State machine diagram:

INITIAL (On Entry)
```
payload  =  new uint8_t[RPMSG_MESSAGE_SIZE];
resource_table_wrapper = new Resource_Table_Wrapper();
one_beep_a_second(PRU_0_GREEN_LED);
// Clear the status of all interrupts
CT_INTC.SECR0 = 0xFFFFFFFF;
CT_INTC.SECR1 = 0xFFFFFFFF;
// Allow OCP master port access by the PRU so the PRU can read external
memories
CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;
// Clear the status of the PRU-ICSS system event that the ARM will use to
'kick' us
//CT_INTC.SICR_bit.STATUS_CLR_INDEX = IRQ_ARM_TO_PRU0;

// Make sure the Linux drivers are ready for RPMsg communication
status = & resource_table_wrapper -> to_resourceTable -> rpmsg_vdev.status;
```

Transition INITIAL → COMMS_READY: `(*status & VIRTIO_CONFIG_S_DRIVER_OK)`

COMMS_READY (On Entry)
```
// Initialize the RPMsg transport structure
pru_rpmsg_init(&transport,
    &resource_table_wrapper -> to_resourceTable -> rpmsg_vring0,
    &resource_table_wrapper -> to_resourceTable -> rpmsg_vring1,
    IRQ_PRU0_TO_ARM,
     IRQ_ARM_TO_PRU0);
two_beeps_a_second(PRU_0_BLUE_LED);
```

Transition COMMS_READY → CHANNEL_CREATED: `pru_rpmsg_channel(RPMSG_NS_CREATE, &transport, CHAN_NAME, CHAN_DESC, CHAN_PORT) == PRU_RPMSG_SUCCESS`

CHANNEL_CREATED

Transition CHANNEL_CREATED → ARM_KICKED_US: `CT_INTC.SECR0_bit.ENA_STATUS_31_0 & (1 << IRQ_ARM_TO_PRU0)`

ARM_KICKED_US (On Entry)
```
//Clear the RPMsg received event status
CT_INTC.SICR_bit.STATUS_CLR_INDEX = IRQ_ARM_TO_PRU0;
```

Transition ARM_KICKED_US → CHANNEL_CREATED: `true`

Transition ARM_KICKED_US → PAYLOAD_RECEIVED: `pru_rpmsg_receive(&transport, &src, &dst, payload, &len) == PRU_RPMSG_SUCCESS`

PAYLOAD_RECEIVED (On Entry)
```
for (uint16_t i=0; i<len; i++) { // convert to Leds pattern and show for one second
    fill_struct_LEDS_unsigned(payload,  &dynamic_Buffer);
}//for

//Send IRQ to PRU1 so it gets ready to receive struct
__R31 = IRQ_XFR_PRU0_TO_PRU1 + PRU_IRQ_out_strobe;
```

Transition PAYLOAD_RECEIVED → SEND_TO_OTHER_PRU: `CT_INTC.SECR0_bit.ENA_STATUS_31_0 & (1 << IRQ_XFR_PRU1_TO_PR...`

SEND_TO_OTHER_PRU (On Entry / On Exit)
```
Comms_PRU_ucFSM local_Buffer = dynamic_Buffer;

CT_INTC.SICR = IRQ_XFR_PRU1_TO_PRU0;        //Clear IRQ
__R31 = IRQ_XFR_PRU0_TO_PRU1 + PRU_IRQ_out_strobe; // signal
sent to PRU1 to immediately receive this
__xout(14, 5, 0, local_Buffer);

five_beeps_a_second(PRU_0_RED_LED);
```

Transition SEND_TO_OTHER_PRU → CHANNEL_CREATED: `true`