LED power supply current configuration using NLM0011/NLM0010

About this document

Scope and purpose

This document describes Infineon's latest near-field communication (NFC) controller NLM0011/NLM0010 for LED power supply current configuration, and how to use it in circuit designs.

The NLM0011/NLM0010 provides all the benefits of contactless configuration of LED power supply current without using a microcontroller (µC) and its peripheral components.

Intended audience

This document is intended for design engineers who want to improve their LED power supply current configuration application

Table of contents

Table of contents

1 Introduction

In this application note, we will discuss and show you how to use NLM0011 to implement two novel features in the LED power supplier: NFC current configuration and constant lumen output (CLO). In LED applications, different LED modules typically have different supply current requirements. Therefore, the output current of a constant current LED power supplier should be configurable. The traditional way to do this was with a plug-in resistor or DIP switches, but they are labor intensive and inflexible. Recently, the advantages of NFC-based wireless configuration were recognized by the LED lighting industry. The first generation of LED power suppliers with NFC configuration features was launched using a µC and NFC tag. In March 2018, the Module-Driver Interface special interest group (MD-SIG) consortium specified a common approach to wireless and mains-voltage-free programming of LED drivers using NFC. (NFC programming of LED driver parameters is standardized by MD-SIG – http://md-sig.org/nfc-programming-of-led-driver-parameters-is-sta ndardized-by-md-sig/.) However, in the cost-sensitive, high-volume mid to low-end LED power supplier market, a solution using µC plus NFC tag is too expensive. Another hot topic in the LED power supplier market is the CLO function. With this function, the luminous flux drop over the entire life of the LED can be compensated. A constant light output of the LED module is achieved by the LED power supplier, which stores the operating hours of the LED module and responds by increasing the output current to the drop in light output. However, the solution currently available for CLO implementation is the expensive µC-based one.

NLM0011 is a dual-mode NFC configuration IC with PWM output primarily designed for LED applications to implement NFC wireless LED current configuration and CLO without the need for an additional µC.

Because it is designed to work with mainstream analog driver ICs, there is minimal effort needed to adopt it into an existing or new design.

2 Application overview

NLM0011 has two operating modes: passive mode and active mode. In passive mode, where the LED driver module is not powered, parameters related to PWM can be configured wirelessly via the NFC interface. In active mode, as soon as the LED driver module is powered, a PWM output is generated according to the stored parameters. Please note that during start up of the lighting mode defined supply voltage is necessary to ensure smooth operation. With an external R/C filter, the PWM signal is converted to a desired DC voltage to control the current output of the LED driver module. With an integrated operation time counter (OTC) and stored LED degradation curve in the CLO table, the PWM signal can be automatically adjusted to compensate for the LED degradation.
By disabling the RF function in active mode, this avoids the NFC-enabled LED power supplier being classified as a radio device, which has restricted EMI and EMC test and certification requirements.

The intended application use case is explained here. The application use case of NLM0011/NLM0010 in lighting operation mode looks as shown below:

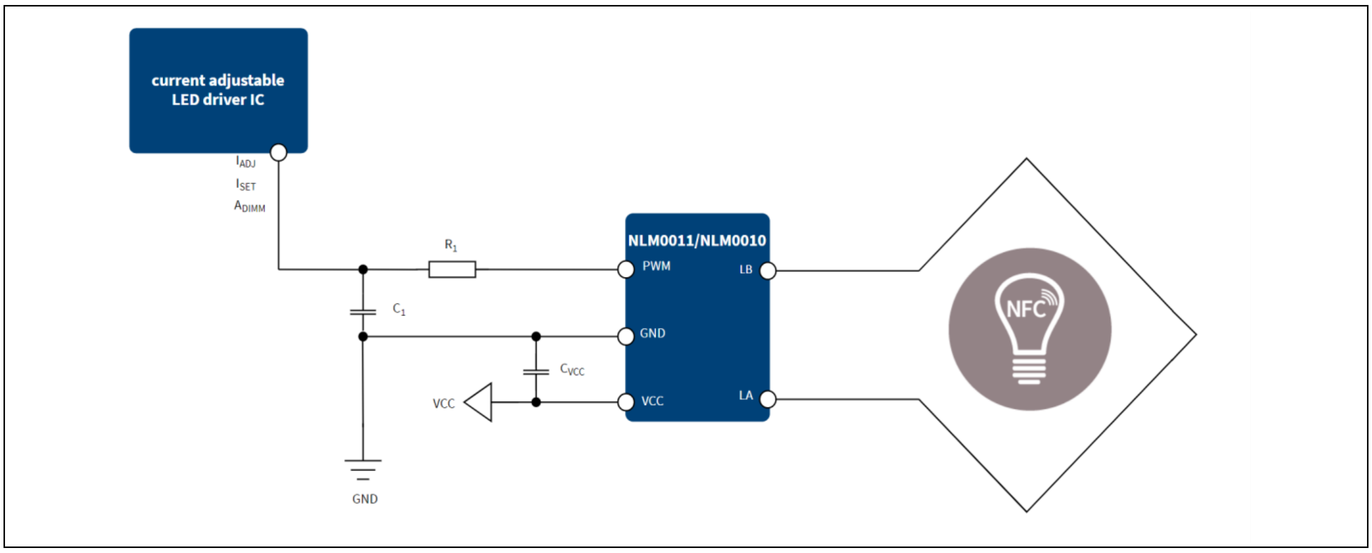


Figure 1 NLM0011/NLM0010 application view

The NLM0011/NLM0010 chip is connected to the LED driver circuit. The driver circuit controls the current of the LEDs. The set-point value of the current is stored in the integrated non-volatile memory (NVM) of the NLM0011/NLM0010. According to the NVM content the NLM0011/NLM0010 generates a PWM output, which corresponds to the value of the set-point. Usually the PWM data pulses are converted into an analog voltage by an external R/C circuitry. This analog voltage is the reference set-point for the LED current regulation loop of the module. The analog value of the set-point is determined by the duty cycle of the PWM output, the time constant of the R/C circuitry and the voltage swing of the PWM output driver. The NLM0011/NLM0010 provides only the set-point definition of the LED current of the regulation loop. The NLM0011/NLM0010 does not execute the regulation loop or any parts of the regulation loop. Within LED lighting systems the set-point definition is usually called “ISET”. Some LED drivers/controllers accept the direct PWM input and convert the PWM data into a set-point value internally. In that case no R/C circuitry is needed between the NLM0011/NLM0010 and the LED driver.

The parameters of the PWM output are period and duty cycle. Both must be configured during NFC configuration. The values of the configuration parameters have to be calculated according to the requirements of the application. Using an external R/C circuitry with a given time constant, the duty cycle and period of the PWM can be calculated to generate any value between minimum and maximum output voltage of the digital PWM output driver. The calculated parameters including the aging function are written into the NVM of the

NLM0011/NLM0010 during NFC configuration. That can only be done if the NLM0011/NLM0010 is powered off. The NLM0011/NLM0010 will harvest the energy from the electromagnetic NFC field and supply itself in that case.

Attention: ***It is important that the NFC connection is disconnected after storing values in NLM0011/NLM0010, before V_{cc} is turned on, otherwise PWM will always be at 100 percent. This can be achieved by moving the antenna away.***

Attention: ***During start up of the lighting mode the system is configured to the correct PWM value. Thus the supply voltage during the start up phase needs to be monotonous rising and needs to be within the specified voltage range for the device to ensure correct operation.***

3 Dimensioning the peripheral components

3.1 Dimensioning the filter

In case an analog DC voltage is used to control the LED driver, R/C circuitry is needed between the NLM0011/NLM0010 and the LED driver, as shown in Figure 1, to convert the PWM signal to an analog voltage.

In the dimensioning of the R/C filter, one important design parameter is the desired cut-off frequency, which is linked to the selected PWM frequency. In case a passive Low Pass Filter (LPF) of first order is used, the stop band roll-off rate is -20 dB per decade. We recommend dimensioning the LPF with minimum -40 dB stop band attenuation at the selected PWM frequency. This means the minimum suggested cut-off frequency is $f_{PWM}/100$.

After the PWM frequency is decided by consideration of the necessary duty cycle resolution (chapter 4.1), the cut-off frequency can be calculated:

$$f_{cut-off} < \frac{f_{PWM}}{100}$$

Afterwards, the RC value can be calculated:

$$f_{cut-off} = \frac{1}{2 \cdot \pi \cdot \tau}$$

$$f_{cut-off} = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

$$RC = \tau = \frac{1}{2 \cdot \pi \cdot f_{cut-off}}$$

From this value, the filter components and values can be derived.

Attention: ***The higher f_{PWM} is, the smaller and less expensive C can be. But the duty cycle resolution decreases at the same time. With a 27 MHz internal clock, NLM0011/NLM0010 offers 9.82 bit resolution at the PWM frequency of 30 kHz.***

Example:

Let us assume f_{PWM} = 15 kHz.

$$f_{cut-off} < \frac{15\ kHz}{100} = 150\ Hz$$

$$RC = \tau = \frac{1}{2 \cdot \pi \cdot f_{cut-off}} = 1{,}06 \cdot 10^{-3} s$$

➔ R = 20 kΩ

➔ C = 50 nF

3.2 Dimensioning the V_{cc} capacitor

According to the datasheet the minimum value for the V_{cc} capacitor (as shown in Figure 1 as C_{VCC}) is given as 22 µF. This value needs to be respected, because if the supply voltage V_{cc} is switched off, NLM0011/NLM0010 detects Under Voltage Lockout (UVLO), and the last values for OTC and on/off switching need a certain amount of time to be written to the internal memory. For this, the voltage must be buffered over this duration. If the value of this capacitor is less than 22 µF, there is no guarantee of correctness for these values across all ranges (especially temperature range).

In case of using NLM0010, which does not support CLO, the value of C_{VCC} must be a minimum of 22 µF, because on/off counting and OTC also need to write the last values to the internal memory.

Only in the case of CLO, if on/off counting and OTC are not used, a smaller value for C_{VCC} is sufficient. A value of 100 nF is recommended.

3.3 Impedance converter

In case the control input for PWM has a low resistance, it is recommended to use an impedance converter, e.g. an operational amplifier. Otherwise it will not be required.

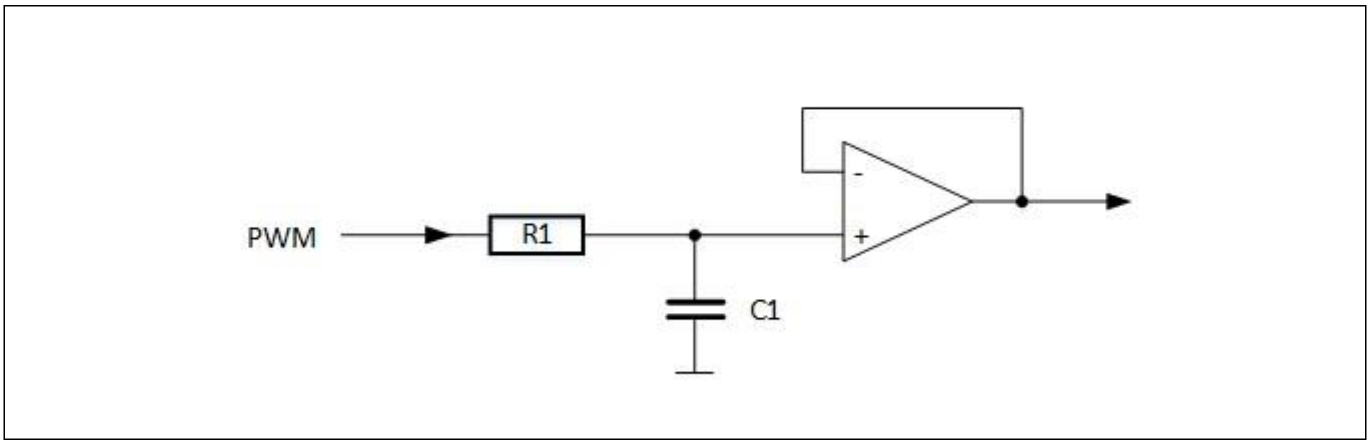


Figure 2 Convert PWM signal using an amplifier

4 Implementation of customer functions, full example

A full example of how to set all parameters is given here:

4.1 Configure PWM

The built-in clock frequency = 27.12 MHz (2 x 13.56 MHz NFC frequency). For a 30 kHz PWM signal, a 27 MHz clock is sufficient to have a 1:900 resolution for duty cycle adjustment.

In this step, it must be decided which resolution is required for the system.

Let us assume a duty cycle resolution of 1:1800, with n = 1800.

The equation for this calculation is:

$$f_{PWM} = \frac{27{,}12\ MHz}{n}$$

So the calculation looks like this:

$$f_{PWM} = \frac{27{,}12\ MHz}{n} = \frac{27{,}12\ MHz}{1800} = 15\ kHz$$

The length of period of the PWM signal T_{PWM} is derived from the content of NVM block 0 B_H (byte (3:2)). This 16-bit value represents the number of internal clock ticks required to achieve the desired frequency.

This value must be stored in NVM as T_{PWM} (PWM period I_{max}) at 0 B_H (31:16). To get the binary code, the following calculation must be done:

$$value\ T_{PWM} = \frac{27120\ kHz}{f_{PWM}} \rightarrow integer\ rounding \rightarrow change\ to\ hexadecimal/binary$$

In our example:

$$value\ T_{PWM} = \frac{27120\ kHz}{15\ kHz} = 1808 \rightarrow (integer\ rounding \rightarrow 1808) \rightarrow 0710_H \rightarrow 11100010000_B$$

In the next step, I_{max} must be configured. The equation for this is:

$$duty\ cycle = \frac{m}{n}$$

m = T_{Imax} (0 – 65536, 2 bytes)

n = duty cycle resolution

In this example, let us assume m = 1600, so the calculation looks like this:

$$Duty\ cycle = \frac{m}{n} = \frac{1600}{1800} = 0{,}83 \rightarrow duty\ cycle = 83\%$$

The maximum value of T_{High} is derived from the content of NVM block 00_H (byte (3:2)). This 16-bit value represents the number of internal clock ticks required to achieve the desired logical high output time. It also represents the value of 100 percent duty cycle according to the CLO table.

This value must be stored in NVM as duty cycle I_{max} at 00_H (31:16). To get the binary code, the following calculation must be done:

$$Duty\ cycle\ I_{max} = \frac{T_{PWM}}{100\%} \cdot duty\ cycle\ in\ \% \rightarrow integer\ rounding\ \rightarrow change\ to\ hexadecimal/binary$$

In our example:

$$Duty\ cycle\ I_{max} = \frac{T_{PWM}}{100\%} \cdot 83\% = 1500{,}64 \rightarrow integer\ rounding \rightarrow\ 1501 \rightarrow 05DD_H \rightarrow 10111011101_B$$

Note: *The CLO calculation unit of the chip finally adapts this value according to the current interpolation result.*

Referring to our example: Let us assume the CLO interpolation unit calculates a current duty cycle of 80 percent (or 102 parts of 128), then the effective current duty cycle will be 83 percent of 80 percent = 66.4 percent.

4.2 Setting up the CLO table

To setup the parameters for the CLO table, three steps must be followed.

In the first step the function of LED degradation must be determined. In the second step, the function for CLO is derived (eight points must be set). And in the third step, the values for CLO are calculated.

Both monotonous rising and falling curves are supported.

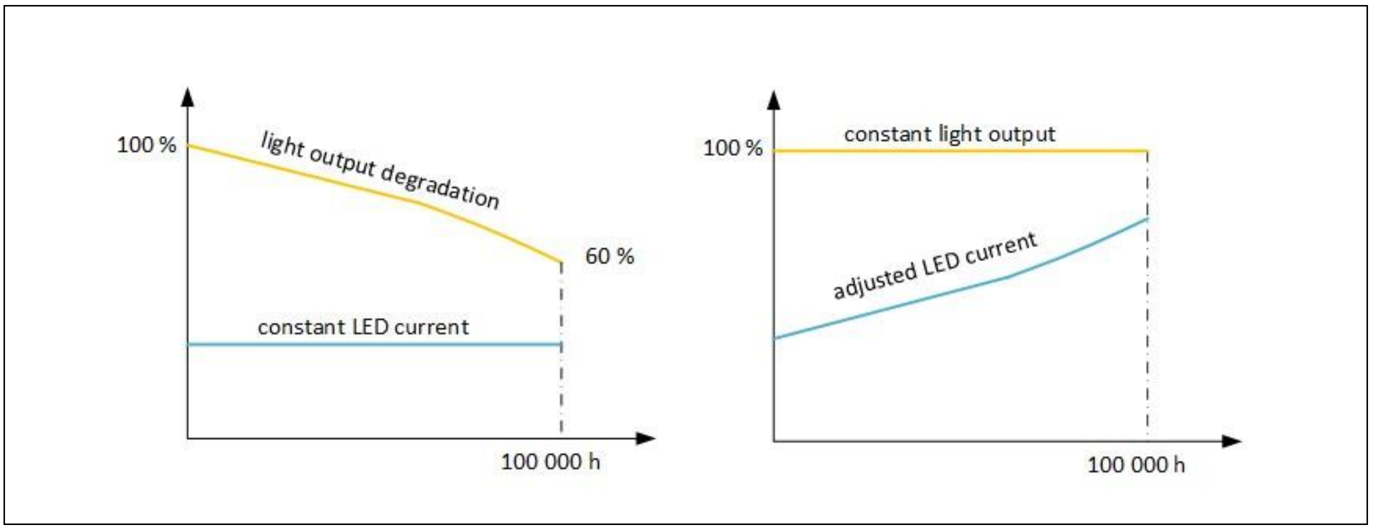
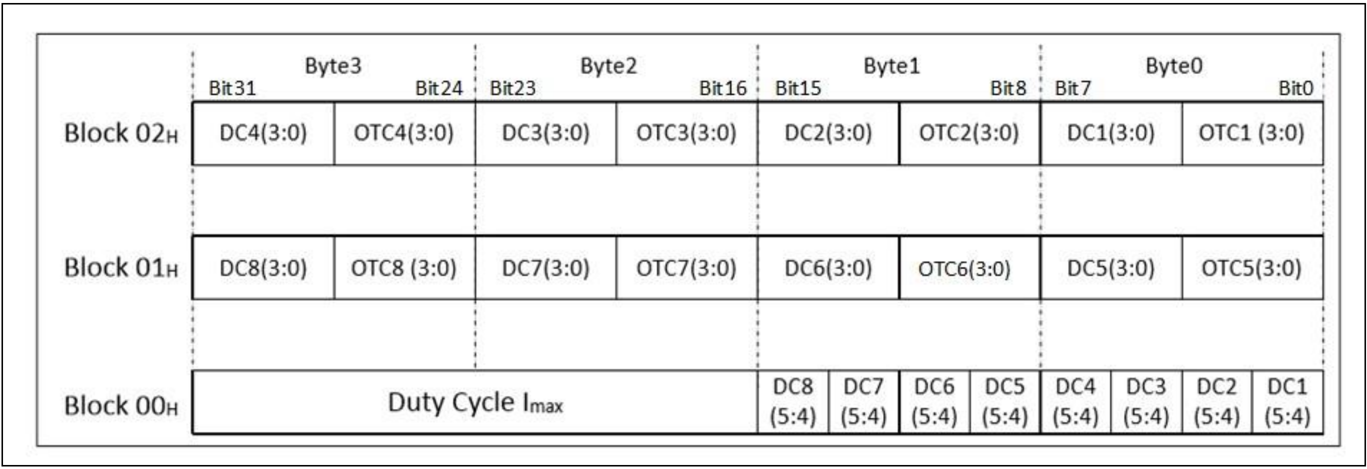


Figure 3 Light output degradation and LED current adjustment

Setting the values for the CLO table

To use the CLO feature of NLM0011, the CLO table must be set as shown:



	Byte3 Bit31	Byte3 Bit24	Byte2 Bit23	Byte2 Bit16	Byte1 Bit15	Byte1 Bit8	Byte0 Bit7	Byte0 Bit0
Block 02H	DC4(3:0)	OTC4(3:0)	DC3(3:0)	OTC3(3:0)	DC2(3:0)	OTC2(3:0)	DC1(3:0)	OTC1 (3:0)
Block 01H	DC8(3:0)	OTC8 (3:0)	DC7(3:0)	OTC7(3:0)	DC6(3:0)	OTC6(3:0)	DC5(3:0)	OTC5(3:0)
Block 00H	Duty Cycle Imax				DC8 (5:4) DC7 (5:4)	DC6 (5:4) DC5 (5:4)	DC4 (5:4) DC3 (5:4)	DC2 (5:4) DC1 (5:4)

Figure 4 CLO table memory layout

For duty cycle calculation the following inputs are needed:

- From NVM (read out during power-on start-up from NVM and stored into registers in lighting operational mode):
  - The reference points of the aging function:
    - The reference values of the operating time. Eight values of the reference operating time must be read out. A single reference value of the operating time is stored in 4-bit format. Reference values for the operating time are represented as multiples of 8192 (reference operating time in hours is equal to the 4-bit value from NVM multiplied by 8192). Therefore the reference values for the operating time cover the range from 0 h to 122880 h (in steps of multiples of 8192 h).
    - The duty cycle reference values. Eight values of the reference duty cycle belonging to the respective operating time reference value must be read out. A single reference value of the duty cycle is stored in 6-bit format. The 6-bit value represents the off-set value of parts of 128 to 64/128 for rising curves and 64/128 to 0/128 for falling curves. Therefore the reference values for the duty cycle cover a range from 64/128 (→ 50 percent) to 128/128 (→ 100 percent) or 64/128 (-> 50 percent) to 0/128 (-> 0 percent).
    - The maximum duty cycle definition belonging to the PWM period. This is an unsigned 16-bit integer value (duty cycle I_{max} has been set in the previous chapter at 00_H (3:2)).
  - From OTC, the operating time in hours. This is a 16-bit value, which represents the operating time. The counter is incremented every four hours. Therefore the operating time in hours is the multiplication of this 16-bit value by 4. In case of power-up the chip restores the current operating time from the NVM into the OTC.

4.2.1 Setting the reference points of the aging function for rising curves

Calculating reference values of the operating time and duty cycle reference values:

The first point of the OTC in the CLO table shall be set to 0 hours. Thus for our example let us assume a reference point at an operating time of 0 hours and a target duty cycle of 82 percent of the duty cycle definition of duty cycle I_{max} (value of block 00_H, byte (3:2), 83 percent). Here is how to calculate the 10-bit reference value of the particular reference point to be stored in the NVM:

- The 4-bit value of OTC is calculated using this equation:

  $$4\ bit\ value\ OTC = \frac{operating\ hours}{8192} \rightarrow integer\ rounding\ \rightarrow change\ to\ hexadecimal/binary$$

  In our example: $\frac{0}{8192} = 0 \rightarrow integer\ rounding = 0 \rightarrow 0000_B$

  Now the value can be stored in NVM at OTC1(3:0) 02_H (3:0).

- The 6-bit value of the duty cycle:

  Convert the given percentage value into parts of 128, do an integer rounding and calculate the off-set as shown in this equation:

  $$6\ bit\ value\ DC\colon \frac{DC\ \%}{100} = \frac{x}{128} \rightarrow x\ rounding\ integer - 64 \rightarrow change\ to\ hexadecimal/binary$$

  In our example: $\frac{82}{100} = \frac{x}{128} \rightarrow x = 104.96 \rightarrow integer\ rounding = 105 - 64 = 41 \rightarrow 101001_B$

  Now the value can be stored in NVM at DC1(3:0) and DC1(5:4), 02_H (7:4) and 00_H (1:0).

According to this scheme all points of the CLO function can be calculated, until all eight points are set.

<u>For our example, the following calculations must be done:</u>

- CLO point 1 (as calculated in this chapter):
  - OTC1(3:0): $0000_B \rightarrow 02_H$ (3:0)
  - DC1(3:0), DC1(5:4): $101001_B \rightarrow 02_H$ (7:4) and 00_H (1:0)
- CLO point 2:
  - OTC2(3:0):

    Let us assume the second point at 75000 hours.

    $$\frac{75000}{8192} = 9.16 \rightarrow integer\ rounding = 9 \rightarrow 1001_B$$

    $1001_B \rightarrow$ OTC2(3:0) $\rightarrow$ 02_H (11:8)
  - DC2(0:3), DC2(5:4):

    Let us assume a duty cycle of 85 percent of duty cycle I_{max} at the second point.

    $$\frac{85}{100} = \frac{x}{128} \rightarrow x = 108.8 \rightarrow integer\ rounding = 109 - 64 = 45 \rightarrow 101101_B$$

    $101101_B \rightarrow$ DC2(0:3), DC2(5:4) $\rightarrow 02_H$ (15:12), 00_H (3:2)
- CLO point 3:
  - OTC3(3:0):

    Let us assume point 3 at 82000 hours.

    $$\frac{82000}{8192} = 10.01 \rightarrow integer\ rounding = 10 \rightarrow 1010_B$$

    $1010_B \rightarrow$ OTC3(3:0) $\rightarrow$ 02_H (19:16)

- DC3(0:3), DC3(5:4):

  Let us assume a duty cycle of 87 percent of duty cycle I_{max} at point 3.

  $$\frac{87}{100} = \frac{x}{128} \rightarrow x = 111.36 \rightarrow integer\ rounding = 111 - 64 = 47 \rightarrow 101111_B$$

  101111_B → DC3(0:3), DC3(5:4) → 02_H (23:20), 00_H (5:4)

- CLO point 4:
  - OTC4(3:0):

    Let us assume point 4 at 87000 hours.

    $$\frac{87000}{8192} = 10.6 \rightarrow integer\ rounding = 11 \rightarrow 1011_B$$

    1011_B → OTC4(3:0) → 02_H (27:24)

  - DC4(0:3), DC4(5:4):

    Let us assume a duty cycle of 88 percent of duty cycle I_{max} at point 4.

    $$\frac{88}{100} = \frac{x}{128} \rightarrow x = 112.64 \rightarrow integer\ rounding = 113 - 64 = 49 \rightarrow 110001_B$$

    110001_B → DC4(0:3), DC4(5:4) → 02_H (31:28), 00_H (7:6)

- CLO point 5:
  - OTC5(3:0):

    Let us assume point 5 at 96000 hours.

    $$\frac{96000}{8192} = 11.72 \rightarrow integer\ rounding = 12 \rightarrow 1100_B$$

    1100_B → OTC5(3:0) → 01_H (3:0)

  - DC5(0:3), DC5(5:4):

    Let us assume a duty cycle of 90 percent of duty cycle I_{max} at point 5.

    $$\frac{90}{100} = \frac{x}{128} \rightarrow x = 115.2 \rightarrow integer\ rounding = 115 - 64 = 51 \rightarrow 110011_B$$

    110011_B → DC5(0:3), DC5(5:4) → 01_H (7:4), 00_H (9:8)

- CLO point 6:
  - OTC6(3:0):

    Let us assume point 6 at 105000 hours.

    $$\frac{105000}{8192} = 12.82 \rightarrow integer\ rounding = 13 \rightarrow 1101_B$$

    1101_B → OTC6(3:0) → 01_H (11:8)

  - DC6(0:3), DC6(5:4):

    Let us assume a duty cycle of 91 percent of duty cycle I_{max} at point 6.

$$\frac{91}{100} = \frac{x}{128} \rightarrow x = 116.48 \rightarrow integer\ rounding = 116 - 64 = 52 \rightarrow 110100_B$$

110100_B → DC6(0:3), DC6(5:4) → 01_H (15:12), 00_H (11:10)

- CLO point 7:
  - OTC7(3:0):

    Let us assume point 7 at 115000 hours.

    $$\frac{115000}{8192} = 14.04 \rightarrow integer\ rounding = 14 \rightarrow 1110_B$$

    1110_B → OTC7(3:0) → 01_H (19:16)
  - DC7(0:3), DC7(5:4):

    Let us assume a duty cycle of 93 percent of duty cycle I_{max} at point 7.

    $$\frac{93}{100} = \frac{x}{128} \rightarrow x = 119.04 \rightarrow integer\ rounding = 119 - 64 = 55 \rightarrow 110111_B$$

    110111_B → DC7(0:3), DC7(5:4) → 01_H (23:20), 00_H (13:12)

- CLO point 8:
  - OTC8(3:0):

    Let us assume point 8 at 120000 hours.

    $$\frac{120000}{8192} = 14.64 \rightarrow integer\ rounding = 15 \rightarrow 1111_B$$

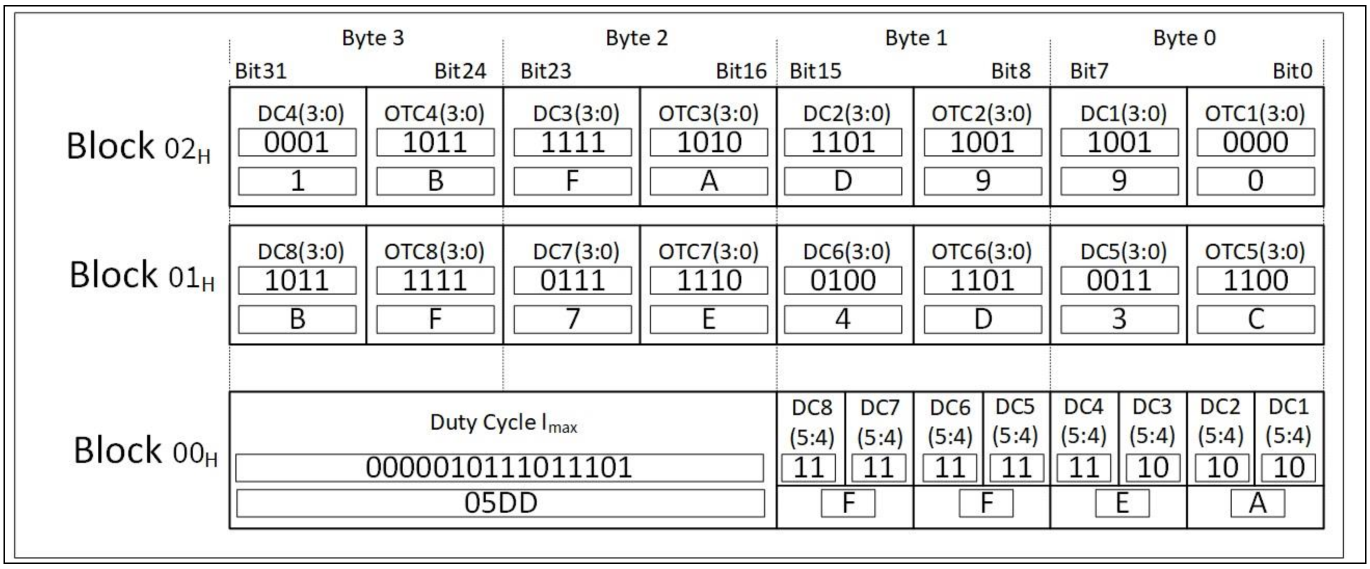
    1111_B → OTC8(3:0) → 01_H (27:24)
  - DC8(0:3), DC8(5:4):

    Let us assume a duty cycle of 96 percent of duty cycle I_{max} at point 8.

    $$\frac{96}{100} = \frac{x}{128} \rightarrow x = 122.88 \rightarrow integer\ rounding = 123 - 64 = 59 \rightarrow 111011_B$$

    111011_B → DC8(0:3), DC8(5:4) → 01_H (31:28), 00_H (15:14)

<u>A fully completed CLO table looks like this for our example:</u>



	Byte 3		Byte 2		Byte 1		Byte 0	
	Bit31	Bit24	Bit23	Bit16	Bit15	Bit8	Bit7	Bit0
Block 02_H	DC4(3:0) 0001 1	OTC4(3:0) 1011 B	DC3(3:0) 1111 F	OTC3(3:0) 1010 A	DC2(3:0) 1101 D	OTC2(3:0) 1001 9	DC1(3:0) 1001 9	OTC1(3:0) 0000 0
Block 01_H	DC8(3:0) 1011 B	OTC8(3:0) 1111 F	DC7(3:0) 0111 7	OTC7(3:0) 1110 E	DC6(3:0) 0100 4	OTC6(3:0) 1101 D	DC5(3:0) 0011 3	OTC5(3:0) 1100 C

	Duty Cycle I_{max}	DC8 (5:4)	DC7 (5:4)	DC6 (5:4)	DC5 (5:4)	DC4 (5:4)	DC3 (5:4)	DC2 (5:4)	DC1 (5:4)
Block 00_H	0000010111011101	11	11	11	11	11	10	10	10
	05DD	F		F		E		A	

Figure 5 **Complete CLO table for our example**

The chip is continuously counting the operating time and is continuously interpolating the resulting duty cycle. The interpolation result is then multiplied by the value of duty cycle I_{max} (value of block 00_H, byte (3:2)) and this product is then divided by 128. This value is then used by the PWM generator and generates a PWM signal of frequency PWM period I_{max} (value of block $0B_H$, byte (3:2)) with the calculated duty cycle.

To disable the CLO function, use a constant 100 percent duty cycle value in the CLO table.

4.2.2 Setting the reference points of the aging function for falling curves

Monotonous falling CLO curves are supported in NLM0011 in chip version A14 (indicated by family code byte value $B9_H$). In order to enable a falling CLO curve the Bit "FR" (bit 31 of NVM block address $0B_H$) must be programmed to value 1_B (for rising CLO curves the default 0_B has to be used. In case of falling CLO curve the covered CLO range is 64 parts of 128 down to 0 of duty cycle I_{max} (for rising CLO curve the covered range is 64 parts of 128 to 128 parts of 128 of duty cycle I_{max}).
The reference point duty cycle value for a falling CLO curve has to be entered as :

$$CLO\ reference\ point\ (falling\ curve) = 64 - < desired\ value\ (0 \ldots 64) >$$

Example:

Table 1 **CLO duty cycle reference values**

Reference point number	**Desired value (parts of 128 of duty cycle I_{max})**	**CLO table value**
Point 1	64	0
Point 2	56	8
Point 3	42	22
Point 4	31	33
Point 5	27	37
Point 6	19	45
Point 7	13	51
Point 8	3	61

Attention: ***Only strict monotonous falling or rising CLO curves are supported.***

In the next step, the value must be converted to hexadecimal/binary.

The calculation for OTC of each point is the same as for rising curves. So the hexadecimal/binary values can be stored in the memory as shown for rising curves.

Calculation example for CLO points for falling curve:

Let us assume the same points for OTC as in the example for rising curves.

The DC points must be calculated like that:

$$6\ bit\ value\ DC(for\ falling\ curve): \frac{DC\ \%}{100} = \frac{x}{128} \rightarrow x\ rounding\ integer; 64 - x\ (integer\ rounded) \rightarrow change\ to\ hexadecimal/binary$$

- CLO point 1:

  Also in case of falling curve, the first point for OTC in CLO shall be set to 0 hours.

  - OTC1(3:0): $0000_B \rightarrow 02_H$ (3:0)
  - DC1(3:0), DC1(5:4):

    Let us assume a duty cycle of 50 percent of duty cycle I_{max}.

    $$\frac{50}{100} = \frac{x}{128} \rightarrow x = 64 \rightarrow integer\ rounding = 64 - 64 = 0 \rightarrow 000000_B$$

    $000000_B \rightarrow 02_H$ (7:4) and 00_H (1:0)

- CLO point 2:
  - OTC2(3:0):

    Let us assume point 2 at 75000 hours.

    $$\frac{75000}{8192} = 9.16 \rightarrow integer\ rounding = 9 \rightarrow 1001_B$$

    $1001_B \rightarrow$ OTC2(3:0) $\rightarrow$ 02_H (11:8)

  - DC2(0:3), DC2(5:4):

    Let us assume a duty cycle of 44 percent of duty cycle I_{max} at point 2.

    $$\frac{44}{100} = \frac{x}{128} \rightarrow x = 56.32 \rightarrow integer\ rounding = 64 - 56 = 8 \rightarrow 001000_B$$

    $001000_B \rightarrow$ DC2(0:3), DC2(5:4) $\rightarrow 02_H$ (15:12), 00_H (3:2)

- CLO point 3:
  - OTC3(3:0):

    Let us assume point 3 at 82000 hours.

    $$\frac{82000}{8192} = 10.01 \rightarrow integer\ rounding = 10 \rightarrow 1010_B$$

1010_B → OTC3(3:0) → 02$_H$ (19:16)

- DC3(0:3), DC3(5:4):

  Let us assume a duty cycle of 33 percent of duty cycle I$_{max}$ at point 3.

  $$\frac{33}{100} = \frac{x}{128} \to x = 42.24 \to integer\ rounding = 64 - 42 = 22 \to 010110_B$$

  010110_B → DC3(0:3), DC3(5:4) → 02$_H$ (23:20), 00$_H$ (5:4)

- CLO point 4:
  - OTC4(3:0):

    Let us assume point 4 at 87000 hours.

    $$\frac{87000}{8192} = 10.6 \to integer\ rounding = 11 \to 1011_B$$

    1011_B → OTC4(3:0) → 02$_H$ (27:24)

  - DC4(0:3), DC4(5:4):

    Let us assume a duty cycle of 24 percent of duty cycle I$_{max}$ at point 4.

    $$\frac{24}{100} = \frac{x}{128} \to x = 30.72 \to integer\ rounding = 64 - 31 = 33 \to 100001_B$$

    100001_B → DC4(0:3), DC4(5:4) → 02$_H$ (31:28), 00$_H$ (7:6)

- CLO point 5:
  - OTC5(3:0):

    Let us assume point 5 at 96000 hours.

    $$\frac{96000}{8192} = 11.72 \to integer\ rounding = 12 \to 1100_B$$

    1100_B → OTC5(3:0) → 01$_H$ (3:0)

  - DC5(0:3), DC5(5:4):

    Let us assume a duty cycle of 21 percent of duty cycle I$_{max}$ at point 5.

    $$\frac{21}{100} = \frac{x}{128} \to x = 26.88 \to integer\ rounding = 64 - 27 = 37 \to 100101_B$$

    100101_B → DC5(0:3), DC5(5:4) → 01$_H$ (7:4), 00$_H$ (9:8)

- CLO point 6:
  - OTC6(3:0):

    Let us assume point 6 at 105000 hours.

    $$\frac{105000}{8192} = 12.82 \to integer\ rounding = 13 \to 1101_B$$

    1101_B → OTC6(3:0) → 01$_H$ (11:8)

  - DC6(0:3), DC6(5:4):

Let us assume a duty cycle of 15 percent of duty cycle I_{max} at point 6.

$$\frac{15}{100} = \frac{x}{128} \rightarrow x = 19.2 \rightarrow integer\ rounding = 64 - 19 = 45 \rightarrow 101101_B$$

101101_B → DC6(0:3), DC6(5:4) → 01_H (15:12), 00_H (11:10)

- CLO point 7:
  - OTC7(3:0):

    Let us assume point 7 at 115000 hours.

    $$\frac{115000}{8192} = 14.04 \rightarrow integer\ rounding = 14 \rightarrow 1110_B$$

    1110_B → OTC7(3:0) → 01_H (19:16)
  - DC7(0:3), DC7(5:4):

    Let us assume a duty cycle of 10 percent of duty cycle I_{max} at point 7.

    $$\frac{10}{100} = \frac{x}{128} \rightarrow x = 12.8 \rightarrow integer\ rounding = 64 - 13 = 54 \rightarrow 110011_B$$

    110011_B → DC7(0:3), DC7(5:4) → 01_H (23:20), 00_H (13:12)

- CLO point 8:
  - OTC8(3:0):

    Let us assume point 8 at 120000 hours.

    $$\frac{120000}{8192} = 14.64 \rightarrow integer\ rounding = 15 \rightarrow 1111_B$$

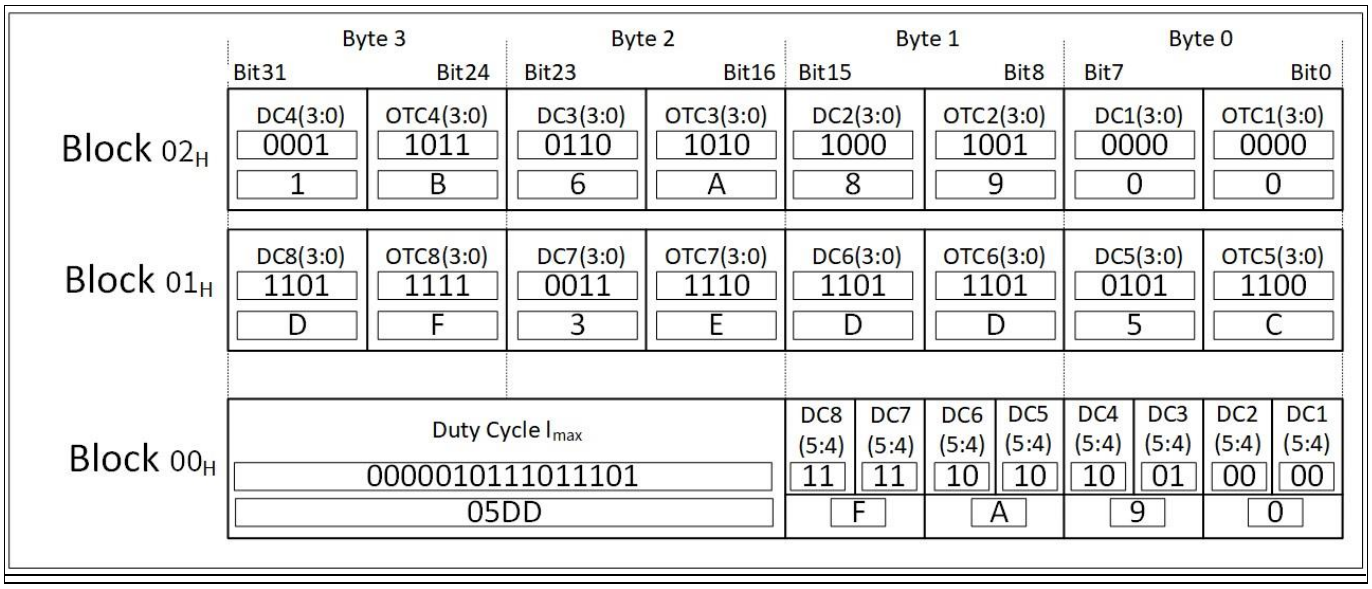
    1111_B → OTC8(3:0) → 01_H (27:24)
  - DC8(0:3), DC8(5:4):

    Let us assume a duty cycle of 2 percent of duty cycle I_{max} at point 8.

    $$\frac{2}{100} = \frac{x}{128} \rightarrow x = 2.56 \rightarrow integer\ rounding = 64 - 3 = 61 \rightarrow 111101_B$$

    111101_B → DC8(0:3), DC8(5:4) → 01_H (31:28), 00_H (15:14)

A fully completed CLO table looks like this for our example for falling curve:



	Byte 3		Byte 2		Byte 1		Byte 0	
	Bit31	Bit24	Bit23	Bit16	Bit15	Bit8	Bit7	Bit0
Block 02$_H$	DC4(3:0)	OTC4(3:0)	DC3(3:0)	OTC3(3:0)	DC2(3:0)	OTC2(3:0)	DC1(3:0)	OTC1(3:0)
	0001	1011	0110	1010	1000	1001	0000	0000
	1	B	6	A	8	9	0	0
Block 01$_H$	DC8(3:0)	OTC8(3:0)	DC7(3:0)	OTC7(3:0)	DC6(3:0)	OTC6(3:0)	DC5(3:0)	OTC5(3:0)
	1101	1111	0011	1110	1101	1101	0101	1100
	D	F	3	E	D	D	5	C

	Duty Cycle I_{max}	DC8 (5:4)	DC7 (5:4)	DC6 (5:4)	DC5 (5:4)	DC4 (5:4)	DC3 (5:4)	DC2 (5:4)	DC1 (5:4)
Block 00$_H$	0000010111011101	11	11	10	10	10	01	00	00
	05DD	F		A		9		0	

Figure 6 **Complete CLO table for our example for falling curve**

4.3 Configure parameters by using the NLM-PWM mobile app

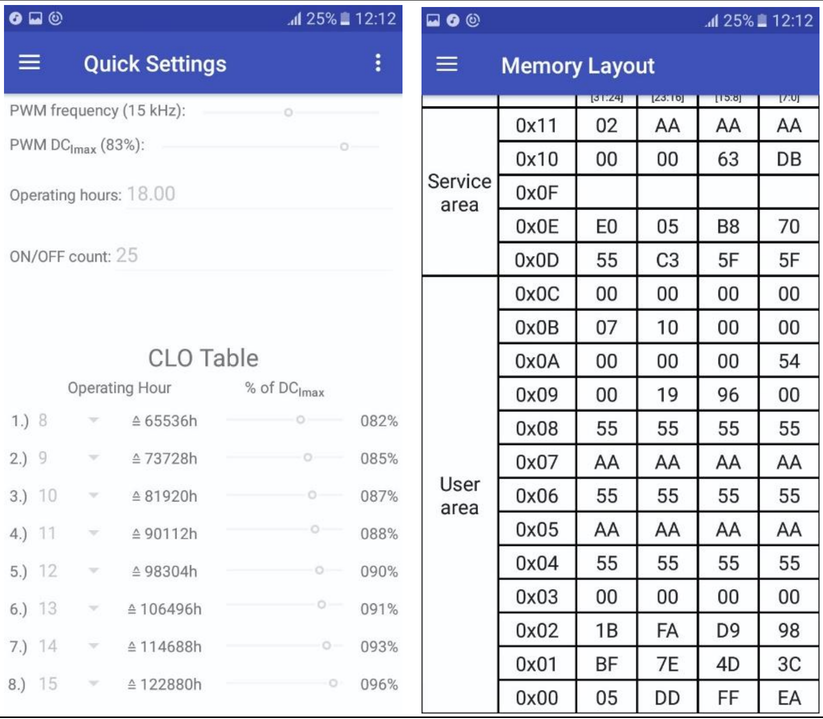
It is also possible to calculate the hexadecimal code for PWM frequency, duty cycle I_{max} and all set-points of the CLO table by using the Infineon NLM-PWM mobile app on Android smartphones.

Attention: ***Since the energy transmitted by different smartphones, a smartphone shall be used which provides enough energy via NFC.***

For that, the values for PWM frequency, T_{PWM} (PWM period I_{max}), duty cycle I_{max} and all set-points of the CLO table must be set in the mobile app and written on the IC (e.g. with Infineon's development kit for NLM0011). Then read out the IC and switch to "Memory Layout".

Note: *Because of internal rounding it is not possible to set exactly 100 percent for PWM $DC_{max\,it}$ in the mobile app.*

Attention: ***The mobile app is not intended for professional commercial use (e.g., production, field maintenance). It is intended for demonstration purposes only.***



25% 12:12

Quick Settings

PWM frequency (15 kHz):

PWM DC_{Imax} (83%):

Operating hours: 18.00

ON/OFF count: 25

CLO Table

	Operating Hour		% of DC_{Imax}
1.)	8	≙ 65536h	082%
2.)	9	≙ 73728h	085%
3.)	10	≙ 81920h	087%
4.)	11	≙ 90112h	088%
5.)	12	≙ 98304h	090%
6.)	13	≙ 106496h	091%
7.)	14	≙ 114688h	093%
8.)	15	≙ 122880h	096%

25% 12:12

Memory Layout

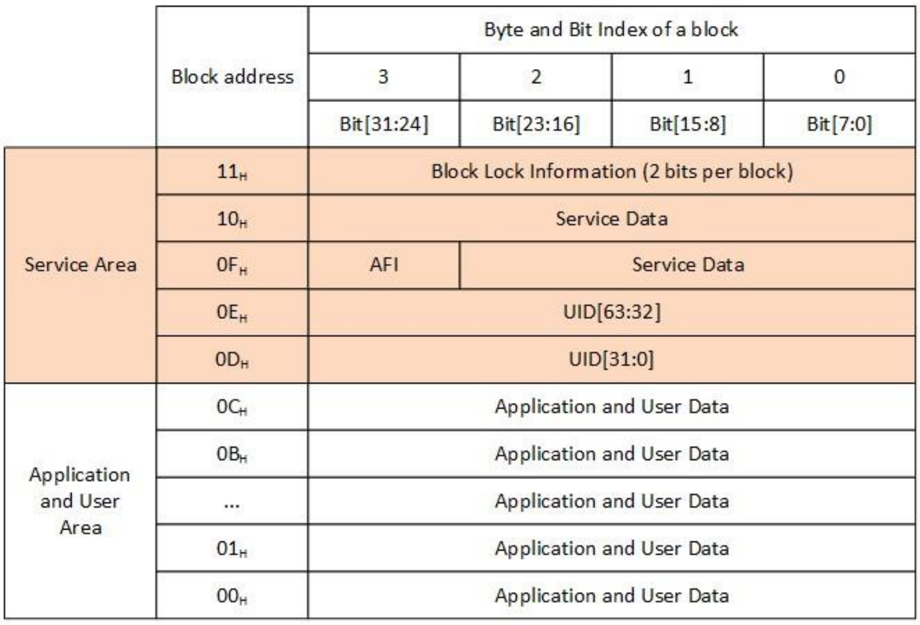
		[31:24]	[23:16]	[15:8]	[7:0]
Service area	0x11	02	AA	AA	AA
	0x10	00	00	63	DB
	0x0F				
	0x0E	E0	05	B8	70
	0x0D	55	C3	5F	5F
User area	0x0C	00	00	00	00
	0x0B	07	10	00	00
	0x0A	00	00	00	54
	0x09	00	19	96	00
	0x08	55	55	55	55
	0x07	AA	AA	AA	AA
	0x06	55	55	55	55
	0x05	AA	AA	AA	AA
	0x04	55	55	55	55
	0x03	00	00	00	00
	0x02	1B	FA	D9	98
	0x01	BF	7E	4D	3C
	0x00	05	DD	FF	EA

Figure 7 Set parameters and memory layout in NLM-PWM mobile app

5 Memory control

5.1 Memory structure, and application and user area

The memory is separated in two areas: the service area, where Infineon stores internal values, and the application and user area, where the LED driver maker and LED lamp maker can store their values.



	Block address	Byte and Bit Index of a block			
		3	2	1	0
		Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]
Service Area	11_H	Block Lock Information (2 bits per block)			
	10_H	Service Data			
	$0F_H$	AFI	Service Data		
	$0E_H$	UID[63:32]			
	$0D_H$	UID[31:0]			
Application and User Area	$0C_H$	Application and User Data			
	$0B_H$	Application and User Data			
	...	Application and User Data			
	01_H	Application and User Data			
	00_H	Application and User Data			

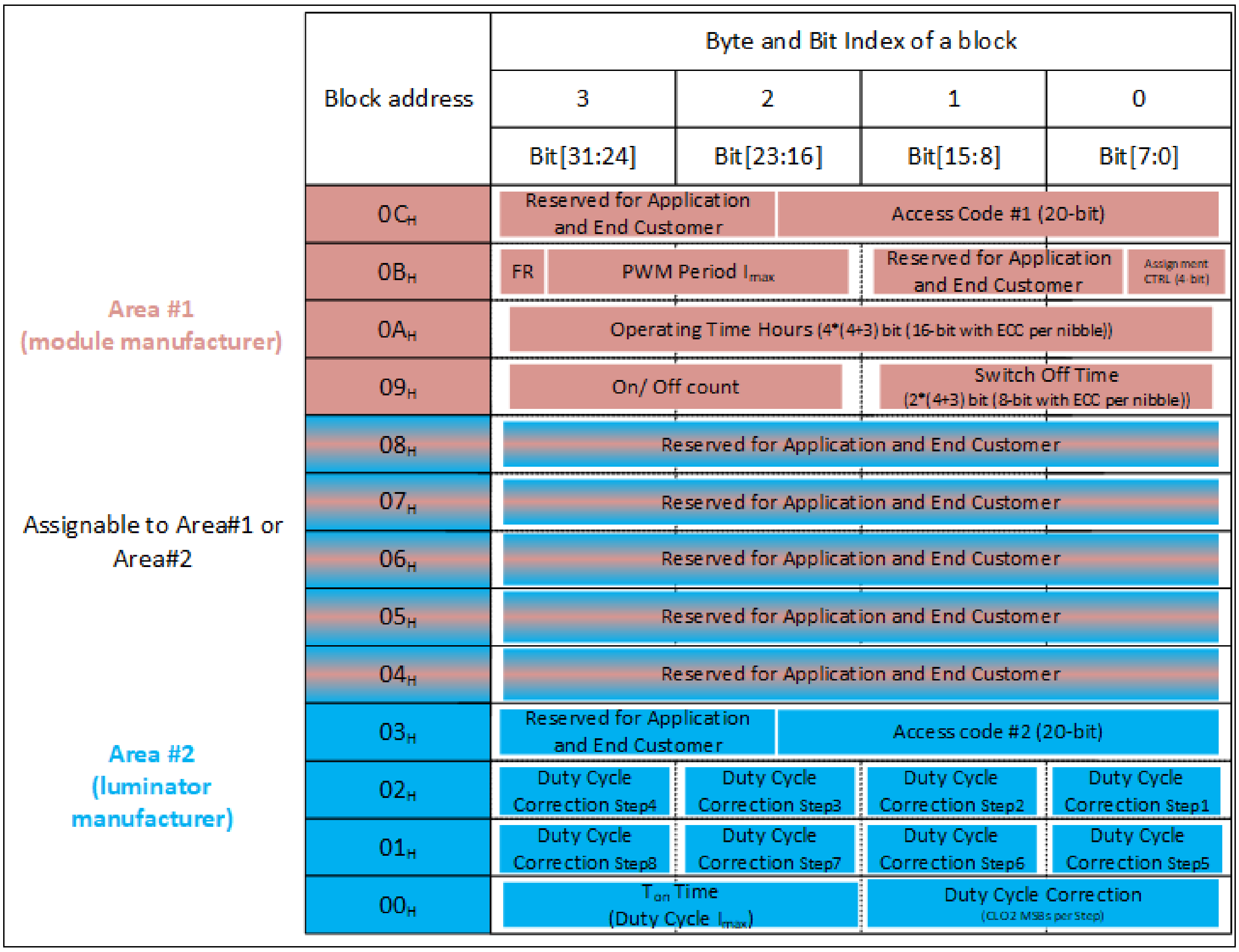
Figure 8 Memory structure

The application and user area is separated into two areas with individual access codes. From the application point of view the user area of the NVM is separated into two exclusive logical areas. One area is usually reserved for the lighting power module manufacturer, and the other area is usually used by the luminaire manufacturer. The area 1 (module area) allocates the blocks at addresses 09_H to $0C_H$. The area 2 (luminaire area) allocates the blocks at addresses 00_H to 03_H. The blocks at addresses 04_H to 08_H, which are reserved for application and end customer specific data, can be assigned to either area 1 or area 2.

The assignment of the blocks at addresses 04_H to 08_H is controlled by the content of the block at address $0B_H$ bit 3 down to 0.

Table 2 Partition of configurable memory

Block $0B_H$ bit (3:0)	Block 08_H	Block 07_H	Block 06_H	Block 05_H	Block 04_H
0100B	Area 1	Area 1	Area 1	Area 1	Area 1
0101B	Area 1	Area 1	Area 1	Area 1	Area 2
0110B	Area 1	Area 1	Area 1	Area 2	Area 2
0111B	Area 1	Area 1	Area 2	Area 2	Area 2
1000B	Area 1	Area 2	Area 2	Area 2	Area 2
1001B	Area 2	Area 2	Area 2	Area 2	Area 2
others	Area 1	Area 1	Area 1	Area 1	Area 1



	Block address	Byte and Bit Index of a block			
		3	2	1	0
		Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]
Area #1 (module manufacturer)	$0C_H$	Reserved for Application and End Customer	Access Code #1 (20-bit)		
	$0B_H$	FR / PWM Period I_{max}		Reserved for Application and End Customer	Assignment CTRL (4 bit)
	$0A_H$	Operating Time Hours (4*(4+3) bit (16-bit with ECC per nibble))			
	09_H	On/ Off count		Switch Off Time (2*(4+3) bit (8-bit with ECC per nibble))	
Assignable to Area#1 or Area#2	08_H	Reserved for Application and End Customer			
	07_H	Reserved for Application and End Customer			
	06_H	Reserved for Application and End Customer			
	05_H	Reserved for Application and End Customer			
	04_H	Reserved for Application and End Customer			
Area #2 (luminator manufacturer)	03_H	Reserved for Application and End Customer	Access code #2 (20-bit)		
	02_H	Duty Cycle Correction Step4	Duty Cycle Correction Step3	Duty Cycle Correction Step2	Duty Cycle Correction Step1
	01_H	Duty Cycle Correction Step8	Duty Cycle Correction Step7	Duty Cycle Correction Step6	Duty Cycle Correction Step5
	00_H	T_{on} Time (Duty Cycle I_{max})		Duty Cycle Correction (CLO2 MSBs per Step)	

Figure 9 **Application and user area**

5.2 Memory access control

Area 1 and area 2 have different access codes. It is possible to read the memory of both areas when entering either access code. But writing the memory in area 1 is only possible when entering the correct access code for area 1. Writing in area 2 is possible when entering the correct access code for area 2 or for area 1.

Each of them is “write protected” by an individual access code (AC1 for area 1 and area 2 and AC2 for area 2).

The access codes can only be updated (rewritten) after a successful code validation (to update AC1, the correct AC1 is provided). But they shall not be read out via NFC interface in any case.

Memory access requires an access code validation, except for answering the “Inventory” command. Without a valid access code, an “Error Code #xx” is answered for any received NFC command.

After a successful access code validation, the user has the following access rights:

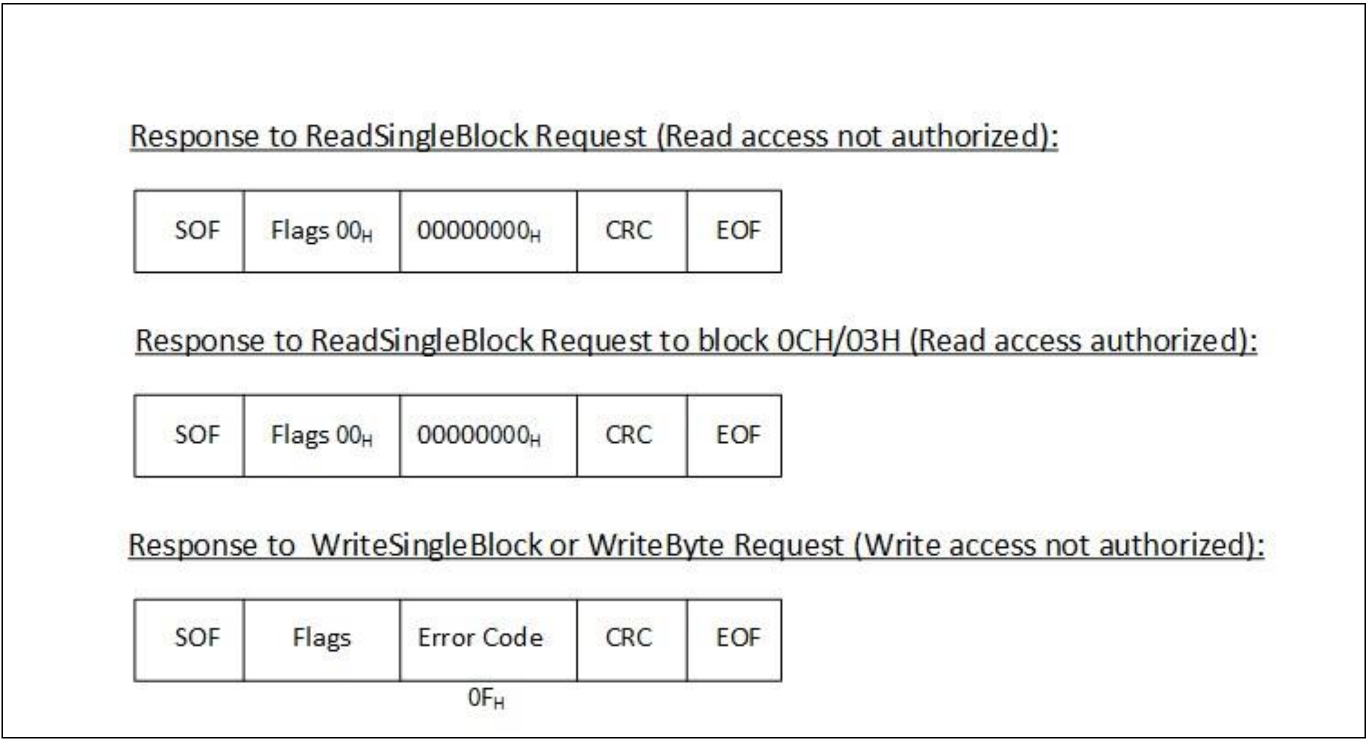
- Read rights in both areas 1 and 2 (no matter which access code is used)
- Writing is possible only with a valid access code (AC1 is required to write in area 1 and area 2. AC2 is required to write in area 2)

Attention: ***In case that access code 1 is equal to access code 2 and different to value 00000$_H$, please check the actual Errata Sheet for NLM0010/11.***

5.3 Access codes and code setting procedure

In NFC configuration mode, it is necessary to control access to the two areas and prevent unauthorized access. Therefore two access codes are defined. The access codes are located in the blocks 0C$_H$ and 03$_H$. The authorization concept can be described as follows (the VICC is the NLM0011/NLM0010 chip):

- VCD must identify and authorize itself by providing an access code to the VICC
- Two access codes are stored in the NVM:
  - Area 1 access code at block 0C$_H$
  - Area 2 access code at block 03$_H$
- VCD provides an access code request (transmits an access code) to VICC
- VICC compares the received access code with the stored access code
  - In case of match: memory access is authorized from now (as long as VICC does not change into power-off state)
  - In case of no match: memory access is not authorized
- Authorized memory access means:
  - Read access:
    - NFC requests ReadSingleBlock are served and responded by the VICC by providing the read data of the requested NVM block, except block 0C$_H$/03$_H$
      - Read rights in both area 1 and area 2 (no matter which of the two access codes are used) are given
      - ReadSingleBlock request to block 0C$_H$ or 03$_H$: the VICC will not send back the content of the block to VCD. The value 00000000H will always be returned instead
  - Write access:
    - NFC requests WriteSingleBlock/Write Byte are served and responded by the VICC; if authorization with AC1 was done complete user memory area can be written
    - If authentication was done with AC2 and no authentication with AC1 was done, write access is only possible to area 2 (luminaire area)
- Non-authorized memory access (and read access to block at address 0C$_H$/03$_H$ when authorized) means:
  - If a ReadSingleBlock request by VCD is not authorized: the VICC responds to the ReadSingleBlock request by non-error read response and data value 00000000$_H$
  - If a WriteSingleBlock request by VCD is not authorized: the VCC responds with error response and error code 0F$_H$



Response to ReadSingleBlock Request (Read access not authorized):

SOF	Flags 00_H	00000000_H	CRC	EOF

Response to ReadSingleBlock Request to block 0CH/03H (Read access authorized):

SOF	Flags 00_H	00000000_H	CRC	EOF

Response to WriteSingleBlock or WriteByte Request (Write access not authorized):

SOF	Flags	Error Code	CRC	EOF
		$0F_H$		

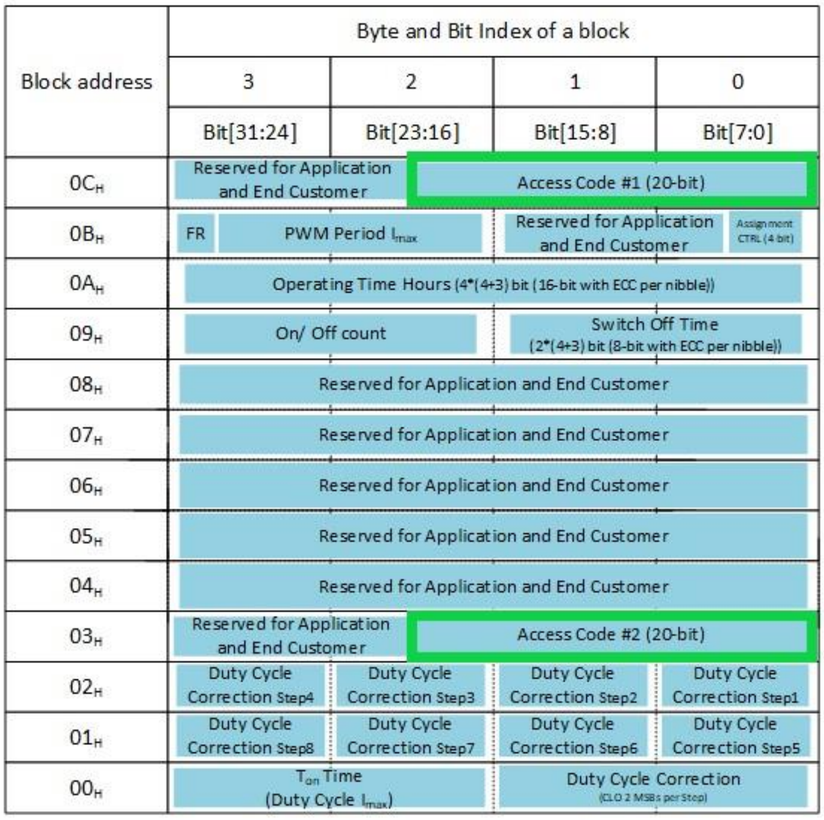
Figure 10 Access controlled responses

The code setting procedure looks as follows:

Access Infineon

Infineon configures the following parameters:

- AC1
- AC2



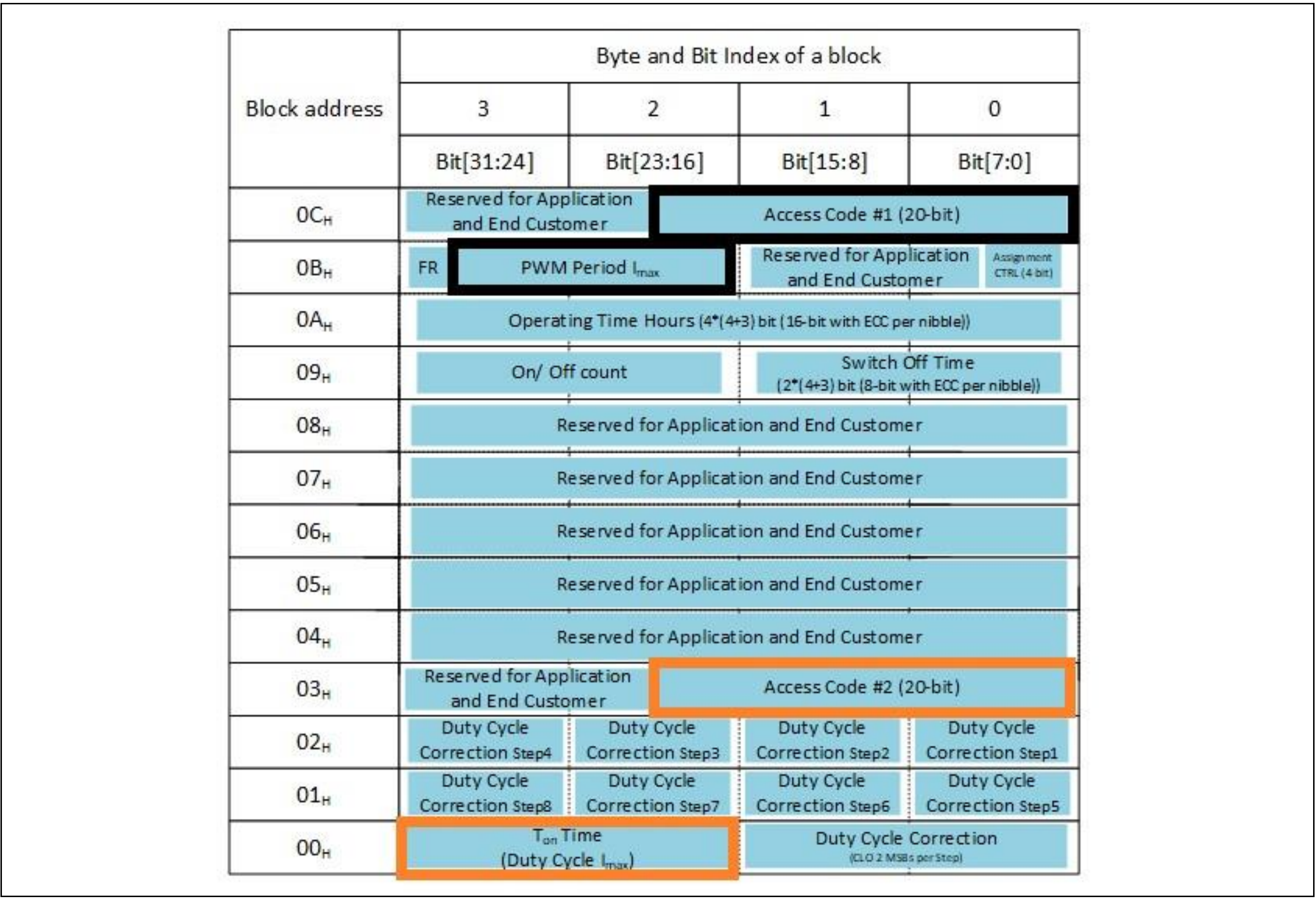
Block address	Byte and Bit Index of a block			
	3	2	1	0
	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]
$0C_H$	Reserved for Application and End Customer	Access Code #1 (20-bit)		
$0B_H$	FR	PWM Period I_{max}	Reserved for Application and End Customer	Assignment CTRL (4 bit)
$0A_H$	Operating Time Hours (4*(4+3) bit (16-bit with ECC per nibble))			
09_H	On/ Off count		Switch Off Time (2*(4+3) bit (8-bit with ECC per nibble))	
08_H	Reserved for Application and End Customer			
07_H	Reserved for Application and End Customer			
06_H	Reserved for Application and End Customer			
05_H	Reserved for Application and End Customer			
04_H	Reserved for Application and End Customer			
03_H	Reserved for Application and End Customer	Access Code #2 (20-bit)		
02_H	Duty Cycle Correction Step4	Duty Cycle Correction Step3	Duty Cycle Correction Step2	Duty Cycle Correction Step1
01_H	Duty Cycle Correction Step8	Duty Cycle Correction Step7	Duty Cycle Correction Step6	Duty Cycle Correction Step5
00_H	T_{on} Time (Duty Cycle I_{max})		Duty Cycle Correction (CLO 2 MSBs per Step)	

Figure 11 Infineon access

Access LED driver maker

The LED driver maker configures following parameters:

- AC1 (update)
- AC2 (update)
- PWM period
- T_{on} period (duty cycle I_{max})



Block address	Byte and Bit Index of a block			
	3	2	1	0
	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]
$0C_H$	Reserved for Application and End Customer	Access Code #1 (20-bit)		
$0B_H$	FR / PWM Period I_{max}		Reserved for Application and End Customer	Assignment CTRL (4 bit)
$0A_H$	Operating Time Hours (4*(4+3) bit (16-bit with ECC per nibble))			
09_H	On/ Off count		Switch Off Time (2*(4+3) bit (8-bit with ECC per nibble))	
08_H	Reserved for Application and End Customer			
07_H	Reserved for Application and End Customer			
06_H	Reserved for Application and End Customer			
05_H	Reserved for Application and End Customer			
04_H	Reserved for Application and End Customer			
03_H	Reserved for Application and End Customer	Access Code #2 (20-bit)		
02_H	Duty Cycle Correction Step4	Duty Cycle Correction Step3	Duty Cycle Correction Step2	Duty Cycle Correction Step1
01_H	Duty Cycle Correction Step8	Duty Cycle Correction Step7	Duty Cycle Correction Step6	Duty Cycle Correction Step5
00_H	T_{on} Time (Duty Cycle I_{max})		Duty Cycle Correction (CLO 2 MSBs per Step)	

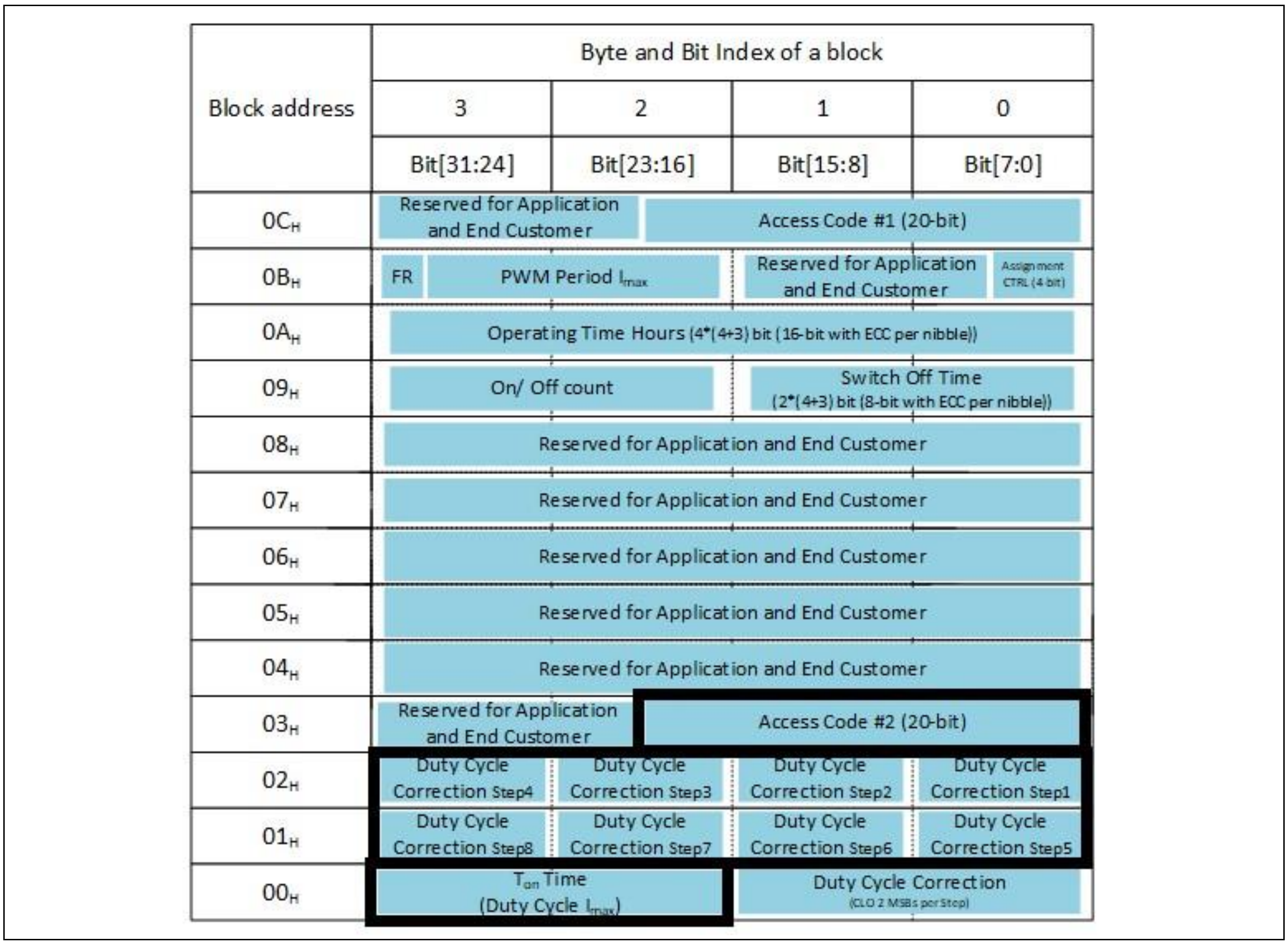
Figure 12 **Access LED driver maker**

Note: *Recommendation for access codes: Implement encryption method in application software to generate individual chip access code by using chip UID.*

Access lamp maker

The lamp maker configures the following parameters:

- AC2 (update)
- T_{on} period (update)
- CLO table



Block address	Byte and Bit Index of a block			
	3	2	1	0
	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]
$0C_H$	Reserved for Application and End Customer	Access Code #1 (20-bit)		
$0B_H$	FR PWM Period I_{max}		Reserved for Application and End Customer	Assignment CTRL (4 bit)
$0A_H$	Operating Time Hours (4*(4+3) bit (16-bit with ECC per nibble))			
09_H	On/ Off count		Switch Off Time (2*(4+3) bit (8-bit with ECC per nibble))	
08_H	Reserved for Application and End Customer			
07_H	Reserved for Application and End Customer			
06_H	Reserved for Application and End Customer			
05_H	Reserved for Application and End Customer			
04_H	Reserved for Application and End Customer			
03_H	Reserved for Application and End Customer	Access Code #2 (20-bit)		
02_H	Duty Cycle Correction Step4	Duty Cycle Correction Step3	Duty Cycle Correction Step2	Duty Cycle Correction Step1
01_H	Duty Cycle Correction Step8	Duty Cycle Correction Step7	Duty Cycle Correction Step6	Duty Cycle Correction Step5
00_H	T_{on} Time (Duty Cycle I_{max})		Duty Cycle Correction (CLO 2 MSBs per Step)	

Figure 13 **Access lamp maker**

Note: *Recommendation for access codes: Implement encryption method in application software to generate individual chip access code by using chip UID.*

6 Development process for NFC reader application software

6.1 Setup the test platform for software development

- Infineon NFC development kit
- NFC reader connected to PC

6.2 Setup the software environment

- Verify the installed firmware version of NFC reader
- Install software development environment (e.g. Eclipse)
- Install NFC reader SDK

6.3 Start software development

- Use Infineon demo software to test the NFC communication (if Feig NFC reader is used)
- Start to develop own application software

6.4 Feig reader application software

To program NLM0011/NLM0010, a NFC reader device is required. It could be a mobile phone with built-in NFC interface or a professional NFC reader device. For industrial applications, like LED lighting, typically a professional NFC reader device is used for reasons of ease of use, long sensing distance and highly reliable operation. The professional NFC reader device is connected to a PC and is controlled by PC software (NFC reader application software). The NFC reader application software is developed either by the vendor of the LED power supply, or the luminaire maker. Typically, vendors of the LED power supply provide their application software to OEM customers.

NFC readers from Feig GmbH are certified by the Module-Driver Interface Special Interest Group (MD-SIG).

6.4.1 Command types and their execution method

NLM0011/NLM0010 support following nine NFC commands: Inventory, Stay Quiet, Read Single Block, Write Single Block, Select, Reset to Ready, Write AFI, Lock AFI, and Write Byte.

Using Feig SDK, a NFC command can be executed by using either the SendProtocol() function or the SendTransparent() function. The three commands below must be executed with the SendTransparent() function:

Read Single Block, Write Single Block, Write Byte.

Execution of other commands – Stay Quiet, Select, Reset to Ready, Write AFI, Lock AFI – are recommend to use the Send Protocol() function.

6.4.2 Code examples

Below two examples using transparent function and protocol function are developed for the Feig NFC reader. For complete source codes, please check the source code file provided. If you are using NFC readers from other vendors, please refer to the corresponding SDK.

6.4.2.1 Inventory command

In this example, the sendProtocol() function is used to execute “inventory” command. The benefit of using the sendProtocol() function is that a short command string can be used.

```
public int inventory(FedmIscTagHandler_Result result) throws
FePortDriverException, FeReaderDriverException, FedmException,
AgedPerfectException
{
        FeIscProtocol localFeIscProtocol = new FeIscProtocol();
        FedmIscReader     iscReader     = reader.getReaderImpl();
        FedmIscTagHandler iscTagHandler = tagHandler.getTagHandlerImpl();
        String cmd = new String();
        String uid = new String();
        uid        = iscTagHandler.getUid();

        result.data    = null;

      //build up NFC command string
        cmd = cmd + "01";                // Inventory command
        cmd = cmd + "00";            // Mode: 00-nonaddressed only

//NFC command execution
        String str = iscReader.sendProtocol((byte)0xB0, cmd);

       //Exception handling
if (str.length() == 0) {
            throw new
FedmException(iscReader.getErrorText(Fedm.ERROR_NO_DATA),
Fedm.ERROR_NO_DATA);
        }

if(reader.getLastStatus() == (byte)0x95) {
            // we have a VICC error
            throw new
AgedPerfectException(AgedPerfectErrorCode.getErrorText(FeHexConvert.hexString
ToInteger(str)),
                        FeHexConvert.hexStringToInteger(str));
        } else if(iscReader.getLastStatus() == 0) {
            // we have a feedback from the Reader
            result.data        = new byte[8];
            result.data[0]     = (byte)localFeIscProtocol.recData[1];
            result.data[1]     = (byte)localFeIscProtocol.recData[2];
            result.data[2]     = (byte)localFeIscProtocol.recData[3];
```

```
                result.data[3]    = (byte)localFeIscProtocol.recData[4];
                result.data[4]    = (byte)localFeIscProtocol.recData[5];
                result.data[5]    = (byte)localFeIscProtocol.recData[6];
                result.data[6]    = (byte)localFeIscProtocol.recData[7];
                result.data[7]    = (byte)localFeIscProtocol.recData[8];
                String TrTypeStr = str.substring(2,4);
                String DsfidStr  = str.substring(4,6);
                String uidStr    = str.substring(6);
                jProtocol.append("TR-TYPE: " + TrTypeStr + "; DSFID: " + DsfidStr
+ "; UID: " + uidStr);
          }
          return reader.getLastStatus();
     }
```

6.4.2.2 Read single block command

In this example, the sendTransparent() function is used to execute the "read single block" command. The benefit of using the sendTransparent() function is that each bit in the command string can be defined. With good software architecture design, it can enable NFC vendor-independent or less-dependent application software development – the command string is identical and independent to the vendor of the NFC reader used.

```
public int readSingleBlock (byte address, byte db_n, FedmIscTagHandler_Result
result)
                throws FePortDriverException, FeReaderDriverException,
FedmException, AgedPerfectException {
          FeIscProtocol localFeIscProtocol = new FeIscProtocol();
          FedmIscReader iscReader          = reader.getReaderImpl();
          FedmIscTagHandler iscTagHandler  = tagHandler.getTagHandlerImpl();
          String cmd        = new String();
          String uid;
          String uidRev = new String();
          uid = iscTagHandler.getUid();
          for (int i=0; i < 15; i=i+2) {
               uidRev = uidRev + uid.substring(14-i, 16-i);
          }
          result.data    = null;


        //build up NFC command string
          cmd = cmd + "02";             // STX
          if (iscTagHandler.nonAddressedMode) {
               cmd = cmd + "000F";      // ALENGTH w/o UID plus CRC: 15
          }
          else {
              cmd = cmd + "0017";     // ALENGTH w UID and CRC: 23
          }
          cmd = cmd + "FF";           // COM-ADR
          cmd = cmd + "BF";           // CMD (Reader)
          cmd = cmd + "01";           // Mode 1: read request
          cmd = cmd + "0038";         // RSP-length in bits w/o SOF and EOF
          cmd = cmd + "0000";         // RSP-delay in multiples of 590ns
0x0000=0x021F=320,4us
          if (iscTagHandler.nonAddressedMode) {
              cmd = cmd + "12";       // flags: selected, high data-rate,
single sub-carrier
          } else {
              cmd = cmd + "22";       // flags: addressed, high data-rate,
single sub-carrier
          }
          cmd = cmd + "20";           // ReadSingleBlock command
          if (!iscTagHandler.nonAddressedMode) {
              cmd = cmd + uidRev;     // UID reverse order
          }
          cmd = cmd + FeHexConvert.byteToHexString(address);     // Address

         //NFC command execution
String str = iscReader.sendTransparent(cmd, true);

//Exception handling
          localFeIscProtocol.recProtocol =
FeHexConvert.hexStringToByteArray(str);
          int i = iscReader.splitRecProtocol(localFeIscProtocol);
          if (i == (int)0x95) {
              int ISOErrorCode = localFeIscProtocol.recData[0];
              throw new
AgedPerfectException(AgedPerfectErrorCode.getErrorText(ISOErrorCode),
ISOErrorCode);
          }
          else if (i == 0) {
              result.data       = new byte[4];
              result.data[0]    = (byte)localFeIscProtocol.recData[4];
              result.data[1]    = (byte)localFeIscProtocol.recData[3];
              result.data[2]    = (byte)localFeIscProtocol.recData[2];
              result.data[3]    = (byte)localFeIscProtocol.recData[1];
              // for debug only:
```

```
            String DataStr    = new String();
            DataStr = DataStr +
FeHexConvert.byteToHexString((byte)localFeIscProtocol.recData[4]);
            DataStr = DataStr +
FeHexConvert.byteToHexString((byte)localFeIscProtocol.recData[3]);
            DataStr = DataStr +
FeHexConvert.byteToHexString((byte)localFeIscProtocol.recData[2]);
            DataStr = DataStr +
FeHexConvert.byteToHexString((byte)localFeIscProtocol.recData[1]);
            // for debug:
            //jProtocol.append("Read-Data: " + DataStr + "\n");
        }
        else {
            throw new
FedmException(iscReader.getErrorText(Fedm.ERROR_NO_DATA),
Fedm.ERROR_NO_DATA);
        }
        return reader.getLastStatus();
    }
```

6.4.2.3 Write single block command

In the „write single block“ example, the sendTranparent() function is used again. Compared to the previous “read single block” example, it can be seen that additional 4 data bytes have to be transferred. Care must be taken, that all multi-byte data fields, such as UID bytes or the 4 data bytes must be transferred in reverse order, i.e., least significant byte first and most significant byte last

```
// [0x21] WriteSingleBlock
    public int writeSingleBlock(byte address, byte db_n, byte db_size, byte[]
db)
            throws FePortDriverException, FeReaderDriverException,
FedmException, NLM0011Exception {
        FeIscProtocol localFeIscProtocol = new FeIscProtocol();
       FedmIscReader iscReader = reader.getReaderImpl();
        FedmIscTagHandler iscTagHandler = tagHandler.getTagHandlerImpl();
        String cmd    = new String();
        String uid;
        String uidRev = new String();
        uid = iscTagHandler.getUid();
        for (int i=0; i < 15; i=i+2) {
            uidRev = uidRev + uid.substring(14-i, 16-i);
        }
        cmd = cmd + "02";            // STX
        if (iscTagHandler.nonAddressedMode) {
            cmd = cmd + "0013";      // ALENGTH w/o UID plus CRC:
19
        } else {
            cmd = cmd + "001B";      // ALENGTH w UID and CRC: 27
        }
        cmd = cmd + "FF";            // COM-ADR
        cmd = cmd + "BF";            // CMD (Reader)
        cmd = cmd + "02";            // Mode 2: write request
        cmd = cmd + "0018";          // RSP-length in bits w/o SOF and EOF:
32bit in error case
        cmd = cmd + "0000";          // RSP-delay in multiples of 590ns
0x0000=0x021F=320,4us
        if (iscTagHandler.nonAddressedMode) {
            cmd = cmd + "12";          // flags: selected, high data-rate,
single sub-carrier
        } else {
            cmd = cmd + "22";          // flags: addressed, high data-rate,
single sub-carrier
        }
        cmd = cmd + "21";            // WriteSingleBlock command
        if (!iscTagHandler.nonAddressedMode) {
            cmd = cmd + uidRev;      // UID reverse order
        }
        cmd = cmd + FeHexConvert.byteToHexString(address);     // Address
        for (int i = (db_size - 1); i >= 0; i--) {
            cmd = cmd + FeHexConvert.byteToHexString(db[i]);   // DataBlock
reverse - 4Bytes
        }
         String str = iscReader.sendTransparent(cmd, true);
        localFeIscProtocol.recProtocol =
FeHexConvert.hexStringToByteArray(str);
        int i = iscReader.splitRecProtocol(localFeIscProtocol);
        if (i == (int)0x95) {
            int ISOErrorCode = localFeIscProtocol.recData[0];
            throw new
NLM0011Exception(NLM0011ErrorCode.getErrorText(ISOErrorCode), ISOErrorCode);
        }
        else if (i == 0) {
            jProtocol.append("OK");


        }
        else {
            throw new 
FedmException(iscReader.getErrorText(Fedm.ERROR_NO_DATA), 
Fedm.ERROR_NO_DATA);
        }
        return reader.getLastStatus();
    }
```

7 Additional implementation topics

7.1 Notes about stored values

7.1.1 Operating time counting (OTC)

The OTC is done in lighting operational mode only. In configuration mode no OTC is performed. The OTC starts working from an on-time of 31 s. All values under the duration of 31 s will not be added to the operating time value.

Maximum countable operating time = 65536 (2 bytes) x 4 h = 262.144 h. The time counter will be frozen after this limit is reached.

7.1.2 CLO table interpolation

Just as with OTC, interpolation between the set-points in the CLO table starts working from an on-time of 31 s. All values under the duration of 31 s will not cause a new calculation of interpolation, because operating time will not increase for a lower on-time than 31 s.

The values according to the interpolation between each point of the CLO table will be recalculated every 4 hours.

7.2 Output current stability and tolerance level consideration

When designing a LED power supply, one important design parameter is the output current stability (output current variation of a single product) and the tolerance level (output current variation from product to product). Normally a +/-10 percent tolerance level is required of the LED power supplier. For a high-quality supplier, a +/-5 percent tolerance level is expected. In some special applications, a lower tolerance level to +/-2 percent could be accepted. The output current tolerance level is a system performance parameter, contributed to by all involved active and passive components in a power supplier. Nevertheless, the stability and accuracy of the analog DC signal, which directly controls the driver IC output, plays an important role in tolerance level achievement.

The analog DC control voltage V_{ave} is calculated as follows:

$$V_{ave} = DC \cdot V_h + (1 - DC) \cdot V_l$$

DC is the PWM duty cycle. Its variation is less than 0.1 percent. It can be ignored in the tolerance calculation. Therefore, the main contributor to the stability and tolerance of the DC control signal is the quality of the V_h and V_l parameters of the PWM signal. Comparing the V_h and V_l, the variation of V_l is at least 10 magnitudes lower than the variation of V_h.

Regarding the PWM stability, with built-in Low Drop-Out (LDO), the PWM output from NLM0011/NLM0010 is very stable and is not influenced by the stability of the power supply. This is a big advantage in design compared to competitors' products whose PWM amplitude is directly coupled to the power supply voltage.

A big advantage of using NFC IC is the ease of calibration of the final driver output current. This would enable a decrease of the overall tolerance down to 1 percent for a lower end driver by simply modifying the reference PWM duty cycle. Later customer current adjustments do not require further calibration.

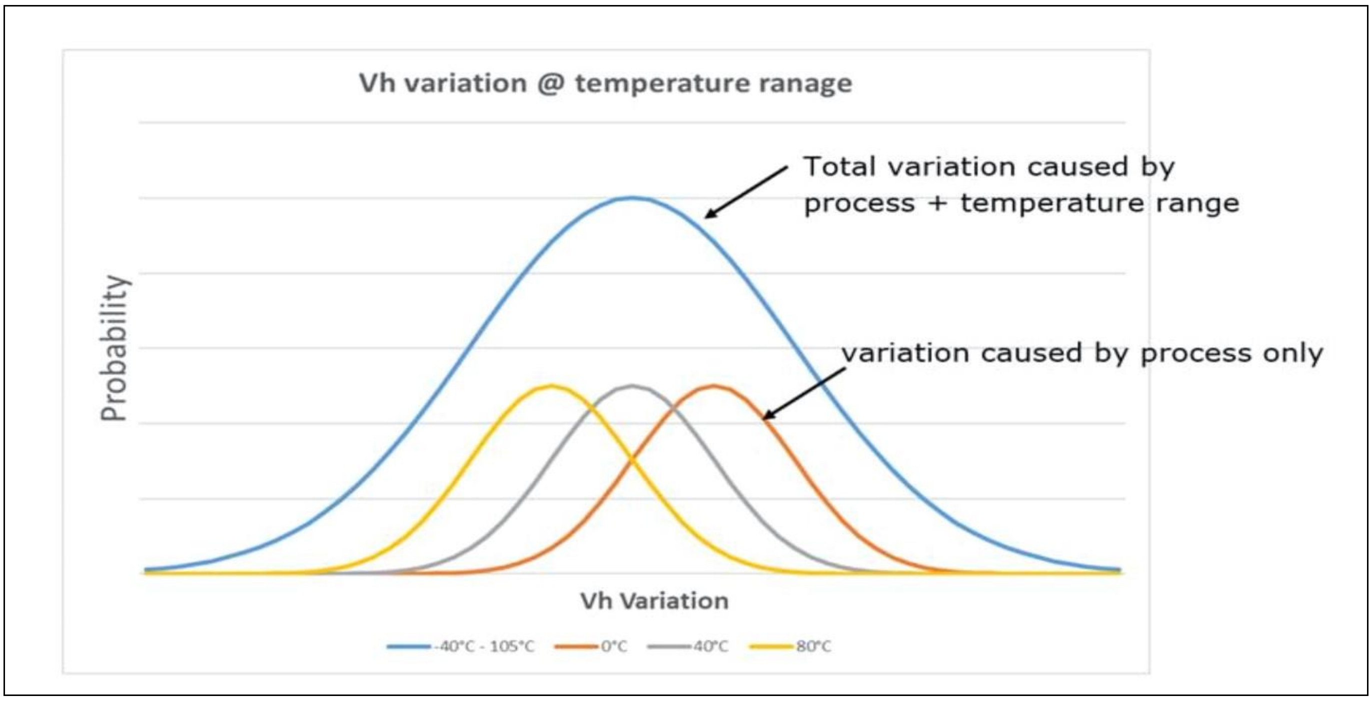


Figure 14 **V_h variation**

7.3 In-production calibration

The current output can be calibrated to the exact nominal value in production testing.

The current output of a LED driver module can be easily calibrated by reconfiguring the duty cycle. Thus all tested modules have the exact nominal current output.

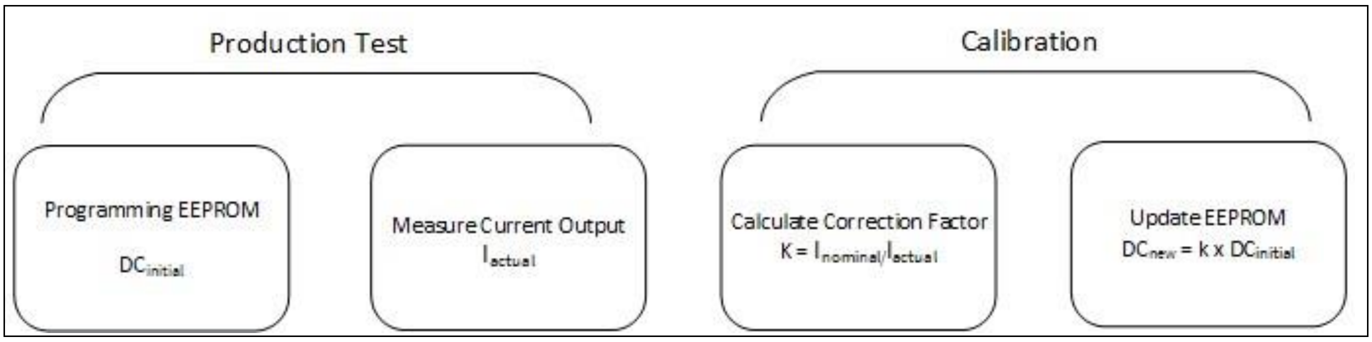

Production Test
Calibration
Programming EEPROM $DC_{initial}$
Measure Current Output I_{actual}
Calculate Correction Factor $K = I_{nominal}/I_{actual}$
Update EEPROM $DC_{new} = k \times DC_{initial}$

Figure 15 **In-production calibration**

<u>How to calibrate a LED module in production:</u>

The design values below are for a 100 mA LED driver module:

f_{PWM} = 4 kHz → PWM period = 6780

Duty cycle = 50 percent → T_{on} period = 6780 x 50 percent

- NFC IC is programmed with design values
- LED module output current is measured in production testing, which is 105 mA; 5 mA more than the specification

- LED driver module maker can calibrate the device to exactly 100 mA output by reprogramming the value of the PWM period I_{max}, but keeping the T_{on} period unchanged. In this case, new $PWM\ period = 6780 \cdot \frac{105}{100} = 7119$
- Write 7119 into EEPROM
- The effective duty cycle is, therefore, changed to 3390/7119 = 47.6 percent. The output current is therefore calibrated to 100 mA.

Table 3 In-production calibration and calculation of T_{on}

I_{LED}	PWM period	T_{on} period
100 mA	6780 → 7119	3390
50 mA		1690
10 mA		339

7.4 Linear extrapolation and LED exchange in field

In case of LED exchange in field, CLO can be programmed with a time shift according to actual operation time.

Output level as a percentage is linearly extrapolated: $DC_x = DC_A + (DC_B - DC_A) \cdot \frac{T_x - T_A}{T_B - T_A}$

$$Effective\ DC = DC_X \cdot \frac{M}{N} \cdot 100\%$$

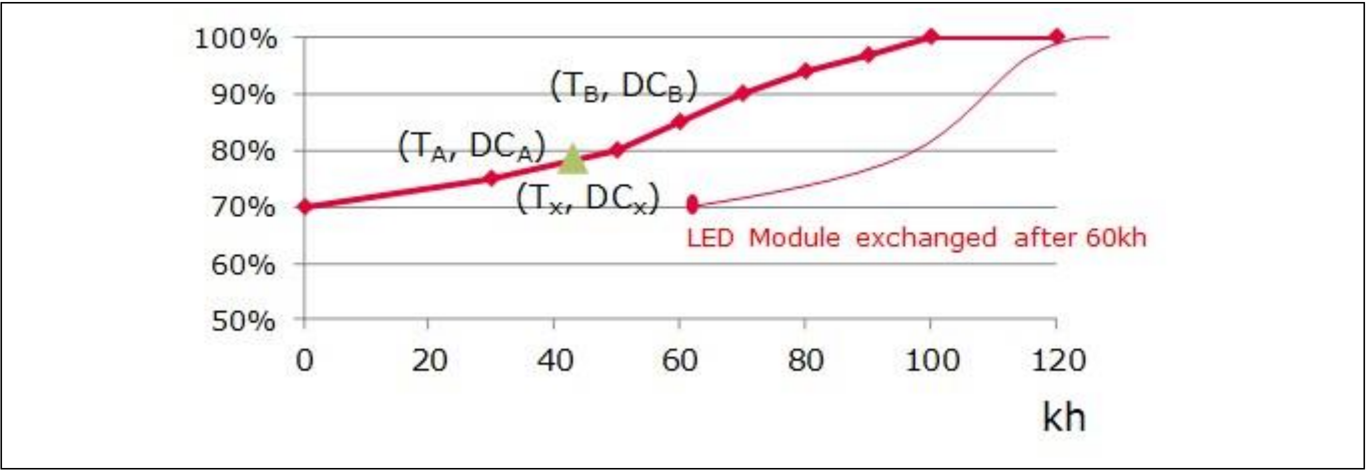


Figure 16 CLO function with LED exchange in field

7.5 Antenna design

The basic principle of the NFC antenna design is to create a circuit resonating at 13.56 MHz with an inductance (antenna) and an internal or external tuning capacitor. This is because NFC readers operate in the 13.56 MHz high frequency band.

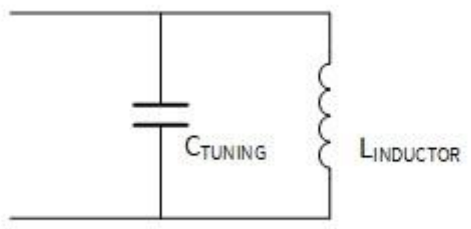


Figure 17 Basic principle

The equation is:

$$f_{TUNING} = \frac{1}{2\pi\sqrt{L_{INDUCTOR} \cdot C_{TUNING}}}$$

therefore:

$$C_{TUNING} = \frac{1}{4\pi^2 f_{TUNING}^2 \cdot L_{INDUCTOR}}$$

If the theoretical value of the inductance is 4.5 µH, a 30 pF tuning capacitor should be used.

If a PCB antenna should be designed, the formula for calculating (or approximating) the inductor value of the antenna is:

$$L(\mu H) = \frac{11.04 \cdot [(A_{mm} + PCB_{th-mm}) \cdot N]^2}{100 \cdot [38.16 \cdot (A_{mm} + PCB_{th-mm}) + 90 \cdot (A_{mm} - a_{mm}) + 100 \cdot PCB_{th-mm}]}$$

where:

- L (µH) is the PCB antenna inductance (in µH)
- A is the outer square dimension (in mm)
- a is the inner square dimension (in mm)
- PCB_{th} is the PCB thickness containing the antenna traces (in mm)

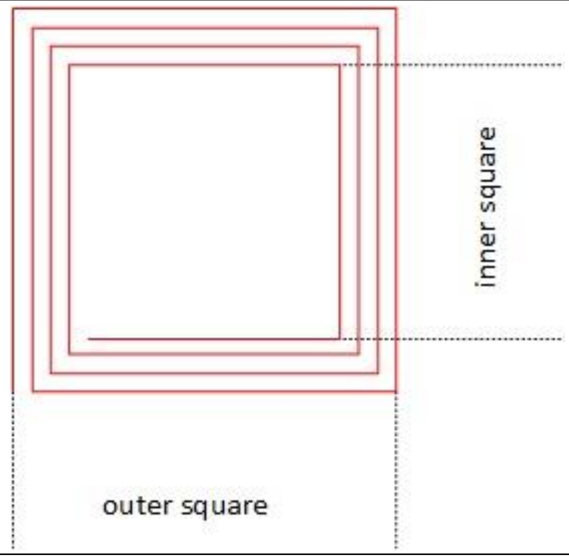


Figure 18 PCB antenna

Additional design tips:

- An antenna that is too small would "waste" magnetic flux due to the small area within the coil. An antenna coil that is too big does not get flux around the outside of the windings. So the shape or size of antenna coil should match the geometry of the reader's antenna (in this case, it is the NFC antenna in the cell phone/Feig reader).
- Tuning the tag only to have 13.56 MHz is not the correct way. Once the tag and the reader are coupled the resonance frequency typically lowers. This is because of the coupling. As a result, the tag antenna resonance circuit must be tuned to a frequency slightly higher than 13.56 MHz. We usually tune our tag resonance frequency to between 13.6 MHz and 13.8 MHz.
- Also important is the type of capacitor used for tuning. There are lots of types, which simply do not behave as capacitors any more above a certain frequency. The wrong capacitor will introduce losses into the LC tank, lowering the quality factor. Therefore we strongly recommend using a COG: Class I (Also known as "NPO").
- Very thin wiring will add ohmic losses. So if there is enough space the width can be 500 µm or more. The gap between can be smaller, which will already introduce a parallel capacitance between the traces. The number of coil turns is not as important; we have seen almost no difference in power delivery between three and four windings if the voltage is shunted to 3.3 V by the chip. This is much more critical if the chip allows higher voltages.
- For full calculation example please check the document antenna design guide on the website.

Attention: ***For a good antenna coupling, it is strongly recommended that the receiving and transmitting antennas are to be arranged parallel to one another and centrally above one another.***

7.6 EMC radiation into the antenna

In comparison to other NFC controllers, the NLM001x has no risk of EMC radiation during operation mode. The antenna is not active during active mode.

The antenna is only switched on in configuration mode.

In this way, the settings in the EEPROM are protected against unwanted changes during active mode and no signal can be emitted via the antenna.

8 User application development suggestion

8.1 Use Infineon mobile app and evaluation kit for fast application development

With Infineon’s NLM0011 evaluation kit a fast evaluation of features, own filter settings and starting to develop the application software for Feig readers without a full system is possible.

8.2 Special note for using the FEIG CPR30-USB

When using the FEIG CPR30-USB please make sure using firmware V1.07 and higher. In this case it is also recommended to set the modulation to 10 percent.

8.3 Feig selection guide and SDK info

For further information please visit www.feig.de.

Revision history

Document version	**Date of release**	**Description of changes**
1.0		Initial version
1.1	2019-07-05	Falling curves added Access code description updated
1.2	2020-06-05	Additional chapters, minor changes

Edition 2020-06-05

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN_1901_PL39_1903_124050