Power Pak Expansion

The Program Pak port on the CoCo provides for almost limitless possibilities for expansion and interface to the real world. It was primarily used for the rom pak games, but Radio Shack also supported disk drives, RS-232 paks, Speech-Sound paks, graphics input pads, and a host of other devices with this port, not to mention all the third party devices. What follows here is a homebuilt interface that adds nearly 16K of rom, plus over 32 input/output lines.

History

Years ago before I had any multi-paks and disk drives, I did all my assembly language programming with cartridge based Edtasm. With all kinds of experiments wired up out the expansion port, it created a need to simultaneously access the cartridge rom and I/O at the same time. My first design was an expander board that contained 2 PIA's and a 40 pin connector to plug Edtasm into. This worked okay until I built my eprom programmer, as it was necessary to load in the programmer software from tape. This Power Pak solved this problem by allowing the programmer software plus the Edtasm rom to reside in the eprom. An added bonus was the compact size, as all these components fit in a game pak size case, with only a ribbon cable leading to the peripheral device.

This is the Power Pak shown with its hood off. A standard program pak cartridge was used for the case. This pak can use a 32K or 16K Eprom, although only 15 1/2 K will be accessed . Two 6821 Peripheral Interface Adapters are used for I /O.

Decoding

As shown in the schematic

Eprom CE (Chip Enable) to CTS, Eprom OE tied to CE
PIA 0 CS2 to SCS, CS1 to +5V, CS0 to A2
PIA1   CS2 to SCS, CS1 to +5V, CS0 to A3

This decoding will address the Eprom at 49152, PIA 0 at 65348-51, PIA 1 at 65352-65355

Parts List

1 Program Pak case
2 6821 PIA's
1 27128 or 27256 eprom
1 28 pin socket
2 40 pin sockets
3 .1 uf capacitors
24 inch 40 conductor ribbon cable

Pin Outs

A listing of Color Computer expansion port signals can be found here. Pinouts for the Eprom chips and the 6821 can be found at the Chip Directory
A simplified schematic is available, download schematic.zip . Note that the power supply decoupling capacitors are not shown on this schematic.

PC Board Artwork

The PC Board is double sided. I don't have the technology to plate through connecting holes, but these layouts allow you to simply solder a wire at the through hole locations. I have scanned in a photocopy of the original artwork. This artwork is missing the Pin 4 connector, add one in if you ever plan on using the interrupt input to the CPU. The artwork can be found here

Constructing the Pak

Start with obtaining a program pak case. Basically pick one of those cartridge games you never play and use it for a donor. Centered under the pak's label is a screw, remove the screw and then pry the pak apart. Remove the rom board. Cut out a piece of double sided PC board that completely fills the case. This is important to have a tight fit. You can use the old rom board as a template to layout the board where it engages the expansion connector. Again, you must be exact here, as a loose fitting, misaligned edge connector is something that you want to avoid. You will need to drill one hole to clear the standoff that fastens the case together.

Once you are satisfied the board fits the case and the expansion connector properly, the next critical step is to align the artwork with the edge connector. I simply install the cleaned up board into the CoCo several times, this will scratch the board where your connector pads should be. As a double check, they should match the old rom board pads also.

Apply your artwork, drill, etch in whatever order or method you use. Connect the through holes if not using a plated through board, install the sockets and other parts. Cut the ribbon cable to align with the pads at the outside of the pak, and route the cable out the back of your pak. Carefully check your board for any defects such as solder bridges etc. As this pak has the cartridge enable connection made, it will need an eprom with some code residing at 49152. Alternatively you can mount a small switch on the pak, routing the CoCo's pin 7 and 8 to it, which will allow you to enable or disable the jump to the pak on startup.

Programming the Eprom

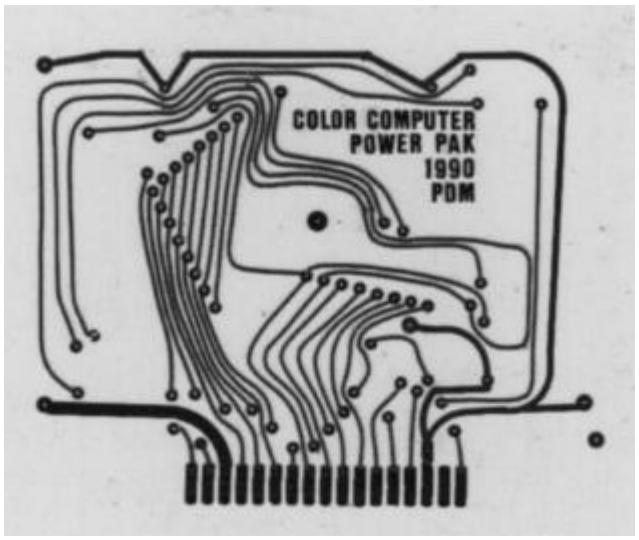
You can can control the power pak using either BASIC or Assembly language. Both types of programs can be located on the pak's eprom. For more info on how to program your own rom on a pak go here .

Power Pak PC Style

I have currently started building a PC equivalent to this. It is an ISA card that currently supports a pair of 8255 Interface adapters. I hear a lot of bad things about PC architecture, but it did impress me that I could pull these chips from a scrapped 1984 XT, and wire them up to my K6-2 500 Based PC with no apparent problems. This project may take off in different directions and is still on breadboard stage. Software is the biggest holdup as I am trying to keep it Windows friendly and I am just a rookie in the PC world. It looks like it may be possible to build a CoCo cartridge port emulator, but I do not want to promise anything at this time as this is basically a learning exercise to step up my knowledge of the PC world. Check here periodically for updates, will post something when it is presentable to the public.

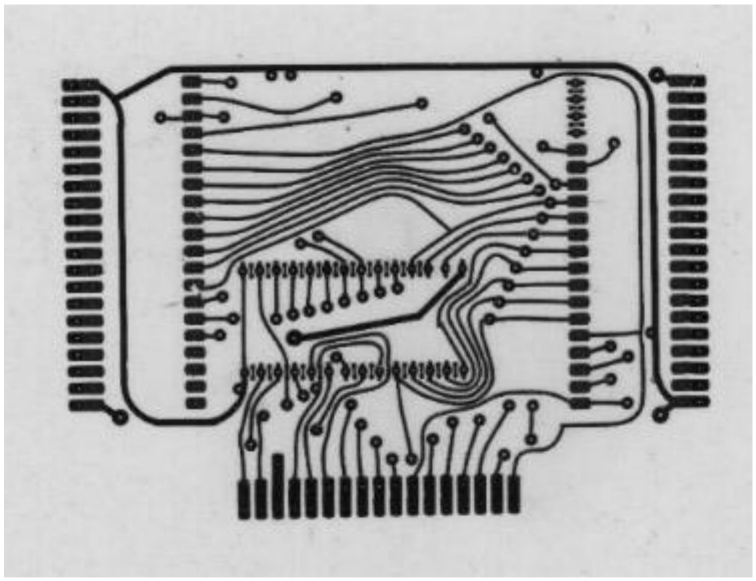
Last updated March 21, 2001

Index



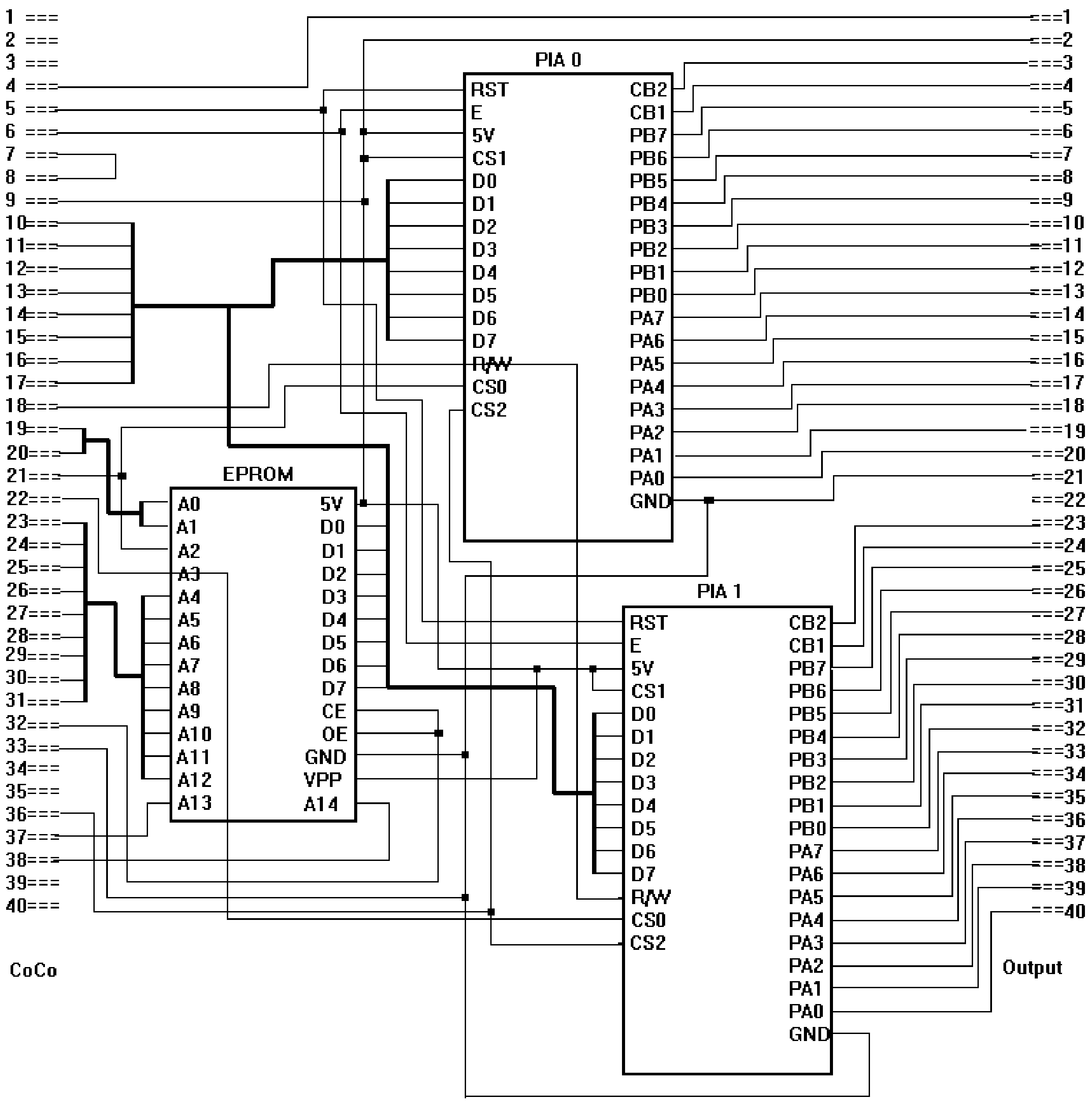
This is the top side trace

Pin 5 **Pin 37**



This is the bottom side trace

Pin 38 **Pin 6**



POWER PAK SCHEMATIC
SHOWN WITH 32K EPROM

Build your own CoCo Eprom Pak

The Color Computer has a large selection on software on rom, available in what RadioShack called a Program Pak. You can't beat software in rom, switch on till your application is ready to work is typically about 2 seconds. If you have a favorite cassette or disk based program, often it can be transferred to a Pak. Or you can write your own assembly language or Basic programs and have them programmed onto the Pak. If you are using a CoCo as a dedicated controller, this also saves the hassle of having to load in your control program every time you power it up.

Assembly language programs can be placed on the pak fairly easily, as long as they are less than
15 3/4 K, and are written in position independant code. Non position independant code programs can often be transferred from the pak to their original load address, and executed from there.

Basic programs require a little more work, as there is some setup required on top of just loading the program in from the pak. The finished results are fairly impressive though, even fairly large programs are automatically running in just seconds, from switch on, with no key input required. If you have a dedicated application for your CoCo, and want to do away with the loading of the software from disk or tape, keep reading for the complete procedure. Remember all addresses/data are shown in base 10 decimal.

Romming a Basic program in a Pak

This procedure has been tested with all 3 CoCo models, cassette or disk based.

1 : Determine the start and end locations of the Basic program you want to rom. Once you have
loaded in or typed in your program, four locations in memory hold the start and end
addresses of the Basic program in memory. The first 16 bit

address is at decimal 25, the second
(end) address is at at 27. Type in the following and enter it a line at a time, DO NOT assign line
numbers and run as a program, as this will change the values you are trying to read.

```
A=PEEK(25):B=(PEEK(26):S=A*256+B:PRINT A;B;S

C=PEEK(27):D=PEEK(28):E=C*256+D:PRINT C;D;E
```

Record values A,B,S,C,D,E

You will find the start address will vary depending on what PCLEAR was set at originally,
and will be different depending on if system is cassette or disk based. Note that the start
address that you read is actually one higher than the start of data. All Basic programs start with
a 0 before the text, but the value you read in the above procedure is the start of text. This is
important in the next step. Take the value you read in (S) and subtract 1 from it, for use in step 2.

2 : Save the tokenized Basic program. This is where it is best that you dust off the old cassette
deck and type in CSAVEM "name", load address (S-1), end address (E), 0
You could alternatively do this to your drive with a SAVEM command, but since the
program is destined to be run in a Program Pak, loading in from the cassette WITHOUT
the drive controller in place, allows a more accurate simulation of how the program will run
once it is auto-booted in from the Pak.

3 : Test the program for operation and conflicting commands before you have it programmed onto eprom. Start with the computer off, power it up. If you have a CoCo3, you have to enter one additional command,   POKE 65502,0 . This will simulate the Program Pak installed. DON'T do this Poke if you have a CoCo1 or 2. CLOADM or LOADM what you had previously saved. Poke in the start and end addresses from above as follows:

```
POKE 25, A
POKE 26, B
POKE 27, C
POKE 28, D
```

Complete initializing Basic by entering the following lines. Remember don't assign line numbers, just enter them one line at a time.

```
POKE 113,85
POKE 114,128
POKE 115,192
EXEC 47452
```

You should now be able to LIST, RUN, EDIT and perform all other Basic functions on the program that was loaded in as machine code. If you have a CoCo3, and your program includes any Super Extended commands, start your Basic program with a POKE 65503,0.

If everything looks good, your code is ready to be programmed onto the cartridge eprom.

The cartridge eprom will contain a small loader and auto-boot

program,
placed at the cartridge start address. You need to add the load address of the Basic program (S-1)
from Step 2 above into Line 200. Source code for this loader and auto start program is as follows:

```
    050  BASDES     EQU       BASLOD - LOADER
    100  LOADER    LDA       #85                          SET WARM
RESET
    110                STA       113
    120                LDD       #32960                     SET EXTENDED
BASIC RESET VECTOR
    130                STD       114
    140                JSR       47452                      SET UP
PARAMETERS FOR BASIC
    150                LDA       #53                        RESTORE
INTERRUPTS
    160                STA       65283                      THAT ARE
    170                LDA       #52                        DISABLED ON
    180                STA       65315                      CARTRIDGE
AUTO START
    190  BASIN       LDX       #BASDES + 49152
    200                LDY       #address                   THIS IS LOAD
ADDRESS OF BASIC (S-1)
    210                STY       25                         SAVE IT IN
BASIC START
    220                INC       26                         CORRECT THE
ACTUAL START VALUE
    230  TNSFER     CLRB                                 SET END
COUNTER TO  ZERO
    240   TNSFR2    LDA       ,X+                          GET FIRST BYTE
FROM ROMPAK
    250                STA       ,Y+                          TRANSFER BYTE
TO BASIC RAM
```

```
      260               BNE      TNSFR2               NON ZERO
DATA, KEEP TRANSFERRING
      270  ENDCHK  INCB                              ZERO DATA
DETECTED, INCREMENT "0" COUNTER
      280               CMPB    #4                    IS THERE 4
CONSECUTIVE ZERO'S?
      290               BEQ      LODDON               IF YES, STOP
TRANSFER
      300               LDA      ,X+                  LOAD NEXT
ROMPAK BYTE AFTER ZERO
      310               STA      ,Y+                  TRANSFER BYTE
TO BASIC RAM
      320               BNE      TNSFER               NON ZERO DATA,
RETURN TO MAIN LOOP
      330               BRA      ENDCHK              ZERO DATA,
INCREMENT COUNTER, STAY IN ZERO LOOP
      340  LODDON  LEAY     -1,Y     ;              CORRECT BASIC
END ADDRESS
      350               STY      27                   SAVE END
ADDRESS FOR BASIC
      360 AUTRUN   LDX      #733                  BASIC LINE
INPUT BUFFER
      370               LDD      #21077                LOAD LETTERS
"RU"
      380               STD      ,X++
      390               LDD      #19968                LOAD "N" AND
END
      400               STD      ,X++
      410               LDB      #4                    INDICATE 4
CHARACTERS
      420               CLRA
      430               STA      112                   SET CONSOLE
IN BUFFER FLAG
      440               LDX      #732                  POINT TO LINE
```

```
INPUT BUFFER
     450              JMP        44159              START MAIN
BASIC LOOP
     460  BASLOD    FCB        255                BASIC PROGRAM
DATA STARTS HERE
     470              END
```

There is 1 absolute address in the above listing that can be changed depending on your
system and the program memory requirements. Line 200, is the load point in ram, common points are 7680, 9728. This can be changed, especially if you will not be running a disk drive. I currently use 7680 as this is the Extended Basic default

Assemble the boot loader and burn on to the eprom at address 0 for 8 and 16K devices. If your rom board is standard and you are using a 32K device, usually 16384 is the correct address to locate the boot loader. (This is due to to the high address line A14 on the rom board tied to 5V) Some paks are tied low, you can write the loader to 0. Burn the Basic program data immediately following the loader . (Overwrite the 255 data byte at BASLOD) The first byte at BASLOD should be a zero, your end of Basic data should be 4 consecutive zeros.

What happens when your CoCo starts with the cartridge in place, is as follows:

The boot loader starts, and moves the Basic in from cartridge rom into ram.
Basic is initialized to recognize this non standard load.
RUN is entered automatically to start the Basic program.

More details on the above items, pulled directly from Extended Basic Unravelled Disassembly

Decimal conversions are in black

0019 25 TXTTAB RMB 2 *PV BEGINNING OF BASIC PROGRAM
001B 27 VARTAB RMB 2 *PV START OF VARIABLES
0071 113 RSTFLG RMB 1 *PV WARM START FLAG: $55 = WARM, OTHER = COLD
0072 114 RSTVEC RMB 2 *PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
B95C 47452 JSR LA35F SET UP TAB WIDTH, ZONE ETC. This routine sets up some I/O parameters for Basic's use

Assembling your own Eprom Pak

Start with a standard Program Pak game that uses a 28 pin DIP chip. Most older cartridges use a 24 pin rom, and some newer paks have no DIP, just a blob on the board. If you can count 14 pins when you slide the protective door back on the Pak, you have a good candidate. On this donor cartridge, you will be destroying the rom chip when you remove it, so take a copy of it before you start if you ever want to play it again. My eprom programmer also has a CoCo cartridge slot that allows me to make a direct copy into a new eprom, before I remove the rom. This rom also makes a good test once you are done assembling your Pak, just plug the eprom into your Eprom Pak and see if the game plays normally.

Open the cartridge, there is a screw hidden under the label. Verify that it has a 28 pin DIP chip before proceeding. If all looks good, remove the chip, taking care not to damage the board. Best method if you do not have desoldering tooling is to cut all the pins, remove the chip, then desolder the pins individually. Boards I have seen will accept 8, 16, or 32K devices, although the maximum addressable in unmodified form is about 15 3/4 K.

Solder in a 28 pin socket, be sure you marked pin one of the original rom, so you can orient the new device correctly. Optionally, you can disconnect connector traces 7 and 8, and install a small switch to control this connection. This switch will allow auto booting, or manual starting of your program, dependant on switch position. Carefully check for solder bridges or other defects before plugging in your Pak. Install your programmed eprom and assemble the Pak.

This page is looking for a new home, as I will be taking my CoCo site down soon. It desperately needs a rewrite as it has lost some of its formatting. If interested, follow the Index link below to my email address to contact me.

This page last revised Dec 28, 2003