ELIZA Business Card Firmware Description

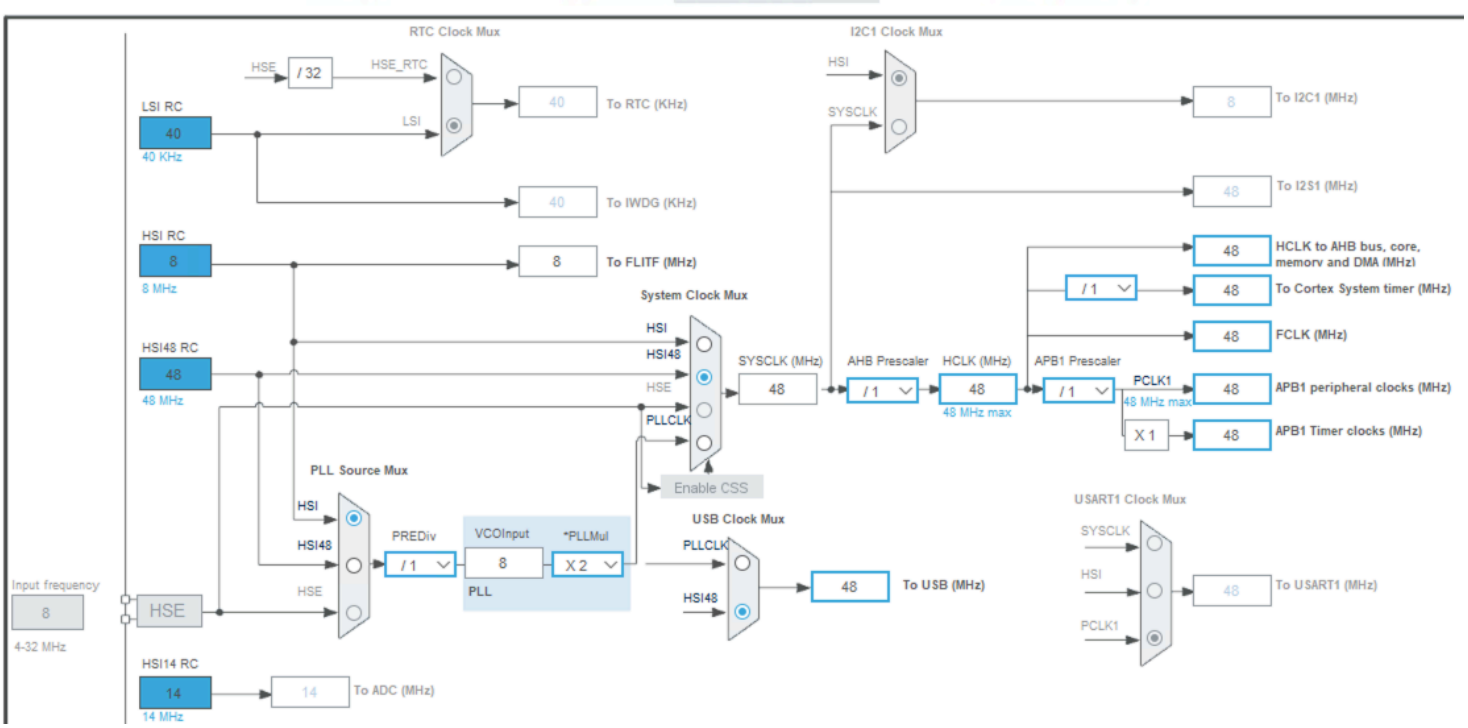
This directory contains the ELIZA Business Card firmware project. It is an STM32CubeIDE project. Note that the ST USB Device Library Middleware component has been removed since it is licensed for use with only ST silicon. The compiled binary files have also been removed since they contain this code. All other ST components are licensed "as-is".

However I'm a little proud of my work on this project so I share that here. I'm releasing my code "as-is" too with no warranty or guarantee of any sort in the hopes it will be helpful or at least amusing.

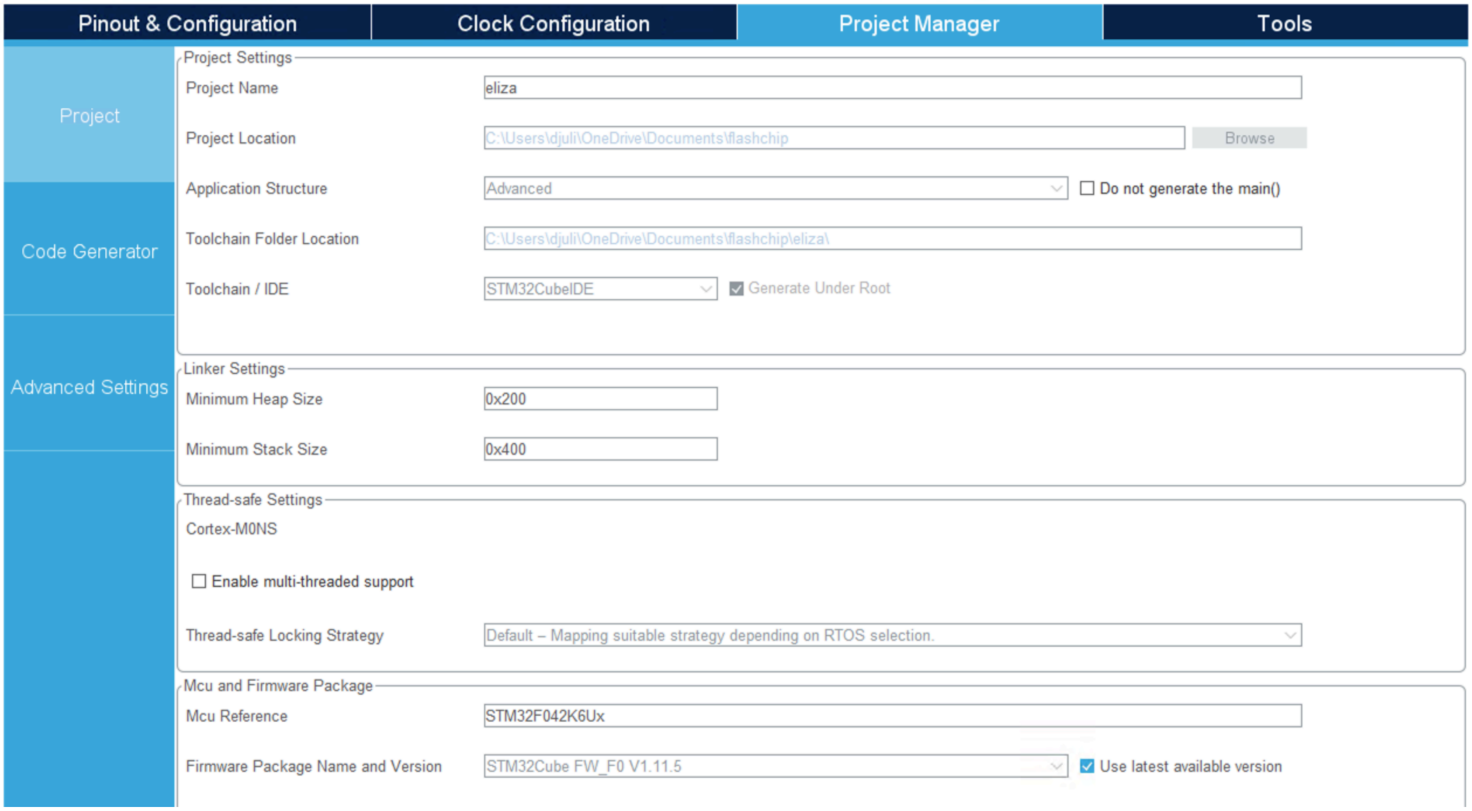
Project Setup

The project was created around a STM32F042K6 chip because it has the same pinout as the 'F096 and seemed to have the clock generation and USB peripherals. However it has only 32KB of Flash and 6KB of RAM (as opposed to 256 KB of Flash and 24 KB of RAM). I used the configurator to set the pins, configure the clock, enable USB and the USB CDC serial port middleware. See below for screenshots showing my project configuration.

Using the internal 48 MHz HSI clock for USB.



Project setup uses default stack and minimum heap sizes.



Project Settings

Project Name: eliza

Project Location: C:\Users\djuli\OneDrive\Documents\flashchip

Application Structure: Advanced ☐ Do not generate the main()

Toolchain Folder Location: C:\Users\djuli\OneDrive\Documents\flashchip\eliza\

Toolchain / IDE: STM32CubeIDE ☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Thread-safe Settings

Cortex-M0NS

☐ Enable multi-threaded support

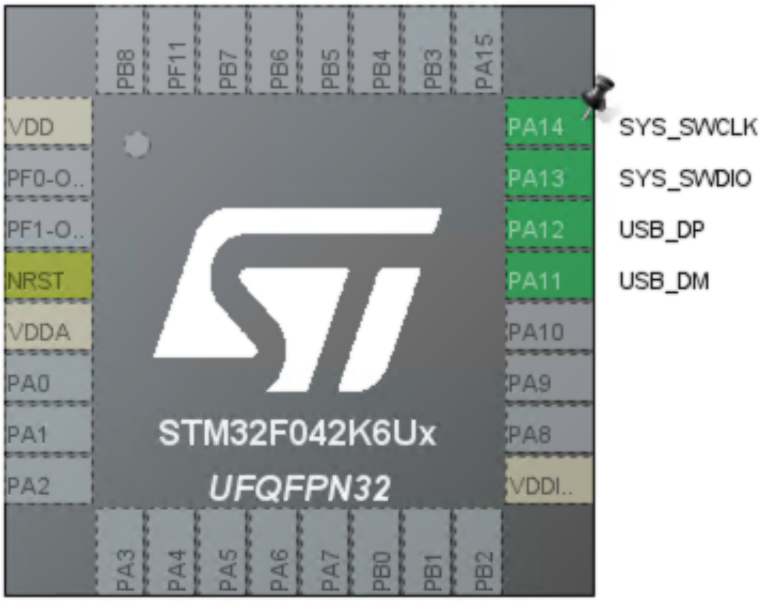
Thread-safe Locking Strategy: Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

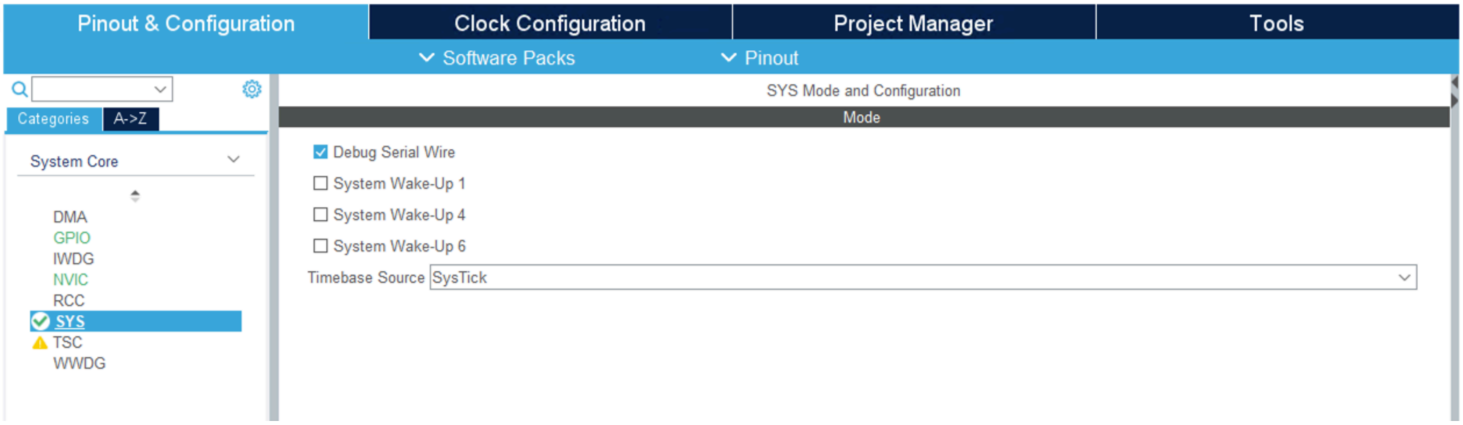
Mcu Reference: STM32F042K6Ux

Firmware Package Name and Version: STM32Cube FW_F0 V1.11.5 ☑ Use latest available version

I only use the USB pins and programming pins.



Sys setup is default too. I never used the serial debug (it probably wouldn't have worked on the non-ST part).



☑ Debug Serial Wire

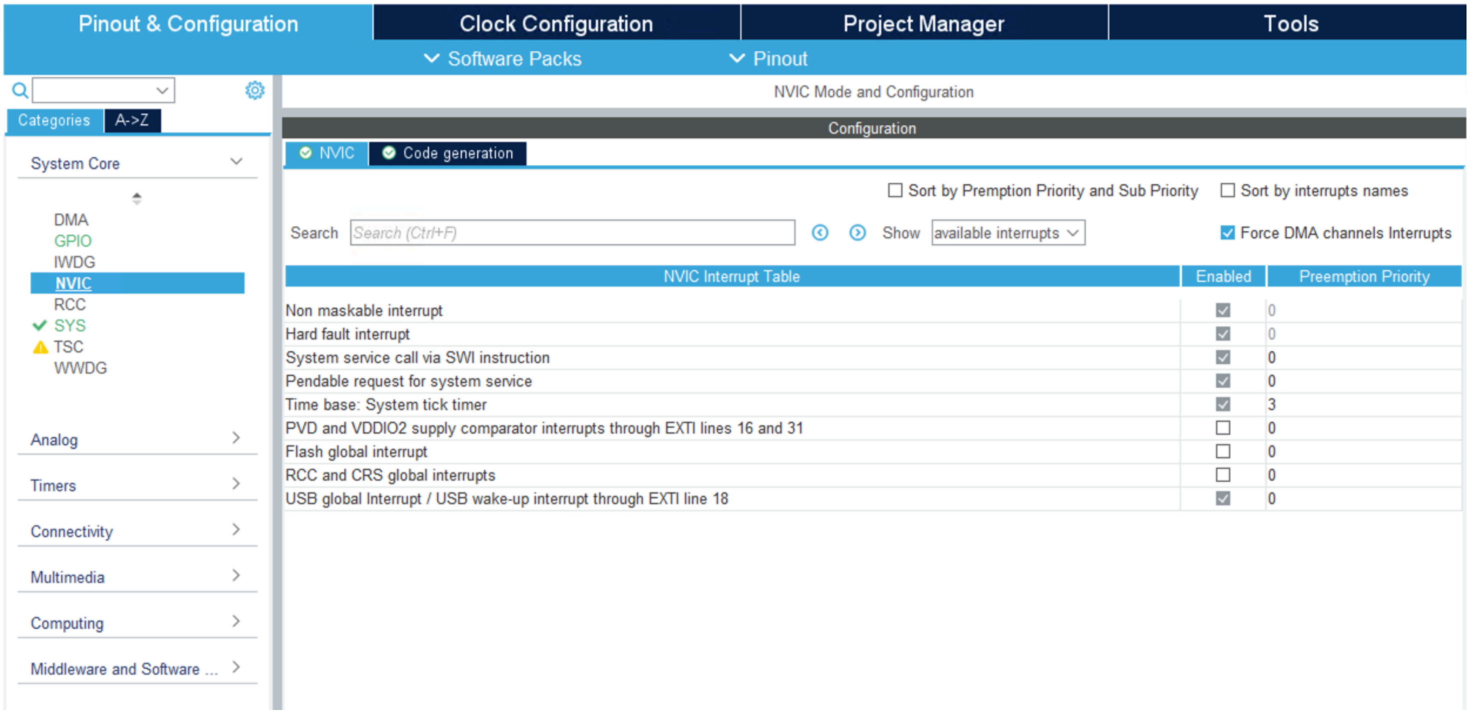
☐ System Wake-Up 1

☐ System Wake-Up 4

☐ System Wake-Up 6

Timebase Source SysTick

NVIC is default with enabled USB interrupt.



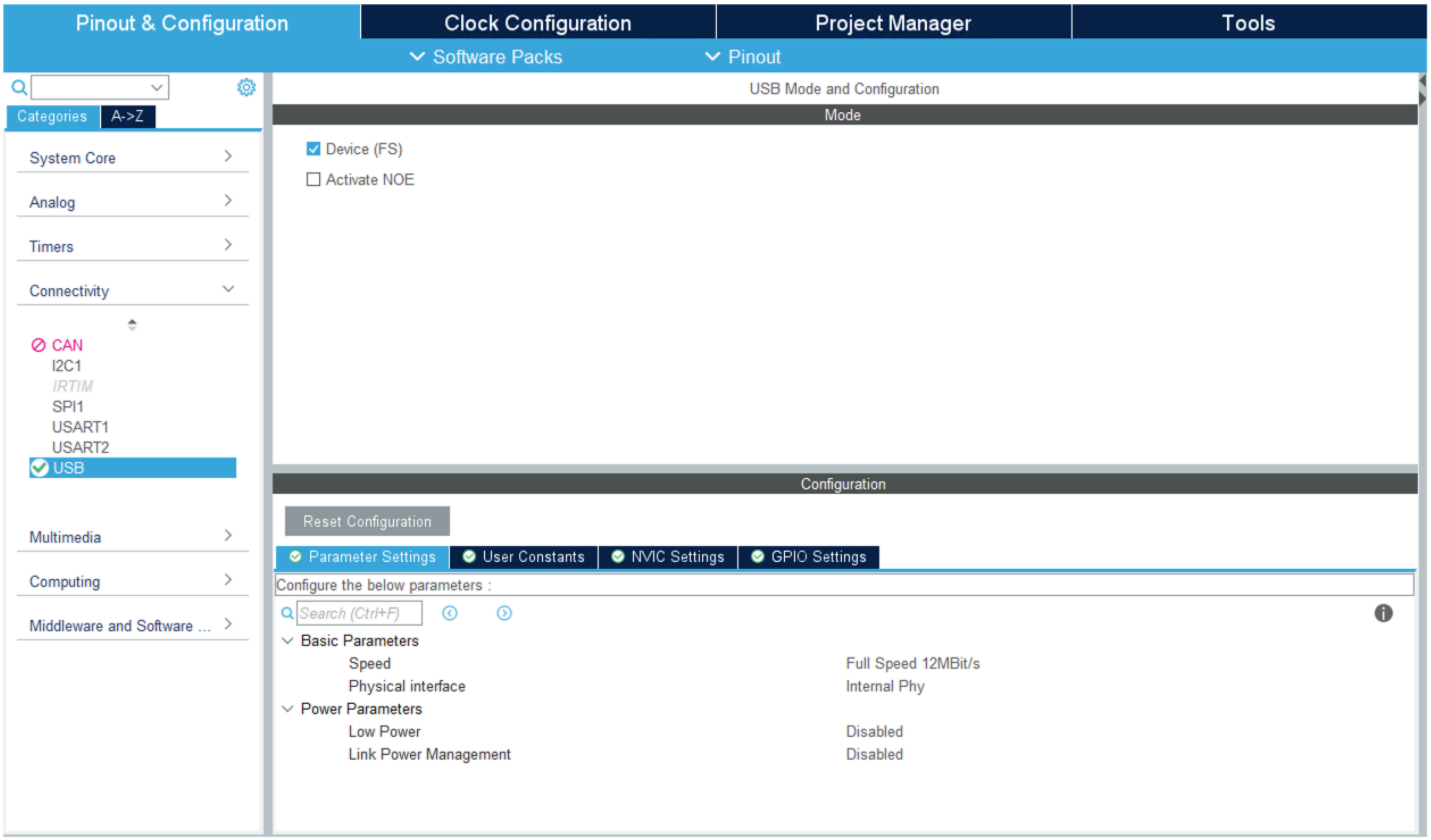
☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

Show available interrupts

☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	☑	0
Hard fault interrupt	☑	0
System service call via SWI instruction	☑	0
Pendable request for system service	☑	0
Time base: System tick timer	☑	3
PVD and VDDIO2 supply comparator interrupts through EXTI lines 16 and 31	☐	0
Flash global interrupt	☐	0
RCC and CRS global interrupts	☐	0
USB global Interrupt / USB wake-up interrupt through EXTI line 18	☑	0

USB is enabled.



USB Mode and Configuration

Mode

☑ Device (FS)

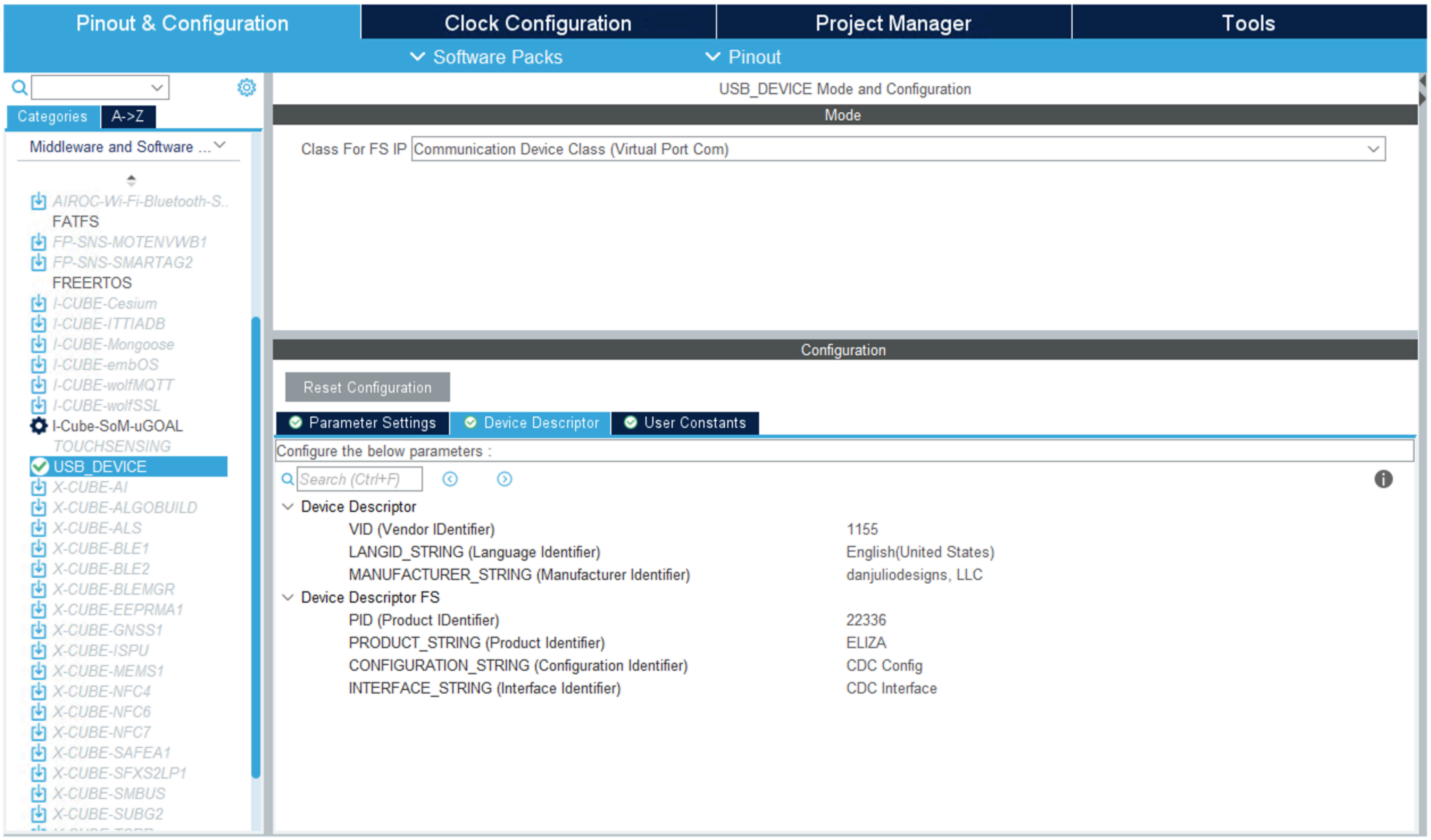
☐ Activate NOE

Configuration

Configure the below parameters :

Basic Parameters	
Speed	Full Speed 12MBit/s
Physical interface	Internal Phy
Power Parameters	
Low Power	Disabled
Link Power Management	Disabled

The USB_DEVICE middleware library is configured in a pretty default way. No doubt my Descriptors are all kinds of wrong. But I made a grand total of 10 cards so I suspect that won't be a problem.



USB_DEVICE Mode and Configuration

Class For FS IP Communication Device Class (Virtual Port Com)

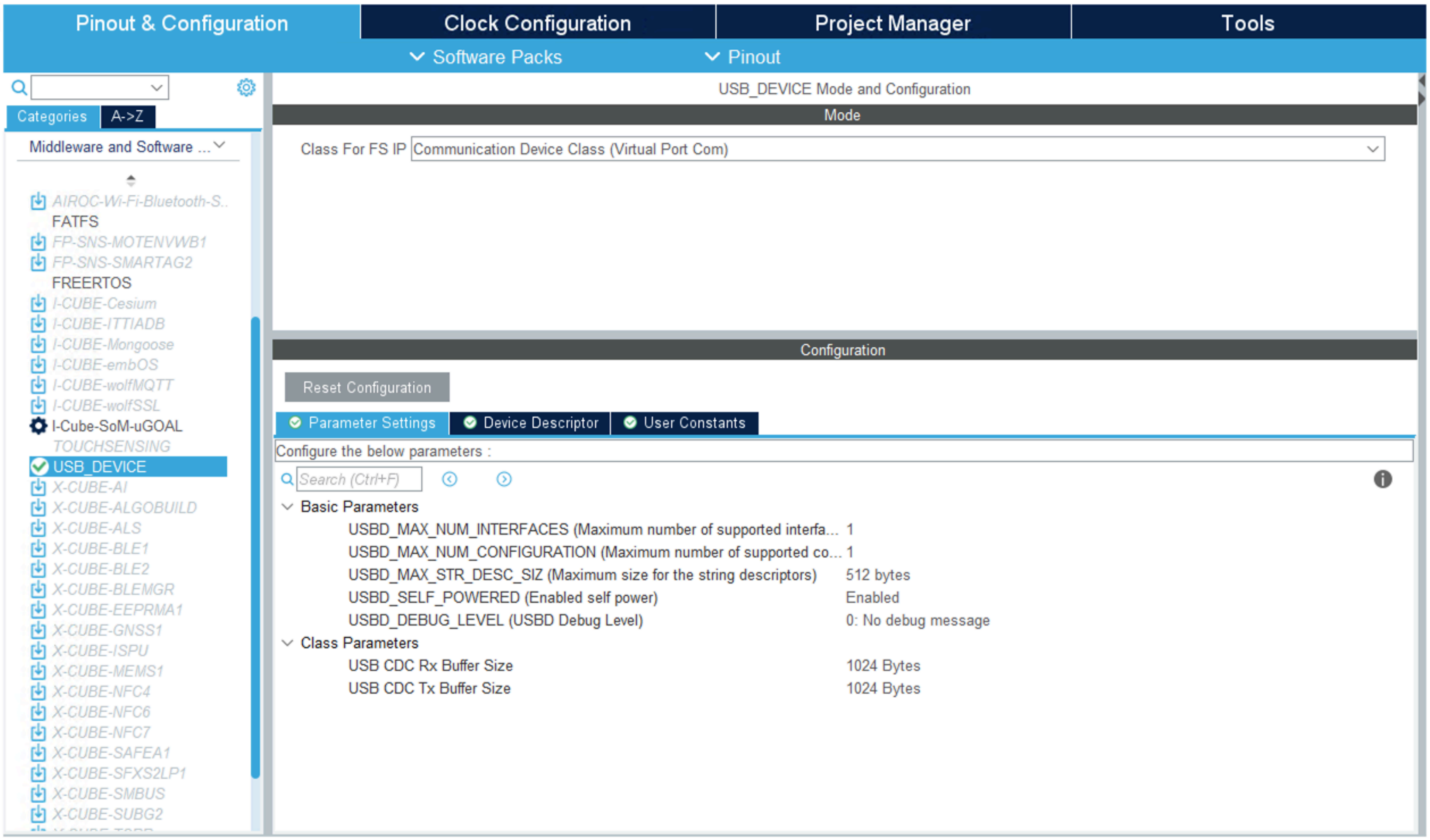
Configure the below parameters :

Device Descriptor

VID (Vendor IDentifier)	1155
LANGID_STRING (Language Identifier)	English(United States)
MANUFACTURER_STRING (Manufacturer Identifier)	danjuliodesigns, LLC

Device Descriptor FS

PID (Product IDentifier)	22336
PRODUCT_STRING (Product Identifier)	ELIZA
CONFIGURATION_STRING (Configuration Identifier)	CDC Config
INTERFACE_STRING (Interface Identifier)	CDC Interface



USB_DEVICE Mode and Configuration

Class For FS IP Communication Device Class (Virtual Port Com)

Configure the below parameters :

Basic Parameters

USBD_MAX_NUM_INTERFACES (Maximum number of supported interfa...	1
USBD_MAX_NUM_CONFIGURATION (Maximum number of supported co...	1
USBD_MAX_STR_DESC_SIZ (Maximum size for the string descriptors)	512 bytes
USBD_SELF_POWERED (Enabled self power)	Enabled
USBD_DEBUG_LEVEL (USBD Debug Level)	0: No debug message

Class Parameters

USB CDC Rx Buffer Size	1024 Bytes
USB CDC Tx Buffer Size	1024 Bytes

Firmware locations

My code lives in the `Eliza` subdirectory. There are two modules.

1. `eliza.h` and `eliza.c` contain the port of the 1977 BASIC program. It's a bit hacky as it was a more-or-less direct copy of the original BASIC translated to C.
2. `virthost.h` and `virthost.c` contain a shim API to interface Eliza with a serial stream of some sort. In this case the USB CDC library middleware. These files came originally from my Eliza port used in the my retrocomputer.

The `Core` subdirectory contains the boilerplate main code. I modified `main.c` to call `eliza_setup()` and `eliza_loop()`.

The `USB_HOST` subdirectory contains the auto-generated code that provides a user API to the USB CDC library middleware. Specifically I modified the `App/usbd_cdc_if.h` and `App/usbd_cdc_if.c` files to support a pair of circular buffers to transfer data between the virthost functions and USB. I also added a flag that is controlled by the Line State so I know if a USB serial port is open or not.

Programming the Flashchip part

To support programming the FCM32F096KCU6 I did the following things.

I modified the automatically generated linker script to support the larger memories.

```
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 24K
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 256K
}
```

This allowed the build process to complete without errors. This has to be redone if the configurator is run again.

And, finally, I used a DAPlink programmer to load the hex file (I had to add hex as an output option in the project preferences) into the 'F096 via the PCB ICSP header. I connected a 3.3V supply on the DAPlink programmer to the V+ line to supply power.