W65C832

INFORMATION, SPECIFICATION AND DATA SHEET

PRELIMINARY

```
                        A
                        B
                  I     O                       R                 P
            M     R     R     R     V     V     E     V     M     H
            L     Q     T     D     P     S     S     D     /     I     B
            -     -     -     Y     -     S     -     A     X     2     E

          ________________________________________________________________
         |                                                                |
         |  6     5     4     3     2     1    44    43    42    41    40 |
    NMI- |  7                                                          39 | E8/E16
     VPA |  8                                                          38 | R/W-
     VDD |  9                                                          37 | VDD
      A0 | 10                                                          36 | D0/A16
      A1 | 11                                                          35 | D1/A17
     VSS | 12                        W65C832                           34 | D2/A18
      A2 | 13                                                          33 | D3/A19
      A3 | 14                                                          32 | D4/A20
      A4 | 15                                                          31 | D5/A21
      A5 | 16                                                          30 | D6/A22
      A6 | 17                                                          29 | D7/A23
         | 18    19    20    21    22    23    24    25    26    27    28 |
         |________________________________________________________________|
           A     A     A     A     A     V     V     A     A     A     A
           7     8     9     1     1     S     S     1     1     1     1
                             0     1     S     S     2     3     4     5
```

MARCH 1991

TABLE OF CONTENTS

TABLE OF CONTENTS

FIGURES

TABLE OF CONTENTS

TABLES

INTRODUCTION

The WDC W65C832 is a CMOS 32-bit microprocessor featuring total software compatibility with their 8-bit NMOS and 8-bit and 16-bit CMOS 6500-series predecessors. The W65C832 is pin-to-pin compatible with 16-bit devices currently available. 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 or 16-bit "emulation" mode, or in the native mode, thus allowing existing systems to use the expanded features.

As shown in the processor programming model, the Accumulator, ALU, X and Y Index registers have been extended to 32 bits. A 16-bit Program Counter, Stack Pointer and 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 for program space and 32-bit 4GByte data space for ASIC use although only 24 bits of address are available in the standard pin-out.

Four 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.

KEY FEATURES OF THE W65C832

* Advanced CMOS design for low power power consumption and increased noise immunity
* Single 1.2-5.25V power supply, as specified
* Emulation mode allows complete hardware and software compatibility with W65C816 designs
* 24-bit address bus allows access to 16 MBytes of memory space
* Full 32-bit ALU, Accumulator, 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
* 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, plus 11 new addressing modes with 91 instructions using 255 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

SECTION 1

W65C832 FUNCTION DESCRIPTION

The W65C832 provides the design engineer with upward mobility and software compatibility in applications where a 32-bit system configuration is desired. The W65C832's 32-bit hardware configuration, coupled with current software allows a wide selection of system applications. In the Emulation mode, the W65C832 offers many advantages, including full software compatibility with 6502, W65C02 or W65C816 coding. In addition, the W65C832's powerful instruction set and addressing modes make it an excellent choice for new 32-bit designs.

Internal organization of the W65C832 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 W65C832 has a 32-bit internal architecture with an 8-bit external data bus.

1.1 Instruction Register and Decode

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 32-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 Accumulator

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 the Accumulator can be 8-, 16- or 32-bits wide.

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 W65C832. 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, both Index Registers are 32 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 bytes if 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 (E8 and E16) select and the Break (B) flags are accessible only through the Processor Status Register. The Emulation (E8) mode select flag is selected by the Exchange Carry and Emulation Bits (XCE) instruction. The XFE instruction exchanges the Emulation (E8 and E16) mode select flags with the Overflow and Carry Flags. Table 1, Emulation and Register Width Control, illustrates the features of the Native and Emulation modes. The M and X flags are always equal to one in the 8-bit 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 value and presented on the Data/Address lines during the first half of a program memory read cycle. The Program Bank Register is initialized to zero during Reset. The PHK instruction pushes the PBR register onto the Stack.

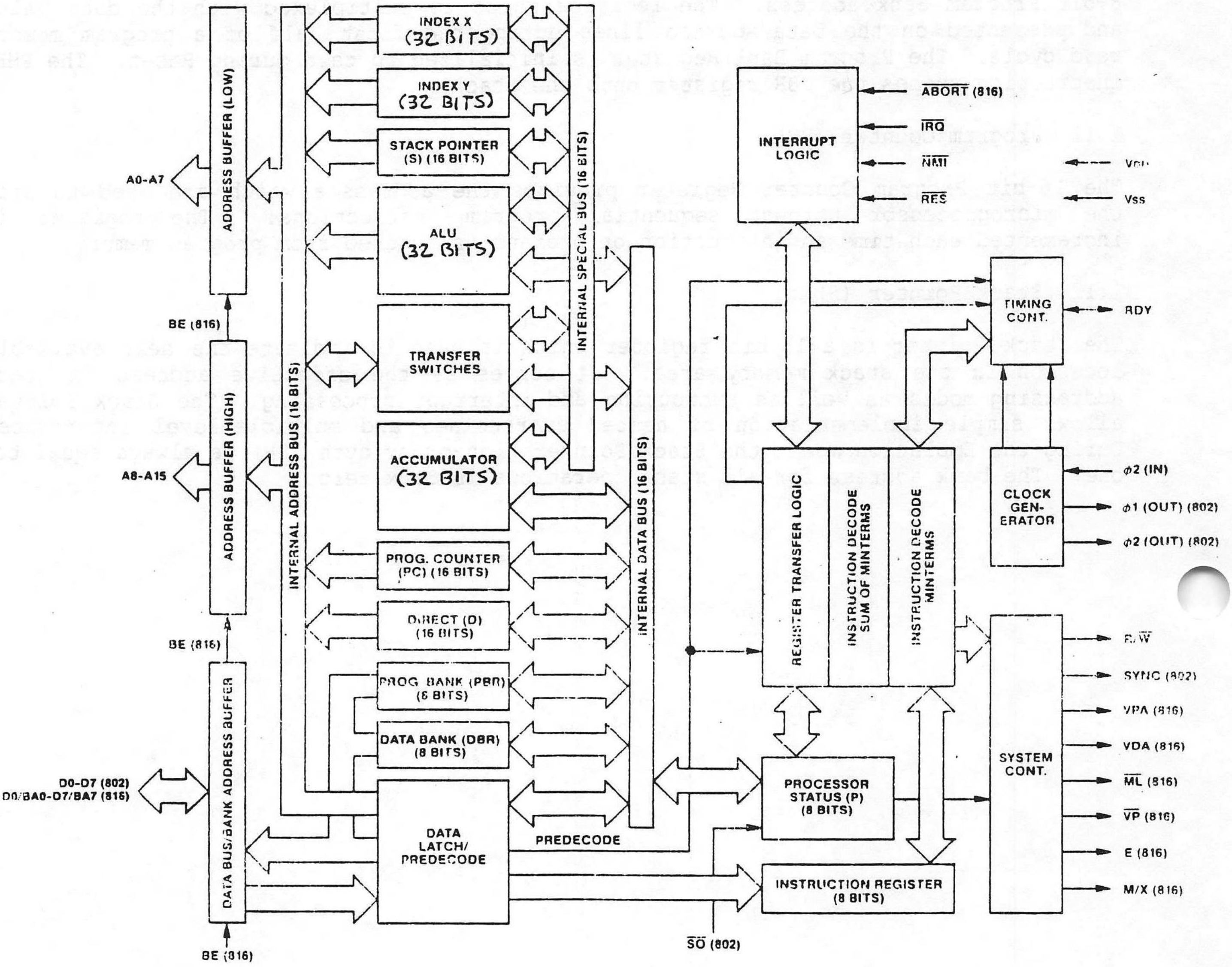
1.11 Program Counter (PC)

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.

Figure 1-1 W65C832 Internal Architecture Simplified Block Diagram



8 Bits	8 Bits	8 Bits	8 Bits	

Index and Data Registers

X Register				X
Y Register				Y
ACCUMULATOR				A

Address Registers

0	Program Bank Register (PBR)	Program Counter		PC
0		Direct Register		D
0		Stack Pointer		S
0	Data Bank Register	0		DBR

Status Register

			Status	P

Figure 1-2 W65C832 Native Mode Programming Model

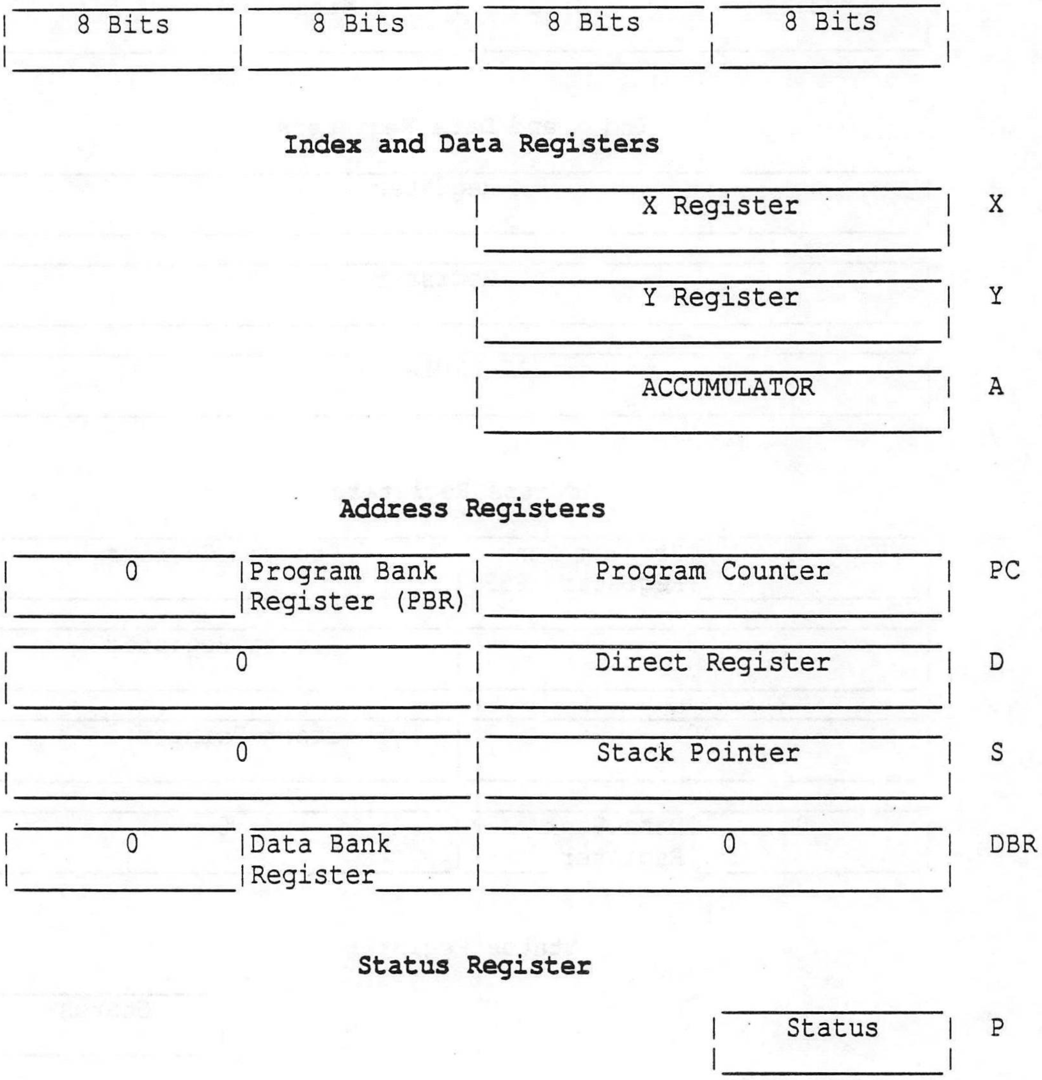


Figure 1-3 W65C816 16-bit Emulation Programming Model

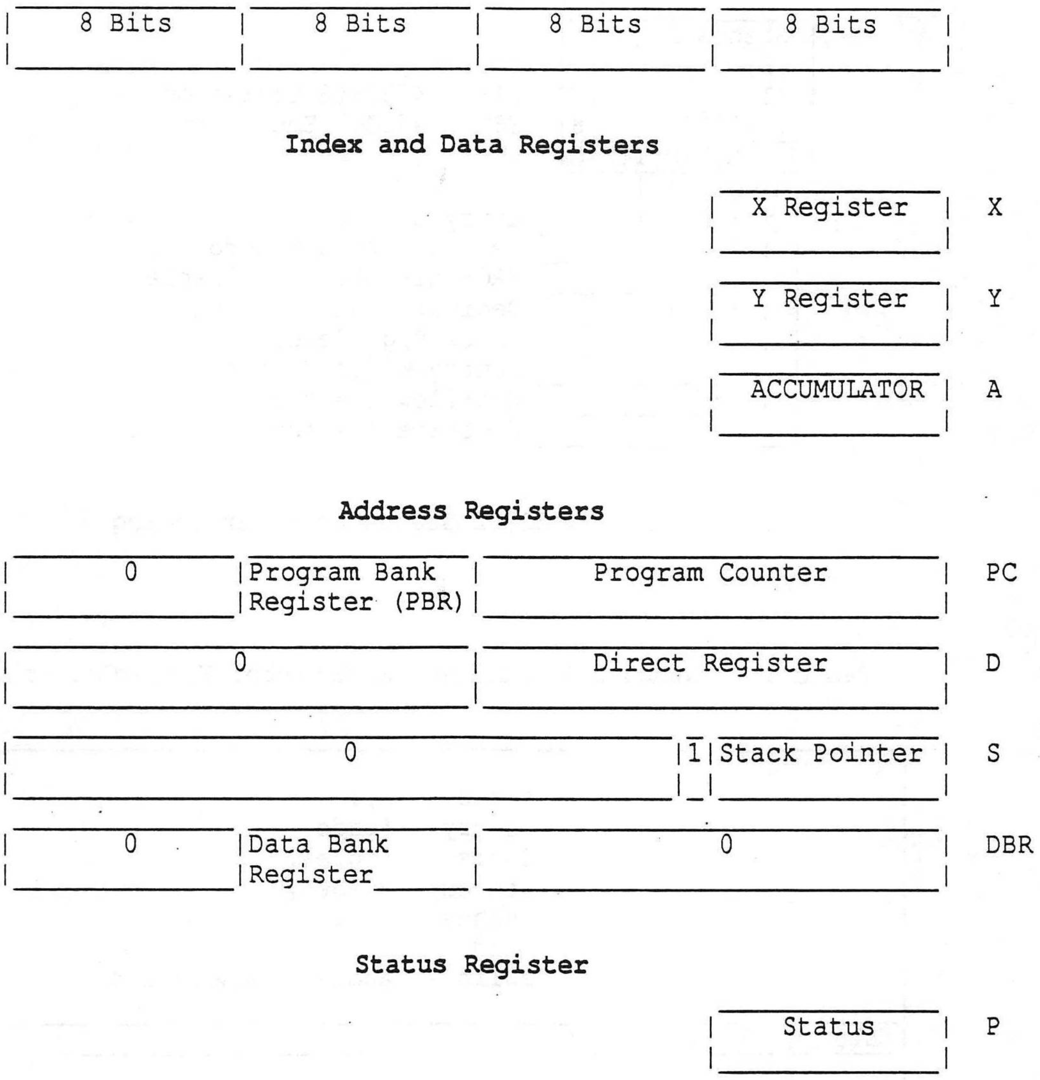


Figure 1-4 W65C02 8-bit Emulation Programming Model

```
 _______________
| Status Reg.(P)_|
| |E|         | |
| |1|___      |E|  E16 - W65C816 Emulation
| |6|1|B|     |8|  E8  - W65C02 Emulation
|N|V|M|X|D|I|Z|C|
 | | | | | | | |
 | | | | | | | |__ Carry 1 = true
 | | | | | | |____ Zero 1 = result zero
 | | | | | |______ IRQ- disable 1 = disable
 | | | | |________ Decimal mode 1 = true
 | | | |__________ Index Reg. Select
 | | |____________ Memory Width Select
 | |______________ Overflow 1 = true
 |________________ Negative 1 = neg.
```

Figure 1-5 W65C832 Status Register Coding

Table 1-1 W65C832 Emulation and Register Width Control

E16	E8	M	X	A and Memory Loads, Stores, Pushes, and Pulls	X,Y Loads, Stores, Pushes, Pulls, and Address Generation	
0	0	0	0	16	32	W65C832 Native
0	0	0	1	16	8	W65C832 Native
0	0	1	0	8	32	W65C832 Native
0	0	1	1	8	8	W65C832 Native
0	1	0	0	32	32	W65C832 Native
0	1	0	1	32	8	W65C832 Native
0	1	1	0	8	32	W65C832 Native
0	1	1	1	8	8	W65C832 Native
1	0	0	0	16	16	W65C816 Emulation
1	0	0	1	16	8	W65C816 Emulation
1	0	1	0	8	16	W65C816 Emulation
1	0	1	1	8	8	W65C816 Emulation
1	1	1	BRK	8	8	W65C02 Emulation

SECTION 2

PIN FUNCTION DESCRIPTION

```
               A
               B
           I   O               R           P
       M   R   R   R   V   V   E   V   M   H
       L   Q   T   D   P   S   S   D   /   I   B
       -   -   -   Y   -   S   -   A   X   2   E
     _____________________________________________
     | 6   5   4   3   2   1   44  43  42  41  40  |
NMI- | 7                                       39  | E8/E16
 VPA | 8                                       38  | R/W-
 VDD | 9                                       37  | VDD
  A0 | 10                                      36  | D0/A16
  A1 | 11                                      35  | D1/A17
 VSS | 12              W65C832                 34  | D2/A18
  A2 | 13                                      33  | D3/A19
  A3 | 14                                      32  | D4/A20
  A4 | 15                                      31  | D5/A21
  A5 | 16                                      30  | D6/A22
  A6 | 17                                      29  | D7/A23
     | 18  19  20  21  22  23  24  25  26  27  28  |
     _____________________________________________
       A   A   A   A   A   V   V   A   A   A   A
       7   8   9   1   1   S   S   1   1   1   1
                   0   1   S   S   2   3   4   5
```

Figure 2-1 W65C832 44 Pin PLCC Pinout

Table 2-1 Pin Function Table

Pin	Description
A0-A15	Address Bus
ABORT-	Abort Input
BE	Bus Enable
PHI2(IN)	Phase 2 In Clock
D0/A16-D7/A23	Data Bus/Address Bus
E8/E16	Emulation Select
IRQ-	Interrupt Request
ML-	Memory Lock
M/X	Mode Select (Pm or Px)
NMI-	Non-Maskable Interrupt
RDY	Ready
RES-	Reset
R/W-	Read/Write
VDA	Valid Data Address
VP-	Vector Pull
VPA	Valid Program Address
VDD	Positive Power Supply (+5 volts)
VSS	Internal Logic Ground

2.1 Abort (ABORT-)

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 ABORT- is a pulse-sensitive signal; i.e., an abort will occur whenever there is a negative pulse (or level) on the ABORT- pin during a PHI2 clock.

2.2 Address Bus (A0-A15)

These sixteen output lines form the low 16 bits of the Address Bus for memory and I/O exchange on the Data Bus. 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 R/W- signal. With Bus Enable high, the R/W- 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/A16-D7/A23)

These eight lines multiplex address bits A16-A23 with the data value D0-D7. 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. Four memory cycles are required to transfer 32-bit values. These lines may be set to the high impedance state by the Bus Enable (BE) signal.

2.5 Emulation Status (E8/E16)

The Emulation Status output E8/E16 reflects the state of the Emulation E8 and E16 mode flags 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 (IRQ-)

The Interrupt Request input signal is used to request that an interrupt sequence be initiated. When the IRQ 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 IRQ- 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.

2.7 Memory Lock (ML-)

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, five or nine cycles of ASL, DEC, INC, LSR, ROL, ROR, TRB, and TSB memory referencing instructions, depending on the state of the M and E8 flags.

2.8 Memory/Index Select Status (M/X)

This multiplexed output reflects the state of the Accumulator (M) and Index (X) select 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 the 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 (NMI-)

A negative transition on the NMI- 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 (8-bit Emulation mode), 00FFEA,B (16-bit Emulation mode) or 00FFDA,B (Native mode). Since NMI- 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 NMI- remains low.

2.10 Phase 2 In (PHI2)

This is the system clock input to the microprocessor internal clock generator. During the low power Standby Mode, PHI2 may held in the high or low state to preserve the contents of internal registers. However, usually it is held in the high state.

2.11 Read/Write (R/W-)

When the R/W- 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. The R/W- signal may be set to the high impedance state by Bus Enable (BE).

2.12 Ready (RDY)

This bidirectional 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 PHI2 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 RES-, ABORT-, NMI-, or IRQ- external interrupt is provided. This feature may be used to eliminate interrupt latency by placing the WAI instruction at the beginning of the IRQ- servicing routine. If the IRQ- Disable flag has been set, the next instruction will be executed when the IRQ- occurs. The processor will not stop after a WAI instruction if RDY has been forced to a high state. However, this feature should only be used on ASIC's and the RDY buffer modified. The Stop (STP) instruction has no effect on RDY.

2.13 Reset (RES-)

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 RES- signal must be held low for at least two clock cycles after VDD reaches operating voltage. Ready (RDY) has no effect while RES- is being held low. During the Reset conditioning period, the following period, the following processor initialization takes place:

Registers

D = 0000
DBR = 00
PBR = 00

SH = 01
XH = 00
YH = 00

	N	V/E16	M	X	D	I	Z	C/E8	
P =	*	*/1	1	1	0	1	*	*/1	* = not initialized

STP and WAI instructions are cleared.

Signals

E8 = 1
E16 = 1
M/X = 1
R/W- = 1
SYNC = 0

VDA = 0
VP- = 1
VPA = 0

When Reset is brought high, an interrupt sequence is initiated:
- o R/W- remains in the high state during the stack address cycles.
- o The Reset vector address is 00FFFC,D.

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

These two output signals indicate valid memory addresses when high logic 1, and are used for memory or I/O address qualification.

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

2.15 VDD and VSS

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

2.16 Vector Pull (VP-)

The Vector Pull output indicates that a vector location is being addressed during an interrupt sequence. VP- is low during the last two interrupt sequence cycles, during which time the processor reads the interrupt vector. The VP- signal may be used to select and prioritize interrupts from several sources by modifying the vector addresses.

SECTION 3

ADDRESSING MODES

The W65C832 is capable of directly addressing 16 MBytes of memory for program space and 4GBytes for data space although only 24 bits (16MBytes) of address space are available on the standard product. 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 00FFD0 and 00FFFF.

3.2 Stack

The Stack may 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 in W65C816 Emulation mode and some, where noted by (*), generate 32-bit effective address in W65C832 native mode.

o Direct Indexed Indirect (d,x)
* Direct Indirect Indexed (d),y
o Direct Indirect (d)
o Direct Indirect Long [d]
* Direct Indirect Long Indexed [d],y
o Absolute a
* Absolute a,x
* Absolute a,y
o 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 W65C832. The 32-bit indexed addressing modes are used with the W65C832; however, the high byte of the address is not available to the hardware on the standard W65C832 but is available on the core for ASIC's. Detailed descriptions of the 24 addressing modes are as follows:

3.5.1 Immediate Addressing-#

The operand is the second byte in 8-bit mode, second and third bytes when in the 16-bit mode, or 2nd thru 5th bytes in 32-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
Address:      |  DBR   | addrh | addrl |
```

3.5.3 Absolute Long-al

```
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 |
Operand              +          | offset |
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. In native mode this creates 32-bit effective addresses.

```
Instruction:   | opcode | offset |
                        | Direct Register |
                      +          | offset |
               |   00   | direct address  |
then:
               |   00   |(direct address) |
              +|  DBR   |
               |       base address       |
Operand       +         |        | Y Reg  |
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. In native mode this creates 32-bit effective addresses.

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

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 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 |
                        | direct address  |
                       +|         | X Reg |
               |   00   |     address     |
then:
               |   00   |    (address)    |
Operand       +|  DBR   |
Address:       |    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.

```
Instruction:  | opcode | offset |
                       | Direct Register |
                     +          | offset |
                       | direct address  |
Operand               +|        | X Reg  |
Address:      |   00   |effective address|
```

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.

```
Instruction:  | opcode | offset |
                       | Direct Register |
                     +          | offset |
                       | direct address  |
Operand               +|        | Y Reg  |
Address:      |   00   |effective address|
```

* 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. In native mode this creates 32-bit effective addresses.

```
Instruction:  | opcode | addrl | addrh |
              |  DBR   | addrh | addrl |
Operand               +|       | X Reg |
Address:      |    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. In native mode this creates 32-bit effective addresses.

```
Instruction:  | opcode | addrl | addrh | baddr |
              | baddr  | addrh | addrl |
Operand               +|       | X Reg |
Address:      |    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. In native mode this creates 32-bit effective addresses.

```
Instruction:  | opcode | addrl | addrh |
              |  DBR   | addrh | addrl |
Operand               +|       | Y Reg |
Address:      |    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 usd 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:
              |   00   |(direct address) |
Operand      +|  DBR   |
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 |
                     +          | offset |
              |   00   | direct address  |
then:
Operand       |     (direct address)     |
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   |
Operand               +        | offset  |
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. In the native mode this creates 32-bit effective addresses.

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

3.5.24 Block Source Bank, Destination Bank-xya

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 Accumulator contains one less than the number of bytes to move. When the Accumulator is zero it will move one byte. The second byte of the block move instructions is also loaded into the Data Bank Register. In W65C832 native mode this X Index Register contains the entire source address and the X Index Register contains the entire destination address; therefore, the instruction is shorter by two bytes and two cycles per byte moved.

```
                 ________________________________
Instruction:   | opcode | dstbnk | srcbnk |
Source                    dstbnk -> DBR
Address:                | srcbnk |       X Reg       |
Destination             |   DBR  |       Y Reg       |
Address:
          Increment (MVN) or decrement (MVP) X and Y.
          Decrement C (if greater than zero), then PC+3->PC.
```

* In W65C832 native mode these addressing modes creates 32-bit effective data space addresses.

Table 3-1 Address Mode Formats

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

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

Table 3-2 Addressing Mode Summary

	Instruction Times In Memory Cycles		Memory Utilization In Number of Program Sequence Bytes	
Address Mode	Original 8-bit NMOS 6502	New W65C832	Original 8-bit NMOS 6502	New W65C832
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	3-11	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(6)	-	3(6)

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. 16 bit operation, add 1 cycle, add 1 byte for immediate.
   32 bit operation, add 3 cycles, add 3 bytes for immediate.
4. Direct register low (DL) not equal zero, add 1 cycle.
5. Read-Modify-Write, add 2 cycles for 8-bit, add 4 cycles for 16-bit, add 8 cycles for 32-bit operation.
6. For W65C832 native mode, subtract 2 cycles and 2 bytes.

SECTION 4

TIMING, AC AND DC CHARACTERISTICS

4.1 Absolute Maximum Ratings: (Note 1)

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
Operating Temperature	TA	0 °C to +70°C
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.

Notes:
1. 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 = 0oC to +70oC

Table 4-2 DC Characteristics

Parameter	Symbol	Min	Max	Unit
Input High Voltage	Vih			
RES-, RDY, IRQ-, Data, BE		2.0	VDD+0.3	V
PHI2, NMI-, ABORT-		0.9*VDD	VDD+0.3	V
Input Low Voltage	Vil			
RES-, RDY, IRQ-, Data, BE		-0.3	0.8	V
PHI2, NMI-, ABORT-		-0.3	0.1*VDD	V
Input Leakage Current (Vin = 0.4 to 2.4)	Iin			
RES-, NMI-, IRQ-, BE, ABORT- (Internal Pullup)		-100	1	uA
RDY (Internal Pullup, Open Drain)		-100	10	uA
PHI2	Iin	-1	1	uA
Address, Data, R/W-, (Off State, BE=0)		-10	10	uA
Output High Voltage (Ioh=-100uA)	Voh			
Data,Address,R/W-,ML-,VP-,M/X,E8/E16 VDA,VPA		0.7 VDD	-	V
Output Low Voltage (Iol = 1.6mA)	Vol			
Data,Address,R/W-,ML-,VP-,M/X,E8/E16 VDA,VPA		-	0.4	V
Supply Current (No Load)	Idd		4	mA/MHz
Standby Current (No Load, Data Bus = VSS orVDD RES-,NMI-,IRQ-,SO-,BE,ABORT-,PHI2=VDD)	Isb	-	1	uA
Capacitance (Vin=0V, TA=25oC, f=2MHz)				
Logic,PHI2	Cin	-	10	pF
Address, Data, R/W-(Off State)	Cts	-	15	pF

4.3 General AC Characteristics: VDD= 5.0V +/- 5%, VSS= 0V,
Ta= 0oC to +70oC

Table 4-3A W65C832 General AC Characteristics, 4-7MHz

		4 MHz		5 MHz		6 MHz		7 MHz		
Parameter	Symbol	Min	Max	Min	Max	Min	Max	Min	Max	Unit
Cycle Time	tCYC	250	DC	200	DC	165	DC	140	DC	nS
Clock Pulse Width Low	tPWL	.125	10	.10	10	.082	10	.07	10	uS
Clock Pulse Width High	tPWH	125		100	-	82		70	-	nS
Fall Time, Rise Time	tF,tR	-	10	-	10	-	5	-	5	nS
A0-A15 Hold Time	tAH	10	-	10	-	10	-	10	-	nS
A0-A15 Setup Time	tADS	-	75	-	67	-	60	-	60	nS
A16-A23 Hold Time	tBH	10	-	10	-	10	-	10	-	nS
A16-A23 Setup Time	tBAS	-	90	-	77	-	65	-	55	nS
Access Time	tACC	130	-	115	-	87	-	60	-	nS
Read Data Hold Time	tDHR	10	-	10	-	10	-	10	-	nS
Read Data Setup Time	tDSR	30	-	25	-	20	-	25	-	nS
Write Data Delay Time	tMDS	-	70	-	65	-	60	-	55	nS
Write Data Hold Time	tDHW	10	-	10	-	10	-	10	-	nS
Processor Control Setup Time	tPCS	30	-	25	-	20	-	20	-	nS
Processor Control Hold Time	tPCH	10	-	10	-	10	-	10	-	nS
E8/E16,MX Output Hold Time	tEH	10	-	10	-	5	-	5	-	nS
E8/E16,MX Output Setup Time	tES	50	-	37	-	25	-	25	-	nS
Capacitive Load *1	CEXT	-	100	-	100	-	35	-	35	pF
BE to Valid Data *2	tBVD	-	30	-	30	-	30	-	30	nS

Table 4-3B W65C832 General AC Characteristics, 8-10MHz

		8 MHz		9 MHz		10 MHz		
Parameter	Symbol	Min	Max	Min	Max	Min	Max	Unit
Cycle Time	tCYC	125	DC	110	DC	100	DC	nS
Clock Pulse Width Low	tPWL	.062	10	.055	10	.05	10	uS
Clock Pulse Width High	tPWH	62	-	55	-	50	-	nS
Fall Time, Rise Time	tF,tR	-	5	-	5	-	5	nS
A0-A15 Hold Time	tAH	10	-	10	-	10	-	nS
A0-A15 Setup Time	tADS	-	40	-	40	-	40	nS
A16-A23 Hold Time	tBH	10	-	10	-	10	-	nS
A16-A23 Setup Time	tBAS	-	45	-	45	-	45	nS
Access Time	tACC	70	-	70	-	70	-	nS
Read Data Hold Time	tDHR	10	-	10	-	10	-	nS
Read Data Setup Time	tDSR	15	-	15	-	15	-	nS
Write Data Delay Time	tMDS	-	40	-	40	-	40	nS
Write Data Hold Time	tDHW	10	-	10	-	10	-	nS
Processor Control Setup Time	tPCS	15	-	15	-	15	-	nS
Processor Control Hold Time	tPCH	10	-	10	-	10	-	nS
E8/E16,MX Output Hold Time	tEH	5	-	5	-	5	-	nS
E8/E16,MX Output Setup Time	tES	15	-	15	-	15	-	nS
Capacitive Load *1	CEXT	-	35	-	35	-	35	pF
BE to Valid Data	tBVD	-	30	-	30	-	30	nS

*1 Applies to Address, Data, R/W

*2 BE to High Impedence State is not testable but should be the same amount of time as BE to Valid Data

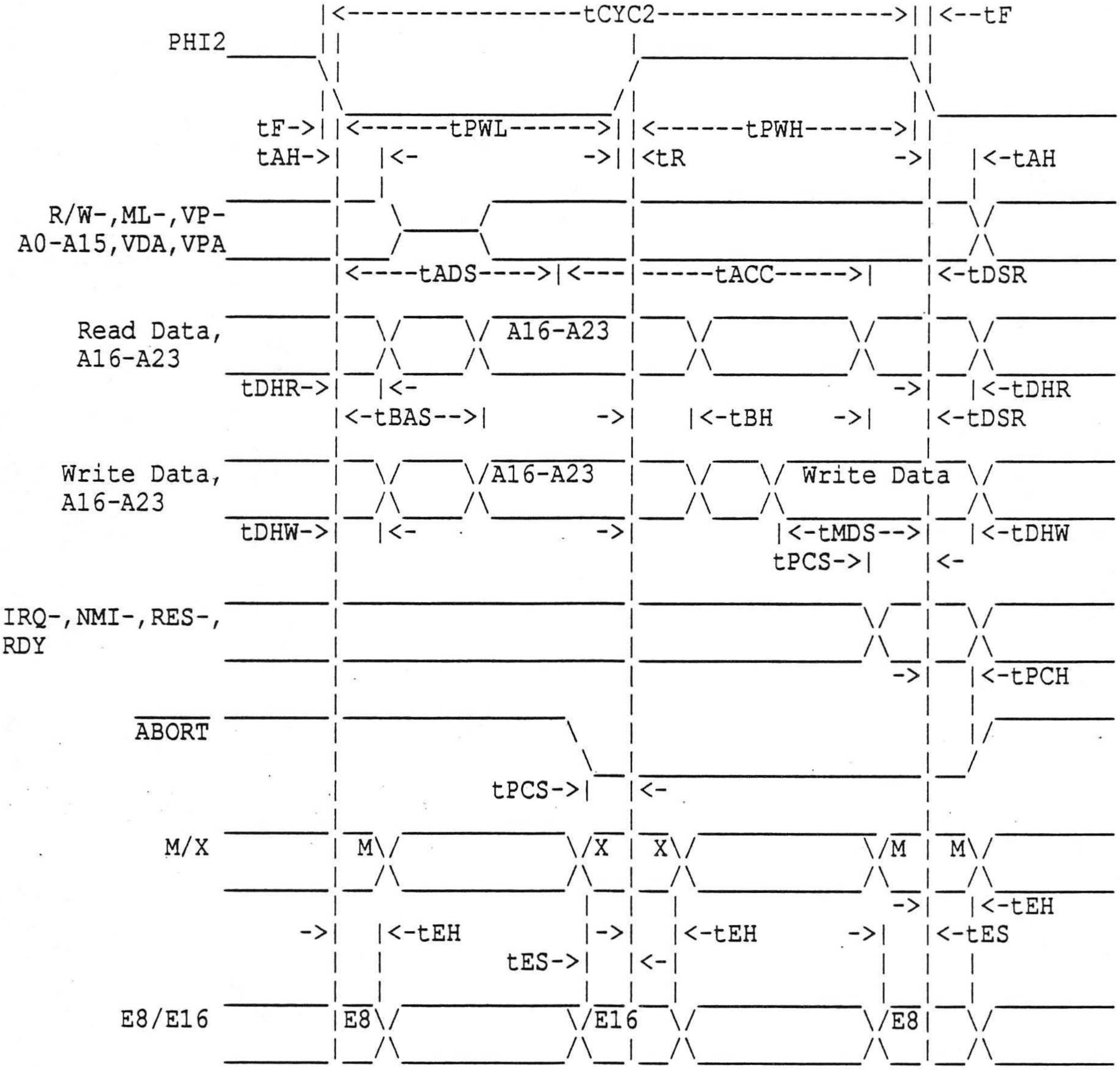
4.4 General AC Characteristics: VDD= 1.2V, VSS= 0V,
Ta= 0oC to +70oC

Table 4-4A W65C832 General AC Characteristics, 40 KHz

		40 KHz		
Parameter	Symbol	Min	Max	Unit
Cycle Time	tCYC	-	25	uS
Clock Pulse Width Low	tPWL	12.5	13	uS
Clock Pulse Width High	tPWH	12.5	-	uS
Fall Time, Rise Time	tF,tR	-	10	nS
A0-A15 Hold Time	tAH	10	-	nS
A0-A15 Setup Time	tADS	-	2	uS
AA0-A23 Hold Time	tBH	10	-	nS
A16-A23 Setup Time	tBAS	-	2	uS
Access Time	tACC	35	-	uS
Read Data Hold Time	tDHR	100	-	nS
Read Data Setup Time	tDSR	1.5	-	uS
Write Data Delay Time	tMDS	-	2	uS
Write Data Hold Time	tDHW	10	-	nS
Processor Control Setup Time	tPCS	1.5	-	uS
Processor Control Hold Time	tPCH	100	-	nS
E8/E16,MX Output Hold Time	tEH	10	-	nS
E8/E16,MX Output Setup Time	tES	100	-	nS
Capacitive Load *1	CEXT	-	100	pF
BE to Valid Data *2	tBVD	-	30	nS

*1 Applied to Address, Data, R/W

*2 BE to High Impedance State is not testable but should be the same amount of time as BE to Valid Data



Timing Notes:

1. Voltage levels are Vl<0.4V, Vh>2.4V.
2. Timing measurement points are 0.8V and 2.0V.

Figure 4-1 General Timing Diagram

SECTION 5

ORDERING INFORMATION

```
                                      W  65C832 PL   -8
Description___________________________|    |     |  |  |
W-Standard                                 |     |  |  |
                                           |     |  |  |
Product Identification Number______________|     |  |  |
                                                 |  |  |
Package__________________________________________|  |  |
PL-44 leaded plastic chip carrier                   |  |
                                                    |  |
Temperature/Processing______________________________|  |
Blank- 0oC to +70oC                                    |
                                                       |
Performance Designator_________________________________|
Designators selected for speed and power.
-4 4MHz  -6 6MHz  -8 8MHz  -10 10MHz  -E Experimental
```

General sales or technical assistance, and information about devices supplied to a custom specification may be requested from:

The Western Design Center, Inc.
2166 East Brown Road
Mesa, Arizona 85213
Phone: 602-962-4545 Fax: 602-835-6442

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.

SECTION 6

APPLICATION INFORMATION

Table 6-1 W65C832 Instruction Set-Alphabetical Sequence

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift One Bit
BCC	Branch on Carry Clear (Pc=0)
BCS	Branch on Carry Set (Pc=1)
BEQ	Branch if Equal (Pz=1)
BIT	Bit Test
BMI	Branch if Result Minus (Pn=1)
BNE	Branch if Not Equal (Pz=0)
BPL	Branch if Result Plus (Pn=0)
BRA	Branch Always
BRK	Force Break
BRL	Branch Always Long
BVC	Branch on Overflow Clear (Pv=0)
BVS	Branch on Overflow Set (Pv=1)
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
COP	Coprocessor
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory or Accumulator by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive OR" Memory with Accumulator
INC	Increment Memory or Accumulator by One
INX	Increment Index X by One
INY	Increment Index Y by One
JML	Jump Long
JMP	Jump to New Location
JSL	Jump Subroutine Long
JSR	Jump to New Location Saving Return
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or Accumulator)
MVN	Block Move Negative
MVP	Block Move Positive
NOP	No Operation
ORA	"OR" Memory with Accumulator
PEA	Push Effective Absolute Address on Stack
PEI	Push Effective Absolute Address on stack
PER	Push Effective Program Counter Relative Address on Stack
PHA	Push Accumulator on Stack
PHB	Push Data Bank Register on Stack
PHD	Push Direct Register on Stack
PHK	Push Program Bank Register on Stack
PHP	Push Processor Status on Stack
PHX	Push Index X on Stack
PHY	Push Index Y on Stack
PLA	Pull Accumulator from Stack
PLB	Pull Data Bank Register from
PLD	Pull Direct Register from Stack
PLP	Pull Processor Status from Stack
PLX	Pull Index X from Stack
PLY	Pull Index Y from Stack
REP	Reset Status Bits
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTL	Return from Subroutine Long
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEP	Set Processor Status Bite
STA	Store Accumulator In Memory
STP	Stop the Clock
STX	Store Index X in Memory
STY	Store Index Y in Memory
STZ	Store Zero in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TCD	Transfer C Accumulator to Direct Register
TCS	Transfer C Accumulator to Stack Pointer Register
TDC	Transfer Direct Register to C Accumulator
TRB	Test and Reset Bit
TSB	Test and Set Bit
TSC	Transfer Stack Pointer Register to C Accumulator
TSX	Transfer Stack Pointer Register to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer Register
TXY	Transfer Index X to Index Y
TYA	Transfer Index Y to Accumulator
TYX	Transfer Index Y to Index X
WAI	Wait for Interrupt
WDM	Reserved for Future Use
XBA	Exchange B and A Accumulator
XCE	Exchange Carry and Emulation E8
XFE	Exchange Carry and Emulation E8 and Exchange Overflow and Emulation E16

For alternate mnemonics, see Table 7-3-1.

Table 6-2 Vector Locations

W65C02 8- Emulation	
00FFFE,F-IRQ-/BRK	Hardware/Software
00FFFC,D-RESET-	Hardware
00FFFA,C-NMI-	Hardware
00FFF8,9-ABORT-	Hardware
00FFF6,7-(Reserved)	
00FFF4,5-COP	Software

W65C816 16-bit Emulation	
00FFEE,F-IRQ-	Hardware
00FFEC,D-(Reserved)	
00FFEA,B-NMI-	Hardware
00FFE8,9-ABORT-	Hardware
00FFE6,7-BRK	Software
00FFE4,5-COP	Software

W65C832 Native	
00FFDE,F-IRQ-	Hardware
00FFDC,D-(Reserved)	
00FFDA,B-NMI-	Hardware
00FFD8,9-ABORT-	Hardware
00FFD6,7-BRK	Software
00FFD4,5-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.

Table 6-3 Opcode Matrix

LSD

MSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
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 ● 6	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 6	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 6	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	

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	★ = New W65C816/802 Opcodes ● = New W65C02 Opcodes Blank = NMOS 6502 Opcodes	ADDRESSING MODE
BASE NO. BYTES		BASE NO. CYCLES

Table 6-4 Operation, Operation Codes and Status Register

MNE-MONIC		#	a	al	d	A	I	(d),y	[d],y	(d,x)	d,x	d,y	a,x	al,x	a,y	r	rl	(a)	(d)	[d]	(a,x)	s	d,s	(d,s),y	xyc	PROCESSOR STATUS CODE								MNE-MONIC
																										7	6	5	4	3	2	1	0	
																										N	V	M	X	D	I	Z	C	E = 0
	OPERATION	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N	V	1	B	D	I	Z	C	E = 1
ADC	A + M + C → A	69	6D	6F	65			71	77	61	75		7D	7F	79				72	67			63	73		N	V	.	.	.	.	Z	C	ADC
AND	A∧M → A	29	2D	2F	25			31	37	21	35		3D	3F	39				32	27			23	33		N	.	.	.	.	.	Z	.	AND
ASL	C ← 15/7 0 ← 0		0E		06	0A					16		1E													N	.	.	.	.	.	Z	C	ASL
BCC	BRANCH IF C = 0															90										.	.	.	.	.	.	.	.	BCC
BCS	BRANCH IF C = 1															B0										.	.	.	.	.	.	.	.	BCS
BEQ	BRANCH IF Z = 1															F0										.	.	.	.	.	.	.	.	BEQ
BIT	A∧M (NOTE 1)	89	2C		24						34		3C													M_7	M_6	.	.	.	.	Z	.	BIT
BMI	BRANCH IF N = 1															30										.	.	.	.	.	.	.	.	BMI
BNE	BRANCH IF Z = 0															D0										.	.	.	.	.	.	.	.	BNE
BPL	BRANCH IF N = 0															10										.	.	.	.	.	.	.	.	BPL
BRA	BRANCH ALWAYS															80										.	.	.	.	.	.	.	.	● BRA
BRK	BREAK (NOTE 2)																					00				.	.	.	●	0	I	.	.	BRK
BRL	BRANCH LONG ALWAYS																82									.	.	.	.	.	.	.	.	* BRL
BVC	BRANCH IF V = 0															50										.	.	.	.	.	.	.	.	BVC
BVS	BRANCH IF V = 1															70										.	.	.	.	.	.	.	.	BVS
CLC	0 → C						18																			.	.	.	.	.	.	.	0	CLC
CLD	0 → D						D8																			.	.	.	.	0	.	.	.	CLD
CLI	0 → 1						58																			.	.	.	.	.	0	.	.	CLI
CLV	0 → V						B8																			.	0	.	.	.	.	.	.	CLV
CMP	A-M	C9	CD	CF	C5			D1	D7	C1	D5		DD	DF	D9				D2	C7			C3	D3		N	.	.	.	.	.	Z	C	CMP
COP	CO-PROCESSOR																					02				.	.	.	.	0	I	.	.	* COP
CPX	X-M	E0	EC		E4																					N	.	.	.	.	.	Z	C	CPX
CPY	Y-M	C0	CC		C4																					N	.	.	.	.	.	Z	C	CPY
DEC	DECREMENT		CE		C6	3A					D6		DE													N	.	.	.	.	.	Z	.	DEC
DEX	X - 1 → X						CA																			N	.	.	.	.	.	Z	.	DEX
DEY	Y - 1 → Y						88																			N	.	.	.	.	.	Z	.	DEY
EOR	A∀M → A	49	4D	4F	45			51	57	41	55		5D	5F	59				52	47			43	53		N	.	.	.	.	.	Z	.	EOR
INC	INCREMENTS		EE		E6	1A					F6		FE													N	.	.	.	.	.	Z	.	INC
INX	X + 1 → X						E8																			N	.	.	.	.	.	Z	.	INX
INY	Y + 1 → Y						C8																			N	.	.	.	.	.	Z	.	INY
JML	JUMP LONG TO NEW LOC.																	DC								.	.	.	.	.	.	.	.	* JML
JMP	JUMP TO NEW LOC.		4C	5C														6C			7C					.	.	.	.	.	.	.	.	JMP
JSL	JUMP LONG TO SUB.			22																						.	.	.	.	.	.	.	.	JSL
JSR	JUMP TO SUB.		20																		FC					.	.	.	.	.	.	.	.	JSR
LDA	M → A	A9	AD	AF	A5			B1	B7	A1	B5		BD	BF	B9				B2	A7			A3	B3		N	.	.	.	.	.	Z	.	LDA
LDX	M → X	A2	AE		A6							B6			BE											N	.	.	.	.	.	Z	.	LDX
LDY	M → Y	A0	AC		A4						B4		BC													N	.	.	.	.	.	Z	.	LDY
LSR	0 → 15/7 0 → C		4E		46	4A					56		5E													0	.	.	.	.	.	Z	C	LSR
MVN	M → M NEGATIVE																								54	.	.	.	.	.	.	.	.	* MVN
MVP	M → M POSITIVE																								44	.	.	.	.	.	.	.	.	* MVP
NOP	NO OPERATION						EA																			.	.	.	.	.	.	.	.	NOP
ORA	AVM → A	09	0D	0F	05			11	17	01	15		1D	1F	19				12	07			03	13		N	.	.	.	.	.	Z	.	ORA
PEA	Mpc + 1, Mpc + 2 → Ms - 1, Ms S - 2 → S																					F4				.	.	.	.	.	.	.	.	* PEA
PEI	M(d), M(d + 1) → Ms - 1, Ms S - 2 → S																					D4				.	.	.	.	.	.	.	.	* PEI
PER	Mpc + rl, Mpc + rl + 1 → Ms - 1, Ms S - 2 → S																					62				.	.	.	.	.	.	.	.	* PER
PHA	A → Ms, S - 1 → S																					48				.	.	.	.	.	.	.	.	PHA
PHB	DBR → Ms, S - 1 → S																					8B				.	.	.	.	.	.	.	.	PHB
PHD	D → Ms, Ms - 1, S - 2 → S																					0B				.	.	.	.	.	.	.	.	* PHD
PHK	PBR → Ms, S - 1 → S																					4B				.	.	.	.	.	.	.	.	* PHK
PHP	P → Ms, S - 1 → S																					08				.	.	.	.	.	.	.	.	PHP
PHX	X → Ms, S - 1 → S																					DA				.	.	.	.	.	.	.	.	● PHX
PHY	Y → Ms, S - 1 → S																					5A				.	.	.	.	.	.	.	.	● PHY
PLA	S + 1 → S, Ms → A																					68				N	.	.	.	.	.	Z	.	PLA
PLB	S + 1 → S, Ms → DBR																					AB				N	.	.	.	.	.	Z	.	* PLB
PLD	S + 2 → S, Ms - 1, Ms → D																					2B				N	.	.	.	.	.	Z	.	* PLD
PLP	S + 1 → S, Ms → P																					28				N	V	M	X	D	I	Z	C	PLP
PLX	S + 1 → S, Ms → X																					FA				N	.	.	.	.	.	Z	.	PLX
PLY	S + 1 → S, Ms → Y																					7A				N	.	.	.	.	.	Z	.	PLY
REP	M∧P → P	C2																								N	V	M	X	D	I	Z	C	* REP
ROL	15/7 0 ← C		2E		26	2A					36		3E													N	.	.	.	.	.	Z	C	ROL
ROR	C → 15/7 0		6E		66	6A					76		7E													N	.	.	.	.	.	Z	C	ROR
RTI	RTRN FROM INT.																					40				N	V	M	X	D	I	Z	C	RTI
RTL	RTRN FROM SUB. LONG																					6B				.	.	.	.	.	.	.	.	* RTL
RTS	RTRN SUBROUTINE																					60				.	.	.	.	.	.	.	.	RTS
SBC	A - M - $\overline{C}$ → A	E9	ED	EF	E5			F1	F7	E1	F5		FD	FF	F9				F2	E7			E3	F3		N	V	.	.	.	.	Z	C	SBC
SEC	1 → C						38																			.	.	.	.	.	.	.	1	SEC
SED	1 → D						F8																			.	.	.	.	1	.	.	.	SED
SEI	1 → I						78																			.	.	.	.	.	1	.	.	SEI
SEP	MVP → P	E2																								N	V	M	X	D	I	Z	C	* SEP
STA	A → M		8D	8F	85			91	97	81	95		9D	9F	99				92	87			83	93		.	.	.	.	.	.	.	.	STA
STP	STOP (1 → ϕ2)						DB																			.	.	.	.	.	.	.	.	● STP
STX	X → M		8E		86							96														.	.	.	.	.	.	.	.	STX
STY	Y → M		8C		84						94															.	.	.	.	.	.	.	.	STY
STZ	00 → M		9C		64						74		9E													.	.	.	.	.	.	.	.	● STZ
TAX	A → X						AA																			N	.	.	.	.	.	Z	.	TAX
TAY	A → Y						A8																			N	.	.	.	.	.	Z	.	TAY
TCD	C → D						5B																			N	.	.	.	.	.	Z	.	* TCD
TCS	C → S						1B																			.	.	.	.	.	.	.	.	* TCS
TDC	D → C						7B																			N	.	.	.	.	.	Z	.	* TDC
TRB			1C		14																					.	.	.	.	.	.	Z	.	● TRB
TSB	AVM → M		0C		04																					.	.	.	.	.	.	Z	.	● TSB
TSC	S → C					3B																				N	.	.	.	.	.	Z	.	* TSC
TSX	S → X					BA																				N	.	.	.	.	.	Z	.	TSX
TXA	X → A					8A																				N	.	.	.	.	.	Z	.	TXA
TXS	X → S					9A																				.	.	.	.	.	.	.	.	TXS
TXY	X → Y					9B																				N	.	.	.	.	.	Z	.	* TXY
TYA	Y → A					98																				N	.	.	.	.	.	Z	.	TYA
TYX	Y → X					BB																				N	.	.	.	.	.	Z	.	* TYX
WAI	0 → RDY					CB																				.	.	.	.	.	.	.	.	● WAI
WDM	NO OPERATION (RESERVED)					42																				.	.	.	.	.	.	.	.	* WDM
XBA	B → A					EB																				N	.	.	.	.	.	Z	.	* XBA
XCE	C → E					FB																				.	.	.	.	.	.	.	E	* XCE

Notes:
1. Bit immediate N and V flags not affected. When M = 0, M_{15} → N and M_{14} → V.
2. Break Bit (B) in Status register indicates hardware or software break.
3. * = New W65C816/802 Instructions
● = New W65C02 Instructions
Blank = NMOS 6502

+ Add
- Subtract
∧ AND
V OR
∀ Exclusive OR

Table 6-5 Instruction Operation

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	(14) VDA,	(14) VPA	(15) ADDRESS BUS	DATA BUS	R/$\overline{W}$
1. Immediate-#		1.	1	1	1	1	PBR,PC	OpCode	1
(LDY,CPY,CPX,LDX,ORA,		2.	1	1	0	1	PBR,PC+1	ID0	1
AND,EOR,ADC,BIT,LDA,	(1)	2a.	1	1	0	1	PBR,PC+2-4	ID1-3	1
CMP,SBC,REP,SEP)									
(14 OpCodes)									
(2, 3 and bytes)									
(2, 3 and 5 cycles)									
2a. Absolute-a		1.	1	1	1	1	PBR,PC	OpCode	1
(BIT,STY,STZ,LDY,		2.	1	1	0	1	PBR,PC+1	AAL	1
CPY,CPX,STX,LDX,		3.	1	1	0	1	PBR,PC+2	AAH	1
ORA,AND,EOR,ADC,		4.	1	1	1	0	DBR,AA	Byte 0	1/0
STA,LDA,CMP,SBC)	(1)	4a.	1	1	1	0	DBR,AA+1-3	Bytes1-3	1/0
(18 OpCodes)									
(3 bytes)									
(4, 5 and 7 cycles									
2b. Absolute-(R-M-W)-a		1.	1	1	1	1	PBR,PC	OpCode	1
(ASL,ROL,LSR,ROR		2.	1	1	0	1	PBR,PC+1	AAL	1
DEC,INC,TSB,TRB)		3.	1	1	0	1	PBR,PC+2	AAH	1
(6 OpCodes)		4.	1	0	1	0	DBR,AA	Byte 0	1
(3 bytes)	(1)	4a.	1	0	1	0	DBR,AA+-3	Bytes 1-3	1
(6 for 8-bit data,	(3)	5.	1	0	0	0	DBR,AA+1 or 3	IO	1
8 for 16-bit data,	(1)	6a.	1	0	1	0	DBR,AA+3-1	Bytes 3-1	0
12 for 32-bit data)		6.	1	0	1	0	DBR,AA	Byte 0	0
2c. Absolute (JUMP)-a		1.	1	1	1	1	PBR,PC	OpCode	1
(JMP)(4C)		2.	1	1	0	1	PBR,PC+1	New PCL	1
(1 OpCode)		3.	1	1	0	1	PBR,PC+2	New PCH	1
(3 bytes)		1.	1	1	1	1	PBR,NEW PC	New OpCode	1
(3 cycles)									
2d. Absolute (Jump to		1.	1	1	1	1	PBR,PC	OpCode	1
subroutine)-a		2.	1	1	0	1	PBR,PC+1	New PCL	1
(JSR)		3.	1	1	0	1	PBR,PC+2	New PCH	1
(1 OpCode)		4.	1	1	0	0	PBR,PC+2	IO	1
(3 bytes)		5.	1	1	1	0	0,S	PCH	0
(6 cycles)		6.	1	1	1	0	0,S-1	PCL	0
(different order from		1.	1	1	1	1	PBR,NEW PC	New OpCode	1
N6502)									
*3a. Absolute Long-al		1.	1	1	1	1	PBR,PC	OpCode	1
(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
(8 OpCodes)		4.	1	1	0	1	PBR,PC+3	AAB	1
(4 bytes)		5.	1	1	1	0	AAB,AA	Byte 0	1/0
(5, 6 and 8 cycles)	(1)	5a.	1	1	1	0	AAB,AA+1	Bytes1-3	1/0
*3b. Absolute Long (JUMP)-al		1.	1	1	1	1	PBR,PC	OpCode	1
(JMP)		2.	1	1	0	1	PBR,PC+1	New PCL	1
(1 OpCode)		3.	1	1	0	1	PBR,PC+2	New PCH	1
(4 bytes)		4.	1	1	0	1	PBR,PC+3	New BR	1
(4 cycles)		1.	1	1	1	1	NEW PBR,PC	OpCode	1

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA		ADDRESS BUS	DATA BUS	R/$\overline{W}$
*3c. Absolute Long (JUMP to		1.	1	1	1	1		PBR,PC	OpCode	1
Subroutine Long)-al		2.	1	1	0	1		PBR,PC+1	New PCL	1
(JSL)		3.	1	1	0	1		PBR,PC+2	New PCH	1
(1 OpCode)		4.	1	1	1	0		0,S	PBR	0
(4 bytes)		5.	1	1	0	0		0,S	IO	1
(7 cycles)		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
4a. Direct-d		1.	1	1	1	1		PBR,PC	OpCode	1
(BIT,STZ,STY,LDY,		2.	1	1	0	1		PBR,PC+1	DO	1
CPY,CPX,STX,LDX,	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
ORA,AND,EOR,ADC,		3.	1	1	1	0		0,D+DO	Byte 0	1/0
STA,LDA,CMP,SBC)	(1)	3a.	1	1	1	0		0,D+DO+1-3	Bytes1-3	1/0
(18 OpCodes)										
(2 bytes)										
(3, 4, 5 and 7 cycles)										
4b. Direct (R-M-W)-d		1.	1	1	1	1		PBR,PC	OpCode	1
(ASL,ROL,LSR,ROR		2.	1	1	0	1		PBR,PC+1	DO	1
DEC,INC,TSB,TRB)	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
(6 OpCodes)		3.	1	0	1	0		0,D+DO	Byte 0	1
(2 bytes)	(1)	3a.	1	0	1	0		0,D+DO+1-3	Bytes 1-3	1
(5,6,7,8,11 and	(3)	4.	1	0	0	0		0,D+DO+1 or 3	IO	1
12 cycles)	(1)	5a.	1	0	1	0		0,D+DO+3-1	Bytes 3-1	0
		5.	1	0	1	0		0,D+DO	Byte 0	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,		2.	1	1	0	0		PBR,PC+1	IO	1
XCE,TYA,TAY,TXA,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)		3.	1	1	0	0		PBR,PC+1	IO	1
(1 byte)										
(3 cycles)										
#6c. Wait-for-Interrupt							RDY			
(WAI)		1.	1	1	1	1	1	PBR,PC	OpCode	1
(1 OpCode)	(9)	2.	1	1	0	0	1	PBR,PC+1	IO	1
(1 byte)		3.	1	1	0	0	0	PBR,PC+1	IO	1
(3 cycles)	$\overline{IRQ}$,$\overline{NMI}$	1.	1	1	1	1	1	PBR,PC+1	IRQ(BRK)	1

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA		ADDRESS BUS	DATA BUS	R/$\overline{W}$
#6d. Stop-the-Clock							RDY			
(STP)		1.	1	1	1	1	1	PBR,PC	OpCode	1
(1 OpCode)		2.	1	1	0	0	1	PBR,PC+1	IO	1
(1 byte)	RES-=1	3.	1	1	0	0	1	PBR,PC+1	IO	1
(3 cycles)	RES-=0	1c.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
	RES-=0	1b.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
	RES-=1	1a.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
(See 21a. Stack		1.	1	1	1	1	1	PBR,PC+1	BEGIN	1
Hardware Interrupt)										
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,	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
STA,LDA,CMP,SBC)		3.	1	1	1	0		0,D+DO	AAL	1
(8 OpCodes)		4.	1	1	1	0		0,D+DO+1	AAH	1
(2 bytes)	(4)	4a.	1	1	0	0		DBR,AAJ,AAL+YL	IO	1
(5,6,7,8,9 and 10		5.	1	1	1	0		DBR,AA+Y	Byte 0	1/0
cycles)	(1)	5a.	1	1	1	0		DBR,AA+Y+1-3	Bytes1-3	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,	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
STA,LDA,CMP,SBC)		3.	1	1	1	0		0,D+DO	AAL	1
(8 OpCodes)		4.	1	1	1	0		0,D+DO+1	AAH	1
(2 bytes)		5.	1	1	1	0		0,D+DO+2	AAB	1
(6,7,8,9 and 10	(17)	5a.	1	1	0	0		0,0+DO+2	IO	1
cycles)		6.	1	1	1	0		AAB,AA+Y	Byte 0	1/0
	(1)	6a.	1	1	1	0		AAB,AA+Y+1-3	Bytes1-3	1/0
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,	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
STA,LDA,CMP,SBC)		3.	1	1	0	0		PBR,PC+1	IO	1
(8 OpCodes)		4.	1	1	1	0		0,D+DO+X	AAL	1
(2 bytes)		5.	1	1	1	0		O,D+DO+X+1	AAH	1
(6,7,8,9 and 10 cycles)		6.	1	1	1	0		DBR,AA	Byte 0	1/0
	(1)	6a.	1	1	1	0		DBR,AA+1-3	Bytes1-3	1/0
10a.Direct,X-d,x		1.	1	1	1	1		PBR,PC	OpCode	1
(BIT,STZ,STY,LDY,		2.	1	1	0	1		PBR,PC+1	DO	1
ORA,AND,EOR,ADC,	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
STA,LDA,CMP,SBC)		3.	1	1	0	0		PBR,PC+1	IO	1
(11 OpCodes)		4.	1	1	1	0		0,D+DO+X	Byte 0	1/0
(2 bytes)	(1)	4a.	1	1	1	0		0,D+DO+X+1	Bytes1-3	1/0
(4,5,6,7 and 8 cycles)										
10b.Direct,X(R-M-W)-d,x		1.	1	1	1	1		PBR,PC	OpCode	1
(ASL,ROL,LSR,ROR,		2.	1	1	0	1		PBR,PC+1	DO	1
DEC,INC)	(2)	2a.	1	1	0	0		PBR,PC+1	IO	1
(6 OpCodes)		3.	1	1	0	0		PBR,PC+1	IO	1
(2 bytes)		4.	1	0	1	0		0,D+DO+X	Byte 0	1
(6,7,8,9,12 and	(1)	4a.	1	0	1	0		0,D+DO+X+1	Bytes1-3	1
13 cycles)	(3)	5.	1	0	0	0		0,D+DO+X+1	IO	1
	(1)	6a.	1	0	1	0		0,D+DO+X+1	Bytes 3-1	0
		6.	1	0	1	0		0,D+DO+X	Byte 0	0

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
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 OpCodes)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
(2 bytes)		3.	1	1	0	0	PBR,PC+1	IO	1
(4,5,6,7 and 8 cycles)		4.	1	1	1	0	O,D+DO+Y	Byte 0	1/0
	(1)	4a.	1	1	1	0	0,D+DO+Y+1-3	Bytes1-3	1/0
12a.Absolute,X-a,x		1.	1	1	1	1	PBR,PC	OpCode	1
(BIT,LDY,STZ,		2.	1	1	0	1	PBR,PC+1	AAL	1
ORA,AND,EOR,ADC,		3.	1	1	0	1	PBR,PC+2	AAH	1
STA,LDA,CMP,SBC)	(4)	3a.	1	1	0	0	DBR,AAH,AAL+XL	IO	1
(11 OpCodes)		4.	1	1	1	0	DBR,AA+X	Byte 0	1/0
(3 bytes)	(1)	4a.	1	1	1	0	DBR,AA+X+1-3	Bytes1-3	1/0
(4,5,6, 7 and 8 cycles)									
12b.Absolute,X(R-M-W)-a,x		1.	1	1	1	1	PBR,PC	OpCode	1
(ASL,ROL,LSR,ROR,		2.	1	1	0	1	PBR,PC+1	AAL	1
DEC,INC)		3.	1	1	0	1	PBR,PC+2	AAH	1
(6 OpCodes)		4.	1	1	0	0	DBR,AAH,AAL+XL	IO	1
(3 bytes)		5.	1	0	1	0	DBR,AA+X	Byte 0	1
(7,9 and 13 cycles)	(1)	5a.	1	0	1	0	DBR,AA+X+1-3	Bytes 1-3	1
	(3)	6.	1	0	0	0	DBR,AA+X+1or3	IO	1
	(1)	7a.	1	0	1	0	DBR,AA+X+3-1	Bytes 3-1	0
		7.	1	0	1	0	DBR,AA+X	Byte 0	0
*13. Absolute Long,X-al,x		1.	1	1	1	1	PBR,PC	OpCode	1
(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
(8 OpCodes)		4.	1	1	0	1	PBR,PC+3	AAB	1
(4 bytes)	(17)	4a.	1	1	0	0	PBR,PC+3	IO	1
(5,6,7 and 8 cycles)		5.	1	1	1	0	AAB,AA+X	Byte 0	1/0
	(1)	5a.	1	1	1	0	AAB,AA+X+1-3	Bytes1-3	1/0
14. Absolute,Y-a,y		1.	1	1	1	1	PBR,PC	OpCode	1
(LDX,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
(9 OpCodes)	(4)	3a.	1	1	0	0	DBR,AAH,AAL+Y	IO	1
(3 bytes)		4.	1	1	1	0	DBR,AA+Y	Byte 0	1/0
(4,5,6,7 and 8 cycles)	(1)	4a.	1	1	1	0	DBR,AA+Y+1-3	Bytes1-3	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	OFF	1
BCS,BNE,BEQ,BRA)	(5)	2a.	1	1	0	0	PBR,PC+1	IO	1
(9 OpCodes)	(6)	2b.	1	1	0	0	PBR,PC+1	IO	1
(2 bytes)		1.	1	1	1	1	PBR,PC+Offset	OpCode	1
(2,3 and 4 cycles)									
*16. Relative Long-rl		1.	1	1	1	1	PBR,PC	OpCode	1
(BRL)		2.	1	1	0	1	PBR,PC+1	OFF Low	1
(1 OpCode)		3.	1	1	0	1	PBR,PC+2	OFF High	1
(3 bytes)		4.	1	1	0	0	PBR,PC+2	IO	1
(4 cycles)		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.	1	1	0	1	PBR,PC+2	AAH	1
(3 bytes)		4.	1	1	1	0	0,AA	New PCL	1
(5 cycles)		5.	1	1	1	0	0,AA+1	New PCH	1
		1.	1	1	1	1	PBR,NEW PC	OpCode	1

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
*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 OpCode)		3.	1	1	0	1	PBR,PC+2	AAH	1
(3 bytes)		4.	1	1	1	0	0,AA	New PCL	1
(6 cycles		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,		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	0,D+DO	AAL	1
(2 bytes)		4.	1	1	1	0	0,D+DO+1	AAH	1
(5,6,7,8 and 9 cycles)		5.	1	1	1	0	DBR,AA	Byte 0	1/0
	(1)	5a.	1	1	1	0	PBR,AA+1-3	Bytes1-3	1/0
*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	0,D+DO	AAL	1
(2 bytes)		4.	1	1	1	0	0,D+DD+1	AAH	1
(6,7,8,9 and 10 cycles)		5.	1	1	1	0	0,D+DO+2	AAB	1
		6.	1	1	1	0	AAB,AA	Byte 0	1/0
	(1)	6a.	1	1	1	0	AAB,AA+1-3	Bytes1-3	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
		1.	1	1	1	1	PBR,NEW PC	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	0,S	PCH	0
(1 OpCode)		4.	1	1	1	0	0,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
		1.	1	1	1	1	PBR,NEW PC	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	0,S	PBR	0
(4 hardware interrupts)	(10)	4.	1	1	1	0	0,S-1	PCH	0
(0 bytes)	(10)	5.	1	1	1	0	0,S-2	PCL	0
(7 and 8 cycles)	(10)(11)	6.	1	1	1	0	0,S-3	P	0
		7.	0	1	1	0	0,VA	AAVL	1
		8.	0	1	1	0	0,VA+1	AAVH	1
		1.	1	1	1	1	0,AAV	New OpCode	1

ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
21b.Stack (Software	(16)	1.	1	1	1	1	PBR,PC	OpCode	1
Interrupts)-s	(16)(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	PCL	1
(6 and 7 cycles)		6.	1	1	1	0	O,S+3	PCH	1
(different order from	(7)	7.	1	1	1	0	O,S+4	PBR	1
N6502)		1.	1	1	1	1	PBR,PC	New OpCode	1
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 Op Code)		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	NEW PC-1	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)(11)	3a.	1	1	1	0	O,S	Bytes3-1	1
(7 Op Codes)		3.	1	1	1	0	O,S-1-3	Byte0	1
(1 byte)									1
(3,4, and 6 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	Byte 0	1
(1 byte)	(1)	4a.	1	1	1	0	O,S+2-4	Bytes1-3	1
(4,5 and 7 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-1	AAH	0
		6.	1	1	1	0	O,S-1	AAL	0

ADDRESS MODE		CYCLE	$\overline{VP}$	$\overline{ML}$	VDA	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
*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	0,S	AAH	0
(3 bytes)		5.	1	1	1	0	0,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	OFF Low	1
Address)-s		3.	1	1	0	1	PBR,PC+2	OFF High	1
(PER)		4.	1	1	0	0	PBR,PC+2	IO	1
(1 Op Code)		5.	1	1	1	0	0,S	PCH+OFF+	0
(3 bytes)								Carry	
(6 cycles)		6.	1	1	1	0	0,S-1	PCL+OFF	0
*22. Stack Relative-d,s		1.	1	1	1	1	PBR,PC	OpCode	1
(ORA,AND,EOR,ADL,		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	0,S+SO	Byte 0	1/0
(2 bytes)	(1)	4a.	1	1	1	0	0,S+SO+1-3	Bytes1-3	1/0
(4,5 and 7 cycles)									
*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,ADC,STA,LDA,		3.	1	1	0	0	PBR+PC+1	IO	1
CMP,SBC)		4.	1	1	1	0	0,S+SO	AAL	1
(8 Op Codes)		5.	1	1	1	0	0,S+SO+1	AAH	1
(2 bytes)		6.	1	1	0	0	0,S+SO+1	IO	1
(7,8 and 10 cycles)		7.	1	1	1	0	DBR,AA+Y	Byte 0	1/0
	(1)	7a.	1	1	1	0	DBR,AA+Y+1-3	Bytes1-3	1/0
*24a.Block Move Positive		1.	1	1	1	1	PBR,PC	OpCode	1
(forward)-xyc	(18)	2.	1	1	0	1	PBR,PC+1	DBA	1
(MVP)	(18)	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
(5 and 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	(18)	2.	1	1	0	1	PBR,PC+1	DBA	1
x,y Decrement	(18)	3.	1	1	0	1	PBR,PC+2	SBA	1
MVP 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 higher (more	C=1	6.	1	1	0	0	DBA,Y-1	IO	1
positive) than the source		7.	1	1	0	0	DBA,Y-1	IO	1
start address.									
		1.	1	1	1	1	PBR,PC	Op Code	1
FFFFFF	(18)	2.	1	1	0	1	PBR,PC+1	DBA	1
^ _Dest Start	(18)	3.	1	1	0	1	PBR,PC+2	SBA	1
| |	N Byte	4.	1	1	1	0	SBA,X-2	SRC Data	1
| _|_Source Start	Last	5.	1	1	1	0	DBA,Y-2	DEST Data	0
| | |_Dest End	C=0	6.	1	1	0	0	DBA,Y-2	IO	1
| |___Source End		7.	1	1	0	0	DBA,Y-2	IO	1
000000		1.	1	1	1	1	PBR,PC+3	New OpCode	1

```
 ADDRESS MODE               CYCLE VP,ML,VDA,VPA ADDRESS BUS   DATA BUS R/W

*24b.Block Move Negative      |‾1.  1  1  1  1  PBR,PC        OpCode      1
     (backward)-xyc       (18)| 2.  1  1  0  1  PBR,PC+1      DBA         1
     (MVN)                (18)| 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(18)| 2.  1  1  0  1  PBR,PC+1      DBA         1
     x,y Increment        (18)| 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  PBR,PC        OpCode      1
     FFFFF                (18)| 2.  1  1  0  1  PBR,PC+1      DBA         1
     |     _Source End    (18)| 3.  1  1  0  1  PBR,PC+2      SBA         1
     |    |‾            N Byte| 4.  1  1  1  0  SBA,X+2       SRC Data    1
     |   _|_Dest.End       C=0| 5.  1  1  1  0  DBA,Y+2       DEST Data   0
     |  |‾|_Source Start      | 6.  1  1  0  0  DBA,Y+2       IO          1
     V  |___Dest.Start        | 7.  1  1  0  0  DBA,Y+2       IO          1
     000000                   |_1.  1  1  1  1  PBR,PC+3      New OpCode  1
```

1. Add 1 byte (for immediate only) for 16-bit data, add 3 bytes for 32-bit data, add 1 cycle for 16-bit data and 3 cycles for 32-bit data.
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 DBR registers will be updated.
4. Add 1 cycle for indexing across page boundaries, or write, or 16-bit or 32-bit Index Registers. When 8-bit Index Registers 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.
7. Subtract 1 cycle for 6502 emulation mode.
8. Add 1 cycle for REP, SEP.
9. Wait at cycle 2 for 2 cycles after NMI- or IRQ- active input.
10. R/W- remains high during Reset.
11. BRK bit 4 equals "0" in Emulation mode.
12. PHP and PLP.
13. Some OpCodes shown are not on the W65C02.
14. VDA and VPA are not valid outputs on the W65C02 but are valid on the W65C832. The two signals, VDA and VPA, are included to point out the upward compatibility to the W65C832. When VDA and VPA are both a one level, this is equivalent to SYNC being a one level.
15. The PBR is not on the W65C02.
16. Co-processors may monitor the signature byte to aid in processor to co-processor communications.
17. Add 1 cycle for 32-bit Index Register mode.
18. Subtract 2 bytes and 2 cycles when in W65C832 Native mode for MVN and MVP.

AAB	Absolute Address Bank
AAH	Absolute Address High
AAL	Absolute Address Low
AAVH	Absolute Address Vector High
AAVL	Absolute Address Vector Low
Byte 0	Data Byte 0
Bytes 1-3	Data Bytes 1-3
C	Accumulator
D	Direct Register
DBA	Destination Bank Address
DBR	Data Bank Register
DEST	Destination
DO	Direct Offset
ID0	Immediate Data Byte 0
ID1-3	Immediate Data Bytes 1-3
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
S	Stack Address
SBA	Source Bank Address
SRC	Source
SO	Stack Offset
VA	Vector Address
x,y	Index Register

* = New W65C816/802 Addressing Modes
# = New W65C02 Addressing Modes
Blank = NMOS 6502 Addressing Modes

SECTION 7

RECOMMENDED ASSEMBLER SYNTAX STANDARDS

7.1 Directives

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

7.2 Comments

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

7.3 The Source Line

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

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

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

7.3.2.1 Many of the operation codes in Table 6-2 have duplicate mnemonics; when two or more machine language instruction have the same mnemonic, the assembler resolves the difference based on the operand.

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

7.3.2.3 In addition to the mnemonics shown in Table 6-2, an assembler may provide the alternate mnemonics show in Table 7-3-1.

Table 7-3-1 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

7.3.2.4 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.

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

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

7.3.3.2 Table 7-3-2 shows the operand formats which shall be recognized by the assembler. The symbol **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.

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

7.3.3.4 The assembler shall use the <, >, and ^ characters after the # character in immediate address to specify which byte or bytes will be selected from the value of the operand. Any calculations in the operand must be performed before the byte selection takes place. Table 7-3-2 defines the action taken by each operand by showing the effect of the operator on an address. The column that shows a two byte immediate value show the bytes in the order in which they appear in memory. The coding of the operand is for an assembler which uses 32 bit address calculations, showing the way that the address should be reduced to a 24 bit value.

Table 7-3-2 Byte Selection Operator

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

7.3.3.5 In any location in an operand where an address, or expression resulting in an address, can be coded, the assembler shall recognize the prefix characters <, |, and >, which force one byte (direct page), two byte (absolute) or three byte (long absolute) addressing. In cases where the addressing modes is not forced, the assembler shall assume that the address is two bytes unless the assembler is able to determine the type of addressing required by context, in which case that addressing mode will be used. Addresses shall be truncated without error in an addressing mode is forced which does not require the entire value of the address. For example,

LDA $0203 LDA |$010203

are completely equivalent. If the addressing mode is not forced, and the type of addressing cannot be determined from context,t he 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 ahs no direct page indexed by Y) and a short address is used in the operand, the assembler shall automatically extend the address by padding the most significant bytes with zeroes in order to extend the address to the length needed. As with immediate address, any expression evaluation shall take place before the address is selected; thus, the address selection character is only used once, before the address of expression.

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

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

7.3.3.8 The operands of a block move instruction are specified as source bank, destination bank-the opposite order of tzz object bytes generated.

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

SECTION 8

CAVEATS

Table 8-1 W65C816 Compatibility Issues

	W65C816/802	W65C02	NMOS 6502
1. S (Stack)	Always page 1 (E=1), 8 bits; 16 bits when (E=0)	Always page 1,8 bits	Always page 1,8 bits
2. X (X Index Reg)	Indexed page zero always in page 0 (E=1), Cross page (E=0)	Always page 0	Always page 0
3. Y (Y Index Reg)	Indexed page zero always in page 0 (E=1), Cross page (E=0)	Always page 0	Always page 0
4. A (Accumulator)	8 bits (M=1), 16 bits (M=0)	8 bits	8 bits
5. (Flag Reg)	N,V, and Z flags valid in decimal mode. D=0 after reset/interrupt.	N,V, and Zflags valid in dec. mode. D=0 after reset/interrupt	N,V, and Z flags invalid in decimal mode. D=unknown after reset. D not modified after interrupt
6. Timing			
A. ABS,X ASL, LSR, ROL, ROR With No Page Crossing	7 cycles	6 cycles	7 cycles
B. Jump Indirect Operand=XXFF	5 cycles	6 cycles	5 cycles and invalid page crossing
C. Branch Across Page	4 cycles (E=1) 3 cycles (E=0)	4 cycles	4 cycles
D. Decimal Mode	No add. cycle	Add 1 cycle	No add. cycle
7. 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-.
8. 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
9. Memory Lock (ML-)	ML-=0 during Read Modify and Write cycles.	ML-=0 during Modify and Write cycles.	Not available

	W65C816/802	W65C02	NMOS 6502
10. Indexed Across Page Boundary (d),y;a,x; a,y	Extra read of invalid address. (Note 1)	Extra read of last instruction fetch.	Extra read of invalid address.
11. RDY Pulled During Write Cycle	Ignored (E=1) for W65C816 only. Processor stops (E=0).	Processor stops.	Ignored.
12. WAI & STP instruct.	Available	Available	Not available
13. Unused OP Codes	One reserved OP Code specified as WDM will be used in future systems The W65C816 performs a no-operation.	No operation.	Unknown and some "hang up" processor.
14. Bank Address Handling	PBR=00 after reset or interrupts	Not available	Not available
15. R/W- During Read-Modify-Write Instructions	E=1, R/W-=0 during Modify and Write cycles. E=0, R/W-=0 only during Write cycle.	R/W-=0 only during Write cycle.	R/W-=0 during Modify and Write cycles.
16. Pin 7	W65C802=SYNC. W65C816=VPA	SYNC	SYNC
17. COP Instruction Signatures 00-7F defined. Signatures 80-FF reserved	Available	Not available	Not available

8.1 Stack Addressing

When in the Native mode, the Stack may use memory locations 000000 to 00FFFF. The effective address of Stack, Stack Relative, and Stack Relative Indirect Indexed addressing modes will always be within this range. In the Emulation mode, the Stack address range is 000100 to 0001FF. The following opcodes and addressing modes will increment or decrement beyond this range when accessing two or three bytes.

JSL; JSR(a,x); PEA, PEI, PER, PHD, PLD, RTL; d, s; (d,s), y

8.2 Direct Addressing

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

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

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

8.3 Absolute Indexed Addressing

The Absolute Indexed addressing modes are used to address data outside the direct addressing range. The W65C02 and W65C832 addressing range is 0000 to FFFF. Indexing from page FFXX may result in a 00YY data fetch when using the W65C02 or W65C832. In contrast, indexing from page ZZFFXX may result in ZZ+1,00YY when using the W65C832.

8.4 ABORT- Input

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

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

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

8.4.1.3 IRQ-, NMI-, ABORT- BRK, COP: When ABORT- is asserted after cycle 2, PBR and DBR will become 00 (Emulation mode) or PBR will become 00 (Native mode).

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

8.5 VDA and VPA Valid Memory Address Output Signals

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

8.6 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.

8.7 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 when the PHI2 clock or RDY is low.

8.8 M/X Output

The M/X output reflects the valid 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 M/X output is invalid during the instruction cycle following REP, SEP and PLP instruction execution. This cycle is used as the opcode fetch cycle of the next instruction.

8.9 All Opcodes Function in All Modes of Operation

8.9.1 It should be noted that all opcodes function in all modes of operation. However, some instructions and addressing modes are intended for W65C832 24-bit addressing and are therefore less useful for the W65C832. The following is a list of instructions and addressing modes which are primarily intended for W65C832 use:

JSL;RTL;[d];[d],y;JMP al;JML;al,al,x

8.9.2 The following instructions may be used with the W65C832 even though a Bank Address is not multiplexed on the Data Bus:

PHK;PHB;PLB

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

8.9.3.1 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).

8.9.3.2 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 W65C832, and 0000 to 00FF for the W65C832.

8.10 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.

8.11 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.

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

8.12.1 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.

8.12.2 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.

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

8.13 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.

8.14 WAI Instruction

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

8.15 The STP instruction disables the PHI2 clock to all circuitry. When disabled, the PHI2 clock is held in the high state. In this case, the Dta 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 RES- 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 RES- input. Note that the external oscillator must be stable and operating properly before RES- goes high.

8.16 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.

8.17 WDM Opcode Use

The WDM opcode will be used on future microprocessors.

8.18 RDY Pulled During Write

The NMOS 6502 does not stop during a write operation. In contrast, both the W65C02 and the W65C832 do stop during write operations. The W65C832 stops during a write when in the Native mode, but does not stop when in the Emulation mode.

8.19 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).

8.20 Interrupt Priorities

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

RES-	Highest Priority
ABORT-	
NMI-	
IRQ-	Lowest Priority

8.21 Transfers from differing register sizes

All transfers from one register to another will result in a full 32-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:

TAS;TSA;TAD;TDA

8.22 Stack Transfers

When in the W65C02 Emulation mode, a 01 is forced into the high byte of the 16-bit stack pointer. When in the Native mode or W65C816 Emulation mode, the A Accumulator is transferred to the 16-bit stack pointer. Note that in both the Emulation and Native modes, the full 16 bits of the Stack Register are transferred to the A Accumulator regardless of the state of the M bit in the Status Register.

8.23 REP/SEP

WDC had problems using the REP and SEP instructions in early versions of the high-speed W65C816 and W65C802 devices and has been corrected on all W65C832 devices.