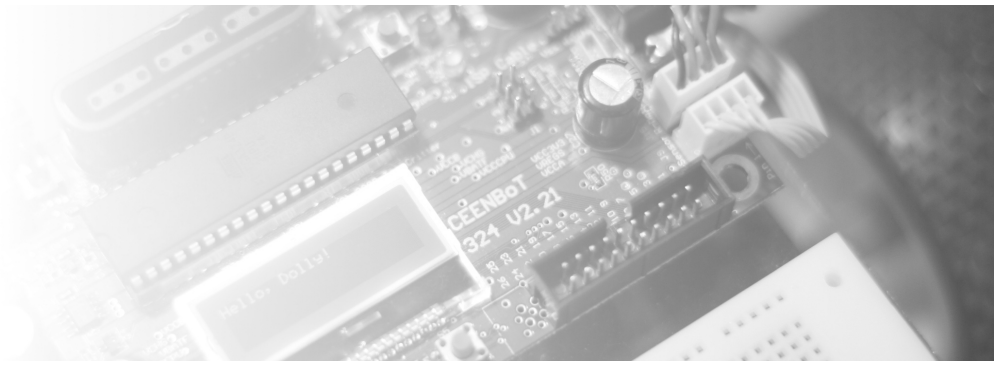
The CEENBoT-API: Getting Started

Tutorial on getting started with the CEENBoT-API on the CEENBoT v2.21 – '324 Platform



Written by ***Jose Santos***,
CEENBoT-API Creator and Developer

Undergraduate Student, Department of Computer and Electronics Engineering (CEEN)
University of Nebraska-Lincoln (Omaha Campus)

Rev 1.07
(Current as of: **v1.09.000R**)

(Blank)

Introduction

The CEEBoT-API is an *application programming interface* that exposes a set of functions which allow you to control and manipulate the CEENBoT in a a simplified manner. Its purpose is not necessarily to completely replace 'bare-metal' programming of the 'BoT, but simply to provide an optional set of tools that allows the end-user to explore the CEENBoT and its capabilities in a more friendly, inviting, and open-ended manner. This opens up the CEENBoT for exploration at multiple skill levels for those who do not wish to be bothered with the intricate details of the CEENBoT's electronics.

The CEENBoT-API exposes a rich set of C functions that allows usage of the various hardware resources available on the CEENBoT itself through simple, well-documented function calls. These set of functions allow the manipulating of peripherals some of which include graphic LCD display, on-board LEDs, stepper motors and more. The API function library is exposed to the user by way of a pre-compiled *static library*. The user merely needs to include the appropriate header files and link against this static library to take advantage of the rich set of functions.

This 'getting started' document is an important first step on beginning to explore the capabilities of the CEENBoT-API. It walks you through a tutorial on how to write programs that run on the CEENBoT which take advantage of the CEENBoT-API. It is *very* important that you read it from beginning to end. The knowledge and experience you gain in this tutorial will get you started to writing your own CEENBoT programs that take advantage of the API. This document is the *spring board* to programming with the API, so if this is your first time, then *this* document is where it all begins. Consequently, it is important that you read this document through to completion.

Requirements

This document assumes you have the following items installed – this document does *not* walk you through the installation of these items – you must consult the installation instruction for any respective software. Therefore, before you begin, you *must* have the following:

- You need to have installed the latest *WinAVR-GCC* tool-chain on your system. At the time this document was being written, you can obtain the *WinAVR* toolchain here:

  http://winavr.sourceforge.net/

- You need to have installed *AVR Studio 4* from *Atmel.* At the time this document was being written, you can obtain *AVR Studio 4* here:

  http://www.atmel.com

  The software is free, but you have to register and posses a valid e-mail address before you are allowed to download. Make sure you also download the *service pack* upgrades.

- You need to download the *static library* and corresponding *header files* for the CEENBoT-API and extract then into a *known* location in your system.

  Presently, you can download the library and header files from the *CEENBoT Portal* here:

  http://ceenbot.digital-brain.info

- It goes without saying – you need your CEENBoT – with the 324 board v2.21 along with a suitable In-System Programmer (or ISP, or AVR-ISP).

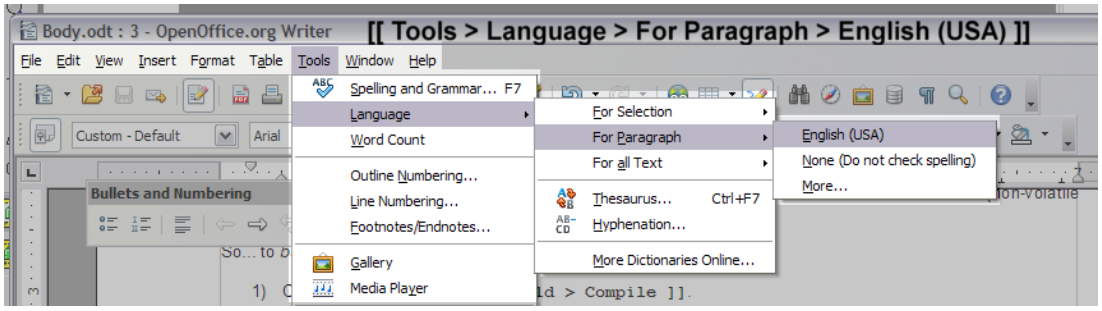
Document Conventions

This document uses the following typographical conventions:

- Code is written using `Lucida Console` type font. It is typically shown as follows:

```
void CBOT_main( void )
{
 // … code here;
} // end CBOT_main()
```

- Sometimes you'll be instructed to access a particular feature by *clicking* your way through a set of menus. For example – take a look at the figure below – it is a screen-shot of a completely unrelated program called *Open Office*:



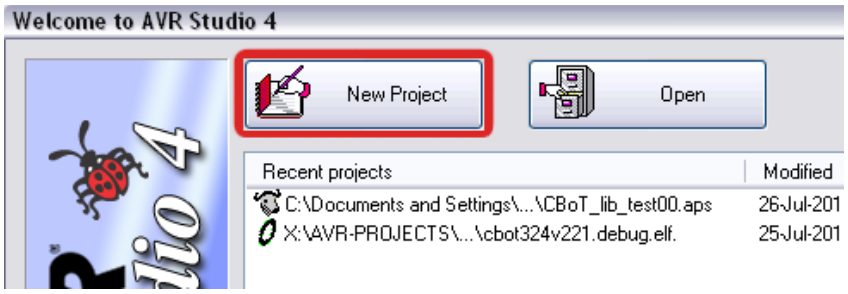
I might describe the above operations using the following notation (without the benefit of the image):

"Go to the top menu and select:

`[[ Tools > Language > For Paragraph > English (USA) ]]`*".*

In particular, note how *menu-driven* directions are enclosed in double square brackets.

- Some panels have *buttons*, which I will instruct you to *click on*. For example, the panel below has a button called *New Project*:



I might describe the following operation:

"In the "***Welcome to AVR Studio 4***" panel, click on **`[ New Project ]`**.

In particular, note that *panel buttons* directions are enclosed in single square brackets.

- In general when I want to attract your attention to an area on a panel or window on the software you're looking at, I will use `courier new` type font to describe such an area. It means you should stop looking at this document and start looking for the specified feature on your computer screen. For example. in the same above figure I might say, “under `Recent projects`, blah, blah”, etc.

Comments, Questions and/or Suggestions...

Comments, questions and/or suggestions should be addressed by e-mail to:

ceenbot.api@digital-brain.info

Check out the *CEENBoT Portal* for latest development news regarding the API:

http://ceenbot.digital-brain.info

WARNING: Before You Begin...

Your CEENBoT may have arrived in your hands with a pre-programmed firmware that showcases some basic functionality. More importantly, this functionality includes *power management* and *battery charging* capability. This capability is NOT yet included in the API. It is *very* important that you either have a backup, or have a copy of the original HEX file of the original CEENBoT firmware before you start writing programs with this API.

At the time this document was being written, the latest firmware can be obtained here:

http://www.ceenbotinc.com/tools/

You need to understand that if you wish to restore your CEENBoT to factory settings – for example, you want to use the CEENBoT to charge your battery after you've done experimenting with the CEENBoT-API – that you need to r*e-flash* your CEENBoT with the factory *firmware* (i.e., the aforementioned HEX file). It is assumed that the end-user understands how to perform this procedure.

with a that said, the following warnings should be taken seriously.

WARNING

Keep a backup of your original *firmware* and know how to *re-flash* your CEENBoT BEFORE YOU BEGIN EXPERIMENTING WITH THE CEENBoT-API!

Finally, and *most* importantly:

NO WARRANTY

THIS PROGRAM (“THE CEENBOT API”) or simply (“API”) IS DISTRIBUTED IN THE HOPE THAT IT WILL BE USEFUL, BUT WITHOUT ANY WARRANTY. IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW THE AUTHOR WILL BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA AND/OR EQUIPMENT OR DATA/EQUPMENT BEING RENDERED INACCURATE OR USELESS OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM OR DEVICEE TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

YOUR USE OF THIS “API” CONSTITUTES YOUR AGREEMENT AND UNDERSTANDING OF THE 'NO-WARRANTY' CLAUSE.

Tutorial Description

We begin this tutorial on how to write programs that take advantage of the CEENBoT-API by creating a simple "*Hello, Dolly*" application that displays this very same message on the CEENBoT's on-board LCD display. Remember that if this is your first time using the API, it is important that you complete the tutorial in its entirety. What you learn from the tutorial will be useful in helping you write your own CEENBoT API-based programs. The tutorial is divided in several parts.

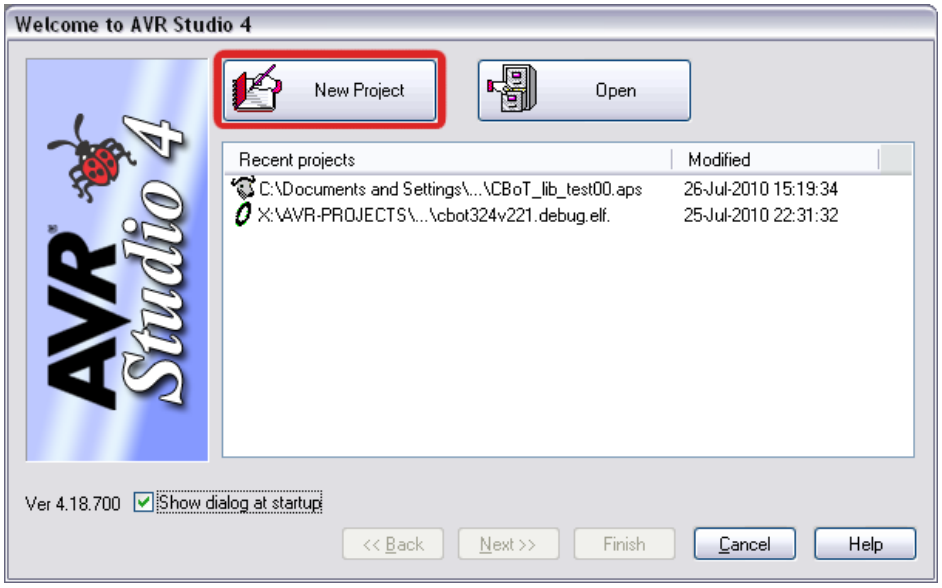
- **Part I**: Starting a new project in *AVR Studio 4*.
- **Part II**: Configuring and setting up the project options for properly compiling and linking against the static library.
- **Part III**: Writing the program and what the proper *program structure* is for a CEENBoT-API based program.
- **Part IV**: Building the program to produce the HEX file that is eventually uploaded to the CEENBoT.
- **Part V**: Uploading the program to the CEENBoT.

Let us now begin.

Part I: Starting a *new* Project

1) Start *AVR Studio 4*.

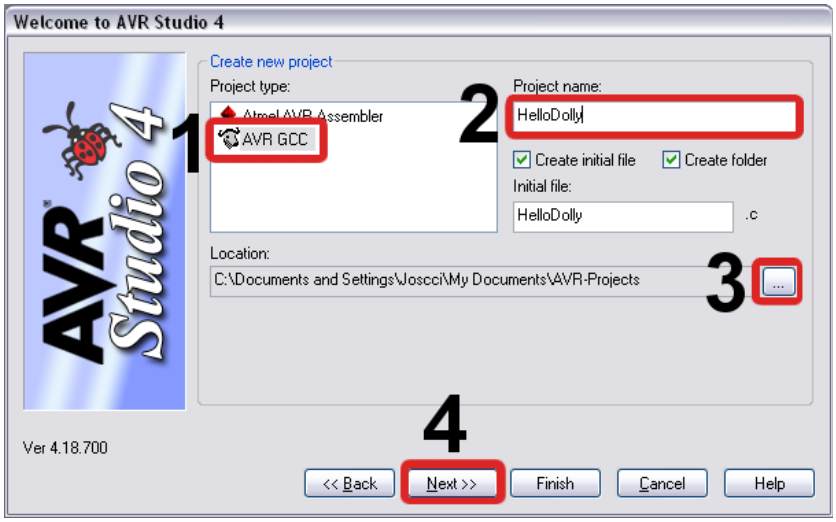
2) The "***Welcome to AVR Studio 4***" panel *might* automatically pop-up; (if it doesn't, select **`[[ Project > Project Wizard ]]`** via the top menu.)



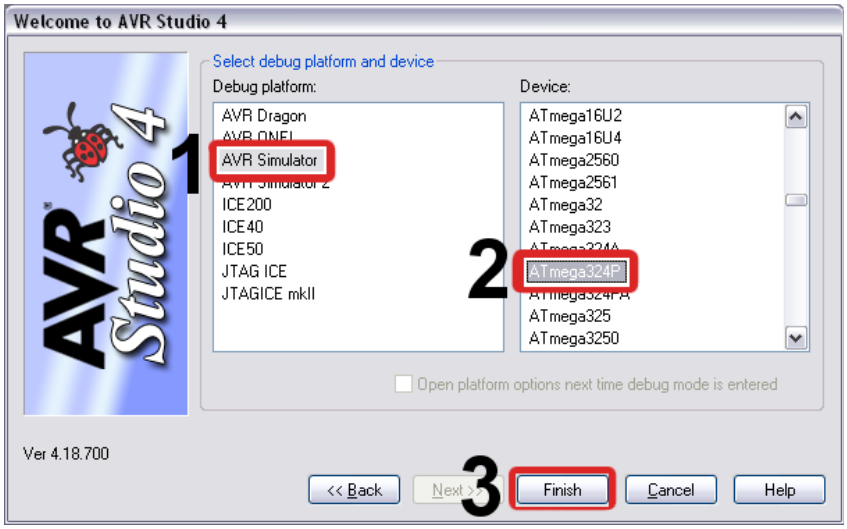
When the panel shows, click on **`[ New Project ]`** button.

3) The panel will now switch content and under **`Project Type`** select **`AVR GCC`**. Then for **`Project name:`** enter "**`HelloDolly`**" (without quotes). Leave the **`Create initial file`** and **`Create folder`** options checked. Also, **`Initial file:`** will be automatically filled with the same name as that you supplied for **`Project name:`**. Next, under **`Location:`** click on the **`[ ... ]`** button to select the destination where the project folder will be created. When you're done, click the **`[ Next >> ]`** button.

> **Note:** Remember where you create the project folder – its *location* – you'll need this info later when we get to uploading your program to the CEENBoT.



4) The panel will switch content once again and under **`Debug platform:`** select **`AVR Simulator`**, and under **`Device:`** select **`ATmega324P`**. Then click the **`[ Finish ]`** button.

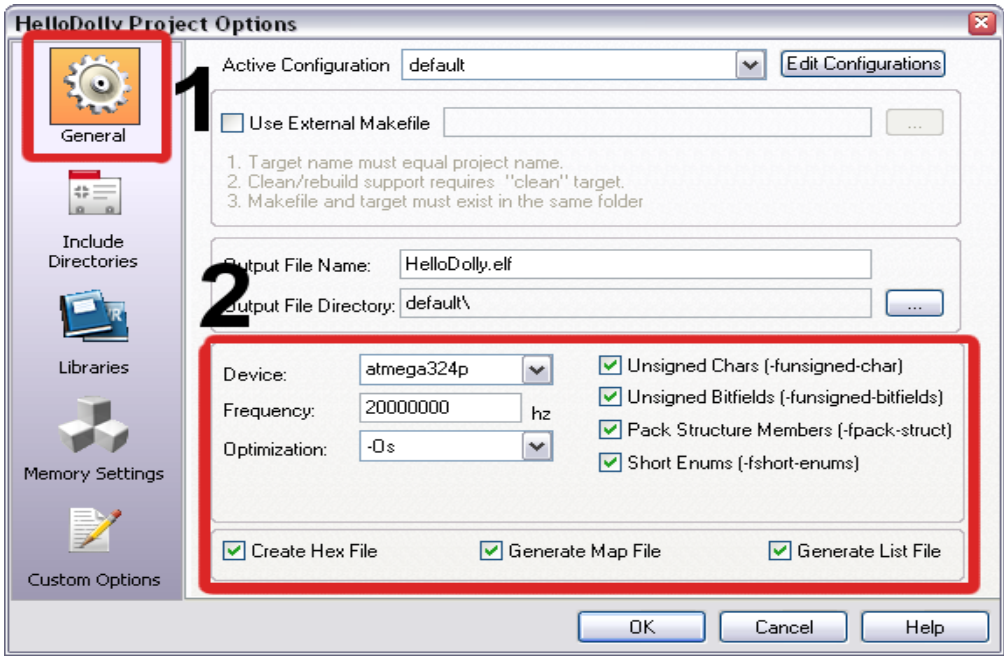


At this point your *new project* has been created. You can always use this procedure for creating a project with the purpose of being used with the CEENBoT-API. We're now ready to move to *Part II*.

Part II: Configuring Project Options

Part I showed you how to start a new Project using the *Project Wizard*. By now the project folder has been created with an *initial* C file, which should be called **`HelloDolly.c`** already open for you to start writing your code in. However, before we proceed, we need to configure the project options.

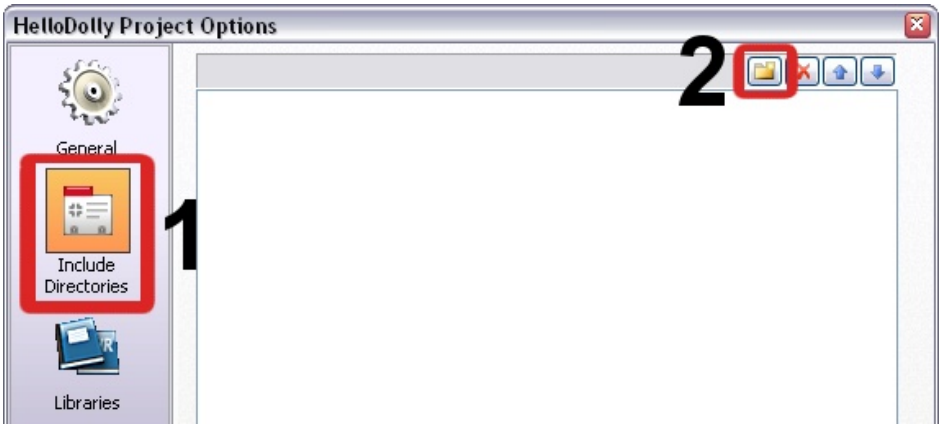
1) So we start by clicking on **`[[ Project > Configuration Options ]]`** on the top menu – this will bring up the "***HelloDolly Project Options***" as shown below:



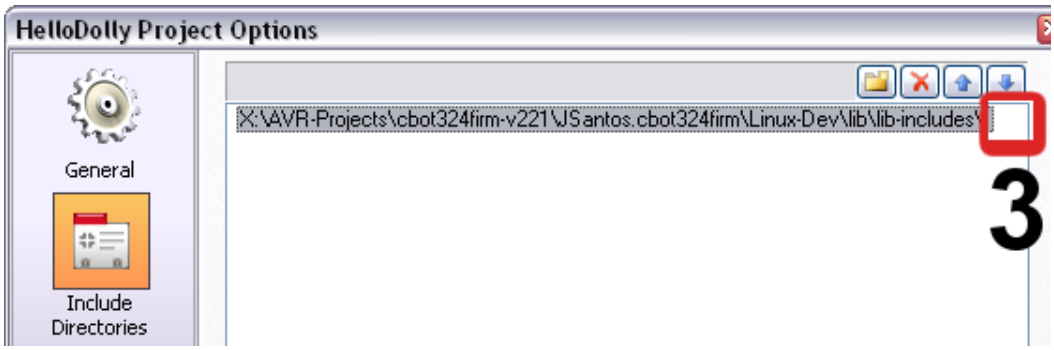
The current *panel* being shown is called **`General`**. In this panel you *must* make sure that <u>all checkboxes are *checked*!</u> (as shown in the above figure). If they're not checked, then make sure you 'check' them.

The **`Device:`** entry should already say **`atmega324p`** (as in the above figure), and **`Optimization:`** should be set to **`-Os`** (that's a capital letter O and not zero). If it isn't make sure you select this setting. Finally, in the **`Frequency:`** entry, enter "**`20000000`**" (without quotes) since the '324 board runs at 20MHz. <u>Make sure you have the correct number of zeros</u>!

2) Next, click on **`Include Directories`** on the left side of this panel. You should see the figure below:

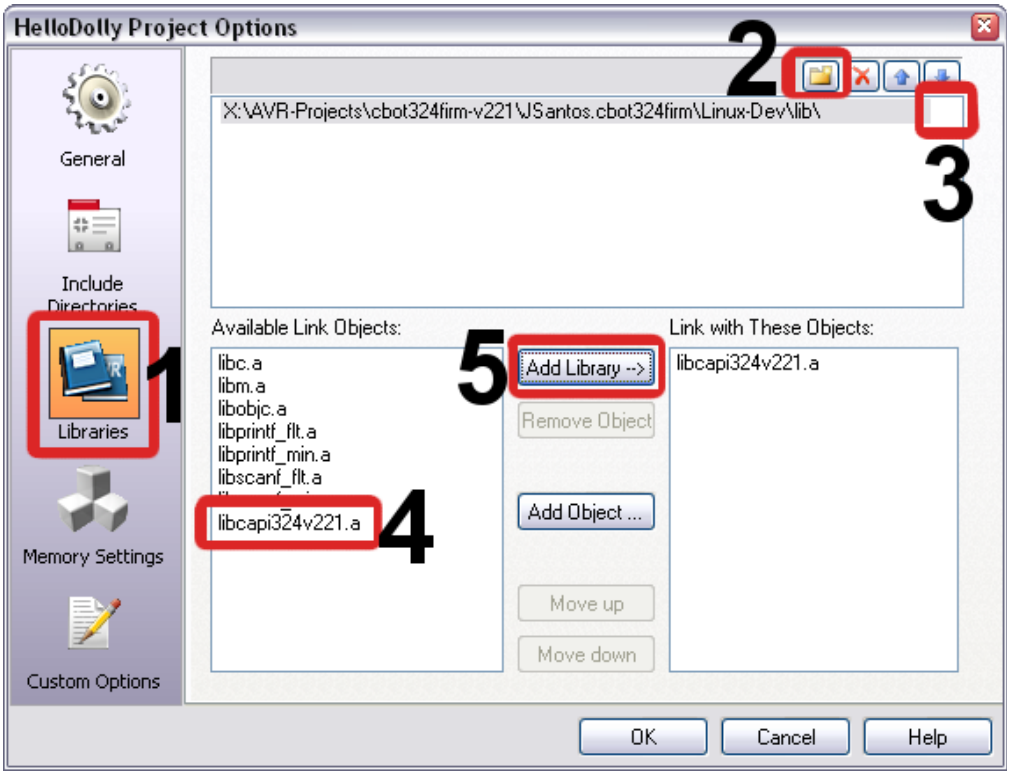


We need to tell *AVR Studio* (which tells *WinAVR-GCC*) where to find the header files for the CEENBoT-API. The header files are the files ending with '**`.h`**'. To do this, click on the *folder icon*. This will add an *empty* directory entry. Click on the **`[ … ]`** button and find the folder location in your system where you have placed the header files for the CEENBoT-API and then click OK.



The figure above shows the selected directory (in *my* system – yours will be different).

3) We're done with the previous panel. Now click on **`Libraries`**. You should see the figure below where we will perform a similar number of steps as step 2 above. We're now going to tell *AVR Studio* where to find the *static library*, so click on the *folder icon* – this will create an *empty* directory entry, then click on the **`[ … ]`** button and find the folder location in your system where you have placed the library file and click the OK button.

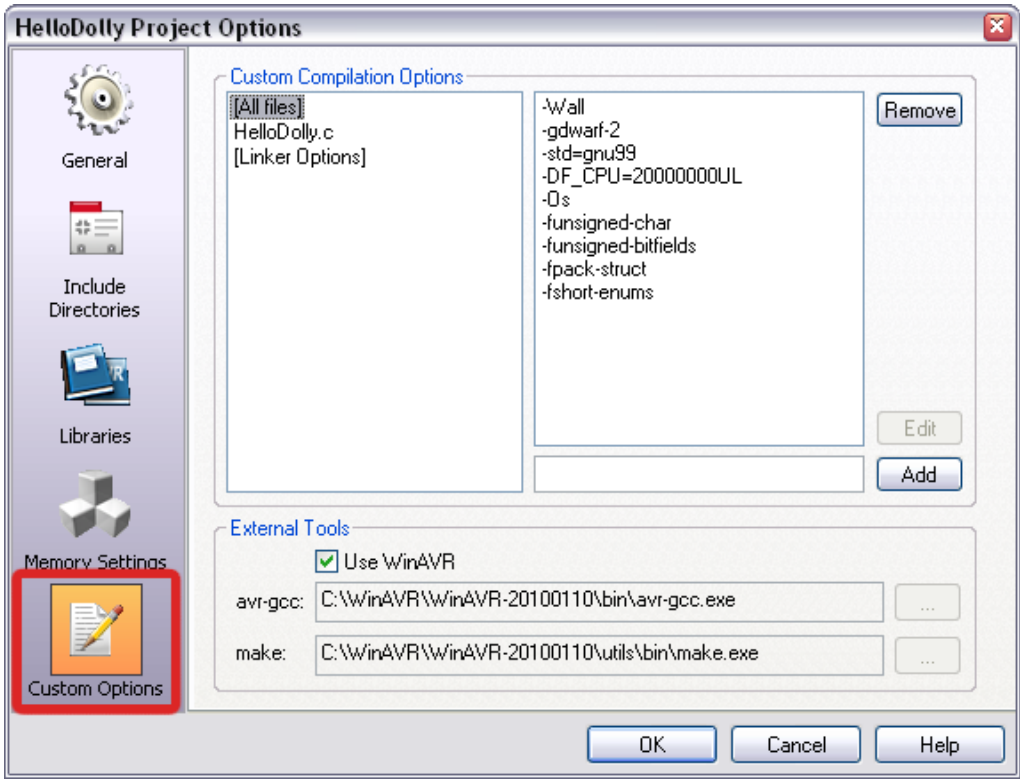


After having selected the directory, you should see **`libcapi324v221.a`** listed under **`Available Link Objects:`**. Select this file and then click on the **`[ Add Library → ]`** button. You should now see **`libcapi324v221.a`** listed under **`Link with These Objects:`** category and your settings should now match the settings shown in the above figure.

Also, while we're here, select **`libm.a`** on the left column, and click **`[ Add Library → ]`** once more, so the right column should now list: **`libcapi324v221.a`** and **`libm.a`** (this step is *not* reflected in the above figure). This is the "math library", and is needed by some of the sound-generating functions (not discussed here) so you should *always* include that in as well so you don't run into problems later on when you try to use any of those functions.

> **Note:** Linking with the **`libm.a`** is a *new* requirement as of API revision **`v1.02.000R`** and up.

4) Now click on **`Custom Options`** on the left side of this panel. You should see the figure below:

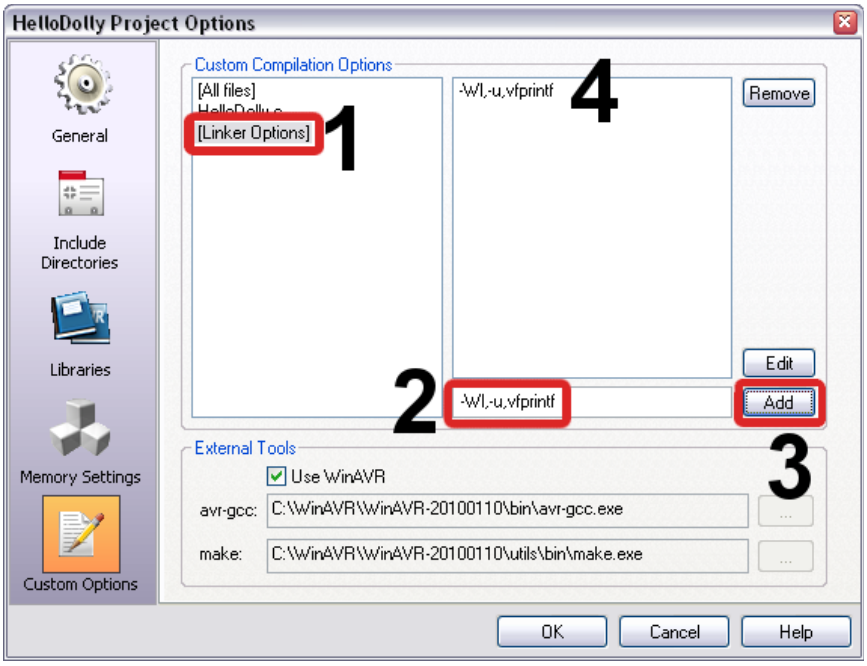


Now, at this stage there's technically nothing left to do, however, while we're at this panel there's some things I'd like you to get notice of as you may want to return here and do some 'tweaking'.

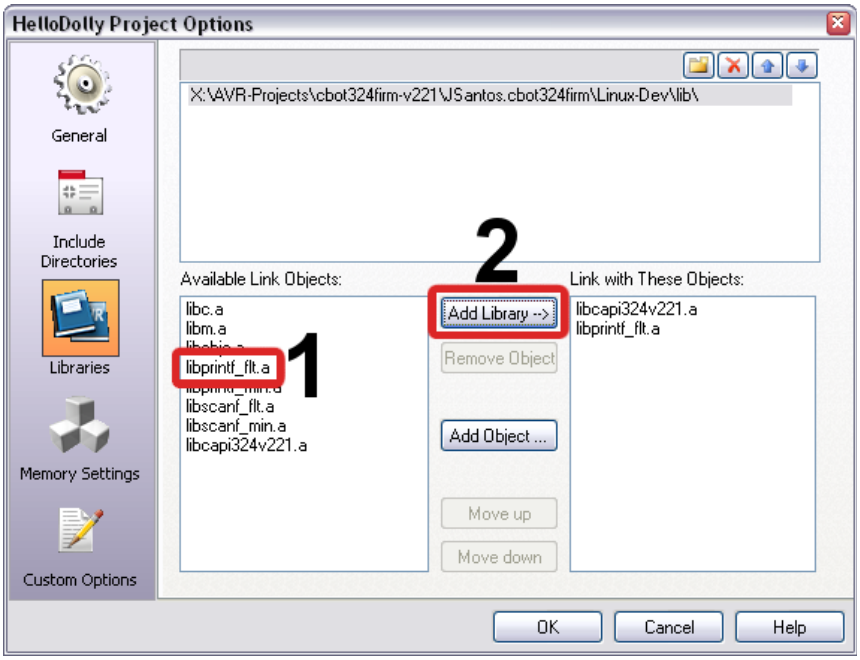
First, at the bottom of the panel, under **`External Tools`** the **`Use WinAVR`** checkbox should be checked. If you installed *WinAVR* before you installed *AVR Studio 4*, paths to **`avr-gcc.exe`** and **`make.exe`** should already be selected and automatically set. However, if it just so happens that this did not happen (for whatever reason), then you can *uncheck* **`Use WinAVR`** and manually point to the folder where *AVR Studio 4* can find avr-gcc.exe and make.exe. So if *AVR Studio 4* has problems finding these programs – *this* is the place to set up the correct locations.

The second detail I'd like to point out has to do with compilation options for the *linker*. Go ahead and click on "**`[Linker Options]`**" under **`Custom Compilation Options`**. You'll notice once you click it that the right-side options disappear. That is, we don't have any *linker-specific* options because, well, we don't need any. However, a common occurrence is that you'd like to use the **`printf()`** family of functions to print *floating point* values, which is not supported by default. This is done to save space. The default **`printf()`** implementations avoid taking floating point values to save code space. When you try to print a floating point value with the default **`printf()`** facilities, you'll end up getting question marks "?" instead.

So, IF you *do* want to use the **`printf()`** facilities with *floating point* values you enable these feature as follows. With "**`[Linker Options]`**" selected, add **`-Wl,-u,vfprintf`** as shown in the next figure and click on the **`[ Add ]`** button. (see next page).



We're not done yet... the last step is to switch to the **`Libraries`** panel once again and under **`Available Link Objects:`** select **`libprintf_flt.a`** and click on **`[Add Library →]`** button (just as you did to add '**`libcapi324v221.a`**'). See the figure below:



Note that you DO NOT need to specify the path to link against **`libprintf_flt.a`** as you did with **`libcapi324v221.a`**. These are standard libraries and the linker [should] know how to automatically find these.

> **Note:** Once again, the procedure outlined in step 4 is optional. Just know this information is here in case you find yourself needing to use the **`printf()`** facilities that support printing of floating point values.

5) Close the Project Options panel by clicking on the **`[ OK ]`** button.

At this point your configuration options for your project are set. We're ready to code!

Part III: The "*Hello, Dolly*" Program

So by now we have our project created, and also properly *configured*. We can now start to write our program. Begin by selecting the "**HelloDolly.c**" file (look for the file listing on the leftmost side of *AVR Studio 4*). Your file should be *empty*. In this blank page enter the following code:

```
// Desc: A 'Hello-Dolly' program that uses the CEENBoT-API.

#include "capi324v221.h"

void CBOT_main( void )
{

  // Open and initialize the LCD-subsystem.
  LCD_open();

  // Clear the LCD.
  LCD_clear();

  // Print a message.
  LCD_printf( "Hello, Dolly!\n" );

  // Don't leave.
  while( 1 );

} // end CBOT_main()
```

Let's talk about the above program and its structure. The first line of importance is `#include "capi324v221.h"`. This is the *main* header file for your CEENBoT-API program. ALL CEENBoT programs that use the CEENBoT-API *must* include this header file!

The next important detail is that you'll notice that '`main()`' is missing. Instead, we have '`CBOT_main()`'. Once again, ALL CEENBoT programs that use the CEENBoT-API start in `CBOT_main()`. If you're missing this function and instead try to use plain '`main()`' you'll end up with compilation errors. Note that `CBOT_main()` takes *no* arguments, nor does it return any.

Let us now talk about the body of the program. The program starts by invoking `LCD_open()`. This function acquires and initializes the necessary resources to make use of the LCD subsystem module. Many parts of the CEENBoT-API work this way. If you want to use the LEDs, you have to open *that* module; if you want to use the stepper motors, you have to open the STEPPER module via the appropriate function call, etc. You must consult the CEENBoT-API reference manual for details on how to use other modules. At the moment our task is on how to compile a simple program.

Next function calls are self explanatory. `LCD_clear()` clears the display, and `LCD_printf()` allows us to print our "**`Hello, Dolly!`**" message on the LCD display.

The last line in our program is the *infinite while* loop: `while( 1 );` Many embedded systems require the MCU to keep running to respond to other events. Therefore, we keep the program running forever in a while loop so that it can respond to other events – such as interrupt triggered events.

In any case, that completes our program – let us now compile it, link it, and generate the HEX file.

Part IV: Building the Program

So we've written some test code and we're ready to try it out on the CEENBoT. We do so by *building the program*. 'Building' refers to *compiling* each source file and *linking* the object files resulting from our source code against one or more static libraries (such as the CEENBoT-API library) and producing what will ultimately be the file that will be *uploaded* to the CEENBoT's main microcontroller. This 'file' that is ultimately uploaded to the CEENBoT is called a HEX file (because it ends with the **`.hex`** extension). A HEX file is simply a format used for encoding raw binary data – this data consists of machine instructions that will reside in the *flash* (non-volatile memory) portion of the MCU.

So... to *build* our program:

1) On the top menu select **`[[ Build > Compile ]]`**.

   While *AVR Studio 4* invokes the compiler keep an eye on the messages that scroll along the bottom panel of *AVR Studio 4*. You might see an error about AVR Studio 4 complaining that there's no 'Makefile'. This is okay because this is the first time we're compiling our *new* project. AVR Studio will create the Makefile in the project folder and you should see this particular error no longer. The line you're really looking for in the messages at the bottom part of AVR Studio is "**`Build succeeded with 0 Warnings...`**".

2) If your program has errors – analyze the errors (from the error messages given) and attempt to correct them, particularly if they're syntactic errors which are often the common cause of errors.

3) Provided you've corrected *all* errors if any select **`[[ Build > Build ]]`**.

   Once again, if no errors have occurred – the line of importance you're looking for is "**`Build succeeded with 0 Warnings...`**".

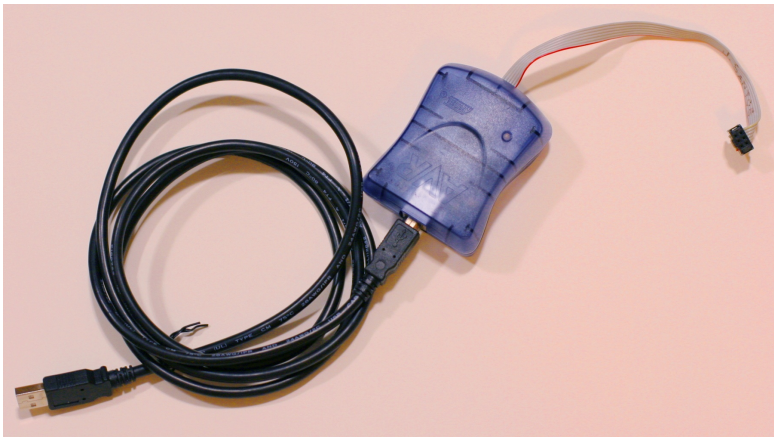
Upon a successful build, *AVR Studio 4* (with the help of *WinAVR-GCC*) will create a HEX file. We will use this HEX file to flash the raw data to the CEENBoT. Proceed to *Part V* to do this.

Part V: Uploading our Program to the CEENBoT

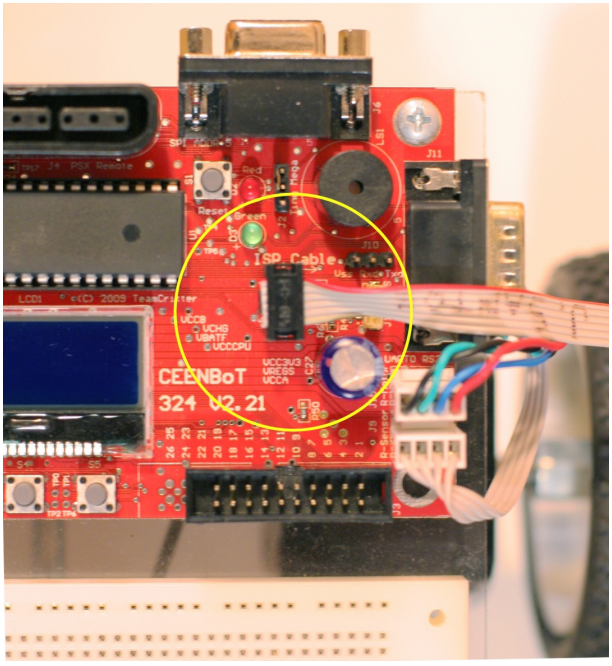
At this point you have written a program that you have successfully *built* without errors. So now it is time to upload our program and try it out. Let us begin:

1) Make sure your AVR-ISP (*In-System Programmer*) is connected to your computer via the appropriate interface (if USB connect to USB port; if Serial connect to suitable COM port, etc).

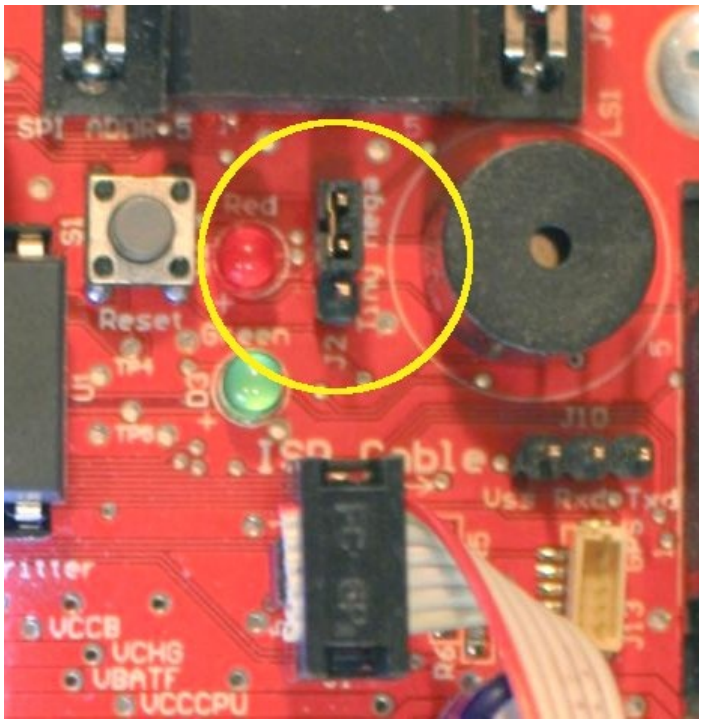
> **Note:** The ISP I'm using is the *AVR-ISP MkII* which connects to a USB port and is shown in the figure below, but please note there are MANY versions of ISP programmers out there – just make sure it compatible with *AVR Studio 4*.



2) Hook up the ISP connector to the CEENBoT's '324 board as shown:



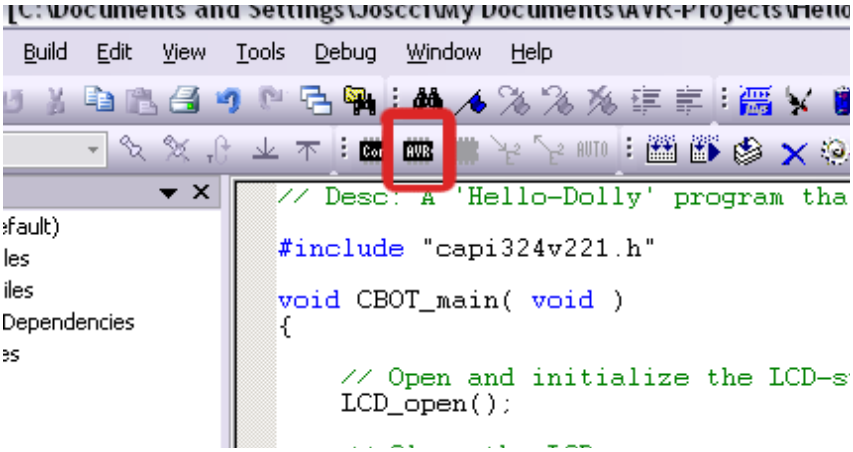
3) While we're here, right above where the ribbon cable connects is a small *header* with three pins as shown in the below figure, labeled **`J2`**:



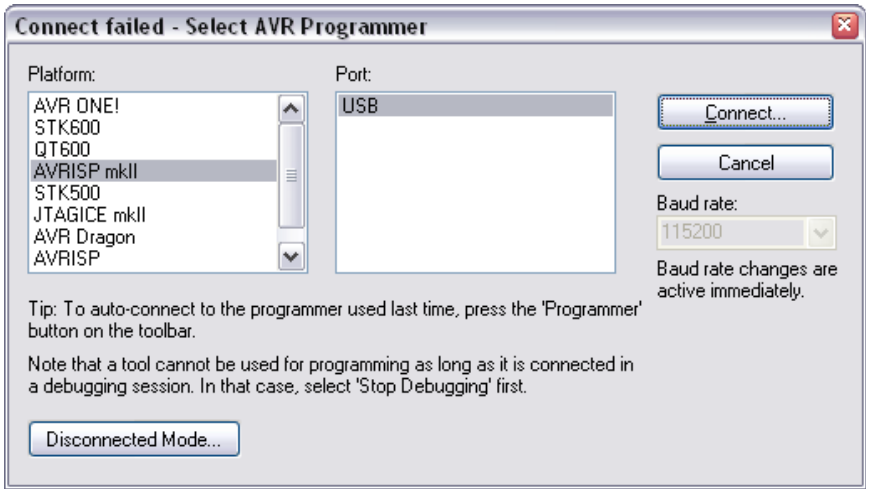
Although difficult to see in the above figure, one of the pins is fully *exposed*, while the remaining two are not. Next to this 3-pin header are the labels "**`Tiny`**" and "**`Mega`**". Your *mission* – should you choose to accept it – is to make sure that the two pins next to where it says "**`Mega`**" are the ones *covered* by the black "Jumper", while the lower pin, next to where it says "**`Tiny`**" remains exposed.

Forgetting to have the jumper (as just described) properly set can be a source of trouble as we try to access and program the MCU with the AVR-ISP programmer. So, please make sure the *Jumper* is [always] set to "**`Mega`**".

4) Click on the 'AVR' icon (see figure below) to connect to a selected ISP programmer:



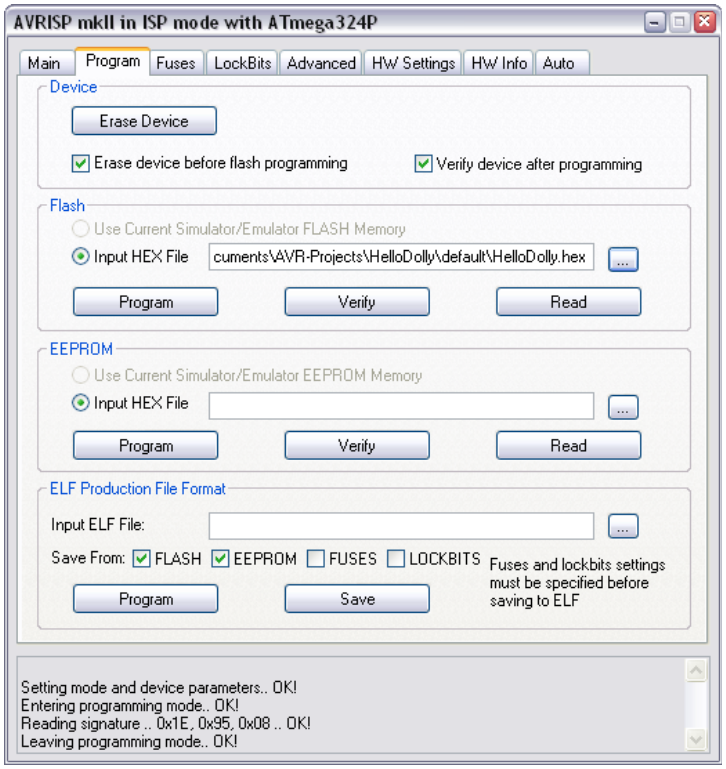
If this is the first time you're trying to access your AVR-ISP programmer, then you'll most likely see the following window to appear: "***Select AVR Programmer***" (see below). If it does, select your **`Platform:`** and the **`Port`** (again, I'm using the AVRISPmkII via USB and hence what is shown in the below figure).



Click the **`[ Connect ]`** button. If connection is successful you should see the panel "***AVR … in ISP mode with ATmega324P***" (see next page).

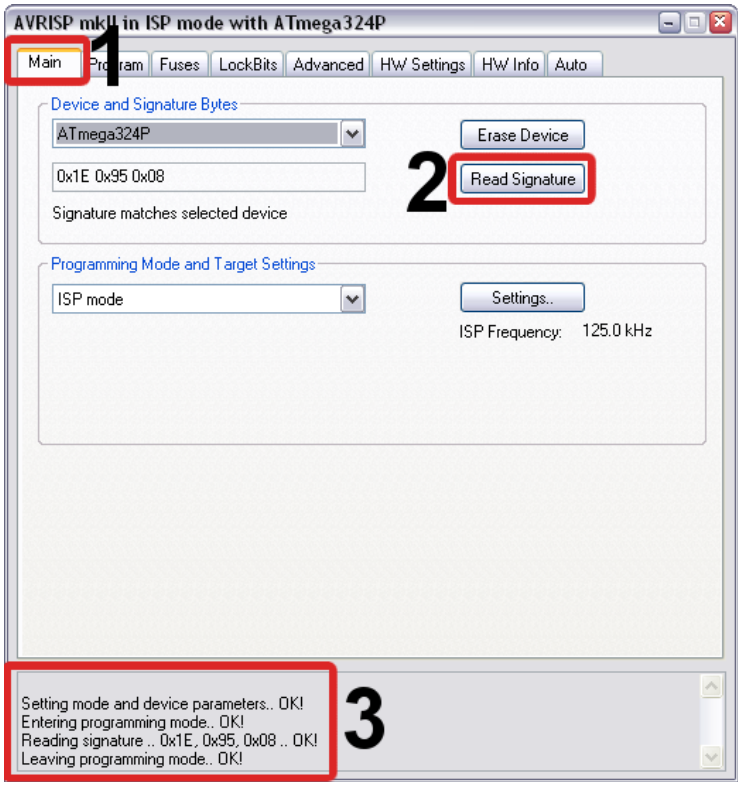
> **Note:** In some instances you might have to keep your CEENBoT turned ON while your AVR-ISP programmer is attached – so if it isn't already make sure it is turned ON.

DON'T DO ANYTHING JUST YET! Just make sure you're here:



Before we proceed to uploading our program, there's a few things I want you to explore, so proceed to the next step.

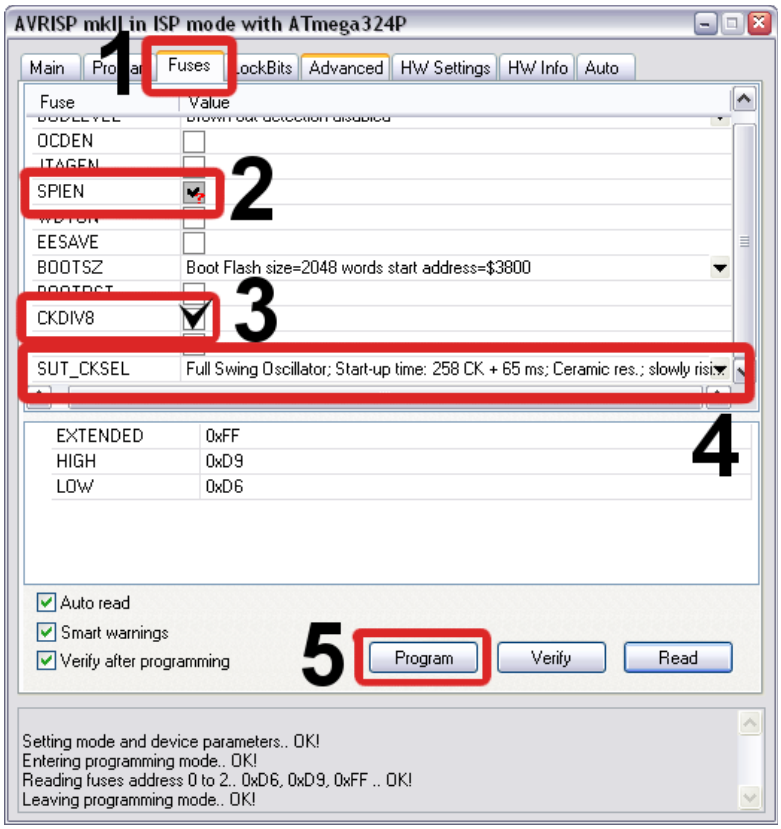
5) On the *top-most* portion of the panel, click on the **`Main`** Tab.



With the CEENBoT powered ON (and the AVR-ISP programmer attached), click on the [Read Signature] button. This will read the signature bytes from the microcontroller to make sure you're talking to the correct device, which in this case is the ATmega324P as listed in this same panel. In addition, performing a signature read can be used as a 'check' to ensure your ISP programmer and CEENBoT are talking to each other correctly. Check the messages at the bottom of this panel to ensure everything went OK. If you cannot read the signature bytes this could signal a potential problem that could be anywhere – your CEENBoT, your cable, your ISP programmer and you need to check for the obvious.

> **Note:** You're not required to perform step 4. It just merely want you to be aware of a way to 'test things out' and make sure your setup works.

6) THIS STEP IS IMPORTANT! **If**, this is your first time programming the CEENBoT with the API, then you need to click on the **`Fuses`** Tab (see figure below):



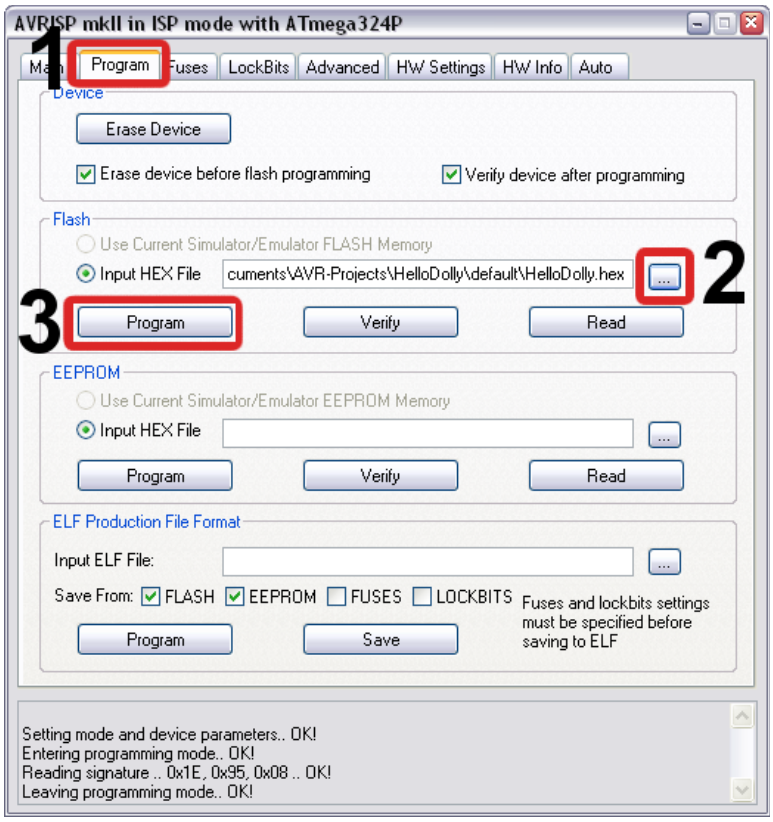
In this panel the ONLY TWO options that should be selected (or *checked*) should be **`SPIEN`** and **`CKDIV8`**. Make sure **`CKDIV8`** *is* checked. Finally, make sure **`SUT_CKSEL`** is set to "**`Full Swing Oscillator; Start-up time: 258CK+65ms; Ceramic res.; Slowly rising.`**".

When you have verified these settings to be correct, click the **`[ Program ]`** button. Take a look at the messages on the bottom portion of the panel and make sure everything went OK (no errors).

Do quick 'verify' by clicking on the **`[ Verify ]`** button to double-check. Once again ensure there are no error messages.

> **Note:** You only need to program the Fuses once. The fuse settings remain programmed in the chip until you explicitly change them again. Just know *how* to do it (and check them) in any case.

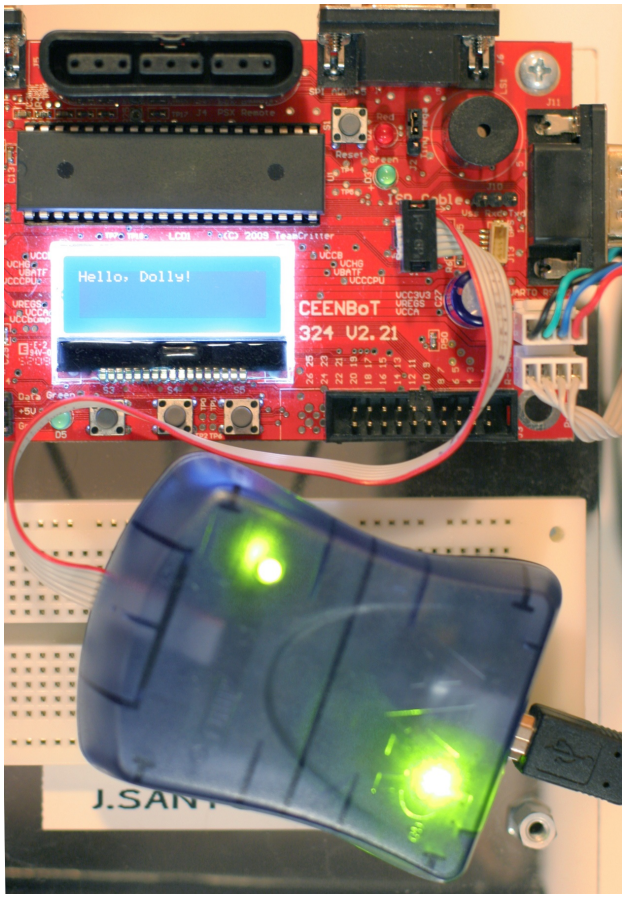
7) We're now ready to program! Switch to the **`Program`** Tab (see figure below):



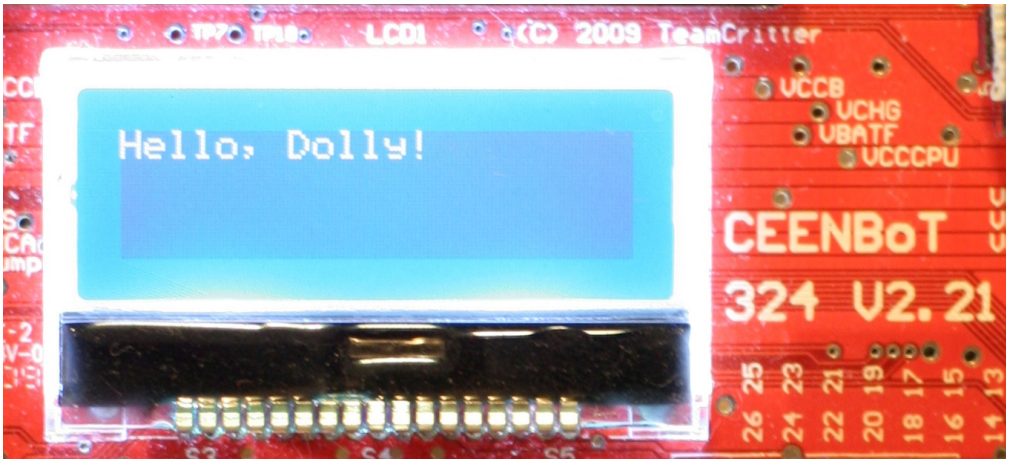
First make sure that **`Erase device before flash programming`** and **`Verify device after programmer`** have *check marks* on them. Then in the **`Flash`** section select **`Input HEX File`** (if it isn't already selected) and click on the **`[ … ]`** button to find the HEX file you just created in the *build* stage. This file is located in **`<your-project-folder>/default`**, where **`<your-project-folder>`** is the path you specified when you created *this* project in the first place. The figure above shows *my* HEX file called "**`HelloDolly.hex`**" after I've selected it.

8) With the CEENBoT powered ON (it should have been ON all this time), click the **`[ Program ]`** button.

Wait for the progress bar to complete. *AVR Studio 4* will, upon completion, read back the uploaded program data to verify it got there and it matches the original – you DO NOT need to click on the *verify* button as the check marked option in this panel ensures this happens automatically as part of the programming process. Once the program is fully uploaded your CEENBoT will most likely automatically RESET and immediately start executing your program, at which point you should see the following amazing result (see next page):



Here's a close up:



*Congratulations! – you've successfully created your first CEENBoT-API based program. Now please continue to the next section "**Additional Notes**" – it is important.*

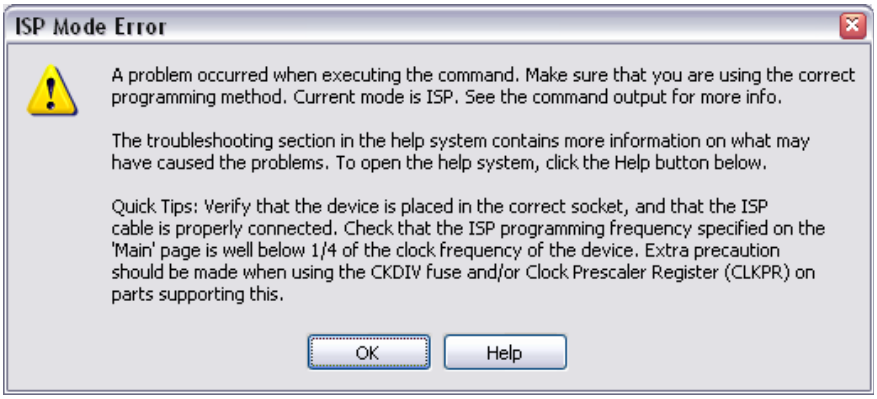
Additional Notes

- Always make sure your CEENBoT is powered ON when you access the programming panel either via the **`Main`** Tab (to read MCU signature bytes), the **`Program`** Tab (to program it), or the **`Fuses`** Tab (to program/view state of the fuses).

- Once again, step 4 in *Part V* (where we read the MCU *signature bytes*) is optional. It is helpful to ensure the your setup is good and that communication exists between the ISP programmer and the CEENBoT (and its MCU). Usually when you open the panel that allows you to program you can go straight to the **`Program`** Tab and get straight to business.

- Step 5 in *Part V* also (fuse programming and verification) is required only once. Once the fuses are properly programmed, the fuse settings are retained by the MCU. In fact, every time you switch to the **`Fuses`** Tab when the CEENBoT is powered ON, the *current* fuse settings will be read. This is a good way to verify the fuse settings are as they're supposed to be. In any case, once the fuses are properly programmed you can just jump straight to the **`Program`** Tab and get straight to business.

- If while you're attempting to access the programming panel you encounter the following:



  It most likely means your CEENBoT isn't powered. Remember you HAVE to have it powered ON for *AVR Studio 4* to communicate and/or program the MCU. If this isn't the case, then you may have another issue: cable not connected correctly, bad ISP connector, bad '324 board, your AVR-ISP is not recognized by your system because you may have incorrectly installed the corresponding drivers, etc – it could be anything!

- If your CEENBoT program requires your 'BoT to move – then it's a good idea to make sure your CEENBoT's wheels are elevated from the surface somehow. Every time you upload your program to your CEENBoT, the program is immediately executed. If that program also makes your CEENBoT move, then your 'BoT is going to get away from you, with programmer cables attached and all! If you happen to be working on an elevated desk, your CEENBoT will run off it and you'll have yourself a broken CEENBoT. So make sure your CEENBoT's wheels are elevated from any surface for SAFETY!

Okay, What Next?

Now that you've completed the tutorial and can successfully create your own CEENBoT programs that use the API, you'll want to check out the *CEENBoT-API: Programmer's Reference Guide*. It discusses all the features of the API in a module-per-module basis and the supported functions that you can call to use those features.