READ ME:

Description of example project

This example implements a Buck Converter in Voltage Control mode for being used together with the XMC Digital Power Explorer kit. Additionally, it includes a number of house-keeping functionalities to typical in SMPS (Switched Mode Power Supply). Some of this cannot be fully tested on an isolated demo board, however, the target is to demonstrate the user some of the added values of digital control with XMC™ in SMPS. The project is implemented using XMC4200, a 32-bit ARM Cortex M4 based microcontroller with FloatingPoint Unit. In addition, it also has dedicated control peripherals such as the HRPWM module for high resolution PWM generation and a Versatile Analog-to-Digital (VADC) module for analog signal measurement, making it suitable for applications that require fast calculation such as the Voltage Control mode.

The voltage Control loop is implemented by a classic 3 poles 3 zeros filter using floating point values. The provided filter coefficients has been selected to have the following controller characteristics:
+ Switching freq = 200kHz
+ Crossover freq = 10kHz
+ Phase margin = 50 degrees
+ PWM master clock = 20,480GHz = 80MHz *256 (8 bits for HRPWM)
+ PWM resolution = 150 ps
+ ADC resolution = 12 bits

VADC conversion is being continuously triggered by the Compare Match 2 of CCU8 (set by default to the maximum = period value). Once the output voltage has being measured by the VADC, an interrupt is generated. Inside the ISR, the 3 poles 3 zeros controller is applied and new value for the duty cycle is updated.
One instance of each of those DAVE 4 APPs have been used to simplify the peripherals configuration:

HRPWM (High resolution PWM, generates the ADC trigger through the CCU8 peripheral and the HRPWM)
ADC_MEASUREMENT_ADV(Output Voltage measurement)
INTERRUPT (NVIC configuration)

Filter related APIs are implemented in the xmc_3p3z_filter_float.h file while the PWM initialization and control ISR are implemented in the main.c.

Hardware Setup

XMC4200 Digital Power Control Card + XMC Digital Power Explorer Kit
Vout P14.6
CCU8 direct output P0.4
CCU8 inverted output P2.15
HRPWM out0 P0.5
HRPWM out1 P0.2
Vin P14.3 (not measured in project)

How to test the application (power conversion part)

a. Import the project
b. Prepare board set up: connect XMC4200 Digital Power Control Card into

XMC Digital Power Explorer and supply power with included power adapter to it.
Make sure power on switch is in the "on" position and that the jumper in power board is in the XMC4000 position

c. Compile and flash the application onto the device
d. Run the application.
e. Check via oscilloscope the different test points available in XMC Digital Power Explorer. Vout is now controlled to be stable at 3,3 V independent of load switching LEDs next to load switches just turn on when that particular load is activated
f. If changes are needed in APPs configuration, make sure code is generated again before compiling

HOUSE-KEEPING add-ons:

Apart from the above explained power control, digital control with XMC™ allows to extend the functionality in SMPS by adding the so-called housekeeping functions. These run in parallel to the main control function of buck converter. In this example following house-keeping functions are exercised:

- **PMBus v1.2 slave functionality**: using USIC peripheral and PMBus_Lib, PMBus slave is implemented with an set of 6 commands for experimenting with PMBus™. Further commands can be added easily thanks to PMBusLib and examples provided PowIRCenter is used as PMBus™ master:
  http://www.infineon.com/cms/en/product/promopages/power-center-software/
  Connect PMBus™ master to PC and XMC Digital Power Explorer (pin header for communication). In PowIRCenter software, open board design in “File->Open Board Design” included in **XMC_PMBus_Lib folder inside DAVE project**. Inside PowIRCenter software, I2C Bus must be configured to 100kHz due to missing clock stretching. After this start code in XMC™.

- **Soft start of reference voltage:** during start up Vref is increased within a systick timer routine

- **Over Current Protection (OCP)**: analog comparator (CSG unit) trips if current value (10bits resolution) is higher than max value allowed. The OCP immediately switches off PWM with a TRAP signal and calls an interrupt for saving status into EEPROM

- **Over-voltage warnings:** using boundary functions in ADCs, Vin and Vout are monitored. If warning levels (high and/or low) are violated, an interrupt is triggered. Status will be stored into EEPROM for future status report

- **Temperature monitoring:** temperature measurement using die temperature measurement in XMC4200

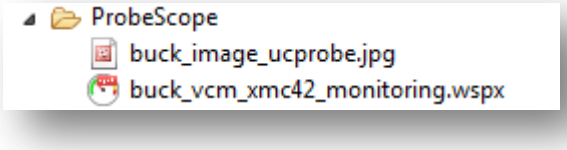
- **uCProbe functionality (real time monitoring, commanding):** added uCProbe capability to monitor in real time variable or command the buck converter. Oscilloscope function can be enabled in control loop function.
  uCProbe can be used to read EEPROM on UI, read out temperature or check step response of buck converter
  In **ProbeScope folder in DAVE project**, a uCProbe workspace can be found with an example UI

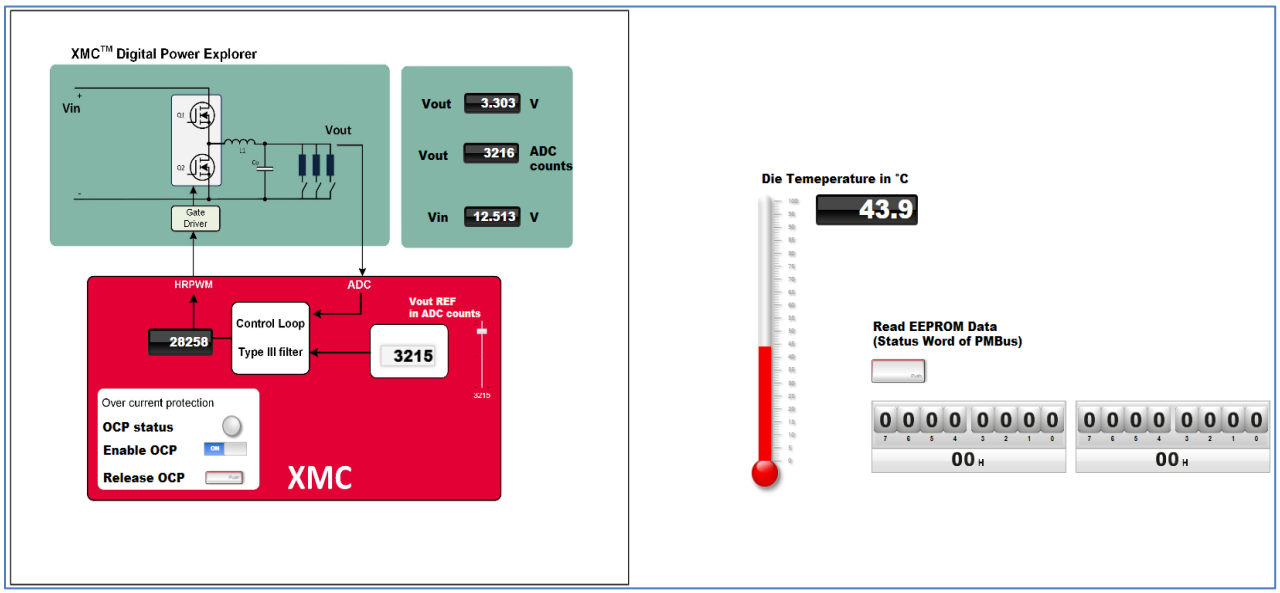
- **Board support file:** added board support file (.bpl) to the project that simplifies the assignment of pins. With bpl file, when assigning pins, user sees also the function of that pin in the board instead of only the pin and port number
- **Debug pins for CPU load calculation**

Some examples of usage of the whole set up are provided in following sections:

Exercise with house-keeping funcitons:

1. Initial status after XMC™ runs, watched from ucProbe workspace -double click in buck_vcm_xmc4200_monitoring.wspx file in ProbeScope folder inside DAVE project.



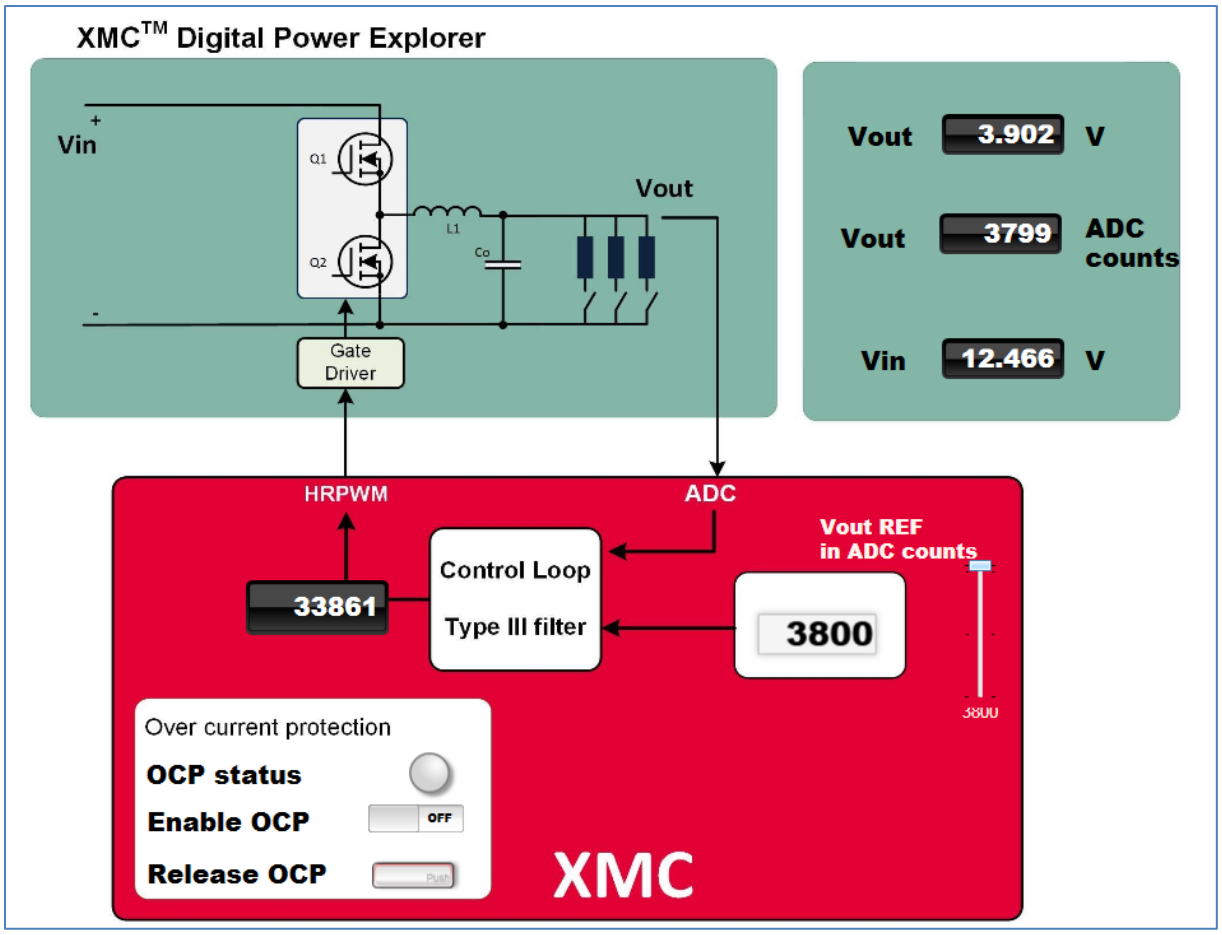


2. Disable OCP by clicking the slide button “Enable OCP” to the “OFF” position. This will avoid protection to be triggered when manipulating reference voltage slider. Once that is done, to generate an over voltage condition, we can move the slider “Vout REF” to higher value than that configure in main.c

```
#define UV_VOUT_PROT_HIGH_LEVEL          (3750U)
```

It is possible as well to enter the value directly into the Vout REF box.

For example, we can set Vout REF to 3800, which is above the wraning level 3750:

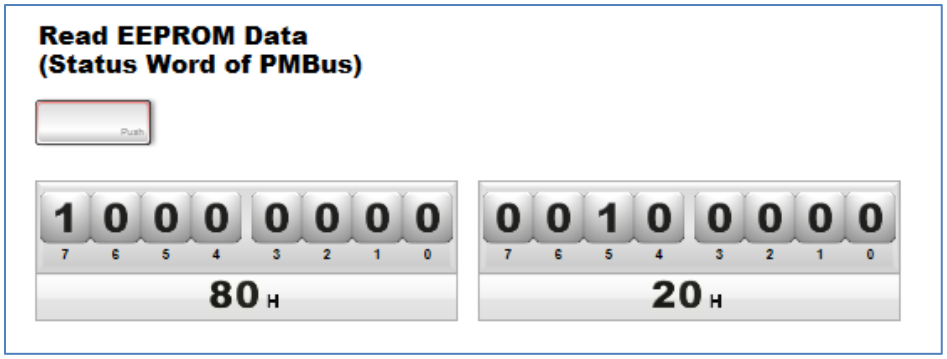


3. Once the value in the reference is higher, an overvoltage condition has happened. To confirm that, it is possible to read the EEPROM where the status of system is stored every time an overload condition happens. To do that just click in the push button showed below. The status of PMBus™ word is shown in binary. If all is “0”, then no overload happened. Any “1” corresponds to an overload condition. The structure of PMBus™ status word in the code, is given here for convenience:

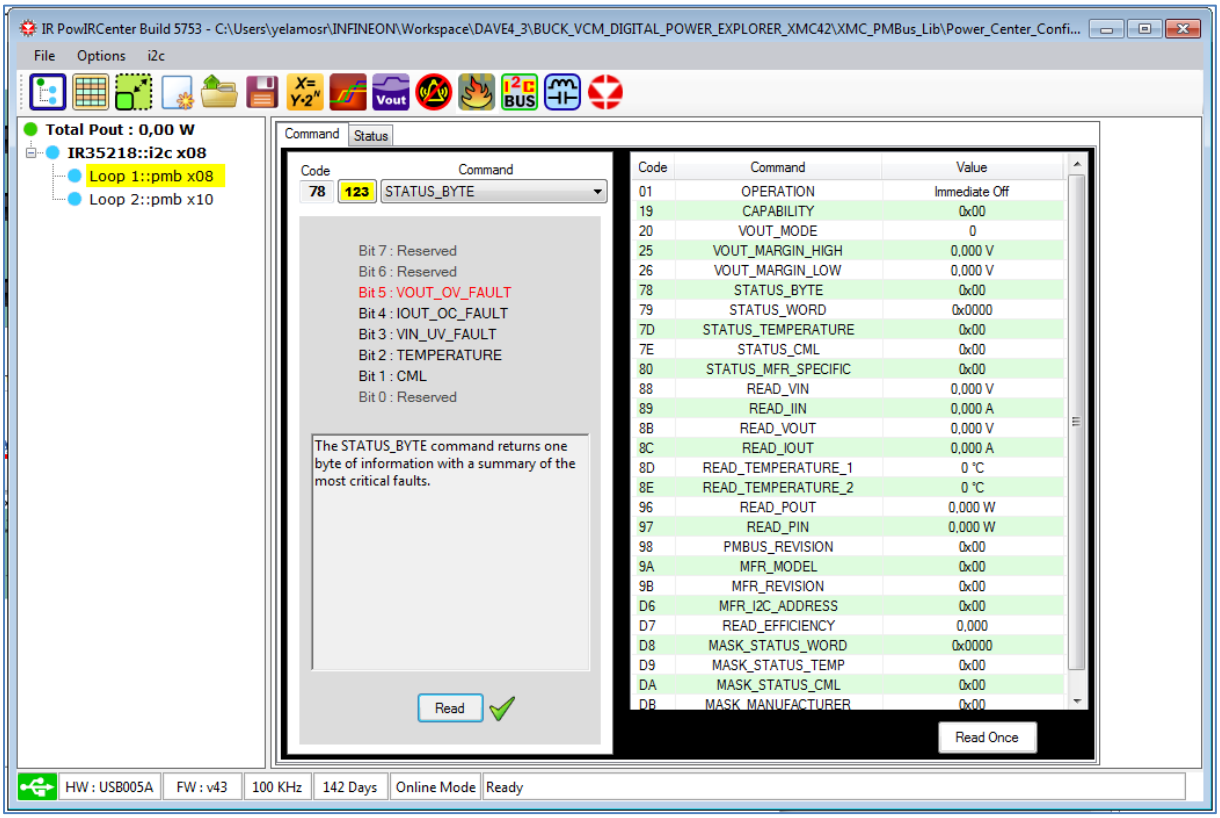
```
typedef struct STATUS_WORD_REG
{
  union
  {
        struct
        {
     uint32_t none_of_the_above   :1; /**< 0 A fault or warning not listed in bits [7:1] of this byte has occurred*/
     uint32_t cml                 :1; /**< 1 A communications, memory or logic fault has occurred*/
     uint32_t temperature         :1; /**< 2 A temperature fault or warning has occurred*/
     uint32_t vin_uv              :1; /**< 3 An input under voltage fault has occurred*/
     uint32_t iout_oc             :1; /**< 4 An output over current fault has occurred*/
     uint32_t vout_ov             :1; /**< 5 An output over voltage fault has occurred */
     uint32_t off                 :1; /**< 6 This bit is asserted if the unit is not providing power to the output,
                                         regardless of the reason, including simply not being enabled. */
     uint32_t busy                    :1; /**< 7 A fault was declared because the device was busy and unable to
respond.*/
    };
    uint8_t  status_lower_bits;
  };

  union
  {
        struct
        {
     uint32_t unknown        :1; /**< 8 */
     uint32_t other          :1; /**< 9 */
     uint32_t fans           :1; /**< 10 A fan or airflow fault or warning has occurred */
     uint32_t power_good     :1; /**< 11 The POWER_GOOD signal, if present, is negated */
     uint32_t mfr            :1; /**< 12 manufacturer specific fault */
     uint32_t input          :1; /**< 13 An input voltage, input current, or input power fault or
```

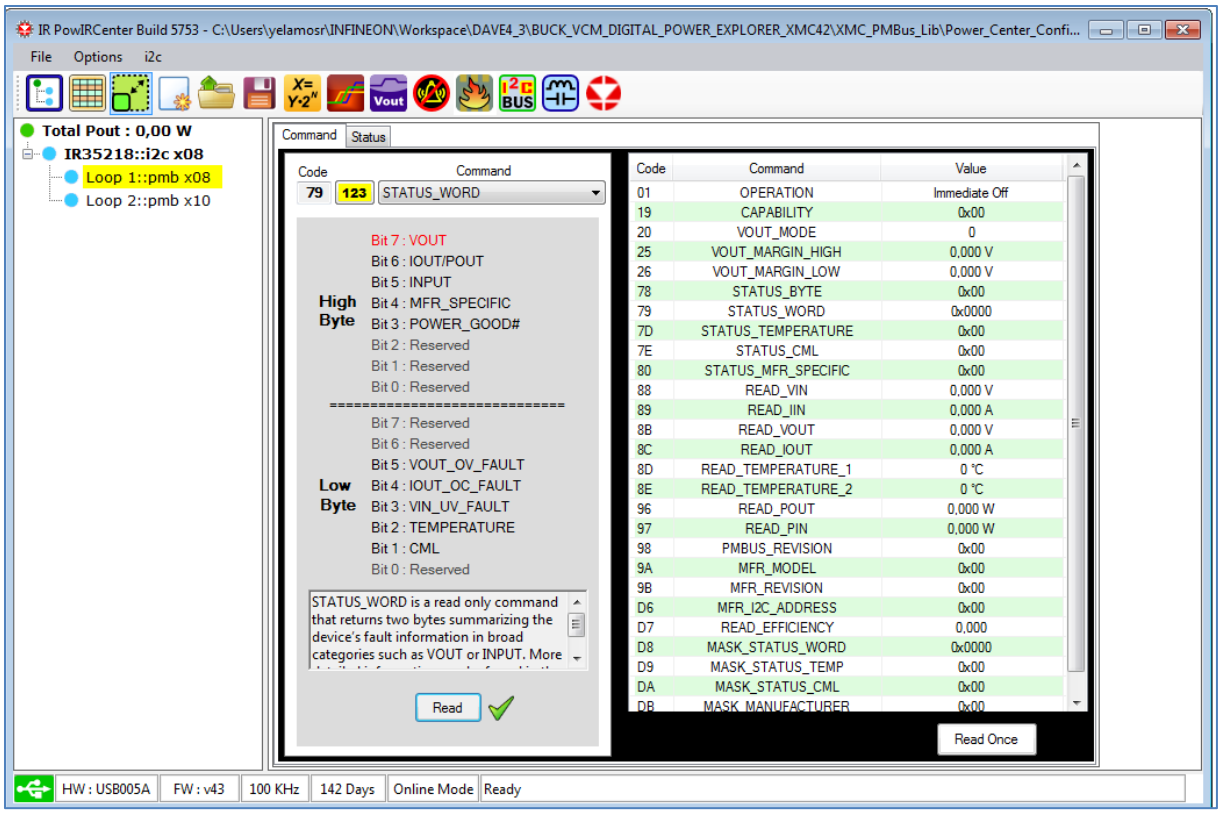
```
                                    warning has occurred */
      uint32_t iout_pout      :1; /**< 14 An output current or output power fault or warning has occurred */
      uint32_t vout           :1; /**< 15 An output voltage fault or warning has occurred */
    };
    uint8_t  status_upper_bits;
  };
}STATUS_WORD_REG_t;
```



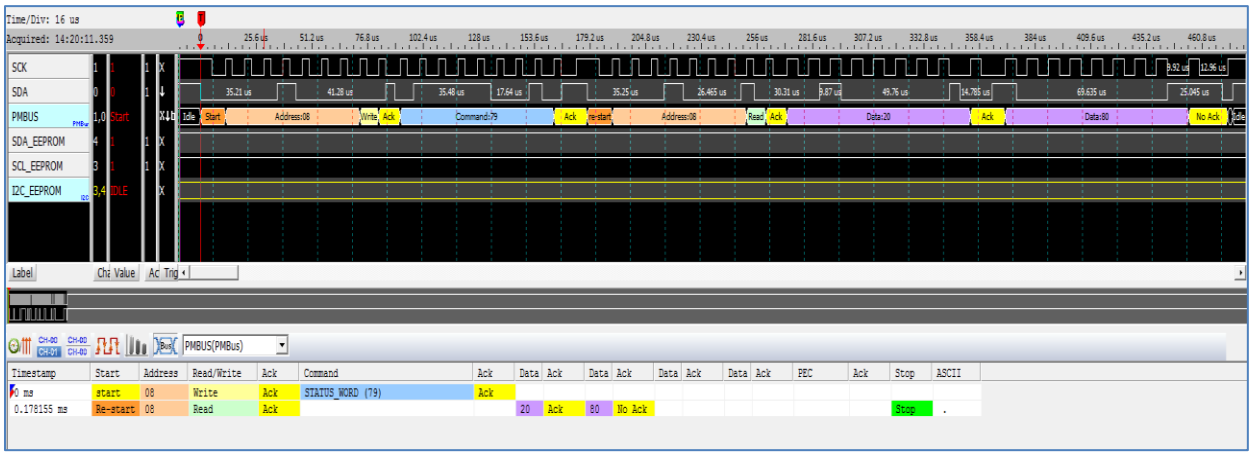
4. Another option is to send a PMBus™ command to read the status in a byte or word format. The bit written in the EEPROM corresponds 1 to 1 to the bits read in PowIRCenter



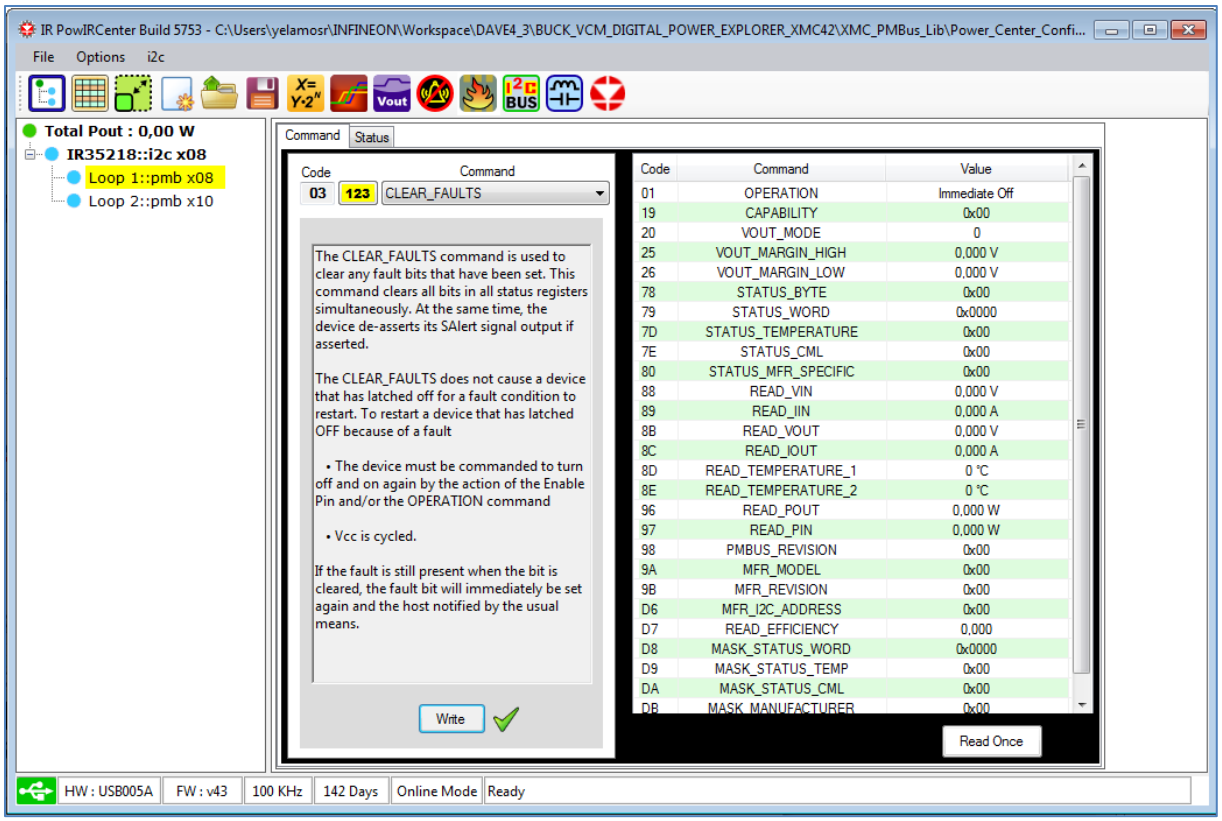
As can be seen in the figure above, VOUT_OV_FAULT appears red after correct read out (green stick in the right side of “Read” button.



If we look into a logic analyzer, we see the I2C and PMBus™ sequence when utilizing the "Status word" command in PMBus™:

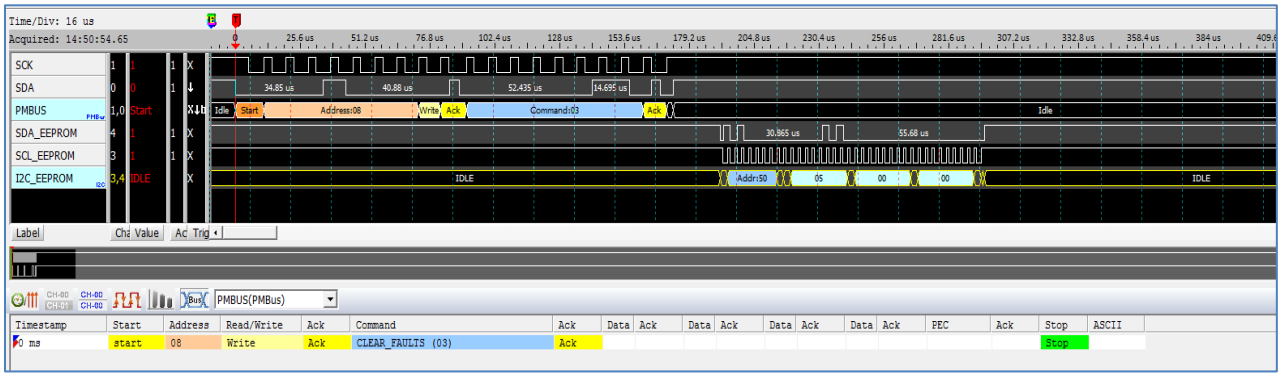


5. Let´s clear now the overvoltage fault with a the corresponding PMBus™ command using PowIRCenter:



In logic analyzer the following is captured. As can be seen, after the PMBus™ command to clear faults, a I2C command to the on board EEPROM is sent to store the new status.

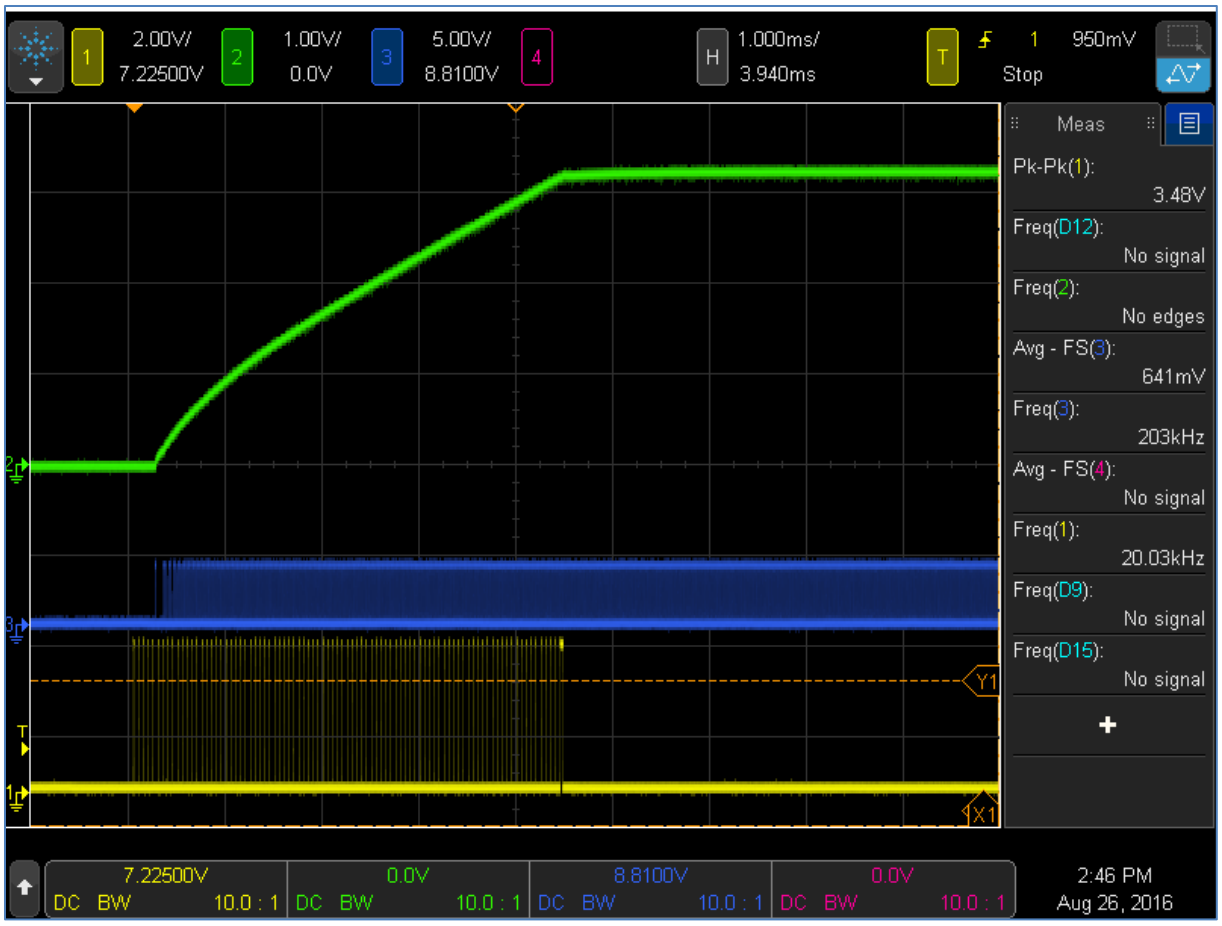
PMBus™ is set to work at 100kHz, while EEPROM I2C is configured to work at 400kHz



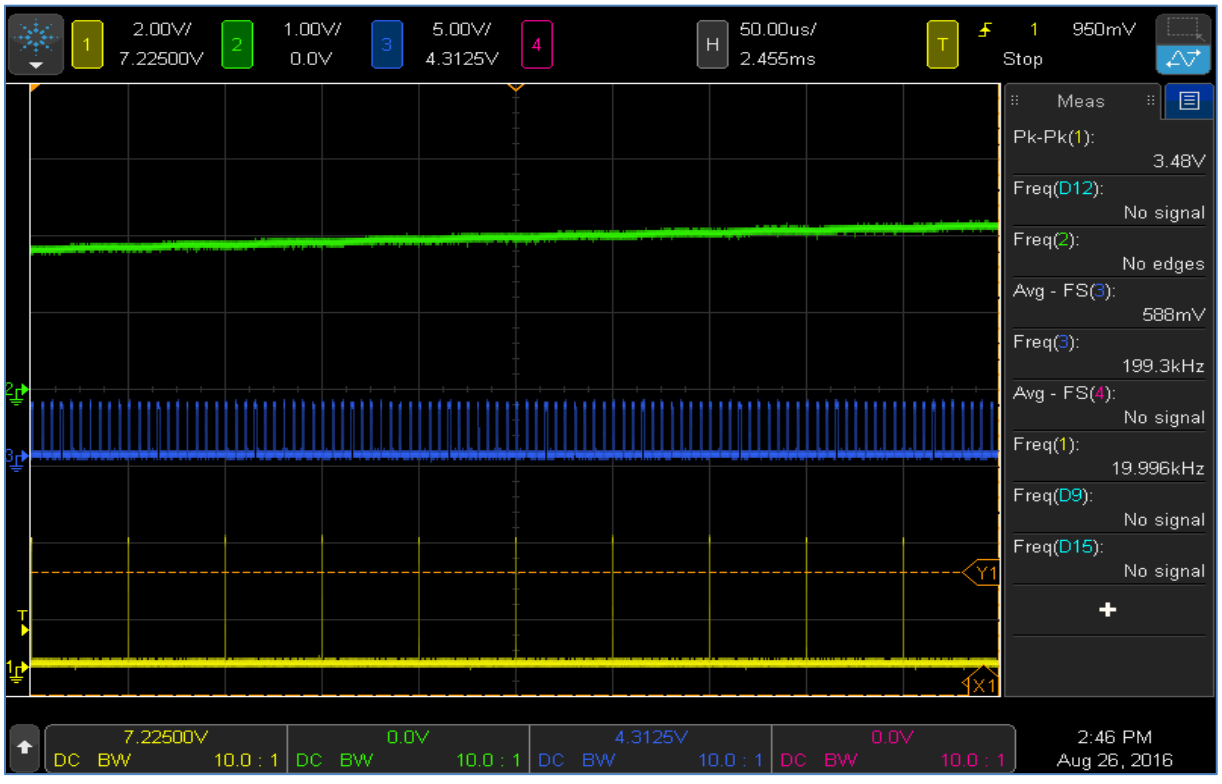
SOFT-START ANALYSIS:

Another feature of this example is that the reference voltage of the buck converter is increased slowly the first time the buck converter stats. This is called soft start and is fully configurable to adjust to different system requirements. In following figure, the output voltage is captured (green signal). A systick timer (executing every time that the yellow signal goes high) is used to increase in step the reference voltage. The blue signal represents PWM signal driving the buck converter. The duty cycle is increased slowly until target value of 3,3 volts is achieved. Current spikes are reduced with this method. During start up, a different overcurrent level has been configured to provide flexibility. One can disable protections during start up or relax conditions.

At the end of soft start sequence, the systick timer for it is stop to avoid CPU load. This is confirmed in oscilloscope where yellow signal stays low from that



In next oscilloscope capture, a zoom in version is shown. Control loop executes every switching cycle (blue is PWM), while the reference changes in soft start every time the yellow signal goes high. There are therefore 10 control loop executions per reference value increase.



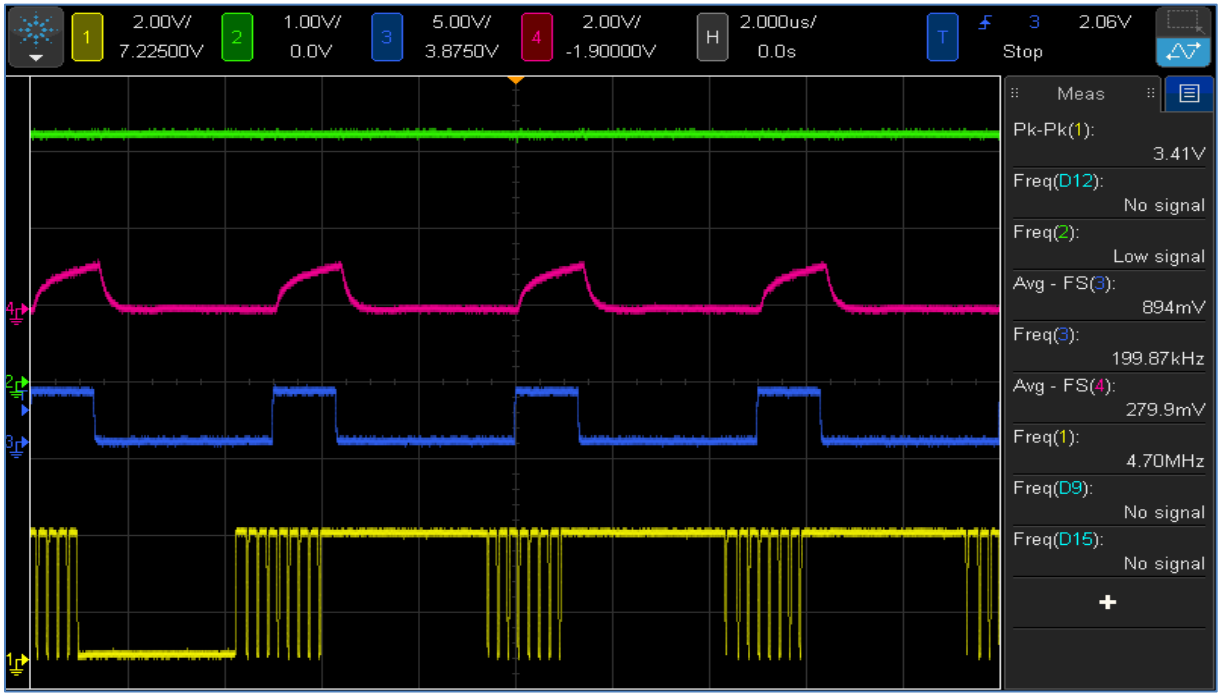
NORMAL RUN TIMING ANALYSIS:

Observer task timing (pink), yellow is control loop (synchronous to PWM switching frequency-200kHz), blue is PWM signal and green signal is stable Vout equal to target 3,3V

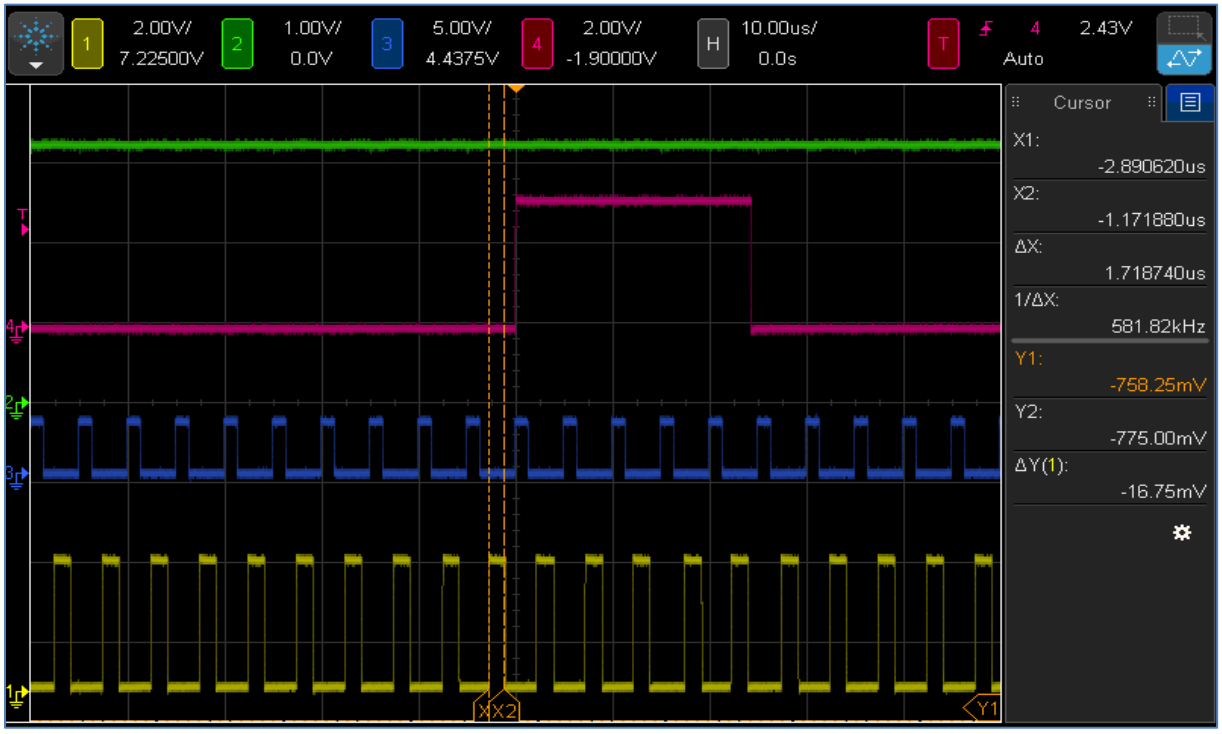


Some other signals during normal run time:

While loop timing on yellow, blue is PWM, pink is current in inductor, green is Vout. What can be seen here is the CPU is free for around 40% of the time (time that while loop executes). It can also be seen where the control loop finishes execution with respect to PWM timing. This is important to understand phase margin degradation due to late update.



Observer systick timer can be interrupted by the higher priority control loop interrupt. This can be seen in following oscilloscope capture, where the pink signal represents the execution time of observer systick and yellow the control loop execution. As can be seen, the control loop executes seamlessly and ignoring systick timer execution. This will cause that the observer functions is interrupted a few times, needing more cycles to finalize completely.



The observer function is enabled after soft start is finished and executes every 50ms as can be seen in next figure

