990975 990976

VL65C816•VL65C802

CMOS 16-BIT MICROPROCESSOR FAMILY

FEATURES

- Advanced CMOS design for low power consumption and increased noise immunity
- Single 3 - 6 V power supply, 5 V specified
- Emulation mode allows complete hardware and software compatibility with 6502 designs
- 24-bit address bus allows access to 16M Bytes of memory space
- Full 16-bit ALU, Accumulator, Stack Pointer, and Index Registers
- Valid Data Address (VDA) and Valid Program Address (VPA) output allows dual cache and cycle steal DMA implementation
- Vector Pull (VP) output indicates when interrupt vectors are being addressed
- May be used to implement vectored interrupt design
- $\overline{\text{ABORT}}$ input and associated vector supports interrupting any instruction without modifying internal registers
- Separate program and data bank registers allow program segmentation
- New Direct Register allows "zero page" addressing anywhere in first 64K bytes
- 24 addressing modes: 13 original 6502 modes plus 11 new addressing modes, with 91 instructions using 255 opcodes
- New Wait for Interrupt (WAI) and Stop the Clock (STP) instructions further reduce power consumption, decrease interrupt latency and allow synchronization with external events
- New Co-Processor instruction (COP) with associated vector supports co-processor configurations (i.e., floating point processors)

DESCRIPTION

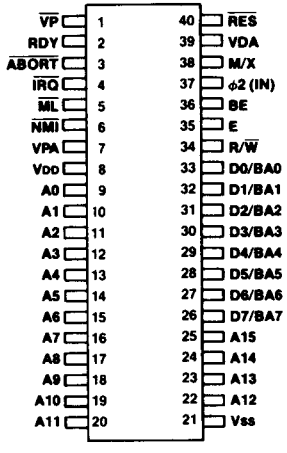
The VLSI VL65C802 and VL65C816 are CMOS 16-bit microprocessors featuring total software compatibility with their 8-bit NMOS and CMOS 6500-series predecessors. The VL65C802 is pin-for-pin compatible with 8-bit devices currently available, while the VL65C816 extends addressing to a full 16 megabytes. These devices offer the many advantages of CMOS technology, including increased noise immunity, higher reliability, and greatly reduced power requirements. A software switch determines whether the processor is in the 8-bit "emulation" mode or in the "native" mode, thus allowing existing systems to use the expanded features.

The Accumulator, ALU, X and Y Index registers, and Stack Pointer Register have all been extended to 16 bits. A new 16-bit Direct PageRegister augments the Direct Page addressing mode (formerly Zero Page addressing). Separate Program Bank and Data Bank Registers allow 24-bit memory addressing.

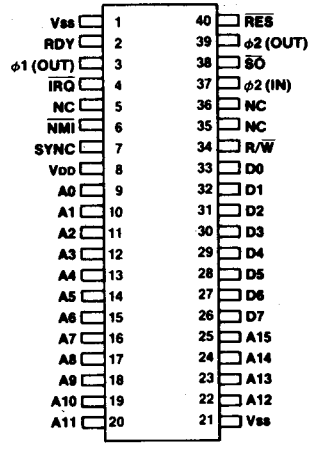
Four new signals provide the system designer with many options. The $\overline{\text{ABORT}}$ input can interrupt the currently executing instruction without modifying internal registers. Valid Data Address (VDA) and Valid Program Address (VPA) outputs facilitate dual cache memory by indicating whether a data segment or program segment is accessed. Modifying a vector is made easy by monitoring the Vector Pull (VP) output.

PIN DIAGRAMS

VL65C816



VL65C802



ORDER INFORMATION

Part Number	Clock Frequency	Package
VL65C802-02PC	2 MHz	Plastic DIP
VL65C802-02CC		Ceramic DIP
VL65C816-02PC		Plastic DIP
VL65C816-02CC		Ceramic DIP
VL65C802-04PC	4 MHz	Plastic DIP
VL65C802-04CC		Ceramic DIP
VL65C816-04PC		Plastic DIP
VL65C816-04CC		Ceramic DIP
VL65C802-06PC	6 MHz	Plastic DIP
VL65C802-06CC		Ceramic DIP
VL65C816-06PC		Plastic DIP
VL65C816-06CC		Ceramic DIP
VL65C802-08PC	8 MHz	Plastic DIP
VL65C802-08CC		Ceramic DIP
VL65C816-08PC		Plastic DIP
VL65C816-08CC		Ceramic DIP

BLOCK DIAGRAM

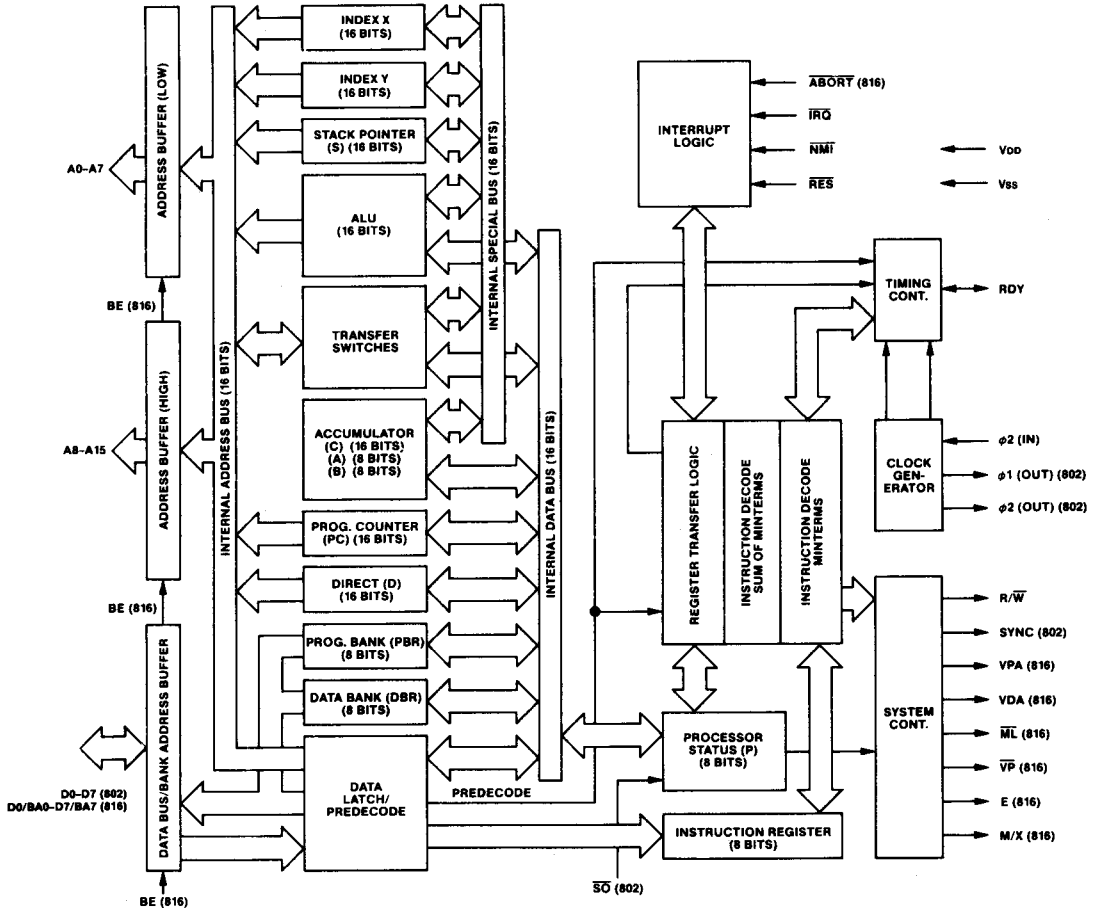


FIGURE 1. STATUS REGISTER

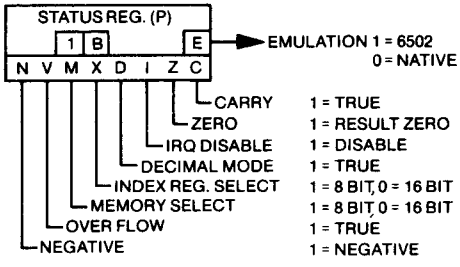
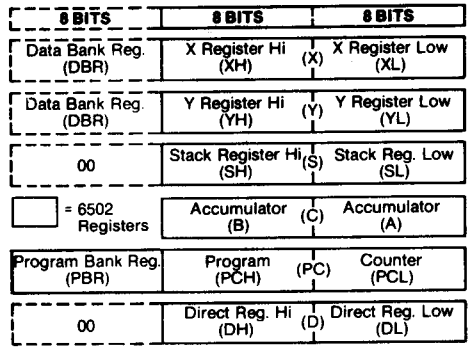


FIGURE 2. PROGRAMMING MODEL



SIGNAL DESCRIPTIONS

Signal Name	Signal Description
Abort ($\overline{ABORT}$) (VL65C816)	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). Since $\overline{ABORT}$ is an edge-sensitive input, an Abort occurs whenever there is a negative pulse (or level) on the $\overline{ABORT}$ line during a phase 2 clock.
Address Bus (A0-A15)	These 16 output lines form the Address Bus for memory and I/O exchange on the Data Bus.. When using the VL65C816, the address lines may be set to the high-impedance state by the Bus Enable (BE) signal.
Bus Enable (BE) (VL65C816)	The Bus Enable input signal allows external control of the Address and Data Buffers, as well as the R/$\overline{W}$ signal. With Bus Enable high, the R/$\overline{W}$ 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.
Data Bus (D0-D7) (VL65C802)	The eight Data Bus lines provide an 8-bit bidirectional Data Bus for use during data exchange between the microprocessor and external memory or peripherals. Two memory cycles are required for the transfer of 16-bit values.
Data/Address Bus (D0/BA0-D7/BA7) (VL65C816)	These eight lines multiplex address bits BA0-BA7 with the data value. The address is present during the first half of a memory cycle, and the data value is read or written during the second half of the memory cycle. Two memory cycles are required to transfer 16-bit values. These lines may be set to the high impedance state by the BE signal.
Emulation Status (E) (VL65C816)	The Emulation Status output reflects the state of the Emulation (E) Mode flag in the Processor Status (P) Register. This signal may be thought of as an op code extension and used for memory and system management.
Interrupt Request ($\overline{IRQ}$)	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 is recognized immediately. The Interrupt Request vector address is 00FFFE,F (Emulation Mode) or 00FFEE,F (Native mode). Since $\overline{IRQ}$ is a level-sensitive input, an interrupt occurs if the interrupt source was not cleared since the last interrupt. Also, no interrupt occurs if the interrupt source is cleared prior to interrupt recognition.
Memory Lock ($\overline{ML}$) (VL65C816)	The Memory Lock output may be used to ensure the integrity of Read-Modify-Write instructions in a multiprocessor system. Memory Lock indicates the need to defer arbitration of the next bus cycle. Memory Lock is low during the last three or five cycles of ASL, DEC, INC, LSR, ROL, ROR, TRB, and TSB memory referencing instructions, depending on the state of the M flag.
Memory/Index Select Status (M/X) (VL65C816)	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.
Non-Maskable Interrupt ($\overline{NMI}$)	A negative transition on the $\overline{NMI}$ 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 (Emulation Mode) or 00FFEA,B (Native Mode). Since $\overline{NMI}$ is an edge-sensitive input, an interrup occurs if there is a negative transition while servicing a previous interrupt. Also, no interrupt occurs if $\overline{NMI}$ remains low.
Phase 1 Out [ø1 (OUT)] (VL65C802)	This inverted clock output signal provides timing for external read and write operations. Executing the Stop (STP) instruction holds this clock in the low state.
Phase 2 In [ø2 (IN)]	This is the system clock input to the microprocessor internal clock generator (equivalent to ø0 (IN) on the 6502). During the low-power Standby Mode, ø2 (IN) should be held in the high state to preserve the contents of internal registers.
Phase 2 Out [ø2 (OUT)] (VL65C802)	This clock output signal provides timing for external read and write operations. Addresses are valid after the address setup time (Tads), following the negative transition of ø2 (Out). Executing the (STP) instruction holds ø2 (Out) in the high state.

SIGNAL DESCRIPTIONS (CONTINUED)

Signal Name	Signal Description
Read/Write (R/$\overline{W}$)	When the R/$\overline{W}$ 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 that is to be stored at the addressed memory location. When using the VL65C816, the R/$\overline{W}$ signal may be set to the high impedance state by Bus Enable (BE).
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 (note that when in the Emulation Mode, the VL65C802 stops only during a read cycle). Returning RDY to the active high state allows the microprocessor to continue following the next Phase 2 in clock negative transition. The RDY signal is internally pulled low following the execution of a Wait for Interrupt (WAI) instruction, and then returned to the high state when a $\overline{RES}$, $\overline{ABORT}$, $\overline{NMI}$, or $\overline{IRQ}$ external interrupt is provided. This feature may be used to eliminate interrupt latency by placing the WAI instruction at the beginning of the $\overline{IRQ}$ servicing routine. If the $\overline{IRQ}$ Disable Flag has been set, the next instruction is executed when the $\overline{IRQ}$ occurs. The processor does stop after a WAI instruction if RDY has been forced to a high state. The Stop (STP) instruction has no effect on RDY.
Reset ($\overline{RES}$)	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 $\overline{RES}$ signal must be held low for a least two clock cycles after VDD reaches operating voltage. Ready (RDY) has no effect while $\overline{RES}$ is being held low. During this Reset conditioning period, the following processor initialization takes place:

Registers

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

	N	V	M	X	D	I	Z	C/E	
P =	*	*	1	1	0	1	*	*/1	* = Not Initialized

STP and WAI instructions are cleared.

Signals

E	= 1	VDA	= 0
M/$\overline{X}$	= 1	$\overline{VP}$	= 1
R/$\overline{W}$	= 1	VPA	= 0
SYNC	= 0		

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

Set Overflow ($\overline{SO}$) (VL65C802)	A negative transition on this input sets the Overflow (V) Flag, bit 6 of the Processor Status (P) Register.
Synchronize (SYNC) (VL65C802)	The SYNC output is provided to identify those cycles during which the microprocessor is fetching an opcode. The SYNC signal is high during an opcode fetch cycle, and when combined with Ready (RDY), can be used for single instruction execution.
Valid Data Address (VDA) and Valid Program Address (VPA) (VL65C816)	These two output signals indicate the type of memory being accessed by the address bus. The following coding applies:

VDA	VPA	
0	0	Internal operation - Address and Data Bus available.
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	Op code fetch - May be used for program cache control and single step control.

VDD and VSS	VDD is the positive supply voltage and VSS is system logic ground. Pin 21 of the two VSS pins on the VL65C802 should be used for system ground.
Vector Pull ($\overline{VP}$) (VL65C816)	The Vector Pull output indicates that a vector location is being addressed during an interrupt sequence. $\overline{VP}$ is low during the last two interrupt sequence cycles, during which time the processor reads the interrupt vector. The $\overline{VP}$ signal may be used to select and prioritize interrupts from several sources by modifying the vector addresses.

FUNCTIONAL DESCRIPTION

The VL65C802 offers the design engineer the opportunity to utilize both existing software programs and hardware configurations, while also achieving the added advantages of increased register lengths and faster execution times. The VL65C802 "ease of use" design and implementation features provide the designer with increased flexibility and reduced implementation costs. In the Emulation Mode, the VL65C802 not only offers software compatibility, but is also hardware (pin-for-pin) compatible with 6502 designs. It provides the advantages of 16-bit internal operation in 6502-compatible applications. The VL65C802 is an excellent direct replacement microprocessor for 6502 designs.

The VL65C816 provides the design engineer with upward mobility and software compatibility in applications in which a 16-bit system configuration is desired. The VL65C816 16-bit hardware configuration, coupled with current software, allows a wide selection of system applications. In the Emulation Mode, the VL65C816 offers many advantages, including full software compatibility with 6502 coding. In addition, the powerful VL65C816 instruction set and addressing modes make it an excellent choice for new 16-bit designs.

Internal organization of the VL65C802 and VL65C816 can be divided into two parts: 1) the Register Section, and 2) the Control Section. Instructions (or op codes) 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. Both the VL65C802 and VL65C816 have a 16-bit internal architecture with an 8-bit external data bus.

INSTRUCTION REGISTER
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.

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.

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

INTERNAL REGISTERS (Refer to figure 2, Programming Model.)

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

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 mulitplexed with the data value and is present on the Data/Address lines during the first half of a data transfer memory cycle for the VL65C816. The Data Bank Register is initialized to zero during Reset.

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 contents. The Direct Register is initialized to zero during Reset.

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

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

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

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.

PROGRAM COUNTER (PC)
The 16-bit Program Counter Register provides the addresses that 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.

STACK POINTER (S)
The Stack Pointer is a 16-bit register that 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.

PIN DESCRIPTIONS

Pin	Description
A0-A15	Address Bus
$\overline{ABORT}$	Abort Input
BE	Bus Enable
ϕ2 (IN)	Phase 2 In Clock
ϕ1 (OUT)	Phase 1 Out Clock
ϕ2 (OUT)	Phase 2 Out Clock
D0-D7	Data Bus
D0/BA0-D7/BA7	Data Bus, Multiplexed
E	Emulation Select
$\overline{IRQ}$	Interrupt Request
$\overline{ML}$	Memory Lock
M/X	Mode Select (P_M or P_X)

Pin	Description
NC	No Connection
$\overline{NMI}$	Non-Maskable Interrupt
RDY	Ready
$\overline{RES}$	Reset
R/$\overline{W}$	Read/Write
$\overline{SO}$	Set Overflow
SYNC	Synchronize
VDA	Valid Data Address
$\overline{VP}$	Vector Pull
VPA	Valid Program Address
V_{DD}	Positive Power Supply (+5 Volts)
V_{SS}	Internal Logic Ground

TABLE 1. COMPATIBILITY ISSUES

	65C816/802	65C02	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 Register)	Indexed page zero always in page 0 (E = 1), Cross page (E = 0).	Always page 0	Always page 0
3. Y (Y Index Register)	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. P (Flag Registor)	N, V, and Z flags valid in decimal mode. D = 0 after reset or interrupt.	N, V, and Z flags valid in decimal mode. D = 0 after reset and 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 additional cycle	Add 1 cycle	No additional 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 ($\overline{ML}$)	$\overline{ML}$ = 0 during Read, Modify and Write cycles.	$\overline{ML}$ = 0 during Modify and Write.	Not available
10. Indexed Across Page Boundary (d),y; a,x; a,y	Extra read of invalid address.	Extra read of last instruction fetch.	Extra read of invalid address.
11. RDY Pulled During Write Cycle.	Ignored (E = 1) for 65C802 only. Processor stops (E = 0).	Processor stops	Ignored
12. WAI and STP Instructions.	Available	Available	Not available
13. Unused OP Codes	One reserved OP Code specified as WDM will be used in future systems. The 65C816 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/$\overline{W}$ During Read-Modify-Write Instructions	E = 1, R/$\overline{W}$ = 0 during Modify and Write cycles. E = 0, R/$\overline{W}$ = 0 only during Write cycle.	R/$\overline{W}$ = 0 only during Write cycle	R/$\overline{W}$ = 0 during Modify and Write cycles.
16. Pin 7	65C802 = SYNC. 65C816 = VPA	SYNC	SYNC
17. COP Instruction Signatures 00-7F user defined Signatures 80-FF reserved	Available	Not available	Not available

TABLE 2. INSTRUCTION SET – ALPHABETICAL SEQUENCE

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift One Bit Left, Memory or Accumulator
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 Address
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 (or Push Immediate Data on Stack)
PEI	Push Effective Indirect Address on Stack (or Push Direct Data 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 Stack
PLD	Pull Direct Register from Stack
PLP	Pull Processor Status from Stack
PLX	Pull Index X from Stack
PLY	Pull Index Y form 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
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
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 Bits

For alternate mnemonics, see Table 7.

TABLE 3. VECTOR LOCATIONS

E = 1		E = 0	
OOFFFE,F	—$\overline{IRQ}$/BRK Hardware/Software	OOFFEE,F	—$\overline{IRQ}$ Hardware
OOFFFC,D	—$\overline{RESET}$ Hardware	OOFFEC,D	—(Reserved)
OOFFFA,B	—$\overline{NMI}$ Hardware	OOFFEA,B	—$\overline{NMI}$ Hardware
OOFFF8,9	—$\overline{ABORT}$ Hardware	OOFFE8,9	—$\overline{ABORT}$ Hardware
OOFFF6,7	—(Reserved)	OOFFE6,7	—BRK Software
OOFFF4,5	—COP Software	OOFFE4,5	—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 4. OPCODE MATRIX

MSD	LSD 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 65C816/802 Opcodes ● = New 65C02 Opcodes Blank = NMOS 6502 Opcodes	ADDRESSING MODE
BASE NO. BYTES		BASE NO. CYCLES

TABLE 5.

	OPERATION	#	a	al	d	A	i	(d),y	[d],y	(d,x)	d,x	d,y	a,x	al,x	a,y	r	rl	(s)	(d)	[d]	(s,x)	s	d,s	(d,s),y	xyc	PROCESSOR STATUS CODE 7 N	6 V	5 M	4 X	3 D	2 I	1 Z	0 C	MNE-MONIC E = 0
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N	V	1	B	D	I	Z	C	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													M7	M6	.	.	.	.	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)																82					00				.	.	.	•	0	I	.	.	BRK
BRL	BRANCH LONG ALWAYS																									.	.	.	.	.	.	.	.	★ 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 → I						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												★ 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			7C												★ JML
JMP	JUMP TO NEW LOC.		4C	5C														6C																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 BACKWARD																								54	.	.	.	.	.	.	.	★ MVN
MVP	M → M FORWARD																								44	.	.	.	.	.	.	.	★ MVP
NOP	NO OPERATION						EA																											NOP
ORA	A∨M → 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																					F4												★ PEA
	S - 2 → S																																	
PEI	M(d), M(d + 1) → Ms - 1, Ms																					D4												★ PEI
	S - 2 → S																																	
PER	Mpc + rl, Mpc + rl + 1 → Ms - 1, Ms																					62												★ PER
	S - 2 → S																																	
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	MAP → 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 - C̄ → 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																											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																									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

TABLE 6. DETAILED INSTRUCTION OPERATION

	ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
1.	Immediate # (LDY,CPY,CPX,LDX,ORA, AND,EOR,ADC,BIT,LDA, CMP,SBC REP,SEP) (14 Op Codes) (2 and 3 bytes) (2 and 3 cycles)	(1)(8)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	IDL	1
			2a.	1	1	0	1	PBR,PC+2	IDH	1
2a.	Absolute a (BIT,STY,STZ,LDY, CPY,CPX,STX,LDX, ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (18 Op Codes) (3 bytes) (4 and 5 cycles)	(1)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
			4.	1	1	1	0	DBR,AA	Data Low	1/0
			4a.	1	1	1	0	DBR,AA+1	Data High	1/0
2b.	Absolute (R-M-W) a (ASL,ROL,LSR,ROR DEC,INC,TSB,TRB) (6 Op Codes) (3 bytes) (6 and 8 cycles)	(1)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
			4.	1	0	1	0	DBR,AA	Data Low	1
		(1)	4a.	1	0	1	0	DBR,AA+1	Data High	1
		(3)	5.	1	0	0	0	DBR,AA+1	IO	1
		(1)	6a.	1	0	1	0	DBR,AA+1	Data High	0
			6.	1	0	1	0	DBR,AA	Data Low	0
2c.	Absolute (JUMP) a (JMP)(4C) (1 Op Code) (3 bytes) (3 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	NEW PCL	1
			3.	1	1	0	1	PBR,PC+2	NEW PCH	1
			1.	1	1	1	1	PBR, NEW PC	Op Code	1
2d.	Absolute (Jump to subroutine) a (JSR) (1 Op Code) (3 bytes) (6 cycles) (different order from N6502)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	NEW PCL	1
			3.	1	1	0	1	PBR,PC+2	NEW PCH	1
			4.	1	1	0	0	PBR,PC+2	IO	1
			5.	1	1	1	0	0,S	PCH	0
			6.	1	1	1	0	0,S-1	PCL	0
			1.	1	1	1	1	PBR,NEW PC	Next Op Code	1
*3a.	Absolute Long al (ORA,AND,EOR,ADC STA,LDA,CMP,SBC) (8 Op Codes) (4 bytes) (5 and 6 cycles)	(1)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
			4.	1	1	0	1	PBR,PC+3	AAB	1
			5.	1	1	1	0	AAB,AA	Data Low	1/0
			5a.	1	1	1	0	AAB,AA+1	Data High	1/0
*3b.	Absolute Long (JUMP) al (JMP) (1 Op Code) (4 bytes) (4 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	NEW PCL	1
			3.	1	1	0	1	PBR,PC+2	NEW PCH	1
			4.	1	1	0	1	PBR,PC+3	NEW BR	1
			1.	1	1	1	1	NEW PBR,PC	Op Code	1
*3c.	Absolute Long (Jump to Subroutine Long) al (JSL) (1 Op Code) (4 bytes) (7 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	NEW PCL	1
			3.	1	1	0	1	PBR,PC+2	NEW PCH	1
			4.	1	1	1	0	0,S	PBR	0
			5.	1	1	0	0	0,S	IO	1
			6.	1	1	1	1	PBR,PC+3	NEW PBR	1
			7.	1	1	1	0	0,S-1	PCH	0
			8.	1	1	1	0	0,S-2	PCL	0
			1.	1	1	1	1	NEW PBR,PC	Next Op Code	1
4a.	Direct d (BIT,STZ,STY,LDY, CPY,CPX,STX,LDX, ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (18 Op Codes) (2 bytes) (3,4 and 5 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
		(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	0,D+DO	Data Low	1/0
		(1)	3a.	1	1	1	0	0,D+DO+1	Data High	1/0
4b.	Direct (R-M-W) d (ASL,ROL,LSR,ROR DEC,INC,TSB,TRB) (6 Op Codes) (2 bytes) (5,6,7 and 8 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
		(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	0	1	0	0,D+DO	Data Low	1
		(1)	3a.	1	0	1	0	0,D+DO+1	Data High	1
		(3)	4.	1	0	0	0	0,D+DO+1	IO	1
		(1)	5a.	1	0	1	0	0,D+DO+1	Data High	0
			5.	1	0	1	0	0,D+DO	Data Low	0
5.	Accumulator A (ASL,INC,ROL,DEC,LSR,ROR) (6 Op Codes) (1 byte) (2 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	0	PBR,PC+1	IO	1
6a.	Implied i (DEY, INY, INX, DEX, NOP, XCE, TYA, TAY,TXA, TXS, TAX,TSX,TCS,TSC,TCD, TDC,TXY,TYX,CLC,SEC, CLI,SEI,CLV,CLD,SED) (25 Op Codes) (1 byte) (2 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	0	PBR,PC+1	IO	1
*6b.	Implied i (XBA) (1 Op Code) (1 byte) (3 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	PBR,PC+1	IO	1
* 6c.	Wait For Interrupt (WAI) (1 Op Code) (1 byte) (3 cycles)	(9)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	0	1	PBR,PC+1	IO	1
			3.	1	1	0	0	0	PBR,PC+1	IO	1
		IRQ,NMI	1.	1	1	1	1	1	PBR,PC+1	IRQ(BRK)	1
* 6d.	Stop-The-Clock (STP) (1 Op Code) (1 byte) (3 cycles)		1.	1	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	0	1	PBR,PC+1	IO	1
		RES=1	3.	1	1	0	0	1	PBR,PC+1	IO	1
		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 (Hardware interrupt)		1.	1	1	1	1	1	PBR,PC+1	BEGIN	1
7.	Direct Indirect Indexed (d),y (ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (8 Op Codes) (2 bytes) (5,6,7 and 8 cycles)	(2)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
			2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	1	0	0,D+DO	AAL	1
			4.	1	1	1	0	0,D+DO+1	AAH	1
		(4)	4a.	1	1	1	0	DBR,AAH,AAL+	YL IO	1
			5.	1	1	1	0	DBR,AA+Y	Data Low	1/0
		(1)	5a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
8.	Direct Indirect Indexed Long [d],y (ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (8 Op Codes) (2 bytes) (6,7 and 8 cycles)	(2)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
			2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	1	0	0,D+DO	AAL	1
			4.	1	1	1	0	0,D+DO+1	AAH	1
			5.	1	1	1	0	0,D+DO+2	AAB	1
			6.	1	1	1	0	AAB,AA+Y	Data Low	1/0
		(1)	6a.	1	1	1	0	AAB,AA+Y+1	Data High	1/0
9.	Direct Indexed Indirect (d,x) (ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (8 Op Codes) (2 bytes) (6,7 and 8 cycles)	(2)	1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
			2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	PBR,PC+1	IO	1
			4.	1	1	1	0	0,D+DO+X	AAL	1
			5.	1	1	1	0	0,D+DO+X+1	AAH	1
			6.	1	1	1	0	DBR,AA	Data Low	1/0
		(1)	6a.	1	1	1	0	DBR,AA+1	Data High	1/0
10a.	Direct,X d,x (BIT,STZ,STY,LDY, ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (11 Op Codes) (2 bytes) (4,5 and 6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
		(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	PBR,PC+1	IO	1
			4.	1	1	1	0	0,D+DO+X	Data Low	1/0
		(1)	4a.	1	1	1	0	0,D+DO+X+1	Data High	1/0
10b.	Direct,X(R-M-W) d,x (ASL,ROL,LSR,ROR, DEC,INC) (6 Op Codes) (2 bytes) (6,7,8 and 9 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
		(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	PBR,PC+1	IO	1
			4.	1	0	1	0	0,D+DO+X	Data Low	1
		(1)	4a.	1	0	1	0	0,D+DO+X+1	Data High	1
		(3)	5.	1	0	0	0	0,D+DO+X+1	IO	1
		(1)	6a.	1	0	1	0	0,D+DO+X+1	Data High	0
			6.	1	0	1	0	0,D+DO+X	Data Low	0
11.	Direct,Y d,y (STX,LDX) (2 Op Codes) (2 bytes) (4,5 and 6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	DO	1
		(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
			3.	1	1	0	0	PBR,PC+1	IO	1
			4.	1	1	1	0	0,D+DO+Y	Data Low	1/0
		(1)	4a.	1	1	1	0	0,D+DO+Y+1	Data High	1/0
12a.	Absolute,X a,x (BIT,LDY,STZ, ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (11 Op Codes) (3 bytes) (4,5 and 6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
		(4)	3a.	1	1	0	0	DBR,AAH,AAL+	XL IO	1
			4.	1	1	1	0	DBR,AA+X	Data Low	1/0
		(1)	4a.	1	1	1	0	DBR,AA+X+1	Data High	1/0
12b.	Absolute,X(R-M-W) a,x (ASL,ROL,LSR,ROR, DEC,INC) (6 Op Codes) (3 bytes) (7 and 9 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
			4.	1	1	0	0	DBR,AAH,AAL+	XL IO	1
			5.	1	0	1	0	DBR,AA+X	Data Low	1
		(1)	5a.	1	0	1	0	DBR,AA+X+1	Data High	1
		(3)	6.	1	0	0	0	DBR,AA+X+1	IO	1
		(1)	7a.	1	0	1	0	DBR,AA+X+1	Data High	0
			7.	1	0	1	0	DBR,AA+X	Data Low	0
*13.	Absolute Long,X al,x (ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (8 Op Codes) (4 bytes) (5 and 6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
			4.	1	1	0	1	PBR,PC+3	AAB	1
			5.	1	1	1	0	AAB,AA+X	Data Low	1/0
		(1)	5a.	1	1	1	0	AAB,AA+X+1	Data High	1/0
14.	Absolute,Y a,y (LDX,ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (9 Op Codes) (3 bytes) (4,5 and 6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
			3.	1	1	0	1	PBR,PC+2	AAH	1
		(4)	3a.	1	1	0	0	DBR,AAH,AAL+	YL IO	1
			4.	1	1	1	0	DBR,AA+Y	Data Low	1/0
		(1)	4a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
15.	Relative r (BPL,BMI,BVC,BVS,BCC, BCS,BNE,BEQ,BRA) (9 Op Codes) (2 bytes) (2,3 and 4 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	Offset	1
		(5)	2a.	1	1	0	0	PBR,PC+1	IO	1
		(6)	2b.	1	1	0	0	PBR,PC+1	IO	1
			1.	1	1	1	1	PBR,PC+Offset	Op Code	1

TABLE 6. DETAILED INSTRUCTION OPERATION (CONT.)

ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
★16. Relative Long rl (BRL) (1 Op Code) (3 bytes) (4 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	Offset Low	1
		3.	1	1	0	1	PBR,PC+2	Offset High	1
		4.	1	1	0	0	PBR,PC+2	IO	1
		1.	1	1	1	1	PBR,PC+Offset	Op Code	1
17a. Absolute Indirect (a) (JMP) (1 Op Code) (3 bytes) (5 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	AAL	1
		3.	1	1	0	1	PBR,PC+2	AAH	1
		4.	1	1	1	0	0,AA	NEW PCL	1
		5.	1	1	1	0	0,AA+1	NEW PCH	1
		1.	1	1	1	1	PBR,NEW PC	Op Code	1
★17b. Absolute Indirect (a) (JML) (1 Op Code) (3 bytes) (6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	AAL	1
		3.	1	1	0	1	PBR,PC+2	AAH	1
		4.	1	1	1	0	0,AA	NEW PCL	1
		5.	1	1	1	0	0,AA+1	NEW PCH	1
		6.	1	1	1	0	0,AA+2	NEW PBR	1
		1.	1	1	1	1	NEW PBR,PC	Op Code	1
● 18. Direct Indirect (d) (ORA,AND,EOR,ADC, STA,LDA,CMP,SBC) (8 Op Codes) (2 bytes) (5,6 and 7 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DO	1
	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	1	0	0,D+DO	AAL	1
		4.	1	1	1	0	0,D+DO+1	AAH	1
		5.	1	1	1	0	DBR,AA	Data Low	1/0
	(1)	5a.	1	1	1	0	DBR,AA+1	Data Low	1/0
★19. Direct Indirect Long [d] (ORA,AND,EOR,ADC STA,LDA,CMP,SBC) (8 Op Codes) (2 bytes) (6,7 and 8 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DO	1
	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	1	0	0,D+DO	AAL	1
		4.	1	1	1	0	0,D+DO+1	AAH	1
		5.	1	1	1	0	0,D+DO+2	AAB	1
		6.	1	1	1	0	AAB,AA	Data Low	1/0
	(1)	6a.	1	1	1	0	AAB,AA+1	Data High	1/0
20a. Absolute Indexed Indirect (a,x) (JMP) (1 Op Code) (3 bytes) (6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	AAL	1
		3.	1	1	0	1	PBR,PC+2	AAH	1
		4.	1	1	0	0	PBR,PC+2	IO	1
		5.	1	1	0	1	PBR,AA+X	NEW PCL	1
		6.	1	1	0	1	PBR,AA+X+1	NEW PCH	1
		1.	1	1	1	1	PBR, NEW PC	Op Code	1
★20b. Absolute Indexed Indirect (Jump to Subroutine Indexed Indirect) (a,x) (JSR) (1 Op Code) (3 bytes) (8 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	AAL	1
		3.	1	1	1	0	0,S	PCH	0
		4.	1	1	1	0	0,S-1	PCL	0
		5.	1	1	0	1	PBR,PC+2	AAH	1
		6.	1	1	0	0	PBR,PC+2	IO	1
		7.	1	1	0	1	PBR,AA+X	NEW PCL	1
		8.	1	1	0	1	PBR,AA+X+1	NEW PCH	1
		1.	1	1	1	1	PBR,NEW PC	Next Op Code	1
21a. Stack (Hardware interrupts) s (IRQ,NMI,ABORT,RES) (4 hardware interrupts) (0 bytes) (7 and 8 cycles)		1.	1	1	1	1	PBR,PC	IO	1
	(3)	2.	1	1	0	0	PBR,PC	IO	1
	(7)	3.	1	1	1	0	0,S	PBR	0
		4.	1	1	1	0	0,S-1	PCH	0
		5.	1	1	1	0	0,S-2	PCL	0
		6.	1	1	1	0	0,S-3	P	0
		7.	0	1	1	0	0,VA	AAVL	1
		8.	0	1	1	0	0,VA+1	AAVH	1
		1.	1	1	1	1	0,AAV	Next Op Code	1
21b. Stack (Software interrupts) s (BRK,COP) (2 Op Codes) (2 bytes) (7 and 8 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
	(3)	2.	1	1	0	1	PBR,PC+1	Signature	1
	(7)	3.	1	1	1	0	0,S	PBR	0
		4.	1	1	1	0	0,S-1	PCH	0
		5.	1	1	1	0	0,S-2	PCL	0
		6.	1	1	1	0	0,S-3 (COP Latches)	P	0
		7.	0	1	1	0	0,VA	AAVL	1
		8.	0	1	1	0	0,VA+1	AAVH	1
		1.	1	1	1	1	0,AAV	Next Op Code	1
21c. Stack (Return from Interrupt) s (RTI) (1 Op Code) (1 byte) (6 and 7 cycles) (different order from N6502)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	0	PBR,PC+1	IO	1
	(3)	3.	1	1	0	0	PBR,PC+1	IO	1
		4.	1	1	1	0	0,S+1	P	1
		5.	1	1	1	0	0,S+2	PCL	1
		6.	1	1	1	0	0,S+3	PCH	1
	(7)	7.	1	1	1	0	0,S+4	PBR	1
		1.	1	1	1	1	PBR,PC	New Op Code	1
21d. Stack (Return from Subroutine) s (RTS) (1 Op Code) (1 byte) (6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	0	0	PBR,PC+1	IO	1
		4.	1	1	1	0	0,S+1	PCL	1
		5.	1	1	1	0	0,S+2	PCH	1
		6.	1	1	1	0	0,S+2	IO	1
		1.	1	1	1	1	PBR,PC	Op Code	1
★21e. Stack (Return from Subroutine Long) s (RTL) (1 Op Code) (1 byte) (6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	0	0	PBR,PC+1	IO	1
		4.	1	1	1	0	0,S+1	NEW PCL	1
		5.	1	1	1	0	0,S+2	NEW PCH	1
		6.	1	1	1	0	0,S+3	NEW PBR	1
		1.	1	1	1	1	NEW PBR,PC	Next Op Code	1

ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
21f. Stack (Push) s (PHP,PHA,PHY,PHX, PHD,PHK,PHB) (7 Op Codes) (1 byte) (3 and 4 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	0	PBR,PC+1	IO	1
	(1)	3a.	1	1	1	0	0,S	Register High	1
		3.	1	1	1	0	0,S-1	Register Low	1
21g. Stack (Pull) s (PLP,PLA,PLY,PLX,PLD,PLB) (Different than N6502) (6 Op Codes) (1 byte) (4 and 5 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	0	0	PBR,PC+1	IO	1
		4.	1	1	1	0	0,S+1	Register Low	1
	(1)	4a.	1	1	1	0	0,S+2	Register High	1
★21h. Stack (Push Effective Indirect Address) s (PEI) (1 Op Code) (2 bytes) (6 and 7 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DO	1
	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
		3.	1	1	1	0	0,D+DO	AAL	1
		4.	1	1	1	0	0,D+DO+1	AAH	1
		5.	1	1	1	0	0,S	AAH	0
		6.	1	1	1	0	0,S-1	AAL	0
★21i. Stack (Push Effective Absolute Address) s (PEA) (1 Op Code) (3 bytes) (5 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	AAL	1
		3.	1	1	0	1	PBR,PC+2	AAH	1
		4.	1	1	1	0	0,S	AAH	0
		5.	1	1	1	0	0,S-1	AAL	0
★21j. Stack (Push Effective Program Counter Relative Address) s (PER) (1 Op Code) (3 bytes) (6 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	Offset Low	1
		3.	1	1	0	1	PBR,PC+2	Offset High	1
		4.	1	1	0	0	PBR,PC+2	IO	1
		5.	1	1	1	0	0,S	PCH+OFF+ CARRY	0
		6.	1	1	1	0	0,S-1	PCL+OFFSET	0
★22. Stack Relative d,s (ORA,AND,EOR,ADL, STA,LDA,CMP,SDC) (8 Op Codes) (2 bytes) (4 and 5 cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	SO	1
		3.	1	1	0	0	PBR,PC+1	IO	1
		4.	1	1	1	0	0,S+SO	Data Low	1/0
	(1)	4a.	1	1	1	0	0,S+SO+1	Data High	1/0
★23. Stack Relative Indirect Indexed (d,s),y (ORA,AND,EOR,ADC, STA,LDA,CMP,SDC) (8 Op Codes) (2 bytes) (7 and 8 Cycles)		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	SO	1
		3.	1	1	0	0	PBR+PC+1	IO	1
		4.	1	1	1	0	0,S+SO	AAL	1
		5.	1	1	1	0	0,S+SO+1	AAH	1
		6.	1	1	0	0	0,S+SO+1	IO	1
		7.	1	1	1	0	DBR,AA+Y	Data Low	1/0
	(1)	7a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
★24a. Block Move Positive (forward) xyc (MVP) (1 Op Code) (3 bytes) (7 cycles) x = Source Address y = Destination c =Number of Bytes to Move -1 x,y Decrement MVP is used when the destination start address is higher (more positive) than the source start address.		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N-2	4.	1	1	1	0	SBA,X	Source Data	1
	Byte	5.	1	1	1	0	DBA,Y	Dest. Data	0
	C=2	6.	1	1	0	0	DBA,Y	IO	1
		7.	1	1	0	0	DBA,Y	IO	1
		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N-1	4.	1	1	1	0	SBA,X-1	Source Data	1
	Byte	5.	1	1	1	0	DBA,Y-1	Dest. Data	0
	C=1	6.	1	1	0	0	DBA,Y-1	IO	1
		7.	1	1	0	0	DBA,Y-1	IO	1
FFFFFF Dest. Start, Source Start, Dest. End, Source End 000000		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N Byte	4.	1	1	1	0	SBA,X-2	Source Data	1
	Last	5.	1	1	1	0	DBA,Y-2	Dest. Data	0
	C=0	6.	1	1	0	0	DBA,Y-2	IO	1
		7.	1	1	0	0	DBA,Y-2	IO	1
		1.	1	1	1	1	PBR,PC+3	Next Op Code	1
★24b. Block Move Negative (backward) xyc (MVN) (1 Op Code) (3 bytes) (7 cycles) x = Source Address y = Destination c = Number of Bytes to Move -1 x,y Increment MVN is used when the destination start address is lower (more negative) than the source start address.		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N-2	4.	1	1	1	0	SBA,X	Source Data	1
	Byte	5.	1	1	1	0	DBA,Y	Dest. Data	0
	C=2	6.	1	1	0	0	DBA,Y	IO	1
		7.	1	1	0	0	DBA,Y	IO	1
		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N-1	4.	1	1	1	0	SBA,X+1	Source Data	1
	Byte	5.	1	1	1	0	DBA,Y+1	Dest. Data	0
	C=1	6.	1	1	0	0	DBA,Y+1	IO	1
		7.	1	1	0	0	DBA,Y+1	IO	1
FFFFFF Source End, Dest.End, Source Start, Dest. Start 000000		1.	1	1	1	1	PBR,PC	Op Code	1
		2.	1	1	0	1	PBR,PC+1	DBA	1
		3.	1	1	0	1	PBR,PC+2	SBA	1
	N Byte	4.	1	1	1	0	SBA,X+2	Source Data	1
	C=0	5.	1	1	1	0	DBA,Y+2	Dest. Data	0
		6.	1	1	0	0	DBA,Y+2	IO	1
		7.	1	1	0	0	DBA,Y+2	IO	1
		1.	1	1	1	1	PBR,PC+3	Next Op Code	1

TABLE 5. NOTES

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 65C816/802 Instructions
   ● = New 65C02 Instructions
   Blank = NMOS 6502

+ Add
- Subtract
∧ AND
V OR
⊻ Exclusive OR
•

TABLE 6. NOTES

Notes:
(1) Add 1 byte (for immediate only) for M=0 or X=0 (i.e. 16 bit data), add 1 cycle for M=0 or X=0.
(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 X=0. When X=1 or in the emulation mode, this cycle contains invalid addresses.
(5) Add 1 cycle if branch is taken.
(6) Add 1 cycle if branch is taken across page boundaries in 6502 emulation mode (E=1).
(7) Subtract 1 cycle for 6502 emulation mode (E=1).
(8) Add 1 cycle for REP,SEP.
(9) Wait at cycle 2 for 2 cycles after $\overline{NMI}$ or $\overline{IRQ}$ active input.

Abbreviations:
AAB Absolute Address Bank
AAH Absolute Address High
AAL Absolute Address Low
AAVH Absolute Address Vector High
AAVL Absolute Address Vector Low
C Accumulator
D Direct Register
DBA Destination Bank Address
DBR Data Bank Register
DO Direct Offset
IDH Immediate Data High
IDL Immediate Data Low
IO Internal Operation
P Status Register
PBR Program Bank Register
PC Program Counter
R-M-W Read-Modify-Write
S Stack Address
SBA Source Bank Address
SO Stack Offset
VA Vector Address
x,y Index Registers
★ = New 65C816/802 Addressing Modes
● = New 65C02 Addressing Modes
Blank = NMOS 6502 Addressing Modes

RECOMMENDED ASSEMBLER SYNTAX STANDARDS

DIRECTIVES

Assembler directives are those parts of the assembly language source program that 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.

COMMENTS

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

THE SOURCE LINE

Any line that causes the generation of a single VL65C816 or VL65C802 machine language instruction should be divided into four fields: a label field, the operation code, the operand, and the comment field.

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, as 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, as long as their use does not conflict with the coding of operand fields.

The Operation Code Field - The operation code consists of a three-character sequence (mnemonic) from table 2. It starts no sooner than column two of the line, or one space after the label if a label is coded.

Many of the operation codes in table 2 have duplicate mnemonics; when two or more machine language instructions have the same mnemonic, the assembler resolves the difference based on the operand.

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

In addition to the mnemonics shown in table 2, an assembler may provide the alternative mnemonics shown in table 7.

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

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 24-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 are recognized by the fact that they start with alphabetic characters. Decimal numbers are recognized as containing only the decimal digits 0 through 9. Hexadecimal constants shall be recognized by prefixing the constant with a dollar sign ($) character, followed by zero or more of either the decimal digits or the hexadecimal digits A through F. If lower case letters are allowed in the label field, then they are also allowed as hexadecimal digits.

All constants, no matter what their format, provide at least enough precision to specify all values that can be represented by a 24-bit signed or unsigned integer represented in two's complement notation.

Table 9 shows the operand formats that are recognized by the assembler. The symbol **d** is a label or value that 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 $10000; the symbol **al** is a label or value that the assembler can recognize as being greater than **$FFFF**. The symbol EXT is a label that cannot be located by the assembler at the time the instruction is assembled. Unless instructed otherwise, an assembler assumes that EXT labels are two bytes long. The symbols r and rl are 8- and 16-bit signed displacements calculated by the assembler.

Note that the operand does not determine whether or not immediate addressing 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 must allow separate settings for the accumulator and index registers.

The assembler shall use the <, >, and ^ characters after the # character in an immediate address to specify which byte or bytes are to be selected from the value of the operand. Any calculations in the operand must be performed before the byte selection takes place. Table 8 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 shows the bytes in the order in which they appear in memory. The coding of the operand is for an asembler that uses 32 bit address calculations, showing the way that the address should be reduced to a 24 bit value.

In any location in an operand in which an address, or expression resulting in an address, can be coded, the assembler recognizes the prefix characters <, |, and >, which force one-byte (direct page), two-byte (absolute) or three-byte (long absolute) addressing. In cases in which the addressing mode 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 is used. Addresses are truncated without error if an addressing mode is forced that 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, the assembler assumes that a two-byte address is to be used. If an instruction does not have a short addressing mode (as in LDA, which has no direct page indexed by Y) and a short address is used in the operand, the assembler automatically extends the address by padding the most significant bytes with zeroes in order to extend the address to the length needed. As with immediate addressing, any expression evaluation takes place before the address is selected; thus, the address selection character is only used once, before the address of expression.

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

A long indirect address is indicated in the operand field of an instruction by surrounding the direct page address where the indirect address is found by square brackets; direct page addresses that contains 16-bit addresses are indicated by being surrounded by parentheses.

The operands of a block move instruction are specified as source bank, destination band (the opposite order of the object bytes generated).

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

TABLE 7. ALTERNATIVE 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

TABLE 8. BYTE SELECTION OPERATOR

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

TABLE 9. 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	(d),y
Indexed	(<d),y
	(<a),y
	(<al),y
	(<EXT),y
Direct Indirect	[d],y
Indexed Long	[<d],y
	[<a],y
	[<al],y
	[<EXT],y
Direct Indexed	(d,x)
Indirect	(<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

Addressing Mode	Format	
Absolute Indexed by Y	!d,y	
	d,y	
	a,y	
	!a,y	
	!al,y	
	!EXT,y	
	EXT,y	
Absolute Long Indexed	>d,x	
by X	>a,x	
	>al,x	
	al,x	
	>EXT,x	
Program Counter	d	(the assembler calculates
Relative and	a	r and rl)
Program Counter	al	
Relative Long	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	(d,s),y	
Indirect Indexed	(<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 10. ADDRESSING MODE SUMMARY

Address Mode	Instruction Times In Memory Cycles		Memory Utilization in Number of Program Sequence Bytes	
	Original 8 Bit NMOS 6502	New 65C816	Original 8 Bit NMOS 6502	New 65C816
1. Immediate	2	2(3)	2	2(3)
2. Absolute	4(5)	4(3,5)	3	3
3. Absolute Long	—	5(3)	—	4
4. Direct	3(5)	3(3,4,5)	2	2
5. Accumulator	2	2	1	1
6. Implied	2	2	1	1
7. Direct Indirect Indexed (d),y	5(1)	5(1,3,4)	2	2
8. Direct Indirect Indexed Long [d], y	—	6(3,4)	—	2
9. Direct Indexed Indirect (d,x)	6	6(3,4)	2	2
10. Direct, X	4(5)	4(3,4,5)	2	2
11. Direct, Y	4	4(3,4)	2	2
12. Absolute, X	4(1,5)	4(1,3,5)	3	3
13. Absolute Long, X	—	5(3)	—	4
14. Absolute, Y	4(1)	4(1,3)	3	3
15. Relative	2(1,2)	2(2)	2	2
16. Relative Long	—	3(2)	—	3
17. Absolute Indirect (Jump)	5	5	3	3
18. Direct Indirect	—	5(3,4)	—	2
19. Direct Indirect Long	—	6(3,4)	—	2
20. Absolute Indexed Indirect (Jump)	—	6	—	3
21. Stack	3-7	3-8	1-3	1-4
22. Stack Relative	—	4(3)	—	2
23. Stack Relative Indirect Indexed	—	7(3)	—	2
24. Block Move X, Y, C (Source, Destination, Block Length)	—	7	—	3

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

ADDRESSING PREFACE

The VL65C816 is capable of directly addressing 16M Bytes of memory. This address space has special significance within certain addressing modes.

RESET AND INTERRUPT VECTORS

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

STACK

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

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).

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.

DATA ADDRESS SPACE

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

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

The following addressing modes are available for use in the VL65C802 and VL65C816 microprocessors. The "long" addressing modes may be used with the VL65C802; however, the high byte of the address is not available to the hardware. Detailed descriptions of the 24 addressing modes are given in the following section.

ADDRESSING MODES

1. Immediate Addressing—#

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

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. Absolute Long—al

The second, third, and fourth byte of the instruction form the 24-bit effective address.

Instruction:	opcode	addrl	addrh	baddr
Operand Address:	baddr	addrh	addrl	

4. Direct—d

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

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

5. Accumulator—A

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

6. Implied—i

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

7. Direct Indirect Indexed—(d),y

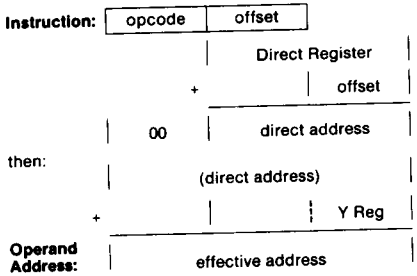
This address mode is often referred to as Indirect,Y. The second byte of the instruction is added to the Direct Register (D). The 16-bit contents of this memory location is then combined with the Data Bank register to form a 24-bit base address. The Y Index Register is added to the base address to form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

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

8. Direct Indirect Long Indexed—[d],y

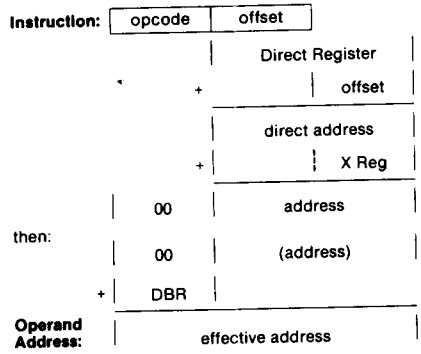
With this addressing mode, the 24-bit base address is pointed to by the sum of the second byte of the instruction and the Direct Register. The effective address is this 24-bit base address plus the Y Index Register.

ADDRESSING MODES (Cont.)



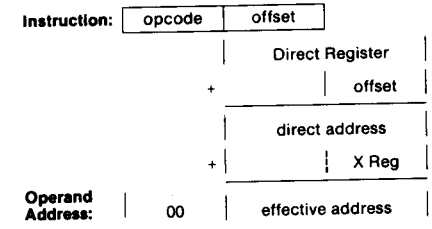
9. Direct Indexed Indirect—(d,x)

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.



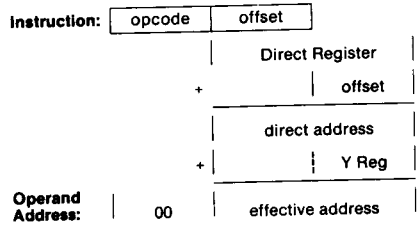
10. Direct Indexed With X—d,x

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.



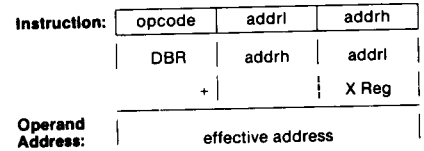
11. Direct Indexed With Y—d,y

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.



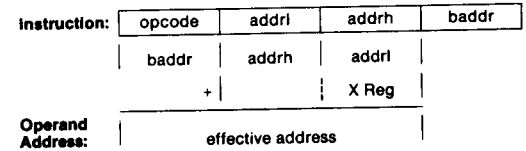
12. Absolute Indexed With X—a,x

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.



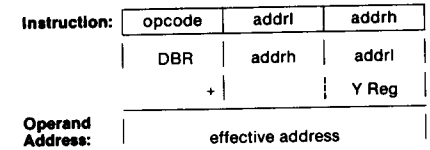
13. Absolute Long Indexed With X—al,x

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.



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.



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.

16. Program Counter Relative Long—rl

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

ADDRESSING MODES (Cont.)

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)

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)
+	DBR	
Operand Address:	effective address	

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 Address:	(direct address)	

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)

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.

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
	+	offset
Operand Address:	00	effective address

23. Stack Relative Indirect Indexed—(d,s),y

The second byte of the instruction is added to the Stack Pointer to form a pointer to the low-order 16-bit base address in Bank 0. The Data Bank Register contains the high-order 8 bits of the base address. The effective address is the sum of the 24-bit base address and the Y Index Register.

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

24. Block Source Bank, Destination Bank—xyc

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

Instruction:	opcode	dstbnk	srcbnk
		dstbnk →	DBR
Source Address:		scrbnk	X Reg
Destination Address:		DBR	Y Reg

Increment (MVN) or decrement (MVP) X and Y.

Decrement C (if greater than zero), then PC+3 → PC.

TABLE 11. VL65C802 TIMING CHARACTERISTICS TA = 0°C to 70°C, VDD = 5.0 V ± 5%

		2 MHz		4 MHz		6 MHz		8 MHz		
Parameter	Symbol	Min	Max	Min	Max	Min	Max	Min	Max	Unit
Cycle Time	tCYC	500	DC	250	DC	167	DC	125	DC	nS
Clock Pulse Width Low	tPWL	0.240	10	0.120	10	0.080	10	0.060	10	μS
Clock Pulse Width High	tPWH	240	∞	120	∞	80	∞	60	∞	nS
Fall Time, Rise Time	tF, tR	—	10	—	10	—	5	—	5	nS
Delay Time, φ2 (IN) to φ1 (OUT)	tDφ1	—	20	—	20	—	20	—	20	nS
Delay Time, φ2 (IN) to φ2 (OUT)	tDφ2	—	40	—	40	—	40	—	40	nS
Address Hold Time	tAH	10	—	10	—	10	—	10	—	nS
Address Setup Time	tADS	—	100	—	75	—	60	—	40	nS
Access Time	tACC	365	—	130	—	87	—	70	—	nS
Read Data Hold Time	tDHR	10	—	10	—	10	—	10	—	nS
Read Data Setup Time	tDSR	40	—	30	—	20	—	15	—	nS
Write Data Delay Time	tMDS	—	100	—	70	—	60	—	40	nS
Write Data Hold Time	tDHW	10	—	10	—	10	—	10	—	nS
Processor Control Setup Time	tPCS	40	—	30	—	20	—	15	—	nS
Processor Control Hold Time	tPCH	10	—	10	—	10	—	10	—	nS
Capacitive Load (Address, Data, and R/W)	CEXT	—	100	—	100	—	35	—	35	pF

FIGURE 3. VL65C802 TIMING DIAGRAM

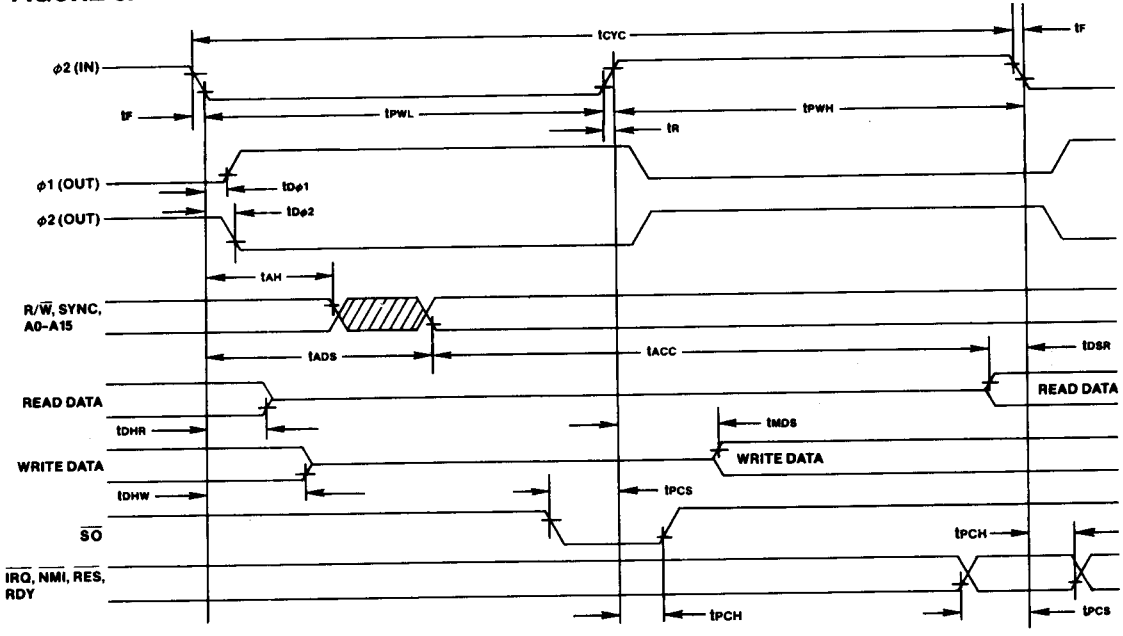
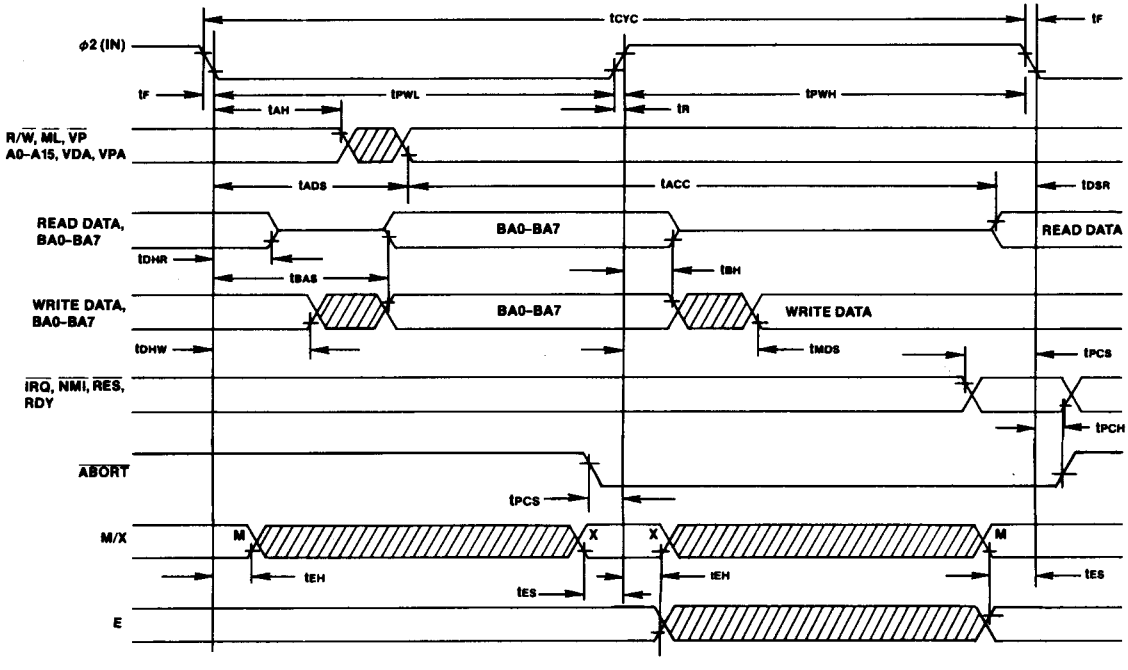


TABLE 12. VL65C816 TIMING CHARACTERISTICS TA = 0°C to 70°C, VDD = 5.0 V ± 5%

		2 MHz		4 MHz		6 MHz		8 MHz		
Parameter	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
Cycle Time	tCYC	500	DC	250	DC	167	DC	125	DC	nS
Clock Pulse Width Low	tPWL	0.240	10	0.120	10	0.080	10	0.060	10	μS
Clock Pulse Width High	tPWH	240	∞	120	∞	80	∞	60	∞	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	—	100	—	75	—	60	—	40	nS
BA0-BA7 Hold Time	tBH	10	—	10	—	10	—	10	—	nS
BA0-BA7 Setup Time	tBAS	—	100	—	90	—	65	—	45	nS
Access Time	tACC	365	—	130	—	87	—	70	—	nS
Read Data Hold Time	tDHR	10	—	10	—	10	—	10	—	nS
Read Data Setup Time	tDSR	40	—	30	—	20	—	15	—	nS
Write Data Delay Time	tMDS	—	100	—	70	—	60	—	40	nS
Write Data Hold Time	tDHW	10	—	10	—	10	—	10	—	nS
Processor Control Setup Time	tPCS	40	—	30	—	20	—	15	—	nS
Processor Control Hold Time	tPCH	10	—	10	—	10	—	10	—	nS
E,MX Output Hold Time	tEH	10	—	10	—	5	—	5	—	nS
E,MX Output Setup Time	tES	50	—	50	—	25	—	15	—	nS
Capacitive Load (Address, Data, and R/W)	CEXT	—	100	—	100	—	35	—	35	pF
BE to High Impedance State	tBHZ	—	30	—	30	—	30	—	30	nS
BE to Valid Data	tBVD	—	30	—	30	—	30	—	30	nS

FIGURE 4. VL65C816 TIMING DIAGRAM



USER INFORMATION

STACK ADDRESSING

When in the Native mode, the Stack Register may use memory locations 000000 to 00FFFF. The effective address of Stack, Stack Relative and Stack Relative Indirect Indexed addressing modes is always within this range. In the Emulation mode, the Stack address range is 000100 to 0001FF. The following opcodes and addressing modes 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

DIRECT ADDRESSING

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 are always 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 increment from 0000FE or 0000FF into Stack area.

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 increment from 00DHFE or 00DHFF into the next higher page.

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

ABSOLUTE INDEXED ADDRESSING (VL65C816 ONLY)

The Absolute Indexed addressing modes are used to address data outside the Direct addressing range. The VL65C02 and VL65C802 addressing range is 0000 to FFFF. Indexing from page FFXX may result in a 00YY data fetch when using the VL65C02 or VL65C802. In contrast, indexing from page ZZFFXX may result in ZZ+1,00YY when using the VL65C816.

$\overline{\text{ABORT}}$ INPUT (VL65C816 ONLY)

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

- Read-Modify-Write: Processor Status Register modified if $\overline{\text{ABORT}}$ is asserted after a modify cycle.
- RTI: Processor Status Register modified if $\overline{\text{ABORT}}$ is asserted after cycle 3.
- $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$, $\overline{\text{ABORT}}$ BRK, COP: When $\overline{\text{ABORT}}$ is asserted after cycle 2, PBR and DBR become 00 (Emulation mode) or PBRbecomes 00 (Native mode).

The Abort Interrupt has been designed for virtual memory systems. For this reason, asynchronous $\overline{\text{ABORT}}$s may cause undesirable results due to the above conditions.

VDA AND VPA (VL65C816 ONLY)

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 that read memory when the Index Register consists of eight 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.

APPLE II, IIe, IIc, AND II+ DISK SYSTEMS (VL65C816 ONLY)

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

DB/BA OPERATION (WHEN RDY IS PULLED LOW -VL65C816 ONLY)

When RDY is low, the Data Bus is held in the data transfer state (i.e., ø2 high). The Bank address external transparent latch should be latched when the ø2 clock or RDY is low.

M/X OUTPUT (VL65C816 ONLY)

The M/X output reflects the value of the M and X bits of the processor Status Register. The REP, SEP, and PLP instructions may change the state of the M and X bits. Note that the 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.

INSTRUCTIONS

Opcodes - It should be noted that all opcodes function in all modes of operation. However, some instructions and addressing modes are intended for VL65C816 24-bit addressing and are therefore less useful for the VL65C802. The following instructions and addressing modes are primarily intended for VL65C816 use:

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

The following instructions may be used with VL65C802 even though a Bank Address is not multiplexed on the Data Bus:

PHK; PHB; PLB

The following instructions have limited use in the Emulation mode:

- 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 are always high (logic 1).
- 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 VL65C816, and 0000 to 00FF for the VL65C802.

USER INFORMATION (CONT.)

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.

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 bytes of the Accumulator (A and B) are not affected by a mode change.

How hardware interrupts, BRK, and COP instructions affect the Program Bank and the Data Bank Registers-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, the previous PBR contents are automatically saved.

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

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.

WAI Instruction-The WAI instruction pulls RDY low and places the processor in the WAI low-power mode. $\overline{NMI}$, $\overline{IRQ}$, or RESET terminate the WAI condition and transfer control to the interrupt handler routine. Note that an $\overline{ABORT}$ input aborts the WAI instruction, but does not restart the processor. When the Status Register 1 flag is set ($\overline{IRQ}$ disabled), the $\overline{IRQ}$ interrupt causes the next instruction (following the WAI instruction) to be executed without going to the $\overline{IRQ}$ interrupt handler. This method results in the highest speed response to an $\overline{IRQ}$ input. When an interrupt is received after an $\overline{ABORT}$ that occurs during the WAI instruction, the processor return to the WAI instruction. Other than $\overline{RES}$ (highest priority), $\overline{ABORT}$ is the next-highest priority, followed by $\overline{NMI}$ or $\overline{IRQ}$ interrupts.

STP Instruction-The STP instruction disables the ø2 clock to all circuitry. When disabled, the ø2 clock is held in the high state. In this case, the Data Bus remains in the data transfer state and the Bank address is not multiplexed onto the Data Bus. Upon executing the STP instruction, the $\overline{RES}$ signal is the only input that can restart the processor. The processor is restarted by enabling the ø2 clock, which occurs on the falling edge of the $\overline{RES}$ input. Note that the external oscillator must be stable and operating properly before $\overline{RES}$ goes high.

COP Signatures-Signatures 00-7F may be user defined, while signatures 80-FF are reserved.

RDY Pulled During Write-The NMOS 6502 does not stop during a write operation. In contrast, both the VL65C02 and the VL65C816 do stop during write operations. The VL65C802 stops during a write when in the Native mode, but does not stop when in the Emulation mode.

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).

INTERRUPTS

Interrupt Priorities-The following interrupt priorities are in effect should more than one interrupt occur at the same time:

$\overline{RES}$	Highest
$\overline{ABORT}$	
$\overline{NMI}$	
$\overline{IRQ}$	Lowest

TRANSFERS

Transfers from 8-Bit to 16-Bit, or 16-Bit to 8-Bit, Registers - All transfers from one register to another result in a full 16-bit output from the source register. The Destination Register size determines the number of bits actually stored in the destination register and the values stored in the processor Status Register. The following are always 16-bit transfers, regardless of the accumulator size:

TCS; TSC; TCD; TDC

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

ABSOLUTE MAXIMUM RATINGS

Ambient Operating Temperature	0°C to +70°C
Storage Temperature	-55°C to +150°C
Supply Voltage to Ground Potential	-0.3 V to +7.0 V
Applied Input Voltage	-0.3 V to VDD + 0.3 V

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of this device under these or any conditions other than those indicated in this data sheet is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

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.

DC CHARACTERISTICS: TA = 0°C to +70°C, VDD = 5 V ± 5%

Parameter	Symbol	Min	Max	Unit
Input High Voltage RES, RDY, IRQ, Data, SO, BE, ϕ2 (IN), NMI, ABORT	V_{IH}	 2.0 0.7 V_{DD}	 V_{DD} + 0.3 V_{DD} + 0.3	 V V
Input Low Voltage RES, RDY, IRQ, Data, SO, BE, ϕ2 (IN), NMI, ABORT	V_{IL}	 -0.3 -0.3	 0.8 0.2	 V V
Input Leakage Current (V_{IN} = 0 to V_{DD}) RES, NMI, RDY, IRQ, SO, BE, ABORT (Internal Pullup) ϕ2 (IN) Address, Data, R/W (Off State, BE = 0)	I_{IN}	 -100 -1 -10	 1 1 10	 μA μA μA
Output High Voltage (I_{OH} = -100μA) SYNC, Data, Address, R/W, ML, VP, M/X, E, VDA, VPA, ϕ1 (OUT), ϕ2 (OUT)	V_{OH}	 0.7 V_{DD}	 —	 V
Output Low Voltage (I_{OL} = 1.6mA) SYNC, Data, Address, R/W, ML, VP, M/X, E, VDA, VPA, ϕ1 (OUT), ϕ2 (OUT)	V_{OL}	 —	 0.4	 V
Supply Current (No Load)	I_{DD}		4	mA/MHz
Standby Current (No Load, Data Bus =V_{SS} or V_{DD} RES, NMI, IRQ, SO, BE, ABORT, ϕ2 = V_{DD})	I_{SB}	—	10	μA
Capacitance (V_{IN} = 0V, T_A = 25°C, f = 2 MHz) Logic, ϕ2 (IN) Address, Data, R/W (Off State)	 C_{IN} C_{TS}	 — —	 10 15	 pF pF