DSSP-HAL
Operation Manual
(TETRTA-DS III PRO™)

Version 1.1

2011. 03.

Dongbu Robot

Table of Contents

Chapter 1. DSSP-HAL Overview

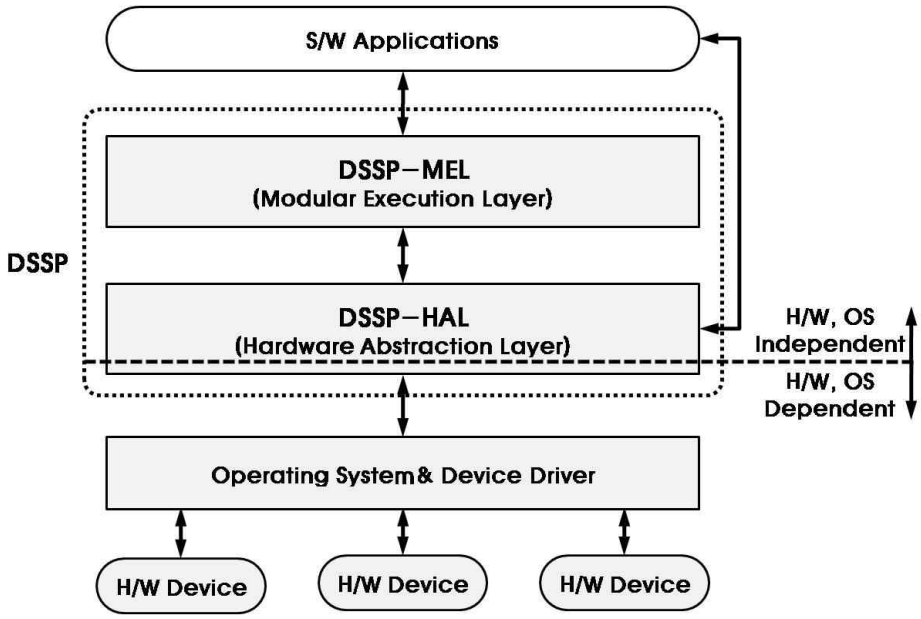
1-1. What is DSSP?

DSSP (DaSarobot Software Platform) is a Network-based Modular Software Platform developed by DasaRoobt to develop autonomous navigation technology for mobile robot platforms. In order to support development of hardware and OS independent application programs, DSSP is composed of two separate layers, DSSP-HAL and DSSP-MEL as seen in the diagram 1-1 below.

DSSP-HAL is a network based API abstraction layer with hardware and OS independent XML designed communications protocol for accessing various devices and hardware installed on the platform. DSSP-MEL is an OS independent Python based modular execution layer consisting of DSSP-HAL APIs.

DSSP-MEL has various modules such as obstacle detection/avoidance module, path planning module, localization module, and map building module that makes up the autonomous movement technology arranged in hierarchical form.

More detailed technical information concerning the DSSP and DSSP-MEL can be found at our website or by contacting the Customer Support Center.

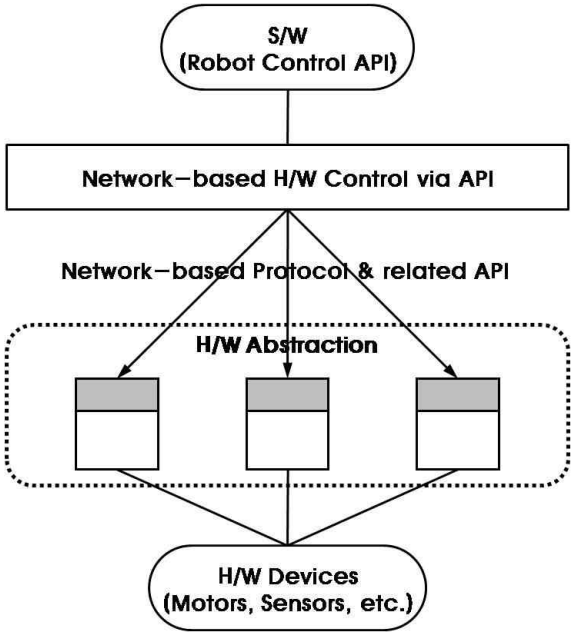


<Diagram 1-1> DSSP Architecture

1-2. What is DSSP-HAL?

'DSSP-HAL(DaSarobot Software Platform — Hardware Abstraction Layer) is a network based hardware abstraction layer with XML based protocol for accessing and controlling all standard and optional devices installed on TETAA-DS III™ in simple and unified manner.

Diagram 1-2 below shows the DSSP-HAL concept.



<Diagram 1-2> DSSP-HAL Concept

As shown diagram 1-2, objective of DSSP-HAL is to make various hardware installed on the robot platform easier to use through abstraction. To achieve such an end, DSSP-HAL supports simple and easy to understand data structure and hardware layer abstraction is accomplished by distributed processing using the network which it' s structure is based on. Communications protocol is designed using system independent XML (eXtended Mark-up Language) to allow implementing hardware abstraction layer on mobile robot platforms based on different systems. Refer to section 2-1 for information on XML based DSSP-HAL communications protocol.

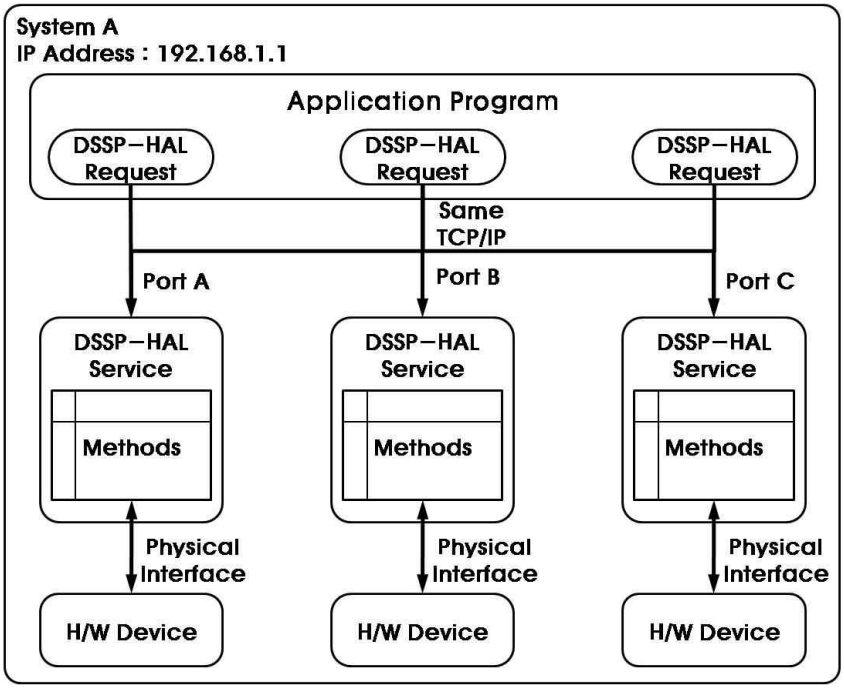
1-3. Implementing DSSP-HAL API

DSSP-HAL operates within the network through the socket interface based on TCP/IP protocol.

Diagram 1-3 and 1-4 shows the DSSP-HAL API working in various networking environment. As seen in the diagram 1-3 and 1-4, DSSP-HAL shows standard network based operating characteristics with "DSSP-HAL Service" acting as a "Server" and providing "service" and "DSSP-HAL Request" acting as a client requesting "service". Hardware abstraction in "method" format of RPC (Remote Procedure Call) is used in the section providing the service to simplify the use of connected robot hardware functions. Also, the data used for service is standardized and simplified by the data structure provided by the DSSP-HAL. The result is the ability to control complex robot hardware by simply requesting the methods remotely within the network using the DSSP-HAL function. As DSSP-HAL operation is based on TCP/IP protocol, it can be implemented in any system that supports TCP/IP protocol.

1-3-1. Implementing API in Same System

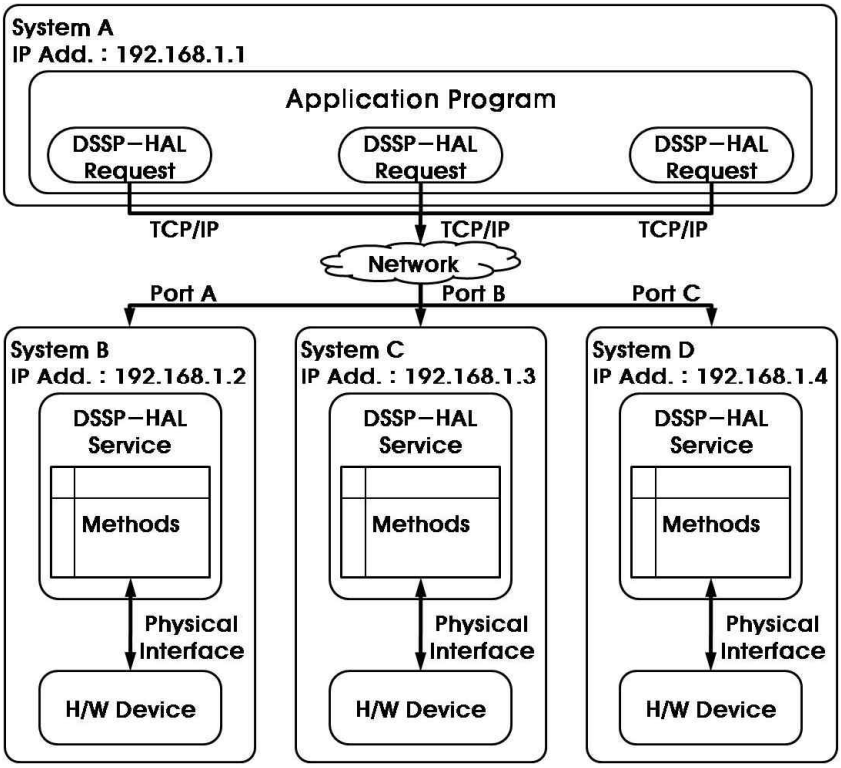
As shown in diagram 1-3, to access and control the hardware within the same network system with the same the IP address, DSSP-HAL Request is sent through a particular port to request specific DSSP-HAL service to access and control the hardware. When DSSP-HAL service assigned to a specific port receives such a DSSP-HAL request command, hardware is accessed through physical interface supported by the method which in turn is supported by DSSP-HAL service to receive requested information or to control the hardware according to the request.



<Diagram 1-3> DSSP-HAL API Implementation in the same system

1-3-2. Implementing API in Different Systems

As shown in diagram 1-4, to access and control the hardware in different network system with different IP address, DSSP-HAL Request is sent through a particular port within the system with particular IP address to request specific DSSP-HAL service to access and control the hardware. When DSSP-HAL service assigned to a specific port within the system with specific IP address receives such a DSSP-HAL service request command, hardware is accessed through the physical interface supported by the method which in turn is supported by DSSP-HAL service to receive requested information or to control the hardware according to the request.



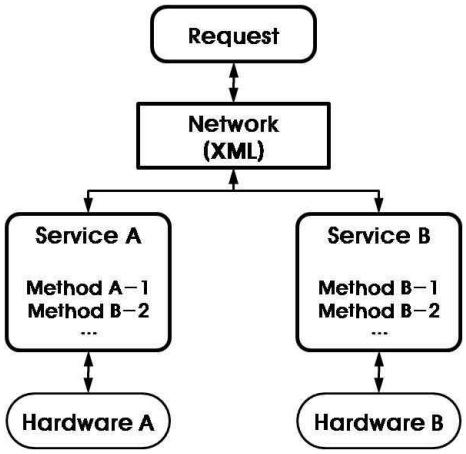
<Diagram 1-4> DSSP-HAL API Implementation among systems

DSSP-HAL is installed in the VIA Embedded Board as a standard feature and it is comprised of services with hardware abstraction for each hardware. Refer to the next chapter for details on DSSP-HAL services.

Chapter 2. DSSP-HAL Structure

2-1. DSSP-HAL Protocol

DSSP-HAL is a network-based hardware abstraction layer with OS independent communications protocol designed using XML (eXtended Mark-up Language). DSSP-HAL supports both Linux and Windows based systems with Windows based systems using winsock2. Diagram 2-1 below shows the DSSP-HAL structure.

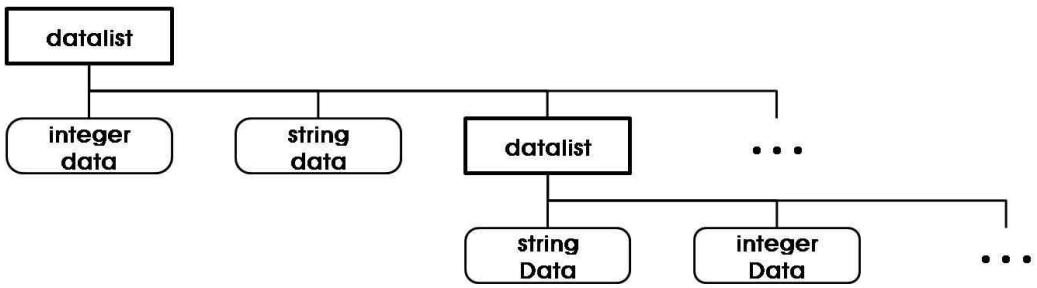


<Diagram 2-1> DSSP-HAL Structure

As shown in diagram 2-1, all the services contain various methods showing the functions available to the particular service. Actual data used to control the robot that is sent back and forth between the "Request" and "Service" is in XML type structure.

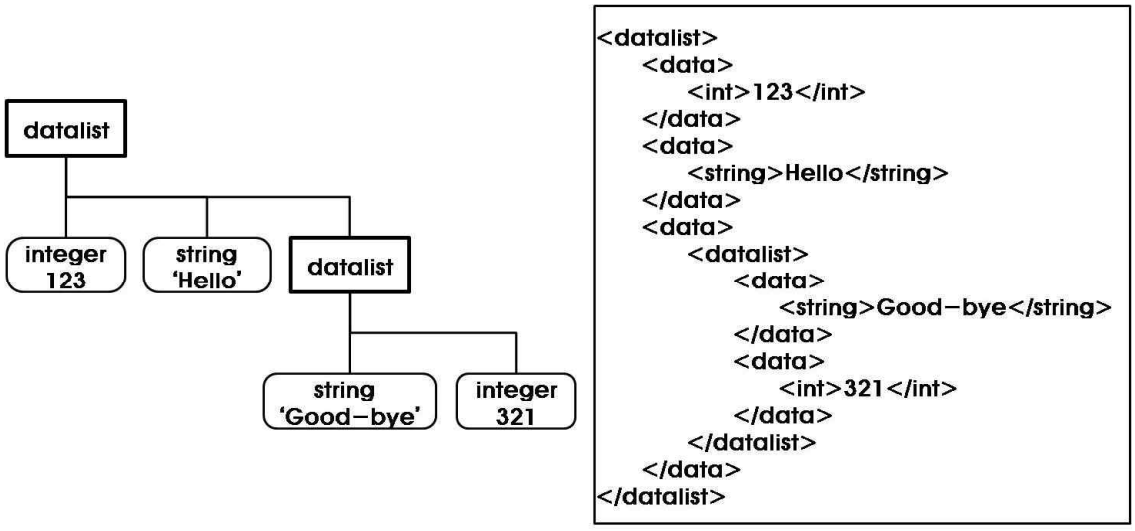
2-2. DSSP-HAL Data

DSSP-HAL supports Integer type data, String type data, and Datalist. As shown in the diagram 2-2, Datalist can contain Integer, String, or Datalist as part of its component. The information sent between the DSSP-HAL "request" and "service" contain these types of data.



<Diagram 2-2> Example of DSSP-HAL Data Structure

As shown in diagram 2-2 below, data supplied by DSSP-HAL service can carry variety of hardware information by combining supported data format in various ways. Also, data supplied by DSSP-HAL service is in OS independent XML type structure as seen in diagram 2-3.

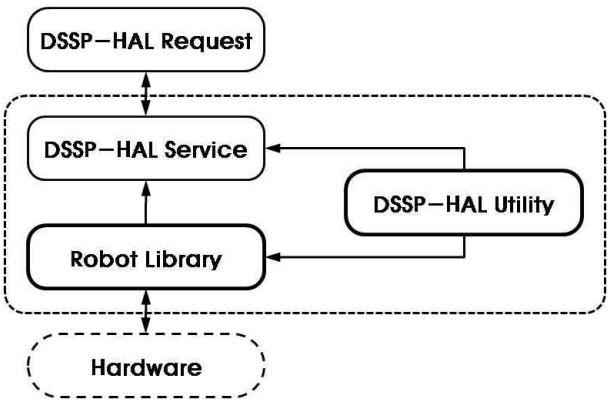


<Diagram 2-3> XML-based Structure Code of DSSP-HAL Data

2-3. DSSP-HAL Request and Service

DSSP-HAL can be divided into "DSSP-HAL service" component, which provides simple and easy way of accessing the various hardware installed on the platform through the network connection, and the "DSSP-HAL Request" component, which requests such services.

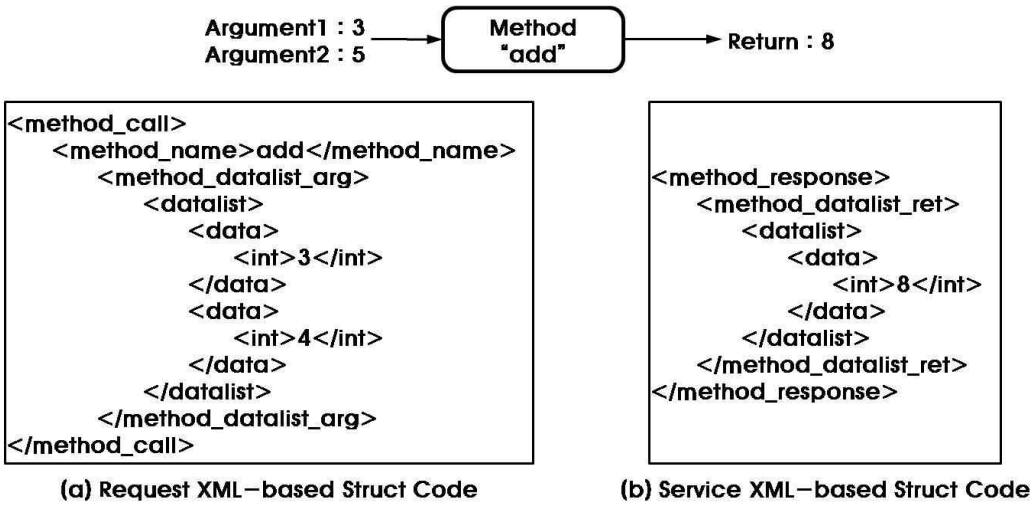
DSSP-HAL provides the services in standard network server format with the "server" providing the hardware related services being called "DSSP-HAL Service" and the part requesting the service viewed as network client. The diagram 2-4 shows the DSSP-HAL communications structure when controlling the mobile robot.



<Diagram 2-4> DSSP-HAL Request & Service Structure

The " robot library" shown is diagram 2-4 above is a UART protocol, predefined collection of libraries of the installed hardware and options that could be used through the DSSP-HAL service method. For example, "TETAA Library" being sold for our mobile robot platform is a type of robot library. Also DSSP-HAL utility shown in diagram 2-4 are collection of predefined functions that could be used for Socket communication, UART communication, and Semaphore by the robot library and DSSP-HAL service.

As explained in the DSSP-HAL data structure, DSSP-HAL communications protocol is designed based on OS independent XML structure. The example in diagram 2-5 below shows the XML based structure of the "request" and "service" when "add" method is called to add two arguments and return the added value. In other words, all methods in DSSP-HAL have OS independent character as they are designed based on XML structure.



<Diagram 2-5> XML-based Structure Code of DSSP-HAL Request & Service

2-4. DSSP-HAL Service Method

When using the DSSP-HAL, access and control of particular hardware is made possible by calling on the specific method registered to the DSSP-HAL service. Diagram 2-7 below shows the codes of the "method_velocity_control" method, one of the methods registered to the "Drive Service" which is a DSSP-HAL service used for controlling the drive motor of the our mobile robot platform TETRA-DS III™.

```
dsphal_method_return_t method_velocity_control(
                dsphal_datalist_t *datalist_arg, dsphal_datalist_t **datalist_ret)
{
        tetra_drive_module_t tetra_drive_module;

        int left_vel;
        int right_vel;
        int semaphore;

        if (!datalist_arg)
                goto ret_err_3;
        if ((semaphore = tetra_semaphore_create(SEMKEY_DRIVE_MODULE, 1)) < 0)
                goto ret_err_3;
        tetra_semaphore_wait(semaphore);
        if (tetra_drive_module_open(&tetra_drive_module, DEVICE_DRIVE_MODULE)) ❶
                goto ret_err_2;
        dsphal_decompose_root_datalist(datalist_arg, "[{i}{i}]", &left_vel, &right_vel); ❷
        if (tetra_drive_module_vel_ctrl(&tetra_drive_module, left_vel, right_vel))
                goto ret_err_1;
        tetra_drive_module_close(&tetra_drive_module);
        tetra_semaphore_signal(semaphore);
        tetra_semaphore_close(semaphore);

        return DSPHAL_METHOD_RETURN_OK;
ret_err_1:
        tetra_drive_module_close(&tetra_drive_module);
ret_err_2:
        tetra_semaphore_signal(semaphore);
        tetra_semaphore_close(semaphore);
ret_err_3:
        return DSPHAL_METHOD_RETURN_ERR;
}
```

<Diagram 2-7> Example of DSSP-HAL Method

As shown in the diagram 2-7 above, each method in the DSSP-HAL service is formed by calling on the defined functions in the "Robot

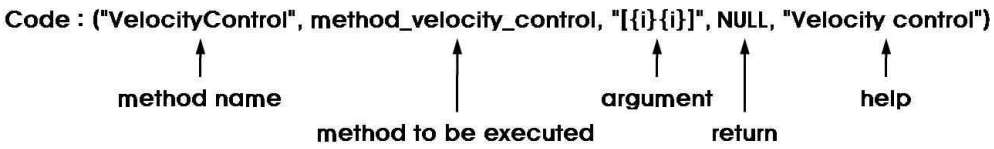
Library" (shown in diagram 2-4) which in this case is "TETAA Library" and DSSP-HAL Utility. Among the codes shown in diagram 2-7, tetra_drive_module_open( ) in Line ① shows "Robot Library" function and the dsphal_decompose_root_datalist( ) in Line ② shows "DSSP-HAL Utility" function.

Similar to the way "server" operates in the network, methods in the DSSP-HAL Service are registered in the DSSP-HAL Service main and when the request command is called, it executes the specified method and goes into wait mode. The diagram 2-8 below shows parts of the main function codes of the "Drive Service" that controls the drive motors installed in the TETAA-DS III™' .

```
int main(int argc, char **argv)
{
    dsphal_tcp_server_t *tcp_server;
    dsphal_tcp_server_t *new_tcp_server;
    dsphal_service_t *service;
    dsphal_registry_t *registry;
    dsphal_initialize();
    if (initialize())
        return -1;
    registry = dsphal_registry_create();
    dsphal_introspection_register(registry);
    dsphal_registry_add_method(registry, dsphal_method_create(
        "VelocityControl", method_velocity_control, "[{i}{i}]", NULL, "Velocity
control"));
    ......
    service = dsphal_service_create(registry);
    tcp_server = dsphal_tcp_server_create(PORT_DRIVE);
    while (1) {
        new_tcp_server = dsphal_tcp_server_accept(tcp_server);
        dsphal_service(new_tcp_server, service);
        dsphal_tcp_server_destroy(new_tcp_server);
    }
    return -1;
}
```

<Diagram 2-8> Example of DSSP-HAL Service Main

Arguments of the functions making up the method in diagram 2-8 are explained in the diagram 2-9 below. The value of the argument or return in diagram 2-9 is defined in "[{?}]" format, "i" in place of the "?" refers to integer type data and "s" refers to string type data, and the number of "?" marks refer to number of the argument or return values. The method "VelocityControl" shown in diagram 2-9 shows that it has 2 integer values for the argument and no values for the return. Diagram 2-9 explains the argument of the "DSSP-HAL Utility" function 'dsphal_registry_add_method' that adds and registers a method in DSSP-HAL service.



<Diagram 2-9> Example of DSSP-HAL Service Code

The diagram 2-10 is an example of the DSSP-HAL request codes displayed when TETAA-DS III™' obeys the control command to turn the left and the right wheels at speed of 100 mm/s.

```
#include "dsphal.h"  ①

#define IP_ADDR_DMP "192.168.51.10"  ②
#define PORT_DRIVE 50010  ③

int main(void)
{
    dsphal_tcp_client_t *tcp_client;
    dsphal_datalist_t *datalist_arg;

    tcp_client = dsphal_tcp_client_create(IP_ADDR_DMP, PORT_DRIVE);
    dsphal_tcp_client_connect(tcp_client);
    datalist_arg = dsphal_build_root_datalist("[{i}{i}]", 100, 100);  ④
    dsphal_request_method_call(tcp_client,  "VelocityControl", datalist_arg);  ⑤
    dsphal_tcp_client_destroy(tcp_client);

    return 0;
}
```

<Diagram 2-10> Example of DSSP-HAL Request

The explanation of the lines in the request codes shown in diagram 2-10 are as follow.

Line ① : DSSP-HAL includes the defined Header.

Line ② : DSSP-HAL defines the IP address of the main control board where service is being performed.

Line ③ : Defines port number of each DSSP-HAL Service.

Line ④ : Creates argument required by the method.

Line ⑤ : Calls method 'VelocityControl'.

Chapter 3. Access through DSSP-HAL Service

This chapter describes the key component of the DSSP-HAL, namely accessing and controlling specific hardware through DSSP-HAL service.

3-1. DSSP-HAL Service Structure Code

3-1-1. dsphal_tcp_client_t

Struct	typedef struct dsphal_tcp_client_t { char *server_ip_addr; int server_port; int socket_fd; } dsphal_tcp_client_t;	
Description	Save DSSP-HAL service TCP information dsphal_tcp_client_create( ) Return value of function	
Argument	server_ip_addr	DSSP-HAL service IP(v4) address
	server_port	DSSP-HAL service Port number
	socket_fd	Socket file descriptor
Example Code	dsphal_tcp_client_t *tcp_client; (define as pointer)	

3-1-2. dsphal_datalist_t

Struct	typedef struct dsphal_datalist_t { size_t size; dsphal_data_t *head; dsphal_data_t *tail; } dsphal_datalist_t;	
Description	Argument(input) or Return(outp) datalist to be received dsphal_request_method_call( ) Return value of function	
Argument	size	Datalist size
	*head	Datalist start pointer
	*tail	Datalist end pointer
Example Code	dsphal_datalist_t * datalist_arg; (define as pointer)	

3-2. DSSP-HAL Service Functions

3-2-1. dsphal_tcp_client_t *dsphal_tcp_client_create(char *server_ip_addr, int server_port)

Function Name	dsphal_tcp_client_t *dsphal_tcp_client_create(char *server_ip_addr, int server_port)	
Argument	server_ip_addr	DSSP-HAL service IP(v4) address

	server_port	DSSP-HAL service Port number
Description	Input DSSP-HAL service IP and Port information	
Return	OK(address)	TCP address value
	NULL	Fail
Example Code	#define IP_ADDR_DSSMP "192.168.51.10" #define PORT_BUMPER 50011 dsphal_tcp_client_t *tcp_client; tcp_client = dsphal_tcp_client_create(IP_ADDR_DSSMP, PORT_BUMPER);	

3-2-2. int dsphal_tcp_client_connect(dsphal_tcp_client_t *dsphal_tcp_client)

Function Name	int dsphal_tcp_client_connect(dsphal_tcp_client_t *dsphal_tcp_client)	
Argument	dsphal_tcp_client	DSSP-HAL service TCP information
Description	Connect to DSSP-HAL service TCP	
Return	OK (0)	Connection successful
	False(-1)	Connection fail

Example Code	#define IP_ADDR_DSSMP "192.168.51.10"
	#define PORT_BUMPER 50011
	
	dsphal_tcp_client_t *tcp_client;
	int connect_state;
	
	tcp_client = dsphal_tcp_client_create(IP_ADDR_DSSMP, PORT_BUMPER);
	connect_state = dsphal_tcp_client_connect(tcp_client);	

3-2-3. int dsphal_tcp_client_destroy(dsphal_tcp_client_t *dsphal_tcp_client)

Function Name	void dsphal_tcp_client_destroy(dsphal_tcp_client_t *dsphal_tcp_client)	
Argument	dsphal_tcp_client	DSSP-HAL service TCP information
Description	Disconnect from DSSP-HAL service	
Return	None	

Example Code	#define IP_ADDR_DSSMP "192.168.51.10"
	#define PORT_BUMPER 50011
	
	int connect_state;
	dsphal_tcp_client_t *tcp_client;
	
	tcp_client = dsphal_tcp_client_create(IP_ADDR_DSSMP, PORT_BUMPER);
	connect_state = dsphal_tcp_client_connect(tcp_client);	

	… … . dsphal_tcp_client_destroy (tcp_client);

3-2-4. dsphal_datalist_t *dsphal_build_root_datalist(char *format, data1, data2 …)

Function Name	dsphal_datalist_t *dsphal_build_root_datalist(char *format, ...)	
Argument	Format	DSSP-HAL data structure format * variable length argument
Description	Create datalist to send to DSSP-HAL service	
Return	dsphal_datalist_t	Datalist address value

Example Code	dsphal_datalist_t *datalist_arg;
	int lvel, rvel;
	lvel = 50;
	rvel = 50;
	
	datalist_arg = dsphal_build_root_datalist("[{i}{i}]", lvel, rvel);	

※ **char *format Data format**

All defined data types in DSSP-HAL is composed of either integer or string type. Char type *format variable appearing above starts with "[" and ends with "]" and if the integer is to appear in between, it is shown by the matching number of "{i}" and if the string is to appear, it is also designed to show matching number of "{s}" .

3-2-5. void dsphal_datalist_destroy(dsphal_datalist_t *dsphal_datalist)

Function Name	void dsphal_datalist_destroy(dsphal_datalist_t *dsphal_datalist)	
Argument	dsphal_datalist	Datalist address value
Description	Clear datalist memory	
Return	None	
Example Code	dsphal_datalist_t *datalist_arg; … … if (datalist_arg) dsphal_datalist_destroy(datalist_arg);	

3-2-6. dsphal_datalist *dsphal_request_method_call(dsphal_tcp_client_t *dsphal_tcp_client, char *method_name, dsphal_datalist_T *dsphal_datalist_arg)

Function Name	dsphal_datalist_t *dsphal_request_method_call(dsphal_tcp_client_t *dsphal_tcp_client,

	char *method_name, dsphal_datalist_t *dsphal_datalist_arg)	
Argument	dsphal_tcp_client	DSSP-HAL service TCP information
	method_name	method name
	dsphal_datalist_arg	Datalist Argument
Description	Request method call to DSSP-HAL service	
Return	None	No return value
	dsphal_datalist_t	If return value, datalist address value

Example Code	**A. When datalist argument exists**
	#define IP_ADDR_TETAA "192.168.51.10"
	#define PORT_DRIVE 50010
	
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_arg;
	dsphal_datalist_t *datalist_ret;
	int lvel
	int rvel;
	lvel = 50;
	rvel = 50;
	
	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETAA, PORT_DRIVE);
	dsphal_tcp_client_connect(tcp_client);
	datalist_arg = dsphal_build_root_datalist("[ii]", lvel, rvel);
	datalist_ret = dsphal_request_method_call(
	tcp_client, "VelocityControl", datalist_arg);
	
	B. When there is no datalist argument
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_ret;
	int i;
	
	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETAA, PORT_BUMPER);
	dsphal_tcp_client_connect(tcp_client);
	datalist_ret = dsphal_request_method_call(
	tcp_client, "SetBumperDirMode", NULL);	

3-2-7. int dsphal_decompose_root_datalist(dsphal_datalist_t *dsphal_datalist, char *format, data1, data2 …)

Function Name	int dsphal_decompose_root_datalist(dsphal_datalist_t *dsphal_datalist, char *format, data1, data2, ..)	
Argument	dsphal_datalist	Datalist decomposition pointer
	format	DSSP-HAL data structure format

		* variable-length argument
Description	dsphal_request_method_call( ) Decompose data requested by function	
Return	OK(0)	Success
	FAIL(-1)	Fail

Example Code	#define IP_ADDR_TETRA "192.168.51.10"
	#define PORT_BUMPER 50011
	
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_ret;
	int i;
	int bumper_val[8];
	
	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETRA, PORT_BUMPER);
	dsphal_tcp_client_connect(tcp_client);
	datalist_ret = dsphal_request_method_call(
	tcp_client, " ReadBumperArray ", NULL);
	if (datalist_ret) {
	dsphal_decompose_root_datalist(datalist_ret, "[{i}{i}{i}{i}{i}{i}{i}{i}]",
	&bumper_val[0],
	&bumper_val[1],
	&bumper_val[2],
	&bumper_val[3],
	&bumper_val[4],
	&bumper_val[5],
	&bumper_val[6],
	&bumper_val[7]);
	dsphal_datalist_destroy(datalist_ret);
	}
	dsphal_tcp_client_destroy(tcp_client);	

3-3. DSSP-HAL Request Example

DSSP-HAL request use examples according to the existence of argument and/or return value as well as the form of the return value when method in DSSP-HAL has been called within the network environment are shown below.

3-3-1. No Argument & Return

When there is no return and no argument after method in DSSP-HAL service has been called, DSSP-HAL request syntax is same as the codes shown below

```
#include "dsphal.h"

#define IP_ADDR_TETRA "192.168.51.10"
#define PORT_POWER 50017

int main(void)
{
```

```
	dsphal_tcp_client_t *tcp_client;

	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETRA, PORT_POWER);
	dsphal_tcp_client_connect(tcp_client);
	dsphal_request_method_call(tcp_client, "PowerDownDriveModule", NULL);
	dsphal_tcp_client_destroy(tcp_client);

	return 0;
}
```

3-3-2. Argument with No Return

When there is argument but no return value after method in DSSP-HAL service has been called, DSSP-HAL request syntax is same as the codes shown below

```
#include  "dsphal.h"

#define IP_ADDR_TETRA "192.168.51.10"
#define PORT_DRIVE 50010

int main(void)
{
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_arg;
	int lvel;
	int rvel;
	lvel = 100;
	rvel = 100;

	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETRA, PORT_DRIVE);
	dsphal_tcp_client_connect(tcp_client);
	datalist_arg = dsphal_build_root_datalist("[{i}{i}]", lvel, rvel);
	dsphal_request_method_call(tcp_client, "VelocityControl", datalist_arg);
	if (datalist_arg)
	dsphal_datalist_destroy(datalist_arg);
	dsphal_tcp_client_destroy(tcp_client);

	return 0;
}
```

3-3-3. Return Value with No Argument

When there is return value but not argument after method in DSSP-HAL service has been called, DSSP-HAL request syntax is same as the codes shown below.

```
#include <stdio.h>
#include  "dsphal.h"

#define IP_ADDR_TETRA "192.168.51.10"
#define PORT_DRIVE 50010
```

```
int main(void)
{
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_ret;
	int l_encoder;
	int r_encoder

	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETRA, PORT_DRIVE);
	dsphal_tcp_client_connect(tcp_client);
	datalist_ret = dsphal_request_method_call(tcp_client, "ReadEncoder", NULL);
	if (datalist_ret) {
	    dsphal_decompose_root_datalist(datalist_ret, "[{i}{i}]", &l_encoder, &r_encoder);
	    dsphal_datalist_destroy(datalist_ret);
	    printf("Encoder : %d, %d\n", l_encoder, r_encoder);
	}

	dsphal_tcp_client_destroy(tcp_client);

	return 0;
}
```

3-3-4. Argument & Return

When there is both return value and argument after method in DSSP-HAL service has been called, DSSP-HAL request syntax is same as the codes shown below. The difference from the previous explanation occurs when there is large number of arguments from the returned datalist as when return is from the device such as the Laser Range Finder. In such instances, DSSP-HAL request syntax becomes impractical due to the large number of "{i}" that has to be repeatedly entered. Example code below shows DSSSP-HAL request syntax with efficient method of parsing the datalist.

```
#include <stdio.h>
#include "dsphal.h"

#define IP_ADDR_TETRA "192.168.51.10"
#define PORT_RANGEFINDER 50014

int main(void)
{
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_arg;
	dsphal_datalist_t *datalist_ret;
	dsphal_data_t *data; // Pointer to receive data from the datalist
	int requested_resolution;
	int i;
	int rangefinder_data[1000];

	requested_resolution = 500;
	tcp_client = dsphal_tcp_client_create(IP_ADDR_TETRA, PORT_RANGEFINDER);
```

```
        dsphal_tcp_client_connect(*tcp_client);
        datalist_arg = dsphal_build_root_datalist("[{i}]", requested_resolution);
        datalist_ret = dsphal_request_method_call(tcp_client, "ReadRangeArray", datalist_arg);
        if (datalist_arg)
            dsphal_datalist_destroy(datalist_arg);
        if (datalist_ret) {
            /*Receive beginning pointer of the datalist */
            data = dsphal_datalist_get_head(datalist_ret);
            for (i = 0; i < requested_resolution; i++) {
                    /*Parse one integral value from the datalist */
                    rangefinder_data[i] = dsphal_data_int_get(dsphal_data_get_data(data));
                    /*Receive data pointer for next data*/
                    data = dsphal_data_get_next(data);
            }
            dsphal_datalist_destroy(datalist_ret);
        }
        for (i = 0; i < requested_resolution; i++) {
            printf("%d\n", rangefinder_ data[i]);
        }
        dsphal_tcp_client_destroy(tcp_client);

        return 0;
}
```

3-4. DSSP-HAL Service Port Configuration

DSSP-HAL service port configuration for standard and optional devices on our mobile robot platform TETAA-DS III™' is shown below in chart 3-1 and 3-2. DSSP-HAL service list shown in the charts maybe added or changed depending on the standard modules and optional devices installed on the TETAA-DS III™. This manual only includes the DSSP-HAL service material on standard devices installed on TETAA-DS III™' . Those wishing to received DSSP-HAL service material on optional devices should contact our Customer Support Center.

<Chart 3-1> DSSP-HAL Service Port Configuration of default TETAA-DS III™

DSSP-HAL Service Name	Assigned TCP Port	Related Device
drive	50010	Drive Motor
bumper	50011	Collision Detection Bumper Sensor
sensors	50012	Ultrasonic Sound Sensor
power	50017	Power B/D
emergency	50022	Emergency Stop Button

<Chart 3-2> DSSP-HAL Service Port Configuration of Option Parts for TETAA-DS III™

DSSP-HAL Service Name	Assigned TCP Port	Related Device
rangefinder	50014	Laser Scanner
cam	50015	Camera

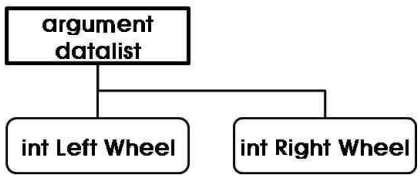
lps	50016	Localization Sensor(LPS)
gyro	50019	Gyroscope Sensor
pantilt	50020	Pan-Tilt Module
exio	50024	External input/output device
stargazer	50028	Localization Sensor(StarGazer)

Those wishing to use additional device should contact our Customer Support Center to receive support in developing DSSP-HAL service for the device.

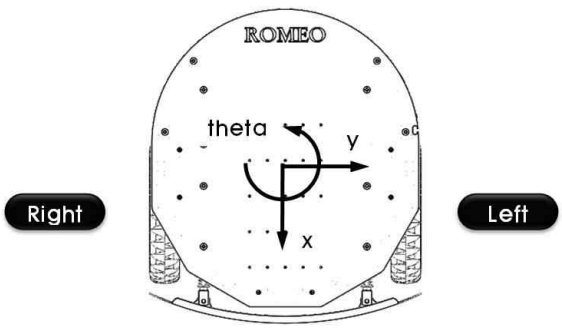
Chapter 4. DSSP-HAL Drive Service

Methods registered in the drive motor related DSSP-HAL service "drive" service provided through TCP 50010 port are as shown below. Drive motor speed can be controlled and information from the encoder installed on the drive motor can be received by calling individual methods through TCP/IP. Please contact our Customer Support Center to receive information on developing additional methods.

Method Name	VelocityControl	
Argument	datalist	Shows left/right wheel speed in integer format Left Wheel, Right Wheel data format (Refer to diagram 4-1 and 4-2) unit mm/sec.
Description	Controls speed of 2 left/right drive motors installed on the platform.	
Return	None	Null
Example Code	Refer to Sample Code	



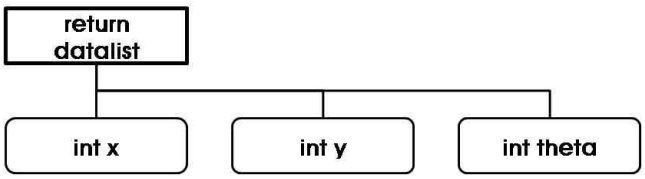
<Diagram 4-1> argument datalist of VelocityControl Method



<Diagram 4-2> Drive Motor Configuration and Coordinates System

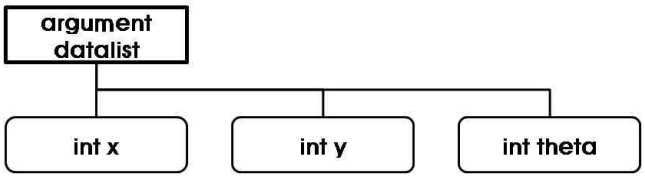
Method Name	ReadPosition	
Argument	None	Null
Description	Receives center position value of the robot through dead-reckoning	
Return	datalist	Center position integer data in x, y, theta format (Refer to diagram 4-3) x, y unit: mm, theta unit: 0.1degree.

Example Code	Refer to Sample Code



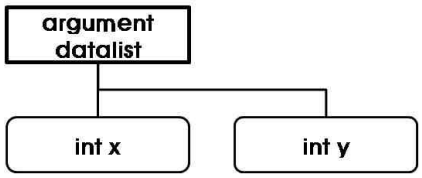
<Diagram 4-3> return datalist of ReadPosition Method

Method Name	ChangePosition	
Argument	datalist	To be renewed Integer data in x, y, theta format (Refer to diagram 4-4) x, y unit: mm, theta unit: 0.1degree.
Description	Renew center position value of the robot.	
Return	None	Null
Example Code	Refer to Sample Code	



<Diagram 4-4> argument datalist of ChangePosition Method

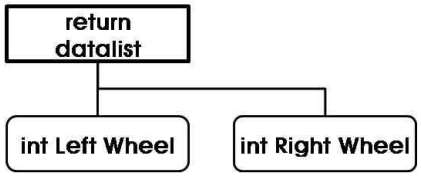
Method Name	ChangePosition2	
Argument	datalist	To be renewed Integer data in x, y, format (Refer to 4-5) x, y unit: mm.
Description	Renew center position value of the robot.	
Return	None	Null
Example Code	Refer to Sample Code	



<Diagram 4-5> argument datalist of ChangePosition2 Method

Method Name	ReadEncoder

Argument	None	Null
Description	Receives encoder data from left/right drive motor installed in the robot	
Return	datalist	Integer data in Left Wheel, Right Wheel format. (Refer to diagram 4-6)
Example Code	Refer to Sample Code	



<Diagram 4-6> return datalist of ReadEncoder Method

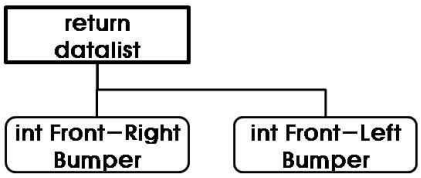
Method Name	ServoOn	
Argument	None	
Description	Turn On drive motor installed in the TETRA-DS III™	
Return	None	
Example Code	Refer to Sample Code	

Method Name	ServoOff	
Argument	None	Null
Description	Turn Off drive motor installed in the TETRA-DS III™	
Return	None	Null
Example Code	Refer to Sample Code	

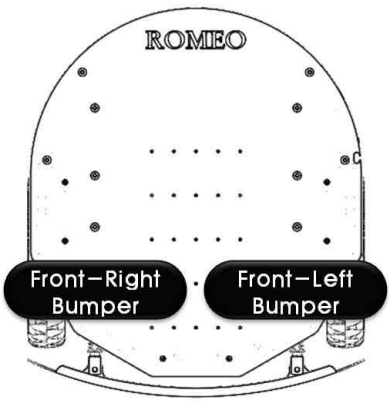
Chapter 5. DSSP-HAL bumper Service

Methods registered in the collision detection bumper sensor related DSSP-HAL service "bumper" service provided through TCP 50011 port are as shown below. Information from the bumper sensors can be received by calling individual methods through TCP/IP. Please contact our Customer Support Center to receive information on developing additional methods.

Method Name	ReadBumperArray	
Argument	None	Null
Description	Receive collision detection information from the forward bumper sensor installed in TETRA-DS III™.	
Return	datalist	Integer data Front-Left Bumper, Front-Right Bumper format. (Refer to diagram 5-1 & 5-2) Collision : 1, No Collision : 0
Example Code	Refer to Sample Code	



<Diagram 5-1> return datalist of ReadBumperArray Method



<Diagram 5-2> Bumper Sensor Configuration

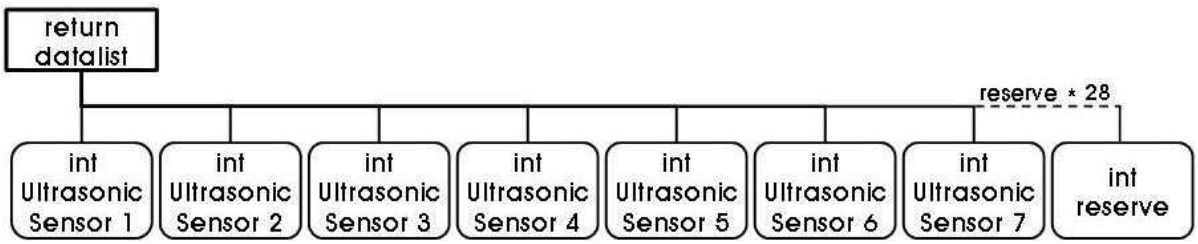
Method Name	SetBumperDirMode	
Argument	None	
Description	Set bumper sensor operation mode to 'Enable Mode'. When bumper sensors detect collision, drive motors stop	
Return	None	Null
Example Code	Refer to Sample Code	

Method Name	SetBumperOffMode"	
Argument	None	Null
Description	Set bumper sensor operation mode to 'Disable Mode'. Drive motor continues to operate even when collision detected..	
Return	None	Null
Example Code	Refer to Sample Code	

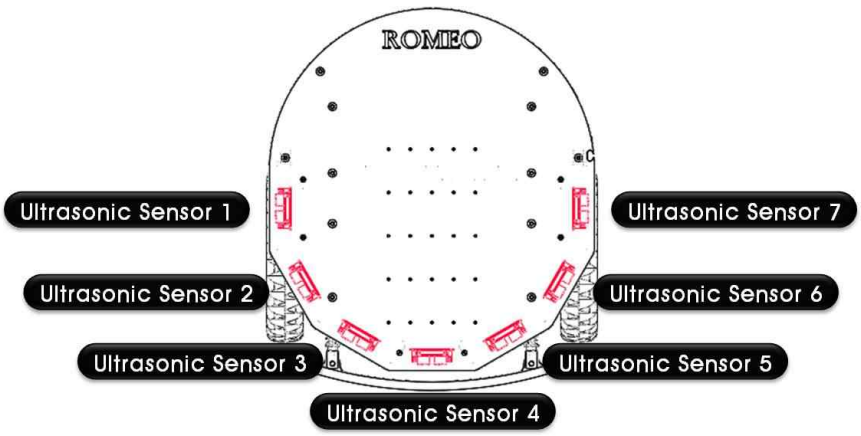
Chapter 6. DSSP-HAL ultrasonic_sensor Service

Methods registered in the ultrasonic sensor related DSSP-HAL service "usonic_sensor" service provided through TCP 50012 port are as shown below. Information from the ultrasonic sensors can be received by calling individual methods through TCP/IP. Please contact our Customer Support Center to receive information on developing additional methods.

Method Name	ReadSensors	
Argument	None	Null
Description	Receives information from 7 ultrasonic sensors installed on TETRA-DS III™. Special case: has 29 reserve data.	
Return	datalist	Integer data from 7 ultrasonic sensors showing distance to detected obstacles. (Refer to diagram 6-1 & 6-2) Unit: cm. Reserved data 28
Example Code	Refer to Sample Code	



<Diagram 6-1> return datalist of ReadUltraSonicSensorArray Method



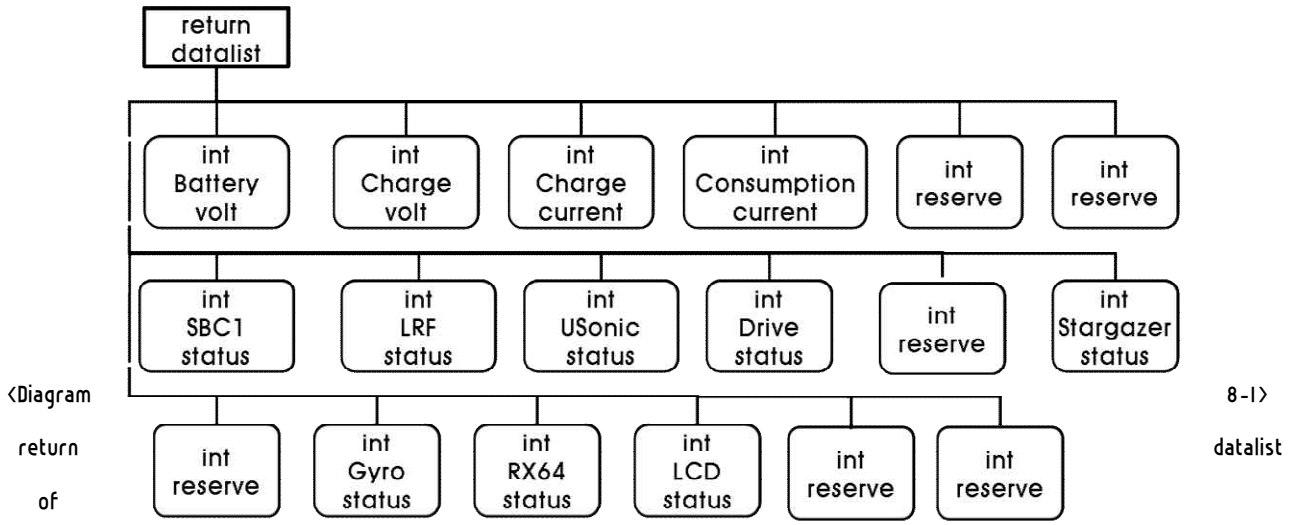
<Diagram 6-2> Ultrasonic Sensors Configuration

Method Name	SetMaxDistance	
Argument	datalist	1 : 1.5m, 2 : 2.0m, 3 : 2.5m, 4 : 3.0m, 5 : 3.5m, 6 : 4.0m , 7 : 4.5m, 8 : 5.0m, 9 : 5.5m (기본값은 3.0m)
Description	Changes the maximum range of the ultrasonic sensors by changing the single digit Integer variable. Default value is 4(3.0m).	
Return	NULL	NULL
Example Code	Refer to Sample Code	

Chapter 7. DSSP-HAL power Service

Methods registered in the system power related DSSP-HAL service "power" service provided through TCP 50017 port are as shown below. Battery information, power usage, device power status, and other power related Information can be received by calling individual methods through TCP/IP. Please contact our Customer Support Center to receive information on developing additional methods.

Method Name	ReadPowerStatus	
Argument	None	Null
Description	Receives system power related information. Shows first battery info, 9th,10th sensor board and drive board power status 1 = on, 0 = off.	
Return	datalist	Battery voltage data and 17 reserved data. (Refer to diagram 8-1)
Example Code	Refer to sample data	



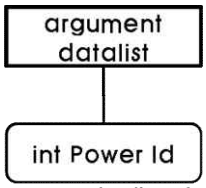
<Diagram 8-1> return of datalist

ReadPowerStatus Method

Method Name	PowerUp	
Argument	datalist	Shows detailed list of robot power management board Integer Power Id type data. (Refer to 8-2 & 8-1)
Description	Activates power on items listed on power ID list of power board	
Return	None	Null
Example Code	Refer to Sample Code	

Method Name	PowerDown	
Argument	datalist	Shows detailed list of robot power management board

		Integer Power Id type data. (Refer to 8-2 & 8-1)
Description	Deactivates power on items listed on power ID list of power board	
Return	None	Null
Example Code	Refer to sample code	



<Diagram 8-2> argument datalist of PowerUp Method

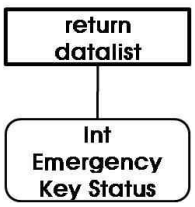
<Chart 8-1> argument datalist of Power Id list

Power Id	Name	Volt	Description
1	Drive	-	Drive power management (System Standard)
2	Ultrasonic	-	Ultrasonic sensor board power management
3	LRF	5V	Laser Scanner power management
4	SBC1	24~27V	SBC power management (use battery voltage)
5	Gyro	5V	Gyro power management
6	StarGazer	5V, 12V	Mange two power terminals for StarGazerat at same time
7	Spare1	12V	Extra 12V (max. 3A) power management
8	Spare3	12V	Extra 12V (max. 3A) power management
9	Spare2	12V	Extra 12V (max. 3A) power management
10	LCD	12V	LCD power management
11	AX64	7.4V	AX64 power management
12	Spare4	12V	Extra 12V (max. 3A) power management

Chapter 8. DSSP-HAL emergency Service

Methods registered in the emergency stop button related DSSP-HAL service "emergency" service provided through TCP 5002 port are as shown below. Information from the emergency stop button can be received by calling individual methods through TCP/IP. Please contact our Customer Support Center to receive information on developing additional methods.

Method Name	ReadEmergencyKey	
Argument	None	Null
Description	Receive emergency stop button status value. When emergency stop button is pushed power to the drive motor is cut regardless of the method request.	
Return	Datalist	Integer data showing emergency stop button status. (Refer to diagram 8-1) Button On : 1, Button Off : 0
Example Code	Refer to Sample Code	



<Diagram 9-1> return datalist of ReadEmergencyKey Method DSSP-HAL Operation Manual_EV1.0

※ Depending on the optional device, separate manual is available for those optional devices without an explanation of the DSSP-HAL method above.