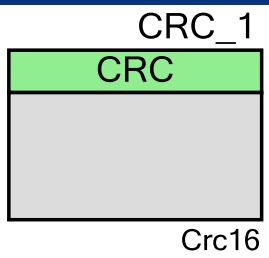
Cyclic Redundancy Check (PDL_CRC)

1.0



Features

- Enables the calculation in:
  - CCITT CRC16 Generator Polynomial: 0x1021
  - IEEE-802.3 CRC32 Generator Polynomial: 0x04C11DB7

General Description

The Peripheral Driver Library CRC (Cyclic Redundancy Check) component is an error detection system. The CRC code is a remainder after an input data string is divided by the pre-defined generator polynomial, assuming the input data string is a high order polynomial. Ordinarily, a data string is suffixed by a CRC code when being sent, and the received data is divided by a generator polynomial as described above. If the received data is dividable, it is judged that the data is correctly received.

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

When to Use a PDL_CRC Component

Use the PDL_CRC component when you need the function block that provide calculation in CCITT CRC16 or IEEE-802.3 CRC32. In this module, the generator polynomial is fixed to the numeric values for those two modes; therefore, the CRC value based on other generator polynomials cannot be calculated.

Quick Start

1. Drag a PDL_CRC component from the Component Catalog FMx/Digital/Cyclic Redundancy Check folder onto your schematic. The placed instance takes the name CRC_1.
2. Double-click to open the component's Configure dialog.
3. On the **Basic** tab, set the following parameters:
   - mode of the calculation
   - result endian
   - initial value

4. Build the project to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate the configuration data for the CRC_1 instance.
5. In the *main.c* file, initialize the peripheral and start the application.

```
#define BUFFER_SIZE (64u)/* Provide here a buffer capacity. */

uint32_t u32CRC_Val;/* The CRC calculation result will be in this integer */
uint8_t u8Count; /* Loop counter*/

/* The buffer which will contain the data which CRC need to calculate */
uint8_t u8DataBuffer[BUFFER_SIZE] = {0};

Crc_Init(&CRC_1_Config);

for(u8Count = 0;u8Count < BUFFER_SIZE; u8Count++)
{
   Crc_Push8(u8DataBuffer[u8Count]);
}

u32CRC_Val= Crc_ReadResult();
```

6. Build and program the device.

Component Parameters

The PDL_CRC 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 basic peripheral initialization settings.

Parameter Name	Description
bDataLittleEndian	CRC feed data byte order
bDataLsbFirst	CRC feed data bit order
bFinalXor	Return CRC result as an XOR value
bResultLittleEndian	CRC result byte order
bResultLsbFirst	CRC result bit order
bUseDma	Enable DMA terminal and trigger
enMode	Select 16- or 32-bit standard CRC
u32CrcInitValue	Initial value of the CRC

Component Usage

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

Generated Data

The PDL_CRC 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. *CRC_1_config.c*). Each variable is also prefixed with the instance name of the component.

Data Structure Type	Name	Description
stc_crc_config_t	CRC_1_Config	Configuration structure. Pass this to Crc_Init() in order to initialize the peripheral.

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.

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.

Industry Standards

The PDL_CRC component conforms to CCITT CRC16 and IEEE-802.3 CRC32 industry standards.

Resources

The PDL_CRC component uses the CRC (Cyclic Redundancy Check) 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.