Buck Converter Application

The `app_buck_converter` module implements the control logic for a buck (step-down) converter used in embedded power management systems. It is designed to regulate output voltage efficiently while ensuring protection mechanisms such as current limiting are enforced. The control strategy is based on a **cascade PID control loop**, where an outer voltage control loop sets the reference for an inner current control loop. This hierarchical approach ensures both stable voltage regulation and fast current response.

A key feature of this module is its integration with a flexible `pid_controller` library, which includes **anti-windup mechanisms**. These mechanisms are crucial for maintaining system safety and stability during current limiting. When the system reaches the configured current limit, the PID controller's windup prevention avoids the accumulation of control error, which could otherwise lead to overshoot, oscillations, or instability. As a result, the converter maintains accurate voltage regulation while ensuring current does not exceed safe levels, even under varying load conditions.

Additionally, the module follows a **configuration-based software architecture**, which greatly enhances its reusability and adaptability across different projects. All tuning parameters, operational limits, and system-specific settings are defined in external configuration files located under the `ProjectConfig` directory. This design allows developers or system integrators to update system behavior—such as PID gains, voltage/current limits, or timing parameters—**without making any changes to the actual source code** of the module.

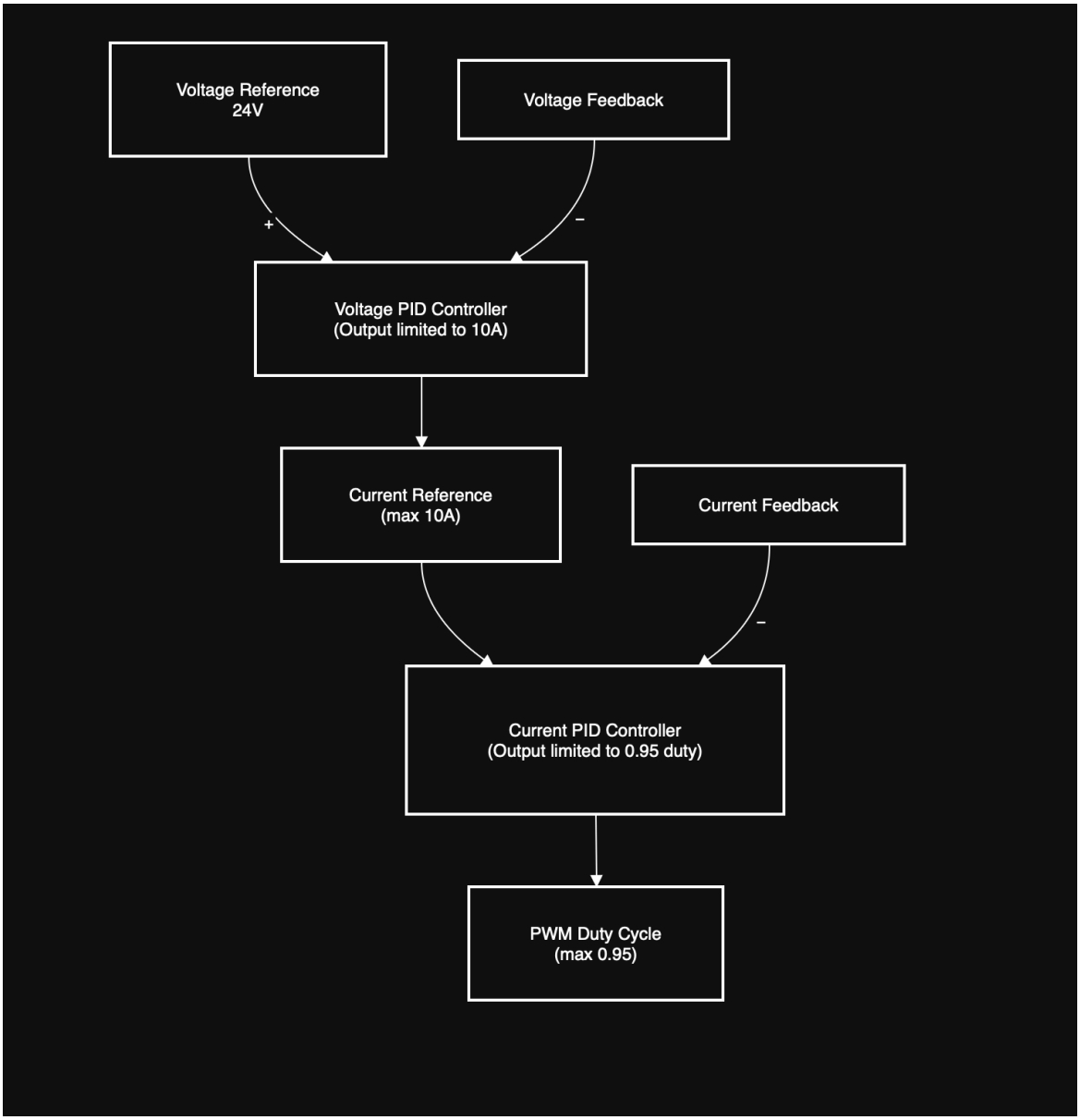
This separation between configuration and implementation provides significant advantages in terms of maintainability and test efficiency. Since changes in system behavior are made through configuration files only, **the core software module remains untouched**, which eliminates the need to re-execute unit tests or revalidate unchanged software. This modular and decoupled approach not only accelerates development but also supports robust software reuse and minimizes regression testing efforts in future projects.

```
static pid_controller_t m_pid_voltage_controller =
{
    .Kp = 3,
    .Ki = 0.02,
    .Kd = 0.1,
    .Kaw = 0.02,
    .TimeStep = 20,
    .controller_output_max = 9.96f, // voltage controller output uses as output current reference
    .controller_output_min = 0.0f, // voltage controller output uses as output current reference
};

static pid_controller_t m_pid_current_controller =
{
    .Kp = 3,
    .Ki = 0.02,
    .Kd = 0.1,
    .Kaw = 0.02,
    .TimeStep = 20,
    .controller_output_max = 0.95f, // current controller output uses as output duty
    .controller_output_min = 0.0f, // current controller output uses as output duty
};

const buck_converter_cfg_t g_buck_converter_config =
{
    .over_current_occurence_time_min = 3,
    .i_out_max = 10.00f,
    .v_out_ref = 24.0f,
    .period_time_process_of_controller_ms = 20,
    .pid_out_voltage_cotroller_ptr = &m_pid_voltage_controller,
    .pid_out_current_cotroller_ptr = &m_pid_current_controller,
};
```

The configurations shown in the figure are stored under the `Project_Configs` folder and are provided to the `init_buck_converter` function to configure the `app_buck_converter` module.



The **first PID controller** operates on the voltage loop. It compares the measured output voltage against a fixed reference voltage of 24V and generates a current reference signal. This output current reference is limited to a maximum of 10A to ensure the system operates safely within current constraints.

The **second PID controller** acts on the current loop. It receives the current reference from the first controller and compares it with the actual measured current. The output of this current PID controller is the PWM duty cycle, which is responsible for controlling the power stage of the buck converter. This duty cycle is capped at 0.95 to prevent overdriving the converter.

By cascading the voltage and current control loops in this manner, the system ensures both tight voltage regulation and robust current limiting. The cascade structure improves dynamic response and stability, while the output limits in each PID controller protect the hardware from unsafe operating conditions.

Table III
SYSTEM AND CONTROLLER PARAMETERS (SIMULATION)

Parameters	Values	Parameters	Values
L, r	2 mH, 0.5 Ω	switching freq.	100 kHz
C	50 μF	k_q	100
E	48 V	i_{max}	2 A
c	1.5×10^5	i_{min}	1 mA

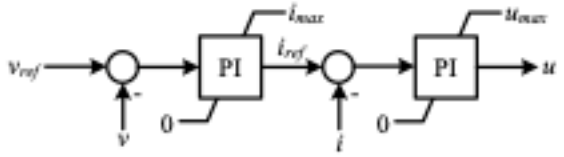


Figure 4. Traditional cascaded PI control with current limitation [7]

- Example image in an essay (https://eprints.whiterose.ac.uk/id/eprint/125488/1/FINAL%20VERSION.pdf)