Type A Board Dev Guide VI

Wilson 2020.11.5
https://github.com/TAMU-Robomasters/Tutorial

Roadmap:

1. STM32CubeMX, Keil uvision
2. LED, GPIO
3. Timer
4. PWM, passive buzzer, servo
5. Buttons
6. USB
7. Flash
8. I2C, IST8310 (magnetic sensor)
9. OLED
10. BMI088 (gyroscope)
11. Motor control with CAN
12. freeRTOS
13. IMU
14. DMA/Remote Control
15. Chassis tasks
16. Gimbal control
17. BIG PICTURE

DMA: Direct Memory Access

DMA enables I/O devices to access memory without the intervene from CPU.

On the Robomaster type A dev board, this feature can be used for the remote control, USART, SPI, I2C.

Note that the remote control receiver is connected to the DBUS port on the dev board. DBUS is similar to USART/UART but has opposite logic. In DBUS, 0 represents high, 1 represents low.

The data frame format used by robomaster remote controller has a fixed format. An implementation is needed to fit the bits into fields for easy access.

DBUS Specs:

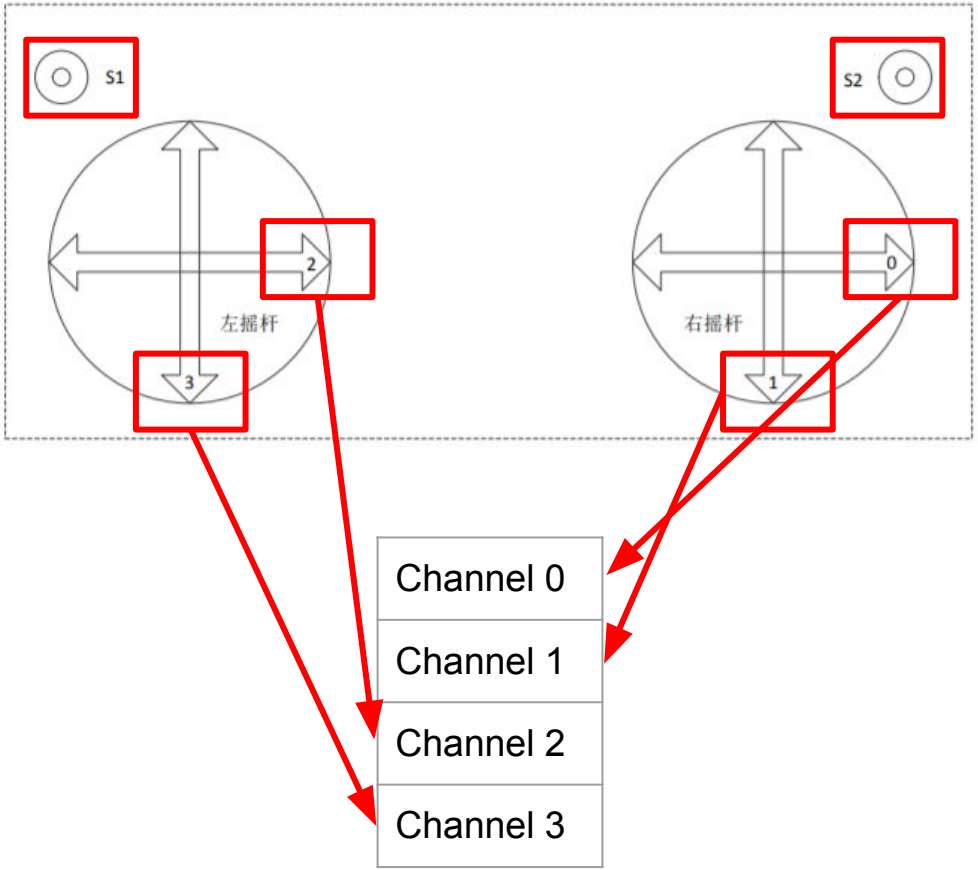
Parameter Name	Value
Baudrate	100 Kbps
Unit data length	8 bits
Checksum	Even checksum
Finishing bit	1
Flow control	None

Unit data length: 8 bits, which is 1 byte; a complete DBUS data frame has 18 bytes

Finishing bit: end of the 18 bytes, 1 bit equals 1 signals the end of the data frame

About the Robomaster Remote Control:

The next 3 slides explain the DBUS data frame structure



What for?	Channel 0	Channel 1	Channel 2	Channel 3	S1	S2
Bit shift	0	11	22	33	44	46
Number of bits	11	11	11	11	2	2
Signed?	No	No	No	No	No	No
Range	Max 1684 Mid 1024 Min 364	Max 1684 Mid 1024 Min 364	Max 1684 Mid 1024 Min 364	Max 1684 Mid 1024 Min 364	Max 3 Min 1	Max 3 Min 1
Function	Right joystick horizontal movement	Right joystick vertical movement	Left joystick horizontal movement	Left joystick vertical movement	1: switch up 2: switch down 3: switch mid	1: switch up 2: switch down 3: switch mid

What for?	Mouse x-axis	Mouse y-axis	Mouse z-axis	Mouse left click	Mouse right click	Keyboard
Bit shift	48	64	80	96	104	112
Number of bits	16	16	16	8	8	16
Signed?	Yes	Yes	Yes	No	No	No
Range	Max 32767 Min -32767 Static 0	Max 32767 Min -32767 Static 0	Max 32767 Min -32767 Static 0	Max 1 Min 0	Max 1 Min 0	See next slide
Function	Mouse movement speed, left: negative value; right: positive value			Mouse left button down: 1 Mouse left button not down: 0	Mouse right button down: 1 Mouse right button not down: 0	

DBUS data frame keyboard keys and bit position, after shifted 112 bits from the start of the DBUS data frame:

Key	Bit position
W	0
S	1
A	2
D	3
Q	4
E	5
Shift	6
Ctrl	7

Note: DBUS data frame is 18 bytes -> 144 bits, the last 16 bits are reserved and not used, but needed for transmission.

Suggestion for writing the DBUS data frame data structure:

1. Use shift operations to extract segments of data for different channels, switches, keyboard keys, mouses, etc
2. Use uint16_t to hold data that are longer than 8 bits
3. Use AND operations to filter out, or “mask” the unrelated bits for the current data segment, i.e. remove bits from channel 1 in the channel 2 bits container
4. Differentiate unsigned and signed data types
5. Use a struct to hold everything
6. Give meaningful names

Hardware setup

1. Remote control receiver
2. Remote control, paired with the receiver. When paired, the green light should be on, without blinking
3. Battery pack to power the dev board. USB power or the JLink downloader may not deliver sufficient power. To check if the power is sufficient, see if the receiver's green light is on.
4. For this tutorial, you do not need to connect the remote control to a computer. But in real competition, the remote control acts as a communication channel whose joysticks, switches, wheels are not used. Operators use a software provided by DJI that captures keyboard inputs and the remote control will send those keyboard/mouse signals to the dev board wirelessly.

STM32CubeMX settings:

1. Change USART1 to asynchronous mode
2. Baudrate set to 100000
3. Word length = 8 bits
4. No parity bit
5. Stop bit1 = 1
6. Enable USART DMA USART1_RX by clicking on the “Add” button in DMA settings
7. Set priority to very high
8. Enable the 6 green LEDs on the dev board. The code for this tutorial will turn on and off the corresponding green LEDs based on the switch positions on the remote control, i.e. 2 groups of 3 LEDs corresponding to the 3 possible positions of the 2 switches.
9. FreeRTOS is used. Remember to change the FreeRTOS task queue.

Keil/programming side techniques:

1. Have a buffer to hold the DMA data
2. Still need interrupt to handle data coming from DBUS port. Such interrupt handler disables DMA, put data to the DMA buffer, then re-enables DMA.
3. For completeness, an if statement can be used: if the data frame has a length of 18 bytes, process the data, i.e. put data into the remote control data struct for easy access
4. Note the direction when counting bits in the 18-byte data frame.
5. Do not read the DMA buffer directly, copy the buffer contents to a separate container, access that data container instead.
6. In UART interrupt handling, to access DMA, disable DMA first, then do the data receiving, then re-enable DMA. Do not forget to copy content to a separate buffer.

Expected outcome:

1. Power on the board with a battery
2. Pair the remote control with the receiver connected to the dev board via DBUS
3. Flip the two switches on the remote control
4. You should be able to see the green LEDs on the dev board on and off based on the combinations of the remote control switches.