Adding a new driver class takes a few steps:

- Add the class in the class enumeration
- Add a header describing the class
    - its request functions
    - a structure containing pointer to request functions
    - utility function prototypes (blocking request wrappers, ?) [1]
- Add reference to the class in the driver structure
- Add a configuration token for the class
- Add a directory for the class
- Add a C-file with utility functions [1]

[1]: This is only needed if the class has such utility functions.

In the examples, we'll call the new class `foobar`.

Adding the class in the device type enumeration

Edit device/include/device/device.h, you'll see a `enum device_class_e`. Add the new device type at the end.

```
    device_class_lcd,
    device_class_gpio,
    device_class_i2c,
+   device_class_foobar,
  };
```

Adding a header describing the class

Create a file in `drivers/include/device/foobar.h`. It may contain

- Callback definitions,
- Request function prototypes,
- Global utility functions.

See trunk/mutekh/drivers/include/device/char.h for sample file.

The `struct dev_class_foobar_s` is a structure holding all the class-specific request functions. You may ensure this structure holds less pointers than the `DRV_MAX_FUNC_COUNT` constant defined in trunk/mutekh/drivers/include/device/driver.h.

Don't forget to protect the new header against circular inclusions with the lines:

```
#ifndef __DEVICE_FOOBAR_H__
#define __DEVICE_FOOBAR_H__

#endif
```

We'll use that macro later.

Adding a reference to the class in the driver structure

Edit trunk/mutekh/drivers/include/device/driver.h and add an entry in the `f` union of the `struct driver_s`.

You must protect it with the macros defined in the header of the class, in case the class is not included in legacy code.

Dont `#include` your driver class in `driver.h`.

```
#ifdef __DEVICE_FOOBAR_H__
    struct dev_class_foobar_s foobar;
#endif
```

Add a configuration token for the class

Edit trunk/mutekh/drivers/device/drivers.config, add a configuration block for the class:

```
%config CONFIG_DRIVER_FOOBAR
flags nodefine
desc At least one foobar device driver must be enabled to have it defined
%config end
```

The `flags nodefine` statement tells the build system an user may not directly define this token, and implies this token is only allowed to be provided by another one.

Create a directory for the class

- create a `driver/device/foobar` directory,
- add a Makefile inside it,
- add `foobar` in the list of subdirectories in `driver/device/Makefile`.

Create the global helper functions file for your class

- create a `driver/device/foobar/device_foobar.c`,
- add it in `driver/device/foobar/Makefile`.

Add a new driver

See NewDriver