

---

## Advanced design tips for the machine learning core and finite state machine using Unico-GUI

---

Main components	
LSM6DSOX	iNEMO inertial module with embedded machine learning core: always-on 3D accelerometer and 3D gyroscope
LSM6DSO32X	iNEMO 6 DoF inertial module with <b>32 g accelerometer</b> and embedded machine learning core
LSM6DSRX	iNEMO inertial module with embedded machine learning core: always-on 3D accelerometer and 3D gyroscope
ISM330DHCX	iNEMO inertial module with embedded machine learning core: always-on 3D accelerometer and 3D gyroscope with digital output for <b>industrial applications</b>
IIS2ICLX	High-accuracy, high-resolution, low-power, <b>2-axis</b> digital inclinometer with embedded machine learning core

### Purpose and benefits

This design tip is a step-by-step guide to perform the following tasks:

- Parse and understand the structure of Unico configuration files (.ucf) to be able to carve out sections related to the machine learning core (MLC) or finite state machines (FSM) and edit the section related to the sensor configuration.
- Merge multiple decision trees and generate one configuration file (.ucf) for the machine learning core (MLC).
- Merge multiple finite state machines and generate one configuration file (.ucf) for the finite state machine (FSM).
- Merge the MLC configuration file with the FSM configuration file to generate one configuration file (.ucf) for the MLC and FSM.
- Configure the target sensor during the initialization phase or reconfigure the sensor at run-time.
- Verify the performance when the sensor is online in real-time or when the sensor is offline using data logs.

### Introduction to Unico-GUI and Unico configuration files (.ucf)

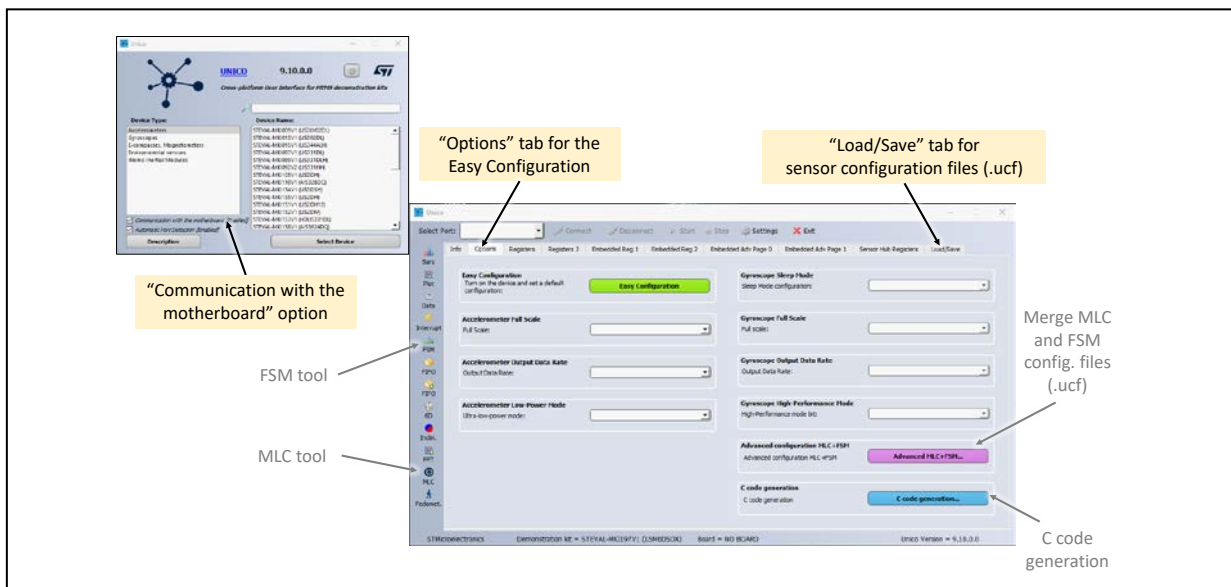
The best tool for ST MEMS sensors is [Unico-GUI](#) which is available for all platforms (Windows, Linux, Mac OS X) and can be downloaded from [www.st.com](http://www.st.com).

Unico-GUI can work in “online” or “offline” mode.

In **online mode**: the ProfiMEMS motherboard ([STEVAL-MKI109V3](#)) is connected via USB to the PC where Unico-GUI is running. The STEVAL-MKI sensor adapter must be plugged into the motherboard (example: [STEVAL-MKI197V1](#) for LSM6DSOX). The user must enable the “Communication with the motherboard” option and confirm the sensor model when Unico-GUI starts, then the main window with tabs and side menu is displayed (see Figure 1).

- In the “Options” tab, the “Easy Configuration” button will automatically configure and activate the sensor so that data can be acquired and visualized in real-time. In the same tab, a few essential parameters can be adjusted: power mode, output data rate or ODR, and full scale.
- Other tabs, such as “Registers” and “Embedded Reg”, allow the user to read and write all the sensor registers in real-time to finely tune the configuration.
- The FSM button allows the user to design, read, and write the finite state machine configuration in real-time. Debugging and step-by-step execution is also supported.
- The “Load/Save” tab allows the user to save the acquired data, and load or save the complete sensor configuration.

Figure 1. Unico-GUI tabs and side menu



In **offline mode**: the ProfiMEMS motherboard is not connected to the laptop but the FSM and MLC development tools are available. The user must disable the “Communication with the motherboard” option (see Figure 1).

- The **FSM** button allows the user to design programs for the finite state machine. The configuration file can be saved at the end of the process. See Figure 2.

- The **MLC** button allows the user to configure feature extraction and design decision trees for the machine learning core. The MLC configuration file is saved at the end of the process. See Figure 3.

Figure 2. FSM development tool for the finite state machine

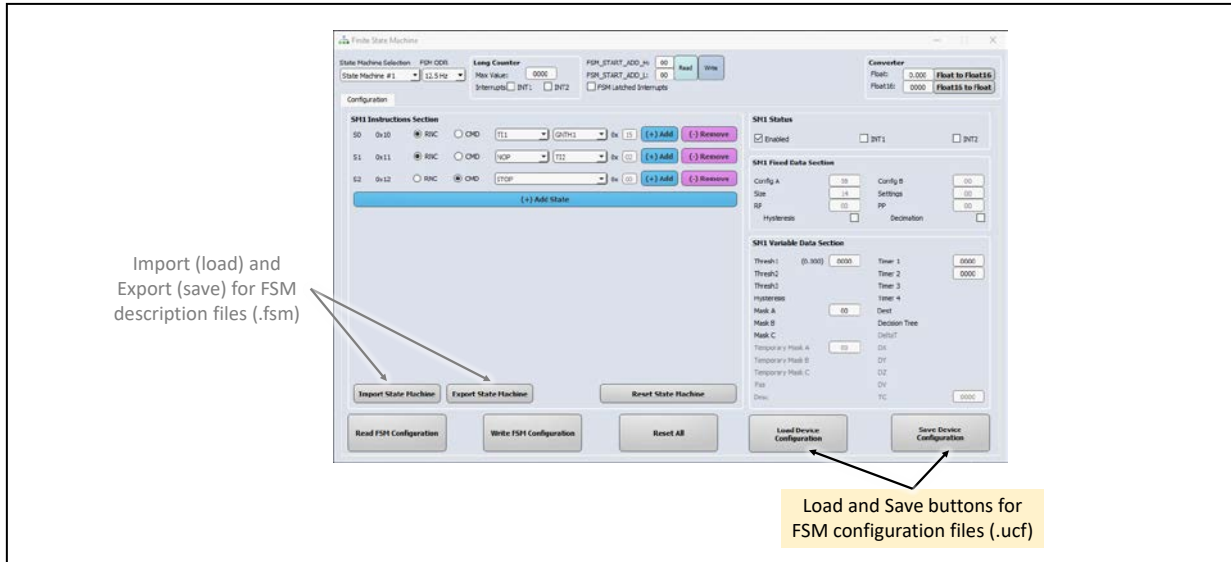
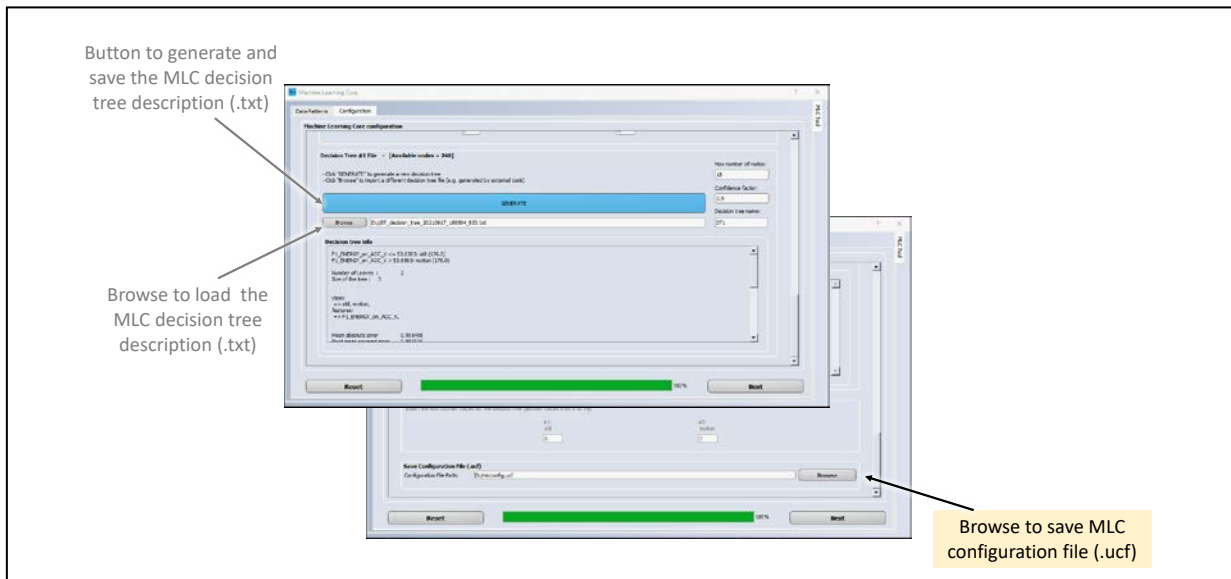


Figure 3. MLC development tool to configure the machine learning core



The configuration is saved as a text file with “.ucf” extension (Unico configuration file). Each line represents a write to a specific register and has the format “Ac xx yy” where Ac is a constant string, xx is the hexadecimal register address, and yy is the hexadecimal register value to be written. Example: “Ac 01 80” corresponds to writing the value 0x80 in register 0x01 (FUNC\_CFG\_ADDRESS).

---

The configuration file can also be converted to C code by clicking on the “C code generation” in the “Options” tab (see Figure 1): the user can browse and select the `.ucf` file to be converted to an `.h` file. The `.h` file contains a const array of struct `{ uint8_t address; uint8_t data }` which represents the sequence of the write to registers. Example: `“const ucf_line_t config[] = {{.address = 0x01, .data = 0x80,}};”`.

## Unico configuration file structure

As explained in the previous section, the Unico configuration file represents a sequence of writes to registers. The sequence is made of several sections dedicated to different banks. The details are presented in the following paragraphs.

Knowledge of the structure is essential to be able to carve out sections related to the machine learning core (MLC) or finite state machines (FSM) and edit the section related to the sensor configuration.

Every sensor has three separate **register banks**. Each register bank is dedicated to a specific sensor function. LSM6DSOX will be referenced here, but other sensors have a similar structure.

- The default bank to configure the **sensor functions** is accessed by setting register 0x01 (FUNC\_CFG\_ACCESS) to 0x00.
- The bank to configure the **embedded functions** (machine learning core or MLC, and finite state machine or FSM) is accessed by setting FUNC\_CFG\_ACCESS to 0x80.
- The bank to configure the **sensor hub functions** is accessed by setting FUNC\_CFG\_ACCESS to 0x40.

The bank dedicated to the embedded functions has an additional 8 pages which can be accessed using PAGE\_RW, PAGE\_SEL, PAGE\_ADDR, and PAGE\_VAL registers (for details check the sensor datasheet).

- Page 0 holds the settings used by the FSM when a magnetometer is connected to the sensor hub: scaling or sensitivity coefficients (to go from LSB to Gauss), hard-iron compensation coefficients (offset to be subtracted), and soft-iron compensation coefficients (6 coefficients for the 3x3 symmetric multiplier matrix).
- Page 1 holds the scaling or sensitivity coefficients used by the MLC when a magnetometer is connected to the sensor hub, other settings for the FSM (number of programs, start address, long counter timeout threshold), and embedded pedometer configuration.
- Pages 2 to 7 hold the actual FSM and MLC configuration. This configuration is generated by the FSM and MLC tool in Unico-GUI. It is uploaded to the sensor by executing the sequence of writes in the corresponding `.ucf` file or `.h` code.

As a practical example, this is the sequence to write to a page after sensor power-up:

- Set FUNC\_CFG\_ACCESS (register 0x01) = 0x80 to enable access to the embedded functions register bank
- Set PAGE\_RW (register 0x17) = 0x40 to enable write operation
- Set PAGE\_SEL (register 0x02) = 0xP1 where P is the number of the page (0x0 - 0xF)

- 
- Set PAGE\_ADDRESS (register 0x08) = byte address (0x00 to 0xFF) in the selected page
  - Set PAGE\_VALUE (register 0x09) = byte value (0x00 to 0xFF); the byte address is automatically incremented for subsequent assignments to PAGE\_VALUE, except when the address is 0xFF, at which point it is required to set PAGE\_SEL and PAGE\_ADDRESS to point to the first byte in the next page

The typical FSM or MLC configuration file is made of the following four types of sections. Some sections may be present multiple times in the generated .ucf file.

1. **Sensor initialization section** to power down the sensor. This is a safe practice to avoid conflicts when configuring the FSM and MLC.
  - Ac 10 00 (CTRL1\_XL = 0 to power down the accelerometer)
  - Ac 11 00 (CTRL2\_G = 0 to power down the gyroscope)
2. **Embedded function configuration start/end section** by writing to the embedded functions register bank (highlighted in blue in Figure 4). This is done to disable the embedded function before programming starts and re-enable it when programming is completed.
  - Ac 01 80 (FUNC\_CFG\_ACCESS = 0x80)
  - Sequence to disable (or enable) the target feature (FSM or MLC)
3. **Embedded function configuration core section** by writing to the pages of the embedded functions register bank (highlight in yellow in Figure 4):
  - Ac 01 80 (FUNC\_CFG\_ACCESS = 0x80)
  - Ac 17 40 (PAGE\_RW = 0x40 to enable write)
  - Ac 02 X1 (PAGE\_SEL = 0xX1 to select page X, X from 1 to 8)
  - Ac 08 XX (PAGE\_ADDRESS = 0xXX to set the address in the selected page)
  - Ac 09 XX (PAGE\_VALUE = 0xXX to write the value at specified address)
  - Ac 09 XX (PAGE\_VALUE = 0xXX to write the value at the next address)
  - Ac 09 XX until another page is selected or another address is selected
  - Ac 17 00 (PAGE\_RW = 0x00 to disable the write at the end of this section)

- 
4. **Sensor configuration section** to configure and activate the sensor as required by the application (highlighted in gray in Figure 4)
- Ac 01 00 (FUNC\_CFG\_ACCESS = 0x00)
  - Sequence to configure the sensor as needed

Regarding the sensor configuration section: care must be taken to configure the sensor in a way which is compatible with the FSM/MLC settings:

- Sensor output data rate (ODR) must be equal to or higher than the FSM/MLC ODR
- Interrupts must be enabled and configured in the embedded function registers and in the sensor registers so they are properly routed to the intended pin and visible (see Figure 5)

As an example, for LSM6DSOX the interrupt registers are: EMB\_FUNC\_INT1/INT2, FSM\_INT\_1A/1B, FSM\_INT\_2A/2B, MLC\_INT1/INT2, and MD1\_CFG, MD2\_CFG. See the [LSM6DSOX datasheet](#) for details.

Figure 4. Annotated FSM and MLC configuration file, FSM program as shown in Figure 5

### FSM .ucf configuration annotated

Sensor: power down

```

1 -- Finite State Machine Tool v9.10.0.9, STEVAL-MKI190V1 (L9M028X2)
2 -- CTRL1_ACCESS embedded function registers
3
4 Ac 18 00 // CTRL1_0 acc power-down, 2g, LFP2 off
5 Ac 11 00 // CTRL2_0 gyro power-down, 2500us
6
7 Ac 01 00 // FUNC_CFG_ACCESS embedded function registers
8
9 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
10 Ac 05 00 // ENB_FUNC_EN_B disable MLC, FIFO compression, FSM
11 Ac 0F 58 // ENB_FUNC_CFG_CTRL8 FSM CSR 1048h
12
13 Ac 46 01 // FSM_ENABLE_A disable FSM1 to FSM8
14 Ac 47 00 // FSM_ENABLE_B disable FSM9 to FSM16
15 Ac 04 00 // ENB_FUNC_INT1 no interrupt from FSM long counter, sig_motion, tilt, pedometer
16 Ac 06 01 // FSM_INT1_A interrupt from FSM1 routed to INT1 pin, no interrupt from FSM2 to FSM8
17 Ac 06 00 // FSM_INT1_B no interrupt from FSM9 to FSM16 to INT1 pin
18 Ac 06 00 // ENB_FUNC_INT2 no interrupt from FSM long counter, sig_motion, tilt, pedometer
19 Ac 06 00 // FSM_INT2_A no interrupt from FSM1 to FSM8 to INT2 pin
20 Ac 06 00 // FSM_INT2_B no interrupt from FSM9 to FSM16 to INT2 pin
    
```

#### Embedded functions (FSM disable and configuration)

```

1 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
2 Ac 05 00 // ENB_FUNC_EN_B disable MLC, FIFO compression, FSM
3 Ac 0F 58 // ENB_FUNC_CFG_CTRL8 FSM CSR 1048h
4 Ac 46 01 // FSM_ENABLE_A disable FSM1 to FSM8
5 Ac 47 00 // FSM_ENABLE_B disable FSM9 to FSM16
6 Ac 04 00 // ENB_FUNC_INT1 no interrupt from FSM long counter, sig_motion, tilt, pedometer
7 Ac 06 01 // FSM_INT1_A interrupt from FSM1 routed to INT1 pin, no interrupt from FSM2 to FSM8
8 Ac 06 00 // FSM_INT1_B no interrupt from FSM9 to FSM16 to INT1 pin
9 Ac 06 00 // ENB_FUNC_INT2 no interrupt from FSM long counter, sig_motion, tilt, pedometer
10 Ac 06 00 // FSM_INT2_A no interrupt from FSM1 to FSM8 to INT2 pin
11 Ac 06 00 // FSM_INT2_B no interrupt from FSM9 to FSM16 to INT2 pin
    
```

#### Embedded functions, page #1 (FSM configuration)

```

20 Ac 17 40 // PAGE_SW write enable
21 Ac 02 11 // PAGE_SEL select page #1
22 Ac 08 7A // PAGE_ADDR address of FSM_LC_TIMEOUT_L in page #1
23 Ac 09 00 // FSM_LC_TIMEOUT_L long counter low byte reset to 0
24 Ac 09 00 // FSM_LC_TIMEOUT_H long counter high byte reset to 0
25 Ac 09 01 // FSM_PROGRAMS set to 1
26 Ac 09 01 // dummy write to keep incrementing the write address
27 Ac 09 00 // FSM_START_ADD_L start address low byte (address 0x00)
28 Ac 09 04 // FSM_START_ADD_H start address high byte (page #4)
29
30 Ac 02 41 // PAGE_SEL select page #4
31 Ac 08 00 // PAGE_ADDR address in page #4
32 Ac 09 96 // 0x00, FSM CONFIG A 2 thresholds, 1 mask, 1 long timer, 2 short timers
33 Ac 09 00 // 0x01, FSM CONFIG B no declination, no hysteresis, no angle computation, no long counter
34 Ac 09 13 // 0x02, SIZE
35 Ac 09 00 // 0x03, SETTINGS
36 Ac 09 12 // 0x04, RESET POINTER
37 Ac 09 00 // 0x05, PROGRAM POINTER
38 Ac 09 33 // 0x06, THRESH1 low byte
39 Ac 09 32 // 0x07, THRESH1 high byte
40 Ac 09 00 // 0x08, THRESH2 low byte
41 Ac 09 3E // 0x09, THRESH2 high byte
42 Ac 09 0F // 0x0A, MASKA
43 Ac 09 00 // 0x0B, MASKB
44 Ac 09 00 // 0x0C, TC low byte
45 Ac 09 00 // 0x0D, TC high byte
46 Ac 09 00 // 0x0E, TIMER1 low byte reload value
47 Ac 09 00 // 0x0F, TIMER1 high byte reload value
48 Ac 09 01 // 0x10, TIMER3 reload value
49 Ac 09 02 // 0x11, TIMER4 reload value
50 Ac 09 51 // 0x12, state a0: reset ONDM, next T11
51 Ac 09 51 // 0x13, state a1: reset ONDM, next T11
52 Ac 09 05 // 0x14, state a2: reset NCP, next ONDM
53 Ac 09 04 // 0x15, state a4: reset NCP, next T14
54 Ac 09 51 // 0x16, state a5: reset ONDM, next T11
55 Ac 09 22 // 0x17, state a6: command CONTROL
    
```

#### Embedded functions (FSM enable)

```

57 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
58 Ac 05 01 // ENB_FUNC_EN_B disable MLC, FIFO compression, enable FSM
59 Ac 17 00 // PAGE_SW write enable
60 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
61
    
```

#### Sensor: configuration

```

62 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
63 Ac 02 00 // PIR_CTRL
64 Ac 04 00 // SAS_TPH_L
65 Ac 05 00 // SAS_TPH_H
66 Ac 06 00 // SAS_IP
67 Ac 07 00 // FIFO_CTRL1
68 Ac 08 00 // FIFO_CTRL2
69 Ac 09 00 // FIFO_CTRL3
70 Ac 0A 00 // FIFO_CTRL4
71 Ac 0B 00 // COUNTER_BDR_RFC1
72 Ac 0C 00 // INT1_CTRL
73 Ac 0E 00 // INT2_CTRL
74 Ac 10 00 // CTRL1_XL
75 Ac 11 00 // CTRL2_0
76 Ac 12 00 // CTRL3_C
77 Ac 13 00 // CTRL4_C
78 Ac 14 00 // CTRL5_C
79 Ac 15 00 // CTRL6_C
80 Ac 16 00 // CTRL7_D
81 Ac 17 00 // CTRL8_XL
82 Ac 18 00 // CTRL9_XL
83 Ac 19 00 // CTRL10_C
84 Ac 56 00 // SAP_CF00
85 Ac 57 00 // SAP_CF01
86 Ac 58 00 // SAP_CF02
87 Ac 59 00 // SAP_THS_60
88 Ac 5A 00 // IME_DUR2
89 Ac 5B 00 // WAKE_UP_DUR
90 Ac 5C 00 // WAKE_UP_DUR
91 Ac 5D 00 // FREE_FALL
92 Ac 5E 00 // MCL_CFG
93 Ac 5F 00 // MCL_CFG
94 Ac 60 00 // SAS_DT_DMD_CODE
95 Ac 61 00 // SAS_DT_RUB
96 Ac 62 00 // I3C_BUS_AVS
97
    
```

### MLC .ucf configuration annotated

Sensor: power down

```

1 -- Machine Learning Core Tool v9.10.0.0, L9M028X2
2 -- MLC0_ACCESS embedded function registers
3
4 Ac 18 00 // CTRL1_0 acc power-down, 2g, LFP2 off
5 Ac 11 00 // CTRL2_0 gyro power-down, 2500us
6
7 Ac 01 00 // FUNC_CFG_ACCESS embedded function registers
8
9 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
10 Ac 05 00 // ENB_FUNC_EN_B disable MLC, FIFO compression, FSM
11 Ac 0F 58 // ENB_FUNC_CFG_CTRL8 FSM CSR 1048h
12
13 Ac 46 01 // FSM_ENABLE_A disable FSM1 to FSM8
14 Ac 47 00 // FSM_ENABLE_B disable FSM9 to FSM16
15 Ac 04 00 // ENB_FUNC_INT1 no interrupt from FSM long counter, sig_motion, tilt, pedometer
16 Ac 06 01 // FSM_INT1_A interrupt from FSM1 routed to INT1 pin, no interrupt from FSM2 to FSM8
17 Ac 06 00 // FSM_INT1_B no interrupt from FSM9 to FSM16 to INT1 pin
18 Ac 06 00 // ENB_FUNC_INT2 no interrupt from FSM long counter, sig_motion, tilt, pedometer
19 Ac 06 00 // FSM_INT2_A no interrupt from FSM1 to FSM8 to INT2 pin
20 Ac 06 00 // FSM_INT2_B no interrupt from FSM9 to FSM16 to INT2 pin
    
```

#### Embedded functions (MLC disable)

```

1 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
2 Ac 05 00 // ENB_FUNC_EN_B disable MLC, FIFO compression, FSM
3 Ac 0F 58 // ENB_FUNC_CFG_CTRL8 FSM CSR 1048h
4 Ac 46 01 // FSM_ENABLE_A disable FSM1 to FSM8
5 Ac 47 00 // FSM_ENABLE_B disable FSM9 to FSM16
6 Ac 04 00 // ENB_FUNC_INT1 no interrupt from FSM long counter, sig_motion, tilt, pedometer
7 Ac 06 01 // FSM_INT1_A interrupt from FSM1 routed to INT1 pin, no interrupt from FSM2 to FSM8
8 Ac 06 00 // FSM_INT1_B no interrupt from FSM9 to FSM16 to INT1 pin
9 Ac 06 00 // ENB_FUNC_INT2 no interrupt from FSM long counter, sig_motion, tilt, pedometer
10 Ac 06 00 // FSM_INT2_A no interrupt from FSM1 to FSM8 to INT2 pin
11 Ac 06 00 // FSM_INT2_B no interrupt from FSM9 to FSM16 to INT2 pin
    
```

#### Embedded functions pages (MLC program)

```

21 Ac 17 40 // PAGE_SW write enable
22 Ac 02 11 // PAGE_SEL select page #1
23 Ac 08 7A // PAGE_ADDR address in page #1
24 Ac 09 00 // PAGE_VALUE for address 0x7A
25 Ac 09 00 //
26 Ac 09 00 //
27 Ac 09 00 //
28 Ac 09 00 //
29 Ac 02 11 // PAGE_SEL select page #1
30 Ac 08 7A // PAGE_ADDR address in page #1
31 Ac 09 00 // PAGE_VALUE for address 0x7A
32 Ac 09 00 //
33 Ac 09 00 //
34 Ac 09 00 //
35 Ac 09 00 //
36 Ac 02 31 // PAGE_SEL select page #3
37 Ac 08 32 // PAGE_ADDR address in page #3
38 Ac 09 00 // PAGE_VALUE for address 0x3C
39 Ac 09 00 //
40 Ac 09 00 //
41 Ac 09 00 //
42 Ac 09 00 //
43 Ac 09 00 //
44 Ac 09 00 //
45 Ac 09 00 //
46 Ac 09 00 //
47 Ac 09 00 //
48 Ac 09 00 //
49 Ac 09 00 //
50 Ac 09 00 //
51 Ac 02 31 // PAGE_SEL select page #3
52 Ac 08 32 // PAGE_ADDR address in page #3
53 Ac 09 00 // PAGE_VALUE for address 0x3C
54 Ac 09 00 //
55 Ac 09 00 //
56 Ac 09 00 //
57 Ac 09 00 //
58 Ac 09 00 //
59 Ac 09 00 //
60 Ac 09 00 //
61 Ac 09 00 //
62 Ac 09 00 //
63 Ac 09 00 //
64 Ac 09 00 //
65 Ac 09 00 //
66 Ac 09 00 //
67 Ac 09 00 //
68 Ac 09 00 //
69 Ac 09 00 //
70 Ac 09 00 //
71 Ac 09 00 //
72 Ac 09 00 //
73 Ac 09 00 //
74 Ac 09 00 //
75 Ac 09 00 //
76 Ac 09 00 //
77 Ac 09 00 //
78 Ac 09 00 //
79 Ac 09 00 //
80 Ac 09 00 //
81 Ac 09 00 //
82 Ac 09 00 //
83 Ac 09 00 //
84 Ac 09 00 //
85 Ac 09 00 //
86 Ac 09 00 //
87 Ac 09 00 //
88 Ac 09 00 //
89 Ac 09 00 //
90 Ac 09 00 //
91 Ac 09 00 //
92 Ac 09 00 //
93 Ac 09 00 //
94 Ac 09 00 //
95 Ac 09 00 //
96 Ac 09 00 //
97 Ac 09 00 //
98 Ac 09 00 //
99 Ac 09 00 //
100 Ac 09 00 //
    
```

#### Sensor: configuration (\*)

```

81 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
82 Ac 18 00 // CTRL1_0 reset BDU and SPI interface
83
    
```

#### (MLC program continued)

```

101 Ac 01 00 // FUNC_CFG_ACCESS embedded function registers
102 Ac 17 40 // PAGE_SW write enable
103 Ac 02 11 // PAGE_SEL select page #1
104 Ac 08 7A // PAGE_ADDR address in page #1
105 Ac 09 37 // PAGE_VALUE for address 0x6E
106 Ac 09 00 //
107 Ac 09 00 //
108 Ac 09 00 //
109 Ac 09 00 //
110 Ac 09 00 //
111 Ac 09 00 //
112 Ac 09 00 //
113 Ac 09 00 //
114 Ac 09 00 //
115 Ac 09 00 //
116 Ac 09 00 //
117 Ac 09 00 //
118 Ac 09 00 //
119 Ac 09 00 //
120 Ac 09 00 //
121 Ac 09 00 //
122 Ac 09 00 //
123 Ac 09 00 //
124 Ac 09 00 //
125 Ac 09 00 //
126 Ac 09 00 //
127 Ac 09 00 //
128 Ac 09 00 //
129 Ac 09 00 //
130 Ac 09 00 //
131 Ac 09 00 //
132 Ac 09 00 //
133 Ac 09 00 //
134 Ac 09 00 //
135 Ac 09 00 //
136 Ac 09 00 //
137 Ac 09 00 //
138 Ac 09 00 //
139 Ac 09 00 //
140 Ac 09 00 //
141 Ac 09 00 //
142 Ac 09 00 //
143 Ac 09 00 //
144 Ac 09 00 //
145 Ac 09 00 //
146 Ac 09 00 //
147 Ac 09 00 //
148 Ac 09 00 //
149 Ac 09 00 //
150 Ac 09 00 //
151 Ac 09 00 //
152 Ac 09 00 //
153 Ac 09 00 //
154 Ac 09 00 //
155 Ac 09 00 //
156 Ac 09 00 //
157 Ac 09 00 //
158 Ac 09 00 //
159 Ac 09 00 //
160 Ac 09 00 //
161 Ac 09 00 //
162 Ac 09 00 //
163 Ac 09 00 //
164 Ac 09 00 //
165 Ac 09 00 //
166 Ac 09 00 //
167 Ac 09 00 //
168 Ac 09 00 //
169 Ac 09 00 //
170 Ac 09 00 //
171 Ac 09 00 //
172 Ac 09 00 //
173 Ac 09 00 //
174 Ac 09 00 //
175 Ac 09 00 //
176 Ac 09 00 //
177 Ac 09 00 //
178 Ac 09 00 //
179 Ac 09 00 //
180 Ac 09 00 //
181 Ac 09 00 //
182 Ac 09 00 //
183 Ac 09 00 //
184 Ac 09 00 //
185 Ac 09 00 //
186 Ac 09 00 //
187 Ac 09 00 //
188 Ac 09 00 //
189 Ac 09 00 //
190 Ac 09 00 //
191 Ac 09 00 //
192 Ac 09 00 //
193 Ac 09 00 //
194 Ac 09 00 //
195 Ac 09 00 //
196 Ac 09 00 //
197 Ac 09 00 //
198 Ac 09 00 //
199 Ac 09 00 //
200 Ac 09 00 //
    
```

#### Embedded functions (MLC enable)

```

181 Ac 04 00 // ENB_FUNC_EN_A disable sig_motion, tilt, pedometer
182 Ac 05 00 // ENB_FUNC_EN_B disable MLC, FIFO compression, FSM
183 Ac 0F 58 // ENB_FUNC_CFG_CTRL8 FSM CSR 1048h
184 Ac 46 01 // FSM_ENABLE_A disable FSM1 to FSM8
185 Ac 47 00 // FSM_ENABLE_B disable FSM9 to FSM16
186 Ac 04 00 // ENB_FUNC_INT1 no interrupt from FSM long counter, sig_motion, tilt, pedometer
187 Ac 06 01 // FSM_INT1_A interrupt from FSM1 routed to INT1 pin, no interrupt from FSM2 to FSM8
188 Ac 06 00 // FSM_INT1_B no interrupt from FSM9 to FSM16 to INT1 pin
189 Ac 06 00 // ENB_FUNC_INT2 no interrupt from FSM long counter, sig_motion, tilt, pedometer
190 Ac 06 00 // FSM_INT2_A no interrupt from FSM1 to FSM8 to INT2 pin
191 Ac 06 00 // FSM_INT2_B no interrupt from FSM9 to FSM16 to INT2 pin
    
```

#### Sensor: configuration (\*)

```

181 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
182 Ac 12 44 // CTRL1_0 enable BDU and auto-increment
183
    
```

#### Embedded functions (MLC ODR)

```

184 Ac 01 00 // FUNC_CFG_ACCESS embedded function registers
185 Ac 00 15 // ENB_FUNC_ODR_CTRL0 set MLC ODR = 35 Hz
186
    
```

#### Sensor: configuration

```

187 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
188 Ac 18 20 // CTRL1_0 acc ODR = 25 Hz Lmode, 2g, LFP2 off
189 Ac 11 00 // CTRL2_0 gyro power-down, 2000us
190 Ac 3E 02 // MCL_CFG embedded function interrupt routed to INT1 pin
191
    
```

#### Embedded functions (MLC interrupts)

```

191 Ac 01 00 // FUNC_CFG_ACCESS embedded function registers
192 Ac 06 01 // MLC_INT1 interrupt from MLC1 routed to INT1 pin
193
    
```

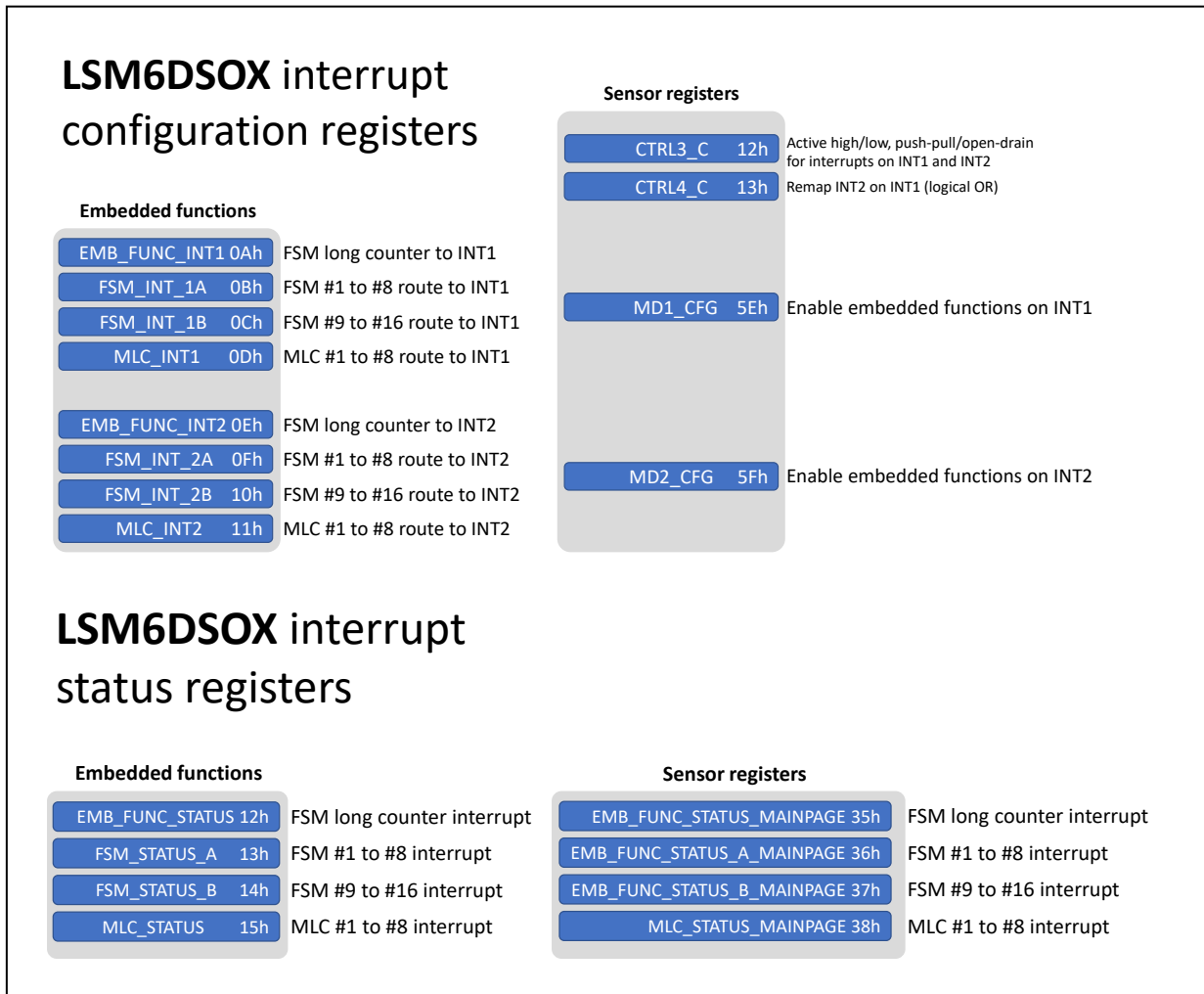
#### Sensor: configuration

```

194 Ac 01 00 // FUNC_CFG_ACCESS sensor registers
195
    
```



Figure 5. Interrupt configuration registers in LSM6DSOX



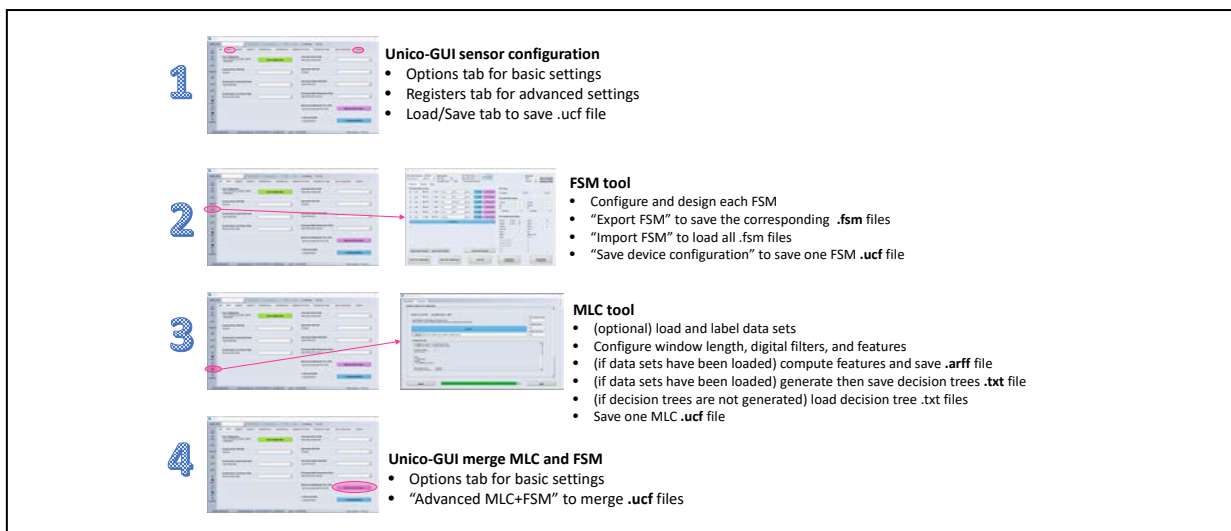
### FSM and MLC configuration: recommended workflow (see Figure 6)

1. **[Unico-GUI online mode only] Tune the sensor configuration** and save it in a dedicated file. When choosing the sensor configuration, it is important to consider the constraints on full-scale and ODR:
  - **The full-scale setting is shared.** The sensor, the FSM and the MLC all use the same setting.
  - **The output data rate setting (ODR) must be equal to or higher** than those used by the FSM and MLC.
    - If the FSM and MLC ODR are lower than the sensor ODR, then the sensor data will be down-sampled by sample selection; there is no anti-aliasing filtering, therefore aliasing can happen and can affect FSM detection and MLC classification performance.
    - FSM data can be further down-sampled independently by each finite state machine.
2. **Design and configure all finite state machines** using the dedicated Unico-GUI tool. It is recommended to export the descriptions using the “Export State Machine” button (.fsm files), to be able to edit or merge them later. Save the FSM configuration file using the “Save Device Configuration” button (.ucf file). This configuration file includes the sensor settings chosen in the first step.



- Finite state machines can be merged by loading their .fsm description and then saving the FSM .ucf file (see Figure 2).
  - The **FSM output data rate must be set equal to the highest in use by the merged FSMs**, while the decimation factor of each FSM must be configured to feed each of them with the respective expected output data rate.
3. **Configure the machine learning core** using the dedicated Unico-GUI tool. If data logs are available, they can be loaded and labeled to enable feature computation (.arff file) and automatic class labeling. The user must choose the MLC settings (ODR, window length and digital filters), the features that will be used by decision trees, and the class labels. The generated decision tree(s) will be stored in a .txt file and then will be converted into a .ucf file. This configuration file will include the sensor settings selected in the first step.
    - Multiple MLC decision trees can be merged by importing the corresponding .txt description and then saving the MLC .ucf file (see Figure 3).
    - The **MLC output data rate (ODR), window length, and full-scale settings are shared and must be equal for all merged decision trees**. The set of digital filters and features is the union of all sets from the merged decision trees.
  4. **If needed, merge the FSM and MLC configurations** by clicking on the “Advanced MLC+FSM” button in the “Options” tab. Unico-GUI will take care to map the FSM/MLC programs in the available memory; if the memory is not large enough, the merge fails, and Unico-GUI will notify the user.
    - The **sensor settings contained in the FSM configuration and MLC configuration must match**. No difference in ODR, full-scale, or other sensor settings is acceptable.
  5. **Configuration step:** upload the merged MLC/FSM configuration to the target device. This will power down the sensor, configure the MLC/FSM, and reactivate the sensor.
    - Note that sensor settings are not saved before power down. If the first step has not been executed, sensor settings are not restored by the merged configuration file. When this happens, the additional configuration step must be performed. Example: the interrupt configuration may need to be restored.
  6. **Additional configuration step:** upload the sensor configuration after the MLC/FSM have been configured. This final step ensures the sensor is working with the correct settings.

Figure 6. Recommended workflow to create one configuration file for MLC and FSM



## How to design and merge finite state machines

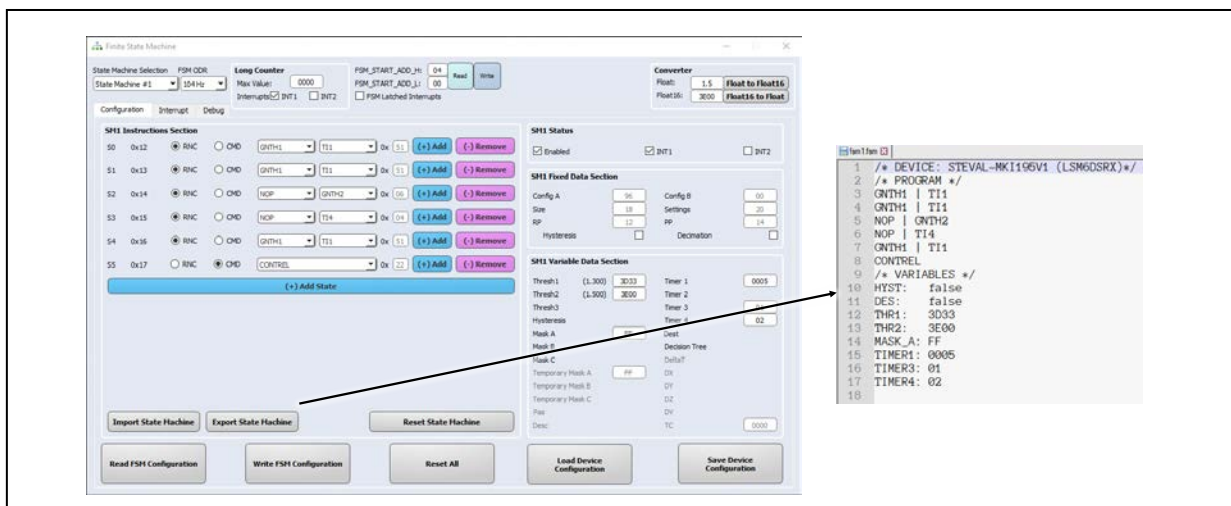
Finite state machines are used to detect a **sequence of events with specified timing** (example: hand or head gestures). The finite state machine is made of a chain of states. The machine can reset itself or advance to the next state when a specific condition is met (input above or below the programmed threshold, or expiration of a timer, or a scenario detected by a decision tree). When the right state is reached, an interrupt can be triggered (usually this is the last state, but any state can trigger an interrupt).

Here is an example for single-tap detection:

- Initial state: if the acceleration along all axes is below a small threshold for at least 100 msec, then move to state 1, otherwise stay in this state.
- State 1: if the acceleration of any axis exceeds another possibly larger threshold, then move to state 2, otherwise move back to the initial state.
- State 2: if the acceleration of all axes returns below that threshold within 10 msec, then move to state 3.
- State 3: the acceleration is ignored for 20 msec (debouncing), move to the next state when the timer expires.
- State 4: if the acceleration of all axes is below the small threshold for 50 msec, then move to the final state, otherwise move back to the initial state.
- Final state: the interrupt is triggered to signal single-tap detection

The finite state machine can be configured as shown in Figure 7. It is recommended to export the .fsm description (a text file) to be able to edit or merge it with other finite state machines at a later time.

Figure 7. FSM example for single-tap detection



The screenshot displays the 'Finite State Machine' configuration interface. On the left, the 'SH11 Instructions Section' lists five states (S0 to S5) with their respective conditions and actions. On the right, the 'SH11 States' section shows configuration parameters for each state, including thresholds, timers, and masks. A code editor on the far right shows the generated assembly code for the FSM, with an arrow pointing from the state configuration to the corresponding code lines.

```
1 /* DEVICE: STEVAL-MKI106V1 (LSM6DSRX) */
2 /* PROGRAM */
3 GNTH1 | TI1
4 GNTH1 | TI1
5 NOP | GNTH2
6 NOP | TI4
7 GNTH1 | TI1
8 CONTROL
9 /* VARIABLES */
10 HYST: false
11 DES: false
12 THR1: 3033
13 THR2: 3000
14 MASK_A: FF
15 TIMER1: 0005
16 TIMER3: 01
17 TIMER4: 02
18
```

---

The designer can program multiple finite state machines which will be executed concurrently. Use the pull-down menu to select a specific state machine, then write the corresponding program or import the .fsm description.

If a finite state machine is imported, care must be taken to ensure that it works with the correct output data rate (ODR).

- If the imported state machine works at a rate lower than the FSM ODR, decimation must be enabled and the DEST field must be set with the necessary decimation factor (2 or higher).
- Conversely, if it works at a rate higher than the FSM ODR, then the FSM ODR must be set equal to this higher ODR, and decimation must be enabled and configured for all other state machines.
- The decimation corresponds to a sample selection, with no anti-aliasing filtering, therefore aliasing can happen and may affect FSM detection performance.

**NOTE:** The .fsm description does not include the ODR information. The designer must take note of this information separately. If the FSM is using filtered signals from the MLC, the designer must also take note of the filter type and coefficients.

Example: The 1<sup>st</sup> state machine is designed to work at 26 Hz, FSM ODR = 26 Hz; then a 2<sup>nd</sup> state machine is imported but it is designed to work at 104 Hz; FSM ODR must be set equal to 104 Hz, and decimation must be configured for the 1<sup>st</sup> state machine (DEST = 4 because  $104 \text{ Hz} / 4 = 26 \text{ Hz}$ ).

When all desired finite state machines have been configured, the designer can press the “Save Device Configuration” button to generate and save the .ucf file which contains the sensor and the FSM configuration (see Figure 2).

The full-scale setting of the sensor can be set to any value because the samples are scaled to the correct unit of measurement (*g* for accelerometer samples, deg/s (dps) for gyroscope samples, Gauss for magnetometer samples). However, if a finite state machine has thresholds set for a given level, that level must be within the programmed full-scale range.

Example: If there is a  $\pm 3 \text{ g}$  threshold for accelerometer data, then the full-scale must be  $\pm 4 \text{ g}$  or larger otherwise that threshold can never be reached. Note that the RMS noise and spurious offset does depend on the full-scale setting and the design must take this into account. The detection performance of the finite state machine may be affected.

The finite state machine input can come from one of the filters configured for the machine learning core (check the SINMUX command for details). When this happens, the designer must also generate and save the .ucf file for the MLC, then merge it with the .ucf for the FSM as explained in the following dedicated paragraph (“How to merge FSM and MLC configurations”). For details see [LSM6DSOX AN5273](#).

---

## How to design and merge decision trees in the machine learning core

The machine learning core can be used to **detect a scenario among N possible scenarios based on its “fingerprint”**. The fingerprint is made of features extracted from the input signal (signal samples are grouped into consecutive non-overlapping windows of specified length, then features are computed on every window). Before feature extraction, the signal can be filtered by 2<sup>nd</sup> order IIR filters to isolate specific frequency bands. After feature extraction, decision trees classify the fingerprint, thereby identifying the scenario.

A meta-classifier can be used to post-process the output and make it more stable: output codes are subdivided into groups based on their numeric value (group 1 for codes 0 to 3, group 2 for codes 4 to 7, and so on). Each group has an associated counter with a threshold. The counter is incremented when the output is within the group, and decremented otherwise. The final output is updated whenever one of the counters reaches its threshold.

Here is an example for fan speed monitoring:

- ODR is set to maximum (104 Hz for LSM6DSOX<sup>1</sup>) to capture high frequencies, the window is set to 52 samples (2 outputs per second will be generated). Longer windows can be more accurate, especially for scenarios characterized by low frequencies, but may be affected by transients going from one scenario to another and have the disadvantage of less frequent outputs. Conversely, shorter windows have the advantage of more frequent outputs.
- No digital filter is applied because no FFT analysis has been performed – in practice it is always convenient to perform a time or frequency domain analysis to decide the best combination of filters and features (see DT0139 for further details).
- Three features are selected (variance, number of zero-crossings, number of peaks) but the automatic design algorithm embedded in Unico-GUI correctly selects only the most relevant (zero-crossings) to achieve >99% accuracy

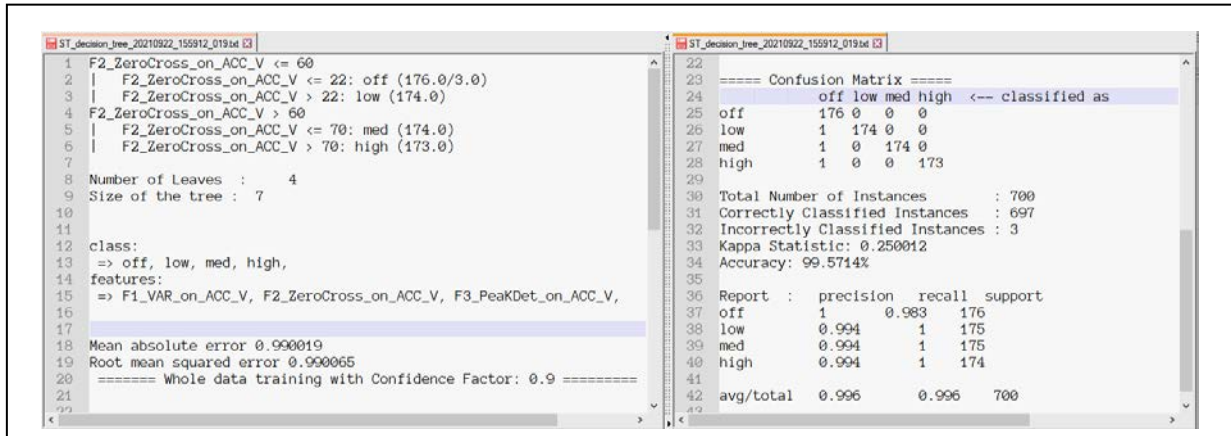
The generated decision tree can be seen in Figure 8. This is the content of the .txt description saved during the process. There is a description of the decision tree itself (for each decision node the designer can see the feature and the threshold), then there is the list of class labels and the list of features, and finally a report on the classification performance (confusion matrix, accuracy, and other metrics).

**NOTE:** The .txt description does not include the information on the ODR and window length. The designer must take note of this information separately. If the MLC is using filtered signals, the designer must take note of the filter type and coefficients. Some features may have additional parameters that must be saved; for example, zero-crossing and peak detector have a threshold parameter, which defaults to 0.0 g, which is not saved in the .txt description.

---

<sup>1</sup> Other devices may have other different limits. Consult the datasheet to determine the maximum FSM and MLC output data rate.

Figure 8. Decision tree for fan speed monitoring: 4 classes (off, low, med, high)



The designer can merge multiple decision trees during the MLC configuration process, but the following constraints must be met:

- All merged decision trees must be designed to work with the same output data rate (ODR), window length, and full-scale setting.
  - Any decision tree that does not match the common settings must be redesigned, that is, data logs with the appropriate ODR, and full-scale settings must be available.
  - For this reason, it is recommended to collect data logs at the highest possible data rate supported by the MLC (104 Hz for LSM6DSOX) so that it is possible to down-sample to any ODR. If this is not possible, the designer can perform sample-rate-conversion on available data logs (e.g., using the `interp1` or `resample` functions in Matlab).
  - The full-scale setting of the sensor can be anything because the samples are scaled to the correct unit of measurement ( $g$  for accelerometer samples,  $\text{deg/s}$  (dps) for gyroscope samples, Gauss for magnetometer samples). However, if a decision tree has thresholds set for a given level, that level must be within the programmed full-scale range. Example: if there is a  $\pm 3 g$  threshold for accelerometer data, then the full-scale must be  $\pm 4 g$  or larger otherwise that threshold can never be reached. Note that the RMS noise and spurious offset does depend on the full-scale setting and the design must take this into account. The classification performance of the decision tree may be affected.
- The set of filters to be configured is the union of the filters used by the merged trees. Similarly, the set of features to be enabled is the union of the features used by the merged trees. The window length for the feature extraction is shared, all decision trees use the same length.
  - The maximum number of features as well as the number of nodes is limited (for LSM6DSOX there can be 31 features and 256 nodes for all decision trees<sup>2</sup>). The number of nodes may be reduced depending on the number of filters and features enabled. Unico-GUI will inform the user when the MLC processing capacity is exceeded.
  - If the merged decision trees have different window lengths, one can try to use the longest window. Otherwise, decision trees must be redesigned, and data logs must be available.

<sup>2</sup> Other devices may have different limits. As an example, LSM6DSRX has an MLC which can handle up to 512 nodes, also it can handle up to 63 features.

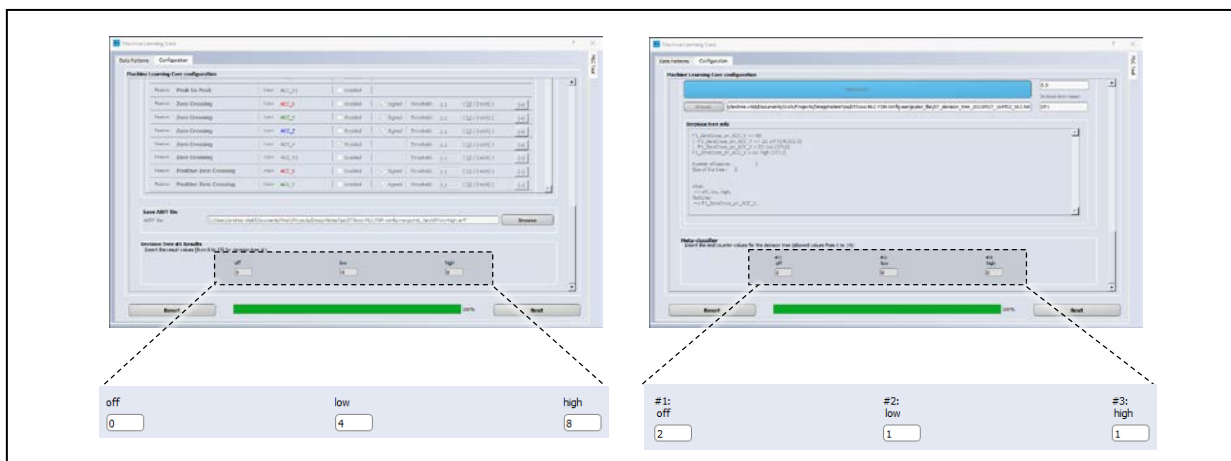
If data logs are available, they can be loaded into Unico-GUI and the decision tree can be re-generated on the fly, the corresponding .txt description will be automatically saved. Otherwise, the .txt description can be loaded.

It is always preferable to load at least one data log for each class, so that feature labels are automatically generated when features are computed and saved to the .arff file. If data logs are not loaded, features cannot be computed, and the tool will ask for feature labels (example: F1\_MEAN\_on\_ACC\_Z, F2\_VAR\_on\_ACC\_V, and so on). Feature labels entered at this point must match those used in the decision tree description which is loaded later in the process.

**If there is only one decision tree and data logs have been loaded and labeled,** Unico-GUI will automatically assign all the labels to that tree and automatically set the corresponding numeric output so that each one is in a separate subgroup. As an example: if there are three classes/labels, the corresponding numeric output will be a multiple of 4 so that they are in different groups. See Figure 9.

- Label “off” is associated with output code 0 in group 1 (threshold = 2 in Figure 9)
- Label “low” is associated with output code 4 in group 2 (threshold = 1 in Figure 9)
- Label “high” is associated with output code 8 in group 3 (threshold = 1 in Figure 9)

**Figure 9. Left: numeric output codes, Right: meta-classifier counter thresholds**



**If data logs have not been loaded and labeled, or if there are two or more decision trees,** Unico-GUI will ask the designer to specify the list of labels for each tree; the corresponding numeric output is the position index (starting from 0); dummy labels can be used to assign a different numeric output to a given label.

As an example: the first decision tree has “off” and “on” labels, the second has “normal” and “abnormal”.

- No dummy labels (see Figure 10)
  - List of labels for first tree: “on; off”
    - Label “on” is associated with output code 0 in group 1
    - Label “off” is associated with output code 1 in group 1
    - Group 1 for the first decision tree has threshold = 2 in Figure 10
  - List of labels for second tree: “normal; abnormal”
    - Label “normal” is associated with output code 0 in group 1
    - Label “abnormal” is associated with output code 1 in group 1
    - Group 1 for the second decision tree has threshold = 1 in Figure 10
- Dummy labels (see Figure 11)
  - List of labels for first decision tree: “off; x; x; x; on”
    - Label “off” is associated with output code 0 in group 1 (threshold = 2 in Figure 11)
    - Label “on” is associated with output code 4 in group 2 (threshold = 2 in Figure 11)
  - List of labels for second decision tree: “normal; x; x; x; abnormal”
    - Label “normal” is associated with code 0 in group 1 (threshold = 1 in Figure 11)
    - Label “abnormal” is associated with code 4 in group 2 (threshold 1 in Figure 11)

Figure 10. Left: MLC label assignment without dummies, Right: meta-classifier counter thresholds

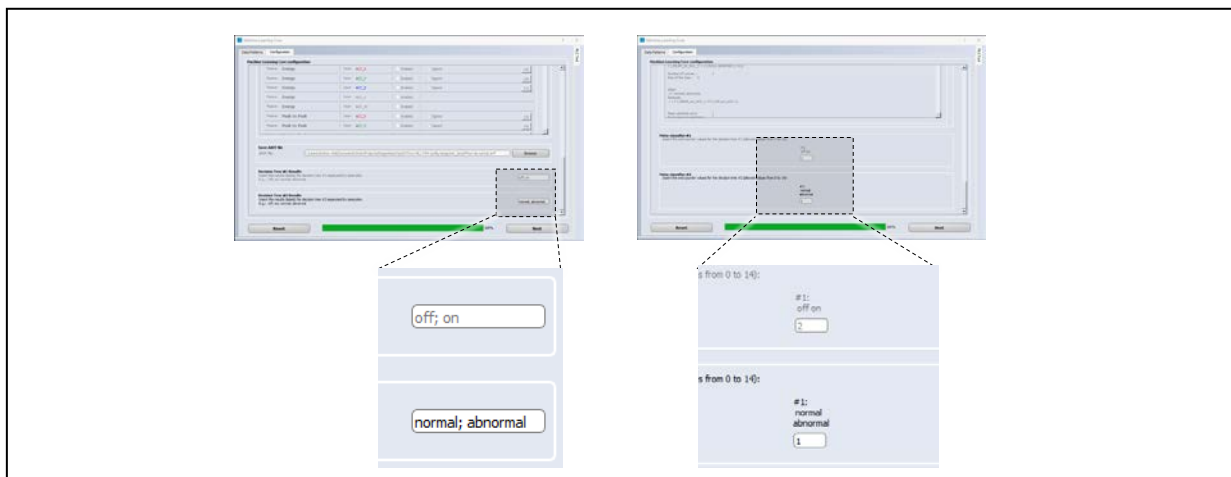
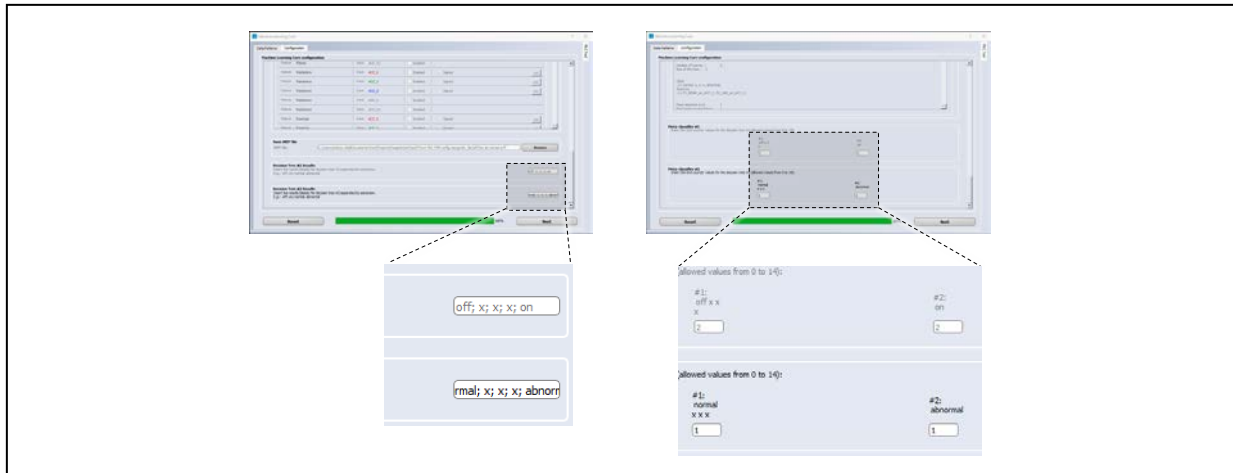




Figure 11. Left: MLC label assignment with dummies, Right: meta-classifier counter thresholds



When all decision trees have been generated or loaded, the designer will be able to browse the file system and save the .ucf file which contains the sensor and MLC configuration (see Figure 3).

### How to merge FSM and MLC configurations

Once the FSM and MLC .ucf configuration files have been generated they can be merged into one .ucf by using the corresponding “Advanced MLC+FSM” button in the “Options” tab in Unico-GUI (see Figure 1).

- The user is prompted to select and load the FSM and the MLC configuration files.
- The tool takes care to map the FSM/MLC programs in the available memory; if the memory is not large enough, the merge fails, and Unico-GUI notifies the user.
- Finally, the user is prompted to select the destination .ucf file where the merged FSM/MLC configuration is saved.

ODR and full-scale settings must be appropriate:

- FSM and MLC can have different output data rates (ODR). They are fed by downsampling the data coming from the sensor. For this reason, **the sensor must have an ODR which is equal to or higher than the largest ODR used by the FSM and MLC**. Downsampling corresponds to sample selection, no filtering, aliasing can happen and can affect the FSM detection and MLC classification performance.
- FSM and MLC take care to rescale sensor data to the appropriate unit of measurement ( $g$  for the accelerometer,  $\text{deg/s}$  (dps) for the gyroscope, Gauss for the magnetometer). However, **the sensor full-scale setting must be set to feed the FSM and MLC with data that has the expected range** otherwise some condition and threshold will never be met. Example: if there is a  $\pm 3 g$  threshold for accelerometer data, then the full-scale must be  $\pm 4 g$  or larger, otherwise that threshold can never be reached. Note that the RMS noise and spurious offset does depend on the full-scale setting and can affect the FSM detection and MLC classification performance.

---

## How to re/configure the target device

The recommended sequence to re/configure the FSM/MLC is the following:

1. Upload the FSM/MLC configuration file: this powers down the sensor, configures the MLC/FSM, and reactivates the sensor.
  - Note that sensor settings are not saved before power-down and the configuration file may not include the necessary sensor settings, this is likely to happen when the configuration file is generated using the “offline” Unico-GUI mode. Example: the interrupt configuration may need to be restored by uploading the sensor configuration.
2. Upload the sensor configuration: this ensures the sensor is configured and activated with the settings needed by the application:
  - Care must be taken to ensure that the **ODR is equal to or higher than the highest ODR used by FSM/MLC**. FSM/MLC are fed by downsampling sensor data, that is, just sample selection, no filtering, aliasing can happen and can affect FSM detection and MLC classification performance.
  - Also, the **full-scale setting must also be equal to or larger than the largest expected by FSM/MLC**. FSM/MLC will rescale sensor data to the appropriate unit of measurement, however RMS noise and spurious offset do depend on the full-scale setting and affect FSM detection and MLC classification performance.

## How to verify the detection performance of FSM

The performance of the finite state machine can only be verified when the sensor is online: its DIL24 adapter must be plugged into the ProfiMEMS evaluation board (STEVAL-MKI109V3), the board must be connected to the laptop via USB, and Unico-GUI must be running with communication with the board enabled.

To verify the performance in real-time (online analysis):

- From the load/save tab in the main Unico-GUI window, load the FSM configuration file, the FSM must be configured to trigger an interrupt on the INT1 or INT2 pin; press Start to activate the sensor, then press the Interrupt button to observe the data and the interrupts in real time.

Alternatively:

- Launch the FSM tool, select the configuration tab, design or import the finite state machine, then write the FSM configuration to the device (see Figure 7).
- Configure the sensor using the Options tab in the main Unico-GUI window, then activate the sensor by pressing Start (see Figure 1).
- Switch back to the FSM tool, select the interrupt tab, then move/tap the device and observe the data and the interrupts generated by the FSM (see Figure 12).

Figure 12. FSM online performance analysis: tap detection has been configured to trigger an interrupt on pin INT1 (see also Figure 7)

Interrupt tab for data injection and debugging

The sensor must be configured and running (press Start on Unico GUI)

Enable communication with STEVAL-MKI109V3 ("ProfiMEMS") and select the target device (here LSM6DSOX)

To verify the performance with data logs (offline analysis):

- Collect data logs: configure the sensor using the Options tab in the main Unico-GUI window, then activate the sensor by pressing Start (see Figure 1); switch to the Load/Save options tab, browse to select the destination file, select the LSB data format (only LSB data must be selected), then press Start/Stop in that tab to start/stop the logging activity. When done, stop logging and stop the sensor.
- Launch the FSM tool, select the configuration tab, design or import the finite state machine; switch to the debug tab, load the data logs by pressing the "Load pattern" button, then use the play/stop buttons to run or step through the FSM code (see Figure 13).

Figure 13. FSM offline performance analysis

Debug tab for data injection and debugging

Load data pattern (log with LSB data) and run/step through the FSM

Enable communication with STEVAL-MKI109V3 ("ProfiMEMS") and select the target device (here LSM6DSOX)

SAMPLE	PP	EP	MASKSEL	IGNED	THRESH	IR_SEL	INT	OUTS	THRESHOLD_1	THRESHOLD_2
60	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
61	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
62	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
63	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
64	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
65	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
66	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
67	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
68	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
69	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
70	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
71	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)
72	0x12	0x12	0	1	0	0	0	00	2066 (0.100)	3000 (1.000)

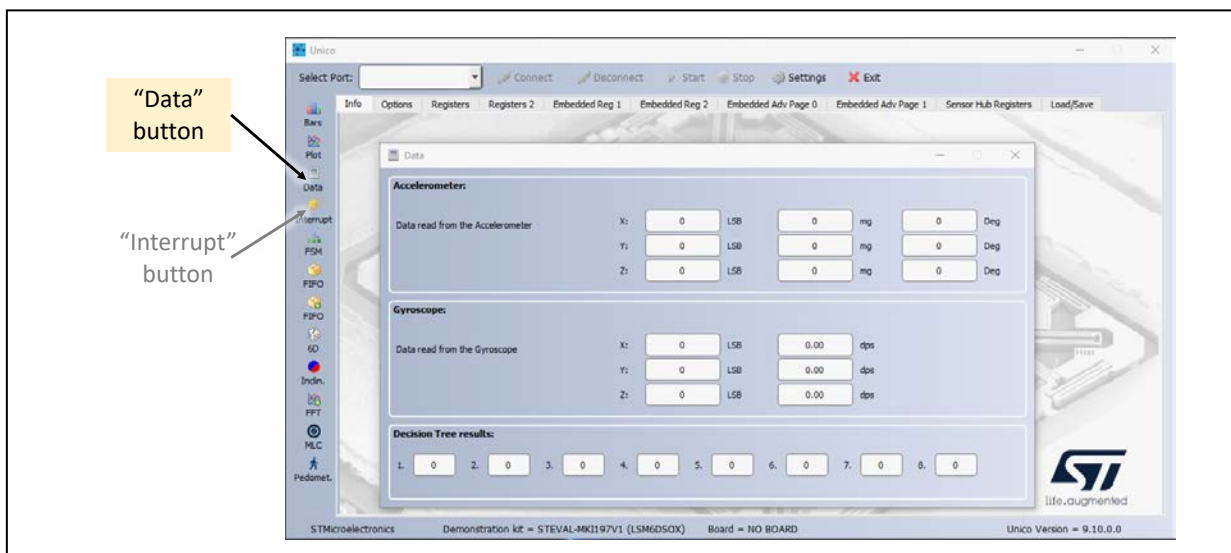
## How to verify the classification performance of the MLC

The performance of the MLC can be verified both in real-time and offline.

To verify the performance in real-time (online analysis):

- The DIL24 adapter of the sensor must be plugged into the ProfiMEMS evaluation board (STEVAL-MK1109V3), the board must be connected to the laptop via USB, Unico-GUI must be running with “Communication with the motherboard” option enabled.
- From the load/save tab in the main Unico-GUI window, load the MLC configuration file, the MLC must be configured to trigger an interrupt on the INT1 or INT2 pin; press Start to activate the sensor, then press the Interrupt button to observe the data and the interrupts in real time, or press the Data button to observe the sensor and the MLC output registers (see Figure 14).

Figure 14. Data window in Unico-GUI



To verify the performance with data logs (offline analysis):

- Collect data logs: configure the sensor using the Options tab in the main Unico-GUI window, then activate the sensor by pressing Start (see Figure 1); switch to the Load/Save options tab, browse to select the destination file, select the desired data (acceleration and/or angular rate), then press Start/Stop in that tab to start/stop the logging activity. When done, stop logging and stop the sensor.

Then use the Python scripts published on the [STMicroelectronics GitHub](https://github.com/STMicroelectronics) site to extract the features from the data logs and run the decision trees. For convenience one can run the MLC Jupyter notebook available at the same link.

---

The notebook covers the following steps:

- Step 1: load and label data logs, set the output classes of all decision trees.
- Step 2: configure filters and features, compute the features, and write the corresponding .arff file, **mlc\_configurator.py** is used in this step.
- Step 3: decision trees are generated, **decision\_tree\_generator.py** is used in this step (this script is based on the Numpy, Scipy, Sklearn, and Pandas packages).
- Step 4: the MLC .ucf configuration file is generated, **mlc\_configurator.py** is used in this step.
- Step 5: decision tree performance is verified by running the decision trees on the computed features saved in the .arff file, **mlc\_test.py** is used in this step.

The scripts `mlc_configurator.py` and `mlc_test.py` are based on Unico-GUI used as a CLI tool, not as a GUI. Unico-GUI executable is run with suitable arguments on the command line. Among the arguments there are input filenames (these are text files generated by the Python scripts: a list of configuration commands to be executed by Unico-GUI), and output filenames (there are text files written by Unico-GUI: the .arff features file, the .ucf configuration file, and the test result file).

## How to build firmware to run the FSM / MLC and other algorithms

This paragraph explains how to quickly build firmware to run the FSM / MLC algorithms as well as other algorithms designed by the user or provided by ST. The recommended tool is **AlgoBuilder GUI**: the processing flow is built by dragging and dropping blocks and connecting them.

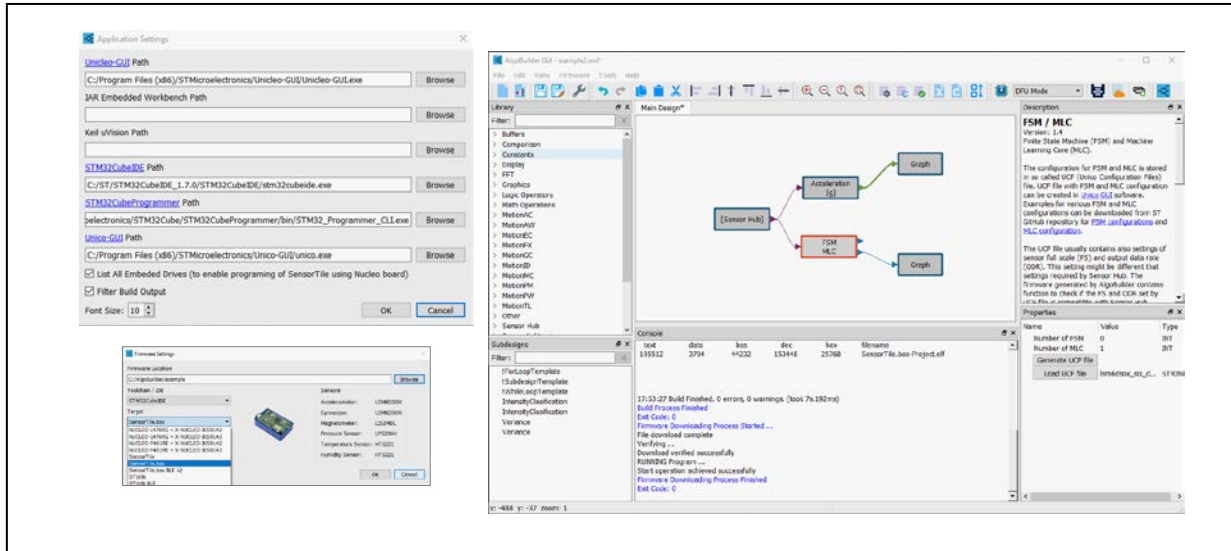
AlgoBuilder GUI uses **STM32CubeIDE** to build the executable **STM32CubeProgrammer** to program the target device, **Unico-GUI** to manage the sensor configuration, and **Unicleo GUI** to display outputs in real-time. After all these programs have been installed, run AlgoBuilder, go to the File menu, select Applications Settings and fill the path for each program as indicated in Figure 15, top left corner. Now AlgoBuilder is ready to be used.

Go to the File menu, select New Design, browse to select a firmware location (a folder that will contain all the project code and the generated binary, it is recommended to use a short path such as `C:\AlgoBuilder\myproject`), then select the target platform. AlgoBuilder currently supports Nucleo L4/F4 paired with X-Nucleo-IKS01A2/A3, SensorTile, SensorTile.Box, STWIN (the last two platforms may be connected to Unicleo GUI via Bluetooth instead of using USB). In this example SensorTile.Box has been selected as shown in Figure 15, bottom left corner.

From the left panel, expand the Sensor Hub item and drag-and-drop the FSM/MLC block and the acceleration block, then from the display panel drag-and-drop two graphs block, the first will be used to display acceleration data, the second to display the MLC output.

AlgoBuilder has also an SD card block which can be used to save the data on the SD card if it is present in the system.

**Figure 15. AlgoBuilder GUI: application settings (top left corner), firmware settings (bottom left corner), main AlgoBuilder window (right side)**



Click on each block and configure it by setting its properties (bottom right corner of the AlgoBuilder window, the top right corner has the description of the block, its inputs, and outputs).

- Click on the Sensor Hub block to configure the output data rate (ODR) and the full-scale setting (in this example: timer clock source, 26 Hz, 2 g, 500 dps).
- Click on the FSM/MLC block and set the number of finite state machines and decision trees in MLC (0 FSM, 1 MLC), then browse to select the corresponding .ucf file (in this example: lsm6dsox\_six\_d\_position.ucf provided with AlgoBuilder, this is designed to identify the 6-tumble orientation of the device).
- Click on the graph that displays acceleration data and select “accelerometer” from the data type pull-down menu.
- Click on the graph that displays the MLC output, select “custom” data type, set the name of the graph and the name of the waveform.
- Connect the Sensor Hub block to the FSM/MLC block and to the acceleration block, then connect those blocks to the respective graph block as shown in Figure 15, right side.

Save the design (it will be saved as an .xml file). From the Firmware menu select “generate C code”. When the code has been generated, from the same menu select “build firmware” and wait for the build to be completed.

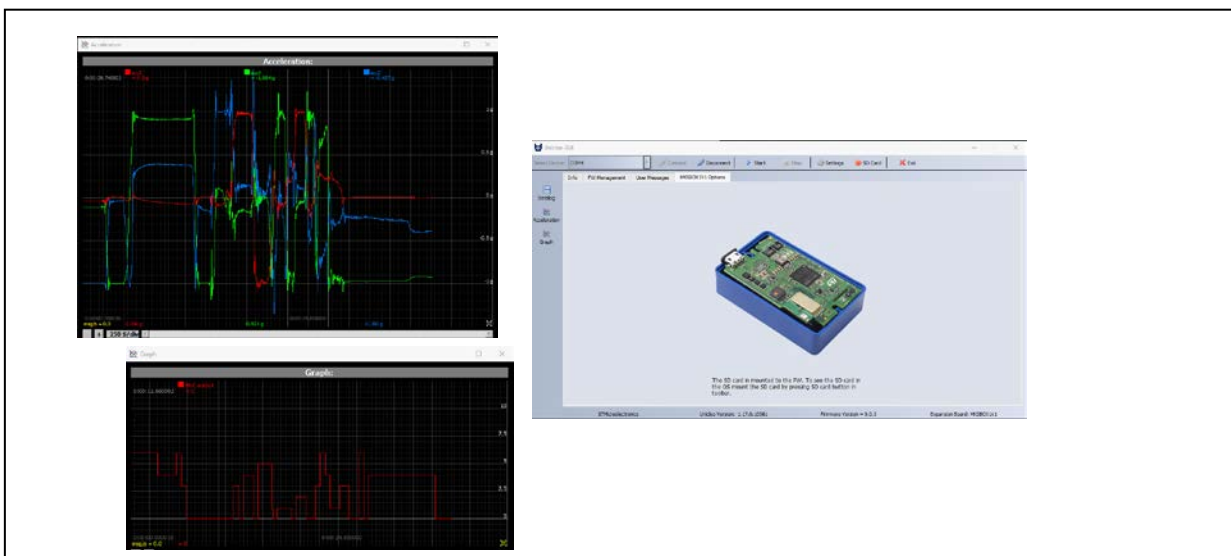
In this example the SensorTile.Box has been selected: disconnect the battery, press and hold both buttons on the board, connect it to the laptop via USB and release the buttons,



this puts the SensorTile.Box in DFU mode (direct firmware upgrade). In AlgoBuilder, go to the Tools menu and select “program target”. Wait for the programming to be completed then close the window. Disconnect and reconnect the SensorTile.Box to reset it.

From the Tools menu select “run Unicleo GUI application”. In Unicleo GUI select the COM port associated with the SensorTile.Box and press connect. If there is an SD card in the SensorTile.Box it must be made available to the firmware: push the SD card button on the top of the window to make the dot red. Then press Start to let the firmware run. On the side there are buttons to open the graphs and display the acceleration and the MLC output. Move the device and observe the output change in real-time, see Figure 16. Note that Unicleo can log all data received from the firmware.

**Figure 16. Unicleo-GUI connected to SensorTile.Box running the AlgoBuilder generated firmware: real-time acceleration graph (top left) and MLC output graph (bottom left)**



## Support material

Documentation
Datasheet DS12814, LSM6DSOX
User manual, UM1049, Unico-GUI
Application note, AN5273, LSM6DSOX Finite State Machine
Application note, AN5259, LSM6DSOX Machine Learning Core
Design Tip, DT0139, Decision tree generation
GitHub MLC examples, <a href="https://github.com/STMicroelectronics/STMems_Machine_Learning_Core">https://github.com/STMicroelectronics/STMems_Machine_Learning_Core</a>



---

## Revision history

Date	Version	Changes
22-Dec-2021	1	Initial release

---

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2021 STMicroelectronics – All rights reserved