

STM32F4 Emulator

User Manual

Version 0.10 – 29 October 2020

Version History

Ver.	Date	Emulator Features	User Manual update
0.1	3 Aug 2020	Initial version that supports GPIO	Initial version
0.2	18 Aug 2020	<ul style="list-style-type: none">Added SysTick, exception handling and GPIO EXTI functionality.Also corrected some machine instructions and implemented MULL and various other instructions.	<ul style="list-style-type: none">Added this tableNumbered sectionsAdded section 5.4 to explain SysTick limitations
0.3	25 Aug 2020	<ul style="list-style-type: none">Added RCC peripheral sectionAdded Timer functionality (timer2 to 5 only)Added UART functionality	<ul style="list-style-type: none">Added section 6 – internal supported peripherals (and moved SysTick section into this section)Added detail about Timer support (6.2) and USART supportAdded detail about Text console (section 5.4)
0.4	1 Sept 2020	<ul style="list-style-type: none">Corrected some machine instruction implementations<ul style="list-style-type: none">a. ADD relative to PCb. SUB.Wc. LDR with negative post-index offsetAdded ability to save and load emulator stateAdded ability to run, suspend and reset independent of GDB connection	<ul style="list-style-type: none">Added section 5 to describe toolbar options
0.5	1 Sept 2020	<ul style="list-style-type: none">Corrected ADD relative to PC machine instruction (caused some “switch” statements to fail)Improved stepping while debug with interrupts active	<ul style="list-style-type: none">None
0.6	6 Sept 2020	<ul style="list-style-type: none">Corrected error in PSR register when exception	<ul style="list-style-type: none">None

		occurred and CPU was busy with if-then-else block	
0.7		<ul style="list-style-type: none"> Added I2C peripheral functionality and “mouse velocity” I2C sensor Added SPI peripheral functionality and emulated SD card 	<ul style="list-style-type: none"> Added section 6.4 to explain mouse velocity I2C sensor Added section 7.4 to explain supported I2C functionality Added section 6.5 with detail about SD card Added section 7.5 with SPI details
0.8	4 Oct 2020	<ul style="list-style-type: none"> Added ADC and analog pad Added I2S and audio player Bug fixes to DMA, and I2S DMA requests 	<ul style="list-style-type: none"> Added section 6.2 with analog input pad details Added section 6.7 with details about audio player Added section 7.6 with DMA capabilities
0.9	7 Oct 2020	<ul style="list-style-type: none"> Added refresh signal from display 	<ul style="list-style-type: none"> Updated description in 6.3 to include refresh signal
0.10	29 Oct 2020	<ul style="list-style-type: none"> Implemented remaining 32-bit instructions 	-

1 Introduction

Modules in the Computer Systems stream at Stellenbosch University, such as Computer Systems 245 and Design (E) 314 require practical experience in developing programs for a microcontroller. Traditionally an STM32 development board, such as the STM32F411-Discovery, or Nucleo-F411RE was used. As part of newer initiatives to provide online tuition for these modules, the practical element had to adapt because practicals cannot be conducted in labs with the actual development boards.

The learning experience is however replaced with a software program which emulates all the functionality that would have been possible with a development board. The STM32IDE is still used to develop programs for the microcontroller, and programming and debugging occurs through the IDE. The “microcontroller” on which the developed code runs is however a Windows program that interprets the ARM binary machine code and mimics memory and peripherals as close as possible.

2 Minimum Requirements

The emulator requires Windows to run. It should run on Windows 7, 8, or 10. It also requires the .NET framework version 4.5 (this should be installed by default on Windows 10).

The emulator will use around 20MB of RAM and take up less than 1MB of hard disk space.

Note that the emulator currently works with STM32CubeIDE up to version 1.3, however it is **not compatible with version 1.4 of STM32CubeIDE**.

3 Installation

To install the emulator, simply unzip the file contents to a convenient location on your hard drive, and run it by double clicking the Emulator.exe file

4 Versions and Functionality

The emulator is still under development, and at present may not support all the required features for all future practicals. Updates to the emulator will be provided on SunLearn. You can view the version of the emulator that you are currently using, as well as the “Release Notes”, that lists the implemented features up to the current version, by clicking the “about” button on the top toolbar.



Figure 1 About button on main toolbar

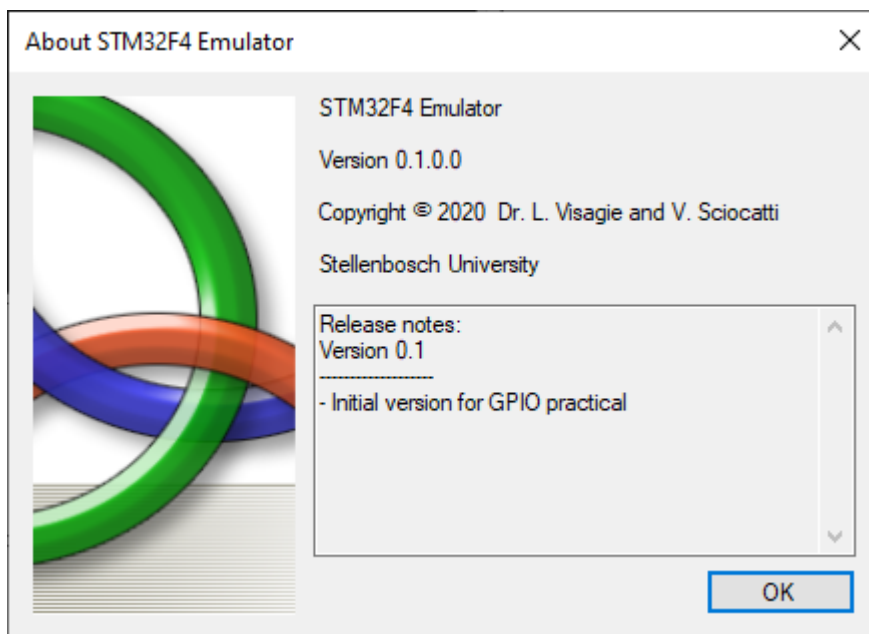







Figure 2 About window, showing version information and Release Notes

5 Toolbar Options

The following options are available from the top toolbar

	Load previously saved emulator state
	Save current emulator state
	Run CPU (only available if GDB is not connected)
	Suspend CPU (only available if GDB is not connected)
	Reset CPU (only available if GDB is not connected)

6 Emulated Hardware

The emulator emulates the STM32F411VE microcontroller. The emulation includes the same set of peripherals, SRAM and Flash memory as the actual microcontroller.

In addition, the following hardware elements have been “connected” to the microcontroller (this will be expanded on in future versions of the emulator)

- 8x user buttons and 4x LED indicators in the form of an arcade controller and joystick
- Memory-mapped 320x200 matrix display

- UART console
- Mouse cursor velocity sensor
- 16MB SD Card

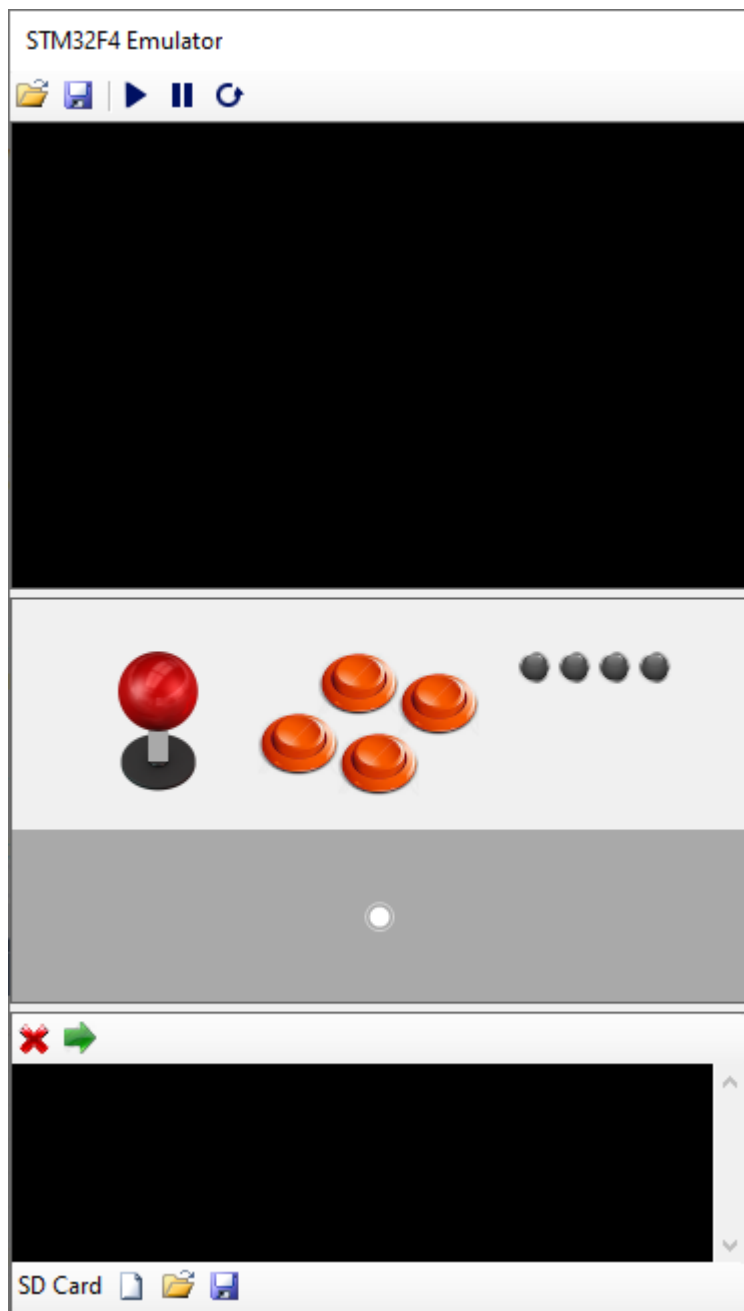


Figure 3 Emulator UI showing display (top), arcade controller (middle) and Text console (bottom)

6.1 Arcade controller

The arcade controller consists of 8x buttons and 4x LEDs.



Figure 4 Arcade controller

The four LEDs are simply numbered from 1 to 4, left to right. The four push-buttons are numbered as in Figure 4, and the joystick is implemented using four buttons – one button each for up, down, left and right. It is possible for the joystick to “push” two buttons at the same time, such as up and left.

The mapping of pushbuttons and LEDs to the GPIO pins of the microcontroller is indicated in Table 1 and Table 2.

Table 1 Button GPIO mapping

Button	GPIO port and pin	Keyboard button
Up	D8	W
Down	D11	S
Left	D10	A
Right	D9	D
1	A0	K
2	A1	O
3	A2	L
4	A3	P

Table 2 LED GPIO mapping

LED	GPIO port and pin	Colour
1	D12	Green
2	D13	Orange
3	D14	Red
4	D15	Blue

Pushbuttons will read as a logical 0 when they are not pushed, and as a logical 1 when pushed in, when reading the IDR – Input Data Register – of the corresponding GPIO port.

For convenience, the emulator also maps actual keyboard events to the button functionality, so that it is possible to create an interactive game or application without using the mouse to click the buttons. The keyboard keys that map to each button is also given in Table 1.

LEDs will be off when a logical 0 is written to the corresponding bit of the Output Data Register (ODR) and on when a logical 1 is written to the ODR bit.

6.2 Analog input pad

The analog input pad will produce an analog voltage signal corresponding to the location on the pad where the mouse is pressed. The X and Y coordinate of the analog pad position each separately produce an analog voltage. When the mouse button is not pressed, the output for both channels will read as $V_{ref}/2$. For the top-left position, the analog pad will output an X and Y V_{out} of 0 V, and for bottom-right position the pad will output V_{ref}, V_{ref} .



Figure 5 Analog input pad

The X and Y voltage outputs are connected to PC4 and PC5 of the emulated microcontroller.

6.3 Memory-mapped Display

The emulator has a matrix display that is mapped to the microcontroller memory, starting at address 0x2002 0000. (The 128kB SRAM starts at address 0x2000 0000 and ends at 0x2001 FFFF).

The display has a resolution of 320 x 200 pixels, and each pixel is mapped to a single byte in memory. The display uses a colour map or “palette” to find the corresponding pixel colour for the byte value in memory. The standard VGA colour map is used.

The colour map table with RGB values is given in the appendix.



Figure 6 VGA 256 default colour palette. Pixel value 0 is top-left (black) and pixel value 15 is top-right (white).

The display is refreshed at a rate of 10Hz. After each refresh, a pulse is generated on pin PB4

6.4 Text Console

The text console makes use of a UART connection to the microcontroller. It is connected to pin PB6 (TX from microcontroller, RX for the text console) and PB7 (RX of the microcontroller, TX from the text console) – this maps to the USART1 peripheral, but you have to select alternative pins. PB6 and PB7 are not the default TX and RX pins for USART1.

The text console uses the following UART setup:

Table 3 Text console UART settings

Mode	Asynchronous
Baud	115200
Stop bits	1
Parity	None
Data word length	8 bits

6.5 Mouse Cursor Velocity Sensor

The emulator implements a “mouse cursor velocity” sensor that is connected to the MCU via I2C. The sensor has no visual representation in the emulator GUI, however it will sample the movement of the mouse cursor and provide the cursor velocity as a measurement that can be requested from the MCU via I2C.

The sensor connections are shown in Table 4.

Table 4 Mouse Cursor Velocity sensor pin connections

Signal	MCU Pin
SCL	PB8
SDA	PB9

The 7-bit address for the sensor is 0011001₂. The 8-bit read and write addresses are given in Table 5

Table 5 Mouse Cursor Velocity sensor I2C address

Signal	Binary	Hexadecimal
8-bit I2C Read Address	0011 0011	33
8-bit I2C Write Address	0011 0010	32

The sensor supports I2C data rates in the range 100 kHz to 400 kHz.

6.5.1 Sensor Characteristics

The sensor measurement range and sensitivity is influenced by the FS (Full-Scale) setting in CTRL_REG4_A.

Table 6 Mouse Cursor Velocity sensor characteristics

Parameter	Condition	Value	Unit
Mouse cursor velocity measurement range	FS=00	+/- 3276	pixels/second
	FS=01	+/- 6553	
	FS=10	+/- 13107	
	FS=11	+/- 32767	
Mouse cursor velocity sensitivity	FS=00	0.1	Pixels/second per LSB
	FS=01	0.2	
	FS=10	0.4	
	FS=11	1.0	

6.5.2 Register Mapping

Table 7 Mouser Course Velocity sensor I2C register mapping

Register name	Type	Register address (hex)	Default/start-up value
CTRL_REG1_A	Read/write	20	0
CTRL_REG4_A	Read/write	23	0
OUT_X_L_A	Read-only	28	-
OUT_X_H_A	Read-only	29	-

OUT_Y_L_A	Read-only	2A	-
OUT_Y_H_A	Read-only	2B	-
OUT_Z_L_A	Read-only	2C	-
OUT_Z_H_A	Read-only	2D	-

6.5.3 Register Detail

6.5.3.1 CTRL_REG1_A (0x20)

7	6	5	4	3	2	1	0
ODR3	ODR2	ODR1	ODR0	-	Zen	Yen	Xen

Bits 7:4 **ODR[3:0]**: Output data rate:

0000: Power down mode

0001: 1 Hz

0010: 10 Hz

0011: 25 Hz

0100: 50 Hz

0101: 100 Hz

0110: 200 Hz

0111: 500 Hz

1xxx: reserved

Bit 2 **Zen**: Z-axis enable

Bit 1 **Yen**: Y-axis enable

Bit 0 **Xen**: X-axis enable

6.5.3.2 CTRL_REG4_A (0x23)

7	6	5	4	3	2	1	0
-	-	FS1	FS0	-	-	-	-

Bits 5:4 **FS[1:0]**: Full-Scale selection as per Table 6

6.5.3.3 OUT_X_L_A (0x28) and OUT_X_H_A (0x29)

X-axis mouse cursor velocity, expressed in 2's complement

6.5.3.4 OUT_Y_L_A (0x2A) and OUT_Y_H_A (0x2B)

Y-axis mouse cursor velocity, expressed in 2's complement

6.5.3.5 OUT_Z_L_A (0x28) and OUT_Z_H_A (0x29)

These registers will always return 0.

6.6 SD Card

The emulator includes a 16MB SD card, connected to the following MCU pins:

Table 8 SD card pin mapping

Signal	MCU Pin
SCK	PA5
MISO	PA6
MOSI	PA7

CS	PA15
----	------

The SD card only supports SPI mode.

The SD card memory is mapped to the host PC RAM. The SD card contents are preserved when the emulator is reset, either through starting a new debug session, or by pressing the reset button from the top toolbar.

However the SD card contents are not automatically persisted to hard drive. It is possible to load and save the SD card state using the buttons from the SD card toolbar, at the bottom of the emulator window. In this case the contents will be stored as a .ISO file on the host PC hard disk.



Figure 7 SD Card toolbar

At start-up, the SD card will be initialised, as if it was formatted with a FAT16 file system. Pressing the New button from the toolbar will restore the SD card to the formatted state, but will erase all other contents on the emulated card.

6.7 Audio player

The emulator includes an external audio player that is connected to the MCU through one of the I2C channels. The audio player accepts single channel, 16-bit data, sampled at 22050 Hz.

The audio player is connected to the following pins:

Table 9 Audio player I2S pin mapping

Signal	MCU Pin
WS	PA4
SD	PC12
CK	PC10

7 Internal Supported Peripherals

7.1 SysTick functionality

The SysTick on the STM32F4 normally causes an interrupt every 1ms, allowing for a software hook to increment a millisecond counter. The SysTick and 1 ms counter is used throughout the HAL code for things like timeouts and delays. For instance, if you use the function HAL_Delay, the implementation of this function will wait for the millisecond counter to reach the desired number of counts.

The emulator also implements the SysTick timer, but it was found that handling the SysTick interrupt (or any interrupt for that matter) at a real-time rate of 1 ms results in detrimental performance in the emulator. The emulator cannot handle interrupts that occur as often as 1 ms gracefully. The compromise that the emulator took is to scale the interrupt interval for the SysTick timer to be 10 times slower. This means that if the autogenerated code sets up the SysTick timer to interrupt every 1 ms (as is the default behaviour), the emulator's SysTick timer will only interrupt every 10 ms. This allows the emulator to still remain responsive and execute code normally.

The implication of this is that whenever you make use of the SysTick functionality for delays and timeouts in your code, you have to scale the delay or timeout value to be 10 times smaller. For

instance, calling HAL_Delay(50) in code that runs on the emulator will result in a delay of 500 ms. (On the actual hardware, it would have resulted in a delay of 50 ms).

This scaling only applies to the SysTick timer. All the timer peripherals will still be setup as normal. However it is not advised that you setup any timer to generate interrupts with period below 10 ms.

7.2 Timers

The emulator currently only implements Timers 2, 3, 4 and 5. The supported functionality is limited to setting up a timer to count (up or down), and to generate an interrupt on the timer update event.

7.3 USART

The emulator implements USART1, 2 and 6 but only in asynchronous mode. Interrupts in transmit mode are not supported, however the RXNE interrupt is supported.

7.4 I2C

The emulator implements only the I2C1 peripheral in master transmitter and master receiver mode. It does not implement any I2C interrupts.

7.5 SPI

SPI1 is supported in Master Full-Duplex mode. The SPI peripheral does not implement interrupts.

7.6 DMA

The emulator supports DMA, however only direct modes. FIFO operation is not supported. Also, MSIZE and PSIZE have to be the same. DMA requests from the following peripherals are supported:

- USART1 TX
- SPI3/I2S3 TX

8 IDE Project Setup

In order to create a new STM32IDE project to use with the emulator, you would use the normal STM32 new project wizard. In the target selection window that appears, **make sure you use the MCU/MPU selector tab (and not the Board selector tab)**. Type in STM32F411VE for the part number.

Select 'Next' and continue setting up the project location with default options.

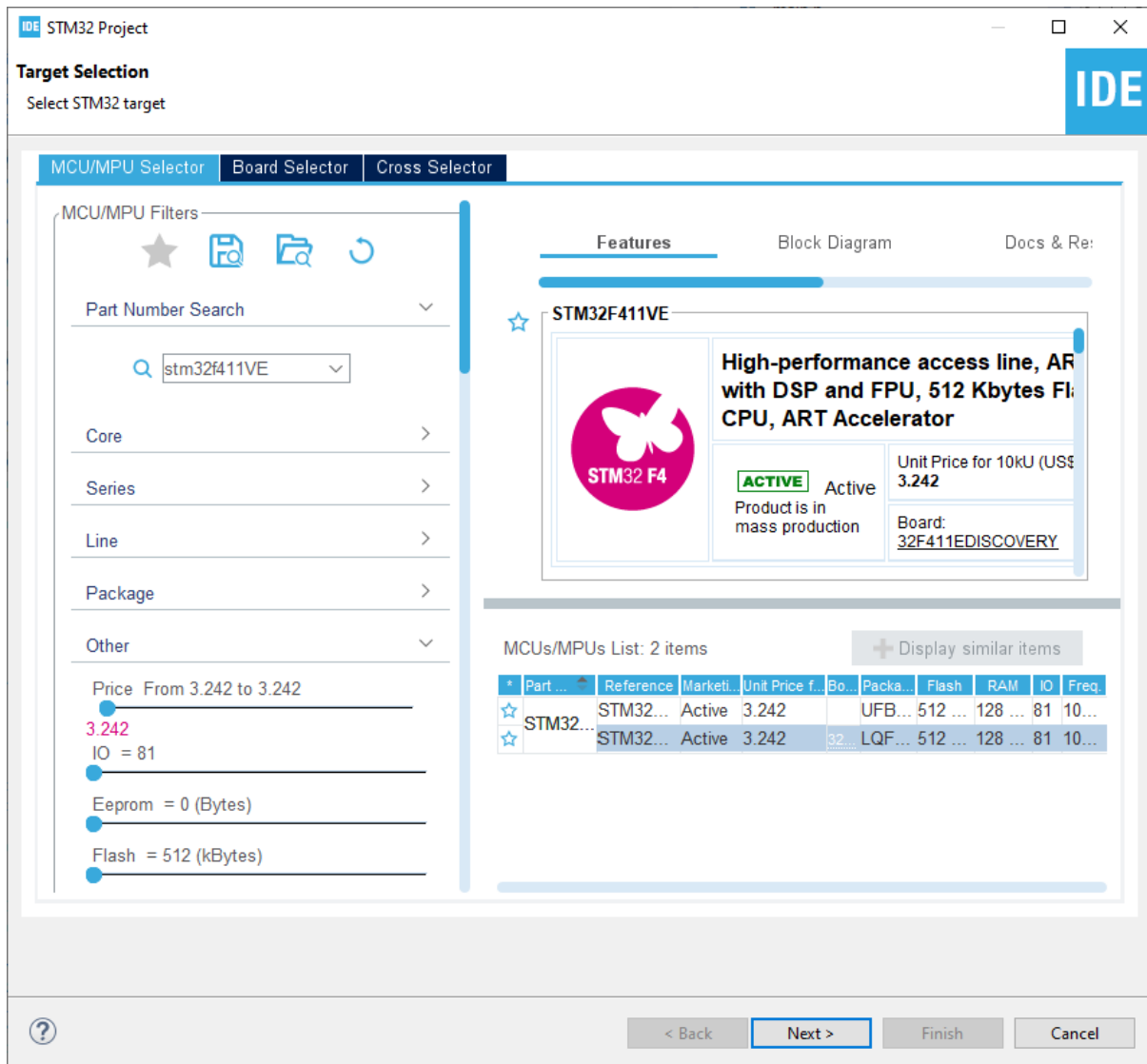



Figure 8 STM32 new project wizard - target selection window

8.1 Debugging

In order to tell the IDE to program and debug using the emulator (as opposed to the physical hardware), you have to make a single modification to the 'Debug launch configuration'. The debug launch configuration window should open up the very first time that you press the debug button () in the IDE, however it can also be changed later by pressing the drop-down arrow next to the debug button on the toolbar, and selection 'Debug Configurations...'.

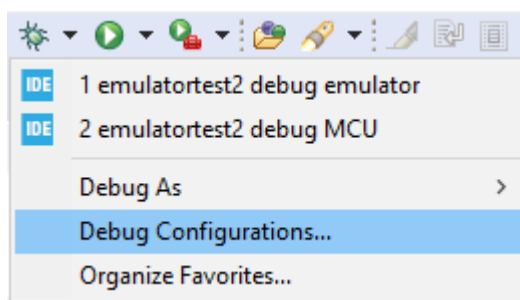


Figure 9 Menu option to open the Debug Configurations window

In the Debug Configurations window, select the 'Debugger' tab, and select the option that says 'Connect to remote GDB server'. Also change the port number entry to 10000.

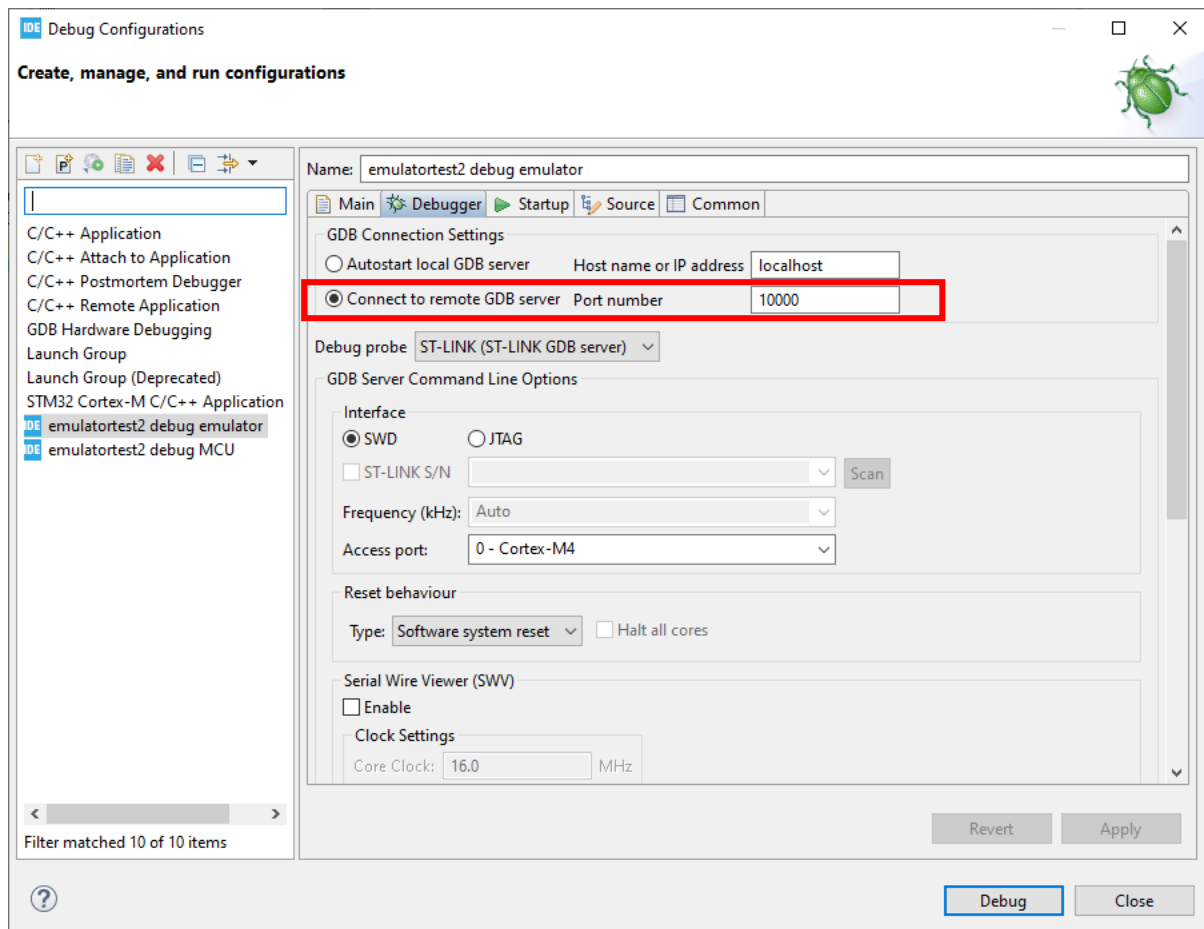


Figure 10 Debug Configuration settings to use with emulator

After doing this setup, (and starting the emulator), you should be able to launch and debug the program in exactly the same way that you would with the actual physical hardware.

9 Appendix

Table 10 VGA index to RGB colour map

Colour index	Red	Green	Blue	Colour index	Red	Green	Blue	Colour index	Red	Green	Blue	Colour index	Red	Green	Blue
0	0	0	0	64	255	130	130	128	57	57	113	192	3	65	0
1	0	2	170	65	255	158	130	129	69	57	113	193	3	65	16
2	20	170	0	66	255	190	130	130	85	57	113	194	2	65	32
3	0	170	170	67	255	223	130	131	97	57	113	195	1	65	49
4	170	0	3	68	255	255	130	132	113	57	113	196	0	65	65
5	170	0	170	69	223	255	130	133	113	57	97	197	0	49	65
6	170	85	0	70	190	255	130	134	113	57	85	198	0	32	65
7	170	170	170	71	158	255	130	135	113	57	69	199	0	16	65
8	85	85	85	72	130	255	130	136	113	57	57	200	32	32	65
9	85	85	255	73	130	255	158	137	113	69	57	201	40	32	65
10	85	255	85	74	130	255	190	138	113	85	57	202	49	32	65
11	85	255	255	75	130	255	223	139	113	97	57	203	57	32	65
12	255	85	85	76	130	255	255	140	113	113	57	204	65	32	65
13	253	85	255	77	130	223	255	141	97	113	57	205	65	32	57
14	255	255	85	78	130	190	255	142	85	113	57	206	65	32	49
15	255	255	255	79	130	158	255	143	69	113	58	207	65	32	40
16	0	0	0	80	186	186	255	144	57	113	57	208	65	32	32
17	16	16	16	81	202	186	255	145	57	113	69	209	65	40	32
18	32	32	32	82	223	186	255	146	57	113	85	210	65	49	32
19	53	53	53	83	239	186	255	147	57	113	97	211	65	57	33
20	69	69	69	84	254	186	255	148	57	113	113	212	65	65	32
21	85	85	85	85	254	186	239	149	57	97	113	213	57	65	32
22	101	101	101	86	255	186	223	150	57	85	113	214	49	65	32
23	117	117	117	87	255	186	202	151	57	69	114	215	40	65	32
24	138	138	138	88	255	186	186	152	81	81	113	216	32	65	32
25	154	154	154	89	255	202	186	153	89	81	113	217	32	65	40
26	170	170	170	90	255	223	186	154	97	81	113	218	32	65	49
27	186	186	186	91	255	239	186	155	105	81	113	219	32	65	57
28	202	202	202	92	255	255	186	156	113	81	113	220	32	65	65
29	223	223	223	93	239	255	186	157	113	81	105	221	32	57	65
30	239	239	239	94	223	255	186	158	113	81	97	222	32	49	65
31	255	255	255	95	202	255	187	159	113	81	89	223	32	40	65
32	0	4	255	96	186	255	186	160	113	81	81	224	45	45	65
33	65	4	255	97	186	255	202	161	113	89	81	225	49	45	65
34	130	3	255	98	186	255	223	162	113	97	81	226	53	45	65
35	190	2	255	99	186	255	239	163	113	105	81	227	61	45	65

36	253	0	255	100	186	255	255	164	113	113	81	228	65	45	65
37	254	0	190	101	186	239	255	165	105	113	81	229	65	45	61
38	255	0	130	102	186	223	255	166	97	113	81	230	65	45	53
39	255	0	65	103	186	202	255	167	89	113	81	231	65	45	49
40	255	0	8	104	1	1	113	168	81	113	81	232	65	45	45
41	255	65	5	105	28	1	113	169	81	113	90	233	65	49	45
42	255	130	0	106	57	1	113	170	81	113	97	234	65	53	45
43	255	190	0	107	85	0	113	171	81	113	105	235	65	61	45
44	255	255	0	108	113	0	113	172	81	113	113	236	65	65	45
45	190	255	0	109	113	0	85	173	81	105	113	237	61	65	45
46	130	255	0	110	113	0	57	174	81	97	113	238	53	65	45
47	65	255	1	111	113	0	28	175	81	89	113	239	49	65	45
48	36	255	0	112	113	0	1	176	0	0	66	240	45	65	45
49	34	255	66	113	113	28	1	177	17	0	65	241	45	65	49
50	29	255	130	114	113	57	0	178	32	0	65	242	45	65	53
51	18	255	190	115	113	85	0	179	49	0	65	243	45	65	61
52	0	255	255	116	113	113	0	180	65	0	65	244	45	65	65
53	0	190	255	117	85	113	0	181	65	0	50	245	45	61	65
54	1	130	255	118	57	113	0	182	65	0	32	246	45	53	65
55	0	65	255	119	28	113	0	183	65	0	16	247	45	49	65
56	130	130	255	120	9	113	0	184	65	0	0	248	0	0	0
57	158	130	255	121	9	113	28	185	65	16	0	249	0	0	0
58	190	130	255	122	6	113	57	186	65	32	0	250	0	0	0
59	223	130	255	123	3	113	85	187	65	49	0	251	0	0	0
60	253	130	255	124	0	113	113	188	65	65	0	252	0	0	0
61	254	130	223	125	0	85	113	189	49	65	0	253	0	0	0
62	255	130	190	126	0	57	113	190	32	65	0	254	0	0	0
63	255	130	158	127	0	28	113	191	16	65	0	255	0	0	0