AN12686

使用FlexIO在i.MX RT上模拟并行摄像头接口

修订版0 — 01/2020 应用说明

作者：恩智浦半导体

目录

1 简介

i.MX RT交叉嵌入式处理器系列采用了恩智浦对Arm®Cortex®-M7内核的高级实现。该处理器家族有一些发展，到目前为止，它从低端部分到高端部分衍生了几个部分。摄像头接口等高级功能通常仅集成在高端部件中。高端部件（例如i.MX RT117x）具有并行和MIPI CSI接口。中端部件（例如i.MX RT105x和RT106x）仅具有并行摄像头接口。低端部件（例如i.MX RT101x和RT102x）没有专用的摄像头接口。但是，有些应用需要具有低性能和低成本要求的摄像头接口。在这种情况下，FlexIO模块是满足此要求的最佳选择。 FlexIO是来自恩智浦的高度可配置的IP外设，集成在各种MCU中。

本文档介绍了如何使用FlexIO仿真并行摄像头接口，以从基于i.MX RT1010的摄像头设备接收实时图像数据。

RT1010处理器集成了运行频率高达500 MHz的Cortex-M7内核。 128 KB的片上RAM可以灵活配置为核心紧密耦合存（TCM）或通用RAM。 80 LQFP封装大大降低了PCB成本。总体而言，i.MX RT1010适合于高计算，低内存和低成本应用。

2 FlexIO 概述

FlexIO是高度可配置的。 它允许您实现各种功能，包括：

- 仿真串行通信接口，例如UART，SPI，I2C，I2S等。
- 并行通信接口的仿真，例如摄像头接口，Motorola 68K总线，Intel 8080总线等。
- 生成PWM波形。
- 逻辑功能的实现
- 执行状态机功能。

2.1 FlexIO功能

i.MX RT FlexIO模块提供以下关键功能：

- 具有发送，接收和数据匹配模式的32位移位寄存器（也称为移位器），用于连续传输的双缓冲结构以及支持大传输量的级联机制。
- 高度灵活的16位定时器，支持各种内部或外部触发以及复位，递减，启用和禁用条件。
- 自动开始/停止位生成和检查。
- 4、8、16或32多位移位宽度，用于并行接口支持。
- 中断，DMA或轮询的发送/接收操作。

- 可编程波特率，在停止模式下支持异步操作。

下图显示了该模块的高级概述。

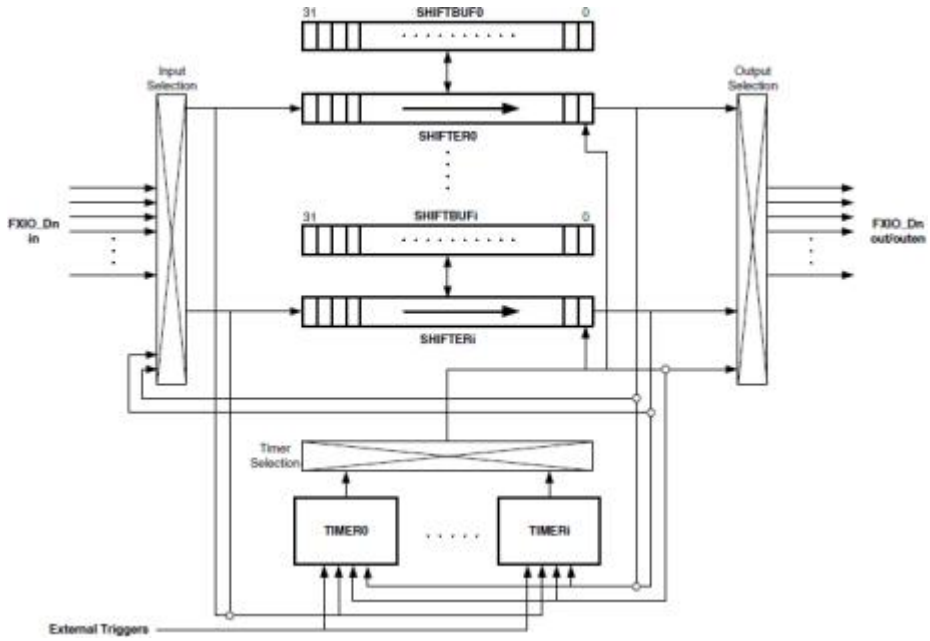

图1.硬件连接图

2.2 内部逻辑连接

为了满足各种要求，内部逻辑连接非常灵活，其中一些复杂。 以下是一些功能：

- 任何一个或多个引脚都可以分配给一个移位器进行输入或输出，如图1所示。
- 可以将任何计时器分配给换挡器进行换挡控制，如图1所示。
- 可以将任何引脚分配给计时器以进行计时器输入或输出，如图1所示。
- 计时器触发可以源自移位器标志，引脚或FlexIO模块外部，如图1所示。
- 移位可以在移位时钟的上升沿或下降沿。
- 引脚方向和极性是可配置的。
- 触发极性是可配置的。
- 计时器的启用，禁用，递减和复位条件可能源自触发器，引脚，相邻计时器等。

2.3 变速杆和计时器

FlexIO由移位器，计时器和引脚组成。 可以从PARAM寄存器中读取给定处理器的这些资源量。 例如，i.MX RT1010上的FlexIO模块具有8个移位器，8个计时器和32个引脚。

发射和接收模式是移位器的两种基本模式。 当移位器处于发送模式时，它将从其缓冲寄存器加载数据，然后将数据移出至其分配的一个或多个引脚。 当移位器处于接收模式时，它将数据从其分配的一个或多个引脚移出，然后将数据存储到其缓冲寄存器中。 装载，存储和移位操作由移位器分配的计时器控制。

如果需要，还可以将定时器配置为不同的工作模式，包括双8位计数器波特/位模式，双8位计数器PWM模式和单16位计数器模式。

双8位计数器的波特率/位模式通常用于构造数据发送器。在此模式下，16位定时器的低8位对模块时钟源进行分频，以生成所需的波特率，而高8位对帧的移位位进行计数。使能后，定时器将从其比较寄存器加载初始值并开始递减计数。当低8位递减为零时，计时器的移位时钟及其输出信号将被触发以产生上升沿或下降沿。高8位递减1。移位时钟驱动移位器。定时器输出信号通常驱动时钟输出引脚，例如SPI主设备的SCK和8080总线的WR。之后，低8位重新加载初始值以开始另一个递减周期。这两个递减循环组成一个换挡循环，从而驱动换挡器移动一个节拍。当所有16位递减为零时，移位中的所有数据位均被移出。然后在另一个传输帧之前禁用计时器。

双8位计数器PWM模式用于生成PWM输出。低8位用于配置定时器移位时钟的高电平周期，高8位用于配置移位时钟的低周期电平。

通常使用单个16位计数器模式来创建同步通信从设备，例如SPI从设备，I2S从设备等。所有16位都用于配置定时器移位时钟。

2.4 串行和并行传输

设备上的FlexIO支持串行和并行传输。 对于两种类型的传输，数据始终在移位器中从MSB移到LSB。 在串行发送器模式下，数据从LSB（位0）逐位移出。 在串行接收器模式下，数据从MSB（第31位）逐位移位。 该过程如图2-a所示。

在并行发送器模式下，数据从移位器的n个LSB中移出。 在并行接收器模式下，数据从n个MSB移入，其中n是并行总线宽度。 图2-b显示了n = 8的用例。

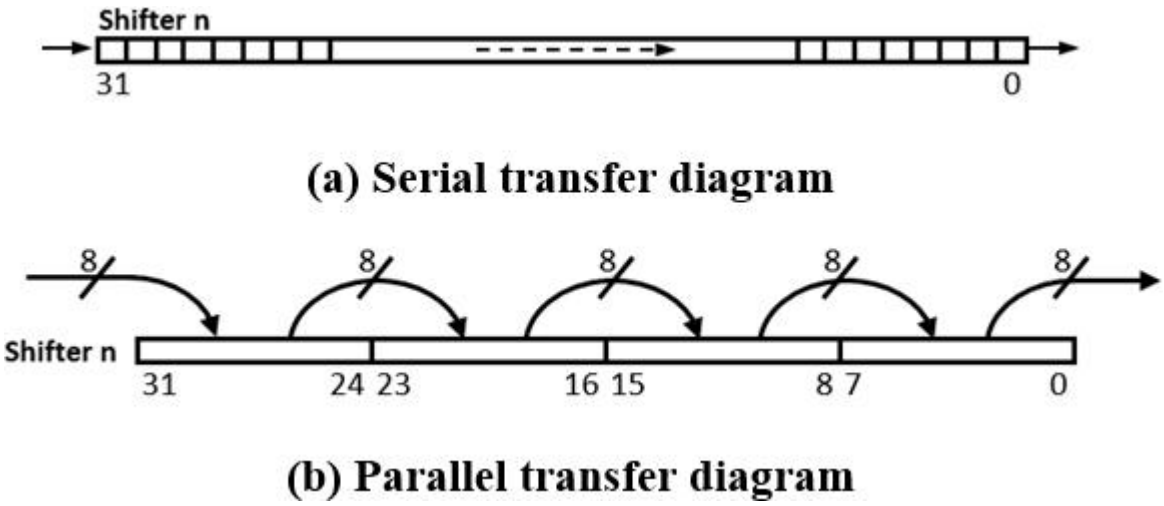

图2.串行和并行传输图

下面介绍并行传输模式：

- 在每个移位时钟上将数据移位n位，其中n是配置的总线宽度。
- 支持4、8、16或32位总线宽度。
- 它将多个移位器组合在一起以支持较大的传输大小，并使用DMA方法访问移位器缓冲寄存器，以进行高速传输和低功耗操作。 下图显示了移位器串联图，其中其他移位器用作FIFO。.

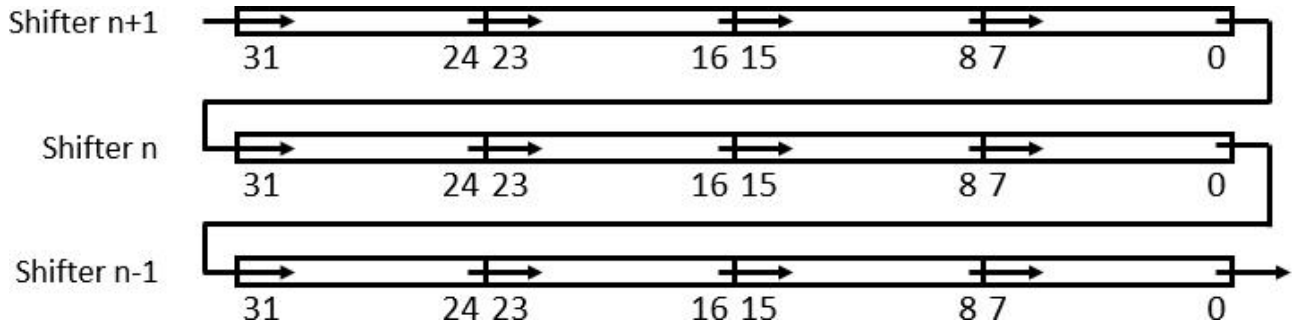

图3. 8位并行传输图的移位器串联

- 仅特定的移位器（SHIFTER0和SHIFTER4）支持将输出并行连接到引脚。 但是，所有移位器（SHIFTER0除外）都支持输出到相邻的低阶移位器。
- 同样，仅特定的移位器（SHIFTER3和SHIFTER7）支持来自引脚的并行输入。 但是，所有移位器（SHIFTER7除外）都支持来自相邻高阶移位器的输入。
- 任何FlexIO引脚都可以是并行数据输出/输入引脚。 但是，对于特定用途，引脚索引必须是连续的，例如8位总线的pin0到pin7，pin1到pin8，依此类推。

2.5 常规配置和操作

FlexIO可以仿真各种通信协议。但是，为了仿真专用外设并处理发送和接收过程，必须通过软件配置FlexIO。

通常，为实现主发送器，将移位器配置为发送模式，将分配的定时器配置为双8位计数器波特/位模式。定时器递减时钟源于模块时钟。定时器触发源自极性相反的移位标志。通过轮询/中断/ DMA填充移位器缓冲区会清除移位器标志，从而使定时器开始递减计数。计时器的减少驱动移位器将数据移出并生成时钟输出信号。

为了实现接收器，将移位器配置为接收模式。定时器以双8位计数器波特率/位模式配置用于同步主接收器，例如8080总线读取实现。此定时器模式还用于异步接收器，例如UART接收器。接收过程与主发送器的接收过程相似，但是数据被移入移位器而不是移出。对于同步接收器，分配的定时器以单个16位计数器模式配置，例如SPI从接收器和并行摄像头接口。递减时钟源自引脚输入，例如SPI SCK和摄像机PCLK信号。定时器触发来自另一个引脚，例如SPI CS和摄像机HREF信号。主设备使能定时器并通过引脚控制减量。类似地，计时器的减少会驱动移位器移入数据。

为了易于使用，NXP在MCUXpresso SDK中提供了API驱动程序和驱动程序示例。您还可以在各种应用笔记中找到这些实现的详细说明和类似的API。

3 并行摄像头接口

并行摄像头传感器通常具有8/10/16/24数据线以输出像素数据，最常用的是8位接口摄像头。 对于这种相机，如果像素为8位大小，则需要一个传输周期。 如果像素的位数超过8位，则需要额外的传输周期。 例如，RGB565（16位）格式的像素需要两个传输周期，而RGB888（24位）格式的像素需要三个传输周期。

除数据线外，还有时序控制输出VSYNC / VREF，HREF / HSYNC和PCLK。 下图显示了此应用示例中使用的摄像机传感器OV7670的时序图。

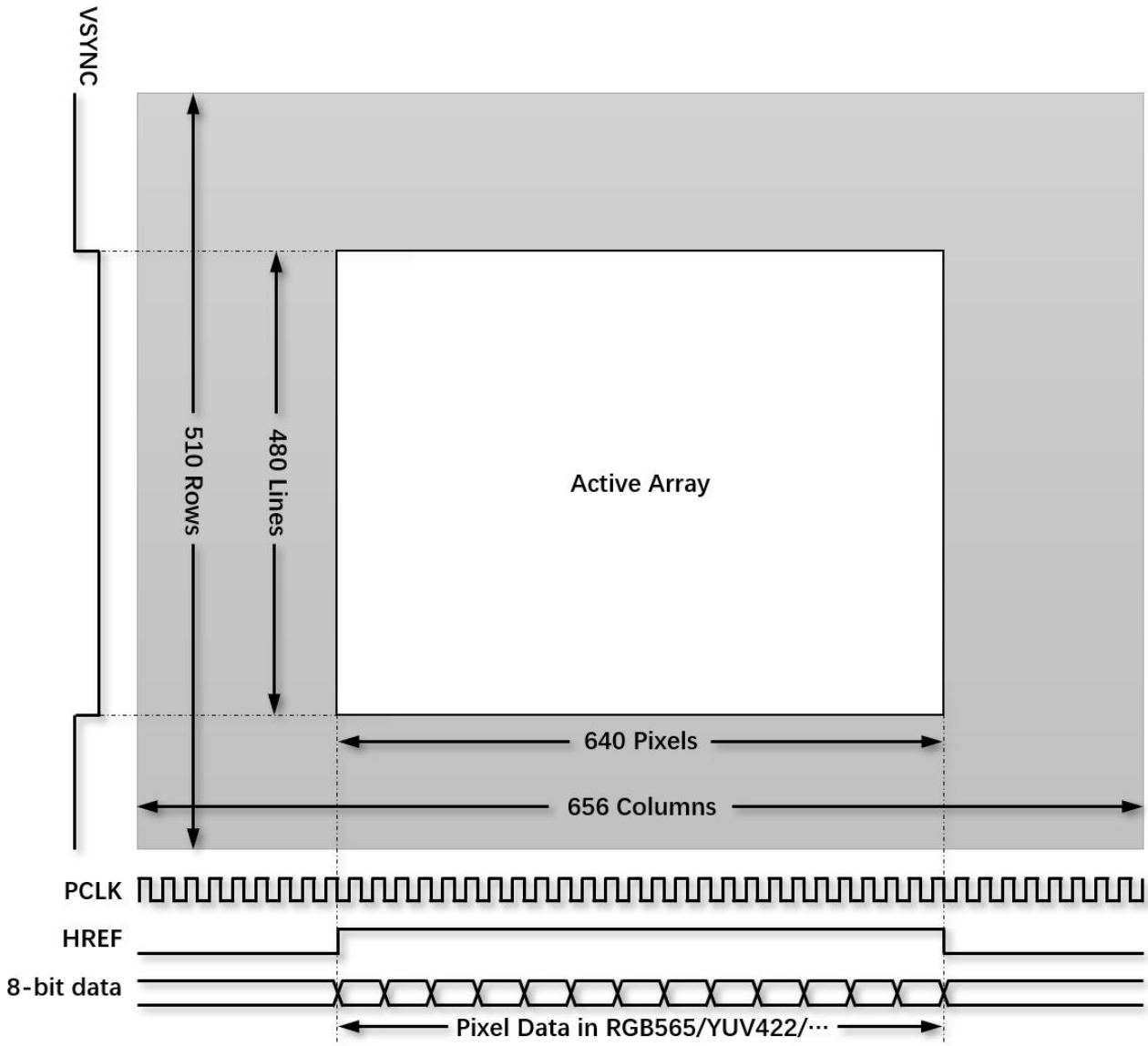

图4.并行摄像头接口时序图

上图显示了一个帧，其中包含510行，包括在VSYNC高电平期间的480条有效数据线。 数据线包含656个点，包括在HREF高电平期间的640个有效像素。 因此，分辨率为VGA（640x480）。

在上图中，两个PCLK周期用于输出16位RGB565 / YUV422格式的像素数据，高字节在后，低字节在后。

数据在PCLK下降沿更改，并在PCLK上升沿锁存。

帧分辨率可调。 不同的分辨率具有相似的信号时序。 下图说明了VGA，QVGA（320x240）和QQVGA（160x120）分辨率之间的时序差异。.

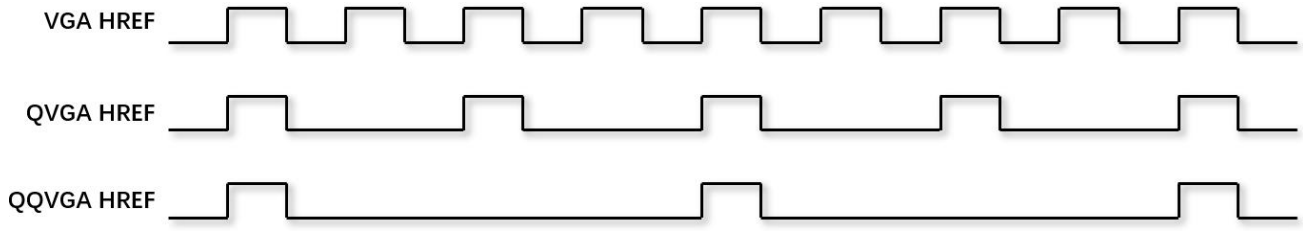

图5. VGA，QVGA和QQVGA时序

上图显示帧分辨率越低，HREF消隐区域越长。 来自不同分辨率的活动区域的大小是相同的。 不同分辨率下其他信号的时序相似。

4 使用FlexIO模拟并行摄像头接口

本节介绍如何使用FlexIO仿真i.MX RT1010的并行摄像头接口

4.1 应用硬件

该应用程序在SOC验证板上运行，因为可用的RT1010 EVK板上没有FlexIO接口接头连接摄像机模块。 使用本文档，您可以轻松地在定制的硬件平台上运行代码。 下图显示了硬件连接。

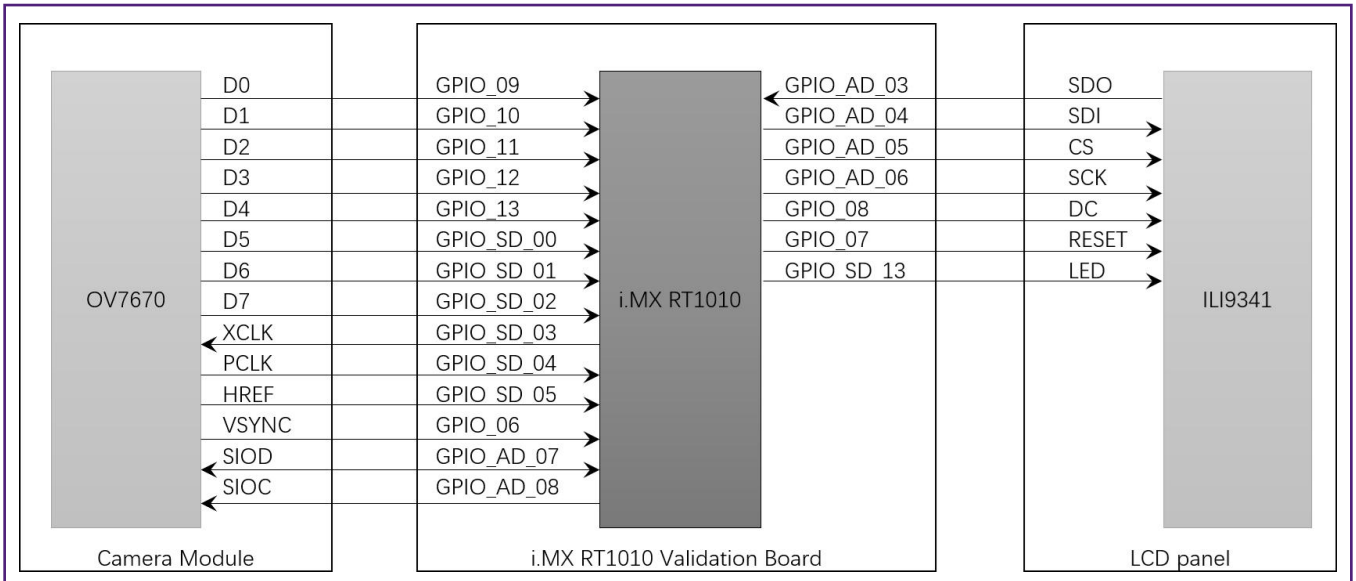

图6.硬件连接图

下表描述了摄像机信号的针脚复用器配置。

表1.相机模块的引脚配置

RT1010 引脚	引脚复用功能	摄像机信号
GPIO_09	FLEXIO1_FLEXIO01	OV7670_D0
GPIO_10	FLEXIO1_FLEXIO02	OV7670_D1
GPIO_11	FLEXIO1_FLEXIO03	OV7670_D2

Table continues on the next page...

表1.相机模块的引脚配置（续）

RT1010 引脚	引脚复用功能	摄像机信号
GPIO_12	FLEXIO1_FLEXIO04	OV7670_D3
GPIO_13	FLEXIO1_FLEXIO05	OV7670_D4
GPIO_SD_00	FLEXIO1_FLEXIO06	OV7670_D5
GPIO_SD_01	FLEXIO1_FLEXIO07	OV7670_D6
GPIO_SD_02	FLEXIO1_FLEXIO08	OV7670_D7
GPIO_SD_03	FLEXIO1_FLEXIO09	OV7670_XCLK
GPIO_SD_04	FLEXIO1_FLEXIO10	OV7670_PCLK
GPIO_SD_05	FLEXIO1_FLEXIO11	OV7670_HREF
GPIO_06	IOMUXC_GPIO_06 (GPIO1_06)	OV7670_VSYNC
GPIO_AD_07	LPI2C2_SDA	OV7670_SIOD
GPIO_AD_08	LPI2C2_SCL	OV7670_SIOC

请注意以下有关摄像机针脚复用器的问题：

- 摄像机D0～D7必须连接到8个连续的FlexIO引脚，例如本示例中的FLEXIO1引脚FLEXIO01～FLEXIO08。
- XCLK，PCLK和HREF连接到三个FlexIO引脚中的任何一个。
- VSYNC连接到GPIO引脚。
- .SIOD和SIOC连接到I2C引脚。

下表描述了LCD信号的引脚复用器配置。.

表2. LCD面板的引脚配置

RT1010 引脚	引脚复用功能	LCD 信号
GPIO_AD_03	LPSPI1_SDI	ILI9341_SDO
GPIO_AD_04	LPSPI1_SDO	ILI9341_SDI
GPIO_AD_05	LPSPI1_PCS0	ILI9341_CS
GPIO_AD_06	LPSPI1_SCK	ILI9341_SCK
GPIO_08	IOMUXC_GPIO_08 (GPIO1_08)	ILI9341_DC
GPIO_07	IOMUXC_GPIO_07 (GPIO1_07)	ILI9341_RESET
GPIO_SD_13	GPIO2_IO13	ILI9341_LED

请注意以下有关LCD引脚复用器的内容：

- LCD SDO，SDI，CS和SDK连接到RT1010 LPSPI引脚。
- DC，RESET和LED连接到GPIO引脚。

4.2 应用软件结构

.MCUXpresso SDK用于构建应用程序软件。 MCUXpresso SDK提供了外围驱动程序，SOC标头和启动文件，各种中间件以及NXP各种处理器（例如i.MX RT，Kinetis，LPC等）的示例代码。

下图显示了应用程序数据流

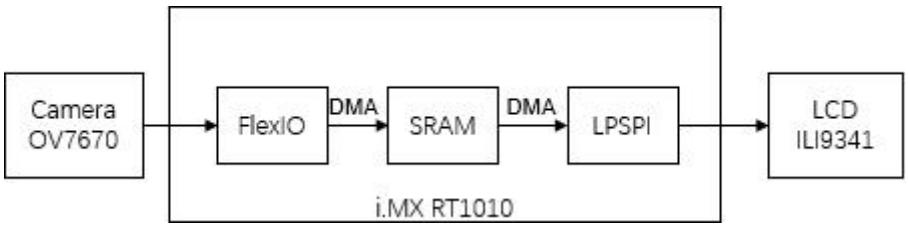

图7.数据流程图

上图显示DMA用于将数据从FlexIO复制到SRAM，以及从SRAM复制到LPSPI。 因此，在这种简单的应用程序中，CPU负载非常低。

由于SRAM空间有限（128 KB），因此使用QQVGA（160x120）帧分辨率。

4.3 FlexIO摄像头接口的操作原理

您可以将FlexIO配置为以不同的方式模拟并行摄像头接口，例如使用不同的数据总线宽度，串联的移位器数量以及所使用的特定移位器，引脚和计时器。

多拍子传输用于支持较大的传输大小。 在此，拍子是指变速操作。 一个传输序列需要计时器生成多个移位时钟。 每一个传输序列的节拍数与串联的移位器数和总线宽度有关。 一个移位器具有32位。 一个移位器支持8位总线的一次4节拍传输。 两个移位器可支持8拍，依此类推。 在此应用中，全部使用了8个移位器。 因此，对于8位总线，支持32个节拍。

下图显示了FlexIO资源的并行摄像头接口组织。.

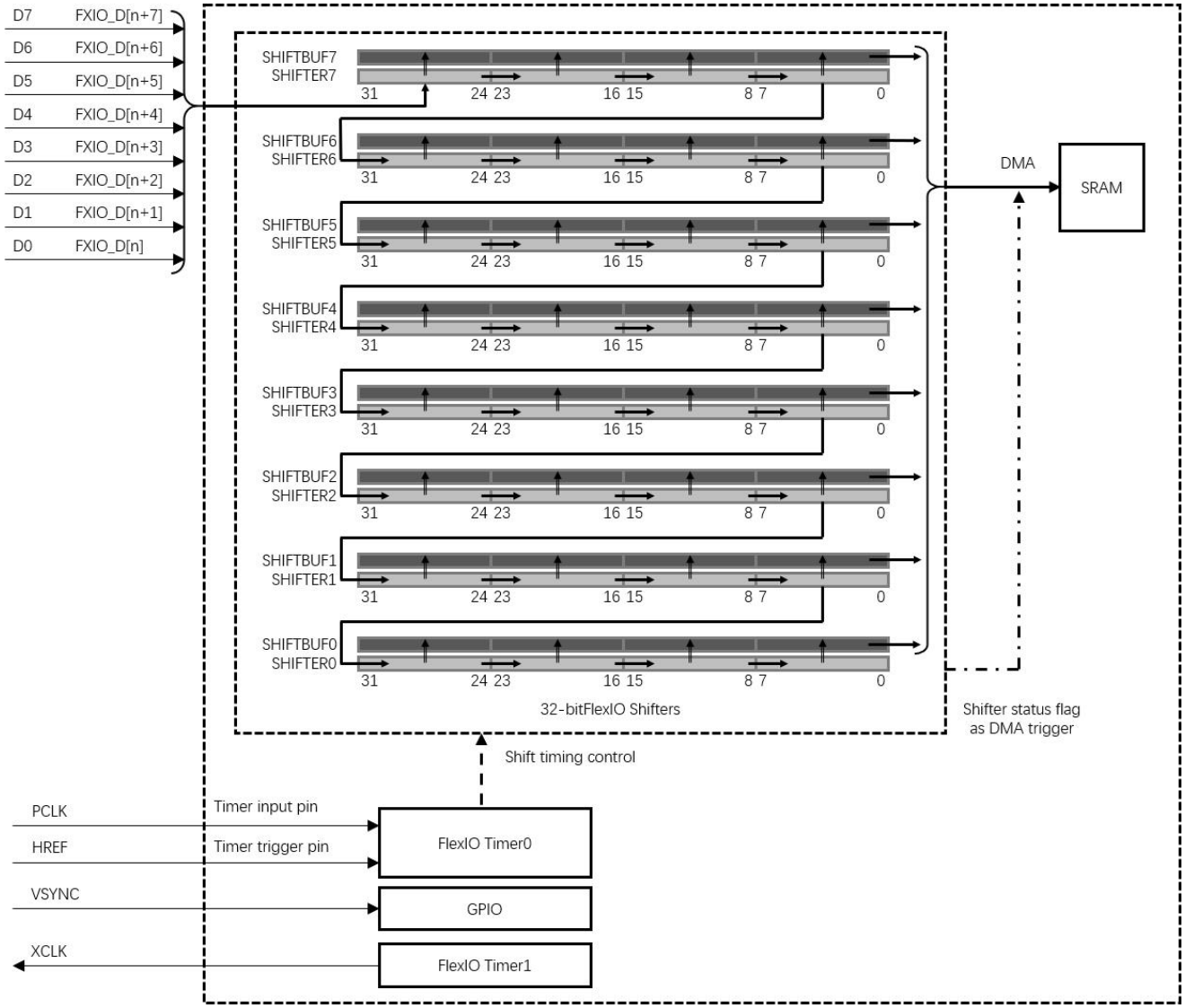

图8.并行摄像机接口的FlexIO配置

在组织中，所有8个移位器都连接在一起。 TIMER0用于控制变速杆的变速。 TIMER1用于生成XCLK（此应用中为24MHz）。 D0-D7，HREF和XCLK基于FlexIO引脚。 另一个GPIO引脚用于接收VSYNC信号。 SHIFTER0状态标志用于触发DMA请求。

转移过程如下：

1. 配置FlexIO，DMA，GPIO等。 以下部分显示了详细的配置。.
2. 在VSYNC上升沿ISR中，将DMA目标地址重新定向到新的帧缓冲区。
3. 一条新线开始转移。 HREF上升沿使能TIMER0。
4. .TIMER0与PCLK输入一起开始递减计数。 同时，TIMER0生成移位时钟，以控制移位器将数据移入。移位时钟是每个PLCK时钟生成的。 每1个移位时钟移位8位数据。
5. TIMER0递减计数为零，并在32个移位时钟后发生比较事件
6. 比较事件表示存储事件。 数据从SHIFTER0-SHIFTER7存储到移位缓冲区SHIFTBUF0-SHIFTBUF7。

7. storeing事件填满了移位器缓冲区，这将设置移位器的状态标志并触发DMA请求。
8. eDMA将数据从移位缓冲区SHIFTBUF0-SHIFTBUF7复制到SRAM。 每个请求复制32个字节。
9. 在比较事件之后，TIMER0再次从TIMCMP0加载初始值。 然后，重复步骤4-9。
10. 传输线后，HREF变为低电平。 禁用TIMER0，直到开始传输新行。 然后，重复步骤3-10。
11. 传输完帧后，VSYNC变为低电平。 然后，重复步骤2-11。

这是捕获一帧的过程。 仅要求CPU在VSYNC ISR中重新定向DMA目标地址。 捕获帧的所有其他操作由FlexIO和eDMA完成。 在VSYNC ISR中也设置了显示DMA源地址配置。

4.4 用于并行摄像机接口的FlexIO配置

摄像头接口的FlexIO配置主要包括移位器和计时器配置。 以下代码是从API驱动程序fsl_flexio_camera.c \ FLEXIO_CAMERA_Init（）中剪切下来的，实现了shifter配置。

```
/* FLEXIO_CAMERA shifter config */
shifterConfig.timerSelect   = base->timerIdx;
shifterConfig.timerPolarity = kFLEXIO_ShifterTimerPolarityOnPositive;
shifterConfig.pinConfig     = kFLEXIO_PinConfigOutputDisabled;
shifterConfig.pinSelect     = base->datPinStartIdx;
shifterConfig.pinPolarity   = kFLEXIO_PinActiveHigh;
shifterConfig.shifterMode   = kFLEXIO_ShifterModeReceive;
shifterConfig.parallelWidth = FLEXIO_CAMERA_PARALLEL_DATA_WIDTH - 1U;
shifterConfig.inputSource   = kFLEXIO_ShifterInputFromNextShifterOutput;
shifterConfig.shifterStop   = kFLEXIO_ShifterStopBitDisable;
shifterConfig.shifterStart = kFLEXIO_ShifterStartBitDisabledLoadDataOnEnable;
/* Configure the shifters as FIFO buffer. */
for (i = base->shifterStartIdx; i < (base->shifterStartIdx + base->shifterCount - 1U); i++)
{
    FLEXIO_SetShifterConfig(base->flexioBase, i, &shifterConfig);
}
shifterConfig.inputSource = kFLEXIO_ShifterInputFromPin;
FLEXIO_SetShifterConfig(base->flexioBase, i, &shifterConfig);
```

上面的代码显示了移位器的配置：

- 用于移位时钟生成的定时器选择：为此应用选择了TIMER0。
- 移位极性：移位位于移位时钟的上升沿。
- 移位器引脚I / O配置：输出被禁用，仅用于输入。
- 第一个数据引脚选择：在此应用中选择了引脚01。 并行宽度为8时，数据引脚为01～08。
- 移位器引脚极性：该引脚为高电平有效，这意味着该引脚与移位器之间的逻辑不反向。
- 移位模式：接收模式。
- 平行宽度：在此应用中为8位。
- 移位器的输入源：在此应用中，SHIFTER0～SHIFTER6的输入源分别是SHIFTER1～SHIFTER7的输出，而SHIFTER7的输入源来自数据引脚。 这是移位器串联功能。
- 移位器停止位：禁用。
- 移位器起始位：禁用。

以下代码是从API驱动程序fsl_flexio_camera.c \ FLEXIO_CAMERA_Init（）中剪切下来的，实现了移位计时器的配置。

```
    timerConfig.triggerSelect   = FLEXIO_TIMER_TRIGGER_SEL_PININPUT(base->hrefPinIdx);
    timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveHigh;
    timerConfig.triggerSource   = kFLEXIO_TimerTriggerSourceInternal;
    timerConfig.pinConfig       = kFLEXIO_PinConfigOutputDisabled;
    timerConfig.pinSelect       = base->pclkPinIdx;
    timerConfig.pinPolarity     = kFLEXIO_PinActiveHigh;
    timerConfig.timerMode       = kFLEXIO_TimerModeSingle16Bit;
    timerConfig.timerOutput     = kFLEXIO_TimerOutputZeroNotAffectedByReset;
    timerConfig.timerDecrement = kFLEXIO_TimerDecSrcOnPinInputShiftPinInput;
    timerConfig.timerReset      = kFLEXIO_TimerResetOnTimerTriggerRisingEdge;
    timerConfig.timerDisable    = kFLEXIO_TimerDisableOnTriggerFallingEdge;
    timerConfig.timerEnable     = kFLEXIO_TimerEnableOnTriggerRisingEdge;
    timerConfig.timerStop       = kFLEXIO_TimerStopBitDisabled;
    timerConfig.timerStart      = kFLEXIO_TimerStartBitDisabled;
    timerConfig.timerCompare    = 8U * base->shifterCount - 1U;
    FLEXIO_SetTimerConfig(base->flexioBase, base->timerIdx, &timerConfig);
```

上面的代码显示了计时器配置：

- 定时器触发选择：HREF信号引脚。
- 触发极性：触发器为高电平有效。
- 触发源：内部触发，表示来自FlexIO模块本身的触发。
- 定时器引脚的I / O配置：禁用输出，这意味着仅用于输入。
- 定时器引脚选择：PCLK信号引脚。
- 定时器引脚极性：该引脚为高电平有效，这意味着该逻辑从该引脚到定时器不会反转。
- 定时器模式：16位计数器模式。
- 计时器输出初始状态：启用时，计时器输出逻辑零，不受计时器复位的影响。
- 定时器递减条件：在引脚输入上递减计数，移位时钟等于引脚输入。
- 定时器复位条件：在触发上升沿。
- 定时器禁用条件：在触发下降沿。
- 定时器启用条件：在触发上升沿。
- 定时器停止位：禁用。
- 定时器启动位：禁用。
- 定时器比较（初始值）：63。移位数为（63 + 1）/ 2 = 32，等于每次DMA传输中的字节数。

4.5 FlexIO timer configuration for camera XCLK

通常，相机传感器需要时钟源才能工作。 时钟输入通常称为PCLK或MCLK。 它可以来自振荡器或处理器的时钟输出。

此应用中使用的OV7670摄像机传感器需要的时钟范围为10～48 MHz，典型值为24 MHz。 在该应用中，另一个FlexIO时钟TIMER1用于生成此要求的24 MHz时钟。 以下代码从flexio_ov7670.c \ FLEXIO_CameraXclkConfig（）中剪切，实现了用于配置TIMER1的功能。

.

```
    timerConfig.triggerSelect = 0u;
    timerConfig.triggerPolarity = kFLEXIO_TimerTriggerPolarityActiveHigh;
    timerConfig.triggerSource = kFLEXIO_TimerTriggerSourceInternal;
```

```
    timerConfig.pinConfig = kFLEXIO_PinConfigOutput;
    timerConfig.pinSelect = BOARD_CAMERA_FLEXIO_XCLK_PIN_INDEX;
    timerConfig.pinPolarity = kFLEXIO_PinActiveHigh;
    timerConfig.timerMode = kFLEXIO_TimerModeDual8BitPWM;
    timerConfig.timerOutput = kFLEXIO_TimerOutputZeroNotAffectedByReset;
    timerConfig.timerDecrement = kFLEXIO_TimerDecSrcOnFlexIOClockShiftTimerOutput;
    timerConfig.timerReset = kFLEXIO_TimerResetNever;
    timerConfig.timerDisable = kFLEXIO_TimerDisableNever;
    timerConfig.timerEnable = kFLEXIO_TimerEnabledAlways;
    timerConfig.timerStop = kFLEXIO_TimerStopBitDisabled;
    timerConfig.timerStart = kFLEXIO_TimerStartBitDisabled;
    timerConfig.timerCompare = 0x0201; /* 120MHz clock source generates 24MHz clock.*/
    FLEXIO_SetTimerConfig(BOARD_CAMERA_FLEXIO_INST, 1u, &timerConfig);
```

上面的代码显示了计时器配置：

- 定时器触发选择：PWM生成无需触发。此分配可以具有任何值。
- 触发极性：触发器为高电平有效。
- 触发源：内部触发。
- 定时器引脚的I / O配置：输出。
- 定时器引脚选择：XCLK信号引脚。
- 定时器引脚极性：该引脚为高电平有效，这意味着该逻辑不会从定时器转换到该引脚。
- 定时器模式：双8位计数器PWM模式。
- 计时器输出初始状态：启用时，计时器输出逻辑零，不受计时器复位的影响。
- 计时器递减条件：从FlexIO模块时钟（在此应用中为120 MHz）开始倒数，移位时钟等于计时器输出。
- 定时器复位条件：永不复位。
- 定时器禁用条件：从不禁用。
- 计时器启用条件：始终启用。
- 定时器停止位：禁用。
- 定时器启动位：禁用。
- 定时器比较（初始值）：0x0201。从120 MHz FlexIO时钟生成的时钟频率为120 /（（（2 + 1）+（1 +1））= 24 MHz。此配置的PWM占空比不是50％。如果需要，您可以将该值配置为0x0202，这将生成20MHz 50％占空比波形。

4.6 DMA TCD配置

当帧数据移入且移位器已满时，必须及时将数据读出到存储器中，以避免溢出。 eDMA用于数据传输以降低CPU负载。 以下代码是从flexio_ov7670.c \ configDMA（）剪切而来的，实现了DMA TCD（传输控制描述符）配置。.

```
/* Configure DMA TCD */
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].SADDR = FLEXIO_CAMERA_GetRxBufferAddress(&s_FlexioCameraDevice);
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].SOFF = 0u;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].ATTR = DMA_ATTR_SMOD(0u) | DMA_ATTR_SSIZE(5u) |
                                            DMA_ATTR_DMOD(0u) | DMA_ATTR_DSIZE(5u);
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].NBYTES_MLNO = 32u;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].SLAST = 0u;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].DADDR = (uint32_t)(*pFlexioCameraFrameBuffer);
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].DOFF = 32u;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].CITER_ELINKNO = (OV7670_FRAME_BYTES / 32u);
```

```
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].DLAST_SGA = 0;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].CSR = 0u;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].CSR |= DMA_CSR_DREQ_MASK;
    DMA0->TCD[FLEXIO_CAMERA_DMA_CHN].BITER_ELINKNO = (OV7670_FRAME_BYTES / 32u);
```

上面的代码显示了DMA TCD配置：

- 源地址：FlexIO SHIFTBUF0地址。
- 源地址偏移量：0，表示无偏移量。
- 源地址模数：0，表示源地址模数被禁用。
- 每次访问的源数据传输大小：5，表示32字节突发。
- 目标地址模数：0，表示目标地址模数被禁用。
- 每次访问的目标数据传输大小：5，表示32字节突发。
- 次循环字节数：32个字节。
- 源次循环偏移映射：禁用。
- 目的地次循环偏移映射：禁用。
- 上次源地址调整：0，表示不调整。
- 目标地址：RAM缓冲区地址。
- 目标地址偏移量：0，表示无偏移量。
- 主循环计数器：（OV7670_FRAME_BYTES / 32u）=（160 x 120 x 2）/ 32 = 1200。
- 最后目的地地址调整：0，表示不调整。在VSYNC ISR中配置了新的目标地址。
- 在每个主循环后禁用DMA请求。每次都会在VSYNC ISR中启用该请求。

4.7 FlexRAM配置

使用FlexRAM功能，可以将RT1010的128 KB内部RAM配置为ITCM，DTCM或OCRAM。 默认情况下，128 KB RAM分为32 KB ITCM，32 KB DTCM和64 KB OCRAM。

另一方面，此应用程序中的一个完整帧占用160 x 120 x 2 = 38.4 KB。 可以将一帧缓冲区放入OCRAM。 但是，如果通过乒乓缓冲方式使用两个帧缓冲区，则默认的RAM分配将不适用。 以下代码是从startup_MIMXRT1011.s剪切而来的，它说明了如何为应用程序重新分配RAM。 之所以在启动程序集文件中实现此配置，是为了确保在重新分配之前不执行任何堆叠操作。 堆栈地

```
        LDR     R0, =0x400AC044 ;IOMUXC_GPR_GPR17 register
        LDR     R1, =0x000000D5
        STR     R1, [R0]
        LDR     R0, =0x400AC040 ;IOMUXC_GPR_GPR16 register
        LDR     R1, =0x00000007
        STR     R1, [R0]
```

址可以在重新分配期间进行更新。

128 KB RAM分为32 KB ITCM和96 KB OCRAM。 此外，必须更新链接描述文件以匹配新的RAM空间。 下面的代码显示了IAR链接程序脚本文件的更新部分。

```
define symbol m_data2_start           = 0x20200000;
define symbol m_data2_end             = 0x20217FFF;
define region DATA2_region = mem:[from m_data2_start to m_data2_end];
define region CSTACK_region = mem:[from m_data2_end-_size_cstack_+1 to m_data2_end];
place in DATA2_region                       { block RW };
place in DATA2_region                       { block ZI };
```

```
place in DATA2_region                        { last block HEAP };
place in DATA2_region                        { block NCACHE_VAR };
place in CSTACK_region                       { block CSTACK };
```

定义了一个名为DATA2_region的区域来表示96 KB OCRAM。 块RW，ZI，HEAP，NCACHE_VAR和CSTACK都放置在该区域

```
typedef uint16_t FrameBuffer_t[OV7670_FRAME_HEIGHT][OV7670_FRAME_WIDTH];
#pragma data_alignment = 32
static FrameBuffer_t g_FlexioCameraFrameBuffer[OV7670_FRAME_BUFFER_CNT] @"NonCacheable";
```

中。

帧缓冲区是在NCACHE_VAR块中定义的，其中的以下代码从flexion_ov7670.c中裁剪而来。

5 结论

本应用笔记介绍了如何使用FlexIO仿真i.MX RT1010处理器的并行摄像头接口。 该演示应用程序使用OV7670摄像机模块，该模块配置为输出QQVGA帧。 帧速率最高为30 fps。

通过此示例，您可以在应用程序中使用低成本RT1010处理器以及备用并行摄像头接口。

6 参考

- i.MX RT1010参考手册（文档IMXRT1010RM）
- 将FlexIO用于并行摄像机接口（文档AN5275）
- 使用FlexIO生成PWM（文档AN5209）
- 使用FlexIO驱动8080总线接口LCD模块（文档AN5313）

如何到达我们的主页：

nxp.com

网络支持：

nxp.com/support

有关更多信息，请访问：http://www.nxp.com

有关销售办事处的地址，请发送电子邮件至：salesaddresses@nxp.com

发布日期：01/2020

文档标识符：AN12686