AN12583

通过状态可配置计时器状态机在 LPC5460x 上启用相机

版本 0 — 2019 年 9 月 16 日 应用笔记

目录

1 简介

LPC5460x 原本不支持硬件的摄像头接口。 然而，为了在 LPC5460x MCU 上启用相机设备，该功能是用状态可配置定时器（SCT）和直接存储器访问（DMA）与适当的同步信号在适当的配置下实现的。SCT 用于捕获同步信号，DMA 用于移动数据，而不需要 CPU 的介入。本笔记描述了在 LPC5460x 上使用 SCT 和 DMA 的软件摄像机接口解决方案的设计。

2 SCT 硬件

SCT 是一个基于硬件计数器/定时器的硬件模块。它有各种条件来创造事件。事件取决于各种条件，包括计数器值、输入、信号和其他条件。然后，事件可以与几个操作起作用如控制计数器、设置输出引脚电压电平、更改状态机等。在 LPC5460x MCU 上的 SCT 模块支持：

- 8 路输入
- 10 路输出
- 10 个匹配/捕获寄存器
- 10 个事件
- 10 个状态机

2.1 SCT 计数器和引脚

根据 SCT_CONFIG[UNIFY]位，SCT 被配置为两个 16 位计数器或一个 32 位计数器。在大多数情况下，默认情况下使用 32 位计数器。

SCT_CONFIG 寄存器用于配置基本计数器。

SCT_CTRL 寄存器用于直接控制基本计数器，不需要等待任何事件。

SCT_OUTPUTDIRCTRL 寄存器指定（每个输出）基本计数器的计数方向。

SCT_COUNT 寄存器保存根据每个 SCT 时钟移动的当前计数器值。

输入和输出寄存器就像 GPIO 的控制寄存器，它们在 SCT 输入和输出信号上读取和写入电压电平。输入寄存器中的每个位都是针对每个 SCT 输入信号的。输出寄存器中的每个位都是针对每个 SCT 输出信号的。

注意

SCT 输入和输出信号不直接连接 GPIO 引脚。输入 MUX 模块对 SCT 输入/输出信号与外部 GPIO 引脚之间的连接进行映射。

2.2 事件的产生

以下条件定义事件：

- 计数器匹配条件。
- 输入（或输出）条件，如上升或下降的边沿或电平。

- 匹配和/或输入/输出条件的组合。
- 在双向模式下，可以根据计数方向产生事件。

SCT_MATCH 寄存器中的值用于通过比较计数器的值来创建事件。匹配事件发生在计数器（或将）递增到下一个值的 SCT 时钟中。在匹配事件之后，SCT_MATCH 将从相应的 SCT_MATCHRELn 寄存器加载新值。

在每个事件的 SCT_EVn_CTRL 寄存器中设置引脚条件，组合模式，双向模式。

2.3 事件的运作

一旦事件发生，可以：

- 限制、终止、启动或停止计数器或改变其方向。
- 改变状态。
- 在输出引脚上输出指示电压电平。
- 捕获当前计数器值。
- 触发中断。
- 触发 DMA。

SCT_LIMIT、SCTHALT、SCT_STOP 和 SCT_START 寄存器将在事件发生时设置计数器操作。 寄存器中的每个位都用于一个事件。 例如，在 LIMIT 中设置位 2 意味着当事件 2 发生时，计数器将直接转向相反的方向或直接回到零(根据寄存器中 SCT_OUTPUTDIRCTRL 设置。

SCT_OUTn_SET 和 SCT_OUTn_CLR 寄存器就像 GPIO 的输出控制寄存器，但只由事件驱动，而不是由软件直接输出到引脚。然后在 SCT 的输出引脚上写入逻辑电压电平。每个 SCT_OUTn_SET/OUTn_CLR 寄存器都是针对特定的 SCT 输出引脚的，而此寄存器中的每个位都定义了哪些事件可以操作此引脚。

如果在寄存器 SCT_CAPCTRL 中某几个事件启用捕获功能，一旦发生任何启用的事件，当前计数器值将在寄存器 SCT_CAPn 中被捕获。在这种情况下，SCT_CAPCTRL 寄存器和 SCT_CAPn 寄存器的每个索引对应于一个捕获监视器，而寄存器中 SCT_CAPCTRL 每个位对应于一个可能导致捕获操作的事件。

当事件发生时，SCT_DMAREQ0 和 SCT_DMAREQ1 用于设置对 DMA 控制器的操作。此寄存器中的每个位用于每个事件的索引号。SCT 有两个 DMA 触发源。

SCT_EVEN 用于设置事件发生时中断控制器的操作。此寄存器中的每个位用于每个事件的索引号。

2.4 SCT 状态机

状态机的状态保存在 SCT_STATE 寄存器中。 它的值可以根据事件更新。在 SCT_EVn_CTRL 寄存器中，对于每个事件，有几个字段可将操作配置给状态机。

STATELD 和 StateV 将告诉 SCT 如何在事件发生时更新状态值：

- 当 STATELD=0 时，STATE V 值被添加到 SCT_STATE 中。
- 当 STATELD=1 时，STATE V 值被加载到 SCT_STATE 中。

重要的是，只有当前状态下启用的事件才能捕获事件条件。每个事件的 SCT_EVn_STATE 寄存器用于保持事件在任何状态下可用。寄存器中的每个位对应于一个可用状态。 例如，在 SCT_EV2_STATE 中设置位 3 意味着事件 2 可以在状态 3 中可用。 然后，使用 SCT_EVn_STATE 寄存器创建状态之间的例程。

3 摄像机信号和与 MCU 的连接

在演示应用中使用了自带晶体时钟源的 OV7620 相机模块。 用一组信号连接到 MCU：

- [输入]I2C/SCCB 总线来设置摄像机传感器的内部寄存器。
- [输出]16b/8b 像素并行像素数据。
- [输出]像素的三个同步信号：PCLK 表示新像素，HREF 表示新行，VSYNC 表示新帧。

表 1. 相机引脚和连接

相机引脚	MCU 引脚	MCU 外设
SCCB_SDA	PIO0_26	I2C2_SDA
SCCB_SCL	PIO0_27	I2C2_SCL
PCLK	PIO0_17	SCT_GPI7 > IN2
HREF	PIO0_14	SCT_GPI1 > IN1
VSYNC	PIO0_13	SCT_GPI0 > IN0
D0	PIO1_24	GPIO1
D1	PIO1_25	GPIO1
D2	PIO1_26	GPIO1
D3	PIO1_27	GPIO1
D4	PIO1_28	GPIO1
D5	PIO1_29	GPIO1
D6	PIO1_30	GPIO1
D7	PIO1_31	GPIO1

要读取可用的像素数据序列，MCU 应该等待 VSYNC START（VSYNC 引脚上的下降边沿）作为一帧，然后等待 HREF START（HREF 引脚上的上升边沿）为新的一行，最后从每个 PCLK（PCLK 引脚上的上升边沿）捕获来自总线的像素数据)。 在一行的末尾，使用 HREF END（HREF 引脚上的下降边沿）来告诉 MCU 在下一次 HREF START 之前，以下像素不可用。在帧的末尾，VSYNC END（VSYNC 引脚上的上升边沿）被用来告诉以下行在下一次 VSYNC START 之前不可用。只有可用行中的可用像素是所需的传感图像数据。

实际上，在 LPC54605 项目中，INPUT MUX 模块用于将外部 SCTGPIX 引脚映射到 SCTINX 信号。

```
#define APP_SCT_INPUT_LINE_VSYNC 0U
#define APP_SCT_INPUT_LINE_HREF 1U
#define APP_SCT_INPUT_LINE_PCLK 2U
     /*
     * Camera IO PinMux SCT line
     * VSYNC -> PIO0_13 -> SCT0_GPI0 -> IN0
     * HREF -> PIO0_14 -> SCT0_GPI1 -> IN1
     * PCLK -> PIO0_17 -> SCT0_GPI7 -> IN2
     */
     INPUTMUX_AttachSignal(INPUTMUX, 0U, kINPUTMUX_SctGpi0ToSct0); /* IN0. */
     INPUTMUX_AttachSignal(INPUTMUX, 1U, kINPUTMUX_SctGpi1ToSct0); /* IN1. */
     INPUTMUX_AttachSignal(INPUTMUX, 2U, kINPUTMUX_SctGpi7ToSct0); /* IN2. */
```

4 设计一个状态机来捕获相机同步信号

由于相机将不断发送大量数据，我们希望完全通过硬件处理同步信号的捕获和数据移动。 如果使用引脚中断来捕获这些同步信号，MCU 将花费大量的工作负载。 即使这样，在 MCU 处理数据时，带有软件的 MCU 可能会丢失一些同步令牌。 因此，在所需的状态机中，SCT 自动处理状态之间的所有转换和发送触发命令到 DMA。

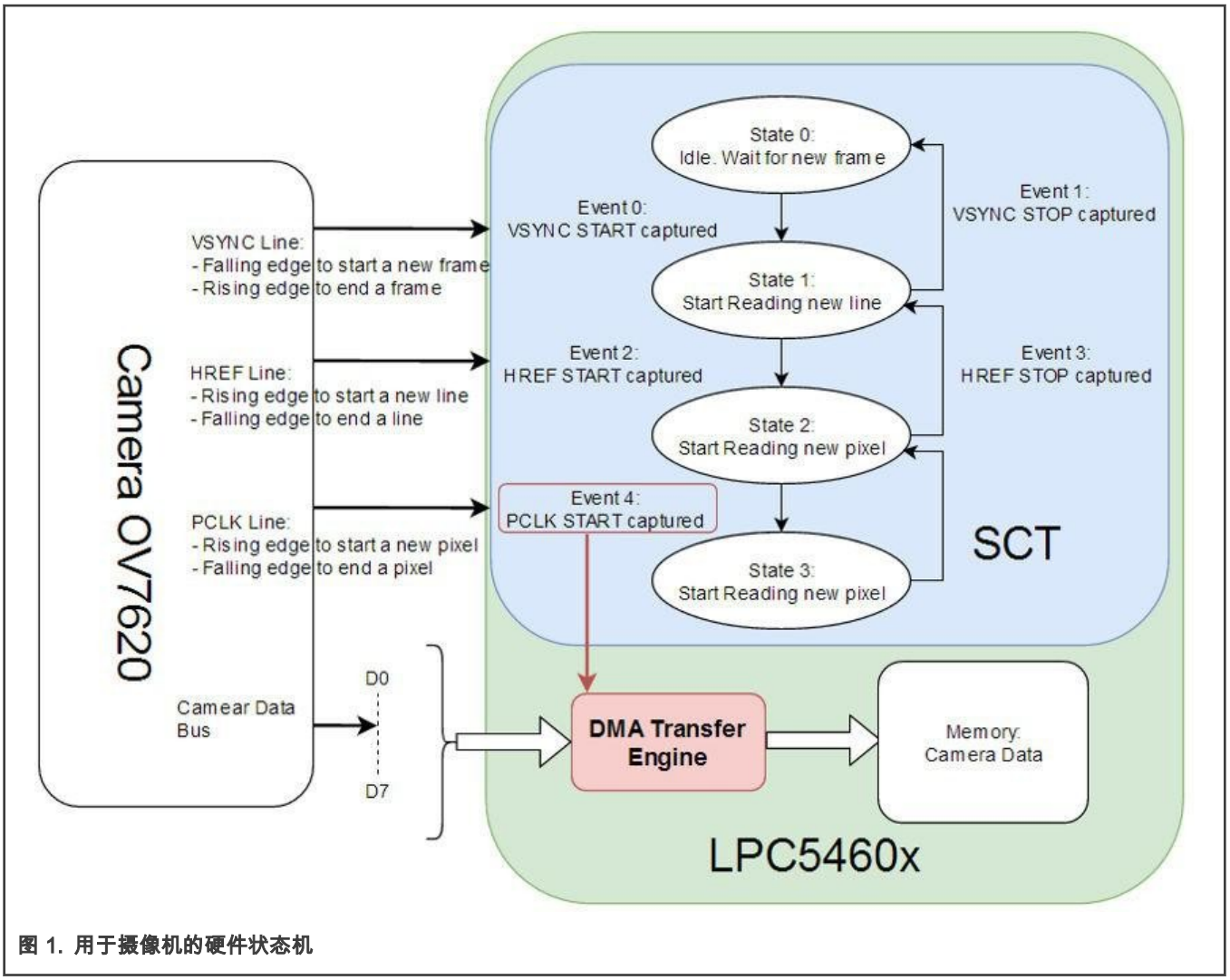

图 1. 用于摄像机的硬件状态机

如 图 1 所示，设计了一种基于 SCT 的状态机，用于捕获摄像机的同步信号，并触发 DMA 将可用像素数据搬移到用户内存。硬件实现所有功能，并自动运行，没有任何软件交互。

五个事件被预先定义为从一个状态移动到另一个状态的条件。

- 事件 0：VSYNC STAET（VSYNC 线上的下降边沿）到 SCT 输入引脚。
- 事件 1：VSYNC END（VSYNC 线上的上升边沿）到 SCT 输入引脚。
- 事件 2：HREF STAET（HREF 线上的上升边沿）到 SCT 输入引脚。
- 事件 3：HREF END（HREF 线上的下降边沿）到 SCT 输入引脚。
- 事件 4：PCLK STAET（PCLK 线上的上升边沿）到 SCT 输入引脚。

在源代码中，它们是用相关代码定义的。

```
#define APP_SCT_EVENT_VSYNC_START 0U /* IN0 falling edge. */
#define APP_SCT_EVENT_VSYNC_END   1U /* IN0 rising  edge. */
#define APP_SCT_EVENT_HREF_START  2U /* IN1 rising  edge. */
#define APP_SCT_EVENT_HREF_END    3U /* IN1 falling edge. */
#define APP_SCT_EVENT_PCLK_START  4U /* IN2 rising  edge. */
```

然后定义四个状态来表示临时状态：

- 状态 0 为初始状态，等待新帧。在状态 0 中，只有事件 0 可用，而其他事件被屏蔽。一旦事件 0 发生（VSYNC START 出现），它就会移动到状态 1 以等待新行。
- 状态 1 正在等待新行。状态 1 有两条出路：
  - 对于事件 1（VSYNC STOP 出现），它将返回到新帧的状态 0。
  - 对于事件 2（HREF START 出现），它将进入状态 2，并等待当前行中的第一个像素数据。
- 状态 2 正在等待一个新的像素。状态 2 两条出路：
  - 对于事件 1（VSYNC STOP 出现），它将返回到新帧的状态 0。
  - 对于事件 3（HREF STOP 出现），它将返回到状态 1 以获得新行。
  - 对于事件 4（PCLK START 出现），它将进入其阴影状态 3。
  - 每次切换到状态 2 时，都会从 SCT 生成一个 DMA 触发，以告诉 DMA 从数据引脚移动像素数据。
- 状态 3 是状态 2 的阴影状态。它也在等待一个新的像素。它接受相同的移动条件返回到状态 0 和状态 1。然而，在这里，我们想要在一行中采样替代像素，以减少数据的计数，因为并非所有像素都是必要的（相机已经配置了替代的行样本模式）。因此，状态 3 可以被认为是跳过（或消耗）到 DMA 的额外的（不必要的）触发。

在源代码中，它们是用相关代码定义的。

```
#define APP_SCT_STATE_WAIT_NEW_FRAME  0U
#define APP_SCT_STATE_WAIT_NEW_LINE   1U
#define APP_SCT_STATE_WAIT_NEW_PCLK   2U
#define APP_SCT_STATE_WAIT_NEXT_FRAME 3U
```

如果状态机在帧传输的中间启动，它将在没有任何移动的情况下等待，因为除了初始状态下的事件 0(等待 VSTART)外，所有事件都被禁用，只有当下一个 VSYNC START 出现时，状态机才会运行于状态之间的转换。

在这里，DMA 被配置为突发模式。一旦 DMA 触发出现，DMA 控制器将一个像素数据从 PIO1[24：31]移动到用户内存。在移动一行像素（当前演示应用程序中一行 320 个像素）之后，将执行 DMA 传输中断。 在 DMA ISR 函数中，必须重新配置传输任务，因为 LPC DMA 不能支持超过 1024 项的更长的传输。在行同步不活动期间，有足够的时间运行该软件。 使用软件计数器来计数行数。当为图像收集足够的行（当前演示应用程序中的 240 行）时，它可以告诉更高层的应用软件，MCU 的 RAM 中已经准备好了完整的图像。

在 SCT 模块中还可以对行启动，行停止，帧启动，帧结束的事件进行监控。例如，可以将事件 1 中的帧结束配置为生成 SCT 中断，并告诉更高层的应用软件图像已经准备好。 这种方式更适合和推荐，因为 DMA 和 SCT 可以分别处理不同的任务：DMA ISR 可以集中在数据传输上，SCT ISR 可以集中在事件检测上。

单片机源项目中最重要的工作是对每个事件进行编程。最后，在对 SCT 配置进行编程时，我们必须将状态机的视图从面向状态转换为面向事件。这意味着，即使是状态机在描述状态和它们之间在事件条件下的转换，我们不得不说它们是处于生存状态的事件。那么，代码是：

- 设置事件的条件和目标状态：

```
/* setup the event operations. */
    /* VSYNC START event:
     * - APP_SCT_INPUT_LINE_VSYNC occurs on VSYNC input falling edge.
     * - switch to APP_SCT_STATE_WAIT_NEW_LINE, wait for a new line.
     */
    SCT0->EVENT[APP_SCT_EVENT_VSYNC_START].CTRL = SCT_EVENT_CTRL_MATCHSEL(0) /* no use. */
                                                | SCT_EVENT_CTRL_HEVENT(0)   /* no use. */
                                                | SCT_EVENT_CTRL_OUTSEL(0)   /* input pin
trigger. */
                                                |
SCT_EVENT_CTRL_IOSEL(APP_SCT_INPUT_LINE_VSYNC) /* VSYNC pin. */
                                                | SCT_EVENT_CTRL_IOCOND(2) /* pin falling edge
trigger. */
                                                | SCT_EVENT_CTRL_COMBMODE(2) /* use io without
counter. */
```

```
                                                | SCT_EVENT_CTRL_STATELD(1) /* load state value.
*/
                                                |
SCT_EVENT_CTRL_STATEV(APP_SCT_STATE_WAIT_NEW_LINE) /* new state. */
                                                | SCT_EVENT_CTRL_MATCHMEM(0) /* no use. */
                                                | SCT_EVENT_CTRL_DIRECTION(0) /* no use. */
                                                ;
    /* HREF START event：
     * - APP_SCT_INPUT_LINE_HREF occurs on HREF input rising edge.
     * - switch to APP_SCT_STATE_WAIT_NEW_PCLK, wait for a new pixel.
     */
    SCT0->EVENT[APP_SCT_EVENT_HREF_START ].CTRL = SCT_EVENT_CTRL_MATCHSEL(0) /* no use. */
                                                | SCT_EVENT_CTRL_HEVENT(0)   /* no use. */
                                                | SCT_EVENT_CTRL_OUTSEL(0)   /* input pin
trigger. */
                                                |
SCT_EVENT_CTRL_IOSEL(APP_SCT_INPUT_LINE_HREF) /* HREF  pin. */
                                                | SCT_EVENT_CTRL_IOCOND(1) /* pin rising trigger.
*/
                                                | SCT_EVENT_CTRL_COMBMODE(2) /* use io without
counter. */
                                                | SCT_EVENT_CTRL_STATELD(1) /* load state value.
*/
                                                |
SCT_EVENT_CTRL_STATEV(APP_SCT_STATE_WAIT_NEW_PCLK) /* new state. */
                                                | SCT_EVENT_CTRL_MATCHMEM(0) /* no use. */
                                                | SCT_EVENT_CTRL_DIRECTION(0) /* no use. */
                                                ;
    /* PCLK START event：
     * - APP_SCT_INPUT_LINE_PCLK occurs on PCLK input rising edge.
     * - switch to APP_SCT_STATE_WAIT_NEW_PCLK itself, wait for a new pixel.
     */
    SCT0->EVENT[APP_SCT_EVENT_PCLK_START ].CTRL = SCT_EVENT_CTRL_MATCHSEL(0) /* no use. */
                                                | SCT_EVENT_CTRL_HEVENT(0)   /* no use. */
                                                | SCT_EVENT_CTRL_OUTSEL(0)   /* input pin
trigger. */
                                                |
SCT_EVENT_CTRL_IOSEL(APP_SCT_INPUT_LINE_PCLK) /* PCLK pin. */
                                                | SCT_EVENT_CTRL_IOCOND(1) /* pin rising trigger.
*/
                                                | SCT_EVENT_CTRL_COMBMODE(2) /* use io without
counter. */
                                                | SCT_EVENT_CTRL_STATELD(1) /* load state value.
*/
                                                |
SCT_EVENT_CTRL_STATEV(APP_SCT_STATE_WAIT_NEW_PCLK) /* new state. */
                                                | SCT_EVENT_CTRL_MATCHMEM(0) /* no use. */
                                                | SCT_EVENT_CTRL_DIRECTION(0) /* no use. */
                                                ;
    /* HREF END event：
     * - APP_SCT_INPUT_LINE_HREF occurs on HREF input falling edge.
     * - switch to APP_SCT_STATE_WAIT_NEW_LINE, wait for a new line.
     */
    SCT0->EVENT[APP_SCT_EVENT_HREF_END   ].CTRL = SCT_EVENT_CTRL_MATCHSEL(0) /* no use. */
                                                | SCT_EVENT_CTRL_HEVENT(0)   /* no use. */
                                                | SCT_EVENT_CTRL_OUTSEL(0)   /* input pin
trigger. */
                                                |
SCT_EVENT_CTRL_IOSEL(APP_SCT_INPUT_LINE_HREF) /* HREF pin.*/
                                                | SCT_EVENT_CTRL_IOCOND(2) /* pin falling
```

```
trigger. */
                                                | SCT_EVENT_CTRL_COMBMODE(2) /* use io without
counter. */
                                                | SCT_EVENT_CTRL_STATELD(1) /* load state value.
*/
                                                |
SCT_EVENT_CTRL_STATEV(APP_SCT_STATE_WAIT_NEW_LINE) /* new state. */
                                                | SCT_EVENT_CTRL_MATCHMEM(0) /* no use. */
                                                | SCT_EVENT_CTRL_DIRECTION(0) /* no use. */
                                                ;
    /* VSYNC END event：
     * - APP_SCT_INPUT_LINE_VSYNC occurs on VSYNC input rising edge.
     * - switch to APP_SCT_STATE_WAIT_NEW_FRAME, wait for a new frame.
     */
    SCT0->EVENT[APP_SCT_EVENT_VSYNC_END  ].CTRL = SCT_EVENT_CTRL_MATCHSEL(0) /* no use. */
                                                | SCT_EVENT_CTRL_HEVENT(0)   /* no use. */
                                                | SCT_EVENT_CTRL_OUTSEL(0)   /* input pin
trigger. */
                                                |
SCT_EVENT_CTRL_IOSEL(APP_SCT_INPUT_LINE_VSYNC) /* VSYNC pin.*/
                                                | SCT_EVENT_CTRL_IOCOND(1) /* pin rising trigger.
*/
                                                | SCT_EVENT_CTRL_COMBMODE(2) /* use io without
counter. */
                                                | SCT_EVENT_CTRL_STATELD(1) /* load state value.
*/
                                                |
SCT_EVENT_CTRL_STATEV(APP_SCT_STATE_WAIT_NEW_FRAME) /* new state */
                                                | SCT_EVENT_CTRL_MATCHMEM(0) /* no use. */
                                                | SCT_EVENT_CTRL_DIRECTION(0) /* no use. */
                                                ;
```

- 设置它们可以生存的状态的事件：

```
  /* setup the enabled event in each state. */
    /* APP_SCT_EVENT_VSYNC_START event is used in these states：
     * - APP_SCT_STATE_WAIT_NEW_FRAME
     * - APP_SCT_STATE_WAIT_NEXT_FRAME (optional)
     */
    SCT0->EVENT[APP_SCT_EVENT_VSYNC_START].STATE = (1U << APP_SCT_STATE_WAIT_NEW_FRAME )
                                                 | (1U << APP_SCT_STATE_WAIT_NEXT_FRAME)
                                                 ;
    /* APP_SCT_EVENT_HREF_START is used in these states：
     * - APP_SCT_STATE_WAIT_NEW_LINE
     */
    SCT0->EVENT[APP_SCT_EVENT_HREF_START].STATE  = (1U << APP_SCT_STATE_WAIT_NEW_LINE  )
                                                 ;
    SCT0->EVENT[APP_SCT_EVENT_PCLK_START].STATE  = (1U << APP_SCT_STATE_WAIT_NEW_PCLK  )
                                                 ;
    /* APP_SCT_EVENT_VSYNC_END event can be available in all state.
     * It would switch to APP_SCT_STATE_WAIT_NEW_FRAME.
     */
    SCT0->EVENT[APP_SCT_EVENT_VSYNC_END  ].STATE = (1U << APP_SCT_STATE_WAIT_NEW_FRAME )
                                                 | (1U << APP_SCT_STATE_WAIT_NEW_LINE  )
                                                 | (1U << APP_SCT_STATE_WAIT_NEW_PCLK  )
                                                 | (1U << APP_SCT_STATE_WAIT_NEXT_FRAME)
                                                 ;
    /* APP_SCT_EVENT_VSYNC_END only occus in APP_SCT_STATE_WAIT_NEW_FRAME state. */
    SCT0->EVENT[APP_SCT_EVENT_HREF_END  ].STATE  = //(1U << APP_SCT_STATE_WAIT_NEW_FRAME )
                                                   //(1U << APP_SCT_STATE_WAIT_NEW_LINE  )
```

```
                                                    (1U << APP_SCT_STATE_WAIT_NEW_PCLK  )
                                                //| (1U << APP_SCT_STATE_WAIT_NEXT_FRAME)
```

5 演示

在实际演示中，使用 OLED 屏幕在运行时显示摄像机感知图像。在运行演示时，绘制一个箭头作为视觉目标。从 图 2 可以看出，带有目标箭头的图像是从摄像机中捕获的，并显示在 OLED 屏幕中。

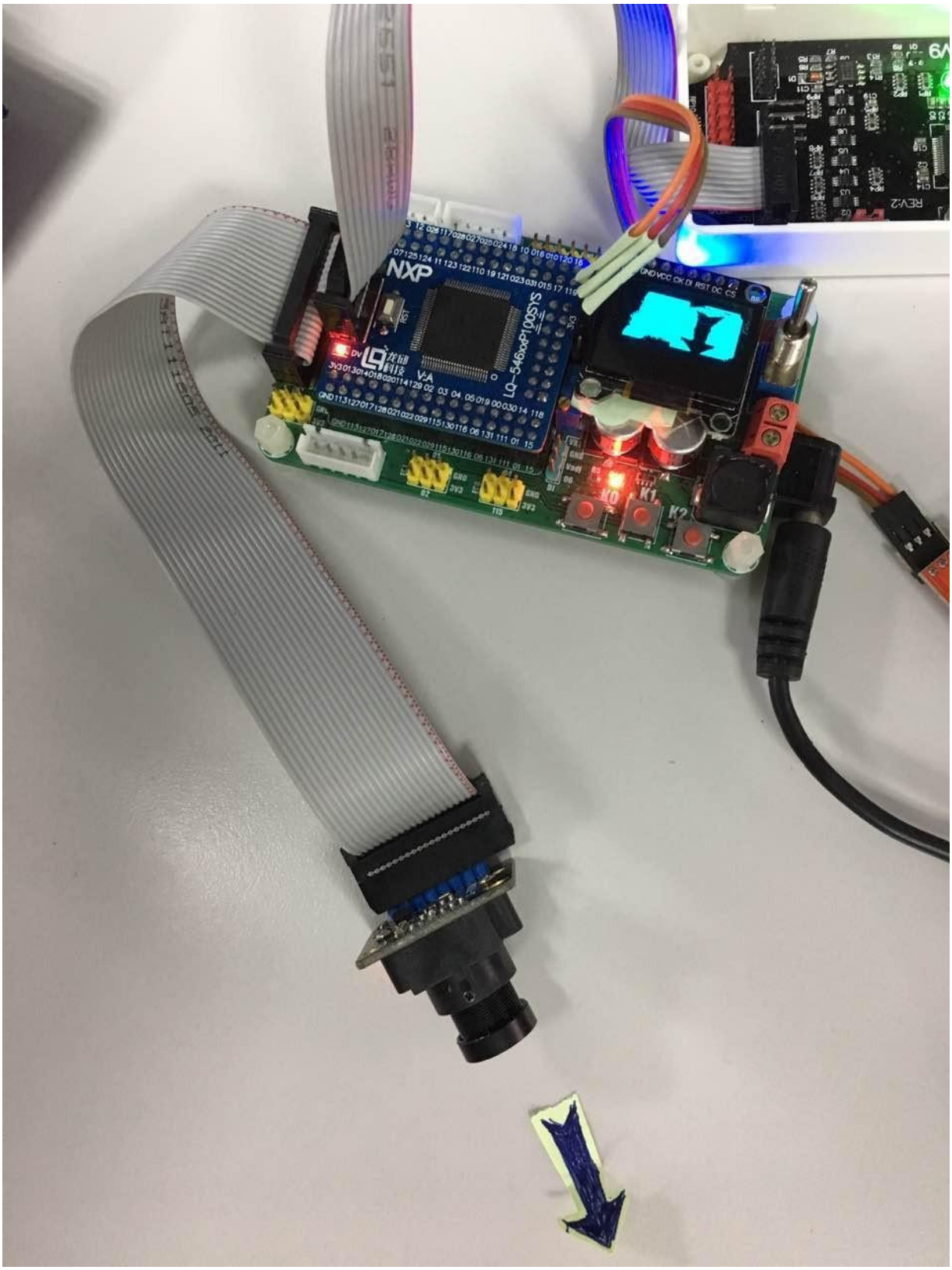

图 2. LPC5460x 相机演示

6 修改记录

版本号	发布日期	说明
0	2019 年 6 月	初次发布

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2019 年 9 月 16 日

Document identifier: AN12583

arm