CESC16 Peripherals:

Peripheral support on the CESC16 Computer:

The computer can access 4 peripherals (IO ports), which are mapped in the highest 256 memory addresses: `[0xFF00, 0xFFFF]`. Reads and writes to one of those addresses will activate the Read/Write lines of the corresponding port.

Each IO is connected to the CPU with a 16-bit data bus and some control signals:

- **Input (CPU → Peripheral)**:
  - `CLK`: *System clock*.
  - `/Out`: *Output Enable* (active low). The peripheral outputs immediately to the bus.
  - `/In`: *Input Enable* (active low). The peripheral reads from the bus on the rising edge of CLK (synchronous signal).
- **Output (Peripheral → CPU)**:
  - `IRQ`: *Interrupt request*. For every rising edge, the CPU will wait until the current instruction finishes and will jump to an interrupt.

I/O Ports:

- **`0xFF00-0xFF3F`**: GPIO port 0 → PS/2 Keyboard (*Input*) + Serial line (*Input, Output*)
- **`0xFF40-0xFF7F`**: GPIO port 1 → VGA Text Terminal (*Output*)
- **`0xFF80-0xFFBF`**: GPIO port 2 → 16-bit timer
- **`0xFFC0-0xFFFF`**: GPIO port 3 → USB flash drive (*Input, Output*)

Custom ASCII table:

The protocols used by the peripherals use a slightly modified version of the ASCII table. Some control characters have been removed to make space for some keys.

HEX	Symbol	Name	Description (Output)	Description (Keyboard)
00	NUL	Null char	No output	No key pressed
01-05	...	RESERVED	(reserved for future commands)	(reserved for future commands)
06	ACK	Acknowledgment	*[CMD]* Input has been read, clear register	UNUSED
07	RDY	Ready	*[CMD]* CPU is ready to be interrupted	UNUSED
08	BS	Back Space	Delete char and move cursor left	BACKSPACE key
09	HT	Horizontal Tab	Insert tab	TAB key
0A	LF	Line Feed	New line	ENTER key
0B	VT	Vertical Tab	New line (without Carriage Return)	Page Up key
0C	FF	Form Feed	Insert a page break	Page Down key
0D	CR	Carriage Return	Move cursor to beginning of line	HOME key
0E	INS	Insert	UNUSED	INSERT key
0F-1A	F1-F12	FUNCTION KEYS	UNUSED	F1 - F12 keys
1B	ESC	Escape	UNUSED	ESC or END key (KEYBOARD INTERRUPT)
1C	LEFT	Left	Move cursor left	Left arrow key
1D	RGHT	Right	Move cursor right	Right arrow key
1E	DOWN	Down	Move cursor down	Down arrow key
1F	UP	Up	Move cursor up	Up arrow key
20-7E	...	PRINTABLE CHARS	(char)	(key)
7F	DEL	Delete	Delete char and move cursor right	DEL key

Port 0: PS/2 Keyboard + Serial

Summary: This port combines 2 peripherals. The *keyboard controller* translates the keystrokes of a PS/2 keyboard to the corresponding ASCII codes, and sends them to the CPU. The *serial line* allows the CPU to communicate with another computer, using *UART over USB*.
Uses microcontrollers: Yes (Arduino Nano): link to source code.
Causes interrupts: Yes, when a keystroke or serial input has been received

How to use: Use the syscall `INPUT.AttachInterrupt` to attach an interrupt handler (address of the routine to be called when a key is pressed). The first and only argument (passed in `a0`) of the handler is the ASCII code of the pressed key.

Port 1: VGA Terminal

Summary: Text terminal with support for some commands (clearing the screen, moving the cursor, etc.). The CPU sends characters in ASCII format and the terminal displays them on a VGA monitor (it generates a VGA signal with a resolution of 320x480@60Hz). The terminal also allows setting the color for each line, 64 colors (6 bit color) are supported.
Uses microcontrollers: Yes (ATmega328P): link to source code.
Causes interrupts: No

How to use: Use the syscall `OUTPUT.char` to print a character to the screen. In order to send a command to the terminal, see the other `OUTPUT` syscalls. Also see the `PRINT` routines to print integers and strings.

Port 2: 16-bit timer

Summary: Consists of a 16-bit binary counter (with a 16x prescaler) that interrupts the CPU when it overflows. This allows the CPU to measure the time that a task takes, and/or schedule a timer after a number of clock cycles.
Uses microcontrollers: No
Causes interrupts: Yes, when the countdown finishes (the timer overflows)

How to use: Use the syscall `TIME.AttachInterrupt` to attach an interrupt handler (address of the routine to be called when the timer finishes). Call `TIME.SetTimer` to start the timer (`a0` is the number of ticks to wait before calling the interrupt, 1 tick = 16 clock cycles). Call `TIME.ReadTimer` to see how many ticks have passed since the timer started.

Port 3: Disk (USB flash drive)

Summary: This peripheral allows the CPU to read the contents of a USB drive, using a CH376S module. It provides the CPU with several commands in order to perform different actions on the file system (open, read, write, change directory, etc.).
Uses microcontrollers: Yes (ATmega328P + CH376S): link to source code.
Causes interrupts: Yes, on specific events (USB becomes disconnected, etc.)

How to use: See the `DISK` routines to perform disk commands such as open/close files, create/delete directories and read/write from disk.