DRSP-HAL
Operation Manual
(TETRA-DS IV™ Platform)

Version 1.2

2012. 01.

Dongbu Robot

Table of Contents

Chapter 1. DRSP-HAL Overview ---------- 3

1-1. What is DRSP? ---------- 3

1-2. What is DRSP-HAL? ---------- 4

1-3.Implementing DRSP-HAL API ---------- 4

1-3-1. Implementing API in same System - - - 5

1-3-2. Implementing API in diverse Systems - - - 6

Chapter 2. DRSP-HAL Structure ---------- 7

2-1. DRSP-HAL Protocol ---------- 7

2-2. DRSP-HAL Data Structure ---------- 7

2-3. DRSP-HAL Service & Request - - - - - - - - - - - - 8

2-4. DRSP-HAL Service Method - 10

Chapter 3. DRSP-HAL Service로의 접근 (Access DRSP-HAL Service) ---------- 13

3-1. DRSP-HAL Service Structure Code - 13

3-2. DRSP-HAL Service Functions - 14

3-3. DRSP-HAL Request Examples - 18

3-4. DRSP-HAL Service Port Configurations - - 2 2

Chapter 4. DRSP-HAL drive Service - 24

Chapter 5. DRSP-HAL power Service - 27

Chapter 6. DRSP-HAL emergency Service - - - - - - - - - - - - 29

Chapter 1. DRSP-HAL Overview

1-1. What is DRSP?

'DRSP (Dongburobot Software Platform) is a Network-based Modular Software Platform developed by DongbuRoobt to facilitate in developing autonomous navigational technology for mobile robot platforms. In order to support development of hardware and OS independent application programs, DRSP 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 type 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 DRSP and DRSP-MEL can be found at our website or by contacting the Customer Support Center.

<Diagram 1-1> DRSP Architecture

1-2. What is DRSP-HAL?

DRSP-HAL(Dongburobot Software Platform – Hardware Abstraction Layer) is a network based hardware abstraction layer with XML type protocol for accessing and controlling all standard and optional devices installed on TETRA-DS IV™ in simple and unified manner.
Diagram 1-2 below shows the DRSP-HAL concept.

<Diagram 1-2> DRSP-HAL Concept

As shown diagram 1-2, objective of DRSP-HAL is to make various hardware installed on the robot platform easier to use through abstraction. To achieve such an end, DRSP-HAL supports simple and easy to understand data structure and hardware layer abstraction is accomplished by distributed processing using the network which its 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 DRSP-HAL API

DRSP-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 "DRSP-HAL Service" acting as a "Server" and providing "service" and "DRSP-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 DRSP-HAL function. As DRSP-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, DRSP-HAL Requests specific DRSP-HAL service which can only be accessed and controlled through a distinct port. When DRSP-HAL service assigned to a distinct port receives a request, request is sent through the DRSP-HAL service supported method and then through the physical interface supported by the said method to control or receive data from the hardware.

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

1-3-2. API Implementation in diverse systems

As shown in diagram 1-4, to access and control the hardware within divers network system with the multiple IP addresses, DRSP-HAL Requests specific DRSP-HAL service which can only be

accessed and controlled through distinct IP address and port. When DRSP-HAL service assigned to a distinct port receives a request through the distinct IP address, request is sent through the DRSP-HAL service supported method and then through the physical interface supported by the said method to control or receive data from the hardware.

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

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

Chapter 2. DRSP-HAL Structure

2-1. DRSP-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> DRSP-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 transmitted between the "Request" and "Service" is in XML type structure

2-2. DRSP-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. Information is transmitted between the DSSP-HAL "request" and "service" containing these types of data.

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

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

.

<Diagram 2-3> XML-based Struct Code of DRSP-HAL Data

2-3. DRSP-HAL Request and Service

DRSP-HAL can be divided between "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.

DRSP-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> DRSP-HAL Request & Service Structure

"Robot library" shown is diagram 2-4 above is collection of libraries from the installed hardware and optional devices defined by UART communications protocol that could be used through the DSSP-HAL service method. For example, "TETRA- DS IV™ Library" being sold for our mobile robot platform is a type of robot library. Also, DSSP-HAL utility shown in diagram 2-4 is 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, DRSP-HAL communications protocol is 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 Struct Code of DRSP-HAL Request & Service

2-4. DRSP-HAL Service Method

When using the DRSP-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 IV™'.

```
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 DRSP-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 "TETRA 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 TETRA-DS IV™'.

```
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 DRSP-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 "DRSP-HAL Utility" function 'dsphal_registry_add_method' that adds and registers a method in DRSP-HAL service.

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

Diagram 2-10 is an example of the DRSP-HAL request codes when TETRA-DS IV™' obeys the control command to drive the left and the right wheels at speed of 100 mm/s.

```
#include "dsphal.h"  ❶

#define IP_ADDR_DMP "192.168.51.10"  ❷
#define PORT_DRIVE 50010  ❸

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);  ❹
    dsphal_request_method_call(tcp_client,  "VelocityControl", datalist_arg);  ❺
    dsphal_tcp_client_destroy(tcp_client);

    return 0;
}
```

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

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. Accessing DRSP-HAL Service

This chapter describes the key function of the DSSP-HAL. Accessing DRSP-HAL service to access and control specific hardware.

3-1. DRSP-HAL Service Structure Codes

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 TCP information of the DRSP-HAL service to be accessed dsphal_tcp_client_create( ) Return value of function	
Argument	server_ip_addr	DRSP-HAL service IP(v4) address
	server_port	DRSP-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(output) data list. 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. DRSP-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	DRSP-HAL service IP(v4) address
	server_port	DRSP-HAL service Port number
Description	Input IP and Port information of the DRSP-HAL service to be accessed.	
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	TCP information of DRSP-HAL service to be accessed.
Description	Connect to TCP of DRSP-HAL service to be accessed	
Return	OK (0)	Connection successful
	False(-1)	Connection failed

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	TCP information of connected DRSP-HAL service
Description	Disconnect from DRSP-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	DRSP-HAL data structure format * variable length argument
Description	Create datalist to be sent to DRSP-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 formt

All defined data types in DSSP-HAL are composed of either integer or string type. Char type *format variable appearing above starts with "[" and ends with "]". if the integer is to appear in between, it is shown by the matching number of "{i}" and if string is to appear, it is shown by 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	TCP information of DRSP-HAL service to be connected
	method_name	Name of method to be called
	dsphal_datalist_arg	Datalist argument
Description	DRSP-HAL 서비스에 메소드 콜(Call) 요청	
Return	None	No return value
	dsphal_datalist_t	Return value, datalist address value
Example Code	**A. When datalist argument exists** #define IP_ADDR_TETRA "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_TETRA, PORT_DRIVE);
	dsphal_tcp_client_connect(tcp_client);
	datalist_arg = dsphal_build_root_datalist("[{i}{i}]", lvel, rvel);
	datalist_ret = dsphal_request_method_call(
	tcp_client, "VelocityControl", datalist_arg);
	
	B. When datalist argument does not exist.
	dsphal_tcp_client_t *tcp_client;
	dsphal_datalist_t *datalist_ret;
	int i;
	
	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, "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 pointer of data to be decomposed.
	format	DRSP-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. DRSP-HAL Request Example

When method within the DRSP-HAL is called in network environment, DRSP-HAL request syntax varies depending on the existence of argument and/or return value as well as the format of the

return value.

3-3-1. No Argument & Return

When there is no return and no argument after method in DSSP-HAL service has been called, DRSP-HAL request syntax is as 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. No Return

When there is argument but no return value after method in DSSP-HAL service has been called, DRSP-HAL request syntax is as 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. No Argument

When there is return value but not argument after method in DSSP-HAL service has been called, DSSP-HAL request syntax is as 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 DRSP-HAL service has been called, DRSP-HAL request syntax is as same as the codes shown below. The difference from the previous explanation occurs when there are large number of arguments from the returned datalist as when return is from the device such as the Laser Range Finder. In such instances, DRSP-HAL request syntax becomes impractical due to the large number of "{i}" that has to be repeatedly entered. Example codes below show more efficient DRSP-HAL request syntax with datalist parsing method.

```
#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) {
        /*Get datalist start pointer*/
        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));
                /*Get 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. DRSP-HAL Service Port Configuration

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

<Table 3-1> DRSP-HAL Service Port Configuration of default TETRA-DS IV™

DRSP-HAL Service Name	Assigned TCP Port	Related Device
drive	50010	Dive Motor
bumper	50011	Collision detection bumper sensor
sensors	50012	Ultrasonic sensor
power	50017	Power B/D
emergency	50022	Emergency stop button

<Table 3-2> DRSP-HAL Service Port Configuration of Option Parts for TETRA-DS IV™

DRSP-HAL Service Name	Assigned TCP Port	Related Device
rangefinder	50014	Laser scnner
cam	50015	Camera
lps	50016	Localization sensor(LPS)
gyro	50019	Gyroscope sensor
pantilt	50020	Pan-tile 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 DRSP-HAL service for the device.

Chapter 4. DRSP-HAL drive Service

Methods registered in the drive motor DRSP-HAL service "drive" provided through TCP 50010 port are shown below. Functions such as drive motor speed control and reading information from the encoder installed on the drive motor is done 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 the two (left/right) drive motors installed in TETRA-DS IV™	
Return	None	Null
Example Code	Refer to Sample Code	

<Diagram 4-1> argument datalist of VelocityControl Method

<Diagram4-2> Drive Motor Configuration and Coordinates System

Method Name	ReadPosition

Argument	None	Null
Description	Receive center of gravity position value of the robot through dead-reckoning.	
Return	datalist	Center of gravity position value in integer 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	Data in Integer x, y, theta format to be refreshed (Refer to diagram 4-3) x, y unit: mm, theta unit: 0.1degree
Description	Refresh center of gravity 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	Data in integer x, y format to be refreshed. (Refer to diagram 4-5) x, y unit: mm.
Description	Refresh center of gravity position value of the robot	
Return	None	Null

Example Code	Refer to sample code

<Diagram4-5> 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	Data from left/right wheel in integer 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 driver of the TETRA-DS IV™ drive motor.	
Return	None	
Example Code	Refer to sample code	

Method Name	ServoOff	
Argument	None	Null
Description	Turn off the driver of the TETRA-DS IV™ drive motor.	
Return	None	Null
Example Code	Refer to sample code	

Chapter 5. DRSP-HAL power Service

Methods registered in the system power DSSP-HAL service "power" provided through TCP 50017 port are as shown below. Battery information, power usage, device power status, and other power related Information is 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	Receive system power related information. Shows first battery info, 9th,10th sensor board and drive board power status Status: 1 = on, 0 = off.	
Return	datalist	Battery voltage data and 17 reserved data. (Refer to diagram 5-1)
Example Code	Refer to sample code	

<Diagram5-1> return datalist of ReadPowerStatus Method

Method Name	PowerUp	
Argument	datalist	Integer Power ID type data showing detailed list of TETRA-DS IV™ power management board. (Refer to diagram 5-2 and table 5-1)
Description	Activate all items in power management board list	

Return	None	Null
Example Code	Refer to sample code	

Method Name	PowerDown	
Argument	datalist	Integer Power ID type data showing detailed list of TETRA-DS IV™ power management board. (Refer to diagram 8-2 and table 8-1)
Description	Deactivate all items in power management board list	
Return	None	Null
Example Code	Refer to sample code	

<Diagram 5-2> argument datalist of PowerUp Method

<Table 5-1> argument datalist of Power Id list

Power Id	Name	Volt	Description
1	Drive	-	Drive power maanagemet (Standard system feature)
2	Ultrasonic	-	Ultrasonic sensor board power management
3	LRF	5V	Laser scanner power management
4	SBC1	24~27V	SBC power management (Battery voltage used)
5	Spare5	5V	Spare 12V (max. 3A) power management
6	StarGazer	5V, 12V	Manage two power terminals for StarGazer at same time
7	Spare1	12V	Spare 12V (max. 3A) power management
8	Spare3	12V	Spare 12V (max. 3A) power management
9	Spare2	12V	Spare 12V (max. 3A) power management
10	LCD	12V	LCD power management
11	Smart Servo	7.4V	Smart Servo power management
12	Spare4	12V	Spare 12V (max. 3A) power management

Chapter 6 DRSP-HAL emergency Service

Methods registered in the emergency stop button DSSP-HAL service "emergency" provided through TCP 5002 port are as shown below. Information from the emergency stop button is 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 pressed, all control to the drive motor is cut regardless of the method request.	
Return	datalist	Integer data showing emergency stop button status. (Refer to diagram 6-1) Button On : 1, Button Off : 0
Example Code	Sample Code 참조	

<Diagram 6-1> return datalist of ReadEmergencyKey Method

※ Refer to the 'DRSP-HAL Optional Devices Operation Manual' for DRSP-HAL information on TETRA-DS IV™ supported option devices.