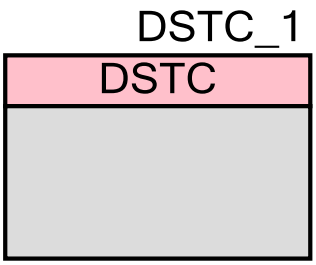
Descriptor System Data Transfer (PDL_DSTC) 1.0



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	

© Cypress Semiconductor Corporation, 2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.