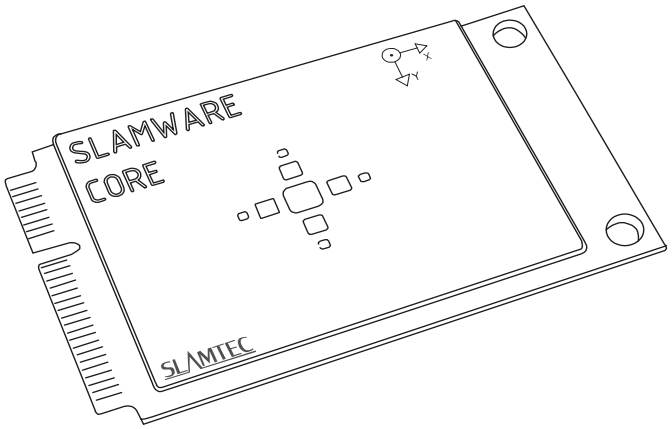
SLAMTEC

SLAMWARE

Autonomous Robot Localization and Navigation Solution

Ctrl Bus Communication Protocol



2016-09-12 rev.1.8

www.slamtec.com

Shanghai Slamtec.Co.,Ltd

Contents

SLAMTEC

CONTENTS **1**

INTRODUCTION **3**

FUNCTION OVERVIEW **4**

Inside Module Structure & Usage 4

INTER-CHIP PROTOCOL SPECIFICATION **5**

Inter-chip Communication Protocol Introduction 5

Inter-chip Communication Protocol Specification 6

Standard Profile 9

PROTOCOL COMMAND **14**

SLAMWARE CORE Ctrl Bus Communication Protocol 14

SLAMWARE CORE Request Command 15

External System Connection Request(CONNECT_BASE) 16

External System Getting Configuration Request(GET_BASE_CONF) 17

External System Getting Binary Configuration Request(GET_BINARY_CONF) 19

External System Getting Base Status Request(GET_BASE_STATUS) 20

External System Getting Base Motor Status Request(GET_BASE_MOTOR_DATA) 20

External System Getting Base Sensor Data Request(GET_BASE_SENSOR_DATA) 21

External System Getting Bump Sensor Data Request(GET_BASE_BUMPER_DATA) 22

External System Getting Auto Home Data Request(GET_AUTO_HOME_DATA) 23

External System Speed Control Request(SET_BASE_MOTOR) 24

External System Set Velocity Request(SET_V_AND_GET_DEADRECKON) 24

External System Poll Command Request(POLL_BASE_CMD) 27

External System Poll Command Response Request(POLL_BASE_ANS_CMD) 29

SLAMWARE CORE System Event Notice Command(SEND_EVENT) 29

External System Health Management Request(HEALTH_MGMT) 30

REVISION HISTORY **32**

APPENDIX **33**

Image and Table Index 33

The SLAMTEC SLAMWARE CORE module is connected with external system via ctrl bus low speed serial bus. With the ctrl bus protocol, SLAMWARE CORE can obtain the status parameter of external system such as power information, charging status, motor speed, sensor status, etc. At the same time, the SLAMWARE CORE controls the motor motion of the External system via the control bus protocol to make the indoor localization and navigation full functional.

The control bus is running on serial port with 115200bps Baud rate. Therefore, the external system needs to meet the requirement of the above bus protocol to make SLAMWARE CORE work normally.

SLAMWARE CORE Communication Ctrl Bus Protocol adopts the extended mode of Inter-chip Communication Protocol. The latter is designed by SLAMTEC for communication between underlying electronic system such as microcontroller and microcontroller, microcontroller and host system. microcontroller and devices. It transfers the data in data packet on communication medium. The data packet is embedded with data checking mechanism and frame synchronization mechanism to ensure reliable communication link. Currently, the protocol is widely used in the robot system, sensors and module devices of SLAMTEC. It is also supported in USART, USB, I^2C, TCP Ethernet.

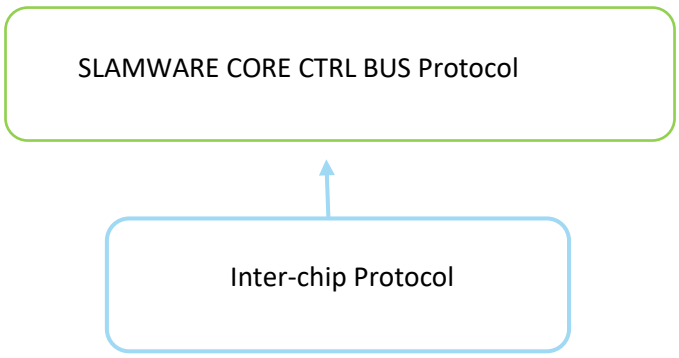


Figure 1-1 SLAMWARE CORE CTRL BUS Protocol and Inter-chip Protocol

Inside Module Structure & Usage

The communication connection of SLAMWARE CORE and external system is described as below:

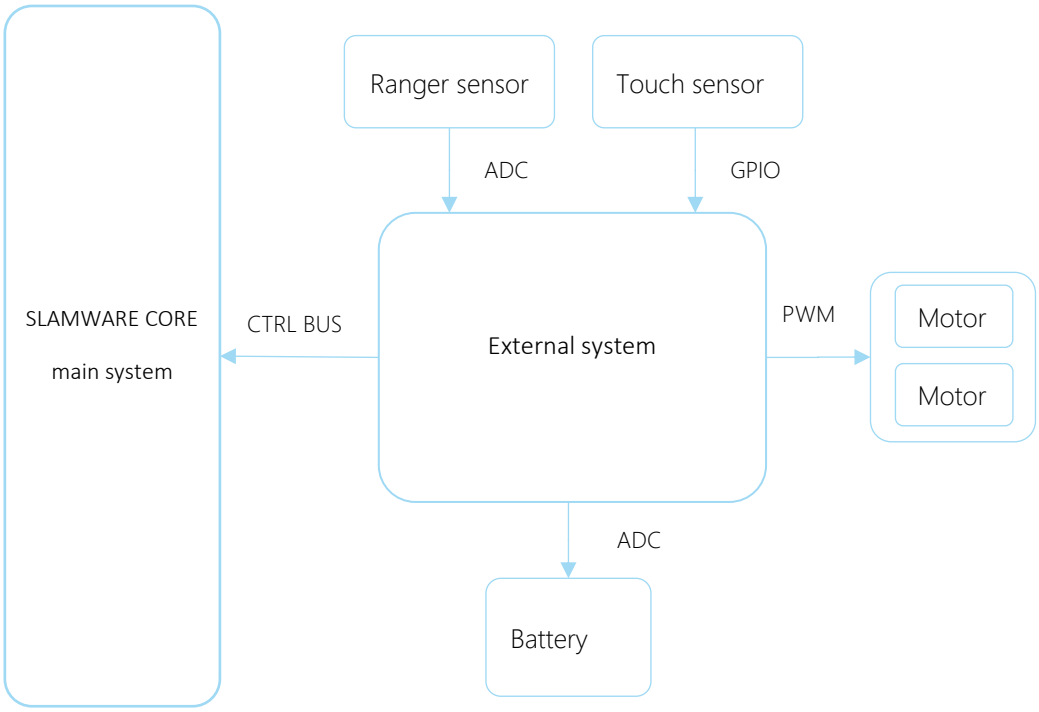


Figure 2-1 SLAMWARE CORE Module Structure

Inter-chip Communication Protocol Introduction

Communication Features

The data communication media assumed in the inter-chip protocol is the network organized by peer-to-peer or bus mode. Only two communication peers at most can communication with each other at the same time. But if the communication media network permits implementing broadcast, a communication peer can send message to multiple other peers in one-way mode at the same time.

The communication media itself can be the channel based on byte stream mode such as USART and TCP Ethernet, or the channel based on data packet such as USB and I^2C. The communication media is not required to implement the synchronization mechanism of data packet, since the inter-chip protocol itself can achieve the synchronization via the synchronization signal in the header of each data packet

When communication occurs, the message transferring on the communication media is always in the data packet form with fixed format. Every data packet can carry data with different length, but there is data with the same length and data structure definition in the header of the data packet. The data in the header of every data packet is used for describing the information of the data packet itself and the expected process mode of the receiving end for handling the data.

Data Transfer Sequence

When transferring data, the inter-chip protocol requires the channel to follow the FIFO rules and transfer the common data structure in little endian format. The common data structure is defined for every data packet in the protocol such as the descriptive data in the header.

Protocol Sub-profile

Since requirements vary in different applications, inter-chip contains several specific sub-profile. Different sub-profiles have defined different network communication mode and header data structure. In the current implementation, standard profile is the default sub-module of inter-chip protocol. It adopts the communication network with request-response mechanism.

Inter-chip Communication Protocol Specification

Data Packet Structure Overview

Every data packet defined by Inter-chip protocol follows the format definition as below:

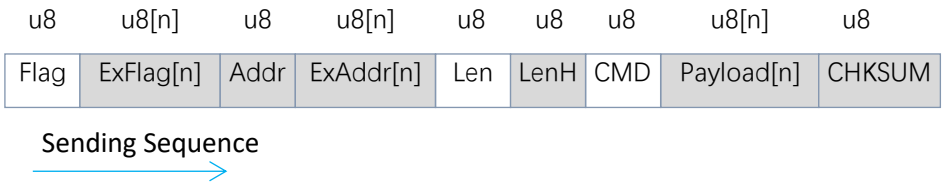


Figure 3-1 Inter-chip Protocol Data Packet Format Definition

In the above format, the parts with gray color are optional fields. Different sub-profile may adopt some of those optional fields. The parts with white color are common in all data packets defined by inter-chip protocol. Detailed description about every filed will be introduced in the following sections.

Flag Field

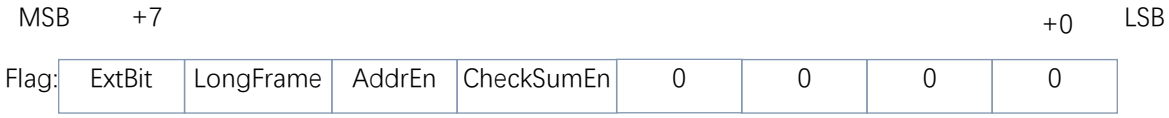


Figure 3-2 Inter-chip Protocol Flag Field Format Definition

This field is the start identifier for a data packet and takes up 1 byte. The definition of this field varies in different sub-profiles. And detailed description about every flag bit are as below:

Field Name	Description
ExtBit	When it is set as 1, it indicates the next is ExFlag field, that is, ExFlag[n] will display in this data packet and it will carry more flag byte information.
LongFrame	When it is set as 1, it indicates this data packet will adopt field in 16 bit and the LenH field will be contained in this data packet. This data packet can carry 65534 bytes' data at most.
AddrEn	When it is set as 1, it indicates Addr will be contained in the data packet, and ExAddr[n] may also be contained. Inter-chip Protocol Stack may parse the address in Addr and ExAddr [1. n] for processing.
CheckSumEn	When it is set as 1. the CHKSUM field will be contained in the end of this data packet for data checking. The Inter-chip Protocol Stack is required to check the data of this data packet and comparing with CHKSUM.

Figure 3-3 Inter-chip Protocol Flag Field Flag Bit Description

Len & LenH Field

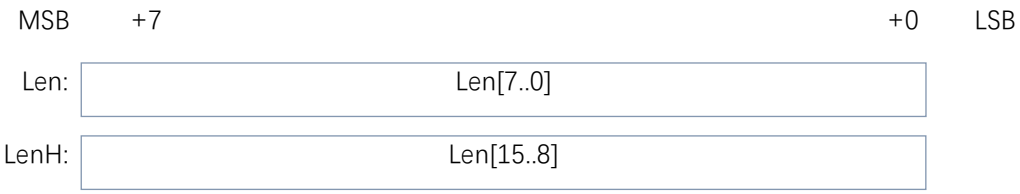


Figure 3-4 Inter-chip Protocol Len and LenH Field Format Definition

This field indicates the length information for the data of the data packet. It is defined as the sum of payload field length and CMD field length. That is:

Len= sizeof(Payload[n]) +1

Since the Len field is 1 byte, when there is only Len field in the packet, the packet can carry payload data with 254 bytes as a maximum.

When the LongFrame is set as 1 in the Flag field, the LenH field will be contained in the data packet and connected with Len field as a 16bit joint field. Then this data packet can carry data with 65534 bytes as a maximum.

Note: when Len/LenH field is set as 0, the data packet will be regarded as invalid and discarded by the protocol stack.

CMD Field

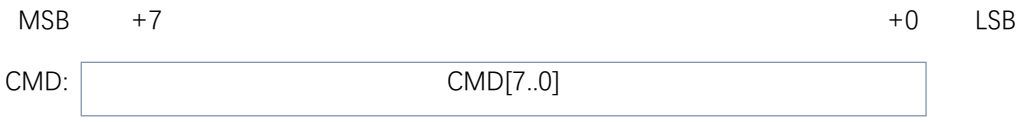


Figure 3-5 Inter-chip Protocol CMD Field Format Definition

This field carries the command information to the protocol stack or the receiving program. That command information indicates how the protocol stack or the receiving program should process the data packet. Different sub-profiles will keep part of CMD value for implementing the communication mechanism of the protocol itself. For example, in standard profile, 0x0 to 0xF are taken up by the protocol stack. For specific definition, please refer to the following sections.

Payload[n] Field

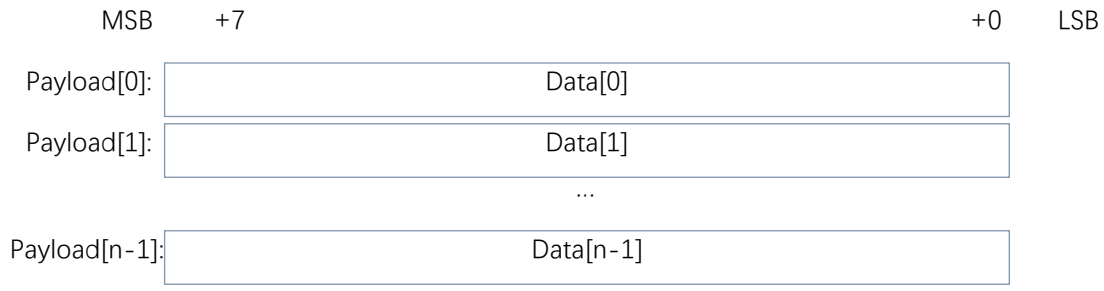


Figure 3-6 Inter-chip Protocol Payload[n] Field Format Definition

This data field records the expected valid message payload from send side. The data length is indicated in Len/LenH field. If there is no message to transfer from send side, this field will be omitted.

CHKSUM Field

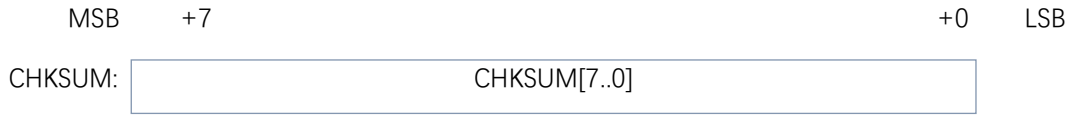


Figure 3-7 Inter-chip Protocol CHKSUM Field Format Definition

When the CheckSumEn of Flag is set as 1, this field will be added in the end of the data packet and used for recording the checksum of correct data in the data packet. The calculating method is as following:

$$\text{CHKSUM}[0..7] = 0\ xor\ \text{Packet}[0]\ xor\ \text{Packet}[1]\ xor\ \ldots xor\ Packet[N]$$

In the above formula, Packet indicates the rest part of the current data packet without CHKSUM field. Packet[n] indicates the N^{th} byte of the data packet which will be sending.

When starting using the CHKSUM field, after receiving a data packet, the receiving protocol stack will re-calculate a new CHKSUM and compare it with the original CHKSUM field in the data packet. If they are not matched, the protocol stack will discard the current data packet and process it according to the operation defined by the sub-profile.

ExFlag[N] Field

Figure 3-8 Inter-chip Protocol ExFlag[N] Field Format Definition

When the ExtBit of the Flag is set as 1, ExFlag field with one or more bytes will be contained in the data packet. This field contains the extra content defined by sub-profile or the extra flag bit used for describing the information of the data packet itself. User application program can also add extra information in this field.

The ExtBit in every ExFlag bit indicates whether there is an extra ExFlag is following the current Exflag. When it is set as 0, it indicates the current ExFlag is the end of the whole ExFlag[N]. Please refer to sub-profile definition for details about this field.

Addr & ExAddr[N] Field

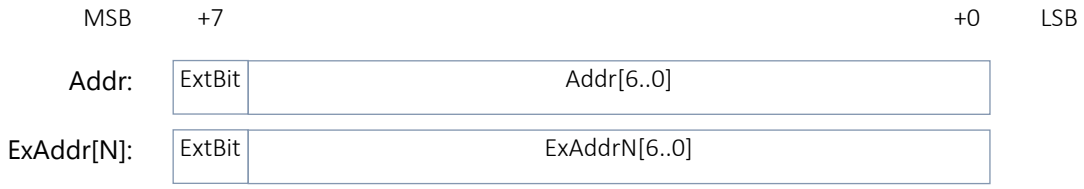


Figure 3-9 Inter-chip Protocol Addr & ExAddr[N] Field Format Definition

When the AddrEn bit in the Flag is set as 1, an Addr field is contained in the data packet to record the target device address that the data packet will be sent to. In different sub-profile implementations, the protocol stack will process the data in this field differently.

In Addr field, 7 bits are taken up for representing 0-127 device address number. If longer device address is required, please set the ExtBit of Addr field as 1, then more address data bits will be used and kept in the following ExAddr[N] field.

When the ExtBit bit in Addr and every ExAddr[N] is set as 1, it indicates an ExAddr byte is following the current byte until the ExtBit is 0.

Standard Profile

The standard profile of Inter-chip protocol defines a reliable peer-to-peer interchange network based on the request and response communication mode and reserves some CMD value for implementing communication mechanism in the protocol stack.

Communication Mode and Terms

Standard Profile requires only two peers communicates with each other at the same time in the network. One peer always sends data packet first to the other

peer, and the other peer always responds after receiving the data packet and sends a corresponding response data packet.

The peer always sending data packet first is defined as Master and the peer always receiving data packet first then sending response data packet is defined as Slave.

Standard Profile adopts the same format in the data packet sent by Master and Slave, that is, the definitions of Flags are the same. But since they are different communication roles, some field of the data packet may have different definitions.

For the data packet sent by Master is defined as the request command data packet (request command in short), and the data packet sent by Slave is defined as response data packet (response in short).

Request command and response must come in pairs. Every response is always matched with a request command. And Master must send the next request command after receiving the response data packet of the last request command from Slave or the communication timing out.

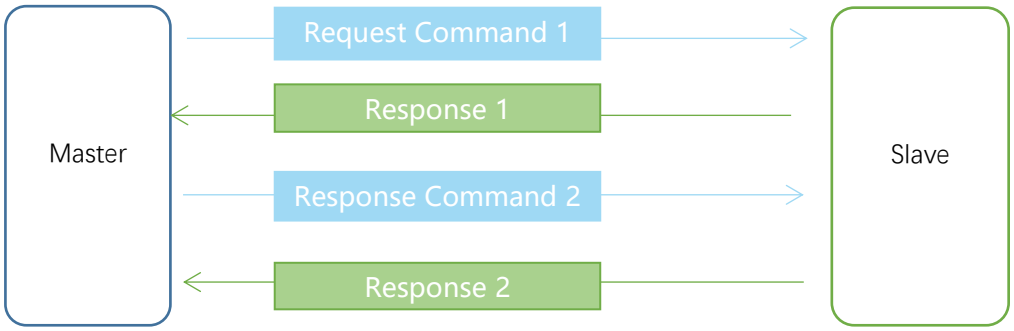


Figure 3-10 The Communication Between Master & Slave Based on Inter-chip Protocol

Flag Field Definition

MSB	+7							+0 LSB
Standard:	0	0	0	1	0	0	0	0
Long Frame:	0	1	0	1	0	0	0	0

Figure 3-11 Inter-chip Protocol Flag Field Format Definition

Standard Profile permits the data packet format with the following two Flag fields.

Packet Type	Flag Value	Description
Standard	0x10	Payload data with 254 bytes as a maximum, no address information, with CHKSUM.
Long Frame	0x50	Payload data with 65534 bytes as a maximum, no address information, with CHKSUM.

Figure 3-12 Inter-chip Protocol Flag Data Packet Format Description

The protocol stack adopted standard profile must support the data packet with standard format. When the protocol stack supports data packet in long frame format, during the communication process, Master and Slave can choose to send standard or long frame data packet according to their actual requirement.

Request Command Data Packet

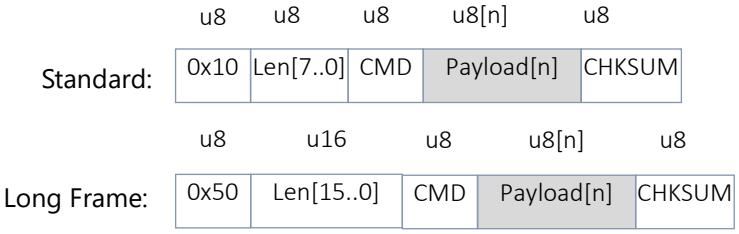


Figure 3-13 Inter-chip Protocol Request Command Data Packet Definition

Request command data packet sent from Master adopts the above two fixed data packet structure format. The most value of the CMD field can be used by application. As for invoking the specific feature or data request of Slave, please refer to the corresponding device documents for detailed value information.

The protocol stack conforming the standard profile will reserve the CMD field value in the following table for processing the task of the protocol stack itself.

Reserved CMD Value	Description
0x00	Forcing synchronization frame, used for achieving the synchronization of communication channel.
0x01	ECHO frame, used for test the Slave communication.
0x02 – 0x0F	Reserved.

Figure 3-14 Inter-chip Protocol Reserved CMD Description

The applications should not use the above reserved CMD field.

Standard profile doesn't make assumptions for the usage of Payload[n]. But specific Slave will ask to make assumptions for the data format of Payload[n] when different CMD field value defined.

Response Data Packet

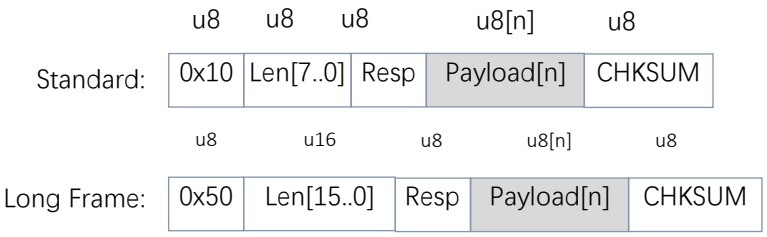


Figure 3-15 Inter-chip Protocol Response Data Packet Format Definition

Response data packet sent from Slave should follow the above two formats. And the CMD field is used for responding the processing status of forgoing request command received by Slave. Its value should be as in the following table.

Reserved Resp(CMD)	Description
0x00	The response packet of forcing synchronization frame.
0x01	The response packet of ECHO frame.
0x02	<OK> indicates the last request command is executed correctly or there is no problem in the protocol stack.
0x03	<Error> indicates error occurs when executing the last request command
0xFF	<Invalid> indicates the last request command is not received completely or the data packet structure is not correct.

Figure 3-16 Inter-chip Protocol Reserved Resp Description

The Payload[n] field in the response data packet keeps the Slave's executed result for forgoing request command sent from Master. When the Resp in the current response data packet is set as <OK>, the specific data definition contained in the Payload[N] is related with corresponding request command. Please refer to the Slave device manual for details.

When Resp is <Error> (0x03) or <Invalid>(0xFF), Payload[N] will keep a 16bit error code to describe the error. The Standard Profile requires the protocol stack to reserve the following error code value. Please refer to the following table for

detailed definition of those error code. The application on Slave can also define more error code.

Error Code	Description
0x40	The checksum of request command data packet is not matched.
0x20	The length of the request command data packet exceeds the limits which the current Slave protocol stack buffer could support.
0x10	The Slave protocol stack channel doesn't finish the synchronization.
0x8000	The CMD field of the request command data packet is not supported by the Slave.
0x8001	Format error occurs when the request command is being executed. For example, the request parameters don't conform the requirements.
0x8002	Operational error occurred when the request command is being executed. For example, executing process cannot continue due to problems.

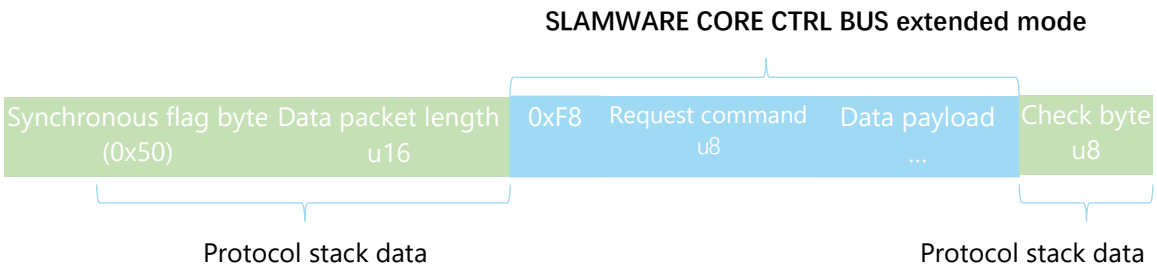
Figure 3-17 Inter-chip Protocol Error Code Description

SLAMWARE CORE Ctrl Bus Communication Protocol

The SLAMWARE CORE CTRL BUS communication protocol adopts the extended mode of inter-chip communication protocol.

Its data packet type field is 0xF8 marked with SLAMWARE CORE CTRL BUS extended mode. In this mode, data payload is in request/response mode and response data packet is in the same format with standard inter-chip communication protocol. There are two formats according to the length of data packet.

Format A:



Format B:

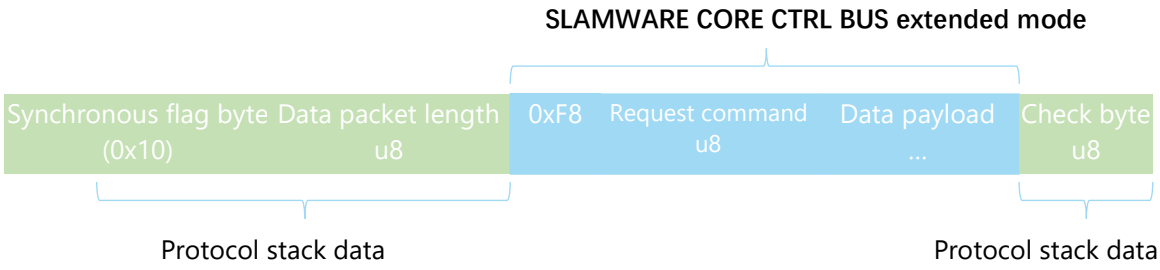


Figure 4-1 Request/Response Communication Mode

The communication protocol between SLAMWARE CORE and external system is one-way mode and always started by SLAMWARE CORE to send data packet request.

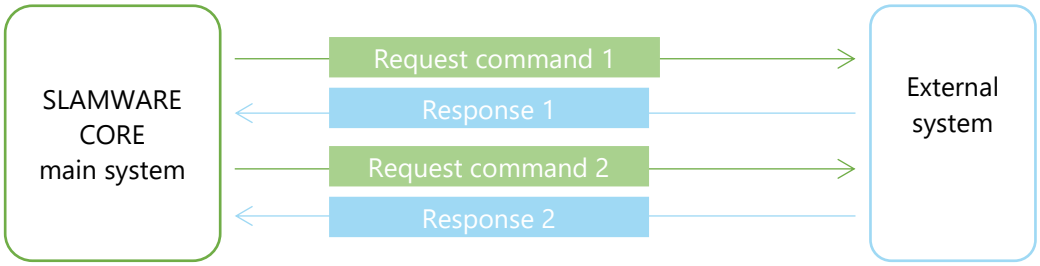


Figure 4-2 Request/Response Communication Mode

The external system should send response to the request command of SLAMWARE CORE in time or the later would think external system is disconnected and stops working.

Request command and response come in pair. Each response is matched with a request command. Only after receiving the response data packet or the communication timeout notice from external system for the last request command sent by SLAMWARE CORE, SLAMWARE CORE would send the next request command.

SLAMWARE CORE can receive control command from external system. However, restricted by one-way communication mode, the external system can send control command only when SLAMWARE CORE termly send the checking request of control command to external system. But the external system can notify SLAMWARE CORE to send out control command checking request quicker via #CMD signal.

SLAMWARE CORE Request Command

SLAMWARE CORE indoor localization and navigation module will send the following command requests and external system can send response to them accordingly, or SLAMWARE CORE CTRL BUS protocol stack will provide a default response. Request commands with * mark are required to response to ensure SLAMWARE CORE working normally.

Request command	Command code	Description
External system connection request (CONNECT_BASE) *	0x10	Request for connecting external system and the latter will send response with hardware and firmware information to show connecting success.
External system parameter obtaining request (GET_BASE_CONF) *	0x20	Request for the configuration and parameter of external system, including dimensions, radius, wheels type, sensor location and angle.
External system configuration obtaining request(GET_BINARY_CONF)	0x21	Request for external system configuring parameters. This command, with advanced extensibility, is a replacement for the command 0x20. SLAMWARE CORE will try this

		command first, if the return value is Not Support, then the command 0x20 will be invoked for obtaining the external system configuration.
External system status obtaining request (GET_BASE_STATUS) *	0x30	Request for obtaining status of external system, including remaining capacity of battery and charging status.
External system wheels status obtaining request (GET_BASE_MOTOR_DATA) *	0x31	Request for the accumulated running distance of the wheels from external system.
External system range sensor data obtaining request (GET_BASE_SENSOR_DATA) *	0x32	Request for range sensor data from external system
External system bump sensor data obtaining request (GET_BASE_BUMPER_DATA) *	0x33	Request for bump sensor data from external system.
External system self-charging data obtaining request (GET_AUTO_HOME_DATA)	0x34	Request for self-charging data from external system.
External system motion control request(SET_BASE_MOTOR) *	0x40	Request external system to control wheels' motion
External system set velocity request (SET_V_AND_GET_DEARECKON) *	0x41	Set the velocity of external system and get deadreckon data of last period
External system command checking request(POLL_BASE_CMD) *	0x50	Request for checking whether there is command sent out from external system.
External system command response request(POLL_BASE_ANS_CMD) *	0x5F	Response to the command request from external system.
SLAMWARE CORE system event notice command(SEND_EVENT) *	0x60	Notify the external system of SLAMWARE CORE event.
SLAMWARE CORE health management command(HEALTH_MGMT) *	0x90	Get the health status of external system

Figure 4-3 SLAMWARE CORE Request Command List

External System Connection Request(CONNECT_BASE)

SLAMWARE CORE indoor localization and navigation module will continually send connection request to external system to ensure that the latter is working normally. Only after external system sends response to the above request, SLAMWARE CORE will start working.

- Request data packet:

Command code	Payload data
0x10	Protocol version u8

- Response data packet:

Response code	Payload data			
<OK>	Mode u8[12]	Firmware version u16	Hardware version u16	Serial No. u32[3]

Data field	Type	Description
Protocol version	u8	The version information of the protocol. If not matched, External system would send response with error message.
Mode	u8[12]	Mode of external system. Up to 12 characters.
Firmware version	u16	Firmware version of external system.
Hardware version	u16	Hardware version of external system.
Serial No.	u32[3]	Serial No. of external system. Up to 12 characters.

Figure 4-4 External System Connection Request Field Description

External System Getting Configuration Request(GET_BASE_CONF)

After connecting with external system, SLAMWARE CORE indoor localization and navigation module will send out getting configuration request to get the basic structure information including dimensions, radius, wheels set type and the information for sensor installation location etc. The current version only supports for rounded and square dimensions and two-wheeled differential motion wheel set.

- Request data packet:

Command code
0x20

- Response data packet:

Response code	Payload data		
<OK>	Dimensions	Radius	Wheel set type
	u8	u32	u8
	Range sensor number	Range sensor installation location	
	u8	pos[8]	
	Touch sensor number	bump sensor installation location	
	u8	pos[8]	

Data field	Type	Description
Dimension type	u8	Shape of external system. Currently only supporting round and square shape.
Radius	u32	Radius of external system. The units come out in mm, Q8 fix-point decimal.
Wheel set type	u8	Wheel set type of external system. Currently only support two-wheeled differential motion wheel set.
Range sensor number	u8	Range sensor number of external system. Support up to 8 sensor.
Range sensor installation location	pos[8]	The installation location for range sensors of external system.
Bump sensor number	u8	Bump sensor number of external system. Support up to 8 sensors.
Bump sensor installation location	pos[8]	The installation location for bump sensors of external system.

Figure 4-5 External System Parameter Obtaining Request Field Description

Note: when using the 0x20 command, 8 sensors can be supported in the most. When using the 0x21 command, 16 sensors can be supported in the most.

Currently supported shape:

Mode No.	Description
0x00	Rounded
0x01	Square

Figure 4-6 Supported Mechanical Dimension of External System

Currently supported wheel set type:

Mode No.	Description
0x00	Two-wheeled differential drive

Figure 4-7 Supported Wheel Set Type

The pos data of the sensor installation location are as below:

Data field	Type	Description
Distance on X-axis	s32	The distance between sensor core and external system core on X-axis, Q8 fix-point decimal.
Distance on Y-axis	s32	The distance between sensor core and external system core on Y-axis, Q8 fix-point decimal.
Distance on Z-axis	s32	The distance between sensor core and external system core on Z-axis, Q8 fix-point decimal.
Angle	u32	The anticlockwise included angle between sensor core and external system core, Q8 fix-point decimal.

Figure 4-8 Sensor Installation Location Pos data

External System Getting Binary Configuration Request(GET_BINARY_CONF)

This is the upgraded command for the command 0x20. SLAMWARE CORE will invoke this command prior to 0x20 to obtain the configuration data from the external system. If the return value is Not Support, then the command 0x20 will be invoked.

Compared with the command 0x20, this command provides more functions such as the configuration for the installation location of Radar.

- Request data packet:

Command code
0x21

- Response data packet:

Response code	Payload data
<OK>	External system configuration data
	u8*n

The configuration data of the external system is generated automatically by the SLAMWARE Configuration Tool included in the SDK. Please add the generated .c file in the project and put the SLAMWARE config variable as the field data for external system configuration.

Note: when using 0x21 Binary Config command to configure the robot, please set all the value in the 0x20 command as 0.

External System Getting Base Status Request(GET_BASE_STATUS)

SLAMWARE CORE indoor localization and navigation module will continually send connection request to the external system to ensure that the latter is working normally. Only after the external system responses to the above request, SLAMWARE CORE will start working. Then SLAMWARE CORE will keep polling for the status of external system and the later should send response in given time. Or SLAMWARE CORE will think external system is disconnected.

- Request data packet:

Command code
0x30

- Response data packet:

Response code	Payload data	
<OK>	Battery percentage	Battery charging status
	u8	u8

Data field	Type	Description
Battery percentage	u8	Battery percentage of external system. From 0 to 100 %.
Battery charging status	u8	Battery charging status. The variable is a bitmap. The definition from low-order to high-order are: whether in charging status; whether connected with external battery; whether connected with the charging station. When the external system has been connected with charger via cable and the battery is in charging status, the status should be 3(the equal in binary format is 00000011). When the external system is charging via charging station, the status should be 5(the equal in binary format is 00000101).

Figure 4-9 External System Status Obtaining Request Field Description

External System Getting Base Motor Status Request(GET_BASE_MOTOR_DATA)

After starting working normally, SLAMWARE CORE indoor localization and navigation module will continually poll for wheels' status from external system and the later should send response in given time. Or SLAMWARE CORE will think external system is disconnected.

- Request data packet:

Command code
0x31

- Response data packet:

Response code	Payload data	
<OK>	Left motor cumulated distance	Right motor cumulated distance
	s32	s32

Data field	Type	Description
Left motor cumulated distance	s32	The cumulated operating distance of left motor. The units come out in mm.
Right motor cumulated distance	s32	The cumulated operating distance of right motor. The units come out in mm.

Figure 4-10 External System Wheel Status Obtaining Request Field Description

External System Getting Base Sensor Data Request(GET_BASE_SENSOR_DATA)

After starting working normally, SLAMWARE CORE indoor localization and navigation module will continually poll for base sensor data from external system and the later should send response in given time. If external system informs no range sensor configured when system parameter is obtaining the request, SLAMWARE CORE will not send the request.

- Request data packet:

Command code
0x32

- Response data packet:

Response code	Payload data
<OK>	base sensor data
	u32[16]

Data field	Type	Description
base sensor data	u32[16]	Base sensor data of external system. The result is the distance information from the sensor to obstacle. The unit come out in mm, Q16 fix-point decimal. It can support up to 16 groups of range sensors. Base sensor is including infrared distance measuring sensor and ultrasonic sensor. **Note:** when using the 0x20 command, 8 sensors can be supported in the most, therefore 8 base sensor data can be obtained in the most. When using the 0x21 command, 16 sensors can be supported in the most and 16 base sensor data can be obtained in the most.

Figure 4-11 Range Sensor Data Obtaining Request Field Description

External System Getting Bump Sensor Data Request(GET_BASE_BUMPER_DATA)

After starting working normally, SLAMWARE CORE indoor localizing and navigation module will continually poll for bump sensor data from external system and the later should response in given time. If external system informs no bump sensor configured when executing the system parameter request, SLAMWARE CORE will not send the request.

- Request data packet:

Command code
0x33

- Response data packet

Response code	Payload data
<OK>	Bump sensor data
	u8

Data field	Type	Description
Bump sensor data	u8	Bump sensor data from external system. Each bit indicates the status of the bump sensor and is in the same sequence with the sensor configuration. When it is specified as 1, it indicates the bump sensor is not triggered; and when as 0, it indicates the bump sensor is triggered. It can support up to 8 groups of bump sensors.

Figure 4-12 Bump sensor Data Obtaining Response Field Description

External System Getting Auto Home Data Request(GET_AUTO_HOME_DATA)

When the SLAMWARE CORE indoor localizing and navigation module tries to recharge by itself, it will obtain related information by invoking this command. Currently, it is used for matching beacon information.

- Request data packet

Command Code	Payload data
0x34	Data type
	U8

Data field	Type	Description
Data Type	u8	The SLAMWARE CORE supports the following data type for obtaining self-charging data: 0 – obtaining self-charging Beacon information If the value is not 0, please send "not support" as the error message to response this request. (0x8000)

Figure 4-13 Self-charging Data Obtaining Request's Request Field Description

- Response data packet (When the datatype is 0)

Command Code	Payload data		
<OK>	The number of the beacon(s)	The number of the receiver(s)	Receiver(s) data
	u8	u8	u8[n]

Data field	Type	Description
The number of beacon(s)	u8	The number of the beacon(s) on the charging dock of external system. It currently supports three beacons. 0 is the left beacon, 1 is the main beacon, and 2 is the right beacon.
The number of the receiver(s)	u8	The number of the receiver(s) of external system. It currently supports three receivers. 0 is the left receiver, 1 is the main receiver, and 2 is the right receiver.
Receiver(s) data	u8[n]	Each receiver is matched with a byte, and each byte stands for the beacon received by the matched receiver. For example, if receiver 1 can receive the data of beacon 0 and beacon 1, then Data[1] = (1 << 0) | (1 << 1).

Figure 4-14 Self-charging Data Obtaining Request's Response Field Description

External System Speed Control Request(SET_BASE_MOTOR)

SLAMWARE CORE indoor localizing and navigation module controls the motion of external system by sending motion control request. And external system controls the system motion accurately according to the content in data packet to make SLAMWARE CORE function properly.

- Request data packet:

Command code	Payload data
0x40	Speed
	s32[4]

- Response data packet

Response code
<OK>

Data field	Type	Description
Speed	s32[4]	The motor speed of the external system wheels. The units come out in mm per second.

Figure 4-15 External System Motion Control Request Field Description

External System Set Velocity Request(SET_V_AND_GET_DEADRECKON)

SLAMWARE CORE indoor localizing and navigation module sets the velocity and gets the deadreckon data of last period.

- Request data packet:

Command code	Payload data		
0x41	Velocity on X-axis	Velocity on Y-axis	Velocity on angular
s32	s32	s32	s32

- Response data packet:

Response code	Payload data		
<OK>	X-axis displacement	Y-axis displacement	Angular displacement
	s32	s32	s32

Data field	Type	Description
Y-axis velocity	s32	Velocity of external system on X-axis, Q16 fix-point decimal.
X-axis velocity	s32	Velocity of external system on Y-axis, Q16 fix-point decimal.
Angular velocity	s32	Velocity of external system on angular, Q16 fix point decimal.
X-axis displacement	s32	Compared with the response of last request, the displacement of external system on X-axis. The units come out in mm, Q16 fix-point decimal.
Y-axis displacement	s32	Compared with the response of last request, the displacement of external system on Y-axis. The units come out in mm, Q16 fix-point decimal.
Angular displacement	s32	Compared with the response of last request, the anti-clock wise displacement of external system. The units come out in degree, Q16 fix-point decimal.

Figure 4-16 External System Set Velocity Request Field Description

Note: the axis here is right-handed coordinate system with the robot as the center. The right ahead of the robot is X axis.

Take the two-wheeled differential drive motor as an example, after the robot receives the command:

1. It should calculate the three displacement values(dx/dy/dyaw) according to the calculated distance of the right and left wheels;
2. Transfer the velocity value of the command(vx/vy/omega) into the line speed of the right and left wheels(vl/vy);
3. Return the result in the first step

The method to calculate the three displacement values(dx/dy/dyaw)

1. Variable definition

Variable name	Definition	Unit

dl	Left wheel displacement , the right ahead of the robot is positive.	m
dr	Right wheel displacement , the right ahead of the robot is positive.	m
dx	Robot forward direction displacement, the right ahead of the robot is positive.	m
dy	Robot side direction displacement, the left of the robot is positive.	m
dyaw	Robot angular displacement, anti-clockwise is positive.	rad

2. Formula

$$dyaw = \frac{dr - dl}{2R}$$

$$dx = \frac{dl + dr}{2}\cos(dyaw)$$

$$dy = \frac{dl + dr}{2}\sin(dyaw)$$

Note: the R is the wheel-track radius, the unit comes out as m.

3. Code sample

```
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;
```

The method to calculate the line speed of the right and left wheels

1. Variable definition

Variable name	Definition	Unit
vx	The forward direction line speed of the robot, the right ahead of the robot is positive.	m/s
vy	The side direction line speed of the robot, the left is the positive. This value is 0 for two-wheeled differential drive motor.	m/s

omega	The angular speed of the robot, anti-clockwise is the positive.	rad/s
vl	The right wheel line speed of the robot. The right ahead is the positive.	m/s
vr	The leftt wheel line speed of the robot. The right ahead is the positive.	m/s

2. Fromula

$$vl = vx - omega * R$$

$$vr = vx + omega * R$$

3. Code sample

```
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;
```

External System Poll Command Request(POLL_BASE_CMD)

SLAMWARE CORE indoor localizing and navigation module can receive command from external system, but restricted by one-way communication mode, the external system can send control command only when SLAMWARE CORE send the request of control command checking to external system. SLAMWARE CORE will termly check the control command of external system. As for the control command that requires a quicker response, the external system can send it via #CMD to notify SLAMWARE CORE send out control command checking request earlier. When #BUSY signal is in high level, it indicates SLAMWARE CORE is busy and cannot response to the control command of external system but the later still requires to response the request command sent from SLAMWARE CORE in time.

Note: SLAMWARE CORE will not response the command of external system. After SLAMWARE CORE receiving the command from external system, external system can check the #BUSY signal pulse. The external system must pull down the #CMD

signal after responding command checking request, or the system would be left in unknown status.

- Request data packet:

Command code
0x50

- Response data packet:

Response code	Payload data
<OK>	External system command
	u8

Data field	Type	Description
External system command code	u8	Only SLAMWARE CORE supported command code can be functional

Figure 4-17 External System Command Checking Request Field Description

External System Command List	Command Name	Description
0x51	GET_INFO	Get SLAMWARE CORE System Information, including software and firmware version and network configurations.
0x52	RESET_WIFI	Reset SLAMWARE CORE wireless network
0x53	FW_UPGRADING	Update SLAMWARE CORE firmware
0x80	START_SWEEP	Start cleaning (Only available in vacuum robot edition SLAMWARE)
0x81	STOP_SWEEP	Stop sweeping(Only available in vacuum robot edition SLAMWARE)
0x82	SPOT_SWEEP	Spot sweeping (Only available in vacuum robot edition SLAMWARE)
0x90	GET_HEALTH	Get health status
0xA0	MOVE_FORWARD	Move forward
0xA1	MOVE_BACKWARD	Move backward
0xA2	TURN_LEFT	Turn left
0xA3	TURN_RIGHT	Turn right

0xAF	CANCEL_ACTION	Cancel the action

Figure 4-18 External System Command List

External System Poll Command Response Request(POLL_BASE_ANS_CMD)

SLAMWARE CORE will give response after processing the command request from external system. The response request always comes in pair with external system command checking request.

The supported external system command code for SLAMWARE CORE is not defined currently.

- Request data packet:

Command code
0x5F

- Response data packet:

Response code	Payload data
<OK>	Answer data
	u8

Data field	Type	Description
Response data	u8	Response data of external system command. Related to SLAMWARE CORE command code.

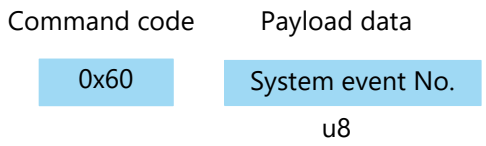
Figure 4-19 External System Command Response Request Field Description

SLAMWARE CORE System Event Notice Command(SEND_EVENT)

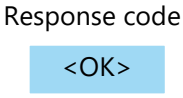
SLAMWARE CORE indoor localizing and navigation module would aperiodically send system event notice to external system and the later will response accordingly.

Note: The supported system event code of SLAMWARE CORE may be changed according to the development.

- Request data packet:



- Response data packet:



Data field	Type	Description
System event code	u8	SLAMWARE CORE's system event code. For details, please refer to the following figure.

Figure 4-20 SLAMWARE CORE Event Notice Command Request Field Description

Currently supported system event codes are as below:

System event code	Command Name	Description
0x61	LIDAR_CONN_FAIL	LIDAR connection fail
0x62	LIDAR_RAMPUP_FAIL	LIDAR launching fail
0x63	SYSTEM_UP_OK	System launching success
0x64	FIRMWARE_UPDATE	System firmware update
0x65	CORE_DISCONNECT	System disconnected
0x66	FIRMWARE_UPDATE_OK	System firmware update success
0x80	START_SWEEP	Inform external system starting sweeping (Only available in vacuum robot edition SLAMWARE)
0x81	END_SWEEP	Inform external system sweeping finished (Only available in vacuum robot edition SLAMWARE)

Figure 4-21 Supported Event Notice Command by SLAMWARE CORE

External System Health Management Request(HEALTH_MGMT)

SLAMWARE CORE indoor localizing and navigation module would aperiodically send this request to external system and the later will response accordingly.

- Request data packet:

Command code	Payload data
0x90	Health management event code
	u8

- Response data packet:

Response code
<OK>

Data field	Type	Description
Health management event code	u8	SLAMWARE CORE's health management event code. For details, please refer to the following figure.

Figure 4-22 SLAMWARE CORE Health Management Command Field Definition

Health management event code	Command Name	Description
0x01	HEALTH_GET_HEALTH	Get health status of external system
0x02	HEALTH_GET_ERROR	Request to get the detailed message of external system
0x03	HEALTH_CLEAR_ERROR	Request to clear error message of external system

Figure 4-23 Supported Event Code by SLAMWARE CORE Health Management Command

Date	Version	Description
2015-03-18	0.1	Initial version. Define CTRL BUS communication protocol.
2015-03-20	0.2	Update protocol.
2015-03-30	0.3	Update protocol. Define fix-point decimal.
2015-09-09	0.4	Update protocol. Update connect command and command to get wheels' status. Add response request command for external system. Add SLAMWARE CORE system event notice command
2015-11-04	0.5	Modify protocol format for the command of SLAMWARE CORE system event notice
2015-12-30	0.6	Update copyright term and polish up the text
2016-03-15	1.7	Add some new commands: External system configuration obtaining request (0x21) ; External system self-charging data obtaining request (0x34) .
2016-05-19	1.8	Added the Inter-chip communication protocol specification and updated the layout.
2016-06-17	1.8	Added the external system command and event notice command about vacuum robot edition SLAMWARE.
2016-07-15	1.8	Added format description about the payload data type of bump sensor data obtaining request.
2016-09-07	1.8	Added SET_V_AND_GET_DEARECKON interface definition
2016-09-12	1.8	Added HEALTH_MANAGEMENT interface definition

Image and Table Index

FIGURE 1-1 SLAMWARE CORE CTRL BUS PROTOCOL AND INTER-CHIP PROTOCOL 3
FIGURE 2-1 SLAMWARE CORE MODULE STRUCTURE 4
FIGURE 3-1 INTER-CHIP PROTOCOL DATA PACKET FORMAT DEFINITION 6
FIGURE 3-2 INTER-CHIP PROTOCOL FLAG FIELD FORMAT DEFINITION 6
FIGURE 3-3 INTER-CHIP PROTOCOL FLAG FIELD FLAG BIT DESCRIPTION 6
FIGURE 3-4 INTER-CHIP PROTOCOL LEN AND LENH FIELD FORMAT DEFINITION 7
FIGURE 3-5 INTER-CHIP PROTOCOL CMD FIELD FORMAT DEFINITION 7
FIGURE 3-6 INTER-CHIP PROTOCOL PAYLOAD[N] FIELD FORMAT DEFINITION 8
FIGURE 3-7 INTER-CHIP PROTOCOL CHKSUM FIELD FORMAT DEFINITION 8
FIGURE 3-8 INTER-CHIP PROTOCOL EXFLAG[N] FIELD FORMAT DEFINITION 8
FIGURE 3-9 INTER-CHIP PROTOCOL ADDR & EXADDR[N] FIELD FORMAT DEFINITION 9
FIGURE 3-10 THE COMMUNICATION BETWEEN MASTER & SLAVE BASED ON INTER-CHIP PROTOCOL 10
FIGURE 3-11 INTER-CHIP PROTOCOL FLAG FIELD FORMAT DEFINITION 10
FIGURE 3-12 INTER-CHIP PROTOCOL FLAG DATA PACKET FORMAT DESCRIPTION 11
FIGURE 3-13 INTER-CHIP PROTOCOL REQUEST COMMAND DATA PACKET DEFINITION 11
FIGURE 3-14 INTER-CHIP PROTOCOL RESERVED CMD DESCRIPTION 11
FIGURE 3-15 INTER-CHIP PROTOCOL RESPONSE DATA PACKET FORMAT DEFINITION 12
FIGURE 3-16 INTER-CHIP PROTOCOL RESERVED RESP DESCRIPTION 12
FIGURE 3-17 INTER-CHIP PROTOCOL ERROR CODE DESCRIPTION 13
FIGURE 4-1 REQUEST/RESPONSE COMMUNICATION MODE 14
FIGURE 4-2 REQUEST/RESPONSE COMMUNICATION MODE 14
FIGURE 4-3 SLAMWARE CORE REQUEST COMMAND LIST 16
FIGURE 4-4 EXTERNAL SYSTEM CONNECTION REQUEST FIELD DESCRIPTION 17
FIGURE 4-5 EXTERNAL SYSTEM PARAMETER OBTAINING REQUEST FIELD DESCRIPTION 18
FIGURE 4-6 SUPPORTED MECHANICAL DIMENSION OF EXTERNAL SYSTEM 18
FIGURE 4-7 SUPPORTED WHEEL SET TYPE 18
FIGURE 4-8 SENSOR INSTALLATION LOCATION POS DATA 19
FIGURE 4-9 EXTERNAL SYSTEM STATUS OBTAINING REQUEST FIELD DESCRIPTION 20
FIGURE 4-10 EXTERNAL SYSTEM WHEEL STATUS OBTAINING REQUEST FIELD DESCRIPTION 21
FIGURE 4-11 RANGE SENSOR DATA OBTAINING REQUEST FIELD DESCRIPTION 22
FIGURE 4-12 BUMP SENSOR DATA OBTAINING RESPONSE FIELD DESCRIPTION 23
FIGURE 4-13 SELF-CHARGING DATA OBTAINING REQUEST' S REQUEST FIELD DESCRIPTION 23
FIGURE 4-14 SELF-CHARGING DATA OBTAINING REQUEST' S RESPONSE FIELD DESCRIPTION 24
FIGURE 4-15 EXTERNAL SYSTEM MOTION CONTROL REQUEST FIELD DESCRIPTION 24
FIGURE 4-16 EXTERNAL SYSTEM SET VELOCITY REQUEST FIELD DESCRIPTION 25
FIGURE 4-17 EXTERNAL SYSTEM COMMAND CHECKING REQUEST FIELD DESCRIPTION 28
FIGURE 4-18 EXTERNAL SYSTEM COMMAND LIST 29

FIGURE 4-19 EXTERNAL SYSTEM COMMAND RESPONSE REQUEST FIELD DESCRIPTION 29
FIGURE 4-20 SLAMWARE CORE EVENT NOTICE COMMAND REQUEST FIELD DESCRIPTION 30
FIGURE 4-21 SUPPORTED EVENT NOTICE COMMAND BY SLAMWARE CORE 30
FIGURE 4-22 SLAMWARE CORE HEALTH MANAGEMENT COMMAND FIELD DEFINITION 31
FIGURE 4-23 SUPPORTED EVENT CODE BY SLAMWARE CORE HEALTH MANAGEMENT COMMAND 31