

# APPLE-1-STEPPER

v1.2



Avete mai desiderato osservare in tempo reale il funzionamento di un microprocessore mentre esegue un programma, con la precisione del singolo ciclo di clock?

Oppure osservare le istruzioni eseguite al boot ?

E ancora: vi siete mai sentiti frustrati perché il vostro programma proprio non funziona come vorreste, e non riuscite a trovare il problema?

Il vostro computer non funziona come dovrebbe e desiderate vedere quello che succede sui vari BUS ma non disponete di un analizzatore logico?

Volete avere la possibilità di scrivere sull'Apple-1 usando un terminale esterno USB per caricare brevi programmi?

## 1. DESCRIZIONE

---

*Apple-1-Stepper* è un dispositivo hardware che si collega a computer Apple-1, Originale o Replica.

Esso consente di:

- fermare l'esecuzione delle istruzioni da parte del microprocessore per scopi di debug, istruzione o troubleshooting per poi riprenderla in seguito,
- comandare manualmente o mediante un timer interno l'esecuzione delle istruzioni,
- visualizzare lo stato e il contenuto dei BUS Dati e Indirizzi, l'istruzione in esecuzione, lo stato della linea R/W, lo stato dei segnali IRQ e NMI, sia su display LCD che su terminale seriale.

Mediante il collegamento supplementare opzionale, potrete inoltre:

- caricare brevi programmi diagnostici,
- scrivere ed inviare programmi all'Apple-1 da dispositivi esterni.

## 2. RISCHIO CARICHE ELETTROSTATICHE

---

*Apple-1-Stepper* è sensibile all'elettricità statica, come il vostro computer Apple, e potrebbe venirne danneggiato.

Prima di qualsiasi operazione sul dispositivo è necessario scaricare l'elettricità statica accumulata dal vostro corpo e prevenirne un nuovo accumulo.

Non ci assumiamo alcuna responsabilità per danni, anche gravi o letali, causati a persone / cose / proprietà intellettuali durante l'installazione o l'utilizzo di questo dispositivo.

### 3. AVVERTENZE IMPORTANTI DURANTE L'USO

---

Per evitare danni al vostro computer Apple-1, se intendete usare la porta USB di *Apple-1-Stepper* osservate sempre scrupolosamente le seguenti istruzioni:

**DISCONNETTERE SEMPRE la porta USB dal computer PRIMA di spegnere l'Apple-1.  
NON COLLEGARE MAI la porta USB al computer se l'Apple-1 è spento.**

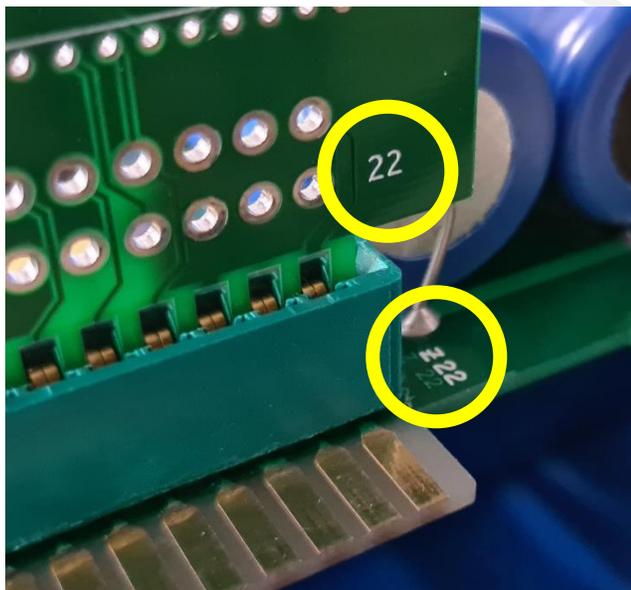
La sequenza corretta di accensione/spegnimento è la seguente:

- 1) **Accendere** Apple-1,
- 2) **Collegare** la porta USB al computer.  
... utilizzare il computer. Quando si decide spegnerlo:
- 3) **Scollegare** la porta USB dal computer,
- 4) **Spegnere** l'Apple-1.

### 4. INSERIMENTO DELLA SCHEDA

---

La scheda va inserita come riportato in figura:



Il numero “22” sulla scheda e il numero “22” sulla motherboard devono trovarsi sulla stessa estremità.

Il numero “22” sulla scheda deve dunque trovarsi rivolto verso il lato esterno della motherboard dell'Apple-1

L'indicazione va rispettata sia quando viene usato il connettore a pettine sia quando viene usato il connettore laterale.

L'utilizzo della scheda in abbinamento a BUS EXTENDER PASSIVI potrebbe generare malfunzionamenti: non utilizzate questo genere di dispositivi.

**ATTENZIONE:** L'accensione con la scheda orientata non correttamente **danneggia istantaneamente** il computer e la scheda stessa.

## 5. DESIGN E COSTRUZIONE

---

La scheda dalla strana forma, che verrà descritta tra poco, può essere inserita nel connettore femmina, tradizionalmente riservato all'Interfaccia Cassette, oppure collegata alla motherboard mediante il connettore laterale.

In questo caso sarà possibile far coesistere *Apple-1-Stepper* con altre schede, tipo la già menzionata Interfaccia Cassette, o le schede Juke-Box / CFFA1 o qualsiasi altra cosa.

*La forma della scheda rende dunque possibile l'alloggiamento di schede supplementari nel connettore femmina tradizionale, senza che vi siano rischi di contatti fisici/elettrici indesiderati tra di esse.*

*La massima altezza consentita per le schede di espansione supplementari è di 100mm.*

Diversamente da altri progetti simili riguardanti la manipolazione del clock, Apple-1 presenta delle difficoltà maggiori:

- il microprocessore tradizionalmente usato, l'NMOS 6502, non consente di ridurre il clock fino a zero, pena la corruzione dei registri interni,
- Apple-1, inoltre, usa Memoria RAM Dinamica. Gli indispensabili cicli necessari al suo refresh dipendono dal clock di sistema, se questo viene rallentato troppo o addirittura fermato, il contenuto della memoria verrà corrotto.
- Anche la temporizzazione dei componenti del segnale video, infine, dipende dall'oscillatore principale.

Non è dunque praticabile una manipolazione del clock di sistema nella accezione classica del termine.

Dal punto di vista funzionale la scheda implementa il circuito suggerito nel Manuale Operativo dell'Apple-1, più molti automatismi che verranno presto descritti.

Il circuito proposto non va dunque a modificare il clock di sistema, ma usa la linea RDY del microprocessore per fermarne le operazioni al momento opportuno.

L'Arduino MEGA presente sulla scheda si occupa della gestione di tale circuito, tenendo inoltre traccia di quanto transita sui BUS Dati e Indirizzi e sulle altre linee di controllo.

Esso gestisce inoltre anche il display LCD e il suo tastierino, la comunicazione USB e l'interfaccia verso la tastiera dell'Apple-1.

Tutte le linee dei BUS dell'Apple-1 sono state elettricamente disaccoppiate in modo da ridurre interferenze / *cross-talking* e consentire l'uso di altre schede in contemporanea senza problemi.

## 6. USO

*Apple-1-Stepper* può essere controllato in due modi: tramite il tastierino sotto il display LCD e tramite terminale collegato alla interfaccia USB dell'Arduino MEGA.

All'avvio esso si predispose in modo totalmente passivo (FREE RUN) e non effettua alcuna manipolazione della linea RDY né alcuna lettura dei BUS e dei suoi segnali di controllo.

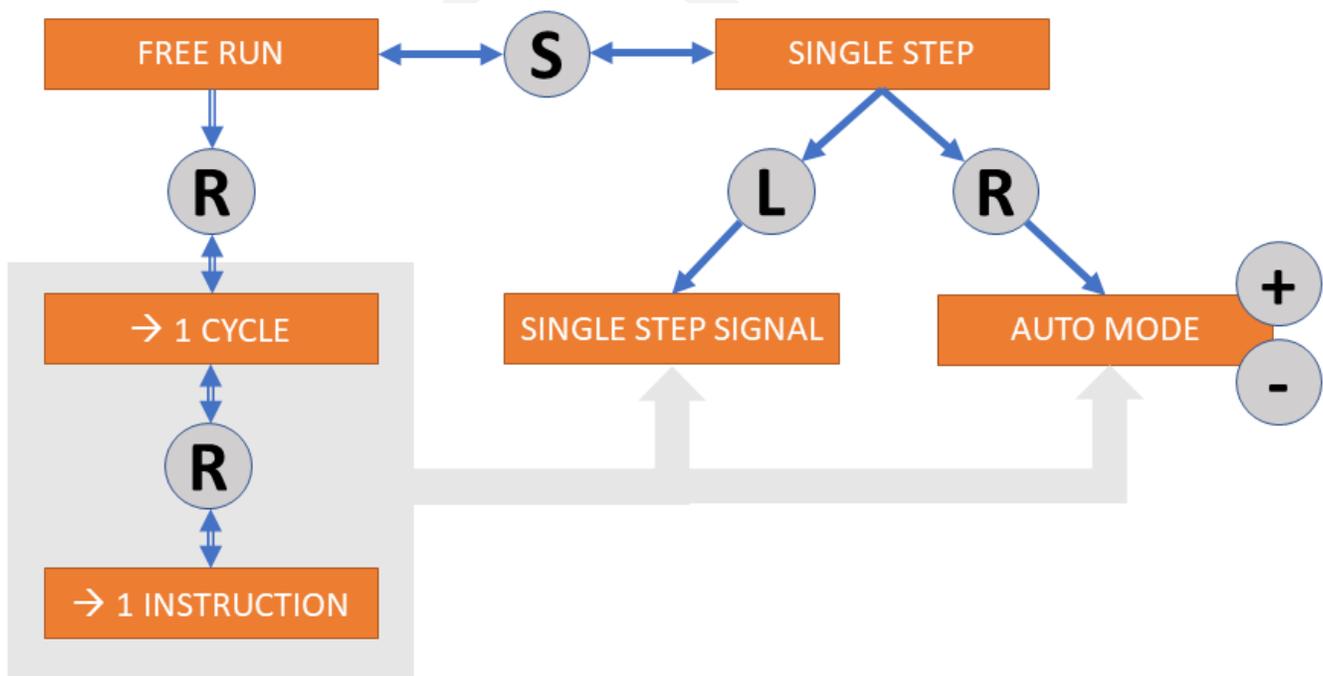
Il dispositivo, come previsto dallo schema presente nel manuale, quando in modalità SINGLE STEP può operare in due modalità:

- SINGLE CYCLE: ad ogni attivazione, manuale o temporizzata, il 6502 esegue un solo ciclo di clock.
- SINGLE INSTRUCTION: ad ogni attivazione, manuale o temporizzata, il 6502 esegue tanti cicli di clock quanti ne servono per completare l'istruzione che viene letta.

Questa predisposizione può essere effettuata solo in FREE RUN mode. Essa verrà tenuta in considerazione dal dispositivo quando esso sarà in modalità SINGLE STEP.

### 6.1 USO TRAMITE I PULSANTI INTEGRATI NEL DISPLAY LCD

Il seguente diagramma rappresenta gli stati del dispositivo in relazione alla pressione dei pulsanti SELECT/LEFT/RIGHT/UP (+)/DN (-).



Dalla modalità **FREE RUN**:

- La pressione ripetuta del tasto **RIGHT** permette di scegliere tra la modalità SINGLE CYCLE / SINGLE INSTRUCTION che verrà utilizzata in seguito. Tale selezione non influenza lo stato del dispositivo, che rimane in FREE RUN.



P-LAB APPLE-1  
RUN -> 1-CYCLE



P-LAB APPLE-1  
RUN -> 1-INSTR

- La pressione del tasto **SELECT** ferma le operazioni del microprocessore e lo pone in stato SINGLE STEP, in attesa.



P-LAB APPLE-1  
SINGLE 2 S/s

Dalla modalità **SINGLE STEP**:

- La pressione del tasto **LEFT** invia al microprocessore il segnale di **eseguire un ciclo di clock o una singola istruzione** come da predisposizione effettuata precedentemente in modo FREE RUN.



\$FF29 \$AD r LDA  
SINGLE 2 S/s

Contestualmente, appare sul display LCD il contenuto dei BUS Dati e Indirizzi in formato esadecimale. Se il contenuto della locazione è letto durante l'operazione di fetch da parte del 6502, viene visualizzato anche l'Opcodice corrispondente.

La pressione ripetuta del tasto **LEFT** causa l'esecuzione del programma uno step alla volta.

- La pressione del tasto **RIGHT** pone il dispositivo in modalità AUTO.



\$FF20 \$BB r BBB  
AUTO 8 S/s

Questa modalità consente di inviare in maniera automatica il segnale di esecuzione del singolo ciclo (oppure istruzione). È possibile selezionare tra 1 e 900 Step/secondo.

La frequenza viene regolata a piacere tramite i pulsanti **UP** e **DOWN** secondo valori prestabiliti.

A frequenze elevate la visualizzazione sul display risulterà inefficace, quindi si suggerisce di utilizzare questa opzione quando si è collegati a un terminale, in modo da catturare ogni singolo step.

Una successiva pressione del tasto **RIGHT** pone il dispositivo nuovamente in modalità manuale.

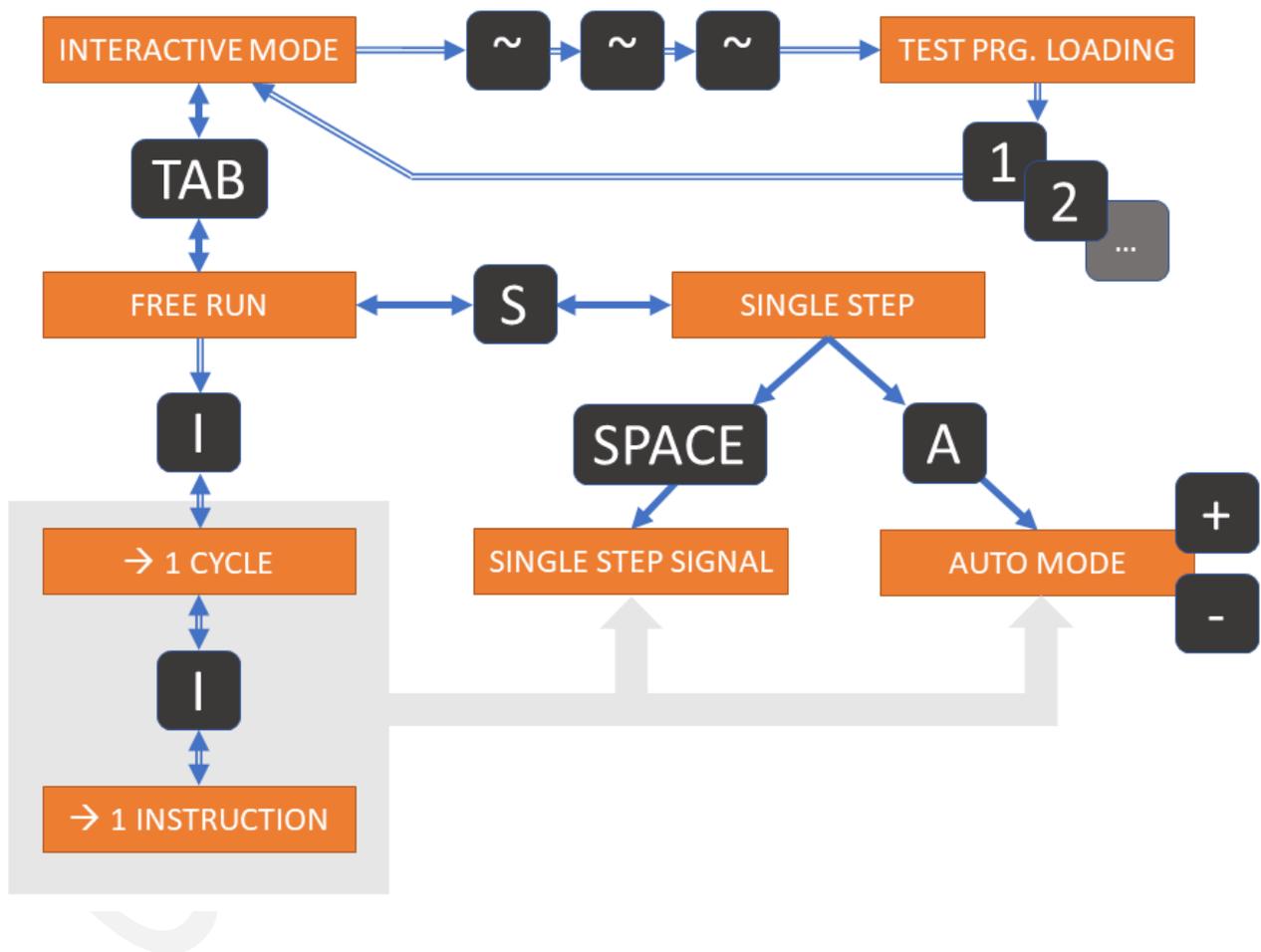
- La pressione del tasto **SELECT** pone il dispositivo in modalità FREE RUN.

## 6.2 USO MEDIANTE MONITOR USB/SERIALE

Le modalità di funzionamento di *Apple-1-Stepper* sono selezionabili anche tramite terminale seriale collegato tramite la tradizionale porta USB di Arduino.

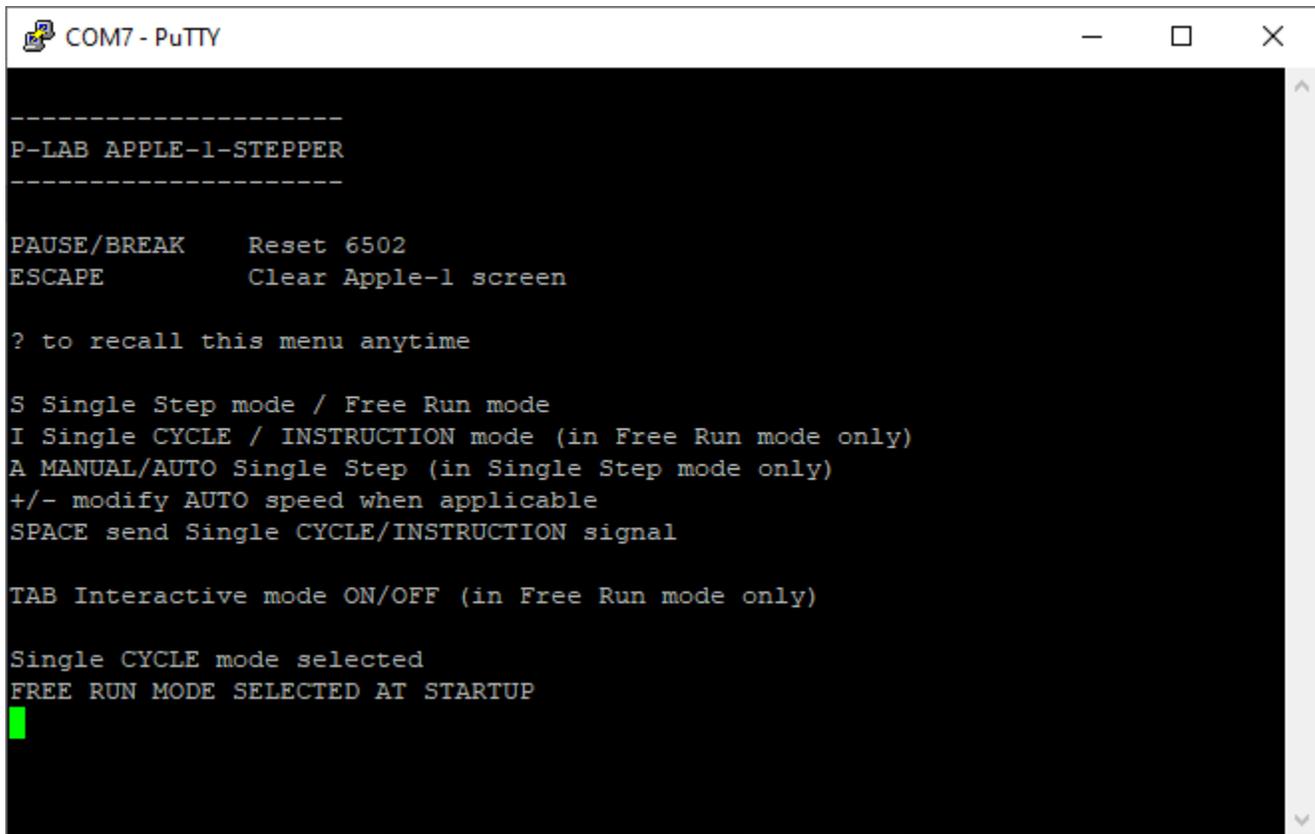
Dopo aver collegato il cavo secondo le indicazioni del paragrafo 3, aprire un programma terminale (ad es. PuTTY oppure minicom) e configurarlo in modo corretto per dialogare con Arduino. La velocità di collegamento è di **115200 baud**.

In virtù delle maggiori funzionalità disponibili in questa modalità, questo è il diagramma funzionale:



## 6.2.1 FUNZIONALITÀ BASE

All'avvio il dispositivo si predispose per la modalità comandi, e viene presentata la seguente schermata:



```
-----  
P-LAB APPLE-1-STEPPER  
-----  
  
PAUSE/BREAK      Reset 6502  
ESCAPE           Clear Apple-1 screen  
  
? to recall this menu anytime  
  
S Single Step mode / Free Run mode  
I Single CYCLE / INSTRUCTION mode (in Free Run mode only)  
A MANUAL/AUTO Single Step (in Single Step mode only)  
+/- modify AUTO speed when applicable  
SPACE send Single CYCLE/INSTRUCTION signal  
  
TAB Interactive mode ON/OFF (in Free Run mode only)  
  
Single CYCLE mode selected  
FREE RUN MODE SELECTED AT STARTUP  
█
```

In questa modalità i comandi per predisporre il dispositivo sono espressi da una sola lettera, in particolare:

**S** per passare dalla modalità FREE RUN alla modalità SINGLE STEP (analogamente al tasto SELECT del display LCD).

**I** per selezionare la modalità SINGLE CYCLE o la SINGLE INSTRUCTION, come con il tasto RIGHT del display LCD. Questa selezione è disponibile solo in modalità FREE RUN.

**A** permette di passare alla modalità AUTO quando ci si trova già in modalità SINGLE STEP. Anche in questo caso i tasti **+** e **-** consentono di aumentare o diminuire la velocità di esecuzione.

**SPAZIO**, quando premuto in modalità SINGLE STEP (ma non in modalità AUTO) consente di mandare il segnale al microprocessore per processare un singolo ciclo o una singola istruzione.

Il tasto **?** permette di richiamare il menu di riepilogo dei comandi in ogni momento, purché si sia in modalità comandi.

Ogni operazione causa un messaggio di conferma a video.

Due tasti speciali sono stati inoltre mappati come segue, e sono sempre attivi:  
(Attenzione: questi tasti potrebbero non funzionare con alcuni programmi terminale)

**PAUSE/BRK** causa il RESET del 6502 e in buona sostanza di tutto l'Apple-1, ma non di *Apple-1-Stepper*. Questa funzione è utile per vedere la consistenza del bootstrap del computer, per esempio.

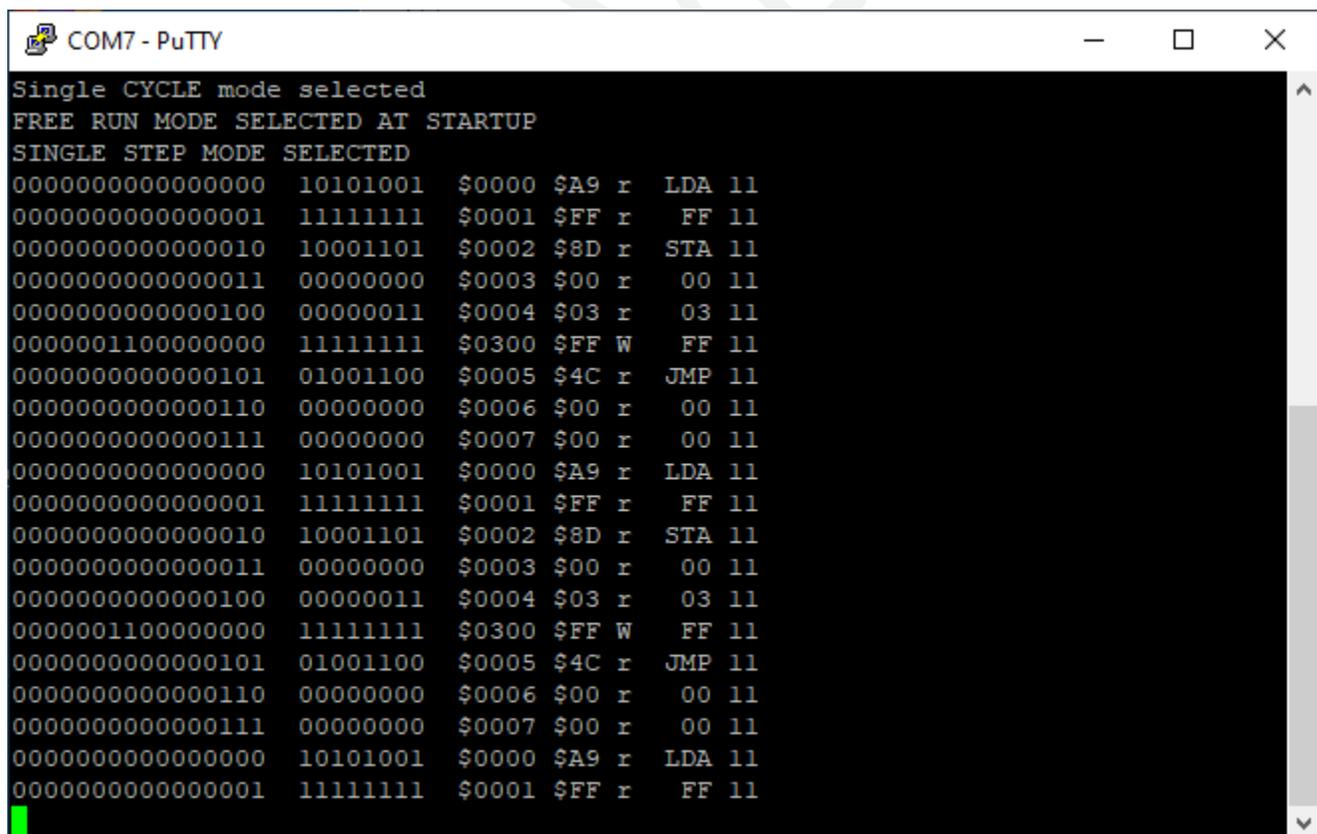
**ESCAPE** causa la cancellazione dello schermo dell'Apple-1. Questo tasto è efficace solo se il collegamento tra *Apple-1-Stepper* e il connettore tastiera è presente (vedi paragrafi successivi).

Per spiegare i dati visualizzati a schermo supponiamo ora di eseguire un semplice programma di prova, che carichi nell'Accumulatore il valore FF per poi depositarlo nella locazione di memoria \$0300 e ricominciare da capo:

```
0000 LDA #$FF
0002 STA $0300
0005 JMP $0000
```

Il codice esadecimale per tale programma, da inserire nell'Apple-1 sarà dunque:  
0: A9 FF 8D 00 03 4C 00 00

Se lo eseguiamo, col comando OR, esso girerà in un loop infinito. Utilizzando *Apple-1-Stepper* in modalità SINGLE CYCLE questo sarà quello che visualizzeremo a terminale dopo aver premuto SPAZIO un po' di volte:



```
COM7 - PuTTY
Single CYCLE mode selected
FREE RUN MODE SELECTED AT STARTUP
SINGLE STEP MODE SELECTED
0000000000000000 10101001 $0000 $A9 r LDA 11
0000000000000001 11111111 $0001 $FF r FF 11
0000000000000010 10001101 $0002 $8D r STA 11
0000000000000011 00000000 $0003 $00 r 00 11
0000000000000100 00000011 $0004 $03 r 03 11
0000001100000000 11111111 $0300 $FF W FF 11
0000000000000101 01001100 $0005 $4C r JMP 11
0000000000000110 00000000 $0006 $00 r 00 11
0000000000000111 00000000 $0007 $00 r 00 11
0000000000000000 10101001 $0000 $A9 r LDA 11
0000000000000001 11111111 $0001 $FF r FF 11
0000000000000010 10001101 $0002 $8D r STA 11
0000000000000011 00000000 $0003 $00 r 00 11
0000000000000100 00000011 $0004 $03 r 03 11
0000001100000000 11111111 $0300 $FF W FF 11
0000000000000101 01001100 $0005 $4C r JMP 11
0000000000000110 00000000 $0006 $00 r 00 11
0000000000000111 00000000 $0007 $00 r 00 11
0000000000000000 10101001 $0000 $A9 r LDA 11
0000000000000001 11111111 $0001 $FF r FF 11
```

Le informazioni a video posso essere così descritte:

- I primi sedici bit rappresentano le sedici linee del BUS Indirizzi,
- Gli otto bit successivi rappresentano le otto linee del BUS Dati,
- I due valori esadecimali seguenti sono rispettivamente Indirizzo e Dato convertiti dai valori binari precedenti,
- Il campo successivo è lo stato della linea R/W. Se è valorizzato “r” il microprocessore ha letto dall’indirizzo sopracitato il valore mostrato, se è valorizzato “w” il microprocessore ha scritto il Dato riportato nella locazione indicata (**solo 65C02**, vedi Appendice 1),
- A seguire viene mostrato il codice mnemonico dell’Opcode presente nel campo dati. La decodifica avviene solo se il microprocessore è nella fase di fetch. Diversamente viene mostrato il puro valore esadecimale,
- Gli ultimi due bit rappresentano rispettivamente lo stato della linea IRQ e della linea NMI. Entrambe sono normalmente inutilizzate, a meno di abbinamenti con schede di espansione (ad esempio: Apple-1 Wi-Fi Modem). Normalmente il loro valore è 1.

### INTERPRETAZIONE DEI RISULTATI

Quanto visualizzato nella schermata precedente è un valido esempio di come i vari eventi sono visualizzati ed è quindi semplice seguire il flusso del programma. Questo approccio può essere utile in fase di debug di programmi scritti in Assembly, per esempio.

Tuttavia, le cose possono essere un po’ più complicate di così: Molte istruzioni impiegano più di un ciclo di clock per essere interpretate ed eseguite. *Apparentemente* però spesso le cose vanno un po’ diversamente.

Prendiamo il seguente programma di prova (e senza utilità alcuna), che dopo una istruzione NOP carica nell’Accumulatore dapprima il valore \$FF, poi il valore \$00.

```
0000 NOP
0001 LDA #$FF
0003 LDA #$00
```

0: EA A9 FF A9 00 è la sua trasposizione in esadecimale. Ci si aspetterebbe un eguale numero di cicli di clock impiegati dalle due istruzioni LDA. E invece...

0000000000000000	11101010	\$0000	\$EA	r	NOP	11	}	3 Cicli
0000000000000001	10101001	\$0001	\$A9	r	A9	11		
0000000000000001	10101001	\$0001	\$A9	r	LDA	11	}	2 Cicli
0000000000000010	11111111	\$0002	\$FF	r	FF	11		
0000000000000011	10101001	\$0003	\$A9	r	LDA	11	}	2 Cicli
0000000000000100	00000000	\$0004	\$00	r	00	11		

Si vede chiaramente che la prima istruzione impiega tre cicli di clock per essere completata, la seconda solo due. Quanto sopra è perfettamente normale ed è una caratteristica prevista dal design del 6502.

In buona sostanza, e senza scendere troppo in tecnicismi, *in certe circostanze* il 6502 è in grado di leggere l’Opcode successivo mentre sta ancora terminando di eseguire l’istruzione corrente. Si tratta dunque di una specie di “sovrapposizione” (overlap) progettato appositamente per ottimizzare l’accesso ai BUS e minimizzare i tempi di inattività.



Sarà possibile passare alla modalità INTERACTIVE solo dalla modalità FREE RUN, per farlo è sufficiente premere il tasto **TAB**.

Comparirà quanto segue:

```
NOW: Interactive Mode   ~~~ for programs
```

Da questo momento ogni carattere digitato sulla tastiera viene inviato all'Apple-1 che lo interpreta come se fosse stato digitato sulla sua tastiera.

Questa caratteristica è utile anche per provare i propri programmi caricandoli direttamente da un computer esterno, si veda tale proposito l'Appendice 3.

L'output dell'Apple-1 ad eventuali comandi avviene sempre sul monitor ad esso collegato.

Sono disponibili due brevi programmi di test. Anch'essi vengono trasmessi maniera automatica come se venissero digitati dalla tastiera.

Per accedere al menù di caricamento premere per tre volte di seguito il tasto tilde ~, comparirà quanto segue:

```
Please choose one:

1  S. Wozniak ASCII Characters test
   - run with OR

2  M. Willegal RAM test
   - set limits in zero page (i.e. 0: 0 E0 0 F0)
   - run with 280R and wait for PASS/error
```

Sono presenti anche delle brevi istruzioni per l'utilizzo dei programmi, che sono:

- Test dei caratteri, incluso nel Manuale Operativo dell'Apple-1
- Test della RAM, di Mike Willegal

Scegliere poi il programma desiderato mediante i tasti 1 o 2 e attendere la fine del caricamento. Eseguirli poi secondo le istruzioni.

Per tornare alla modalità Comandi (FREE RUN), premere nuovamente il tasto **TAB**:

```
NOW: Stepper Command Mode
```

## 7. ESEMPIO DI BOOT DELL'APPLE-1 DALLA PRESSIONE DEL TASTO RESET

```
COM7 - PuTTY
-----
P-LAB APPLE-1-STEPPER
-----

PAUSE/BREAK   Reset 6502
ESCAPE        Clear Apple-1 screen

? to recall this menu anytime

S Single Step mode / Free Run mode
I Single CYCLE / INSTRUCTION mode (in Free Run mode only)
A MANUAL/AUTO Single Step (in Single Step mode only)
+/- modify AUTO speed when applicable
SPACE send Single CYCLE/INSTRUCTION signal

TAB Interactive mode ON/OFF (in Free Run mode only)

Single CYCLE mode selected
FREE RUN MODE SELECTED AT STARTUP
SINGLE STEP MODE SELECTED
1111111100101001 10101101 $FF29 $AD r LDA 11
1111111100101001 10101101 $FF29 $AD r AD 11
0000000111101110 11111111 $01EE $FF r FF 11
0000000111101101 00101001 $01ED $29 r 29 11
0000000111101100 01100001 $01EC $61 r 61 11
1111111111111100 00000000 $FFFC $00 r 00 11
1111111111111101 11111111 $FFFD $FF r FF 11
1111111100000000 11011000 $FF00 $D8 r CLD 11
1111111100000001 01011000 $FF01 $58 r 58 11
1111111100000001 01011000 $FF01 $58 r CLI 11
1111111100000010 10100000 $FF02 $A0 r A0 11
1111111100000010 10100000 $FF02 $A0 r LDY 11
1111111100000011 01111111 $FF03 $7F r 7F 11
1111111100000100 10001100 $FF04 $8C r STY 11
1111111100000101 00010010 $FF05 $12 r 12 11
1111111100000110 11010000 $FF06 $D0 r D0 11
1101000000010010 01111111 $D012 $7F W 7F 11
1111111100000111 10101001 $FF07 $A9 r LDA 11
1111111100001000 10100111 $FF08 $A7 r A7 11
1111111100001001 10001101 $FF09 $8D r STA 11
1111111100001010 00010001 $FF0A $11 r 11 11
1111111100001011 11010000 $FF0B $D0 r D0 11
1101000000010001 10100111 $D011 $A7 W A7 11
1111111100001100 10001101 $FF0C $8D r STA 11
1111111100001101 00010011 $FF0D $13 r 13 11
1111111100001110 11010000 $FF0E $D0 r D0 11
```

RESET VECTOR: \$FF00

BOOT STARTS AT \$FF00

## 8. CREDITI / RINGRAZIAMENTI

---

Questo progetto è stato ispirato dal lavoro e dai video di Ben Eater...

<https://eater.net>

...e da Erturk Kocalar di 8bitforce:

<https://www.8bitforce.com>

E inoltre:

- Steve Wozniak per tutto il suo lavoro, incluso il test caratteri.
- Mike Willegal per il test della RAM  
<https://www.willegal.net/appleii/6502mem.htm>

Non da ultimo, per il suo supporto e la sua collaborazione a distanza:

- Massimiliano Larocca

Bibliografia:

Apple-1 Operation Manual

<https://www.applefritter.com/files/a1man.pdf>

Western Design Center

[https://www.westerndesigncenter.com/wdc/AN-002\\_W65C02S\\_Replacements.php](https://www.westerndesigncenter.com/wdc/AN-002_W65C02S_Replacements.php)

---

Ci auguriamo che tu possa divertirti ad utilizzare *Apple-1 -Stepper* !

**APPLE-1-STEPPER**

INFO [p-l4b@protonmail.com](mailto:p-l4b@protonmail.com)

**P-L4B @ PROTONMAIL.COM**

## APPENDICE 1 – NMOS 6502 vs CMOS 65C02

---

Quando il microprocessore 6502 fu ideato e progettato, le memorie di tipo ROM disponibili sul mercato erano molto più lente della velocità di lettura che il microprocessore poteva raggiungere. Per far sì che il 6502 potesse leggere anch'esse venne ideata la linea RDY (Ready).

Se essa viene portata al livello logico '0' in certe circostanze il microprocessore sospende ogni attività lasciando BUS e linee di controllo nello stato in cui si trovano, per consentire alla memoria "lenta" di completare le sue operazioni.

Al momento opportuno, la linea RDY viene riportata a livello logico '1' e le attività riprendono da dove si erano fermate.

Semplificando, possiamo dire che la linea RDY si comporta come un semaforo, ma solo per il microprocessore. Il resto del traffico (il refresh dei registri del 6502 stesso, della RAM, la generazione dei segnali video, etc.) continua indisturbato.

Ci sono tuttavia circostanze in cui il semaforo non viene tenuto in considerazione: è il caso delle operazioni di scrittura in memoria.

Se la linea RDY viene attivata quando il microprocessore sta scrivendo qualcosa in una certa area di memoria, l'operazione di scrittura verrà completata, e *le operazioni verranno sospese alla prima operazione di lettura che si incontrerà.*

Sostanzialmente le operazioni di scrittura verranno eseguite sempre e comunque e mai fermate o ritardate. Esse verranno eseguite a piena velocità, perché l'NMOS 6502 è stato progettato per funzionare proprio così.

Facciamo un esempio pratico.

Il seguente programma di esempio in Assembly carica (LDA) nell'Accumulatore del 6502 il valore \$FF, lo deposita (STA) poi nella locazione \$0300, infine effettua un salto incondizionato (JMP) all'inizio del programma (locazione \$0000), che quindi ricomincia da capo.

```
0000  A9 FF      LDA #$FF
0002  8D 00 03   STA $0300
0005  4C 00 00   JMP $0000
```

Nella memoria del nostro Apple-1 avremo dunque il seguente contenuto:

INDIRIZZO	DATO
0000	A9
0001	FF
0002	8D
0003	00
0004	03
0005	4C
0006	00
0007	00

Quando il programma viene eseguito, il microprocessore effettua *elettricamente* le seguenti operazioni sui BUS e sulla linea di controllo R/W:

INDIRIZZO	R/W	DATO
0000	R	A9
0001	R	FF
0002	R	8D
0003	R	00
0004	R	03
<b>0300</b>	<b>W</b>	<b>FF</b>
0005	R	4C
0006	R	00
0007	R	00

Come si può vedere, sono tutte operazioni di lettura (Read) perché di fatto il microprocessore sta leggendo il programma, tranne una riga contrassegnata in grassetto con W (Write).

Questa è l'operazione con la quale il microprocessore dà seguito all'istruzione precedente ossia STA \$0300: scrivi il contenuto dell'Accumulatore (\$FF in questo caso) nell'indirizzo di memoria \$0300.

Alla luce di quanto detto, in quali "passi" del nostro programma la manipolazione della linea RDY avrà dunque effetto, fermando l'esecuzione del programma?

INDIRIZZO	R/W	DATO	STOP
0000	R	A9	SI
0001	R	FF	SI
0002	R	8D	SI
0003	R	00	SI
0004	R	03	SI
<b>0300</b>	<b>W</b>	<b>FF</b>	<b>NO</b>
0005	R	4C	SI
0006	R	00	SI
0007	R	00	SI

Il fatto che l'operazione di Write venga eseguita a piena velocità di clock impedisce ad Arduino di identificarla con precisione e, soprattutto, di leggere le 16 linee di indirizzo più le 8 di dati in tempo utile prima che esse cambino: esso è troppo lento. Altre soluzioni basate su diverse piattaforme di sviluppo potrebbero funzionare, ma non sono state prese in considerazione in questo progetto.

Con la sola manipolazione della linea RDY (senza toccare il Clock principale perché avrebbe effetti deleteri) risulta dunque estremamente complicato catturare gli eventi di Write direttamente dal BUS a piena velocità.

Possiamo quindi dire che **le operazioni di scrittura risulteranno invisibili all'Apple-1-Stepper.**

Ciò detto, il fatto di non "vedere" i BUS Dati e Indirizzi che onorano la fase di scrittura toglie poco valore all'utilità complessiva dell'oggetto, soprattutto in ottica di debug di programmi o di ricerca di guasti.

Il 65C02 si comporta in modo diverso dal suo predecessore: vediamo come.

Il microprocessore 65C02 è la versione CMOS del NMOS 6502 ed è "quasi" pin compatibile con esso, tanto che è possibile adattarlo con facilità per funzionare sui computer Apple-1 (Vedi Appendice 2).

Senza entrare nei dettagli delle differenze tra i due, ampiamente trattate da fonti autorevoli, la cosa che ci interessa è che **il 65C02 tiene in considerazione la linea RDY anche durante i cicli di scrittura.**

Nel caso dell'esempio precedente, dunque, sarà possibile fermare l'esecuzione del programma in qualunque punto e visualizzare correttamente anche la scrittura "fisica" all'indirizzo \$0300 da parte della istruzione STA (e di ogni altra istruzione che scrive qualcosa da qualche parte, naturalmente).

INDIRIZZO	R/W	DATO	STOP
0000	R	A9	SI
0001	R	FF	SI
0002	R	8D	SI
0003	R	00	SI
0004	R	03	SI
<b>0300</b>	<b>W</b>	<b>FF</b>	<b>SI</b>
0005	R	4C	SI
0006	R	00	SI
0007	R	00	SI

Tutte le scritture in memoria sono dunque pienamente visualizzabili.

## APPENDICE 2 – SOSTITUIRE NMOS 6502 CON CMOS 65C02

---

La sostituzione del NMOS 6502 con il CMOS 65C02 nell'architettura Apple-1 è trattata dal documento:  
[https://www.westerndesigncenter.com/wdc/AN-002\\_W65C02S\\_Replacements.php](https://www.westerndesigncenter.com/wdc/AN-002_W65C02S_Replacements.php)

di cui vengono riportati i passaggi salienti tradotti dall'originale:

- 1) VPB (Pin 1) – piegare verso l'esterno questo piedino in modo che risulti fuori dallo zoccolo.
- 2) MLB (Pin 5) - piegare verso l'esterno questo piedino in modo che risulti fuori dallo zoccolo. L'Apple-1 identifica il pin5 come VMA (Valid Memory Address) per compatibilità con il microprocessore 6800. Sia 6502 che W65C02 hanno sempre il segnale VMA valido. Apple-1 ha un jumper che dovrebbe essere sempre collegato quando si usa 6502 o W65C02S.
- 3) BE (Pin 36) – Collegare questo pin a VDD (Pin 8).
- 4) Apple-1 ha una resistenza di pullup da 3K per tenere a livello 1 la linea RDY (Pin 2). WDC non raccomanda l'uso delle istruzioni STOP o WAIT sull'Apple-1.

## APPENDICE 3. CARICAMENTO DI PROGRAMMI TRAMITE TERMINALE USB (Linux)

---

È possibile trasferire programmi mediante la modalità INTERACTIVE già descritta.

Per fare ciò:

- Lasciare attivo il programma terminale (es. minicom) come di consueto,
- Aprire un'altra shell e con privilegi di amministratore eseguire il seguente script (supponiamo l'abbiate chiamato `loader.sh`) seguito dal nome del file che desiderate caricare sull'Apple-1. Ad es.:  

```
#sudo loader.sh demo.txt
```
- Attendere la fine del caricamento.

Lo script sottoriportato legge il file di input carattere per carattere e lo invia all'Arduino MEGA che a sua volta lo invia all'Apple-1.

Le pause presenti sono state determinate sperimentalmente per venire incontro alle limitate capacità di input del computer.

Il *device file* relativo ad Arduino potrebbe essere diverso, adattatelo alla vostra situazione.

```
#!/bin/bash
#Send file to device character by character. Optimized for Apple-1
DEVICE="/dev/ttyACM0"
stty -F $DEVICE 115200 cs8 -cstopb -parenb
INPUT=$1
while IFS= read -r -n1 char
do
    if [ "$char" = "" ];
    then
        echo -n -e '\x8d' > $DEVICE
        sleep 0.5
        echo
    fi
    echo -n "$char" > $DEVICE
    echo -n "$char"
    sleep 0.06
done < "$INPUT"
exit 0
```