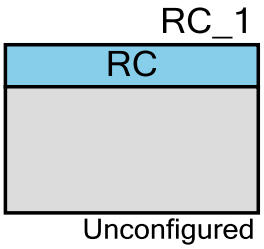
Remote Control (PDL_RC)
1.0

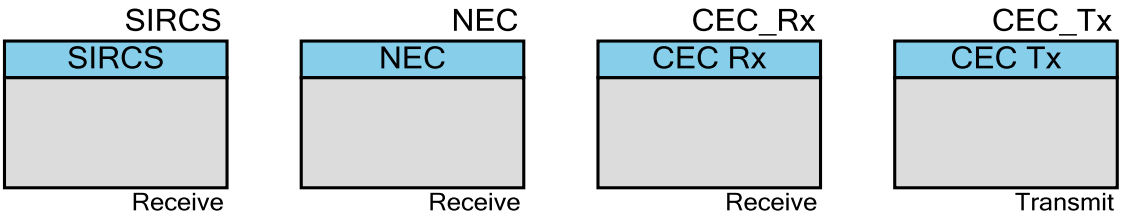


Features

- Up to 2 Channels
- HDMI-CEC/ High Definition Multimedia Interface Consumer Electronics Control transmitter/receiver
- SIRCS/Sony Infrared Remote Control mode
- NEC/Association for Electric Home Appliances mode
- Capable of adjusting detection timings for start bit and data bit
- Equipped with noise filter

General Description

The Peripheral Driver Library (PDL) Remote Control (PDL_RC) component is a multifunctional peripheral block with the following operational modes: SIRCS, NEC, CEC_Rx, CEC_Tx. Each mode is available as a pre-configured schematic in the PSoC Creator Component Catalog.



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

When to Use a PDL_RC Component

Use the PDL_RC component for receiving HDMI-CEC signals and infrared remote control signals.

Quick Start

1. Drag a PDL_RC component from the Component Catalog FMx/Communication/Remote Control/ folder onto your schematic. The placed instance takes the name RC_1.

2. Double-click to open the component's Configure dialog.
3. On the **Basic** tab select the configuration of the component.
4. Depending on the selected component configuration, there will be added tab(s) with specific parameters
5. Assign pins in your device using the Pin Editor. If you are creating a design for a development kit, refer the kit User Guide for suitable pin assignments.
6. 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 RC_1 instance.
7. In the *main.c* file, initialize the peripheral and start the application.

```
RC_1_SetPinFunc_CEC();
Rc_Rx_Cec_Init(&RC_1_HW ,&RC_1_Config);
```

8. Build and program the device.

Component Parameters

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

General Tab

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

Parameter Name	Description
RCConfig	Selects the timer operating mode to one of the following modes: ▪ Unconfigured – This is the default mode. The RC component must be configured at run time when configured in this mode. ▪ SIRCS - configured to be in SIRCS mode. ▪ NEC - configured to be in NEC mode. ▪ CEC_Rx - configured to be in CEC receiver mode. ▪ CEC_Tx – configured to be in CEC transmitter mode.

Interrupts Tab

This tab contains the Interrupt configuration settings.

Parameter Name	Description
bTouchNvic	Install interrupts in NVIC.
bRcRxSircsAckIrq	ACK detection interrupt enable. Visible when SIRCS mode selected.

Parameter Name	Description
pfnRcRxSircsAckIrqCb	Callback routine for ACK detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when SIRCS mode selected.
bRcRxSircsCntOvfIrq	Counter overflow interrupt enable. Visible when SIRCS mode selected.
pfnRcRxSircsCntOvfIrqCb	Callback routine for counter overflow interrupt. Note: this generates a declaration only - USER must implement the function. Visible when SIRCS mode selected.
bRcRxSircsStartIrq	Start bit detection interrupt enable. Visible when SIRCS mode selected.
pfnRcRxSircsStartIrq	Callback routine for start bit detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when SIRCS mode selected.
bRcRxNecAckIrq	ACK detection interrupt enable. Visible when NEC mode selected.
pfnRcRxNecAckIrqCb	Callback routine for ACK detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when NEC mode selected.
bRcRxNecCntOvfIrq	Counter overflow interrupt enable. Visible when NEC mode selected.
bRcRxNecRepeatCodeIrq	Repeat code interrupt enable. Visible when NEC mode selected.
pfnRcRxNecCntOvfIrqCb	Callback routine for counter overflow interrupt. Note: this generates a declaration only - USER must implement the function. Visible when NEC mode selected.
pfnRcRxNecRepeatCodeIrqCb	Callback routine for repeat code interrupt. Note: this generates a declaration only - USER must implement the function. Visible when NEC mode selected.
bRcRxNecStartIrq	Start bit detection interrupt enable. Visible when NEC mode selected.
pfnRcRxNecStartIrqCb	Callback routine for start bit detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when NEC mode selected.
bRcRxCecAckIrq	ACK detection interrupt enable. Visible when CEC_Rx mode selected.
pfnRcRxCecAckIrqCb	Callback routine for ACK detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Rx mode selected.
bRcRxCecCntOvfIrq	Counter overflow interrupt enable. Visible when CEC_Rx mode selected.
bRcRxCecMaxDataIrq	Enable maximum data bit width violation detection interrupt. Visible when CEC_Rx mode selected.
bRcRxCecMinDataIrq	Enable minimum data bit width violation detection interrupt. Visible when CEC_Rx mode selected.
pfnRcRxCecCntOvfIrqCb	Callback routine for counter overflow interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Rx mode selected.
pfnRcRxCecMaxDataIrqCb	Callback routine for maximum data width violation interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Rx mode selected.
pfnRcRxCecMinDataIrqCb	Callback routine for minimum data width violation interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Rx mode selected.

Parameter Name	Description
bRcRxCecStartIrq	Start bit detection interrupt enable. Visible when CEC_Rx mode selected.
pfnRcRxCecStartIrqCb	Callback routine for start bit detection interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Rx mode selected.
bRcTxCecBusErrorIrq	Bus error interrupt enable. Visible when CEC_Tx mode selected.
pfnRcTxIrqBusErrorCb	Callback routine for bus error interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Tx mode selected.
bRcTxCecStatusIrq	Tx status interrupt enable. Visible when CEC_Tx mode selected.
pfnRcTxIrqTxStatusCb	Callback routine for Tx status interrupt. Note: this generates a declaration only - USER must implement the function. Visible when CEC_Tx mode selected.

Timing Tab

This tab contains the Timing configuration settings.

Parameter Name	Description
enSrcClk	RC source clock.
u16DivVal	Source clock divider.
enOverflowCycle	Time before overflow. Visible when SIRCS or NEC or CEC_Rx mode is selected.

Receive Tab

This tab contains the Timing configuration settings. This tab is visible when SIRCS or NEC, or CEC_Rx mode is selected

Parameter Name	Description
bAddrCmpEn	Compare against address in the address register. Visible when SIRCS or NEC, or CEC_Rx mode is selected.
u8Addr1	RC receiver address 1 to compare with device address. Visible when SIRCS or NEC, or CEC_Rx mode is selected.
u8Addr2	RC receiver address 2 to compare with device address. Visible when SIRCS or NEC, or CEC_Rx mode is selected.
enThresholdType	Threshold type. Visible when SIRCS or NEC, or CEC_Rx mode is selected
u8MinPulseWidth	Minimum pulse duration.
u8RepeatWidth	Repeat code width. Visible when NEC mode selected.
u8StartBitWidth	Duration of the start bit. Visible when SIRCS or NEC, or CEC_Rx mode is selected.
u8ThresholdWidth	Threshold width. Visible when SIRCS or NEC, or CEC_Rx mode is selected.

Parameter Name	Description
bBusErrorPulseOutput	Enable Bus error pulse detection. Visible when CEC_Rx mode is selected.
bMaxDataBitDetect	Enable maximum data bit detection. Visible when CEC_Rx mode is selected.
bMinDataBitDetect	Enable minimum data bit detection. Visible when CEC_Rx mode is selected.
u8MaxDataWidth	Maximum data width. Visible when CEC_Rx mode is selected.
u8MinDataWidth	Minimum data width. Visible when CEC_Rx mode is selected.

Transmit Tab

This tab contains the Interrupt configuration settings. This tab is visible when CEC_Tx mode selected.

Parameter Name	Description
u8FreeCycle	Signal free time.

Component Usage

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

Generated Data

The PDL_RC 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. *RC_1_config.c*). Each variable is also prefixed with the instance name of the component.

Data Structure Type	Name	Description
stc_rc_rx_sircs_irq_en_t	RC_1_IrqEn	Interrupt enable structure. Generated if the SIRCS mode selected.
stc_rc_rx_sircs_irq_cb_t	RC_1_IrqCb	Interrupt callback structure. Generated if the SIRCS mode selected.
stc_rc_rx_sircs_config_t	RC_1_Config	Configuration structure. Generated if the SIRCS mode selected.
stc_rc_rx_nec_irq_en_t	RC_1_IrqEn	Interrupt enable structure. Generated if the NEC mode selected.
stc_rc_rx_nec_irq_cb_t	RC_1_IrqCb	Interrupt callback structure. Generated if the NEC mode selected.
stc_rc_rx_nec_config_t	RC_1_Config	Configuration structure. Generated if the NEC mode selected.
stc_rc_rx_cec_irq_en_t	RC_1_IrqEn	Interrupt enable structure. Generated if the CEC_Rx mode selected.

Data Structure Type	Name	Description
stc_rc_rx_cec_irq_cb_t	RC_1_IrqCb	Interrupt callback structure. Generated if the CEC_Rx mode selected.
stc_rc_rx_cec_config_t	RC_1_Config	Configuration structure. Generated if the CEC_Rx mode selected.
stc_rc_tx_cec_irq_en_t	RC_1_IrqEn	Interrupt enable structure. Generated if the CEC_Tx mode selected.
stc_rc_tx_cec_irq_cb_t	RC_1_IrqCb	Interrupt callback structure. Generated if the CEC_Tx mode selected.
stc_rc_tx_cec_config_t	RC_1_Config	Configuration structure. Generated if the CEC_Tx mode selected.

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…**" option in the drop-down menu.

Preprocessor Macros

The RC component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the component (e.g. "RC_1").

Macro	Description
RC_1_SetPinFunc_CEC	Macro to assign RC CEC signal in the device pin.
RC_1_HW	Hardware pointer to the block instance in the device. This should be used in all API calls when specifying the block to access.

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 RC component is specified to trigger interrupts, it will generate the callback function declaration that will be called from the RC 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
RC_1_RcRxSircsStartIrqCb	Start interrupt callback function. Generated if the SIRCS mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxSircsAckIrqCb	ACK interrupt callback function. Generated if the SIRCS mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxSircsCntOvfIrqCb	Counter overflow interrupt callback function. Generated if the SIRCS mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxNecStartIrqCb	Start interrupt callback function. Generated if the NEC mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxNecAckIrqCb	ACK interrupt callback function. Generated if the NEC mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxNecCntOvfIrqCb	Counter overflow interrupt callback function. Generated if the NEC mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxNecRepeatCodeIrqCb	Repeat code interrupt callback function. Generated if the NEC mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxCecStartIrqCb	Start interrupt callback function. Generated if the CEC_Rx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxCecAckIrqCb	ACK interrupt callback function. Generated if the CEC_Rx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxCecCntOvfIrqCb	Counter overflow interrupt callback function. Generated if the CEC_Rx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxCecMinDataIrqCb	Minimum data interrupt callback function. Generated if the CEC_Rx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcRxCecMaxDataIrqCb	Maximum data interrupt callback function. Generated if the CEC_Rx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcTxCecBusErrorIrqCb	Bus error interrupt callback function. Generated if the CEC_Tx mode selected. Note: this generates a declaration only - USER must implement the function.
RC_1_RcTxCecAckIrqCb	Transfer status interrupt callback function. Generated if the CEC_Tx mode selected. 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_RC component uses the RC (Remote Control) 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.