



OpenCores.Org

R65C02 IP Core Specification

Author: Jens Gutschmidt
opencores@vivare-services.com

Rev. 1.00
October 14, 2018

This page has been intentionally left blank.

Revision History

| Rev | Date | Author | Description |
|------|-------------|-----------------|--|
| . | | | |
| 0.1 | 12/18/06 | Jens Gutschmidt | First Draft |
| 0.2 | 01/02/07 | Jens Gutschmidt | Pictures of FSM's |
| 0.3 | 08/20/08 | Jens Gutschmidt | <ul style="list-style-type: none">- Tables and timing diagrams- Deleting pictures of FSM |
| 0.4 | 10/01/08 | Jens Gutschmidt | <ul style="list-style-type: none">- New ideas for timing diagrams |
| 0.5 | 01/02/09 | Jens Gutschmidt | <ul style="list-style-type: none">- Textual changes / spell checking- Insert R6502_TC block diagram |
| 0.6 | 02/01/09 | Jens Gutschmidt | <ul style="list-style-type: none">- Work on Timing Diagrams |
| 0.7 | 23/02/09 | Jens Gutschmidt | <ul style="list-style-type: none">- Added new instruction codes (Timing diagrams for all new op codes missing) |
| 0.8 | 16/04/14 | Jens Gutschmidt | <ul style="list-style-type: none">- Corrected description for BIT- Corrected timing for JSR, RTS |
| 0.9 | 09-Sep-2018 | Jens Gutschmidt | <ul style="list-style-type: none">- Corrected ADC/SBC descriptions (D=1 -> 1 additional cycle)- Corrected all branch descriptions (page crossing computed wrong)- PHP writes always E&B = '1' to stack- Adding Interrupt section- Adding ADC / SBC Decimal Mode |
| 0.91 | 04-Oct-2018 | Jens Gutschmidt | <ul style="list-style-type: none">- Corrected the instruction table (all ABS,X ABS,Y and (IND),Y address modes are depend on page crossing – wrong in vendor's documentation) |
| 1.00 | 14-Oct-2018 | Jens Gutschmidt | <ul style="list-style-type: none">- Section "Timing Diagrams" has been deleted |

Contents

| | | |
|----------|---------------------------|-----------|
| 1 | INTRODUCTION | 1 |
| 2 | ARCHITECTURE | 2 |
| 3 | OPERATION | 6 |
| | <i>ADC.....</i> | <i>7</i> |
| | <i>AND.....</i> | <i>8</i> |
| | <i>ASL.....</i> | <i>9</i> |
| | <i>BBR.....</i> | <i>10</i> |
| | <i>BBS.....</i> | <i>11</i> |
| | <i>BCC.....</i> | <i>12</i> |
| | <i>BCS.....</i> | <i>13</i> |
| | <i>BEQ.....</i> | <i>14</i> |
| | <i>BIT.....</i> | <i>15</i> |
| | <i>BMI.....</i> | <i>16</i> |
| | <i>BNE.....</i> | <i>17</i> |
| | <i>BPL.....</i> | <i>18</i> |
| | <i>BRA.....</i> | <i>19</i> |
| | <i>BRK.....</i> | <i>19</i> |
| | <i>BVC.....</i> | <i>20</i> |
| | <i>BVS.....</i> | <i>21</i> |
| | <i>CLC.....</i> | <i>22</i> |
| | <i>CLD.....</i> | <i>22</i> |
| | <i>CLI.....</i> | <i>23</i> |
| | <i>CLV.....</i> | <i>23</i> |
| | <i>CMP.....</i> | <i>24</i> |
| | <i>CPX.....</i> | <i>24</i> |
| | <i>CPY.....</i> | <i>25</i> |
| | <i>DEC.....</i> | <i>25</i> |
| | <i>DEX.....</i> | <i>26</i> |
| | <i>DEY.....</i> | <i>26</i> |
| | <i>EOR.....</i> | <i>27</i> |
| | <i>INC.....</i> | <i>28</i> |
| | <i>INX.....</i> | <i>28</i> |
| | <i>INY.....</i> | <i>29</i> |
| | <i>JMP.....</i> | <i>29</i> |
| | <i>JSR.....</i> | <i>30</i> |
| | <i>LDA.....</i> | <i>30</i> |
| | <i>LDX.....</i> | <i>31</i> |
| | <i>LDY.....</i> | <i>31</i> |
| | <i>LSR.....</i> | <i>32</i> |
| | <i>NOP.....</i> | <i>32</i> |
| | <i>ORA.....</i> | <i>33</i> |
| | <i>PHA.....</i> | <i>33</i> |
| | <i>PHP.....</i> | <i>34</i> |
| | <i>PHX.....</i> | <i>34</i> |
| | <i>PHY.....</i> | <i>34</i> |
| | <i>PLA.....</i> | <i>34</i> |
| | <i>PLP.....</i> | <i>35</i> |
| | <i>PLX.....</i> | <i>35</i> |

| | |
|--|-----------|
| <i>PLY</i> | 35 |
| <i>ROL</i> | 36 |
| <i>ROR</i> | 36 |
| <i>RMB</i> | 37 |
| <i>RTI</i> | 37 |
| <i>RTS</i> | 37 |
| <i>SBC</i> | 38 |
| <i>SEC</i> | 38 |
| <i>SED</i> | 39 |
| <i>SEI</i> | 39 |
| <i>SMB</i> | 40 |
| <i>STA</i> | 40 |
| <i>STX</i> | 41 |
| <i>STY</i> | 41 |
| <i>STZ</i> | 41 |
| <i>TAX</i> | 42 |
| <i>TAY</i> | 42 |
| <i>TRB</i> | 43 |
| <i>TSB</i> | 44 |
| <i>TSX</i> | 44 |
| <i>TXA</i> | 45 |
| <i>TXS</i> | 45 |
| <i>TYA</i> | 46 |
| 4 | 47 |
| <i>Registers</i> | 47 |
| <i>List of Registers</i> | 47 |
| <i>PSW – Description</i> | 47 |
| 5 | 48 |
| <i>Clocks</i> | 48 |
| 6 | 49 |
| <i>Interrupts</i> | 49 |
| <i>NMI</i> | 49 |
| <i>IRQ</i> | 50 |
| <i>BRK</i> | 50 |
| APPENDIX A | 51 |
| ADC / SBC DECIMAL MODE | 51 |
| INSTRUCTION TABLE | 53 |
| INDEX | 54 |
| LIST OF FIGURES | |
| Figure (1): R65C02 architecture..... | 2 |
| Figure (2): R65C02_TC IP core architecture | 5 |
| Figure (3): Interrupt NMI – Timing Diagram | 49 |
| Figure (4): Interrupt IRQ – Timing Diagram | 50 |

LIST OF TABLES

| | |
|---|----|
| Table (1): ADC – Short Reference | 7 |
| Table (2): AND – Short Reference | 8 |
| Table (3): ASL – Short Reference | 9 |
| Table (4): BBR – Short Reference | 10 |
| Table (5): BBS – Short Reference | 11 |
| Table (6): BCC – Short Reference | 12 |
| Table (7): BCS – Short Reference | 13 |
| Table (8): BEQ – Short Reference | 14 |
| Table (9): BIT – Short Reference | 15 |
| Table (10): BMI – Short Reference | 16 |
| Table (11): BNE – Short Reference | 17 |
| Table (12): BPL – Short Reference | 18 |
| Table (13): BRA – Short Reference | 19 |
| Table (14): BRK – Short Reference | 19 |
| Table (15): BVC – Short Reference | 20 |
| Table (16): BVS – Short Reference | 21 |
| Table (17): CLC – Short Reference | 22 |
| Table (18): CLD – Short Reference | 22 |
| Table (19): CLI – Short Reference | 23 |
| Table (20): CLV – Short Reference | 23 |
| Table (21): CMP – Short Reference | 24 |
| Table (22): CPX – Short Reference | 24 |
| Table (23): CPY – Short Reference | 25 |
| Table (24): DEC – Short Reference | 25 |
| Table (25): DEX – Short Reference | 26 |
| Table (26): DEX – Short Reference | 26 |
| Table (27): EOR – Short Reference | 27 |
| Table (28): INC – Short Reference | 28 |
| Table (29): INX – Short Reference | 28 |
| Table (30): INY – Short Reference | 29 |
| Table (31): JMP – Short Reference | 29 |
| Table (32): JSR – Short Reference | 30 |
| Table (33): LDA – Short Reference | 30 |

| | |
|---|----|
| Table (34): LDX – Short Reference | 31 |
| Table (35): LDY – Short Reference | 31 |
| Table (36): LSR – Short Reference | 32 |
| Table (37): NOP – Short Reference | 32 |
| Table (38): OR – Short Reference | 33 |
| Table (39): PHA – Short Reference | 33 |
| Table (40): PHP – Short Reference | 34 |
| Table (41): PHX – Short Reference | 34 |
| Table (42): PHY – Short Reference | 34 |
| Table (43): PLA – Short Reference | 34 |
| Table (44): PLP – Short Reference | 35 |
| Table (45): PLX – Short Reference | 35 |
| Table (46): PLY – Short Reference | 35 |
| Table (47): ROL – Short Reference | 36 |
| Table (48): ROR – Short Reference | 36 |
| Table (49): RMB – Short Reference | 37 |
| Table (50): RTI – Short Reference | 37 |
| Table (51): RTS – Short Reference | 37 |
| Table (52): SBC – Short Reference | 38 |
| Table (53): SEC – Short Reference | 38 |
| Table (54): CLD – Short Reference | 39 |
| Table (55): SEI – Short Reference | 39 |
| Table (56): SMB – Short Reference | 40 |
| Table (57): STA – Short Reference | 40 |
| Table (58): STX – Short Reference | 41 |
| Table (59): STY – Short Reference | 41 |
| Table (60): STZ – Short Reference | 41 |
| Table (61): TAX – Short Reference | 42 |
| Table (62): TAY – Short Reference | 42 |
| Table (63): TRB – Short Reference | 43 |
| Table (64): TSB – Short Reference | 44 |
| Table (65): TSX – Short Reference | 44 |
| Table (66): TXA – Short Reference | 45 |

| | |
|---|----|
| Table (67): TXS – Short Reference | 45 |
| Table (68): TYA – Short Reference | 46 |

1 Introduction

The Central Processing Unit (CPU) 6502 was introduced at 1976 by Commodore Computers. It is an 8 Bit processor which is well known worldwide. The also most known computer system in the 70s/80s was the APPLE based on this famous 6502.

In this century building of little computer systems based on an 8 Bit architecture is easier than ever before. Many CAD/CAM tool are existing on the market to give you help to do this job.

In the last decade the technology give us more and more possibilities to reach our dreams – higher, faster, wider. FPGA's (Field Programmable Gate Arrays) and CPLD's (Complex Programmable Logic Devices) are coming up. They shorten the time of development, simulation and verification of such systems dramatically. On the other hand stands the rising complex of designs and well knowing of desired description languages – VHDL (Very high scale integration Hardware Description Language) or Verilog.

Many operations and functions of computer tasks aren't need really the power of “modern” processors today like Intel's Pentium. Nevertheless it is not usable – sometimes also impossible - to build simple systems with discrete IC's. The necessity of flexibility compel us the using of high tech parts and tools...to build a “simple” system.

The using of standard IP's (Intellectual Properties) is the key to reach that goal.

Now, talking about the specification of the **R65C02 True Cycle Core IP**. Written in VHDL but developed with Mentor's HDL Designer. The R65C02 is very easy to handle for people were have experience with other cpu. Also suitable for beginners who will learning to build her own first cpu system.

2 Architecture

The following figure shows the internal architecture of the Rockwell's R65C02.

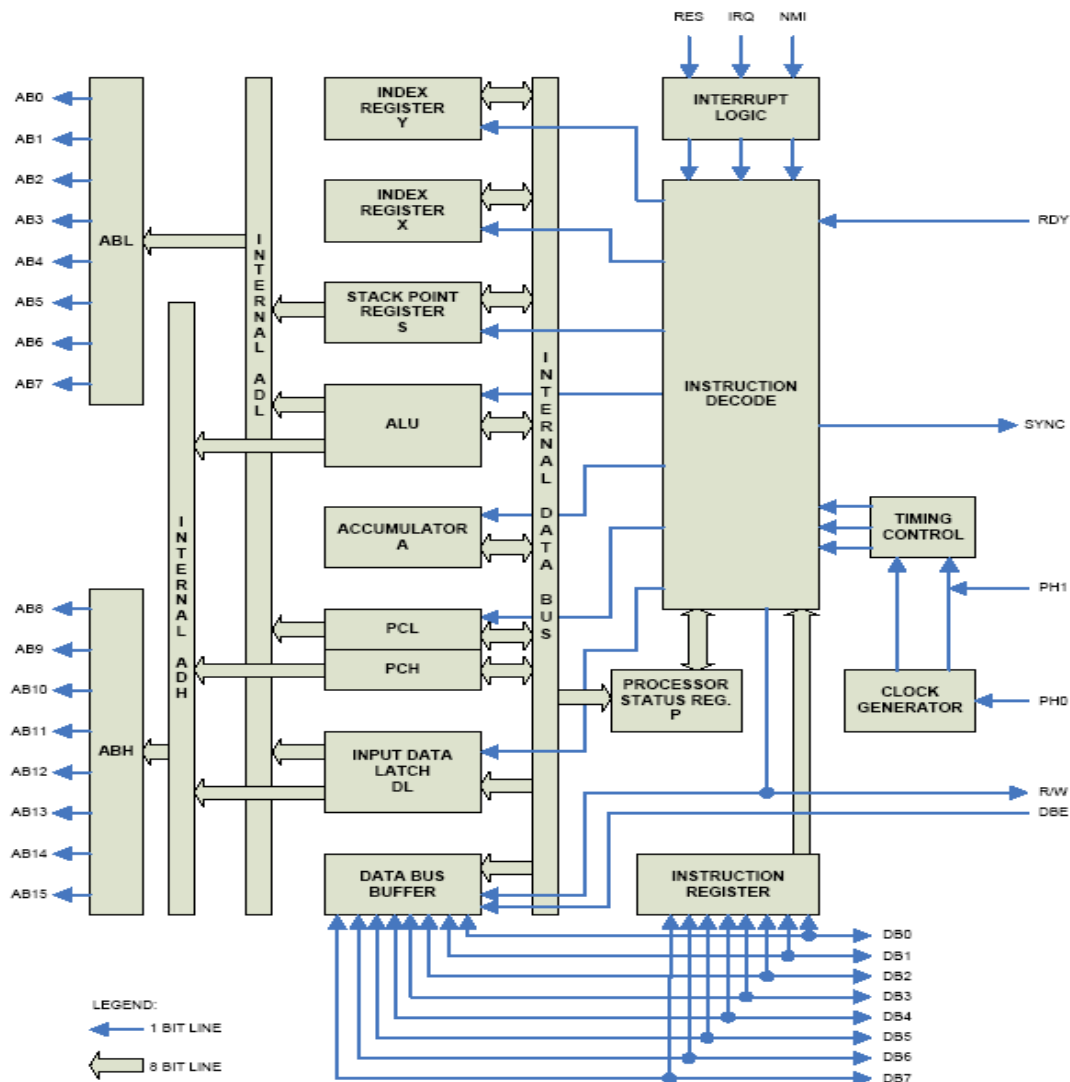


Figure (1): R65C02 architecture

First, for more clearance and better understanding of the **R65C02 True Cycle Core IP** let me summarize the most important quality of Rockwell's R65C02.

This architecture based on asynch logic. The two phase clock is a special attribute of that. The Data Bus is only active while clock phase PH1 is "1" for reading and writing (PH0 is "0"). All transfers of data are occurring at active phase of PH1.

The whole R65C02 is fully statically, so all registers holding her values if the clocking will be going to D.C. Internally all operation of 16 Bit are splitted into two 8 Bit busses. This is very interesting and causes one more clock cycle in some operations (see “PCL”, “PCH” -> Program Counter Low/High) which can operate over page boundaries.

Address lines are 16 Bit wide. 65.536 addresses for byte access are possible. There are no differences exist between memory and I/O cycles. All I/O devices must be connected via memory mapped I/O. The mnemonics are also all the same for both. The usage of address space is normally organized from top to the button “ROM – I/O – RAM” (0xFFFF – 0x0000).

Only one pin $\overline{R/W}$ controls the read and writes operations.

The R65C02 can hold in every cycle by applying RDY. Wait states can be generated by using RDY.

There are two interrupts \overline{IRQ} and \overline{NMI} . \overline{IRQ} is mask able, \overline{NMI} is not. Note that \overline{RES} is internally handled as an interrupt, but is resetting the R65C02 to the starting point of operation at vector address 0xFFFF. Every interrupt has its own vector which can point elsewhere to the 16 Bit address space. The vectors are located from 0xFFFF to 0xFFFA.

The Stack Pointer is 256 Bytes deep and is hard-wired to 0x01FF (internally 0x0200 – 0x01FF going to the address bus) for the first memory location and grows downward to 0x0100. Decrementing again will produce a wraparound back to 0x01FF.

Arithmetical operations like ADC and SBC can handle binary values from 0-255 and decimal values from 0-99 without using other op codes. Only one user changeable bit into the Status Register determine the exact arithmetic mode of that operations.

The R65C02 generally operates into a pipeline mode. That means that a finish of an OP code falls every time into a fetch cycle of the next OP. This save one clock cycle

Every operation consume between one and eight clock cycles.

To build a useful and backward software and hardware compatible VHDL Core for the R65C02 many unavoidable changes may be forced to simulation and verification before any real core will be made.

The requirements:

- Vendor independent for implementation in any FPGA
- True Cycle for all operations as described in original publications

Because the original publications until late 80's shows wrong cycle counts for some R65C02 instructions, the r65c02_tc's cycle timing was finally referenced against [4] (Programming the 65816).

- No "Mixed Mode" – Only R65C02, not R6502 -> for performance and area
- No fantastic or useful extensions – as like the original R65C02 as possible
- Only one clock source
- Full synch design
- Operating speed $f_{max} > 100\text{MHz}$ for most public used FPGA devices
- Easily to change for future requirement – building variants
- The activity on data and address busses since the execution of OP codes is not forced to be like the original – may be or maybe not.

The design based on finite state machines (fsm). Some registers are for internally use only and stores temporally values. These registers are not accessible by the programmer.

The next picture shows the hierarchical structure of the R65C02 IP core.

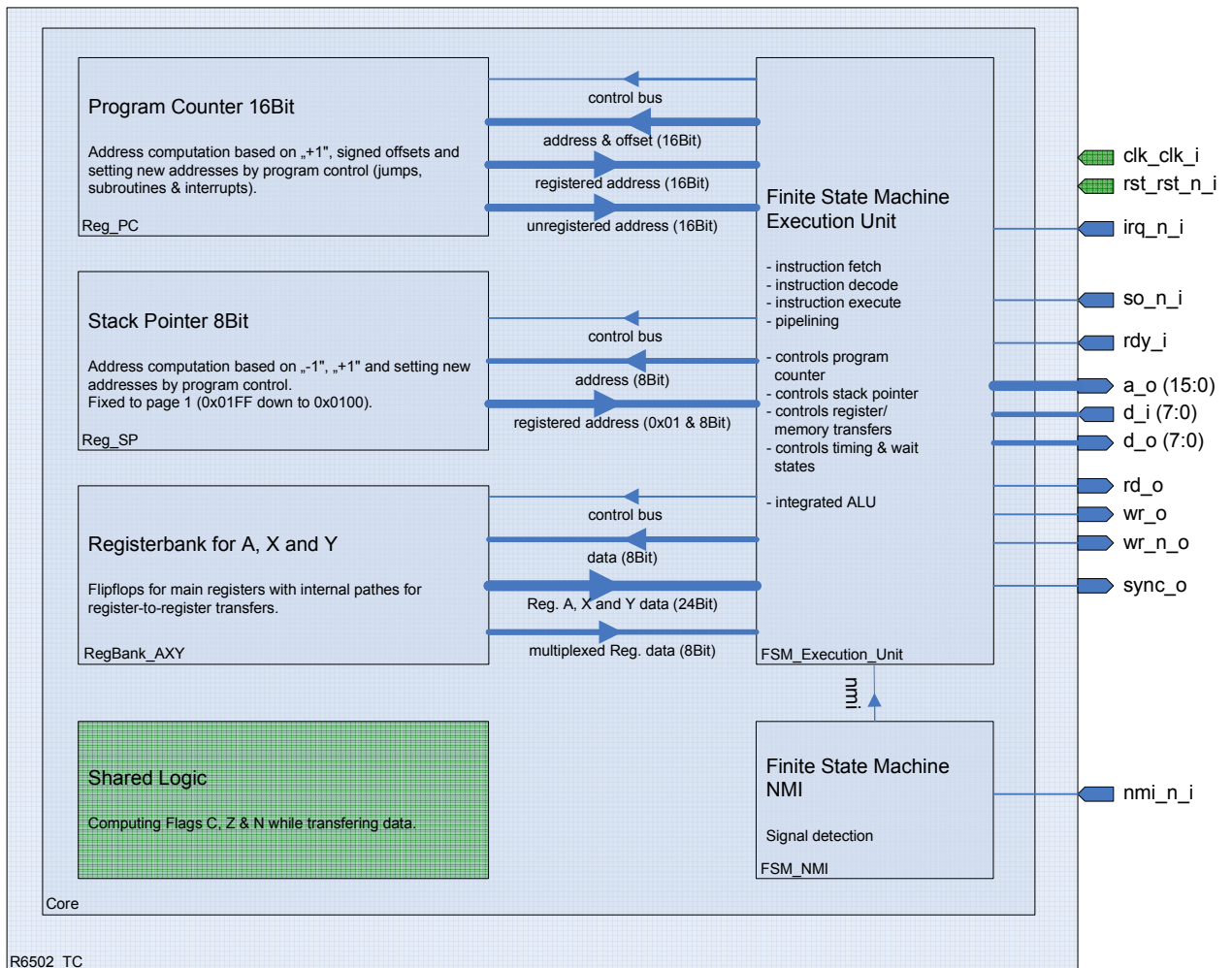


Figure (2): R65C02_TC IP core architecture

3 Operation

ADC

Operation: Add memory or immediate value to accumulator A with carry ($A + M + C \rightarrow A, C$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | ADC # Oper | 69 | 2 | 2+ | √ | √ | - | - | - | - | √ | √ |
| Zero Page | ADC Oper | 65 | 2 | 3+ | √ | √ | - | - | - | - | √ | √ |
| Zero Page, X | ADC Oper, X | 75 | 2 | 4+ | √ | √ | - | - | - | - | √ | √ |
| Absolute | ADC Oper | 6D | 3 | 4+ | √ | √ | - | - | - | - | √ | √ |
| Absolute, X | ADC Oper, X | 7D | 3 | 4+* | √ | √ | - | - | - | - | √ | √ |
| Absolute, Y | ADC Oper, Y | 79 | 3 | 4+* | √ | √ | - | - | - | - | √ | √ |
| (Indirect, X) | ADC (Oper, X) | 61 | 2 | 6+ | √ | √ | - | - | - | - | √ | √ |
| (Indirect), Y | ADC (Oper), Y | 71 | 2 | 5+* | √ | √ | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | ADC (Oper) | 72 | 2 | 5+ | √ | √ | - | - | - | - | √ | √ |

* => Add 1 if page boundary is crossed

+ => Add 1 if mode is "DECIMAL" (D='1')

Table (1): ADC – Short Reference

Example Immediate (assumed A=\$FE, C=0, D=0):

| Address | Bytes | Mnemonic |
|---------|-------|---|
| \$9000 | 69 03 | ADC #\$03 ; |
| \$9002 | ... | next OP ; A is now \$01, C=1, N=0, Z=0, V=0 |

1111 1110 (\$FE)
ADC 0000 0011 (\$03)
0000 0001 (\$01), C=1, N=0, Z=0, V=0

AND

Operation: “AND” memory or immediate value with accumulator A ($A \cap M \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | AND #Oper | 29 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | AND Oper | 25 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | AND Oper, X | 35 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | AND Oper | 2D | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, X | AND Oper, X | 3D | 3 | 4* | √ | - | - | - | - | - | √ | - |
| Absolute, Y | AND Oper, Y | 39 | 3 | 4* | √ | - | - | - | - | - | √ | - |
| (Indirect, X) | AND (Oper, X) | 21 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| (Indirect), Y | AND (Oper), Y | 31 | 2 | 5* | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | AND (Oper) | 32 | 2 | 5 | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (2): AND – Short Reference

Example Immediate (assumed A is \$C5):

| Address | Bytes | Mnemonic | |
|---------|-------|-----------|---------------------------|
| \$9000 | 29 71 | AND #\$71 | ; |
| \$9002 | ... | next OP | ; A is now \$41, N=0, Z=0 |

```

1100 0101 ($C5)
AND 0111 0001 ($71)
0100 0001 ($41), N=0, Z=0

```

ASL

Operation: Shift Left One Bit (Memory or Accumulator) ($C \leftarrow \boxed{7} \boxed{6} \boxed{5} \boxed{4} \boxed{3} \boxed{2} \boxed{1} \boxed{0} \leftarrow 0$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator | ASL A | 0A | 1 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | ASL Oper | 06 | 2 | 5 | √ | - | - | - | - | - | √ | √ |
| Zero Page, X | ASL Oper, X | 16 | 2 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute | ASL Oper | 0E | 3 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute, X | ASL Oper, X | 1E | 3 | 7 | √ | - | - | - | - | - | √ | √ |

Table (3): ASL – Short Reference

Example Accumulator (assumed A=\$55):

| Address | Bytes | Mnemonic | |
|---------|-------|----------|---------------------------|
| \$9000 | 0A | ASL A | ; |
| \$9001 | ... | next OP | ; A is now \$AA, N=1, Z=0 |

ASL 0101 0101 (\$55)
1010 1010 (\$AA), C=0, N=1, Z=0

BBR

Operation: Branch on memory bit x is reset, where x=0-7 MSB of op code (Branch on Mx = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BBR(0-7) ZP,offs8 | 0F-7F | 3 | 5* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (4): BBR – Short Reference

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|---|
| \$9000 | 2F 32 10 | BBR2 \$32, \$10 | ; BBR to same page, - NO BRANCH - jumps to \$9003 |
| \$9003 | ... | <i>next OP</i> | ; M2(\$0032) = 1 => no branch, 5 cycles |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|--|
| \$90FF | 5F 32 10 | BBR5 \$32, \$10 | ; BBR to different page, - NO BRANCH – jumps to \$9102 |
| \$9102 | ... | <i>next OP</i> | ; M5(\$0032) = 1 => no branch, 5 cycles (!!!) |

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|---|
| \$9000 | 0F 32 10 | BBR0 \$32, \$10 | ; BBR to same page, - BRANCH - jumps to \$9003 + \$10 |
| ... | | | |
| \$9013 | ... | <i>next OP</i> | ; M0(\$0032)=0 => branch, 6 cycles |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|--|
| \$90FC | 0F 32 10 | BBR0 \$32, \$10 | ; BBR to different page, - BRANCH - jumps to \$900F + \$10 |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; M0(\$0032)=0 => branch, 7 cycles |

BBS

Operation: Branch on memory bit x is set, where x=0-7 MSB-8 of op code (Branch on Mx = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BBS(0-7) ZP,offs8 | 8F-FF | 3 | 5* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (5): BBS – Short Reference

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|---|
| \$9000 | 8F 32 10 | BBS0 \$32, \$10 | ; BBS to same page, - NO BRANCH - jumps to \$9003 |
| \$9003 | ... | <i>next OP</i> | ; M0(\$0032) =0 => no branch, 5 cycles |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|--|
| \$90FF | FF 32 10 | BBS7 \$32, \$10 | ; BBS to different page, - NO BRANCH – jumps to \$9102 |
| \$9102 | ... | <i>next OP</i> | ; M7(\$0032) =0 => no branch, 5 cycles (!!!) |

Example Relative (assumed \$0032=\$24, same page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|---|
| \$9000 | AF 32 10 | BBS2 \$32, \$10 | ; BBS to same page, - BRANCH - jumps to \$9003 + \$10 |
| ... | | | |
| \$9013 | ... | <i>next OP</i> | ; M2(\$0032)=1 => branch, 6 cycles |

Example Relative (assumed \$0032=\$24, different page):

| Address | Bytes | Mnemonic | |
|---------|----------|-----------------|--|
| \$90FC | DF 32 10 | BBS5 \$32, \$10 | ; BBS to different page, - BRANCH - jumps to \$900F + \$10 |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; M5(\$0032)=1 => branch, 7 cycles |

BCC

Operation: Branch on Carry Clear (Branch on C = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BCC Oper | 90 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (6): BCC – Short Reference

Example Relative (assumed C=1, no branch):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 90 10 | BCC \$10 | ; BCC to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; C=1 => no branch, 2 cycles |

Example Relative (assumed C=1, no branch):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FF | 90 10 | BCC \$10 | ; BCC to same page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; C=1 => no branch, 2 cycles |

Example Relative (assumed C=0, branch to same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 90 10 | BCC \$10 | ; BCC to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; C=0 => branch, 3 cycles |

Example Relative (assumed C=0, branch to different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | 90 10 | BCC \$10 | ; BCC to different page, - BRANCH - jumps to \$9109 |
| ... | | | |
| \$9109 | ... | <i>next OP</i> | ; C=0 => branch, 4 cycles |

BCS

Operation: Branch on Carry Set (Branch on C = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BCS Oper | B0 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (7): BCS – Short Reference

Example Relative (assumed C=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | B0 10 | BCS \$10 | ; BCS to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; C=0 => no branch, 2 cycles |

Example Relative (assumed C=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | B0 10 | BCS \$10 | ; BCS to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; C=0 => no branch, 2 cycles (!!!) |

Example Relative (assumed C=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | B0 10 | BCS \$10 | ; BCS to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; C=1 => branch, 3 cycles |

Example Relative (assumed C=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | B0 10 | BCS \$10 | ; BCS to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; C=1 => branch, 4 cycles |

BEQ

Operation: Branch on result zero (Branch on Z = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BEQ Oper | F0 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (8): BEQ – Short Reference

Example Relative (assumed Z=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | F0 10 | BEQ \$10 | ; BEQ to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; Z=0 => no branch, 2 cycles |

Example Relative (assumed Z=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | F0 10 | BEQ \$10 | ; BEQ to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; Z=0 => no branch, 2 cycles (!!!) |

Example Relative (assumed Z=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | F0 10 | BEQ \$10 | ; BEQ to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; Z=1 => branch, 3 cycles |

Example Relative (assumed Z=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | F0 10 | BEQ \$10 | ; BEQ to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; Z=1 => branch, 4 cycles |

BIT

Operation (ZP and ABS): Bit 6 and 7 of the operand are transferred to the status register. If the result of $A \cap M = 0$ then $Z = 1$, otherwise $Z = 0$ ($M7 \rightarrow N$, $M6 \rightarrow V$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate <i>NEW</i> | BIT # Oper | 89 | 2 | 2 | - | - | - | - | - | - | √ | - |
| Zero Page | BIT Oper | 24 | 2 | 3 | 7 | 6 | - | - | - | - | √ | - |
| Zero Page, X <i>NEW</i> | BIT Oper, X | 34 | 2 | 4 | 7 | 6 | - | - | - | - | √ | - |
| Absolute | BIT Oper | 2C | 3 | 4 | 7 | 6 | - | - | - | - | √ | - |
| Absolute, X <i>NEW</i> | BIT Oper, X | 3C | 3 | 4* | 7 | 6 | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (9): BIT – Short Reference

In mode “Immediate” BIT leaves flags N&V unchanged. This is a special case for R65C02!

BMI

Operation: Branch on result minus (Branch on N = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BMI Oper | 30 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (10): BMI – Short Reference

Example Relative (assumed Z=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 30 10 | BMI \$10 | ; BMI to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; N=0 => no branch, 2 cycles |

Example Relative (assumed Z=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | 30 10 | BMI \$10 | ; BMI to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; N=0 => no branch, 2 cycles (!!!) |

Example Relative (assumed Z=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 30 10 | BMI \$10 | ; BMI to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; N=1 => branch, 3 cycles |

Example Relative (assumed Z=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | 30 10 | BMI \$10 | ; BMI to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; N=1 => branch, 4 cycles |

BNE

Operation: Branch on result not zero (Branch on Z = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BNE Oper | D0 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (11): BNE – Short Reference

Example Relative (assumed Z=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | D0 10 | BNE \$10 | ; BNE to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; Z=1 => no branch, 2 cycles |

Example Relative (assumed Z=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | D0 10 | BNE \$10 | ; BNE to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; Z=1 => no branch, 2 cycles (!!!) |

Example Relative (assumed Z=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | D0 10 | BNE \$10 | ; BNE to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; Z=0 => branch, 3 cycles |

Example Relative (assumed Z=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | D0 10 | BNE \$10 | ; BNE to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; Z=0 => branch, 4 cycles |

BPL

Operation: Branch on result plus (Branch on N = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BPL Oper | 10 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (12): BPL – Short Reference

Example Relative (assumed N=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 10 10 | BPL \$10 | ; BPL to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; N=1 => no branch, 2 cycles |

Example Relative (assumed N=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | 10 10 | BPL \$10 | ; BPL to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; N=1 => no branch, 2 cycles (!!!) |

Example Relative (assumed N=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 10 10 | BPL \$10 | ; BPL to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; N=0 => branch, 3 cycles |

Example Relative (assumed N=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | 10 10 | BPL \$10 | ; BPL to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; N=0 => branch, 4 cycles |

BRA

Operation: Branch always (Branch always)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|---------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative <i>NEW</i> | BRA Oper | 80 | 2 | 3* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to different page

Table (13): BRA – Short Reference

BRK

Operation: Forced Interrupt (PC + 2 ↓, P ↓ NV11DIZC, E & B are written as 1 to stack)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | BRK | 00 | 1 | 7 | - | - | - | - | 0 | 1 | - | - |

Table (14): BRK – Short Reference

BVC

Operation: Branch on no overflow (Branch on V = 0)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BVC Oper | 50 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (15): BVC – Short Reference

Example Relative (assumed V=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 50 10 | BVC \$10 | ; BVC to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; V=1 => no branch, 2 cycles |

Example Relative (assumed V=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | 50 10 | BVC \$10 | ; BVC to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; V=1 => no branch, 2 cycles (!!!) |

Example Relative (assumed V=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 50 10 | BVC \$10 | ; BVC to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; V=0 => branch, 3 cycles |

Example Relative (assumed V=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | 50 10 | BVC \$10 | ; BVC to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; V=0 => branch, 4 cycles |

BVS

Operation: Branch on overflow (Branch on V = 1)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Relative | BVS Oper | 70 | 2 | 2* | - | - | - | - | - | - | - | - |

* => Add 1 if branch occurs to same page

* => Add 2 if branch occurs to different page

Table (16): BVS – Short Reference

Example Relative (assumed V=0, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 70 10 | BVS \$10 | ; BVS to same page, - NO BRANCH - jumps to \$9002 |
| \$9002 | ... | <i>next OP</i> | ; V=0 => no branch, 2 cycles |

Example Relative (assumed V=0, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--|
| \$90FF | 70 10 | BVS \$10 | ; BVS to different page, - NO BRANCH - jumps to \$9101 |
| \$9101 | ... | <i>next OP</i> | ; V=0 => no branch, 2 cycles (!!!) |

Example Relative (assumed V=1, same page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 70 10 | BVS \$10 | ; BVS to same page, - BRANCH - jumps to \$9002 + \$10 |
| ... | | | |
| \$9012 | ... | <i>next OP</i> | ; V=1 => branch, 3 cycles |

Example Relative (assumed V=1, different page):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$90FD | 70 10 | BVS \$10 | ; BVS to different page, - BRANCH - jumps to \$910F |
| ... | | | |
| \$910F | ... | <i>next OP</i> | ; V=1 => branch, 4 cycles |

CLC

Operation: Clear Carry flag ($0 \rightarrow C$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | CLC | 18 | 1 | 2 | - | - | - | - | - | - | - | 0 |

Table (17): CLC – Short Reference

Example:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--------------|
| \$9000 | 18 | CLC | ; |
| \$9001 | ... | <i>next OP</i> | ; C is now 0 |

CLD

Operation: Clear Decimal flag ($0 \rightarrow D$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | CLD | D8 | 1 | 2 | - | - | - | - | 0 | - | - | - |

Table (18): CLD – Short Reference

Example:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--------------|
| \$9000 | D8 | CLD | ; |
| \$9001 | ... | <i>next OP</i> | ; D is now 0 |

CLI

Operation: Clear Interrupt Disable flag (0 → I)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | CLI | 58 | 1 | 2 | - | - | - | - | - | 0 | - | - |

Table (19): CLI – Short Reference

Example:

| Address | Bytes | Mnemonic |
|---------|-------|-----------------------------|
| \$9000 | 58 | CLI ; |
| \$9001 | ... | <i>next OP</i> ; I is now 0 |

CLV

Operation: Clear Overflow flag (0 → O)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | CLV | B8 | 1 | 2 | - | 0 | - | - | - | - | - | - |

Table (20): CLV – Short Reference

Example:

| Address | Bytes | Mnemonic |
|---------|-------|-----------------------------|
| \$9000 | B8 | CLV ; |
| \$9001 | ... | <i>next OP</i> ; V is now 0 |

CMP

Operation: Compare Memory and Accumulator (A - M)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | CMP # Oper | C9 | 2 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | CMP Oper | C5 | 2 | 3 | √ | - | - | - | - | - | √ | √ |
| Zero Page, X | CMP Oper, X | D5 | 2 | 4 | √ | - | - | - | - | - | √ | √ |
| Absolute | CMP Oper | CD | 3 | 4 | √ | - | - | - | - | - | √ | √ |
| Absolute, X | CMP Oper, X | DD | 3 | 4* | √ | - | - | - | - | - | √ | √ |
| Absolute, Y | CMP Oper, Y | D9 | 3 | 4* | √ | - | - | - | - | - | √ | √ |
| (Indirect, X) | CMP (Oper, X) | C1 | 2 | 6 | √ | - | - | - | - | - | √ | √ |
| (Indirect), Y | CMP (Oper), Y | D1 | 2 | 5* | √ | - | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | CMP (Oper) | D2 | 2 | 5 | √ | - | - | - | - | - | √ | √ |

* => Add 1 if page boundary is crossed

Table (21): CMP – Short Reference

CPX

Operation: Compare Memory and Index X (X - M)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | CPX # Oper | E0 | 2 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | CPX Oper | E4 | 2 | 3 | √ | - | - | - | - | - | √ | √ |
| Absolute | CPX Oper | EC | 3 | 4 | √ | - | - | - | - | - | √ | √ |

Table (22): CPX – Short Reference

CPY

Operation: Compare Memory and Index Y (Y - M)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | CPY # Oper | C0 | 2 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | CPY Oper | C4 | 2 | 3 | √ | - | - | - | - | - | √ | √ |
| Absolute | CPY Oper | CC | 3 | 4 | √ | - | - | - | - | - | √ | √ |

Table (23): CPY – Short Reference

DEC

Operation: Decrement Memory by one ($M - 1 \rightarrow M$ or $A - 1 \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator <i>NEW</i> | DEC A | 3A | 1 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | DEC Oper | C6 | 2 | 5 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | DEC Oper, X | D6 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| Absolute | DEC Oper | CE | 3 | 6 | √ | - | - | - | - | - | √ | - |
| Absolute, X | DEC Oper, X | DE | 3 | 7 | √ | - | - | - | - | - | √ | - |

Table (24): DEC – Short Reference

DEX

Operation: Decrement index X by one ($X - 1 \rightarrow X$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | DEX | CA | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (25): DEX – Short Reference

Example Implied (assume X=\$01):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | CA | DEX | ; |
| \$9001 | ... | <i>next OP</i> | ; X is now \$00, N=0, Z=1 |

DEX 0000 0001 (\$01)
0000 0000 (\$00), N=0, Z=1

DEY

Operation: Decrement index Y by one ($Y - 1 \rightarrow Y$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | DEY | 88 | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (26): DEX – Short Reference

Example Implied (assume Y=\$00):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | 88 | DEY | ; |
| \$9001 | ... | <i>next OP</i> | ; Y is now \$FF, N=1, Z=0 |

DEY 0000 0000 (\$00)
1111 1111 (\$FF), N=1, Z=0

EOR

Operation: “Exclusive-Or” memory or immediate value with accumulator A ($A \nabla M \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | EOR # Oper | 49 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | EOR Oper | 45 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | EOR Oper, X | 55 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | EOR Oper | 4D | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, X | EOR Oper, X | 5D | 3 | 4* | √ | - | - | - | - | - | √ | - |
| Absolute, Y | EOR Oper, Y | 59 | 3 | 4* | √ | - | - | - | - | - | √ | - |
| (Indirect, X) | EOR (Oper, X) | 41 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| (Indirect), Y | EOR (Oper), Y | 51 | 2 | 5* | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | EOR (Oper) | 52 | 2 | 5 | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (27): EOR – Short Reference

Example Immediate (assumed A is \$23):

| Address | Bytes | Mnemonic | |
|---------|-------|-----------|---------------------------|
| \$9000 | 49 63 | EOR #\$63 | ; |
| \$9002 | ... | next OP | ; A is now \$40, N=0, Z=0 |

0110 0011 (\$63)
EOR 0010 0011 (\$23)
0100 0000 (\$40), N=0, Z=0

INC

Operation: Increment Memory by one ($M + 1 \rightarrow M$ or $A + 1 \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator <i>NEW</i> | INC A | 1A | 1 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | INC Oper | E6 | 2 | 5 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | INC Oper, X | F6 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| Absolute | INC Oper | EE | 3 | 6 | √ | - | - | - | - | - | √ | - |
| Absolute, X | INC Oper, X | FE | 3 | 7 | √ | - | - | - | - | - | √ | - |

Table (28): INC – Short Reference

INX

Operation: Increment index X by one ($X + 1 \rightarrow X$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | INX | E8 | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (29): INX – Short Reference

Example Implied (assume $X = \$0C$):

| Address | Bytes | Mnemonic | |
|---------|-------|----------|---------------------------|
| \$9000 | E8 | INX | ; |
| \$9001 | ... | next OP | ; X is now \$0D, N=0, Z=0 |

INX 0000 1100 (\$0C)
0000 1101 (\$0D), N=0, Z=0

INY

Operation: Increment index Y by one ($Y + 1 \rightarrow Y$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | INY | C8 | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (30): INY – Short Reference

Example Implied (assume $Y = \$FF$):

| Address | Bytes | Mnemonic |
|---------|-------|----------------|
| \$9000 | C8 | INY |
| \$9001 | ... | <i>next OP</i> |

INY 1111 1111 (\$FF)
0000 0000 (\$00), N=0, Z=1

JMP

Operation: Jump to new location ($(PC + 1) \rightarrow PCL$, $(PC + 2) \rightarrow PCH$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Absolute | JMP Oper | 4C | 3 | 3 | - | - | - | - | - | - | - | - |
| (Absolute, X) <i>NEW</i> | JMP (Oper, X) | 7C | 3 | 6 | - | - | - | - | - | - | - | - |
| Indirect | JMP (Oper) | 6C | 3 | 6 | - | - | - | - | - | - | - | - |

Table (31): JMP – Short Reference

In mode “Indirect” JMP fetches the high-order byte of the effective address from the first byte of the next page, if the indirect address location spans a page boundary. The 6502 fetches it from the first byte of the current page.

R65C02: JMP (\$02FF) gets ADL from \$02FF, ADH from \$0300 (6 cycles)

6502: JMP (\$02FF) gets ADL from \$02FF, ADH from \$0200 (5 cycles)

JSR

Operation: Jump to subroutine saving return address ($PC + 2 \downarrow$, $(PC + 1) \rightarrow PCH$, $(PC + 2) \rightarrow PCL$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Absolute | JSR Oper | 20 | 3 | 6 | - | - | - | - | - | - | - | - |

Table (32): JSR – Short Reference

Example:

| Address | Bytes | Mnemonic |
|---------|----------|--------------|
| \$9000 | 20 72 F0 | JSR \$F072 ; |
| F072 | ... | next OP |

LDA

Operation: Load accumulator A with memory or immediate value ($M \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | LDA # Oper | A9 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | LDA Oper | A5 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | LDA Oper, X | B5 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | LDA Oper | AD | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, X | LDA Oper, X | BD | 3 | 4* | √ | - | - | - | - | - | √ | - |
| Absolute, Y | LDA Oper, Y | B9 | 3 | 4* | √ | - | - | - | - | - | √ | - |
| (Indirect, X) | LDA (Oper, X) | A1 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| (Indirect), Y | LDA (Oper), Y | B1 | 2 | 5* | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | LDA (Oper) | B2 | 2 | 5 | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (33): LDA – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic |
|---------|-------|-----------------------------------|
| \$9000 | A9 00 | LDA #\$00 ; |
| \$9002 | ... | next OP ; A is now \$00, N=0, Z=1 |

LDX

Operation: Load index X with memory or immediate value ($M \rightarrow X$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | LDX # Oper | A2 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | LDX Oper | A6 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, Y | LDX Oper, Y | B6 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | LDX Oper | AE | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, Y | LDX Oper, Y | BE | 3 | 4* | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (34): LDX – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic |
|---------|-------|-----------------------------------|
| \$9000 | A2 8F | LDX #\$8F ; |
| \$9002 | ... | next OP ; X is now \$8F, N=1, Z=0 |

LDY

Operation: Load index Y with memory or immediate value ($M \rightarrow Y$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | LDY # Oper | A0 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | LDY Oper | A4 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | LDY Oper, X | B4 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | LDY Oper | AC | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, X | LDY Oper, X | BC | 3 | 4* | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (35): LDY – Short Reference

Example Immediate:

| Address | Bytes | Mnemonic |
|---------|-------|-----------------------------------|
| \$9000 | A0 02 | LDY #\$02 ; |
| \$9002 | ... | next OP ; Y is now \$02, N=0, Z=0 |

LSR

Operation: Shift Right One Bit (Memory or Accumulator) ($0 \rightarrow \boxed{7} \boxed{6} \boxed{5} \boxed{4} \boxed{3} \boxed{2} \boxed{1} \boxed{0} \rightarrow C$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator | LSR A | 4A | 1 | 2 | 0 | - | - | - | - | - | √ | √ |
| Zero Page | LSR Oper | 46 | 2 | 5 | 0 | - | - | - | - | - | √ | √ |
| Zero Page, X | LSR Oper, X | 56 | 2 | 6 | 0 | - | - | - | - | - | √ | √ |
| Absolute | LSR Oper | 4E | 3 | 6 | 0 | - | - | - | - | - | √ | √ |
| Absolute, X | LSR Oper, X | 5E | 3 | 7 | 0 | - | - | - | - | - | √ | √ |

Table (36): LSR – Short Reference

NOP

Operation: No Operation

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|--|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | NOP | x3 xB | 1 | 1 | - | - | - | - | - | - | - | - |
| Implied | NOP | EA | 1 | 2 | - | - | - | - | - | - | - | - |
| Implied | NOP | 02 22 42 62 82 C2 E2 | 1(+1) | 2 | - | - | - | - | - | - | - | - |
| Implied | NOP | 44 | 1(+1) | 3 | - | - | - | - | - | - | - | - |
| Implied | NOP | 54 D4 F4 | 1(+1) | 4 | - | - | - | - | - | - | - | - |
| Implied | NOP | DC FC | 1(+2) | 4 | - | - | - | - | - | - | - | - |
| Implied | NOP | 5C | 1(+2) | 8 | - | - | - | - | - | - | - | - |

Table (37): NOP – Short Reference (“+1” or “+2” add 1 or 2 to the PC for fetching the next instruction)

ORA

Operation: “Or” memory or immediate value with accumulator A ($A \vee M \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | OR # Oper | 09 | 2 | 2 | √ | - | - | - | - | - | √ | - |
| Zero Page | OR Oper | 05 | 2 | 3 | √ | - | - | - | - | - | √ | - |
| Zero Page, X | OR Oper, X | 15 | 2 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute | OR Oper | 0D | 3 | 4 | √ | - | - | - | - | - | √ | - |
| Absolute, X | OR Oper, X | 1D | 3 | 4* | √ | - | - | - | - | - | √ | - |
| Absolute, Y | OR Oper, Y | 19 | 3 | 4* | √ | - | - | - | - | - | √ | - |
| (Indirect, X) | OR (Oper, X) | 01 | 2 | 6 | √ | - | - | - | - | - | √ | - |
| (Indirect), Y | OR (Oper), Y | 11 | 2 | 5* | √ | - | - | - | - | - | √ | - |
| (Indirect) <i>NEW</i> | OR (Oper) | 12 | 2 | 5 | √ | - | - | - | - | - | √ | - |

* => Add 1 if page boundary is crossed

Table (38): OR – Short Reference

Example Immediate (assumed A is \$8A):

| Address | Bytes | Mnemonic | |
|---------|-------|----------|---------------------------|
| \$9000 | 09 11 | OR #\$11 | ; |
| \$9002 | ... | next OP | ; A is now \$9B, N=1, Z=0 |

1000 1010 (\$8A)
 OR 0001 0001 (\$11)
1001 1011 (\$9B), N=1, Z=0

PHA

Operation: Push accumulator on stack ($A \downarrow$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | PHA | 48 | 1 | 3 | - | - | - | - | - | - | - | - |

Table (39): PHA – Short Reference

PHP

Operation: Push processor status on stack (P ↓ NV11DIZC, E & B are written as 1 to stack)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | PHP | 08 | 1 | 3 | - | - | - | - | - | - | - | - |

Table (40): PHP – Short Reference

PHX

Operation: Push index register X on stack (X ↓)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PHX | DA | 1 | 3 | - | - | - | - | - | - | - | - |

Table (41): PHX – Short Reference

PHY

Operation: Push index register Y on stack (Y ↓)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PHY | 5A | 1 | 3 | - | - | - | - | - | - | - | - |

Table (42): PHY – Short Reference

PLA

Operation: Pull accumulator from stack (A ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | PLA | 68 | 1 | 4 | √ | - | - | - | - | - | √ | - |

Table (43): PLA – Short Reference

PLP

Operation: Pull processor status from stack (P ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|------------|---|---|---|---|---|---|---|
| Implied | PLP | 28 | 1 | 4 | From Stack | | | | | | | |

Table (44): PLP – Short Reference

PLX

Operation: Pull index register X from stack (X ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PLX | FA | 1 | 4 | √ | - | - | - | - | - | √ | - |

Table (45): PLX – Short Reference

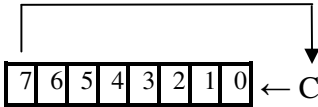
PLY

Operation: Pull index register Y from stack (Y ↑)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|--------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied <i>NEW</i> | PLY | 7A | 1 | 4 | √ | - | - | - | - | - | √ | - |

Table (46): PLY – Short Reference

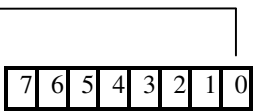
ROL

Operation: Rotate one bit left (memory or accumulator) ()

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator | ROL A | 2A | 1 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | ROL Oper | 26 | 2 | 5 | √ | - | - | - | - | - | √ | √ |
| Zero Page, X | ROL Oper, X | 36 | 2 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute | ROL Oper | 2E | 3 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute, X | ROL Oper, X | 3E | 3 | 7 | √ | - | - | - | - | - | √ | √ |

Table (47): ROL – Short Reference

ROR

Operation: Rotate one bit right (memory or accumulator) (C → )

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Accumulator | ROR A | 6A | 1 | 2 | √ | - | - | - | - | - | √ | √ |
| Zero Page | ROR Oper | 66 | 2 | 5 | √ | - | - | - | - | - | √ | √ |
| Zero Page, X | ROR Oper, X | 76 | 2 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute | ROR Oper | 6E | 3 | 6 | √ | - | - | - | - | - | √ | √ |
| Absolute, X | ROR Oper, X | 7E | 3 | 7 | √ | - | - | - | - | - | √ | √ |

Table (48): ROR – Short Reference

RMB

Operation: Reset memory bit x . Where $x = 0-7$ MSB of op code ($0 \rightarrow Mx$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | RMB(0-7) Oper | 07-77 | 2 | 5 | - | - | - | - | - | - | - | - |

Table (49): RMB – Short Reference

Example Zero Page:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|----------------------|
| \$9000 | 37 14 | RMB3 \$14 | ; \$0014 = \$C9 |
| \$9002 | ... | <i>next OP</i> | ; \$0014 is now \$C1 |

RTI

Operation: Return from interrupt ($P \uparrow$, $PC \uparrow$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|------------|---|---|---|---|---|---|---|
| Implied | RTI | 40 | 1 | 6 | From Stack | | | | | | | |

Table (50): RTI – Short Reference

RTS

Operation: Return from subroutine ($PC \uparrow$, $PC + 1 \rightarrow PC$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | RTS | 60 | 1 | 6 | - | - | - | - | - | - | - | - |

Table (51): RTS – Short Reference

SBC

Operation: Subtract memory or immediate value from accumulator with borrow ($A - M - \overline{C} \rightarrow A, C$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Immediate | SBC # Oper | E9 | 2 | 2+ | √ | √ | - | - | - | - | √ | √ |
| Zero Page | SBC Oper | E5 | 2 | 3+ | √ | √ | - | - | - | - | √ | √ |
| Zero Page, X | SBC Oper, X | F5 | 2 | 4+ | √ | √ | - | - | - | - | √ | √ |
| Absolute | SBC Oper | ED | 3 | 4+ | √ | √ | - | - | - | - | √ | √ |
| Absolute, X | SBC Oper, X | FD | 3 | 4+* | √ | √ | - | - | - | - | √ | √ |
| Absolute, Y | SBC Oper, Y | F9 | 3 | 4+* | √ | √ | - | - | - | - | √ | √ |
| (Indirect, X) | SBC (Oper, X) | E1 | 2 | 6+ | √ | √ | - | - | - | - | √ | √ |
| (Indirect), Y | SBC (Oper), Y | F1 | 2 | 5+* | √ | √ | - | - | - | - | √ | √ |
| (Indirect) <i>NEW</i> | SBC (Oper) | F2 | 2 | 5+ | √ | √ | - | - | - | - | √ | √ |

* => Add 1 if page boundary is crossed

+ => Add 1 if mode is "DECIMAL" (D='1')

Table (52): SBC – Short Reference

SEC

Operation: Set Carry flag ($1 \rightarrow C$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | SEC | 38 | 1 | 2 | - | - | - | - | - | - | - | 1 |

Table (53): SEC – Short Reference

Example:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--------------|
| \$9000 | 38 | SEC | ; |
| \$9001 | ... | <i>next OP</i> | ; C is now 1 |

SED

Operation: Set Decimal flag (1 → D)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | SED | F8 | 1 | 2 | - | - | - | - | 1 | - | - | - |

Table (54): CLD – Short Reference

Example:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--------------|
| \$9000 | F8 | SED | ; |
| \$9001 | ... | <i>next OP</i> | ; D is now 1 |

SEI

Operation: Set Interrupt Disable flag (1 → I)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | SEI | 78 | 1 | 2 | - | - | - | - | - | 1 | - | - |

Table (55): SEI – Short Reference

Example:

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|--------------|
| \$9000 | 78 | SEI | ; |
| \$9001 | ... | <i>next OP</i> | ; I is now 1 |

SMB

Operation: Set memory bit x. Where x = 0-7 (MSB – 8) of op code (1 → Mx)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | SMB(0-7) Oper | 87-F7 | 2 | 5 | - | - | - | - | - | - | - | - |

Table (56): SMB – Short Reference

Example Zero Page:

| Address | Bytes | Mnemonic |
|---------|-------|-------------------------------------|
| \$9000 | F7 14 | SMB \$F0 ; \$00F0 = \$43 |
| \$9002 | ... | <i>next OP</i> ; \$00F0 is now \$C3 |

STA

Operation: Store accumulator in memory (A → M)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page | STA Oper | 85 | 2 | 3 | - | - | - | - | - | - | - | - |
| Zero Page, X | STA Oper, X | 95 | 2 | 4 | - | - | - | - | - | - | - | - |
| Absolute | STA Oper | 8D | 3 | 4 | - | - | - | - | - | - | - | - |
| Absolute, X | STA Oper, X | 9D | 3 | 5 | - | - | - | - | - | - | - | - |
| Absolute, Y | STA Oper, Y | 99 | 3 | 5 | - | - | - | - | - | - | - | - |
| (Indirect, X) | STA (Oper, X) | 81 | 2 | 6 | - | - | - | - | - | - | - | - |
| (Indirect), Y | STA (Oper), Y | 91 | 2 | 6 | - | - | - | - | - | - | - | - |
| (Indirect) <i>NEW</i> | STA (Oper) | 92 | 2 | 5 | - | - | - | - | - | - | - | - |

Table (57): STA – Short Reference

STX

Operation: Store index X in memory ($X \rightarrow M$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page | STX Oper | 86 | 2 | 3 | - | - | - | - | - | - | - | - |
| Zero Page, Y | STX Oper, Y | 96 | 2 | 4 | - | - | - | - | - | - | - | - |
| Absolute | STX Oper | 8E | 3 | 4 | - | - | - | - | - | - | - | - |

Table (58): STX – Short Reference

STY

Operation: Store index Y in memory ($Y \rightarrow M$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page | STY Oper | 84 | 2 | 3 | - | - | - | - | - | - | - | - |
| Zero Page, X | STY Oper, X | 94 | 2 | 4 | - | - | - | - | - | - | - | - |
| Absolute | STY Oper | 8C | 3 | 4 | - | - | - | - | - | - | - | - |

Table (59): STY – Short Reference

STZ

Operation: Store value # $\$00$ in memory ($\#\$00 \rightarrow M$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-------------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | STZ Oper | 64 | 2 | 3 | - | - | - | - | - | - | - | - |
| Zero Page, X <i>NEW</i> | STZ Oper, X | 74 | 2 | 4 | - | - | - | - | - | - | - | - |
| Absolute <i>NEW</i> | STZ Oper | 9C | 3 | 4 | - | - | - | - | - | - | - | - |
| Absolute, X <i>NEW</i> | STZ Oper, X | 9E | 3 | 5 | - | - | - | - | - | - | - | - |

Table (60): STZ – Short Reference

TAX

Operation: Transfer accumulator A to index X ($A \rightarrow X$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TAX | AA | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (61): TAX – Short Reference

Example Implied (assume A=\$5D):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | AA | TAX | ; |
| \$9001 | ... | <i>next OP</i> | ; X is now \$5D, N=0, Z=0 |

TAY

Operation: Transfer accumulator A to index Y ($A \rightarrow Y$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TAY | A8 | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (62): TAY – Short Reference

Example Implied (assume A=\$89):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | A8 | TAY | ; |
| \$9001 | ... | <i>next OP</i> | ; Y is now \$89, N=1, Z=0 |

TRB

Operation: Test memory with accumulator A and reset memory bits. If the result of $A \cap M = 0$ then $Z = 1$, otherwise $Z = 0$ ($\bar{A} \cap M \rightarrow M$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | TRB Oper | 14 | 2 | 5 | - | - | - | - | - | - | √ | - |
| Absolute <i>NEW</i> | TRB Oper | 1C | 3 | 6 | - | - | - | - | - | - | √ | - |

Table (63): TRB – Short Reference

Example Zero Page (assume A=\$03):

| Address | Bytes | Mnemonic |
|---------|-------|--|
| \$9000 | 14 D2 | TRB \$D2 ; \$00D2 = \$61 |
| \$9002 | ... | <i>next OP</i> ; \$00D2 is now \$60, Z=0 |

```

0000 0011 (A=$03)
NOT  1111 1100
AND  0110 0001 (M=$61)
0110 0000 ($60) New value for memory

0000 0011 (A=$03)
AND  0110 0001 (M=$61)
0000 0001 Z=0 New value for flag Z

```

TSB

Operation: Test memory with accumulator A and set memory bits. If the result of $A \cap M = 0$ then $Z = 1$, otherwise $Z = 0$ ($A \cup M \rightarrow M$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|----------------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Zero Page <i>NEW</i> | TSB Oper | 04 | 2 | 5 | - | - | - | - | - | - | √ | - |
| Absolute <i>NEW</i> | TSB Oper | 0C | 3 | 6 | - | - | - | - | - | - | √ | - |

Table (64): TSB – Short Reference

Example Zero Page (assume A=\$0C):

| Address | Bytes | Mnemonic |
|---------|-------|--|
| \$9000 | 04 D2 | TSB \$D2 ; \$00D2 = \$11 |
| \$9002 | ... | <i>next OP</i> ; \$00D2 is now \$1D, Z=1 |

0000 1100 (A=\$0C)
OR 0001 0001 (M=\$11)
0001 1101 (\$1D) New value for memory

0000 1100 (A=\$0C)
AND 0001 0001 (M=\$11)
0000 0000 Z=1 New value for flag Z

TSX

Operation: Transfer stack pointer SP to index X ($SP \rightarrow X$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TSX | BA | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (65): TSX – Short Reference

Example Implied (assume SP=\$F2):

| Address | Bytes | Mnemonic |
|---------|-------|--|
| \$9000 | BA | TSX |
| \$9001 | ... | <i>next OP</i> ; X is now \$F2, N=1, Z=0 |

TXA

Operation: Transfer index X to accumulator A ($X \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TXA | 8A | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (66): TXA – Short Reference

Example Implied (assume X=\$00):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | 8A | TXA | |
| \$9001 | ... | <i>next OP</i> | ; A is now \$00, N=0, Z=1 |

TXS

Operation: Transfer index X to stack pointer SP ($X \rightarrow SP$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TXS | 9A | 1 | 2 | - | - | - | - | - | - | - | - |

Table (67): TXS – Short Reference

Example Implied (assume X=\$E0):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---|
| \$9000 | 9A | TXS | |
| \$9001 | ... | <i>next OP</i> | ; SP is now \$E0 (all flags are uneffected) |

TYA

Operation: Transfer index Y to accumulator A ($Y \rightarrow A$)

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles | N | V | E | B | D | I | Z | C |
|-----------------|------------------------|---------|-----------|------------|---|---|---|---|---|---|---|---|
| Implied | TYA | 98 | 1 | 2 | √ | - | - | - | - | - | √ | - |

Table (68): TYA – Short Reference

Example Implied (assume Y=\$14):

| Address | Bytes | Mnemonic | |
|---------|-------|----------------|---------------------------|
| \$9000 | 98 | TYA | ; |
| \$9001 | ... | <i>next OP</i> | ; A is now \$14, N=0, Z=0 |

4

Registers

List of Registers

| Name | Address | Width | Access | Description |
|------|---------|-------|-------------|-----------------------|
| A | - | 8 | Instruction | Accumulator |
| X | - | 8 | Instruction | X Register |
| Y | - | 8 | Instruction | Y Register |
| S | - | 8 | Instruction | Stack |
| PSW | - | 8 | Instruction | Processor Status Word |
| PC | - | 16 | Instruction | Program Counter |

Table 1: List of registers

PSW – Description

| Bit # | Access | Description |
|-------|--------|---|
| 7 | | N-Flag, negative result (1 -> negative) |
| 6 | | V-Flag, overflow (1 -> overflow) |
| 5 | | <i>RESERVED (always read as '1')</i> |
| 4 | | B-Flag, BRK instruction |
| 3 | | D-Flag, decimal mode (1 -> decimal) |
| 2 | | I-Flag, interrupt disable (1 -> disabled) |
| 1 | | Z-Flag, zero result (1 -> zero) |
| 0 | | C-Flag, carry |

Table 2: PSW Description

Reset Value: 24h

5

Clocks

| Name | Source | Rates (MHz) | | | Remarks | Description |
|-----------|--------|-------------|---------|-----|---|--------------|
| | | Max | Typ | Min | | |
| clk_clk_i | - | | 110-180 | 0 | Achievable rates depend on synthesis and place&route. | Master Clock |

Table 3: List of clocks

The following table gives you an overview of standard clocking rates for some different families and vendors (only standalone core – no system):

| Vendor | Device | Rates (MHz) | | | Logic Utilisation | Remarks |
|--------|--------------------------------|-------------|-----|-----|-------------------|--------------------|
| | | Max | Typ | Min | | |
| Intel | Stratix II: EP2S60F672C5ES | | 146 | 0 | 1161 ALUTs | “SLOW” model |
| Intel | Cyclone III: EP3C5F256C6 | | 125 | 0 | 1593 Elements | “SLOW” model |
| Intel | MAX V: 5M2210ZF256C4 | | 62 | 0 | 1542 Elements | “SLOW” model |
| Intel | MAX 10: 10M16DCF256C7G | | 113 | 0 | 1553 Elements | “SLOW” model |
| Intel | Stratix V: 5SGXMA3K2F35C3 | | 242 | 0 | 692 ALMs | “SLOW” model |
| Xilinx | Artix-7: xc7a15tcp236-3 | | 181 | 0 | 1511 LUTs | “Default” strategy |
| Xilinx | Virtex-7: xc7v585tffg1157-3 | | 144 | 0 | 1349 LUTs | “Default” strategy |
| Xilinx | Kintex-7: xc7k70tfbg484-3 | | 150 | 0 | 1349 LUTs | “Default” strategy |

Table 4: List of clock rates

6

Interrupts

The priority of interrupts is as followed if all of three interrupts occur at the same time:

BRK – NMI – IRQ

In contrast to the chip version of the R65C02, the interrupts are not delay able by some instructions like “CLF”.

NMI

The interrupt input “nmi_n_i” is edge triggered at a falling edge ‘1’ -> ‘0’. It is not mask-able.

There is a need of one cycle ‘1’ following with one cycle ‘0’ to detect a NMI interrupt as “valid”. “nmi_n_i” can switch back to ‘1’ or held on ‘0’ within cycle 3.

After the internal IRQ sequence of 7 cycles has finished, the “valid” flag re-set and allow another incoming NMI to detect. The maximum frequency of falling edges on “nmi_n_i” must be carefully defined to avoid a stack overflow while a currently processing interrupt is interrupted by another one.

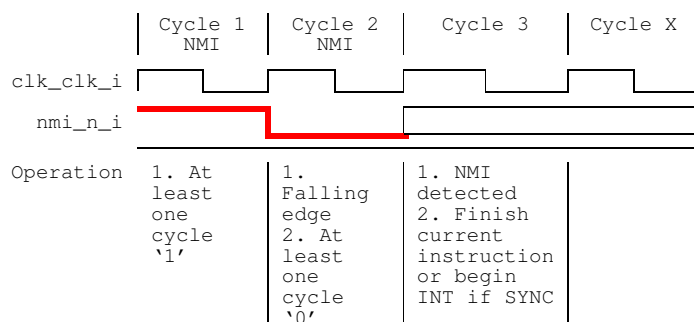


Figure (3): Interrupt NMI – Timing Diagram

IRQ

The interrupt input “irq_n_i” is level triggered at a level of ‘0’. It is mask-able by using the I-Flag (I=’1’ to disable IRQ).

There is a need of level ‘0’ at a rising edge of “clk_clk_i” and “SYNC=’1’” to detect an IRQ interrupt as “valid”. “irq_n_i” can switch back to ‘1’ or held on ‘0’ within cycle 3.

After the internal IRQ sequence of 7 cycles has finished, the I-Flag is set to ‘1’ automatically to disallow another incoming IRQ to detect. The starting ISR (Interrupt Service Routine) is responsible to manage and service the devices which were generating the interrupt and enable IRQ again (I-Flag=’0’).

Devices must switch back “irq_n_i” to ‘1’ after the interrupt service is finished.

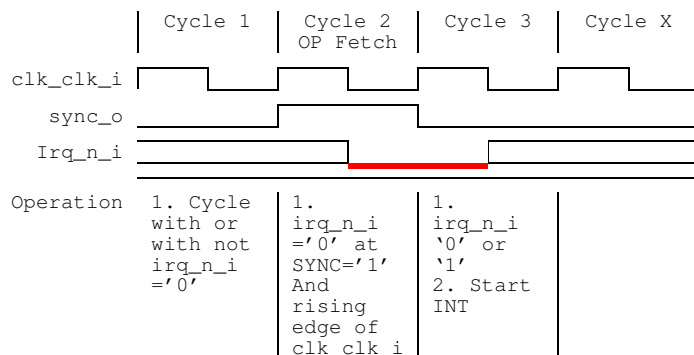


Figure (4): Interrupt IRQ – Timing Diagram

BRK

“BRK” is an instruction to force an interrupt by software. Internally it is handled like a real hardware interrupt and is mostly used by debug software.

“BRK” transfers the B-Flag = 1 onto the stack. The B-Flag itself leaves untouched and the I-Flag after “BRK” is 1.

Appendix A

ADC / SBC Decimal Mode

The R65C02 allows handling of Binary Coded Decimals (BCD) directly if the D-Flag is set to 1.

BCD defines a representation of decimal numbers 0-9 within an underlying hexadecimal number system.

For example:

99d => 63h

But in BCD the representation is

99d => 99h (BCD)

The R65C02 has the ability to convert the result value into BCD automatically if the nibbles exceed the decimal range of 0-9 with some limitations. Input values are not corrected in any way.

Some examples with ADC IMM (DECIMAL):

| ACCU | IMM | Carry in | Real World Result HEX | Real World Result BCD | R65C02 Result BCD | Low Nibble (LN) High Nibble (HN) |
|------|-----|----------|-----------------------|-----------------------|-------------------|--|
| 01h | 09h | 0 | 0Ah | 10h | 10h | LN > 9 (Ah), Carry to HN (=10h) |
| 00h | 09h | 1 | 0Ah | 10h | 10h | LN > 9 (Ah), Carry to HN (=10h) |
| 01h | 0Fh | 0 | 10h | 16h | 00h | LN < 9 (0h), NO Carry to HN (=00h) |
| 0Fh | 0Fh | 0 | 1Eh | 30h | 14h | LN > 9 (Eh), Carry to HN (=1h new), HN _{new} = HN _a + HN _{imm} + 1 |
| F0h | F0h | 0 | (1)E0h | (4)80h | 40h | LN < 9 (0h), NO Carry to HN (=Eh new), HN _{new} = HN _a + HN _{imm} + 0 |

| | | | | | | |
|-----|-----|---|--------|--------|-----|---|
| FFh | FFh | 0 | (1)FEh | (5)10h | 54h | LN > 9 (Eh), Carry to HN (=Fh new), HN _{new} = HN _a + HN _{imm} + 1 |
| FFh | FFh | 1 | (1)FFh | (5)11h | 55h | LN > 9 (Fh), Carry to HN (=Fh new), HN _{new} = HN _a + HN _{imm} + 1 |

Table 5: List of registers

In general (NOT valid for 65x02), there are four levels of decoding and corrections for hexadecimal values to convert to BCD:

| Level | Result pure HEX | | Corrected Result | System | Tasks |
|-------|-----------------|----|------------------|--------|---|
| L1 | 04h | => | 04h | (BCD) | no correction needed |
| L2 | 0Ch | => | 12h | (BCD) | correction of low nibble, add carry to the high nibble |
| L3 | 12h | => | 18h | (BCD) | correction of low nibble from high nibble (backward correction) if carry occur from low to high nibble. |
| L4 | 1Bh | => | 27h | (BCD) | correction of low nibble from high nibble , add carry to the high nibble (backward correction) |

Table 6: List of registers

Even the R65C02 correct the nibbles if a nibble value is greater than 9, the handling and generation of carry bits from one nibble to the other covers not all possible decoding levels L1 to L4. Incorrect results may occur if using invalid BCD numbers containing the hex parts A-F. The backward correction to the low nibble and new generation of the carry to correct the high nibble (L4) was not implemented into the R65C02 chip and the r65c02_tc IP core also.

Instruction Table

| LSB | | | | | | | | | | | | | | | |
|-----|---------------------------------------|--------------------------------------|---|--------------------|---------------------------------|--------------------|-------------------|-----------------------|-----------------------------------|-----------------------|---|-----------------------------------|-----------------------------------|-----------------------------------|----------------------------------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | BRK Implied 1 7 | ORA (IND,X) 2 6 | | TSB ZP 2 5 | ORA ZP 2 3 | ASL ZP 2 5 | RMB0 ZP 2 5 | PHP Implied 1 3 | ORA IMM 2 2 | ASL Accum 1 2 | | TSB ABS 3 6 | ORA ABS 3 4 | ASL ABS 3 6 | BBR0 ZP 3 5 ^{***} |
| 1 | BPL Relative 2 2 ^{***} | ORA (IND) 2 5 | | TRB ZP 2 5 | ORA ZP,X 2 4 | ASL ZP,X 2 6 | RMB1 ZP 2 5 | CLC Implied 1 3 | ORA ABS,Y 3 4 ^{**} | INC Accum 1 2 | | TRB ABS 3 6 | ORA ABS 3 4 ^{**} | ASL ABS,X 3 6 ^{**} | BBR1 ZP 3 5 ^{***} |
| 2 | JSR ABS 3 6 | AND (IND,X) 2 6 | | BIT ZP 2 3 | AND ZP 2 3 | ROL ZP 2 5 | RMB2 ZP 2 5 | PLP Implied 1 4 | AND IMM 2 2 | ROL Accum 1 2 | | BIT ABS 3 4 | AND ABS 3 4 | ROL ABS 3 6 | BBR2 ZP 3 5 ^{***} |
| 3 | BMI Relative 2 2 ^{***} | AND (IND,Y) 2 5 | | BIT ZP,X 2 4 | AND ZP,X 2 4 | ROL ZP,X 2 6 | RMB3 ZP 2 5 | SEC Implied 1 3 | AND ABS,Y 2 4 ^{**} | DEC Accum 1 2 | | BIT ABS,X 3 4 ^{**} | AND ABS,X 3 4 ^{**} | ROL ABS,X 3 6 ^{**} | BBR3 ZP 3 5 ^{***} |
| 4 | RTI Implied 1 6 | EOR (IND,X) 2 6 | | | EOR ZP 2 3 | LSR ZP 2 5 | RMB4 ZP 2 5 | PHA Implied 1 3 | EOR IMM 2 2 | LSR Accum 1 2 | | JMP ABS 3 3 | EOR ABS 3 4 | LSR ABS 3 6 | BBR4 ZP 3 5 ^{***} |
| 5 | BVC Relative 2 2 ^{***} | EOR (IND,Y) 2 5 | | | EOR ZP,X 2 4 | LSR ZP,X 2 6 | RMB5 ZP 2 5 | CLI Implied 1 3 | EOR ABS,Y 3 4 ^{**} | PHY Implied 1 3 | | | EOR ABS 3 4 ^{**} | LSR ABS,X 3 6 ^{**} | BBR5 ZP 3 5 ^{***} |
| 6 | RTS Implied 1 6 | ADC (IND,X) 2 6 | | STZ ZP 2 3 | ADC ZP 2 3 | ROR ZP 2 5 | RMB6 ZP 2 5 | PLA Implied 1 4 | ADC IMM 2 2 ⁺ | ROR Accum 1 2 | | JMP (IND) ABS 3 6 | ADC ABS 3 4 ⁺ | ROR ABS 3 6 | BBR6 ZP 3 5 ^{***} |
| 7 | BVS Relative 2 2 ^{***} | AND (IND,Y) 2 5 ^{***} | | STZ ZP,X 2 4 | ADC ZP,X 2 4 ⁺ | ROR ZP,X 2 6 | RMB7 ZP 2 5 | SEI Implied 1 3 | ADC ABS,Y 3 4 ^{**} | PLY Implied 1 4 | | JMP (IND,X) ABS 3 6 | ROR ABS,X 3 4 ⁺ | ROR ABS,X 3 6 ^{**} | BBR7 ZP 3 5 ^{***} |
| 8 | BRA Relative 2 3 [*] | STA (IND,X) 2 6 | | STY ZP 2 3 | STA ZP 2 3 | STX ZP 2 3 | SMB0 ZP 2 5 | DEY Implied 1 3 | BIT IMM 2 2 | TXA Implied 1 2 | | STY ABS 3 4 | STA ABS 3 4 | STX ABS 3 4 | BBR8 ZP 3 5 ^{***} |
| 9 | BCC Relative 2 2 ^{***} | STA (IND,Y) 2 6 ^{**} | | STY ZP,X 2 4 | STA ZP,X 2 4 | STX ZP,Y 2 4 | SMB1 ZP 2 5 | TYA Implied 1 3 | STA ABS,Y 3 5 ^{**} | TXS Implied 1 2 | | STZ ABS 3 4 | STA ABS,X 3 5 ^{**} | STZ ABS,X 3 5 ^{**} | BBR9 ZP 3 5 ^{***} |
| A | LDY IMM 2 2 | LDA (IND,X) 2 6 | | LDY ZP 2 3 | LDA ZP 2 3 | LDX ZP 2 3 | SMB2 ZP 2 5 | TAY Implied 1 3 | LDA IMM 2 2 | TAX Implied 1 2 | | LDY ABS 3 4 | LDA ABS 3 4 | LDX ABS 3 4 | BBR2 ZP 3 5 ^{***} |
| B | BCS Relative 2 2 ^{***} | LDA (IND,Y) 2 5 ^{**} | | LDY ZP,X 2 4 | LDA ZP,X 2 4 | LDX ZP,Y 2 4 | SMB3 ZP 2 5 | CLV Implied 1 3 | LDA ABS,Y 3 4 ^{**} | TSX Implied 1 2 | | LDY ABS,X 3 4 ^{**} | LDA ABS,X 3 4 ^{**} | LDX ABS,X 3 4 ^{**} | BBR3 ZP 3 5 ^{***} |
| C | CPY IMM 2 2 | CMP (IND,X) 2 6 | | CPY ZP 2 3 | CMP ZP 2 3 | DEC ZP 2 5 | SMB4 ZP 2 5 | INY Implied 1 3 | CMP IMM 2 2 | DEX Implied 1 2 | | CPY ABS 3 4 | CMP ABS 3 4 | DEC ABS 3 6 | BBR4 ZP 3 5 ^{***} |
| D | BNE Relative 2 2 ^{***} | CMP (IND,Y) 2 5 ^{**} | | | CMP ZP,X 2 4 | DEC ZP,X 2 6 | SMB5 ZP 2 5 | CLD Implied 1 3 | CMP ABS,Y 3 4 ^{**} | PHX Implied 1 3 | | | CMP ABS,X 3 4 ^{**} | DEC ABS,X 3 6 ^{**} | BBR5 ZP 3 5 ^{***} |
| E | CPX IMM 2 2 | SBC (IND,X) 2 6 ⁺ | | CPX ZP 2 3 | SBC ZP 2 3 ⁺ | INC ZP 2 5 | SMB6 ZP 2 5 | INX Implied 1 3 | SBC IMM 2 2 ⁺ | NOP Implied 1 2 | | CPX ABS 3 4 | SBC ABS 3 4 ⁺ | INC ABS 3 6 | BBR6 ZP 3 5 ^{***} |
| F | BEQ Relative 2 2 ^{***} | SBC (IND,Y) 2 5 ^{***} | | | SBC ZP,X 2 4 ⁺ | INC ZP,X 2 6 | SMB7 ZP 2 5 | SED Implied 1 3 | SBC ABS,Y 3 4 ^{**} | PLX Implied 1 4 | | | SBC ABS,X 3 4 ^{**} | INC ABS,X 3 6 ^{**} | BBR7 ZP 3 5 ^{***} |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E |

+ Add 1 to N if in decimal mode, * ADD 1 to N if page boundary is crossed

** Add 1 to N if branch occurs to same page or add 2 to N if branch occurs to different page

Index

- [1] Rockwell, Product Description R65C02, R65C102 and R65C112 – R65C00 Microprocessors
Document No. 29651N52, Rev. 6, June 1987, Order No. 2149
- [2] MOS Technology, MCS6500 Microcomputer Family, Hardware Manual
Publication No. 6500-10A, Rev. A, January 1976
- [3] MOS Technology, MCS6500 Microcomputer Family, Software Manual
Publication No. 6500-50A, January 1976
- [4] Western Design Center, Programming the 65816
2007