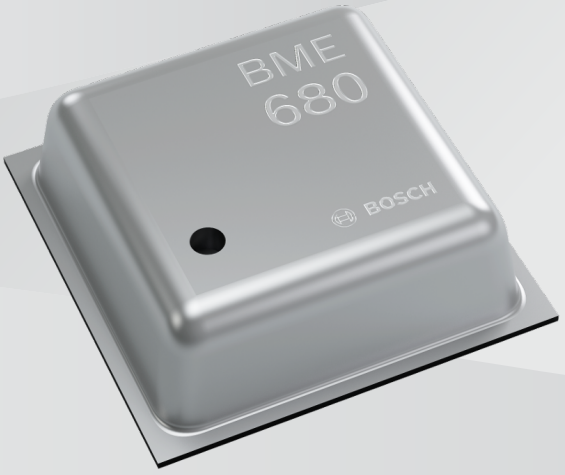
Integration Guide
Bosch Software Environmental Cluster (BSEC)



Contents

1 Instructions for using the BSEC Arduino Library 1.4.6.0 for Arduino 1.8.5

Until the Arduino IDE natively supports pre-compiled libraries, the following steps/hacks will need to be followed to integrate the BSEC library into your project.

Installation and getting started

If you have already used the previous example code for, some of the changes need to be reverted.

Primarily,

1. Adding the linker flag `-libalgobsec` in the platform.txt file.
2. Copying of the `libalgobsec.a` file to a specific directory.

1. Install the latest Arduino IDE

As of this publication, the latest Arduino IDE 1.8.5 can be downloaded from this link

2. Install the BSEC library

Either download this library as a zip and import it into the Arduino IDE. Refer to this guide on how to import libraries.

3. Replace the arduino-builder

The `arduino-builder-PR219` now has the ability to select the appropriate pre-compiled library from the library folder. This avoids the need to add the linker flag `-libalgobsec` in the platform.txt file and copying the pre-compiled library `libalgobsec.a` to a specific directory.

Replace the arduino-builder executable from the Arduino folder with the appropriate file based on your OS from this link arduino-builder-219.zip.

4. Modify the platform.txt file

The arduino-builder passes the linker flags under `compiler.c.elf.extra_flags`. Most platform.txt files in the combine recipe place this previously unused variable in the start of the link recipe whereas it should be near the end along with the other pre-compiled libraries flags.

Examples

ESP8266 community forum's ESP8266 core Original line 91, 92,

```
## Combine gc-sections, archives, and objects
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" {compiler.c.elf.flags}
       {compiler.c.elf.extra_flags} -o "{build.path}/{build.project_name}.elf" -Wl,--start-group {object_files}
       "{build.path}/arduino.ar" {compiler.c.elf.libs} -Wl,--end-group  "-L{build.path}"
```

should become

```
## Combine gc-sections, archives, and objects
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" {compiler.c.elf.flags}  -o
       "{build.path}/{build.project_name}.elf" -Wl,--start-group {object_files} "{build.path}/arduino.ar" {compiler.c.elf.libs}
       {compiler.c.elf.extra_flags} -Wl,--end-group  "-L{build.path}"
```

Arduino's SAMD core Original line 95, 96,

```
## Combine gc-sections, archives, and objects
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}"  "-L{build.path}" {compiler.c.elf.flags}
       {compiler.c.elf.extra_flags} "-T{build.variant.path}/{build.ldscript}"
       "-Wl,-Map,{build.path}/{build.project_name}.map" --specs=nano.specs --specs=nosys.specs {compiler.ldflags} -o
       "{build.path}/{build.project_name}.elf" {object_files} -Wl,--start-group {compiler.arm.cmsis.ldflags} -lm "{build.path}/{archive
       -Wl,--end-group
```

Should be,

```
## Combine gc-sections, archives, and objects
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}"  "-L{build.path}" {compiler.c.elf.flags}
       "-T{build.variant.path}/{build.ldscript}" "-Wl,-Map,{build.path}/{build.project_name}.map" --specs=nano.specs
       --specs=nosys.specs {compiler.ldflags} -o "{build.path}/{build.project_name}.elf" {object_files}
       -Wl,--start-group {compiler.arm.cmsis.ldflags} -lm {compiler.c.elf.extra_flags} "{build.path}/{archive_file}"
       -Wl,--end-group
```

5. Only for the ESP8266 - modify the linker script

Due to the current size of the BSEC library, upon compilation, you will receive an error: `section '.text' will not fit in region 'iram1_0_seg'`. In order to solve this, you will need to modify the linker script and specifically define where the library should be placed in memory.

You will need to modify the file `eagle.app.v6.common.ld` typically found in `{YourESP8266PPackage↩ Directory}\tools\sdk\ld`.

With reference to the linker script here,

After line 117 `*libwps.a:(.literal.* .text.*)`, add `*libalgobsec.a:(.literal.* .text.*)`, which should look like,

```
  *libupgrade.a:(.literal.* .text.*)
  *libwpa.a:(.literal.* .text.*)
  *libwpa2.a:(.literal.* .text.*)
  *libwps.a:(.literal.* .text.*)
  *libalgobsec.a:(.literal.* .text.*)
  *(.irom0.literal .irom.literal .irom.text.literal .irom0.text .irom.text .irom.text.*)
  _irom0_text_end = ABSOLUTE(.);
  _flash_code_end = ABSOLUTE(.);
} >irom0_0_seg :irom0_0_phdr
```

6. Verify and upload the example code

Start or restart the Arduino IDE. Open the example code found under `File>Examples>Bsec software library>Basic`.

Select your board and COM port. Upload the example. Open the Serial monitor. You should see an output on the terminal.

Note that not all supported cores have been tested. In such cases, the examples can be found under `File>Examples>INCOMPATIBLE>Bsec software library>Basic`

The current list of tested micro-controllers include,

- atmega2560 : Arduino MEGA 2560
- cortex-m0
- cortex-m0plus : Arduino Zero
- cortex-m3 : Arduino Due
- cortex-m4f : Adafruit NRF52 BlueFruit LE (cortex-m4 directory),