Introducing Your

COMMANDER X16

Personal Computer

CONTENTS

1.0 X16 Motherboard and Case Setup

Carefully remove the X16 mainboard from any packaging padding.

Remember to ground yourself or touch metal (keys or paperclip) before handling electronic equipment.

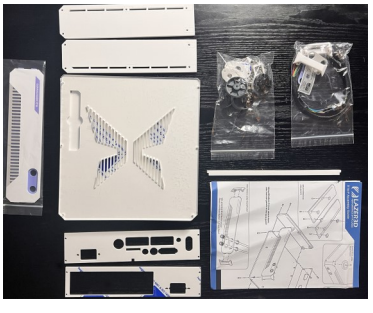
IF YOU DON'T YET HAVE A CASE:
The X16 board can safely sit atop the thin padding that it came with. Avoid "flexing" or twisting the board (keep it flat).

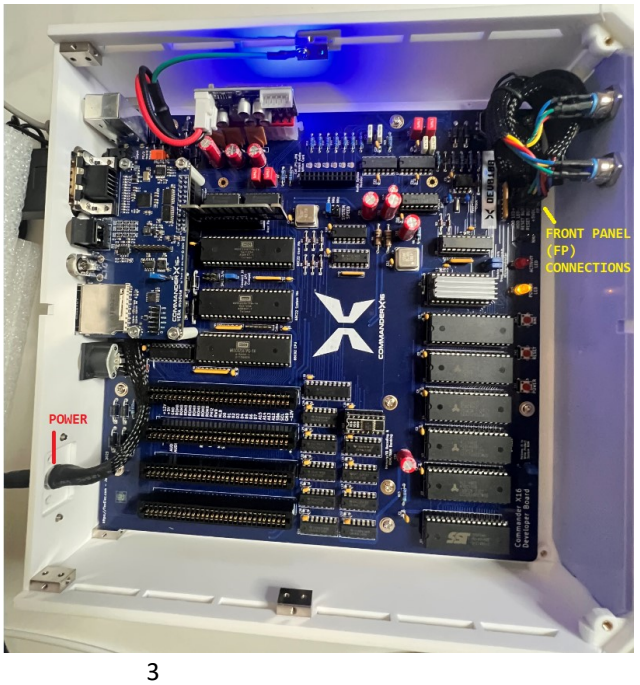
IF YOU NEED A CASE:
Seek an "ATX" case. The ATX standard for computer cases has been around since 1995. The standard mostly helps establish the spots for "stand-off" screws to mount a motherboard to a metal chassis, along with standard holes for mounting an "ATX-style" power supply unit (PSU). ATX cases are plentiful to find (new or used) and are available in "desktop" (lower profile) or "tower" (upright) styles.

Some ATX cases have built-in stand-offs.

Most new-cases will come with a set of stand-off screws. But if not, "computer screw assortment kits" can be found online. The rubber washer sits below the screw, to help make it easier to remove those screws in the future. The X16 motherboard has 9x stand-off holes.

IF YOU HAVE THE OFFICIAL CASE:
The Lazer3D case comes pre-fitted with the 9x stand-off screws, but does not include the rubber washers (they aren't really necessary). The assembly of the case takes about 20-40 minutes and requires only a regular sized Philips head screwdriver.





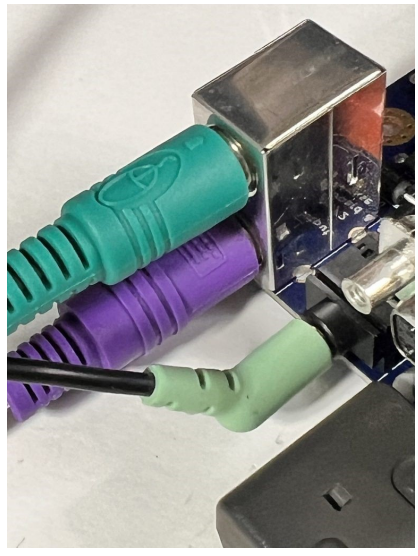
The picoPSU can also be used to power many 8" to 13" flat panel screens using a 12V adapter.

1.1 Keyboard and Mouse Connections

The X16 uses PS/2 style mouse and keyboard connections. These type of connections are simpler and have less overhead to implement than USB. And, arguably, PS/2 connections are more reliable (particularly at very early stages of the system startup process) and more secure (also offering more equalized performance in tournaments) than USB.

Because PS/2 style connectors are relatively easy to make, they remain widely supported worldwide, making them very affordable.

The PS/2 Keyboard connector is typically identified with a purple connector end, with an embedded keyboard-icon identifying the "top" position. The PS/2 mouse connector typically has a green connector end, with a mouse-icon. These connectors are at the rear -right corner of the mainboard.



PS/2 style connections were introduced by IBM in 1987, replacing the aging DIN style.

DIN to PS/2 adapters are supported. Use the 6" ones since the shorter adapters block the mouse connector.

A full-sized PS/2 keyboard can also be used!

1.2 Audio Connection

For audio output, the X16 includes a two-ring "stereo jack" located next to the keyboard and mouse connectors. This jack can use headphones or amplified stereo speakers. Some speakers or stereo equipment may use the red/white RCA style inputs. In that case, the stereo-output cable of the X16 can be adapted to a Y-splitter (or "stereo to dual mono" adapters).

There are countless desktop stereo speaker options, just avoid the ones with USB connections. More expensive speaker options may include a sub-woofer. 4Watts power output is sufficient for a small room.

Optional Cables (may be necessary if using a stereo receiver)

CA-2014RB (non-USB)
4W Peak Output

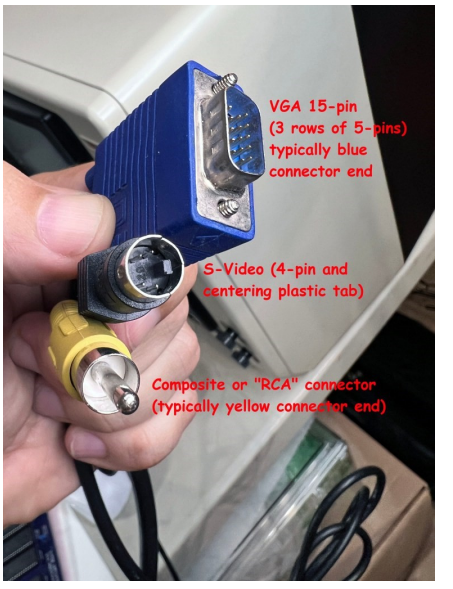
1.3 Video Connections

SHIFT+SCROLL LOCK is used to cycle between video output.

The X16 supports three video output options:

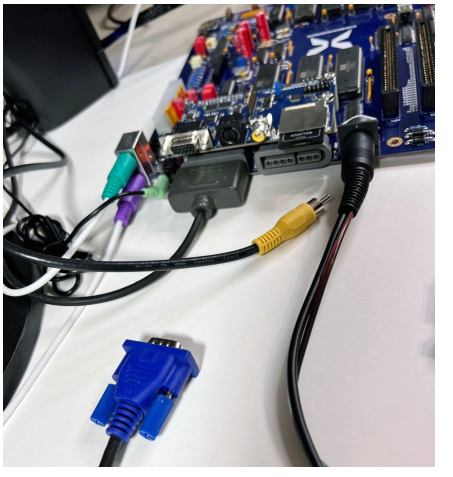
- VGA (DB15)
- S-Video
- Composite (RCA)

While all three connectors can be connected at the same time, the system will only have one active video output at a time.



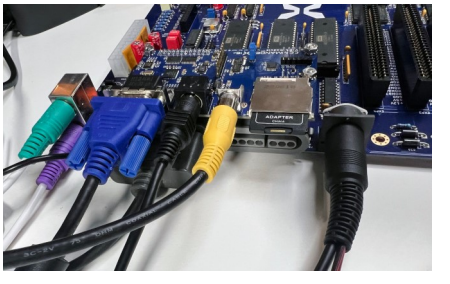
VGA was introduced by IBM in 1987.

In general, VGA provides the best quality and is easiest to adapt to modern video-display equipment. Modern VGA cables typically have a blue connector end, but some can also have black.



S-Video also became popular around 1987.

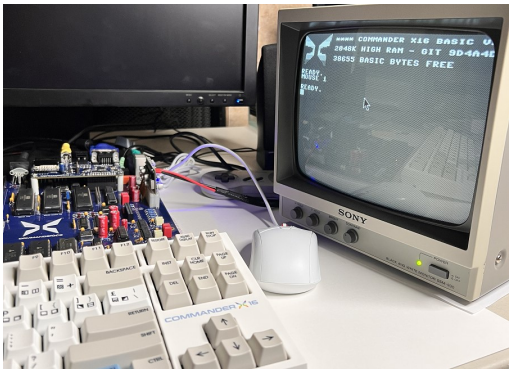
S-Video is technically better quality than Composite video. Sometimes VCR's will have an S-Video input. Take care to avoid bending the 4-pins within the connector end.



RCA cables were introduced in 1956!

Composite video (or also called RCA) typically has a yellow end connector. It is the easiest to connect because the orientation of the cable does not matter.

S-Video and Composite is intended for older CRT type displays. To go really old-school, you can also find Composite to RF or BNC adapters!



1.4 Adapting to HDMI

Each of the X16 video outputs (VGA, S-Video, Composite) can be adapted to HDMI. Many modern televisions and display devices will use HDMI. Smaller or thinner devices (such as portable LCD screens) may use mini-HDMI or micro-HDMI.

The HDMI standard was first introduced in 2002.

Adapting **VGA to HDMI** requires a "VGA to HDMI" adapter (or some vendors will use the term "converter" or "dongle"). Look for one that is "active" (powered by 5V USB) and also "with audio." Then consider the cable length you want: short (for a display that is on a desktop) or long (such as a larger display that is up on a wall). The 5V USB source will not go into the X16— instead use a suitable USB wall adapter (and a USB extension if necessary).

USB Wall Outlet Charge Adapter

USB Extension Cable

VGA to HDMI Adapter (active, with audio)

Adapting **Composite or S-Video to HDMI** is the same idea, but will generally require a larger "box" type adapter that also requires its own DC power source. **If you have an HDMI display available, there is little reason to use these options over VGA.** The Composite and S-Video output of the X16 is generally used to connect older CRT displays that still have these as native inputs (which gives a more "vintage experience").

Stereo Audio Splitter

Stereo Audio Cable (two rings)

Composite Video Cable

S-Video Cable

Composite to HDMI Adapter (USB 5V, 720/1080 up-convert)

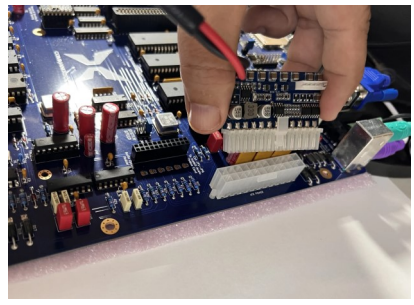
S-Video + Composite to HDMI Adapter (DC5V, up-convert, stereo audio pass-thru)

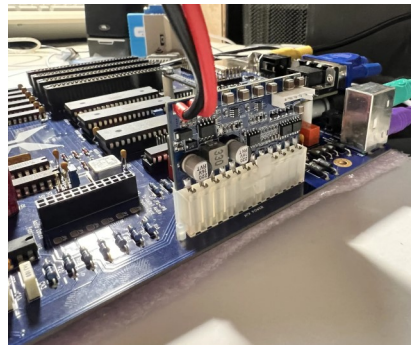
1.5 Power Supply Connection

The X16 motherboard has a standard ATX power connector. This can use a regular full size ATX power supply unit (PSU) that is typically in the 300W-600W range, or use a simpler ATX picoPSU in the 70-160W range.

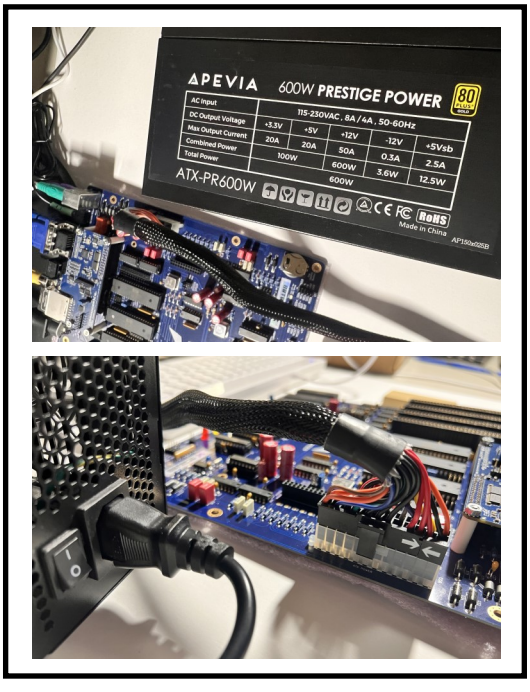
Make sure the ATX connector is fully seated. The connector should only fit one way, per the shape of the connector itself. The older 20-pin ATX connector can be used, but more modern PSUs will have a 24-pin connector (also sometimes called “20+4”).

If using a picoATX, a wall AC adapter to 12VDC is needed (with about 800mA to 2A).

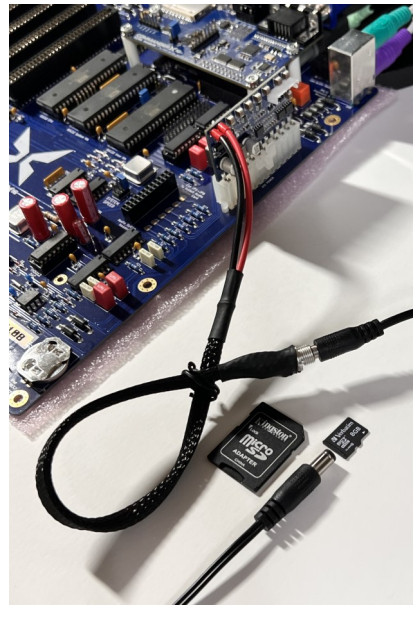




A 20-pin ATX connector can also be used!

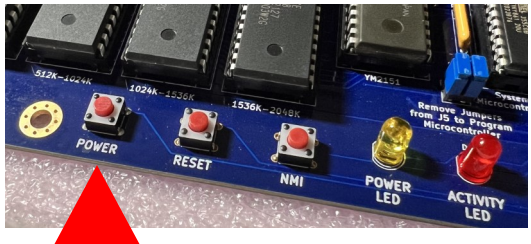


OR



Use a power plug barrel size of 5.5 x 2.5 mm.

SATA or Molex to 12V barrel plugs can be used to power various peripherals.



If you don’t yet have the Front Panel (FP) connectors in place, then press the red POWER button here to toggle the system on and off.

Be sure the adapter has the symbol for “center positive”:

2.0 Getting Started: File Management Commands

After starting the system, you are "booted" into what is called the "X16 BASIC." The flashing cursor indicates that the system is waiting for your commands. You can begin entering BASIC programming commands right away. But it is better to first get familiar with a few "file management" commands, so that you can save your work or load existing files.

X16 may not support certain older SD cards that are under 8GB in capacity.

The X16 has a built-in SD-card reader which by default is referenced as Device 8. This card represents the "disk drive" of the system and therefore the X16 BASIC extensions used to interact with this device are referred to as "CMDR-DOS" (Commander X16 Disk Operating System).

Listed below is a summary of the most typical commands used.

The X16 will still boot to X16 BASIC even if no SD-card is inserted!

```
DOS"$                List current directory (same as F7)
DOS"S:FILENAME"      Scratch (delete/erase) specified FILENAME
DOS"R:NEW=OLD"       Rename a file to NEW from OLD
DOS"MD:FOLDER"       Make a sub-folder called FOLDER
DOS"RD:FOLDER"       Remove sub-folder called FOLDER (if empty)
DOS"CD:/"            Back to root folder
DOS"CD:FOLDER"       Go to specified FOLDER
DOS"CD:.."           Move back up one folder
DOS 9                Set device 9 as default (such as SD2IEC)
DOS 8                Set device 8 as default (onboard SD-card)
```

To initialize or clear the contents of an SD-card that already has a FAT32 partition, use the following: (**WARNING:** DOS"N command erases all data content!)

```
DOS"I                          Initialize (re-mount filesystem)
DOS"$=P                        List partitions (optional)
DOS"CP1                        Select partition 1
DOS"N1:SAMPLE,X16,FAT32"       New (format) WARNING: Data Loss!
                               Sets volume name "SAMPLE" and
                               the ID "X16" can be anything.
```

Be aware that Windows may insist that SD cards larger than 32GB be formatted using exFAT, which is not compatible with the X16. Use "FAT32 Formatter" to force the use of FAT32 partition format on larger SD-cards.

There are a few "system commands" that are not in the category of CMDR-DOS or BASIC. The most common commands are:

The first thing to do on an X16 is type the **MENU** command!

Use this to set the clock and adjust the screen output.

```
MENU          Activate the system-settings menu
MON           Run the internal Machine Language Monitor
CODEX         Run CodeX (an advanced System Monitor)
RESET         Reset the system (like pressing CTRL-ALT-DEL)
POWEROFF      Power down the system (like pressing power switch)
HELP          Show ROM versions and link to support website
```

2.1 Introducing X16 BASIC

The X16 BASIC supports many of the typical BASIC keywords (such as LIST and PRINT), but has many new feature as well (such as MOUSE and BANK). The X16 BASIC also includes commands for using the advanced audio and video capabilities of the X16 VERA hardware! For review, here is a summary of the typical BASIC commands and programming syntax.

X16 BASIC is derived from Commodore BASIC, which was derived from the very original MicroSoft BASIC for the Altair 8800 in 1975!

```
CLS             Clear the screen
LIST            List the current program contents
REN             Renumber all BASIC line numbers w/ increment 10
SAVE "FILE"     Save the program to the current device with
                a filename of FILE (prefix with "@: to overwrite)
LOAD "FILE"     Load the specified FILE from the current device
NEW             Clear current BASIC program from memory
RUN             Run the current program
CONT            Continue running the current program
```

While most BASIC programs start with line numbers, some BASIC keywords can be used "interactively" and don't require a line number. For example, "PRINT A" can be used without a line number to show the current value of variable A. But in order to "remember" your program (to SAVE and RUN it), line numbers are required.

The following is a sample program to generate and display a random number between 1 and 10. After typing this program, use command "RUN" to see the result.

```
10 A=INT(RND(TI)*10)+1
20 PRINT A
RUN
```

Like a calculator, BASIC supports all the typical arithmetic operations (+, −, *, / for divide) on variables. But programming also involves aspects such as Boolean-logic, string-manipulation, and input-data stream processing. BASIC is a great way to be introduced to these core concepts. Listed below is a summary of the typical BASIC programming keywords. For more complete examples, see Appendix A.

Remember to **SAVE** your programs! The system will <u>not</u> save periodically for you.

Press **CTRL-C** while a BASIC program is running in order to stop its runtime.

```
PRINT "ABC"     Print string "ABC" (can use ? for PRINT)
INPUT A         Prompt for a numeric input, store into A
INPUT A$        Prompt for a string input, store into A$
PRINT A,B$      Print numeric A, tab, then string B$
RND(TI)         Random number floating-point value 0 to 1
MID$(A$,N,M)    Return M characters from string A$ at index N
STR$(A)         Convert numeric A into a string
TIME$           Get current time hhmmss (or TI$)
DATE$           Get current date yyyymmdd (or DA$)
TI              Get current counter time
MX / MY / MB    Mouse X, Y and button state values
MOUSE 1 (or 0)  Turn mouse cursor ON (or OFF)
FOR N = A TO B  Loop numeric variable N from A to B (inclusive)
NEXT N          Increment numeric variable N (and loop back to FOR)
GOTO 100        Go to line number 100 as next line to run
IF A > 5 THEN   If logic <condition> is true, then perform some code
GOSUB 1000      Go to line number 1000 and allow a RETURN
RETURN          Resume from the last GOSUB position
DIM M(300)      Initialize M to an array of dimension 300
CLR             Reset all variables (including DIM's)
```

2.2 The X16 Machine Language Monitor

The X16 has a built-in System Monitor that can be used to explicitly monitor (and adjust) the state of the system. BASIC reserves a specific portion of memory for use by its limited number of available variables. But eventually more sophisticated programs need the raw performance and resources of the system. The System Monitor is a way to load such programs, or to access system resources beyond what BASIC permits.

You can develop 6502 machine code externally, then load it into X16 memory using the L-command of the monitor.

The System Monitor is accessed by typing the **MON** command from within BASIC.

```
MON
C*
   PC  IRQ  BK AC XR YR SP NV#BDIZC
.;03A1 D0DA 00 B1 EB 00 ED *.**...*
.
```

A sample of the most typical commands used in the monitor is as follows:

```
M 0800           Monitor memory contents starting at address $0800 (hex)
D 0800           Debug (with disassembly) starting at address $0800
R                Display state of the CPU Registers
O n              Set BANK to n
X                Exit the monitor and go back to BASIC
F 0800 0900 00   Fill addresses $0800 to $0900 with byte value $00
G n              Start executing at the specified address n
T 0800 0900 A000 Transfer (copy) all contents of $0800 to $0900 over
                 into starting address $A000
S"@:SAMPLE.BIN",08,0000,FFFF
                 Save the contents of $0000 to $FFFF (hex) to device 8
                 using filename SAMPLE.BIN ("@:" means overwrite)
L"SAMPLE.BIN",08,0800
                 Load the contents of SAMPLE.BIN from device 8 into
                 memory starting at office $0800 (full file is loaded)
```

To stop monitoring or debugging, press **ENTER** twice.

During monitoring, use F3/F5 to scroll UP/DOWN across addresses.

As an example of what the System Monitor can be used for, suppose you have a simple machine code sample (there are online 6502 assemblers that can give you the equivalent object machine code):

src:

```
JSR $FECF ; invoke 24-bit RNG
STA $B000 ; store first byte   (8)
STX $B001 ; store second byte  (16)
STY $B002 ; store third byte   (24)
JSR $FECC ; enter monitor to examine $B000
```

object code:

```
0000: 20 CF FE 8D 00 B0 8E 01
0008: B0 8C 02 B0 20 CC FE
```

You can use the monitor to key in the object code at a specific address (M-command), save that memory content to a file (S-command), run and exercise the sample (G-command), and load the sample back in a future session (L-command).

In this example, whenever you run the code ("**G 2000**"), it generates a new random set of values and writes them into address B000, then returns to the monitor. You can use "**M B000**" command to view the new values.

```
READY.
MON

C*
   PC  IRQ  BK AC XR YR SP NV#BDIZC
.;03A1 D0DA 00 B1 EB 00 ED *.**...*
.M 2000
.:2000 20 CF FE 8D 00 B0 8E 01
.:2008 B0 8C 02 B0 20 CC FE 94
.:2010
.G 2000 run
C*
   PC  IRQ  BK AC XR YR SP NV#BDIZC
.;03A1 D0DA 00 B0 DE D9 E0 *.**....
.M B000
.:B000 B5 1D D9 36 90 52 8A 96
.:B008
.G 2000 run
C*
   PC  IRQ  BK AC XR YR SP NV#BDIZC
.;03A1 D0DA 00 B0 D1 DC D3 *.**....
.M B000
.:B000 B3 23 DC 36 90 52 8A 96
.:B008
```

APPENDIX A: Sample X16 BASIC Programs

(A.1) The PETSCII Chart Program— this program shows the available printable PETSCII symbols.

Key Concepts:

- Assigning values to variables and simple logic handling
- Special handling of double-quote (“) output
- Use of GOTO for branching

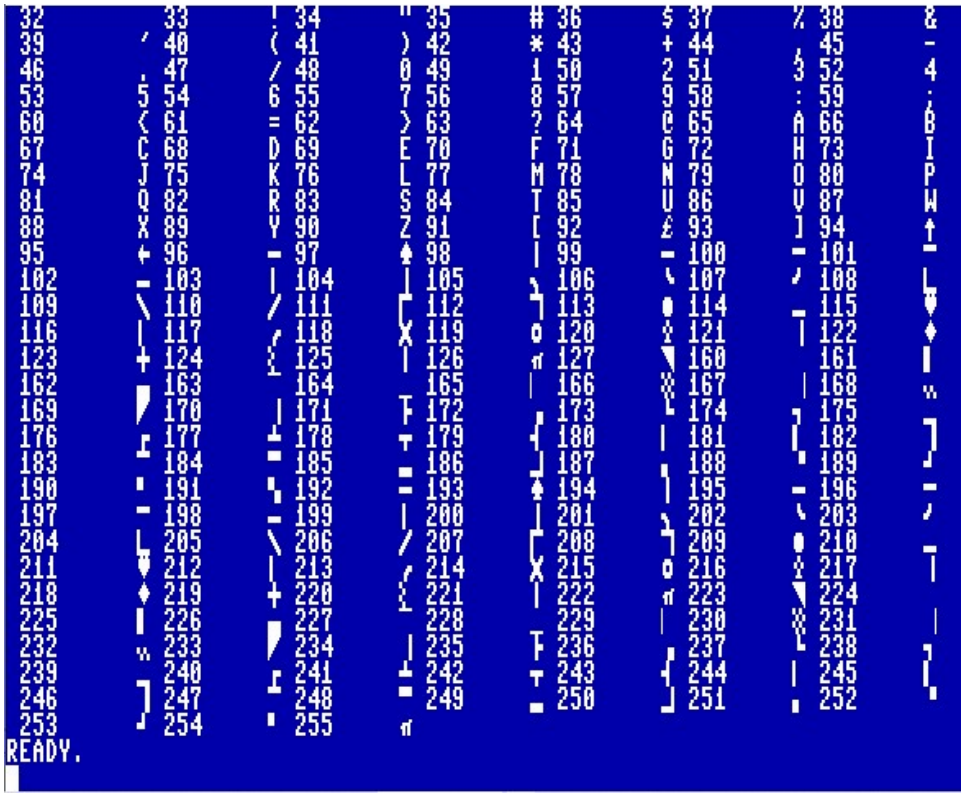
Type “**RUN**” to run your program starting at its first line number.

Use **CTRL-C** to stop a program and “**CONT**” to continue a program.

```
10 SCREEN 1:C=1
20 A=32:B=127
30 FOR I=A TO B
40 PRINT I,CHR$(I);
50 IF I = 34 THEN PRINT CHR$(34);:PRINT CHR$(157);
60 C=C+1
70 IF C>7 THEN GOTO 120
80 NEXT I
90 IF A > 100 THEN END
100 A=160:B=255:REM PREP NEXT PRINTABLE GROUP
110 GOTO 30
120 C=1:PRINT
130 GOTO 80
```

The non-printable codes do other effects such as color, reverse, or move the cursor.

Once entered correctly, this program will have an output that looks like the following. After running this program, press **SHIFT+ALT** to swap between the character sets.



(A.2) **The Confetti Program**— this is the X16 "celebration program" for the June 2023 Developer Board launch!

Key Concepts:

- Changing screen resolution modes
- Asynchronous user input query (scanning for ESC)
- String to numeric conversions
- PRINT control codes for color output
- Use of Random Number generation
- PRINT output of PETSCII symbols

The **REM** (reminder) portions are not required for the program to run properly.

```
10 SCREEN 6
20 W=20:H=15
30 GET A$:REM GET KEY INPUT (IF ANY)
40 A=ASC(A$)
50 IF A = 27 GOTO 160:REM ESCAPE TO EXIT
60 PRINT CHR$(INT(149+RND(TIME)*9));:REM COLOR
70 X=INT(RND(TIME)*W)+1
80 Y=INT(RND(TIME)*H)+1
90 IF ((X=W) AND (Y=H)) THEN GOTO 130
100 LOCATE Y,X
110 PRINT CHR$(160+RND(TIME)*95);
120 GOTO 30
130 REM HANDLE LAST ROW/COL ISSUE
140 REM TBD - PRINT CAUSES SCREEN SCROLLING
150 GOTO 30
160 REM USER PRESSED ESCAPE
170 SCREEN 1
180 COLOR 1,6:REM WHITE ON BLUE
190 END
```

Proposed Challenges:

- Why does Line 70 and Line 80 end with +1 ?
  HINT: Try running it with the +1 removed.
- Try correctly supporting different screen resolutions
- Allow user to press SPACE to cycle to different screen resolutions
  HINT: Keyboard code for space is 32
- Solve the screen scrolling issue during the X=W and Y=H condition
- Play background audio while "tossing the confetti"
  HINT: See PSGPLAY command used in example A.5

(A.3) **Experimenting with BSAVE**— determine an appropriate filename and then write the contents of memory to that filename.

Key Concepts:

- FOR-LOOP control loop
- Using MID$ to get sub-string portions
- Concatenating strings together
- Working with banks within BASIC

```
10 FOR I = 0 TO 3
20 A$=MID$(STR$(I),2):REM REMOVE PREFIX SIGN
30 IF LEN(A$) < 4 THEN GOTO 90
40 F$="@:TEST"+A$+".BIN"
50 PRINT I,F$:REM SHOW STATUS
60 BSAVE F$,8,I,$0000,$FFFF
70 NEXT I
80 END
90 A$="0"+A$
100 GOTO 30
```

You can increase the loop on up to 255!

Proposed Challenges:

- Examine the resulting .BIN files, explain why only the portion starting at $A000 differs across these files
- Revise the status to remain on the same line
- Consider: What does BSAVE do if the SD-card is full?
- Can the GOTO in Line 30 be replaced with a GOSUB?

(A.4) **Examine Top of a File** — open a specified file and examine the first few bytes.

Key Concepts:

- Using devices and file handles
- Converting portions of strings to individual bytes

Use this program to examine the files that were created in the previous example!

IMPORTANT: Use "BINPUT#" exactly as shown. (no spaces)

```
10 OPEN 9,8,9,"TEST.BIN":REM FILE HANDLE 9, DEVICE 8
15 M=10:REM MAX BYTES TO PEEK AT
20 BINPUT#9,A$,M
30 PRINT LEN(A$);"BYTES"
40 CLOSE 9:REM CLOSE FILE HANDLE 9
50 FOR I = 1 TO M
60 B$=MID$(A$,I,1)
70 PRINT I,ASC(B$)
80 NEXT I
```

(A.5) **Simple Paint Program** — use the mouse to draw on the display, with left mouse to change color and right mouse to erase.

Key Concepts:

- Invoking a system call from BASIC and processing its results
- Enable mouse pointer and working with mouse-state variables
- Working with audio output

```
10 C=13:PSGINIT:COLOR 15,0
20 MOUSE 0:SCREEN 6:MOUSE 1
30 REM MODE 3 MIGHT BE BETTER
40 SYS $FFED
50 W=PEEK($30D):H=PEEK($30E)
60 X=INT(MX/8)+1
70 Y=INT(MY/8)+1
80 IF (MB=1) THEN C=RND(TIME)*15
85 IF (MB=2) THEN C=0
90 IF ((X=W) AND (Y=H)) THEN GOTO 170
100 COLOR 14,1
110 LOCATE 1,1:PRINT STR$(X)+" ",STR$(Y)+" ";
120 COLOR C,0
130 REM CAUTION: MOUSE COULD BE OUT OF BOUNDS
140 LOCATE Y,X
150 PRINT CHR$(160+RND(TIME)*80);
160 GOTO 50
170 LOCATE 1,1:REM NOT NECESSARY
180 PSGPLAY 0,"T180A20": REM BEEP
190 GOTO 50
```

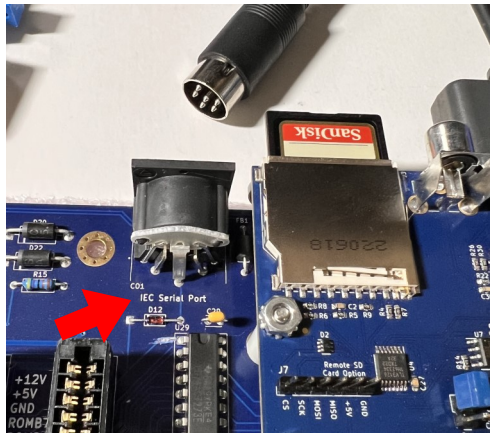
"MB" is for Mouse Button!

Proposed Challenges:

- Try different screen resolutions
- Add a SAVE/LOAD feature of the current screen content
- Allow user to toggle the X/Y cursor status on or off
- Why is a beep warning being emitted when touching the lower right corner?
  HINT: Related to same issue as A.2 sample.

APPENDIX B: Use of the IEC Connection

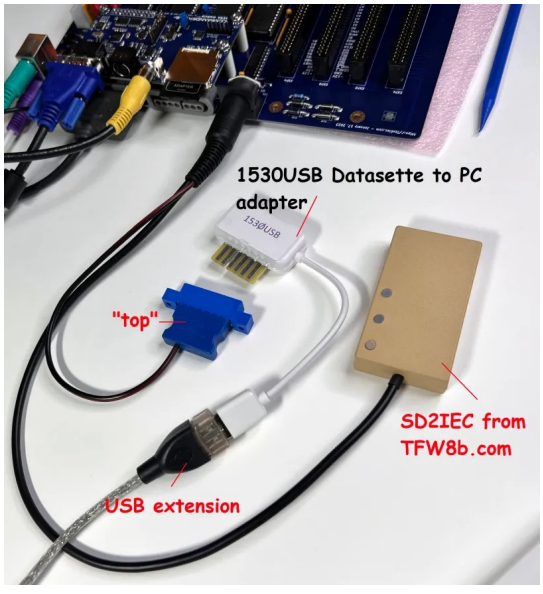
Next to the SD-card is a round 6-pin socket called the IEC Serial Port. This is the same as the 6-pin Serial connector used on the Commodore 64, which means that CBM disk drives can still be connected to the X16. Other compatible devices include various plotters and printers, and perhaps some future devices may yet choose to use this IEC bus interface.



IEC-462 is associated with IEEE-488, which are international standards derived from HP (known as GPIB) in 1978.

IEC-462/IEEE-488 was originally a 24-pin parallel data standard. Commodore modified this to a 6-pin serial standard for the VIC-20 in 1981.

The SD2IEC requires its own 5V power. In this setup, the 1530USB adapter is used only to provide that 5V.



There are also modern SD-card based replacements to the CBM 1541 disk drives. Typically these are called SD2IEC. They support the 1541 floppy formats and disk images thereof, so it can provide a means of directly transferring legacy C64 data over to the X16. Or it can just be used to provide an alternative means of saving/loading files besides the main SD-card slot of the system.

Be aware that the IEC interface is noticeably slower than the built-in SD-card storage.

The X16 SD-card is initially set as Device 8. If an IEC device is also configured as device 8, then it will need to be reconfigured (typically to device 9). For the TFW8b SD2IEC device shown here, the follow commands are used to reconfigure its device number:

```
OPEN1,<current address>,15,"U0>"+CHR$(<new address>):CLOSE1
OPEN1,<new address>,15,"XW":CLOSE1
```

Where `<current address>` is 8 and `<new address>` is 9. Other device numbers may also be used, though some may be reserved. You can then swap between device using the following commands:

```
DOS 8
DOS 9
```

That is, X16 DOS commands will act upon the last specified device number (unless additional command parameters indicate otherwise).

APPENDIX C: Updating System ROM Software

ROM is Read Only Memory, used to hold software that does not need to be loaded.

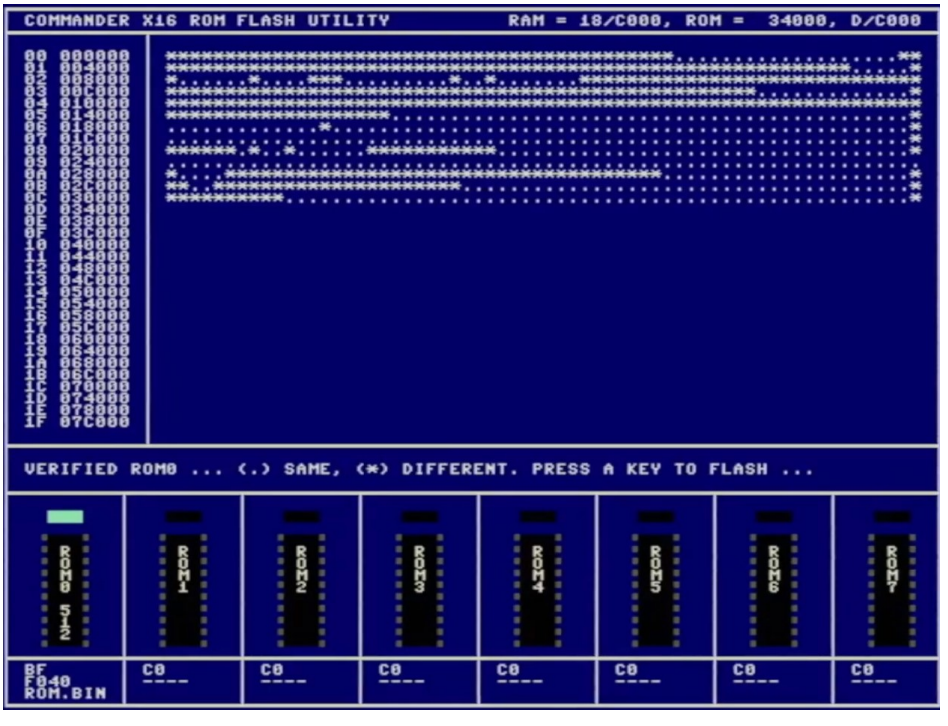
ROM updates should be rarely done. Always consult the X16 community (forums, Discord, etc.) to be sure to have the newest information about the procedure and location of correct ROM software. The following is a general overview of the steps:

The **SMC ROM** contains code such as interpreting PS/2 keyboard inputs into codes the System ROM can understand, and interfacing with the SD-card. The **System ROM** contains the built in BASIC, MONITOR, general screen editing features (blinking the cursor, moving cursor around), and File I/O routines. The **VERA ROM** contains code related to interacting with the audio and video hardware. Each of these ROMs can be updated independently of each other.

The SD card included with the system contains a SYSTEM folder. In that folder is the **FLASHVERA.PRG** and **SMCUPDATE.PRG** programs. Then there is a ROMFLASH folder containing the **FLASH-CX16.PRG** program.

FLASHVERA.PRG requires the file named **VERA.BIN**, while the FLASH-CX16.PRG requires the filename to be **ROM.BIN**. In all cases, the filename should be in uppercase when copying the new ROM file to the SD-card. Run the respective ROM update program, and follow the on-screen prompts.

Most ROM updates will take under one minute. But the worse case scenario is to lose power during a ROM update, since the system may no longer correctly startup with only a partially loaded ROM. In such a situation, the system can likely be recovered, but you will need to consult with the X16 community on what corrective action to take (such as installing a correctly flashed ROM provided by someone else).



Next Steps:

This manual is just the beginning— a stepping stone—to your journey of exploring the X16's capabilities further! This brief introduction to the Commander X16 system is intended as a "***welcome back***" to those who have been away from such systems for too long, or to help guide those who are completely new to these kinds of systems.

The Commander X16 has far more capabilities than has been described here and will continue to evolve in its capabilities. With that in mind, in this introductory manual we focused on the "core" aspects that (most likely) won't be changing over time.

`https://www.commanderx16.com/` (forum)

`https://github.com/x16community/x16-docs`

`https://www.facebook.com/groups/CommanderX16`

`https://discord.com/invite/nS2PqEC`

NOTES

Appendix D:
X16 Quick Reference

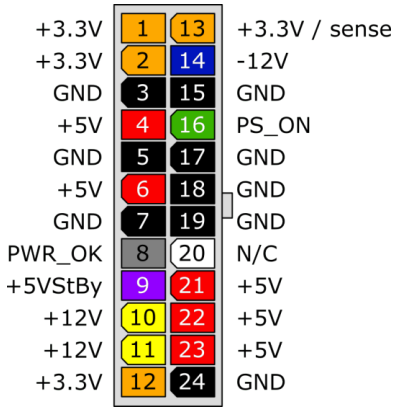
X16 BASIC Function Keys

```
F1 = LIST <ENTER>
F2 = SAVE"@:
F3 = LOAD"
F4 = SCREEN-1
F5 = RUN  <ENTER>
F6 = MONITOR <ENTER>
F7 = DOS"$ <ENTER>
F8 = DOS"
```

Conversion Chart

DEC	HEX	BIN
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

ATX Power Pins



X16 BASIC CHR$ Codes

```
CHR$(7)   BELL (beep)
CHR$(14)  lower case mode
CHR$(19)  cursor home
CHR$(28)  red
CHR$(30)  green
CHR$(31)  blue
CHR$(129) orange
CHR$(142) upper case mode
CHR$(144) black
CHR$(147) clear screen
CHR$(149) brown
CHR$(150) pink
CHR$(151) dark gray
CHR$(152) medium gray
CHR$(153) light green
CHR$(154) light blue
CHR$(155) light gray
CHR$(156) purple
CHR$(158) yellow
CHR$(159) cyan
```

X16 BASIC Keyboard Codes

```
LEFT ARROW   = 157
RIGHT ARROW  = 29
UP ARROW     = 145
DOWN ARROW   = 17
SPACE        = 32
ENTER        = 13
ESCAPE       = 27
TAB          = 9
```

X16 Keyboard Hotkeys

```
CTRL+ALT+DELETE
   reset the system
CTRL+ALT+PRINT SCREEN
   break to BASIC (NMI)
SHIFT+SCROLL LOCK
   cycle video output mode
SHIFT+ALT
   toggle character set
```

The Commander X16 project began around 2019, with an idea of making an updated “Commodore-like” desktop home computer system. Updates such as built-in VGA support, more than 64K system RAM, faster clock speed, and more modern available components. And then, the Pandemic of 2020 very suddenly disrupted many aspects of day-to-day life. Physical production of the system had to be put on hold. But through emulators and virtual design, the volunteer development team pressed on. They had to deal with legal issues of incorporating aspects of original Commodore software, make judgement calls on what hardware components would be sustainable as things eventually came back to normal, and do the technical work of enhancing the original Commodore core to make use of the new system capabilities.

After years of designing and developing, the initial batch of Developer Board hardware was first sold on Thursday, June 22nd, 2023. This booklet is a brief overview of setting up and using that system.