life.augmented

AN5273

Application note

LSM6DSOX：有限状态机

引言

本文档旨在提供有关 ST 的 LSM6DSOX 嵌入式有限状态机的使用和配置的信息。

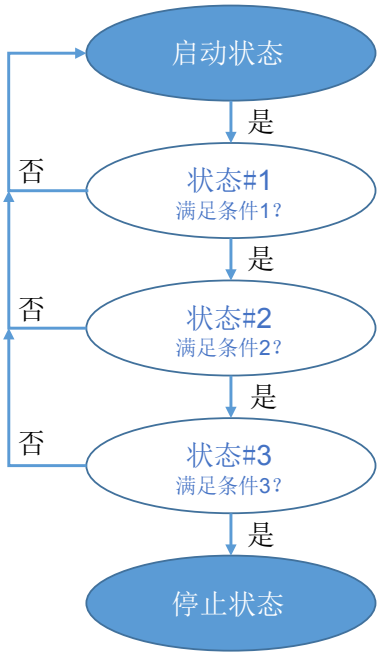
LSM6DSOX 可配置为由用户定义的运动模式激活中断信号生成。为此，最多可以为运动检测独立编程 16 组嵌入式有限状态机。

1 有限状态机(FSM)

1.1 有限状态机定义

有限状态机(FSM)是用于设计逻辑连接的数学抽象。它是由有限数量的状态和状态之间的转换组成的行为模型，类似于流程图，在该流程图中，可在满足特定条件时检查逻辑运行方式。状态机从启动状态开始，通过依赖于输入的转换进入不同状态，最终能以特定状态（被称为停止状态）结束。当前状态取决于系统在过去的状态。下图描述了通用状态机的流程。

图 1. 通用状态机



1.2 中的有限状态机 LSM6DSOX

LSM6DSOX 用作组合式加速度计-陀螺仪传感器，可生成加速度和角速率输出数据；可使用传感器集合功能（模式 2）连接外部传感器（如磁力计）。所有这些数据均可以用作嵌入式有限状态机中最多 16 组程序的输入（请参见下图）。

图 2. 中的状态机 LSM6DSOX

FSM 采用高度模块化的结构：可轻松编写多达 16 组程序，每组程序均可以识别特定手势。

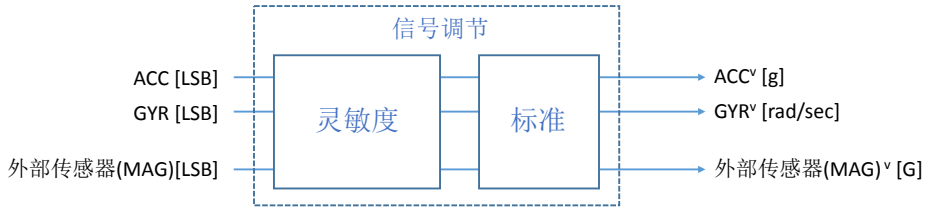
全部 16 组有限状态机都是独立的：每个都有自己的专用存储区，并可以独立执行。在达到结束状态或执行某些特定命令时，将产生中断。通常在识别特定手势时产生中断。

2 信号调节块

信号调节块如下图所示，用作输入传感器数据和 FSM 块之间的接口。使用以下约定单位来转换输出传感器数据（用[LSB]表示）时需要该块：

- 以[*g*]为单位的加速度计数据；
- 以[rad/sec]为单位的陀螺仪数据；
- 外部传感器：如果为磁力计，则必须将数据转换为[G]。

图 3. 信号调节块



该块旨在将灵敏度应用于[LSB]输入数据，然后将这些数据转换为半精度浮点(HFP)格式，并将这些数据传递至 FSM 块。更详细来讲：

- LSM6DSOX 加速度计数据转换系数由设备自动处理；
- LSM6DSOX 陀螺仪数据转换系数由设备自动处理；
- 设备不会自动处理外部传感器数据转换系数：用户必须按以下步骤在设备中正确设置（例如）磁力计转换系数。请注意，磁力计数据必须以[G]为单位转换，并以 HFP 格式表示。

例如：LIS2MDL 磁力计灵敏度为 1.5 mG/LSB → 0.0015 G/LSB → 1624h HFP；这是 LSM6DSOX 设备的默认外部传感器灵敏度值。

对外部磁力计数据应用正确的转换系数的步骤：

1.	将 80h 写入寄存器 01h	// 启用嵌入式功能寄存器访问
2.	将 40h 写入寄存器 17h	// PAGE_RW (17h) = ‘40h’：启用写操作
3.	将 01h 写入寄存器 02h	// PAGE_SEL (02h) = ‘01h’：选择嵌入式高级功能寄存器页面 0
4.	将 BAh 写入寄存器 08h	// PAGE_ADDRESS (08h) = ‘BAh’（MAG_SENSITIVITY_L 地址）
5.	将[LSB]转换系数（以 LIS2MDL 为例，24h）写入寄存器 09h	// 将[LSB]转换系数值写入寄存器 MAG_SENSITIVITY_L (BAh)
6.	将[MSB]转换系数（以 LIS2MDL 为例，16h）写入寄存器 09h	// 将[MSB]转换系数值写入寄存器 MAG_SENSITIVITY_H (BBh)
7.	将 01h 写入寄存器 02h	// PAGE_SEL (02h) = ‘01h’：选择嵌入式高级功能寄存器页面 0
8.	将 00h 写入寄存器 17h	// PAGE_RW (17h) = ‘00h’：禁用读/写操作
9.	将 00h 写入寄存器 01h	// 禁用嵌入式功能寄存器访问

除转换为 HFP 格式以外，信号调节块还计算定义如下的输入数据标准：

$$V = \sqrt{x^2 + y^2 + z^2}$$

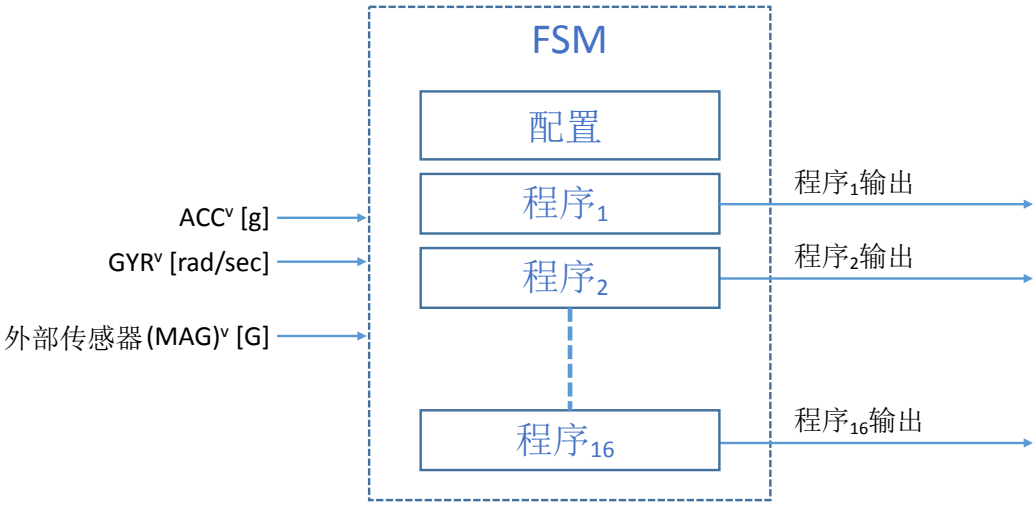
输入数据标准可在状态机程序中使用，以确保为用户提供高水平的程序定制。

3 FSM 块

将来自信号调节块的输出数据信号发送至 FSM 块，具体如下图所示。FSM 块主要包括：

- 一个通用 FSM 配置块：它影响所有程序，并且包括一些必须正确初始化的寄存器，以便配置和定制整个 FSM 模块；
- 最多 16 组可配置程序：每个程序处理输入数据并生成输出。

图 4. FSM 块



以下各节介绍了 FSM 配置和程序块。

3.1 配置块

配置块由 FSM 配置（FSM ODR、中断、程序配置等）所涉及的一组寄存器组成。

嵌入式功能寄存器可用于正确配置 FSM：将 FUNC_CFG_ACCESS (01h)寄存器中的 FUNC_CFG_EN 位置“1”并将 SHUB_REG_ACCESS 位置“0”时，可访问这些寄存器。

LSM6DSOX 器件在嵌入功能寄存器组内部配有更多数量的寄存器，该寄存器被称为高级功能寄存器，并按页划分。必须遵循特定的读/写程序才能访问嵌入式功能寄存器。此特定程序所涉及的寄存器如下：

- PAGE_SEL (02h)：选择所需页；
- PAGE_ADDRESS (08h)：在选定页中选择所需寄存器地址；
- PAGE_VALUE (09h)：设置要写入所选寄存器的值（仅在写入操作中）；
- PAGE_RW (17h)：用于选择读/写操作。

以下脚本显示了在嵌入式功能寄存器组的页码 Z 内的地址为 XXh 的寄存器中写入 YYh 值的通用程序：

1. 将 80h 写入寄存器 01h // 启用嵌入式功能寄存器访问
2. 将 40h 写入寄存器 17h // PAGE_RW (17h) = ‘40h’：启用写操作
3. 将 Z1h 写入寄存器 02h // PAGE_SEL (02h) = ‘Z1h’：选择嵌入式高级功能寄存器页面 Z
4. 将 XXh 写入寄存器 08h // PAGE_ADDRESS (08h) = ‘XXh’：XXh 是要配置的寄存器的地址
5. 将 YYh 写入寄存器 09h // PAGE_VALUE (09h) = ‘YYh’：YYh 是要写入的值
6. 将 01h 写入寄存器 02h // PAGE_SEL (02h) = ‘01h’：选择嵌入式高级功能寄存器页面 0。这对于设备的正确操作是必需的。
7. 将 00h 写入寄存器 17h // PAGE_RW (17h) = ‘00h’：禁用读/写操作
8. 将 00h 写入寄存器 01h // 禁用嵌入式功能寄存器访问

注：在写操作后，PAGE_ADDRESS (08h)寄存器会自动递增。

必须从 FSM_START_ADD_L (7Eh)和 FSM_START_ADD_H (7Fh)寄存器指示的寄存器地址开始，将程序配置写入嵌入式高级功能寄存器中。必须将所有程序写入连续寄存器中，包括两个重要方面：

- 从一个页面移至另一个页面时（即从页面 03h，地址 FFh 转到页面 04h，地址 00h），必须正确更新 PAGE_SEL (02h)寄存器和 PAGE_ADDRESS (08h)寄存器。LSM6DSOX 提供可通过 PAGE_SEL (02h)寄存器进行寻址的 8 个页面。要对最后一页进行寻址，必须将 PAGE_SEL (02h)设为 71h；
- 程序 SIZE 字节必须为偶数：如果为奇数，则必须在指令部分的末尾添加一个附加的 STOP 状态。

有关如何配置整个 FSM 的详细示例，请参阅第 8 节 FSM 配置示例。

3.1.1 FSM 寄存器

下表提供了与 FSM 和相应地址有关的寄存器列表。

表 1. FSM 寄存器

寄存器名	类型	地址	位 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EMB_FUNC_STATUS _MAINPAGE	r	35h	IS_FSM_LC	0	-	-	-	0	0	0
FSM_STATUS_A _MAINPAGE	r	36h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
FSM_STATUS_B _MAINPAGE	r	37h	IS_FSM16	IS_FSM15	IS_FSM14	IS_FSM13	IS_FSM12	IS_FSM11	IS_FSM10	IS_FSM9

3.1.1.1 EMB_FUNC_STATUS_MAINPAGE (35h)

EMB_FUNC_STATUS_MAINPAGE (35h)寄存器包含关于长计数器的中断状态信息。

表 2. EMB_FUNC_STATUS_MAINPAGE (35h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM_LC	0	-	-	-	0	0	0

当 FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)嵌入功能寄存器中的当前长计数器值等于 FSM_LC_TIMEOUT_L (7Ah)和 FSM_LC_ TIMEOUT_H (7Bh)寄存器中配置的长计数器超时值时，IS_FSM_LC 位置自动置“1”.

3.1.1.2 FSM_STATUS_A_MAINPAGE（36h）

FSM_STATUS_A_MAINPAGE (36h)寄存器包含有关程序 1-8 的中断状态信息。

表 3. FSM_STATUS_A_MAINPAGE（36h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1

在 FSM 程序 x 中执行 OUTC / CONT / CONTREL 命令时，将 IS_FSMx 位置“1”。有关这些命令的更多详细信息，请参见专用章节/段落。

3.1.1.3 FSM_STATUS_B_MAINPAGE（37h）

FSM_STATUS_B_MAINPAGE (37h)寄存器包含有关程序 9-16 的中断状态信息。

表 4. FSM_STATUS_B_MAINPAGE（37h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM16	IS_FSM15	IS_FSM14	IS_FSM13	IS_FSM12	IS_FSM11	IS_FSM10	IS_FSM9

在 FSM 程序 x 中执行 OUTC / CONT / CONTREL 命令时，将 IS_FSMx 位置“1”。有关这些命令的更多详细信息，请参见专用章节/段落。

3.1.2 FSM 嵌入式功能寄存器

表 5. 嵌入功能寄存器

寄存器名	类型	地址	位 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EMB_FUNC_EN_B	r/w	05h	0(1)	0(1)	0(1)	-	-	0(1)	0(1)	FSM_EN
EMB_FUNC_INT1	r/w	0Ah	INT1_FSM _LC(2)	01	-	-	-	01	01	01
FSM_INT1_A	r/w	0Bh	INT1_FSM8(2)	INT1_FSM7(2)	INT1_FSM6(2)	INT1_FSM5(2)	INT1_FSM4(2)	INT1_FSM3(2)	INT1_FSM2(2)	INT1_FSM1(2)
FSM_INT1_B	r/w	0Ch	INT1_FSM16(2)	INT1_FSM15(2)	INT1_FSM14(2)	INT1_FSM13(2)	INT1_FSM12(2)	INT1_FSM11(2)	INT1_FSM10(2)	INT1_FSM9(2)
EMB_FUNC_INT2	r/w	0Eh	INT2_FSM_LC(3)	0(1)	-	-	-	0(1)	0(1)	0(1)
FSM_INT2_A	r/w	0Fh	INT2_FSM8(3)	INT2_FSM7(3)	INT2_FSM6(3)	INT2_FSM5(3)	INT2_FSM4(3)	INT2_FSM3(3)	INT2_FSM2(3)	INT2_FSM1(3)
FSM_INT2_B	r/w	10h	INT2_FSM16(3)	INT2_FSM15(3)	INT2_FSM14(3)	INT2_FSM13(3)	INT2_FSM12(3)	INT2_FSM11(3)	INT2_FSM10(3)	INT2_FSM9(3)
EMB_FUNC_STATUS	r	12h	IS_FSM_LC	0	-	-	-	0	0	0
FSM_STATUS_A	r	13h	IS_FSM_8	IS_FSM_7	IS_FSM_6	IS_FSM_5	IS_FSM_4	IS_FSM_3	IS_FSM_2	IS_FSM_1
FSM_STATUS_B	r	14h	IS_FSM_16	IS_FSM_15	IS_FSM_14	IS_FSM_13	IS_FSM_12	IS_FSM_11	IS_FSM_10	IS_FSM_9
PAGE_RW	r/w	17h	EMB_FUNC_LIR	-	-	0	0	0	0	0
FSM_ENABLE_A	r/w	46h	FSM8_EN	FSM7_EN	FSM6_EN	FSM5_EN	FSM4_EN	FSM3_EN	FSM2_EN	FSM1_EN
FSM_ENABLE_B	r/w	47h	FSM16_EN	FSM15_EN	FSM14_EN	FSM13_EN	FSM12_EN	FSM11_EN	FSM10_EN	FSM9_EN
FSM_LONG_COUNTER_L	r	48h	FSM_LC7	FSM_LC6	FSM_LC5	FSM_LC4	FSM_LC3	FSM_LC2	FSM_LC1	FSM_LC0
FSM_LONG_COUNTER_H	r	49h	FSM_LC15	FSM_LC14	FSM_LC13	FSM_LC12	FSM_LC11	FSM_LC10	FSM_LC9	FSM_LC8
FSM_LONG_COUNTER_CLEAR	r/w	4Ah	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	FSM_LC _CLEARED(4)	FSM_LC _CLEAR
FSM_OUTS1	r	4Ch	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS2	r	4Dh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS3	r	4Eh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS4	r	4Fh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS5	r	50h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS6	r	51h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS7	r	52h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS8	r	53h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS9	r	54h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS10	r	55h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS11	r	56h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS12	r	57h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS13	r	58h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS14	r	59h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS15	r	5Ah	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS16	r	5Bh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
EMB_FUNC_ODR_CFG_B	r/w	5Fh	0(1)	1(5)	0(1)	FSM_ODR1	FSM_ODR0	0(1)	1(5)	1(5)
FSM_INIT	r/w	67h	0(1)	0(1)	0(1)	0(1)	-	0(1)	0(1)	FSM_INIT

1. *为了设备的正确运行，此位必须置为“0”。*
2. *如果将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置“1”，则此位有效。*
3. *如果将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置“1”，则此位有效。*
4. *只读位。*
5. *为了设备的正确运行，此位必须置为“1”。*

3.1.2.1 *EMB_FUNC_EN_B (05h)*

EMB_FUNC_EN_B (05h)寄存器用于启用 FSM 嵌入式功能。

表 6. EMB_FUNC_EN_B (05h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	-	-	0	0	FSM_EN

FSM_EN 位用于启用 FSM。当此位被置“1”时，所有启用的 FSM 程序均开始执行。

3.1.2.2 *EMB_FUNC_INT1（0Ah）*

EMB_FUNC_INT1 (0Ah)寄存器用于向 INT1 引脚发送 FSM 长计数器中断：将 INT1_FSM_LC 位置“1”，以使能发送。

表 7. EMB_FUNC_INT1 (0Ah)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT1_ FSM_LC	0	-	-	-	0	0	0

如果将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置“1”，则 INT1_FSM_LC 位有效。

3.1.2.3 *FSM_INT1_A（0Bh）*

FSM_INT1_A (0Bh)寄存器用于向 INT1 引脚发送 FSM 程序 1-8 中断。

表 8. FSM_INT1_A (0Bh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT1_FSM8	INT1_FSM7	INT1_FSM6	INT1_FSM5	INT1_FSM4	INT1_FSM3	INT1_FSM2	INT1_FSM1

如果将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置“1”，则这些位有效。
该寄存器的每个位使信号可在 INT1 上传输。引脚输出将提供所选信号的或组合。

3.1.2.4 *FSM_INT1_B（0Ch）*

FSM_INT1_B (0Ch)寄存器用于向 INT1 引脚发送 FSM 程序 9-16 中断。

表 9. FSM_INT1_B (0Ch)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT1_FSM16	INT1_FSM15	INT1_FSM14	INT1_FSM13	INT1_FSM12	INT1_FSM11	INT1_FSM10	INT1_FSM9

如果将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置“1”，则这些位有效。
该寄存器的每个位使信号可在 INT1 上传输。引脚输出将提供所选信号的或组合。

3.1.2.5 *EMB_FUNC_INT2（0Eh）*

EMB_FUNC_INT2 (0Eh)寄存器用于向 INT2 引脚发送 FSM 长计数器中断：将 INT2_FSM_LC 位置“1”，以使能发送。

表 10. **EMB_FUNC_INT2 (0Eh)寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT2_ FSM_LC	0	-	-	-	0	0	0

如果将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置“1”，则这些位有效。

3.1.2.6 *FSM_INT2_A（0Fh）*

FSM_INT2_A (0Fh)寄存器用于向 INT2 引脚发送 FSM 程序 1-8 中断。

表 11. **FSM_INT2_A (0Fh)寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT2_FSM8	INT2_FSM7	INT2_FSM6	INT2_FSM5	INT2_FSM4	INT2_FSM3	INT2_FSM2	INT2_FSM1

如果将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置“1”，则这些位有效。
该寄存器的每个位使信号可在 INT2 上传输。引脚输出将提供所选信号的或组合。

3.1.2.7 *FSM_INT2_B（10h）*

FSM_INT2_B (10h)寄存器用于向 INT2 引脚发送 FSM 程序 9-16 中断。

表 12. **FSM_INT2_B (10h)寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INT2_FSM16	INT2_FSM15	INT2_FSM14	INT2_FSM13	INT2_FSM12	INT2_FSM11	INT2_FSM10	INT2_FSM9

如果将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置“1”，则这些位有效。
该寄存器的每个位使信号可在 INT2 上传输。引脚输出将提供所选信号的或组合。

3.1.2.8 *EMB_FUNC_STATUS（12h）*

EMB_FUNC_STATUS (12h)寄存器包含关于长计数器的中断状态信息。

表 13. EMB_FUNC_STATUS (12h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM_LC	0	-	-	-	0	0	0

当 FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器中的当前长计数器值等于 FSM_LC_TIMEOUT_L (7Ah)和 FSM_LC_ TIMEOUT_H (7Bh)寄存器中配置的长计数器超时值时，IS_FSM_LC 位置自动置“1”.

3.1.2.9 *FSM_STATUS_A（13h）*

FSM_STATUS_A (13h)寄存器包含有关程序 1-8 的中断状态信息。

表 14. FSM_STATUS_A（13h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1

在 FSM 程序 x 中执行 OUTC / CONT / CONTREL 命令时，将 IS_FSMx 位置“1”。有关这些命令的更多详细信息，请参见专用章节/段落。

3.1.2.10 *FSM_STATUS_B（14h）*

FSM_STATUS_B (14h)寄存器包含有关程序 9-16 的中断状态信息。

表 15. FSM_STATUS_B（14h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IS_FSM16	IS_FSM15	IS_FSM14	IS_FSM13	IS_FSM12	IS_FSM11	IS_FSM10	IS_FSM9

在 FSM 程序 x 中执行 OUTC / CONT / CONTREL 命令时，将 IS_FSMx 位置“1”。有关这些命令的更多详细信息，请参见专用章节/段落。

3.1.2.11 *PAGE_RW（17h）*

PAGE_RW (17h)寄存器用于将 FSM 中断从脉冲（默认）更改为锁存。

表 16. PAGE_RW (17h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EMB_ FUNC_LIR	-	-	0	0	0	0	0

3.1.2.12 *FSM_ENABLE_A（46h）*

FSM_ENABLE_A (46h)寄存器用于使能 FSM 的程序 1-8。

表 17. FSM_ENABLE_A（46h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM8_EN	FSM7_EN	FSM6_EN	FSM5_EN	FSM4_EN	FSM3_EN	FSM2_EN	FSM1_EN

3.1.2.13 *FSM_ENABLE_B（47h）*

FSM_ENABLE_B (47h)寄存器用于使能 FSM 的程序 9-16。

表 18. FSM_ENABLE_B（47h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM16_EN	FSM15_EN	FSM14_EN	FSM13_EN	FSM12_EN	FSM11_EN	FSM10_EN	FSM9_EN

3.1.2.14 *FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)*

FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器用于读/写长计数器值。有关如何访问这些寄存器的信息，请参见第 3.1 节 配置块。

表 19. FSM_LONG_COUNTER_L（48h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC7	FSM_LC6	FSM_LC5	FSM_LC4	FSM_LC3	FSM_LC2	FSM_LC1	FSM_LC0

表 20. FSM_LONG_COUNTER_H（49h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC15	FSM_LC14	FSM_LC13	FSM_LC12	FSM_LC11	FSM_LC10	FSM_LC9	FSM_LC8

3.1.2.15 *FSM_LONG_COUNTER_CLEAR（4Ah）*

FSM_LONG_COUNTER_CLEAR (4Ah)寄存器用于重置 FSM 长计数器值。

表 21. FSM_LONG_COUNTER_CLEAR（4Ah）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	0	0	0	FSM_LC_ CLEARED(1)	FSM_LC_ CLEAR

1. *只读位。*

将 FSM_LC_CLEAR 位置“1”，用以在下次执行 INCR 命令时重置 FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器的值。长计数器复位完成后，自动将 FSM_LC_CLEARED 位置“1”。请参见第 5.1 节 长计数器。

3.1.2.16 *FSM_OUTS[1:16]（4Ch - 5Bh）*

FSM[1:16]输出寄存器。

表 22. **FSM_OUTS[1:16]（4Ch - 5Bh）寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V

这些寄存器为只读寄存器，每组状态机各一个，包含执行 OUTC / CONT / CONTREL 命令时更新的当前有效临时掩码值。

3.1.2.17 *EMB_FUNC_ODR_CFG_B（5Fh）*

EMB_FUNC_ODR_CFG_B (5Fh)寄存器用于配置 FSM 的 ODR（FSM_ODR[1:0]位）。

表 23. **EMB_FUNC_ODR_CFG_B (5Fh)寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	1	0	FSM_ODR1	FSM_ODR0	0	1	1

所有程序均以该配置的速率执行。有关如何以不同数据速率运行程序的信息，请参见第 6 节 可变数据部分中的第 6.6 节 抽取器。

下表列出了可能的 ODR 配置。

表 24. **FSM 输出数据速率**

FSM_ODR[1:0]	ODR [Hz]
00	12.5
01	26
10	52
11	104

注：FSM ODR 在内部被限制为加速度计和陀螺仪之间的最高传感器数据速率。建议将加速度计和/或陀螺仪的数据速率设为高于或等于已配置的 FSM ODR。

3.1.2.18 *FSM_INIT（67h）*

FSM_INIT (67h)寄存器用于将 FSM 程序重置为其默认配置。

表 25. FSM_INIT（67h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	0	-	0	0	FSM_INIT

FSM_INIT 位用于触发新的“启动例程”请求。当该位被置“1”时，设备执行启动例程，如第 9 节 启动例程中所述。启动例程完成后，自动将 FSM_INIT 位置“0”。

此外，将 EMB_FUNC_EN_B (05h)寄存器的 FSM_EN 位置“0”时（并在启动例程完成时重置为“0”）时，该位自动变为“1”。

3.1.3 FSM 嵌入式高级功能寄存器

下表提供了与 FSM 相关的嵌入式高级功能页面 0 和 1 的寄存器列表。通过配置 *PAGE_SEL (02h)*中的 PAGE_SEL[3:0]位可访问这些寄存器。

表 26. **FSM 嵌入式高级功能寄存器**

寄存器名	页	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SENSITIVITY_L	0	BAh	MAG_SENS_L7	MAG_SENS_L6	MAG_SENS_L5	MAG_SENS_L4	MAG_SENS_L3	MAG_SENS_L2	MAG_SENS_L1	MAG_SENS_L0
MAG_SENSITIVITY_H	0	BBh	MAG_SENS_H7	MAG_SENS_H6	MAG_SENS_H5	MAG_SENS_H4	MAG_SENS_H3	MAG_SENS_H2	MAG_SENS_H1	MAG_SENS_H0
MAG_OFFX_L	0	C0h	MAG_OFFX_L7	MAG_OFFX_L6	MAG_OFFX_L5	MAG_OFFX_L4	MAG_OFFX_L3	MAG_OFFX_L2	MAG_OFFX_L1	MAG_OFFX_L0
MAG_OFFX_H	0	C1h	MAG_OFFX_H7	MAG_OFFX_H6	MAG_OFFX_H5	MAG_OFFX_H4	MAG_OFFX_H3	MAG_OFFX_H2	MAG_OFFX_H1	MAG_OFFX_H0
MAG_OFFY_L	0	C2h	MAG_OFFY_L7	MAG_OFFY_L6	MAG_OFFY_L5	MAG_OFFY_L4	MAG_OFFY_L3	MAG_OFFY_L2	MAG_OFFY_L1	MAG_OFFY_L0
MAG_OFFY_H	0	C3h	MAG_OFFY_H7	MAG_OFFY_H6	MAG_OFFY_H5	MAG_OFFY_H4	MAG_OFFY_H3	MAG_OFFY_H2	MAG_OFFY_H1	MAG_OFFY_H0
MAG_OFFZ_L	0	C4h	MAG_OFFZ_L7	MAG_OFFZ_L6	MAG_OFFZ_L5	MAG_OFFZ_L4	MAG_OFFZ_L3	MAG_OFFZ_L2	MAG_OFFZ_L1	MAG_OFFZ_L0
MAG_OFFZ_H	0	C5h	MAG_OFFZ_H7	MAG_OFFZ_H6	MAG_OFFZ_H5	MAG_OFFZ_H4	MAG_OFFZ_H3	MAG_OFFZ_H2	MAG_OFFZ_H1	MAG_OFFZ_H0
MAG_SI_XX_L	0	C6h	MAG_SI_XX_L7	MAG_SI_XX_L6	MAG_SI_XX_L5	MAG_SI_XX_L4	MAG_SI_XX_L3	MAG_SI_XX_L2	MAG_SI_XX_L1	MAG_SI_XX_L0
MAG_SI_XX_H	0	C7h	MAG_SI_XX_H7	MAG_SI_XX_H6	MAG_SI_XX_H5	MAG_SI_XX_H4	MAG_SI_XX_H3	MAG_SI_XX_H2	MAG_SI_XX_H1	MAG_SI_XX_H0
MAG_SI_XY_L	0	C8h	MAG_SI_XY_L7	MAG_SI_XY_L6	MAG_SI_XY_L5	MAG_SI_XY_L4	MAG_SI_XY_L3	MAG_SI_XY_L2	MAG_SI_XY_L1	MAG_SI_XY_L0
MAG_SI_XY_H	0	C9h	MAG_SI_XY_H7	MAG_SI_XY_H6	MAG_SI_XY_H5	MAG_SI_XY_H4	MAG_SI_XY_H3	MAG_SI_XY_H2	MAG_SI_XY_H1	MAG_SI_XY_H0
MAG_SI_XZ_L	0	CAh	MAG_SI_XZ_L7	MAG_SI_XZ_L6	MAG_SI_XZ_L5	MAG_SI_XZ_L4	MAG_SI_XZ_L3	MAG_SI_XZ_L2	MAG_SI_XZ_L1	MAG_SI_XZ_L0
MAG_SI_XZ_H	0	CBh	MAG_SI_XZ_H7	MAG_SI_XZ_H6	MAG_SI_XZ_H5	MAG_SI_XZ_H4	MAG_SI_XZ_H3	MAG_SI_XZ_H2	MAG_SI_XZ_H1	MAG_SI_XZ_H0
MAG_SI_YY_L	0	CCh	MAG_SI_YY_L7	MAG_SI_YY_L6	MAG_SI_YY_L5	MAG_SI_YY_L4	MAG_SI_YY_L3	MAG_SI_YY_L2	MAG_SI_YY_L1	MAG_SI_YY_L0
MAG_SI_YY_H	0	CDh	MAG_SI_YY_H7	MAG_SI_YY_H6	MAG_SI_YY_H5	MAG_SI_YY_H4	MAG_SI_YY_H3	MAG_SI_YY_H2	MAG_SI_YY_H1	MAG_SI_YY_H0
MAG_SI_YZ_L	0	CEh	MAG_SI_YZ_L7	MAG_SI_YZ_L6	MAG_SI_YZ_L5	MAG_SI_YZ_L4	MAG_SI_YZ_L3	MAG_SI_YZ_L2	MAG_SI_YZ_L1	MAG_SI_YZ_L0
MAG_SI_YZ_H	0	CFh	MAG_SI_YZ_H7	MAG_SI_YZ_H6	MAG_SI_YZ_H5	MAG_SI_YZ_H4	MAG_SI_YZ_H3	MAG_SI_YZ_H2	MAG_SI_YZ_H1	MAG_SI_YZ_H0
MAG_SI_ZZ_L	0	D0h	MAG_SI_ZZ_L7	MAG_SI_ZZ_L6	MAG_SI_ZZ_L5	MAG_SI_ZZ_L4	MAG_SI_ZZ_L3	MAG_SI_ZZ_L2	MAG_SI_ZZ_L1	MAG_SI_ZZ_L0
MAG_SI_ZZ_H	0	D1h	MAG_SI_ZZ_H7	MAG_SI_ZZ_H6	MAG_SI_ZZ_H5	MAG_SI_ZZ_H4	MAG_SI_ZZ_H3	MAG_SI_ZZ_H2	MAG_SI_ZZ_H1	MAG_SI_ZZ_H0
FSM_LC_TIMEOUT_L	1	7Ah	FSM_LC_TIMEOUT 7	FSM_LC_TIMEOUT 6	FSM_LC_TIMEOUT 5	FSM_LC_TIMEOUT 4	FSM_LC_TIMEOUT 3	FSM_LC_TIMEOUT 2	FSM_LC_TIMEOUT 1	FSM_LC_TIMEOUT 0
FSM_LC_TIMEOUT_H	1	7Bh	FSM_LC_TIMEOUT 15	FSM_LC_TIMEOUT 14	FSM_LC_TIMEOUT 13	FSM_LC_TIMEOUT 12	FSM_LC_TIMEOUT 11	FSM_LC_TIMEOUT 10	FSM_LC_TIMEOUT 9	FSM_LC_TIMEOUT 8
FSM_PROGRAMS	1	7Ch	FSM_N_PROG7	FSM_N_PROG6	FSM_N_PROG5	FSM_N_PROG4	FSM_N_PROG3	FSM_N_PROG2	FSM_N_PROG1	FSM_N_PROG0
FSM_START_ADD_L	1	7Eh	FSM_START7	FSM_START6	FSM_START5	FSM_START4	FSM_START3	FSM_START2	FSM_START1	FSM_START0
FSM_START_ADD_H	1	7Fh	FSM_START15	FSM_START14	FSM_START13	FSM_START12	FSM_START11	FSM_START10	FSM_START9	FSM_START8

3.1.3.1 *MAG_SENSITIVITY_L (BAh)和MAG_SENSITIVITY_H (BBh)*

外部磁力计灵敏度寄存器(r/w)。

表 27. **MAG_SENSITIVITY_L（BAh）寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SENS_L7	MAG_SENS_L6	MAG_SENS_L5	MAG_SENS_L4	MAG_SENS_L3	MAG_SENS_L2	MAG_SENS_L1	MAG_SENS_L0

表 28. **MAG_SENSITIVITY_H（BBh）寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SENS_H7	MAG_SENS_H6	MAG_SENS_H5	MAG_SENS_H4	MAG_SENS_H3	MAG_SENS_H2	MAG_SENS_H1	MAG_SENS_H0

该寄存器对应于外部磁力计传感器的 LSB 至高斯转换值。该寄存器值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。MAG_SENS[15:0]的默认值为 0x1624，对应于 0.0015 高斯/LSB（LIS2MDL 磁力计灵敏度）。

3.1.3.2 *MAG_OFFX_L (C0h)和MAG_OFFX_H (C1h)*

X 轴硬铁补偿寄存器(r/w)的偏移量。

表 29. **MAG_OFFX_L（C0h）寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFX_L7	MAG_OFFX_L6	MAG_OFFX_L5	MAG_OFFX_L4	MAG_OFFX_L3	MAG_OFFX_L2	MAG_OFFX_L1	MAG_OFFX_L0

表 30. **MAG_OFFX_H（C1h）寄存器**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFX_H7	MAG_OFFX_H6	MAG_OFFX_H5	MAG_OFFX_H4	MAG_OFFX_H3	MAG_OFFX_H2	MAG_OFFX_H1	MAG_OFFX_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.3 *MAG_OFFY_L (C2h)和MAG_OFFY_H (C3h)*

Y 轴硬铁补偿寄存器(r/w)的偏移量。

表 31. MAG_OFFY_L（C2h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFY_L7	MAG_OFFY_L6	MAG_OFFY_L5	MAG_OFFY_L4	MAG_OFFY_L3	MAG_OFFY_L2	MAG_OFFY_L1	MAG_OFFY_L0

表 32. MAG_OFFY_H（C3h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFY_H7	MAG_OFFY_H6	MAG_OFFY_H5	MAG_OFFY_H4	MAG_OFFY_H3	MAG_OFFY_H2	MAG_OFFY_H1	MAG_OFFY_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.4 *MAG_OFFZ_L (C4h)和MAG_OFFZ_H (C5h)*

Z 轴硬铁补偿寄存器(r/w)的偏移量。

表 33. MAG_OFFZ_L（C4h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFZ_L7	MAG_OFFZ_L6	MAG_OFFZ_L5	MAG_OFFZ_L4	MAG_OFFZ_L3	MAG_OFFZ_L2	MAG_OFFZ_L1	MAG_OFFZ_L0

表 34. MAG_OFFZ_H（C5h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_OFFZ_H7	MAG_OFFZ_H6	MAG_OFFZ_H5	MAG_OFFZ_H4	MAG_OFFZ_H3	MAG_OFFZ_H2	MAG_OFFZ_H1	MAG_OFFZ_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.5 *MAG_SI_XX_L (C6h)和MAG_SI_XX_H (C7h)*

软铁（3x3 对称）矩阵 row1 col1 校正寄存器(r/w)。

表 35. MAG_SI_XX_L（C6h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XX_L7	MAG_SI_XX_L6	MAG_SI_XX_L5	MAG_SI_XX_L4	MAG_SI_XX_L3	MAG_SI_XX_L2	MAG_SI_XX_L1	MAG_SI_XX_L0

表 36. MAG_SI_XX_H（C7h）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XX_H7	MAG_SI_XX_H6	MAG_SI_XX_H5	MAG_SI_XX_H4	MAG_SI_XX_H3	MAG_SI_XX_H2	MAG_SI_XX_H1	MAG_SI_XX_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.6 *MAG_SI_XY_L (C8h)和MAG_SI_XY_H (C9h)*

软铁（3x3 对称）矩阵 row1 col2（和 row2 col1）校正寄存器(r/w)。

表 37. MAG_SI_XY_L (C8h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XY_L7	MAG_SI_XY_L6	MAG_SI_XY_L5	MAG_SI_XY_L4	MAG_SI_XY_L3	MAG_SI_XY_L2	MAG_SI_XY_L1	MAG_SI_XY_L0

表 38. MAG_SI_XY_H (C9h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XY_H7	MAG_SI_XY_H6	MAG_SI_XY_H5	MAG_SI_XY_H4	MAG_SI_XY_H3	MAG_SI_XY_H2	MAG_SI_XY_H1	MAG_SI_XY_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.7 *MAG_SI_XZ_L (CAh)和MAG_SI_XZ_H (CBh)*

软铁（3x3 对称）矩阵 row1 col3（和 row3 col1）校正寄存器(r/w)。

表 39. MAG_SI_XZ_L (CAh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XZ_L7	MAG_SI_XZ_L6	MAG_SI_XZ_L5	MAG_SI_XZ_L4	MAG_SI_XZ_L3	MAG_SI_XZ_L2	MAG_SI_XZ_L1	MAG_SI_XZ_L0

表 40. MAG_SI_XZ_H (CBh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_XZ_H7	MAG_SI_XZ_H6	MAG_SI_XZ_H5	MAG_SI_XZ_H4	MAG_SI_XZ_H3	MAG_SI_XZ_H2	MAG_SI_XZ_H1	MAG_SI_XZ_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.8 *MAG_SI_YY_L (CCh)和MAG_SI_YY_H (CDh)*

软铁（3x3 对称）矩阵 row2 col2 校正寄存器(r/w)。

表 41. MAG_SI_YY_L (CCh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_YY_L7	MAG_SI_YY_L6	MAG_SI_YY_L5	MAG_SI_YY_L4	MAG_SI_YY_L3	MAG_SI_YY_L2	MAG_SI_YY_L1	MAG_SI_YY_L0

表 42. MAG_SI_YY_H (CDh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_YY_H7	MAG_SI_YY_H6	MAG_SI_YY_H5	MAG_SI_YY_H4	MAG_SI_YY_H3	MAG_SI_YY_H2	MAG_SI_YY_H1	MAG_SI_YY_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.9 *MAG_SI_YZ_L (CEh)和MAG_SI_YZ_H (CFh)*

软铁（3x3 对称）矩阵 row2 col3（和 row3 col2）校正寄存器(r/w)。

表 43. MAG_SI_YZ_L (CEh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_YZ_L7	MAG_SI_YZ_L6	MAG_SI_YZ_L5	MAG_SI_YZ_L4	MAG_SI_YZ_L3	MAG_SI_YZ_L2	MAG_SI_YZ_L1	MAG_SI_YZ_L0

表 44. MAG_SI_YZ_H (CFh)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_YZ_H7	MAG_SI_YZ_H6	MAG_SI_YZ_H5	MAG_SI_YZ_H4	MAG_SI_YZ_H3	MAG_SI_YZ_H2	MAG_SI_YZ_H1	MAG_SI_YZ_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.10 *MAG_SI_ZZ_L (D0h)和MAG_SI_ZZ_H (D1h)*

软铁（3x3 对称）矩阵 row3 col3 校正寄存器(r/w)。

表 45. MAG_SI_ZZ_L (D0h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_ZZ_L7	MAG_SI_ZZ_L6	MAG_SI_ZZ_L5	MAG_SI_ZZ_L4	MAG_SI_ZZ_L3	MAG_SI_ZZ_L2	MAG_SI_ZZ_L1	MAG_SI_ZZ_L0

表 46. MAG_SI_ZZ_H (D1h)寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MAG_SI_ZZ_H7	MAG_SI_ZZ_H6	MAG_SI_ZZ_H5	MAG_SI_ZZ_H4	MAG_SI_ZZ_H3	MAG_SI_ZZ_H2	MAG_SI_ZZ_H1	MAG_SI_ZZ_H0

该值表示为半精度浮点格式：SEEEEEFFFFFFFFFF（S：1 个符号位；E：5 个指数位；F：10 个分数位）。

3.1.3.11 *FSM_LC_TIMEOUT_L (7Ah)和FSM_LC_TIMEOUT_H (7Bh)*

FSM_LC_TIMEOUT_L (7Ah)和 FSM_LC_TIMEOUT_H (7Bh)寄存器用于设置长计数器超时寄存器值。

表 47. FSM_LC_TIMEOUT_L（7Ah）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC_ TIMEOUT7	FSM_LC_ TIMEOUT6	FSM_LC_ TIMEOUT5	FSM_LC_ TIMEOUT4	FSM_LC_ TIMEOUT3	FSM_LC_ TIMEOUT2	FSM_LC_ TIMEOUT1	FSM_LC_ TIMEOUT0

表 48. FSM_LC_TIMEOUT_H（7Bh）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC_ TIMEOUT16	FSM_LC_ TIMEOUT15	FSM_LC_ TIMEOUT14	FSM_LC_ TIMEOUT13	FSM_LC_ TIMEOUT12	FSM_LC_ TIMEOUT11	FSM_LC_ TIMEOUT10	FSM_LC_ TIMEOUT9

3.1.3.12 *FSM_PROGRAMS（7Ch）*

FSM_PROGRAMS (7Ch)寄存器用于设置已配置状态机的数量。

表 49. FSM_N_PROG（7Ch）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_N_ PROG7	FSM_N_ PROG6	FSM_N_ PROG5	FSM_N_ PROG4	FSM_N_ PROG3	FSM_N_ PROG2	FSM_N_ PROG1	FSM_N_ PROG0

为使设备正常运行，该寄存器必须采用与已配置的状态机相关的配置。允许的最大值为 16 (0x10)。

3.1.3.13 *FSM_START_ADD_L (7Eh)和FSM_START_ADD_H (7Fh)*

FSM_START_ADD_L (7Eh)和 FSM_START_ADD_H (7Fh)寄存器用于设置 FSM 程序的起始地址。

表 50. FSM_START_ADD_L（7Eh）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_START7	FSM_START6	FSM_START5	FSM_START4	FSM_START3	FSM_START2	FSM_START1	FSM_START0

为使设备正常运行，如果未通过 Unico 工具对其进行其他配置，则必须将该寄存器的值设为等于“00h”。

表 51. FSM_START_ADD_H（7Fh）寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_START16	FSM_START15	FSM_START14	FSM_START13	FSM_START12	FSM_START11	FSM_START10	FSM_START9

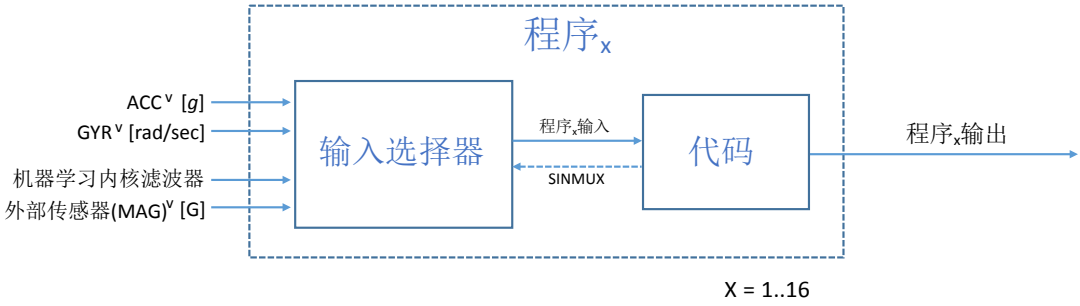
为使设备正常运行，如果未通过 Unico 工具对其进行其他配置，则必须将该寄存器的值设为等于“04h”。

3.2 程序块

来自信号调节模块的输出数据被发送至 FSM 模块，该模块由 16 个程序块组成。如下图所示，每个程序块包括：

- 输入选择器块，用于选择将由程序处理的所需输入数据信号；
- 代码块，由数据和将要执行的指令组成。

图 5. 程序块



3.2.1 输入选择器块

输入选择器块允许在以下物理传感器数据信号之间选择输入数据信号或从内部计算数据信号：

- LSM6DSOX 加速度计数据，通过预先计算的范数(V)；
- LSM6DSOX 陀螺仪数据，通过预先计算的范数(V)；
- 外部传感器（如磁力计）数据，通过预先计算的范数(V)；
- 内部滤波数据，通过适当地配置机器学习内核；
- 内部计算角度，通过预先计算的范数(V)。

通过以下公式在内部计算范数(V)：

$$V = \sqrt{x^2 + y^2 + z^2}$$

机器学习内核允许配置应用于传感器数据的高通、带通、IIR1 和 IIR2 滤波器。

下图显示了加速度计和陀螺仪数字链中的有限状态机块输入。对于 LSM6DSOX 中可用的全部四种连接模式，两个数字链中的有限状态机(FSM)块位置相同。

图 6. FSM 输入（加速度计）

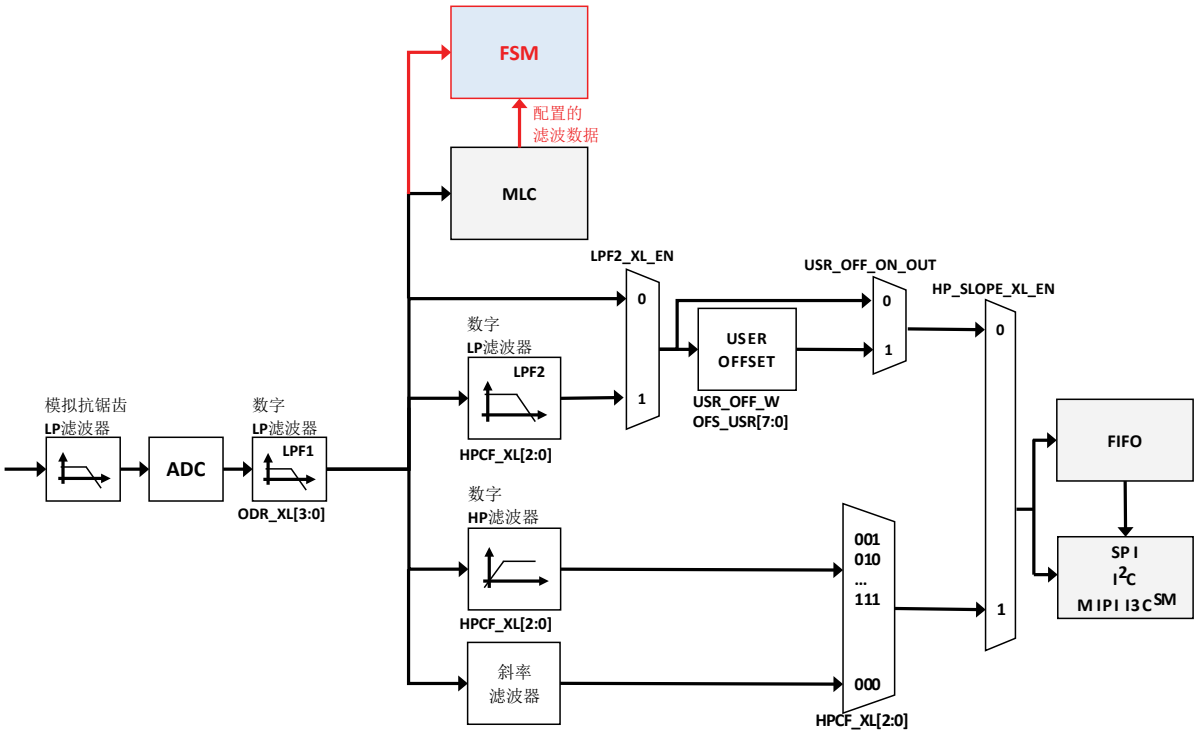
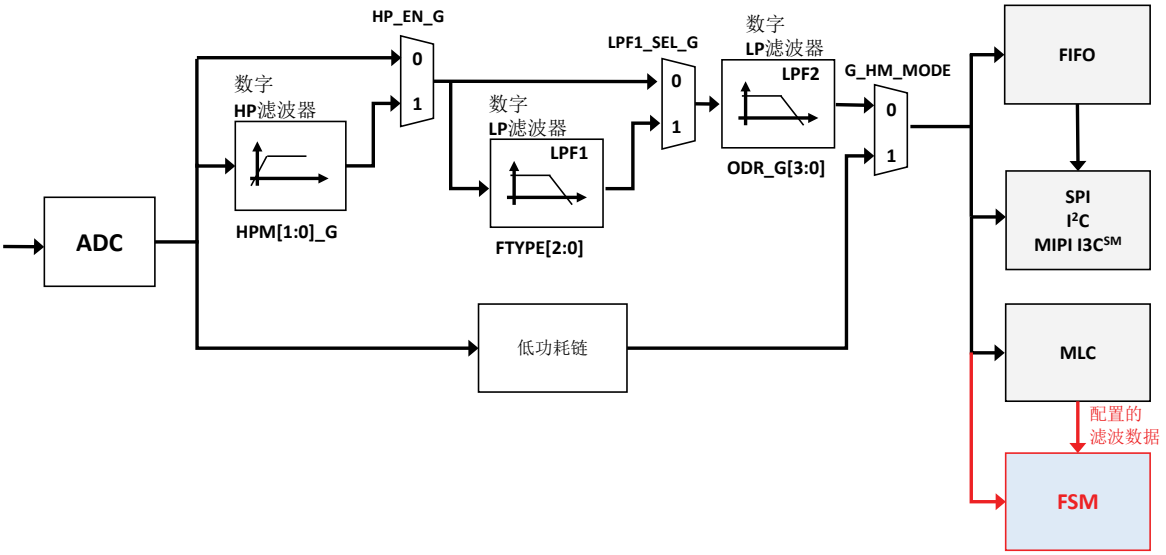


图 7. FSM 输入（陀螺仪）



加速度计和陀螺仪的信号带宽取决于设备配置。有关更多信息，请参见 www.st.com 中的 AN5192。根据程序目的，程序块通过处理所选输入信号并生成相应的程序输出信号来执行已配置的程序（代码块）。

注：程序指令部分中的用户可使用 SINMUX 命令为程序块动态切换所需的输入信号。有关 SINMUX 命令的其他详细信息，请参见 SINMUX（23h）。

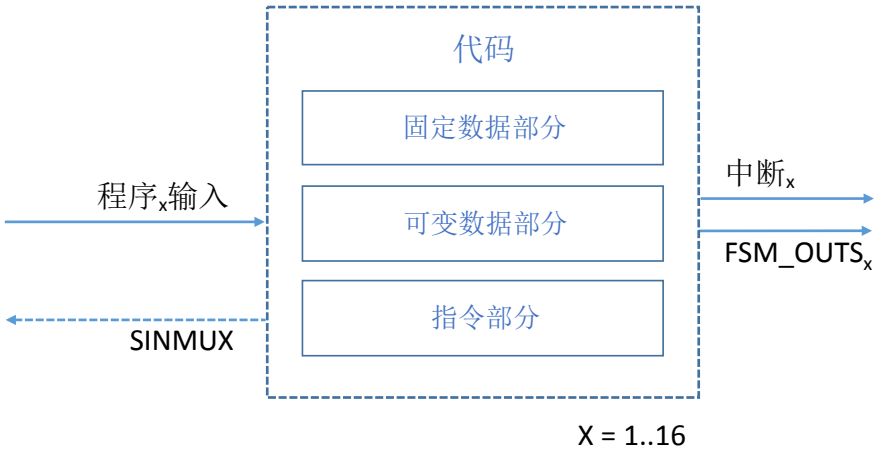
3.2.2 代码块

FSM 程序 $_x$ 代码块包含状态机程序。下图显示了单个程序的结构，由以下部分组成：

- 数据部分，由固定部分（所有 FSM 的大小相同）和可变部分（每个 FSM 具有特定大小）组成；
- 指令部分，由条件和命令组成。

每个程序都可以根据输入 x 信号的已处理样本设置生成中断 x 信号和修改相应的 FSM_OUTS$_x$ 寄存器值。

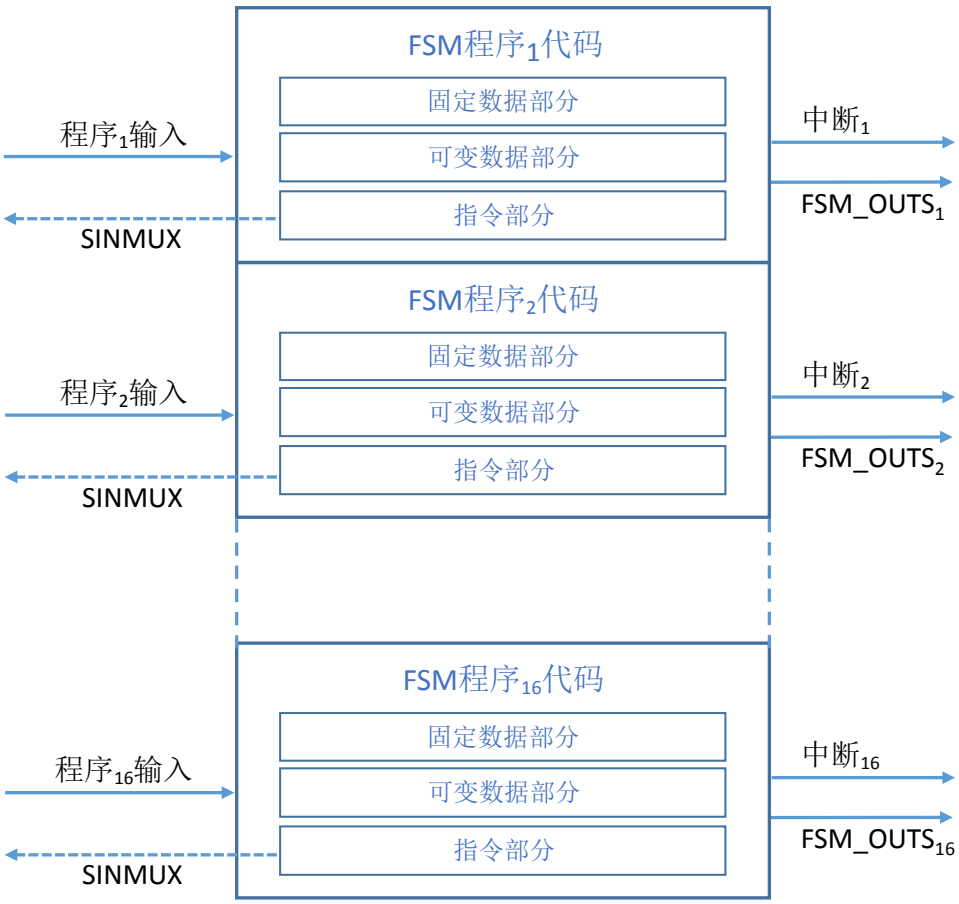
图 8. FSM 程序 x 代码结构



所有 FSM 程序都连续存储在一组保留的嵌入式高级功能寄存器中，如下图所示。每个程序的最大允许大小为 256 字节。

注意：每次器件上电时均必须重新配置 FSM。

图 9. FSM 程序 x 存储区



4 FSM 中断

当达到结束状态或执行某些特定命令（OUTC / CONT / CONTREL 命令）时，将产生 FSM 中断信号。产生中断时，相应的临时掩码值将传输到其相应的 FSM_OUTS 嵌入式功能寄存器。

当 FSM_LONG_COUNTER_L/H 嵌入功能寄存器中存储的长计数器值达到 FSM_LC_TIMEOUT_L/H 嵌入式高级功能寄存器（页面 1）中配置的长计数器超时值时，将产生 FSM 长计数器中断信号。

可以通过读取专用寄存器来检查 FSM 中断和 FSM 长计数器中断信号：

- EMB_FUNC_STATUS_MAINPAGE (35h)寄存器或 EMB_FUNC_STATUS (12h)嵌入式功能寄存器的长计数器中断状态；
- FSM_STATUS_A_MAINPAGE (36h)或 FSM_STATUS_B_MAINPAGE (37h)寄存器或 FSM_STATUS_A (13h)或 FSM_STATUS_B (14h)嵌入式功能寄存器的 FSM 中断状态。

通过设置专用位，可将 FSM 中断信号驱动至 INT1/INT2 中断引脚：

- 将 EMB_FUNC_INT1/EMB_FUNC_INT2 嵌入式功能寄存器的 INT1_FSM_LC/INT2_FSM_LC 位置 1；
- 将 FSM_INT1_A/FSM_INT1_B/FSM_INT2_A/FSM_INT2_B 嵌入式功能寄存器的 INT1_FSM[1:16]/INT2_FSM[1:16]位置 1；

注意：在以上两种情况下，均必须通过设置 MD1_CFG/MD2_CFG 寄存器的 INT1_EMB_FUNC/INT2_EMB_FUNC 位，来使能向 INT1/INT2 中断引脚发送嵌入式功能事件。

默认情况下，中断信号的行为为脉冲行为。脉冲持续时间取决于是否启用更快速的传感器：

- 如果加速度计 ODR 大于陀螺仪 ODR，则脉冲持续时间等于 1/ODRXL；
- 如果陀螺仪 ODR 大于加速度计 ODR，则脉冲持续时间等于 1/ODRG；

注意：最小脉冲持续时间为 1/104 Hz（约 9.6 毫秒）。

通过将 PAGE_RW (17h)嵌入式功能寄存器的 EMB_FUNC_LIR 位置 1，可启用锁存模式。在这种情况下，可通过读取以下状态来重置中断信号：

- EMB_FUNC_STATUS_MAINPAGE (35h)寄存器或 EMB_FUNC_STATUS (12h)嵌入式功能寄存器的长计数器中断状态；
- FSM_STATUS_A_MAINPAGE (36h)或 FSM_STATUS_B_MAINPAGE (37h)寄存器或 FSM_STATUS_A (13h)或 FSM_STATUS_B (14h)嵌入式功能寄存器的 FSM 中断状态。

5 固定数据部分

固定数据部分存储有关可变数据部分和指令部分的信息：它由六个字节组成，位于每个程序的开头。下图显示了固定数据部分的结构。

图 10. 固定数据部分

	NAME	7	6	5	4	3	2	1	0
0	CONFIG A	NR_THRESH(1:0)		NR_MASK(1:0)		NR_LTIMER(1:0)		NR_TIMER(1:0)	
1	CONFIG B	DES	HYST	ANGLE	PAS	DECTREE	STOPDONE	LC	JMP
2	SIZE	PROGRAM SIZE(7:0)							
3	SETTINGS	MASKSEL(1:0)		SIGNED	R_TAM	THRS3SEL	IN_SEL(2:0)		
4	RESET POINTER	RESET POINTER(7:0)							
5	PROGRAM POINTER	PROGRAM POINTER(7:0)							

注意：必须根据程序目的设置绿色位，将程序载入嵌入式高级功能寄存器页面时（通过 FSM 逻辑自动配置它们），必须将红色位置“0”。

前两个字节用于存储程序使用的资源量，而其他字节则供设备存储程序状态。

- 使用 CONFIG_A 可以声明：
  - 最多 3 个阈值（NR_THRESH 位）；
  - 最多 3 个掩码（NR_MASK 位）；
  - 最多 2 个长（16 位）定时器（NR_LTIMER 位）；
  - 最多 2 个短（8 位）定时器（NR_TIMER 位）。
- 使用 CONFIG_B 可以声明：
  - 输入 ODR 的抽取因子（DES 位）；
  - 迟滞值（HYST 位）；
  - 陀螺仪角度的使用，必须计算和存储这些角度（ANGLE 位）；
  - 先前的轴符号的使用，必须计算和存储这些轴符号（PAS 位）；
  - 决策树接口的使用（DECTREE 位）；
  - 长计数器的使用，所有状态机通用（LC 位）。
- SIZE 参数以字节为单位存储整个程序的长度（固定数据部分的大小、可变数据部分的大小以及指令部分的大小之和）。SIZE 字节必须始终为偶数。如果程序大小为奇数，则必须在指令部分的底部添加一个附加 STOP 状态。
- SETTINGS 参数存储当前程序状态（选定掩码、选定阈值、输入信号等...）。
- RESET POINTER (RP)和 PROGRAM POINTER (PP)分别存储复位指针的相对地址（RESET 条件为真时的跳转地址）和程序指针的相对地址（当前采样时间内正在执行的指令的地址）。地址 00h 是指 CONFIG_A 字节。

注意：当 PP 等于“0”时，设备自动运行启动例程（有关更多信息，请参阅第 9 节 启动例程），以正确初始化状态机的内部变量和参数。这是器件正常运行所必须的。

5.1 长计数器

长计数器为用户可用的临时计数器资源。可使用 INCR 命令增加存储在 FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器中的值。

该资源为所有程序所公用，并且在可变数据部分中不需要其他分配的资源。要使用长计数器资源，必须将 CONFIG_B 字节的 LC 位置“1”。

当长计数器值（FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器）等于配置的长计数器超时值（FSM_LC_TIMEOUT_L (7Ah)和 FSM_LC_TIMEOUT_H (7Bh)寄存器）时，EMB_FUNC_STATUS (12h)寄存器的 IS_FSM_LC 状态位置“1”。

可将该信号发送至：

- INT1 引脚，如果：
  - 将 EMB_FUNC_INT1 (0Ah)寄存器的 INT1_FSM_LC 位置 1；
  - 将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置 1。
- INT2 引脚，如果：
  - 如果将 EMB_FUNC_INT2 (0Eh)寄存器的 INT2_FSM_LC 位置 1；
  - 将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置 1。

通过读取 EMB_FUNC_STATUS_MAINPAGE (35h)寄存器或 EMB_FUNC_STATUS (12h)嵌入式功能寄存器来清除 IS_FSM_LC。但是，如果未重置 FSM_LONG_COUNTER 值，则每次发出 INCR 命令时都会产生中断。

为正确重置 FSM 长计数器，用户必须将 FSM_LONG_COUNTER_CLEAR (4Ah)寄存器的 FSM_LC_CLEAR 位置 1。在这种情况下，下次执行 INCR 命令时，启动重置程序。完成该重置程序后，自动将 FSM_LONG_COUNTER_CLEAR (4Ah)寄存器的 FSM_LC_CLEARED 位置“1”。最后，必须将 FSM_LONG_COUNTER_CLEAR（4Ah）寄存器的 FSM_LC_CLEAR 位手动重置为“0”。

6 可变数据部分

可变数据部分位于程序的相应固定数据部分下方，其大小取决于在固定数据部分中定义的资源量。
然后，在固定数据部分中枚举的每个资源都将以适当的大小和适当的位置分配到可变数据部分中。下图显示了可变数据部分的结构。

图 11. 可变数据部分

	NAME	7	6	5	4	3	2	1	0
6 7	THRESH1	THRESH1(15:0)							
8 9	THRESH2	THRESH2(15:0)							
10 11	THRESH3	THRESH3(15:0)							
12 13	HYST	HYSTERESIS(15:0)							
14	MASKA	MASKA(7:0)							
15	TMASKA	TMASKA(7:0)							
16	MASKB	MASKB(7:0)							
17	TMASKB	TMASKB(7:0)							
18	MASKC	MASKC(7:0)							
19	TMASKC	TMASKC(7:0)							
20 21	DELTAT	DELTAT(15:0)							
22 23	DX	DX(15:0)							
24 25	DY	DY(15:0)							
26 27	DZ	DZ(15:0)							
28 29	DV	DV(15:0)							
30 31	TC	TC(15:0) or TC(7:0)							
32 33	TIMER1	TIMER1(15:0)							
34 35	TIMER2	TIMER2(15:0)							
36	TIMER3	TIMER3(7:0)							
37	TIMER4	TIMER4(7:0)							
38	DEST	DEST(7:0)							
39	DESC	DESC(7:0)							
40	PAS	SCTC	CANGLE	MSKIT	MSKITEQ	SIGN_X	SIGN_Y	SIGN_Z	SIGN_V
41	DECTREE	-	DTSEL(2:0)			DTRES(3:0)			

如上表所示，可变数据部分的最大容量为 36 字节。如果程序需要较少的资源，则分配给可变数据部分的容量较小。上表中未显示的从 0 到 5 的字节被分配给固定数据部分的 CONFIG A、CONFIG B、SIZE、SETTINGS、RP 和 PP 字节。

注意：始终从最低资源编号开始使用固定数据部分中声明的资源。例如，如果用户在固定数据部分（定义了两个阈值）中定义了 NR_THRESH = '10'，则程序中可以使用的可用阈值为 THRESH1 和 THRESH2，THRESH3 不可用，并且未分配与 THRESH3 对应的字节（低于 THRESH2 的所有资源均向上移动）。

6.1 阈值

阈值资源用于在比较条件下检查和验证通过所选输入信号（通过 SINMUX 命令）和轴（通过 MASKS）假定的值。
阈值度量单位为所选信号的度量单位：

- 如果选择 LSM6DSOX 加速度计信号，则阈值单位为[*g*]；
- 如果选择 LSM6DSOX 陀螺仪信号，则阈值单位为[rad/sec]；
- 如果选择 LSM6DSOX 集成陀螺仪信号，则阈值单位为[rad]；
- 如果选择外部磁力计传感器信号，则阈值单位为[G]；
- 如果选择 MLC 滤波器信号，则阈值单位与滤波信号单位相同（对于加速度计为[*g*]，对于陀螺仪为[rad/sec]，对于外部磁力计为[G]）。

阈值可以有符号或无符号：可使用 SSIGN0 / SSIGN1 命令从有符号模式转换为无符号模式。在有符号模式下，信号和阈值在比较时保留其原始符号。在无符号模式下，在信号和阈值的绝对值之间进行比较。
通过设置 CONFIG_A 字节的 NR_THRESH[1:0]位，可在可变数据部分中配置相应数量的阈值，如下所述：

- NR_THRESH[1:0] = ‘00’：在可变数据部分中未分配阈值。
- NR_THRESH[1:0] = ‘01’：在可变数据部分中只分配了 THRESH1[15:0]。
- NR_THRESH[1:0] = ‘10’：在可变数据部分中分配了 THRESH1[15:0]和 THRESH2[15:0]。
- NR_THRESH[1:0] = ‘11’：在可变数据部分中分配了 THRESH1[15:0]、THRESH2[15:0]和 THRESH3[15:0]。

涉及命令：

- STHR1 / STHR2；
- SELTHR1 / SELTHR3；
- SSIGN0 / SSIGN1。

涉及条件：

- GNTH1 / GNTH2 / GLTH1 / GRTH1；
- LNTH1 / LNTH2 / LLTH1 / LRTH1。

6.2 迟滞

执行阈值比较时，迟滞资源会影响当前阈值。如果使用迟滞资源，则迟滞值将自动：

- 添加到用于所有“大于”条件（GNTH1、GNTH2、GLTH1 和 GRTH1）的阈值；
- 从用于所有“小于”条件（LNTH1、LNTH2、LLTH1 和 LRTH1）的阈值减去。

例如：

- 如果执行“GNTH1”条件，则使用的阈值为：THRESH1 + 迟滞；
- 如果执行“LNTH2”条件，则使用的阈值为：THRESH2 – 迟滞。

通过将 CONFIG_B 字节的 HYST 位置“1”，可在可变数据部分中正确配置 HYSTERESIS[15:0]资源。
涉及命令：

- N/A.

涉及条件：

- GNTH1 / GNTH2 / GLTH1 / GRTH1；
- LNTH1 / LNTH2 / LLTH1 / LRTH1。

注意：迟滞不影响过零条件。

6.3 掩码/临时掩码

掩码资源用于在执行条件时启用或禁用输入数据(X, Y, Z, V)上的掩码操作。如果将掩码位置 1，则启用相应的轴和符号，否则将其禁用。掩码用于阈值比较条件或过零检测。掩码允许通过使用负号使能相应轴位来反转输入信号的符号。掩码由 8 位组成（每个轴 2 位），如下所示：

+X	-X	+Y	-Y	+Z	-Z	+V	-V

对于每个轴，可配置四种不同的掩码设置：

1. 正轴位 = 0 / 负轴位 = 0，禁用轴；
2. 正轴位 = 0 / 负轴位 = 1，启用相反符号的轴；
3. 正轴位 = 1 / 负轴位 = 0，启用当前符号的轴；
4. 正轴位 = 1 / 负轴位 = 1，启用当前符号的轴和相反符号的轴。

启用程序后，将每个掩码的值复制到相关临时掩码(TM)中，该临时掩码将在条件执行期间使用。每次发出条件时，将条件结果再次存储在临时掩码中（这也会影响临时条件）。

示例：

- “GNTH1”条件；
- THRESH1 = 0.50 *g*；
- MASKA = 12h (00010010b) → 启用-Y 和+V；
- 当前输入加速度计样本 = [0.72 -0.45 0.77 1.15]。

TM before the condition	0	0	0	1	0	0	1	0
Accelerometer sample	0.72	-0.72	-0.45	0.45	0.77	-0.77	1.15	-1.15
TM after the condition	0	0	0	0	0	0	1	0

在以下情况下，可以将临时掩码值重置为掩码值：

- 有复位条件的任何时候；
- 执行 CONTREL 命令时；
- 执行 REL 命令时；
- 在每个真的下一个条件后，如果先前已发出 SRTAM1 命令。

通过设置 CONFIG_A 字节的 NR_MASK[1:0]位，可在可变数据部分中配置相应数量的掩码，如下所述：

- NR_ MASK[1:0] = ‘00’：在可变数据部分中未分配任何掩码；
- NR_ MASK[1:0] = ‘01’：在可变数据部分中只分配 MASKA[7:0]/TMASKA[7:0]；
- NR_ MASK[1:0] = ‘10’：在可变数据部分中分配 MASKA[7:0]/TMASKA[7:0]和 MASKB[7:0]/TMASKB[7:0]；
- NR_ MASK[1:0] = ‘11’：在可变数据部分中分配 MASKA[7:0]/TMASKA[7:0]、MASKB[7:0]/TMASKB[7:0]和 MASKC[7:0]/TMASKC[7:0]。

涉及命令：

- SELMA / SELMB / SELMC；
- SMA / SMB / SMC；
- REL；
- SRTAM0 / SRTAM1；
- SWAPMSK；
- SISW。

涉及条件：

- GNTH1 / GNTH2 / GLTH1 / GRTH1；
- LNTH1 / LNTH2 / LLTH1 / LRTH1；
- PZC / NZC。

6.4 角度计算

发出条件时，角度资源可用于代替角速度数据。角度计算在内部执行：陀螺仪数据自动乘以 DELTAT 值，并将结果添加到相应的角度轴字节（DX、DY、DZ 和 DV）。将固定数据部分的 CONFIG B 字节的 ANGLE 位置 1 时，启用此功能。每次由 FSM 处理新样本时，都会独立于所选信号进行积分。

有两种复位角度的方式：

- 默认情况下，每次复位或下一个条件为真时，都会清除角速度积分。在这种情况下，当新样本抵达时，计算角度（DX、DY、DZ 和 DV 字节）将从零开始；
- 如果程序包含 CANGLE 命令，则使用其他复位角度模式。在这种情况下，清除积分角度：
  - 如果执行 CANGLE 命令（新样本到达时）；
  - 仅在复位条件为真时。

将 CONFIG_B 字节的 ANGLE 位置“1”时，在可变数据部分中分配 10 个字节（DELTAT、DX、DY、DZ 和 DV）：必须将 DELTAT 资源设为等于当前的 FSM_ODR 周期时间（以秒为单位）（半浮点（16 位）格式）。如果期望使用 CANGLE 命令，则还必须将 CONFIG_B 字节的 PAS 位置“1”。

涉及命令：

- CANGLE。

涉及条件：

- GNTH1 / GNTH2 / GLTH1 / GRTH1；
- LNTH1 / LNTH2 / LLTH1 / LRTH1；
- PZC / NZC。

6.5 TC 和定时器

定时器资源用于管理事件持续时间。可以声明两种定时器资源：长定时器（16 位）和短定时器（8 位）。时基由 EMB_FUNC_ODR_CFG_B (5Fh)寄存器的 FSM_ODR[1:0]位设置，包括抽取因子（如果使用）。长定时器资源被称为 TI1 和 TI2，短定时器资源则被称为 TI3 和 TI4。附加的内部定时器计数器(TC)用作临时计数器，以检查计时是否已结束。TC 值可以两种不同的方式预载，可使用 SCTC0 / SCTC1 命令加以选择：

- SCTC0 模式（默认）：当程序指针移至具有超时条件的状态时，TC 值始终预装到相应的定时器值。在这种模式下，定时器持续时间仅影响一种状态。
- SCTC1 模式：当程序指针移至具有超时条件的状态时，根据在新状态中使用哪个定时器，有两种不同的方案：
  - 如果在新状态下使用的定时器与前一状态下使用的定时器不同，则将 TC 值预装入相应的定时器值中。在这种模式下，定时器持续时间仅影响一种状态（与 SCTC0 模式相同）；
  - 如果新状态下使用的定时器与前一状态下使用的定时器相同，则不会预装 TC 值。TC 值从其先前值开始继续减小。在这种模式下，定时器持续时间可能会影响更多状态。

每次出现新样本时，TC 值都会减 1：如果 TC 达到“0”，则条件为真。

示例：

- 定时器 TI3 设为等于 10。考虑以下状态：
  - S0 - SCTC0 或 SCTC1
  - S1 - TI3 | GNTH1
  - S2 - TI3 | LNTH2
  - S3 - TI3 | GNTH1
- TI3 = 0Ah（10 个样本）；

根据 S0，有两种不同的状态机行为：

- SCTC0 情况：总是预装 TC 字节（程序指针移至状态 S1、S2 和 S3），每个条件最多检查 10 个样本。这意味着最多可在 30 个样本中验证所有条件；
- SCTC1 情况：仅在程序指针移至 S1 时才预装 TC 字节（在移至 S2 和 S3）时不预装，并且必须在最多 10 个样本中验证所有条件。

当必须在同一时间窗口中验证不同条件时，通常使用 SCTC1 模式。

通过设置 CONFIG_A 字节的 NR_LTIMER[1:0]位，可在可变数据部分中配置相应数量的长计数器，如下所述：

- NR_LTIMER[1:0] = ‘00’：未在可变数据部分中分配长定时器；
- NR_LTIMER[1:0] = ‘01’：将 TIMER1[15:0]分配到可变数据中；

- NR_LTIMER(1:0) = ‘10’：将 TIMER1[15:0]和 TIMER2[15:0]分配到可变数据部分。

通过设置 CONFIG_A 字节的 NR_TIMER[1:0]位，可在可变数据部分中配置相应数量的短定时器，如下所述：

- NR_TIMER[1:0] = ‘00’：在可变数据部分中未分配任何短定时器；
- NR_TIMER[1:0] = ‘01’：将 TIMER3[7:0]分配到可变数据中；
- NR_TIMER[1:0] = ‘10’：将 TIMER3[7:0]和 TIMER4[7:0]分配到可变数据部分中。

以下为 TC 资源的大小：

- 如果 NR_LTIMER[1:0] = ‘00’且 NR_TIMER[1:0] = ‘00’，则不分配 TC 资源；
- 如果 NR_LTIMER[1:0] = ‘00’且 NR_TIMER[1:0] ≠ ‘00’，则 TC 资源占用一个字节；
- 如果 NR_LTIMER[1:0] ≠ ‘00’且 NR_TIMER[1:0] = ‘00’，则 TC 资源占用两个字节；
- 如果 NR_LTIMER[1:0] ≠ ‘00’且 NR_TIMER[1:0] ≠ ‘00’，则 TC 资源占用两个字节；

涉及命令：

- STIMER3 / STIMER4；
- SCTC0 / SCTC1。

涉及条件：

- TI1 / TI2 / TI3 / TI4。

6.6 抽取器

抽取器资源用于减少进入有限状态机的数据的采样率。

通过将 CONFIG_B 字节的 DES 位设置为“1”，可在可变数据部分中正确配置 DEST 和 DESC 字节。DEST 值为所需抽取因子，而 DESC 值为内部计数器（由设备自动管理）。根据以下公式，抽取因子与 EMB_FUNC_ODR_CFG_B (5Fh)寄存器的 FSM_ODR[1:0]位相关：

PROGRAM_ODR = FSM_ODR / DEST

启动时：

DESC = DEST（初始抽取值）

采样时钟发生时：

DESC = DESC - 1

当 DESC 等于 0 时，将当前样本用作状态机的新输入，并再次将 DESC 值设为初始抽取值。

涉及命令：

- N/A.

涉及条件：

- N/A.

注：DEST 的最小有意义值为“2”。

6.7 前一个轴符号

前一个轴符号资源主要用于存储前一个样本的符号：此信息用于过零条件。此外，也用于存储其他信息，如所选的定时器复位方法（SCTC0 或 SCTC1），用于重置积分陀螺仪数据（可变数据部分中的 DX、DY、DZ 和 DV 字节）和所选中断屏蔽类型（MSKIT、MSKITEQ 或 UMSKIT）的清除角标志(CANGLE)。

通过将 CONFIG_B 字节的 PAS 位置“1”，可将 PAS 字节分配到可变数据部分中（PAS 字节值由器件自动管理）。如果下面列出的命令或条件至少有一个将用于程序，则这具有强制性。

涉及命令：

- SCTC0 / SCTC1 / CANGLE / MSKIT / MSKITEQ / UMSKIT。

涉及条件：

- PZC / NZC。

注意：如果执行了 SSIGN0 命令，则将 NZC 和 PZC 用作常规 ZC 条件。

6.8 决策树接口

可使用 CHKDT 条件访问决策树接口资源，该条件可用于检查机器学习内核算法中可用的八个决策树之一的结果。当期望将机器学习逻辑与 FSM 程序结合使用时，这可能非常有用。

通过将 CONFIG_B 字节的 DECTREE 位置 1，可在可变数据部分中正确配置 DECTREE 字节。DECTREE 字节包含有关要触发的决策树累进数字（DTSEL(2:0)位，0 到 7）和对应的期望值（DTRES(3:0)位，0 到 15）。

使用 SETP 命令可动态重新配置程序流中的 DECTREE 字节，以触发不同的决策树及其期望值。有关 SETP 命令的详细信息，请参见其专用段落。

涉及命令：

- N/A

涉及条件：

- CHKDT

注意：无法执行以下条件：

- PZC | CHKDT 操作码(0xDF)等于 SMB 操作码；
- CHKDT | NZC 操作码(0xFE)等于 SMC 操作码；
- NZC | CHKDT 操作码(0xEF)等于 MSKITEQ 操作码；
- CHKDT | GNTH1 操作码(0xF5)等于 MSKIT 操作码。

7 指令部分

指令部分在可变数据部分下面定义，并由实现算法逻辑的一系列状态组成。每个状态都有一个 8 位操作码 (opcode)，每个操作码均可以实现命令或 RESET/NEXT 条件：

- 这些命令用于执行特殊的流控、输出和同步任务。某些命令可能有参数，以单步指令的形式执行；
- RESET/NEXT 条件是两个条件的组合（4 位用于 RESET 条件，4 位用于 NEXT 条件），用于复位或继续执行程序流。

操作码对寄存器和内部状态机存储器有直接影响。对于某些操作码，可能会产生其他副作用（如更新状态信息）。

RESET/NEXT 条件或命令，后面带参数，表示一条指令，也被称为程序状态。它们为程序指令部分的模块。

7.1 Reset/Next 条件

RESET/NEXT 条件用于复位或继续执行程序流程。新样本集合就绪时，RESET/NEXT 条件将在一种状态下执行。

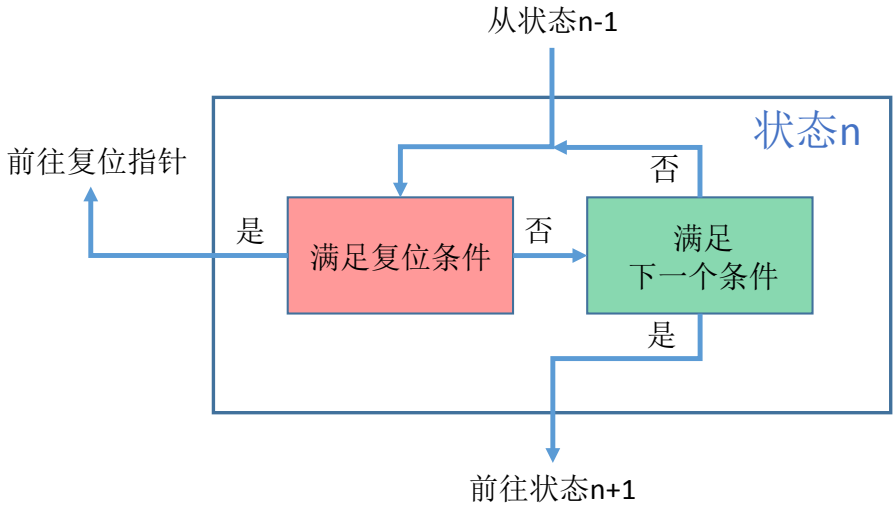
RESET 条件在操作码 MSB 部分中定义，而 NEXT 条件在操作码 LSB 部分中定义。如下图所示，始终在 NEXT 条件之前执行 RESET 条件，即仅在不满足 RESET 条件时才进行评估。

当两个条件（NEXT 和 RESET）都未满足时，状态机将等待新样本集合(X, Y, Z, V)，然后以相同状态再次开始评估。

只要“RESET 条件”为真(PP = RP)，就会向复位指针转移。

只要“RESET 条件”为假且“NEXT 条件”为真且(PP = PP + 1)，就会向复位指针转移。

图 12. 单状态说明



注释：始终在 NEXT 条件之前评估 RESET 条件。默认情况下，将复位指针(RP)设为第一个状态，但可以使用 SRP/CRP 命令动态更改复位指针(RP)。

由于条件通过四个位编码，因此最多可对十六种不同的条件进行编码：下表显示了可用条件列表。有三种类型的条件：

- 超时：当预装定时器值的 TC 计数器达到零时，这些条件为真。
- 阈值比较：当使能的输入（如加速度计 X、Y、Z 轴或范数）高于（或低于）已编程阈值时，这些条件为真；

$$V = \sqrt{x^2 + y^2 + z^2}$$

- 过零检测：当使能的输入跨过零值时，这些条件为真。

表 52. 条件

操作码	助记符	说明	注释	所需资源
0h	NOP	无操作	执行移至另一个条件	N/A
1h	TI1	定时器 1（16 位值）有效	未对数据样本进行评估	TC，TIMER1
2h	TI2	定时器 2（16 位值）有效		TC，TIMER1，TIMER2
3h	TI3	定时器 3（8 位值）有效		TC，TIMER3
4h	TI4	定时器 4（8 位值）有效		TC，TIMER3，TIMER4
5h	GNTH1	任何触发轴 > THRESH1	输入信号，用掩码触发，与阈值比较	THRESH1，一个 MASK
6h	GNTH2	任何触发轴 > THRESH2		THRESH1，THRESH2，一个 MASK
7h	LNTH1	任何触发轴 ≤ THRESH1		THRESH1，一个 MASK
8h	LNTH2	任何触发轴 ≤ THRESH2		THRESH1，THRESH2，一个 MASK
9h	GLTH1	所有触发轴 > THRESH1		THRESH1，一个 MASK
Ah	LLTH1	所有触发轴 ≤ THRESH1		THRESH1，一个 MASK
Bh	GRTH1	任何触发轴 > -THRESH1		THRESH1，一个 MASK
Ch	LRTH1	任何触发轴 ≤ - THRESH1		THRESH1，一个 MASK
Dh	PZC	任何触发轴均以正斜率越过零值	输入信号，通过掩码触发，越过零值	PAS
Eh	NZC	任何触发轴均以负斜率越过零值		PAS
Fh	CHKDT	对比预期结果检查决策树结果	需要机器学习内核配置	DECTREE

上表的最后一列指示条件所需的资源。这些资源在可变数据部分中分配，并且不同 FSM 之间的资源可能不同。为确保 FSM 行为正确，必须在固定数据部分中设置每个程序所需的资源量。

注意：让 NEXT 条件和 RESET 条件相同没有意义。因此，诸如 11h 之类的操作码无法实现 TI1 | TI1 条件，但可以实现一些命令：例如，操作码 11h 实现了 CONT 命令。

7.1.1 NOP（0h）

说明：对于某些不需要有效相反条件的特定条件，NOP（无操作）用来填充 RESET/NEXT。

动作：

- 如果 NOP 处于 RESET 条件：当新样本集合就绪时，仅评估 NEXT 条件；
- 如果 NOP 处于 NEXT 条件：当新样本集合就绪时，仅评估 RESET 条件。

7.1.2 TI1（1h）

说明：在 TI1 条件下，对 TC 字节的计数器值进行计数和评估。

动作：

- 当程序指针移至具有 TI1 条件的状态时，TC = TIMER1；
- 当出现新的样本集合(X, Y, Z, V)时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较（等待新样本）；
  - 如果 TC = 0，则条件有效：
    - 如果 TI1 处于 RESET 位置，PP = RP；
    - 如果 TI1 处于 NEXT 位置，PP = PP + 1。

7.1.3 TI2（2h）

说明：在 TI2 条件下，对 TC 字节的计数器值进行计数和评估。

动作：

- 当程序指针移至具有 TI2 条件的状态时，TC = TIMER2；
- 当出现新的样本集合(X, Y, Z, V)时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较（等待新样本）；
  - 如果 TC = 0，则条件有效：
    - 如果 TI2 处于 RESET 位置，PP = RP；
    - 如果 TI2 处于 NEXT 位置， PP = PP + 1。

7.1.4 TI3（3h）

说明：在 TI3 条件下，对 TC 字节的计数器值进行计数和评估。

动作：

- 当程序指针移至具有 TI3 条件的状态时，TC = TIMER3；
- 当出现新的样本集合(X, Y, Z, V)时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较（等待新样本）；
  - 如果 TC = 0，则条件有效：
    - 如果 TI3 处于 RESET 位置，PP = RP；
    - 如果 TI3 处于 NEXT 位置， PP = PP + 1。

7.1.5 TI4（4h）

说明：在 TI4 条件下，对 TC 字节的计数器值进行计数和评估。

动作：

- 当程序指针移至具有 TI4 条件的状态时，TC = TIMER4；
- 当出现新的样本集合(X, Y, Z, V)时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较（等待新样本）；
  - 如果 TC = 0，则条件有效：
    - 如果 TI4 处于 RESET 位置，PP = RP；
    - 如果 TI4 处于 NEXT 位置，PP = PP + 1。

7.1.6 GNTH1（5h）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均大于阈值 1 水平，则 GNTH1 条件有效。阈值为：THRESH1 + HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 GNTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 GNTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.7 GNTH2（6h）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均大于阈值 2 水平，则 GNTH2 条件有效。阈值为：THRESH2 + HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 GNTH2 有效且处于 RESET 位置，则 PP = RP；
  - 如果 GNTH2 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.8 LNTH1（7h）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均小于或等于阈值 1 水平，则 LNTH1 条件有效。阈值为：THRESH1 - HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 LNTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 LNTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.9 LNTH2（8h）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均小于或等于阈值 2 水平，则 LNTH2 条件有效。阈值为：THRESH2 - HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 LNTH2 有效且处于 RESET 位置，则 PP = RP；
  - 如果 LNTH2 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.10 GLTH1（9h）

说明：如果数据样本集合(X, Y, Z, V)的所有轴均大于阈值 1 水平，则 GLTH1 条件有效。阈值为：THRESH1 + HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 GLTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 GLTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.11 LLTH1（Ah）

说明：如果数据样本集合(X, Y, Z, V)的所有轴均小于或等于阈值 1 水平，则 LLTH1 条件有效。阈值为：THRESH1 - HYST。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 LLTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 LLTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.12 GRTH1（Bh）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均大于阈值 1 水平，则 GRTH1 条件有效。阈值为：–(THRESH1 + HYST)。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 GRTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 GRTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.13 LRTH1（Ch）

说明：如果数据样本集合(X, Y, Z, V)的任何触发轴均小于或等于阈值 1 水平，则 LRTH1 条件有效。阈值为：–(THRESH1 – HYST)。

注：仅在将固定数据部分的 CONFIG_A(HYST) 位置“1”且可变数据部分中的 HYST 值不为“0”时，HYST 值才参与阈值比较。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果 LRTH1 有效且处于 RESET 位置，则 PP = RP；
  - 如果 LRTH1 有效且处于 NEXT 位置，则 PP = PP + 1；

7.1.14 PZC（Dh）

说明：如果数据采样集合的任意触发轴(X, Y, Z, V)以正斜率越过零值，则 PZC 条件有效。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果发生具有正斜率的过零事件且 PZC 位于 RESET 位置，则 PP = RP；
  - 如果发生具有正斜率的过零事件且 PZC 处于 NEXT 位置，则 PP = PP + 1。

7.1.15 NZC（Eh）

说明：如果数据样本集合的任意触发轴(X, Y, Z, V)以负斜率越过零值，则 NZC 条件有效。

动作：

- 出现新样本集合(X, Y, Z, V)时，请检查条件：
  - 如果发生负斜率过零事件且 NZC 处于 RESET 位置，则 PP = RP；
  - 如果发生负斜率过零事件且 NZC 处于 NEXT 位置，则 PP = PP + 1。

7.1.16 CHKDT（Fh）

说明：如果所选决策树的结果为预期结果，则 CHKDT 条件有效。有关如何正确配置决策树接口的更多信息，请参阅第 6.8 节 决策树接口。

动作：

- 出现新样本集合(X, Y, Z, V)时，检查所选决策树的输出，如果输出为预期输出：
  - 如果 CHKDT 位于 RESET 位置，则 PP = RP；
  - 如果 CHKDT 位于 NEXT 位置，则 PP = PP + 1。

7.2 指令

指令用于修改程序的流控、输出和同步行为。

指令将立即执行（无需新的样本集合）：执行指令时，将程序指针设为下一行，即立即进行评估：

- 如果新行是指令，则立即再次执行；
- 如果新行是条件，则在处理下一个样本时执行。

某些指令可能需要必须在指令操作码下面定义参数（通过报告参数值的专用操作码）。请参考以下示例，该示例显示了三个连续操作码，这些操作码用于在执行 STHR1 命令时动态更改“THRESH1”资源的值：

“AAh”（STHR1 命令）

“CDh”（第 1 个参数）

“3Ch”（第 2 个参数）

当程序指针达到“AAh”（STHR1 命令）状态时，设备会识别出这是一个需要两个参数的命令：这三种状态将立即执行，而不等待新的样本集合。在指令执行完成后，THRESH1 资源值被设为“3CCDh”，等于“1.2”。

表 53. 命令列表

操作码	助记符	说明	参数
00h	STOP	停止执行，然后等待复位指针重新开始	无
11h	CONT	从复位指针继续执行	无
22h	CONTREL	从复位指针继续执行，重置临时掩码	无
33h	SRP	将复位指针设置为下一个地址/状态	无
44h	CRP	将复位指针清除到第一个程序行	无
55h	SETP	设置程序存储器中的参数	字节 1：地址 字节 2：值
66h	SELMA	将 MASKA 和 TMASKA 选为当前掩码	无
77h	SELMB	将 MASKB 和 TMASKB 选为当前掩码	无
88h	SELMC	将 MASKC 和 TMASKC 选为当前掩码	无
99h	OUTC	将临时掩码写入输出寄存器	无
AAh	STHR1	设置 THRESH1 寄存器的新值	字节 1：THRESH1 [LSB] 字节 2：THRESH1 [MSB]
BBh	STHR2	设置 THRESH2 寄存器的新值	字节 1：THRESH2 [LSB] 字节 2：THRESH2 [MSB]
CCh	SELTHR1	选择 THRESH1 来替代 THRESH3	无
DDh	SELTHR3	选择 THRESH3 来替代 THRESH1	无
EEh	SISW	将选定掩码中的符号信息设为相反	无
FFh	REL	将临时掩码重置为默认值	无
12h	SSIGN0	设置 UNSIGNED 比较模式	无
13h	SSIGN1	设置 SIGNED 比较模式	无
14h	SRTAM0	下一个条件为真后不重置临时掩码	无
21h	SRTAM1	下一个条件为真后重置临时掩码	无
23h	SINMUX	设置输入多路复用器	字节 1：多路复用器的输入值
24h	STIMER3	设置 TIMER3 寄存器的新值	字节 1：TI3 值
31h	STIMER4	设置 TIMER4 寄存器的新值	字节 1：TI4 值
32h	SWAPMSK	切换掩码选择 MASKA <=> MASKB；MASKC 不受影响	无
34h	INCR	增加长计数器+1，检查长计数器超时并清除	无

操作码	助记符	说明	参数
41h	JMP	两个 Next 条件的跳转地址	字节 1：条件 字节 2：重置跳转地址 字节 3：下一个跳转地址
42h	CANGLE	清除角度	
43h	SMA	设置 MASKA 和 TMASKA	字节 1：MASKA 值
DFh	SMB	设置 MASKB 和 TMASKB	字节 1：MASKB 值
FEh	SMC	设置 MASKC 和 TMASKC	字节 1：MASKC 值
5Bh	SCTC0	下一个条件为真时清除时间计时器 TC	无
7Ch	SCTC1	下一个条件为真时不清除时间计时器 TC	无
C7h	UMSKIT	设置 OUTS 时取消屏蔽中断生成	无
EFh	MSKITEQ	如果 OUTS 不变，则在设置 OUTS 时屏蔽中断生成	无
F5h	MSKIT	设置 OUTS 时屏蔽中断生成	无

7.2.1 STOP（00h）

说明：STOP 命令可停止执行并等待主机重启。该命令用于控制程序结束。

参数：无。

动作：

- 将结果掩码输出到 $OUTS_x$ 寄存器；
- 产生中断（如果使能，则相应地使用 MSKIT / MSKITEQ / UMSKIT 命令）；
- 通过将固定数据部分的 CONFIG_B(STOPDONE)位置“1”来停止自身。要重启程序，用户应禁用和启用 FSM_ENABLE_A (46h)或 FSM_ENABLE_B (47h)寄存器中的相应状态机位。在这种情况下，执行启动例程。有关启动例程的更多信息，请参阅第 9 节 启动例程。

7.2.2 CONT（11h）

说明：CONT 命令循环执行到复位点。该命令用于控制程序结束。

参数：无。

动作：

- 将结果掩码输出到 $OUTS_x$ 寄存器；
- 产生中断（如果使能，则相应地使用 MSKIT / MSKITEQ / UMSKIT 命令）；
- PP = RP。

7.2.3 CONTREL（22h）

说明：CONTREL 命令循环执行到复位点。该命令用于控制程序结束。同时，将临时掩码值重置为其默认值。

参数：无。

动作：

- 将结果掩码输出到 $OUTS_x$ 寄存器；
- 将临时掩码重置为默认值；
- 产生中断（如果使能，则相应地使用 MSKIT / MSKITEQ / UMSKIT 命令）；
- PP = RP。

7.2.4 SRP（33h）

说明：SRP 命令将复位指针设为下一个地址/状态。该命令用于修改程序的起点。

参数：无。

动作：

- RP = PP + 1；
- PP = PP + 1。

7.2.5 CRP（44h）

说明：CRP 命令将复位指针清除到起始位置（程序代码开头）。

参数：无。

动作：

- RP = 程序代码开头；
- PP = PP + 1。

7.2.6 SETP（55h）

说明：SETP 命令可用于修改当前所使用的状态机配置。该命令用于修改当前状态机所需地址处的字节值。

参数：两个字节。

- 第 1 个参数：要修改的字节地址（8 位）。该地址与当前状态机有关（地址 00h 是指 CONFIG_A 字节）；
- 第 2 个参数：要写入第 1 个参数地址的新值（8 位）。

动作：

- 通过第 1 个参数 = 第 2 个参数对字节值进行寻址
- PP = PP + 3。

7.2.7 SELMA（66h）

说明：SELMA 命令将 MASKA / TMASKA 设为当前掩码。

参数：无。

动作：

- MASK_A 选择。将固定数据段的 SETTINGS(MASKSEL[1:0])位设为“00”；
- PP = PP + 1。

7.2.8 SELMB（77h）

说明：SELMB 命令将 MASKB / TMASKB 设为当前掩码。

参数：无。

动作：

- 选择 MASK_B。将固定数据段的 SETTINGS(MASKSEL[1:0])位设为“01”；
- PP = PP + 1。

7.2.9 SELMC（88h）

说明：SELMC 命令将 MASKC / TMASKC 设为当前掩码。

参数：无。

动作：

- 选择 MASK_C。将固定数据段的 SETTINGS(MASKSEL[1:0])位设为“10”；
- PP = PP + 1

7.2.10 OUTC（99h）

说明：OUTC 代表输出命令。该命令用于将 OUTS 寄存器的值更新为当前的临时掩码值，并产生中断（如果使能）。

参数：无。

动作：

- 将当前状态机的 OUTS 寄存器更新为选定的临时掩码值；
- 产生中断（如果使能，则相应地使用 MSKIT / MSKITEQ / UMSKIT 命令）；
- PP = PP + 1。

7.2.11 STHR1（AAh）

说明：STHR1 命令将 THRESH1 值设为新的所需值。THRESH1 是一个半浮点数（16 位）。

参数：两个字节。

- 第 1 个参数：THRESH1 LSB 值（8 位）；
- 第 2 个参数：THRESH1 MSB 值（8 位）。

动作：

- 为 THRESH1 设置新值；
- PP = PP + 3。

7.2.12 STHR2（BBh）

说明：STHR2 命令将 THRESH2 值设为新的所需值。THRESH2 为一个半浮点数（16 位）。

参数：两个字节。

- 第 1 个参数：THRESH2 LSB 值（8 位）；
- 第 2 个参数：THRESH2 MSB 值（8 位）。

动作：

- 为 THRESH2 设置新值；
- PP = PP + 3。

7.2.13 SELTHR1（CCh）

说明：执行 SELTHR1 命令后，在执行 GNTH1、LNTH1、GLTH1、LLTH1、GRTH1、LRTH1 条件时，使用 THRESH1 值替代 THRESH3 值。

参数：无。

动作：

- 选择 THRESH1 来替代 THRESH3。将固定数据部分的 SETTINGS(THRS3SEL)位置“0”；
- PP = PP + 1。

7.2.14 SELTHR3（DDh）

说明：执行 SELTHR3 命令后，在执行 GNTH1、LNTH1、GLTH1、LLTH1、GRTH1、LRTH1 条件时，使用 THRESH3 值替代 THRESH1 值。

参数：无。

动作：

- 选择 THRESH3 来替代 THRESH1。将固定数据部分的 SETTINGS(THRS3SEL)位置“1”；
- PP = PP + 1。

7.2.15 SISW（EEh）

说明：SISW 命令将临时轴掩码符号替换为相反符号。

参数：无。

动作：

- 将选定的临时掩码轴符号更改为相反符号：
  - 如果符号（轴）为正，则新符号（轴）为负；
  - 如果符号（轴）为负，则新符号（轴）为正；
  - 如果轴信息为零，则不改变。
- PP = PP + 1。

7.2.16 REL（FFh）

命令：REL 命令可释放临时掩码轴信息。

参数：无。

动作：

- 将当前的临时掩码重置为默认值；
- PP = PP + 1。

7.2.17 SSIGN0（12h）

说明：SSIGN0 命令将比较模式设为“无符号”。

参数：无。

动作：

- 将比较模式设为“无符号”。将固定数据部分的 SETTINGS(SIGNED)位置“0”；
- PP = PP + 1。

7.2.18 SSIGN1（13h）

说明：SSIGN1 命令将比较模式设为“有符号”（默认行为）。

参数：无。

动作：

- 将比较模式设为“有符号”。将固定数据部分的 SETTINGS(SIGNED)位置“1”；
- PP = PP + 1。

7.2.19 SRTAM0（14h）

说明：SRTAM0 命令用于在 NEXT 条件为真（默认行为）时保留临时掩码值。

参数：无。

动作：

- 临时轴掩码值在有效 NEXT 条件后不变。将固定数据段的 SETTINGS(R_TAM)位置“0”；
- PP = PP + 1。

7.2.20 SRTAM1（21h）

说明：SRTAM1 命令用于在 NEXT 条件为真时重置临时掩码。

参数：无。

动作：

- 临时轴掩码值在有效 NEXT 条件后复位。将固定数据段的 SETTINGS(R_TAM)位置“1”；
- PP = PP + 1。

7.2.21 SINMUX（23h）

说明：SINMUX 命令用于更改当前状态机的输入源。如果未执行 SINMUX 命令，则自动选择加速度计信号作为默认输入源。

SINMUX 命令也可用于经 MLC 滤波的数据，为此，MLC 滤波器的结构必须采用以下配置：

- 必须将第一个 MLC 滤波器[F_x F_y F_z F_v(2)]应用于传感器轴。
- 第二个 MLC 滤波器[0 0 0 G_v(3)]必须应用于传感器范数；
- 第三个 MLC 滤波器[H_x H_y H_z H_v(2)]必须用于传感器轴。
- 第四个 MLC 滤波器[0 0 0 J_v(3)]必须用于传感器范数。

注意：如果用户只需将两个滤波器应用于传感器轴上（不需要传感器范数上的滤波器），则即使不使用，也需要配置传感器范数上的第二个 MLC 滤波器。此外，如果用户只需将两个滤波器应用于传感器范数（不需要传感器轴上的滤波器），则必须配置上述全部四个 MLC 滤波器。

参数：一个字节。

- 第 1 个参数：用于选择以下输入源的值：
  - 0：加速度计[a_x a_y a_z a_v]；
  - 1：陀螺仪[g_x g_y g_z g_v]；
  - 2：校准过的磁力计[m_x m_y m_z m_v]；
  - 3：来自机器学习内核(1)的第一个滤波信号[F_x F_y F_z F_v(2)]；
  - 4：来自机器学习内核(1)的第三个滤波信号[H_x H_y H_z H_v(2)]；
  - 5：来自机器学习内核(1)的第二个滤波信号范数[0 0 0 G_v(3)]；
  - 6：来自机器学习内核(1)的第四个滤波信号范数[0 0 0 J_v(3)]；
  - 7：陀螺仪积分信号[d_x d_y d_z d_v]。

动作：

- 根据设置参数相应地选择输入信号。根据所选输入源信号（可能为 000b、001b、010b、011b、100b、101b、110b 或 111b）配置固定数据部分的 SETTINGS(IN_SEL[2:0])位；
- PP = PP + 2。

(1) 滤波器类型可能为 HP / LP / IIR1 / IIR2，具体取决于机器内核配置。

(2) F_v（和 H_v）由 FSM 在内部计算，并从 MLC 提供的 F_x、F_y 和 F_z（或 H_x、H_y 和 H_z）滤波数据值开始计算。

(3) G_v（和 J_v）由 MLC 提供。

7.2.22 STIMER3（24h）

说明：STIMER3 命令用于为 TIMER3 设置新值。

参数：一个字节。

- 第 1 个参数：新的 TIMER3 值。

动作：

- 设置新的 TIMER3 值；
- PP = PP + 2。

7.2.23 STIMER4（31h）

说明：STIMER4 命令用于为 TIMER4 设置新值。

参数：一个字节。

- 第 1 个参数：新的 TIMER4 值。

动作：

- 设置新的 TIMER4 值；
- PP = PP + 2。

7.2.24 SWAPMSK（32h）

说明：SWAPMSK 命令用于替换 MASKA 和 MASKB 选择。MASKC 不受影响。

参数：无。

动作：

- 将 MASKA 切换为 MASKB；
- PP = PP + 1。

7.2.25 INCR（34h）

说明：INCR 命令用于在 FSM_LONG_COUNTER_CLEAR (4Ah)寄存器的 FSM_LC_CLEAR 位置“1”时或在长计数器值加 1 时重置长计数器。长计数器值存储在 FSM_LONG_COUNTER_L (48h)和 FSM_LONG_COUNTER_H (49h)寄存器中。

参数：无。

动作：

- 如果将 FSM_LONG_COUNTER_CLEAR (4Ah)寄存器的 FSM_LC_CLEAR 位置“1”，或将长计数器值加 1，则重置长计数器值；
- PP = PP + 1。

7.2.26 JMP（41h）

说明：JMP 命令是以“NEXT1 | NEXT2”条件为特征的特殊命令，具有两个不同的跳转地址。

参数：三个字节。

- 第 1 个参数：NEXT1 | NEXT2 条件；
- 第 2 个参数：如果 NEXT1 条件为真，则跳转地址；
- 第 3 个参数：如果 NEXT2 条件为真，则跳转地址。

在 NEXT2 条件之前评估 NEXT1 条件。跳转地址与当前状态机有关（地址 00h 是指 CONFIG_A 字节）。

动作：

- 将固定数据部分的 CONFIG_B(JMP)位设置为“1”。评估“NEXT1 | NEXT2”条件：
  - 如果“NEXT1”条件为真，则 PP = 第 2 个参数条件；
  - 否则，如果“NEXT2”条件为真，则 PP = 第 3 个参数条件；
  - 否则，等待新的样本集合并再次评估“NEXT1 | NEXT2”条件。

7.2.27 CANGLE（42h）

说明：CANGLE 命令用于清除陀螺仪积分值。如果执行此命令，则在下一个条件为真（默认行为）时，不再清除角度积分值，但以下情况例外：

- 每次执行 CANGLE 命令时（新样本到达时）；
- 如果复位条件为真。

参数：无。

动作：

- 清除角度值。
- PP = PP + 1。

7.2.28 SMA（43h）

说明：SMA 命令用于为 MASKA 和 TMASKA 设置新值。

参数：一个字节。

- 第 1 个参数：新的 MASKA 和 TMASKA 值。

动作：

- 设置新的 MASKA 和 TMASKA 值；
- PP = PP + 2。

7.2.29 SMB（DFh）

说明：SMB 命令用于为 MASKB 和 TMASKB 设置新值。

参数：一个字节。

- 第 1 个参数：新的 MASKB 和 TMASKB 值。

动作：

- 设置新的 MASKB 和 TMASKB 值；
- PP = PP + 2。

7.2.30 SMC（FEh）

说明：SMC 命令用于为 MASKC 和 TMASKC 设置新值。

参数：一个字节。

- 第 1 个参数：新的 MASKC 和 TMASKC 值。

动作：

- 设置新的 MASKC 和 TMASKC 值；
- PP = PP + 2。

7.2.31 SCTC0（5Bh）

说明：SCTC0 命令用于在 NEXT 条件为真（默认行为）时复位 TC 字节（时间计数器）。

参数：无。

动作：

- TC（时间计数器）字节值在有效 NEXT 条件后复位；
- PP = PP + 1。

7.2.32 SCTC1（7Ch）

说明：SCTC1 命令用于在 NEXT 条件为真时保留 TC 字节（时间计数器）。

参数：无。

动作：

- TC（时间计数器）字节值在有效 NEXT 条件后不会改变；
- PP = PP + 1。

7.2.33 UMSKIT（C7h）

说明：UMSKIT 命令用于在更新 OUTS 寄存器值（默认行为）时取消屏蔽中断生成。有关中断生成的更多信息，请参见 OUTC / CONT / CONTREL 命令。

参数：无。

动作：

- 设置 OUTS 寄存器时取消屏蔽中断生成；
- PP = PP + 1。

7.2.34 MSKITEQ（EFh）

说明：MSKITEQ 命令用于在 OUTS 寄存器值更新但其值不变的情况下（临时掩码值等于当前 OUTS 寄存器值）屏蔽中断生成。有关中断生成的更多信息，请参见 OUTC / CONT / CONTREL 命令。

参数：无。

动作：

- 如果 OUTS 不变，则在设置 OUTS 寄存器时屏蔽中断生成。
- PP = PP + 1。

7.2.35 MSKIT（F5h）

说明：MSKIT 命令用于在 OUTS 寄存器值更新时屏蔽中断生成。有关中断生成的更多信息，请参见 OUTC / CONT / CONTREL 命令。

参数：无。

动作：

- 设置 OUTS 寄存器时屏蔽中断生成。
- PP = PP + 1。

8 FSM 配置示例

本节包含一个示例，该示例解释了配置 LSM6DSOX FSM 必须完成的所有写操作。必须遵循以下几个步骤：

- 配置嵌入式功能寄存器组中的 FSM 寄存器；
- 配置嵌入式高级功能寄存器组中的 FSM 寄存器；
- 配置 LSM6DSOX 传感器（加速度计和/或陀螺仪）。

在该示例中，配置了两个简单程序：

- 程序 1：手腕倾斜（绕 x 轴）算法，发送至 INT1 引脚；
- 程序 2：唤醒算法，发送至 INT2 引脚。

两种算法均旨在仅以 26 Hz 的采样率使用加速度计数据。
有关程序数据部分和指令部分的详细信息，请参见下图。

图 13. FSM 配置示例

	PAGE - ADDRESS	NAME	7	6	5	4	3	2	1	0
PROGRAM 1	4 - 00h	CONFIG A	01 (1 threshold)		01 (1 mask)		00		01 (1 short timer)	
	4 - 01h	CONFIG B	0	0	0	0	0	0	0	0
	4 - 02h	SIZE	10h (16 bytes)							
	4 - 03h	SETTINGS	00		0	0	0	00		
	4 - 04h	RESET POINTER	00h							
	4 - 05h	PROGRAM POINTER	00h							
	4 - 06h	THRESH1	B7AEh (-0.480)							
	4 - 07h									
	4 - 08h	MASKA	80h (+X)							
	4 - 09h	TMASKA	00h							
	4 - 0Ah	TC	00h							
	4 - 0Bh	TIMER3	10h (16 samples)							
	4 - 0Ch	GNTH1 | TI3	53h							
	4 - 0Dh	OUTC	99h							
	4 - 0Eh	GNTH1 | NOP	50h							
	4 - 0Fh	STOP	00h							
PROGRAM 2	4 - 10h	CONFIG A	01 (1 threshold)		01 (1 mask)		00		00	
	4 - 11h	CONFIG B	0	0	0	0	0	0	0	0
	4 - 12h	SIZE	0Ch (12 bytes)							
	4 - 13h	SETTINGS	00		0	0	0	00		
	4 - 14h	RESET POINTER	00h							
	4 - 15h	PROGRAM POINTER	00h							
	4 - 16h	THRESH1	3C66h (1.100)							
	4 - 17h									
	4 - 18h	MASKA	02h (+V)							
	4 - 19h	TMASKA	00h							
	4 - 1Ah	NOP | GNTH1	05h							
	4 - 1Bh	CONTREL	22h							

必须在加速度计和陀螺仪传感器均处于掉电模式下的情况下执行 FSM 配置。有关完整的设备配置，请参考以下脚本：

1. 将 00h 写入寄存器 10h // 将加速度计传感器设为下电模式
2. 将 00h 写入寄存器 11h // 将陀螺仪传感器设为下电模式

3. 将 80h 写入寄存器 01h // 使能对嵌入功能寄存器的访问
4. 将 01h 写入寄存器 05h // EMB_FUNC_EN_B(FSM_EN) = '1'
5. 将 4Bh 写入寄存器 5Fh // EMB_FUNC_ODR_CFG_B (FSM_ODR) = '01' (26Hz)
6. 将 03h 写入寄存器 46h // FSM_ENABLE_A = '03h'
7. 将 00h 写入寄存器 47h // FSM_ENABLE_B = '00h'
8. 将 01h 写入寄存器 0Bh // FSM_INT1_A = '01h'
9. 将 00h 写入寄存器 0Ch // FSM_INT1_B = '00h'
10. 将 02h 写入寄存器 0Fh // FSM_INT2_A = '02h'
11. 将 00h 写入寄存器 10h // FSM_INT2_B = '00h'
12. 将 40h 写入寄存器 17h // PAGE_RW：使能写操作
13. 将 11h 写入寄存器 02h // 使能对嵌入式高级功能寄存器的访问，PAGE_SEL = 1
14. 将 7Ah 写入寄存器 08h // PAGE_ADDRESS = 7Ah
15. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_LONG_COUNTER_L
16. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_LONG_COUNTER_H
17. 将 02h 写入寄存器 09h // 将 02h 写入寄存器 FSM_PROGRAMS
18. 将 02h 写入寄存器 09h // 执行空写入，以增加写入地址
19. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_START_ADDRESS_L
20. 将 04h 写入寄存器 09h // 将 04h 写入寄存器 FSM_START_ADDRESS_H
21. 将 41h 写入寄存器 02h // PAGE_SEL = 4
22. 将 00h 写入寄存器 08h // PAGE_ADDRESS = 00h
23. 将 51h 写入寄存器 09h // CONFIG_A
24. 将 00h 写入寄存器 09h // CONFIG_B
25. 将 10h 写入寄存器 09h // SIZE
26. 将 00h 写入寄存器 09h // SETTINGS
27. 将 00h 写入寄存器 09h // RESET POINTER
28. 将 00h 写入寄存器 09h // PROGRAM POINTER
29. 将 AEh 写入寄存器 09h // THRESH1 LSB
30. 将 B7h 写入寄存器 09h // THRESH1 MSB
31. 将 80h 写入寄存器 09h // MASKA
32. 将 00h 写入寄存器 09h // TMASKA
33. 将 00h 写入寄存器 09h // TC
34. 将 10h 写入寄存器 09h // TIMER3
35. 将 53h 写入寄存器 09h // GNTH1 | TI3
36. 将 99h 写入寄存器 09h // OUTC
37. 将 50h 写入寄存器 09h // GNTH1 | NOP
38. 将 00h 写入寄存器 09h // STOP（必须包含偶数个 SIZE 字节）
39. 将 50h 写入寄存器 09h // CONFIG_A
40. 将 00h 写入寄存器 09h // CONFIG_B
41. 将 0Ch 写入寄存器 09h // SIZE
42. 将 00h 写入寄存器 09h // SETTINGS
43. 将 00h 写入寄存器 09h // RESET POINTER
44. 将 00h 写入寄存器 09h // PROGRAM POINTER
45. 将 66h 写入寄存器 09h // THRESH1 LSB

46. 将 3Ch 写入寄存器 09h // THRESH1 MSB
47. 将 02h 写入寄存器 09h // MASKA
48. 将 00h 写入寄存器 09h // TMASKA
49. 将 05h 写入寄存器 09h // NOP | GNTH1
50. 将 22h 写入寄存器 09h // CONTREL
51. 将 01h 写入寄存器 02h // 禁用对嵌入式高级功能寄存器的访问，PAGE_SEL = 0
52. 将 00h 写入寄存器 17h // PAGE_RW：禁用写操作
53. 将 00h 写入寄存器 01h // 禁用对嵌入功能寄存器的访问
54. 将 02h 写入寄存器 5Eh // MD1_CFG(INT1_EMB_FUNC) = '1'
55. 将 02h 写入寄存器 5Fh // MD2_CFG(INT2_EMB_FUNC) = '1'
56. 将 20h 写入寄存器 10h // CTRL1_XL = '20h' (26 Hz, ±2 *g*)

9 启动例程

启用 FSM 后，将自动执行启动例程。该例程执行以下任务：

- 将 CONFIG_B(STOPDONE)和 CONFIG_B(JMP)位复位；
- 将 PP 和 RP 指针初始化到代码第一行；
- 将 SETTINGS 字段初始化为默认值 0x20，即：
  - MASKSEL = ‘00’；
  - SIGNED = ‘1’；
  - R_TAM = ‘0’；
  - THRS3SEL = ‘0’；
  - IN_SEL = ‘000’。
- 清除相关输出寄存器 OUTS；
- 为所有声明的临时掩码分配相应的原始掩码值($TMASK_x = MASK_x$)；
- 如果声明了定时器，则将时间计数器初始化为 0 (TC = 0)；
- 如果声明了抽取因子，则使用编程的抽取时间值(DESC = DEST)初始化抽取计数器；
- 如果声明了先前的轴符号，则将其初始化为 0 (PAS = 0)；
- 如果声明了陀螺仪角度计算，则将四个角度初始化为 0 (DX = DY = DZ = DV = 0)；
- 如果 CONFIG_B(LC)有效，则长计数器复位。

执行启动例程时，程序始终从已知状态启动，这与停止方式无关。但是，应注意，默认模式表示：

- 将 MASKA 选作运行掩码(MASKSEL = ‘00’)；
- 有符号比较模式(SIGNED = ‘1’)；
- 下一个条件为真后(R_TAM = ‘0’)，不释放临时掩码；
- 选择阈值 1 代替阈值 3 进行比较(THRS3SEL = ‘0’)；
- 输入多路复用器设置，以选择加速度计数据(IN_SEL = ‘000’)。

10 状态机配置示例

10.1 翻转

翻转是每隔 n 个采样产生一次中断的简单状态机配置。
该理念是使用定时器对 n 个采样进行计数。

图 14. 翻转状态机示例

BYTE #	NAME	7	6	5	4	3	2	1	0
00h	CONFIG A	00		00		00		01 (1 short timer)	
01h	CONFIG B	0	0	0	0	0	0	0	0
02h	SIZE	0Ah (10 bytes)							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h	TC	00h							
07h	TIMER3	10h (16 samples)							
08h	NOP | TI3	03h							
09h	CONTREL	22h							

指令部分说明

PP = 08h：第一次达到此状态时，TC = TI3。每次生成新样本集合时，TC 字节都会减 1。当 TC = 0 时，PP = PP + 1。

PP = 09h：无需样本集即可执行 CONTREL 命令：这将产生中断并使程序复位(PP = RP = 08h)。

在该示例中，每 16 次采样产生一个中断。可通过配置 TI3 获得所需的触发周期，该周期取决于所配置的 FSM_ODR。

10.2 唤醒

对于超低功耗应用，最好有一个在移动后唤醒系统的中断信号。
该理念是使用 1.0 *g* 的标称重力值，并对标称重力值施加一点迟滞。

图 15. 唤醒状态机示例

BYTE #	NAME	7	6	5	4	3	2	1	0
00h	CONFIG A	01 (1 threshold)		01 (1 mask)		00		00	
01h	CONFIG B	0	1	0	0	0	0	0	0
02h	SIZE	12h (18 bytes)							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h	THRESH1	3C00h (1.000)							
07h									
08h	HYST	27AEh (0.030)							
09h									
0Ah	MASKA	02h (+V)							
0Bh	TMASKA	00h							
0Ch	JMP	41h							
0Dh	LNTH1 | GNTH1	75h							
0Eh	10h	10h (jump address if LNTH1 condition is true)							
0Fh	10h	10h (jump address if GNTH1 condition is true)							
10h	CONTREL	22h							
11h	STOP	00h							

指令部分说明
PP = 0Ch：无需设置样本即可执行的 JMP 命令：CONFIG_B(JMP)位被置“1”。PP = PP + 1。
PP = 0Dh：对阈值 1 执行双重条件（默认情况下选择 MASKA）。由于使用了迟滞，因此比较阈值为：

- COND1 (LNTH1): THRESH1 – HYST。跳转地址为 10h；
- COND2 (GNTH1): THRESH1 + HYST。跳转地址为 10h。

当向量（幅度）位于迟滞区域之外时（上述条件之一为真），PP 被设为地址 10h。
PP = 10h：无需样本集即可执行 CONTREL 命令：这将产生中断并使程序复位(PP = RP = 0Ch)。
在示例中，唤醒阈值为 1.0 *g* ± 30 mg。配置迟滞值时，应考虑加速度计偏移。

10.3 自由落体

该功能用于检测系统何时落下（例如，用于保护硬盘驱动器上的数据）。如果物体处于自由落体状态，则 X 轴、Y 轴和 Z 轴的加速度将为零。

要实现此功能，所有轴上的加速度应小于所配置阈值，并保持最小配置持续时间。检测到这一情况时，将产生一个中断。

图 16. 自由落体状态机示例

BYTE #	NAME	7	6	5	4	3	2	1	0
00h	CONFIG A	01 (1 threshold)		01 (1 mask)		00		01 (1 short timer)	
01h	CONFIG B	0	0	0	0	0	0	0	0
02h	SIZE	12h (18 bytes)							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h	THRESH1	34CDh (0.300)							
07h									
08h	MASKA	A8h (+X, +Y, +Z)							
09h	TMASKA	00h							
0Ah	TC	00h							
0Bh	TIMER3	03h (3 samples)							
0Ch	SSIGN0	12h							
0Dh	SRP	33h							
0Eh	GNTH1 | TI3	53h							
0Fh	OUTC	99h							
10h	GNTH1 | NOP	50h							
11h	STOP	00h							

指令部分说明

PP = 0Ch：无需样本集合即可执行 SSIGN0 命令：将 SETTINGS(SIGNED)位置“0”，表示已设置无符号比较模式。PP = PP + 1。

PP = 0Dh：无需样本集合即可执行 SRP 命令：将 RESET POINTER 设为下一个状态，0Eh。PP = PP + 1。

PP = 0Eh：如果一个轴上的加速度大于 THRESH1，则 PP = RP。如果连续 3 个样本的所有轴上的加速度均小于 THRESH1，则 PP 增加(PP = PP + 1)。

PP = 0Fh：无需样本集合即可执行 OUTC 命令：这会产生一个中断并增加 PP (PP = PP + 1)。

PP = 10h：如果一个轴上的加速度大于 THRESH1，则 PP = RP。这意味着设备不再处于自由落体状态，因此必须重置程序。

在该例中，将自由落体阈值设为 0.3 *g*，并将自由落体持续时间设为 3 个样本。

注意：自由落体持续时间与 FSM_ODR 严格相关：例如，如果将 FSM_ODR 设为 26 Hz，则自由落体持续时间为 115 毫秒（3 个样本/ 26 Hz）。

10.4 决策树接口

本示例说明了决策树接口如何与 FSM 一起使用。假定机器学习内核的配置如下：

- 0 号决策树（第一个决策树）实现了一种活动识别算法，该算法能够检测三种用户活动（类别）：静止、行走和跑步；
- 输出值与每个识别的活动有关：
  - “静止”为 0；
  - “行走”输出为 1；
  - “跑步”输出为 2；
- 用于特征计算的窗口长度为 2 秒（ODR 等于 26 Hz 的 52 个样本）

FSM 实现了一种简单的唤醒算法，该算法在决策树输出等于“静止”后启用。在这种情况下，如果设备再次开始移动，FSM 会产生唤醒中断。

图 17. 决策树接口示例

BYTE #	NAME	7	6	5	4	3	2	1	0
00h	CONFIG A	01 (1 threshold)		01 (1 mask)		00		10 (2 short timer)	
01h	CONFIG B	0	0	0	0	1	0	0	0
02h	SIZE	12h (18 bytes)							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h 07h	THRESH1	3C33h (1.050)							
08h	MASKA	02h (+V)							
09h	TMASKA	00h							
0Ah	TC	00h							
0Bh	TIMER3	02h (2 samples)							
0Ch	TIMER4	35h (53 samples)							
0Dh	DECTREE	00h (selected decision tree number ‘0’, expected output is ‘0’)							
0Eh	NOP | CHKDT	0Fh							
0Fh	TI3 | GNTH1	35h							
10h	OUTC	99h							
11h	TI4 | NOP	40h							

指令部分说明

PP = 0Eh：根据 DECTREE 字节检查决策树输出。配置 DECTREE 字节，以检查决策树编号“0”，并期望输出等于“0”（即“静止”）。如果检测到的活动为“静止”，则 PP 会增加(PP = PP + 1)。

PP = 0Fh：如果 TI3 到期，则 PP = RP（重置程序并再次检查决策树接口）。如果加速度计的向量（幅度）大于 THRESH1，则 PP 增加(PP = PP + 1)。

PP = 10h：无需样本集合即可执行 OUTC 命令：这会产生一个中断并增加 PP (PP = PP + 1)。

PP = 11h：如果 TI4 到期，则 PP = RP。

11 有限状态机工具

通过专用工具可以实现设备中的有限状态机可编程性，该专用工具可作为 Unico GUI 的扩展提供。

11.1 Unico GUI

Unico 是意法半导体产品组合中提供的所有 MEMS 传感器演示板的图形用户界面。它可以与基于 STM32 微控制器（Professional MEMS Tool）的主板进行交互，从而可以在 MEMS 传感器和 PC GUI 之间进行通信。

有关 Professional MEMS Tool 板的详细信息，请参见 STEVAL-MKI109V3。

Unico GUI 提供用于所支持的三种操作系统的三种软件包。

- Windows
  - STSW-MKI109W
- Linux
  - STSW-MKI109L
- Mac OS X
  - STSW-MKI109M

Unico GUI 允许以图形和数字格式显示传感器输出，并允许用户保存或广泛管理来自器件的数据。

Unico 允许访问 MEMS 传感器寄存器，从而实现寄存器设置的快速原型设计，以及直接在器件中轻松测试配置。可将当前寄存器的配置保存在文本文件中，并从现有文件中加载配置。这样就可以在几秒内对传感器进行重新编程。

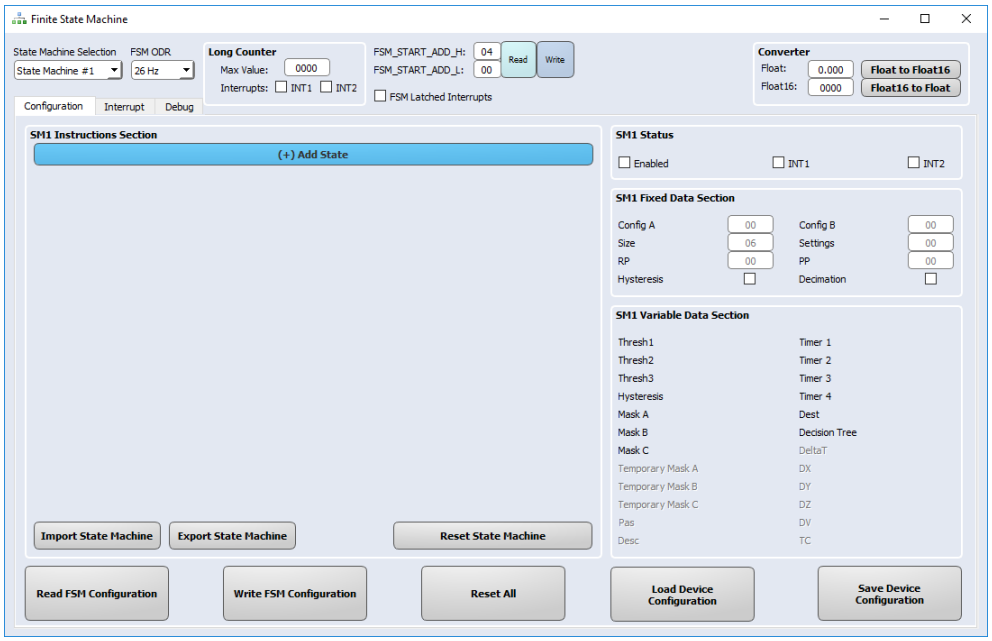
Unico GUI 中提供的有限状态机工具可通过自动生成器件的配置文件来帮助处理寄存器配置。通过点击几个按钮，可使用配置文件。用户可通过这些配置文件创建自己的器件配置库。

要执行有限状态机工具，用户必须点击专用的“FSM”按钮，该按钮位于 UNICO GUI 主窗口的左侧，如下图所示。

图 18. 运行有限状态机工具

加载后，将显示有限状态机工具的主窗口。

图 19. 有限状态机工具



在有限状态机工具主窗口的顶部，用户可以选择选定哪个状态机（该选择同时应用于配置标签和调试标签）。还可以配置 FSM ODR、长计数器参数和 FSM 锁存中断。FSM 起始地址由 Unico 工具自动管理，用户不应更改。最后，可以使用从 float32 到 float16 格式的转换器，反之亦然。转换器用于生成要在变量数据部分的阈值资源中设置的值。

有限状态机工具主要由三个标签组成，将在专门部分中解释这三个标签：

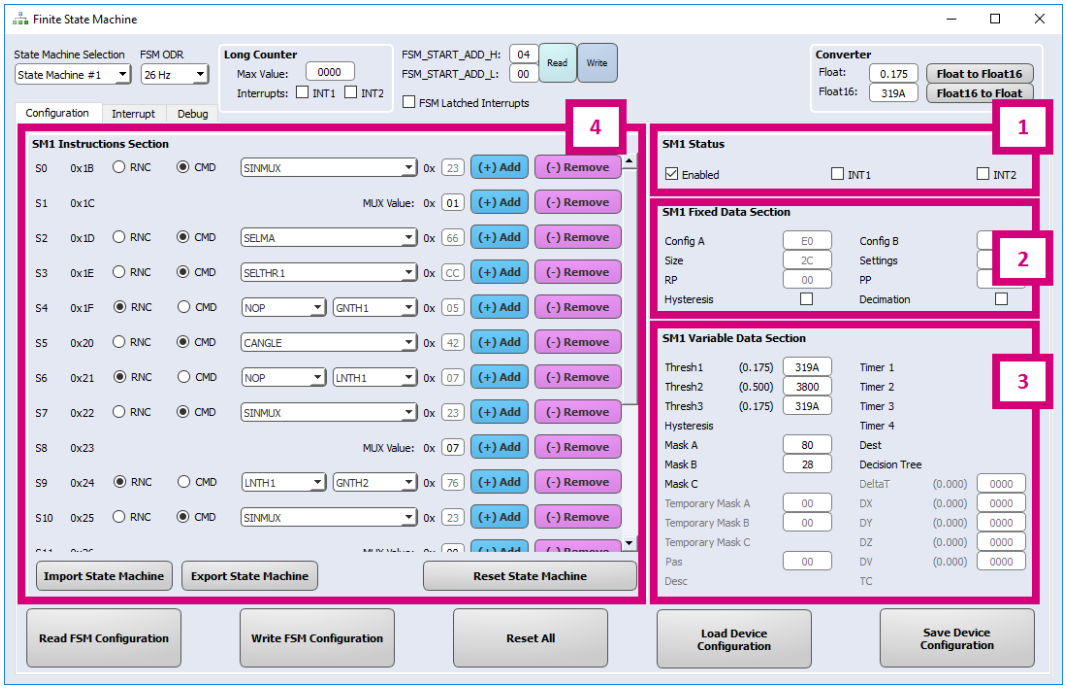
- 配置标签（默认选中的标签）；
- 中断标签；
- 调试标签。

11.1.1 配置标签

有限状态机工具的配置标签允许用户实现程序逻辑。UI 能够抽象 FSM 程序结构：为此，显示 4 个组框：

1. SM_x 状态；
2. SM_x 固定数据部分；
3. SM_x 可变数据部分；
4. SM_x 指令部分。

图 20. 有限状态机工具 - 配置标签



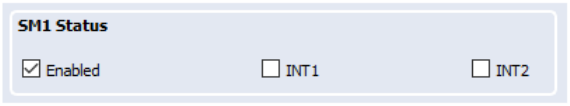
在配置标签的底部，用户可以使用专用按钮管理设备配置：

- 读取 FSM 配置：用于读取 FSM 寄存器并基于当前 FSM 配置和程序以图形方式构建 UI；
- 写入 FSM 配置：用于写入整个 FSM 配置（包括 FSM ODR、长计数器参数、中断状态和程序）；
- 全部复位：用于复位整个有限状态机工具 UI；
- 加载设备配置：用于加载.ucf 文件；
- 保存设备配置：用于生成.ucf 文件，其中包括传感器和 FSM 寄存器配置。

11.1.1.1 *SMx 状态*

SMx 状态组框位于配置标签的右上角。

图 21. 配置标签 - SMx 状态



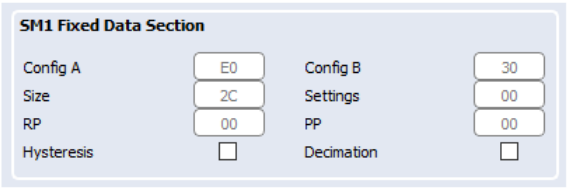
SMx 状态组框允许用户启用/禁用状态机并将中断状态发送到 INT1/INT2 引脚上。具体而言：

- “Enabled”复选框用于启用/禁用状态机。如果程序至少包含一条指令，则自动置位，如果程序不包含任何指令，则将自动复位；
- “INT1”复选框用于使能向 INT1 引脚发送状态机中断。如果将 MD1_CFG (5Eh)的 INT1_EMB_FUNC 位置“1”，则有效；
- “INT2”复选框用于使能向 INT2 引脚发送状态机中断。如果将 MD2_CFG (5Fh)的 INT2_EMB_FUNC 位置“1”，则有效。

11.1.1.2 *SMx 固定数据部分*

SMx 固定数据部分组框在配置标签的右侧。

图 22. 配置标签 - SMx 固定数据部分

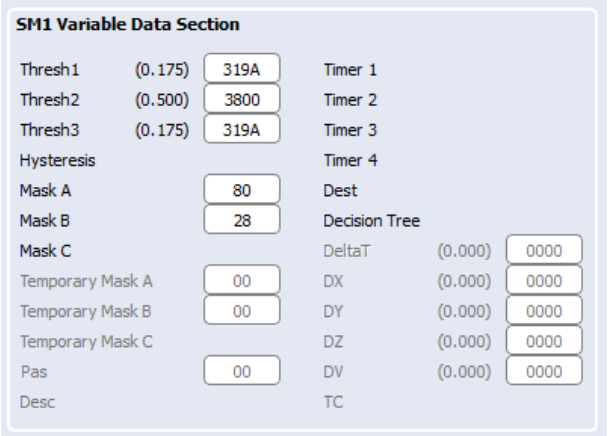


SMx 固定数据部分组框使用户可以获取有关程序的固定数据部分字节的信息。这些字节由有限状态机工具自动管理。也可以根据用户需要启用/禁用迟滞和抽取资源。如果启用，则相应资源将显示在 SMx 可变数据部分组框中。

11.1.1.3 *SM_x 可变数据部分*

SM_x 可变数据组框位于配置标签的右下角。

图 23. 配置标签 – SM_x 可变数据部分



SM_x 可变数据组框可简化资源分配过程：根据构成 SM_x 指定部分的指令，所有需要的资源均会自动显示或隐藏在 SM_x 可变数据部分组框中。用户只需设置显示资源的值。

11.1.1.4 *SM_x 指令部分*

SM_x 指令部分组框位于配置标签的左侧。

图 24. 配置标签 – SM_x 指令部分

SM_x 指令部分组框可帮助用户构建算法逻辑。SM_x 可变数据部分组框根据 SM_x 指令部分组框中使用的资源动态更新。在 SM_x 指令部分组框中，可执行更多操作：

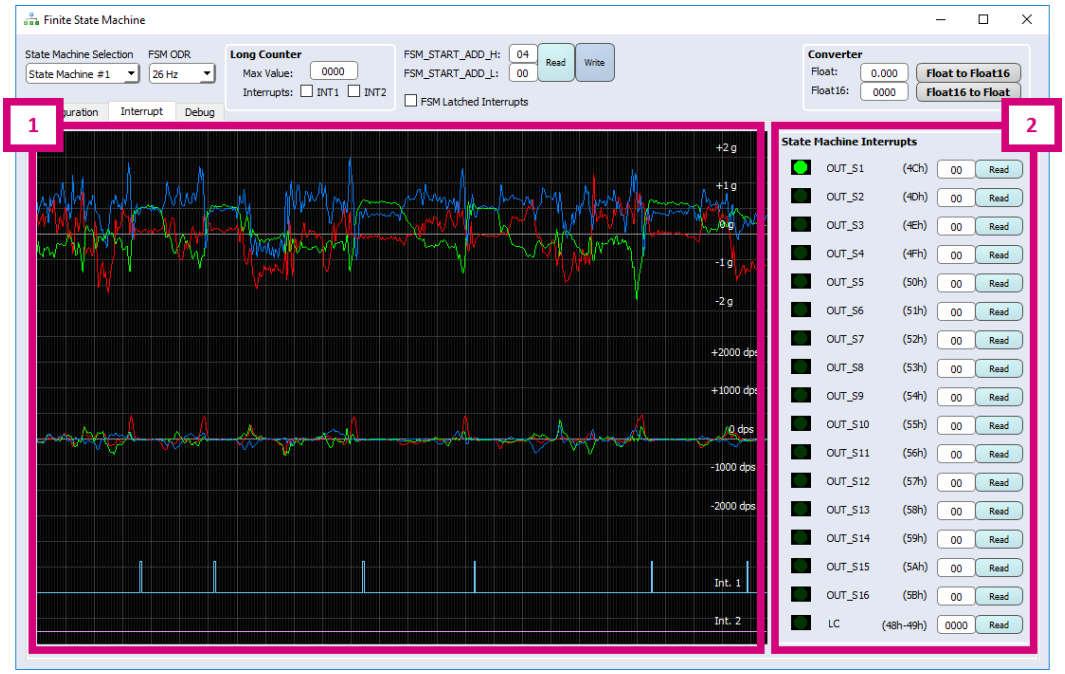
1. 自定义现有状态。单个状态包括：
   - 状态编号 S_x
   - 状态程序的相对十六进制地址（地址 0x00 对应于固定数据部分中的 CONFIG_A 字节）
   - 状态类型和操作码：用户可以使用如下所述的单选按钮和下拉列表来定制状态：
     - “RNC”单选按钮：状态为 RESET/NEXT 条件。在这种情况下，显示两个下拉列表。左边的下拉列表与 RESET 条件有关，而右边的下拉列表与 NEXT 条件有关。
     - “CMD”单选按钮：状态为命令。在这种情况下，将显示一个下拉列表。具有一个或多个参数（通过工具自动显示）的命令需要用户手动配置参数值。
   - “Add”按钮用于在当前状态之前插入新状态；
   - “Remove”按钮用于删除当前状态。
2. “Add State”按钮用于在状态机末尾添加新状态。此按钮始终位于状态机状态的底部；
3. “Import / Export State Machine”按钮用于以.fsm 格式导入/导出状态机程序。.fsm 格式用于使用户能够通过一组.fsm 状态机程序构建整个 FSM 配置。
4. “Reset State Machine”按钮用于使状态机指令部分复位（仅在 UI 上，不在设备中）。

11.1.2 中断标签

有限状态机工具的中断标签允许用户在程序逻辑运行时检查所配置程序的功能。UI 由两部分组成，如图 25 中所示。

1. 信号图：根据使能的传感器和中断配置显示加速度计、陀螺仪和中断信号图。
2. 状态机中断状态：在该组框中，显示两列信息：
   – 图形中的绿色 LED 与相应状态机中断源位有关。默认情况下，LED 熄灭。将相应的源位置“1”时，LED 点亮约 300 毫秒；
   – 点击相应的“Read”按钮，可手动读取 OUT_Sx 寄存器值和长计数器寄存器值。

图 25. 有限状态机工具 - 中断标签

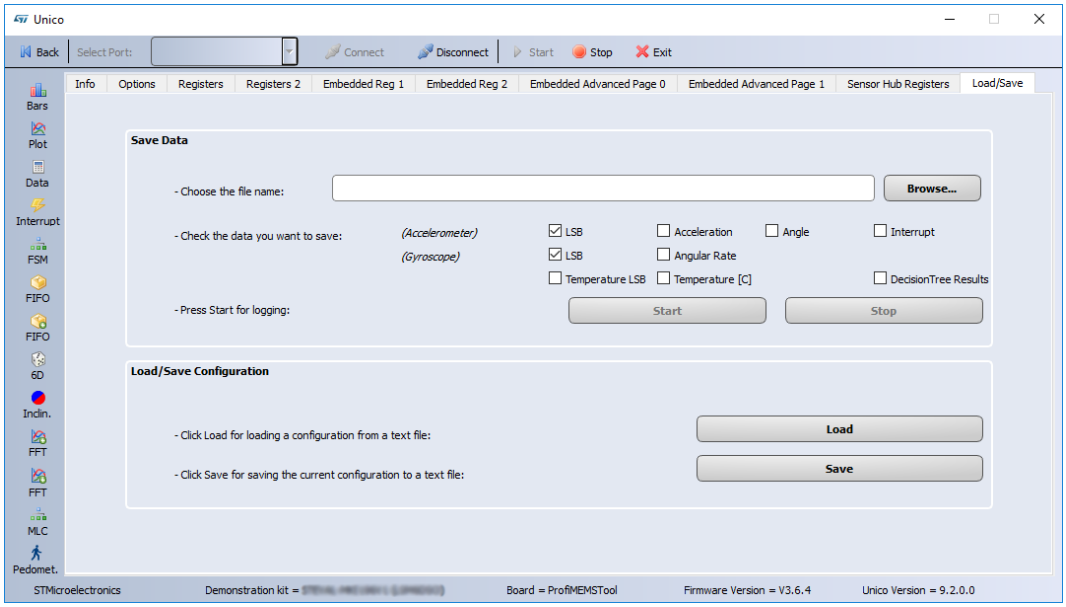


11.1.3 调试标签

调试标签可用于将数据注入设备，以检查所配置程序的功能。

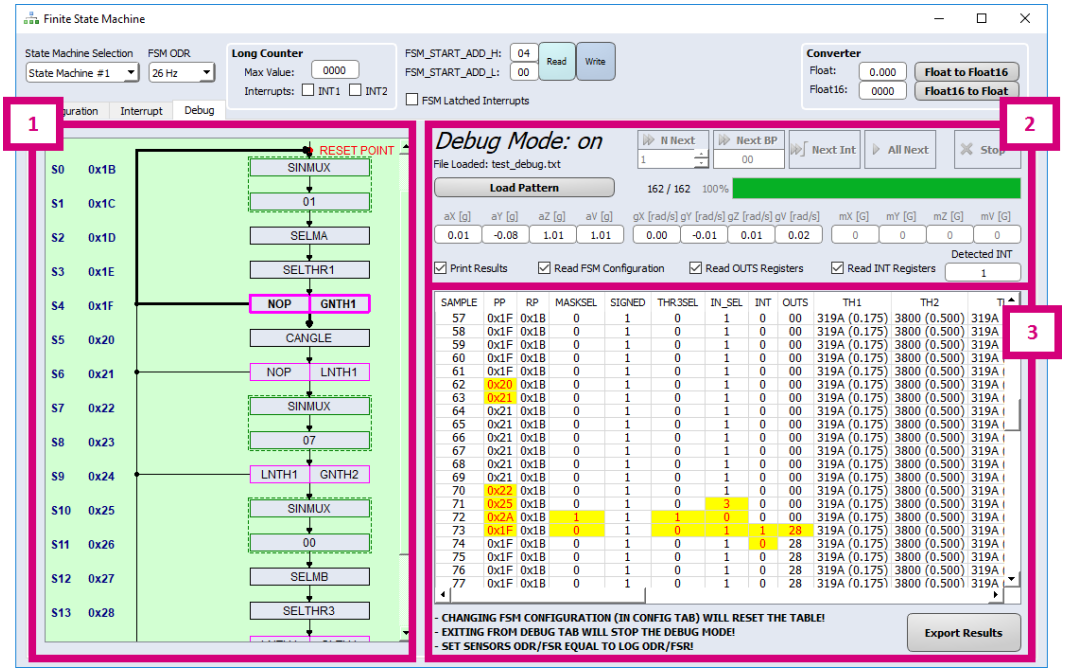
UNICO GUI 载入/保存标签（如下图所示）允许用户获取正确格式化的数据注入日志文件：这些日志文件必须只包含[LSB]数据（加速度计和/或陀螺仪，具体取决于用户需求和程序逻辑）。

图 26. **UNICO GUI – 载入/保存标签**



调试标签窗口如下图所示。

图 27. **有限状态机工具 – 调试标签**



调试标签主要由三个 UI 部分组成：

1. 状态机流程：状态机流程在此处以图形方式显示。启用调试模式后，当前状态将高亮显示，并根据注入示例和程序行为动态更新。

2. 调试指令：默认情况下，调试模式为关闭。加载日志文件后，自动打开调试模式，用户可以开始向设备注入数据，以验证程序功能。注入的样本数据和检测到的中断数如此处所示。
3. 输出结果：将样本注入设备后，根据“打印结果”复选框的状态将新行添加到表中。表列代表状态机参数和资源，而表行与注入的样本的有关。参数或资源值改变时，将突出显示相应的单元格。最后，可以以文本文件格式导出表结果。

版本历史

表 54. 文档版本历史

日期	版本	变更
2019 年 1 月 28 日	1	初始版本
2019 年 2 月 1 日	2	更新了“SINMUX（23h）”
2019 年 8 月 13 日	3	在 3.1.2.17 节“EMB_FUNC_ODR_CFG_B（5Fh）”中添加了有关 FSM ODR 的注释 更新了第 3.1.3.13 节“FSM_START_ADD_L (7Eh)和 FSM_START_ADD_H (7Fh)” 更新了第 3.2.1 节“输出选择器块” 更新了第 4 节“FSM 中断” 更新了第 5.1 节“长计数器” 更新了第 6 节“可变数据部分” 更新了第 6.1 节“阈值” 更新了第 6.4 节“角度计算” 更新了第 6.8 节“决策树接口” 更新了 7.2.21 节“SINMUX（23h）” 更新了第 8 节“FSM 配置示例”中的脚本 添加了第 10.4 节“决策树”接口
2019 年 10 月 2 日	4	更新了 第 6.7 节 前一个轴符号 更新了 第 6.8 节 决策树接口 更新了 第 11.1 节 Unico GUI 更新了 图 20. 有限状态机工具 - 配置标签 更新了 图 23. 配置标签 – SM_x 可变数据部分 更新了 图 25. 有限状态机工具 - 中断标签 更新了 第 11.1.3 节 调试标签

目录

表一览

图一览