# Application Note

**Tag detector function of MLX90130 / MLX90132**
**13.56MHz RFID / NFC Transceivers Family**

## Scope

The Tag Detector is a special function of the MLX90130/32 ICs, designed to save power when working in reader mode. This function allows detecting the presence of a card near the antenna. When a Card is present, device exits power-saving mode and sends an interrupt request to the application host microcontroller.

This application note explains the functionality of the Tag Detector feature, the configuration of the different parameters and finally describes an example of calibration procedure. The working principle of the tag detector function is firstly explained, followed by the explanation of the command structure and finally, the calibration procedure. In order to simplify the reading and the search of the document, the command parameters are highlighted in blue while, some internal values of registers are highlighted in violet.

## Applications

- RFID / NFC General Purpose

## Related Melexis Products

| Part Code | Temperature Code | Package Code |
|---|---|---|
| MLX90130 | R (-40°C to 105°C) | LQ (Lead free QFN 5x5 32 leads) |
| MLX90130 | S (-20°C to 85°C) | LQ (Lead free QFN 5x5 32 leads) |
| MLX90132 | R (-40°C to 105°C) | LQ (Lead free QFN 5x5 32 leads) |

## Introduction


The MLX90130/32 serie is a 13.56MHz RFID / NFC transceiver ICs family. It has been designed to support communications with RF memory transponders compliant with ISO/IEC14443, ISO/IEC15693 and ISO/IEC18092 (MLX90132) RFID standards.

The digital section of the MLX90130/32 handles the low protocol layers from API to physical layer using advanced bit and frame decoding functions. It contains a digital demodulator based on sub-carrier detection and a programmable bit/symbol encoder/decoder. It also encodes and decodes the start and stop bits, parity bits, extra guard time (EGT), start and end of frame (SOF/EOF) and CRC.

Its 528 bytes buffer allows buffering of an entire RFID frame and its SPI/UART communication interfaces guarantee an easy control from the majority of microcontrollers.

# 1 TAG Detector Mode: Principle of Working

The Tag Detector is based on the detection of amplitude changes in its HF field. In fat, If a metallic object approaches the antenna or moves away from it, it influences the generated field. This effect is even bigger if the object has a resonant circuit tuned around the generated HF frequency.

Then it is possible for the device to detect any change in the amplitude of the HF field belongings to the presence of an object in front of its antenna. In order to reduce the overall power consumption, this detection can be made with a very short periods of HF field while the IC remains in low-power mode (sleep mode) the rest of the time.

Once any variation of the HF field amplitude has been detected, the MLX90130/32 device wakes-up and informs the application microcontroller by a short pulse on its pin IRQ_OUT. Then, the application microcontroller takes-over the control of the MLX90130/32 and initiates the TAG recognition procedure by sending successive commands according to the requirements of the application.

In case the environment is changing leading to a constant modification of the HF field amplitude generated, there is always the possibility to measure again the new reference amplitude by performing a new calibration procedure (procedure explained further of this document).

In order to detect any variation of the HF field (decrease of increase), two thresholds levels are stored into the MLX90130/32 device: DacDataL and DacDataH. Basically, the device wakes-up and generates an IRQ pulse if the measured HF amplitude becomes lower/higher than respectively DacDataL/DacDataH. While the measured HF field amplitude remains within the range defined by DacDataL and DacDataH, the device stays in TAG detection mode (no wake-up).

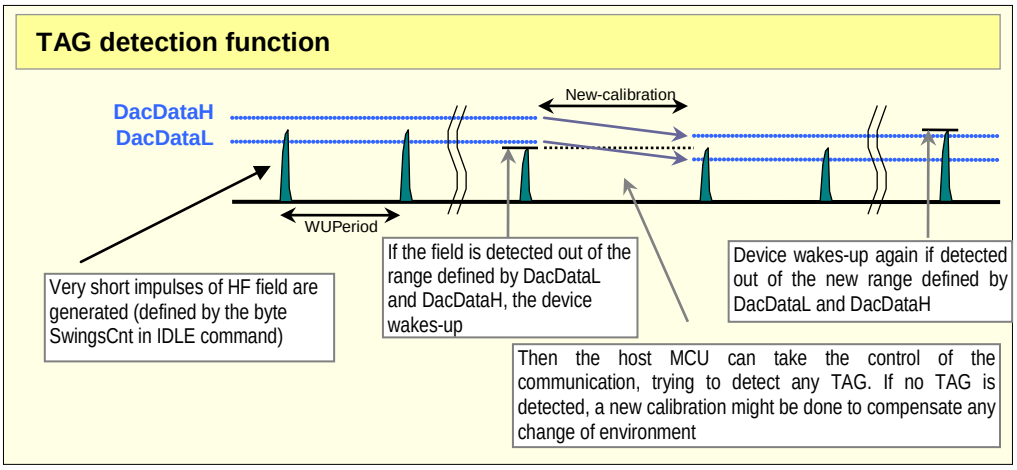


Figure 1: **TAG detection wake-up principle**

# 2 Timings definition

The period of time in which no HF field is generated is called “Sleep_TagDet” period. During this period the High Frequency Oscillator (HFO) as well as the DAC are turned OFF; only the internal Low Frequency Oscillator (LFO) is ON.

**Equation 1:**

$$T_{Sleep\_TAGdet} = 256 \cdot T_{LFO} \cdot (WUPeriod + 2)$$

With:

- $f_{LFO}$ = 32kHz
- WUPeriod = Time with the MLX90130/32 in Idle mode, configuration byte define in “IDLE” command

The bytes OscStart and DacStart are set to default value of 0x60, those bytes defines some delay before the HF field is set ON for the TAG detection. When these delays are over, the HF field is generated during:

**Equation 2:**

$$T_{HFon\_TAGdet} = SwingCnt \cdot T_{carrier}$$

With:

- SwingCnt = Number of HF period activity, configuration byte define in “IDLE” command

A Timeout time is available to leave automatically from the Tag detector mode after “MaxSleep” numbers of executions. There is also the possibility to disable this feature and let the Tag Detector mode running indefinitely, until a real TAG event occurs

**Equation 3:**

$$T_{MaxSleep\_TAGdet} = \left( T_{HFOon\_TAGdet} + T_{Sleep\_TAGdet} \right) \cdot (MaxSleep + 1)$$

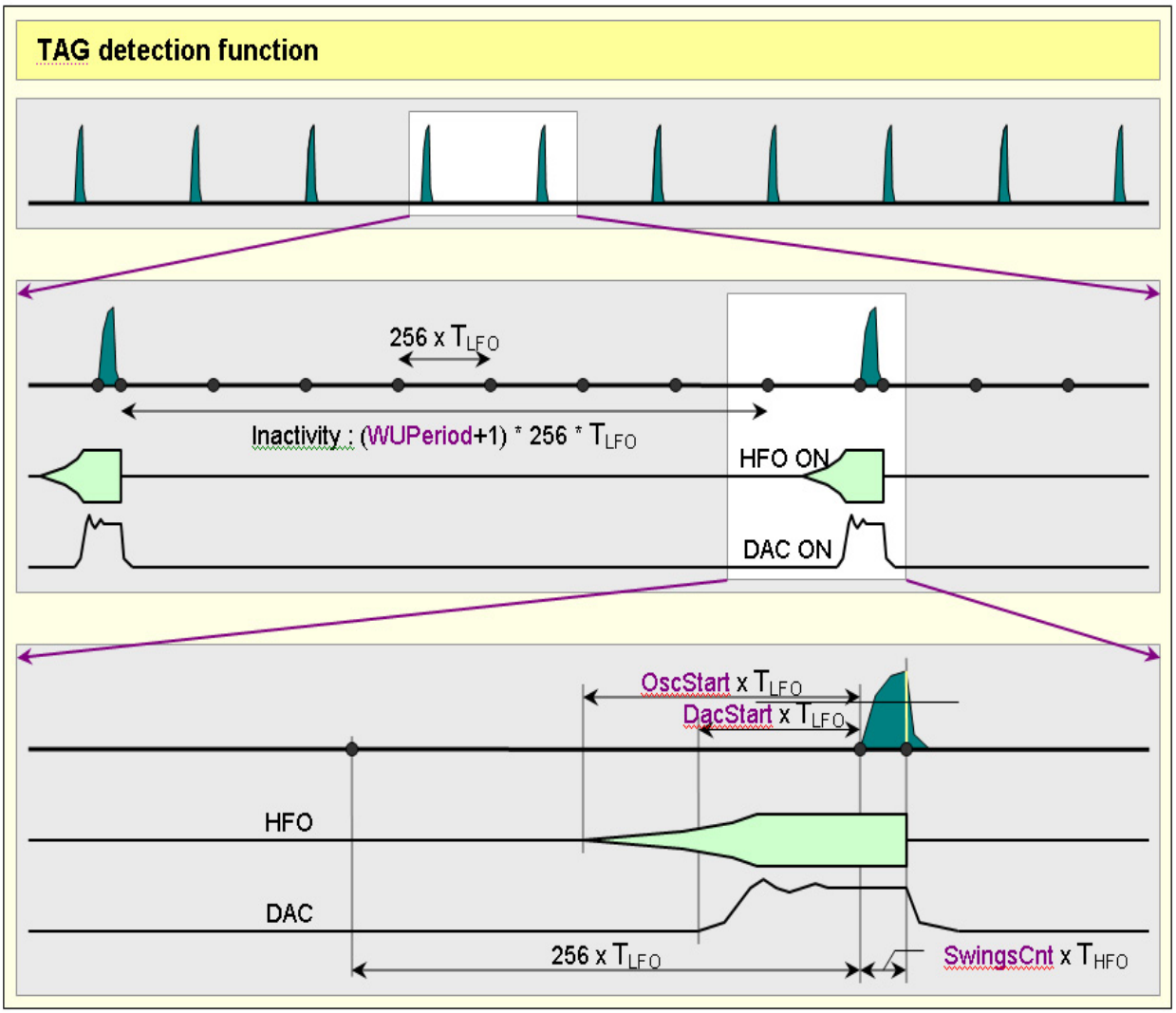


Figure 2: **TAG detection function**

# 3 Interruption during Tag Detector Mode

During TAG detection mode activity, the MLX90130/32 can be interrupted by a low-pulse on the pin $IRQ_{IN}$ (defined in the WUflags register of Idle command). As soon as the IRQ pulse is detected, the MLX90130/32 wakes-up and updates the WUFlag register at address 0x62. The detection of IRQ and wake-up procedure is different according to whether TAG detector is activated or not.

Two wake-up sources can be identified:

- The device wakes up due to a pulse on $IRQ_{IN}$: The host has to wait for $t_{SU(HFO)}$ (about 2ms)
- The device wakes up due to a TAG detection: HFO has been already started and is stable

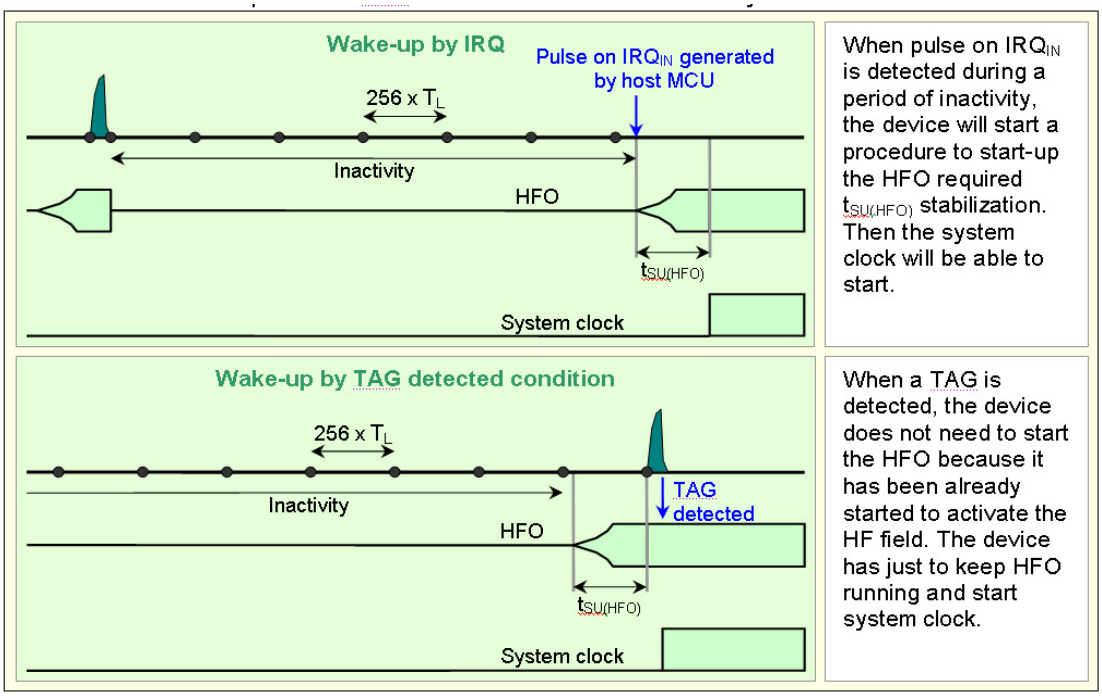


Figure 3: **TAG detection wake-up source vs $t_{SU(HFO)}$ stabilization time**

# 4 Idle command (0x07) definition

The command **IDLE** is used to activate and configure the Tag detector function.

| Idle 0x07 | | | |
|---|---|---|---|
| Direction | Data | Comments | Example |
| MCU – device | 07 | Command code | 0x070E0221003801180022C0C054603F00– Tag detector with LFO set at 32kHz<br><br>0x070ECB21003801180022C0C054603F10 – Tag detector with LFO set at 4kHz + possibility to WU on low level on RX and time out set with MaxSleep = 10 |
| | 0E | Length of data | |
| | <WUFlags> | Specifies wake-up sources and LFO frequency. Refer to Table 2 | |
| | <EnterCtrlL> | 2 bytes: Settings to enter Idle mode, refer to Table 3 | |
| | <EnterCtrlH> | | |
| | <WUCtrlL> | 2 bytes: Settings to wake-up from Idle mode, refer to Table 3 | |
| | <WUCtrlH> | | |
| | <LeaveCtrlL> | 2 bytes: Settings to leave Idle mode (recommended value = 0x1800), refer to Table 3 | |
| | <LeaveCtrlH> | | |
| | <WUPeriod> | Period of time between two TAG detection bursts. Also used to specify the duration before timeout. Refer to Table 2 | |
| | <OscStart> | Waiting time for the HFO to stabilize (based time: LFO)<br>(recommended value = 0xC0) | |
| | <DacStart> | Waiting time for the DAC to stabilize (based time: LFO)<br>(recommended value = 0xC0) | |
| | <DacDataL> | Lower compare value for TAG detection. Note: Only the 6 MSB bits are available | |
| | <DacDataH> | Higher compare value for TAG detection. Note: Only the 6 MSB bits are available | |
| | <SwingsCnt> | Number of HF periods during TAG detection refer to Table 2 | |
| | <MaxSleep4:0> | Maximal number of TAG detection trials before timeout. Value set to 0x01 during TAG detection calibration.<br>0x00 < MaxSleep < 0x1F (bit 7 to 5 are RFU and must be set to 0)<br>Also used to specify duration before timeout, refer to Table 2 | |
| Device - MCU | 0x00 | Error code | 0x0001XX |
| | 0x01 | Length of data | |
| | <WUFlags> | Content of WUFlags, please refer to Table 4 below | |
| Device - MCU | 0x82 | Error code | 0x8200, Invalid command length |
| | 0x00 | Length of data | |

Table 1: **"Idle" command description**

| Values of parameters | | |
|---|---|---|
| A | Register | Comment |
| 0 | WUFlags | Default value = 0x02: only TAG detector is activated, with LFO = 32kHz<br>7:6 = 00, LFO = 32kHz<br>7:6 = 01, LFO = 16kHz<br>7:6 = 10, LFO = 8kHz<br>7:6 = 11, LFO = 4kHz<br>5: RFU<br>The following bits Specifies the possible source on which to exit from idle mode, each bit corresponds to one wake-up source. Those Wake-up source flags are updated and returned when the MLX90132 leaves the Idle routine without error.<br>4: Low level on SPI_NSS<br>3: Low level on IRQ_IN<br>2: Field detector: shall not be used in this mode<br>1: Tag Detector<br>0: Wake up at the end of cycle: TimeOut (see Byte MaxSleep)<br><br>*Example: Tag detector is used, but external MCU can additionally wake-up device by the impulse on UART_RX pin. Bits 1 and 3 must be set. So that WUFlags = 0x0A* |
| 1 | EnterCtrL | Enable specific block of the device, in our case we just need LFO and Sleep mode, i.e: 0x21 |
| 2 | EnterCtrlH | : 0x00 |
| 3 | WUCtrlL | On wake up we need HFO, LFO and Vdda (DAC enable is already set ON by default): 0x38 |
| 4 | WUCtrlH | Iref must be enable: 0x01 |
| 5 | LeaveCtrlL | For leaving Tag det mode: HFO and Vdda should be enable: 0x18 |
| 6 | LeaveCtrlH | : 0x00 |
| 7 | WUPeriod | Period of inactivity. Bigger value saves power, lower value decreases the reaction time. A good compromise is 8 times per second : 0x0F |
| 8 | OscStart | : 0xC0 (recommended value) |
| 9 | DacStart | : 0xC0 (recommended value) |
| A | DacDataL | Low threshold value for Field change detection. Note: Only the 6 MSB bits are available. See following chapter on Tag detector calibration |
| B | DacDataH | High threshold value for Field change detection. Note: Only the 6 MSB bits are available. See following chapter on Tag detector calibration |
| C | SwingsCnt | Number of pulses during HF burst for TAG detection, may depend on the environment (e.g. antenna). The recommended value with the DVK90130/32 is 0x1F. This value must be determined experimentally in the final application and environment. |
| D | MaxSleep | Used only when bit 0 in WUFlags is set to 1. It indicates the number of Sleep periods before wake-up even if TAG is not detected (timeout). |

Table 2**: "Idle" command parameters description**

| Meaning of power settings CtrlH:CtrlL | | |
|---|---|---|
| A | Register | Comment |
| 0 | CtrlL | 7 : Initial DAC compare index<br>6 : DAC enable<br>5 : LFO enable<br>4 : HFO enable<br>3 : VddA enable<br>2 : Hibernate enable<br>1 : Melexis reserved, must be set to 0<br>0 : Sleep mode enable<br>Note: bit 0,1 and 2 cannot be combined ; only one can be selected, the 2 other should be set to '0'. |
| 1 | CtrlH | 7 : RFU<br>6 - 2 : Melexis Reserved, must be set to '0'<br>1 : FDet enable<br>0 : IREF (needs to be set for TAG detector mode on wakeup - otherwise must be set to '0') |

Table 3: **Fields <EnterCtrl>, <WUCtrl> and <LeaveCtrl> definition in "Idle" command**

# 5 Timeout time MaxSleep

A Timeout time is available to leave automatically from the Tag detector mode after "MaxSleep+2" numbers of executions. There is also the possibility to disable this feature and so let the Tag Detector mode running indefinitely, until a real TAG event occurs.
The Timeout time is enabled by setting the bit **WUFlags[0]** to 1 and the parameter MaxSleep should then be specified. The timeout value is defined by the Equation 3.

# 6 Wake-Up Status

Before waking-up, the internal WUFlags register of the MLX90130/32 is updated with the status of the current wake-up event (e.g. TAG detected, Timeout after MaxSleep period …). Then $IRQ_{OUT}$ switches to a logical zero to inform the host microcontroller that the MLX90130/32 has been woken-up. The host microcontroller needs to read-back the results of the "Idle" command (see Table 1), then the $IRQ_{OUT}$ pin is set back to a logical one by the MLX90130/32.

| Wake-up register of the MLX90130/32 | | | |
|---|---|---|---|
| A | Register | Bit | Function |
| 62 | Wake-Up Event | 7:6 | LFO Prescaler:<br>Divides the LFO for the state machine<br>00: 32kHz<br>01: 16kHz<br>10: 8kHz<br>11: 4kHz |
| | | 5 | RFU |
| | | 4:0 | WUFlags:<br>Specifies the source of Wake-up:<br>bit4 : Low level on SS<br>bit3 : Low level on RX<br>bit2 : Field Detector<br>bit1 : TAG Detector<br>bit0 : WakeUp at the end of MaxSleep cycles |

Table 4: **Register Wake-up Event description**

# 7 Tag Detector Calibration

The calibration process should be performed with no tag in its near environment. It consists of executing a successive tag detection sequence using a well-known configuration, in order to establish two specific reference thresholds: DacDataL and DacDataH which will be programmed in the device before entering Tag Detector Mode, **these both thresholds are coded in 6 bits**.

The calibration process must be performed by the external host microcontroller. The parameters SwingsCnt OscStart and DacStart, should remain the same during the calibration process. For optimizing the calibration's timings, we will perform it with binary search method on parameter DacDataL. As the DAC has got 6 bits (the 6 MSB bits), 6 iterations are enough to determine the calibration levels.

The value of MaxSleep is set to 0x00, so only one burst will be performed at each calibration step. The WUFlags is configured to 0x03 in order to stop at the end of cycle.

The calibration process is started with the following values:

- MaxSleep = 0x00
- WUFlags = 0x03
- SwingsCnt = 0x3F (defined by the application)
- OscStart = 0xC0 (recommended value)
- DacStart = 0xC0 (recommended value)
- DacDataH = 0xFC
- DacDataL = 0xFC/2 = 0x7C (only first 6 MSB are taken in account for the DAC Thresholds)

Consequently, the resulting "Idle" command sent to configure the MLX90130/32 will be :

- Idle = 0x070E**03**21003801180022**C0C07CFC3F00**

At each step we will read back the content of WUFlags register status which can have only 2 possible values:

- **0x01** = Wake-up after MaxSleep number expired: it indicates that the HF field level is above DacDataL, so we have to **increase** this parameter to continue the calibration
- **0x02** = Wake-up by TAG detected event: it indicates that the field level is below DacDataL, so we have to **decrease** this parameter to continue the calibration.

The value found during the calibration process will be used as reference value **DacDataRef** to define the final values of DacDataL and DacDataH. To avoid too much sensitivity in the tag detection process, we recommend using a guard band. This value should correspond to 2 DAC steps: **Guard** = 0x08 (recommended value).

Final recommended values with guard band:

- DacDataL = **DacDataRef** – **Guard**
- DacDataH = **DacDataRef** + **Guard**

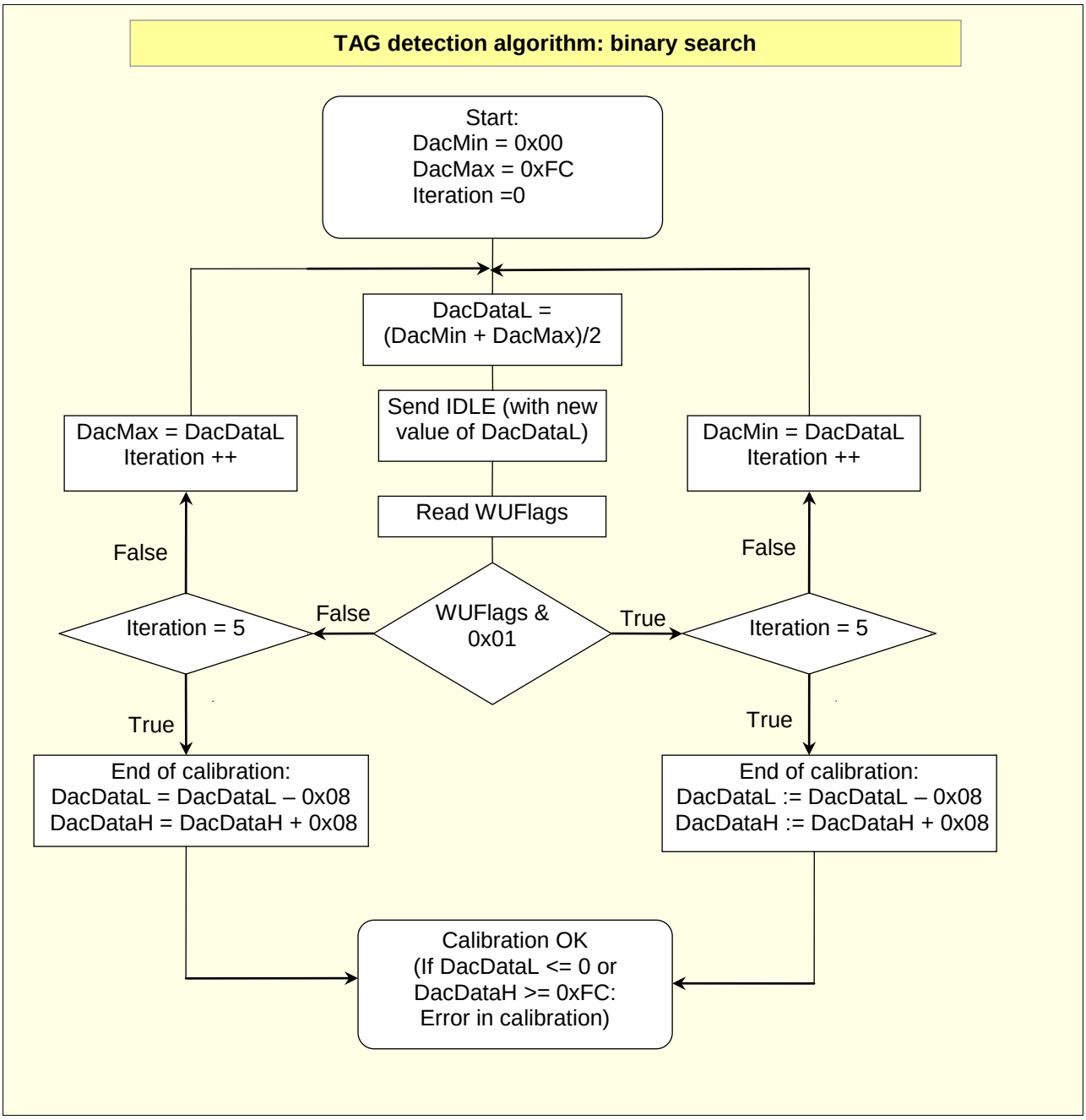


Figure 4: **TAG detection calibration procedure flow chart**

*Please note that only the 6 MSB are used for DacDataL and DacDataH, resulting in a maximum value of 0xFC.*

# 8 TAG Detector average consumption

The following chart illustrates the average power consumption measured on the MLX90132 evaluation board, with different settings of WUPERIOD with the following conditions

- fLFO = 32kHz
- OscStart = 0x3D (2ms)
- SwingCnt = 0x3F (burst of 64 carrier periods)
- WUPeriod = between 5 (56ms) up to 0xE3 (1832ms)

| WUPeriod [hex] | Sleep Time [ms] | Consumption [uA] |
|---|---|---|
| **5** | 56 | 83.8 |
| **12** | 160 | 58.2 |
| **1A** | 224 | 45.3 |
| **37** | 456 | 38.2 |
| **70** | 912 | 32.5 |
| **AA** | 1376 | 32.5 |
| **E3** | 1832 | 32.5 |

Figure 5: **Table of TAG detector average consumption measured on EVB90132**

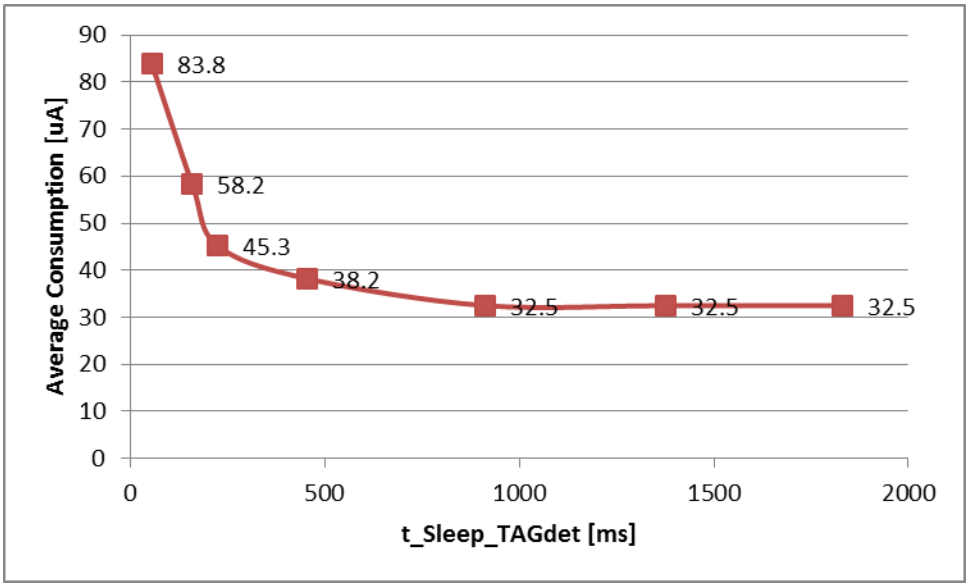


Figure 6: **Chart of TAG detector average consumption measured on EVB90132**

# 9 Appendix A. Tag Detector script source example in TCL

```
# ------------------------------------------------------------------------
# Name                              : Tag detector Calibration
# Description                       : This script performs an automatic
calibration on the TAG detector, by using the IDLE command.
#                                   : Once finished, it automatically enters the tag
detector mode for around 10s, (set by the timeout counter)
#                                   : At the end of timeout, it will tell if  a TAG
has been detected or not during this period
# Version                           : 001.3 (March 2013)
# Contact                           : dru@melexis.com
#-------------------------------------------------------------------------
#Load the TCL Plugin for MLX90132 applications
load D://Strfnfcaplugin.dll

set WUflags  03
set EnterCtrlL  21
set EnterCtrlH  00
set WUCtrlL 38
set WUCtrlH 01
set LeaveCtrlL  18
set LeaveCtrlH  00
set WUperiod  22
set OscStart  C0
set Dacstart C0
set dacL 24
set dacH FC
set Swingcnt 3F
set MaxSleep 00

#read back command via read_register command (@62)
set read_wup_reg 620100

# define delay
set ms 300

# Set Field OFF
FieldOff

#algorythme to check DacCal and genearte DacCabH, DacCabL
set loop 0
set dacmin 00
set dacmax FC

while {$loop <6} {
      set dac0 [expr 0x$dacmin / 2 + 0x$dacmax / 2 ]
      set dac1 [expr $dac0 & 0xFC ]
      set dac2 [format %02x $dac1]
      set dacL $dac2
      set tagdetcmd1
$WUflags$EnterCtrlL$EnterCtrlH$WUCtrlL$WUCtrlH$LeaveCtrlL$LeaveCtrlH
```

```
      set tagdetcmd2 $WUperiod$OscStart$Dacstart$dacL$dacH$Swingcnt$MaxSleep
      set Tagdetcommand1 $tagdetcmd1$tagdetcmd2

      set tagdetmode  [Idle $Tagdetcommand1]
      after $ms
      set ECHO [ECHO]
      set CheckAnswer [ReadRegister $read_wup_reg]

      puts "dac level L: $dacL, loop: $loop,  DacCheck Answer= $CheckAnswer"
      if {[expr $CheckAnswer & 0x000102] > 0 } {
            set dacmax $dacL
      } else {
      set dacmin $dacL
      }
      #increase iteration
      incr loop 1

}

# set thresholds: check last answer:
#if answer = 102 we passed the threshold,
# set high threshold value at this level +1 and low threshold value  at this
level 2
#if answer = 101 we have not passed the threshold,
# set high threshold value  +2 and low threshold value  at this level -1

      if {[expr $CheckAnswer & 0x000102] > 0 } {
            set dac1 [expr 0x$dacL + 0x04 ]
            set DacCabH [format %02x $dac1]
            set dac1 [expr 0x$dacL - 0x08 ]
            set DacCabL [format %02x $dac1]
      } else {
            set dac1 [expr 0x$dacL + 0x08 ]
            set DacCabH [format %02x $dac1]
            set dac1 [expr 0x$dacL - 0x04]
            set DacCabL [format %02x $dac1]
      }

puts "Low Threshold calibrated value: $DacCabL "
puts "High Threshold calibrated value: $DacCabH "

# set Tagdetector mode in loop with timeout of 10s
puts "Set TagDetector mode in loop with timeout of 10s"

# calculations of WUPeriod for a  time out = 10s
# t = 256*t_LFO *(WUperiod +2) (MaxSleep +1)
# if we want t = 10s, WUPeriod = 22, t_LFO = 1/20k (minimum value )
# Max sleep  = 1F,  (max value)
 # WUPeriod 10/ (256/ 32000 * (0x1F +1) ) -2 = 22, i.e. 0x16
set MaxSleep 1F
set WUperiod  16

# prepare command with new calibrated thresholds
```

```
set tagdetcmd1
$WUflags$EnterCtrlL$EnterCtrlH$WUCtrlL$WUCtrlH$LeaveCtrlL$LeaveCtrlH
set tagdetcmd2 $WUperiod$OscStart$Dacstart$DacCabL$DacCabH$Swingcnt$MaxSleep
set Tagdetcommand1 $tagdetcmd1$tagdetcmd2

# send command
set tagdetmode  [Idle $Tagdetcommand1]

# Wait for around 10s before reading back the status
puts "Wait for around 10s ...."
set loop 0

set ms 1000
while {$loop <10} {
      after $ms
      incr loop 1
}

      set ECHO [ECHO]
      set CheckAnswer [ReadRegister $read_wup_reg]

if {[expr $CheckAnswer & 0x000102] > 0 } {
      puts "TAG detector mode result:  $CheckAnswer, a TAG was detected "
} else {
      puts "TAG detector mode result:  $CheckAnswer, NO TAG was detected "
}
```