NXP Semiconductors
User's Guide

Document identifier: IMXRT500SDMAAPIUG
Rev. 0, 02/2021

i.MX RT500 SmartDMA API Documentation

Contents

Chapter 1
Overview

The purpose of the SmartDMA is to perform graphics handling to offload the work from the Arm processor in the system.

Chapter 2
API components and conventions

The SmartDMA API uses a naming convention scheme where definitions are preceded by “smartdma”.

2.1 Structures used in API

The structures in the driver are as follows:

smartdma_flexio_mculcd_param_t Structure

Members:

- **p_buffer**
- **buffersize**
- **smartdma_stack**

smartdma_flexio_onelane_mculcd_param_t Structure

Members:

- **p_buffer**
- **buffersize**
- **offset**
- **smartdma_stack**

smartdma_dsi_param_t Structure

Members:

- **p_buffer**
- **buffersize**
- **smartdma_stack**

smartdma_rgb565_rgb888_param_t Structure

Members:

- **inBuf**
- **outBuf**
- **buffersize**
- **smartdma_stack**

Chapter 3
SmartDMA API

This section describes the API functions to initialize the SmartDMA and control.

3.1 Initialization and control

SMARTDMA_Init

Description:

This function initializes the SmartDMA.

Syntax:

```
void SMARTDMA_Init(
uint32_t apiMemAddr,
const void *firmware,
uint32_t firmwareSizeByte);
```

Parameters:

- apiMemAddr - The address to which the firmware is copied.
- firmware - The firmware to use.
- firmwareSizeByte - The size of the firmware.

> **NOTE**
>
> Do not use this function. It is superseded by GPIO_PinWrite. Reference SMARTDMA_InitWithoutFirmware and SMARTDMA_InstallFirmware.

SMARTDMA_InitWithoutFirmware

Description:

This function initializes the SmartDMA. This function is similar to SMARTDMA_Init, but this function does not install the firmware. The firmware can be installed using SMARTDMA_InstallFirmware.

Syntax:

```
void SMARTDMA_InitWithoutFirmware (
void );
SMARTDMA_InstallFirmware
```

Description:

This function installs the firmware used for display based on the memory address and the specified display size.

Syntax:

```
void SMARTDMA_InstallFirmware (
uint32_t apiMemAddr,
const void *firmware,
uint32_t firmwareSizeByte);
```

Parameters:

- apiMemAddr - The address to which the firmware is copied.
- firmware - The firmware to use.

- firmwareSizeByte - The size of the firmware.

NOTE

Call this function only when the SmartDMA is not busy.

SMARTDMA_InstallCallback

Description:

This function installs the complete callback function.

Syntax:

```
void SMARTDMA_InstallCallback (
smartdma_callback_t callback,
void *param);
```

Parameters:

- callback - The callback is called when the SmartDMA program finishes.
- param - The parameter for the callback.
- firmwareSizeByte - The size of the firmware.

NOTE

Call this function only when the SmartDMA is not busy.

SMARTDMA_Boot

Description:

This function boots the SmartDMA to run the program.

Syntax:

```
void SMARTDMA_Boot(
uint32_t apiIndex,
void *pParam,
uint8_t mask);
```

Parameters:

- apiIndex - The index of the API to call.
- pParam - The pointer to the parameter.
- mask - The value is set to SMARTDMA_ARM2SMARTDMA[0:1].

NOTE

Call this function only when SmartDMA is not busy.

SMARTDMA_ Deinit

Description:

This function deinitializes the SmartDMA.

Syntax:

```
void SMARTDMA_Deinit(
void);
```

SMARTDMA_ Reset

Description:

This function resets the SMARTDMA.

Syntax:

```
void SMARTDMA_Reset(
void);
```

SMARTDMA_ HandleIRQ

Description:

This is the SmartDMA IRQ.

Syntax:

```
void SMARTDMA_HandleIRQ(
void);
```

Chapter 4
FlexIO MCULCD SmartDMA API

FLEXIO_MCULCD_TransferCreateHandleSMARTDMA

Description:

This function initializes the FlexIO MCULCD master SmartDMA handle. This function initializes the FlexIO MCULCD master SMARTDMA handle which can be used for other FlexIO MCULCD transactional APIs. For a specified FlexIO MCULCD instance, call this API once to get the initialized handle.

Syntax:

```
status_t FLEXIO_MCULCD_TransferCreateHandleSMARTDMA(
FLEXIO_MCULCD_Type *base,
 flexio_mculcd_smartdma_handle_t *handle,
 const flexio_mculcd_smartdma_config_t *config,
 flexio_mculcd_smartdma_transfer_callback_t callback,
 void *userData);
```

Parameters:

- base - The pointer to the FLEXIO_MCULCD_Type structure.
- handle - The pointer to the flexio_mculcd_smartdma_handle_t structure to store the transfer state.
- config - The pointer to the configuration.
- callback - The MCULCD transfer-complete callback; NULL means no callback.
- userData - The callback function parameter.
- kStatus_Success - The handle is successfully created.

FLEXIO_MCULCD_TransferSMARTDMA

Description:

This function performs a non-blocking FlexIO MCULCD transfer using SmartDMA. This function returns immediately after the transfer initiates. Use the callback function to check whether the transfer is completed.

Syntax:

```
status_t FLEXIO_MCULCD_TransferSMARTDMA(
FLEXIO_MCULCD_Type *base,
 flexio_mculcd_smartdma_handle_t *handle,
 flexio_mculcd_transfer_t *xfer);
```

Parameters:

- base - The pointer to the FLEXIO_MCULCD_Type structure.
- handle - The pointer to the flexio_mculcd_smartdma_handle_t structure to store the transfer state.
- xfer - The pointer to the FlexIO MCULCD transfer structure.

FLEXIO_MCULCD_TransferAbortSMARTDMA

Description:

This function gets the remaining bytes for the FlexIO MCULCD SmartDMA transfer.

Syntax:

```
status_t FLEXIO_MCULCD_TransferGetCountSMARTDMA(
FLEXIO_MCULCD_Type *base,
 flexio_mculcd_smartdma_handle_t *handle,
 size_t *count);
```

Parameters:

- base - The pointer to the FLEXIO_MCULCD_Type structure.
- handle - The FlexIO MCULCD SMARTDMA handle pointer.
- count - The number of counts transferred by the SmartDMA transaction so far.

FLEXIO_MCULCD_TransferAbortSMARTDMA

Description:

This function aborts a FlexIO MCULCD transfer using SmartDMA.

Syntax:

```
void FLEXIO_MCULCD_TransferAbortSMARTDMA(
FLEXIO_MCULCD_Type *base,
flexio_mculcd_smartdma_handle_t *handle);
```

Parameters:

- base - The pointer to the FLEXIO_MCULCD_Type structure.
- handle - The FlexIO MCULCD SmartDMA handle pointer.

Chapter 5
MIPI DSI SmartDMA API

DSI_TransferCreateHandleSMARTDMA

Description:

This function creates the MIPI DSI SmartDMA handle.

Syntax:

```
status_t DSI_TransferCreateHandleSMARTDMA(
MIPI_DSI_HOST_Type *base,
 dsi_smartdma_handle_t *handle,
 dsi_smartdma_callback_t callback,
 void *userData);
```

Parameters:

- base - The MIPI DSI host peripheral base address.
- handle - The handle pointer.
- callback - The callback function.
- userData - The user data.

DSI_TransferWriteMemorySMARTDMA

Description:

This function writes to the display controller video memory using SmartDMA. It performs the data transfer using SmartDMA. When the transfer finishes, the upper layer can be informed through a callback function.

Syntax:

```
status_t DSI_ DSI_TransferWriteMemorySMARTDMA (
MIPI_DSI_HOST_Type *base,
 dsi_smartdma_handle_t *handle,
 dsi_smartdma_write_mem_transfer_t *xfer);
```

Parameters:

- base - The MIPI DSI host peripheral base address.
- handle - The pointer to the dsi_smartdma_handle_t structure, which stores the transfer state.
- xfer - The pointer to the transfer structure.

DSI_TransferAbortSMARTDMA

Description:

This function aborts the current APB data transfer.

Syntax:

```
void DSI_TransferAbortSMARTDMA(
MIPI_DSI_HOST_Type *base,
dsi_smartdma_handle_t *handle);
```

Parameters:

- base - The MIPI DSI host peripheral base address.

- handle - The pointer to the dsi_smartdma_handle_t structure, which stores the transfer state.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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: 02/2021

Document identifier: IMXRT500SDMAAPIUG

arm