C74-6502 Datasheet

The C74-6502 is a cycle-accurate, pin-compatible implementation of the classic 6502 8-bit microprocessor. Built using strictly discrete components, it can run at clock-speeds of up to 20MHz and may be configured to operate as an 6502, a 6510, or 65C02 processor.

With the addition of an optional "K24 Card", the CPU acquires certain WDC 65816 capabilities, namely a 24-bit address bus (16MB memory space), additional addressing modes and dozens of new opcodes. The K24 Card also enables the instruction-set to be switched programmatically, allowing separate 6502 and 65C02 programs to run concurrently in independent 64K "partitions".

Architecturally, the C74-6502 is a microcode-based design where a 32-bit control-word is decoded "on-the-fly" prior to use by the Control Unit. A single-stage microcode pipeline pre-fetches micro-instructions from ROM, effectively eliminating the microcode fetch time from the critical path, and allowing the CPU to execute one microinstruction per clock-cycle at 20MHz.

The C74-6502 implements all NMOS 6502 and WDC 65C02 instructions, interrupts and functions, including Decimal Mode and “Undocumented Opcodes” (with some exceptions, as detailed below). The C74-6502 uses 7400 series ICs from the AC, LVC and CBT logic families for its circuitry, which includes a discrete-component ALU, an integrated 6510 I/O port and a built-in SPI interface.

The CPU can operate in place of an existing 6502 IC in a host-computer through the use of a custom 40-pin Socket-Adapter. Alternatively, it may be installed in new designs by way of standard pin headers.

See C74Project.wordpress.com for further information on the C74 Project.

Specifications

- Cycle-accurate, pin-compatible 6502, 6510 & 65C02 operation[2]
- Implements all 6502 and 65C02 instructions and functions, including Decimal Mode[2]
- Supports NMOS 6502 Undocumented Opcodes[2]
- Includes an on-board 6-bit, bidirectional I/O port which is compatible with the MOS 6510
- Custom 40-Pin Host Socket Adapter
- Compatible with TTL or CMOS Host Systems
- Auxiliary Power Supply Connections
- 0 to 20MHz Clock-Rate
- Operating Voltage: 5V[3]
- Current Draw: 130mA at 1MHz, 750mA at 20MHz[3]

K24 Card Optional Features: (untested)

- 24-bit address bus
- Additional 65816 instructions and addressing modes
- Software selectable instruction-sets
- Built-in SPI Interface and custom Opcode

Notes:

[1] The control word is 48 bits when the K24 card is installed

[2] There are specific limitations and caveats to compatibility with the target processors. These are described in detail below and summarized in Appendix J.

[3] Quoted figures reflect approximate measurements only

C74-6502

Physical Assembly

The C74-6502 is comprised of two main Eurocard PCBs, and a third optional card, as follows:

Card A - Registers
This card holds the processor's main registers, the 6510 port and clock management logic. Two separate headers allow for both 6502 and 6510 pinout connections, with the first also being compatible with WDC and Rockwell 65C02 pinouts. On-board jumpers are used to enable the /VP pin on the 6502 header and the 6510 port. Jumpers also select between host or auxiliary power supplies, and can optionally engage a Zero Delay Buffer when running at the highest clock-rates.

Card B - ALU & CU
This card holds the discrete-component ALU, the P status register, flag management logic, microcode sequencer, and interrupt handling logic. On-board jumpers on this card are used to select the microcode for the active instruction-set, which then also set other operating options accordingly.

Card C - K24 (Optional)
This card holds logic specific to implementing a 24-bit address bus. It is entirely optional, and the CPU will function correctly without it. When installed, an 8-bit Address Bus extension (ADX) is available as a separate header on the K24 Card (which requires custom connections to the host system for use). A special K24 configuration register (CFG) enables programmatic switching of the active instruction-set and associated operating options. The K24 Card also implements a simple SPI Mode 3 interface on a dedicated header. This interface may be accessed directly by the CPU, without the need of any additional circuitry or bit-banging software.

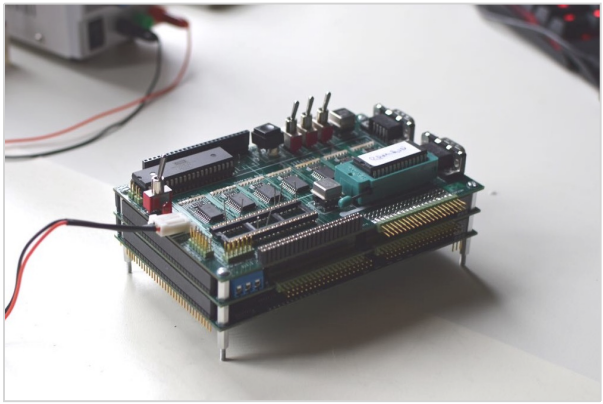


Fig 1. C74-6502 & SBC In Stacked Configuration

Microcode

The C74-6502 will operate with one of several available instructions sets. Each is implemented as separate microcode and is selectable via a pair of on-board jumpers on Card-B labeled ALT and CMOS as follows:

	Jumpers Closed	Microcode
0)	**None**	6502
1)	**CMOS**	65C02
2)	**ALT**	6502+NOPs
3)	**ALT & CMOS**	K24

The different microcode options are described in more detail below.

6502 Microcode
Implements all NMOS 6502 operations and functions, including, with minor exceptions, all bugs and other quirky behaviour. Decimal Mode operations reproduce the same values for all status flags as the original, including those that are invalid. The JMP (ind) page-crossing bug is replicated, as is the wrap-around behaviour of zero-page indexed addressing. Similarly, Read-Modify-Write instructions in abs,X addressing mode take seven cycles, whether or not a page is crossed. "Undocumented" opcodes are reproduced as well, except for ARR ($6B) in Decimal Mode. "Unstable" opcodes are given a fixed but sensible behaviour as detailed in Appendix F. Other compatibility notes are given on Appendix J.

65C02 Microcode
Implements all WDC 65C02 instructions, and is cycle-accurate, including reproducing single and multi-cycle cycle NOPs. Opcodes BBR, BBS, RMS, SMB, WAI and STP are also included. Read-Modify-Write (RMW) instructions reflect the fact that DEC and INC (abs,X) always take seven cycles on the original processor, but ASL, ASR, ROL and ROR (abs,X) all take only six cycles if no page boundary is crossed. The /ML signal is held low during the read, modify and write cycles of RMW instructions (see Appendix B for details on connecting the /ML pin to host systems). The microcode corrects the JMP (ind) bug of the NMOS 6502, and also executes this instruction in six cycles, rather than five. Decimal Mode operations will generate correct values for all flags (and take an extra cycle to do so). Note that Decimal mode SBC operations using invalid BCD inputs will generate results compatible with the NMOS 6502 only (see *Decimal Mode* below). Finally, interrupts will automatically clear the Decimal Flag, as the 65C02 does (unless configured otherwise through the K24 CFG register, see below).

6502+NOPs Microcode
This microcode is useful when trying to avoid the accidental execution of undocumented opcodes - for instance, to prevent a KIL operation from halting the CPU. It implements the NMOS 6502 instruction set also, but replaces undocumented opcodes with cycle-accurate NOPs (i.e. NOPs which take the same number of bytes and cycles as the undocumented opcodes they replace, but perform no function). It is otherwise cycle and function-accurate.

K24 Microcode
The K24 microcode implements additional WDC 65816 instructions using a 24-bit address space and 8-bit registers. This is behaviour analogous to the 65816's Emulation Mode (E flag = 1 and M and X flags are implied to be 1). All but MVN, MVP, XCE and COP opcodes are implemented in

the microcode. There a number of specific incompatibilities to the 65816 worth noting (full details are documented in Appendix H).

In addition, this microcode implements a couple of opcodes particular to the K24 card. Namely, WDM ($42) is replaced by a custom CFG opcode which sets the K24 CFG register; and COP ($02) is replaced by a custom SPI opcode to enable the CPU to communicate directly with an external SPI device (details of both these opcodes can be found below).

In order to facilitate two-way switching of the active instruction-set, the CFG opcode ($42) also appears in the 6502 and 65C02 microcode when the K24 Card is installed and the CFG.EN jumper is engaged. Leaving CFG.EN open will disable the CFG opcode and CFG register. In that instance, $42 reverts back to its original function, and the active microcode is then selected via the dedicated jumpers on Card B as usual.

The 65816 instruction-set may be used when the K24 Card is not present, but of course addressing will be restricted to 16 bits. Operations which read 65816 specific registers will read a zero in that case, and write operations to those registers will be NOPs.

CPU Functions

Decimal Mode

Decimal Mode operations differ in the three supported processors in two minor respects — the treatment of the status flags and invalid BCD inputs. The NMOS 6502 sets only the C-flag to a valid state after a Decimal operation, leaving the N, Z and V flags in an invalid state. The 65C02, on the other hand, sets all four flags to a valid state, but takes an extra cycle to do so. Meanwhile, the 65816 sets all flags correctly and does it all in one cycle.

The C74-6502 ALU can emulate the behaviour of all three processors in this respect, as controlled by two operating options: 2-Cycle-BCD and BCD-Flags-Valid. 2-Cycle-BCD adds a cycle to Decimal Mode instructions, while BCD-Flags-Valid will cause the ALU to produce correct values for all flags, as the 65C02 does.

These options are set automatically at RESET based on the active instruction-set, but may also be set manually through the CFG opcode (see CFG Opcode below). Note that the 65816 compatible configuration, i.e., enabling BCD-Flags-Valid while leaving 2-Cycle-BCD disabled, will add few nanoseconds of delay to the minimum CPU-cycle and will therefore reduce the maximum clock-rate. Since the 65816 implementation is not cycle-accurate in any case, there is no harm in enabling 2-Cycle-BCD for the 65816 instruction-set if operating at the highest clock-rates is a priority. This is what the hardware will do if configuring this option automatically.

Decimal Mode SBC operations using invalid BCD input values generate different results on the 65C02. Valid BCD values use strictly “0” through “9” as digits. Any value using the hex digits “A” through “F” is considered invalid, and generates undefined results when used in Decimal Mode. The NMOS 6502 and the 65816 produce matching results, but the 65C02 has minor differences for SBC operations. For example:

Operation	6502	65816	65C02
“0 - 11”	89	89	89
“0 - $B”	$9F	$9F	$**8F**

All three processors generate “89” for “0 - 11”. On the other hand, “0 - $B” generates “$8F” on the 65C02 and “$9F” otherwise.

The C74-6502 ALU generates results compatible the NMOS 6502 and 65816 behaviour in this respect, regardless of the microcode or Decimal Mode operating options in effect. This incompatibility is a potential source of errors when executing 65C02 software programs on the C74-6502.

Interrupt Handling

The C74-6502 always processes interrupts with the following priority: /RES then /NMI then /IRQ. As with the NMOS 6502, an NMI will preempt an IRQ if it occurs prior to the interrupt vector being fetched from memory. The IRQ will then be processed after the NMI completes. /RES going low inhibits all writes to memory but the processor otherwise continues normal execution. The rising-edge of /RES thereafter will trigger the Reset interrupt, which is then processed at the next SYNC cycle.

All interrupt signals are sampled on the falling-edge of PHI2 and take-effect on the next SYNC cycle following detection. This rule is strictly observed, even after branch instructions, which is NOT always the case on 6502 processors. Depending on the CPU variant and the conditions of the branch, a branch instruction may delay the detection of interrupts. (see http://forum.6502.org/viewtopic.php?f=4&t=1634 and http://forum.6502.org/viewtopic.php?f=4&t=4129&p=45773#p45703). Although strictly speaking incompatible, the behaviour of the C74-6502 is more consistent and yields lower interrupt latency.

By default, the interrupt service sequence will clear the D Flag when the 65C02 or K24 microcode is selected, but will leave it unchanged for NMOS 6502 operation. The "Interrupt-CLD" internal operating option controls this behaviour. This option is automatically set by the hardware on RESET based on the selected instruction-set, but may also be set manually through the CFG opcode (see CFG Opcode below).

Interrupt after BRK fetch: If an interrupt occurs after a BRK instruction has been fetched but before the interrupt vector is invoked, the NMOS 6502 will perform the interrupt and ignore the BRK. By contrast, the CMOS 65C02 will execute the BRK and process the interrupt thereafter (according to the Rockwell R65C02 datasheet). The C74-6502, on the other hand, combines these two behaviours so that an interrupt will preempt the BRK, but the BRK will then execute once the interrupt completes.

Unnecessary Reads

The 6502 interacts with memory on every cycle, even if it does not yet have a fully formed address. During indexed addressing, for example, the NMOS 6502 will perform a spurious read from a *partially formed* address if a page boundary is crossed, and discards the data. A similar problem manifests during absolute-indexed store operations, where the CPU always performs a read just prior to the write. Here, the address will be *fully formed* if no page crossing takes place, but the read is nevertheless spurious.

Although normally benign, such unnecessary reads can be destructive and trigger unwanted (or in rare cases, wanted) side-effects. A change was introduced in the 65C02 to re-use the address on the bus (a.k.a the Previous Bus Address), as a reasonably “safe” address, during read cycles when the CPU is busy incrementing the high-byte of a target address. Unfortunately, that still leaves the processor performing potentially destructive reads during absolute-indexed store operations when a page is not crossed.

The C74-6502 correctly replicates the behaviour of both processors with regards to unnecessary reads, except that absolute-indexed store operations always use the Previous Bus Address when the 65C02 instruction-set is active. See Appendix K for a complete description of the behaviour of the NMOS 6502 and 65C02 during such unnecessary reads, as well as what the C74-6502 does in each instance.

RMW Instructions

The 65C02 performs two reads and a single write during Read-Modify-Write instructions, vs. the NMOS 6502 which does one read and two writes. In both situations, the middle cycle is a dummy-cycle when the CPU is performing the internal "modify" portion of the instruction. Performing a write in the middle-cycle can have unintended (or in rare cases, intended) consequences. The C74-6502 matches the behaviour of each target processor during the Read-Modify-Write dummy-cycle (i.e., a write for the NMOS 6502 and a read for the 65C02).

Microcode Pipeline

A single-stage microcode pipeline enables the CPU to support clock speeds of up to 20 MHz. The basic function of the pipeline is to pre-fetch the next micro-instruction while the current one executes - in essence, overlapping the traditional fetch and execute cycles for microcode. This pre-fetching has the effect of ensuring a new micro-instruction is always available when needed, and the CPU can execute one micro-instruction every 50ns without waiting.

K24 Card Initialization

The K24 card hosts several dedicated registers that are specifically initialized. On Reset, the microcode zeroes the CFG register and both the PBR and DBR Bank registers. This defaults the CPU to NMOS 6502 compatible behaviour running in Bank 0 on startup.

K24 ADX Bus

The ADX bus on the K24 Card header implements the upper 8 bits of the optional 24-bit address bus. It requires custom

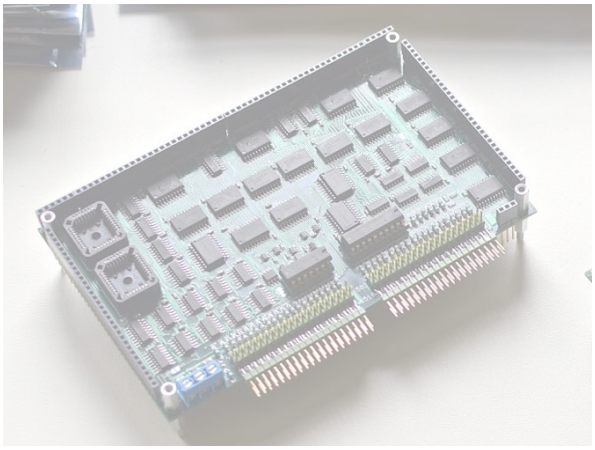


Fig. 2 - K24 Card & ADX Pinout Header (pic *)**

connections to the host system for use. See Appendix B for a detailed description of the K24 Pinout header.

When the K24 microcode is active, the value emitted on ADX will depend on the operation and addressing mode in effect as follows:

Operation	Addressing Mode	ADX[1]
Interrupt	Vector Fetch	$00
Stack Access	Implied	$00
Data Access	Zero Page	$00
Data Access	Standard Addressing	DBR
Data Access	Long Addressing	Operand
Program Access	Opcode Fetch, JMP (abs), JSR(abs,x)	PBR

[1]ADX is adjusted to reflect bank boundary crossings as appropriate.

When either the 6502 or 65C02 microcode is active, the value emitted on ADX is follows:

Operation	Addressing Mode	ADX[1]
Interrupt	Vector Fetch	PBR
Stack Access	Implied	DBR
Data Access	Zero Page	DBR
Data Access	Standard Addressing	DBR
Program Access	Opcode Fetch, JMP (abs), JSR(abs,x)	PBR

[1]ADX is NOT adjusted at bank boundaries.

If enabled, the 6510 port is mapped to bank 0 for the K24 microcode and to the current data bank (DBR) otherwise.

CFG Opcode

The CFG opcode ($42) appears in all instruction sets when the CFG register is enabled (which requires the K42 Card to be present and the CFG.EN jumper on that card to be closed). CFG replaces the KIL undocumented opcode on the NMOS 6502, a multi-cycle NOP on 65C02, and WDM on the 65816. These Opcodes will revert back to their original function if the CFG register is disabled. CFG is a 2-byte, 3-cycle opcode that will exchange the value of the A register with the CFG register. Flags are unchanged. The operand is ignored.

The format of the CFG Register is as follows:

Bit	Function
0..1	Microcode Select[1]
4	Enable 2-Cycle-BCD
5	Enable BCD-Flags-Valid
6	Enable Interrupt-CLD
7	AUX Control Signal

[1]Microcode 0 = 6502, 1 = 65C02, 2 = 6502+NOPs, and 3 = K24.

The Bit 7 AUX control signal is output on pin-5 of the K24 header and may used for any creative purpose.

If the CFG register is disabled, the active microcode is selected via jumpers on Card B, and other operating options are set by the hardware accordingly. In that instance, pin-5 on the K24 header will pull to GND.

Note that switching microcode using the CFG Opcode does not alter either of the 65816-style Program or Data Bank Registers resident in the K24 Card. This means that multiple 6502 and 65C02 programs can be made to run within specifically assigned 64K banks in memory transparently and without conflict. Sample code to illustrate a rudimentary pre-emptive multitasking scheme using NMI interrupts and CFG to switch between tasks can be found in Appendix I below.

SPI Opcode

The K24 Card incorporates a simple SPI

Mode 3 interface which enables the CPU to communicate directly with a wide variety of SPI enabled devices. A dedicated SPI Header is available on the K24 Card with 3 independent Slave Select lines (See Appendix B). The COP opcode ($02) in the 65816 instruction-set has been replaced with a custom SPI opcode. SPI will exchange 8 bits in the A accumulator with the selected SPI device over 8 consecutive clock-cycles, MSB first. SPI is a 2-byte, 10 cycle instruction. Flags are unchanged. The instruction's single-byte operand is formatted as follows:

Bit	Function
0	SS0, Slave Select 0
1	SS1, Slave Select 1
2	SS2, Slave Select 2
7	Continue[1]

[1]Continue=0 will turn off the Slave Select lines at the end of the instruction, while Continue=1 will leave them on so data interchange may continue with a following SPI instruction.

Pinout

The C74-6502 implements two distinct pinout headers on Card A (physically, each is composed of two 40-pin connectors, providing 40 pins for CPU-pin signals, and 40 pins for GND lines).

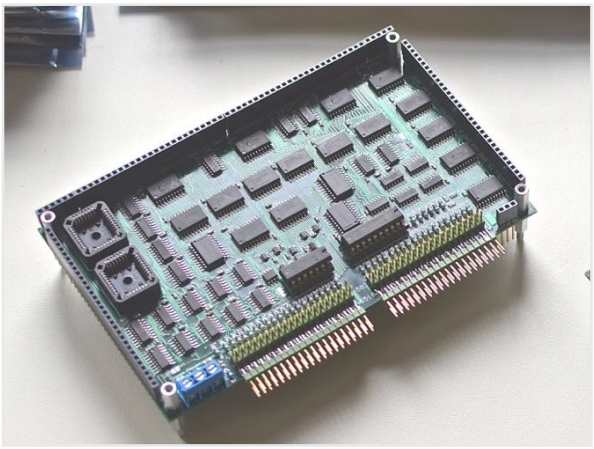


Fig. 3 - Card A - 6502 & 6510 Pinout Headers

The pinout header perpendicular to the board is compatible with the NMOS 6502 and the CMOS 65C02, while the horizontal one is compatible with a 6510. Certain signals differ very slightly in function to the originals but in all cases the circuitry is "well-behaved" so the CPU should be pin-compatible in all but rare circumstances. The K24 Card provides access to the ADX bus and useful control signals for 24-bit address decoding on a dedicated header. Details are provided in Appendix B.

Clock Signals

The NMOS 6502 expects a PHI0 clock input on pin-37, and generates PHI1 and PHI2 as output signals. By contrast, the WDC 65C02 uses different nomenclature and refers to the input clock as PHI2 and the output clocks as PHI1O and PHI2O. The WDC documentation indicates there is an "unspecified delay" between the PHI2 input and the output clocks, and recommends that the input clock signal be used as a system clock

The C74-6502 uses the NMOS 6502 nomenclature (referring to the input clock as PHI0), and allows the use of either the PHI0 input clock or the PHI2 output clock as the system clock. As with the 65C02, the clock may be stopped indefinitely at any time. There is a delay of approximately 20ns between PHI0 and PHI2 (typical tpd), which can be safely ignored in most circumstances. If the PHI0 input clock is used as a system clock, however, this internal clock latency may hinder the performance of the CPU at clock-rates in excess of 14MHz. In those circumstances, the on-board Zero Delay Buffer should be enabled through the "ZDB" jumper located on Card A.

When enabled, ZDB will eliminate any internal delay such that PHI2 will be exactly co-incident with PHI0. When ZDB is enabled, care must be taken to provide the CPU with a constant-frequency, CMOS-level clock signal of at least 5MHz. Note that the Zero Delay Buffer requires 1ms to stabilize internal clock signals. As a result, dynamically adjusting the frequency of the clock when ZDB is engaged will cause the CPU to fail.

6510 On-Board I/O

The C74-6502 implements a discrete-component 6510 compatible I/O port on addresses $0000 and $0001. Although only 6 bits are on the CPU pins, the port is 8 bits wide internally. The port can be enabled via an on-board jumper (SJ2) located on Card A. If the K24 card is present, the 6510 port will be mapped to Bank 0 if the K24 microcode is active, and to all banks otherwise. The port pins are available on the 6510 Pinout Header on Card A.

TTL Level Inputs

The C74-6502 may be configured with "T" variant ICs (i.e., 74ACT) on all input signals if TTL input levels (rather than CMOS) are expected on the CPU. Through-hole IC sockets on Card A are provided for this purpose. These are clearly marked on the PCB as "74'132" and "74'245". They should be fitted with 74AHCT132 and 74ACT245 ICs respectively to accept TTL level signals.

C74 Socket Adapter

The C74-6502 may be connected to a host computer using the C74 Socket Adapter and ribbon cable. The Adapter can replace either a 6502, 6510 or a 65C02 IC on the host system. The ribbon cable should be connected to either the 6502 or 6510 headers on Card A as appropriate.

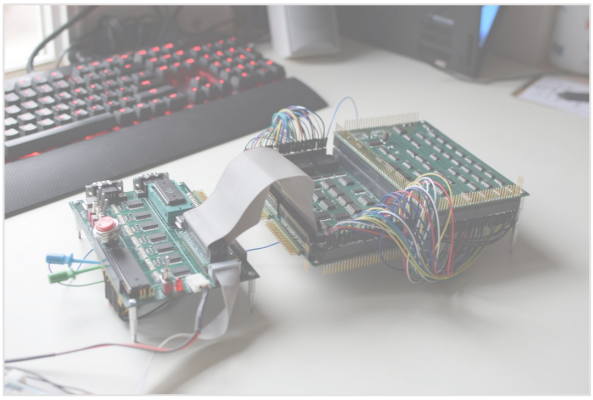


Fig. 4 C74-6502 Using 6502 Pinout Header (pic*)**

The Adapter has header connectors topside, where the ribbon cable connects, and machined socket-pins below, which will plug into a standard 40-pin IC socket.

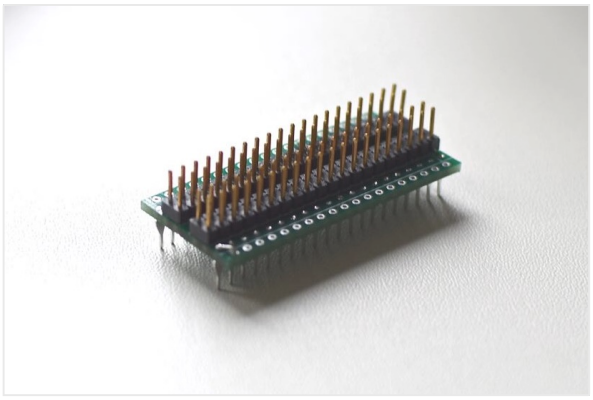


Fig. 5 - C74 Socket Adapter

Special care should be taken to orient the ribbon cable connectors according to the pin markings on Card A and on the adapter itself. GND and VDD lines on the adapter must be explicitly connected or bypassed to ground before use, as described on Appendix C.

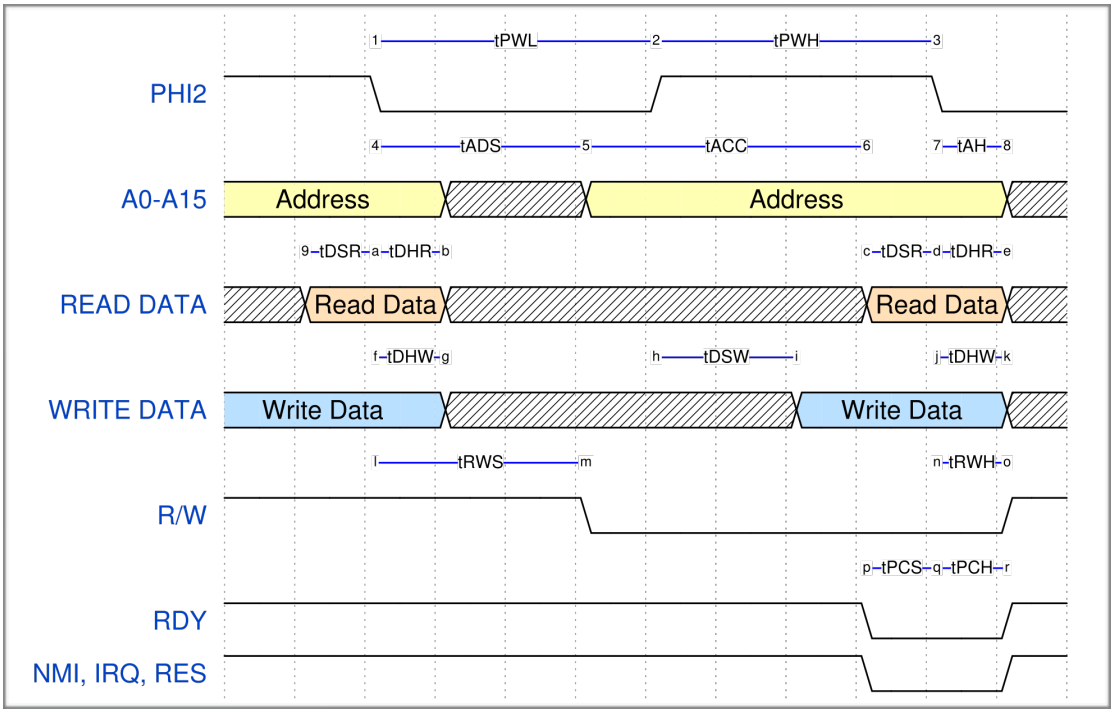
Power Supply

The C74-6502 may be powered from the host computer through the Socket Adapter's VDD pin. However, if the host is unable to supply sufficient current, the Auxiliary Power

terminal connectors on Card A can be used to provide supplemental power the CPU. The "Host Power" jumper on Card A should be left open to select Auxiliary Power, or closed to rely solely on the host. If external power is used, care should be taken in the design of the external supply circuit that ensures voltage levels are synchronized with the host. Details can be found in Appendix I.

CPU Timing Requirements

Fig. 6 - General Timing Diagram



Switching Characteristics (TA = 25°, V = 5V)

Symbol	Parameter	Typical[1]	Measured[1]	Unit
tACC	Max. Access Time (at 20MHz)	22	30	ns
tADS	Address Setup Time	23	15	ns
tAH	Address Hold Time	23	15	ns
tDHR	Read Data Hold Time	0		ns
tDHW	Write Data Hold Time	22	15	ns
tDSR	Read Data Setup Time	5		ns
tDSW	Write Data Setup Time	22	15	ns
tPCH	Processor Control Hold Time	0.5		ns
tPCS	Processor Control Setup Time	11		ns
tPWH	Minimum Clock Pulse Width High	25		ns
tPWL	Minimum Clock Pulse Width Low	25		ns
tRWH	R/W Signal Hold Time	24	20	ns
tRWS	R/W Signal Setup Time	24	20	ns

[1]**Note: “**Measured” times above reflect an average of measurements taken and are accurate within 2.5ns. “Typical” times are derived by calculating signal propagation delays using “typical” figures given on component manufacturer datasheets. Where only Min and Max values are available for such figures , the mid-point between those two is used as a proxy.

Appendix A
Configuration Summary

The table below illustrates how the C74-6502 may be configured to emulate the behaviour of the various target CPUs as closely as possible. There are, nevertheless, limitations to the compatibility achieved in each case, and these are detailed in Appendix J.

Jumper/Parameter	NMOS 6502	NMOS 6510	WDC 65C02	WDC 65816[1]
Pin Header	6502	6510	6502	6502 & K24
Card A: 6510 Port	Disable	Enable	Disable	Disable
Card A: /VP[2]	Disable	Disable	Enable	Enable
Card A: Host Power	Enable if powering the CPU solely from the host computer			
Card A: *DP	Enable	Enable	Disable	Disable
Card A: 74'245 Socket	Use 74ACT245 for TTL levels, otherwise 74AC245			
Card A: 74'132 Socket	Use 74AHCT132 for TTL levels, otherwise 74AC132			
Card A: ZDB	Enable for >14MHz if PHI0 is used for peripherals			
Card B: ROM.SEL	None or ALT	None or ALT	CMOS	CMOS & ALT
Card C: K24 Header	ADX & 816 signals available if K24 Card is installed			
Card C: CFG.EN	Enable for dynamically selecting the Instruction-Set and options			
CFG[1..0]: Microcode	6502 or 6502+NOPs	6502 or 6502+NOPs	65C02	K24
CFG[4]: 2-Cycle-BCD	Disable	Disable	Enable	Enable
CFG[5]: BCD-Flags-Valid	Disable	Disable	Enable	Enable
CFG[6]: Interrupt-CLD	Disable	Disable	Enable	Enable
CFG[7]: AUX	AUX Control Signal Output on Pin-5 of the K24 header			
Socket Adapter VDD	Pin 8	Pin 6	Pin 8	N/A
Socket Adapter GND[2]	Pin 1 & 21	Pin 21	Pin 21	N/A
Socket Adapter PIN 5[3]	N. C.	N. C.	Connected	N/A

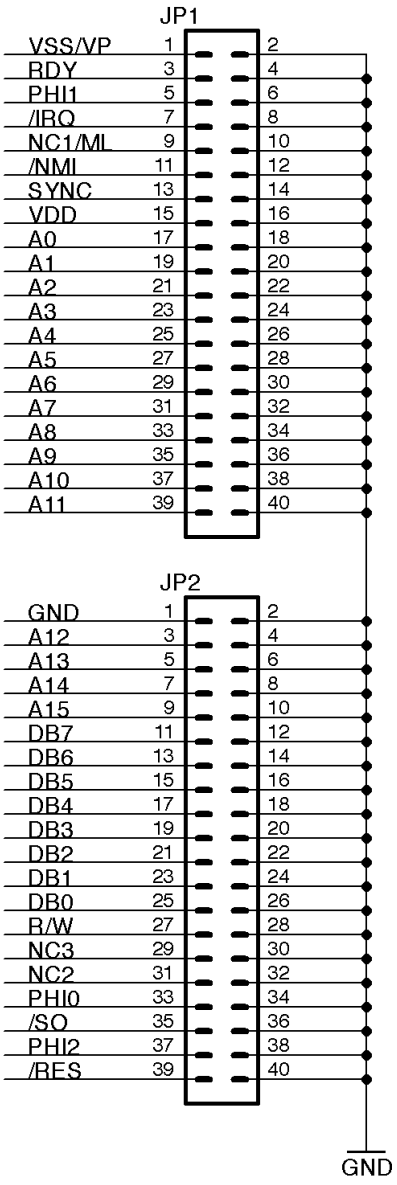
[1]**Note**: The C74-6502 closely emulates the behaviour of the WDC 65816's Emulation Mode only (i.e., E flag = 1, M and X flags = 0). See Appendix H for details and limitations

[2]**Note**: When connecting the C74-6502 to a host-system which expects a Rockwell 65C02, Pin 1 should be connected to ground at the Socket Adapter and the /VP jumper on Card A should be disabled. See Appendix B for further details.

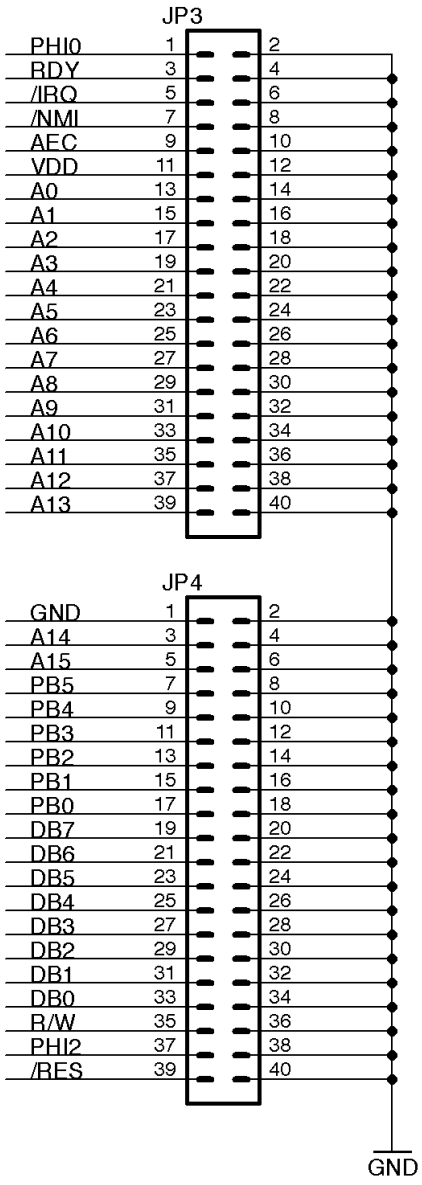
[3]**Note:** Pin 5 is Not Connected on NMOS 6502 and 6510 microprocessors. The C74-6502 drives Pin 5 with /ML. If the host system has PIN 5 grounded, then the pin on the Adapter should be removed so as to leave it floating for the CPU to drive.

Appendix B
Headers & Pinouts

6502 Pinout



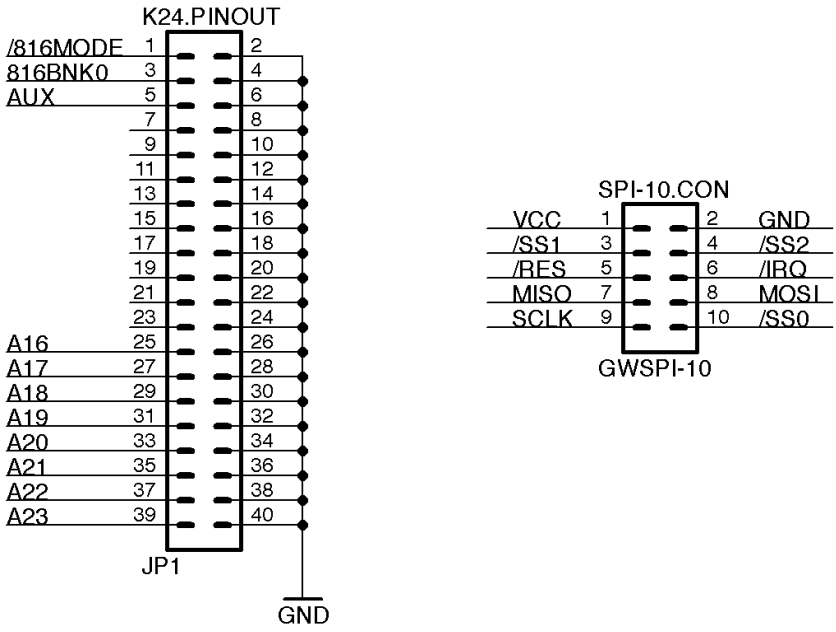
6510 Pinout



6502 & 6510 CPU Pinout

- The 6502 header should be used when connecting to a host system intended for either a 6502 or 65C02.
- A jumper (SJ3) on Card A is used to enable the /VP signal on Pin 1 for compatibility with a WDC 65C02 socket. The jumper should be left open otherwise.
- RDY will pause the CPU if it is taken low anytime before the falling of PHI2, including for write cycles.

- RDY can operate as an input-only or as a bi-directional pin, for use with 65C02 WAI instruction. If use of the WAI instruction is anticipated, it is recommended that RDY be driven through a Schottky Diode and an accompanying 470 Ohm pull-up resistor. If RDY is driven directly by external circuitry, then the WAI instruction may not pause the CPU.
- The 65C02 BE pin and the 6510 AEC pin are equivalent in function and are connected to the same circuitry internally. When low, either will take the address bus, data bus and R/W pin to a high-impedance state asynchronously.
- Signals are active simultaneously on both headers so one can interrogate /VP on the 6502 header while using the 6510 header for all other signals. Note that the /ML signal is only active if the 65C02 or K24 microcode is selected.



Optional K24 Header and SPI.CON Pinout

- Two control signals are available on the K24 header to aid in address decoding. They are "/816MODE" on pin1 (low when the K24 microcode is active) and "/816BNKH" on pin 3 (low when an instruction addresses a bank other than bank 0). An internal address decoder, for example, uses these signals to enable the 6510 port when /816BNKH is high. This maps the 6510 port to Bank 0 when the K24 microcode is active but to all banks otherwise.
- The K24 header outputs the AUX control signal on Pin-5. It is taken directly from the CFG register bit-7 and may be used for any user-defined purpose. For example, the signal may be used to switch clock-rates under software control. This signal is pulled low if the CFG register is disabled.
- The K24 Card implements a simple SPI Mode 3 interface on the SPI.CON header, as shown above (based on Garth Wilson's SPI-10 standard, see http://forum.6502.org/viewtopic.php?f=4&t=4264). Up to three Slave Select signals are available. When in use, the CPU will toggle SCLK once per CPU cycle to provide very fast access to SPI devices.

Appendix C
Series Resistors & Socket Adapter

Series Resistors

The series resistors on Card A are meant to enhance signal integrity on connections to the host computer. Ideally the value of these resistors plus the output impedance of the drivers should match the characteristic impedance of the ribbon cable used. Series resistors R8, R21, and R35 are marked on the PCB with a white line to indicate they are 0 Ohm resistors. Their leads are connected internally so physical resistors need not be installed.

C74 Socket Adapter

There are two pinouts headers on the TTL CPU, one for the 6502 pinout and another for the 6510. Each pinout header has a total of 80 pins, with 40 pins carrying signals for the particular processor's pins, and 40 being ground connections for better signal integrity. Correspondingly, the C74 Socket Adapter has 80 pins as well, arranged as two 40-pin headers.

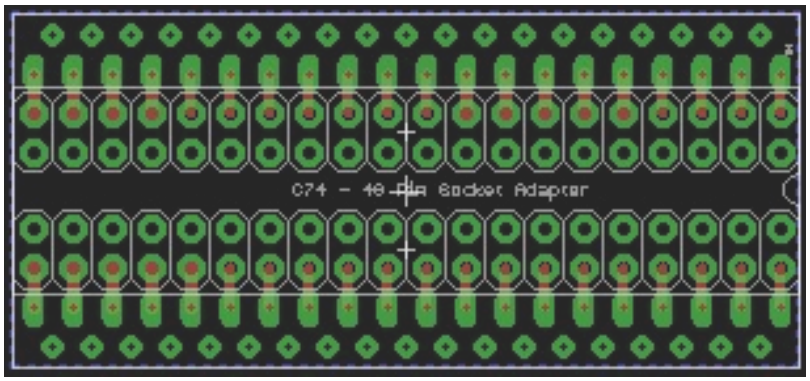


Fig. 7 - C74 40-Pin Socket Adapter

The inner rows of pins on the headers are connections to ground, while the outer rows carry processor signals to the Adapter's underside socket pins. The pads corresponding to GND and VDD socket pins of a given pinout must be explicitly connected to the grounded vias located on the top and bottom edges of the PCB. GND pins should be connected to a nearby via with a jumper or SMD 0 Ohm resistor. For VDD pins, a 0.1uF capacitor should be used. SMD 0603 or 0805 packages will all fit in the available space between pads and vias.

Refer to Appendix A above for correct VDD and GND pin assignments for the various supported CPUs. It is desirable to build multiple adapters, one for each pinout variation, to match pins assignments required for the various host systems[1].

[1]**Note:** The C74-6502 drives Pin 5 with /ML when either the WDC 65C02 or K24 microcode is selected. It may be desirable to remove Pin 5 from the Adapter if Pin 5 is grounded on the host system and use of either microcode is anticipated.

Appendix D
Sample Power Supply Circuit

The C74-6502 may be powered externally if the host does not provide enough current for the CPU to operate[1]. To do so, the “Host Power” jumper on Card A should be dis-engaged and the on-board power terminals connected to an external power source. Below is a suggested external power circuit in which a buffer amplifier tracks the host-power as it ramps up, causing the CPU to ramp up with the same waveform. The circuit requires an op-amp and a power transistor that supply power from the external supply to the CPU. The "+" input of the op-amp attaches to the host "+5". The "-" input of the op-amp is for feedback, creating a voltage follower. This simple circuit ensures that the instant that the host computer is at 2V, for example, the CPU will also be at 2V, and so on.

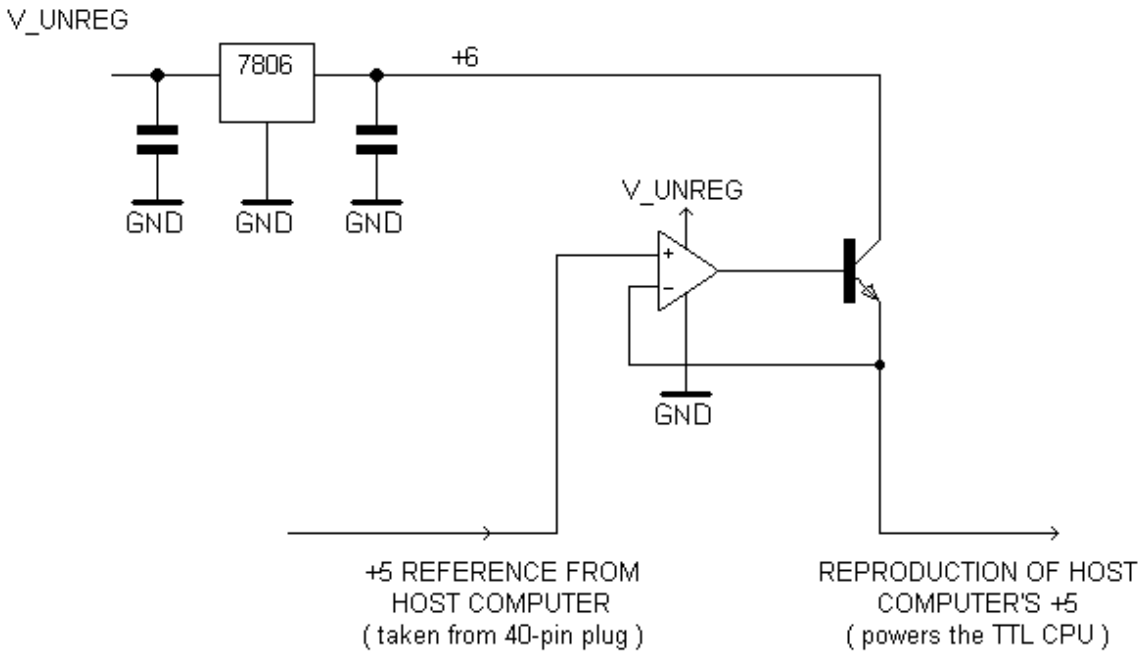


Fig. 8 - Power Supply Circuit

[1]The C74-6502’s current draw at 1MHz is comparable to the 130mA the NMOS 6502 and 6510 microprocessors require. The need for external supplemental power is therefore unlikely when connecting to legacy systems that use these microprocessors, such as the Commodore VIC 20 or C64 computers. Current draw may become an issue, however, when replacing a CMOS 65C02 microprocessor within and existing motherboard. In such cases, the auxiliary Power terminals on Card A may be used.

Appendix E
NMOS 6502 Undocumented Opcodes

The table below summarizes the C74-6502 implementation of NMOS Undocumented Opcodes. All Stable Opcodes are cycle and function accurate, with the exception of the notoriously complicated $6B (ARR) in Decimal Mode (See http://forum.6502.org/viewtopic.php?f=4&t=3493&start=117 for a detailed description of ARR by ttlworks). Note that some Undocumented Opcodes do not meet the requirements for 20MHz operation (see Appendix J for details).

Opcode (Aka)	OP	Addr	Bytes	Cycles	Flags	Function
KIL	$02	imp	1	?	None	Halt CPU (performs a 65C02 STP)
	$12	imp	1	?		
	$22	imp	1	?		
	$32	imp	1	?		
	$42	imp	1	?		
	$52	imp	1	?		
	$62	imp	1	?		
	$72	imp	1	?		
	$92	imp	1	?		
	$B2	imp	1	?		
	$D2	imp	1	?		
	$F2	imp	1	?		
LAX	$A7	zp	2	3	NZ	A, X <= MEM(adr)
	$B7	zp,y	2	4		
	$AF	abs	3	4		
	$BF	abs,y	3	4*		
	$A3	(ind,X)	2	6		
	$B3	(ind),y	2	5*		
LDD (NOP)	$80	imm	2	2	None	Load and Discard - Multi-cycle NOP

Opcode (Aka)	OP	Addr	Bytes	Cycles	Flags	Function
	$82	imm	2	2		
	$89	imm	2	2		
	$C2	imm	2	2		
	$E2	imm	2	2		
	$04	zp	2	3		
	$44	zp	2	3		
	$64	zp	2	3		
	$14	zp,x	2	4		
	$34	zp,x	2	4		
	$54	zp,x	2	4		
	$74	zp,x	2	4		
	$D4	zp,x	2	4		
	$F4	zp,x	2	4		
	$0C	abs	3	4		
	$1C	abs,x	3	4*		
	$3C	abs,x	3	4*		
	$5C	abs,x	3	4*		
	$7C	abs,x	3	4*		
	$DC	abs,x	3	4*		
	$FC	abs,x	3	4*		
NOP	$1A	imp	1	2	None	No Operation - 2-cycle NOP
	$3A	imp	1	2		
	$5A	imp	1	2		
	$7A	imp	1	2		
	$DA	imp	1	2		
	$FA	imp	1	2		
RLA	$27	zp	2	5	NZC	MEM(Adr) <= ROL(adr), A <= A AND (adr)
	$37	zp,x	2	6		
	$2F	abs	3	6		

Opcode (Aka)	OP	Addr	Bytes	Cycles	Flags	Function
	$3F	abs,x	3	7		
	$3B	abs,y	3	7		
	$23	(ind,x)	2	8		
	$33	(ind),y	2	8		
RRA	$67	zp	2	5	NZCV	MEM(adr) <= ROR (adr), A <= A ADC (adr)
	$77	zp,x	2	6		
	$6F	abs	3	6		
	$7F	abs,x	3	7		
	$7B	abs,y	3	7		
	$63	(ind,x)	2	8		
	$73	(ind),y	2	8		
SLO	$07	zp	2	5	NZC	MEM(adr) <= ASL(adr), A <= A ORA(adr)
	$17	zp,x	2	6		
	$0F	abs	3	6		
	$1F	abs,x	3	7		
	$1B	abs,y	3	7		
	$03	(ind,x)	2	8		
	$13	(ind),y	2	8		
SRE	$47	zp	2	5	NZC	MEM(adr) <= LSR (adr), A <= A EOR (adr)
	$57	zp,x	2	6		
	$4F	abs	3	6		
	$5F	abs,x	3	7		
	$5B	abs,y	3	7		
	$43	(ind,x)	2	8		
	$53	(ind),y	2	8		
DCP	$C7	zp	2	5	NZC	MEM(adr) <= DEC (adr), A <= A CMP (adr)

Opcode (Aka)	OP	Addr	Bytes	Cycles	Flags	Function
	$D7	zp,x	2	6		
	$CF	abs	3	6		
	$DF	abs,x	3	7		
	$DB	abs,y	3	7		
	$C3	(ind,x)	2	8		
	$D3	(ind),y	2	8		
ISC (ISB)	$E7	zp	2	5	NZCV	MEM(adr) <- INC (adr), A <= A SBC (adr)
	$F7	zp,x	2	6		
	$EF	abs	3	6		
	$FF	abs,x	3	7		
	$FB	abs,y	3	7		
	$E3	(ind,x)	2	8		
	$F3	(ind),y	2	8		
SBC	$EB	imm	2	2	NZCV	A <= A SBC (imm)
ANC	$0B	Imm	2	2	NZC	A <= A AND Imm, Carry <= A.7;
	$2B	Imm	2	2		
LAS (LAR)	$BB	abs,y	3	4*	NZ	A,X,SP <= MEM(adr) & SP
			“A&X” Group			
SAX (AAX)	$87	zpg	2	3	None	MEM(Adr) <- A&X
	$97	zpg,y	2	4		
	$83	(ind,X)	2	6		
	$8F	Abs	3	4		
AXS	$CB	imm	2	2	NZC	X <= A&X SBC1 Imm;
						where SBC1 means SBC w/ C = 1.
						Always in binary mode

Opcode (Aka)	OP	Addr	Bytes	Cycles	Flags	Function
	"Unstables" - Magic Constant Group					
ALR	$4B	imm	2	2	NZC	A <= A AND (MAGIC | #IMM), A <= LSR A
ARR	$6B	imm	2	2	NZCV	A <= A & (MAGIC | #IMM), ROR A;
						Special Flags Handling in Binary Mode:
						C <= A.6, V <= A.6 XOR A.5
						Decimal Mode NOT IMPLEMENTED
						Will behave the same as Binary Mode
LXA	$AB	imm	2	2	NZ	A,X <= (MAGIC | A) & #IMM
XAA	$8B	imm	2	2	None	(MAGIC | A) & X & #IMM => A$FF"
	"Unstables" - & (H + 1) Group					
AHX	$9F	abs,Y	3	5	None	MEM(adr) <= A&X AND hi(adr) + 1*
	$93	(ind),Y	2	6		
SHX	$9E	abs,y	3	5	None	MEM(adr) <= X AND hi(adr) + 1*
SHY	$9C	abs,x	3	5	None	MEM(adr) <= Y AND hi(addr) + 1*
TAS	$9B	abs,y	3	5	None	SP <= A&X,
						MEM(Adr) <= A&X AND hi(adr) + 1*
						*where hi(adr) + 1 refers to the upper byte of the target address plus 1

Appendix F
NMOS 6502 Unstable Opcodes

"Unstable Opcodes" are known as such because their behaviour varies based on the characteristics of the host computer, and with the operating environment at the time of execution. This variability presents immense challenges when trying to model these opcodes. There is simply too much ambiguity to define a single deterministic function; any one behaviour will work in some instances and fail in others. A practical approach, therefore, is to settle on one set of "common" behaviours and hope to address a meaningful subset of all possible variations. The C74-6502 adopts just such an approach, and treats unstable opcodes as follows:

1. **"& (H+1)" Group: AHX($9F, $93), SHX($9E), SHY$(9C), TAS($9B)** - these opcodes exhibit two instabilities - the "& (H+1)" operation is "sometimes" dropped and the high-byte of the target address may be corrupted on page-crossings. Tests on a VIC20 show that page-crossings that stay within the lower nibble of the high-byte appears to work correctly, but it's hard to be definitive about this. Consequently, the C74-6502 honours the "& (H+1)" construct in all cases and ignores any page-crossing anomalies.

2. **MAGIC Constant Group: ALR($4B), ARR($6B), XAA/ANE($8B), LXA ($AB)** - these opcodes are dependent on a so-called MAGIC constant which in fact changes across systems and even within specific systems at different times. Some claim that of these opcodes, only $8B and $AB are in fact unstable, but specific tests showed differences do exist with the others as well. There simply seems to be no one right answer for the value of MAGIC. The C74-6502 somewhat arbitrarily uses specific values for MAGIC as convenient and plausible options knowing full well this will not work in all instances. With the chosen values, these opcodes reduce to the following functions:

- **ALR($4B**): A <= A & (MAGIC | #IMM), A <= LSR A, where (MAGIC = $00) **becomes**
  A <= A & #IMM, A <= LSR A
- **ARR($6B)**: A <= A & (MAGIC | #IMM), ROR A, where (MAGIC = $00) **becomes** A <= A & #IMM, ROR A
- **XAA($8B)**: A <= (MAGIC | A) & X & #IMM, where (MAGIC = $FF) **becomes** A <= X & #IMM
- **LXA($AB)**: A,X <= (MAGIC | A) & #IMM, where (MAGIC = $FF) **becomes** A,X <= #IMM.

As noted above, this set of behaviours is at best a compromise, but nevertheless provide a reasonable approach to these opcodes. Namely, that software that properly accounts for the variabilities in the "unstables" will run on the C74-6502 without a problem. This can be done, as the excellent paper "No More Secrets" (http://csdb.dk/release/?id=143981) suggests, by not relying on a given value for MAGIC and by keeping page boundaries well away from "& (H+1)" instructions. Otherwise, all bets are off on this as well as all other 6502 implementations.

“Do not use [Magic Constant Group Opcodes] with any other immediate value than 0, or when the accumulator value is $ff (both takes the magic constant out of the equation)! (Or, more generalized, these are safe if all bits that could be 0 in A are 0 in either the immediate value or X or both.)”

NO MORE SECRETS, NMOS 6502 Unintended Opcodes.

Appendix G
65C02 NOPs

The 65C02 instruction set guarantees that all unused opcodes are NOPs, but in fact not all NOPs are created equal. Several NOP operations in fact follow standard addressing modes and perform specific memory reads, just as LDA might, but then simply discard the value retrieved. Dr Jefyll has dubbed these opcodes “LDD” operations, for Load and Discard, which is a fitting name[1]. Other NOPs are single-byte, single-cycle operations which immediately fetch the next opcode - we’ll call these “NOP1” opcodes. In order to maintain compatibility with the 65C02, the C74-6502 implements these opcodes as follows:

Opcode	Mnemonic	Bytes	Cycles
$x3, $xB (all opcodes ending in $3 or $B)[2]	NOP1	1	1
$02, $22, $42, $62, $82, $C2, $E2	LDD #	2	2
$44	LDD zpg	2	3
$54, $D4, $F4	LDD zpg, X	2	4
$5C	N/A	3	8
$DC, $FC	LDD abs	3	4

It’s interesting to note that LDD opcodes follow exactly the same bus cycles as the corresponding load instructions, except $5C which takes 8 cycles.

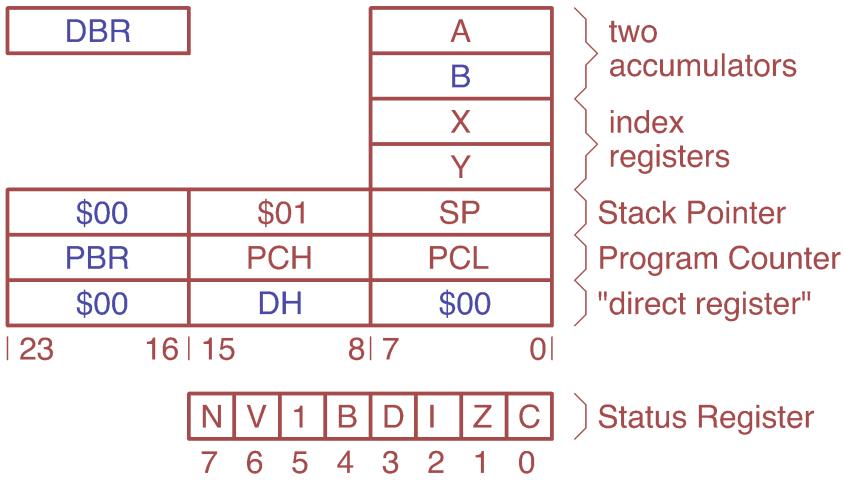
[1]See http://laughtonelectronics.com/Arcana/KimKlone/Kimklone_opcode_mapping.html

[2]Except for WAI ($CB) and STP ($DB) opcodes

Appendix H
K24 Microcode Compatibility Notes

The C74-6502 K24 microcode implements 65816 instructions as they behave when that processor's Emulation Mode is active (i.e., E-flag is set to 1). In that situation, the 65816's 24-bit address space and extended addressing modes are available, but the A, B, X and Y registers are all 8 bits (M and X flags are both implied to be 1).

Fig. 9 - Programming Model (K24 Additions In Blue)



The C74-6502 K24 implementation of 65816 instructions has the following incompatibilities:

- It is not cycle-accurate
- The MVN, MVP, COP and XCE opcodes are not implemented.
- The ABORT and COP Interrupt Vectors are not implemented. K24 uses the same addresses as the 6502 for RES, NMI and IRQ vectors.
- Opcode $42 (WDM) is replaced by the CFG instruction, which will exchange the contents of the A accumulator with the CFG register
- Opcode $02 (COP) is replaced by the SPI instruction, which will swap 8 bits with a selected SPI device as documented above
- The Direct Page register (DH) is 8 bits and points only to page boundaries. The PHD and PLD instructions still push and pull 16 bits on the stack with $00 as the low byte.
- The A & B accumulators cannot be combined into a 16 bit C register. TCS, TSC use only the A accumulator (low-byte); TCD, TDC only the B accumulator (high-byte).
- SP is 8 bits. The stack is fixed to bank $00 page $01 and all opcodes which push or pull values on the stack will wrap on a page boundary.
- Other rules for crossing bank boundaries are correctly applied (e.g., Abs,Y or [Ind],Y)

Appendix I
Rudimentary Multitasking

The C74-6502's CFG opcode enables the processor to switch instruction-sets under program control. Since CFG does not alter the values in the Data Bank Register or the Program Bank Register, 6502 and 65C02 tasks can be made to run in different 64K memory banks concurrently and independently. To illustrate, below is sample code for a very rudimentary preemptive multitasking scheme which executes a monitor program in Bank 0 and runs three tasks concurrently. This example runs Klaus Dormann's NMOS 6502 Test Suite in Bank 1, his CMOS 65C02 Test Suite in Bank 2 and Bruce Clark's NMOS 6502 Decimal Test in Bank 3.

The monitor program runs in K24 mode and interrupts the currently executing task at regular intervals on an NMI. It saves state, switches banks, restores state and performs an RTI to launch the previously interrupted task. NMIISR and RESUME are stub code segments that need to placed in each bank at some open same location. NMI vectors in every bank should point to NMIISR and the long-JMP at the end of the monitor program must point back to the RESUME stub (note that the monitor program self-modifies the long-jump address with the current bank). The monitor's RESET entry point performs initialization and launches the first task. Bank 0 RESET vectors should be pointed to it.

```
; NMI interrupt Service
; S/b loaded into every bank
; NMI vectors point here

NMIISR  sei
        pha             ; save state
        txa
        pha
        tya
        pha

        lda #3          ; Switch to K24
        .byte $42       ; CFG opcode
        .byte $00

        .byte $5C       ; JMP lng
        .word BNKSW     ; to Monitor
        .byte $00       ; in bank 0
```

```
; Resume entry point to re-launch task.
; S/b loaded into every bank
; JMP target from BNKSW below

RESUME .byte $42        ; CFG in A
        .byte $00
        pla             ; restore state
        tay
        pla
        tax
        pla
        rti
```

```
; MONITOR loads at $0000 bank 0
; s/b invoked on RESET to initialize

*  = $0000

RESET    sei

; Enable 65816 Mode

            lda #3
            .byte $42        ; CFG Opcode
            .byte 00

; Initialize SP in Task Table

            lda #F9          ;Set SP to $F9 to
            sta csp+1        ;allow room for
            sta csp+2        ;initial stk frame
            sta csp+3        ; 3 bytes RTI
                             ; 3 bytes A, X, Y

; initialize RTI stack frame in each bank
; initial values for A, X, Y are left undefined

            lda #$04         ; Start @ $0400
            sta $0101FF
            sta $0201FF
            lad #$02         ; Start @ $0200
            sta $0301FF

            lda #$00
            sta $0101FE
            sta $0101FD
            sta $0201FE
            sta $0201FD
            sta $0301FE
            sta $0301FD

            stz cbnk         ; start in bnk 0
```

```
; BANK SWITCH entry Point

BNKSW    lda #0           ; go to bnk0
            pha
            plb

            ldx cbnk         ; index task tabl
            tsc              ; save SP to tabl
            sta csp,x

            inx              ; next bnk
            cpx #maxbnks
            bne skp
            ldx #$01

skp         stx cbnk

            lda csp,x        ; restore SP
            tcs
            lda ccfg,x       ; restor CFG to A

            phx              ; save cbnk
            plb              ; switch cbnk

            ; jump to the target bank (cbnk)

            .byte $5C        ; JMP lng
            .word RESUME
cbnk        .byte $00

; Task Table

maxbnks  = 4

csp         .byte $F9        ; SP
            .byte $F9
            .byte $F9
            .byte $F9

ccfg        .byte $00        ; CFG
            .byte $00        ; 6502
            .byte $71        ; 65C02
            .byte $02        ; 6502+NOP
```

Appendix J
Notes & Caveats

Functional Differences For NMOS 6502 Operation:

- A branch-taken does NOT delay detection of interrupts to the next instruction
- A BRK that is pre-empted by an interrupt will not be ignored but will execute once the interrupt completes
- “Unstable” opcodes are Undocumented opcodes whose behaviour is non-deterministic — that is, their varies depending on the characteristics of the host system and other environmental factors. On the C74-6502 they have been given a fixed functionality as outlined in Appendix F
- The Undocumented Opcode ARR behaves the same way in Decimal as in Binary mode

Other Compatibility Notes:

- Decimal Mode SBC operations with invalid BCD inputs generate results matching the NMOS 6502 in all instances, regardless of the selected microcode
- All 6502 variants perform a spurious read of the target address during absolute-indexed store operations where a page-crossing does NOT occur. The C74-6502 reads the Previous Bus Address in such circumstances when the 65C02 instruction set is active.
- Writes to memory are inhibited when /RES goes low. The Reset sequence is invoked on the next SYNC cycle following the rising-edge of /RES
- If RDY is driven directly externally, the WAI instruction will not drive RDY low and the CPU will not pause. It is recommended that RDY be driven externally through a Shottky diode and accompanying pull-up resistor for use with the WAI instruction
- The 65816 instruction-set incompatibilities are outlined in Appendix H

Notes On 20MHz Operation:

- "Typical" Total Propagation Delay figures have been used to calculate the CPU's critical path. The C74-6502 has been tested for stable operation at 20MHz. Actual performance may slower (or faster) under various operating conditions.
- The PHI2 output clock signal is delayed by approximately 20ns relative to the PHI0 input clock (when the ZDB jumper is not engaged). If clocking using PHI0 input clock, enable ZDB.
- The operating option BCD-Flags-Valid provides compatibility with 65C02 Decimal Mode but imposes a slight delay doing so. The 2-Cycle-BCD operating option provides an internal wait-state to accommodate this delay, exactly as the 65C02 does. These two options, therefore, should be enabled together for full compatibility. Leaving 2-Cycle-BCD disabled removes this internal wait-state and may therefore prevent the correct operation of the Decimal Mode circuitry at high clock-rates.

- NMOS 6502 Undocumented Opcodes are typically used only in rare and specialized situations - mostly to obfuscate copy-protection mechanisms. When used, a select few of these may limit the C74-6502's maximum clock-rate as follows:

Opcode	Clock-Rate[1]
SAX, AXS, ANE	19MHz
Decimal Mode RRA, ISC	17MHz
AHX, SHX, SHY, TAS	14MHz

[1]The clock-rate quoted is based on "typical" tdp figures. Actual limits may be higher in practice.

Appendix K
Unnecessary Reads (Dead Cycles)

A Dead Cycle is one in which the CPU is busy with an internal operation and does not make specific use of the external buses. The following table summarizes the behaviour of the NMOS 6502 and CMOS 65C02 processors, and the corresponding implementation of the C74-6502 instructions-sets. Note the highlight in line 3 of the table where the C74-6502 behaviour differs from the CMOS 65C02.

	6502 Dead Cycles[1] Opcode/Addressing Mode	Cycle[2,3]	Internal Operation[4]	External Operation 6502[15]	External Operation 65C02[15]	External Operation C74-6502	External Operation C74-65C02
1	zpg,X/Y	3	BAL + X	PFA[6]	PBA[7]	PFA[6]	PBA[7]
2	abs,X/Y — read w/ pg. crossing	4*	BAH + 1	PFA	PBA	PFA	PBA
3	abs,X/Y — write w/o pg. crossing[8]	4	BAH + 0	FFA[5,8]	FFA[5,8]	FFA[5,8]	PBA
4	abs,X/Y — write w/ pg. crossing	4	BAH + 1	PFA	PBA	PFA	PBA
5	(zpg),Y — read w/ pg. crossing	5*	BAH + 1	PFA	PBA	PFA	PBA
6	(zpg),Y — write w/o pg.crossing	5	BAH + 0	FFA	PBA	FFA	PBA
7	(zpg),Y — write w/ pg.crossing	5	BAH + 1	PFA	PBA	PFA	PBA
8	(zpg,X)	3	BAL + X	PFA	PBA	PFA	PBA
9	RMW zpg,X — "Read, Modify, Write" Opcodes[11]	3	BAL + X	PFA	PBA	PFA	PBA
10	RMW abs,X — 6502 w/o pg. crossing	4	BAH + 0	FFA		FFA	
11	RMW abs,X — 6502 w/ pg. crossing	4	BAH + 1	PFA		PFA	
12	RMW abs,X — 65C02 INC/DEC w/o pg. crossing	4	BAH + 0		FFA		FFA
13	RMW abs,X — 65C02 INC/DEC w/ pg. crossing	4	BAH + 1		PBA		PBA
14	RMW abs,X — 65C02 ASL LSR ROL ROR w/ pg. crossing[12]	4*	BAH + 1		PBA		PBA
15	RMW zpg	4	Modify	FFA (W)[9]	FFA	FFA (W)[9]	FFA
16	RMW abs	5	Modify	FFA (W)	FFA	FFA (W)	FFA
17	RMW zpg,X	5	Modify	FFA (W)	FFA	FFA (W)	FFA
18	RMW abs,X	[6]12	Modify	FFA (W)	FFA	FFA (W)	FFA
19	All 1 Byte Opcodes	2	—	PC[10]	PC[10]	PC[10]	PC[10]
20	JMP (abs)/(abs,X) — 65C02	4	—		PBA		PBA
21	JSR	3	—	SP	SP	SP	SP
22	RTS, RTI, PLP, PLA, PLY, PLX	3	SP + 1	SP	SP	SP	SP
23	RTS	6	PC + 1	PC[10]	PC[10]	PC[10]	PC[10]
24	Branch (taken)[14]	3	PCL + BOF[13]	PC[10]	PC[10]	PC[10]	PC[10]
25	Branch (taken) — w/ pg. crossing[14]	4*	PCH +/- 1	PC[10]	PC[10]	PC[10]	PC[10]
26	Extra BCD Cycle — 65C02	*	BCD		PBA		PBA

NOTES:

1. A Dead Cycle is one in which the CPU is busy with an internal operation and does not make specific use of the external buses.
2. Cycle numbers are with reference to Fetch_Opcode as cycle 1.
3. "*" denotes the cycle is added on a page crossing.
4. BAL and BAH refer to "Base Address Low" and "Base Address High" respectively.
5. FFA refers to a Fully Formed Address, which is the final target address of a given addressing mode.
6. PFA refers to a Partially-Formed Address, one which has yet to be adjusted, as follows
   - zpg,X/Y — (""00"", BAL) Base address before the index register is added to the low-byte (BAL doesn't yet reflect the index offset
   - abs,X/Y — (BAH, BAL+X/Y) Base address before the low-byte carry is added to the high-byte (BAH doesn't yet reflect the page crossing)
   - (zpg),Y — (BAH, BAL+Y) Base address before the low-byte carry is added to the high-byte (BAH doesn't yet reflect the page crossing)

- (zpg,X) — (""00"", BAL) Base address before the index register is added to the low-byte (BAL doesn't yet reflect the index offset)

7. PBA refers to the Previous Bus Address (i.e., the value on the address bus from the previous cycle). This is the “fix” introduced by the 65C02. Re-reading the PBA is assumed to be a safe action, preferable to generating a "stray" read with an "invalid address", aka a Partially Formed Address (PFA), as the NMOS 6502 does. A PBA is also used on the 65C02 as a "safe" address for the dead cycle in JMP (abs) and the extra cycle in BCD operations.
8. abs,X/Y write operations without a page-crossing actually read from the Fully Formed Address before writing to it. The read can be troublesome when accessing I/O devices where reads are destructive. For 65C02 there's a software workaround, which is to ensure that the write to abs,X/Y triggers a page-crossing, which means the address during the dead cycle will be a PBA, not the Fully Formed Address. The only software workaround that works on both NMOS and CMOS 6502 is to avoid abs,X/Y address mode when writing to the read-sensitive device.
9. External Operations are Reads unless otherwise noted by "(W)".
10. PC refers to the address at PC, as follows:
    - 1-Byte Opcodes Cycle 2 — Address of next opcode
    - RTS Cycle 6 — Return address - 1 (as retrieved from the stack
    - Branch (taken) Cycle 3 — Address of next opcode after Branch
    - Branch (taken) w/ pg. crossing Cycle 4 — High-byte unchanged from prior cycle/Low-byte of target address (PCL + BOF[13])
11. "Read, Write, Modify" Opcodes refers to INC, DEC, ASL, LSR, ROL and ROR.
12. On the 65C02, the "Modify" operation occurs in cycle 5 for ASL, LSR, ROL and ROR if a page is not crossed.
13. BOF refers to the Branch Offset value.
14. Dead cycles are 5 and 6* for 65C02 BBR and BBS (rather than cycles 3 and 4* for standard branches).
15. The behaviour shown for each CPU has been verified on the Visual 6502 (www.visual6502.org) and on a WDC 65C02 respectively. See also http://archive.6502.org/books/mcs6500_family_hardware_manual.pdf Appendix A.