Documentation, pre-released

created: 27.01.2016
author: support@iseg-hv.de
version: 1.1.1.5

iseg HARDWARE ABSTRACTION LAYER - isegHAL

The isegHAL library offers a string based application interface - API. The data collection from the iseg high voltage modules is done in background. All communication hand shake is handled by the isegHAL.

1.1. COMMUNICATION VIA ISEG ITEMS

The communication takes place about iseg items. An iseg item consists of an object, a value, the timestamp item last refreshed, the timestamp item last changed and the quality for that value. The objects representing the fully qualified names of the data points from the connected hardware (hardware interface or virtual lines, crates and modules). That means such an object specifies the hierarchical sequence to address a specific data point in the connected system.

Any item will be addressed via the fully qualified names. A fully qualified name consists of up to four layers:

- system ; system information
- line ; hardware interface or virtual line (0..15)
- device ; module (0..63), crate (1000, 2000..2015) [1]
- channel ; high voltage channels (0..63)

Items of the system layer come without any additional address coding and owns their item name only e.g. CycleCounter.
The other layers will be divided by a dot e.g. 0.1.2.VoltageSet.
The layer “line” can address up to 16 hardware interfaces or virtual lines of iseg master-slave crate systems e.g. an ECH44A as master and an ECH43A plus an ECH238 as slave crates.
The layer “device” addresses up to 64 HV modules and up to 16 crates.
The layer “channel” addressed up to 48 channels.

Example for such an object consisting of line.device.channel.item:

0.1.2.VoltageSet
line(0).module(1).channel(2).item

0.1000.Status
line(0).crate(0).item

2.63.47.CurrentMeasure
line(2).module(63).channel(47).CurrentMeasure

[1] isegHAL since version 1.1.0.0

1.2. COMMON FUNCTIONS

1.2.1. TYpe Iseg RESULT

```
typedef enum IsegResult {
        ISEG_OK = 0,
        ISEG_ERROR = 1,
        ISEG_WRONG_SESSION_NAME,
        ISEG_WRONG_USER,
        ISEG_WRONG_PASSWORD,
        ISEG_NOT_AUTHORIZED
} IsegResult;
```

1.2.2. Function iseg_getVersion

unsigned int iseg_getVersion(void)

Returns the library four digit version number as unsigned int value.

Example: 0x01010000

1.2.3. Function iseg_getVersionString

const char *iseg_getVersionString(void)

Returns the library four digit version number as null-terminated string.

Example: "1.1.0.0"

1.2.4. Function iseg_connect

Connects the isegHAL via the specified hardware interface to the iseg devices.

IsegResult iseg_connect(const char *name, const char *interface, void *reserved) [1]

Parameter name is a free user selectable label to identify the connection (for example “IF0”). This label is used for all following function calls to set and get.
Parameter interface: can0, can1, .. can15; (in future designated usb0, usb1; com1; ip_address)
The string value “test” for name and interface is forbidden.
Parameter reserved for future applications, at the moment a zero pointer should be used.

Returns ISEG_OK for a successful connection. If the IsegResult value is unequal to zero an error is occurred and the connection fails.

Example:

```
iseg_connect(“can0”, “CAN0”, NULL);
iseg_connect(“usb1”, “iseg://COM17” , NULL);
iseg_connect(“usb1”, “scpi://ttyUSB0” , NULL);
iseg_connect(“eth2”, “hal://192.168.16.56:1454/can0,user,pass”,NULL);
iseg_connect(“eth2”, “hal://ics.iseg-hv.com:1454/can0,user,pass”,NULL);
iseg_connect(“eth2”, “snmp-iseg://192.168.16.245,username=iseg,password=geheim”, NULL);
iseg_connect(“eth2”, “snmp-wiener://192.168.16.245:10001”, NULL);
```

1.2.5. Function iseg_disconnect

Disconnects the connected interface and stops the communication threads.

IsegResult iseg_disconnect(const char *name) [1]

Parameter name must be the same that is used for *iseg_connect().*
Returns ISEG_OK for a successful disconnection. If the IsegResult value is unequal to zero an error is occurred and the connection fails.

[1] isegHAL since version 1.1.0.0

1.3. ISEG ITEM FUNCTIONS

1.3.1. Structure IsegItem

```
typedef struct IsegItem {
  char object[FULLY_QUALIFIED_OBJECT_SIZE]; // line.module.channel.item 0.0.1.VoltageSet
  char value[VALUE_SIZE];
  char quality[QUALITY_SIZE];
  char timeStampLastRefreshed[TIME_SIZE];
  char timeStampLastChanged[TIME_SIZE];
} IsegItem; [1]
```

object: The objects representing the fully qualified names of the data points from the connected hardware. That means such an object specifies the hierarchical sequence to address a specific data point in the connected system (interface line, device and channel address). Example “0.0.0.Status.VoltageMeasure”

value: the value of that item.

quality: the quality of the item.

timeStampLastRefreshed: the UTC time of the last item refresh by the isegHAL (lower hardware abstrction layer). Time in seconds separated by a dot from the milliseconds and the tenth part of milliseconds since 1970-01-01 00:00:00 UTC.

timeStampLastChanged:the UTC time of the last item changement by the isegHAL (lower hardware abstrction layer). Time in seconds separated by a dot from the milliseconds and the tenth part of milliseconds since 1970-01-01 00:00:00 UTC.

Valid object strings for system items:

- "Status"
- “CrateNumber”
- “CrateList”
- “ModuleNumber”
- “CycleCounter”
- “Configuration”
- “LogLevel”
- “LogPath”
- “LiveInsertionMode”

- “ServerVersion”[2]
- “NetworkTimeout”[2] to set the NetworkTimeout use before iseg_connect()
- “SessionName”[2] generates a unique session name

Valid object strings for line items:

- "Status"
- “BitRate”
- “ModuleNumber”
- “ModuleList”
- “CrateNumber”
- “CrateList”
- “Read”[1]
- “Write”[1]
- “LogLevel”[1]

Valid object strings for device (crate) items:

- "Connected"
- "Alive"
- "Control"
- "Status"
- "EventStatus"
- “EventMask”
- “FanSpeed”
- “PowerOn”
- “SerialNumber”
- "DeviceClass"
- "FirmwareRelease"
- "FirmwareName”
- “Temperatures[i]”
- “Supplies[j]”
- “Article”

i 0..8; j 0..8

[1] isegHAL since version 1.1.0.0
[2] isegHAL since version 1.1.1.4

Valid object strings for device (module) items:

- "Connected"
- "Alive"
- "Control"
- "Status"
- "EventStatus"
- “EventMask”
- “SampleRate”
- “DigitalFilter”
- "VoltageRampSpeed"
- "CurrentRampSpeed"
- “VoltageLimit”
- “CurrentLimit”
- "Temperature"
- "SerialNumber"
- "ChannelNumber"
- "DeviceClass"
- "FirmwareRelease"
- "FirmwareName"
- “Temperatures[i]”
- “Supplies[i]”
- “HighVoltageOk”
- “Article”

i 0..7

The items “Alive” and “Connected” displays basically the same information. The item “Alive” is recommended for the common operation instead the item “Connected” can be used for maintenance because in addition it is writable in order to set a HV module back to disconnect state.

Valid object strings for channel items:

- "VoltageSet"
- "CurrentSet"
- "VoltageMeasure"
- "CurrentMeasure"
- "VoltageBounds"
- "CurrentBounds"
- "VoltageNominal"
- "CurrentNominal"
- "Control"
- "Status"
- "EventStatus"
- “EventMask”
- “DelayedTripAction”
- “DelayedTripTime”
- “ExternalInhibitAction”
- “TemperatureExternal”
- “VctCoefficient”

Allowed data types of item value:

• DataTypeUI1	“UI1”	one byte unsigned integer value
• DataTypeUI4	“UI4”	four byte unsigned integer value
• DataTypeR4	“R4”	four byte floating point value
• DataTypeString	“STR”	string value
• DataTypeBool	“BOOL”	boolean value

Definitions for quality of item values:

- ISEG_ITEM_QUALITY_INVALID “000”
- ISEG_ITEM_QUALITY_INITIALIZE “001”
- ISEG_ITEM_QUALITY_OK "002"
- ISEG_ITEM_QUALITY_COMMUNICATION_BAD "003"
- ISEG_ITEM_QUALITY_ERROR "004

Definition for time stamps of item values:
There are two time stamps for the item values. The timeStampLastRefreshed will be actualized whenever data from the corresponding data point has been received. The timeStampLastChanged will be actualized when data from the corresponding data point has been received and these new data is different to the cached data. The time stamp represents the seconds since the 01.01.1970 with a resolution of 100 microseconds.

Examples:

{ “0.1.VoltageRampSpeed”, “1.234”, “002”, “1431352497.9019 “, “1431352497.8472”}
{ “0.1.15.VoltageSet”, “1234.567”, “002”, “1431352686.5022 “, “1431352415.3583” }

1.3.2. Item LogLevel and LogPath

The items LogLevel and LogPath configures the logging function for debugging purposes. LogLevel is pre-initialized to value 3 and LogPath to “/tmp.

```
LogErrors = 1                  //log CAN bus error messages, loss of communication (Stop)
LogInformation = 2             //log state of initializing, switch on/off live insertion mode
LogApiRead = 4                 //log API read accesses (APIRX)
LogApiWrite = 8                //log API write accesses (APITX)
LogCanReceive = 32             //log CAN receive messages (CANRX)
LogCanReadRequest = 64         //log CAN read request messages (CANRQ)
LogCanWriteRequest = 128       //log CAN write request messages (CANTX)
```

1.3.3. Function iseg_setItem

Writes an iseg item value which is specified by the item object to the data point.

IsegResult iseg_setItem(const char *name, const char *object, const char *value) [1]

Parameter name is a free user selectable label to identify the connection (for example “IF0”).
Parameter object contains the fully qualified name for setting a new item value.
Parameter value contains the new value.
Return value is ISEG_OK if the object was stored into the data cache or ISEG_ERROR
The value will be stored into the data cache, written to the device in background process and read back afterwards.

Example: iseg_setItem(“IF0”, “0.4.1.VoltageSet”, “1000.5”);

There are two different possibilities to set a control, an event status or an event mask data point, as a word instruction or as a bit array instruction.
Example: iseg_setItem(“IF0”, “0.4.1.Control”, “8”);
Switch on HV channel 1 via an instruction by word manipulation. Please note, that only bit 3, HV on will be set. All other bits will be reset!
iseg_setItem(“IF0”, “0.4.1.Control:3”, “1”);
Switch on HV channel 1 via an instruction by bit manipulation. Please note, that bit 3, HV on will be set without affecting the other bits of the control word!

Please see the attachment for a detailed description of the Control-, Status-, EventStatus and EventMask registers.

[1] isegHAL since version 1.1.0.0

1.3.4. Function iseg_getItem

Read an item value, specified by the item object, from the data cache

IsegItem iseg_getItem(const char *name, const char *object) [1]

Parameter name is a free user selectable label to identify the connection (for example “IF0”).
Parameter object contains the fully qualified name to read out an iseg item from data cache.
Return value is an IsegItem.

- if the function is interrupted in cause of a wrong object string, the item quality contains ISEG_ITEM_QUALITY_ERROR

A request of “0.Status” from that function returns the state of the connected interface:

CAN interface:

- CAN_STATE_ERROR_ACTIVE_STRING "00"
- CAN_STATE_ERROR_WARNING_STRING "01"
- CAN_STATE_ERROR_PASSIVE_STRING "02"
- CAN_STATE_BUS_OFF_STRING "03"
- CAN_STATE_STOPPED_STRING "04"
- CAN_STATE_SLEEPING_STRING "05"
- CAN_STATE_SERIOUSLY_ERROR_STRING "06"
- CAN_STATE_UNDEFINED_STRING "99"

Other interfaces will be supported in future.

Example: iseg_getItem(“IF0”, “0.4.1.VoltageMeasure”);
Object: “0.4.1.VoltageMeasure”
Value: “100.345”
Quality: “002”
TimeStampLastRefreshed: “1431520867.2152“
TimeStampLastChanged: “1431520867.0982”

Example : iseg_getItem(“IF0”, “0.15.Supplies[3]”);
Object: “0.4.1.Supplies[3]”
Value: “15.2358”
Quality: “002”
TimeStampLastRefreshed: “1431520992.6796“
TimeStampLastChanged: “1431520992.6796”

Example: iseg_getItem(“IF0”, “0.ModuleList”);
Object: “0.ModuleList”
Value: “0,2,4,8”
Quality: “002”
TimeStampLastRefreshed: “1431518076.5514“
TimeStampLastChanged: “1431518076.5514”

Example Crate Status: iseg_getItem(“IF0”, “0.1000.Status”);
Object: “0.1000.Status”
Value: “327680”
Quality: “002”
TimeStampLastRefreshed: “1431521205.1390 “
TimeStampLastChanged: “1431518076.6171”

[1] isegHAL since version 1.1.0.0

1.3.5. Function iseg_getItemString

Read an item value from the data cache without the use of structures. For applications like LabView this second read function uses basic data types only.

IsegItem iseg_getItemString(const char *name, const char *object, const char *result, int size) [1]

Parameter name is a free user selectable label to identify the connection (for example “IF0”).
Parameter object contains the fully qualified name to read out an iseg item from data cache.
Parameter result references an IsegItem formatted as semicolon-separated-string.
Parameter size: The size of the given buffer.
Return value is ISEG_OK if the object could address a valid IsegItem of data cache or ISEG_ERROR

Example:
iseg_getItemString("can0", "0.4.1.VoltageMeasure", buffer, sizeof(buffer));
buffer = “0.4.1.VoltageMeasure;100.345;002;1431520867.2152;1431520867.0982”

[1] isegHAL since version 1.1.1.4

1.4. ISEG ITEM PROPERTIES

1.4.1. Structure isegItemProperty

```
typedef struct IsegItemProperty {
  char object[FULLY_QUALIFIED_OBJECT_SIZE]; //line.module.channel.item 0.0.1.VoltageSet
  char type[DATA_TYPE_SIZE]; // “R4”, “UI4”,  “STR”, “BOOL”
  char unit[UNIT_SIZE]; // "V", "A", "C", "%", "%/s", "s"
  char access[ACCESS_SIZE]; // "R", "W", "RW"
  char quality[QUALITY_SIZE];
} IsegItemProperty;
```

Valid object strings for system item properties:

- "Status"
- “BitRate”
- “CrateNumber”
- “CrateList”
- “ModuleNumber”
- “CycleCounter”
- “Configuration”
- “Read”
- “Write”
- “LogLevel”
- “LogPath”
- “LiveInsertionMode”

- “ServerVersion”
- “NetworkTimeout” to set the NetworkTimeout use before iseg_connect()
- “SessionName” generates a unique session name

Valid object strings for line item properties:

- "Status"
- “BitRate”
- “ModuleNumber”
- “ModuleList”
- “CrateNumber”
- “CrateList”

Valid object strings for device (crate) items properties:

- "Connected"
- "Alive"
- "Control"
- "Status"
- "EventStatus"
- “EventMask”
- “FanSpeed”
- “PowerOn”
- “SerialNumber”
- "DeviceClass"
- "FirmwareRelease"
- "FirmwareName
- “Temperatures[i]”
- “Supplies[j]”
- “Article”

i 0..8, j 0..8

Valid object strings for device (module) item properties:

- "Connected"
- "Alive"
- "Control"
- "Status"
- "EventStatus"
- “EventMask”
- “SampleRate”
- “DigitalFilter”
- "VoltageRampSpeed"
- "CurrentRampSpeed"
- “VoltageLimit”
- “CurrentLimit”
- "Temperature"
- "SerialNumber"
- "ChannelNumber"
- "DeviceClass"
- "FirmwareRelease"
- "FirmwareName"
- “Temperatures[i]”
- “Supplies[i]”
- “HighVoltageOk”
- “Article”

i 0..7

Valid object strings for channel items properties:

- "VoltageSet"
- "CurrentSet"
- "VoltageMeasure"
- "CurrentMeasure"
- "VoltageBounds"
- "CurrentBounds"
- "VoltageNominal"
- "CurrentNominal"
- "Status"
- "Control"
- "EventStatus"
- “EventMask”
- “DelayedTripAction”
- “DelayedTripTime”
- “ExternalInhibitAction”
- “TemperatureExternal”
- “VctCoefficient”

Allowed data types of item value:

DataTypeUI1	“UI1	one byte unsigned integer value
DataTypeUI4	“UI4	four byte unsigned integer value
DataTypeR4	“R4”	four byte floating point value
DataTypeString	“STR”	string value
DataTypeBool	“BOOL”	boolean value

Units of Item properties:

"V"	Volt
"A"	Ampere
"s"	Second
"%/s"	Percent per Second
"C"	Degree Celsius
“b/s”	Bit per scond

Access rights of Item properties:

- "R" item is readable
- "W" item is writable
- "R/W" item is read-/ writable

1.4.2. Function iseg_getItemProperty

Read an item property, specified by the item object, from the data cache.

IsegItemProperty iseg_getItemProperty(const char *name, const char *object)

Parameter name is a free user selectable label to identify the connection (for example “IF0”).
Parameter object contains the fully qualified name to read out iseg item properties.
Return value is an IsegItemProperty {object, datatype, unit, access, quality}

- if the function is interrupted in cause of a wrong object string, the item quality contains ISEG_ITEM_QUALITY_ERROR

Example: iseg_getItemProperty(“IF0”, “0.4.1.VoltageSet”);
Object: “0.4.1.VoltageSet”
DataType: “R4”
Unit: “V”
Access rights: “RW”
Quality: “002”

Example: iseg_getItemProperty(“IF0”, “0.4.FirmwareName”);
Object: “0.4.FirmwareName”
DataType: “STR”
Unit: “”
Access rights: “R”
Quality: “002”

Example: iseg_getItemProperty(“IF0”, “0.4.1.FirmwareName”); (item doesn’t exist)
Object: “0.4.1.FirmwareName”
DataType: “”
Unit: “”
Access rights: “”
Quality: “004”

Example: iseg_getItemProperty(“IF0”, “0.4.1.Control”);
Object: “0.4.1.Control”
DataType: “UI4”
Unit: “”
Access rights: “RW”
Quality: “002”

Example: iseg_getItemProperty(“IF0”, “0.4.1.Control:3”);
Object: “0.4.1.Control:3”
DataType: “BOOL”
Unit: “”
Access rights: “RW”
Quality: “002”

1.4.3. Function iseg_getItemPropertyString

Read iseg item properties without the use of structures. For applications like LabView this second read function uses basic data types only.

IsegItem iseg_getItemPropertyString(const char *name, const char *object, const char *result, int size) [1]

Parameter name is a free user selectable label to identify the connection (for example “IF0”).
Parameter object contains the fully qualified name to read out the iseg item properties.
Parameter result references an IsegItemProperty formatted as semicolon-separated-string.
Parameter size: The size of the given buffer.
Return value is ISEG_OK if the object could address a valid IsegItemProperty or ISEG_ERROR

Example:
iseg_getItemPropertyString("test", "0.4.1.VoltageMeasure", buffer, sizeof(buffer));
buffer = “0.4.1.VoltageMeasure;R4;V;R;002”

[1] isegHAL since version 1.1.1.4

1.5. CAN TUNNEL ACCESS FOR MAINTAINACE

The CAN Tunnel access is implemented to access any of the data points via EDCP or NMT Messages. Before the CAN Tunnel can be used, the isegHAL must be configured.

Configuration isegHAL to have an access via the CAN Tunnel:
iseg_setItem{“IF0”, “Configuration”, “1”);
Configuration isegHAL for normal working:
Iseg_setItem(“IF0”, “Configuration”, “0”);

Write access via the CAN Tunnel: iseg_setItem(“IF0”, “Write”, “004#C8”);

Read access via the CAN Tunnel: iseg_setItem(“IF0”, “Read”, “201#1000”);
isegItem = iseg_getItem(“IF0”, “Read:0”);
isegItem == {“Read:0”,”201#1000”, “”, “”, “”}
isegItem = iseg_getItem(“IF0”, “Read:1”);
isegItem == {“Read:1”,” 204#10001800”, “”, “”, “”}

1.6. APPENDIX BIT ADDRESSING CRATE

Control:25	do set crate enable active (CC44A only)
Control:24	set crate enable active (CC44A only)
Control:1	do clear statistic (CC44A only)
Control:0	do clear all events
Status:21	crate fast off (CC44A only)
Status:20	crate enabled (CC44A only)
Status:19	shut down (CC44A only)
Status:18	high voltage on (CC44A only)
Status:17	power fail
Status:16	power on
Status:12	sum error (CC44A only)
Status:11	high +3.3 crate controller (CC44A only)
Status:10	low +3.3 crate controller (CC44A only)
Status:9	high +5 crate controller (CC44A only)
Status:8	low +5 crate controller (CC44A only)
Status:7	high temperature
Status:6	service (CC44A only)
Status:5	high +24 backplane
Status:4	low +24 backplane
Status:3	high +5 backplane
Status:2	low +5 backplane
Status:1	high +24 battery
Status:0	low +24 battery
EventStatus:19	event shut down (CC44A only)
EventStatus:18	event high voltage on (CC44A only)
EventStatus:17	event power fail
EventStatus:16	event power on
EventStatus:11	event high +3.3 crate controller (CC44A only)
EventStatus:10	event low +3.3 crate controller (CC44A only)
EventStatus:9	event high +5 crate controller (CC44A only)
EventStatus:8	event low +5 crate controller (CC44A only)
EventStatus:7	event high temperature
EventStatus:6	event service (CC44A only)
EventStatus:5	event high +24 backplane
EventStatus:4	event low +24 backplane
EventStatus:3	event high +5 backplane
EventStatus:2	event low +5 backplane
EventStatus:1	event high +24 battery
EventStatus:0	event low +24 battery
EventMask:19	mask event shut down (CC44A only)
EventMask:18	mask event high voltage on (CC44A only)
EventMask:17	mask event power fail (CC44A only)
EventMask:16	mask event power on (CC44A only)
EventMask:11	mask event high +3.3 crate controller (CC44A only)
EventMask:10	mask event low +3.3 crate controller (CC44A only)
EventMask:9	mask event high +5 crate controller (CC44A only)
EventMask:8	mask event low +5 crate controller (CC44A only)
EventMask:7	mask event high temperature (CC44A only)
EventMask:6	mask event service (CC44A only)
EventMask:5	mask event high +24 backplane (CC44A only)
EventMask:4	mask event low +24 backplane (CC44A only)
EventMask:3	mask event high +5 backplane (CC44A only)
EventMask:2	mask event low +5 backplane (CC44A only)
EventMask:1	mask event high +24 battery (CC44A only)
EventMask:0	mask event low +24 battery (CC44A only)

1.7. APPENDIX BIT ADDRESSING MODULE

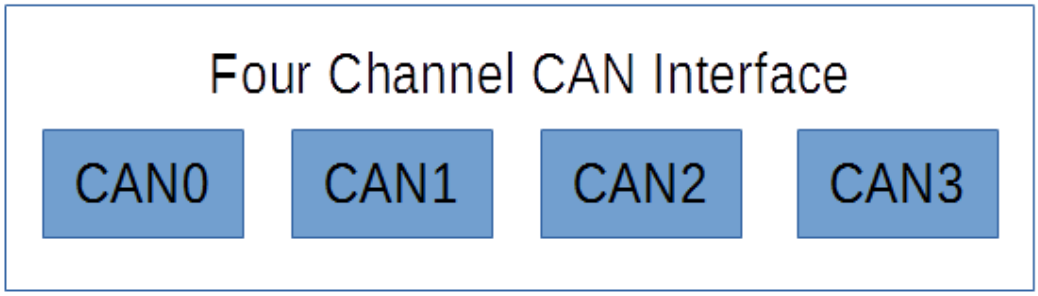
Control:14	set kill enable
Control:12	set fine adjustment
Control:6	do clear all events
Status:15	is kill enable
Status:14	is temperature good
Status:13	is supplies good
Status:12	is module good
Status:11	is event active
Status:10	is safety loop good
Status:9	is no ramp
Status:8	is no sum error
Status:6	is input error
Status:5	is hardware voltage limit good [1]
Status:4	is service needed
Status:2	is live insertion
Status:0	is fine adjustment active
EventStatus:14	event temperature not good
EventStatus:13	event supplies not good
EventStatus:10	event safety loop not good
EventStatus:6	event input error
EventStatus:5	event hardware voltage limit not good [1]
EventStatus:4	event service needed
EventStatus:2	event live insertion
EventMask:14	mask event temperature not good
EventMask:13	mask event supplies not good
EventMask:10	mask event safety loop not good
EventMask:6	mask event input error
EventMask:5	mask event hardware voltage limit not good [1]
EventMask:4	mask event service needed
EventMask:2	event live insertion

[1] modules with FirmwareName E16D0 only

1.8. APPENDIX BIT ADDRESSING CHANNEL

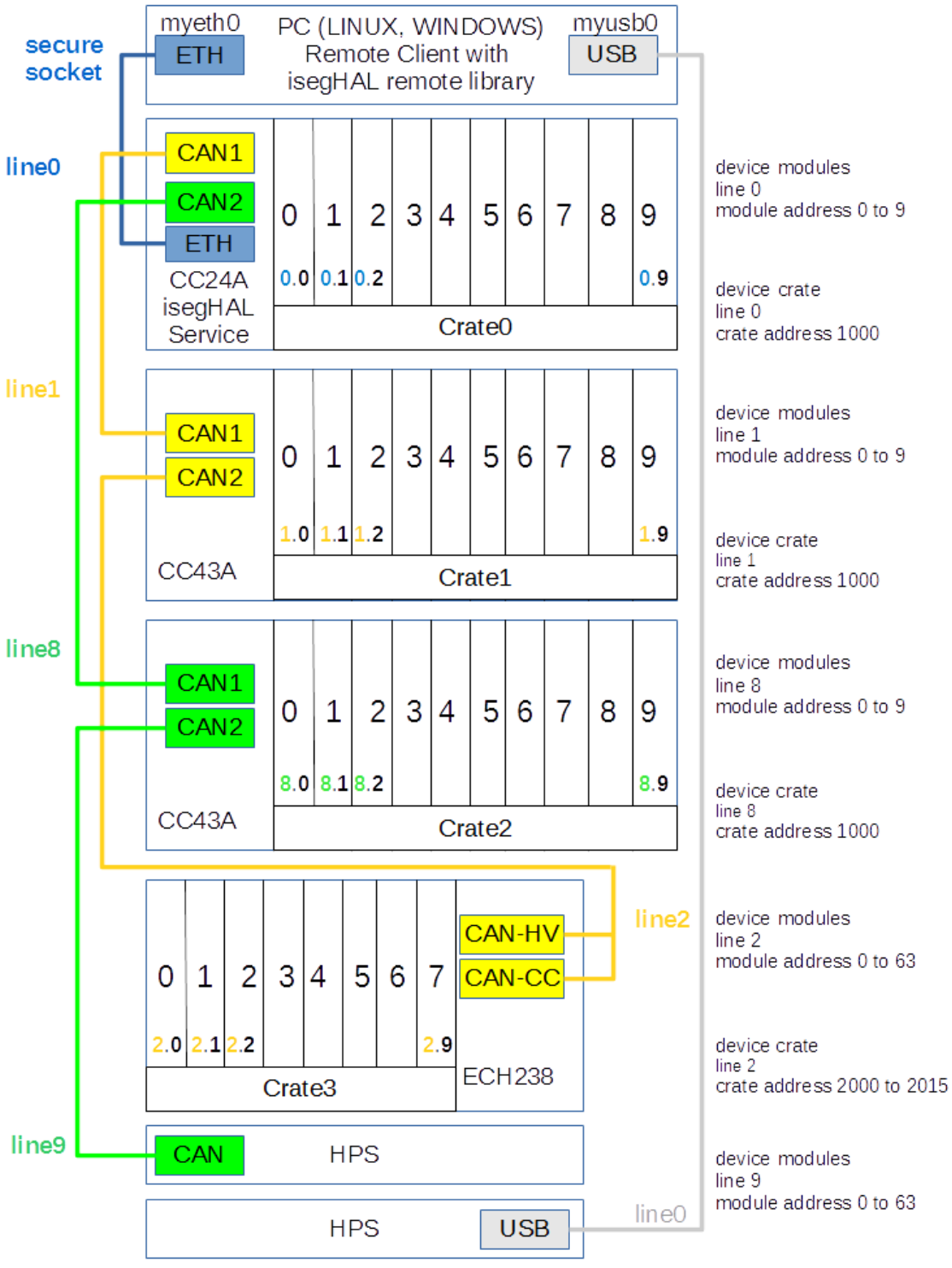
Control:5	set emergency off
Control:3	set on
Status:15	is voltage limit exceeded
Status:14	is current limit exceeded
Status:13	is trip exceeded
Status:12	is external inhibit
Status:11	is voltage bounds exceeded
Status:10	is current bounds exceeded
Status:9	is arc error
Status:8	is low current range
Status:7	is constant voltage
Status:6	is constant current
Status:5	is emergency off
Status:4	is ramp
Status:3	is high voltage on
Status:2	is input error
Status:1	is arc / is regulation error
EventStatus:15	event voltage limit exceeded
EventStatus:14	event current limit exceeded
EventStatus:13	event trip
EventStatus:12	event external inhibit
EventStatus:11	event voltage bounds exceeded
EventStatus:10	event current bounds exceeded
EventStatus:9	event arc error
EventStatus:7	event constant voltage
EventStatus:6	event constant current
EventStatus:5	event emergency off
EventStatus:4	event end of ramp
EventStatus:3	event on to off
EventStatus:2	event input error
EventStatus:1	event arc or event regulation error
EventMask:15	mask event voltage limit exceeded
EventMask:14	mask event current limit exceeded
EventMask:13	mask event trip
EventMask:12	mask event external inhibit
EventMask:11	mask event voltage bounds exceeded
EventMask:10	mask event current bounds exceeded
EventMask:9	mask event arc error
EventMask:7	mask event constant voltage
EventMask:6	mask event constant current
EventMask:5	mask event emergency off
EventMask:4	mask event end of ramp
EventMask:3	mask event on to off
EventMask:2	mask event input error
EventMask:1	mask event arc or mask event regulation error

1.9. CAN SYSTEM STRUCTURES OVERVIEW




```
iseg_connect("can0", "CAN0");  // line0
iseg_connect("can1", "CAN1");  // line1
iseg_connect("can2", "CAN3");  // line2
```

1.10. SYSTEM STRUCTURES OVERVIEW



Examples connect:

```
iseg_connect("can0", "CAN0", NULL);          // isegHAL within CC44
iseg_connect("myeth0", "s://IpAddess:Port/can0",Login,Password", NULL);
                                             //isegHAL remote service
iseg_connect("usb0", "COM10", NULL);         // isegHAL USB SCPI
```

Example get item voltage measurement for crate2, module3, channel4:
IsegItem vmeas = iseg_getItem(“myeth0”,” 8.5.6.VoltageMeasurement”);

Example set voltage set for crate2, module3, channel4:
Int isegResult = iseg_setItem(“myeth0”, “8.5.6.VoltageSet”,”1000”);