T1601/T117-OW/MTS4-OW/T1820

升级替代 M1601/M601/M1820 应用指南

敏源第 6 代高精度温度芯片 T1601（SOT23 封装）、T117-OW（DFN6L 封装）、MTS4-OW（DFN4L 封装）、T1820（TO92S 封装），最高测温精度±0.1℃，同时也有±0.5℃精度的产品（选型可参见后附表）。芯片内置 16 bit ADC，分辨率 0.004°C，温度转换时间 15.3/8.5/5.2/2.2ms 可配置，宽工作电压范围 1.8~5.5V，低功耗（测温平均电流 2μA@AVG=8，1Hz），可替代或并行敏源第 4 代数字温度芯片：M1601（SOT23 封装）、M601（DFN8 封装）、M1820（TO92S 封装），程序上只需做如下简要修定（下面案例以 T1601 升级替代 M1601 为例，其他型号升级替代类同）：

1、温度转换时间

M1601 温度转换时间 10.5/5.5/4ms 可配置，T1601 系列温度转换时间 15.3/8.5/5.2/2.2ms（默认出厂配置 5.2ms），配置方式见下：

```
#define SingleIC      /*总线上只有一颗芯片*/
//#define AllIC       /*总线上有一颗以上数量芯片*/
//#define SingleConvert /*总线上所有芯片一颗一颗分别测温转换*/

ConvertTemp();
Delay_ms(7);//不同重复性下转换时间不同，此处延时必须大于对应的转换时间。
/*读取方式在此处定义*/
/**
  * @brief   多点设置周期测量频率、平均次数和低功耗模式
  * @param   mps：要设置的周期测量频率（每秒测量次数），可能为下列其一
    *   @arg FRE_8TIMES     ：每执行 ConvertTemp 一次，启动每秒 8 次重复测量
    *   @arg FRE_4TIMES     ：每执行 ConvertTemp 一次，启动每秒 4 次重复测量
    *   @arg FRE_2TIMES     ：每执行 ConvertTemp 一次，启动每秒 2 次重复测量
    *   @arg FRE_1TIMES     ：每执行 ConvertTemp 一次，启动每秒 1 次温度测量
```

```
     *   @arg FRE_2S        ：每执行 ConvertTemp 一次，启动每 2 秒 1 次重复测量
     *   @arg FRE_4S        ：每执行 ConvertTemp 一次，启动每 4 秒 1 次重复测量
     *   @arg FRE_8S        ：每执行 ConvertTemp 一次，启动每 8 秒 1 次重复测量
     *   @arg FRE_16S       ：每执行 ConvertTemp 一次，启动每 16 秒 1 次重复测量
    * @param   avg：要设置的平均次数，可能为下列其一
     *   @arg AVG_1         ：平均次数 1 次
     *   @arg AVG_8         ：平均次数 8 次
     *   @arg AVG_16        ：平均次数 16 次
     *   @arg AVG_32        ：平均次数 32 次
    * @param   sleep：要设置的低功耗模式，可能为下列其一
     *   @arg OFF_PD        ：位清 0，不进入低功耗模式
     *   @arg ON_PD         ：进入低功耗模式
    * @param   x：要匹配的芯片在 Search 到的总线上全部芯片中的序号（单点使用时,x 可为 0）
    * @retval 无
    */
    int OW_SetConfig(uint8_t mps, uint8_t avg, uint8_t sleep, uint8_t x)
    {
        uint8_t scrb[sizeof(SCRATCHPAD_READ)], ID1[9];
        SCRATCHPAD_READ* scr = (SCRATCHPAD_READ*)scrb;
        /*读 9 个字节。第 7 字节是系统配置寄存器，第 8 字节是系统状态寄存器。最后字节是前 8 个的
校验和--CRC。*/
        if (OW_ReadScratchpad(scrb, x) == 0)//FALSE
        {
            return 0;//FALSE   /*CRC 验证未通过*/
        }
    #ifdef SingleIC     /*总线上只有一颗芯片*/
        if (scrb[sizeof(scrb) - 1] != MY_CRC8(scrb, sizeof(scrb) - 1))
        {
            return 0;//FALSE   /*CRC 验证未通过*/
        }
    #endif
    #ifndef SingleIC     /*总线上有一颗以上数量芯片*/
        for (uint8_t j = 0; j < 7; j++)
        {
            ID1[j] = ID_Buff[x][j];
    }
        for (uint8_t j = 0; j < sizeof(scrb); j++)
```

```
    {
        ID1[j + 7] = scrb[j];
    }
    if (scrb[sizeof(scrb) - 1] != MY_CRC8(ID1, sizeof(scrb) + 6))
    {
        return 0;
    }
#endif
    /*计算接收的前 8 个字节的校验和，并与接收的第 9 个 CRC 字节比较。*/
    scr->Temp_Cfg &= 0x00;
    scr->Temp_Cfg |= mps;
    scr->Temp_Cfg |= avg;
    scr->Temp_Cfg |= sleep;
    OW_WriteScratchpad(scrb, x);
    return 1;//TRUE
}
```

2、温度寄存器

T1601、M1601 系列计算公式不同，修改如下：

```
float T1601_OutputtoTemp(int16_t out)
{
    return ((float)out/256.0 + 25.0);
}
```

3、温度读取指令

T1601 系列的温度读取指令是 0xBC，读取 Temperature 寄存器的 3 个字节，，读取函数按如下进行修改：

```
/*读取方式在此处定义*/
#define SingleIC      /*总线上只有一颗芯片*/
//#define AllIC       /*总线上有一颗以上数量芯片*/
//#define SingleConvert /*总线上所有芯片一颗一颗分别测温转换*/
#define    READ_TEMP          0xbc
```

```
typedef struct
{
    uint8_t T_lsb;                  /*The LSB of 温度结果, RO*/
    uint8_t T_msb;                  /*The MSB of 温度结果, RO*/
    uint8_t Crc_temp;
}TEMP_READ;

/**
  * @brief  等待转换结束后读测量结果。和@ConvertTemp 联合使用
  * @param  iTemp：返回的 16 位温度测量结果
  * @param  x：要匹配的芯片在 Search 到的总线上全部芯片中的序号（单点使用时,x 可为 0）
  * @retval 读状态
*/
int OW_ReadTempWaiting(uint16_t* iTemp, uint8_t x)
{
    uint8_t scrb[sizeof(TEMP_READ)], ID1[9];
    TEMP_READ* scr = (TEMP_READ*)scrb;
    if (OW_ReadTemp(scrb, x) == 0)//FALSE
    {
        return 0; //FALSE /*读寄存器失败*/
    }
#ifdef SingleIC     /*总线上只有一颗芯片*/
    if (scrb[sizeof(scrb) - 1] != MY_CRC8(scrb, sizeof(scrb) - 1))
    {
        return 0;//FALSE  /*CRC 验证未通过*/
    }
#endif
#ifndef SingleIC     /*总线上有一颗以上数量芯片*/
    for (uint8_t j = 0; j < 7; j++)
    {
        ID1[j] = ID_Buff[x][j];
}
    for (uint8_t j = 0; j < sizeof(scrb); j++)
    {
        ID1[j + 7] = scrb[j];
    }
    if (scrb[sizeof(scrb) - 1] != MY_CRC8(ID1, sizeof(scrb) + 6))
```

```
    {
        return 0;
    }
#endif
    /*将温度测量结果的两个字节合成为 16 位字。*/
    * iTemp = (uint16_t)scr->T_msb << 8 | scr->T_lsb;
    return 1;//TRUE
}
```

4、睡眠模式

T1601 系列默认开启睡眠，进入睡眠时，需要把 Temperature 寄存器的 3 个字节全部读完，无需发送发送指令，读取代码修改见上述第 3 条**温度读取指令**中的读温函数，T1601 睡眠模式可配置，配置代码修改见上述第 1 条**温度转换时间**中的转换时间配置函数。

5、暂存器

使用 T1601 系列时，暂存器配置只需按芯片手册定义进行修订即可。

表 10.1 寄存器列表与 E^2PROM 映射关系

寄存器名称	位宽	寄存器逻辑地址	单总线读取	单总线写入	E^2PROM 逻辑地址	单总线写入 E^2PROM	单总线装载 E^2PROM	上电复位默认值
Temp_lsb	8	00h	Read temperature (0xBC)	NA	NA	NA	NA	00h
Temp_msb	8	01h		NA	NA	NA	NA	00h
Crc_temp	8	02h		NA	NA	NA	NA	NA
Status	8	03h	Read scratchpad (0xBE)	NA	NA	NA	NA	00h
Temp_Cmd	8	04h		Write config (0x4E)	NA	NA	NA	40h
Temp_Cfg	8	05h			00h	Copy page (0x48)	Recall EE (0xB8) / Recall page (0xB6)	69h
Alert_Mode	8	06h			01h			00h
Th_lsb	8	07h			02h			FFh
Th_msb	8	08h			03h			7Fh
Tl_lsb	8	09h			04h			00h
Tl_msb	8	0Ah			05h			80h
Crc_scratch	8	0Bh		NA	NA			NA
User_define_0	8	0Ch	Read scratchpad	Write scratchpad	06h			00h
User_define_1	8	0Dh			07h			00h

User_define_2	8	0Eh	extension (0xDD)	extension (0x77)	08h			00h
User_define_3	8	0Fh			09h			00h
User_define_4	8	10h			0Ah			00h
User_define_5	8	11h			0Bh			00h
User_define_6	8	12h			0Ch			00h
User_define_7	8	13h			0Dh			00h
User_define_8	8	14h			0Eh			00h
User_define_9	8	15h			0Fh			00h
Crc_scratch_ext	8	16h		NA	NA	NA	NA	NA
E^2PROM_Cmd	8	17h	NA	NA	NA	NA	NA	00h
Romcode1	8	18h	Read romcode (0x33)	NA	NA	NA	NA	01h
Romcode2	8	19h		NA	NA	NA	NA	16h
Romcode3	8	1Ah		NA	10h	NA	Recall EE (0xB8)	(1)
Romcode4	8	1Bh			11h			(1)
Romcode5	8	1Ch			12h			(1)
Romcode6	8	1Dh			13h			(1)
Romcode7	8	1Eh			14h			(1)
crc_romcode	8	1Fh		NA	NA			NA
Misc_Cfg	8	63h	(2)	(2)	(2)	(2)	(2)	00h
Mode_Cfg	8	68h	(2)	(2)	(2)	(2)	(2)	00h

备注：

（1）该字节为出厂值；

（2）对该字节的修改与其他用于内部测试的字节相关联，修改指令可与厂家联系获取。

附：产品选型表

型号	封装	尺寸
T1601X	SOT23-3	2.9*2.8*1.1mm
T117X-OW	DFN6L	2*2*0.75mm
MTS4X-OW	DFN4L	1.6*1.2*0.55mm
T1820X	TO92S	/