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

```
 1 /*
 2  * Copyright (C) 2016 Texas Instruments Incorporated - http://www.ti.com/
 3  *
 4  *
 5  * Redistribution and use in source and binary forms, with or without
 6  * modification, are permitted provided that the following conditions
 7  * are met:
 8  *
 9  *  * Redistributions of source code must retain the above copyright
10  *    notice, this list of conditions and the following disclaimer.
11  *
12  *  * Redistributions in binary form must reproduce the above copyright
13  *    notice, this list of conditions and the following disclaimer in the
14  *    documentation and/or other materials provided with the
15  *    distribution.
16  *
17  *  * Neither the name of Texas Instruments Incorporated nor the names of
18  *    its contributors may be used to endorse or promote products derived
19  *    from this software without specific prior written permission.
20  *
21  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25  * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28  * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29  * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30  * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32  */
33
34 /**
35  *  File    :   pru_virtqueue.c
36  *
37  *  Summary :   A virtual queue implementation to simplify vring usage.
38  *
39  *  Notes   :
```

```
 *  - Implementaion of the interface described in "pru_virtqueue.h"
 */
#include <pru_virtqueue.h>

volatile register uint32_t __R31;

/* bit 5 is the valid strobe to generate system events with __R31 */
#define INT_ENABLE  (1 << 5)

/* __R31[3:0] can generate 15-0 which maps to system events 31-16
 * e.g. to generate PRU-ICSS System Event 17 (pru_mst_intr[1])
 * __R31 = (INT_ENABLE | (17 - INT_OFFSET));
 */
#define INT_OFFSET  16

void pru_virtqueue_init(
    struct pru_virtqueue        *vq,
    struct fw_rsc_vdev_vring    *vring,
    uint32_t            to_arm_event,
    uint32_t            from_arm_event
)
{
    vq->id = vring->notifyid;
    vq->to_arm_event = to_arm_event;
    vq->from_arm_event = from_arm_event;
    vq->last_avail_idx = 0;

    vring_init(&vq->vring, vring->num, (void*)vring->da, vring->align);
}

int16_t pru_virtqueue_get_avail_buf(
    struct pru_virtqueue    *vq,
    void            **buf,
    uint32_t            *len
)
{
    int16_t         head;
    struct vring_desc   desc;
    struct vring_avail  *avail;

    avail = vq->vring.avail;

    /* There's nothing available */
    if (vq->last_avail_idx == avail->idx)
        return PRU_VIRTQUEUE_NO_BUF_AVAILABLE;

    /*
     * Grab the next descriptor number the ARM host is advertising, and
     * increment the last available index we've seen.
     */
    head = avail->ring[vq->last_avail_idx++ & (vq->vring.num - 1)];

    desc = vq->vring.desc[head];
    *buf = (void *)(uint32_t)desc.addr;
    *len = desc.len;

    return (head);
}

int16_t pru_virtqueue_add_used_buf(
    struct pru_virtqueue    *vq,
```

```
101     int16_t         head,
102     uint32_t            len
103 )
104 {
105     struct vring_used_elem  *used_elem;
106     uint32_t        num;
107     struct vring_used   *used;
108 
109     num = vq->vring.num;
110     used = vq->vring.used;
111 
112     if (head > num)
113         return PRU_VIRTQUEUE_INVALID_HEAD;
114 
115     /*
116      * The virtqueue's vring contains a ring of used buffers.  Get a pointer to
117      * the next entry in that used ring.
118      */
119     used_elem = &used->ring[used->idx++ & (num - 1)];
120     used_elem->id = head;
121     used_elem->len = len;
122 
123     return PRU_VIRTQUEUE_SUCCESS;
124 }
125 
126 int16_t pru_virtqueue_kick(
127     struct pru_virtqueue    *vq
128 )
129 {
130     /* If requested, do not kick the ARM host */
131     if (vq->vring.avail->flags & VRING_AVAIL_F_NO_INTERRUPT)
132         return PRU_VIRTQUEUE_NO_KICK;
133 
134     /* Generate a system event to kick the ARM */
135     __R31 = (INT_ENABLE | (vq->to_arm_event - INT_OFFSET));
136 
137     return PRU_VIRTQUEUE_SUCCESS;
138 }
139 
```