LSM6DSV16X：有限状态机

简介

本文档旨在介绍如何使用和配置 ST 的 LSM6DSV16X 中嵌入的有限状态机。

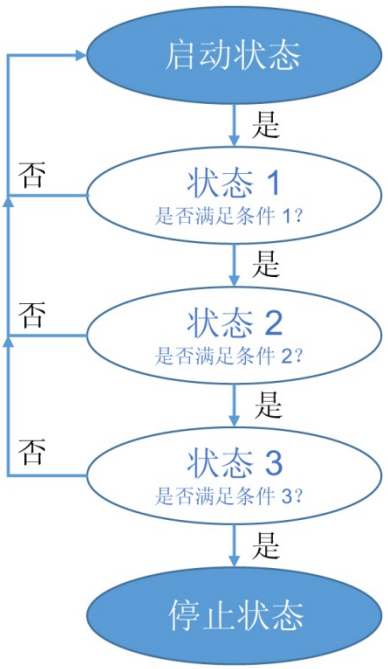
LSM6DSV16X 可配置为由用户定义的运动模式激活中断信号的生成。为此，最多可以为运动检测独立编程八组嵌入有限状态机。

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。它是一种行为模式，由有限数量的状态和状态之间的转换组成，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（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 LSM6DSV16X 中的有限状态机

LSM6DSV16X 作为加速度计-陀螺仪二合一传感器，生成加速度和角速率输出数据。此外，还可以使用传感器集线器功能（模式 2）连接外部传感器，如磁力计或压力传感器等。这些数据均可用作嵌入有限状态机中最多八组程序的输入（请参见下图）。

图 2. **LSM6DSV16X 中的状态机**

FSM 采用高度模块化的结构：可轻松编写多达八组程序，每组程序均可识别特定手势。

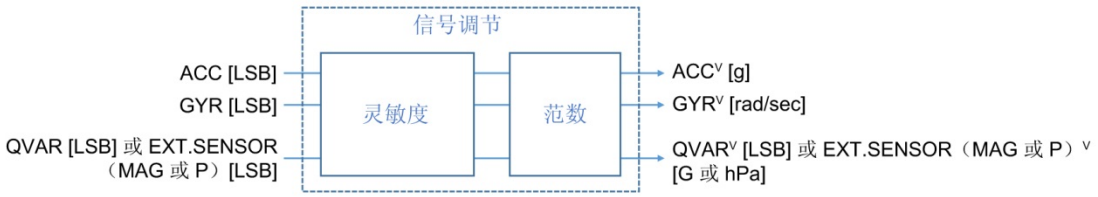
这八组有限状态机都是独立的：每个都有专用的存储区，都可以独立执行。在达到结束状态或执行某些特定命令时，会生成中断。通常，在识别到特定手势时，会生成中断。

2 信号调节模块

信号调节模块如下图所示，该模块用作输入传感器数据和 FSM 模块之间的接口。转换输出传感器数据（用 [LSB] 表示）时需要使用该模块，并且单位约定如下：

- 加速度计数据的单位为 [*g*]
- 陀螺仪数据的单位为 [rad/sec]
- Qvar 数据以 [LSB] 表示
- 外部传感器：如果为磁力计，则必须将数据转换为 [G]。如果是压力传感器，则必须将数据转换为 [hPa]。

图 3. 信号调节模块



该模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，并传递至 FSM 模块。更详细的说明如下：

- LSM6DSV16X 的加速度计数据转换系数由器件自动处理。
- LSM6DSV16X 的陀螺仪数据转换系数由器件自动处理。
- Qvar 数据和外部传感器数据转换系数不会由器件自动处理。LSM6DSV16X FSM 支持处理下列两种不同格式的外部传感器数据：
  - 具有 16 位数据的 3 轴传感器（如磁力计传感器）数据，通过将 EXT_FORMAT (00h) 嵌入高级功能寄存器的 EXT_FORMAT_SEL 位置 0（默认值）来使能。在这种情况下，外部传感器数据在内部进行如下处理。

    1. 将外部传感器的原始数据转换为 HFP 格式（不应用灵敏度）。
    2. 应用硬铁偏移。嵌入高级功能寄存器 FSM_EXT_OFFX_L (C0h)、FSM_EXT_OFFX_H (C1h)、 FSM_EXT_OFFY_L (C2h)、FSM_EXT_OFFY_H (C3h)、FSM_EXT_OFFZ_L (C4h) 和 FSM_EXT_OFFZ_H (C5h) 必须包含 HFP 格式的原始硬铁偏移。
    3. 应用软铁补偿。嵌入高级功能寄存器 FSM_EXT_MATRIX_XX_L (C6h)、FSM_EXT_MATRIX_XX_H (C7h)、FSM_EXT_MATRIX_XY_L (C8h)、FSM_EXT_MATRIX_XY_H (C9h)、FSM_EXT_MATRIX_XZ_L (CAh)、FSM_EXT_MATRIX_XZ_H (CBh)、FSM_EXT_MATRIX_YY_L (CCh)、FSM_EXT_MATRIX_YY_H (CDh)、FSM_EXT_MATRIX_YZ_L (CEh)、FSM_EXT_MATRIX_YZ_H (CFh)、FSM_EXT_MATRIX_ZZ_L (D0h) 和 FSM_EXT_MATRIX_ZZ_H (D1h) 必须包含 HFP 格式的软铁系数。
    4. 应用灵敏度。嵌入高级功能寄存器 FSM_EXT_SENSITIVITY_L (BAh) 和 FSM_EXT_SENSITIVITY_L (BBh)必须包含 HFP 格式的灵敏度值。
    5. 旋转数据。嵌入高级功能寄存器 EXT_CFG_A (D4h) 和 EXT_CFG_B (D5h) 必须包含外部传感器方向配置。

  - 具有 24 位数据的单轴传感器（如压力传感器）数据，通过将嵌入高级功能寄存器 EXT_FORMAT (00h) 的 EXT_FORMAT_SEL 位置 1 来使能。在这种情况下，外部传感器数据在内部进行如下处理。

    1. 外部传感器的原始数据保持 LSB 格式（不应用灵敏度）。
    2. 应用偏移。嵌入高级功能寄存器 EXT_3BYTE_OFFSET_XL (06h)、EXT_3BYTE_OFFSET_L (07h) 和 EXT_3BYTE_OFFSET_H (08h) 必须包含 LSB 格式的偏移。
    3. 应用灵敏度。嵌入高级功能寄存器 EXT_3BYTE_SENSITIVITY_L (02h) 和 EXT_3BYTE_SENSITIVITY_H (03h) 必须包含 HFP 格式的灵敏度值。

Qvar 传感器按 3 轴传感器进行管理，只提供一个值作为 X 轴数据。

提示 *Qvar 数据必须转换为 [LSB]（因此将灵敏度配置为 1），磁力计数据必须转换为 [G]，而压力数据必须转换为 [hPa]。*

示例：LSM6DSV16X Qvar 灵敏度等于 1 → 3C00h HFP。

示例：LIS2MDL 磁力计灵敏度为 1.5 mG/LSB → 0.0015 G/LSB → 1624h HFP；这是 LSM6DSV16X 器件的默认外部传感器灵敏度值。

对 LSM6DSV16X Qvar 数据应用适当转换系数的步骤：

1. 将 80h 写入寄存器 01h	// 使能对嵌入功能寄存器的访问
2. 将 40h 写入寄存器 17h	// PAGE_RW (17h) = 40h：使能写操作
3. 将 01h 写入寄存器 02h	// PAGE_SEL (02h) = 01h：选择嵌入高级功能寄存器页 0
4. 将 BAh 写入寄存器 08h	// PAGE_ADDRESS (08h) = BAh（FSM_EXT_SENSITIVITY_L 地址）
5. 将 [LSB] 转换系数（以 Qvar 为例，00h）写入寄存器 09h	// 将 [LSB] 转换系数值写入寄存器 FSM_EXT_SENSITIVITY_L (BAh)
6. 将 [MSB] 转换系数（以 Qvar 为例，3Ch）写入寄存器 09h	// 将 [MSB] 转换系数值写入寄存器 FSM_EXT_SENSITIVITY_H (BBh)
7. 将 01h 写入寄存器 02h	// PAGE_SEL (02h) = 01h：选择嵌入高级功能寄存器页 0
8. 将 00h 写入寄存器 17h	// PAGE_RW (17h) = 00h：禁止读/写操作
9. 将 00h 写入寄存器 01h	// 禁止对嵌入功能寄存器的访问

除转换为 HFP 格式以外，信号调节模块还会计算定义如下的输入数据范数：

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

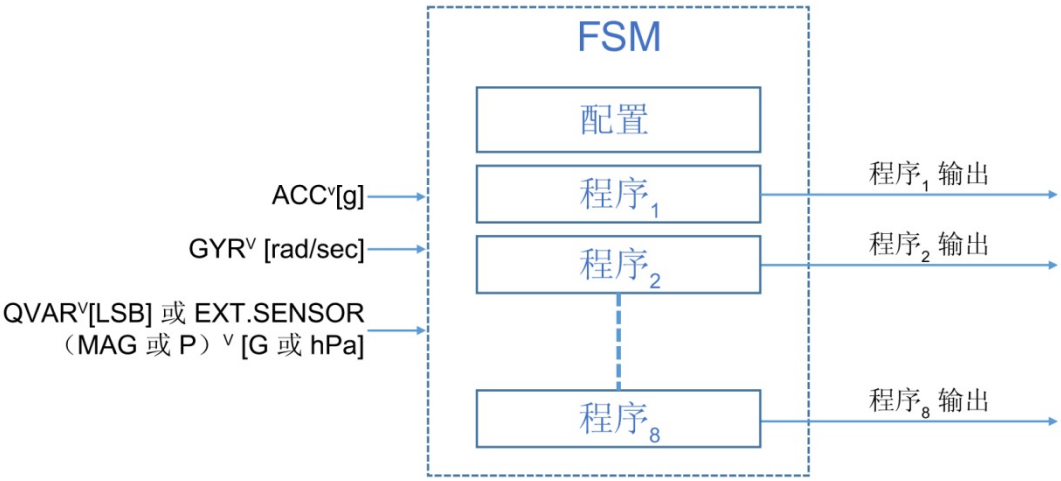
输入数据的范数可在状态机程序中使用，以确保为用户提供较高的程序定制水平。

3 FSM 模块

来自信号调节模块的输出数据信号将发送至下图所示的 FSM 模块。FSM 模块主要包括：

- 一个通用 FSM 配置模块：该模块会影响所有程序，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。
- 最多八组可配置程序：每个程序都用于处理输入数据并生成输出。

图 4. **FSM 模块**



后续各节将介绍 FSM 配置模块和程序模块。

3.1 配置模块

配置模块由 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。

LSM6DSV16X 器件在嵌入功能寄存器组内配有更多数量的寄存器，这些寄存器称为嵌入高级功能寄存器，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。

以下脚本显示了将 YYh 值写入嵌入功能寄存器组中编号为 Z 的页内地址为 XXh 的寄存器的通用程序：

1. 将 80h 写入寄存器 01h // 使能对嵌入功能寄存器的访问
2. 将 40h 写入寄存器 17h // PAGE_RW (17h) = 40h：使能写操作
3. 将 Z1h 写入寄存器 02h // PAGE_SEL (02h) = Z1h：选择嵌入高级功能寄存器的页 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：选择嵌入高级功能寄存器页 0。这是确保器件正常运行的强制性要求。
7. 将 00h 写入寄存器 17h // PAGE_RW (17h) = 00h：禁止读/写操作
8. 将 00h 写入寄存器 01h // 禁止对嵌入功能寄存器的访问

提示 *在写操作后，PAGE_ADDRESS (08h) 寄存器会自动递增。*

必须将程序配置写入嵌入高级功能寄存器，并且必须从 FSM_START_ADD_L (7Eh) 和 FSM_START_ADD_H (7Fh) 寄存器指示的寄存器地址开始。必须将所有程序写入连续的寄存器，写入时应注意以下两点重要事项：

- 从一个页转到另一个页时（即从页 03h，地址 FFh 转到页 04h，地址 00h），必须正确更新 PAGE_SEL (02h) 寄存器和 PAGE_ADDRESS (08h) 寄存器。LSM6DSV16X 器件提供可通过 PAGE_SEL (02h) 寄存器寻址的六个页。要对最后一个页进行寻址，必须将 PAGE_SEL (02h) 设为 51h。
- 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，请参见第 9 节 FSM 配置示例。

3.1.1 寄存器

下表列出的所有 FSM 相关寄存器都只能从主 SPI/I²C/MIPI I3C®接口访问。

表 1. 寄存器

寄存器名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FUNC_CFG_ACCESS	01h	EMB_FUNC_ REG_ACCESS	SHUB_ REG_ACCESS	0	0	FSM_WR_ CTRL_EN	-	-	-
FIFO_CTRL2	08h	-	-	0	-	0	-	-	XL_DualC_BATCH _FROM_FSM
CTRL7	16h	AH_QVAR_EN	INT2_DRDY_ AH_QVAR	AH_QVAR_ C_ZIN_1	AH_QVAR_ C_ZIN_0	0	0	0	-
CTRL_STATUS	1Ah	0	0	0	0	0	FSM_WR_CTRL_ STATUS	-	0
EMB_FUNC_STATUS_ MAINPAGE	49h	IS_FSM_LC	0	-	-	-	0	0	0
FSM_STATUS_ MAINPAGE	4Ah	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
MD1_CFG	5Eh	-	-	-	-	-	-	INT1_EMB_FUNC	-
MD2_CFG	5Fh	-	-	-	-	-	-	INT2_EMB_FUNC	-

3.1.2 嵌入功能寄存器

下表列出了器件中可用嵌入功能的 FSM 相关寄存器和相应地址。当 FUNC_CFG_ACCESS (01h) 寄存器中的 EMB_FUNC_REG_ACCESS 位设为 1 时，嵌入功能寄存器可访问。

表 2. 嵌入功能寄存器

寄存器名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAGE_SEL	02h	PAGE_SEL3	PAGE_SEL2	PAGE_SEL1	PAGE_SEL0	0	0	0	1
EMB_FUNC_EN_B	05h	0	0	0	-	-	0	0	FSM_EN
PAGE_ADDRESS	08h	PAGE_ADDR7	PAGE_ADDR6	PAGE_ADDR5	PAGE_ADDR4	PAGE_ADDR3	PAGE_ADDR2	PAGE_ADDR1	PAGE_ADDR0
PAGE_VALUE	09h	PAGE_VALUE7	PAGE_VALUE6	PAGE_VALUE5	PAGE_VALUE4	PAGE_VALUE3	PAGE_VALUE2	PAGE_VALUE1	PAGE_VALUE0
EMB_FUNC_INT1	0Ah	INT1_FSM_LC	0	-	-	-	0	0	0
FSM_INT1	0Bh	INT1_FSM8	INT1_FSM7	INT1_FSM6	INT1_FSM5	INT1_FSM4	INT1_FSM3	INT1_FSM2	INT1_FSM1
EMB_FUNC_INT2	0Eh	INT2_FSM_LC	0	-	-	-	0	0	0
FSM_INT2	0Fh	INT2_FSM8	INT2_FSM7	INT2_FSM6	INT2_FSM5	INT2_FSM4	INT2_FSM3	INT2_FSM2	INT2_FSM1
EMB_FUNC_STATUS	12h	IS_FSM_LC	0	-	-	-	0	0	0
FSM_STATUS	13h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
PAGE_RW	17h	EMB_FUNC_LIR	PAGE_WRITE	PAGE_READ	0	0	0	0	0
FSM_ENABLE	46h	FSM8_EN	FSM7_EN	FSM6_EN	FSM5_EN	FSM4_EN	FSM3_EN	FSM2_EN	FSM1_EN
FSM_LONG_ COUNTER_L	48h	FSM_LC_7	FSM_LC_6	FSM_LC_5	FSM_LC_4	FSM_LC_3	FSM_LC_2	FSM_LC_1	FSM_LC_0
FSM_LONG_ COUNTER_H	49h	FSM_LC_15	FSM_LC_14	FSM_LC_13	FSM_LC_12	FSM_LC_11	FSM_LC_10	FSM_LC_9	FSM_LC_8
FSM_OUTS1	4Ch	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS2	4Dh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS3	4Eh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS4	4Fh	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS5	50h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS6	51h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS7	52h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS8	53h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_ODR	5Fh	0	1	FSM_ODR_2	FSM_ODR_1	FSM_ODR_0	0	1	1
EMB_FUNC_INIT_B	67h	0	0	0	-	-	0	0	FSM_INIT

3.1.3 嵌入高级功能页

下表列出了嵌入高级功能 page 0 的 FSM 相关寄存器。当 PAGE_SEL (02h) 寄存器中的 PAGE_SEL[3:0] 位设为 0000 时，这些寄存器可访问。

表 3. 嵌入高级功能寄存器 - page 0

寄存器名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_EXT_SENSITIVITY_L	BAh	FSM_EXT_S_7	FSM_EXT_S_6	FSM_EXT_S_5	FSM_EXT_S_4	FSM_EXT_S_3	FSM_EXT_S_2	FSM_EXT_S_1	FSM_EXT_S_0
FSM_EXT_SENSITIVITY_H	BBh	FSM_EXT_S_15	FSM_EXT_S_14	FSM_EXT_S_13	FSM_EXT_S_12	FSM_EXT_S_11	FSM_EXT_S_10	FSM_EXT_S_9	FSM_EXT_S_8
FSM_EXT_OFFX_L	C0h	FSM_EXT_OFFX_7	FSM_EXT_OFFX_6	FSM_EXT_OFFX_5	FSM_EXT_OFFX_4	FSM_EXT_OFFX_3	FSM_EXT_OFFX_2	FSM_EXT_OFFX_1	FSM_EXT_OFFX_0
FSM_EXT_OFFX_H	C1h	FSM_EXT_OFFX_15	FSM_EXT_OFFX_14	FSM_EXT_OFFX_13	FSM_EXT_OFFX_12	FSM_EXT_OFFX_11	FSM_EXT_OFFX_10	FSM_EXT_OFFX_9	FSM_EXT_OFFX_8
FSM_EXT_OFFY_L	C2h	FSM_EXT_OFFY_7	FSM_EXT_OFFY_6	FSM_EXT_OFFY_5	FSM_EXT_OFFY_4	FSM_EXT_OFFY_3	FSM_EXT_OFFY_2	FSM_EXT_OFFY_1	FSM_EXT_OFFY_0
FSM_EXT_OFFY_H	C3h	FSM_EXT_OFFY_15	FSM_EXT_OFFY_14	FSM_EXT_OFFY_13	FSM_EXT_OFFY_12	FSM_EXT_OFFY_11	FSM_EXT_OFFY_10	FSM_EXT_OFFY_9	FSM_EXT_OFFY_8
FSM_EXT_OFFZ_L	C4h	FSM_EXT_OFFZ_7	FSM_EXT_OFFZ_6	FSM_EXT_OFFZ_5	FSM_EXT_OFFZ_4	FSM_EXT_OFFZ_3	FSM_EXT_OFFZ_2	FSM_EXT_OFFZ_1	FSM_EXT_OFFZ_0
FSM_EXT_OFFZ_H	C5h	FSM_EXT_OFFZ_15	FSM_EXT_OFFZ_14	FSM_EXT_OFFZ_13	FSM_EXT_OFFZ_12	FSM_EXT_OFFZ_11	FSM_EXT_OFFZ_10	FSM_EXT_OFFZ_9	FSM_EXT_OFFZ_8
FSM_EXT_MATRIX_XX_L	C6h	FSM_EXT_MAT_XX_7	FSM_EXT_MAT_XX_6	FSM_EXT_MAT_XX_5	FSM_EXT_MAT_XX_4	FSM_EXT_MAT_XX_3	FSM_EXT_MAT_XX_2	FSM_EXT_MAT_XX_1	FSM_EXT_MAT_XX_0
FSM_EXT_MATRIX_XX_H	C7h	FSM_EXT_MAT_XX_15	FSM_EXT_MAT_XX_14	FSM_EXT_MAT_XX_13	FSM_EXT_MAT_XX_12	FSM_EXT_MAT_XX_11	FSM_EXT_MAT_XX_10	FSM_EXT_MAT_XX_9	FSM_EXT_MAT_XX_8
FSM_EXT_MATRIX_XY_L	C8h	FSM_EXT_MAT_XY_7	FSM_EXT_MAT_XY_6	FSM_EXT_MAT_XY_5	FSM_EXT_MAT_XY_4	FSM_EXT_MAT_XY_3	FSM_EXT_MAT_XY_2	FSM_EXT_MAT_XY_1	FSM_EXT_MATXY_0
FSM_EXT_MATRIX_XY_H	C9h	FSM_EXT_MAT_XY_15	FSM_EXT_MAT_XY_14	FSM_EXT_MAT_XY_13	FSM_EXT_MAT_XY_12	FSM_EXT_MAT_XY_11	FSM_EXT_MAT_XY_10	FSM_EXT_MAT_XY_9	FSM_EXT_MAT_XY_8
FSM_EXT_MATRIX_XZ_L	CAh	FSM_EXT_MAT_XZ_7	FSM_EXT_MAT_XZ_6	FSM_EXT_MAT_XZ_5	FSM_EXT_MAT_XZ_4	FSM_EXT_MAT_XZ_3	FSM_EXT_MAG_SI_XZ_2	FSM_EXT_MAT_XZ_1	FSM_EXT_MAT_XZ_0
FSM_EXT_MATRIX_XZ_H	CBh	FSM_EXT_MAT_XZ_15	FSM_EXT_MAT_XZ_14	FSM_EXT_MAT_XZ_13	FSM_EXT_MAT_XZ_12	FSM_EXT_MAT_XZ_11	FSM_EXT_MAT_XZ_10	FSM_EXT_MAT_XZ_9	FSM_EXT_MAT_XZ_8
FSM_EXT_MATRIX_YY_L	CCh	FSM_EXT_MAT_YY_7	FSM_EXT_MAT_YY_6	FSM_EXT_MAT_YY_5	FSM_EXT_MAT_YY_4	FSM_EXT_MAT_YY_3	FSM_EXT_MAT_YY_2	FSM_EXT_MAT_YY_1	FSM_EXT_MAT_YY_0
FSM_EXT_MATRIX_YY_H	CDh	FSM_EXT_MAT_YY_15	FSM_EXT_MAT_YY_14	FSM_EXT_MAT_YY_13	FSM_EXT_MAT_YY_12	FSM_EXT_MAT_YY_11	FSM_EXT_MAT_YY_10	FSM_EXT_MAT_YY_9	FSM_EXT_MAT_YY_8
FSM_EXT_MATRIX_YZ_L	CEh	FSM_EXT_MAT_YZ_7	FSM_EXT_MAT_YZ_6	FSM_EXT_MAT_YZ_5	FSM_EXT_MAT_YZ_4	FSM_EXT_MAT_YZ_3	FSM_EXT_MAT_YZ_2	FSM_EXT_MAT_YZ_1	FSM_EXT_MAT_YZ_0
FSM_EXT_MATRIX_YZ_H	CFh	FSM_EXT_MAT_YZ_15	FSM_EXT_MAT_YZ_14	FSM_EXT_MAT_YZ_13	FSM_EXT_MAT_YZ_12	FSM_EXT_MAT_YZ_11	FSM_EXT_MAT_YZ_10	FSM_EXT_MAT_YZ_9	FSM_EXT_MAT_YZ_8
FSM_EXT_MATRIX_ZZ_L	D0h	FSM_EXT_MAT_ZZ_7	FSM_EXT_MAT_ZZ_6	FSM_EXT_MAT_ZZ_5	FSM_EXT_MAT_ZZ_4	FSM_EXT_MAT_ZZ_3	FSM_EXT_MAT_ZZ_2	FSM_EXT_MAT_ZZ_1	FSM_EXT_MAT_ZZ_0
FSM_EXT_MATRIX_ZZ_H	D1h	FSM_EXT_MAT_ZZ_15	FSM_EXT_MAT_ZZ_14	FSM_EXT_MAT_ZZ_13	FSM_EXT_MAT_ZZ_12	FSM_EXT_MAT_ZZ_11	FSM_EXT_MAT_ZZ_10	FSM_EXT_MAT_ZZ_9	FSM_EXT_MAT_ZZ_8
EXT_CFG_A	D4h	0	EXT_Y_AXIS2	EXT_Y_AXIS1	EXT_Y_AXIS0	0	EXT_Z_AXIS2	EXT_Z_AXIS1	EXT_Z_AXIS0
EXT_CFG_B	D5h	0	0	0	0	0	EXT_X_AXIS2	EXT_X_AXIS1	EXT_X_AXIS0

下表列出了嵌入高级功能 page 1 的 FSM 相关寄存器。当 PAGE_SEL (02h) 寄存器中的 PAGE_SEL[3:0] 位设为 0001 时，这些寄存器可访问。

表 4. 嵌入高级功能寄存器 - page 1

寄存器名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC_TIMEOUT_L	7Ah	FSM_LC_TIMEOUT7	FSM_LC_TIMEOUT6	FSM_LC_TIMEOUT5	FSM_LC_TIMEOUT4	FSM_LC_TIMEOUT3	FSM_LC_TIMEOUT2	FSM_LC_TIMEOUT1	FSM_LC_TIMEOUT0
FSM_LC_TIMEOUT_H	7Bh	FSM_LC_TIMEOUT15	FSM_LC_TIMEOUT14	FSM_LC_TIMEOUT13	FSM_LC_TIMEOUT12	FSM_LC_TIMEOUT11	FSM_LC_TIMEOUT10	FSM_LC_TIMEOUT9	FSM_LC_TIMEOUT8
FSM_PROGRAMS	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	7Eh	FSM_START7	FSM_START6	FSM_START5	FSM_START4	FSM_START3	FSM_START2	FSM_START1	FSM_START0
FSM_START_ADD_H	7Fh	FSM_START15	FSM_START714	FSM_START13	FSM_START12	FSM_START11	FSM_START10	FSM_START9	FSM_START8

下表列出了嵌入高级功能 page 2 的 FSM 相关寄存器。当 PAGE_SEL (02h) 寄存器中的 PAGE_SEL[3:0] 位设为 0010 时，这些寄存器可访问。

表 5. 嵌入高级功能寄存器 - page 2

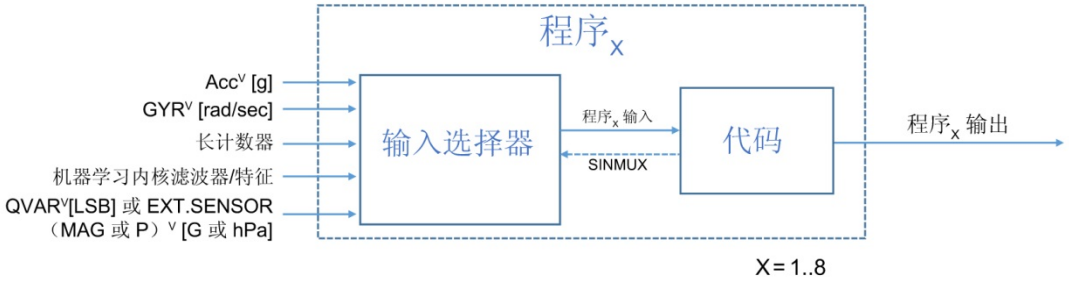
寄存器名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EXT_FORMAT	00h	0	0	0	0	0	EXT_FORMAT_SEL	0	0
EXT_3BYTE_SENSITIVITY_L	02h	EXT_3BYTE_S_7	EXT_3BYTE_S_6	EXT_3BYTE_S_5	EXT_3BYTE_S_4	EXT_3BYTE_S_3	EXT_3BYTE_S_2	EXT_3BYTE_S_1	EXT_3BYTE_S_0
EXT_3BYTE_SENSITIVITY_H	03h	EXT_3BYTE_S_15	EXT_3BYTE_S_14	EXT_3BYTE_S_13	EXT_3BYTE_S_12	EXT_3BYTE_S_11	EXT_3BYTE_S_10	EXT_3BYTE_S_9	EXT_3BYTE_S_8
EXT_3BYTE_OFFSET_XL	06h	EXT_3BYTE_OFF_7	EXT_3BYTE_OFF_6	EXT_3BYTE_OFF_5	EXT_3BYTE_OFF_4	EXT_3BYTE_OFF_3	EXT_3BYTE_OFF_2	EXT_3BYTE_OFF_1	EXT_3BYTE_OFF_0
EXT_3BYTE_OFFSET_L	07h	EXT_3BYTE_OFF_15	EXT_3BYTE_OFF_14	EXT_3BYTE_OFF_13	EXT_3BYTE_OFF_12	EXT_3BYTE_OFF_11	EXT_3BYTE_OFF_10	EXT_3BYTE_OFF_9	EXT_3BYTE_OFF_8
EXT_3BYTE_OFFSET_H	08h	EXT_3BYTE_OFF_23	EXT_3BYTE_OFF_22	EXT_3BYTE_OFF_21	EXT_3BYTE_OFF_20	EXT_3BYTE_OFF_19	EXT_3BYTE_OFF_18	EXT_3BYTE_OFF_17	EXT_3BYTE_OFF_16

3.2 程序模块

来自信号调节模块的输出数据将发送至 FSM 模块，该模块由八个程序模块组成。如下图所示，每个程序模块包括：

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

图 5. 程序模块



3.2.1 输入选择器模块

输入选择器模块用于在以下物理传感器数据信号或内部计算的数据信号之间选择输入数据信号：

- LSM6DSV16X 加速度计数据，包含预先计算的范数 (V)
- LSM6DSV16X 陀螺仪数据，包含预先计算的范数 (V)
- LSM6DSV16X Qvar 数据，可由作为外部传感器的 FSM 处理
- 具有 16 位数据的外部 3 轴传感器（如磁力计传感器）的数据，包含预先计算的范数 (V)
- 具有 24 位数据的外部单轴传感器（如压力传感器）的数据
- 长计数器的值
- 通过适当地配置机器学习内核在内部滤波的数据和在内部计算的特征
- 内部计算的角度，包含预先计算的范数 (V)

范数 (V) 通过以下公式在内部计算：

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

使用机器学习内核可将器件配置为，计算应用于内部/外部传感器数据的特征（如均值、方差、峰峰值、能量等）或滤波器（如高通、带通、IIR1 和 IIR2 滤波器）。有关机器学习内核功能的更多详细信息，请参见应用指南 AN5804。

下图显示了加速度计和陀螺仪数字链中有限状态机模块的输入。对于 LSM6DSV16X 支持的所有四种连接模式，两个数字链中的有限状态机 (FSM) 模块位置相同。

图 6. **FSM 输入（加速度计）**

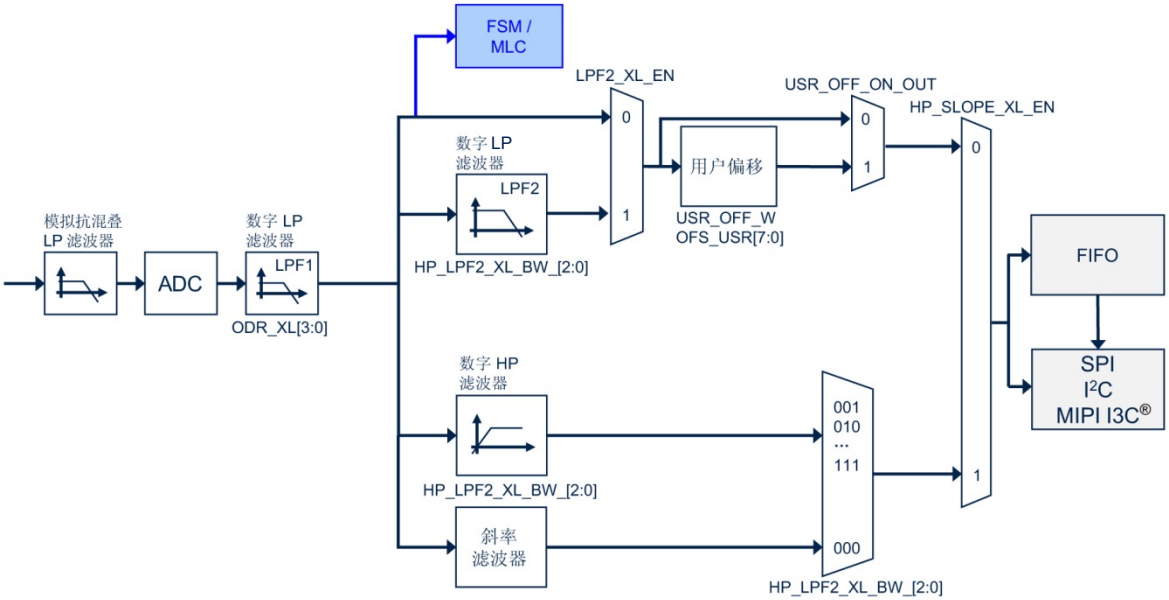
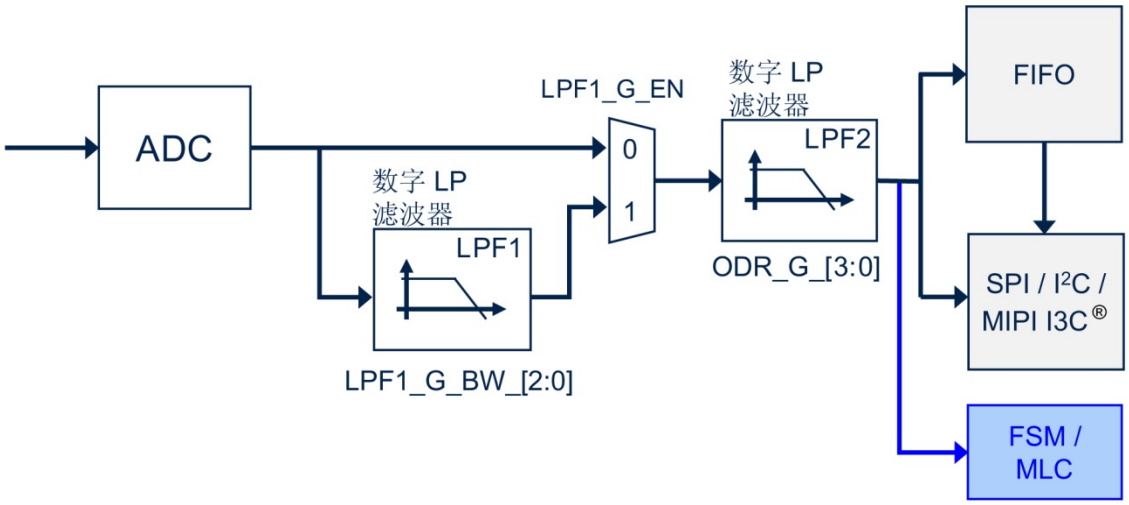


图 7. **FSM 输入（陀螺仪）**



加速度计和陀螺仪的信号带宽取决于器件配置。有关更多信息，请参见 www.st.com 中的 AN5763。根据程序目的，程序模块会执行已配置的程序（代码模块），即处理所选输入信号并生成相应的程序输出信号。

提示　*程序指令部分中的用户可使用 SINMUX 命令动态切换程序模块所需的输入信号。有关 SINMUX 命令的其他详细信息，请参见 SINMUX (23h)。*

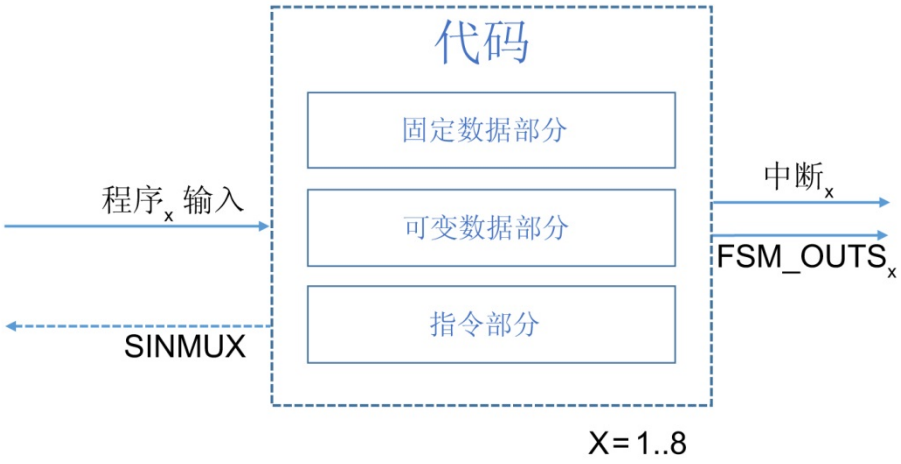
3.2.2 代码模块

FSM 程序 x 代码模块包含状态机程序。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 固定部分的大小均相同）和可变部分（每个 FSM 的可变部分都有特定大小）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 信号的已处理样本集合生成中断 x 信号并修改相应的 FSM_OUTSx 寄存器值。

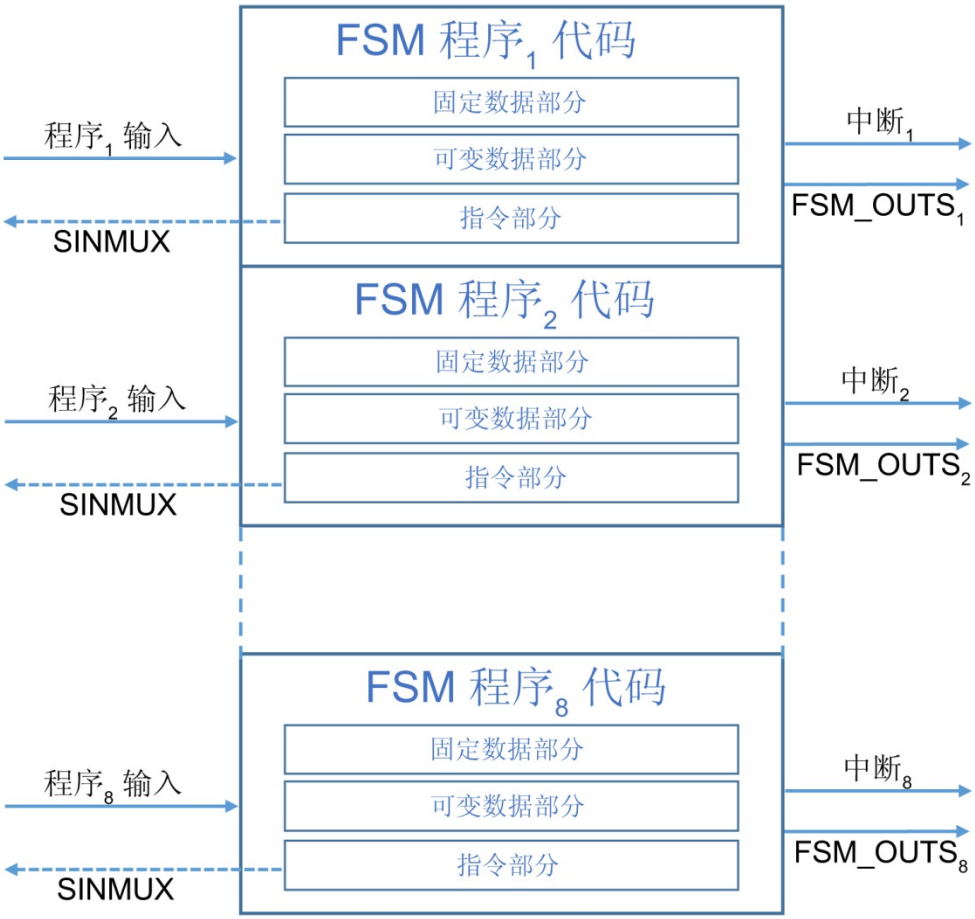
图 8. FSM 程序 x 代码结构



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

提示 *每次器件上电时均必须重新配置 FSM（基于所有嵌入功能）。*

图 9. FSM 程序 x 存储区



4 FSM 中断状态和信号

FSM 支持产生两种不同的中断信号：FSM 程序中断信号和 FSM 长计数器中断信号。

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

当嵌入功能寄存器 FSM_LONG_COUNTER_L (48h) 和 FSM_LONG_COUNTER_H (49h) 中存储的长计数器值达到嵌入高级功能寄存器 FSM_LC_TIMEOUT_L (7Ah) 和 FSM_LC_TIMEOUT_H (7Bh)（页 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。

FSM 中断状态可以通过读取以下专用寄存器进行查看：

- FSM_STATUS_MAINPAGE (4Ah) 寄存器或 FSM_STATUS (13h) 嵌入功能寄存器，专用于存储 FSM 中断状态
- EMB_FUNC_STATUS_MAINPAGE (49h) 寄存器或 EMB_FUNC_STATUS (12h) 嵌入功能寄存器，专用于存储长计数器中断状态

FSM 中断信号可通过将专用位置位传输至 INT1/INT2 中断引脚：

- 将嵌入功能寄存器 FSM_INT1/FSM_INT2 的 INT1_FSM[1:8]/INT2_FSM[1:8] 位置 1
- 将嵌入功能寄存器 EMB_FUNC_INT1/EMB_FUNC_INT2 的 INT1_FSM_LC/INT2_FSM_LC 位置 1

提示 *在以上两种情况下，均必须通过将 MD1_CFG/MD2_CFG 寄存器的 INT1_EMB_FUNC/INT2_EMB_FUNC 位置 1 来使能向 INT1/INT2 中断引脚发送嵌入功能事件。*

默认情况下，中断信号表现为脉冲形式。脉冲持续时间取决于使能的更快速的传感器：

- 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。

提示 *脉冲最短持续时间为 1/960 Hz（约 1 ms）。*

将 PAGE_RW (17h) 嵌入功能寄存器的 EMB_FUNC_LIR 位置 1，可使能锁存模式。在这种情况下，读取以下寄存器时会复位中断信号和状态位：

- FSM_STATUS_MAINPAGE (4Ah) 寄存器或 FSM_STATUS (13h) 嵌入功能寄存器（专用于 FSM 中断）
- EMB_FUNC_STATUS_MAINPAGE (49h) 寄存器或 EMB_FUNC_STATUS (12h) 嵌入功能寄存器（专用于存储长计数器中断）

5 长计数器

长计数器是一种可供用户使用的 15 位临时计数器资源。使用 INCR、DECR 或 RESET 命令，可以分别递增、递减或重置存储在 FSM_LONG_COUNTER_L (48h) 和 FSM_LONG_COUNTER_H (49h) 寄存器中的长计数器值。
长计数器的最小（默认）值为 0，而长计数器的最大值为在嵌入高级功能寄存器 FSM_LC_TIMEOUT_L (7Ah) 和 FSM_LC_TIMEOUT_H (7Bh) 中存储的所配置的超时值。
当长计数器的值等于配置的长计数器超时值时，EMB_FUNC_STATUS_MAINPAGE (49h) 寄存器和 EMB_FUNC_STATUS (12h) 嵌入功能寄存器的 IS_FSM_LC 状态位置 1。
有关 FSM 长计数器中断的详细信息，请参见第 4 节 FSM 中断状态和信号。
该资源为所有程序所公用，并且无需在 **[可变数据部分]** 中分配的其他资源。

提示 *当 FSM_LC_TIMEOUT 等于 0 时，禁止长计数器功能。*

提示 *FSM_LC_TIMEOUT 值必须设为小于 2^{15}（15 位无符号格式）。*

6 固定数据部分

[固定数据部分] 用于存储有关 **[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]

	名称	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	EXT_SINMUX	ANGLE	PAS	DECTREE	STOPDONE	-	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)							

提示 以绿色标出的位必须根据程序目的设置，以红色标出的位必须在将程序载入嵌入高级功能寄存器页时置 0（它们通过 FSM 逻辑自动配置）。

前两个字节用于存储程序使用的资源量，而其他字节则供器件用来存储程序状态。

- 使用 CONFIG_A 可以声明：
  - 最多三个阈值（NR_THRESH 位）
  - 最多三个掩码（NR_MASK 位）
  - 最多两个长（16 位）定时器（NR_LTIMER 位）
  - 最多两个短（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）
  - 扩展 sinmux 功能，用于选择长计数器值或由机器学习核心计算出的滤波器/特征的值作为 FSM 输入（EXT_SINMUX 位）
  - 必须计算和存储的陀螺仪角度的使用（ANGLE 位）
  - 必须计算和存储的先前轴符号的使用（PAS 位）
  - 决策树接口的使用（DECTREE 位）
- SIZE 参数用于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 参数用于存储当前程序状态（所选掩码、所选阈值、输入信号等）。
- RESET POINTER (RP) 和 PROGRAM POINTER (PP) 分别用于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。

提示 当 PP 等于 0 时，器件自动运行启动例程（有关更多信息，请参见第 10 节启动例程），以正确初始化状态机的内部变量和参数。这是确保器件正常运行的强制性要求。

7 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。

在 **[固定数据部分]** 中枚举的每个资源都将以适当的大小和位置在 **[可变数据部分]** 中分配。下图显示了 **[可变数据部分]** 的结构。

图 11. **[可变数据部分]**

	名称	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	EXT_SINMUX	SINMUX_ADDR_B				SINMUX_ADDR_A			
13		IN_SEL(3)	THRXYZ1	NEXT_ODR	-	SINMUX_ADDR_C			
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) 或 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(1:0)		DTRES(3:0)			

如上表所示，**[可变数据部分]** 的最大容量为 36 字节。如果程序需要的资源较少，则分配给 **[可变数据部分]** 的容量也较小。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。

提示

*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]** 中定义了 NR_THRESH = 10（定义了两个阈值），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）。*

7.1 阈值

阈值资源用于在比较条件下检查和验证由所选输入信号（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 LSM6DSV16X 加速度计信号，则阈值单位为 [*g*]。
- 如果选择 LSM6DSV16X 陀螺仪信号，则阈值单位为 [rad/sec]。
- 如果选择 LSM6DSV16X 集成陀螺仪信号，则阈值单位为 [rad]。
- 如果选择外部传感器信号，则阈值单位与应用灵敏度后使用的单位相同。
- 如果选择长计数器值，则阈值以 15 位无符号格式表示。
- 如果选择 MLC 滤波信号或计算特征值，则阈值单位与所选滤波信号或计算特征值的单位相同。

阈值既可以有符号，也可以无符号。可以使用 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]。

相关命令：

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

相关条件：

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

7.2 扩展 sinmux

扩展 sinmux 资源主要用于选择长计数器值（第一个参数等于 8 的 SINMUX）或由机器学习内核计算出的特征/滤波值（第一个参数等于 9 的 SINMUX）作为 FSM 输入。此外，当需要在一个 ODR (THRXYZ1) 中执行多个条件时，也需要该资源。
通过将 CONFIG_B 字节的 EXT_SINMUX 位置 1，可在 **[可变数据部分]** 中分配 EXT_SINMUX 字节（EXT_SINMUX 字节值由器件自动管理）。如果下面列出的命令中至少有一个在程序中使用，则必须这样操作。
相关命令：

- 第一个参数等于 8（选择长计数器）或 9（选择 MLC 特征值或滤波值）的 SINMUX
- THRXYZ0、THRXYZ1

相关条件：

- N/A

7.3 掩码/临时掩码

掩码资源用于在执行条件时使能或禁止对输入数据 (X, Y, Z, V) 执行的掩码操作。如果将掩码位置 1，则使能相应的轴和符号，否则将其禁止。如果输入数据由单轴外部传感器以 3 字节数据格式生成，则该数据位于掩码的 X 通道中。掩码用于阈值比较条件或过零检测。掩码可以通过使能带有负号的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，禁止轴；
2. 正轴位 = 0 / 负轴位 = 1，使能相反符号的轴；
3. 正轴位 = 1 / 负轴位 = 0，使能当前符号的轴；
4. 正轴位 = 1 / 负轴位 = 1，使能当前符号的轴和相反符号的轴。

使能程序后，会将每个掩码的值复制到相关临时掩码 (TM) 中，在条件执行期间将使用该临时掩码。每次发出条件时，将条件结果再次存储在临时掩码中（这也会影响后续条件）。

示例：

- GNTH1 条件
- THRESH1 = 0.50 *g*
- MASKA = 12h (00010010b) →使能 -Y 和 +V
- 加速度计当前输入样本 = [0.72 -0.45 0.77 1.15]

执行条件前的 TM	0	0	0	1	0	0	1	0
加速度计样本	0.72	-0.72	-0.45	0.45	0.77	-0.77	1.15	-1.15
执行条件后的 TM	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
- 在每个为真的 NEXT 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]** 中只分配了 MASKA[7:0]/TMASKA[7:0]。
- NR_ MASK[1:0] = 10：在 **[可变数据部分]** 中分配了 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]** 中分配了 MASKA[7:0]/TMASKA[7:0]、MASKB[7:0]/TMASKB[7:0] 和 MASKC[7:0]/TMASKC[7:0]。

相关命令：

- SELMA / SELMB / SELMC
- SMA / SMB / SMC
- REL
- SRTAM0 / SRTAM1

相关条件：

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

7.4 角度计算

条件发出时，使用角度资源而不是角速度数据。角度计算在内部执行：陀螺仪数据自动乘以 DELTAT 值，结果添加到相应的角度轴字节（DX、DY、DZ 和 DV）。**[固定数据部分]** 的 CONFIG B 字节的 ANGLE 位置 1 时，使能该功能。每次 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，计算的角度（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。
相关命令：

- CANGLE

相关条件：

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

7.5 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）。时间基准由 FSM_ODR_ODR_CFG_B (5Fh) 寄存器的 FSM_ODR[2:0] 位设置，包括抽取因子（如果使用）。长定时器资源称为 TI1 和 TI2，短定时器资源称为 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加载，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 值始终预加载到相应的定时器值中。在这种模式下，定时器持续时间仅影响一种状态。
- SCTC1 模式：当程序指针移至具有超时条件的状态时，视新状态中使用的定时器而定，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 TC 值会被预加载到相应的定时器值中。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，则不会预加载 TC 值。TC 值从其先前值开始持续减小。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 10 个样本。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）

视 S0 而定，有两种不同的状态机行为：

- SCTC0：始终预加载 TC 字节（程序指针移至状态 S1、S2 和 S3 时），每个条件下最多检查 10 个样本。这意味着可以在最多 30 个样本中验证所有条件；
- SCTC1：仅在程序指针移至 S1 时才预加载 TC 字节（移至 S2 和 S3 时不预加载），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定时器，如下所述：

- NR_LTIMER[1:0] = 00：在 **[可变数据部分]** 中未分配长定时器。
- NR_LTIMER[1:0] = 01：在 **[可变数据部分]** 中分配 TIMER1[15:0]。
- NR_LTIMER(1:0) = 10：在 **[可变数据部分]** 中分配 TIMER1[15:0] 和 TIMER2[15:0]。

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：在 **[可变数据部分]** 中分配了 TIMER3[7:0]。
- NR_TIMER[1:0] = 10：在 **[可变数据部分]** 中分配了 TIMER3[7:0] 和 TIMER4[7:0]。

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。

相关命令：

- STIMER3 / STIMER4
- SCTC0 / SCTC1

相关条件：

- TI1 / TI2 / TI3 / TI4

7.6 降频器

降频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 位置 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（由器件自动管理）。根据以下公式，抽取因子与 FSM_ODR (5Fh) 寄存器的 FSM_ODR[2:0] 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，当前样本将用作状态机的新输入，并且 DESC 值被再次设为初始抽取值。

相关命令：

- N/A

相关条件：

- N/A

提示 *DEST 的最小有意义值为“2”。*

7.7 前轴符号

前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）、用于重置陀螺仪积分数据（**[可变数据部分]** 中的 DX、DY、DZ 和 DV 字节）的清除角标志 (CANGLE) 和所选中断屏蔽类型（MSKIT、MSKITEQ 或 UMSKIT）。

通过将 CONFIG_B 字节的 PAS 位置 1，可在 **[可变数据部分]** 中分配 PAS 字节（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，则必须这样操作。

相关命令：

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

相关条件：

- PZC / NZC

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

7.8 MLC 接口

FSM 的 MLC 接口可以对决策树的输出或计算出的滤波器/特征的值实施条件。当希望将机器学习逻辑或定制滤波器与 FSM 程序结合使用时，这可能非常有用。

决策树的输出可以使用 CHKDT 条件访问，该条件可用于评估机器学习内核算法中可用的四个决策树之一的结果。

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[1:0] 位，0 到 3）和对应的期望值（DTRES[3:0] 位，0 到 15）的信息。

由 MLC 计算出的滤波器/特征的值可以使用扩展 sinmux 功能进行选择。

通过将 CONFIG_B 字节的 EXT_SINMUX 位置 1，可将 EXT_SINMUX 字节分配到 **[可变数据部分]** 中（EXT_SINMUX 字节由器件自动管理）。有关如何选择由 MLC 计算出的滤波器/特征的更多详细信息，请参见 第 7.2 节扩展 sinmux 和第 8.2.21 节 SINMUX (23h)。

提示 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，请参见相关段落。

提示 有关如何配置 MLC 的更多详细信息，请参见 AN5804。

相关命令：

- SINMUX

相关条件：

- CHKDT

8 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：

- 命令用于执行特殊的流控、输出和同步任务。有些命令可能有参数，以单步命令的形式执行。
- RESET/NEXT 条件是两种条件的组合（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。它们是程序指令部分的模块。

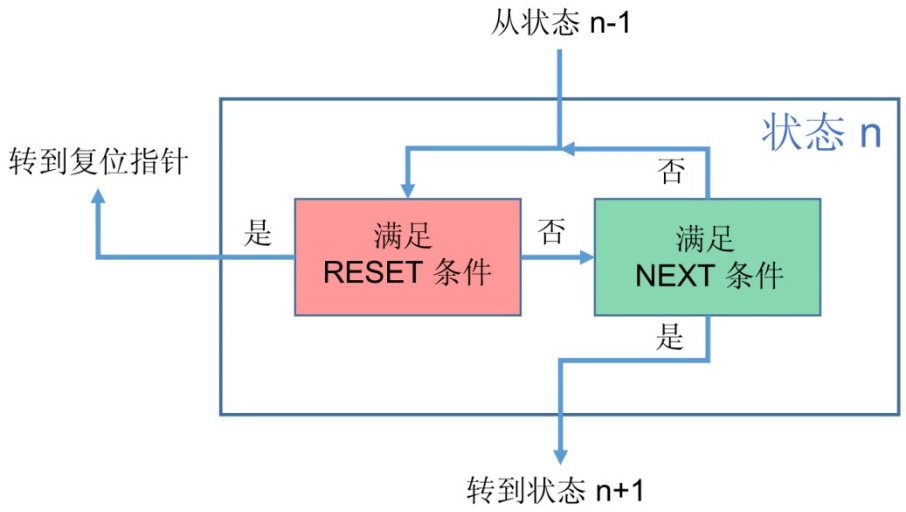
8.1 RESET/NEXT 条件

RESET/NEXT 条件用于复位或继续执行程序流，并且只有一个状态。
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。
RESET/NEXT 条件影响程序流，如下所示：

- 只要 RESET 条件为真 (PP = RP)，就会跳转至复位指针。
- 只要 RESET 条件为假且 NEXT 条件为真 (PP = PP + 1)，就会跳转至下一状态。
- 当 RESET 和 NEXT 条件都为假时，不会发生跳转。

如下图所示，NEXT 条件始终在 RESET 条件之后执行，即仅在不满足 RESET 条件时才进行评估。

图 12. 单状态说明



默认情况下，当 FSM 处理新数据样本时，会执行单个 RESET/NEXT 条件，但可以使用 THRXYZ1/THRXYZ0 命令对同一数据样本执行多个条件。有关如何使能此模式的详细信息，请参见第 8.2.27 节 THRXYZ1 (F7h) 和第 8.2.28 节 THRXYZ0 (F8h)。

提示 *RESET 条件始终在 NEXT 条件之前评估。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 16 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 6. 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	不适用
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	任何触发轴以负斜率越过零值		PAS
Fh	CHKDT	对照预期结果检查决策树结果	需要配置机器学习内核	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。

提示

让 NEXT 和 RESET 这两个位置的条件相同没有意义。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。
此外，无法执行以下条件，因为它们被识别为命令：

- *PZC | CHKDT 操作码 (0xDF) 等于 SMB 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。*
- *CHKDT | LNTH1 操作码 (0xF7) 等于 THRXYZ1 操作码。*
- *CHKDT | GNTH2 操作码 (0xF6) 等于 RSTLC 操作码。*
- *CHKDT | LNTH2 操作码 (0xF8) 等于 THRXYZ0 操作码。*
- *CHKDT | PZC 操作码 (0xFD) 等于 DECR 操作码。*
- *CHKDT | NZC 操作码 (0xFE) 等于 SMC 操作码。*

8.1.1 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 对。

动作：

- 如果 NOP 处于 RESET 条件，则 FSM 只评估 NEXT 条件。
- 如果 NOP 处于 NEXT 条件，则 FSM 只评估 RESET 条件。

8.1.2 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 FSM 处理新的数据样本 (X, Y, Z, V) 时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较。
  - 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。

提示 *有关如何在 THRXYZ1 / THRXYZ0 命令中使用定时器的详细信息，请参见第 8.2.27 节 THRXYZ1 (F7h) 和 第 8.2.28 节 THRXYZ0 (F8h)。*

8.1.3 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 FSM 处理新的数据样本 (X, Y, Z, V) 时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较。
  - 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。

提示 *有关如何在 THRXYZ1 / THRXYZ0 命令中使用定时器的详细信息，请参见第 8.2.27 节 THRXYZ1 (F7h) 和 第 8.2.28 节 THRXYZ0 (F8h)。*

8.1.4 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 FSM 处理新的数据样本 (X, Y, Z, V) 时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较。
  - 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。

提示 *有关如何在 THRXYZ1 / THRXYZ0 命令中使用定时器的详细信息，请参见第 8.2.27 节 THRXYZ1 (F7h) 和 第 8.2.28 节 THRXYZ0 (F8h)。*

8.1.5 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 FSM 处理新的数据样本 (X, Y, Z, V) 时，TC = TC – 1：
  - 如果 TC > 0，在当前状态下继续比较。
  - 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。

提示 *有关如何在 THRXYZ1 / THRXYZ0 命令中使用定时器的详细信息，请参见第 8.2.27 节 THRXYZ1 (F7h) 和 第 8.2.28 节 THRXYZ0 (F8h)。*

8.1.6 GNTH1 (5h)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴大于或等于阈值 1，则 GNTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X ≥ 阈值 || -X ≥阈值）。*

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。

8.1.7 GNTH2 (6h)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴大于或等于阈值 2，则 GNTH2 条件有效。比较过程中使用的阈值是 THRESH2。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X ≥ 阈值 || -X ≥阈值）。*

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。

8.1.8 LNTH1 (7h)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴小于阈值 1，则 LNTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X < 阈值 || -X < 阈值）。*

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。

8.1.9 LNTH2 (8h)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴小于阈值 2，则 LNTH2 条件有效。比较过程中使用的阈值是 THRESH2。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X < 阈值 || -X < 阈值）。*

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。

8.1.10 GLTH1 (9h)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的所有轴均大于或等于阈值 1，则 GLTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X ≥ 阈值 || -X ≥阈值）。*

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。

8.1.11 LLTH1 (Ah)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的所有轴均小于阈值 1，则 LLTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X < 阈值 || -X < 阈值）。*

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。

8.1.12 GRTH1 (Bh)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴大于或等于反向阈值 1，则 GRTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 -THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 -THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X ≥ 阈值 || -X ≥阈值）。*

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。

8.1.13 LRTH1 (Ch)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任何触发轴小于反向阈值 1，则 LRTH1 条件有效。比较过程中使用的阈值取决于 **[固定数据部分]** SETTINGS 字节中的 THRS3SEL 位，具体如下所述：

- THRS3SEL = 0（默认值或执行 SELTHR1 命令后假定的值）：使用的阈值是 -THRESH1。
- THRS3SEL = 1（执行 SELTHR3 命令后假定的值）：使用的阈值是 -THRESH3。

提示 *如果同一个轴同时使用“+”和“-”符号，则只需正值或负值满足条件即可，这意味着将应用 OR 运算符（例如，+X < 阈值 || -X < 阈值）。*

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。

8.1.14 PZC (Dh)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任意触发轴以正斜率越过零值，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：
  - 如果发生以正斜率过零事件且 PZC 处于 RESET 位置，则 PP = RP。
  - 如果发生以正斜率过零事件且 PZC 处于 NEXT 位置，则 PP = PP + 1。

8.1.15 NZC (Eh)

说明：如果 FSM 处理的数据样本 (X, Y, Z, V) 的任意触发轴以负斜率越过零值，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：
  - 如果发生以负斜率过零事件且 NZC 处于 RESET 位置，则 PP = RP。
  - 如果发生以负斜率过零事件且 NZC 处于 NEXT 位置，则 PP = PP + 1。

8.1.16 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，请参见第 7.8 节 MLC 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 处于 RESET 位置，则 PP = RP。
  - 如果 CHKDT 处于 NEXT 位置，则 PP = PP + 1。

8.2 命令

命令用于修改程序的流控、输出和同步行为。

命令将立即执行（无需等待新的样本集合）。执行命令时，将程序指针设为下一行，即立即进行评估。

某些命令可能必须在命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，用于在执行 STHR1 命令时动态更改“THRESH1”资源的值：

- AAh（STHR1 命令）
- CDh（第一个参数）
- 3Ch（第二个参数）

当程序指针达到 AAh（STHR1 命令）状态时，器件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，而无需等待新的样本集合。在命令执行完成后，THRESH1 资源值被设为 3CCDh，等于 1.2。

表 7. 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
B5h	SETR	设置器件寄存器值 (ASC) 或使能在 FIFO 中对加速度计通道 2 进行由 FSM 触发的批处理	字节 1：地址 (ASC) 或命令（FIFO 中的批处理） 字节 2：值 (ASC) 或状态（FIFO 中的批处理）
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
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	NEXT 条件为真后不重置临时掩码	无
21h	SRTAM1	NEXT 条件为真后重置临时掩码	无
23h	SINMUX	设置输入多路复用器	有关所需参数的详细信息，请参见 第 8.2.21 节 SINMUX (23h)。
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
34h	INCR	将长计数器的值加 1，检查长计数器超时值	无
FDh	DECR	将长计数器的值减 1	无
F6h	RSTLC	复位长计数器	无
F7h	THRXYZ1	使能在无需等待样本集合的情况下执行多个条件	无

操作码	助记符	说明	参数
F8h	THRXYZ0	禁止在无需等待样本集合的情况下执行多个条件	无
41h	JMP	两个 NEXT 条件的跳转地址	字节 1：条件 字节 2：RESET 条件跳转地址 字节 3：NEXT 条件跳转地址
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	NEXT 条件为真时清零时间计数器 TC	无
7Ch	SCTC1	NEXT 条件为真时不清零时间计数器 TC	无
C7h	UMSKIT	设置 OUTS 时取消屏蔽中断生成	无
EFh	MSKITEQ	在设置 OUTS 时如果 OUTS 不变，则屏蔽中断生成	无
F5h	MSKIT	设置 OUTS 时屏蔽中断生成	无

8.2.1 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，基于使用的 MSKIT / MSKITEQ / UMSKIT 命令）
- 通过将 **[固定数据部分]** 的 CONFIG_B 字节的 STOPDONE 位置 1 来停止自身。要重启程序，用户应禁止和使能 FSM_ENABLE (46h) 寄存器中的相应状态机位。在这种情况下，启动例程将执行。有关启动例程的更多信息，请参见第 10 节 启动例程。

8.2.2 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，基于使用的 MSKIT / MSKITEQ / UMSKIT 命令）
- PP = RP

8.2.3 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。此外，还会将临时掩码值复位为其默认值。
参数：无
动作：

- 将结果掩码输出到 OUTSx 寄存器
- 将临时掩码复位为默认值
- 产生中断（如果使能，基于使用的 MSKIT / MSKITEQ / UMSKIT 命令）
- PP = RP

8.2.4 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

8.2.5 CRP (44h)

说明：CRP 命令将复位指针清零至起始位置（程序代码开头）。
参数：无
动作：

- RP = 程序代码开头
- PP = PP + 1

8.2.6 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

- 第一个参数：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）。
- 第二个参数：要写入第一个参数地址的新值（8 位）。

动作：

- 通过第一个参数寻址的字节值 = 第二个参数
- PP = PP + 3

8.2.7 SETR (B5h)

8.2.7.1 *ASC 功能*

说明：SETR 命令用于实现自适应自配置 (ASC) 功能，该功能支持器件自行重新配置，无需主机干预。下表列出了可以由 FSM 执行写操作的寄存器。

表 8. **ASC FSM 主页寄存器**

主页寄存器	寄存器地址（主机）	寄存器地址 (FSM)
FIFO_CTRL1	07h	07h
FIFO_CTRL2	08h	08h
FIFO_CTRL3	09h	09h
FIFO_CTRL4	0Ah	0Ah(1)
CTRL1	10h	10h
CTRL2	11h	11h
CTRL6	15h	15h
CTRL7	16h	16h
CTRL8	17h	17h
CTRL9	18h	18h
CTRL10	19h	19h

1. *不支持更改 FIFO_MODE。*

表 9. **ASC FSM 嵌入功能寄存器**

嵌入功能寄存器	寄存器地址（主机）	寄存器地址 (FSM)
EMB_FUNC_EN_A	04h	01h(1)
EMB_FUNC_EN_B	05h	02h(1)
FSM_ENABLE	46h	03h(1)
EMB_FUNC_FIFO_EN_A	44h	05h(1)
EMB_FUNC_FIFO_EN_B	45h	06h(1)

1. *对嵌入功能寄存器的访问由 FSM 自动处理。*

对上述器件寄存器的写访问是互斥的：FUNC_CFG_ACCESS (01h) 寄存器的 FSM_WR_CTRL_EN 位用于使能主机或 FSM 对上述器件寄存器的写访问。对该位执行写操作后，通过查看 CTRL_STATUS (1Ah) 寄存器的 FSM_WR_CTRL_STATUS 位置 1 还是清零来确认控制器的更改。具体如下所述：

- FSM_WR_CTRL_STATUS 清零：所有器件寄存器可以写入，但仅可通过标准接口写入。
- FSM_WR_CTRL_STATUS 置 1：上述器件寄存器受 FSM 控制，处于只读模式且仅可通过标准接口读取。

参数：两个字节

- 第一个参数：要修改值的寄存器的地址（8 位），即上表中“寄存器地址 (FSM)”列中的地址。如果此参数等于 0x00，则第二个参数用作此后第一个 SETR 命令写操作的位掩码。
- 第二个参数：要写入第一个参数寄存器地址的新值（8 位）。如果第一个参数为 0x00，则此参数用作此后第一个 SETR 命令写操作的位掩码。

动作：

- 通过第一个参数寻址的寄存器值 = 第二个参数
- PP = PP + 3

提示 *默认的写操作位掩码是 0xFF，表示写入所有位。通过设置位掩码，可以只更改寄存器中特定位的值，而不更改其他位的值。例如，如果必须在不更改加速度计功耗模式的情况下更改加速度计 ODR，则必须连续执行以下两条 SETR 命令：*

1. *SETR 0x00 0x0F（位掩码等于 0x0F）*
2. *SETR 0x10 0x06（新的加速度计 ODR 等于 120 Hz）*

8.2.7.2 *由 FSM 触发的 FIFO 中的批处理*

说明：SETR 命令可用于在 FSM 检测到特定运动模式时，使能在 FIFO 中对加速度计通道 2 进行批处理。要使能该功能，需将 FIFO_CTRL2 (08h) 寄存器的 XL_DualC_BATCH_FROM_FSM 位置 1。置 1/清零后，便可使用 SETR 命令来使能/禁止在 FIFO 中对加速度计通道 2 进行批处理，具体如下所述。

参数：两个字节

- 第一个参数：0x32
- 第二个参数：0x01（使能 FIFO 中的批处理）或 0x00（禁止 FIFO 中的批处理）

动作：

- 使能/禁止在 FIFO 中对加速度计通道 2 进行批处理
- PP = PP + 3

8.2.8 SELMA (66h)

说明：SELMA 命令用于将 MASKA / TMASKA 设为当前掩码。

参数：无

动作：

- 选择 MASK_A。将 **[固定数据部分]** 的 SETTINGS(MASKSEL[1:0]) 位设为 00。
- PP = PP + 1

8.2.9 SELMB (77h)

说明：SELMB 命令用于将 MASKB / TMASKB 设为当前掩码。

参数：无

动作：

- 选择 MASK_B。将 **[固定数据部分]** 的 SETTINGS(MASKSEL[1:0]) 位设为 01。
- PP = PP + 1

8.2.10 SELMC (88h)

说明：SELMC 命令用于将 MASKC / TMASKC 设为当前掩码。

参数：无

动作：

- 选择 MASK_C。将 **[固定数据部分]** 的 SETTINGS(MASKSEL[1:0]) 位设为 10。
- PP = PP + 1

8.2.11 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，基于使用的 MSKIT / MSKITEQ / UMSKIT 命令）
- PP = PP + 1

8.2.12 STHR1 (AAh)

说明：STHR1 命令将 THRESH1 值设为所需的新值。THRESH1 是一个半精度浮点数（16 位）。

参数：两个字节

- 第一个参数：THRESH1 LSB 值（8 位）
- 第二个参数：THRESH1 MSB 值（8 位）

动作：

- 为 THRESH1 设置新值
- PP = PP + 3

8.2.13 STHR2 (BBh)

说明：STHR2 命令将 THRESH2 值设为所需的新值。THRESH2 是一个半精度浮点数（16 位）。

参数：两个字节

- 第一个参数：THRESH2 LSB 值（8 位）
- 第二个参数：THRESH2 MSB 值（8 位）

动作：

- 为 THRESH2 设置新值
- PP = PP + 3

8.2.14 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

8.2.15 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

8.2.16 REL (FFh)

命令：REL 命令用于释放轴的临时掩码信息。

参数：无

动作：

- 将当前的临时掩码重置为默认值
- PP = PP + 1

8.2.17 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

8.2.18 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

8.2.19 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 **[固定数据部分]** 的 SETTINGS(R_TAM) 位设为 0。
- PP = PP + 1

8.2.20 SRTAM1 (21h)

说明：SRTAM1 命令用于在 NEXT 条件为真时复位临时掩码。
参数：无
动作：

- 临时轴掩码值在有效 NEXT 条件后复位。将 **[固定数据部分]** 的 SETTINGS(R_TAM) 位置 1。
- PP = PP + 1

8.2.21 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。
标准 SINMUX 命令还可用于选择经 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。
- 必须将第二个 MLC 滤波器 [0 0 0 $G_v^{(3)}$] 应用于传感器范数。
- 必须将第三个 MLC 滤波器 [H_x H_y H_z $H_v^{(2)}$] 应用于传感器轴。
- 必须将第四个 MLC 滤波器 [0 0 0 $J_v^{(3)}$] 应用于传感器范数。

可以使用扩展 sinmux 功能来克服上述对 MLC 滤波器的顺序和类型（轴或范数）的要求，该功能可用于获取任何经 MLC 滤波的数据。

提示 *如果用户只需将两个滤波器应用于传感器轴（不需要对传感器范数应用滤波器），也需要配置对传感器范数应用的第二个 MLC 滤波器，即使该滤波器不会使用也是如此。此外，如果用户只需将两个滤波器应用于传感器范数（不需要对传感器轴应用滤波器），则必须配置上述所有四个 MLC 滤波器。*

此外，扩展 sinmux 功能支持使用 SINMUX 命令来选择长计数器的值或任何计算出的 MLC 滤波器或特征。

参数：第一个参数等于 9 时为三个字节，否则为一个字节。

- 第一个参数：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：外部传感器 [m_x m_y m_z m_v]
  - 3：来自机器学习内核(1) 的第一个经滤波的信号 [F_x F_y F_z F_v(2)]
  - 4：来自机器学习内核(1) 的第三个经滤波的信号 [H_x H_y H_z H_v(2)]
  - 5：来自机器学习内核(1) 的第二个经滤波的信号范数 [0 0 0 G_v(3)]
  - 6：来自机器学习内核(1) 的第四个经滤波的信号范数 [0 0 0 J_v(3)]
  - 7：陀螺仪积分信号 [d_x d_y d_z d_v]
  - 8：长计数器的值 [LC_x 0 0 0]
  - 9：来自机器学习内核的任何滤波器或特征 [K_x K_y K_z K_v]。滤波器或特征通过第二个和第三个参数进行选择
- 第二个参数（仅当第一个参数等于 9 时需要使用）：MLC 滤波器或特征 IDENTIFIER[7:0]
- 第三个参数（仅当第一个参数等于 9 时需要使用）：MLC 滤波器或特征 IDENTIFIER[15:8]

配置 MLC 后，意法半导体工具会生成配置文件，其中会指明滤波器和特征的标识符，如下图所示。

图 13. MLC 滤波器和特征的标识符

```
-- <MLC1_SRC>DT1,0='negative',4='positive'

-- FILTER_IIR1_ACC_X -> 023Ch
-- FILTER_IIR1_ACC_Y -> 023Eh
-- FILTER_IIR1_ACC_Z -> 0240h
-- F1_MEAN_on_ACC_X -> 0242h
-- F2_MEAN_on_ACC_Y -> 0244h
-- F3_MEAN_on_ACC_Z -> 0246h
```

提示　使用 SINMUX 2 命令将 Qvar 传感器作为外部传感器进行处理。在这种情况下，在 Qvar 数据值比较过程中要使用的掩码值可以是 0x80（对应于 +X 轴）或 0x40（对应于 -X 轴）。

提示　当打算通过 SINMUX 9 命令选择对轴应用的滤波器时，建议配置与滤波后的 X 轴相关的标识符。在这种情况下，比较过程中使用的掩码值遵循轴的正常顺序。

提示　在长计数器值比较过程中要使用的掩码值为 0x80（对应于 +X 轴），阈值为无符号的 15 位值。

提示　在范数滤波或特征值比较过程中要使用的掩码值为 0x80（对应于 +X 轴）或 0x40（对应于 -X 轴）。

动作：

- 根据设置的参数相应地选择输入信号。根据所选输入源信号配置 **[固定数据部分]** 的 SETTINGS(IN_SEL[2:0]) 位和 **[可变数据部分]** 的 EXT_SINMUX(IN_SEL[3]) 位。
- 如果第一个参数等于 9，PP + 4；否则，PP + 2。

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

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

(3) G_v / J_v 由 MLC 提供。

8.2.22 STIMER3 (24h)

说明：STIMER3 命令用于为 TIMER3 设置新值。

参数：一个字节

- 第一个参数：新的 TIMER3 值

动作：

- 设置新的 TIMER3 值
- PP = PP + 2

8.2.23 STIMER4 (31h)

说明：STIMER4 命令用于为 TIMER4 设置新值。

参数：一个字节

- 第一个参数：新的 TIMER4 值

动作：

- 设置新的 TIMER4 值
- PP = PP + 2

8.2.24 INCR (34h)

说明：INCR 命令用于将长计数器的值加一。长计数器的值存储在 FSM_LONG_COUNTER_L (48h) 和 FSM_LONG_COUNTER_H (49h) 嵌入功能寄存器中，并钳位到 FSM_LC_TIMEOUT_L (7Ah) 和 FSM_LC_TIMEOUT_H (7Bh) 嵌入高级功能寄存器中存储的长计数器超时值。当长计数器的值等于长计数器超时值时，将产生中断。有关 FSM 长计数器中断的详细信息，请参见第 4 节 FSM 中断状态和信号。

参数：无

动作：

- 将长计数器的值加一，如果长计数器的值等于长计数器超时值，则产生中断。
- PP = PP + 1

8.2.25 DECR (FDh)

说明：DECR 命令用于将长计数器的值减一。长计数器值存储在 FSM_LONG_COUNTER_L (48h) 和 FSM_LONG_COUNTER_H (49h) 嵌入功能寄存器中，并被钳位至零。

参数：无

动作：

- 将长计数器值减一。
- PP = PP + 1

8.2.26 RSTLC (F6h)

说明：RSTLC 命令用于重置长计数器的值。长计数器的值存储在 FSM_LONG_COUNTER_L (48h) 和 FSM_LONG_COUNTER_H (49h) 嵌入功能寄存器中。

参数：无

动作：

- 重置长计数器的值。
- PP = PP + 1

8.2.27 THRXYZ1 (F7h)

说明：THRXYZ1 命令用于使能这样的模式：针对同一个数据样本执行多个条件。使能该模式后，FSM 使用同一个数据样本执行所有条件，直到 THRXYZ0 命令执行为止。THRXYZ1 命令与 THRXYZ0 命令之间的所有指令可视为一条指令。在这种情况下，命令和条件会影响程序流，具体如下所述：

- 如果当前状态是命令，则立即执行。
- 如果当前状态是条件：
  - 如果 RESET 条件为真，则程序指针设为 RP，并且 **[可变数据部分]** 的 EXT_SINMUX 字节的 THRXYZ1 位设为 0，以恢复默认条件执行模式（参见 THRXYZ0 命令）。处理下一个数据样本时会执行新状态。
  - 如果 NEXT 条件为真，则程序指针设为下一个状态，并且会立即执行该状态。
  - 如果 NEXT 条件为假，则程序指针设为 THRXYZ1 命令的地址，并且 FSM 在处理下一个数据样本时会再次评估 THRXYZ1 命令与 THRXYZ0 命令之间的指令。

在使能该模式期间，必须考虑以下几点注意事项。

- 在 SCTC1 模式下只能使用一个定时器。
- 如果在到达 THRXYZ0 命令之前发出了多个 OUTC 命令，则 OUTS 寄存器将只包含与最后执行的条件的结果相关的信息。
- 不支持 JMP 和 SRP 命令。
- 不支持 SINMUX 0 命令，必须用 SINMUX 9 01D4h 命令代替，这意味着将 IDENTIFIER[7:0] 参数设为 D4h，将 IDENTIFIER[15:8] 参数设为 01h。

参数：无

动作：

- 将 **[可变数据部分]** 的 EXT_SINMUX 字节的 THRXYZ1 位置 1。
- PP = PP + 1

8.2.28 THRXYZ0 (F8h)

说明：THRXYZ0 命令用于恢复默认模式，在该模式下，每个条件仅针对一个数据样本执行。在这种情况下，命令和条件会影响程序流，具体如下所述：

- 如果当前状态是命令，则立即执行。
- 如果当前状态是条件：
  - 如果 RESET 条件为真，则程序指针设为 RP。处理下一个数据样本时会执行新状态。
  - 如果 NEXT 条件为真，则程序指针设为下一个状态。处理下一个数据样本时会执行新状态。
  - 如果 RESET 和 NEXT 条件均为假，则 PP 不变。处理下一个数据样本时会再次评估该条件。

参数：无

动作：

- 将 **[可变数据部分]** 的 EXT_SINMUX 字节的 THRXYZ1 位设为 0。
- PP = PP + 1

8.2.29 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

- 第一个参数：NEXT1 | NEXT2 条件
- 第二个参数：如果 NEXT1 条件为真，则跳转地址
- 第三个参数：如果 NEXT2 条件为真，则跳转地址

NEXT1 条件在 NEXT2 条件之前评估。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**] 中 CONFIG_B 字节的 JMP 位设为 1。评估 NEXT1 | NEXT2 条件：
  - 如果 NEXT1 条件为真，则 PP = 第二个参数地址。
  - 否则，如果 NEXT2 条件为真，则 PP = 第三个参数地址。
  - 否则，等待新的样本集合并再次评估 NEXT1 | NEXT2 条件。

8.2.30 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时不再清除角度积分值（默认行为），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

8.2.31 SMA (43h)

说明：SMA 命令用于为 MASKA 和 TMASKA 设置新值。

参数：一个字节

- 第一个参数：新的 MASKA 和 TMASKA 值。

动作：

- 设置新的 MASKA 和 TMASKA 值
- PP = PP + 2

8.2.32 SMB (DFh)

说明：SMB 命令用于为 MASKB 和 TMASKB 设置新值。

参数：一个字节

- 第一个参数：新的 MASKB 和 TMASKB 值

动作：

- 设置新的 MASKB 和 TMASKB 值
- PP = PP + 2

8.2.33 SMC (FEh)

说明：SMC 命令用于为 MASKC 和 TMASKC 设置新值。

参数：一个字节

- 第一个参数：新的 MASKC 和 TMASKC 值

动作：

- 设置新的 MASKC 和 TMASKC 值
- PP = PP + 2

8.2.34 SCTC0 (5Bh)

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

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

8.2.35 SCTC1 (7Ch)

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

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

8.2.36 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

8.2.37 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：

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

8.2.38 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

9 FSM 配置示例

本节包含一个示例，该示例解释了配置 LSM6DSV16X FSM 必须完成的所有写操作。必须遵循以下配置步骤：

- 配置嵌入功能寄存器组中的 FSM 寄存器。
- 配置嵌入高级功能寄存器组中的 FSM 寄存器。
- 配置 LSM6DSV16X 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 30 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 14. FSM 配置示例

	页 - 地址	名称	7	6	5	4	3	2	1	0
PROGRAM1	4 - 00 h	CONFIG A	01（1 个阈值）		01（1 个掩码）		00		01（1 个短定时器）	
	4 - 01h	CONFIG B	0	0	0	0	0	0	0	0
	4 - 02 h	SIZE	10h（16 字节）							
	4 - 03 h	SETTINGS	00		0	0	0	00		
	4 - 04 h	RESET POINTER	00h							
	4 - 05 h	PROGRAM POINTER	00h							
	4 - 06 h	THRESH1	B7AEh (-0.480)							
	4 - 07 h									
	4 - 08 h	MASKA	80h (+X)							
	4 - 09 h	TMASKA	00h							
	4 - 0Ah	TC	00h							
	4 - 0Bh	TIMER3	10h（16 个样本）							
	4 - 0Ch	GNTH1 | TI3	53h							
	4 - 0Dh	OUTC	99h							
	4 - 0Eh	GNTH1 | NOP	50h							
	4 - 0Fh	STOP	00h							
PROGRAM2	4 - 10 h	CONFIG A	01（1 个阈值）		01（1 个掩码）		00		00	
	4 - 11h	CONFIG B	0	0	0	0	0	0	0	0
	4 - 12 h	SIZE	0Ch（12 字节）							
	4 - 13 h	SETTINGS	00		0	0	0	00		
	4 - 14 h	RESET POINTER	00h							
	4 - 15 h	PROGRAM POINTER	00h							
	4 - 16 h	THRESH1	3C66h (1.100)							
	4 - 17 h									
	4 - 18 h	MASKA	02h (+V)							
	4 - 19 h	TMASKA	00h							
	4 - 1Ah	NOP | GNTH1	05h							
	4 - 1Bh	CONTREL	22h							

FSM 配置操作必须在加速度计和陀螺仪传感器均处于掉电模式下执行。有关完整的器件配置，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掉电模式
2. 将 00h 写入寄存器 11h // 将陀螺仪传感器设为掉电模式
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 // FSM_ODR[2:0] = 001 (30 Hz)
6. 将 03h 写入寄存器 46h // FSM_ENABLE = 03h
7. 将 01h 写入寄存器 0Bh // FSM_INT1 = 01h
8. 将 02h 写入寄存器 0Fh // FSM_INT2 = 02h
9. 将 40h 写入寄存器 17h // PAGE_RW：使能写操作
10. 将 11h 写入寄存器 02h // 使能对嵌入高级功能寄存器的访问，PAGE_SEL = 1
11. 将 7Ah 写入寄存器 08h // PAGE_ADDRESS = 7Ah
12. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_LONG_COUNTER_L
13. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_LONG_COUNTER_H
14. 将 02h 写入寄存器 09h // 将 02h 写入寄存器 FSM_PROGRAMS
15. 将 02h 写入寄存器 09h // 执行空写操作，以增加写入地址
16. 将 00h 写入寄存器 09h // 将 00h 写入寄存器 FSM_START_ADDRESS_L
17. 将 04h 写入寄存器 09h // 将 04h 写入寄存器 FSM_START_ADDRESS_H
18. 将 41h 写入寄存器 02h // PAGE_SEL = 4
19. 将 00h 写入寄存器 08h // PAGE_ADDRESS = 00h
20. 将 51h 写入寄存器 09h // CONFIG_A
21. 将 00h 写入寄存器 09h // CONFIG_B
22. 将 10h 写入寄存器 09h // SIZE
23. 将 00h 写入寄存器 09h // SETTINGS
24. 将 00h 写入寄存器 09h // RESET POINTER
25. 将 00h 写入寄存器 09h // PROGRAM POINTER
26. 将 AEh 写入寄存器 09h // THRESH1 LSB
27. 将 B7h 写入寄存器 09h // THRESH1 MSB
28. 将 80h 写入寄存器 09h // MASKA
29. 将 00h 写入寄存器 09h // TMASKA
30. 将 00h 写入寄存器 09h // TC
31. 将 10h 写入寄存器 09h // TIMER3
32. 将 53h 写入寄存器 09h // GNTH1 | TI3
33. 将 99h 写入寄存器 09h // OUTC
34. 将 50h 写入寄存器 09h // GNTH1 | NOP
35. 将 00h 写入寄存器 09h // STOP（必须包含偶数个 SIZE 字节）
36. 将 50h 写入寄存器 09h // CONFIG_A
37. 将 00h 写入寄存器 09h // CONFIG_B
38. 将 0Ch 写入寄存器 09h // SIZE
39. 将 00h 写入寄存器 09h // SETTINGS
40. 将 00h 写入寄存器 09h // RESET POINTER
41. 将 00h 写入寄存器 09h // PROGRAM POINTER
42. 将 66h 写入寄存器 09h // THRESH1 LSB
43. 将 3Ch 写入寄存器 09h // THRESH1 MSB
44. 将 02h 写入寄存器 09h // MASKA
45. 将 00h 写入寄存器 09h // TMASKA
46. 将 05h 写入寄存器 09h // NOP | GNTH1
47. 将 22h 写入寄存器 09h // CONTREL
48. 将 01h 写入寄存器 02h // 禁止对嵌入高级功能寄存器的访问，PAGE_SEL = 0

49. 将 00h 写入寄存器 17h	// PAGE_RW：禁止写操作
50. 将 00h 写入寄存器 01h	// 禁止对嵌入功能寄存器的访问
51. 将 02h 写入寄存器 5Eh	// MD1_CFG(INT1_EMB_FUNC) = 1
52. 将 02h 写入寄存器 5Fh	// MD2_CFG(INT2_EMB_FUNC) = 1
53. 将 04h 写入寄存器 10h	// CTRL1 = 04h (30 Hz)

10 启动例程

FSM 使能后，将自动执行启动例程。该例程在 PROGRAM POINTER 字节设为 0 的情况下执行，具体执行以下任务：

- 将 CONFIG_B 字节中的 STOPDONE 和 JMP 位复位。
- 将 PP 和 RP 指针初始化为第一行代码的地址。
- 将 SETTINGS 位域初始化为默认值 0x20，表示：
  - MASKSEL = 00
  - SIGNED = 1
  - R_TAM = 0
  - THRS3SEL = 0
  - IN_SEL = 000
- 清零相关输出寄存器 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，则使用设定的抽取时间值 (DESC = DEST) 初始化抽取计数器。
- 如果声明了先前的轴符号资源，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)。

启动例程执行时，程序始终从已知状态重启，而与停止方式无关。但应注意，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 而不是阈值 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)

11 状态机配置示例

11.1 切换

切换是一个简单状态机配置，在该配置下，每出现 n 个样本就会产生一次中断。
目的是使用定时器对 n 个样本进行计数。

图 15. 切换状态机示例

字节编号	名称	7	6	5	4	3	2	1	0
00h	CONFIG A	00		00		00		01（1 个短定时器）	
01h	CONFIG B	0	0	0	0	0	0	0	0
02h	SIZE	0Ah（10 字节）							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h	TC	00h							
07h	TIMER3	10h（16 个样本）							
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：无需样本集合即可执行 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 个样本产生一个中断。可通过配置 TI3 获得所需的切换周期，该周期取决于所配置的 FSM_ODR。

11.2 自适应自配置 (ASC)

本示例说明如何使用 ASC 功能根据具体的运动事件重新配置器件。

下面的程序一开始将加速度计配置为 30 Hz 的低功耗模式，并将陀螺仪配置为掉电模式。当检测到唤醒事件时，两个传感器均配置为 120 Hz 的高性能模式。如果静止了一段时间，加速度计将设回 30 Hz 的低功耗模式，陀螺仪将设回掉电模式。

图 16. **ASC 状态机示例**

字节编号	名称	7	6	5	4	3	2	1	0
00h	CONFIG A	01（1 个阈值）		01（1 个掩码）		0		01（1 个短定时器）	
01h	CONFIG B	0	0	0	1	0	0	0	0
02h	SIZE	20h（32 字节）							
03h	SETTINGS	0		0	0	0	0		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h 07h	THRESH1	3C66h (1.100)							
08h	MASKA	02h (+V)							
09h	TMASKA	00h							
0Ah	TC	00h							
0Bh	TIMER3	78h（120 个样本）							
0Ch	PAS	00h							
0Dh	MSKIT	F5h							
0Eh	SETR	B5h							
0Fh	10h	10h							
10h	63h	63h							
11h	SETR	B5h							
12h	11h	11h							
13h	00h	10h							
14h	NOP | GNTH1	05h							
15h	SETR	B5h							
16h	10h	10h							
17h	06h	06h							
18h	SETR	B5h							
19h	11h	11h							
1Ah	06h	06h							
1Bh	SRP	33h							
1Ch	GNTH1 | TI3	53h							
1Dh	CRP	44h							
1Eh	CONTREL	22h							
1Fh	STOP	00h							

指令部分说明

PP = 0Dh：无需样本集合即可执行 MSKIT 命令。PAS 字节中的 MSKIT 位置 1。PP = PP + 1。

PP = 0Eh：无需样本集合即可执行 SETR 命令。寄存器 10h 设为 63h（加速度计传感器配置为 30 Hz 的低功耗模式）。PP = PP + 3。

PP = 11h：无需样本集合即可执行 SETR 命令。寄存器 11h 设为 00h（陀螺仪传感器配置为掉电模式）。PP = PP + 3。

PP = 14h：每次生成新样本时都会评估此条件。如果加速度信号的向量（幅度）大于 THRESH1，则 PP = PP + 1。

PP = 15h：无需样本集合即可执行 SETR 命令。寄存器 10h 设为 06h（加速度计传感器配置为 120 Hz 的高性能功耗模式）。PP = PP + 3。

PP = 18h：无需样本集合即可执行 SETR 命令。寄存器 11h 设为 06h（陀螺仪传感器配置为 120 Hz 的高性能模式）。PP = PP + 3。

PP = 1Bh：无需样本集合即可执行 SRP 命令。RESET POINTER 设为下一个状态，即 1Ch。PP = PP + 1。

PP = 1Ch：每次生成新样本时都会评估此条件。如果加速度信号的向量（幅度）大于 THRESH1，则 PP = RP，TC 设为 TI3。如果连续 TI3 个样本的加速度信号的向量（幅度）均小于 THRESH1，则 PP = PP + 1。

PP = 1Dh：无需样本集合即可执行 CRP 命令。RESET POINTER 设为其默认值，即 0Dh。PP = PP + 1。

PP = 1Eh：无需样本集合即可执行 CONTREL 命令。这种情况下并不会因为执行 MSKIT 命令而产生中断并使程序复位。PP = RP = 0Dh。

在本示例中，唤醒阈值为 1.1 *G*，定时器为 120 个样本（即 1 秒）。

11.3 自由落体

该功能用于检测系统的自由落体（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，所有轴上的加速度均应小于所配置的阈值，并保持所配置的最短持续时间。检测到这一情况时，将生成一个中断。

图 17. 自由落体检测状态机配置示例

字节编号	名称	7	6	5	4	3	2	1	0
00h	CONFIG A	01（1 个阈值）		01（1 个掩码）		00		01（1 个短定时器）	
01h	CONFIG B	0	0	0	0	0	0	0	0
02h	SIZE	12h（18 字节）							
03h	SETTINGS	00		0	0	0	00		
04h	RESET POINTER	00h							
05h	PROGRAM POINTER	00h							
06h 07h	THRESH1	34CDh (0.300)							
08h	MASKA	A8h (+X, +Y, +Z)							
09h	TMASKA	00h							
0Ah	TC	00h							
0Bh	TIMER3	03h（3 个样本）							
0Ch	SSIGNO	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。SETTINGS 字节的 SIGNED 位设为 0，表示设置了无符号比较模式。PP = PP + 1。

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

PP = 0Eh：如果某一轴上的加速度大于 THRESH1，则 PP = RP。如果对于连续的三个样本，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。这将产生中断并使 PP 增加 (PP = PP + 1)。

PP = 10h：如果某一轴上的加速度大于 THRESH1，则 PP = RP。这意味着器件不再处于自由落体状态，因此必须重置程序。

在该示例中，将自由落体阈值设为 0.3 *g*，并将自由落体持续时间设为三个样本。

提示 *自由落体持续时间与 FSM_ODR 密切相关：例如，如果将 FSM_ODR 设为 30 Hz，则自由落体持续时间约为 100 ms（30 Hz 时为三个样本）。*

11.4 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相关联：
  - 静止输出为 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 30 Hz 的 60 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使能。在这种情况下，如果器件再次开始移动，FSM 会产生唤醒中断。

图 18. 决策树接口示例

字节编号	名称	7	6	5	4	3	2	1	0
00h	CONFIG A	01（1 个阈值）		01（1 个掩码）		00		10（2 个短定时器）	
01h	CONFIG B	0	0	0	0	1	0	0	0
02h	SIZE	12h（18 字节）							
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 个样本）							
0Ch	TIMER4	3Dh（61 个样本）							
0Dh	DECTREE	00h（选定 0 号决策树，预期输出为 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 0 号决策树，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 失效，则 PP = RP（程序复位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。这将产生中断并使 PP 增加 (PP = PP + 1)。

PP = 11h：如果 TI4 失效，则 PP = RP。

12 有限状态机工具

器件中的有限状态机可通过专用工具进行编程，这种专用工具可作为 Unico GUI 的扩展提供。

12.1 Unico GUI

Unico 是意法半导体产品组合中可用的所有 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（专业 MEMS 工具）的主板进行交互，从而可以在 MEMS 传感器与 PC GUI 之间进行通信。

有关专业 MEMS 工具板的详细信息，请参见 STEVAL-MKI109V3。

Unico GUI 提供支持三种操作系统的三个软件包。

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

Unico GUI 允许以图形或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，以及从现有文件中加载配置。这样，就可以在几秒内对传感器重新编程。

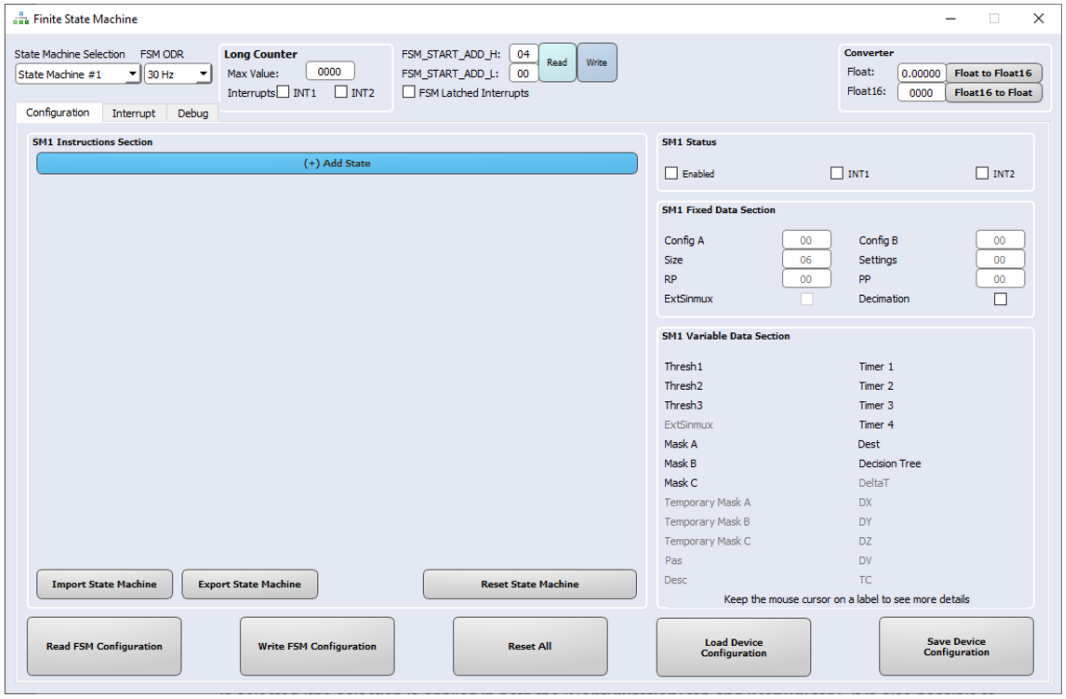
Unico GUI 中提供的有限状态机工具可以自动生成器件的配置文件，从而帮助处理寄存器配置。只需点击几个按钮，便可生成配置文件。用户可利用这些配置文件为器件创建自己的配置库。

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 19. 运行有限状态机工具

加载后，将显示 **[Finite State Machine]**（有限状态机）工具的主窗口。

图 20. [Finite State Machine]（有限状态机）工具



在 **[Finite State Machine]**（有限状态机）工具主窗口的顶部，用户可以选择状态机（该选择也用于 **[Configuration]**（配置）选项卡和 **[Debug]**（调试）选项卡）。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 格式转换为 float16 格式的转换器，反之亦然。转换器用于生成要在 **[可变数据部分]** 的阈值资源中设置的值。

[Finite State Machine]（有限状态机）工具主要由以下三个选项卡组成，将在专门章节中介绍这些选项卡：

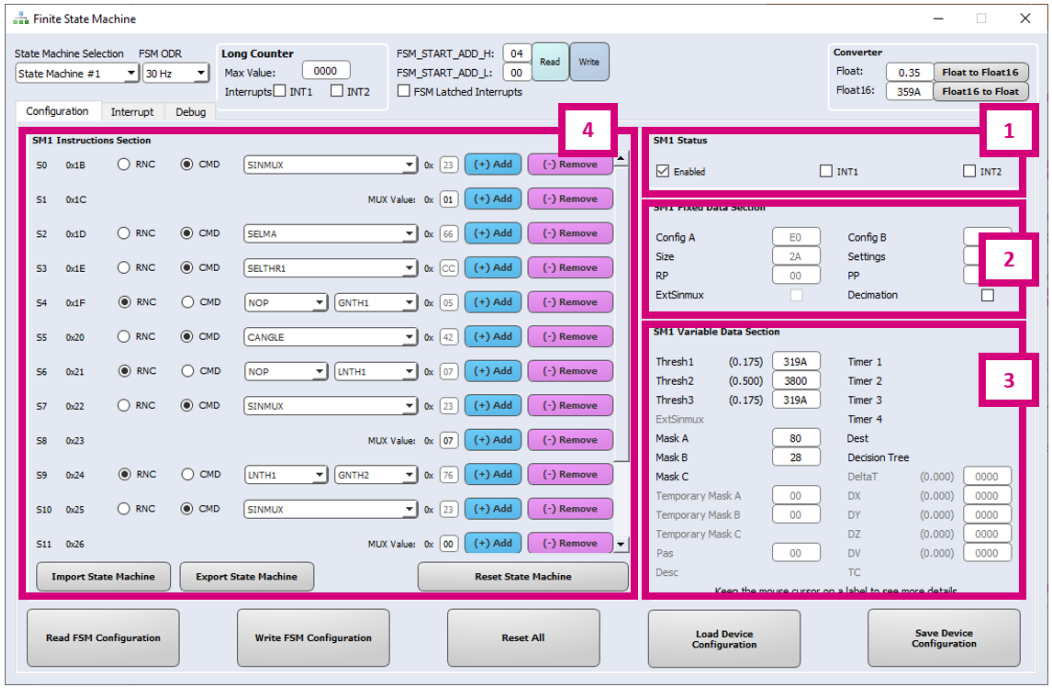
- **[Configuration]**（配置）选项卡（默认选中的选项卡）
- **[Interrupt]**（中断）选项卡
- **[Debug]**（调试）选项卡

12.1.1 配置选项卡

[Finite State Machine]（有限状态机）工具的 **[Configuration]**（配置）选项卡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. **[SMx Status（SMx 状态）]**
2. **[SMx Fixed Data Section（SMx 固定数据部分）]**
3. **[SMx Variable Data Section（SMx 可变数据部分）]**
4. **[SMx Instructions Section（SMx 指令部分）]**

图 21. **[Finite State Machine]（有限状态机）工具 - [Configuration]（配置）选项卡**



在 **[Configuration]**（配置）选项卡的底部，用户可以使用专用按钮管理器件配置：

- **[Read FSM Configuration]**（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。
- **[Write FSM Configuration]**（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）。
- **[Reset All]**（全部复位）：用于复位整个 **[Finite State Machine]**（有限状态机）工具 UI。
- **[Load Device Configuration]**（加载器件配置）：用于加载 .ucf 文件。
- **[Save Device Configuration]**（保存器件配置）：用于生成 .ucf 文件，其中包括传感器配置和 FSM 寄存器配置。

12.1.1.1 *SMx 状态*

[SMx Status]（SMx 状态）组框位于 **[Configuration]**（配置）选项卡的右上角。

图 22. **[Configuration]（配置）选项卡 - [SMx Status]（SMx 状态）**

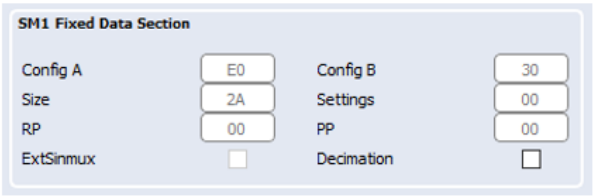
[SMx Status]（SMx 状态）组框允许用户使能/禁止状态机并将中断状态发送到 INT1/INT2 引脚。具体而言：

- **[Enabled]**（使能）复选框用于使能/禁止状态机。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。

12.1.1.2 *SMx 固定数据部分*

[SMx Fixed Data Section]（SMx 固定数据部分）组框在 **[Configuration]**（配置）选项卡的右侧。

图 23. **[Configuration]（配置）选项卡- [SMx Fixed Data Section]（SMx 固定数据部分）**

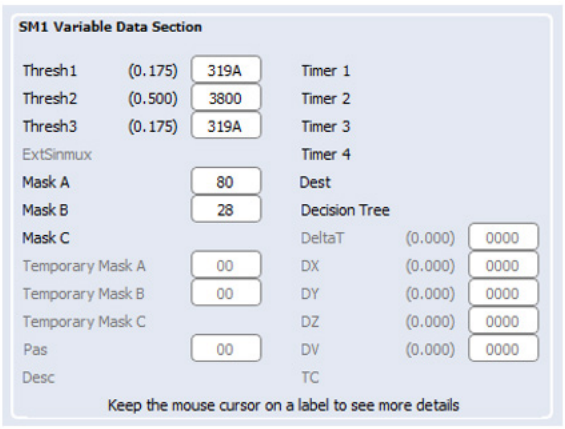


[SMx Fixed Data Section]（SMx 固定数据部分）组框使用户可以获取有关程序的固定数据部分字节的信息。这些字节由 **[Finite State Machine]**（有限状态机）工具自动管理。也可以根据用户需要使能/禁止迟滞和抽取资源。如果使能，则相应资源将显示在 **[SMx Variable Data Section]**（SMx 可变数据部分）组框中。

12.1.1.3 *SMx 可变数据部分*

[SMx Variable Data Section]（SMx 可变数据部分）组框位于 **[Configuration]**（配置）选项卡的右下角。

图 24. **[Configuration]（配置）选项卡 - [SMx Variable Data Section]（SMx 可变数据部分）**

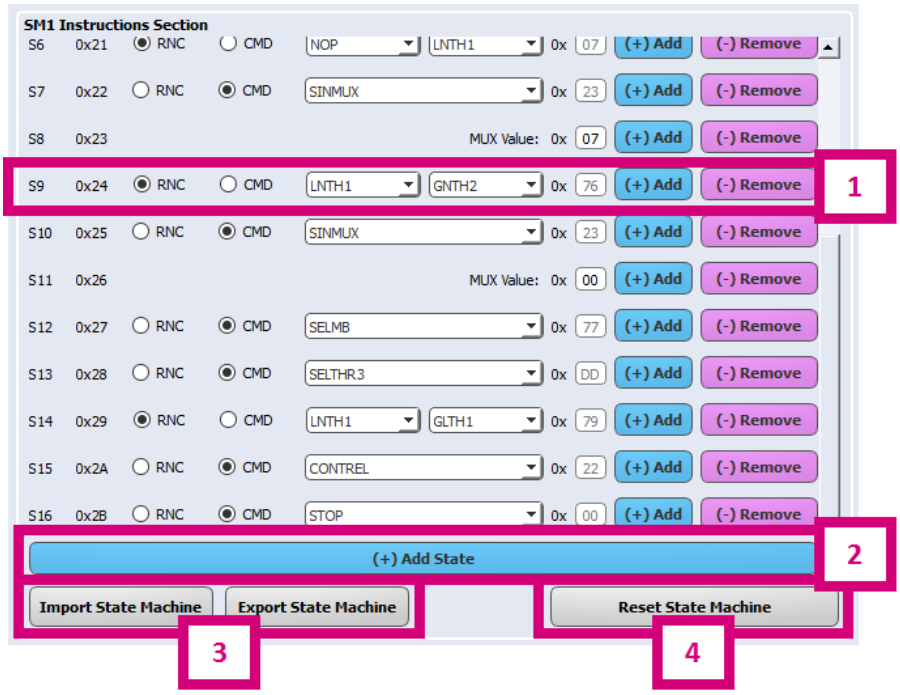


[SMx Variable Data Section]（SMx 可变数据部分）组框可简化资源分配过程：视构成 **[SMx Instruction Section]**（SMx 指令部分）的指令而定，所有需要的资源均会自动显示或隐藏在 **[SMx Variable Data Section]**（SMx 可变数据部分）组框中。用户只需设置显示资源的值。

12.1.1.4 *Smx 指令部分*

[SMx Instructions Section]（SMx 指令部分）组框位于 **[Configuration]**（配置）选项卡的左侧。

图 25. [Configuration]（配置）选项卡– [SMx Instructions Section]（SMx 指令部分）



[SMx Instructions Section]（SMx 指令部分）组框可帮助用户构建算法逻辑。**[SMx Variable Data Section]**（SMx 可变数据部分）组框根据 **[SMx Instructions Section]**（SMx 指令部分）组框中使用的资源动态更新。在 **[SMx Instructions Section]**（SMx 指令部分）组框中，可执行更多操作：

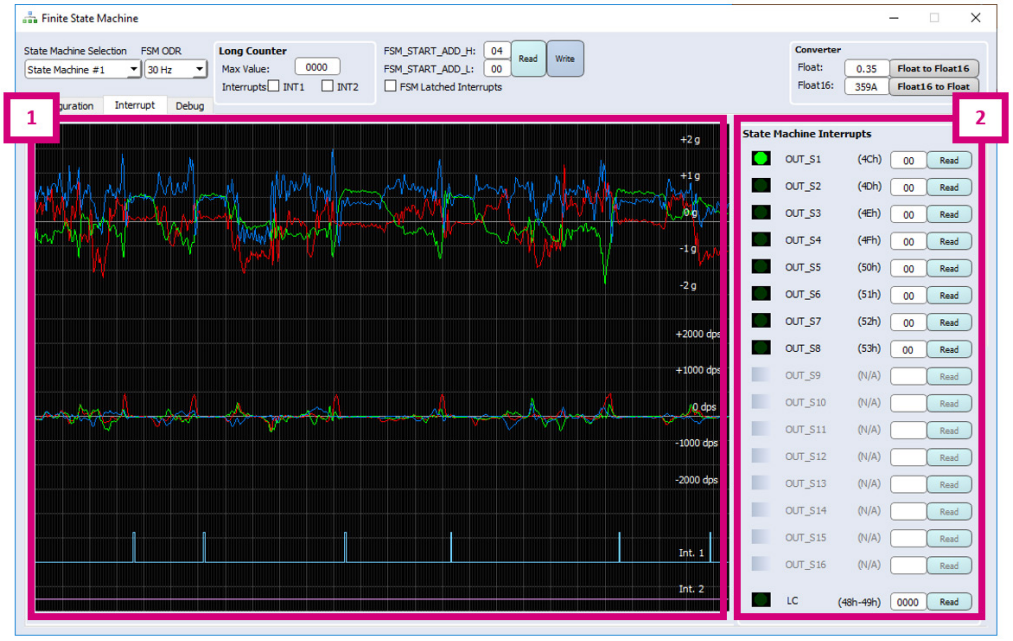
1. 自定义现有状态。单个状态包括：
   - 状态编号 Sx
   - 状态程序的相对十六进制地址（地址 0x00 对应于固定数据部分中的 CONFIG_A 字节）
   - 状态类型和操作码：用户可以使用如下所述的单选按钮和下拉列表来自定义状态：
     - **[RNC]** 单选按钮：状态为 RESET/NEXT 条件。在这种情况下，显示两个下拉列表。左边的下拉列表与 RESET 条件相关，右边的下拉列表与 NEXT 条件相关。
     - **[CMD]** 单选按钮：状态为命令。在这种情况下，将显示一个下拉列表。具有一个或多个参数（通过工具自动显示）的命令需要用户手动配置参数值。
   - **[Add]**（添加）按钮用于在当前状态之前插入新状态。
   - **[Remove]**（删除）按钮用于删除当前状态。
2. **[Add State]**（添加状态）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 State Machine]**（导入状态机）/**[Export State Machine]**（导出状态机）按钮用于以 .fsm 格式导入/导出状态机程序。用户可以使用 .fsm 格式通过一组 .fsm 状态机程序构建整个 FSM 配置。
4. **[Reset State Machine]**（复位状态机）按钮用于将状态机指令部分复位（仅在 UI 上，不在器件中）。

12.1.2 Interrupt（中断）选项卡

[Finite State Machine]（有限状态机）工具的 **[Interrupt]**（中断）选项卡支持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 26 所示。

1. 信号曲线图：根据使能的传感器和中断配置，显示加速度计、陀螺仪和中断信号曲线图。
2. **[State Machine Interrupts]**（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，该 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]**（读取）按钮，可手动读取 OUT_Sx 寄存器的值和长计数器寄存器的值。

图 26. **[Finite State Machine]（有限状态机）工具 - [Interrupt]（中断）选项卡**

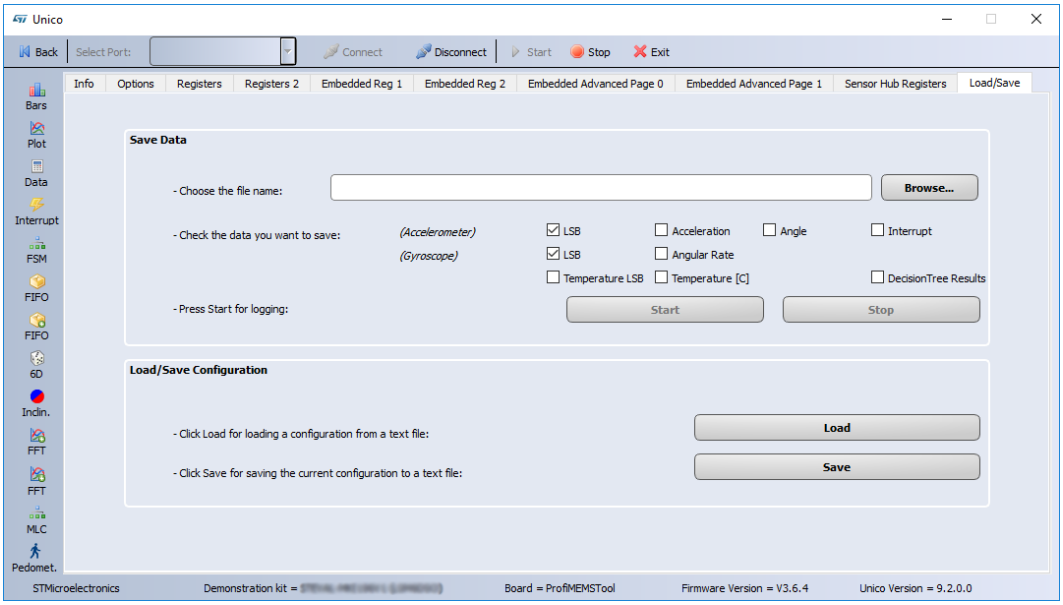


12.1.3 调试选项卡

[Debug]（调试）选项卡可用于将数据注入器件，以检查所配置程序的功能。

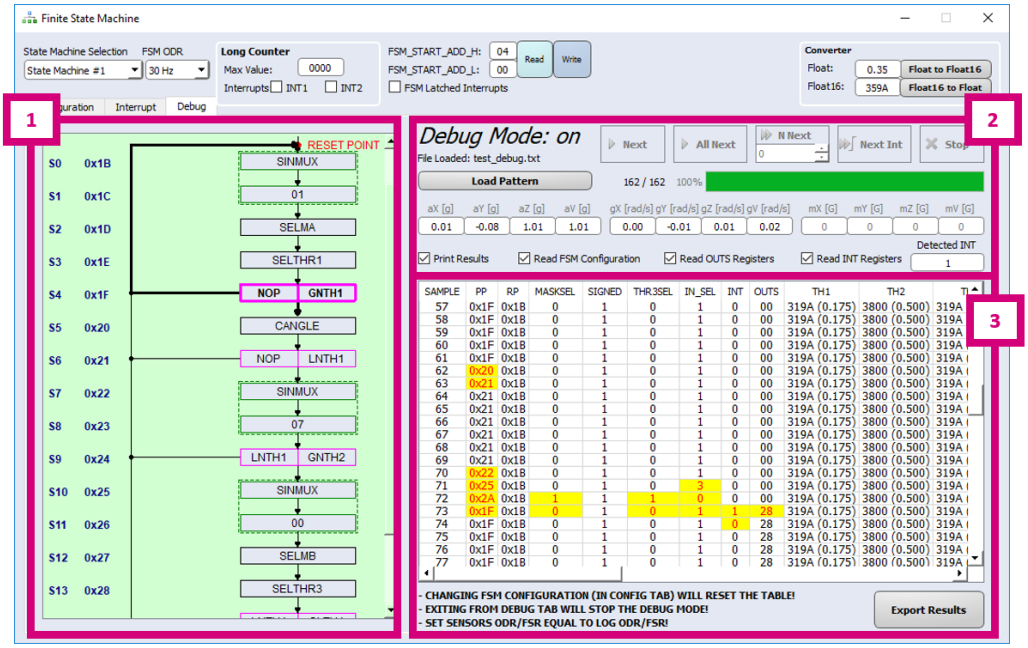
UNICO GUI **[Load/Save]**（加载/保存）选项卡（如下图所示）允许用户获取正确格式化的数据注入程序的日志文件：这些日志文件必须只包含 [LSB] 数据（加速度计和/或陀螺仪，具体取决于用户需求和程序逻辑）。

图 27. **UNICO GUI – [Load/Save]（加载/保存）选项卡**



[Debug]（调试）选项卡窗口如下图所示。

图 28. **[Finite State Machine]（有限状态机）工具 – [Debug]（调试）选项卡**



[Debug]（调试）选项卡的 UI 主要分为三个部分：

1. 状态机流程：状态机流程在此处以图形方式显示。使能调试模式后，当前状态将高亮显示，并根据注入样本和程序行为动态更新。
2. 调试命令：默认情况下，调试模式为关闭。加载日志文件后，调试模式会自动打开，用户可以开始向器件注入数据，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：将样本注入器件后，根据 **[Print Results]**（打印结果）复选框的状态将新行添加到表中。表中的列代表状态机参数和资源，表中的行则与注入的样本有关。参数或资源值改变时，将突出显示相应的单元格。最后，可以导出文本文件格式的表结果。

修订历史

表 10. 文档修订历史

日期	版本	变更
2022 年 11 月 18 日	1	初始版本
2023 年 1 月 31 日	2	少量文本更新

目录

表一览

图一览