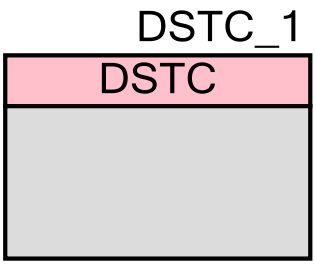
Descriptor System Data Transfer (PDL_DSTC) 1.0



Features

- Up to 64 transfer channels
- Software start
- Hardware start
- Supports a 32-bit address space

General Description

The Peripheral Driver Library (PDL) Descriptor System Data Transfer Controller (PDL_DSTC) component is a function block that can transfer data at high speed, bypassing the CPU. One set of transfer control details (basic transfer settings, number of transfers, transfer source address, transfer destination address) is specified in one DSTC. The DSTC can build up to 64 transfer channels in the FM0+ family devices.

The data transfer operation can be started by one of the following three methods:

- Direct start by the CPU (software start)
- Start by an interrupt signal from a peripheral device (hardware start)
- Chain Start function

This component uses firmware drivers from the PDL_DSTC module, which is automatically added to your project after a successful build.

When to Use a PDL_DSTC Component

Use the DSTC component when you need transfer data at high speed bypassing the CPU.

Quick Start

1. Drag a PDL_DSTC component from the Component Catalog FMx/System/ folder onto your schematic. The placed instance takes the name DSTC_1.
2. Double-click to open the component's Configure dialog.
3. On the **Basic** tab, set the following parameters:
    - enable or disable stop of the transmission when error occurred

- enable or disable Read Skip Buffer
- priority of the transmission

4. On the **Interrupts** tab, initialize needed interrupts and their callback functions.
5. Build the project to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer, and generate configuration data for the DSTC_1 instance.
6. In the *main.c* file, initialize the peripheral and start the application.

```
stc_dstc_des0123_t descriptor;/* Initialize the descriptor with
data/peripheral pointers used in the transmission */
stc_dstc_config_t dstc_config =
{
    0,
    DSTC_1_Config,
    /* here balow put callback function pointers e.g.
    pfnDstcAdc0PrioCallback,
    …,
    pfnDstcWcCallback */
};

dstc_config.u32Destp = (uint32_t)&descriptor;/* Set the descriptor address
in the configuration structure */

Dstc_Init(&DSTC_1_Config);
Dstc_SwTrigger(0u);   /* Start SW transfer with DES + 0 offset */
```

7. Build and program the device.

Component Parameters

The PDL_DSTC component Configure dialog allows you to edit the configuration parameters for the component instance.

Basic Tab

This tab contains the component parameters used in the general peripheral initialization settings.

Parameter Name	Description
bErrorStopEnable	Enable Error Stop
bReadSkipBufferDisable	Disable the Read Skip Buffer
enSwTransferPriority	Software transfer priority

Interrupts Tab

This tab contains the Interrupts configuration settings.

Parameter Name	Description
bTouchNvic	Install interrupts in NVIC
bErInterruptEnable	Enable error interrupt
bSwInterruptEnable	Enable software interrupt
pfnErrorCallback	Error status callback
pfnNotifySwCallback	Notification SW Callback Function Pointer

Component Usage

After a successful build, firmware drivers from the PDL_DSTC module are added to your project in the pdl/drivers/dstc folder. Pass the generated data structures to the associated PDL functions in your application initialization code to configure the peripheral.

Generated Data

The PDL_DSTC component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the component (e.g. *DSTC_1_config.c*). Each variable is also prefixed with the instance name of the component.

Data Structure Type	Name	Description
stc_dstc_config_t	DSTC_1_Config	Configuration structure.

Once the component is initialized, the application code should use the peripheral functions provided in the referenced PDL files. Refer to the PDL documentation for the list of provided API functions. To access this document, right-click on the component symbol on the schematic and choose "**Open API Documentation…**" in the drop-down menu.

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the **Built-In** tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Support

If the PDL_DSTC component is specified to trigger interrupts, it will generate the callback function declaration that will be called from the DSTC ISR. The user is then required to provide the actual callback code. If a null string is provided the struct is populated with zeroes and the callback declaration is not generated. In that case it is the user's responsibility to modify the struct in firmware.

The component generates the following function declarations.

Function Callback	Description
DSTC_1_NotifySwCallback	Notification SW callback function. Note: this generates a declaration only - USER must implement the function.
DSTC_1_ErrorCallback	Error callback function. Note: this generates a declaration only - USER must implement the function.

Code Examples and Application Notes

There are numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

Cypress also provides a number of application notes describing how FMx devices can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes.

Resources

The PDL_DSTC component uses the DSTC (Descriptor System data Transfer Controller) peripheral block.

References

- FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers Peripheral Manuals
- Cypress FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0	Initial Version	