

FT64F0AX

ADC APPLICATION NOTE

目录

1. 12BIT ADC 模块	3
1.1. ADC 相关寄存器汇总	4
1.2. ADC 的配置	8
1.2.1. 端口配置	8
1.2.2. 通道选择	8
1.2.3. 触发方式选择	8
1.2.4. 触发源选择	8
1.2.5. 触发类型选择	8
1.2.6. 触发延时配置	9
1.2.7. ADC 参考电压	9
1.2.8. 转换时钟	9
1.2.9. 中断	10
1.2.10. 转换结果的格式	11
1.2.11. 阈值比较	11
1.3. ADC 的工作原理	11
1.3.1. 启动转换	11
1.3.2. 转换完成	12
1.3.3. 终止转换	12
1.3.4. 休眠模式下 ADC 的工作	12
1.3.5. 外部触发器	12
1.3.6. A/D 转换步骤	13
1.4. A/D 采集时间要求	14
2. 应用范例	16
1 联系信息	20

FT64F0Ax ADC 应用

1. 12bit ADC 模块

模数转换器 (Analog-to-digital Converter, ADC) 可将模拟输入信号转换为相应的 12 位二进制表征值。该系列器件采用多个模拟输入复用到一个采样保持电路。采样保持电路的输出与转换器的输入相连接。转换器通过逐次逼近法产生 12 位二进制值, 并将转换结果保存在 ADC 结果寄存器 (ADRESL:ADRESH) 中。ADC 参考电压可用软件选择为 VDD、外部参考电压或内部产生的参考电压。ADC 可在转换完成时产生中断。该中断可用于将器件从休眠唤醒。

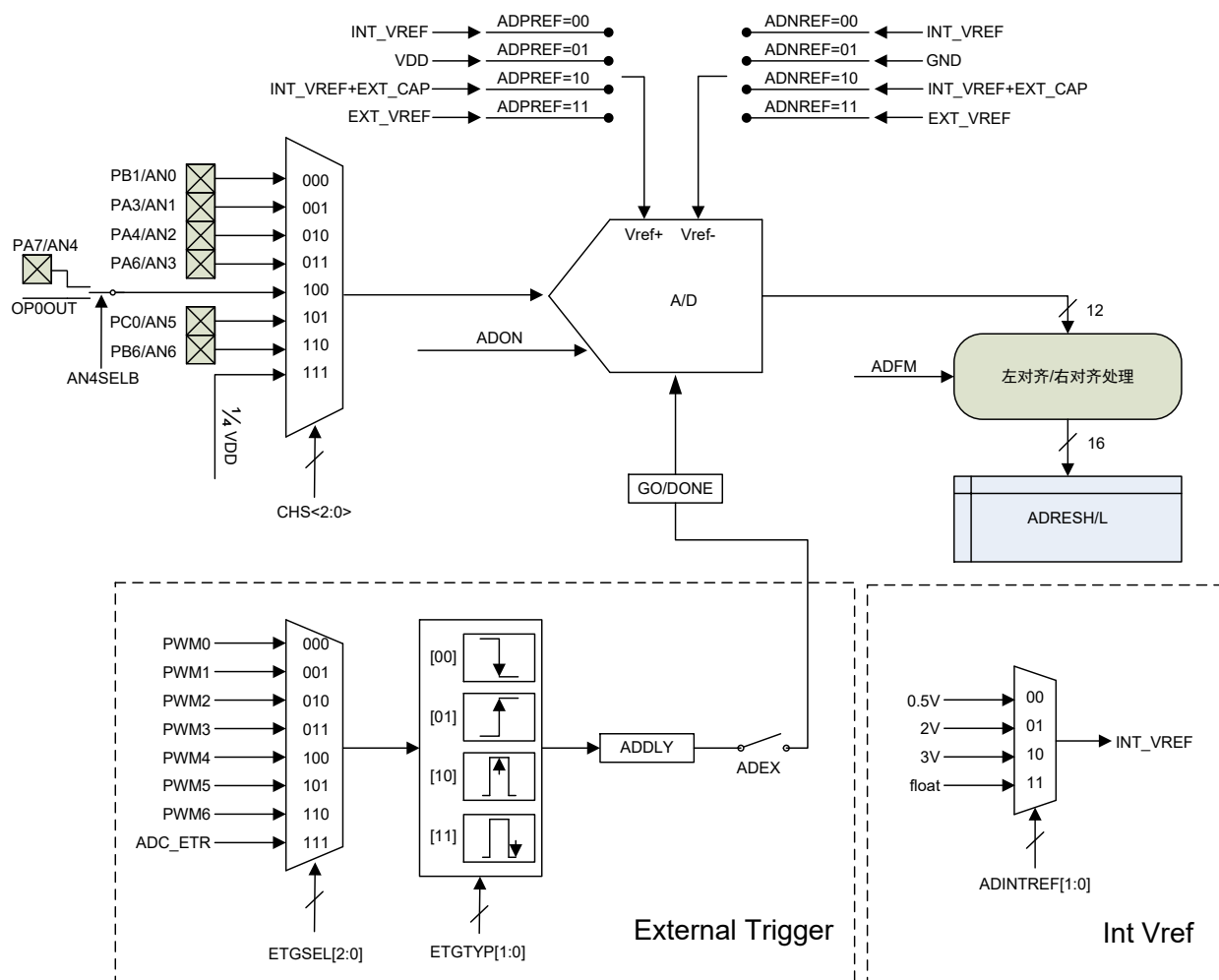


图 1-1 ADC 原理框图

1.1. ADC 相关寄存器汇总

名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
PCKEN	0x9A	UART2EN	I2CEN	UART1EN	SPIEN	TIM4EN	TIM2EN	TIM1EN	ADCEN	0000 0000
CKOCON	0x95	SYSON	CCORDY	DTYSEL		CCOSEL[2:0]			CCOEN	0010 0000
ADRESL	0x09B	A/D 结果寄存器的低字节								0000 0000
ADRESH	0x09C	A/D 结果寄存器的高字节								0000 0000
ADCON0	0x09D	—	CHS[2:0]			—	ADEX	GO/DONE	ADON	-000 -000
ADCON1	0x09E	ADFM	ADCS[2:0]			ADNREF[1:0]		ADPREF[1:0]		0000 0000
ADCON2	0x09F	ADINTREF[1:0]		ETGTYP[1:0]		ADDLY.8	ETGSEL[2:0]			0000 0000
ADDLY	0x01F	ADDLY[7:0] / LEBPRL[7:0]								0000 0000
ADCON3	0x41A	ADFBEN	ADCMPOP	ADCMPEN	ADCMPO	LEBADT	AN4SELB	ELVDS[1:0]		0000 0000
ADCMPLH	0x41B	ADCMPLH[7:0]								0000 0000
LEBCON	0x41C	LEBEN	LEBCH		—	EDGS	BKS2	BKS1	BKS0	0000 -000

表 1-1 ADC 相关用户寄存器地址

名称	状态		寄存器	地址	复位值
GIE	<u>全局中断</u> 1 = 使能 (PEIE, ADCIE 适用) 0 = 全局关闭 (唤醒不受影响)		INTCON[7]	Bank 首地址+ 0x0B	RW-0
PEIE	<u>外设总中断</u>	1 = 使能 (ADCIE 适用) 0 = 关闭 (无唤醒)	INTCON[6]		RW-0
ADCIE	<u>ADC 转换完成中断</u>	1 = 使能 0 = 关闭 (无唤醒)	PIE1[0]	0x91	RW-0
ADCIF ¹	ADC 转换完成中断 标志位	1 = Yes (锁存) 0 = No	PIR1[0]	0x11	R_W1C-0

表 1-2 ADC 中断使能和状态位

¹ 写 1 清 0，写 0 无效。建议只使用 STR、MOVWI 指令进行写操作，而不要用 BSR 或 IOR 指令。

名称	状态		寄存器	地址	复位值
ADRESL	<u>ADC 转换结果低有效位 (LSB)</u> ADFM=0: ADRESL[7:4] = 低 4 位 (其余为“0”) ADFM=1: ADRESL[7:0] = 低 8 位		ADRESL[7:0]	0x9B	RW-0000 0000
ADRESH	<u>ADC 转换结果高有效位 (MSB)</u> ADFM=0: ADRESH[7:0] = 高 8 位 ADFM=1: ADRESH[3:0] = 高 4 位 (其余为“0”)		ADRESH[7:0]	0x9C	RW-0000 0000
ADCEN	<u>ADC 模块时钟</u>	1 = 使能 0 = 关闭	PCKEN[0]	0x9A	RW-0
CHS	<u>ADC 模拟输入通道</u> 000 = AN0 100 = AN4 / OP0OUT 001 = AN1 101 = AN5 010 = AN2 110 = AN6 011 = AN3 111 = 1/4 V _{DD}		ADCON0[6:4]	0x9D	RW-000
AN4SELB	<u>AN4 通道选择位</u> 1 = 接内部运放输出 (OP0OUT) 0 = AN4 接 PA7		ADCON3[2]	0x41A	RW-0
ADEX	<u>ADC 触发条件 (GO/DONE)</u> 1 = 由 PA4/PB3 或 PWM 置位 GO/DONE (硬件触发) 0 = 由指令置位 GO/DONE (软件触发)		ADCON0[2]	0x9D	RW-0
GO/DONE	<u>ADC 转换启动和状态位</u> 1 = 由软件, PA4/PB3 或 PWM 启动 A/D 转换 (转换完成后自动清零) 0 = 转换完成 / 未进行转换		ADCON0[1]		RW-0
ADON	1 = ADC 使能 0 = ADC 关闭 (无电流消耗)		ADCON0[0]		RW-0
LFMOD	1: LIRC = 256 kHz 0: LIRC = 32 kHz		TCKSRC[7]	0x31F	RW-0
ADFM	<u>A/D 转换结果格式 (参阅 “ADRESH”)</u> 1 = 右对齐 0 = 左对齐		ADCON1[7]	0x9E	RW-0
ADCS	<u>ADC 转换时钟源</u> 000 = SysClk/2 100 = SysClk/4 001 = SysClk/8 101 = SysClk/16 010 = SysClk/32 110 = SysClk/64 011 = LIRC 111 = LIRC		ADCON1[6:4]		RW-000

名称	状态	寄存器	地址	复位值
ADNREF	<u>$V_{ADC-REF-}$ (负参考电压)</u> 00 = 内部 $V_{ADC-REF}$ 01 = GND 10 = 内部 $V_{ADC-REF}$ + 外部电容 Cap 11 = 外部参考电压 (I/O)	ADCON1[3:2]		RW-00
ADPREF	<u>$V_{ADC-REF+}$ (正参考电压)</u> 00 = 内部 $V_{ADC-REF}$ 01 = V_{DD} 10 = 内部 $V_{ADC-REF}$ + 外部电容 Cap 11 = 外部参考电压 (I/O)	ADCON1[1:0]		RW-00
ADINTREF	<u>内部 $V_{ADC-REF}$</u> 00 = 0.5 01 = 2.0 10 = 3.0 11 = (未连接)	ADCON2[7:6]	0x9F	RW-00
ETGTYP	<u>外部触发沿 (当 ADEX=1 时适用)</u> 00 = (PWM 或 PA4/PB3-ADC_ETR) 下降沿 01 = (PWM 或 PA4/PB3-ADC_ETR) 上升沿 10 = 一个 PWM 周期的中点 11 = 一个 PWM 周期的终点 注: 中点和终点触发仅中心对齐模式 PWM 输出使用	ADCON2[5:4]		RW-00
ADDLY.8 / LEBPR9	ADC 延迟计数器或 LEB 计数器的第 8 位 (参阅“ADDLY”)	ADCON2[3]		RW-0
ETGSEL	<u>外部触发源 (当 ADEX=1 时适用)</u> 000 = PWM0, TIM1_CH1 100 = PWM4, TIM2_CH1 001 = PWM1, TIM1_CH2 101 = PWM5, TIM2_CH2 010 = PWM2, TIM1_CH3 110 = PWM6, TIM2_CH3 011 = PWM3, TIM1_CH4 111 = ADC_ETR	ADCON2[2:0]		RW-000
ADDLY / LEBPRL	<u>ADC 延迟/LEB (非软件触发, ADEX = 1)</u> (此为低 8 位, ADDLY.8 为高有效位) 延迟时间 = (ADDLY+6)/ F_{ADC} (如果启用 PWM 输出触发 ADC, 在 PWM 运行过程中不得更改 ADDLY)	ADDLY[7:0]	0x1F	RW-0000 0000
ADFBEN	<u>ADC 阈值比较结果匹配事件触发 PWM 故障刹车</u> 1 = 使能 0 = 关闭	ADCON3[7]	0x41A	RW-0

名称	状态	寄存器	地址	复位值
ADCMPOP	<u>ADC 阈值比较的极性</u> 1 = ADC 结果的高 8 位 < ADCMPH[7:0] 0 = ADC 结果的高 8 位 ≥ ADCMPH[7:0]	ADCON3[6]		RW-0
ADCMPEM	<u>ADC 阈值比较</u> 1 = 使能 0 = 关闭	ADCON3[5]		RW-0
ADCMPO	<u>ADC 比较结果输出位</u> 输出比较结果 (每次 AD 转换结束都会更新输出)	ADCON3[4]		RO-0
LEBADT	<u>LEB 结束后, ADC 开始自动转换</u> 1 = 触发 ADC 转换 0 = 不触发 ADC 转换	ADCON3[3]		RW-0
ADCMPEM	ADC 比较阈值 (仅高 8 位, 0.4% steps)	ADCMPEM[7:0]	0x41B	RW-0000 0000
LEBEN	<u>ADC 触发和 BKIN 的 LEB 使能位</u> 1 = 使能 (当 GO/DONE=1 时进行切换将产生不可预知的结果) 0 = 关闭	LEBEN[7]	0x41C	RW-0
LEBCH	<u>LEB 信号源</u> 00 = TIM1_CH1 10 = TIM1_CH3 01 = TIM1_CH2 11 = TIM1_CH4	LEBCH[6:5]		RW-00
EDGS	<u>LEB 触发沿</u> 1 = 下降沿 0 = 上升沿	LEBEN[3]		RW-0

表 1-3 ADC 相关用户寄存器

1.2. ADC 的配置

配置和使用 ADC 时，必须考虑以下功能：

- 端口配置
- 通道选择
- 触发方式选择
- 触发源选择
- 触发类型选择
- 触发延时配置
- ADC 参考电压的选择
- ADC 转换时钟源
- 中断控制
- 转换结果的格式
- 阈值比较

注意：在进行各项配置更改的时候，需要确保 AD 转换并未正在进行或外部触发功能未开启。建议在 ADON 关闭时进行更改。

1.2.1. 端口配置

ADC 可用于转换模拟和数字信号。转换模拟信号时，应将相关的 TRIS 和 ANSELA 位置 1 将 I/O 引脚应配置为模拟功能。更多信息请参见相应的端口章节。

注意：如果定义为数字输入的引脚上存在模拟电压，会导致输入缓冲器传导过大的电流。

1.2.2. 通道选择

ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样保持电路。改变通道时，根据采样稳定的需要可在启动转换前加入一定延时，硬件已固定有 $1.5T_{AD}$ 的采样延时。更多信息请参见[章节 1.3](#)“ADC 的工作原理”。

1.2.3. 触发方式选择

ADCON0 寄存器的 ADEX 位决定是否使用外部触发信号。

若 ADEX=0 时，GO/DONE 位可由程序置位，AD 转换完成自动清零。

若 ADEX=1 时，GO/DONE 位将由外部硬件触发置位，AD 转换完成清零。

注意：若选择了前沿消隐触发 ADC，即 LEBADT 设为 1 时，需要先置位 ADEX 和 ADON。

1.2.4. 触发源选择

在设定 ADEX 后，ADCON2 寄存器的 ETGSEL 位决定使用哪个外部触发信号。其中可选 I/O 引脚触发，需要配置相关寄存器。具体请参见相应的端口章节。

1.2.5. 触发类型选择

ADCON2 寄存器的 ETGTYP 位决定外部触发信号的触发类型。

其中选择 PWM 的中点或终点类型时，触发源将会默认选择 TIM1 中心对齐的 PWM 输出信号。具体请参见相应的 TIM1 章节。

1.2.6. 触发延时配置

ADCON2 寄存器的 ADDLY.8 位和 ADDLY 寄存器组成 9 位延时计数器，共同决定外部触发信号的触发延时时间。由于需要同步异步信号，实际延迟时间为： $(ADDLY+6)/F_{ADC}$ 。

注意：若选择了前沿消隐触发功能时，则实际延迟时间为： $(ADDLY+3)/F_{TIM1} + 3/F_{ADC}$ 。

1.2.7. ADC 参考电压

ADCON1 寄存器的 ADPREF 位提供对正参考电压的控制，ADNREF 位提供对负参考电压的控制。正/负参考电压可以是内部参考电压、VDD/GND、内部参考电压加外部电容、外部参考电压。正/负参考电压可以有各种组合，但不可以同时选择内部参考电压。若发生则强制负参考电压选择 GND。

ADCON2 寄存器的 ADINTREF 位提供对内部参考电压的控制。内部参考电压可以选择 0.5V、2V、3V 或者悬空。

1.2.8. 转换时钟

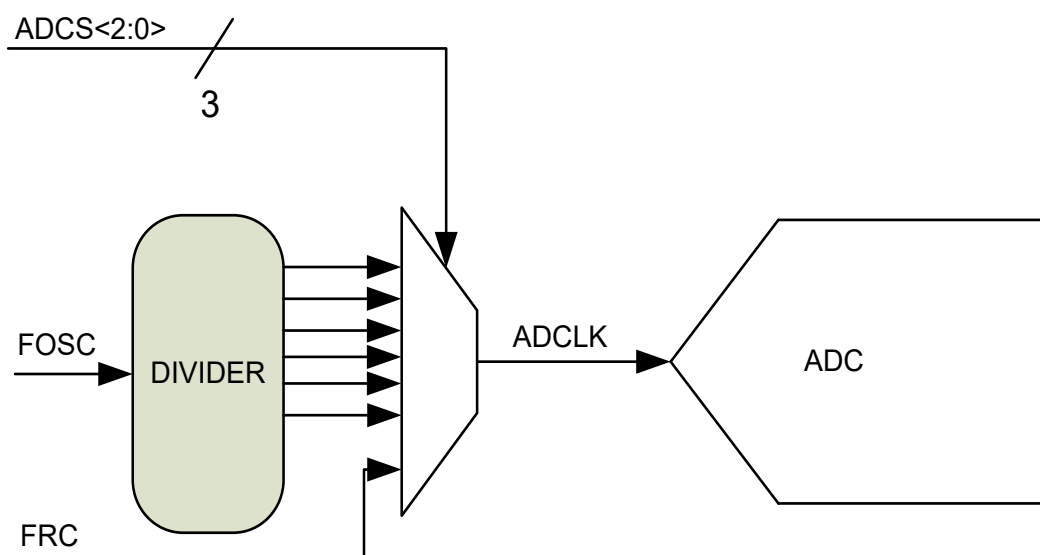


图 1-2 ADC 的时钟配置原理

转换时钟源可通过 ADCON1 寄存器的 ADCS 位用软件选择。有以下 7 种时钟选项：

- $F_{OSC}/2$
- $F_{OSC}/4$
- $F_{OSC}/8$
- $F_{OSC}/16$
- $F_{OSC}/32$
- $F_{OSC}/64$
- F_{RC} (内部慢时钟振荡器)

完成一位 (bit) 的转换时间定义为 T_{AD} 。完成 12 位转换需要 14 个 T_{AD} 周期 (12 T_{AD} 数据转换时间, 1 T_{AD} 的数据传输处理时间, 1 T_{AD} 的数据清除时间), 如图 1-3 和 1-6 所示。

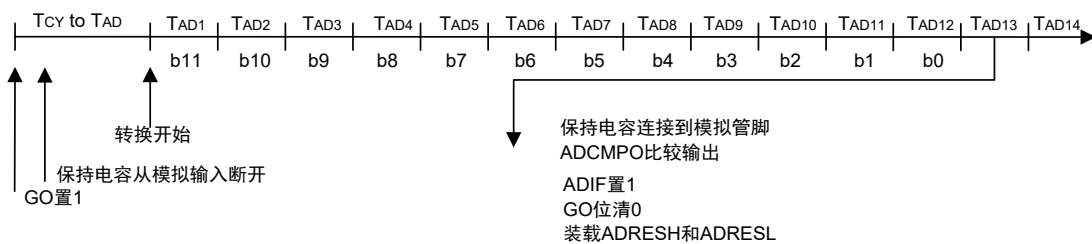


图 1-3 模数转换 T_{AD} 周期

进行正确的转换必须满足相应的 T_{AD} 规范。更多信息请参见[章节错误!未找到引用源。](#)“电气特性”中的 A/D 转换要求。表 1-4 所示为正确选择 ADC 时钟的示例。

ADC 时钟周期 (T_{AD})		系统时钟频率 (F_{OSC})			
ADC 时钟源	ADCS[2:0]	16MHz	8MHz	4MHz	1MHz
$F_{OSC}/2$	000	125ns	250ns	500ns	2.0 μ s
$F_{OSC}/4$	100	250ns	500ns	1.0 μ s	4.0 μ s
$F_{OSC}/8$	001	0.5 μ s	1.0 μ s	2.0 μ s	8.0 μ s
$F_{OSC}/16$	101	1.0 μ s	2.0 μ s	4.0 μ s	16.0 μ s
$F_{OSC}/32$	010	2.0 μ s	4.0 μ s	8.0 μ s	32.0 μ s
$F_{OSC}/64$	110	4.0 μ s	8.0 μ s	16.0 μ s	64.0 μ s
F_{RC}	x11	4.0 μ s	4.0 μ s	4.0 μ s	4.0 μ s

表 1-4 ADC 时钟周期和器件工作频率

注意:

- 除非使用的是 F_{RC} , 否则任何系统时钟频率的变化均会改变 ADC 时钟频率, 这将对 ADC 结果产生负面影响;
- F_{RC} 可以是 256kHz 或者是 32kHz, 取决于 LFMOD 为何值;
- 当 SYSON 为 0 时, 在睡眠模式下 AD 转换结束并不能唤醒 MCU, 要想实现 AD 转换结束后唤醒, 必须把 SYSON 置 1;
- 表格中的灰色部分是不支持的频率。

1.2.9. 中断

ADC 模块可使中断在模数转换完成时产生。ADC 中断标志为 PIR1 寄存器中的 ADIF 位。ADC 中断使能为 PIE1 寄存器中的 ADIE 位。ADIF 位必须用软件置 1 清零。

注意:

- 无论 ADC 中断是否被打开, ADIF 位在每次正常转换完成时均置 1。
- 软件停止 AD 转换不会置位 ADIF。
- 仅当置位 SYSON 时, ADC 才能在休眠期间工作。

器件工作或处于休眠状态时均可产生中断。如果器件处于休眠状态, 中断可唤醒器件。从休眠唤醒时, 始终执行 SLEEP 指令后的那条指令。如果用户试图唤醒器件并恢复顺序执行代码, 必须禁止全局中断。如果允许全局中断, 代码执行将转至中断服务程序。

1.2.10. 转换结果的格式

12 位 A/D 转换结果有两种格式，即左对齐和右对齐。ADCON1 寄存器的 ADFM 位控制输出格式。AD 自动校准值也受输出格式影响。

两种输出格式：

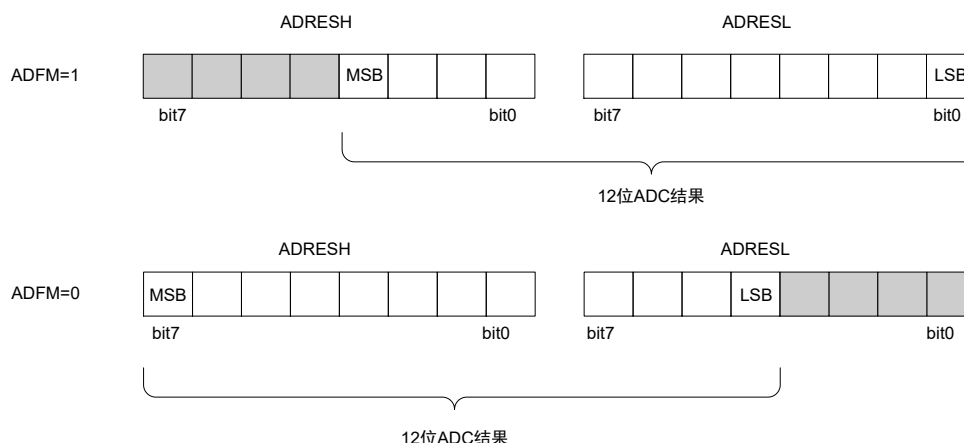


图 1-4 ADC 转换结果格式示意图

1.2.11. 阈值比较

ADCMPOH 寄存器为 ADC 结果比较阈值，ADCON3 寄存器的 ADCMPEN 位控制比较功能使能，ADCMPOP 位控制比较极性，ADCMPO 指示比较结果。

AD 可以在每次转换完成时进行比较。比较结果会一直保持，直到下次转换完成被更新。ADCMPOH 或 ADON 的清零可以关闭比较功能或 AD 模块，同时可以清零 ADCMPO。进入睡眠不会清零 ADCMPO。

在每次比较完成时可以产生故障刹车事件，由 ADCON3 寄存器的 ADFBEN 控制。

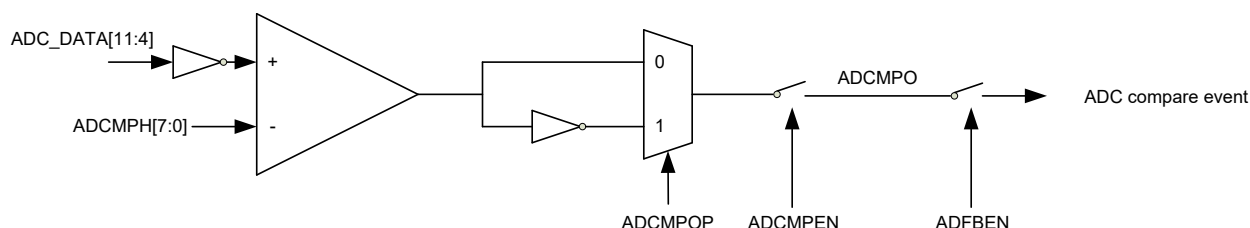


图 1-5 ADC 阈值比较功能框图

1.3. ADC 的工作原理

1.3.1. 启动转换

要使能 ADC 模块，必须将 ADCON0 寄存器的 ADON 位置 1。

若 ADEX=0 时，将 ADCON0 寄存器的 GO/DONE 位置 1 将启动 AD 转换。

若 ADEX=1 时，需要外部触发信号才能启动，并且硬件置位 GO/DONE 位，程序置位 GO/DONE 位无效。

注意：

1. 不应在打开 ADC 的那条指令中将 GO/DONE 位置 1。请参见[章节 1.3.6](#)“A/D 转换步骤”
2. 不应在启动 ADC 转换后或等待外部触发时更改 AD 配置。
3. 置位 GO/DONE 位后需要等待一个系统周期才可读回 GO/DONE 标志位。

1.3.2. 转换完成

转换完成时，ADC 模块将：

- 将 GO/DONE 位清零
- 将 ADIF 标志位置 1
- 用新的转换结果更新 ADRESH:ADRESL 寄存器
- 若使能阈值比较功能，则更新 ADCMPO 比较结果

1.3.3. 终止转换

当 ADEX=0，ADC 处于软件触发状态，如果转换必须在完成前被终止，可用软件将 GO/DONE 清零。那么 ADC 将不会更新这次转换数据。当 ADEX=1，ADC 处于硬件触发状态，如果需要终止转换，则需要将 ADON 置 0，关闭 ADC 的使能开关。

注意：器件复位将强制所有寄存器回到其复位状态。这样，ADC 模块就被关闭，并且任何待处理的转换均被终止。

1.3.4. 休眠模式下 ADC 的工作

ADC 模块可在休眠期间工作，这要求打开 SYSON 位。

ADC 需要等待 $4 \cdot T_{AD}$ 后才开始转换。这允许软件在设置 ADGO 后，执行一个 SLEEP 指令置 MCU 于 SLEEP 模式，从而降低 ADC 转换期间的系统噪声。

如果允许 ADC 中断，转换完成后器件将从休眠唤醒。如果禁止 ADC 中断，ADC 模块在转换完成后关闭，尽管 ADON 位保持置 1 状态。

如果 SYSON 未打开，执行一条 SLEEP 指令将使当前转换强制中止，ADC 模块被直接关闭，尽管 ADON 位保持置 1 状态。

如果需要在休眠模式下搭配其它模块一起使用，具体请参见相应的功能配置章节，如 TIMER、GPIO、CLK 管理模块。

1.3.5. 外部触发器

除了通过软件启动 AD 转换外，还可以通过硬件触发方式启动 AD 转换。在 ADEX 置 1 后，可选择 PWM 通道的边沿或周期、管脚边沿等触发信号自动触发启动 AD 转换（硬件自动置位 GO/DONE 位）。这允许在没有软件介入的情况下，定期进行 AD 转换。

通过 ETGSEL (ADCON2[2:0]) 和 ETGTYP (ADCON2[5:4]) 设置来选择触发源和触发类型。同时，还可以在外部触发信号与启动 AD 转换之间插入触发延时。

在 AD 模块转换过程中 (GO/DONE = 1)，任何软件或硬件触发信号都是无效的。若这时停止转换，清零 GO/DONE 位并不会停止触发延时计数。可清零 ADEX 停止触发延时计数。

只有配置 TIMER 为 PWM 输出模式并且使能 PWM 输出时，才会产生 AD 触发信号。更多信息请参见相应的 TIMER 章节。

注意：

当 LEBEN=1 时，外部触发器被禁止。这种情况下，可以选择把 LEBADT 置 1，此时 ADON 和 ADEX 必须设置为 1。在消隐周期结束后会触发一次 AD 转换（硬件自动置位 GO/DONE 位）。

1.3.6. A/D 转换步骤

以下是使用 ADC 进行模数转换的步骤示例：

1. 配置端口：
 - 禁止引脚输出驱动器（见 TRIS 寄存器）
 - 将引脚配置为模拟
2. 配置 ADC 模块：
 - 选择 ADC 转换时钟
 - 配置参考电压
 - 选择 ADC 输入通道
 - 配置触发源、类型及延时
 - 选择转换结果的格式
 - 配置 ADC 结果阈值比较
 - 打开 ADC 模块
3. 配置 ADC 中断（可选）：
 - 将 ADC 中断标志清零
 - 允许 ADC 中断
 - 允许外设中断
 - 允许全局中断
4. 等待所需稳定时间 $T_{ST}^{(1)}$ ；
5. 将 GO/DONE 位置 1 启动转换或等待硬件触发；
6. 等待一个系统周期才可回读 GO/DONE 位；
7. 通过以下情况之一等待 ADC 转换完成：
 - 查询 GO/DONE 位
 - 等待 ADC 中断（允许中断时）
8. 读取 ADC 结果；
9. 将 ADC 中断标志清零（在允许了中断的情况下这一步是必需的）。

以下是一段示例代码：

```
BANKSEL    ADCON1                ;
LDWI      B'11110101'            ;Right justify, ADC Frc clock
STR       ADCON1                  ;Vref+ VDD , Vref- GND

BANKSEL    TRISB                  ;
BSR       TRISB,1                 ;Set PB1 to input

BANKSEL    ANSELA                  ;
```

```

BSR      ANSELA,0           ;Set PB1 to analog

BANKSEL   ADCON0           ;
LDWI      B'00000001'      ;Select channel AN0,
STR      ADCON0           ;Turn ADC On
CALL      StableTime       ;ADC stable time, tST

BSR      ADCON0,ADGO       ;Start conversion (ADC stable time)
NOP      ;ADGO ReadBack WaitTime

BTSC      ADCON0,ADGO      ;Is conversion done?
LJUMP     $-1              ;No, test again

BANKSEL   ADRESH           ;
LDR      ADRESH,W         ;Read upper 4 bits
STR      RESULTHI         ;store in GPR space

BANKSEL   ADRESL           ;
LDR      ADRESL,W         ;Read lower 8 bits
STR      RESULTLO         ;Store in GPR space
    
```

注意:

1. T_{ST} 时间是 ADC 的稳定时间,当使用内部参考时,ADC 首次启动还需要考虑参考电压的稳定时间 T_{VRINT} ,等待时间应取两者的较大者,即 $\max(T_{VRINT}, T_{ST})$;

1.4. A/D 采集时间要求

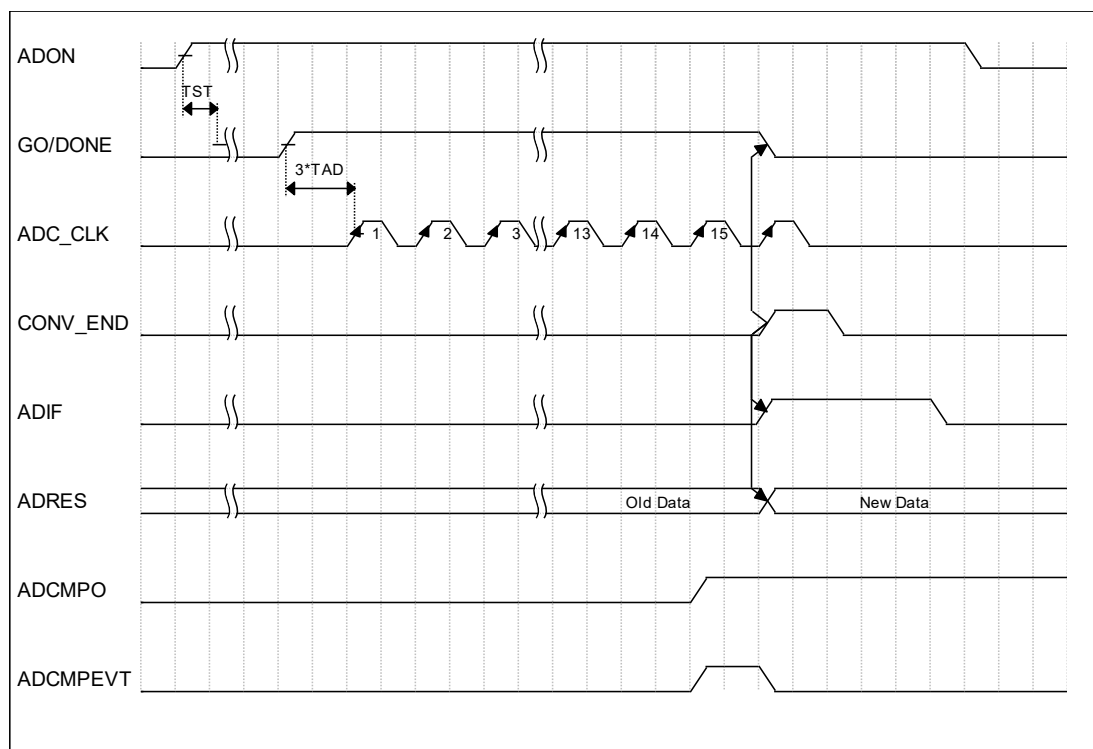
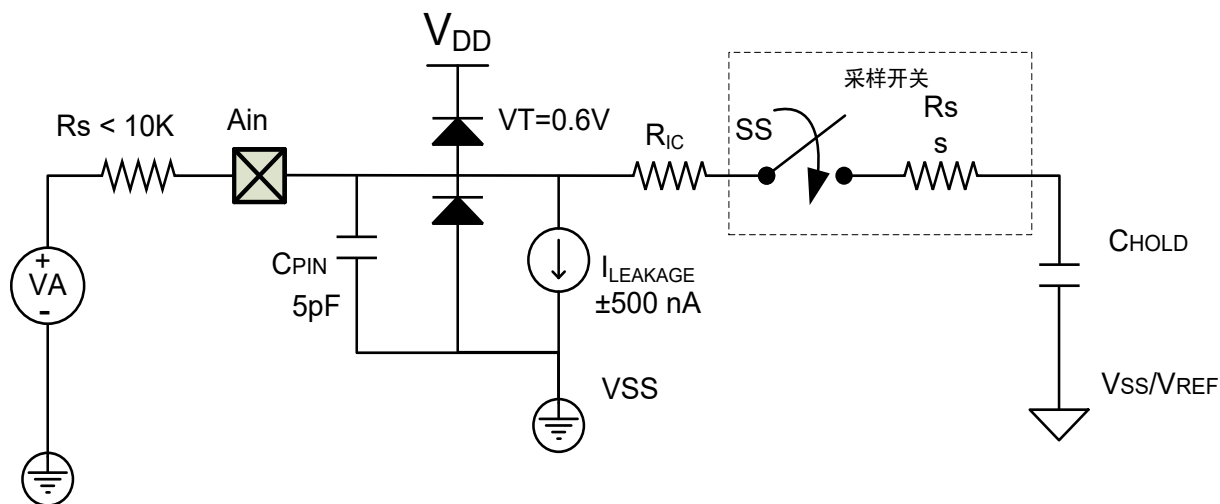


图 1-6 ADC 软件触发转换时序图

为了使 ADC 达到规定的精度，必须使充电保持电容 (CHOLD) 充满至输入通道的电平。模拟输入模型请参见图 1-。源阻抗 (R_S) 和内部采样开关 (RSS) 阻抗直接影响电容 CHOLD 的充电时间。采样开关 (RSS) 阻抗随器件电压 (V_{DD}) 的变化而变化，参见图 1-。建议模拟信号源的最大阻抗为 $10k\Omega$ 。采集时间随着源阻抗的降低而缩短。在选择 (或改变) 模拟输入通道后，必须在开始转换前完成采集。



图注：

- CPIN = 输入电容
- VT = 门限电压
- ILEAKAGE = 结点漏电流
- RIC = 互联电阻
- SS = 采样开关
- CHOLD = 采样保持电容

图 1-7 模拟输入模型

2. 应用范例

```
//*****
/* 文件名: TEST_64F0Ax_ADC.c
* 功能:    FT64F0Ax_ADC 功能演示
* IC:      FT64F0A5  TSSOP20
* 内部:    16M/2T
* 说明:    程序采样 AN0 口变阻器的 AD 值并计算其电压
*
*          FT64F0A5  TSSOP20
*          -----
* NC-----|1(PA5)    (PA4)20|-----NC
* NC-----|2(PA6)    (PA3)19|-----NC
* NC-----|3(PA7)    (PA2)18|-----NC
* NC-----|4(PC0)    (PA1)17|-----NC
* NC-----|5(PC1)    (PA0)16|-----NC
* NC-----|6(PB7)    (PB0)15|-----NC
* GND-----|7(GND)   (PB1)14|-----AN0
* NC-----|8(PB6)    (PB2)13|-----NC
* VDD-----|9(VDD)   (PB3)12|-----NC
* NC-----|10(PB5)   (PB4)11|-----NC
*
*          -----
*/
//*****
#include "SYSCFG.h";
#include "FT64F0AX.h";
//***** 宏定义 *****
#define uchar  unsigned char
#define uint   unsigned int
#define ulong  unsigned long

volatile uint  adcData;
volatile uint  theVoltage;
/*-----
* 函数名:  POWER_INITIAL
* 功能:    上电系统初始化
* 输入:    无
* 输出:    无
*-----*/
void POWER_INITIAL(void)
{
    OSCCON=0B01110001;          //系统时钟选择为内部振荡器 16MHz,分频比为 1:1

    PCKEN|=0B00000001;          //AD 模块时钟使能
```



```

INTCON=0;                //禁止所有中断

PORTA=0B00000000;
PORTB=0B00000000;
PORTC=0B00000000;

WPUA=0B00000000;        //弱上拉的开关，0-关，1-开
WPUB=0B00000000;
WPUC=0B00000000;

WPDA=0B00000000;        //弱下拉的开关，0-关，1-开
WPDB=0B00000000;
WPDC=0B00000000;

TRISA=0B00000000;        //输入输出设置，0-输出，1-输入
TRISB=0B00000000;
TRISC=0B00000000;

PSRC0=0B11111111;        //源电流设置最大
PSRC1=0B11111111;
PSRC2=0B00001111;

PSINK0=0B11111111;        //灌电流设置最大
PSINK1=0B11111111;
PSINK2=0B00000011;

ANSELA=0B00000000;        //设置对应的 IO 为数字 IO
}
/*-----
* 函数名： SPI_INITIAL
* 功能：   初始化 IIC
* 输入：   无
* 输出：   无
-----*/
void ADC_INITIAL(void)
{
    ANSELA=0B00000001;        //控制 IO 的数模输入，1：对应的 IO 为模拟管脚，0：对应的
IO 为数字 IO，设置 AN0 为模拟管脚
    ADCON0=0B00000000;
    //Bit[6:4]:000-选择模拟通道 AN0
    //Bit2:当软件设定 GO/DONE 位时启动 AD 转换（即不用外部触发源）

    ADCON1=0B11100100;

```

```
//Bit7: 1-ADC 转换结果右对齐, 即装入转换结果时, ADRESH 的高 4 位被设置为 0
//Bit[6:4]:110-ADC 转换时钟设置为 Fosc/64
//Bit[3:2]:01-负参考电压-GND
//Bit[1:0]:00-正参考电压-内部参考电压
```

```
ADCON2=0B01000000;
//Bit[7:6]:01-ADC 内部参考电压 2V
```

```
ADCON3=0B00000000;
ADDLY=0B00000000;           //外部触发延时
ADCMPLH=0B00000000;         //ADC 比较阈值, 用于 ADC 结果高 8 位比较
ADON=1;                      //使能 ADC
```

```
}
```

```
/*-----
```

```
* 函数名: DelayUs
* 功能:   短延时函数
* 输入:   Time 延时时间长度 延时时长 Time  $\mu$ s
* 输出:   无
```

```
-----*/
```

```
void DelayUs(uchar Time)
```

```
{
```

```
    uchar a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
```

```
}
```

```
/*-----
```

```
* 函数名: DelayMs
* 功能:   短延时函数
* 输入:   Time 延时时间长度 延时时长 Time ms
* 输出:   无
```

```
-----*/
```

```
void DelayMs(uchar Time)
```

```
{
```

```
    uchar a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197);
        }
    }
}
```

```
}
```

```

/*-----
* 函数名: GET_ADC_DATA
* 功能:   读取通道 ADC 值
* 输入:   adcChannel 通道序号
* 输出:   INT 类型 AD 值（单次采样无滤波）
*-----*/
uint GET_ADC_DATA(uchar adcChannel)
{
    ADCON0&=0B00001111;
    ADCON0|=adcChannel<<4;
    DelayUs(40);
    GO=1;
    NOP();
    NOP();
    while(GO);

    return (uint)(ADRESH<<8|ADRESL);
}

/*-----
* 函数名: main
* 功能:   主函数
* 输入:   无
* 输出:   无
*-----*/
void main(void)
{
    POWER_INITIAL();           //系统初始化
    ADC_INITIAL();             //ADC 初始化

    while(1)
    {
        adcData=GET_ADC_DATA(0);
        theVoltage=(ulong)adcData*2*1000/4096;
        NOP();
        NOP();
    }
}

```

1 联系信息

Fremont Micro Devices Corporation

#5-8, 10/F, Changhong Building
Ke-Ji Nan 12 Road, Nanshan District,
Shenzhen, Guangdong, PRC 518057

Tel: (+86 755) 8611 7811

Fax: (+86 755) 8611 7810

Fremont Micro Devices (HK) Limited

#16, 16/F, Block B, Veristrong Industrial Centre,
34-36 Au Pui Wan Street, Fotan, Shatin, Hong Kong SAR

Tel: (+852) 2781 1186

Fax: (+852) 2781 1144

<http://www.fremontmicro.com>

* Information furnished is believed to be accurate and reliable. However, Fremont Micro Devices Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents of other rights of third parties, which may result from its use. No license is granted by implication or otherwise under any patent rights of Fremont Micro Devices Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. Fremont Micro Devices Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of Fremont Micro Devices Corporation. The FMD logo is a registered trademark of Fremont Micro Devices Corporation. All other names are the property of their respective owners.