Document Feedback

together, to get there!

AB155x RACE Command Protocol Application Note

V1.0

AIROHA

INTERNAL USE

Document Feedback

Agenda

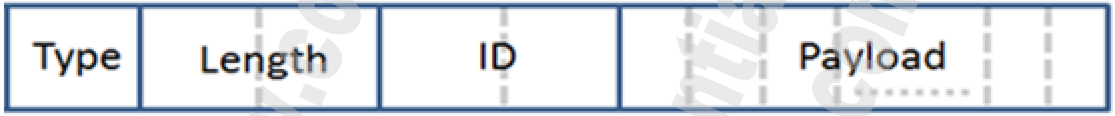
- Purpose
- Packet Format
- Architecture
- How to Add New RACE Commands
- Expected Result
- Demo
- RACE ID Category

Purpose

- Run-time Application Command Environment (RACE)
- RACE command is a powerful control interface for customers to implement **unique features**.
- **Real time** control SW/HW configuration or function to adjust the product behavior.
- With different communication protocols, enhance the ability of RACE command (e.g., **FOTA**, **Audio turning, etc.)**.
- RACE command now only supports binary format.

Document Feedback

RACE Command Packet Format


Type

Length

ID

Payload

- Type: 1 byte
  - 0x5A Command with Response
  - 0x5B Response
  - 0x5C Command without Response
  - 0x5D Notification
- Length: 2 bytes
  - Include ID and Payload
- ID: 2 bytes
  - Command or Response ID
- Little Endian:
  - 0xABCD -> 0xCD, 0xAB

Example: Command

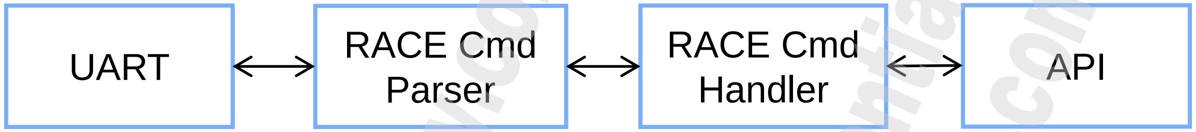
Header	Type	Length	ID	Payload
0x05	0x5A	0x04	0x200	0xABCD
0x05	0x5A	**0x04**, 0x00	**0x00,0x02**	**0xCD, 0xAB**
Raw Data: 0x05, 0x5A, 0x04, 0x00, 0x00, 0x02, 0xCD, 0xAB				

Example: Response

Header	Type	Length	ID	Payload
0x05	0x5B	0x03	0x200	0x01
Raw Data: 0x05, 0x5B, 0x03, 0x00, 0x00, 0x02, 0x01				

Architecture

Receive RACE Command and processing:

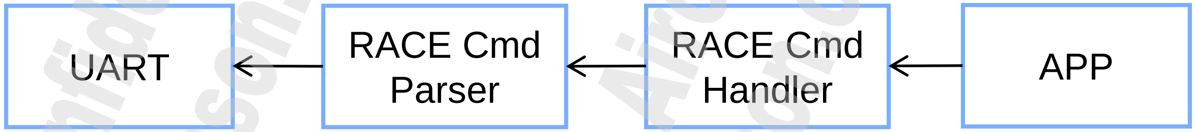


UART ↔ RACE Cmd Parser ↔ RACE Cmd Handler ↔ API

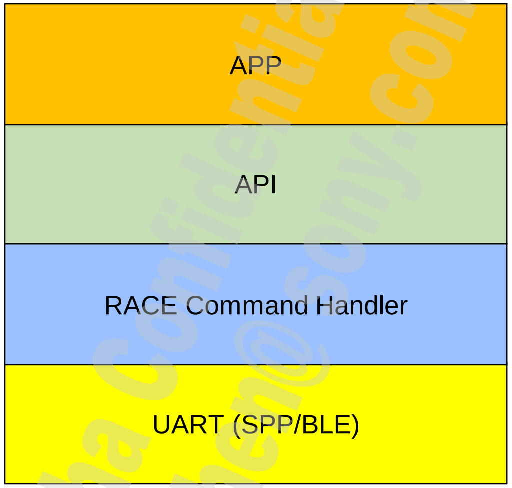
→ Processing RACE Command

← RACE Event Response
(If Type = 0x5C, without Response)

RACE Notification:



UART ← RACE Cmd Parser ← RACE Cmd Handler ← APP



APP

API

RACE Command Handler

UART (SPP/BLE)

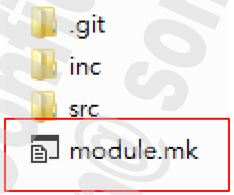
Add New RACE Command (1)

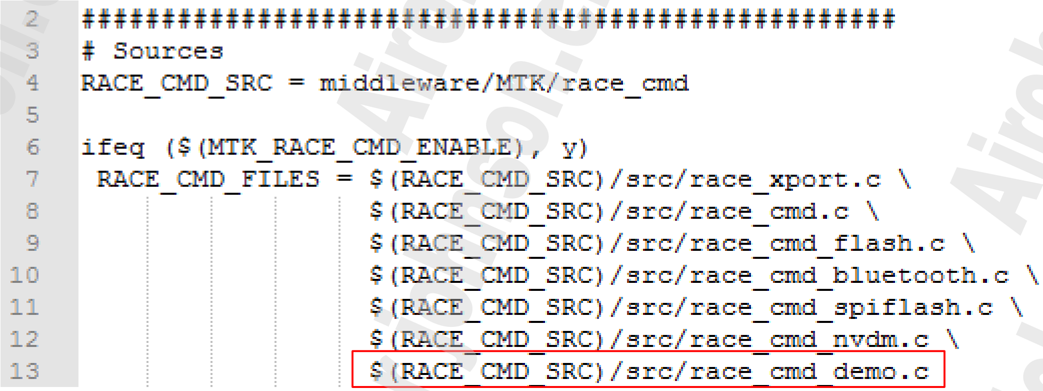
Step 1: Define RACE command packet format as follows:

Race Command						Race Event Response			
ID Range	NAME	Type	Length	ID	RACE Command / Notification Payload	Type	Length	ID	RACE Response Payload
0xF000~0xF100	RACE_CMD_DEMO_1	0x5A	6	0x0000	Para[4]	0x5B	7	0x0000	Status, Para[4]

Step 2: Open the folder: <sdk_path\mcu\middleware\MTK\race_cmd>.

Step 3: Edit module.mk and add C file path.

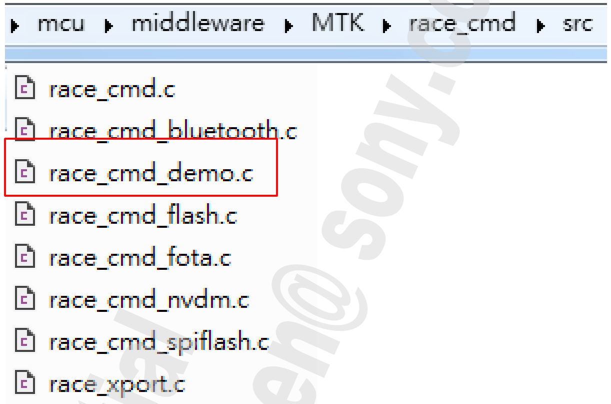




```
###################################################
# Sources
RACE_CMD_SRC = middleware/MTK/race_cmd

ifeq ($(MTK_RACE_CMD_ENABLE), y)
 RACE_CMD_FILES = $(RACE_CMD_SRC)/src/race_xport.c \
                  $(RACE_CMD_SRC)/src/race_cmd.c \
                  $(RACE_CMD_SRC)/src/race_cmd_flash.c \
                  $(RACE_CMD_SRC)/src/race_cmd_bluetooth.c \
                  $(RACE_CMD_SRC)/src/race_cmd_spiflash.c \
                  $(RACE_CMD_SRC)/src/race_cmd_nvdm.c \
                  $(RACE_CMD_SRC)/src/race_cmd_demo.c
```

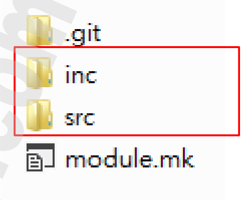
Add New RACE Command(2)

Step 4: Add C file in “src” folder.



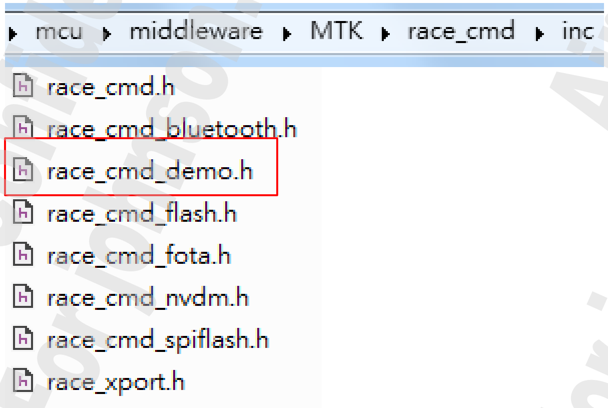
▸ mcu ▸ middleware ▸ MTK ▸ race_cmd ▸ src

- race_cmd.c
- race_cmd_bluetooth.c
- race_cmd_demo.c
- race_cmd_flash.c
- race_cmd_fota.c
- race_cmd_nvdm.c
- race_cmd_spiflash.c
- race_xport.c



- .git
- inc
- src
- module.mk

Step 5: Add Header file in “inc” folder.

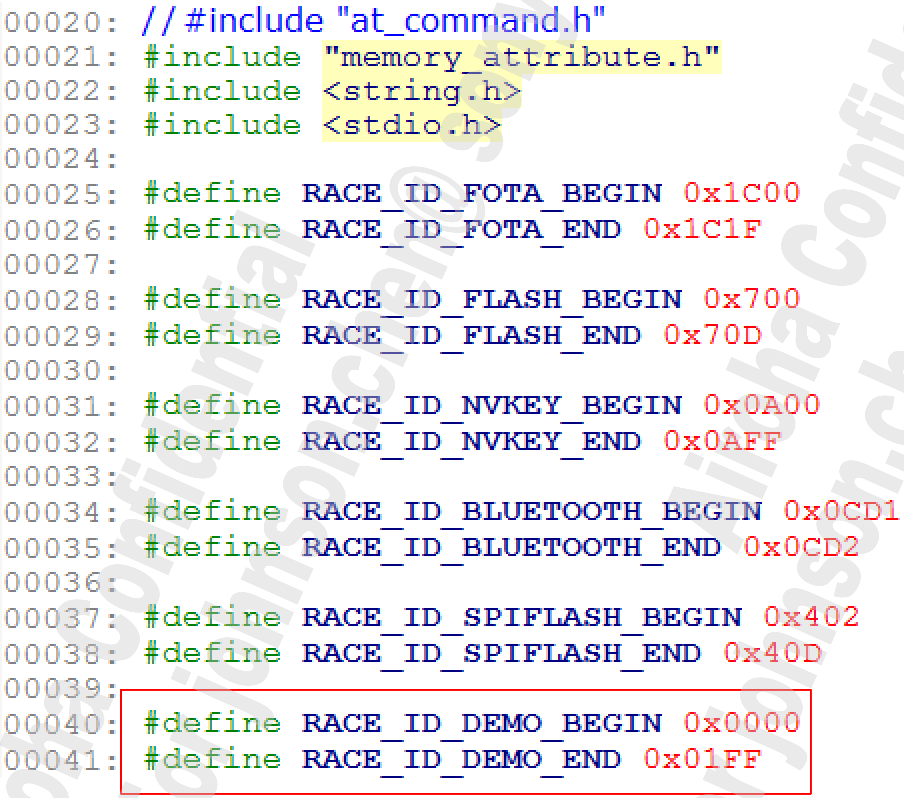


▸ mcu ▸ middleware ▸ MTK ▸ race_cmd ▸ inc

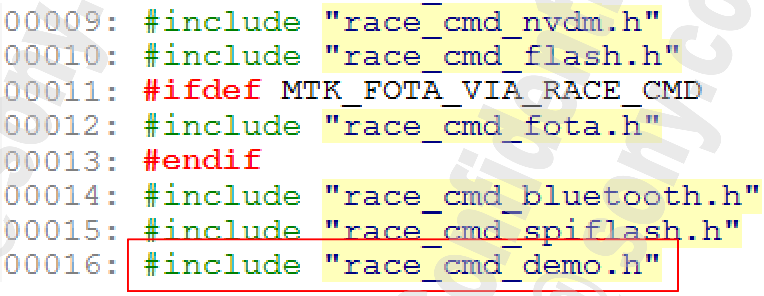
- race_cmd.h
- race_cmd_bluetooth.h
- race_cmd_demo.h
- race_cmd_flash.h
- race_cmd_fota.h
- race_cmd_nvdm.h
- race_cmd_spiflash.h
- race_xport.h

Add New RACE Command(3)

Step 6: Open Race_Cmd.c and define new race command range. Include the header file “race_cmd_demo.h”.



```
//#include "at_command.h"
#include "memory_attribute.h"
#include <string.h>
#include <stdio.h>

#define RACE_ID_FOTA_BEGIN 0x1C00
#define RACE_ID_FOTA_END 0x1C1F

#define RACE_ID_FLASH_BEGIN 0x700
#define RACE_ID_FLASH_END 0x70D

#define RACE_ID_NVKEY_BEGIN 0x0A00
#define RACE_ID_NVKEY_END 0x0AFF

#define RACE_ID_BLUETOOTH_BEGIN 0x0CD1
#define RACE_ID_BLUETOOTH_END 0x0CD2

#define RACE_ID_SPIFLASH_BEGIN 0x402
#define RACE_ID_SPIFLASH_END 0x40D

#define RACE_ID_DEMO_BEGIN 0x0000
#define RACE_ID_DEMO_END 0x01FF
```

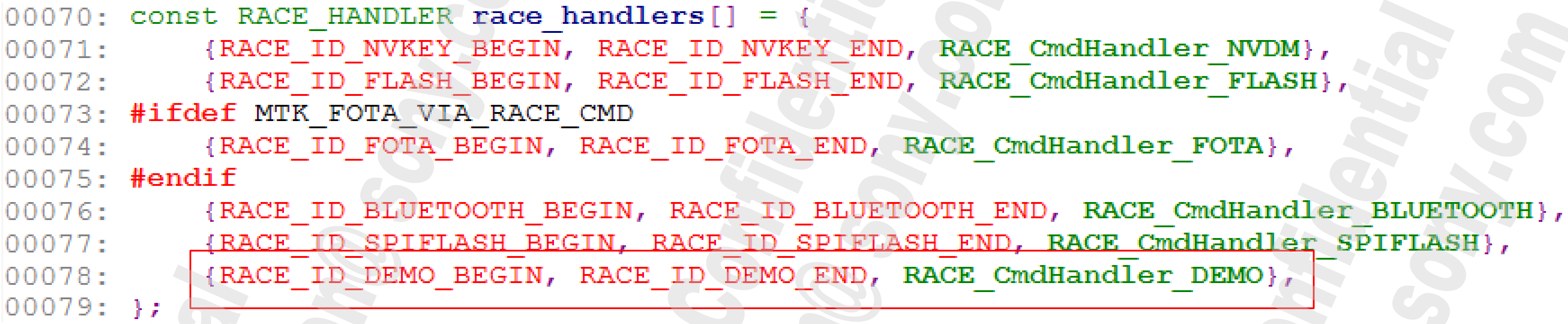


```
#include "race_cmd_nvdm.h"
#include "race_cmd_flash.h"
#ifdef MTK_FOTA_VIA_RACE_CMD
#include "race_cmd_fota.h"
#endif
#include "race_cmd_bluetooth.h"
#include "race_cmd_spiflash.h"
#include "race_cmd_demo.h"
```

Note: RACE command ID range 0x0000~0x1FFF is released for customers’ application

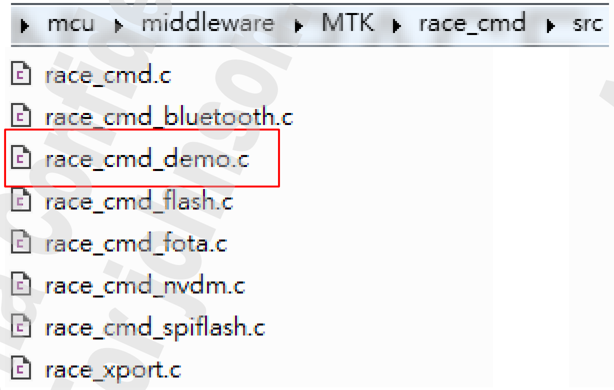
Add New RACE Command(4)

Step 7: In Race_Cmd.c, define the unique function entry as follows:



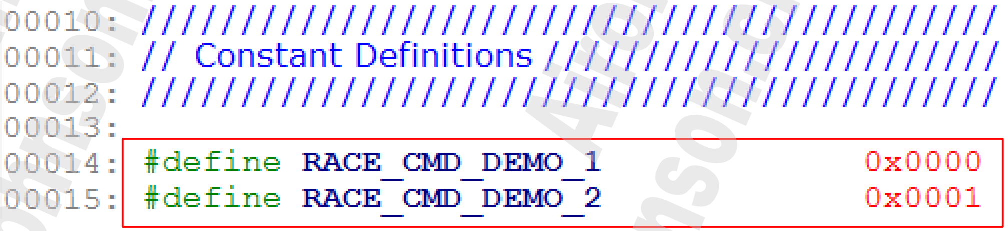
```
const RACE_HANDLER race_handlers[] = {
    {RACE_ID_NVKEY_BEGIN, RACE_ID_NVKEY_END, RACE_CmdHandler_NVDM},
    {RACE_ID_FLASH_BEGIN, RACE_ID_FLASH_END, RACE_CmdHandler_FLASH},
#ifdef MTK_FOTA_VIA_RACE_CMD
    {RACE_ID_FOTA_BEGIN, RACE_ID_FOTA_END, RACE_CmdHandler_FOTA},
#endif
    {RACE_ID_BLUETOOTH_BEGIN, RACE_ID_BLUETOOTH_END, RACE_CmdHandler_BLUETOOTH},
    {RACE_ID_SPIFLASH_BEGIN, RACE_ID_SPIFLASH_END, RACE_CmdHandler_SPIFLASH},
    {RACE_ID_DEMO_BEGIN, RACE_ID_DEMO_END, RACE_CmdHandler_DEMO},
};
```

Step 8: Open race_cmd_demo.c and define race cmd ID.



▸ mcu ▸ middleware ▸ MTK ▸ race_cmd ▸ src

- race_cmd.c
- race_cmd_bluetooth.c
- race_cmd_demo.c
- race_cmd_flash.c
- race_cmd_fota.c
- race_cmd_nvdm.c
- race_cmd_spiflash.c
- race_xport.c



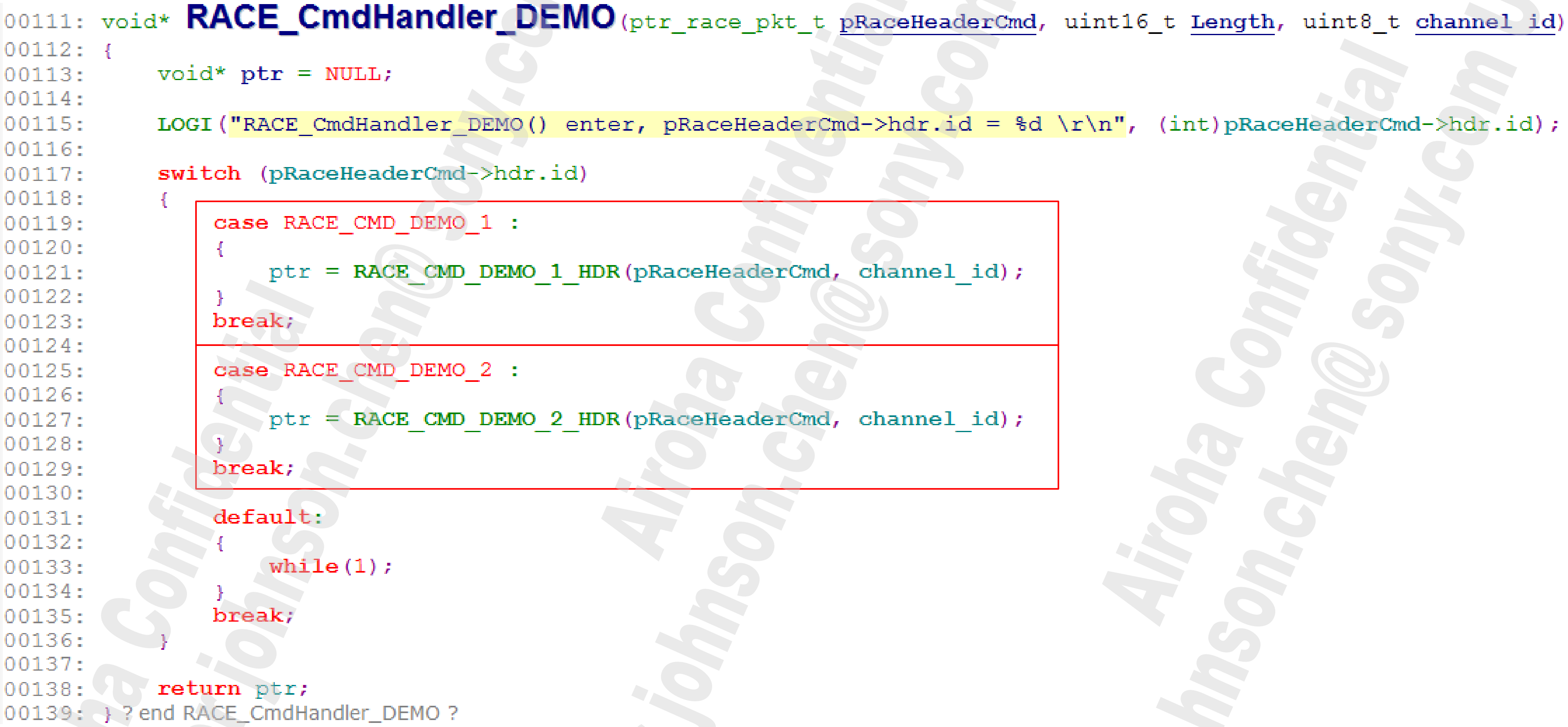
```
//////////////////////////////////////////////
// Constant Definitions //////////////////////
//////////////////////////////////////////////

#define RACE_CMD_DEMO_1                  0x0000
#define RACE_CMD_DEMO_2                  0x0001
```

Note: RACE command ID range is 0x0000~0x1FFF for customer

Document Feedback

Add New RACE Command(5)

Step 9: Define the RACE command sub-handler.

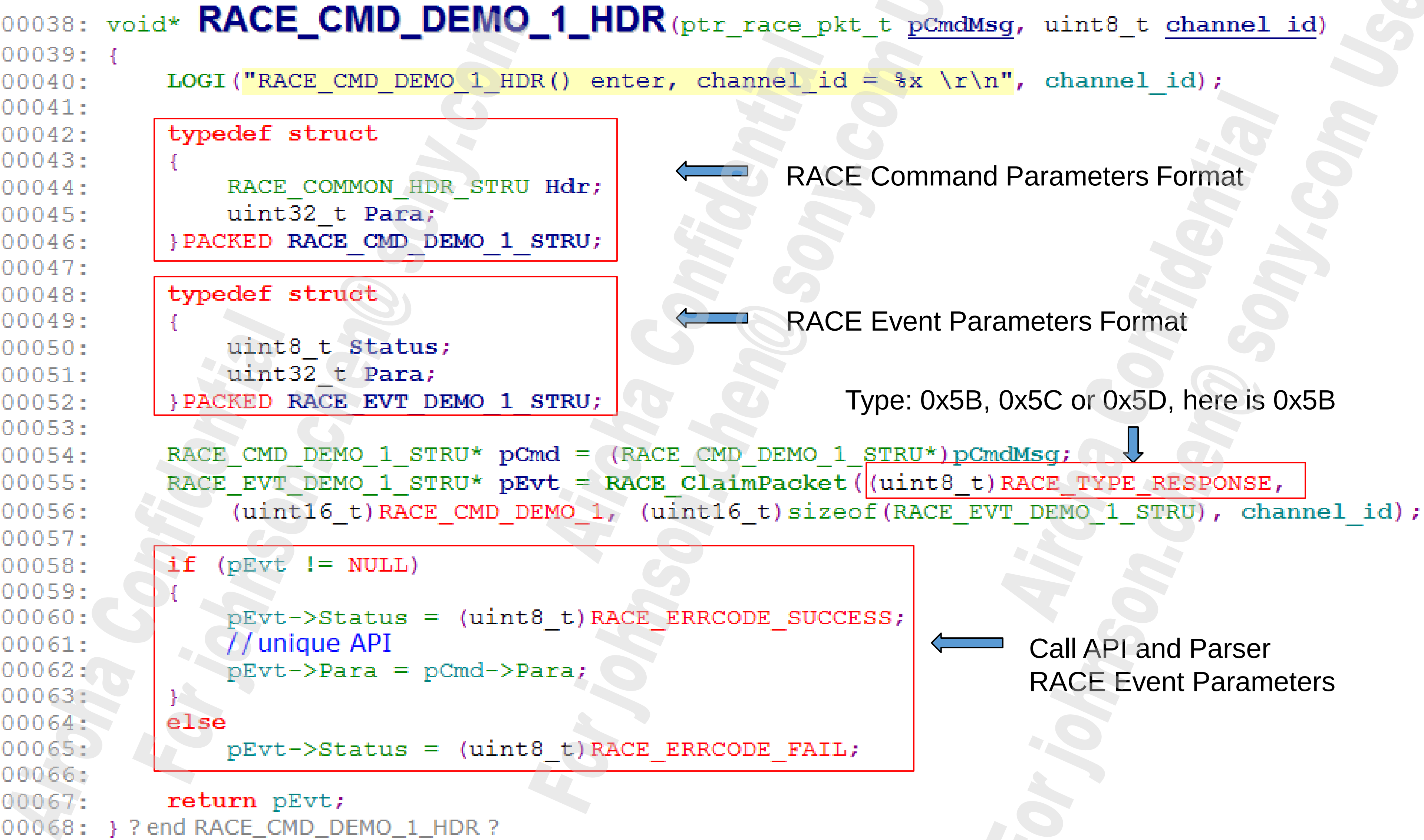


```
void* RACE_CmdHandler_DEMO(ptr_race_pkt_t pRaceHeaderCmd, uint16_t Length, uint8_t channel_id)
{
    void* ptr = NULL;

    LOGI("RACE_CmdHandler_DEMO() enter, pRaceHeaderCmd->hdr.id = %d \r\n", (int)pRaceHeaderCmd->hdr.id);

    switch (pRaceHeaderCmd->hdr.id)
    {
        case RACE_CMD_DEMO_1 :
        {
            ptr = RACE_CMD_DEMO_1_HDR(pRaceHeaderCmd, channel_id);
        }
        break;

        case RACE_CMD_DEMO_2 :
        {
            ptr = RACE_CMD_DEMO_2_HDR(pRaceHeaderCmd, channel_id);
        }
        break;

        default:
        {
            while(1);
        }
        break;
    }

    return ptr;
} ? end RACE_CmdHandler_DEMO ?
```

Document Feedback

Add New RACE Command(6)

Step 9 (continued): Program RACE cmd sub-handler.



```
void* RACE_CMD_DEMO_1_HDR(ptr_race_pkt_t pCmdMsg, uint8_t channel_id)
{
    LOGI("RACE_CMD_DEMO_1_HDR() enter, channel_id = %x \r\n", channel_id);

    typedef struct
    {
        RACE_COMMON_HDR_STRU Hdr;
        uint32_t Para;
    }PACKED RACE_CMD_DEMO_1_STRU;

    typedef struct
    {
        uint8_t Status;
        uint32_t Para;
    }PACKED RACE_EVT_DEMO_1_STRU;

    RACE_CMD_DEMO_1_STRU* pCmd = (RACE_CMD_DEMO_1_STRU*)pCmdMsg;
    RACE_EVT_DEMO_1_STRU* pEvt = RACE_ClaimPacket((uint8_t)RACE_TYPE_RESPONSE,
        (uint16_t)RACE_CMD_DEMO_1, (uint16_t)sizeof(RACE_EVT_DEMO_1_STRU), channel_id);

    if (pEvt != NULL)
    {
        pEvt->Status = (uint8_t)RACE_ERRCODE_SUCCESS;
        //unique API
        pEvt->Para = pCmd->Para;
    }
    else
        pEvt->Status = (uint8_t)RACE_ERRCODE_FAIL;

    return pEvt;
} ? end RACE_CMD_DEMO_1_HDR ?
```

RACE Command Parameters Format

RACE Event Parameters Format

Type: 0x5B, 0x5C or 0x5D, here is 0x5B

Call API and Parser
RACE Event Parameters

Expected Result

- Send RACE Command:

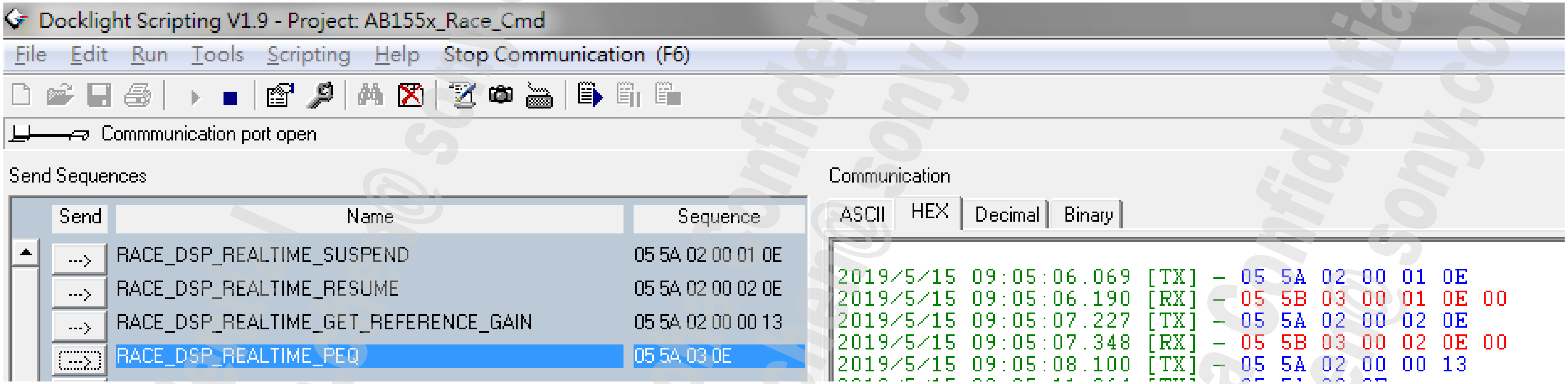
  0x05, 0x5A, 0x06, 0x00, 0x00, 0x00, 0xDD, 0xCC, 0xBB, 0xAA

- Receive RACE Event Response:

  0x05, 0x5B, 0x07, 0x00, 0x00, 0x00, 0x00, 0xDD, 0xCC, 0xBB, 0xAA

Demo

- The following information is a simple demonstration of how to use the RACE command using the Docklight simulation tool for serial communication protocols.



Send	Name	Sequence
-->	RACE_DSP_REALTIME_SUSPEND	05 5A 02 00 01 0E
-->	RACE_DSP_REALTIME_RESUME	05 5A 02 00 02 0E
-->	RACE_DSP_REALTIME_GET_REFERENCE_GAIN	05 5A 02 00 00 13
-->	RACE_DSP_REALTIME_PEQ	05 5A 03 0E

```
2019/5/15 09:05:06.069 [TX] - 05 5A 02 00 01 0E
2019/5/15 09:05:06.190 [RX] - 05 5B 03 00 01 0E 00
2019/5/15 09:05:07.227 [TX] - 05 5A 02 00 02 0E
2019/5/15 09:05:07.348 [RX] - 05 5B 03 00 02 0E 00
2019/5/15 09:05:08.100 [TX] - 05 5A 02 00 00 13
```

RACE ID Category

- ID Range 0x0000 ~ 0x01FF is reserved for custom RACE commands.

RACE ID	Description
0x0000 ~ 0x01FF	RACE command for customer
0x0200 ~ 0x03FF	Reserved for future used
0x0400 ~ 0x04FF	RACE command for storage
0x0500 ~ 0x06FF	Reserved for future used
0x0700 ~ 0x07FF	Race command for internal flash
0x0800 ~ 0x09FF	Reserved for future used
0x0A00 ~ 0x0AFF	Race command for NVDM
0x0B00 ~ 0x0BFF	Reserved for future used
0x0C00 ~ 0x0CFF	Race command for Bluetooth
0x0D00 ~ 0x0DFF	Reserved for future used
0x0E00 ~ 0x0EFF	Race command for DSP
0x0F00 ~ 0x1BFF	Reserved for future used
0x1C00 ~ 0x1C19	RACE command for FOTA
0x1C1A ~ 0x15FF	Reserved for future used
0x1600 ~ 0x167F	Race command for cap touch
0x1E00 ~ 0x1E1F	Race command for boot reason