SMART ARM-based Microcontrollers

AT12861: Image Sensor Interface in SAM V7/E7/S7 Devices

APPLICATION NOTE

Introduction

This application note explains Image Sensor Interface (ISI) in Atmel® | SMART SAM V71/V70/E70/S70 devices. It describes the steps to configure ISI to capture an image from an image sensor and display it in the LCD module. It also explains the steps to integrate a camera sensor with ISI using software package.

The examples included in this application note are available in SAM V71/V70/E70/S70 software package.

Features

The following features are covered in this application note:

- Image Sensor Interface in SAM S7/V7/E7 devices
- Preview Path and Codec path output formats
- ISI example implementation in software package
- Steps to follow to integrate new camera sensor
- JPEG library integration with software package

Table of Contents

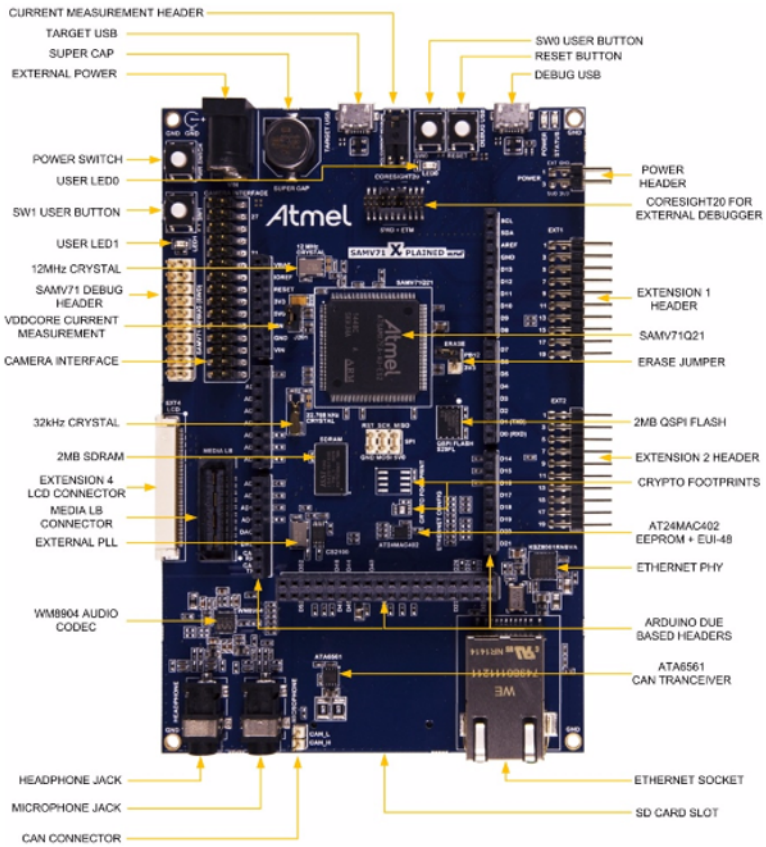
1. Hardware / Software Requisites

The following hardware and software environments are required to evaluate the examples. For more information, see the References section of this application note.

1.1. SAM V71 Xplained Ultra Evaluation Kit

The SAM V71 Xplained Ultra evaluation kit is ideal for evaluating and prototyping using the SAM V71, SAM V70, SAM S70, and SAM E70 ARM® Cortex®-M7 based microcontrollers. The Extension boards for the SAM V71 Xplained Ultra can be purchased separately. The ATSAMV71-XULT evaluation kit does not include extension boards. The detailed view of SAM V71 Xplained Ultra Evaluation Kit is provided in the following illustration.

Figure 1-1. SAM V71 Xplained Ultra Evaluation Kit

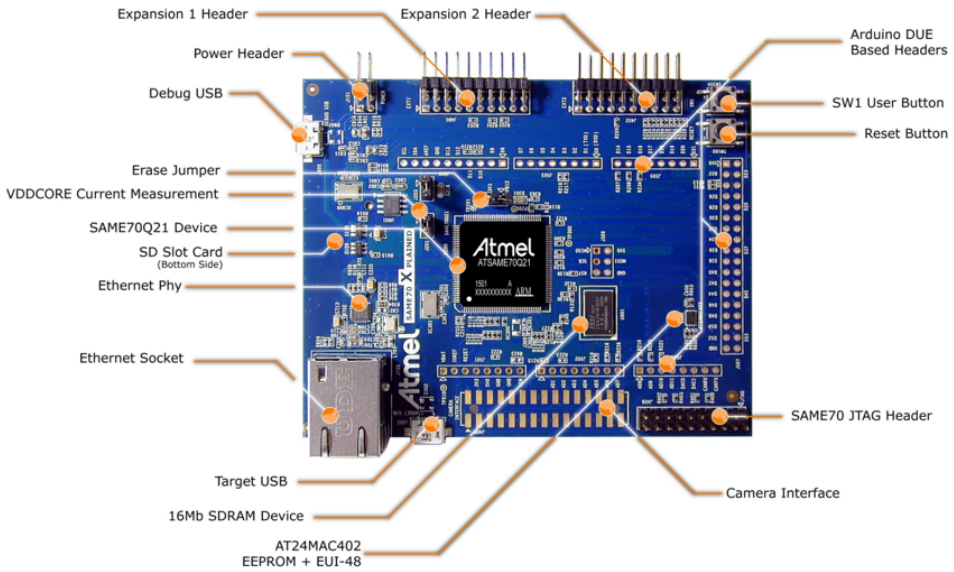


1.2. SAM E70 Xplained Evaluation Kit

The SAM E70 Xplained evaluation kit is ideal for evaluating and prototyping using SAM S70 and SAM E70 ARM Cortex-M7 based microcontrollers. Extension boards for the SAM E70 Xplained can be purchased separately.

The SAM E70 product is a superset of the S70 series. The SAM E70 Xplained evaluation kit can be used for evaluation. The Xplained Pro MCU series evaluation kits includes an on-board Embedded Debugger. No external tools are required to program or debug the ATSAME70Q21.

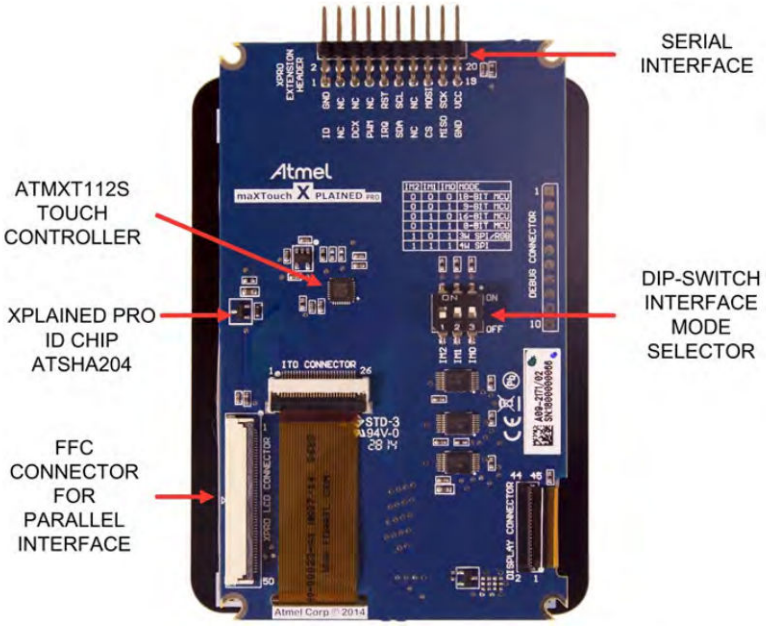
Figure 1-2. SAM E70 Xplained Evaluation Kit



1.3. Atmel maXTouch Xplained Pro

Atmel maXTouch® Xplained Pro is an extension board for the Xplained Pro platform with a 320x480 RGB LCD and a capacitive touch sensor with a maXTouch controller. The LCD can be controlled using different interfaces, including 3- and 4-wire SPI, and Parallel and RGB Parallel interface mode using the DIP-switch to select the interface. The maXTouch Xplained Pro kit connects to any Xplained Pro standard extension header on any Xplained Pro MCU board using the 20-pin header, but is limited to 3- and 4-wire SPI mode. Atmel maXTouch Xplained Pro also features a standard Xplained Pro LCD connector (FFC), which enables the use of the parallel interfaces. Both connections feature SPI interface for the LCD and I^2C for the maXTouch device. The ILI9488 controller is used to drive the LCD on this board.

Figure 1-3. maXTouch Xplained Pro



1.4. Camera Sensor Extension Boards

By default, the software package supports the following camera profiles:

- OV2640
- OV2643
- OV5640
- OV7740
- OV9740

Note: To purchase the camera module used in this example, contact the nearest Atmel Sales office. The contact details of Atmel Sales Office are available at http://www.atmel.com.

1.5. SAM V71/V70/E70/S70/ Software Package

The software package provides basic drivers, software services, and libraries for Atmel SAM V71, SAM V70, SAM E70, and SAM S70 Cortex-M7 based microcontrollers. It contains the source code, usage examples, documentation, and ready-to-use projects for Atmel Studio, GNU, IAR™ EWARM, and ARM Keil® MDK.

1.6. Atmel Studio Version 6.2 SP2 or Later

Atmel Studio is the integrated development platform (IDP) for developing and debugging AVR® and ARM Cortex-M based Atmel microcontroller applications.

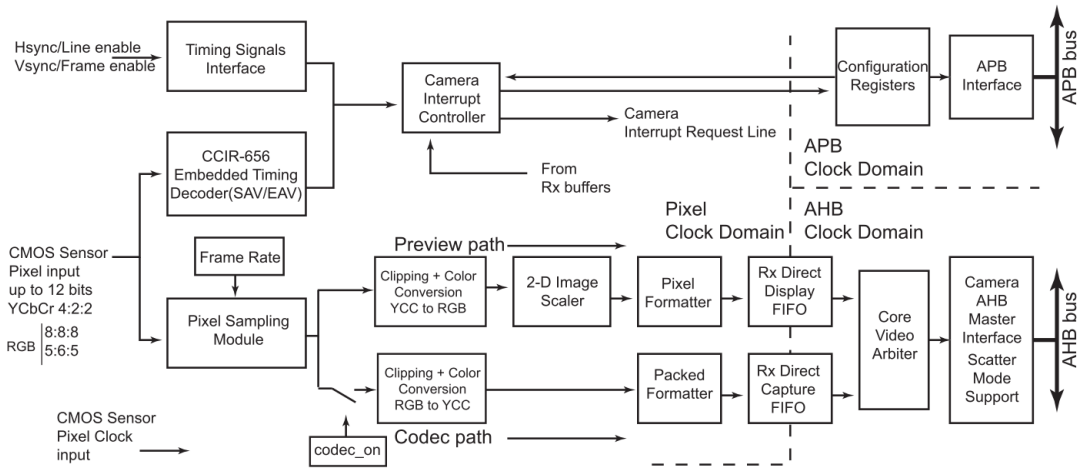
2. Image Sensor Interface (ISI)

The Image Sensor Interface (ISI) supports direct connection to the ITU-R BT. 601/656 8-bit mode compliant sensors and up to 12-bit grayscale sensors. The ISI can capture various data formats from image sensor. Before storing the data in the memory frame buffer using the DMA, it performs the data conversion, if required.

The ISI supports color CMOS image sensor and grayscale image sensors with a reduced set of functionalities. Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

Two separate processing paths are available; preview path and codec path. Each path has an internal FIFO to store the captured image sensor.

Figure 2-1. ISI Block Diagram



For detailed description of each block, refer to the respective device datasheet.

2.1. ISI Features

- ITU-R BT. 601/656 8-bit Mode External Interface Support
- Supports up to 12-bit Grayscale CMOS Sensors
- Support for ITU-R BT.656-4 SAV and EAV Synchronization
- Vertical and Horizontal Resolutions up to 2048 x 2048
- Preview Path up to 640 x 480 in RGB Mode
- Codec Path up to 2048 x 2048
- 16-byte FIFO on Codec Path
- 16-byte FIFO on Preview Path
- Support for Packed Data Formatting for YCbCr 4:2:2 Formats
- Preview Scaler to Generate Smaller Size image
- Programmable Frame Capture Rate
- VGA, QVGA, CIF, QCIF Formats Supported for LCD Preview
- Custom Formats with Horizontal and Vertical Preview Size as Multiples of 16

2.2. Data Timing

Two synchronization methods are available to capture the image sensor data. They are:

1. VSYNC/HSYNC Data Timing
2. SAV/EAV Data Timing

The VSYN/HSYNC Data Timing method uses the HSYN and VSYNC signal lines from the camera sensor. The SAV/EAV Data Timing uses timing reference signals embedded in the data lines and thereby reducing the use of I/O lines.

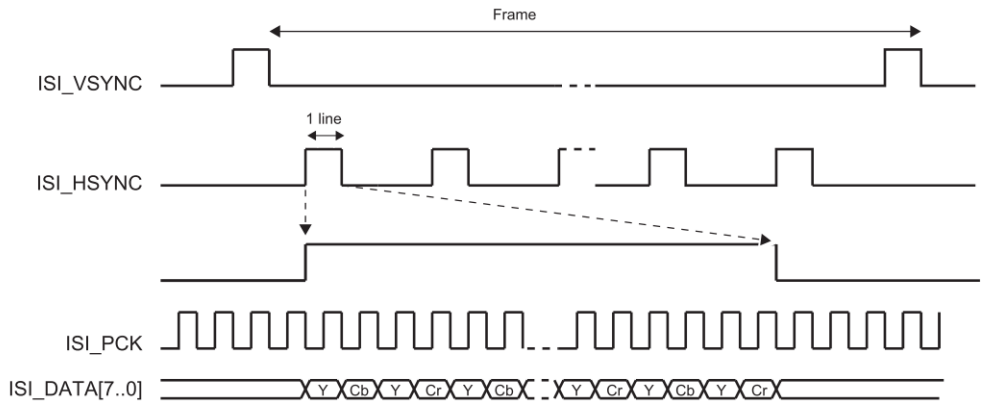
2.2.1. VSYNC/HSYNC Data Timing

VSYNC indicates the start of each frame. HSYNC indicates the start of a line. For example, in an image with a 640 x 320 resolution, HSYNC triggers for each line (i.e., 640) and captures 320 pixel data in a row at rising edge of pixel clock. VSYNC asserts for start of each frame (i.e., after 640 x 320 image is captured, VSYCN asserts to capture next frame of data). A delay can be added at the start of each line and each frame through SLD and SFD configuration.

- **SLD**: Start of Line Delay
  SLD pixel clock periods to wait before the beginning of a line.
- **SFD**: Start of Frame Delay
  SFD lines are skipped at the beginning of the frame.

In the VSYNC/HSYNC synchronization, the valid data is captured with the active edge of the pixel clock (ISI_PCK), after SFD lines of vertical blanking and SLD pixel clock periods delay programmed in the ISI_CR.

Figure 2-2. VSYNC/HSYNC Data Timing

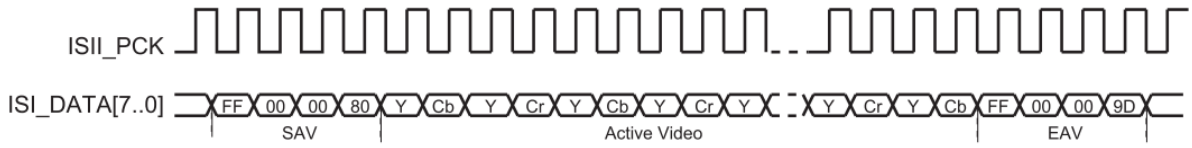


2.2.2. SAV/EAV Data Timing

The ITU-RBT.656-4 standard defines the functional timing for an 8-bit interface. There are two timing reference signals, one at the beginning of each video data block SAV (0xFF000080) and the other one at the end of each video data block EAV (0xFF00009D). Only the data sent between EAV and SAV is captured.

The horizontal blanking and vertical blanking are ignored. Use of the SAV and EAV synchronization eliminates the ISI_VSYNC and ISI_HSYNC signals from the interface, thereby reducing the pin count. To retrieve both frame and line synchronization properly, at least one line of vertical blanking is mandatory.

Figure 2-3. SAV/EAV Data Timing



2.3. Preview Path

When the preview DMA channel is configured and enabled, the preview path is activated and an 'RGB frame' is moved to memory. The preview path frame rate is configured using the FRATE field of the ISI_CFG1 register.

It captures YCrCb or YUV pixels and converts them to RBG 5:6:5 color space. It contains a data formatter that converts RGB 8:8:8 pixel to RGB 5:6:5 format compliant with the 16-bit format of the LCD controller.

For preview path, the frame buffer is always stored in RGB 5:6:5 in little endian format. The first pixel received is stored at the lowest address and the next pixel onwards are stored in increasing order.

2.3.1. Data Ordering

All sensors do not output the YCbCr or RGB components in the same order. The ISI allows the user to program the same component order similar to the sensor. Thereby reduces the software processing to restore the right format. The details about various possible combinations of sensor output format (in RGB and YCbCr modes) are provided in the datasheet.

2.3.2. Scaling, Decimation (Subsampling)

This module resizes the captured 8-bit sensor images to fit the format suitable for the LCD such as RGB 5:6:5. This module can only downscale the data and cannot upscale. The same ratio is applied for resizing both horizontal and vertical resolution, then a fractional decimation algorithm is applied. Example calculation is as follows:

If the input image is captured in 1280 × 1024 resolution and the output format must be stored in 640 × 480, the calculations are as follows:

H_{ratio} is calculated as 2 (1280/640 = 2)

Similarly, V_{ratio} = 1024/480 = 2.1333

The decimation factor is 2. So the decimation value is 32, as it is a multiple of 16 (32/16 =2).

The decimation factor is configured in "ISI Preview Decimation Factor Register" of ISI.

2.4. Codec Path

When the codec DMA channel is configured and enabled, the codec path is activated and a 'YCbCr 4:2:2 frame' is captured as soon as the ISI_CDC bit of the ISI Control Register (ISI_CR) is set. If the RGB input stream is selected, this module converts RGB to YCrCb 4:2:2 color space. Depending on user selection, this can be bypassed so that input YCrCb stream is directly connected to the format converter module.

The codec data path is mainly used to capture a single frame and encode it later in a user-space application. The data available in frame buffer of codec path will always be stored in the memory format as shown.

Figure 2-4. Codec Path - Data Storage Format

MSB			LSB
Y(n+1)	Cr(n)	Y(n)	Cb(n)

When the FULL bit of the ISI_CFG1 register is set, both preview DMA channel and codec DMA channel can operate simultaneously. When a zero is written to the FULL bit of the ISI_CFG1 register, a hardware scheduler checks the FRATE field. If the value is zero, a preview frame is skipped and a codec frame is moved to memory, instead. If its value is non- zero, at least one free frame slot is available. The scheduler postpones the codec frame to that free available frame slot. The data stream may be sent on both preview path and codec path if the value of bit ISI_CDC in the ISI_CR is one.

To optimize the bandwidth, the codec path should be enabled only when a capture is required.

2.5. Grayscale Mode

In the grayscale mode, input data stream is stored in memory without any processing. The 12-bit data, which represents the grayscale level for the pixel is stored in memory one or two pixels per word, depending on the GS_MODE bit in the ISI_CFG2 register. In grayscale mode, the input data does not perform any conversion. It is not compatible for LCD display which is RGB 5:6:5 format. The software must perform the necessary conversion. The codec data path should be used to capture the image data in grayscale mode. When grayscale mode is enabled, all data conversion in the path are skipped and the RAW image data is stored in the frame buffer configured.

2.5.1. 12-bit Gray Scale Mode

In the 12-bit gray scale mode, ISI_DATA[11:0] is the physical interface to the ISI. These bits are sampled and written to frame buffer. When 12-bit grayscale mode is enabled, the two memory formats supported are:

Figure 2-5. ISI_CFG2.GS_MODE = 0 : Two Pixels per Word

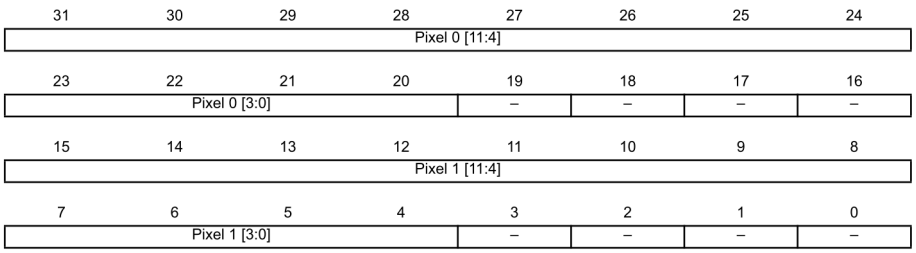
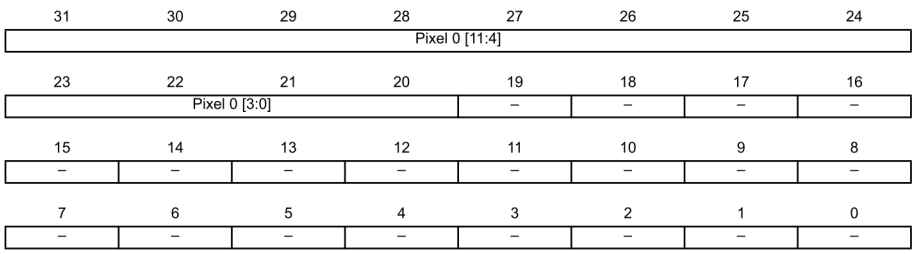


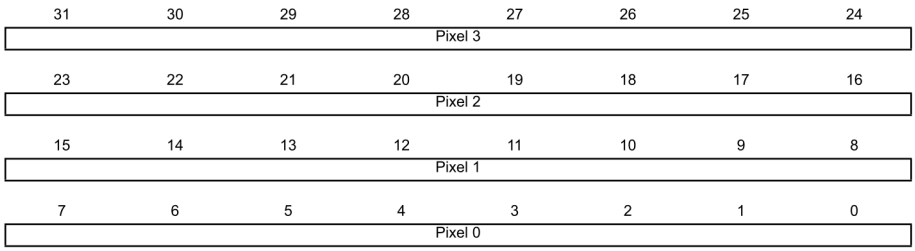
Figure 2-6. ISI_CFG2.GS_MODE = 1 : One Pixel per Word



2.5.2. 8-bit Grayscale Mode

For an 8-bit grayscale mode, ISI_DATA[7:0] on the 12-bit data bus is the physical interface to the ISI. These bits are sampled and written to the memory.

Figure 2-7. 8-bit Grayscale Mode Data Storage Format



2.6. FIFO and DMA Features

The FIFO feature is available for each codec path and preview path. These asynchronous buffers are used to safely transfer formatted pixels from the pixel clock domain to the AHB clock domain. A video arbiter is used to manage FIFO thresholds and triggers a relevant DMA request through the AHB master interface. Thus, depending on the FIFO state, a specified length of burst is asserted. The AHB master interface supports Scatter DMA mode through linked list operation. This mode of operation improves the flexibility of the image buffer location and allows the user to allocate two or more frame buffers.

The destination frame buffers are defined by a series of Frame Buffer Descriptors (FBD). It controls the transfer of one entire frame and optionally loads another frame buffer address in case of linked list operation.

The frame buffer descriptor consist of three words:

- *First word* defines the address of current frame buffer
- *Second word* controls the information
- *Third word* defines the next frame address

3. Interfacing Camera Sensor with ISI

This section provides details about interfacing a camera sensor with ISI peripheral. The following points must be taken care of while interfacing the camera sensor:

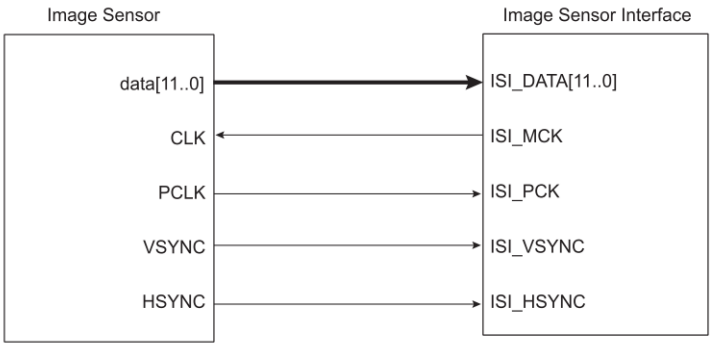
- Ensure that the camera sensor is compatible with the ISI
- Decide the input format
- Camera control settings
- Decide the memory allocation for frame buffer
- Ensure that the software package and ISI configuration

3.1. Check the Sensor Compatibility

Ensure that the sensor is compatible to interface with the ISI peripheral before starting the integration.

The selected camera sensor must have a parallel interface to communicate with the ISI peripheral. This includes the pixel data lines, HSYNC/VSYNC, and clock lines. A typical connection between ISI and image sensor is illustrated in the following diagram.

Figure 3-1. ISI Connection Example



3.2. Check for the Input Format and Desired Frame Rate

Check the required data format which is compatible with ISI. ISI on SAMV/S/E7 series supports RGB or YUV data format. It does not support RAW Bayer/JPEG formats in the hardware. However, while using grayscale mode in ISI, RAW image data can be captured to frame buffer and processing must be performed in the software.

ISI peripheral can support resolution up to 640 x 480 in preview path and up to 2048 x 2048 in codec path.

The camera sensor captures an image at a particular frame rate and data is available on data lines. For continuous capturing of images such as video streaming, the frame rate plays an important role. The MCU must be able to capture the data appropriately without missing any part of the data.

The frame rate can be calculated as follows:

$$\frac{(Maximum\ pixel\ \text{clok})}{(No.of\ pixels\ per\ frame * no.of\ \text{bytes per pixel})}$$

For example: Consider the pixel clock is 24MHz, the data line over ISI is 8-bits.

For capturing a VGA image (640 x 480) in RGB 5:6:5 format, the number of pixels per frame is 307200 (640 * 480).

If the number of bytes per pixel (RGB 5:6:5 format) is 2:

Frame rate = (24M) / (307200 * 2) ~ 39 fps

3.3. Camera Control Settings

The camera sensor is generally controlled by using the I^2C interface with which we can perform the read/write operations on the control registers. The registers must be configured as per requirement. The details about each register is available in the respective camera module datasheet.

In software package, the I^2C communication with sensor is performed at `…\libraries\libboard\source\image_sensor_inf.c`.

To integrate a new camera sensor, a file containing the register setting for various formats should be filled in an array. The following example settings for camera sensors are available in the software package at `…\libraries\libboard\source\` :

- `mt9v022_config.c`
- `ov2640_config.c`
- `ov2643_config.c`
- `ov5640_config.c`
- `ov7740_config.c`
- `ov9740_config.c`

3.4. Frame Buffer Allocation in the Application

Memory allocation plays a vital role in storing captured image data in the frame buffer. The higher the resolution of the image, the higher the frame buffer size. For example, VGA (640 x 480) image is captured in RGB 5:6:5 format using preview path. Frame buffer size of ~614400 bytes (i.e. 640 * 480 * 2) is required; where 2 is the number of bytes per pixel.

As internal SRAM will not be sufficient to store the data, external memory can be used. The example application in the software package captures RGB image in VGA format and the frame buffer is allocated to external SDRAM memory.

3.5. Integration of Camera Sensor Configuration with Software Package

When the new camera sensor configuration file is ready, it can be added to the application in the software package. In case of Atmel Studio, the file can be copied to the respective location in the software package. To add the project, right click on the folder > **Add** > **Existing Item** > Select File > Use **Add as a Link** option.

For an ISI peripheral, perform the following initialization steps:

1. The respective I/O lines multiplexed should be enabled for ISI peripheral.
2. Initialize TWI lines which are used to control the camera sensor configuration.
3. Initialize the required sensor.
4. Initialize ISI and configure for preview or codec path based on the requirement.

5. Configure Frame buffer for respective paths.
6. Inside ISI handler, the captured image can be processed by the application.

The following topics explains these steps with reference to the example implementation.

4. Example implementation

This upcoming topics cover the description about the example implementation available in the software package.

4.1. ISI Example Implementation in the Software Package

This example is available in the software package in the path "\examples\Atmel \SAMV71_Xplained_Ultra\examples\isi\".

The image is captured in preview mode (RGB 5:6:5 format) and displayed on the LCD. The maXTouch Xplained Pro kit is used to interface the LCD module used in this example. The external SDRAM available on the board is used to store the frame buffer.

The Initialization step involves:

1. Initialize and configure External SDRAM.

```
BOARD_ConfigureSdram();
```

2. Initialize TWI to control the image sensor configuration.

```
/** * \brief TWI initialization. */
 static void _twiInit(void)
{
    /* Configure TWI pins. */
    PIO_Configure(pinsTWI, PIO_LISTSIZE(pinsTWI));
    /* Enable TWI peripheral clock */
    PMC_EnablePeripheral(BOARD_ID_TWI_ISI);
    /* Configure TWI */
    TWI_ConfigureMaster(BOARD_BASE_TWI_ISI, TWCK, BOARD_MCK);
    TWID_Initialize(&twid, BOARD_BASE_TWI_ISI);
    /* Configure TWI interrupts */
    NVIC_ClearPendingIRQ(TWIHS0_IRQn);
    NVIC_EnableIRQ(TWIHS0_IRQn);
}
```

3. Initialize the sensor for the required format.

```
sensor_setup(&twid, sensorsProfile, VGA);
```

4. LCD is configured with the frame buffer as source to update the window.

```
static void  _lcd_Configure(void)
{
    //……..other code…..//
   LCDD_Initialize(LCD_MODE, &lcdEbiDma, lcdDisplayMode);
   LCDD_SetCavasBuffer((void*)gIsiBuffer, BOARD_LCD_HEIGHT * BOARD_LCD_WIDTH * 2);
   //……..other code…..//
}
```

5. Initialize the ISI frame buffer.
   The frame buffer is configured for preview path, pointing to SDRAM address as current frame buffer, with preview channel descriptor fetch enabled. The next descriptor points to the first descriptor again.

```
static void _isi_AllocateFBD(void)
{
     uint32_t i;
     for(i = 0; i < ISI_MAX_PREV_BUFFER; i++) {
         preBufDescList[i].Current = (uint32_t)ISI_BASE;
         preBufDescList[i].Control = ISI_DMA_P_CTRL_P_FETCH;
         preBufDescList[i].Next    = (uint32_t)&preBufDescList[0];
     }
}
```

6. Initialize ISI for preview path and enable transfer complete interrupt. In this example, preview path captures the image in YUV format. The processed data stored in the frame buffer is always in RGB 5:6:5 format making it compatible with LCD.

```
static void _isiInit(void)
{
    ISI_Y2R y2r;
    /* Enable ISI peripheral clock */
     PMC_EnablePeripheral(ID_ISI);

    /* Set up Frame Buffer Descriptors(FBD) for preview path. */
    _isi_AllocateFBD();

    /* Reset ISI peripheral */
     ISI_Reset();

    /* Set the windows blank */
    ISI_SetBlank(0, 0);

    /* Set vertical and horizontal Size of the Image Sensor for preview path*/
    ISI_SetSensorSize(wImageWidth, wImageHeight);

    /* Set preview size to fit LCD size*/
    if (lcdDisplayMode == LCD_REVERSE_MODE) {
        /* LCD Height = BOARD_LCD_WIDTH  LCD Width =  BOARD_LCD_HEIGHT */
        wDisplayHeight = wImageHeight * 1000 / (wImageWidth * 1000 /
BOARD_LCD_HEIGHT);
        wDisplayWidth = BOARD_LCD_HEIGHT;
        ISI_setPreviewSize(wDisplayWidth, wDisplayHeight);
     } else {
         /* LCD Height = BOARD_LCD_HEIGHT  LCD Width =  BOARD_LCD_WIDTH */
        wDisplayHeight = wImageHeight * 1000 / (wImageWidth * 1000 /
BOARD_LCD_WIDTH);
        wDisplayWidth = BOARD_LCD_WIDTH;
        ISI_setPreviewSize(BOARD_LCD_WIDTH, wDisplayHeight );
      }

    /*  Set data stream in YUV format.*/
     ISI_setInputStream(YUV_INPUT);

    /* Defines YCrCb swap format.*/
     ISI_YCrCbFormat(ISI_CFG2_YCC_SWAP_MODE2);

     /* calculate scaler factor automatically. */
     ISI_calcScalerFactor();

     /* Configure DMA for preview path. */
     ISI_setDmaInPreviewPath((uint32_t)&preBufDescList[0], ISI_DMA_P_CTRL_P_FETCH,
                  (uint32_t)ISI_BASE);
     /* Set Matrix color space for YUV to RBG convert */
     y2r.C0 = 0x95;
     y2r.C1 = 0xFF;
     y2r.C2 = 0x68;
     y2r.C3 = 0x32;
     y2r.C4 = 0xCC;
     y2r.Yoff = 1;
     y2r.Croff = 1;
     y2r.Cboff = 1;

     ISI_SetMatrix4Yuv2Rgb(&y2r);
     NVIC_ClearPendingIRQ(ISI_IRQn);
     NVIC_SetPriority(ISI_IRQn, 7);
     NVIC_EnableIRQ(ISI_IRQn);
}
```

7. Initialize ISI peripheral and enable interrupt after the frame buffer is filled and the LCD window is updated with the frame buffer value. As preview path image is already RGB 5:6:5 format, the value can be updated to the windows without performing any post processing in the software.

```
/**
  * \brief ISI interrupt handler.
*/
void ISI_Handler(void)
```

```
{
    uint32_t status,imr;
    status = ISI->ISI_SR;
    imr = ISI->ISI_IMR;
    if ((status & ISI_SR_PXFR_DONE) && (imr & ISI_IMR_PXFR_DONE)) {
        ISI_DmaChannelDisable(ISI_DMA_CHDR_P_CH_DIS);
        ISI_DisableInterrupt(ISI_IDR_PXFR_DONE);
        LCDD_UpdateWindow();
        ISI_DmaChannelEnable(ISI_DMA_CHER_P_CH_EN);
        ISI_EnableInterrupt(ISI_IER_PXFR_DONE);
     }
}
```

4.2. Example Test Setup

1. Connect the maXTouch Xplained kit with the SAMV7 Xplained LCD connecter using a flat cable as shown in the figure below. Switch IM0 = IM1 = OFF and IM2 = ON to enable LCD in parallel interface mode using EBI.

Figure 4-1. Example Setup

2. Connect the camera sensor module to the “Camera Connector” header on the board. The software package supports the following camera profiles.
   - OV2640
   - OV2643
   - OV5640
   - OV7740
   - OV9740
3. Download the application to the SAM V7 board.
4. The camera starts capturing image and it is displayed on the LCD.

Figure 4-2. ISI Example Result

4.3. ISI Grayscale Example Implementation

This example implementation is available in the software package in the path `\examples\Atmel\SAMV71_Xplained_Ultra\examples\isi_gray\`.

The camera used in this example is monochrome sensor which outputs only monochrome data in RGB format. In this example, the preview path is used to capture the sensor data. It is configured in RGB565 mode. Preview path always outputs RGB5:6:5 processed data. As the output format and configured input format (RGB 5:6:5) are the same in the preview path, it bypasses the conversion and the input sensor data is stored in the frame buffer. The monochrome camera sensor contains only luminance information. So the software conversion is performed in the ISI handler to RGB5:6:5 format to make it compatible with the LCD controller. The color conversion in preview path is bypassed as the data processed on Y information alone is not sufficient for proper LCD display.

```
/** * \brief ISI interrupt handler. */
void ISI_Handler(void)
{
    uint32_t status,imr;
    status = ISI->ISI_SR;
    imr = ISI->ISI_IMR;
    if ((status & ISI_SR_PXFR_DONE) && (imr & ISI_IMR_PXFR_DONE)) {
        if (ISI->ISI_DMA_P_ADDR != (uint32_t)ISI_BASE + (wImageWidth *
wImageHeight ))
            return;
        ISI_DisableInterrupt(ISI_IDR_PXFR_DONE);
        ISI_DmaChannelDisable(ISI_DMA_CHDR_P_CH_DIS);
        y2rgb565();
        LCDD_UpdateWindow();

ISI_setDmaInPreviewPath((uint32_t)&preBufDescList,
            ISI_DMA_P_CTRL_P_FETCH, (uint32_t)ISI_BASE);
        ISI_DmaChannelEnable(ISI_DMA_CHER_P_CH_EN);
        ISI_EnableInterrupt(ISI_IER_PXFR_DONE);
       }
}
```

In ISI handler, it is processed to RGB 5:6:5 format to make it compatible with the LCD interface.

```
void ISI_Handler(void)
{
    //…other code…//
    y2rgb565();
    LCDD_UpdateWindow();
    //… other code….//
}
```

4.4. JPEG Compression

JPEG stands for Joint Photographic Experts Group. It is designed for compressing either full-color or grayscale images of natural, real-world scenes. It is more suitable for photographs, naturalistic artwork, and similar materials.

It is a lossy compression method which discards a lot of details from the original information. So the original image cannot be extracted from it. The human eye can perceive less accurately the small color changes than small changes in brightness. JPEG is designed to exploit this limitation that it compresses images considering that it will be looked by humans and discards other information.

4.5. IJG JPEG Compression Library

IJG is an informal group that writes and distributes a widely used free library for JPEG image compression. Compression and decompression library ported for Atmel devices are available in the Atmel software package. This library can be integrated with the application, if required. It is available in the path `examples\Atmel\SAMV71_Xplained_Ultra\libraries\libjpeg`.

To encode an image into JPEG, the following functions are used:

```
    //Source buffer pointer and length which points to the starting of the buffer that is to
be compressed.
    JpegData_SetSource()
    //Destination buffer pointer and length, which points to the starting of the buffer to
store the results.
    JpegData_SetDestination()
    JpegData_SetDimensions() // Set Image dimensions
    JpegData_SetParameters() // To set image quality, input format and compression method
    ijg_compress() // Start JPEG encoding
```

To decode JPEG, the following functions are used:

```
    // Source buffer pointer and length, which points to the starting of the buffer that is
to be decompressed.
    JpegData_SetSource()
    // Destination buffer pointer and length, which points to the starting of the buffer to
store the results.
    JpegData_SetDestination()
    JpegData_SetDimensions() // Set Image dimensions
    JpegData_SetParameters() // To set image quality, input format and compression
    methodijg_decompress() // Start JPEG decoding
```

The IJG library supports several compression/decompression methods including IFAST and ISLOW. While IFAST is faster, it is a less accurate method. ISLOW is slower and more accurate method. One of these methods can be selected based on the requirement. For detailed information about IJG library, refer to the website http://www.ijg.org/.

5. References

1. Atmel web page for the SAM V71 Xplained Ultra Evaluation Kit: http://www.atmel.com/tools/ATSAMV71-XULT.aspx
2. Atmel web page for the SAM E70 Xplained Evaluation Kit: http://www.atmel.com/tools/ATSAME70-XPLD.aspx
3. maXTouch Xplained Pro : http://www.atmel.com/tools/ATMXT-XPRO.aspx
4. Software Package : http://www.atmel.com/tools/samv71-samv70-same70-sams70-software-package.aspx
5. Atmel Studio : http://www.atmel.com/tools/atmelstudio.aspx?tab=overview
6. AT12863: Interfacing LCD Controllers for SAM S70/E70/V70 : http://www.atmel.com/Images/Atmel-42646-SAM-S70-E70-V70-Interfacing-LCD-Controllers_ApplicationNote_AT12863.pdf
7. SAM V7 Device Datasheet : http://www.atmel.com/products/microcontrollers/arm/sam-v-mcus.aspx?tab=documents
8. SAM S7 MCUs: http://www.atmel.com/products/microcontrollers/arm/sam-s.aspx?tab=documents
9. SAM E7 MCUs : http://www.atmel.com/products/microcontrollers/arm/sam-e.aspx

6. Revision History

Doc Rev.	Date	Comments
42702A	04/2016	Initial document release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**