AN2101
应用笔记

UM16436 配置指南

版本：V1.0

UNICMICRO
广芯微电子

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

本文档的所有部分，其著作产权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

目录

1 摘要

本篇应用笔记主要介绍UM16436配置指南。

本篇应用笔记主要包括：

- 概述
- 驱动相关配置
- 字库
- 点亮屏幕

2 概述

本文主要介绍 UM16436 的配置方法以及步骤，用户可根据实际需求进行相应的配置，以能够进行快速开发。软件开发包中提供了 SPI 和 I2C 接口的示例代码，具体见“UM16436 SDK V1.0\Driver&Example\Example”路径下的工程示例。

3 驱动相关配置

为了能够使UM16436成功驱动芯片，需要进行以下相关的配置。

3.1 屏幕参数配置

以4COM屏幕为例，LCD电压为3.0V，驱动模式为1/4占空比，1/3偏压：

- LCD分压比调节：配置LCDTEST寄存器的VLCD_SET位，此处需要根据实际的硬件进行相应设置。若硬件设计上没有单独为VLCD进行供电，且VDD供电为3.3V，则该位可以设置为4b’0010，即使用内部分压给到VLCD供电，VLCD=0.9*VDD=0.9*3.3≈3.0V；若硬件设计上单独为VLCD提供3.0V供电，则该位可以设置为4b’1111，即VLCD电压来自外部引脚。
- LCD显示模式：配置LCDSET寄存器的DISPLAY_MODE位，设置为2b’00，即36SEG*4COM显示模式，占空比为1/COM。
- LCD偏压：配置LCDSET寄存器的BIAS_SET位，设置为2b’01，即1/3 偏压。

此时将DISPCTRL寄存器的LCD_EN位置1，即可正确驱动屏幕显示，但为了获得更好的驱动显示效果，可再进行以下设置：

- 驱动波形：配置LCDSET寄存器的INV_MODE位，在A类和B类驱动显示效果一样的情况下，可设置为B类波形，能够降低整体功耗水平。
- 显示帧频：配置DFSET寄存器的DFSET位，根据需要的驱动波形选择显示帧频设置，在不影响显示效果的情况下，可适当降低显示帧频以降低整体功耗水平。
- 偏置电流：配置LCDSET寄存器的IBIAS_SET位，为了保证COM&SEG端输出波形在高帧频下不失真，可根据实际需求将偏置电流设大。
- 快速充电期间的驱动强度：配置FCSET寄存器的FC_DS位，若显示效果不佳，可适当加大快速充电期间的驱动强度，共有16档可设。
- 偏快速充电时间：配置FCSET寄存器的FC_PW位，若显示效果不佳，可适当加长快速充电时间，共有16档可设。

以SPI接口为例，完整的配置见以下代码段：

```
    umlcd_write_reg(UMLCD_TEST_REG, UMLCD_TESTEN_OFF | UMLCD_LCCTRL_OFF | UMLCD_VLCD_SET_VDD | UMLCD_RCL_CAT_EN_OFF);//设置 VLCD 为 VDD

    umlcd_write_reg(UMLCD_DFSET_REG, UMLCD_DFSET_EN_ON | UMLCD_DFSET_TYPE_B_64Hz);//帧频 64Hz
```

```
    umlcd_write_reg(UMLCD_LCDSET_REG, UMLCD_DISPLAY_MODE_36x4 |
UMLCD_INV_MODE_TYPE_B | UMLCD_BIAS_SET_ONE_THIRD |
UMLCD_IBIAS_SET_100nA);//4COM、36SEG、B 类、1/3 Bias、偏置电流 100nA

    umlcd_write_reg(UMLCD_FCSET_REG, UMLCD_FC_PW_CLOCK_2 |
UMLCD_FC_DS_160UA);//快充时间 2Clock、快充期间驱动强度 160uA

    umlcd_write_reg(UMLCD_SYS_CTRL_REG, UMLCD_CLK_SEL_INTERNAL |
UMLCD_SPI_LOCK_OFF | UMLCD_I2C_LOCK_OFF | UMLCD_CLK32K_EN_ON |
UMLCD_WDT_EN_OFF | UMLCD_CLK_OUT_OFF);//使用内部 32KHz 时钟源，开启时钟

    umlcd_write_reg(UMLCD_DISPCTRL_REG, UMLCD_FELUSE_EN_OFF |
UMLCD_PWEN_ON | UMLCD_LCD_EN_ON);//开启 LCD
```

3.2 时钟自动校准配置

UM16436内部32kHz时钟频率可以使用通信接口的时钟信号来自动校准。以SPI通信接口为例，按以下步骤操作：

- 主控MCU的SPI SCK速率设置为1MHz。
- 配置LCDTEST寄存器的RCL_CAT_EN位，设置为1，即开启时钟频率校准。
- 利用通信过程中产生的1MHz时钟频率进行校准，校准时间必须等待250个SCK时钟周期，即延时250us。
- 250个SCK时钟周期后，配置LCDTEST寄存器的RCL_CAT_EN位，设置为0，即关闭时钟频率校准，完成32kHz时钟频率校准。

```
    umlcd_write_reg(UMLCD_TEST_REG, UMLCD_TESTEN_OFF | UMLCD_LCCTRL_OFF |
UMLCD_VLCD_SET_VDD | UMLCD_RCL_CAT_EN_ON);

    delay_us(250);//必要的时间，不能去掉

    umlcd_write_reg(UMLCD_TEST_REG, UMLCD_TESTEN_OFF | UMLCD_LCCTRL_OFF |
UMLCD_VLCD_SET_VDD | UMLCD_RCL_CAT_EN_OFF);
```

实际使用中可直接调用该函数进行时钟自动校准。

```
    /*************************************************************************
     * function     : umlcd_clock_auto_calibrate
     * Description: umlcd_clock_auto_calibrate 时钟自动校准
```

```
    * input : none
    * return: none
    ***************************************************************************/
   void umlcd_clock_auto_calibrate(void);
```

4 字库

UM16436 字库在 *um16436_fonts.h* 文件中定义，其中包括定义段码的数组和定义数字的数组。在实际开发中，用户需要根据不同款式的屏幕，以及实际中该款屏幕的 COM&SEG 与 UM16436 的接线顺序来重新定义段码数组。

下图为 4COM、16SEG 段码屏幕的段码定义。

PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
COM1	COM1	---	---	---	S7	1D	1C	2D	2C	3D	3C	4D	4C	5D	5C	6D	6C	7D	7C	P1
COM2	---	COM2	---	---	S8	1E	1G	2E	2G	3E	3G	4E	4G	5E	5G	6E	6G	7E	7G	P2
COM3	---	---	COM3	---	S9	1F	1B	2F	2B	3F	3B	4F	4B	5F	5B	6F	6B	7F	7B	S6
COM4	---	---	---	COM4	S10	1A	S11	2A	S1	3A	col1	4A	S2	5A	col2	6A	S3	7A	S4	S5

图 4-1：段码定义

段码数组“*const char fonts_table[304][4]*”中定义了 304 个 SEG 数据，从 *fonts_table[x][4]*到 *fonts_table[304][4]* ，每 8 个 SEG 对应一个 DATAx 寄存器，具体的 DATAx 寄存器与 COMx&SEGx 的关系可以参考 UM16436 数据手册。其中需要注意，字库中的字符定义最大 4 个字节，如果是 2 字节定义，需要在第三个字节上补上空字符，大于 2 字节的定义，则不需要补空字符。

以 4COM 段码屏为例，若实际中屏幕 COM0~3 是与 UM16436 的 COM0~3 连接，即正序连接，则段码数组中定义的 COM0~4 的顺序也需要正序，见以下代码段。

```
/*COM0*/
{"P1\0"},{"7C\0"},{"7D\0"},{"6C\0"},{"6D\0"},{"5C\0"},{"5D\0"},{"4C\0"},      /* DATA0 */
{"4D\0"},{"3C\0"},{"3D\0"},{"2C\0"},{"2D\0"},{"1C\0"},{"1D\0"},{"S7\0"},      /* DATA1 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA2 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA3 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA4 */
/*COM1*/
{"P2\0"},{"7G\0"},{"7E\0"},{"6G\0"},{"6E\0"},{"5G\0"},{"5E\0"},{"4G\0"},      /* DATA5 */
{"4E\0"},{"3G\0"},{"3E\0"},{"2G\0"},{"2E\0"},{"1G\0"},{"1E\0"},{"S8\0"},      /* DATA6 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA7 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA8 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA9 */
/*COM2*/
{"S6\0"},{"7B\0"},{"7F\0"},{"6B\0"},{"6F\0"},{"5B\0"},{"5F\0"},{"4B\0"},      /* DATA10 */
{"4F\0"},{"3B\0"},{"3F\0"},{"2B\0"},{"2F\0"},{"1B\0"},{"1F\0"},{"S9\0"},      /* DATA11 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA12 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA13 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA14 */
/*COM3*/
{"S5\0"},{"S4\0"},{"7A\0"},{"S3\0"},{"6A\0"},{"CO2"},{"5A\0"},{"S2\0"},      /* DATA15 */
{"4A\0"},{"CO1"},{"3A\0"},{"S1\0"},{"2A\0"},{"S11"},{"1A\0"},{"S10"},        /* DATA16 */
```

```
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA17 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA18 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA19 */
```

若实际中屏幕 COM0~3 是与 UM16436 的 COM3~0 连接，即反序连接，则段码数组中定义的 COM0~4 的顺序也需要反序，见以下代码段。

```
/*COM3*/
{"S5\0"},{"S4\0"},{"7A\0"},{"S3\0"},{"6A\0"},{"CO2"},{"5A\0"},{"S2\0"},        /* DATA15 */
{"4A\0"},{"CO1"},{"3A\0"},{"S1\0"},{"2A\0"},{"S11"},{"1A\0"},{"S10"},          /* DATA16 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA17 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA18 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA19 */
/*COM2*/
{"S6\0"},{"7B\0"},{"7F\0"},{"6B\0"},{"6F\0"},{"5B\0"},{"5F\0"},{"4B\0"},        /* DATA10 */
{"4F\0"},{"3B\0"},{"3F\0"},{"2B\0"},{"2F\0"},{"1B\0"},{"1F\0"},{"S9\0"},        /* DATA11 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA12 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA13 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA14 */
/*COM1*/
{"P2\0"},{"7G\0"},{"7E\0"},{"6G\0"},{"6E\0"},{"5G\0"},{"5E\0"},{"4G\0"},        /* DATA5 */
{"4E\0"},{"3G\0"},{"3E\0"},{"2G\0"},{"2E\0"},{"1G\0"},{"1E\0"},{"S8\0"},        /* DATA6 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA7 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA8 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA9 */
/*COM0*/
{"P1\0"},{"7C\0"},{"7D\0"},{"6C\0"},{"6D\0"},{"5C\0"},{"5D\0"},{"4C\0"},        /* DATA0 */
{"4D\0"},{"3C\0"},{"3D\0"},{"2C\0"},{"2D\0"},{"1C\0"},{"1D\0"},{"S7\0"},        /* DATA1 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA2 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA3 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA4 */
```

以 16SEG 段码屏为例，若实际中屏幕 SEG0~15 是与 UM16436 的 SEG0~15 连接，即正序连接，则段码数组中定义的 SEG0~15 的顺序也需要正序，见以下代码段。

```
/*COM0*/
{"P1\0"},{"7C\0"},{"7D\0"},{"6C\0"},{"6D\0"},{"5C\0"},{"5D\0"},{"4C\0"},        /* DATA0 */
{"4D\0"},{"3C\0"},{"3D\0"},{"2C\0"},{"2D\0"},{"1C\0"},{"1D\0"},{"S7\0"},        /* DATA1 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA2 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL}, /* DATA3 */
```

```
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA4 */
```

若实际中屏幕 SEG0~15 是与 UM16436 的 SEG15~0 连接，即反序连接，则段码数组中定义的 SEG0~15 的顺序也需要反序，见以下代码段。

```
/*COM0*/
{"S7\0"},{"1D\0"},{"1C\0"},{"2D\0"},{"2C\0"},{"3D\0"},{"3C\0"},{"4D\0"},        /* DATA1 */
{"4C\0"},{"5D\0"},{"5C\0"},{"6D\0"},{"6C\0"},{"7D\0"},{"7C\0"},{"P1\0"},        /* DATA0 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA2 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA3 */
{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},{NULL},  /* DATA4 */
```

5 点亮屏幕

在配置完 UM16436 后，用户可利用以下两个接口来点亮任意段码。

5.1 任意段码显示接口

```
/*****************************************************************************
 * function     : umlcd_display
 * Description: umlcd_display 用于控制单个段码的显示
 * input :
 *                 char *display_table ： 显示段码的字符数组
 *                 uint8_t length                  ： 显示段码的个数
 * return: none
 *****************************************************************************/
void umlcd_display(char *display_table, uint8_t length);
```

该接口用于显示单个段码，根据输入字符数组的长度，可连续显示多个单段码。

在使用前我们需要定义一个显示特定内容的段码字符数组，或只定义单个段码字符。将定义的数组的需要显示段码的对应起始地址输入到第一个入口参数，第二个入口参数输入显示段码的个数。以下面代码段为例，以下代码段的功能为*table1[0]~ table1[7]*对应段码显示500ms，随后显示*table1[8]~ table1[9]*对应段码500ms。

```
char table1[][3] = {"S5","S6","S7","S8","S9","S10","S1","S2","S3","S4"};
umlcd_display(table1[0], 8);delay_ms(500);
umlcd_display(table1[8], 2);delay_ms(500);
```

5.2 数字显示接口

```
/*****************************************************************************
 * function     : umlcd_num_display
 * Description: umlcd_num_display 段码数字显示 0-9
 * input :
 *                         uint8_t num_location ： 第几个段码
 *                         uint8_t num                 ： 需要显示的数字
 * return: none
 *****************************************************************************/
void umlcd_num_display(uint8_t num_location, uint8_t num);
```

该接口用于显示指定位置的单个段码数字，数字的位置与段码屏数字定义位置一致。

假设有下图数字段码定义，若需要在第3个数字位置上显示数字“7”，则函数接口中的*num_location入口参数*输入“3”，*num入口参数*输入“7”，即*umlcd_num_display(3,7)* 。

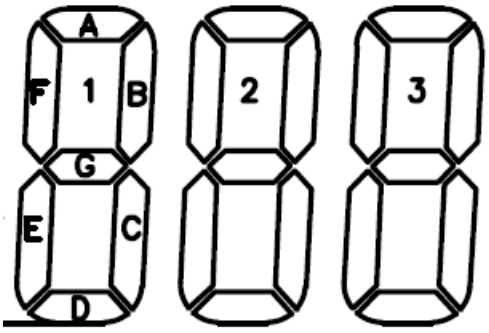

图 5-1：数字段码定义

6 版本修订

版本	日期	描述
V1.0	2023.06.30	初始版