tell me about sparkfun red-b boards which riscv is on them?

As of my last knowledge update in September 2021, SparkFun doesn't have a specific "RedBoard" that features a RISC-V microprocessor. However, SparkFun is known for its RedBoard series, which are development boards based on the Arduino platform, and they mostly feature AVR and ARM Cortex-M0+ microcontrollers.

RISC-V is an open-source instruction set architecture (ISA) based on the reduced instruction set computing (RISC) principles. It has gained popularity due to its flexibility, modularity, and scalability, which makes it a suitable choice for a wide range of applications, from microcontrollers to high-performance computing.

Although SparkFun doesn't have a RedBoard with a RISC-V processor, there are other development boards featuring RISC-V microcontrollers available in the market. Some examples include:

1. SiFive HiFive1 Rev B: This development board features the SiFive Freedom E310 RISC-V microcontroller, which is a 32-bit RV32IMAC core running at 320 MHz.
2. Longan Nano: This board is powered by the GD32VF103CBT6 RISC-V microcontroller from GigaDevice, which has a 32-bit Bumblebee core running at 108 MHz.
3. SeeedStudio Wio Lite RISC-V: This board features the GD32VF103 RISC-V microcontroller and is compatible with the Arduino development environment.

These boards are just a few examples, and there might be other options available in the market. It is essential to keep in mind that the development board landscape is continuously evolving, and new products may have been released since my knowledge update in September 2021.

need I add here files as VMEX.c to be project built using this makefile?

```
# Copyright 2019 SiFive, Inc #
# SPDX-License-Identifier: Apache-2.0 #
```

```
PROGRAM ?= VMEX

$(PROGRAM): $(wildcard *.c) $(wildcard *.h) $(wildcard *.S)
	$(CC) $(CFLAGS) $(LDFLAGS) $(filter %.c %.S,$^) $(LOADLIBES) $(LDLIBS) -o $@

clean:
	rm -f $(PROGRAM) $(PROGRAM).hex
```

Based on the provided Makefile, it looks like you don't need to explicitly add the `VMEX.c` file or any other source files in the Makefile. The Makefile is set up to automatically build the project using all `.c`, `.h`, and `.S` files present in the same directory as the Makefile.

The line `$(PROGRAM): $(wildcard *.c) $(wildcard *.h) $(wildcard *.S)` specifies that the target `$(PROGRAM)` (in this case, `VMEX`) depends on all `.c`, `.h`, and `.S` files in the current directory. The `wildcard` function in the Makefile will find all files with those extensions.

When you run `make` in the same directory as the Makefile, it will use the provided rules to build the `VMEX` executable using all the `.c`, `.h`, and `.S` files in the directory.

So, if you have a `VMEX.c` file in the same directory as the Makefile, it will be included in the build automatically. Just make sure that your `VMEX.c` and any other source files you want to include are in the same directory as the Makefile.

---

umm, this was SiFive FreedomStudio (eclipse) ... I finally tested my code in proven Segger using just J-Link to Sparkfun RED-V board based on SiFive FV310 ... know you please how can I use in such plain project some includes/defines for FV310 registers etc ??

To use register definitions and other useful utilities for the SiFive FE310 microcontroller in your plain project, you can use the Freedom E SDK provided by SiFive. The SDK includes a set of libraries, examples, and register definitions for SiFive E31 and E51 RISC-V cores, including the FE310.

To get started, follow these steps:

1. Clone the Freedom E SDK repository:

bash Copy code

```bash
git clone https://github.com/sifive/freedom-e-sdk.git
```

2. Initialize and update the submodules:

css Copy code

```css
cd freedom-e-sdk git submodule update --init --recursive
```

3. You will find the necessary header files for FE310 in the following directory:

bash Copy code

```bash
freedom-e-sdk/bsp/env/freedom-e300-hifive1/
```

In this directory, you will find two important header files:

`encoding.h`: Contains register names, constants, and macros related to the RISC-V ISA.

`platform.h`: Contains memory-mapped I/O register addresses, peripheral base addresses, and other board-specific definitions for the HiFive1 board, which uses the FE310 microcontroller.

4. Include these header files in your project. For example, in your main C file, you can use:

c Copy code

```c
#include "platform.h" #include "encoding.h"
```

Make sure to update the include paths in your build system (e.g., Makefile) to point to the directory containing the header files.

With these headers included, you can access the registers and other hardware-specific features of the FE310 microcontroller in your code.

If you want to use the provided libraries for the peripherals (like GPIO, UART, PWM, etc.), you can find them in the `freedom-e-sdk/software` directory. You can also explore the example projects provided in the SDK to get a better understanding of how to use the libraries and header files.

---

so, when I use that includes in Segger IDE, it will be the same? I only want some named access to registers and areas, probably for now

---

Yes, you can use the same header files in the SEGGER Embedded Studio (SES) IDE to get named access to registers and memory areas for the SiFive FE310 microcontroller. The process is quite similar:

1. Follow the steps mentioned in the previous answer to clone the Freedom E SDK and locate the necessary header files (`platform.h` and `encoding.h`).
2. Create a new project in SEGGER Embedded Studio (SES) for the Sparkfun RED-V board (or open an existing one).
3. Configure the include paths in the project settings:
   Right-click on the project in the Project Explorer and select "Options".
   In the "Common" section, go to the "Code" tab.
   In the "Additional Include Directories" field, add the path to the directory containing the header files, e.g., `<path_to_freedom_e_sdk>/freedom-e-sdk/bsp/env/freedom-e300-hifive1/`.
4. Include the header files in your C source files where you need access to the registers and memory areas:

c Copy code

```c
#include "platform.h" #include "encoding.h"
```

Once you've set up the include paths and included the header files in your source files, you will have named access to the registers and memory areas of the SiFive FE310 microcontroller, and you can use them in your code as needed.

thanks :-)

You're welcome! If you have any more questions or need further assistance, feel free to ask. Good luck with your project! 😊