Getting Started

# Get the IDE

Go here:

<https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html>

Get the version for your environment (I’m using Windows, Brian is using a Mac).

See the video on this page if you’re stuck.

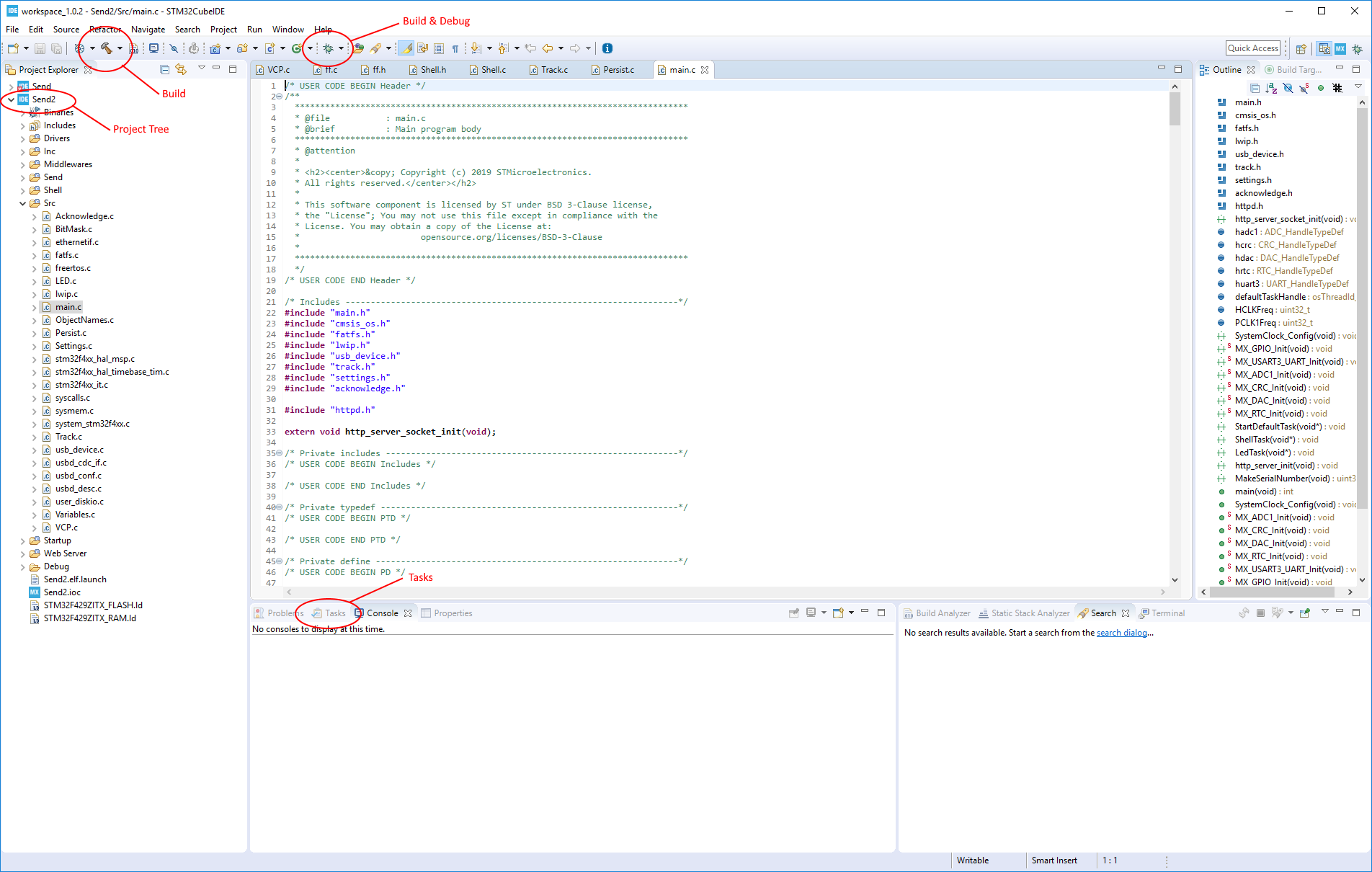
Open the IDE. Let it create the default workspace.

## Get the project

Copy the project into the workspace root.

Go to File – Open Projects from Filesystem… and select Send2 (that name can be changed, the 2 was just to differentiate it from the project for the old Nucleo board). Maybe it should be 4.

It should load. Initially the project tree will be collapsed – click the down arrow to expand it. You can then double-click any source file to open it.



Build – press this to build the project.

Debug – press this to Build, Download, and switch to the Debug perspective. You can now set breakpoints and single step through the code. There is an ‘issue’ with the IDE that the ‘run button’ gets enabled long before the debugger is ready to run. Wait for the first line of code in main to be highlighted before you push run.

Tasks – any comment in the code with key words, like “todo”, “fixme”, “ToDo”, show up in this pane. There are a bunch in the source for the libraries, so if we all agree to a common spelling (I’m using “ToDo” now), the rest on them can be filtered out. I’m hoping to have an issue tracker soon, which *may* hook into this.

Console (next to Tasks) – this is the IDE output window.

Terminal – there is a way to have the serial terminal in one of the panes in the IDE, it’s just not that easy to setup, so I’m just using teraterm (or just about any serial terminal program).

## Project Tree

08/01/2019 11:27 AM <DIR> .settings

08/16/2019 06:32 AM <DIR> Debug

08/01/2019 11:27 AM <DIR> Drivers

08/01/2019 11:27 AM <DIR> CMSIS

08/01/2019 11:27 AM <DIR> STM32F4xx\_HAL\_Driver

08/06/2019 06:50 AM <DIR> Inc

07/28/2019 05:09 PM 404 Acknowledge.h

07/23/2017 05:50 PM 283 BitMask.h

08/06/2019 12:01 AM 1,815 ethernetif.h

08/06/2019 12:01 AM 1,534 fatfs.h

08/06/2019 12:01 AM 12,369 ffconf.h

08/13/2019 12:28 PM 6,710 FreeRTOSConfig.h

08/06/2019 12:01 AM 2,129 lwip.h

08/06/2019 12:01 AM 5,069 lwipopts.h

08/10/2019 04:37 PM 3,909 main.h

08/11/2019 11:53 AM 394 ObjectNames.h

08/06/2019 08:40 AM 1,286 Persist.h

08/06/2019 11:49 AM 866 Settings.h

08/06/2019 12:01 AM 16,408 stm32f4xx\_hal\_conf.h

08/06/2019 12:01 AM 2,127 stm32f4xx\_it.h

08/11/2019 11:39 AM 1,114 Track.h

08/06/2019 12:01 AM 2,825 usbd\_cdc\_if.h

08/06/2019 12:01 AM 4,109 usbd\_conf.h

08/06/2019 12:01 AM 3,158 usbd\_desc.h

08/06/2019 12:01 AM 2,515 usb\_device.h

08/06/2019 12:01 AM 1,624 user\_diskio.h

08/06/2019 08:40 AM 560 Variables.h

07/22/2019 12:32 PM 1,538 VCP.h

08/01/2019 11:33 AM <DIR> Middlewares

08/16/2019 06:31 AM <DIR> ST

08/16/2019 06:31 AM <DIR> STM32\_USB\_Device\_Library

08/16/2019 06:31 AM <DIR> Third\_Party

08/16/2019 06:31 AM <DIR> FatFs

08/16/2019 06:31 AM <DIR> FreeRTOS

08/16/2019 06:31 AM <DIR> LwIP

08/01/2019 12:02 PM <DIR> Send

08/01/2019 12:02 PM <DIR> inc

07/30/2019 11:54 AM 7,137 BITS.h

06/06/2019 02:43 PM 1,611 CKSUM.h

06/06/2019 02:43 PM 11,406 DCC.h

06/06/2019 02:43 PM 6,799 ZLOG.h

06/06/2019 02:43 PM 4,398 ZTYPES.h

06/06/2019 02:43 PM 5,286 Z\_CORE.h

08/01/2019 12:02 PM <DIR> src

07/30/2019 11:43 AM 30,320 ARGS.cpp

07/31/2019 10:50 AM 7,204 ARGS.h

07/30/2019 12:05 PM 133,822 DEC\_TST.cpp

07/30/2019 12:01 PM 9,981 DEC\_TST.h

07/30/2019 09:07 AM 6,221 PDS601.h

08/10/2019 03:31 PM 3,016 port.c

07/31/2019 07:37 AM 1,058 port.h

07/30/2019 11:41 AM 26,254 SELF\_TST.cpp

07/30/2019 11:42 AM 5,508 SELF\_TST.h

07/31/2019 01:30 PM 45,035 SEND.cpp

07/31/2019 11:52 AM 10,321 SEND.h

07/30/2019 11:41 AM 54,367 SEND\_REG.cpp

07/31/2019 11:45 AM 8,364 SEND\_REG.h

07/30/2019 11:41 AM 5,693 SR\_CORE.cpp

06/06/2019 02:44 PM 3,854 SR\_CORE.h

07/30/2019 11:42 AM 40,547 ST\_VECT.cpp

06/06/2019 02:44 PM 1,590 ST\_VECT.h

07/30/2019 11:42 AM 11,124 TEST\_REG.cpp

07/30/2019 11:41 AM 5,529 TEST\_REG.h

06/06/2019 02:44 PM 2,330 T\_STAT.h

08/06/2019 06:46 AM <DIR> Shell

07/19/2019 11:33 AM 4,834 Ansi.c

09/27/2017 06:02 PM 1,246 Ansi.h

07/19/2019 11:33 AM 1,490 GetLine.c

02/19/2017 10:04 PM 219 GetLine.h

08/13/2019 12:51 PM 27,029 Shell.c

08/06/2019 06:53 AM 2,812 Shell.h

07/19/2019 06:29 PM 9,582 ShellFile.c

02/26/2017 06:58 PM 1,318 ShellFile.h

07/19/2019 06:30 PM 10,792 ShellScript.c

05/07/2018 09:37 PM 1,270 ShellScript.h

11/14/2017 07:41 PM 3,317 strsep.c

08/06/2019 06:41 AM <DIR> Src

07/28/2019 05:08 PM 8,848 Acknowledge.c

07/23/2017 05:50 PM 795 BitMask.c

08/06/2019 12:01 AM 22,934 ethernetif.c

08/06/2019 12:01 AM 1,666 fatfs.c

08/06/2019 12:01 AM 2,085 freertos.c

08/10/2019 04:37 PM 953 LED.c

08/10/2019 02:57 PM 5,200 lwip.c

08/13/2019 12:52 PM 17,529 main.c

08/11/2019 11:50 AM 1,876 ObjectNames.c

07/28/2019 06:40 AM 12,540 Persist.c

08/06/2019 08:37 AM 3,803 Settings.c

08/06/2019 12:01 AM 9,024 stm32f4xx\_hal\_msp.c

08/06/2019 12:01 AM 4,165 stm32f4xx\_hal\_timebase\_tim.c

08/06/2019 12:01 AM 5,915 stm32f4xx\_it.c

08/01/2019 11:27 AM 4,058 syscalls.c

08/01/2019 11:27 AM 2,851 sysmem.c

08/01/2019 11:16 AM 28,181 system\_stm32f4xx.c

08/11/2019 11:43 AM 12,615 Track.c

08/10/2019 04:15 PM 8,881 usbd\_cdc\_if.c

08/06/2019 12:01 AM 19,578 usbd\_conf.c

08/06/2019 12:01 AM 12,511 usbd\_desc.c

08/06/2019 12:01 AM 2,657 usb\_device.c

08/06/2019 12:01 AM 5,179 user\_diskio.c

08/13/2019 12:31 PM 20,163 Variables.c

07/31/2019 07:20 AM 1,313 VCP.c

08/01/2019 11:27 AM <DIR> Startup

08/13/2019 12:34 PM <DIR> Web Server

08/02/2019 11:26 AM 4,263 httpd\_structs.h

08/12/2019 07:33 PM 3,515 WebServer.c

08/08/2019 12:19 PM 22,244 WebServer2.c

08/16/2019 06:31 AM 85,699 .cproject

08/06/2019 12:02 AM 47,139 .mxproject

08/06/2019 06:51 AM 1,113 .project

08/13/2019 12:18 PM 5,493 Send2.elf.launch

08/06/2019 12:02 AM 12,445 Send2.ioc

08/12/2019 07:57 PM 6,191 STM32F429ZITX\_FLASH.ld

08/06/2019 12:02 AM 6,059 STM32F429ZITX\_RAM.ld

In general, source files with a lower case first letter are generated by CubeMX.

The .ioc file will open the CubeMX perspective for major project changes. It has a code generator and will alter the project, so for now, you can look, but don’t save. I haven’t done this in a while, and I am not sure the results. I did notice that some of the extra folders (Send, Shell, Web Server), get excluded from the project and have to be re-included (more on this later).

# Build

Click the build button. The project should build without error. There are some warnings that I will take care of after I add the code for the sender board. GOAL: warning-less compile.

## Include/Exclude Files and Folders

You may notice that some of the Send files don’t build. The are ‘excluded’ from the project. To include them, right click the folder and choose Resource Configurations – Exclude From Build… If the boxes are checked, the folder is excluded (I see 4 instead of 2 sometimes and I don’t know why – it worked OK though). Deselect All and press OK. This same procedure works for individual c/cpp file as well.

# Shell

The shell is extensible, and you may want to add a command to make testing easier or add features. The new command may be a permanent addition or may be removed once you are done with it. Since it may be hidden from normal users, it can stay as long as you like.

### Write the Command

The commands are C functions with the following prototype:

CMD\_RETURN **Sh[name]**(uint8\_t bPort, **int** argc, **char** \*argv[]);

It is recommended to start the name with “Sh”, but it isn’t mandatory.

* argc and argv work just like the dos equivalent with argv[0] the name of the command.
* bPort is the port which initiated the command, it isn’t important to know who, just pass this to the console output methods. We may have to change the meaning of this in the future to support multiple Telnet connections.

Return on one the following codes:

*CMD\_OK*

*CMD\_BAD\_PARAMS*

*CMD\_NOT\_FOUND*

*CMD\_FAILED*

*CMD\_CANCEL\_KEY*

*CMD\_BAD\_NUMBER*

*CMD\_READ\_ONLY*

We can add more if necessary.

Add the command to the table:

**typedef** **struct** SHELL\_TABLE\_t

{

**char** szCommand[11]; // command string

**unsigned** **char** bClass; // command class – for help

**unsigned** **char** bFlags; // 'help' control flags

// pointer to function

CMD\_RETURN (\*Command)(uint8\_t bPort, **int** argc, **char** \*argv[]);

**char**\* pszHelp; // help string

} SHELL\_TABLE;

Example:

{"help", *CL\_SYS*, NO\_FLAGS, ShHelp, "command help"},

The flags include:

NO\_FLAGS Help will list this command

VALUE\_RETURN This was to let a script return a value – support for shell variables is not implemented yet.

SCRIPT\_HELP This command is meant for scripts

It is only listed for “help script” and “help all”

SUPPRESS\_HELP This will only be listed for “help all”

The Class includes:

*CL\_NONE No specific class*

*CL\_ANSI A command which uses ANSI escape sequences to control the terminal*

*CL\_FILE A file system command*

*CL\_SCRIPT Commands especially for scripting*

*CL\_CS Command Station commands*

*CL\_SYS System commands*

*CL\_TEST Decoder test commands*

## Shell Output

**void** **ShCharOut**(uint8\_t port, **char** c);

Output 1 character to ‘port’.

**void** **ShBuffOut**(uint8\_t port, **char**\* s, **int** len);

Output a set of characters to ‘port’.

**void** **ShStringOut**(uint8\_t port, **char**\* s);

Output a string to ‘port’.

**void** **ShFieldOut**(uint8\_t port, **char**\* szBuffer, **int** iFieldWidth);

**void** **ShFieldNumberOut**(uint8\_t bPort, **char**\* szBuffer, **int** number, **int** iFieldWidth);

These last two are the primary methods for the output of shell commands, the FieldWidth parameter makes it easy to keep output data in columns. Pass in 0 for no width. The Number variant outputs the buffer and then converts the number to a string and outputs it.

**void** **ShNL**(uint8\_t bPort);

If using the two methods above, use this for a new line.

**void** **shprintf**(uint8\_t port, **const** **char** \*fmt, ... );

A printf variant to output to a ‘port’.

**char** **ShGetChar**(uint8\_t port);

It is possible for a shell command to stay in the command and wait for user input, but at the moment, that command will block the shell for all users. If we need this, that functionality can be added. The ‘send’ command will call non-blocking input methods but may still need thread management.

Sample Shell Command

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* ShArgs

\*

\* @catagory Shell Command

\* @brief A Shell command that just echos the argument list

\*

\* @param bPort - port that issued this command

\* argc - argument count

\* argv - argc array of arguments

\*

\* @return CMD\_RETURN - shell result

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMD\_RETURN **ShArgs**(uint8\_t bPort, **int** argc, **char** \*argv[])

{

ShNL(bPort);

ShFieldNumberOut(bPort, "Command Arguments: ", argc, 0);

**for**(**int** i = 0; i < argc; i++)

{

ShNL(bPort);

ShFieldOut(bPort, argv[i], 0);

}

**return** *CMD\_OK*;

}

# Variables

Variables are extensible. Variables allow you to have changeable, persistent settings. They can be accesses and used in comparisons in shell scripts.

In ObjectNames.c add a string and initialize it (the convention is lower case)

**const** **char** szPort[] = "port";

In ObjectNames.h add an extern to the string

**extern** **const** **char** szPort[];

In Variables.c add the variable. Add the string, variable, and action to the Variables table

uint32\_t CabPort;

In Variable.h add an extern to the variable.

**extern** uint32\_t CabPort;

If the new variable does not have defined type, support must be added to the following:

**char**\* **VarToString**(uint32\_t idx)

**int** **SetVariable**(**const** **char**\* szObject, **char**\* pStrValue)

If it is persistent and a type that is not an int or a string, support must be added to the following in Persist.c & Persist.h:

add a GetProfileXXX and a WriteProfileXXX

## Variables Table

**#define** VAR\_TYPE\_INT 1

**#define** VAR\_TYPE\_STRING 2

**#define** VAR\_TYPE\_VER 3

**#define** VAR\_TYPE\_TIME 4

**#define** VAR\_TYPE\_DATE 5

**#define** VAR\_TYPE\_12\_24 6

**#define** VAR\_TYPE\_ON\_OFF 7

**#define** VAR\_TYPE\_PORT 8

**#define** VAR\_TYPE\_PORT 9

**#define** VAR\_TYPE\_IP 10

**#define** VAR\_TYPE\_LED 11

**#define** VAR\_TYPE\_INPUTS 12

**#define** VAR\_TYPE\_LOOP\_CNT 13

**#define** VAR\_TYPE\_TRACK 20

**#define** VAR\_TYPE\_MAX 63

**#define** VAR\_TYPE\_MASK 0x3f

**#define** VAR\_TYPE\_PERSIST 0x100

**#define** VAR\_TYPE\_READ\_ONLY 0x200

**#define** VAR\_TYPE\_BAT\_BACKED 0x400 // not implemented yet

**typedef** **struct** var\_table\_t

{

**const** **char**\* szCmdString;

**void**\* var;

uint32\_t type;

**char**\* pszDefault;

**char**\* pszHelp;

} VAR\_TABLE;

{szPort, &CabPort, (VAR\_TYPE\_PORT | VAR\_TYPE\_PERSIST), “NCE”, "Port = NCE | XpressNet" },

The variable is now accessible in the shell and help will list the name and ‘variables’ will display the names and values of all the variables. They may also be used is scripts. Variables with named outputs in the shell, like the above ‘port’, will be numeric. Entering help or ? And the variable name will display the help string.