Typographical Conventions for C

Normal Text	Control Flow	Keyword	Data Type
Decimal	Octal	Hex	Binary
Float	Standard Suffix	Char	String
String Char	Comment	Symbol	Preprocessor
Prep. Lib	Region Marker	Error	

Doxygen text

Normal Text	Tags	Custom Tags	Word
HTML Tag	*Entities*	Description	Comment
Region	Identifier	HTML Comment	Types
Code	*Dot Graph*	*Formulas*	*Message Sequence Chart*
Verbatim	Note	Warning	Attention
Todo	Error		

Alerts text

Normal Text	Alert Level 1	Alert Level 2	Alert Level 3
Region Marker			

Modelines text

Comment	Keyword	Variable	Number
String	Value	Option ON	Option OFF

```c
/*
 * Copyright (C) 2016 Texas Instruments Incorporated - http://www.ti.com/
 *
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   * Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   * Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the
 *     distribution.
 *
 *   * Neither the name of Texas Instruments Incorporated nor the names of
 *     its contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 *  File     :   pru_rpmsg.c
 *
 *  Summary :   An RPMsg implementation for the PRU to use while communicating
 *              with the ARM host.
 *
 *  Notes   :
 *  - Implementaion of the interface described in "pru_rpmsg.h"
 */

#include <pru_rpmsg.h>

struct pru_rpmsg_hdr {
    uint32_t    src;
    uint32_t    dst;
    uint32_t    reserved;
    uint16_t    len;
    uint16_t    flags;
    uint8_t     data[0];
};

struct pru_rpmsg_ns_msg {
    char        name[RPMSG_NAME_SIZE];
    char        desc[RPMSG_NAME_SIZE];
    uint32_t    addr;
    uint32_t    flags;
};

int16_t pru_rpmsg_init(
    struct pru_rpmsg_transport  *transport,
    struct fw_rsc_vdev_vring    *vring0,
    struct fw_rsc_vdev_vring    *vring1,
    uint32_t            to_arm_event,
    uint32_t            from_arm_event
)
{
    if (to_arm_event > MAX_VALID_EVENT || to_arm_event < MIN_VALID_EVENT)
        return PRU_RPMSG_INVALID_EVENT;

    if (from_arm_event > MAX_VALID_EVENT || from_arm_event < MIN_VALID_EVENT)
        return PRU_RPMSG_INVALID_EVENT;

    pru_virtqueue_init(&transport->virtqueue0, vring0, to_arm_event, from_arm_event);
    pru_virtqueue_init(&transport->virtqueue1, vring1, to_arm_event, from_arm_event);

    return PRU_RPMSG_SUCCESS;
}

int16_t pru_rpmsg_send(
    struct pru_rpmsg_transport  *transport,
    uint32_t            src,
    uint32_t            dst,
    void            *data,
    uint16_t            len
)
{
    struct pru_rpmsg_hdr    *msg;
    uint32_t        msg_len;
    int16_t         head;
    struct pru_virtqueue    *virtqueue;

    /*
     * The length of our payload is larger than the maximum RPMsg buffer size
     * allowed
     */
    if (len > (RPMSG_BUF_SIZE - sizeof(struct pru_rpmsg_hdr)))
        return PRU_RPMSG_BUF_TOO_SMALL;

    virtqueue = &transport->virtqueue0;

    /* Get an available buffer */
    head = pru_virtqueue_get_avail_buf(virtqueue, (void **)&msg, &msg_len);

    if (head < 0)
        return PRU_RPMSG_NO_BUF_AVAILABLE;

    /* Copy local data buffer to the descriptor buffer address */
    memcpy(msg->data, data, len);
    msg->len = len;
    msg->dst = dst;
    msg->src = src;
    msg->flags = 0;
    msg->reserved = 0;

    /* Add the used buffer */
    if (pru_virtqueue_add_used_buf(virtqueue, head, msg_len) < 0)
        return PRU_RPMSG_INVALID_HEAD;

    /* Kick the ARM host */
    pru_virtqueue_kick(virtqueue);

    return PRU_RPMSG_SUCCESS;
}

int16_t pru_rpmsg_receive(
    struct pru_rpmsg_transport  *transport,
    uint16_t            *src,
    uint16_t            *dst,
    void            *data,
    uint16_t            *len
)
{
    int16_t         head;
    struct pru_rpmsg_hdr    *msg;
    uint32_t        msg_len;
    struct pru_virtqueue    *virtqueue;

    virtqueue = &transport->virtqueue1;

    /* Get an available buffer */
    head = pru_virtqueue_get_avail_buf(virtqueue, (void **)&msg, &msg_len);

    if (head < 0)
        return PRU_RPMSG_NO_BUF_AVAILABLE;


    /* Copy the message payload to the local data buffer provided */
    memcpy(data, msg->data, msg->len);
    *src = msg->src;
    *dst = msg->dst;
    *len = msg->len;

    /* Add the used buffer */
    if (pru_virtqueue_add_used_buf(virtqueue, head, msg_len) < 0)
        return PRU_RPMSG_INVALID_HEAD;

    /* Kick the ARM host */
    pru_virtqueue_kick(virtqueue);
```

```c

    return PRU_RPMSG_SUCCESS;
}

int16_t pru_rpmsg_channel(
    enum pru_rpmsg_ns_flags flags,
    struct pru_rpmsg_transport  *transport,
    char                *name,
    char                *desc,
    int32_t             port
)
{
    struct pru_rpmsg_ns_msg ns_msg;
    uint8_t             i;

    for (i = 0; i < RPMSG_NAME_SIZE; i++) {
        ns_msg.name[i] = name[i];
        ns_msg.desc[i] = desc[i];
    }
    ns_msg.addr = port;
    ns_msg.flags = flags;

    return pru_rpmsg_send(transport, port, 53, &ns_msg, sizeof(ns_msg));
}

```