CMD

Microcircuits

G65SC802
G65SC816

CMOS 8/16-Bit Microprocessor Family

Features

- Advanced CMOS design for low power consumption and increased noise immunity
- Emulation mode for total software compatibility with 6502 designs
- Full 16-bit ALU, Accumulator, Stack Pointer, and Index Registers
- Direct Register for "zero page" addressing
- 24 addressing modes (including 13 original 6502 modes)
- Wait for Interrupt (WAI) and Stop the Clock (STP) instructions for reduced power consumption and decreased interrupt latency
- 91 instructions with 255 opcodes
- Co-Processor (COP) instruction and associated vector
- Powerful Block Move instructions

Features (G65SC802 Only)

- 8-Bit Mode with both software and hardware (pin-to-pin) compatibility with 6502 designs (64 KByte memory space)
- Program selectable 16-bit operation
- Choice of external or on-board clock generation

Features (G65SC816 Only)

- Full 16-bit operation with 24 address lines for 16 MByte memory
- Program selectable 8-Bit Mode for 6502 coding compatibility.
- Valid Program Address (VPA) and Valid Data Address (VDA) outputs for dual cache and DMA cycle steal implementation
- Vector Pull ($\overline{VP}$) output indicates when interrupt vectors are being fetched. May be used for vectoring/prioritizing interrupts
- Abort interrupt and associated vector for interrupting any instruction without modifying internal registers
- Memory Lock ($\overline{ML}$) for multiprocessor system implementation

General Description

The G65SC802 and G65SC816 are ADV-CMOS (ADVanced CMOS) 16-bit microprocessors featuring total software compatibility with 8-bit NMOS and CMOS 6500 series microprocessors. The G65SC802 is pin-to-pin compatible with 8-bit 6502 devices currently available, while also providing full 16-bit internal operation. The G65SC816 provides 24 address lines for 16 MByte addressing, while providing both 8-bit and 16-bit operation.

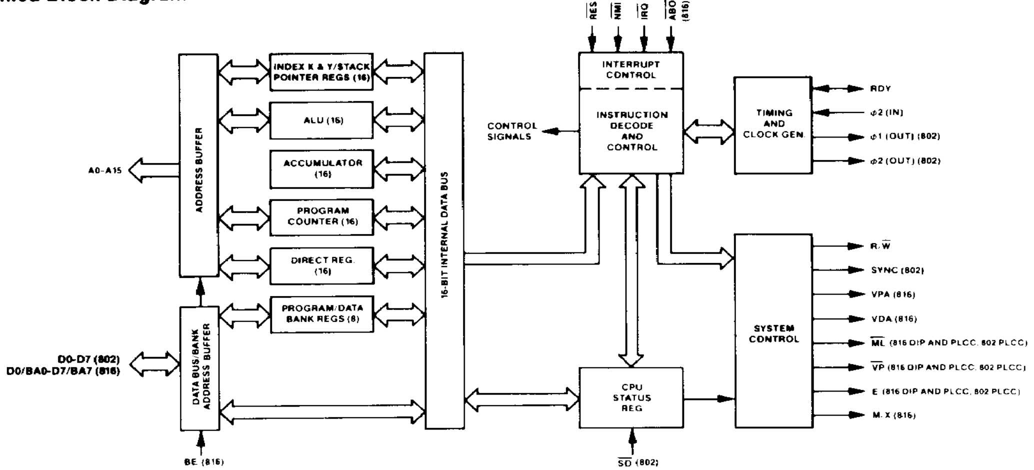
Each microprocessor contains an Emulation (E) mode for emulating 8-bit NMOS and CMOS 6500-Series microprocessors. A software switch determines whether the processor is in the 8-bit emulation mode or in the Native 16-bit mode. This allows existing 8-bit system designs to use the many powerful features of the G65SC802 and G65SC816.

The G65SC802 and G65SC816 provide the system engineer with many powerful features and options. A 16-bit Direct Page Register is provided to augment the Direct Page addressing mode, and there are separate Program Bank Registers for 24-bit memory addressing. Other valuable features include:

- An Abort input which can interrupt the current instruction without modifying internal registers.
- Valid Data Address (VDA) and Valid Program Address (VPA) outputs which facilitate dual cache memory by indicating whether a data or program segment is being accessed.
- Vector modification by simply monitoring the Vector Pull ($\overline{VP}$) output.
- Block Move instructions.

CMD Microcircuits' G65SC802 and G65SC816 microprocessors offer the design engineer a new freedom of design and application, and the many advantages of state-of-the-art ADV-CMOS technology.

Simplified Block Diagram



This is advance information and specifications are subject to change without notice.

Absolute Maximum Ratings: (Note 1)

Rating	Symbol	Value
Supply Voltage	V_{DD}	-0.3V to +7.0V
Input Voltage	V_{IN}	-0.3V to V_{DD} +0.3V
Operating Temperature	T_A	0°C to +70°C
Storage Temperature	T_S	-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 cause permanent damage. Functional operation under these conditions is not implied.

DC Characteristics (All Devices): V_{DD} = 5.0V ±5%, V_{SS} = 0V, T_A = 0°C to +70°C

Parameter	Symbol	Min	Max	Unit
Input High Voltage $\overline{RES}$, RDY, $\overline{IRQ}$, Data, $\overline{SO}$, BE $\overline{ABORT}$, $\overline{NMI}$, φ2 (IN)	V_{IH}	 2.0 0.7 V_{DD}	 V_{DD} + 0.3 V_{DD} + 0.3	 V V
Input Low Voltage $\overline{RES}$, RDY, $\overline{IRQ}$, Data, $\overline{SO}$, BE $\overline{ABORT}$, $\overline{NMI}$, φ2 (IN)	V_{IL}	 -0.3 -0.3	 0.8 0.2	 V V
Input Leakage Current (V_{IN} = 0 to V_{DD}) $\overline{RES}$, $\overline{NMI}$, $\overline{IRQ}$, $\overline{SO}$, BE, $\overline{ABORT}$ (Internal Pullup) RDY (Internal Pullup, Open Drain) φ2 (IN) Address, Data, R/$\overline{W}$ (Off State, BE = 0)	I_{IN}	 -100 -100 -1 -10	 1 10 1 10	 µA µA µA µA
Output High Voltage (I_{OH} = -100µA) SYNC, Data, Address, R/$\overline{W}$, $\overline{ML}$, $\overline{VP}$, 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/$\overline{W}$, $\overline{ML}$, $\overline{VP}$, M/X, E, VDA, VPA, φ1 (OUT), φ2 (OUT)	V_{OL}	—	0.4	V

Supply Current f = 2 MHz
(No Load) f = 4 MHz
f = 6 MHz
f = 8 MHz
	I_{DD}	—
		—
		—
		—
			10
			20
			30
			40
				mA
				mA
				mA
				mA

Standby Current (No Load; Data Bus = V_{SS} or V_{DD}; φ2(IN) = $\overline{ABORT}$ = $\overline{RES}$ = $\overline{NMI}$ = $\overline{IRQ}$ = $\overline{SO}$ = BE = V_{DD})	I_{SB}	—	10	µA
Capacitance (V_{IN} = 0V, T_A = 25°C, f = 2 MHz) Logic, φ2 (IN) Address, Data, R/$\overline{W}$ (Off State)	 C_{IN} C_{TS}	 — —	 10 15	 pF pF

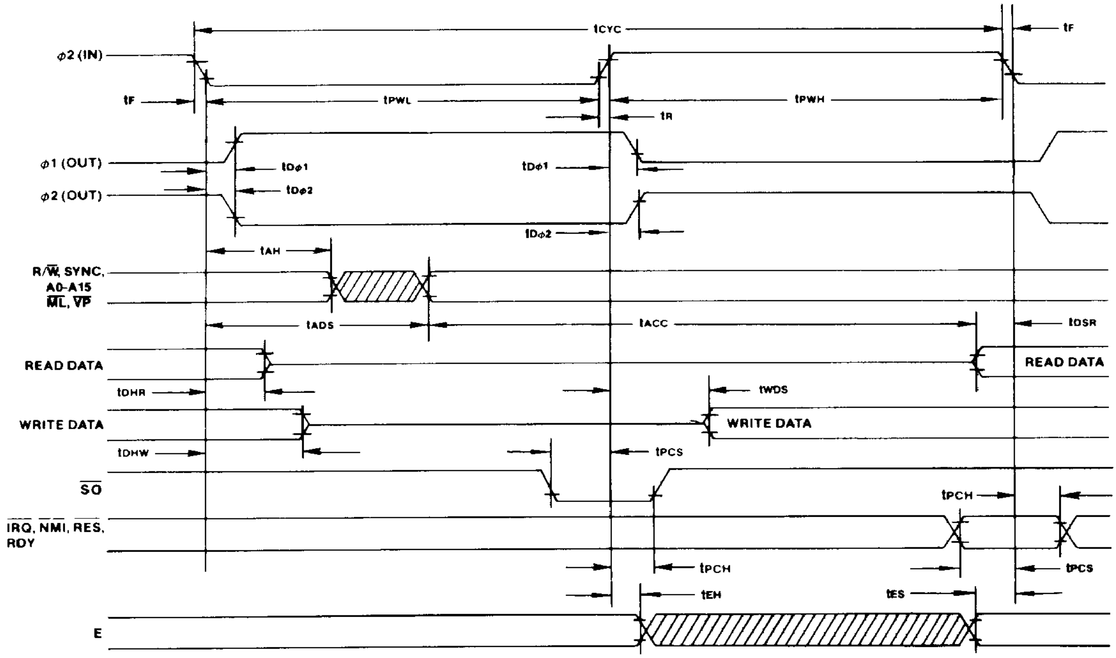
AC Characteristics (G65SC802): V_{DD} = 5.0V ±5%, Vss = 0V, T_A = 0° to +70° C

		2MHz		4MHz		5MHz		6MHz		
Parameter	Symbol	Min	Max	Min	Max	Min	Max	Min	Max	Unit
Cycle Time	t_{CYC}	500	DC	250	DC	200	DC	167	DC	nS
Clock Pulse Width Low	t_{PWL}	0.240	10	0.120	10	0.095	10	0.080	10	µS
Clock Pulse Width High	t_{PWH}	240	∞	120	∞	95	∞	80	∞	nS
Fall Time, Rise Time	t_F, t_R	—	10	—	10	—	5	—	5	nS
Delay Time, φ2 (IN) to φ1 (OUT)	$t_{D\phi 1}$	—	40	—	40	—	35	—	30	nS
Delay Time, φ2 (IN) to φ2 (OUT)	$t_{D\phi 2}$	—	40	—	40	—	35	—	30	nS
Address Hold Time	t_{AH}	10	—	10	—	10	—	10	—	nS
Address Setup Time	t_{ADS}	—	100	—	75	—	65	—	60	nS
Access Time	t_{ACC}	355	—	130	—	100	—	85	—	nS
Read Data Hold Time	t_{DHR}	10	—	10	—	10	—	10	—	nS
Read Data Setup Time	t_{DSR}	40	—	30	—	25	—	20	—	nS
Write Data Delay Time	t_{WDS}	—	100	—	70	—	65	—	60	nS
Write Data Hold Time	t_{DHW}	10	—	10	—	10	—	10	—	nS
Processor Control Setup Time	t_{PCS}	40	—	30	—	25	—	20	—	nS
Processor Control Hold Time	t_{PCH}	10	—	10	—	10	—	10	—	nS
E Output Hold Time	t_{EH}	10	—	10	—	10	—	10	—	nS
E Output Setup Time	t_{ES}	50	—	50	—	35	—	25	—	nS
Capacitive Load (Address, Data, and R/W)	C_{EXT}	—	100	—	100	—	35	—	35	pF

AC Characteristics (G65SC802): V_{DD} = 5.0V ±5%, V_{SS} = 0V, T_A = 0° to +70° C

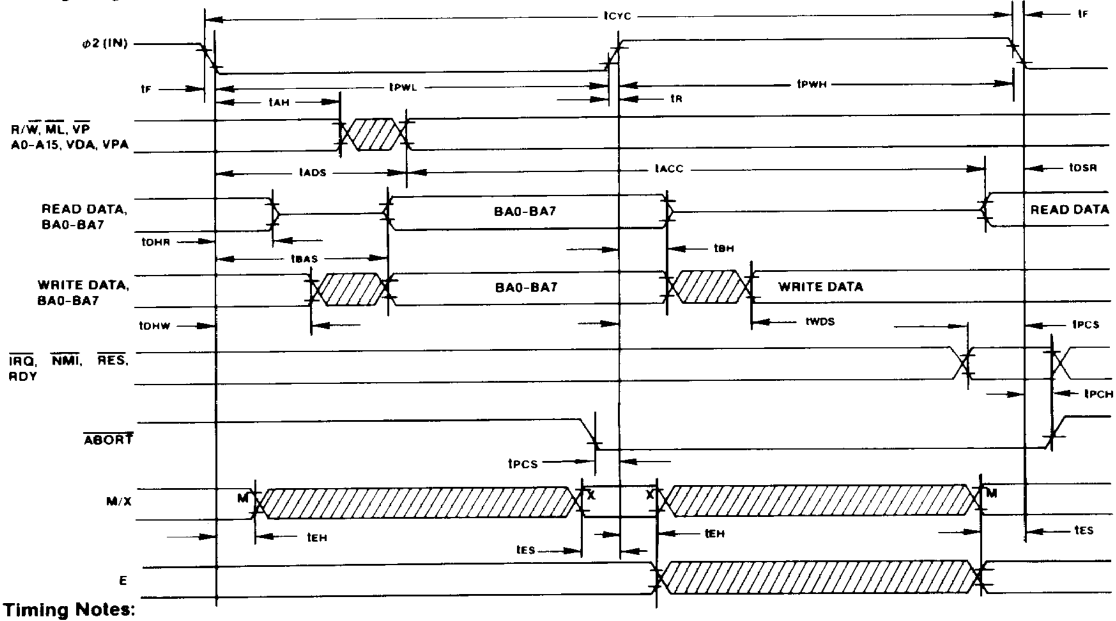
Parameter	Symbol	2MHz		4MHz		5MHz		6MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
Cycle Time	t_{CYC}	500	DC	250	DC	200	DC	167	DC	nS
Clock Pulse Width Low	t_{PWL}	0.240	10	0.120	10	0.095	10	0.080	10	μS
Clock Pulse Width High	t_{PWH}	240	∞	120	∞	95	∞	80	∞	nS
Fall Time, Rise Time	t_F, t_R	—	10	—	10	—	5	—	5	nS
A0–A15 Hold Time	t_{AH}	10	—	10	—	10	—	10	—	nS
A0–A15 Hold Time	t_{ADS}	—	100	—	75	—	65	—	60	nS
BA0–BA7 Hold Time	t_{BH}	10	—	10	—	10	—	10	—	nS
BA0–BA7 Setup Time	T_{BAS}	—	100	—	90	—	75	—	65	nS
Access Time	t_{ACC}	355	—	130	—	100	—	85	—	nS
Read Data Hold Time	t_{DHR}	10	—	10	—	10	—	10	—	nS
Read Data Setup Time	t_{DSR}	40	—	30	—	25	—	20	—	nS
Write Data Delay Time	t_{WDS}	—	100	—	70	—	65	—	60	nS
Write Data Hold Time	t_{DHW}	10	—	10	—	10	—	10	—	nS
Processor Control Setup Time	t_{PCS}	40	—	30	—	25	—	20	—	nS
Processor Control Hold Time	t_{PCH}	10	—	10	—	10	—	10	—	nS
E,MX Output Hold Time	t_{EH}	10	—	10	—	10	—	10	—	nS
E,MX Output Setup Time	t_{ES}	50	—	50	—	35	—	25	—	nS
Capacitive Load (Address, Data, and R/W)	C_{EXT}	—	100	—	100	—	35	—	35	pF
BE to High Impedance State	t_{BHZ}	—	30	—	30	—	30	—	30	nS
BE to Valid Data	t_{BVD}	—	30	—	30	—	30	—	30	nS

Timing Diagram (G65SC802)



Timing Notes:
1. Typical output load = 100 pF
2. Voltage levels are $V_L < 0.4V$, $V_H > 2.4V$
3. Timing measurement points are 0.8V and 2.0V

Timing Diagram (G65SC816)



Timing Notes:

1. Typical output load = 100 pF
2. Voltage levels are $V_L < 0.4V$, $V_H > 2.4V$
3. Timing measurement points are 0.8V and 2.0V

Functional Description

The G65SC802 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 G65SC802's "ease of use" design and implementation features provide the designer with increased flexibility and reduced implementation costs. In the Emulation mode, the G65SC802 not only offers software compatibility, but is also hardware (pin-to-pin) compatible with 6502 designs . . . plus it provides the advantages of 16-bit internal operation in 6502-compatible applications. The G65SC802 is an excellent direct replacement microprocessor for 6502 designs.

The G65SC816 provides the design engineer with upward mobility and software compatibility in applications where a 16-bit system configuration is desired. The G65SC816's 16-bit hardware configuration, coupled with current software allows a wide selection of system applications. In the Emulation mode, the G65SC816 offers many advantages, including full software compatibility with 6502 coding. In addition, the G65SC816's powerful instruction set and addressing modes make it an excellent choice for new 16-bit designs.

Internal organization of the G65SC802 and G65SC816 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. Both the G65SC802 and the G65SC816 have a 16-bit internal architecture with an 8-bit external data bus.

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.

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 and Logic Unit (ALU)

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

Internal Registers (Refer to Figure 2, Programming Model)

Accumulator (A)

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

Data Bank (DB)

During the Native mode (E=0), 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 G65SC816. 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. 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 (providing the Index Select Bit (X) equals zero). If the Index Select Bit (X) equals one, both registers will be 8 bits wide.

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 2, G65SC802 and G65SC816 Mode Comparison, 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 (PB)

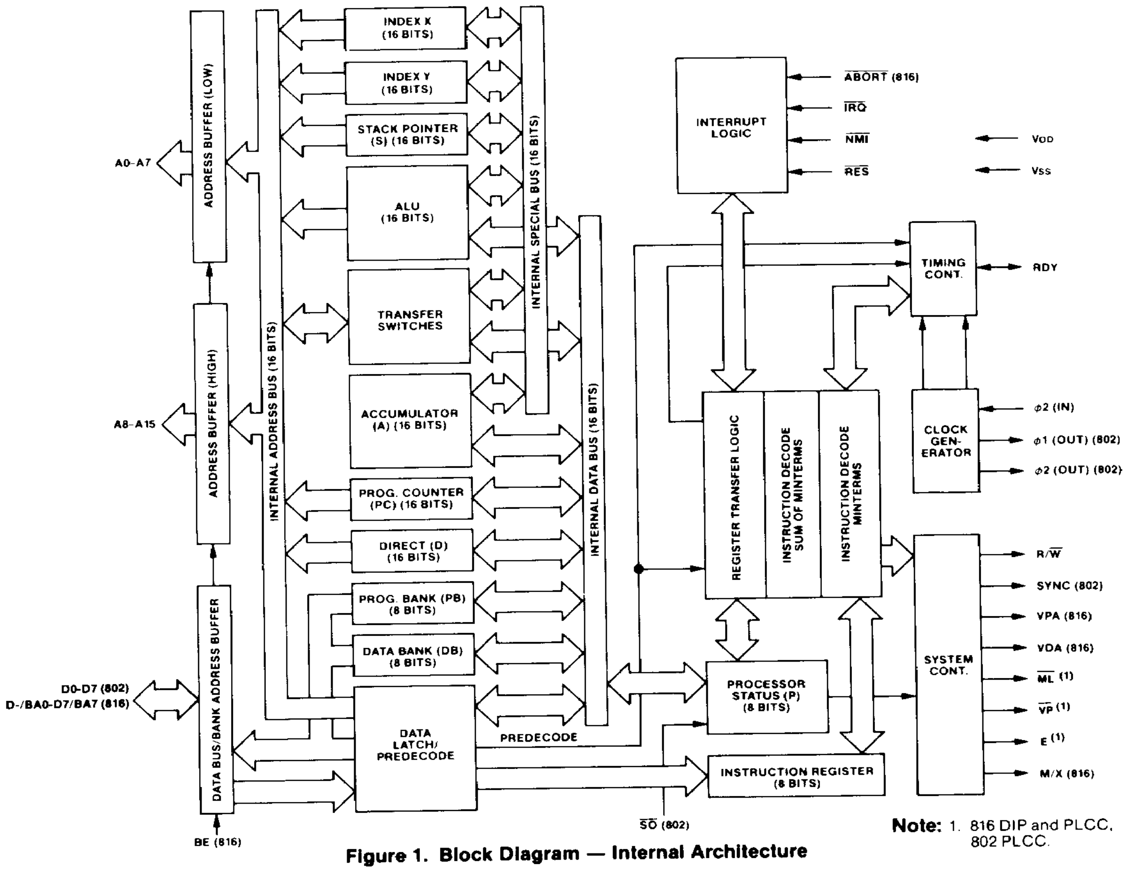
The 8-bit Program Bank Register holds the bank address for all instruction fetches. The 24-bit address consists of the 16-bit instruction effective address and the 8-bit Program Bank address. The register value is multiplexed with the data value and presented on the Data/Address lines during the first half of a program memory read cycle. The Program Bank Register is initialized to zero during Reset.

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.

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 01. The Bank Address is 00 for all Stack operations.



Note: 1. 816 DIP and PLCC, 802 PLCC.

Figure 1. Block Diagram — Internal Architecture

Signal Description

The following Signal Description applies to both the G65SC802 and the G65SC816 except as otherwise noted.

Abort ($\overline{\text{ABORT}}$)—G65SC816

The Abort input prevents modification of any internal registers during execution of 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). Abort is asserted whenever there is a low level on the Abort input, and the $\phi 2$ clock is high. The Abort internal latch is cleared during the second cycle of the interrupt sequence. This signal may be used to handle out-of-bounds memory references in virtual memory systems.

Address Bus (A0-A15)

These sixteen output lines form the Address Bus for memory and I/O exchange on the Data Bus. When using the G65SC816, the address lines may be set to the high impedance state by the Bus Enable (BE) signal.

Bus Enable (BE)

The Bus Enable input signal allows external control of the Address and Data Buffers, as well as the R/$\overline{\text{W}}$ signal. With Bus Enable high, the R/$\overline{\text{W}}$ 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)—G65SC802

The eight Data Bus lines provide an 8-bit bidirectional Data Bus for use during data exchanges 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)—G65SC816

These eight lines multiplex bits BA0-BA7 with the data value. The Bank address is present during the first half of a memory cycle, and the data value is read or written during the second half of the memory cycle. The Bank address external transparent latch should be latched when the $\phi 2$ clock is high or RDY is low. Two memory cycles are required to transfer 16-bit values. These lines may be set to the high impedance state by the Bus Enable (BE) signal.

Emulation Status (E)—G65SC816 (Also Applies to G65SC802, 44-Pin Version)

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

Interrupt Request ($\overline{\text{IRQ}}$)

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 $\overline{\text{IRQ}}$ 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.

Memory Lock ($\overline{\text{ML}}$)—G65SC816 (Also Applies to G65SC802, 44-Pin Version)

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)—G65SC816

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 $\phi 2$ clock positive transition. Instructions PLP, REP, RTI and SEP may change the state of these bits. Note that the M/X output may be invalid in the cycle following a change in the M or X bits. These bits may be thought of as opcode extensions and may be used for memory and system management.

Non-Maskable Interrupt ($\overline{\text{NMI}}$)

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 $\overline{\text{NMI}}$ 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 $\overline{\text{NMI}}$ remains low.

Phase 1 Out ($\phi 1$ (OUT))—G65SC802

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 ($\phi 2$ (IN))

This is the system clock input to the microprocessor internal clock generator (equivalent to $\phi 0$ (IN) on the 6502). During the low power Standby Mode, $\phi 2$ (IN) should be held in the high state to preserve the contents of internal registers.

Phase 2 Out ($\phi 2$ (OUT))—G65SC802

This clock output signal provides timing for external read and write operations. Addresses are valid (after the Address Setup Time (T_{ADS})) following the negative transition of Phase 2 Out. Executing the Stop (STP) instruction holds Phase 2 Out in the High state.

Read/Write (R/$\overline{\text{W}}$)

When the R/$\overline{\text{W}}$ 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. When using the G65SC816, the R/$\overline{\text{W}}$ 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 G65SC802 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 $\overline{\text{RES}}$, $\overline{\text{ABORT}}$, $\overline{\text{NMI}}$, or $\overline{\text{IRQ}}$ external interrupt is provided. This feature may be used to eliminate interrupt latency by placing the WAI instruction at the beginning of the $\overline{\text{IRQ}}$ servicing routine. If the $\overline{\text{IRQ}}$ Disable flag has been set, the next instruction will be executed when the $\overline{\text{IRQ}}$ occurs. The processor will not stop after a WAI instruction if RDY has been forced to a high state. The Stop (STP) instruction has no effect on RDY.

Reset ($\overline{\text{RES}}$)

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 $\overline{\text{RES}}$ signal must be held low for at least two clock cycles after V_{DD} reaches operating voltage. Ready (RDY) has no effect while $\overline{\text{RES}}$ is being held low. During this Reset conditioning period, the following processor initialization takes place:

Registers

D	= 0000	SH	= 01
DB	= 00	XH	= 00
PB	= 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/X	= 1	$\overline{\text{VP}}$	= 1
R/$\overline{\text{W}}$	= 1	VPA	= 0
SYNC	= 0		

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

- R/$\overline{\text{W}}$ remains in the high state during the stack address cycles.
- The Reset vector address is 00FFFC,D.

Set Overflow ($\overline{SO}$)—G65SC802

A negative transition on this input sets the Overflow (V) flag, bit 6 of the Processor Status (P) Register.

Synchronize (SYNC)—G65SC802

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)—G65SC816

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. Address outputs may be invalid due to low byte additions only.
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.

VDD and Vss

VDD is the positive supply voltage and Vss is system ground. When using only one ground on the G65SC802 DIP package, pin 21 is preferred.

Vector Pull (VP)—G65SC816 (Also Applies to G65SC802, 44-Pin Version)

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.

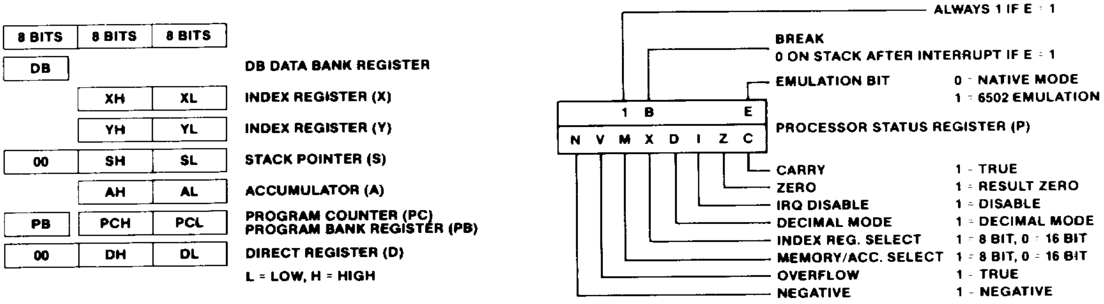


Figure 2. Programming Model

Table 1. G65SC802 and G65SC816 Compatibility

Function	G65SC802/816 Emulation	G65SC02	NMOS 6502
Decimal Mode: • After Interrupts • N, Z Flags • ADC, SBC	0 → D Valid No added cycle	0 → D Valid Add 1 cycle	Not initialized Undefined No added cycle
Read-Modify-Write: • Absolute Indexed, No Page Crossing • Write • Memory Lock	7 cycles Last 2 cycles Last 3 cycles	6 cycles Last cycle Last 2 cycles	7 cycles Last 2 cycles Not available
Jump Indirect: • Cycles • Jump Address, Operand = XXFF	5 cycles Correct	6 cycles Correct	5 cycles Invalid
Branch or Index Across Page Boundary	Read last program byte	Read last program byte	Read invalid address
0 → RDY During Write	G65SC802: Ignored until read G65SC816: Processor stops	Processor stops	Ignored until read
Write During Reset	No	Yes	No
Unused Opcodes	No operation	No operation	Undefined
ϕ1 (OUT), ϕ2 (OUT), $\overline{SO}$, SYNC Signals	Available with G65SC802 only	Available	Available
RDY Signal	Bidirectional	Input	Input

Table 2. G65SC802 and G65SC816 Mode Comparison

Function	Emulation (E = 1)	Native (E = 0)
Stack Pointer (S)	8 bits in page 1	16 bits
Direct Index Address	Wrap within page	Crosses page boundary
Processor Status (P): • Bit 4	Always one, except zero in stack after hardware interrupt	X flag (8/16-bit Index)
• Bit 5	Always one	M flag (8/16-bit Accumulator)
Branch Across Page Boundary	4 cycles	3 cycles
Vector Locations: ABORT BRK COP IRQ NMI RES	 00FFF8,9 00FFFE,F 00FFF4,5 00FFFE,F 00FFFA,B 00FFFC,D	 00FFE8,9 00FFE6,7 00FFE4,5 00FFEE,F 00FFEA,B 00FFFC,D (1 → E)
Program Bank (PB) During Interrupt, RTI	Not pushed, pulled	Pushed and pulled
0 → RDY During Write	G65SC802: Ignored until read G65SC816: Processor stops	Processor stops
Write During Read-Modify-Write	Last 2 cycles	Last 1 or 2 cycles depending on M flag

G65SC802 and G65SC816 Microprocessor Addressing Modes

The G65SC816 is capable of directly addressing 16 MBytes of memory. This address space has special significance within certain addressing modes, as follows:

Reset and Interrupt Vectors

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

Stack

The Native mode Stack address will always be within the range 000000 to 00FFFF. In the Emulation mode, the Stack address range is 000100 to 0001FF. The following opcodes and addressing modes can 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

The Direct addressing modes are often used to access memory registers and pointers. The contents of the Direct Register (D) is added to the offset contained in the instruction operand to produce an address in the range 000000 to 00FFFF. Note that in the Emulation mode, [Direct] and [Direct],y addressing modes and the PEI instruction will increment from 0000FE or 0000FF into the Stack area, even if D=0.

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 16 MByte address space. Words, arrays, records, or any data structures may span 64 KByte bank boundaries with no compromise in code efficiency. As a result, indexing from page FF in the G65SC802 may result in data accessed in page zero. 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 Indexed Long [d],y
- Absolute
- Absolute,x
- Absolute,y
- Absolute long
- Absolute long indexed
- Stack Relative Indirect Indexed (d,s),y

The following addressing mode descriptions provide additional detail as to how effective addresses are calculated.

Twenty-four addressing modes are available for use with the G65SC802 and G65SC816 microprocessors. The "long" addressing modes may be used with the G65SC802; however, the high byte of the address is not available to the hardware. Detailed descriptions of the 24 addressing modes are as follows:

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:	DB	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

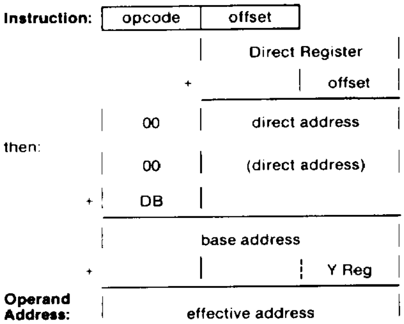
This form of addressing always uses a single byte instruction. The operand is the Accumulator.

6. Implied—·

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

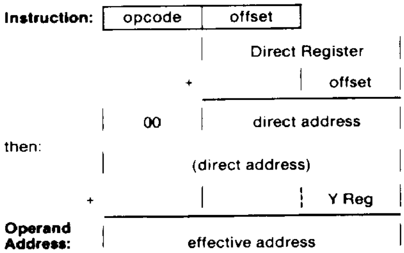
7. Direct Indirect Indexed—(d),y

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.



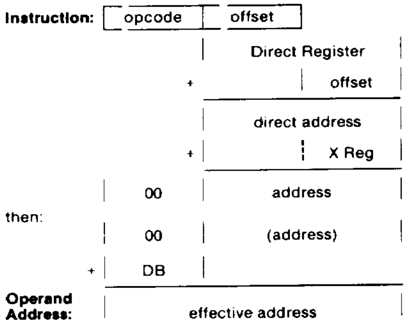
8. Direct Indirect Indexed Long—[d],y

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.



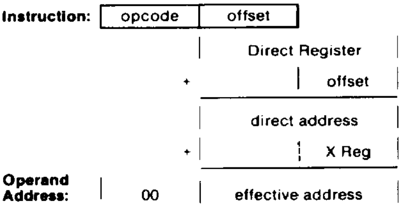
9. Direct Indexed Indirect—(d,x)

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.



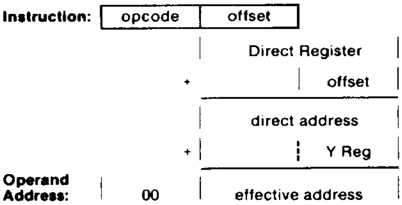
10. Direct Indexed With X—d,x

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.



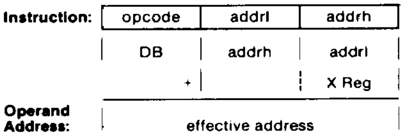
11. Direct Indexed With Y—d,y

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.



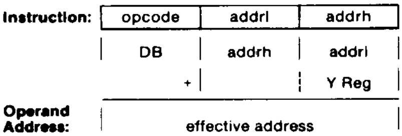
12. Absolute Indexed With X—a,x

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.



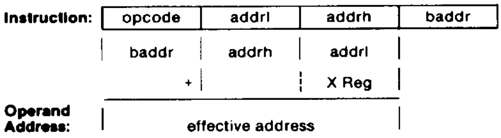
13. Absolute Indexed With Y—a,y

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.



14. 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.



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 and result are both an unsigned 16-bit quantity in the range 0 to 65535.

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 PB = (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)  |
            + |   DB   |
              ----------------------------
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  |
                       -------------------
              |   00   |     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     |
            + |   DB   |
              ----------------------------
              |       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 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  →  DB
Source
Address:               | srcbnk |      X Reg      |
Destination
Address:               |   DB   |      Y Reg      |

       Increment (MVN) or decrement (MVP) X and Y.
       Decrement A, (if greater than zero), then PC-3 → PC.
```

Notes on G65SC802/816 Instructions

All Opcodes Function in All Modes of Operation

It should be noted that all opcodes function in all modes of operation. However, some instructions and addressing modes are intended for G65SC816 24-bit addressing and are therefore less useful for the G65SC802. The following is a list of instructions and addressing modes which are primarily intended for G65SC816 use:

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

The following instructions may be used with the G65SC802 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 will always be high (logic 1).
- When in the Emulation mode, the MVP and MVN instructions only move date in page zero since X and Y Index Register high byte is zero.

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 byte of the Accumulator AL and AH are not affected by a mode change.

WAI Instruction

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

STP Instruction

The STP instruction disables the $\phi 2$ clock to all circuitry. When disabled, the $\phi 2$ clock is held in the high state. In this case, the Data Bus will remain in the data transfer state and the Bank address will not be multiplexed onto the Data Bus. Upon executing the STP instruction, the $\overline{\text{RES}}$ signal is the only input which can restart the processor. The processor is restarted by enabling the $\phi 2$ clock, which occurs on the falling edge of the $\overline{\text{RES}}$ input. Note that the external oscillator must be stable and operating properly before $\overline{\text{RES}}$ goes high.

Tranfers from 8-Bit to 16-Bit, or 16-Bit to 8-Bit Registers

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

TCS; TSC; TCD; TDC

Stack Transfers

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

Interrupt Processing Sequence

The interrupt processing sequence is initiated as the direct result of hardware Abort, Interrupt Request, Non-Maskable Interrupt, or Reset inputs. The interrupt sequence can also be initiated as a result of the Break or Co-Processor instructions within the software. The following listings describe the function of each cycle in the interrupt processing sequence:

Hardware Interrupt—$\overline{ABORT}$, $\overline{IRQ}$, $\overline{NMI}$, $\overline{RES}$ Inputs

Cycle No.		Address	Data	$R/\overline{W}$	SYNC	VDA	VPA	$\overline{VP}$	Description
E = 0	E = 1								
1	1	PC	X	1	1	1	1	1	Internal Operation
2	2	PC	X	1	0	0	0	1	Internal Operation
3	[1]	S	PB	0	0	1	0	1	Write PB to Stack, S-1 → S
4	3	S	PCH [2]	0 [3]	0	1	0	1	Write PCH to Stack, S-1 → S
5	4	S	PCL [2]	0 [3]	0	1	0	1	Write PCL to Stack, S-1 → S
6	5	S	P [4]	0 [3]	0	1	0	1	Write P to Stack, S-1 → S
7	6	VL	(VL)	1	0	1	0	0	Read Vector Low Byte, 0 → P_D, 1 → P_I,00 → PB
8	7	VH	(VH)	1	0	1	0	0	Read Vector High Byte

Software Interrupt—BRK, COP Instructions

Cycle No.		Address	Data	$R/\overline{W}$	SYNC	VDA	VPA	$\overline{VP}$	Description
E = 0	E = 1								
1	1	PC-2	X	1	1	1	1	1	Opcode
2	2	PC-1	X	1	0	0	1	1	Signature
3	[1]	S	PB	0	0	1	0	1	Write PB to Stack, S-1 → S
4	3	S	PCH	0	0	1	0	1	Write PCH to Stack, S-1 → S
5	4	S	PCL	0	0	1	0	1	Write PCL to Stack, S-1 → S
6	5	S	P	0	0	1	0	1	Write P to Stack, S-1 → S
7	6	VL	(VL)	1	0	1	0	0	Read Vector Low Byte, 0 → P_D, 1 → P_I, 00 → PB
8	7	VH	(VH)	1	0	1	0	0	Read Vector High Byte

Notes:

[1] Delete this cycle in Emulation mode.
[2] Abort writes address of aborted opcode.
[3] $R/\overline{W}$ remains in the high state during Reset.
[4] In Emulation mode, bit 4 written to stack is changed to 0.

Table 3. Vector Locations

Name	Source	Emulation (E = 1)	Native (E = 0)	Priority Level
$\overline{ABORT}$	Hardware	00FFF8,9	00FFE8,9	2
BRK	Software	00FFFE,F	00FFE6,7	N/A
COP	Software	00FFF4,5	00FFE4,5	N/A
$\overline{IRQ}$	Hardware	00FFFE,F	00FFEE,F	4
$\overline{NMI}$	Hardware	00FFFA,B	00FFEA,B	3
$\overline{RES}$	Hardware	00FFFC,D	00FFFC,D (1 → E)	1

Table 4. G65SC802 and G65SC816 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 (add one cycle if DL ≠ 0)
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 Bits
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 Accumulator to Direct Register
TCS*	Transfer Accumulator to Stack Pointer Register
TDC*	Transfer Direct Register to Accumulator
TRB	Test and Reset Bit
TSB	Test and Set Bit
TSC*	Transfer Stack Pointer Register to 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
XBA*	Exchange AH and AL
XCE	Exchange Carry and Emulation Bits

***Common Mnemonic Aliases**

Mnemonic	Alias
BCC	BLT
BCS	BGE
CMP	CPA
DEC A	DEA
INC A	INA
TCD	TAD
TCS	TAS
TDC	TDA
TSC	TSA
XBA	SWA

**JSL should be recognized as equivalent to JSR when it is specified with long absolute addresses.

JML is equivalent to JMP with long addressing forced.

Table 5. Arithmetic and Logical Instructions

MNE-MONIC	M/X	OPERATION E=1 or E=0 and M/X=1	 E=0 and M/X=0	Addressing Mode immed	 accum	 dir	 dir,x	 dir,y	 (dir)	 (dir,x)	 (dir),y	 [dir]	 [dir],y	 abs	 abs,x	 abs,y	 abs1	 abs1,x	 d,s	 (d,s),y	STATUS N V M X D I Z C	MNEMONIC
ADC	Pm	AL+B+Pc -AL	A+W+Pc -A	69		65	75		72	61	71	67	77	6D	7D	79	6F	7F	63	73	N V . . . . Z C	ADC
AND	Pm	AL∧B -AL	A∧W -A	29		25	35		32	21	31	27	37	2D	3D	39	2F	3F	23	33	N . . . . . Z .	AND
ASL (2)	Pm	Pc← B← 0	Pc← W← 0		0A	06	16							0E	1E						N . . . . . Z C	ASL
BIT (1)	Pm	AL∧B	A∧W	89		24	34							2C	3C						N V . . . . Z .	BIT
CMP	Pm	AL-B	A-W	C9		C5	D5		D2	C1	D1	C7	D7	CD	DD	D9	CF	DF	C3	D3	N . . . . . Z C	CMP
CPX	Px	XL-B	X-W	E0		E4								EC							N . . . . . Z C	CPX
CPY	Px	YL-B	Y-W	C0		C4								CC							N . . . . . Z C	CPY
DEC (2)	Pm	B-1 -B	W-1 -W		3A	C6	D6							CE	DE						N . . . . . Z .	DEC
EOR	Pm	AL∀B -AL	A∀W -A	49		45	55		52	41	51	47	57	4D	5D	59	4F	5F	43	53	N . . . . . Z .	EOR
INC (2)	Pm	B+1 -B	W+1 -W		1A	E6	F6							EE	FE						N . . . . . Z .	INC
LDA	Pm	B -AL	W -A	A9		A5	B5		B2	A1	B1	A7	B7	AD	BD	B9	AF	BF	A3	B3	N . . . . . Z .	LDA
LDX	Px	B -XL	W -X	A2		A6		B6						AE		BE					N . . . . . Z .	LDX
LDY	Px	B -YL	W -Y	A0		A4	B4							AC	BC						N . . . . . Z .	LDY
LSR (2)	Pm	0 -B -Pc	0 -W -Pc		4A	46	56							4E	5E						0 . . . . . Z C	LSR
ORA	Pm	ALVB -AL	AVW -A	09		05	15		12	01	11	07	17	0D	1D	19	0F	1F	03	13	N . . . . . Z .	ORA
ROL (2)	Pm	Pc← B← Pc	PC← W←Pc		2A	26	36							2E	3E						N . . . . . Z C	ROL
ROR (2)	Pm	PC -B -Pc	Pc -W -Pc		6A	66	76							6E	7E						N . . . . . Z C	ROR
SBC	Pm	AL-B-$\overline{Pc}$ -AL	A-W-$\overline{Pc}$ -A	E9		E5	F5		F2	E1	F1	E7	F7	ED	FD	F9	EF	FF	E3	F3	N V . . . . Z C	SBC
STA (7)	Pm	AL -B	A -W			85	95		92	81	91	87	97	8D	9D	99	8F	9F	83	93	. . . .	STA
STX	Px	XL -B	X -W			86		96						8E							. . . .	STX
STY	Px	YL -B	Y -W			84	94							8C							. . . .	STY
STZ (7)	Pm	0 -B	0 -W			64	74							9C	9E						. . . .	STZ
TRB (8)	Pm	$\overline{AL}$∧B -B	$\overline{A}$∧W -W			14								1C							. . . . . . Z .	TRB
TSB (8)	Pm	ALVB -B	AVW -W			04								0C							. . . . . . Z .	TSB
						← add one cycle if DL ≠ 0 →																
Emulation (E=1) or Native (E=0) Mode. 8 bit (M/X=1)			cycles	2	2	3	4	4	5	6	5 (3)	6	6	4	4 (3)	4 (3)	5	5	4	7		
			bytes	2	1	2	2	2	2	2	2	2	2	3	3	3	4	4	2	2		
Native Mode (E=0). 16 bit (M/X=0)			cycles	3	2	4	5	5	6	7	6	7	7	5	5	5	6	5	5	8		
			bytes	3	1	2	2	2	2	2	2	2	2	3	3	3	4	4	2	2		

V	logical OR	B	byte per effective address
Λ	logical AND	W	word per effective address
∀	logical exclusive OR	r	relative offset
+	arithmetic addition	A	Accumulator, AL low half of Accumulator
-	arithmetic subtraction	X	Index Register, XL low half of X register
≠	not equal	Y	Index Register, YL low half of Y register
•	status bit not affected	Pc	carry bit
		M/X	effective mode bit in Status Register (Pm or Px)
		Ws	word per stack pointer
		Bs	byte per stack pointer

Notes:

1. BIT instruction does not affect N and V flags when using immediate addressing mode. When using other addressing modes, the N and V flags are respectively set to bits 7 and 6 or 15 and 14 of the addressed memory depending on mode (byte or word).
2. For all Read/Modify/Write instruction addressing modes except accumulator—
   Add 2 cycles for E=1 or E=0 and Pm=1 (8-bit mode).
   Add 3 cycles for E=0 and Pm=0 (16-bit mode).
3. Add one cycle when indexing across page boundary and E=1 except for STA and STZ instructions.
4. If E=1 then 1 -SH and XL -SL. If E=0 then X→S regardless of Pm or Px.
5. Exchanges the carry (Pc) and E bits. Whenever the E bit is set the following registers and status bits are locked into the indicated state: XH=0, YH=0, SH=1, Pm=1, Px=1.
6. Add 1 cycle if branch is taken. In Emulation (E=1) mode only—add 1 cycle if the branch is taken and crosses a page boundary.
7. Add 1 cycle in Emulation mode (E=1) for (dir),y; abs,x; and abs,y addressing modes.
8. With TSB and TRB instruction, the Z flag is set or cleared by the result of A∧B or A∧W.
   For all Read/Modify/Write instruction addressing modes except accumulator—
   Add 2 cycles for E=1 or E=0 and Pm=1 (8-bit mode).
   Add 3 cycles for E=0 and Pm=0 (16-bit mode).

Table 6. Branch, Transfer, Push, Pull, and Implied Addressing Mode Instructions

Mnemonic	Bytes	M/X	Cycles	Operation 8 Bit	Cycles	Operation 16 Bit	Implied	Stack	Relative	Status N V M X D I Z C	Mnemonic
BCC (6)	2	—	2	PC+r→PC	2	PC+r→PC			90	. . . .	BCC
BCS (6)	2	—	2	PC+r→PC	2	PC+r→PC			B0	. . . .	BCS
BEQ (6)	2	—	2	PC+r→PC	2	PC+r→PC			F0	. . . .	BEQ
BMI (6)	2	—	2	PC+r→PC	2	PC+r→PC			30	. . . .	BMI
BNE (6)	2	—	2	PC+r→PC	2	PC+r→PC			D0	. . . .	BNE
BPL (6)	2	—	2	PC+r→PC	2	PC+r→PC			10	. . . .	BPL
BRA (6)	2	—	2	PC+r→PC	2	PC+r→PC			80	. . . .	BRA
BVC (6)	2	—	2	PC+r→PC	2	PC+r→PC			50	. . . .	BVC
BVS (6)	2	—	2	PC+r→PC	2	PC+r→PC			70	. . . .	BVS
CLC	1	—	2	0→Pc	2	0→Pc	18			 0	CLC
CLD	1	—	2	0→Pd	2	0→Pd	D8			. . . . 0 . . .	CLD
CLI	1	—	2	0→Pi	2	0→Pi	58			. . . . . 0 . .	CLI
CLV	1	—	2	0→Pv	2	0→Pv	B8			. 0	CLV
DEX	1	Px	2	XL-1→XL	2	X-1→X	CA			N . . . . . Z .	DEX
DEY	1	Px	2	YL-1→YL	2	Y-1→Y	88			N . . . . . Z .	DEY
INX	1	Px	2	XL+1→XL	2	X+1→X	E8			N . . . . . Z .	INX
INY	1	Px	2	YL+1→YL	2	Y+1→Y	C8			N . . . . . Z .	INY
NOP	1	—	2	no operation	2	no operation	EA			. . . .	NOP
PEA	3	—	5	W→Ws, S-2→S	5	same		F4		. . . .	PEA
PEI	2	—	6	W→Ws, S-2→S	6	same		D4		. . . .	PEI
PER	3	—	6	W→Ws, S-2→S	6	same		62		. . . .	PER
PHA	1	Pm	3	AL→Bs, S-1→S	4	A→Ws, S-2→S		48		. . . .	PHA
PHB	1	—	3	DB→Bs, S-1→S	3	same		8B		. . . .	PHB
PHD	1	—	4	D→Ws, S-2→S	4	same		0B		. . . .	PHD
PHK	1	—	3	PB→Bs, S-1→S	3	same		4B		. . . .	PHK
PHP	1	—	3	P→Bs, S-1→S	3	same		08		. . . .	PHP
PHX	1	Px	3	XL→Bs, S-1→S	4	X→Ws, S-2→S		DA		. . . .	PHX
PHY	1	Px	3	YL→Bs, S-1→S	4	Y→Ws, S-2→S		5A		. . . .	PHY
PLA	1	Pm	4	S+1→S, Bs→AL	5	S+2→S, Ws→A		68		N . . . . . Z .	PLA
PLB	1	—	4	S+1→S, Bs→DB	4	same		AB		N . . . . . Z .	PLB
PLD	1	—	5	S+2→S, Ws→D	5	same		2B		N . . . . . Z .	PLD
PLP	1	—	4	S+1→S, Bs→P	4	same		28		N V M X D I Z C	PLP
PLX	1	Px	4	S+1→S, Bs→XL	5	S+2→S, Ws→X		FA		N . . . . . Z .	PLX
PLY	1	Px	4	S+1→S, Bs→YL	5	S+2→S, Ws→Y		7A		N . . . . . Z .	PLY
SEC	1	—	2	1→Pc	2	1→Pc	38			 1	SEC
SED	1	—	2	1→Pd	2	1→Pd	F8			. . . . 1 . . .	SED
SEI	1	—	2	1→Pi	2	1→Pi	78			. . . . . 1 . .	SEI
TAX	1	Px	2	AL→XL	2	A→X	AA			N . . . . . Z .	TAX
TAY	1	Px	2	AL→YL	2	A→Y	A8			N . . . . . Z .	TAY
TCD	1	—	2	A→D	2	A→D	5B			N . . . . . Z .	TCD
TCS	1	—	2	A→S	2	A→S	1B			. . . .	TCS
TDC	1	—	2	D→A	2	D→A	7B			N . . . . . Z .	TDC
TSC	1	—	2	S→A	2	S→A	3B			N . . . . . Z .	TSC
TSX	1	Px	2	SL→XL	2	S→X	BA			N . . . . . Z .	TSX
TXA	1	Pm	2	XL→AL	2	X→A	8A			N . . . . . Z .	TXA
TXS	1	—	2	see note 4	2	X→S	9A			. . . .	TXS
TXY	1	Px	2	XL→YL	2	X→Y	9B			N . . . . . Z .	TXY
TYA	1	Pm	2	YL→AL	2	Y→A	98			N . . . . . Z .	TYA
TYX	1	Px	2	YL→XL	2	Y→X	BB			N . . . . . Z .	TYX
XCE	1	—	2	see note 5	2	see note 5	FB			 C	XCE

See Notes on page 2-106.

Table 7. Other Addressing Mode Instructions

Mnemonic	Addressing Mode	Op Code	Cycles	Bytes	Status N V M X D I Z C	Mnemonic	Function
BRK	stack	00	7/8	2	. . . . 0 1 . .	BRK	See discussion in Interrupt Processing Sequence section.
BRL	relative long	82	3	3	. . . .	BRL	PC+r→PC where −32768<r<32767.
COP	stack	02	7/8	2	. . . . 0 1 . .	COP	See discussion in Interrupt Processing Sequence section.
JML	absolute indirect	DC	6	3	. . . .	JML	W→PC, B→PB
JMP	absolute	4C	3	3	. . . .	JMP	W→PC
JMP	absolute indirect	6C	5	3	. . . .	JMP	W→PC
JMP	absolute indexed indirect	7C	6	3	. . . .	JMP	W→PC
JMP	absolute long	5C	4	4	. . . .	JMP	W→PC, B→PB
JSL	absolute long	22	8	4	. . . .	JSL	PB→Bs, S-1→S, PC→Ws, S-2→S, W→PC, B→PB
JSR	absolute	20	6	3	. . . .	JSR	PC→Ws, S-2→S, W→PC
JSR	absolute indexed indirect	FC	6	3	. . . .	JSR	PC→Ws, S-2→S, W→PC
MVN	block	54	7/byte	3	. . . .	MVN	See discussion in Addressing Mode section
MVP	block	44	7/byte	3	. . . .	MVP	
REP	immediate	C2	3	2	N V M X D I Z C	REP	$P \wedge \bar{B} \rightarrow P$
RTI	stack	40	6/7	1	N V M X D I Z C	RTI	S+1→S, Bs→P, S+2→S, Ws→PC, if E=0 then S+1→S, Bs→PB
RTL	stack	6B	6	1	. . . .	RTL	S+2→S, Ws+1→PC, S+1→S, Bs→PB
RTS	stack	60	6	1	. . . .	RTS	S+2→S, Ws+1→PC
SEP	immediate	E2	3	2	N V M X D I Z C	SEP	$P \vee B \rightarrow P$
STP	implied	DB	3 +	1	. . . .	STP	Stop the clock. Requires reset to continue.
WAI	implied	CB	3 +	1	. . . .	WAI	Wait for interrupt. RDY held low until interrupt.
XBA	implied	EB	3	1	N . . . . . Z .	XBA	Swap AH and AL. Status bits reflect final condition of AL.

See Notes on page 2-106.

Table 8. Opcode Matrix

M S D	LSD																M S D
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK s 2 8	ORA (d,x) 2 6	COP s 2 8	ORA d,s 2 4	TSB d 2 5	ORA d 2 3	ASL d 2 5	ORA [d] 2 6	PHP s 1 3	ORA # 2 2	ASL A 1 2	PHD s 1 4	TSB a 3 6	ORA a 3 4	ASL a 3 6	ORA al 4 5	0
1	BPL r 2 2	ORA (d),y 2 5	ORA (d) 2 5	ORA (d,s),y 2 7	TRB d 2 5	ORA d,x 2 4	ASL d,x 2 6	ORA [d],y 2 6	CLC i 1 2	ORA a,y 3 4	INC A 1 2	TCS i 1 2	TRB a 3 6	ORA a,x 3 4	ASL a,x 3 7	ORA al,x 4 5	1
2	JSR a 3 6	AND (d,x) 2 6	JSL al 4 8	AND d,s 2 4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND [d] 2 6	PLP s 1 4	AND # 2 2	ROL A 1 2	PLD s 1 5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4 5	2
3	BMI r 2 2	AND (d),y 2 5	AND (d) 2 5	AND (d,s),y 2 7	BIT d,x 2 4	AND d,x 2 4	ROL d,x 2 6	AND [d],y 2 6	SEC i 1 2	AND a,y 3 4	DEC A 1 2	TSC i 1 2	BIT a,x 3 4	AND a,x 3 4	ROL a,x 3 7	AND al,x 4 5	3
4	RTI s 1 7	EOR (d,x) 2 6	reserve 2 2	EOR d,s 2 4	MVP xya 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 xya 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 indexed long
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 indexed long
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	xya	block move

legend	
instruction mnemonic	addressing mode
base number of bytes	base number of cycles

Table 9. Detailed Instruction Operation

	ADDRESS MODE		CYCLE	$\overline{\text{VP}}$,	$\overline{\text{ML}}$,	VDA,	VPA		ADDRESS BUS	DATA BUS	R/$\overline{\text{W}}$
1	Immediate ·#		1	1	1	1	1		PBR PC	Op Code	1
	(LDY.CPY.CPX LDX.ORA.		2	1	1	0	1		PBR.PC+1	IDL	1
	AND.EOR ADC.BIT LDA	(1)(8)	2a	1	1	0	1		PBR.PC+2	IDH	1
	CMP SBC REP SEP)										
	(14 Op Codes)										
	(2 and 3 bytes)										
	(2 and 3 cycles)										
2a	Absolute—a		1	1	1	1	1		PBR.PC	Op Code	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	Data Low	1/0
	STA.LDA.CMP.SBC)	(1)	4a	1	·	1	0		DBR.AA+1	Data High	1/0
	(16 Op Codes)										
	(3 bytes)										
	(4 and 5 cycles)										
2b	Absolute (R-M-W)—a		1	1	1	1	1		PBR.PC	Op Code	1
			2	1	1	0	1		PBR.PC+1	AAL	1
	(ASL.ROL.LSR.ROR		3	1	1	0	1		PBR.PC+2	AAH	1
	DEC INC.TSB.TRB)		4	1	0	1	0		DBR.AA	Data Low	1
	(8 Op Codes)	(1)	4a	1	0	1	0		DBR.AA+1	Data High	1
	(3 bytes)	(3)	5	1	0	0	0		DBR.AA+1	IO	1
	(6 and 8 cycles)	(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		1	1	1	1	1		PBR.PC	Op Code	1
	(JMP)(4C)		2	1	1	0	1		PBR.PC+1	NEW PCL	1
	(1 Op Code)		3	1	1	0	1		PBR.PC+2	NEW PCH	1
	(3 bytes)		1	1	1	1	1		PBR, NEW PC	New Op Code	1
	(3 cycles)										
2d	Absolute (Jump to		1	1	1	1	1		PBR.PC	Op Code	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 Op Code)		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 N6502)		1	1	1	1	1		PBR.NEW PC	New Op Code	1
★3a	Absolute Long—al		1	1	1	1	1		PBR.PC	Op Code	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 Op Codes)		4	1	1	0	1		PBR.PC+3	AAB	1
	(4 bytes)		5	1	1	1	0		AAB.AA	Data Low	1/0
	(5 and 6 cycles)	(1)	5a	1	1	1	0		AAB.AA+1	Data High	1/0
★3b	Absolute Long (JUMP)—al		1	1	1	1	1		PBR.PC	Op Code	1
	(JMP)		2	1	1	0	1		PBR.PC+1	NEW PCL	1
	(1 Op Code)		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	New Op Code	1
★3c	Absolute Long (Jump to		1	1	1	1	1		PBR.PC	Op Code	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 Op Code)		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 Op Code	1
4a	Direct—d		1	1	1	1	1		PBR.PC	Op Code	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	Data Low	1/0
	STA.LDA.CMP.SBC)	(1)	3a	1	1	1	0		0,D+DO+1	Data High	1/0
	(16 Op Codes)										
	(2 bytes)										
	(3.4 and 5 cycles)										
4b	Direct (R-M-W)—d		1	1	1	1	1		PBR.PC	Op Code	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
	(8 Op Codes)		3	1	0	1	0		0,D+DO	Data Low	1
	(2 bytes)	(1)	3a	1	0	1	0		0,D+DO+1	Data High	1
	(5.6 7 and 8 cycles)	(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		1	1	1	1	1		PBR.PC	Op Code	1
	(ASL INC.ROL DEC.LSR.ROR)		2	1	1	0	0		PBR.PC+1	IO	1
	(6 Op Codes)										
	(1 byte)										
	(2 cycles)										
6a	Implied—i		1	1	1	1	1		PBR.PC	Op Code	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 Op Codes)										
	(1 byte)										
	(2 cycles)										
★6b	Implied—i		1	1	1	1	1		PBR.PC	Op Code	1
	(XBA)		2	1	1	0	0		PBR.PC+1	IO	1
	(1 Op Code)		3	1	1	0	0		PBR.PC+1	IO	1
	(1 byte)										
	(3 cycles)										
								RDY			
● 6c	Wait For Interrupt										
	(WAI)		1	1	1	1	1	1	PBR.PC	Op Code	1
	(1 Op Code)	(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)	$\overline{\text{IRQ}}$.$\overline{\text{NMI}}$	1	1	1	1	1	1	PBR.PC+1	IRQ(BRK)	1
● 6d	Stop-The-Clock										
	(STP)		1	1	1	1	1	1	PBR.PC	Op Code	1
	(1 Op Code)		2	1	1	0	0	1	PBR.PC+1	IO	1
	(1 byte)	$\overline{\text{RES}}$=1	3	1	1	0	0	1	PBR.PC+1	IO	1
	(3 cycles)	$\overline{\text{RES}}$=0	1c	1	1	0	0	1	PBR.PC+1	RES(BRK)	1
		$\overline{\text{RES}}$=0	1b	1	1	0	0	1	PBR.PC+1	RES(BRK)	1
		$\overline{\text{RES}}$=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)										

	ADDRESS MODE		CYCLE	$\overline{\text{VP}}$,	$\overline{\text{ML}}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{\text{W}}$
7	Direct Indirect Indexed—(d).y		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		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 and 8 cycles)	(4)	4a	1	1	0	0	DBR.AAH.AAL+YL	IO	1
			5	1	1	1	0	DBR.AA+Y	Data Low	1/0
		(1)	5a	1	1	1	0	DBR.AA+Y+1	Data High	1/0
8	Direct Indirect		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		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 and 8 cycles)		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)		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		3	1	1	0	0	PBR.PC+1	IO	1
	(2 bytes)		4	1	1	1	0	0,D+DO+X	AAL	1
	(6.7 and 8 cycles)		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		1	1	1	1	1	PBR.PC	Op Code	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
	(12 Op Codes)		4	1	1	1	0	0,D+DO+X	Data Low	1/0
	(2 bytes)	(1)	4a	1	1	1	0	0,D+DO+X+1	Data High	1/0
	(4.5 and 6 cycles)									
10b	Direct.X (R-M-W)—d.x		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		3	1	1	0	0	PBR.PC+1	IO	1
	(2 bytes)		4	1	0	1	0	0,D+DO+X	Data Low	1
	(6.7.8 and 9 cycles)	(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		1	1	1	1	1	PBR.PC	Op Code	1
	(STX.LDX)		2	1	1	0	1	PBR.PC+1	DO	1
	(2 Op Codes)	(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 and 6 cycles)		4	1	1	1	0	0,D+DO+Y	Data Low	1/0
		(1)	4a	1	1	1	0	0,D+DO+Y+1	Data High	1/0
12a	Absolute.X—a.x		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		4	1	1	1	0	DBR.AA+X	Data Low	1/0
	(3 bytes)	(1)	4a	1	1	1	0	DBR.AA+X+1	Data High	1/0
	(4.5 and 6 cycles)									
12b	Absolute.X (R-M-W)—a.x		1.	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		4.	1	1	0	0	DBR.AAH.AAL+XL	IO	1
	(3 bytes)		5.	1	0	1	0	DBR.AA+X	Data Low	1
	(7 and 9 cycles)	(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		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		4	1	1	0	1	PBR.PC+3	AAB	1
	(4 bytes)		5	1	1	1	0	AAB.AA+X	Data Low	1/0
	(5 and 6 cycles)	(1)	5a	1	1	1	0	AAB.AA+X+1	Data High	1/0
14	Absolute.Y—a.y		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)	(4)	3a	1	1	0	0	DBR.AAH.AAL+YL	IO	1
	(3 bytes)		4	1	1	1	0	DBR.AA+Y	Data Low	1/0
	(4.5 and 6 cycles)	(1)	4a	1	1	1	0	DBR.AA+Y+1	Data High	1/0
15	Relative—r		1	1	1	1	1	PBR.PC	Op Code	1
	(BPL.BMI.BVC.BVS.BCC.		2	1	1	0	1	PBR.PC+1	Offset	1
	BCS.BNE.BEQ.BRA)	(5)	2a	1	1	0	0	PBR.PC+2	IO	1
	(9 Op Codes)	(6)	2b	1	1	0	0	PBR.PC+2+OFF	IO	1
	(2 bytes)		1	1	1	1	1	PBR.New PC	New Op Code	1
	(2.3 and 4 cycles)									
★16	Relative Long—rl		1	1	1	1	1	PBR.PC	Op Code	1
	(BRL)		2	1	1	0	1	PBR.PC+1	Offset Low	1
	(1 Op Code)		3	1	1	0	1	PBR.PC+2	Offset High	1
	(3 bytes)		4	1	1	0	0	PBR.PC+2	IO	1
	(4 cycles)		1	1	1	1	1	PBR.New PC	New Op Code	1
17a	Absolute Indirect—(a)		1	1	1	1	1	PBR.PC	Op Code	1
	(JMP)		2	1	1	0	1	PBR.PC+1	AAL	1
	(1 Op Code)		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	Op Code	1
★17b	Absolute Indirect—(a)		1	1	1	1	1	PBR.PC	Op Code	1
			2	1	1	0	1	PBR.PC+1	AAL	1
	(JML)		3	1	1	0	1	PBR.PC+2	AAH	1
	(1 Op Code)		4	1	1	1	0	0.AA	NEW PCL	1
	(3 bytes)		5	1	1	1	0	0.AA+1	NEW PCH	1
	(6 cycles)		6	1	1	1	0	0.AA+2	NEW PBR	1
			1	1	1	1	1	NEW PBR.PC	New Op Code	1
● 18	Direct Indirect—(d)		1	1	1	1	1	PBR.PC	Op Code	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 Op Codes)		3	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4	1	1	1	0	0,D+DO+1	AAH	1
	(5.6 and 7 cycles)		5	1	1	1	0	DBR.AA	Data Low	1/0
		(1)	5a	1	1	1	0	DBR.AA+1	Data Low	1/0

Table 9. Detailed Instruction Operation (continued)

	ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
★19.	Direct Indirect Long—(d)		1	1	1	1	1	PBR,PC	Op Code	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 Op Codes)		3.	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4	1	1	1	0	0,D+DO+1	AAH	1
	(6,7 and 8 cycles)		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)		1	1	1	1	1	PBR,PC	Op Code	1
	(JMP)		2	1	1	0	1	PBR,PC+1	AAL	1
	(1 Op Code)		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	New Op Code	1
★20b	Absolute Indexed Indirect		1	1	1	1	1	PBR,PC	Op Code	1
	(Jump to Subroutine Indexed		2	1	1	0	1	PBR,PC+1	AAL	1
	Indirect) —(a,x)		3	1	1	1	0	0,S	PCH	0
	(JSR)		4	1	1	1	0	0,S-1	PCL	0
	(1 Op Code)		5	1	1	0	1	PBR,PC+2	AAH	1
	(3 bytes)		6.	1	1	0	0	PBR,PC+2	IO	1
	(8 cycles)		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 Op Code	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 Op Code	1
21b	Stack (Software		1	1	1	1	1	PBR,PC	Op Code	1
	Interrupts) —s	(3)	2	1	1	0	1	PBR,PC+1	Signature	1
	(BRK,COP)	(7)	3	1	1	1	0	0,S	PBR	0
	(2 Op Codes)		4	1	1	1	0	0,S-1	PCH	0
	(2 bytes)		5	1	1	1	0	0,S-2	PCL	0
	(7 and 8 cycles)		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	New Op Code	1
21c	Stack (Return from		1	1	1	1	1	PBR,PC	Op Code	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	0,S+1	P	1
	(1 byte)		5	1	1	1	0	0,S+2	New PCL	1
	(6 and 7 cycles)		6	1	1	1	0	0,S+3	New PCH	1
	(different order from N6502)	(7)	7	1	1	1	0	0,S+4	PBR	1
			1	1	1	1	1	PBR,New PC	New Op Code	1
21d	Stack (Return from		1	1	1	1	1	PBR,PC	Op Code	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	0,S+1	New PCL-1	1
	(1 byte)		5	1	1	1	0	0,S+2	New PCH	1
	(6 cycles)		6	1	1	0	0	0,S+2	IO	1
			1	1	1	1	1	PBR,New PC	New Op Code	1
★21e	Stack (Return from		1	1	1	1	1	PBR,PC	Op Code	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	0,S+1	NEW PCL	1
	(1 byte)		5	1	1	1	0	0,S+2	NEW PCH	1
	(6 cycles)		6	1	1	1	0	0,S+3	NEW PBR	1
			1	1	1	1	1	NEW PBR,PC	New Op Code	1
21f	Stack (Push) —s		1	1	1	1	1	PBR,PC	Op Code	1
	(PHP,PHA,PHY,PHX,		2	1	1	0	0	PBR,PC+1	IO	1
	PHD,PHK,PHB)	(1)	3a	1	1	1	0	0,S	Register High	0
	(7 Op Codes)		3	1	1	1	0	0,S-1	Register Low	0
	(1 byte)									
	(3 and 4 cycles)									
21g	Stack (Pull) —s		1	1	1	1	1	PBR,PC	Op Code	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	0,S+1	Register Low	1
	(1 byte)	(1)	4a	1	1	1	0	0,S+2	Register High	1
	(4 and 5 cycles)									
★21h	Stack (Push Effective		1	1	1	1	1	PBR,PC	Op Code	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	0,D+DO	AAL	1
	(2 bytes)		4	1	1	1	0	0 D+DO+1	AAH	1
	(6 and 7 cycles)		5	1	1	1	0	0,S	AAH	0
			6	1	1	1	0	0 S-1	AAL	0
★21i	Stack (Push Effective		1	1	1	1	1	PBR,PC	Op Code	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	Op Code	1
	Program Counter Relative		2	1	1	0	1	PBR,PC+1	Offset Low	1
	Address) —s		3	1	1	0	1	PBR,PC+2	Offset High	1
	(PER)		4	1	1	0	0	PBR,PC+2	IO	1
	(1 Op Code)		5	1	1	1	0	0,S	PCH + Offset + CARRY	0
	(3 bytes)									
	(6 cycles)		6	1	1	1	0	0,S 1	PCL + Offset	0
★22	Stack Relative —d,s		1	1	1	1	1	PBR,PC	Op Code	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	Data Low	1/0
	(2 bytes)	(1)	4a	1	1	1	0	0,S+SO+1	Data High	1/0
	(4 and 5 cycles)									

	ADDRESS MODE		CYCLE	$\overline{VP}$,	$\overline{ML}$,	VDA,	VPA	ADDRESS BUS	DATA BUS	R/$\overline{W}$
★23.	Stack Relative Indirect		1	1	1	1	1	PBR,PC	Op Code	1
	Indexed —(d,s),y		2	1	1	0	1	PBR,PC+1	SO	1
	(ORA,AND,EOR,ADC,		3	1	1	0	0	PBR+PC+1	IO	1
	STA,LDA,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 and 8 Cycles)		7	1	1	1	0	DBR,AA+Y	Data Low	1/0
		(1)	7a	1	1	1	0	DBR,AA+Y+1	Data High	1/0
★24a.	Block Move Positive		1	1	1	1	1	PBR,PC	Op Code	1
	(forward) —xyc		2	1	1	0	1	PBR,PC+1	DBA	1
	(MVP)		3	1	1	0	1	PBR,PC+2	SBA	1
	(1 Op Code)	N-2	4	1	1	1	0	SBA,X	Source 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
	x - Source Address		7	1	1	0	0	DBA,Y	IO	1
	y - Destination		1	1	1	1	1	PBR,PC	Op Code	1
	c =Number of Bytes to Move -1		2	1	1	0	1	PBR,PC+1	DBA	1
	x,y Decrement		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	Source Data	1
	destination start address	Byte	5	1	1	1	0	DBA,Y-1	Dest Data	0
	is higher (more positive)	C-1	6	1	1	0	0	DBA,Y 1	IO	1
	than the source start address		7	1	1	0	0	DBA,Y-1	IO	1
			1	1	1	1	1	PBR,PC	Op Code	1
	FFFFFF		2	1	1	0	1	PBR,PC+1	DBA	1
	Dest Start	N Byte	3	1	1	0	1	PBR,PC+2	SBA	1
		Last	4	1	1	1	0	SBA,X-2	Source Data	1
	Source Start	C=0	5	1	1	1	0	DBA,Y-2	Dest Data	0
	Dest End		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 Op Code	1
★24b	Block Move Negative		1	1	1	1	1	PBR,PC	Op Code	1
	(backward) —xyc		2	1	1	0	1	PBR,PC+1	DBA	1
	(MVN)	N-2	3	1	1	0	1	PBR,PC+2	SBA	1
	(1 Op Code)	Byte	4	1	1	1	0	SBA,X	Source Data	1
	(3 bytes)	C-2	5	1	1	1	0	DBA,Y	Dest Data	0
	(7 cycles)		6	1	1	0	0	DBA,Y	IO	1
	x = Source Address		7	1	1	0	0	DBA,Y	IO	1
	y - Destination									
	c = Number of Bytes to Move -1		1	1	1	1	1	PBR,PC	Op Code	1
	x,y Increment		2	1	1	0	1	PBR,PC+1	DBA	1
	FFFFFF	N-1	3	1	1	0	1	PBR,PC+2	SBA	1
	Source End	Byte	4	1	1	1	0	SBA,X+1	Source Data	1
		C=1	5	1	1	1	0	DBA,Y+1	Dest Data	0
	Dest End		6	1	1	0	0	DBA,Y+1	IO	1
	Source Start		7	1	1	0	0	DBA,Y+1	IO	1
	Dest Start		1	1	1	1	1	PBR,PC	Op Code	1
			2	1	1	0	1	PBR,PC+1	DBA	1
	000000	N Byte	3	1	1	0	1	PBR,PC+2	SBA	1
		C-0	4	1	1	1	0	SBA,X+2	Source Data	1
	MVN is used when the		5	1	1	1	0	DBA,Y+2	Dest Data	0
	destination start address		6	1	1	0	0	DBA,Y+2	IO	1
	is lower (more negative)		7	1	1	0	0	DBA,Y+2	IO	1
	than the source start		1	1	1	1	1	PBR,PC+3	New Op Code	1
	address									

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

(10) R/$\overline{W}$ remains high during Reset.

(11) BRK bit 4 equals "0" in Emulation mode.

Abbreviations

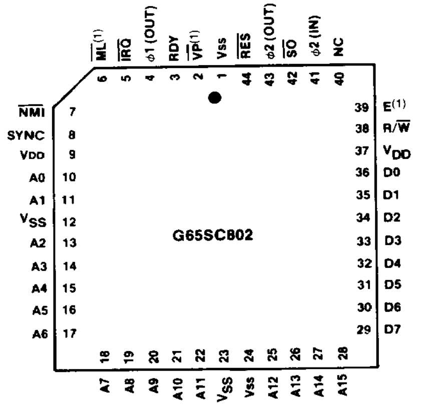
AAB	Absolute Address Bank
AAH	Absolute Address High
AAL	Absolute Address Low
AAVH	Absolute Address Vector High
AAVL	Absolute Address Vector Low
C	Accumulator
D	Direct Register
DBA	Destination Bank Address
DBR	Data Bank Register
DO	Direct Offset
IDH	Immediate Data High
IDL	Immediate Data Low
IO	Internal 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 G65SC816/802 Addressing Modes
●	New G65SC02 Addressing Modes
Blank	NMOS 6502 Addressing Modes

Pin Function Table

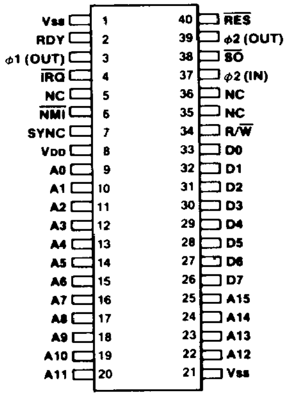
Pin	Description
A0-A15	Address Bus
$\overline{\text{ABORT}}$	Abort Input
BE	Bus Enable
φ2 (IN)	Phase 2 In Clock
φ1 (OUT)	Phase 1 Out Clock
φ2 (OUT)	Phase 2 Out Clock
D0-D7	Data Bus (G65SC802)
D0/BA0-D7/BA7	Data Bus, Multiplexed (G65SC816)
E	Emulation Select
$\overline{\text{IRQ}}$	Interrupt Request
$\overline{\text{ML}}$	Memory Lock
M/X	Mode Select (PM or Px)

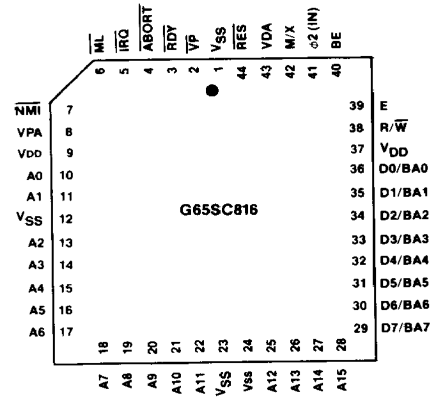
Pin	Description
NC	No Connection
$\overline{\text{NMI}}$	Non-Maskable Interrupt
RDY	Ready
$\overline{\text{RES}}$	Reset
R/$\overline{\text{W}}$	Read/Write
$\overline{\text{SO}}$	Set Overflow
SYNC	Synchronize
VDA	Valid Data Address
$\overline{\text{VP}}$	Vector Pull
VPA	Valid Program Address
VDD	Positive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

Pin Configuration

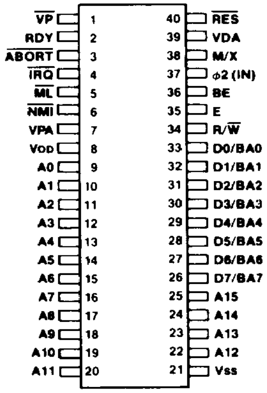


G65SC802



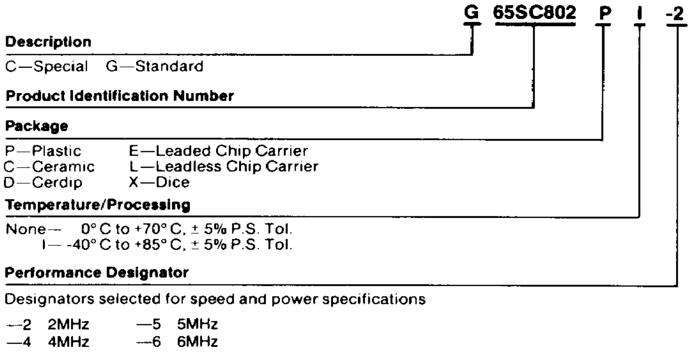


G65SC816



Notes:
1. New signal pins on G65SC802 not available on 40-pin DIP version.

Ordering Information



G 65SC802 P I -2

Description

C—Special G—Standard

Product Identification Number

Package

P—Plastic E—Leaded Chip Carrier
C—Ceramic L—Leadless Chip Carrier
D—Cerdip X—Dice

Temperature/Processing

None— 0° C to +70° C, ± 5% P.S. Tol.
I— -40° C to +85° C, ± 5% P.S. Tol.

Performance Designator

Designators selected for speed and power specifications

—2 2MHz —5 5MHz
—4 4MHz —6 6MHz