Protopad Developer Guide

Version 2.2 · Date 2023-06-18

Contents

1. Introduction

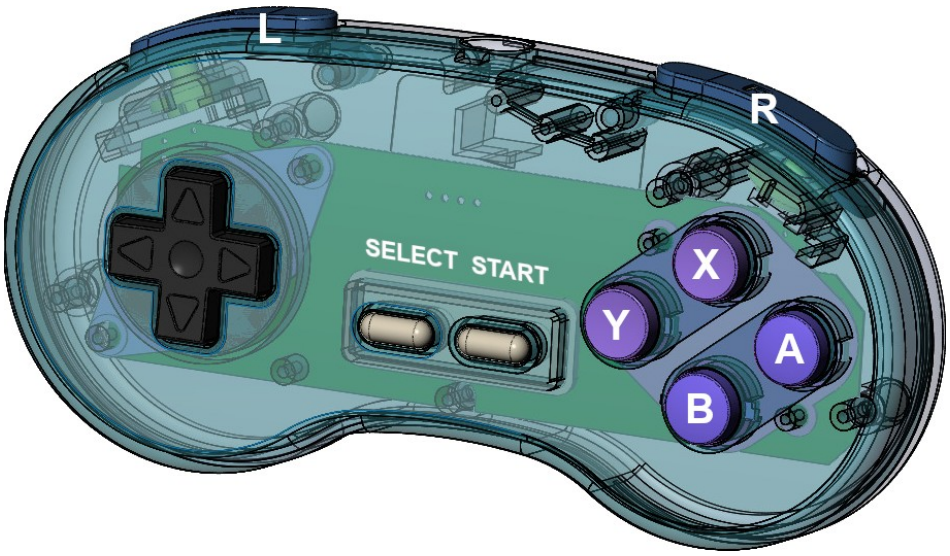
Protopad is a game controller with a total of 12 digital control buttons. It has a female DE-9 connector and is designed to be used with a multitude of systems, like the Commodore 64, the Commodore Amiga, the Atari XL, or the Amstrad CPC.

Protopad has different operation modes for different purposes and target platforms.

This documentation will explore the operation modes relevant to the Commodore 64/128/SX64/64GS platform family. Other operation modes are still in development and thus not part of this documentation yet.

2. Button Layout

Protopad's button layout resembles that of Nintendo's Super NES standard controller.



On the left, there are 4 directional buttons combined into one cross-shaped button called the *directional pad* or the *D-pad*.

On the right, there are two pairs of action buttons, labelled A, B and X, Y.

The two shoulder buttons on the top are called L and R.

The SELECT and START buttons are located in the center.

3. Operation Modes

Protopad comes with two operation modes that are C64 compatible: the *General Compatibility Mode* which supports up to 7 buttons, and the much more powerful *Native Mode* which supports all 12 buttons.

3.1. General Compatibility Mode

By default, Protopad operates in *General Compatibility Mode*.

In this mode, it simply mimics the behavior of a generic Atari-compatible joystick with four directional buttons and one action button.

This enables Protopad to be used with any existing C64 game without modification. It serves as a direct replacement for Atari-compatible joysticks like the Competition Pro or the Quickshot.

Protopad also provides two additional action buttons that are not part of the official Atari joystick port specification.

This enables Protopad to be used with existing C64 games supporting multi-button joysticks like the Cheetah Annihilator.

3.1.1 Reading the Button States

3.1.1.1 Reading the Atari-Compatible Buttons

The states of the directional buttons and the primary action button are delivered to the five

digital data lines of the C64's control port.

The digital data lines of control port 2 are handled by CIA 1 port A.
The digital data lines of control port 1 are handled by CIA 1 port B.

Button	*C64 IO Register*	*Role*	*Description*
UP	CIA 1 Port A/B ($DC00/$DC01) Bit#0	input	0 = pressed
DOWN	CIA 1 Port A/B ($DC00/$DC01) Bit#1	input	0 = pressed
LEFT	CIA 1 Port A/B ($DC00/$DC01) Bit#2	input	0 = pressed
RIGHT	CIA 1 Port A/B ($DC00/$DC01) Bit#3	input	0 = pressed
ACTION (primary)	CIA 1 Port A/B ($DC00/$DC01) Bit#4	input	0 = pressed

The following code example reads the button states from control port 2:

```
        ; set all bits of CIA 1 port A to input

        lda #%000 00000
        sta $dc02

        ; read buttons from CIA 1 port A

        lda $dc00
        and #%000 11111
```

The following code example reads the button states from control port 1:

```
        ; set all bits of CIA 1 port B to input

        lda #%000 00000
        sta $dc03

        ; read buttons from CIA 1 port B

        lda $dc01
        and #%000 11111
```

3.1.1.2 Reading the Additional Action Buttons

The states of the secondary and tertiary action buttons are delivered to the analogue POTX and POTY lines of the C64's control port, respectively.

The analogue POTX and POTY lines are internally wired to the SID's A/D converters.

Button	*C64 IO Register*	*Role*	*Description*
ACTION (secondary)	SID POTX ($D419)	input	< $FF = pressed
ACTION (tertiary	SID POTY ($D41A)	input	< $FF = pressed

Two things must be taken into account prior to reading the additional action buttons:

POTX and POTY only exist once and they are shared by both control ports. Thus, the control port of interest must be selected programmatically.

Bits 6 and 7 of CIA 1 port A are used to select the control port for POTX and POTY.

Control Port	*C64 IO Register*	*Role*	*Description*
Control Port 2	CIA 1 Port A ($DC00) Bit#6	output	1 = selected
Control Port 1	CIA 1 Port A ($DC00) Bit#7	output	1 = selected

Also, POTX and POTY are updated by the SID every 512 clock cycles, so after selecting the control port, one full update has to be awaited in order to get correct values.

The following code example will read the additional buttons from control port 2 of the C64:

```
; set bits 6 and 7 of CIA 1 port A to output
; to enable control port selection

lda #%11000000
sta $dc02

; set bit 6 and reset bit 7 of CIA 1 port A
; to select control port 2 for POTX/POTY

lda #%01000000
sta $dc00

; after 2 x 512 cycles, the values are stable

lda $d419 ; read secondary action button from POTX
lda $d41a ; read tertiary action button from POTY
```

The following code example will read the additional buttons from control port 1 of the C64:

```
; set bits 6 and 7 of CIA 1 port A to output
; to enable control port selection

lda #%11000000
sta $dc02

; reset bit 6 and set bit 7 of CIA 1 port A
; to select control port 1 for POTX/POTY

lda #%10000000
sta $dc00

; after 2 x 512 cycles, the values are stable

lda $d419 ; read secondary action button from POTX
lda $d41a ; read tertiary action button from POTY
```

3.2. Native Mode

Protopad's key feature is the powerful *Native Mode* which grants full control over all 12 buttons to the programmer. It is therefore ideal for newly developed games that intend to support Protopad.

Games with Protopad support can check for the presence of a Protopad at runtime and provide a conventional Atari-style control scheme as a fallback.

3.2.1 Communicating with the Protopad

In order to use Protopad's *Native Mode*, the C64 needs to implement a specific communication protocol.

The communication protocol uses the five digital data lines of the C64's control port.

Two lines are used as output from the C64 to the Protopad and three lines are used both as output and input, depending on the current protocol phase.

The digital data lines of control port 2 are handled by CIA 1 port A.
The digital data lines of control port 1 are handled by CIA 1 port B.

Function	*C64 IO Register*	*Role*
Init / Data 0	CIA 1 Port A/B ($DC00/$DC01) Bit#0	output / input
Init / Data 1	CIA 1 Port A/B ($DC00/$DC01) Bit#1	output / input
Init / Data 2	CIA 1 Port A/B ($DC00/$DC01) Bit#2	output / input
Init / Clock	CIA 1 Port A/B ($DC00/$DC01) Bit#3	output
Init / Keep	CIA 1 Port A/B ($DC00/$DC01) Bit#4	output

Using the protocol, the C64 can attempt to initiate a *Native Mode* transaction on the C64's control port.

If a Protopad is present at the control port, it will acknowledge the attempt and await further instructions from the C64. The C64 can then request and receive the button states.

3.2.1.1 Initiating a Native Mode Transaction

To initiate a *Native Mode* transaction, the C64 sets port bits 0-4 to output and its values to zero. Then it sets port bits 0-2 back to input and checks their values.

A Protopad present at the control port would recognize all five port bits to be zero, latch its button states and keep port bits 0-2 zero as a handshake.

If the C64 detects one or more of port bits 0-2 to be non-zero, no Protopad is available. The C64 should terminate the *Native Mode* transaction and perform conventional Atari-style joystick handling instead.

If the C64 detects port bits 0-2 to be zero, a Protopad is available and the *Native Mode* transaction has been initiated successfully.

3.2.1.2 Reading the Button States

Once a *Native Mode* transaction is initiated, it is kept alive by the C64 via keeping port bit 4 (Keep) zero.

The C64 can then use port bit 3 (Clock) to request the button states from the Protopad. Every time port bit 3 is toggled, Protopad delivers the next three button states to port bits 0-2 (Data) from where the C64 can read them.

	Keep	*Clock*	*Data 2*	*Data 1*	*Data 0*
State after Initiation	0	0	0	0	0
Iteration 1	0	1	B	A	➡
Iteration 2	0	0	⬅	⬇	⬆
Iteration 3	0	1	START	SELECT	R
Iteration 4	0	0	L	Y	X

After four iterations, the button states of all 12 buttons have been delivered and the C64 should terminate the *Native Mode* transaction.

3.2.1.3 Terminating the Native Mode Transaction

If no Protopad was present at the control port or all button states have been read, the C64 sets port bits 0-4 back to input which terminates the *Native Mode* transaction and sets Protopad back to *General Compatibility Mode*.

3.2.1.4 Hardware Limitations

Due to weak CIA port drivers, bad signal quality, bus noise and damping, a delay of 10 µs should be considered between two port writes or between a port write and a port read in order to get stable and valid results.

3.2.2 Example Code

3.2.2.1 Naive Implementation For Demonstrational Purposes

The following code example attempts a *Native Mode* transaction to retrieve all 12 buttons of a Protopad in control port 2. The code honors the 10 µs delay recommendation and is not optimized at all.

```
;======================================================
; Protopad Example Code
; Naive Implementation For Demonstrational Purposes
; Written by Chester Kollschen
; Copyright © 2023 by Knights of Bytes
; Version 1.0.1
;======================================================

;------------------------------------------------------
; Results
;------------------------------------------------------

buttons1: .byte $00 ; bits 7-0: 00BA→←↓↑
buttons2: .byte $00 ; bits 7-0: 00TERLYX

;------------------------------------------------------
; Read
;
; Pre-Condition: CIA 1 port A bits 4-0: input
;------------------------------------------------------

read:

        ; initiate native transaction on Protopad in control port 2

        lda #%000 00000
```

```
        sta $dc00 ; CIA 1 port A bits 4-0: 00000
        lda #%00011111
        sta $dc02 ; CIA 1 port A bits 4-0: output

        nop
        nop
        nop

        lda #%00011000
        sta $dc02 ; CIA 1 port A bits 4-3: output, bits 2-0: input

        ; check for handshake from Protopad in control port 2

        nop
        nop
        nop

        lda $dc00
        and #%00000111
        bne fail ; no Protopad detected

        ; retrieve and store button states

        lda #%00001000
        sta $dc00 ; clock iteration 1

        nop
        nop
        nop

        lda $dc00 ; read 3 buttons
        and #%00000111
        asl
        asl
        asl
        sta buttons1

        lda #%00000000
        sta $dc00 ; clock iteration 2

        nop
        nop
        nop

        lda $dc00 ; read 3 buttons
        and #%00000111
        ora buttons1
        sta buttons1

        lda #%00001000
        sta $dc00 ; clock iteration 3

        nop
        nop
        nop

        lda $dc00 ; read 3 buttons
        and #%00000111
        asl
        asl
        asl
        sta buttons2
```

```
        lda #%000**00**000
        sta $dc00 ; clock iteration 4

        nop
        nop
        nop

        lda $dc00 ; read 3 buttons
        and #%00000**111**
        ora buttons2
        sta buttons2

        ; terminate native transaction on Protopad in control port 2

        lda #%000**00000**
        sta $dc02 ; CIA 1 port A bits 4-0: input

        rts

fail:

        ; terminate native transaction on Protopad in control port 2

        lda #%000**00000**
        sta $dc02 ; CIA 1 port A bits 4-0: input

        ; perform conventional Atari-style button handling

        lda $dc00

        [...]

        rts
```

3.2.2.2 Optimized Implementation For Speed

The following code example attempts a *Native Mode* transaction to retrieve all 12 buttons of a Protopad in control port 2. The code honors the 10 µs delay recommendation and is optimized for speed.

```
;=====================================================
; Protopad Examble Code
; Optimized Implementation For Speed
; Written by Chester Kollschen
; Copyright © 2023 by Knights of Bytes
; Version 1.0.1
;=====================================================

;-----------------------------------------------------
; Results
;-----------------------------------------------------

buttons1: .byte $00 ; bits 7-0: 01BA→←↓↑
buttons2: .byte $00 ; bits 7-0: 01TERLYX

;-----------------------------------------------------
; Read
;
; Pre-Condition: CIA 1 port A bits 7-0: input
;-----------------------------------------------------
```

```
read:

        ; preload X and Y to optimize transaction

        ldx #%00000000 ; port value to clock iterations 2 and 4
        ldy #%00001000 ; port value to clock iterations 1 and 3

        ; initiate native transaction on Protopad in control port 2

        stx $dc00 ; CIA 1 port A bits 7-0: 00000000
        lda #%11111111
        sta $dc02 ; CIA 1 port A bits 7-0: output
        nop
        nop
        lda #%11111000
        sta $dc02 ; CIA 1 port A bits 7-3: output, bits 2-0: input

        ; check for handshake from Protopad in control port 2

        nop
        nop
        nop
        lda $dc00
        sty $dc00 ; clock iteration 1
        and #%00000111
        bne fail ; no Protopad detected

        ; retrieve and store button states

        nop
        lda $dc00 ; read 3 buttons
        stx $dc00 ; clock iteration 2

        asl
        asl
        asl
        eor $dc00 ; read 3 buttons
        sty $dc00 ; clock iteration 3
        sta buttons1

        nop
        lda $dc00 ; read 3 buttons
        stx $dc00 ; clock iteration 4

        asl
        asl
        asl
        eor $dc00 ; read 3 buttons
        sta buttons2

        ; terminate native transaction on Protopad in control port 2

        stx $dc02 ; CIA 1 port A bits 7-0: input

        rts

fail:

        ; terminate native transaction on Protopad in control port 2

        stx $dc02 ; CIA 1 port A bits 7-0: input

        ; perform conventional Atari-style button handling
```

```
        lda $dc00
        [...]
        rts
```

3.2.3 Recommended Button Usage

While the purpose of each button is defined entirely by the application, we have some recommendations for you on what functions to assign to which buttons.

For example, it's a de-facto standard that games can be paused with the START button.

Please have a look at the following table for more examples.

Button	*Recommended Usage*
UP, DOWN, LEFT, RIGHT	Up, Down, Left, Right
A	CANCEL/BACK in menus special weapon in shooters special action in platformers
B	CONFIRM in menus fire in shooters jump in platformers
X	smart bombs in shooters call inventory/map
Y	accelerate in platformers dock/undock probes in shooters fire in platform shooters
L	change weapon/speed in shooters brake in racing games cycle items
R	change weapon/speed in shooters accelerate in racing games cycle items
SELECT	Enter menu, select menu item
START	Start game, Pause, Resume

4. Support

For further questions, feel free to write us an e-mail to: contact@protovision.games