including math functions in module

v923z

Posts: 158
Joined: Mon Dec 28, 2015 6:19 pm

including math functions in module

by **v923z** » Wed Nov 28, 2018 8:57 pm

Hi all,

I would like to include a couple of functions from math in my C module. I can compile the module all right, but I get a linker error like so

CODE: SELECT ALL

```
test.c:(.text.test_vec+0x5e): undefined reference to `acos'
```

I have tried to add the file to SRC_LIBM in Makefile, but that didn't help (I got an other type of error). I would like to stress that the module itself works: if I kick out the offending function, I can compile, link, and upload the firmware, and then import and use all implemented functions without issues. What I want to say with that is that I have problems only, when I want to access a function from math. (I include math.h). Including micropython modules appears to be beyond the scope of tutorials on adding your own module.

What really confuses me is the fact that the built-in math functions are scattered all over the place, and there seems to be no central location with a single source and header file. This might be related to copyright issues, mightn't it?

In any case, I would appreciate any illuminating comments.

Cheers,
Zoltán

dhylands

Posts: 3686
Joined: Mon Jan 06, 2014 6:08 pm
Location: Peachland, BC, Canada
Contact: 💬

Re: including math functions in module

by **dhylands** » Wed Nov 28, 2018 9:15 pm

The function acos takes doubles as arguments and Micropython (at least on the pyboard) uses floats. The libm library that comes with Micropython has the acosf function which takes float arguments and returns floats.

Dave Hylands

v923z

Posts: 158
Joined: Mon Dec 28, 2015 6:19 pm

Re: including math functions in module

by **v923z** » Wed Nov 28, 2018 9:45 pm

This is definitely true, thanks! However, there are still degrees and radians that are part of modmath.c, and I don't think they are exposed anywhere.

dhylands

Posts: 3686
Joined: Mon Jan 06, 2014 6:08 pm
Location: Peachland, BC, Canada
Contact: 💬

Re: including math functions in module

by **dhylands** » Thu Nov 29, 2018 12:24 am

On the pyboard, they're exposed to python in the math module:

CODE: SELECT ALL

```
MicroPython v1.9.4-479-g828f771e3-dirty on 2018-09-12; PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>> import math
>>> math.radians(90)
1.570796
>>> math.degrees(1.570796)
89.99998
>>>
```

Those functions aren't exposed to C code (at least not directly since they're declared as STATIC). You could always look up the python functions, but those functions are simple enough you could also copy/paste them.

Dave Hylands

v923z

Posts: 158
Joined: Mon Dec 28, 2015 6:19 pm

Re: including math functions in module

by **v923z** » Thu Nov 29, 2018 6:59 am

> **dhylands wrote:** ↑ Thu Nov 29, 2018 12:24 am
> On the pyboard, they're exposed to python in the math module:
>
> **CODE: SELECT ALL**

```
MicroPython v1.9.4-479-g828f771e3-dirty on 2018-09-12; PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>> import math
>>> math.radians(90)
1.570796
>>> math.degrees(1.570796)
89.99998
>>>
```

Those functions aren't exposed to C code (at least not directly since they're declared as STATIC).

Yeah, sorry. I meant exposed to C code.

> **dhylands wrote:** ↑ Thu Nov 29, 2018 12:24 am
> You could always look up the python functions, but those functions are simple enough you could also copy/paste them.

This is what I am doing at the moment, but was wondering, whether there is a more elegant fashion. I wanted to avoid code duplication, that's all.

Is the STATIC declaration related to compile-size limitations, or something else? I think I would definitely like to have access to the core C functionality in my modules, and it would be great, if the modules could be kept in synch with the micropython base, so copy/paste is not my preferred way.

dhylands

Posts: 3686
Joined: Mon Jan 06, 2014 6:08 pm
Location: Peachland, BC, Canada

Re: including math functions in module

by **dhylands** » Thu Nov 29, 2018 4:56 pm

Well, if you were going to expose the functions at a C level, you would probably factor them out from the existing functions, since the existing functions take mp_obj_t's rather than floats, so you'll wind up doing a bunch of conversions.

For such a small function, it's quite possibly cheaper (both from a code size perspective and from a performance perspective) to copy/paste the contents, but only profiling will tell you.

Contact:

Dave Hylands

v923z

Posts: 158
Joined: Mon Dec 28, 2015 6:19 pm

Re: including math functions in module

by **v923z** » Thu Nov 29, 2018 9:30 pm

> **dhylands wrote:** ↑ Thu Nov 29, 2018 4:56 pm
> Well, if you were going to expose the functions at a C level, you would probably factor them out from the existing functions, since the existing functions take mp_obj_t's rather than floats, so you'll wind up doing a bunch of conversions.
>
> For such a small function, it's quite possibly cheaper (both from a code size perspective and from a performance perspective) to copy/paste the contents, but only profiling will tell you.

What you write is OK for math functions, and I definitely see your point on the type conversion, but it seems to me that hardware functions (e.g., ADC) are also enclosed in their respective source files. This would mean that one cannot write an extra module using those functions. I think, in those cases copy/paste is really not an option.

Let us suppose I want to interface to some sensor via ADC/I2C/SPI/USART etc. I can do that from python, but that has an overhead. Wouldn't it be better, if one could have access to the hardware from anywhere? What I am trying to understand here is why those functions have to be declared static. There might be a very legitimate reason that I don't know.

dhylands

Re: including math functions in module

by **dhylands** » Thu Nov 29, 2018 9:39 pm

The general tendency is to make things STATIC unless they need to be shared. From a maintenance standpoint a function that is declared STATIC can be changed arbitrarily and you only need to look within the same source file to find all of the "users" of that function.

Posts: 3686
Joined: Mon Jan 06, 2014 6:08 pm
Location: Peachland, BC, Canada
Contact: 💬

If a function is made public, then if you want to change the API of the function you need to find all of its uses in the rest of the codebase and fix those as well.

Some examples of functions which are made public:
https://github.com/micropython/micropyt ... int.c#L162
(which is sued within this source file and from the switch module).

Various led functions (like led_state, led_toggle, etc which have both a C and python API.
https://github.com/micropython/micropyt ... tm32/led.c

Dave Hylands

v923z

Posts: 158
Joined: Mon Dec 28, 2015 6:19 pm

Re: including math functions in module

by **v923z** » Fri Nov 30, 2018 7:18 am

> **dhylands wrote:** ↑ Thu Nov 29, 2018 9:39 pm
> The general tendency is to make things STATIC unless they need to be shared. From a maintenance standpoint a function that is declared STATIC can be changed arbitrarily and you only need to look within the same source file to find all of the "users" of that function.
>
> If a function is made public, then if you want to change the API of the function you need to find all of its uses in the rest of the codebase and fix those as well.

I think there is absolutely no disagreement between us; I fully understand and appreciate your points. All I wanted to say is that I have already run into difficulties with this restriction. For now, I solved the problem by copying small chunks of code, but I believe, this can't work in the long run.

For one thing, this is also a maintenance issue: if the original code (meaning the one that is part of the micropython code base) changes/improves, you have to copy that chunk, or at least, think about, whether it is worth copying. I would even argue that this case might even be worse, because the maintainer of

mymodule.c would always have to check for changes in the code base, if they don't want to miss out on the improvements. On the other hand, if the API of the original is made public, and it changes, then this fact will be immediately obvious to everyone, because mymodule.c won't compile anymore.

Second, every time you copy code, you are probably going to make the firmware larger. I don't think that the compiler/linker is smart enough to notice that a static function in objarray.c is the very same as a static function in mymodule.c. I believe, this argument is especially important in the case of a microcontroller, where resources are sparse.

The question really is, whether the benefits of making at least a subset of the functions public outweigh the drawbacks. I wrote a module that renders fonts in 16-bit colours as described in https://github.com/peterhinch/micropyth ... /writer.py . @peterhinch reported render times in the order of 5 ms for 12-pixel high fonts. I can manage it under 100 us with a font height of 20 pixels. That is a gain of at least 50. This sounds convincing to me. I will try to release the code over the weekend, so you can see it for yourself.

Post Reply

9 posts • Page **1** of **1**

Return to "Development of MicroPython"

Jump to