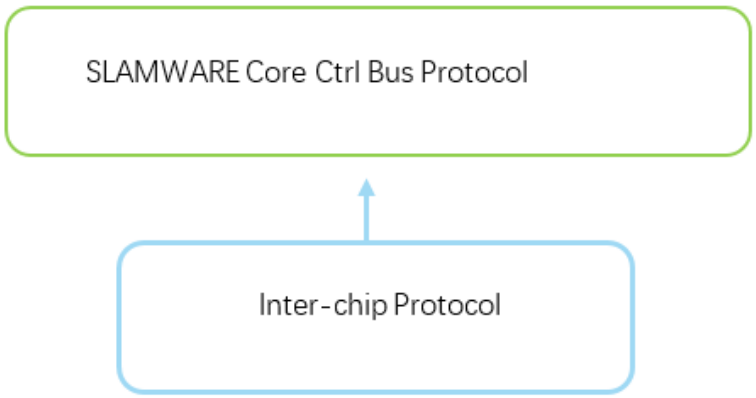
KBSW180104 SLAMWARE Control Bus

Control Bus Control Bus

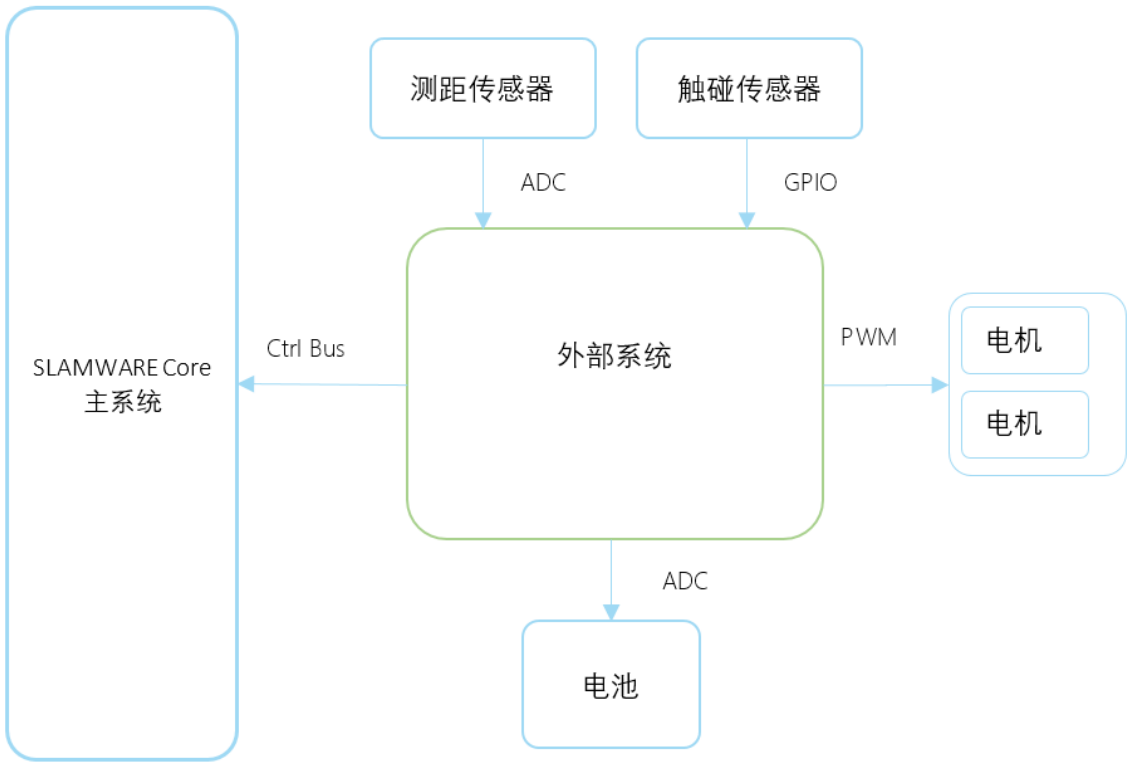
SLAMWARE CoreControl BusControl BusSLAMWARE CoreSLAMWARE CoreControl Bus Control Bus115200bpsSLAMWARE Core SLAMWARE Core Control BusInter-chipInter-chipSLAMTECSLAMTECUSARTUSBI^{2}CTCP

Ctrl BusInter-chip



SLAMWARE Core

SLAMWARE Core



Inter-chip

Inter-chip

2(Peer)
USARTTCPUSBI2CInter-chip Protocol

Inter-chip ProtocolFIFO

Inter-chip Protocol(sub-profile)Standard Profile-

Inter-chip

Inter-chip Protocol

	u8	u8[n]	u8	u8[n]	u8	u8	u8	u8[n]	u8
	Flag	ExFlag[n]	Addr	ExAddr[n]	Len	LenH	CMD	Payload[n]	CHKSUM

Sub-ProfileInter-chip Protocol

Flag

	MSB +7							+0 LSB
Flag:	ExtBit	LongFrame	AddrEn	CheckSumEn	0	0	0	0

Sub-Profile

ExtBit	1ExFlagExFlag[n]
LongFrame	116bitLenH65534
AddrEn	1AddrExAddr[n]Inter-chip ProtocolAddrExAddr[1..n]
CheckSumEn	1CHKSUMInter-chip ProtocolCHKSUM

LenLenH

	MSB +7		+0 LSB
Len:	Len[7..0]		
LenH:	Len[15..8]		

PayloadCMD

Len=sizeof(Payload[n])+1

LenLen254Payload

FlagLongFrame1LenHLen16bit65534

> ⚠
>
> Len/LenH0

CMD

	MSB +7		+0 LSB
CMD:	CMD[7..0]		

Sub-ProfileCMDStandard Profile0x00xF

Payload[n]

	MSB +7		+0 LSB
Payload[0]:	Data[0]		
Payload[1]:	Data[1]		
...			
Payload[n-1]:	Data[n-1]		

Len/LenH

CHKSUM

	MSB +7		+0 LSB
CHKSUM:	CHKSUM[7..0]		

FlagCheckSumEn1

CHKSUM[0..7]=0 xor Packet[0] xor Packet[1] xor…xor Packet[N]

PacketCHKSUMPacket[N]N

CHKSUMCHKSUMCHKSUMSub-Profile

ExFlag[n]

	MSB +7		+0 LSB
ExFlag[N]:	ExtBit	ExFlagN[6..0]	

FlagExtBit1ExFlagSub-Profile
ExFlagExtBitExFlagExFlag0ExFlagExFlag[N]Sub-Profile

AddrExAddr[N]

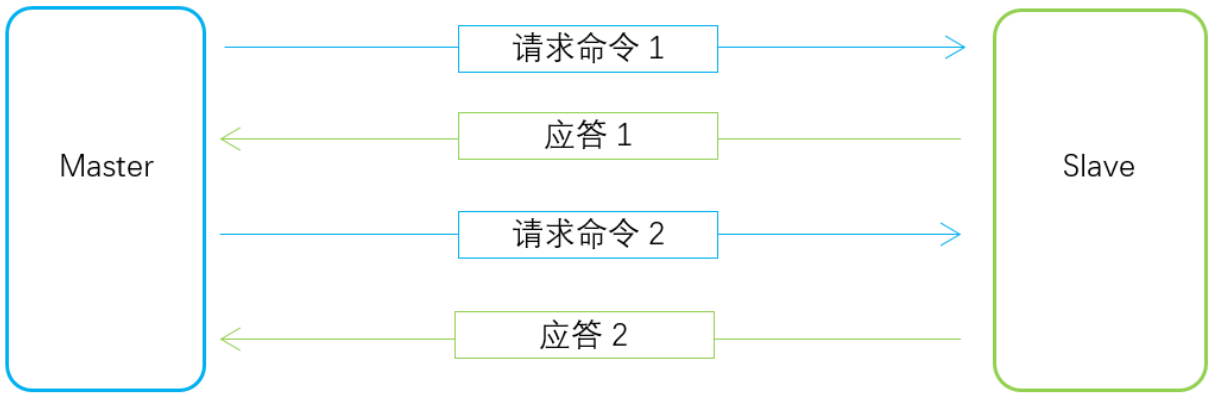
	MSB +7		+0 LSB
Addr:	ExtBit	Addr[6..0]	
ExAddr[N]:	ExtBit	ExAddrN[6..0]	

FlagAddrEn1AddrSub-Profile
Addr7bit0-127AddrExtBit1ExAddr[N]
AddrExAddr[N]ExtBit1ExAddrExtBit0

Standard Profile

Inter-chip ProtocolStandard ProfileCMD

Standard Profile2
(Master)(Slave)
Standard ProfileMasterSlaveFlag
MasterSlave
MasterSlave



Flag

	MSB +7		+0 LSB

Standard:	0	0	0	1	0	0	0	0
Long Frame:	0	1	0	1	0	0	0	0

Standard Profile2Flag

	Flag	
Standard	0x10	Payload254CHKSUM
Long Frame	0x50	Payload65534CHKSUM

Standard ProfileStandardLong FrameMasterSlaveStandardLong Frame

	u8	u8	u8	u8[n]	u8
Standard:	0x10	Len[7..0]	CMD	Payload[n]	CHKSUM
	u8	u16	u8	u8[n]	u8
Long Frame:	0x50	Len[15..0]	CMD	Payload[n]	CHKSUM

Master2CMDSlave
Standard ProfileCMD

CMD	
0x00	
0x01	ECHOSlave
0x02 – 0x0F	

CMD

	u8	u8	u8	u8[n]	u8
Standard:	0x10	Len[7..0]	Resp	Payload[n]	CHKSUM
	u8	u16	u8	u8[n]	u8
Long Frame:	0x50	Len[15..0]	Resp	Payload[n]	CHKSUM

Slave2CMDSlave

Resp(CMD)	
0x00	
0x01	ECHO
0x02	<OK>
0x03	<Error>
0xFF	<Invalid>

Payload[n]SlaveMasterResp<OK>Paylaod[N]Slave

Resp<Error> (0x03)<Invalid>(0xFF)Payload[N]16bitStandard ProfileSlave

0x40	

0x20	Slave
0x10	Slave
0x8000	CMDSlave
0x8001	
0x8002	

SLAMWARE Core

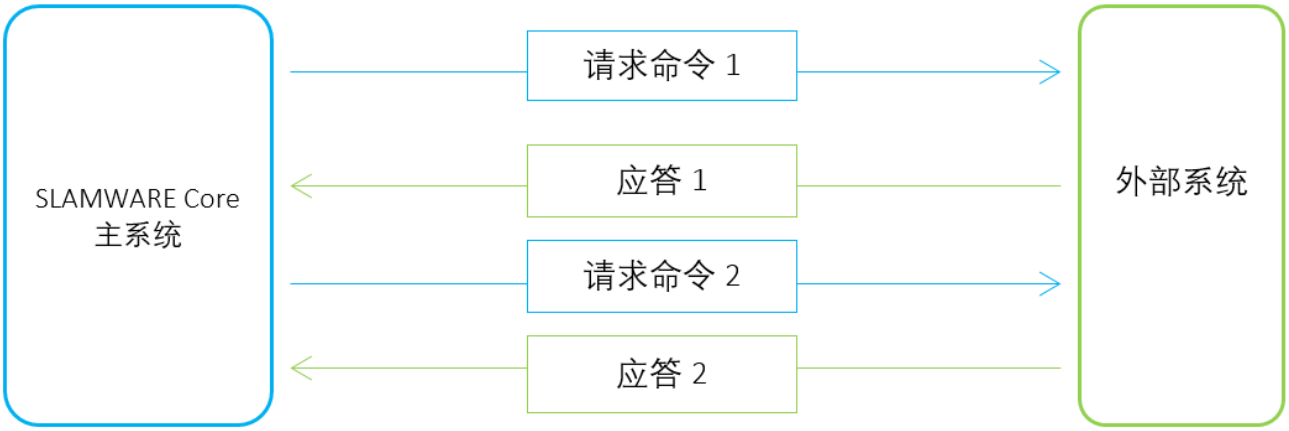
SLAMWARE Core Ctrl Bus

SLAMWARE Core Ctrl BusInter-chip
SLAMWARE Core Ctrl Bus0xF8/Inter-chip

		SLAMWARE Core Ctrl Bus			
(0x50)	u16	0xF8	u8	...	u8

		SLAMWARE Core Ctrl Bus			
(0x10)	u8	0xF8	u8	...	u8

SLAMWARE CoreCtrl BusSLAMWARE Core



SLAMWARE CoreSLAMWARE Core
SLAMWARE Core
SLAMWARE CoreSLAMWARE Core#CMDSLAMWARE Core

SLAMWARE CORE

SLAMWARE CoreSLAMWARE Core Ctrl Bus*SLAMWARE Core

(CONNECT_BASE) *	0x10	
(GET_BASE_CONF) *	0x20	

(GET_BINARY_CONF)	0x21	0x20SLAMWARE CoreNot Support0x20
(GET_BASE_STATUS) *	0x30	
(GET_BASE_MOTOR_DATA) *	0x31	
(GET_BASE_SENSOR_DATA) *	0x32	
(GET_BASE_BUMPER_DATA) *	0x33	
(GET_AUTO_HOME_DATA) *	0x34	
GET_AUXILIARY_ANCHOR	0x35	
(SET_BASE_MOTOR) *	0x40	
Deadreckon SET_V_AND_GET_DEADRECKON *	0x41	Deadreckon
POLL_BASE_CMD *	0x50	
POLL_BASE_ANS_CMD *	0x5F	
SLAMWARE Core(SEND_EVENT) *	0x60	SLAMWARE Core
(HEALTH_MGMT) *	0x90	

SET_V_AND_GET_DEADRECKONGET_BASE_MOTOR_DATASET_BASE_MOTOR

1. slamware SET_V_AND_GET_DEADRECKON, SET_BASE_MOTORGET_BASE_MOTOR_DATA
2. GET_BASE_MOTOR_DATASET_BASE_MOTORSET_V_AND_GET_DEADRECKON

(CONNECT_BASE)

SLAMWARE CoreSLAMWARE Core

0x10	u8			
<OK>	u8[12]	u16	u16	u32[3]

	u8	
	u8[12]	12
	u16	
	u16	
	u32[3]	12

(GET_BASE_CONF)

SLAMWARE CoreSLAMWARE Core

0x20	

<OK>	u8	u32	u8	u8	pose[8]
				u8	pose[8]

	u8	
	u32	mm,Q8
	u8	
	u8	8
	pos[8]	
	u8	8
	pos[8]	

⚠

0x2080x21 Binary Config16

0x00	
0x01	

0x00	

pos

X	s32	XQ8
Y	s32	YQ8
Z	s32	ZQ8
	u32	Q8

(GET_BINARY_CONF)

0x20SLAMWARE CoreNot Support0x20

0x20

0x21	
<OK>	u8*n

SDKSlamware Configuration Tool.cslamware_config

> ⚠
>
> 0x21 Binary Config 0x200

(GET_BASE_STATUS)

SLAMWARE CoreSLAMWARE CoreSLAMWARE CoreSLAMWARE Core

0x30		
<OK>	u8	u8

	u8	0-100
	u8	300000011 500000101

(GET_BASE_MOTOR_DATA)

SLAMWARE CoreSLAMWARE Core

0x31		
<OK>	s32	s32

	s32	mm
	s32	mm

> ⚠

(GET_BASE_SENSOR_DATA)

SLAMWARE Core
SLAMWARE Core

0x32	
<OK>	u32[16]

	u32[16]	mmQ1616 0x2088 0x21 Binary Config1616

(GET_BASE_BUMPER_DATA)

SLAMWARE Core
SLAMWARE Core

0x33	
<OK>	u32

	u32	bit 1032

(GET_AUTO_HOME_DATA)

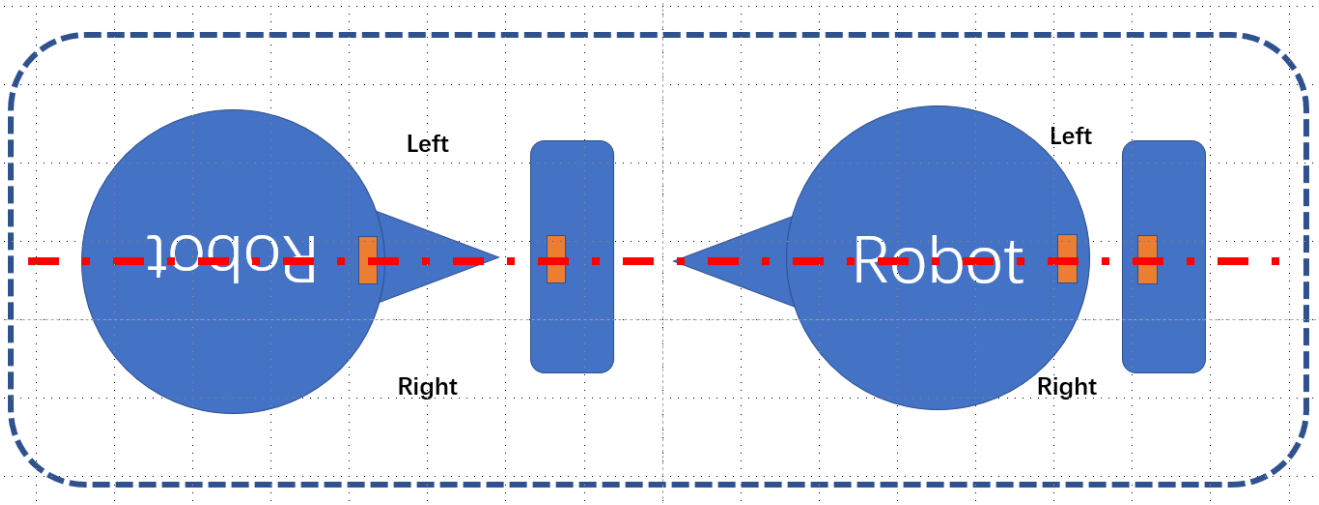
SLAMWARE CoreBeacon

0x34	u8

<OK>	u8	u8	u8

0x34

	u8	SLAMWARE Core: 0 – 0not support0x8000
	u8	 3012
	u8	 3012
	u8[n]	 101 Data[1]=(10)(11)



(SET_BASE_MOTOR)

SLAMWARE CoreSLAMWARE Core

0x40	s32[4]
<OK>	

	s32[4]	mm/s

Deadreckon(SET_V_AND_GET_DEADRECKON)

0x41	Xs32	Ys32	s32
<OK>	Xs32	Ys32	s32

X	s32	XQ16
Y	s32	YQ16
	s32	Q16
X	s32	XmmQ16
Y	s32	YmmQ16
	s32	Q16

⚠

, x

1. dx, dy, dyaw
2. vx, vy, omegavl, vy
3.

1.

dl		m
dr		m
dx		m
dy		m
dyaw		rad

2.

dyaw=(dr-dl)/2R

dx=cos(dyaw)×(dl+dr)/2

dy=sin(dyaw)×(dl+dr)/2

⚠

Rm

3.

base_set_velocity_request

```
float d_yaw = (d_dist_r_mm_f - d_dist_l_mm_f)/2.0f/robot_radius_mm;
float displacement = (d_dist_l_mm_f + d_dist_r_mm_f)/2.0f;
float dx = cos(d_yaw)*displacement;
float dy = sin(d_yaw)*displacement;
ans_pkt->base_dx_mm_q16 = (_32)(dx*(1<<16))
ans_pkt->base_dy_mm_q16 = (_32)(dy*(1<<16))
ans_pkt->base_dtheta_degree_q16 = (_32)(d_yaw/M_PIF*180*(1<<16))
```

1.

vx		m/s
vy	, 0	m/s
omega		rad/s
vl		m/s
vr		m/s

2. ()
   vl=vx-omega*R
   vr=vx+omega*R
3.

base_set_velocity_request

```
base_set_velocity_request_t *req = (base_set_velocity_request_t*)request->payload;
float speed_l_mm = (float)req->velocity_x_q16 * 1000.0 / (1 << 16);
float speed_r_mm = speed_l_mm;
float line_speed_mm = (float)req->angular_velocity_q16 / (1 << 16) * robot_radius_mm;
speed_l_mm -= line_speed_mm;
speed_r_mm += line_speed_mm;
```

(POLL_BASE_CMD)

SLAMWARE CoreSLAMWARE CoreSLAMWARE CoreSLAMWARE Core#CMDSLAMWARE Core#BUSYSLAMWARE CoreSLAMWARE Core
SLAMWARE CoreSLAMWARE Core#BUSY#CMD

0x50	
<OK>	u8

	u8	SLAMWARE Core

SLAMWARE Core

0x51	GET_INFO	SLAMWARE CORE
0x52	RESET_WIFI	SLAMWARE CORE
0x53	FW_UPGRADING	SLAMWARE CORE
0x80	START_SWEEP	()
0x81	STOP_SWEEP	()
0x82	SPOT_SWEEP	()
0x90	GET_HEALTH	
0xA0	MOVE_FORWARD	
0xA1	MOVE_BACKWARD	
0xA2	TURN_LEFT	
0xA3	TURN_RIGHT	
0xAF	CANCEL_ACTION	
0xB0	GET_AUXILIARY_ANCHOR	

(POLL_BASE_ANS_CMD)

SLAMWARE Core
SLAMWARE Core

0x5F	
<OK>	u8

	u8	SLAMWARE CORE

SLAMWARE CORE(SEND_EVENT)

SLAMWARE CORE
SLAMWARE CORE

0x60	u8
<OK>	

	u8	SLAMWARE CORE

0x61	LIDAR_CONN_FAIL	LIDAR
0x62	LIDAR_RAMPUP_FAIL	LIDAR
0x63	SYSTEM_UP_OK	
0x64	FIRMWARE_UPDATE	
0x65	CORE_DISCONNECT	
0x66	FIRMWARE_UPDATE_OK	
0x80	START_SWEEP	()
0x81	END_SWEEP	()

(HEALTH_MGMT)

SLAMWARE CORE

0x01	HEALTH_GET_HEALTH	
0x02	HEALTH_GET_ERROR	
0x03	HEALTH_CLEAR_ERROR	

HEALTH_GET_HEALTH

0x90	0x01	
<OK>	Health_flagu8	Error_countu8

Health_flag	u8	
Error_count	u8	

Health flag

Health Flag	Bit	
	[7:3]	
FATAL	[2]	FATAL 0 = Has no FATAL 1 = Has FATAL

ERROR	[1]	ERROR 0 = Has no ERROR 1 = Has ERROR
WARN	[0]	WARN 0 = Has no WARN 1 = Has WARN

HEALTH_GET_ERROR

0x90	0x02	error idu8
<OK>	error_codeu32	error_messageu8[32]

error_id	u8	errorerror count.
error_code	u32	
error_message	u8[32]	.

Error code

Error Code	Bit	
	[31:24]	0x01 = warn 0x02 = error 0x03 = fatal
	[23:16]	0 = USER 1 = SYSTEM 2 = POWER 3 = MOTION 4 = SENSOR
	[15:8]	
	[7:0]	

Sensor Errors

```
// sensor errors
#define BASE_SENSOR_FATAL_CONTROLLER_DOWN        (SLAMWARECORE_HEALTH_ERROR_FATAL | BASE_COMPONENT_SENSOR |
0x0000u)
#define BASE_SENSOR_WARN_BUMPER_DOWN               (SLAMWARECORE_HEALTH_ERROR_WARN  | BASE_COMPONENT_SENSOR
| 0x0100u)
#define BASE_SENSOR_ERROR_BUMPER_DOWN               (SLAMWARECORE_HEALTH_ERROR_ERROR |
BASE_COMPONENT_SENSOR | 0x0100u)
#define BASE_SENSOR_FATAL_BUMPER_DOWN               (SLAMWARECORE_HEALTH_ERROR_FATAL |
BASE_COMPONENT_SENSOR | 0x0100u)
#define BASE_SENSOR_WARN_CLIFF_DOWN                       (SLAMWARECORE_HEALTH_ERROR_WARN  |
BASE_COMPONENT_SENSOR | 0x0200u)
#define BASE_SENSOR_ERROR_CLIFF_DOWN               (SLAMWARECORE_HEALTH_ERROR_ERROR | BASE_COMPONENT_SENSOR
| 0x0200u)
#define BASE_SENSOR_FATAL_CLIFF_DOWN               (SLAMWARECORE_HEALTH_ERROR_FATAL | BASE_COMPONENT_SENSOR
| 0x0200u)
#define BASE_SENSOR_WARN_SONAR_DOWN                       (SLAMWARECORE_HEALTH_ERROR_WARN  |
BASE_COMPONENT_SENSOR | 0x0300u)
```

HEALTH_CLEAR_ERROR

0x90	0x03	error codeu32
<OK>		

GET_AUXILIARY_ANCHOR

SLAMWARE CORE

0x35		
<OK>	Flagu8	Anchorsu8[*]

Flag	u8	
Anchors	u8[*]	AnchorInfo

Flag

Flag	Bit	
	[7:5]	0x00 = UWB
	[4]	0 = 1 =

	[3:0]	

AnchorInfo

AnchorInfo[0]	0AnchorInfo	n
AnchorInfo[1]	1AnchorInfo	
...		
AnchorInfo[n]	nAnchorInfo	

AnchorInfo

AnchorInfo		
id	u16	Anchorid
distance	u16	Anchor
Max error	u8	