Embedded Programming

Software Engineering
Embedded Programming
Hardware Specs

Recall the hardware specs for your boards:

- MKL25Z128VLK4 microcontroller
  - 48MHz ARM Cortex-M0+
  - 128KB flash
  - 16KB SRAM

What might make embedded programming different from desktop programming?

FRDM-KL25Z Board

image from KL25Z User’s Manual
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main () {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different `.c` file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {  
    motor_velocity_target = 3.0;  
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:

```c
if (kill_switch) motor_velocity_target = 0;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:

```c
if (kill_switch) motor_velocity_target = 0;
```

And why not have it dependent on tracking, perhaps in a different .c file:

```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:

```c
if (kill_switch) motor_velocity_target = 0;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:

```c
if (kill_switch) motor_velocity_target = 0;
```

And why not have it dependent on tracking, perhaps in a different .c file:

```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main () {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```

And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main () {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
Software Engineering
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main () {
    motor_velocity_target = 3.0;
    // rest of code here
}

So far, so good, right?
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
Globals

While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global
```

```c
void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:

```c
if (kill_switch) motor_velocity_target = 0;
```

And why not have it dependent on tracking, perhaps in a different .c file:

```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; // global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different .c file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```
While we’re talking about globals, what anti-patterns can arise from this?

```c
float motor_velocity_target; //global

void main() {
    motor_velocity_target = 3.0;
    // rest of code here
}
```

So far, so good, right?

Perhaps I also have a kill switch in another function:
```c
if (kill_switch) motor_velocity_target = 0;
```
And why not have it dependent on tracking, perhaps in a different `.c` file:
```c
if (bad_tracking) motor_velocity_target -= 0.1;
```

Soon, you have no clue what the target actually is - dataflow spaghetti!
Avoid dynamic memory allocation
Watch out for the limited RAM and stack overflow
Watch out for synchronization errors
Write code that conforms to user expectations
Avoid dataflow spaghetti