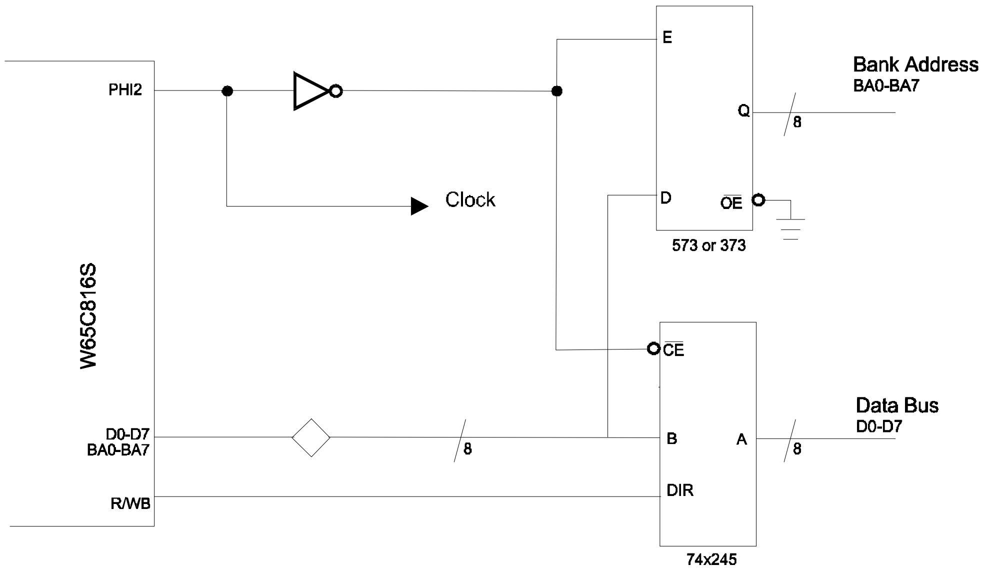


Figure 5-1 Bank Address Latching Circuit

SECTION 6

RECOMMENDED W65C816S ASSEMBLER SYNTAX STANDARDS

6.1 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.

6.2 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.

6.3 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.

6.3.1 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.

6.3.2 The Operation Code Field --The operation code shall consist of a three character sequence (mnemonic) from Table 6-1. It shall start no sooner than column 2 of the line, or one space after the label if a label is coded.

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

6.3.2.2 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.

6.3.2.3 In addition to the mnemonics shown in Table 6-1, an assembler may provide the alternate mnemonics show in Table 7-1.

Table 6-1 Alternate Mnemonics

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

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

6.3.3 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.

6.3.3.1 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.

6.3.3.2 Table 7-3-2 shows the operand formats which 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 6-2 Address Mode Formats

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

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

6.3.3.3 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.

6.3.3.4 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 6-3 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

6.3.3.5 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.

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

6.3.3.7 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.

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

SECTION 7

CAVEATS

Table 8-1 W65C816S Compatibility Issues(following 2 pages)

Compatability Issue	W65C816/802	W65C02	NMOS 6502	W65C816S	W65C02S
S (Stack)	Always page 1 (E=1),8 bits, 16 bits when E=0	Always Page 1, 8 bits	Always Page 1, 8 bits	Always page 1 (E=1),8 bits, 16 bits when E=0	Always page 1, 8 bits
X (X Index Reg)	Indexed page zero always in page 0 (E=1), Cross Page (E=0)	Always Page 0	Always Page 0	Indexed page zero always in page 0 (E=1) Cross page (E=0)	Always page 0
Y (Y Index Reg)	Indexed page zero always in page 0 (E=1), Cross page (E=0)	Always Page 0	Always Page 0	Indexed page zero always in page 0 (E=1), Cross page (E=0)	Always Page 0
A (Accumulator)	8 bits (M=1), 16 bits (M=0)	8 bits	8 bits	8 bits (M=1), 16 BITS (M=0)	8 bits
(Flag Reg)	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 invalid in decimal mode. D=unknown after reset. D not modified after interrupt	N,V, and Z flags vaild in decimal mode. D=0 after reset/interrupt	N,V, and Z flags vaild in decimal mode. D=0 after reset/interrupt
Timing A. ABS,X,ASL,LSR, ROL with no Page Crossing	7 cycles	6 cycles	7 cycles	7 cycles	6 cycles
B. Jump Indirect Operand=XXFF	5 cycles	6 cycles	5 cycles and invalid page crossing	5 cycles	6 cycles
C. Branch Across Page	4 cycles (E=1)	4 cycles	4 cycles	4 cycles (E=1)	4 cycles
D. Decimal Mode	No add. cycle	Add 1 cycle	No add. cycle	No add. cycle	Add 1 cycle
BRK Vector	00FFFE,F(E=1) BRK bit=0 on stack if IRQ-, NMI-, ABORT-, 00FFE6,7 (E=0), X=X on stack always	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-, NMI-, ABORT-, 00FFE6,7 (E=0), X=X on stack always	FFFE,F BRK bit=0 on stack if IRQ, NMI
Interrupt or Break Bank Address	PBR not pushed (E=1), RTI, PBR, not pulled (E=1), PBR pushed (E=0) Rti, PBR pulled (E=0)	Not available	Not available	PBR not pushed (E=1), RTI, PBR, not pulled (E=1), PBR pushed (E=0) Rti, PBR pulled (E=0)	Not available
Memory Lock (ML)	ML-=0 during Read Modify and Write cycles	ML-=0 during Modify and Write cycles	Not available	ML-=0 during Read Modify and Write cycles	ML-=0 during 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 invalid address	Extra read of invalid address	Extra read of last instruction fetch
RDY Pulled during Write cycle	Ignored (E=1) for W65C802 only Processor Stops (E=0)	Processor stops	Ignored	Ignored (E=1) for W65C802 only Processor Stops (E=0)	Processor stops

Stack Addressing

When in the Native mode, the Stack may use memory locations 000000 to 00FFFFF. 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

7.2 Direct Addressing

7.2.1 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.

7.2.2 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.

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

7.3 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.

7.4 ABORTB Input

7.4.1 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:

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

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

7.4.1.3 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).

7.4.2 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.

7.5 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.

7.6 Apple II, IIe, IIc and II+ Disk Systems

VDA and VPA should not be used to qualify addresses during disk operation on Apple systems. Consult your Apple representative for hardware/software configurations.

7.7 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.

7.8 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.

7.9 All OpCodes Function in All Modes of Operation

7.9.1 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 following is a list of instructions and addressing modes which are primarily intended for W65C816S use: JSL,RTL,JMP al;JML,al,

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

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

7.9.3.1 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).

7.9.3.2 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.

7.10 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.

7.11 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.

7.12 How Hardware Interrupts, BRK, and COP Instructions Affect the Program Bank and the Data Bank Registers

7.12.1 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 is automatically saved on Stack.

7.12.2 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.

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

7.13 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.

7.14 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.

7.15 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.

7.16 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.

7.17 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.

7.18 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.

7.19 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).

7.20 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

7.21 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,TDC

7.22 Stack Transfers

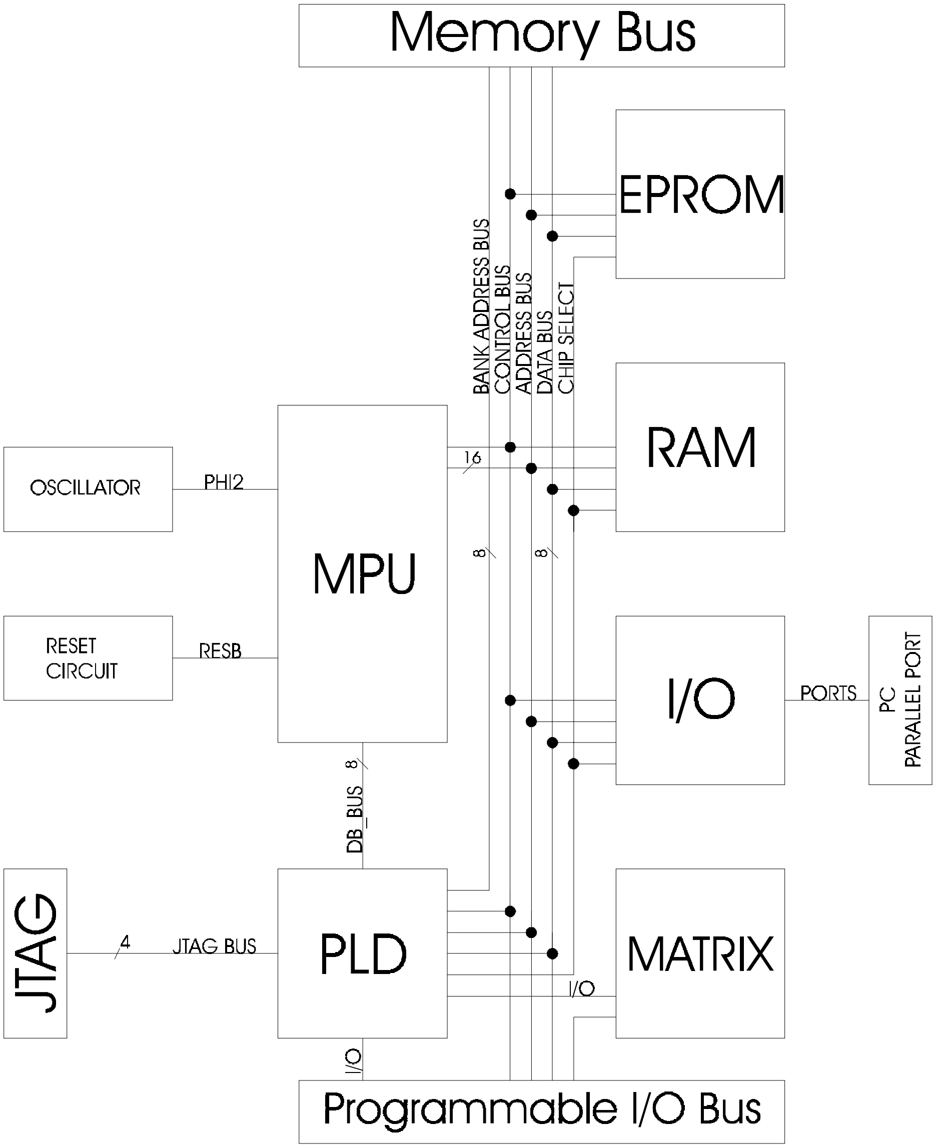
When in the Emulation mode, a 01 is forced into SH. In this case, the B Accumulator will not be loaded into SH during a TCS instruction. When in the Native mode, the B Accumulator is transferred to SH. Note that in both the Emulation and Native modes, the full 16 bits of the Stack Register are transferred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

7.23 BRK Instruction

The BRK instruction for both 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.

Section 8

W65C816DB Developer Board



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

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 XC9572 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 XC9572 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.

Memory map:

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

SECTION 9

HARD CORE MODEL

9.1 W65C816 Core Information

1. The W65C816C uses the same instruction set as the W65C816S.

2. The only functional difference between the W65C816S and W65C816C is the RDY pin. The W65C816S RDY pin is bidirectional utilizing an active pullup. The W65C816C RDY function is split into 2 pins, RDYINand WAITN. The WAITN output goes low when a WAI instruction is executed.
3. The W65C816C will be a smaller die since the I/O buffers have been removed.

4. The outputs are the N-channel and P-channel output transistors drivers.

5. The following inputs, if not used, must be held in the high state: RDY input, IRQB, MIB, BE and ABORTB.

6. The timing of the W65C816C is the same as the W65C816S.

SECTION 10

SOFT CORE RTL MODEL

Under Construction

SECTION 11

FIRM CORE MODEL

UNDER CONSTRUCTION

SECTION 5

ORDERING INFORMATION

W65C816S8PL-14	
Description W65C = standard product	W65C
Product Identification Number	816S
Foundry Process Blank = 1.2u 8 = .8u	8

Package
P = Plastic Dual-In-Line, 40 pins
PL = Plastic Leaded Chip Carrier, 44 pins
Q = Quad Flat Pack, 44 pins
	PL

Temperature/Processing Blank = 0°C to + 70°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
e-mail: information@wdesignc.com
WEB: http://www.wdesignc.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.