**NVHR HMI**

**Software Design Document**

**Volution Ventilation Group**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Date** | **Authors** | **Changes** |
| *0.1* | *17th July 2018* | *Hormoz Rebati* | *Draft Issue* |

**Referenced Documents**

The following items are referenced throughout this document:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref** | **Document No.** | **Issue** | **Description** |
|  | NVHR GUI (BB NVHR GRAPHICS) | V20 | NVHR HMI Graphics requirements |
|  | Commissioning GUI | V5.3 | Commissioning Screens (Setup Screens) |
|  | NVHR Advanced Controls Spec – BB Control Spec | DRAFT 4 Rev1 | A.K.A. Low Cost Controller – NVHR System |
| 4. | NVHR Advance Controls Protocol at a Glance | V1.0 | EEPROM Memory Map, NVHR Vars and Message from Mainboard to Peripheral and messages from Peripheral to Mainboard |
| 5. | NVHR Advance Controls HMI Home Screen Logic | V 1.03 | The logic for Home Screen states |
| 6. | NVHR Advance Controls Shared hardware IO | V1.0 | NVHR LCC Shared Hardware I/O and Model |

Glossary

|  |  |
| --- | --- |
| **Key** | **Description** |
| *BACnet* | *Building Automation Controls Network* |
| *BB* | *Breathing Buildings* |
| *BMS* | *Building Management System* |
| *EEPROM* | *Electrically Erasable Programmable Read-Only Memory* |
| *GUI* | *Graphical User Interface* |
| *HMI* | *Human Machine Interface* |
| *MCB* | *Main Controller Board* |
| *NVHR* | *Natural Ventilation with Heat Recovery* |
| *RTC* | *Real-Time Clock* |
| *UART* | *Universally Asynchronous Receiver/Transmitter* |
| *UTL* | *Utility tools* |

[1. Introduction 6](#_Toc520355844)

[1.1 Scope 6](#_Toc520355845)

[1.1.1 Identification 6](#_Toc520355846)

[1.1.2 Exclusions 7](#_Toc520355847)

[1.1.3 SYSTEM Overview 7](#_Toc520355848)

[1.1.3.1 Overview 7](#_Toc520355849)

[1.1.3.2 System Context 8](#_Toc520355850)

[2. Architectural Design 9](#_Toc520355851)

[2.1 Software Components 9](#_Toc520355852)

[2.2 Software Modules 9](#_Toc520355853)

[2.2.1 Package Diagram 10](#_Toc520355854)

[2.2.2 Use Case relationships 11](#_Toc520355855)

[2.2.2.1 Scenario 1 – ordinary user 12](#_Toc520355856)

[2.2.2.2 Scenario 2 – Maintenance User 12](#_Toc520355857)

[2.2.2.3 Scenario 3 – Setup (Commissioning) User 13](#_Toc520355858)

[2.2.3 Dynamic behaviour 14](#_Toc520355859)

[2.2.3.1 Machine settings and non-volatile memory 14](#_Toc520355860)

[3. Software Detail Design and Hierarchy – HMI and COMMS 16](#_Toc520355861)

[3.1 Communications module 16](#_Toc520355862)

[3.1.1 Messages from the main board 16](#_Toc520355863)

[3.1.1.1 Common Header 16](#_Toc520355864)

[3.1.1.2 Payloads 17](#_Toc520355865)

[3.1.1.2.1 Payload for MCB\_MESSAGE\_MEASUREMENTS\_1 17](#_Toc520355866)

[3.1.1.2.2 Payload for MCB\_MESSAGE\_MEASUREMENTS\_2 19](#_Toc520355867)

[3.1.1.2.3 Payload for MCB\_MESSAGE\_EE\_VALUES\_1 20](#_Toc520355868)

[3.1.1.3 HMI incoming Message data type 22](#_Toc520355869)

[3.1.2 Messages to the main board 23](#_Toc520355870)

[3.1.2.1 Message types 23](#_Toc520355871)

[3.1.2.2 Message Body 23](#_Toc520355872)

[3.1.2.3 HMI outgoing Message data type 27](#_Toc520355873)

[3.2 VAR module 28](#_Toc520355874)

[3.2.1 VAR functionality 29](#_Toc520355875)

[3.2.2 NVHR Vars 32](#_Toc520355876)

[3.2.3 NVHR 16 bit EEPROM Memory Map 35](#_Toc520355877)

[3.3 HMI Software Modules 41](#_Toc520355878)

[3.3.1 Home screen splash 41](#_Toc520355879)

[3.3.2 Home screen logic 42](#_Toc520355880)

[3.3.3 HMI software components 43](#_Toc520355881)

[3.3.3.1 HMI User\_display 44](#_Toc520355882)

[3.3.3.2 HMI Settings (User\_settings) 44](#_Toc520355883)

[3.3.3.3 HMI User Maintenance 45](#_Toc520355884)

[3.3.3.4 HMI User Setup 45](#_Toc520355885)

[3.3.3.5 HMI Common 45](#_Toc520355886)

[3.3.3.6 HMI System\_info 57](#_Toc520355887)

[3.3.4 Summary 57](#_Toc520355888)

[3.3.5 Further Work Recommendations 57](#_Toc520355889)

[4. HMI Build Tools 58](#_Toc520355890)

[5. HMI Build Process 58](#_Toc520355891)

[6. Appendix A – NVHR HMI Screens Workflow 59](#_Toc520355892)

[7. Appendix B – NVHR HMI Screenshots 63](#_Toc520355893)

# ­Introduction

The Volution Group’s heat recovery and ventilation unit is designed for energy efficient ventilation of school classrooms and offices.

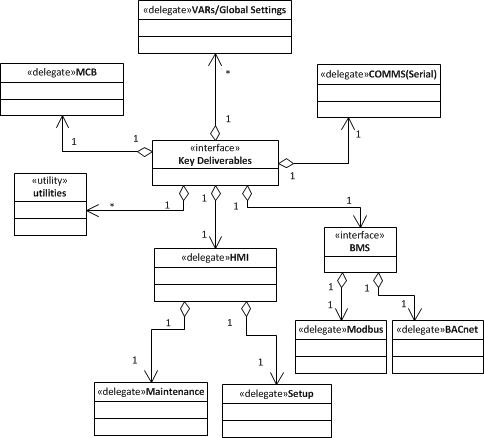
The system is designed for continuous ventilation of rooms within a property, providing significant energy recovery. The features for the current system are specified based on NVHR 1.5. The unit is controlled via touch screen HMI display, WIFI or via a third party building control system (BMS). The touch screen control unit provides the user interface for Setup (commissioning) and Maintenance purposes.

# Scope

# Identification

This document has been produced by Volution Group’s technical design team. The contents constitute the amalgamation of high level and low level software design processes and their components. This document describes only the software components of the NVHR HMI part of the overall system.

The design of the NVHR system is based on the key engineering deliverables as illustrated below:



**The Main NVHR System Key Deliverables**

# Exclusions

This document does not contain any software design information for the following components:

* The operating system structure and the NVHR integration
* The login and security procedures
* The Qt framework and Qt Simulator application
* The C/C++/C++11 optimisation features
* Application user guide
* System hardware structure

# SYSTEM Overview

# Overview

The system context diagram below depicts the communication relationship between components of the system. The machine’s mainboard is linked to the remote controls by serial data links.

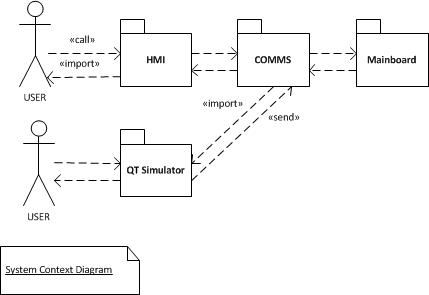
The remote controls are:

* Wired remote (HMI console)
* WiFi
* BMS

The communication between HMI and Mainboard is by 100-byte Mainboard to remote transmission packets and 84-byte remote to Mainboard replies. The low level comms are handled by MCB and VAR global modules described later in this document.

# System Context

The following diagram depicts the operational context of the NVHR system along with its interfaces with the external system.



# Architectural Design

The architectural design process is based on the following general requirements:

* The HMI application makes a distinction between the User, Maintenance and the Setup (Commissioning) user modes. A selective set of operations are available to different category of users:  
  + The Ordinary User, no lock code required
  + The Maintenance mode – Pin code required
  + The Setup (Commissioning) mode – Pin code required
* The embedded HMI is based on PIC32MX processor with memory constraints.
* There is a Qt Simulator that is based on PC and Qt graphics library, primarily used for debugging, rapid prototyping and feature functional testing.
* The HMI application communicates to the mainboard (with Processor PIC24FJ) via comms subsystem using a bespoke protocol. See later in this document for further details.
* User training is required to proficiently operate the system

# Software Components

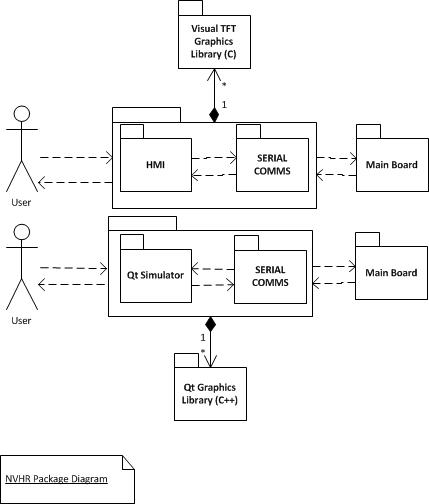
The sections 4 and 5 provide detailed list of the software modules (section 3.3) as well as tools required to build the HMI application.

# Software Modules

This section introduces the high level components and the system behaviour based on the software requirements together with HMI screen flows (appendix A).

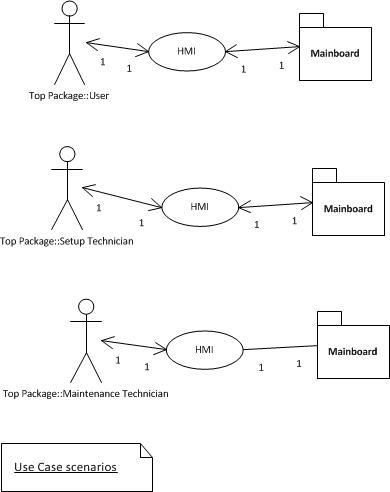
# Package Diagram

This section introduces the high level components of BB NVHR, depicting their structural relationship. This enables us to organise the model elements into groups with their global stereotype associations. This is effectively the business architecture diagram. The following diagram represents this structure. The aggregation, composition and dependancy notations represent the object referencing and the object containment strategies.



# Use Case relationships

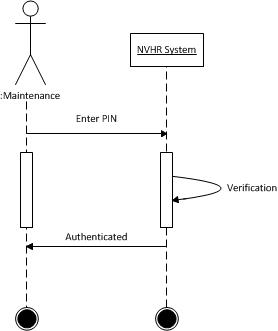
This section describes the behaviour of the system from the user’s perspective by using actions and reactions. They are derived from the user requirement specification; see NVHR HMI screens workflow in Appendix A. They depict the user interaction with the NVHR HMI sub-system and the environment. Each use case scenario corresponds to the specific type of system use. The system’s functionality is depicted when triggered in response to the simulation of an external actor, such as a system maintainer or an ordinary user.



# Scenario 1 – ordinary user

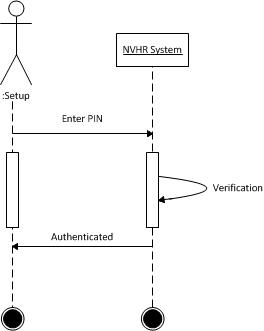
This scenario “extends” stereotype relationship between the ordinary user without any privileges and the sub-system. In case of system settings, only the “System Information” is available to the user. This includes, Date, Time, Unit Status, Room Temp Set Point, information about the system (e.g. NVHR Model, Project No).

# Scenario 2 – Maintenance User



Successful pin code entry permits the user to enter the maintenance menu. This allows system On/Off/Test Sequence performance, System Information, Sensor readings, Set Points Readings, Fan and Damper Status, User Screen options settings, Notices and providing Network information. See appendix B for a complete list of screenshots.

# Scenario 3 – Setup (Commissioning) User



Successful pin code entry enables the user to enter the commissioning menu. This allows system setup On/Off/Test Sequence performance, System Information, Sensor readings, Set Points Readings, Fan and Damper Status, User Screen options settings, Notices, providing Network information, setup Night Cool Period, Occupied Days & Hours and more. See appendix B for the screenshots and Appendix A for the HMI screenshot flow diagrams.

# Dynamic behaviour

The machine’s main board is physically linked to the wired HMI system by serial data link. The communication between the two systems is achieved by series of messages using a bespoke protocol. The messages from the main board are 100 bytes and those sent to the board from HMI are 84 bytes. The serial data is running at 9600 baud rate, 8 data bits, no parity, 1 stop bit, +3.3V TTL level UART signals. The break-down of each package is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Transmit Header | 34 bytes | Receive Header | 10 bytes |
| Transmit Payload | 64 bytes | Receive Payload | 72 bytes |
| Transmit Checksum | 2 bytes | Receive Checksum | 2 bytes |

The machine broadcasts its messages to all devices at one second interval, timed from the crystal controlled RTC chip. The header information from the board contains all global and common information that is the same for all transmitted messages. However, the payload changes for each message. Generally, a local copy of all non-volatile memory (EE for EEPROM) are stored are stored in HMI. When the user has finished adjusting a particular value, the HMI will send the value back to the main board in an ee write message to be copied in the EE memory. For details of messages and EE value, see ***ref 4*** (NVHR Advance Controls Protocol at a Glance). The detailed info about the data types responsible for communications is given later in this document. The message types are normally cycled around (from message type 0 to 8, 15).

# Machine settings and non-volatile memory

The machine’s settings are stored in non-volatile memory in the form of EE values. This memory is split into four areas, Working, Defaults, Setup and PCB calibration.

**Settings/Working:**  
 All settings are kept in this area. Any change in setting will be saved here, which will be used in machine’s control logic. The settings need a value when a unit is first used. These are the default values copied from the Defaults area of EE, when a unit is reset to factory defaults.

**Defaults:**Thedefaults are kept in this area. They are loaded in EE from a configuration file using external software. The software:  
 Erases this part of EE first  
 Copies the default values to Working/Settings area of EE  
 It is to these defaults that a ‘reset to factory defaults’ sets the machine.

**Set up:**  
When a machine is first switched on, the serial number, machine type and size are recorded into this section of EE by the external software. This is done only once.

**PCB Calibration:**  
On a brand new board at manufacturing test, test points are measured and adjusted from the diagnostic menu and once in the test limits, the calibration values are recorded into this section. This is done once. The file ***mcb\_ee.h*** contains the <*name> (2 bytes) and the <value> tuple.* Each value is written to EE with the message type MCB\_MESSAGE\_EE\_WRITE using**mcb\_write\_reply ()** function*.*

# Software Detail Design and Hierarchy – HMI and COMMS

This section will explore in detail, the components, attributes and the methods developed per module, bringing together the overall concept. The instantiation of these modules will provide the NVHR HMI and MCB system.

# Communications module

The communication between HMI and mainboard (MCB) is through this module. The reads and writes to the serial port are carried out through interfaces defined in ***mcb.***c using a bespoke protocol. See **ref 4** for more details.

# Messages from the main board

# Common Header



# Payloads

With existing HMI, the following message types are supported – defined in **mcb\_config.h**:

#define MCB\_MESSAGE\_MEASUREMENTS\_1 0u

#define MCB\_MESSAGE\_MEASUREMENTS\_2 1u

#define MCB\_MESSAGE\_EE\_VALUES\_1 2u

#define MCB\_MESSAGE\_EE\_VALUES\_2 3u

#define MCB\_MESSAGE\_EE\_VALUES\_3 4u

#define MCB\_MESSAGE\_EE\_VALUES\_4 5u

#define MCB\_MESSAGE\_EE\_VALUES\_5 6u

#define MCB\_MESSAGE\_EE\_VALUES\_6 7u

#define MCB\_MESSAGE\_EE\_VALUES\_7 8u

#define MCB\_MESSAGE\_EE\_ARRAYS\_1 15u

# Payload for MCB\_MESSAGE\_MEASUREMENTS\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 34 | 0 | 1 | Year | 2-digit year (+2000) |
| 35 | 1 | Month | 1=Jan .. 12=Dec |
| 36 | 1 | Day of Month | 1..31 |
| 37 | 1 | Day of Week | 1=Monday .. 7=Sunday |
| 38 | 1 | Daylight Savings | 0..1 |
| 39 | 2 | Hours Run (÷2) | MSB - units of 2 hours |
| 40 | LSB |
| 41 | 2 | Hours to next service | MSB |
| 42 | LSB |
| 43 | 1 | Commissioning Source | As per override source (x\_SERIAL\_ID) |
| 44 | 1 | OSCTUNE | signed |
| 45 | 2 | EE OSCTUNE | MSB |
| 46 | LSB |
| 47 | 1 | EE block in use | Block A (79) or block B (217) |
| 48 | 2 | EE addr last used | MSB |
| 49 | LSB |
| 50 | 2 | Maintenance PIN | MSB |
| 51 | LSB - 0..9999 |
| 52 | 2 | Commissioning PIN | MSB |
| 53 | LSB - 0..9999 |
| 54 | 2 | Room Temperature (P1) | 0-10V input P1 (T\_r = recirc temp) MSB (deg C x10) |
| 55 | LSB |
| 56 | 2 | Room CO2 level (P2) | 0-10V input P2 MSB (0..2500ppm) |
| 57 | LSB |
| 58 | 2 | Extract NTC thermistor (L) | MSB (deg C x10) |
| 59 | LSB |
| 60 | 2 | Mixed NTC thermistor (LC) | MSB (deg C x10) |
| 61 | LSB |
| 62 | 2 | External T\_e NTC thermistor (RC) | MSB (deg C x10) |
| 63 | LSB |
| 64 | 2 | Spare NTC thermistor (R) | MSB (deg C x10) |
| 65 | LSB |
| 66 | 1 | Fresh Damper direction | Stopped (0) / Opening (1) / Closing (2) |
| 67 | 1 | Fresh Damper drive duration | seconds |
| 68 | 1 | Fresh Damper drive stop | seconds |
| 69 | 1 | Exhaust Damper direction | Stopped (0) / Opening (1) / Closing (2) |
| 70 | 1 | Exhaust Damper drive duration | seconds |
| 71 | 1 | Exhaust Damper drive stop sec | seconds |
| 72 | 1 | Fresh fan (inner) state |  |
| 73 | 1 | Fresh fan endState speed |  |
| 74 | 1 | Fresh fan start (parent) | 0..100% |
| 75 | 1 | Fresh fan start (child) | 0..100% |
| 76 | 1 | Fresh fan end (parent) | 0..100% |
| 77 | 1 | Fresh fan state duration | seconds |
| 78 | 1 | Fresh fan state elapsed | seconds |
| 79 | 1 | Recirc fan (inner) state |  |
| 80 | 1 | Recirc fan endState speed |  |
| 81 | 1 | Recirc fan start (parent) | 0..100% |
| 82 | 1 | Recirc fan start (child) | 0..100% |
| 83 | 1 | Recirc fan end (parent) | 0..100% |
| 84 | 1 | Recirc fan state duration | seconds |
| 85 | 1 | Recirc fan state elapsed | seconds |
| 86 | 2 | V1 out (NVHR1.0 Fresh Child) | MSB |
| 87 | LSB |
| 88 | 2 | V2 out (NVHR1.0 Recirc Child) | MSB |
| 89 | LSB |
| 90 | 2 | L fan PWM (always Fresh Parent) | MSB |
| 91 | LSB |
| 92 | 2 | R fan PWM (NVHR1.0 Recirc Parent) | MSB |
| 93 | LSB |
| 94 | 2 | L fan RPM (always Fresh Parent) | MSB |
| 95 | LSB |
| 96 | 2 | R fan RPM (NVHR1.0 Recirc Parent) | MSB |
| 97 | LSB |

# Payload for MCB\_MESSAGE\_MEASUREMENTS\_2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 34 | 1 | 2 | Tr\_max\_today | MSB (deg C x10) |
| 35 | LSB |
| 36 | 2 | Te\_max\_today | MSB (deg C x10) |
| 37 | LSB |
| 38 | 2 | Assert error triggers | MSB |
| 39 | LSB |
| 40 | 1 | Extract T override source | see config.h (x\_SERIAL\_ID) |
| 41 | 1 | Intake T override source | see config.h (x\_SERIAL\_ID) |
| 42 | 1 | Mixed T override source | see config.h (x\_SERIAL\_ID) |
| 43 | 1 | Spare T override source | see config.h (x\_SERIAL\_ID) |
| 44 | 1 | Room T override source | see config.h (x\_SERIAL\_ID) |
| 45 | 1 | Room CO2 override sauce | see config.h (x\_SERIAL\_ID) |
| 46 |  |  |  |
| 47 |  |  |  |
| 48 |  |  |  |
| 49 |  |  |  |
| 50 |  |  |  |
| 51 |  |  |  |
| 52 |  |  |  |
| 53 |  |  |  |
| 54 |  |  |  |
| 55 |  |  |  |
| 56 |  |  |  |
| 57 |  |  |  |
| 58 | 8 | Diag status |  |
| 59 |  |
| 60 |  |
| 61 |  |
| 62 |  |
| 63 |  |
| 64 |  |
| 65 |  |
| 66 | 8 | BMS status | Model |
| 67 | Stack State |
| 68 | Traffic Status |
| 69 |  |
| 70 |  |
| 71 |  |
| 72 |  |
| 73 |  |
| 74 | 8 | WiFi status | Signal Strength (signed dBm) |
| 75 | remote\_status |
| 76 | status |
| 77 |  |
| 78 |  |
| 79 |  |
| 80 |  |
| 81 |  |
| 82 | 8 | Wired HMI status |  |
| 83 |  |
| 84 |  |
| 85 |  |
| 86 |  |
| 87 |  |
| 88 |  |
| 89 |  |
| 90 | 8 | RF HMI status |  |
| 91 |  |
| 92 |  |
| 93 |  |
| 94 |  |
| 95 |  |
| 96 |  |
| 97 |  |

# Payload for MCB\_MESSAGE\_EE\_VALUES\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 34 | 2 | 2 | EE 0 | Number of Values |
| 35 |
| 36 | 2 | EE 1 | Number of Arrays |
| 37 |
| 38 | 2 | EE 2 | Defaults Version |
| 39 |
| 40 | 2 | EE 3 | Model |
| 41 |
| 42 | 2 | EE 4 | Brand |
| 43 |
| 44 | 2 | EE 5 | Sensors required |
| 45 |
| 46 | 2 | EE 6 | Spare |
| 47 |
| 48 | 2 | EE 7 | Spare |
| 49 |
| 50 | 2 | EE 8 | Language |
| 51 |
| 52 | 2 | EE 9 | Country |
| 53 |
| 54 | 2 | EE 10 | Service interval |
| 55 |
| 56 | 2 | EE 11 | Mainboard setup flags |
| 57 |
| 58 | 2 | EE 12 | HMI setup flags |
| 59 |
| 60 | 2 | EE 13 | CO2 Traffic Light Low threshold |
| 61 |
| 62 | 2 | EE 14 | CO2 Traffic Light High threshold |
| 63 |
| 64 | 2 | EE 15 | CO2 Traffic Light Colour Selection |
| 65 |
| 66 | 2 | EE 16 | PID p coefficient |
| 67 |
| 68 | 2 | EE 17 | PID i coefficient |
| 69 |
| 70 | 2 | EE 18 | PID d coefficient |
| 71 |
| 72 | 2 | EE 19 | PID dt |
| 73 |
| 74 | 2 | EE 20 | 0 |
| 75 |
| 76 | 2 | EE 21 | 0 |
| 77 |
| 78 | 2 | EE 22 | 0 |
| 79 |
| 80 | 2 | EE 23 | 0 |
| 81 |
| 82 | 2 | EE 24 | 0 |
| 83 |
| 84 | 2 | EE 25 | 0 |
| 85 |
| 86 | 2 | EE 26 | 0 |
| 87 |
| 88 | 2 | EE 27 | 0 |
| 89 |
| 90 | 2 | EE 28 | 0 |
| 91 |
| 92 | 2 | EE 29 | 0 |
| 93 |
| 94 | 2 | EE 30 | BACnet / MODbus Baud Rate + Parity |
| 95 |
| 96 | 2 | EE 31 | BACnet / MODbus Address |
| 97 |

For the payloads for other messages, see **ref 4**.

# HMI incoming Message data type

The data type to handle incoming messages from the main board is defined in **mcb.c** and depicted below:

**// NVHR message**

// --------------------------------------------------------------------------

typedef struct CPL\_ATTRIB\_PACKED MCB\_msg\_t

{

// standard message header (FROM mainboard)

uint8 msg\_num; // 0 to 255 rolling

uint8 msg\_type;

uint8 fw\_ver;

uint8 serial[4]; // 32bit big endian

uint8 op\_mode;

uint8 override\_src;

uint8 fresh\_damper\_status;

uint8 window\_advice\_state;

uint8 season;

uint8 graph\_state;

uint8 sub\_mode;

uint8 fresh\_fan\_state;

uint8 recirc\_fan\_state;

uint8 current\_override;

uint8 exhaust\_damper\_status;

uint8 bms\_stop\_source;

uint8 diag\_code;

uint8 warn\_code;

uint8 remote\_override\_time\_req[2];

uint8 remote\_override\_time\_rem[2];

uint8 remote\_override\_source;

uint8 boards\_fitted;

uint8 ls\_status;

uint8 switch\_status;

uint8 port\_output\_bits;

uint8 usb\_status;

uint8 rtc\_hours;

uint8 rtc\_minutes;

uint8 rtc\_second;

// **payload** depends upon msg\_type;

uint8 payload[64];

// 16bit additive checksum

uint8 chksum\_msb;

uint8 chksum\_lsb;

} **MCB\_msg\_t**;

All the incoming messages are handled by   
 **bool\_t mcb\_handle\_msg(MCB\_msg\_t const\* msg)**  defined in mcb.c

# Messages to the main board

# Message types

Currently the following message types are supported – defined in ***mcb\_config.h***

// message type - incoming

#define MCB\_MESSAGE\_STANDARD\_REPLY 100 // standard reply header values no payload

#define MCB\_MESSAGE\_CLOCKSET 101 // clockset

#define MCB\_MESSAGE\_COMMISSIONING 102 // air flow that the fans run at in payload

#define MCB\_MESSAGE\_CONTROLLED\_MODE 103 // takes over control

#define MCB\_MESSAGE\_EE\_WRITE 104 // <name> value(s)/array(s) in payload

#define MCB\_MESSAGE\_SENSOR\_OVERRIDE 105 // override a sensor input

#define MCB\_MESSAGE\_LOAD\_PRESET\_FLOWS 106

#define MCB\_MESSAGE\_RESTORE\_DEFAULTS 107 //

#define MCB\_MESSAGE\_USB 108 // read write on payload

#define MCB\_MESSAGE\_NUMBER\_OF\_TYPES 109 // reduced normal or extended

#define MCB\_MESSAGE\_BMS\_STOP 111 // NOT used by HMI

#define MCB\_MESSAGE\_BMS\_START 112 // NOT used by HMI

#define MCB\_MESSAGE\_RESET\_SERVICE 114

#define MCB\_MESSAGE\_MANUAL\_OVERRIDE 115

#define MCB\_MESSAGE\_ACK 255

# Message Body

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE STANDARD REPLY (100)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Not used | 0 |
| 4 | 1 | Not used | 0 |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 8 | Peripheral Status | Meaning varies depending on peripheral |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE CLOCKSET (101)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Not used | 0 |
| 4 | 1 | Not used | 0 |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 1 | Minute | 0..59 | Set bit 7 to set clock |
| 11 | 1 | Hour | 0..23 |
| 12 | 1 | Day of Month | 1..31 |
| 13 | 1 | Month | 1..12 |
| 14 | 1 | Day of Week | Not used (set to 1) |
| 15 | 1 | Year | 15..99 |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE\_COMMISSIONING (102)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Not used | 0 |
| 4 | 1 | Verification code | MESSAGE\_COMMISSIONING\_NUMBER (2) |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 2 | Parent Fresh AF | MSB (0..1000 = 0.0-100.0%) |
| 11 | LSB |
| 12 | 2 | Parent Recirc AF | MSB (0..1000 = 0.0-100.0%) |
| 13 | LSB |
| 14 | 2 | Child Fresh AF | MSB (0..1000 = 0.00-10.00V) |
| 15 | LSB |
| 16 | 2 | Child Recirc AF | MSB (0..1000 = 0.00-10.00V) |
| 17 | LSB |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE CONTROLLED MODE (103)** |
| 2 | 1 | Not used |  |
| 3 | 1 | Verification code | MESSAGE\_CONTROLLED\_MODE\_CODE (238) |
| 4 | 1 | Verification number | MESSAGE\_CONTROLLED\_MODE\_NUMBER (5) |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 1 | Fresh Damper State | TBC |
| 11 | 1 | Exhaust Damper State | TBC |
| 12 | 1 | Fresh fan-speed | 0=stopped, 1=starting, 2=min, 3=slow, 4=fast, 5=ultra |
| 13 | 1 | Recirc fan-speed |
| 14 | 1 | Window Advice | TBC |
| 15 | 1 | Season | TBC |
| 16 | 1 | Sub-mode | TBC |
| 17 | 2 | Voltage controlled Damper (0.0-100.0%) | MSB |
| 18 | LSB |
| 19 | 1 | Output freeze bits | bit 1=FanL, 2=FanR, 3=V1, 4=V2, 5=Relays, 6=H-Bridge (bits 7+8 spare) |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE EE WRITE (104)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Num of Arrays | 1..4 for arrays, 0 for values |
| 4 | 1 | Num of Values | 1..18 for values, 0 for arrays |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | *See command arrays & values sheet for details* | | |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE OVERRIDE\_SENSOR (105)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Num of Arrays | 150 (magic value) |
| 4 | 1 | Num of Values | 0 |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 1 | Sensor ID | TBC |
| 11 | 2 | Override Value | MSB |
| 12 | LSB |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |
|  |  |  |  |  |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE LOAD\_PRESET\_FLOWS (106)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Num of Arrays | MESSAGE\_RESTORE\_DEFAULTS\_CODE (73) |
| 4 | 1 | Num of Values | 256 - MESSAGE\_RESTORE\_DEFAULTS\_CODE |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 1 | Preset number | 0=rectangular, 1=20pa spigot |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE RESTORE DEFAULTS (107)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Num of Arrays | MESSAGE\_RESTORE\_DEFAULTS\_CODE (73) |
| 4 | 1 | Num of Values | 256 - MESSAGE\_RESTORE\_DEFAULTS\_CODE |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | *None* | | |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |
|  |  |  |  |  |
| Byte | Message Type | Length | Field | Description |
| 0 | Header | 1 | Message Number | Cycles 0..255 |
| 1 | 1 | Message Type | **MESSAGE USB (108)** |
| 2 | 1 | Not used | 0 |
| 3 | 1 | Not used | 0 |
| 4 | 1 | Not used | 0 |
| 5 | 1 | Message types | 0=normal, 1=extended, other=reduced |
| 6 | 1 | Spare |  |
| 7 | 1 | Spare |  |
| 8 | 1 | F/W Ver High | 0..255 (0=pre-release, 1=first major release) |
| 9 | 1 | F/W Ver Low | 0..255 |
| 10 | Payload | 1 | USB Command | TBC |
| 82 | Footer | 2 | Checksum | MSB |
| 83 | LSB |

Further details and messages are described in **ref 4** document.

# HMI outgoing Message data type

The following data type defines the outgoing messages, specified as:

// --------------------------------------------------------------------------

typedef struct CPL\_ATTRIB\_PACKED MCB\_reply\_t

{

// standard message header (TO mainboard)

uint8 msg\_num; // cycles round 1 to 255

uint8 msg\_type;

uint8 btn\_value;

uint8 num\_arrays;

uint8 num\_values;

uint8 message\_types; // 0 to 9 (tbc) or 46 if not yet adjusted

uint8 remote\_onoff;

uint8 spare;

uint8 fw\_ver\_major;

uint8 fw\_ver\_minor;

// **payload** depends upon msg\_type;

uint8 payload[72];

// 16bit additive checksum

uint8 chksum\_msb;

uint8 chksum\_lsb;

} **MCB\_reply\_t;**

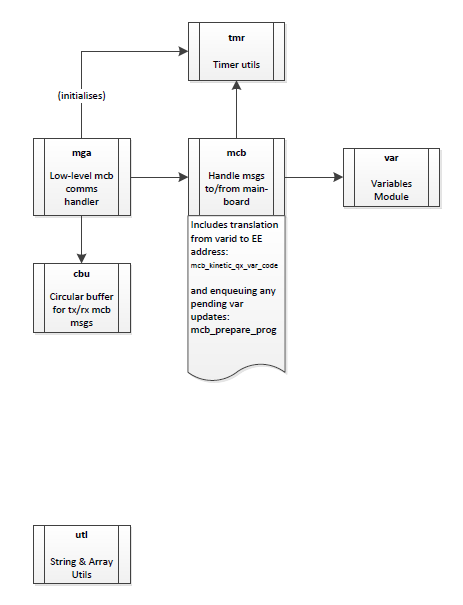
All the outgoing messages are handled by the function:  
 **bool\_t mcb\_prepare\_prog(MCB\_reply\_t \* reply)** defined in ***mcb.c*** file.

# VAR module

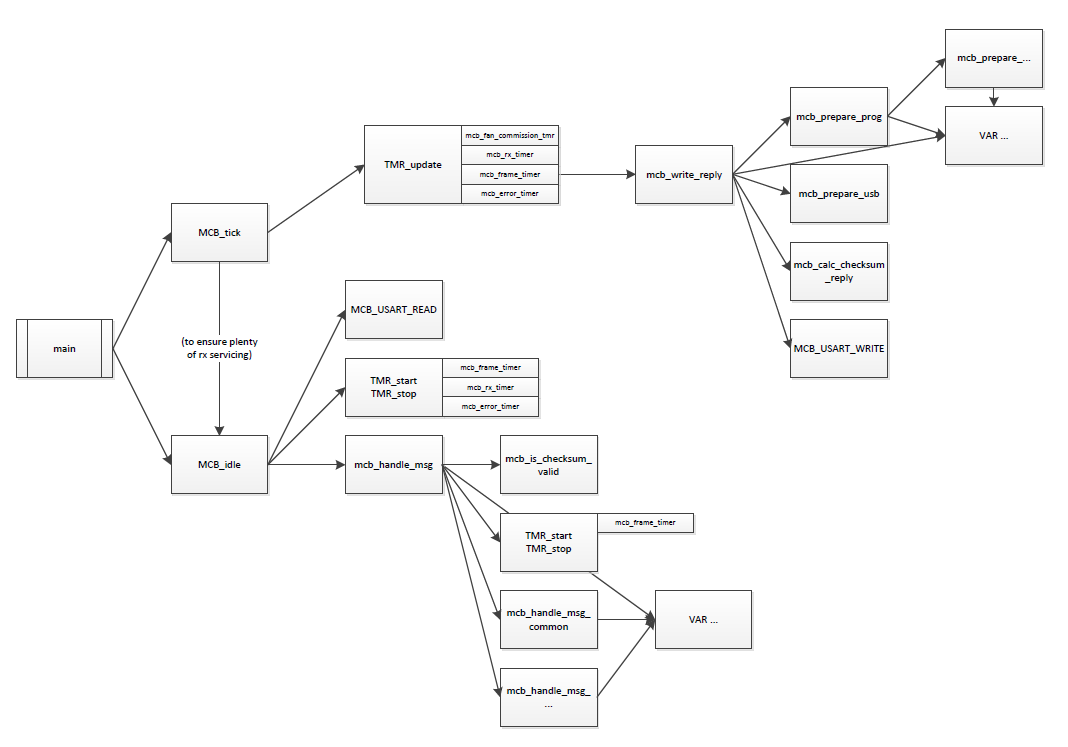
HMI and the Mainboard convey data through global variable set (VAR). They are classified in file ***var.h*** as:

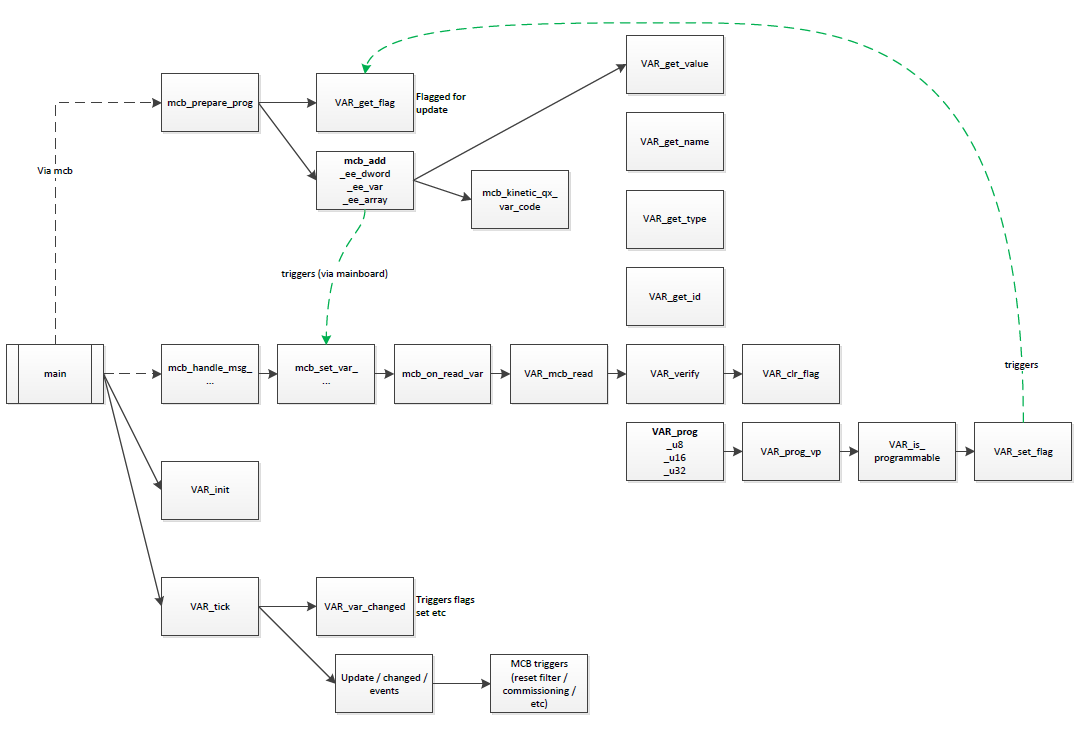
VAR\_TABLE\_STAT.  
VAR\_TABLE\_SINK,  
VAR\_TABLE\_INTERN,  
VAR\_TABLE\_EE

To read from these variables, ***var\_data\_all->*** is de-referenced.  
To write to these variables, VAR***\_prog\_u8(VAR\_ID\_var, value), VAR\_prog\_u16(VAR\_ID\_var, value), VAR\_prog\_u32(VAR\_ID\_var, value)*** are utilised.



# VAR functionality





# NVHR Vars







# NVHR 16 bit EEPROM Memory Map

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EE Hex | EE Dec | Name | Description | Write | Var Name | Msg |
| *Working Block – 16 bit values* | | | | | | |











|  |
| --- |
| *Working Block – 16 byte Arrays* |



|  |
| --- |
| *Machine Setup – 16 bit Values* |





|  |
| --- |
| *Machine Setup – 16 byte Arrays* |



|  |
| --- |
| *PCB Calibration – 16 bit Values* |

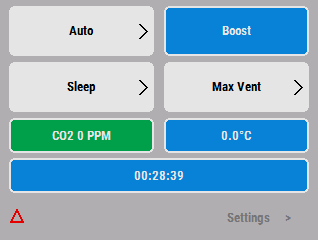


# HMI Software Modules

The main entry to the NVHR HMI system is through the home screen. Any changes to the system may be carried out through the *settings* option.

# Home screen splash

The following screen shall contain the specified controls derived from the workflow, allowing the user navigation and access to system maintenance, setup (commissioning) and the system information.

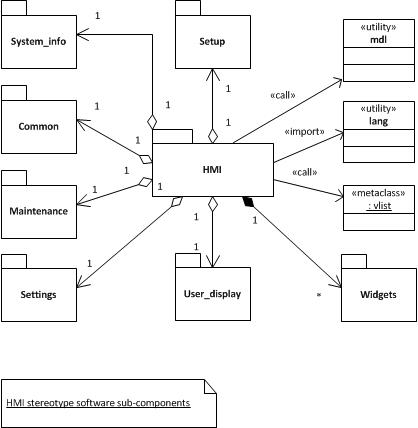


# Home screen logic

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **VentManualOp** | **Override Source** | **Info Display** | **Auto Vent** | **More Vent** | **Disable Vent** | **Max Vent** |
| AUTO | NONE  (no others should ever show) | Window Advice | Highlighted  Not clickable | Not highlighted  Clickable | Not highlighted  Clickable | Not highlighted  Clickable |
| MANUAL\_MORE\_VENT | BUTTON\_P1\_to\_6 (HMI) | Countdown | Not highlighted  Clickable | Highlighted  Not clickable | Not highlighted  Clickable | Not highlighted  Clickable |
| MANUAL\_MORE\_VENT | (any other) | Override source | Not highlighted  Not clickable | Highlighted  Not clickable | Not highlighted  Clickable | Not highlighted  Clickable |
| MANUAL\_DISABLE\_VENT | BUTTON\_P1\_to\_6 (HMI) | Countdown | Not highlighted  Clickable | Not highlighted  Clickable | Highlighted  Not clickable | Not highlighted  Clickable |
| MANUAL\_DISABLE\_VENT | STOP\_SW1  STOP\_DIAGNOSTIC  STOP\_FAULT\_CODE  STOP\_SETUP\_OFF  STOP\_BMS | **OFF[[1]](#footnote-1)** | | | | |
| MANUAL\_DISABLE\_VENT | (any other) | Override source | Not highlighted  Not clickable | Not highlighted  Clickable | Highlighted  Not clickable | Not highlighted  Clickable |
| MANUAL\_MAX\_VENT | BUTTON\_P1\_to\_6 (HMI) | Countdown | Not highlighted  Clickable | Not highlighted  Clickable | Not highlighted  Clickable | Highlighted  Not clickable |
| MANUAL\_MAX\_VENT | (any other) | Override source | Not highlighted  Not clickable | Not highlighted  Clickable | Not highlighted  Clickable | Highlighted  Not clickable |
| ACTIVE\_NIGHT\_COOL  PASSIVE\_NIGHT\_COOL | NONE  (no others should ever show) | Night Cooling | Highlighted  Not clickable[[2]](#footnote-2) | Not highlighted  Clickable | Not highlighted  Clickable | Not highlighted  Clickable |
| REMOTE\_CONTROLLED  COMMISSIONING  DIAGNOSTICS  (any other / undefined) | (any) | VentManualOp Name | Not highlighted  Not clickable | Not highlighted  Not clickable | Not highlighted  Not clickable | Not highlighted  Not clickable |

# HMI software components

The following diagram depicts the stereotyped software components of the HMI subsystem.



Each component package will be described in further details in the following sections. The function prototypes and data structures are specified in file ***scr.h***

# HMI User\_display

|  |  |  |
| --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** |
| ***void***  ***NVHR\_SCR\_screen\_user\_display(void)*** | Sets display to ON or OFF |  |
| ***void***  ***NVHR\_SCR\_screen\_user\_display\_off(void)*** | Sets display to OFF state. Also used in Home Screen Logic and tick functions | typedef struct NVHR\_SCR\_screen\_user\_display\_off\_t  {  TButton\_Round nvhr\_message;  TButton nvhr\_settings;  } NVHR\_SCR\_screen\_user\_display\_off\_t; |
| ***void***  ***NVHR\_SCR\_screen\_user\_display\_on(void)*** | Sets display to ON state. Also used in Home Screen Logic and tick functions | typedef struct NVHR\_SCR\_screen\_user\_display\_t  {  TButton\_Round nvhr\_auto;  TButton\_Round nvhr\_boost;  TButton\_Round nvhr\_sleep;  TButton\_Round nvhr\_maxvent;  TButton\_Round nvhr\_co2;  TButton\_Round nvhr\_temperature;  TButton\_Round nvhr\_info\_display;  TButton nvhr\_settings;  SCR\_custom\_t nvhr\_auto\_right\_arrow;  SCR\_custom\_t nvhr\_boost\_right\_arrow;  SCR\_custom\_t nvhr\_maxvent\_right\_arrow;  SCR\_custom\_t nvhr\_sleep\_right\_arrow;  SCR\_custom\_t nvhr\_triangle;  } NVHR\_SCR\_screen\_user\_display\_t; |
| ***void***  ***NVHR\_SCR\_boost\_settings(uint32 index)*** | Sets boost times. Index determines if for maxvent, boost or sleep | typedef struct NVHR\_SCR\_boost\_settings\_t  {  TCircleButton nvhr\_help;  TButton nvhr\_title;  TButton\_Round nvhr\_time1;  TButton\_Round nvhr\_time2;  TButton\_Round nvhr\_time3;  TButton\_Round nvhr\_time4;  SCR\_custom\_t nvhr\_back;  SCR\_custom\_t nvhr\_time1\_right\_arrow;  SCR\_custom\_t nvhr\_time2\_right\_arrow;  SCR\_custom\_t nvhr\_time3\_right\_arrow;  SCR\_custom\_t nvhr\_time4\_right\_arrow;  } NVHR\_SCR\_boost\_settings\_t; |
| ***void***  ***NVHR\_SCR\_boost\_help(void)*** | Boost help screen | typedef struct NVHR\_SCR\_help\_struct\_t  {  TButton nvhr\_title;  TButton\_Round nvhr\_help\_bg;  SCR\_custom\_t nvhr\_back;  TLabel nvhr\_help\_lbl[SCR\_VLIST\_MAX\_ROWS];  char nvhr\_help\_txt[SCR\_VLIST\_MAX\_ROWS][42];  } NVHR\_SCR\_help\_t; |

# HMI Settings (User\_settings)

|  |  |  |
| --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** |
| ***void NVHR\_SCR\_screen\_user\_settings(void)*** | The user settings sub-menu | typedef struct NVHR\_SCR\_screen\_user\_settings\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_screen\_user\_settings\_t; |

# HMI User Maintenance

|  |  |  |
| --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** |
| ***void***  ***NVHR\_SCR\_user\_maintenance(void)*** | The user Maintenance sub-menu | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_user\_maintenance\_t; |
|  |  |  |
|  |  |  |

# HMI User Setup

|  |  |  |
| --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** |
| ***void***  ***NVHR\_SCR\_user\_setup(void)*** | The user Setup sub-menu | typedef struct NVHR\_SCR\_user\_setup\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_user\_setup\_t; |

# HMI Common

|  |  |  |  |
| --- | --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** | |
| ***void***  ***NVHR\_SCR\_screen\_lock\_code(void)*** | Used for locking screen – not utilised | typedef struct SCR\_screen\_lock\_code\_struct  {  SCR\_screen\_adjuster\_quad\_t a;  TButton done;  } SCR\_screen\_lock\_code\_t; | |
| ***void***  ***NVHR\_SCR\_screen\_unlock(uint8 index);*** | Used for unlocking screen – quad adjuster |  | |
| ***void***  ***NVHR\_SCR\_system\_on\_off(void)*** | Toggle screen | typedef struct SCR\_screen\_chklist\_struct  {  TButton title;  TButton action;  SCR\_custom\_t action\_arrow;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_chklist\_t chklist;  uint8 indices[SCR\_CHKLIST\_MAX\_ROWS];  char strings[SCR\_CHKLIST\_MAX\_ROWS][32];  } NVHR\_SCR\_screen\_system\_on\_off\_t; | |
| ***void***  ***NVHR\_SCR\_system\_yes\_no(void);*** | Toggle screen | typedef struct SCR\_screen\_chklist\_struct  {  TButton title;  TButton action;  SCR\_custom\_t action\_arrow;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_chklist\_t chklist;  uint8 indices[SCR\_CHKLIST\_MAX\_ROWS];  char strings[SCR\_CHKLIST\_MAX\_ROWS][32];  } NVHR\_SCR\_screen\_system\_yes\_no\_t; | |
| ***void*** ***NVHR\_SCR\_screen\_display\_settings(void);*** | Single adjuster for display settings | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } SCR\_screen\_display\_settings\_t; | |
| ***void NVHR\_SCR\_screen\_address\_settings(void)*** | Single adjuster for address settings | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_screen\_address\_settings\_t | |
| ***void***  ***NVHR\_SCR\_test\_sequence(void)*** | Test sequence screens | typedef struct NVHR\_SCR\_test\_sequence\_t  {  TButton title;  TButton\_Round message1\_container;  TButton\_Round confirm;  TButton\_Round cancel;  SCR\_custom\_t back;  TLabel test\_sequence\_lbl[SCR\_VLIST\_MAX\_ROWS];  char test\_sequence\_txt[SCR\_VLIST\_MAX\_ROWS][40];  } NVHR\_SCR\_test\_sequence\_t; | |
| ***void***  ***NVHR\_SCR\_option\_unavailable(void)*** | Screen for unavailable options | typedef struct NVHR\_SCR\_not\_done\_t  {  TButton title;  TButton\_Round message\_container;  TButton\_Round confirm;  TLabel not\_done\_lbl[SCR\_VLIST\_MAX\_ROWS];  char not\_done\_txt[SCR\_VLIST\_MAX\_ROWS][40];  } NVHR\_SCR\_not\_done\_t; | |
| ***void***  ***NVHR\_SCR\_display\_settings(void)*** | Display brightness settings | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } SCR\_screen\_display\_settings\_t; | |
| ***void NVHR\_SCR\_room\_temp\_setpoint(void);*** | Adjuster for room temperature setpoint | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_temp\_setpoint\_t; |
| ***void NVHR\_SCR\_room\_temp\_setpoint\_help(void);*** | Help screen | typedef struct NVHR\_SCR\_help\_struct\_t  {  TButton nvhr\_title;  TButton\_Round nvhr\_help\_bg;  SCR\_custom\_t nvhr\_back;  TLabel nvhr\_help\_lbl[SCR\_VLIST\_MAX\_ROWS];  char nvhr\_help\_txt[SCR\_VLIST\_MAX\_ROWS][42];  } NVHR\_SCR\_room\_temp\_setpoint\_help\_t; |
| ***void***  ***NVHR\_SCR\_winsum\_xover(void)*** | Winter to Summer cross over | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_temp\_setpoint\_t; |
| ***void***  ***NVHR\_SCR\_winsum\_xover\_help(void)*** | Help screen for Winter Summer crossover | typedef struct NVHR\_SCR\_help\_struct\_t  {  TButton nvhr\_title;  TButton\_Round nvhr\_help\_bg;  SCR\_custom\_t nvhr\_back;  TLabel nvhr\_help\_lbl[SCR\_VLIST\_MAX\_ROWS];  char nvhr\_help\_txt[SCR\_VLIST\_MAX\_ROWS][42];  } NVHR\_SCR\_room\_temp\_setpoint\_help\_t; |
| ***void***  ***NVHR\_SCR\_common\_sysinfo(void)*** | System info for Maintenance and Setup | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_sysinfo\_t; |
| ***void***  ***NVHR\_SCR\_common\_sysinfo\_readme(void)*** | Help screen for system info | typedef struct NVHR\_SCR\_system\_info\_readme\_t  {  TButton nvhr\_title;  TButton\_Round nvhr\_help\_bg;  SCR\_custom\_t nvhr\_back;  TLabel nvhr\_help\_lbl[SCR\_VLIST\_MAX\_ROWS];  char nvhr\_help\_txt[SCR\_VLIST\_MAX\_ROWS][32];  } NVHR\_SCR\_system\_info\_readme\_t; |
| ***void NVHR\_SCR\_screen\_common\_sysinfo\_about(void)*** | Info about sysinfo | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_sysinfo\_about\_t; |
| ***void***  ***NVHR\_SCR\_common\_setpoints(void)*** | Setpoints screen for Maintenance and Setup | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_setpoints\_t; |
| ***Void***  ***NVHR\_SCR\_common\_sensor\_readings(void)*** | Sensor readings | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_sensor\_readings\_t; |
| ***void***  ***NVHR\_SCR\_common\_fandamp\_status(void)*** | Fan and Damper status for Maintenance and Setup | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_fandamp\_t; |
| ***void NVHR\_SCR\_common\_timedate\_settings(void)*** | Time and date settings | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_datetime\_t; |
| ***void NVHR\_SCR\_common\_userscr\_settings(void)*** | User settings sub-menu | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_common\_userscr\_t; |
| ***void***  ***NVHR\_SCR\_common\_notices(void)*** | Notices screen | Option unavailable |
| ***void***  ***NVHR\_SCR\_screen\_date(void)*** | Date setup widget | typedef struct NVHR\_SCR\_screen\_date\_struct  {  TButton title;  SCR\_custom\_t back;  TButton\_Round inc\_year;  TButton\_Round inc\_month;  TButton\_Round inc\_day;  TButton\_Round dec\_year;  TButton\_Round dec\_month;  TButton\_Round dec\_day;  TLine line1;  TLine line2;  TButton year\_title;  TButton month\_title;  TButton day\_title;  TButton year\_value;  TButton month\_value;  TButton day\_value;  char str\_year[8];  char str\_month[8];  char str\_day[8];  uint8 date\_year;  uint8 date\_month;  uint8 date\_day;  } NVHR\_SCR\_screen\_date\_t; |
| ***void***  ***NVHR\_SCR\_screen\_time(void)*** | Time setup widget | typedef struct NVHR\_SCR\_screen\_time\_struct  {  TButton title;  SCR\_custom\_t back;  TButton\_Round inc\_hour;  TButton\_Round inc\_mins;  TButton\_Round dec\_hour;  TButton\_Round dec\_mins;  TLine line1;  TLine line2;  TButton hour\_title;  TButton mins\_title;  TButton hour\_value;  TButton mins\_value;  TLabel separator;  char str\_hour[8];  char str\_mins[8];  uint8 time\_hour;  uint8 time\_mins;  } NVHR\_SCR\_screen\_time\_t; |
| ***void***  ***NVHR\_SCR\_occupied\_days\_hours(void)*** | Widget to specify occupied days and hours | typedef struct NVHR\_SCR\_common\_date\_t  {  SCR\_screen\_vlist\_t vscr;  SCR\_weekdays\_t weekdays;  char time\_str[16];  } NVHR\_SCR\_common\_date\_t; |
| ***void***  ***NVHR\_SCR\_nightcool\_days\_hours(void)*** | Widget to specify nightcool days and hours | typedef struct NVHR\_SCR\_common\_date\_t  {  SCR\_screen\_vlist\_t vscr;  SCR\_weekdays\_t weekdays;  char time\_str[16];  } NVHR\_SCR\_common\_date\_t; |
| ***void***  ***NVHR\_SCR\_nightcool\_period(void)*** | Dual adjuster to specify nightcool period | typedef struct SCR\_screen\_adjuster\_dual\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[2];  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t values[2];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_dual\_struct\* a, uint8 index);  } NVHR\_SCR\_nightcool\_period\_t; |
| ***void***  ***NVHR\_SCR\_common\_bacnet(void)*** | BACnet subsystem menu | typedef struct NVHR\_SCR\_common\_bacnet\_modbus\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_common\_bacnet\_modbus\_t; |
| ***void***  ***NVHR\_SCR\_common\_network\_baud(void)*** | Setup baud rate | typedef struct SCR\_screen\_chklist\_struct  {  TButton title;  TButton action;  SCR\_custom\_t action\_arrow;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_chklist\_t chklist;  uint8 indices[SCR\_CHKLIST\_MAX\_ROWS];  char strings[SCR\_CHKLIST\_MAX\_ROWS][32];  } NVHR\_SCR\_common\_network\_baud\_t; |
| ***void***  ***NVHR\_SCR\_common\_modbus(void)*** | Modbus subsystem menu | typedef struct NVHR\_SCR\_common\_bacnet\_modbus\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_common\_bacnet\_modbus\_t; |
| ***void NVHR\_SCR\_screen\_common\_modbus\_parity(void)*** | Checklist for parity setup | typedef struct SCR\_screen\_chklist\_struct  {  TButton title;  TButton action;  SCR\_custom\_t action\_arrow;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_chklist\_t chklist;  uint8 indices[SCR\_CHKLIST\_MAX\_ROWS];  char strings[SCR\_CHKLIST\_MAX\_ROWS][32];  } NVHR\_SCR\_common\_network\_parity\_t; |
| ***void***  ***NVHR\_SCR\_common\_fans(void)*** | Fan setup menu | typedef struct NVHR\_SCR\_common\_fandamp\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_common\_fan\_damp\_t; |
| ***void***  ***NVHR\_SCR\_common\_dampers(void)*** | Damper setup menu | typedef struct NVHR\_SCR\_common\_fandamp\_struct  {  TButton title;  SCR\_custom\_t back;  SCR\_vlist\_t vlist;  char strings[SCR\_VLIST\_MAX\_ROWS][16];  } NVHR\_SCR\_common\_fan\_damp\_t; |
| ***void***  ***NVHR\_SCR\_screen\_service\_input(char const \*title, VAR\_phone\_num\_t const \*ph)*** | Service input widget | typedef struct SCR\_screen\_service\_phone\_t  {  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_custom\_t phone;  TButton\_Round btn\_left;  TButton\_Round btn\_right;  TButton\_Round btn\_digit[12];  char str\_digit[12][2];  SCR\_event\_handler\_t evt\_handler;  uint8 active\_digit;  VAR\_phone\_num\_t phone\_num;  } SCR\_screen\_service\_phone\_t; |
| ***void***  ***SCR\_screen\_invalid\_defaults(void)*** | Error message dialog | typedef struct SCR\_screen\_error\_t  {  TButton title;  SCR\_custom\_t back;  TButton info1;  TButton info2;  } SCR\_screen\_error\_t; |
| ***void***  ***NVHR\_SCR\_screen\_invalid\_model(void)*** | Error message dialog | typedef struct SCR\_screen\_error\_t  {  TButton title;  SCR\_custom\_t back;  TButton info1;  TButton info2;  } SCR\_screen\_error\_t; |
| ***void***  ***NVHR\_SCR\_vent\_on\_CO2(void)*** | Dual adjuster for Vent on CO2 | typedef struct SCR\_screen\_adjuster\_dual\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[2];  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t values[2];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_dual\_struct\* a, uint8 index);  } NVHR\_SCR\_screen\_vent\_on\_CO2\_t; |
| ***void***  ***NVHR\_SCR\_room\_CO2\_indicator(void)*** | Dual adjuster for CO2 low and CO2 high | N typedef struct SCR\_screen\_adjuster\_dual\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[2];  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t values[2];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_dual\_struct\* a, uint8 index);  } NVHR\_SCR\_screen\_rm\_co2\_indicator\_t; |
| ***void***  ***NVHR\_SCR\_winter\_offset(void)*** | Single adjuster for winter offset temperature | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_winter\_offset\_setpoint\_t; |
| ***void***  ***NVHR\_SCR\_nightcool\_trigger(void)*** | Dual adjuster for Day max setpoint and room temp point | typedef struct SCR\_screen\_adjuster\_dual\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[2];  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t values[2];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_dual\_struct\* a, uint8 index);  } NVHR\_SCR\_screen\_nightcool\_trigger\_t; |
| ***void***  ***NVHR\_SCR\_mixed\_delivery(void)*** | List of mixed delivery items – temp set point and PID control constants | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_mixed\_air\_delivery\_t; |
| ***void***  ***NVHR\_SCR\_air\_delivery\_temp(void)*** | Single adjuster for air delivery temperature set point | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_air\_delivery\_temp\_t; |
| ***void***  ***NVHR\_SCR\_air\_delivery\_pid(void)*** | Triple adjuster for Kp, Ki and Kd coefficients (+/- 373) | typedef struct SCR\_screen\_adjuster\_triple\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[3];  TLine line1;  TLine line2;  // Data  uint16 tag;  uint16 tag2;  SCR\_adjuster\_value\_t values[3];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_triple\_struct\* a, uint8 index);  } NVHR\_SCR\_air\_delivery\_pid\_t; |
| ***void***  ***NVHR\_SCR\_fan\_speeds(void)*** | The menu lists for Fan speed setup | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_fan\_speed\_t; |
| ***Static void***  ***nvhr\_scr\_fan\_single\_speed(uint8\_t index, const char \* title)*** | Dual adjusters for parent fresh (PF) setting up (following index):  STARTING MINIMUM  WINTER\_SLOW  WINTER\_FAST  SUMMER\_SLOW  SUMMER\_FAST  ULTRA | typedef struct SCR\_screen\_adjuster\_single\_struct  {  // Widgets  TButton title;  TCircleButton nvhr\_help;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster;  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t value;  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_single\_struct\* a);  } NVHR\_SCR\_screen\_fan\_single\_speed\_t; |
| ***static void***  ***nvhr\_scr\_fan\_dual\_speed(uint8 index, const char \* title)*** | Dual adjusters for parent-child fresh (PF – CF) setting up (following index):  STARTING MINIMUM  WINTER\_SLOW  WINTER\_FAST  SUMMER\_SLOW  SUMMER\_FAST  ULTRA | typedef struct SCR\_screen\_adjuster\_dual\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[2];  TLine line1;  TLine line2;  // Data  uint16 tag;  SCR\_adjuster\_value\_t values[2];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_dual\_struct\* a, uint8 index);  } NVHR\_SCR\_screen\_fan\_dual\_speed\_t; |
| ***void***  ***NVHR\_SCR\_override\_durations(uint8 index)*** | Override durations setup with End-of-day:  Duration1  Duration2  Duration3  Duration4 with E-O-D | typedef struct SCR\_screen\_adjuster\_triple\_struct  {  // Widgets  TButton title;  SCR\_custom\_t back;  SCR\_custom\_t next;  SCR\_adjuster\_t adjuster[3];  TLine line1;  TLine line2;  // Data  uint16 tag;  uint16 tag2;  SCR\_adjuster\_value\_t values[3];  // Events  // This is not only for notifications but it is also  // required for formatting of display strings.  // Otherwise no values will be displayed.  SCR\_event\_handler\_t evt\_handler;  void(\*changed)(struct SCR\_screen\_adjuster\_triple\_struct\* a, uint8 index);  } NVHR\_SCR\_override\_durations\_t; |
| ***void***  ***NVHR\_SCR\_fan\_dual\_speed\_perform(uint8 index)*** | Calls - 2 speeds ***nvhr\_scr\_fan\_dual\_speed()***  with index:  STARTING  MINIMUM  WINTER\_SLOW  WINTER\_FAST  SUMMER\_SLOW  SUMMER\_FAST  ULTRA |  |
| ***void NVHR\_SCR\_fan\_single\_speed\_perform(uint8 index)*** | Calls – 1 speed  ***nvhr\_scr\_fan\_single\_speed()***  with index:  STARTING  MINIMUM  WINTER\_SLOW  WINTER\_FAST  SUMMER\_SLOW  SUMMER\_FAST  ULTRA |  |
| ***uint8 NVHR\_SCR\_translate\_fan\_to\_mcb\_speed(uint8 fan\_speed\_index )*** | Convenient function returning equivalent MCB speed enumeration given the fan\_speed\_index:  STARTING => WS\_STARTING  MIN =>  WS\_MIN  WINT\_SLOW =>  WINTER\_SLOW  WINT\_FAST =>  WINTER\_FLAST  SUM\_SLOW =>  SUMMER\_SLOW  SUM\_FAST =>  SUMMER\_FAST  ULTRA =>  WS\_ULTRA |  |

# HMI System\_info

|  |  |  |
| --- | --- | --- |
| **Public Method** | **Description** | **Data Structure** |
| ***void***  ***NVHR\_SCR\_system\_info(void)*** | The screens for user information display | typedef struct SCR\_screen\_vlist\_ex\_struct  {  SCR\_screen\_vlist\_t vscr;  char strings[SCR\_VLIST\_MAX\_ROWS][32];  uint8 indices[SCR\_VLIST\_MAX\_ROWS];  uint8 user;  } NVHR\_SCR\_screen\_system\_info\_t; |
|  |  |  |

# Summary

In this document, only the HMI’s public functions (with exception of ***nvhr\_scr\_fan\_single\_speed ()*** and ***nvhr\_scr\_fan\_dual\_speed () private functions***) prototyped in ***scr.h*** are covered.

# Further Work Recommendations

* Provide Logging mechanism.
* Provide a facility to distinguish Errors/Exceptions/Warnings/Information and  
  depict them in HMI accordingly. This should give the user a better understanding  
  of the nature issue flagged by the HMI system.
* Support other messages not covered yet (Defined in ***mcb\_config.h***):
  + MCB\_MESSAGE\_EE\_ARRAYS\_2
  + MCB\_MESSAGE\_EE\_ARRAYS\_3
  + MCB\_MESSAGE\_EE\_ARRAYS\_4
  + MCB\_MESSAGE\_EE\_ARRAYS\_5
  + MCB\_MESSAGE\_EE\_ARRAYS\_6
  + MCB\_MESSAGE\_EE\_ARRAYS\_7
  + MCB\_MESSAGE\_EE\_ARRAYS\_8
* Provide mechanism for setting user pin code.

# HMI Build Tools

* MikroC Pro for PIC32 IDE and development environment – Version 4.0.0   
  From [www.mikroe.com](http://www.mikroe.com)
* MPLAB IPE V.4.05 (Integrated Programming Environment)
* MPLAB ICD3 for firmware transfer to the HMI board
* HMI board with PIC32MX460F512L device
* SVN code repository

# HMI Build Process

* Make sure the MikroC Pro project ***nvhrAdvanceHmi.mcp32*** exists
* Compile and build the project using the MikroC Pro
* If successful, then *<project>.hex* is created (***nvhrAdvanceHmi.hex***)
* In IPE, select device ***PIC32MX460F512L***
* In IPE, Connect to ICD3 (e.g. ICD 3 S.No: JIT171610281)
* In IPE, Browse and select ***nvhrAdvanceHmi.hex*** file from a pre-defined location
* In IPE, Program the device
* In IPE, make sure ***Settings->Release from Reset*** is selected
* The HMI board should now be loaded with the new firmware

# Appendix A – NVHR HMI Screens Workflow

The following user requirement diagram shows the fundamental user interaction flow blocks of BB NVHR HMI.

This includes User, Maintenance, and commissioning screens, colour coded (Blue for User, Green for Maintenance and Red for Commissioning).

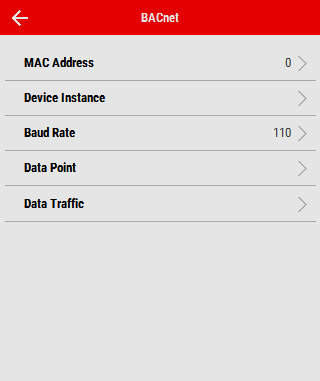
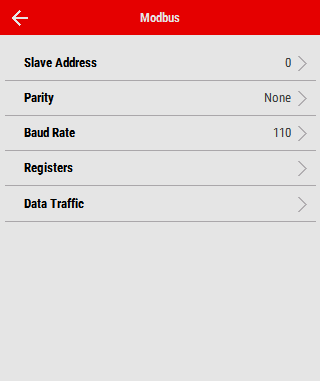
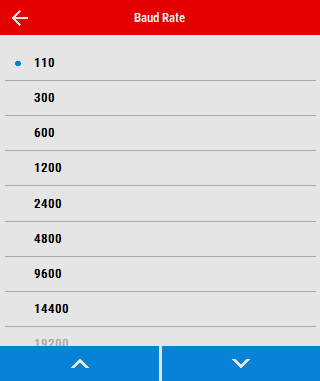
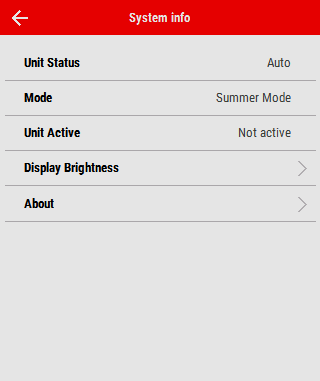
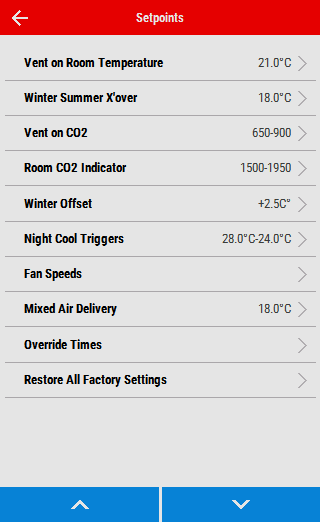
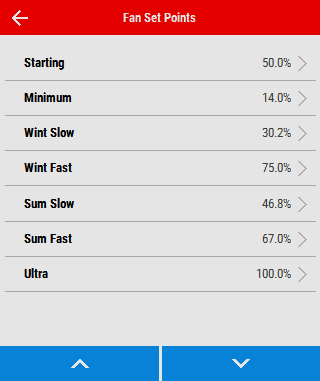
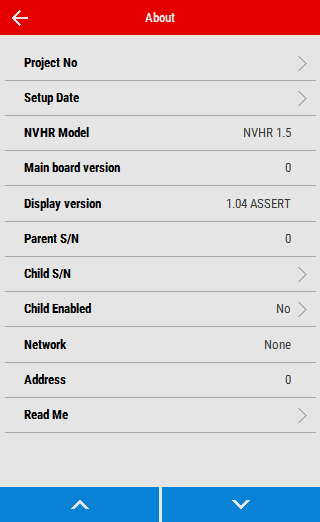
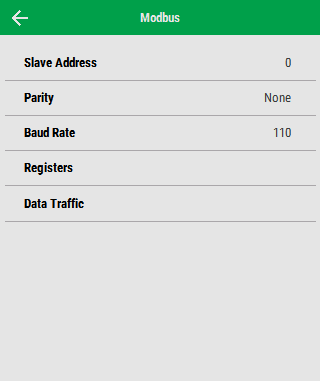
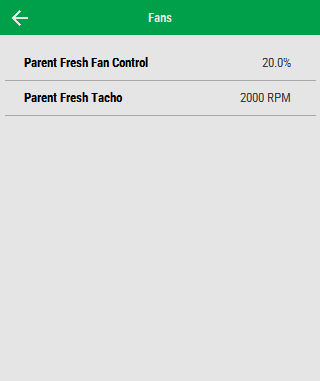
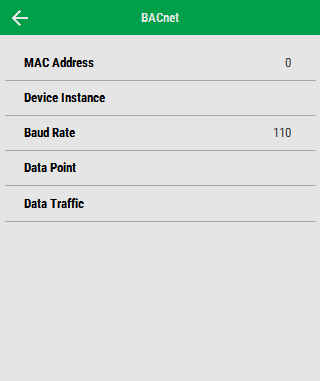
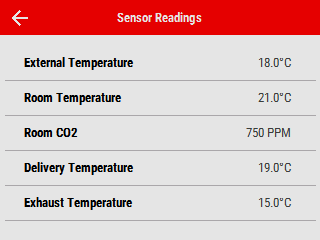
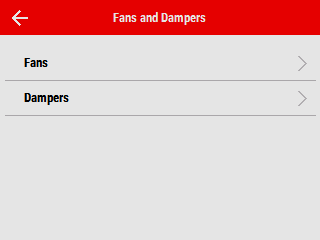
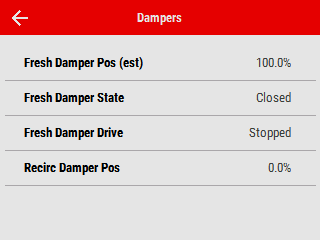
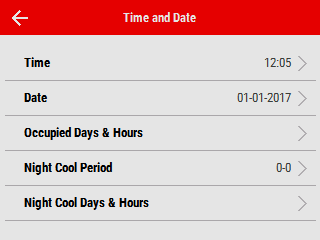
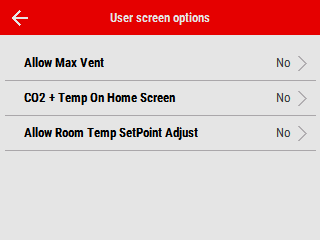
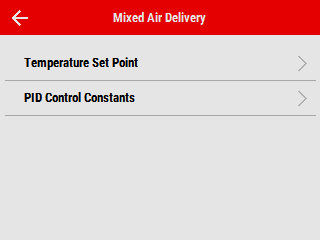
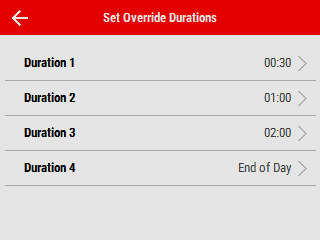
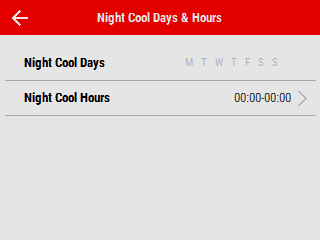
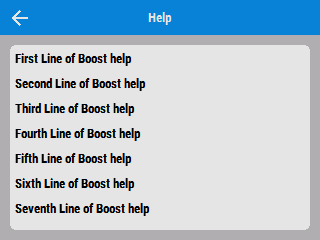
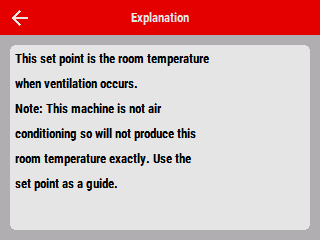
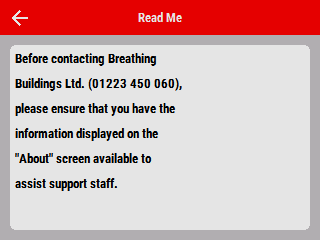
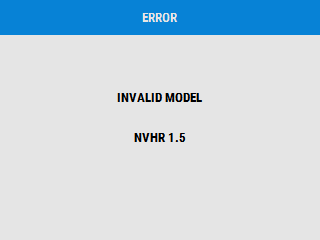
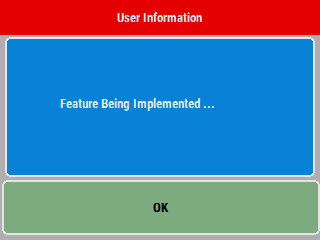
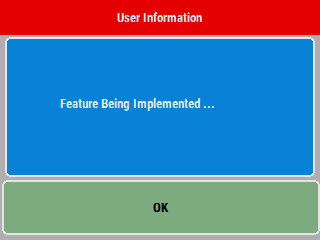
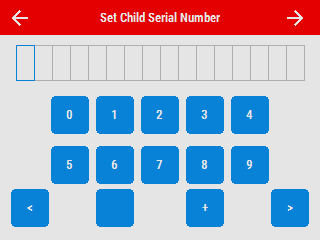
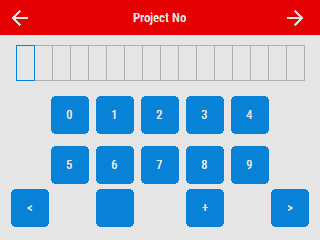
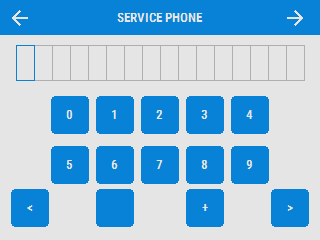
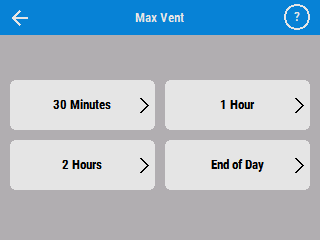
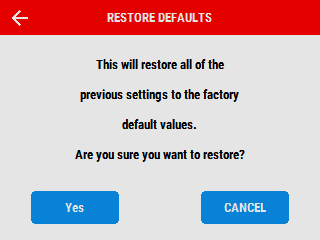
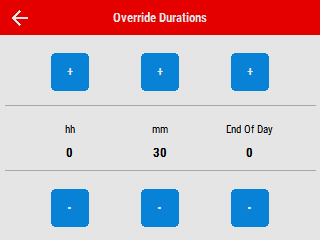
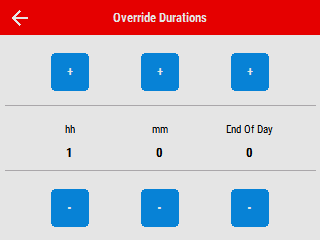
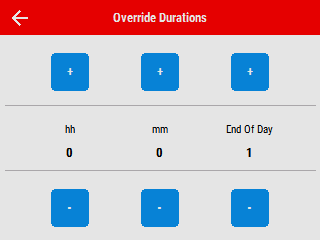
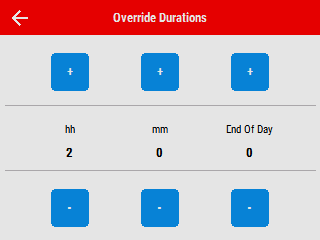
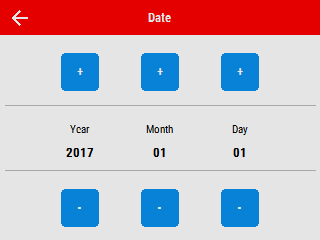
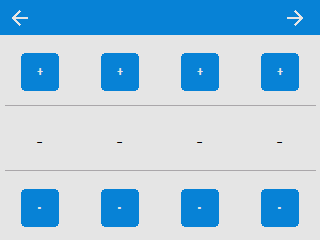
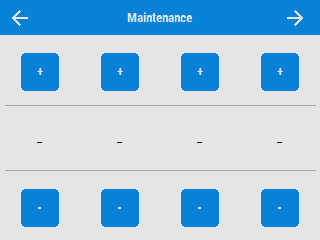
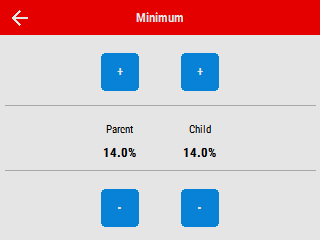
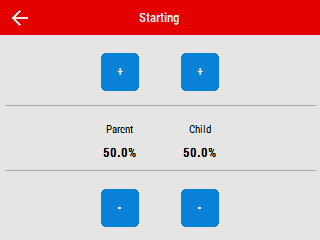
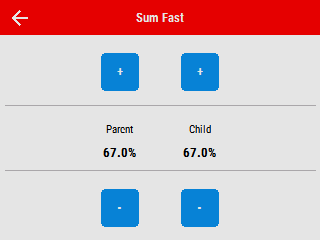
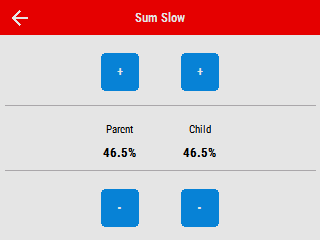
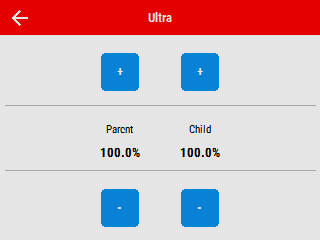
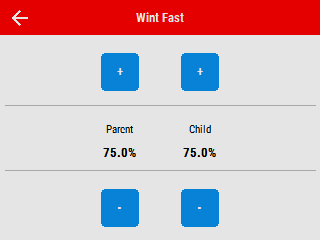
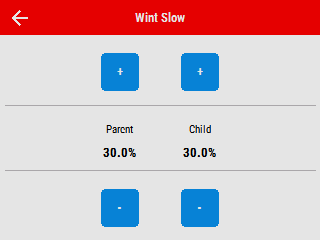
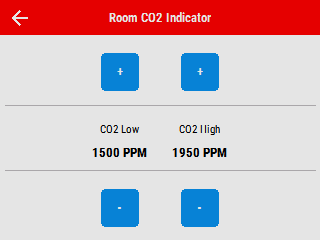
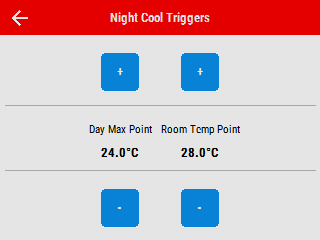
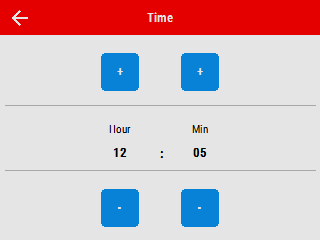
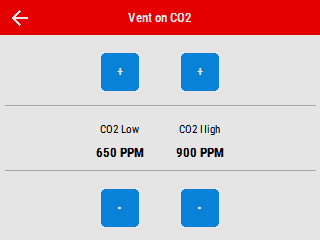
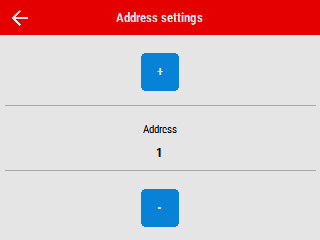
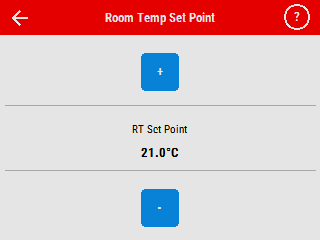
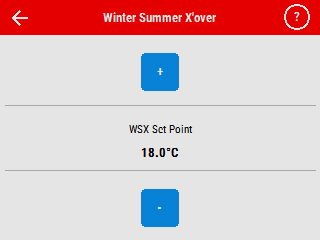
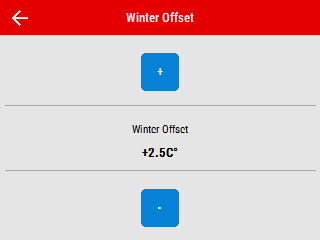
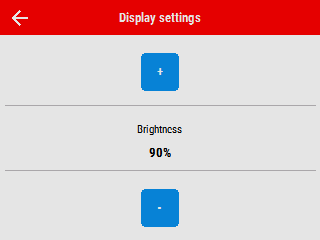
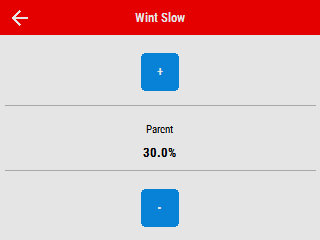
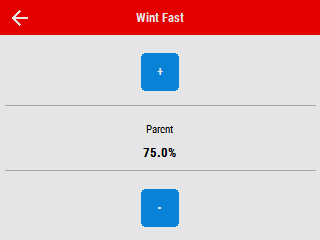
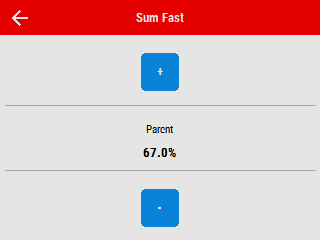
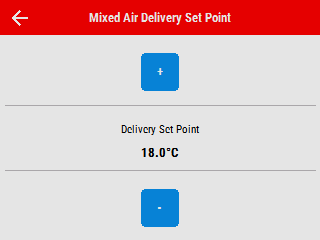
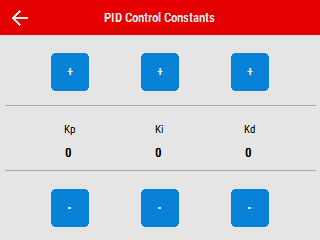
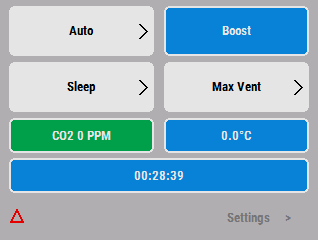
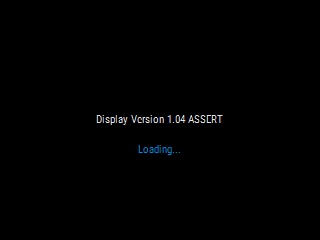
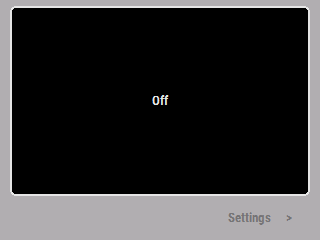
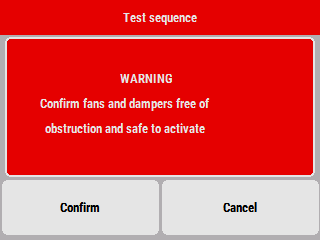
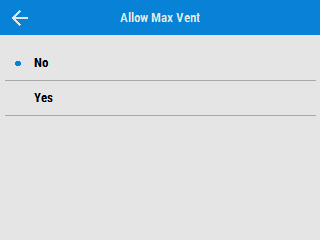
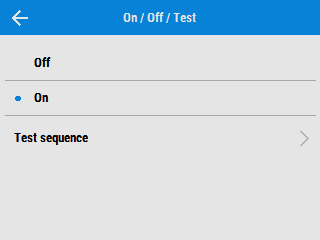
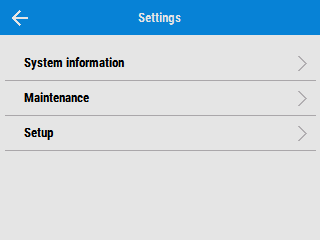
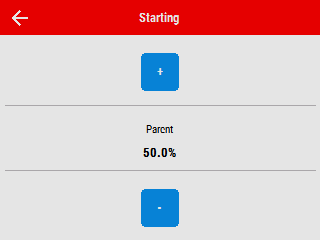
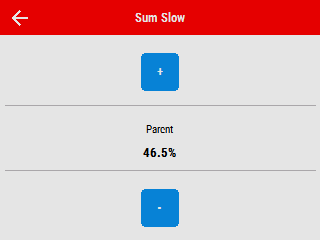
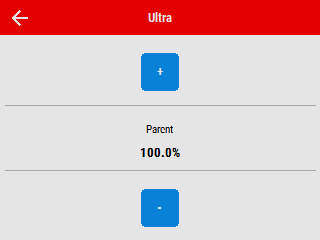
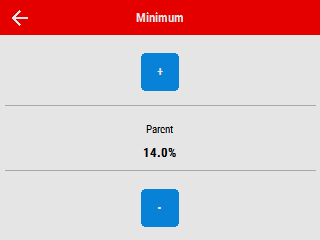
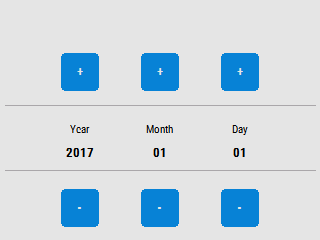
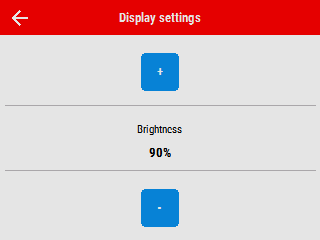
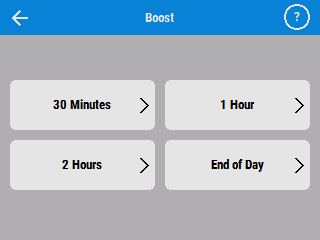
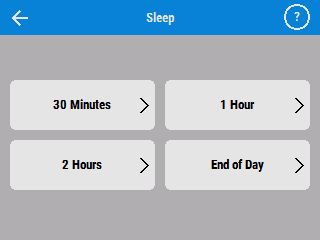
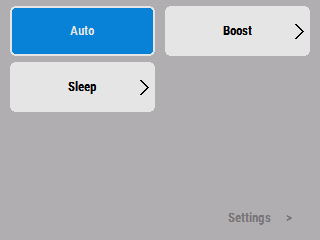
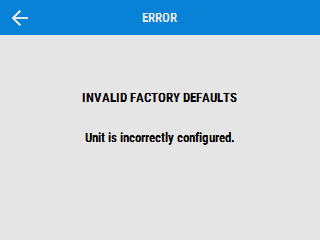
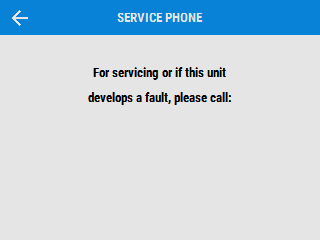
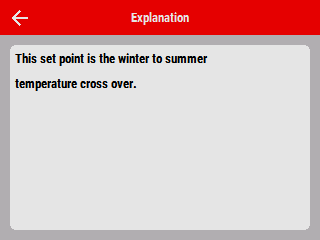
**User Menu**

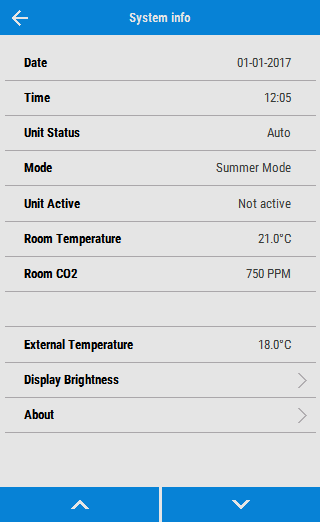
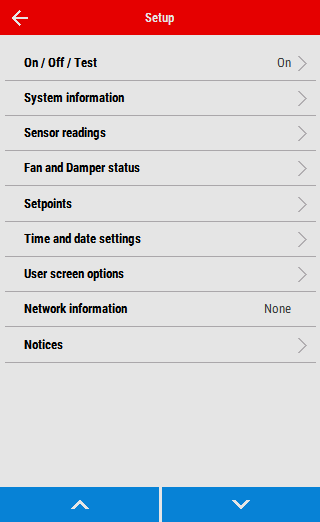
|  |  |  |  |
| --- | --- | --- | --- |
| ***Landing Menu*** |  |  |  |
| Auto \* |  |  |  |
| Boost > | ***Boost Menu*** |  |  |
|  | 30 min |  |  |
|  | 1 hr |  |  |
|  | 2 hr |  |  |
|  | End of Day |  |  |
| Sleep > | ***Sleep Menu*** |  |  |
|  | 30 min |  |  |
|  | 1 hr |  |  |
|  | 2 hr |  |  |
|  | End of Day |  |  |
| Max Vent > | ***Max Vent Menu*** |  |  |
|  | 30 min |  |  |
|  | 1 hr |  |  |
|  | 2 hr |  |  |
|  | End of Day |  |  |
| Settings > | ***Settings Menu*** |  |  |
|  | System Info > | ***System Info Menu*** |  |
|  |  | Date & Time |  |
|  |  | Unit Status |  |
|  |  | Mode |  |
|  |  | Unit Activity |  |
|  |  | Room Temp |  |
|  |  | Room CO2 |  |
|  |  | Room Temp Setpoint > | ***RT Set Point*** |
|  |  |  | Room Temp Set Point Adj |
|  |  | External Temp |  |
|  |  | About |  |
|  | Screen Options > | ***Screen Option*** |  |
|  |  | Brightness Adjust |  |
|  | Maintenance > |  |  |
|  |  | ***Maintenance Pin*** |  |
|  |  | Enter Pin > |  |
|  |  |  | ***Maintenance Menu*** |
|  | Setup > | ***Setup Pin*** |  |
|  |  | Enter Pin > |  |
|  |  |  | ***Setup Menu*** |
|  |  |  |  |

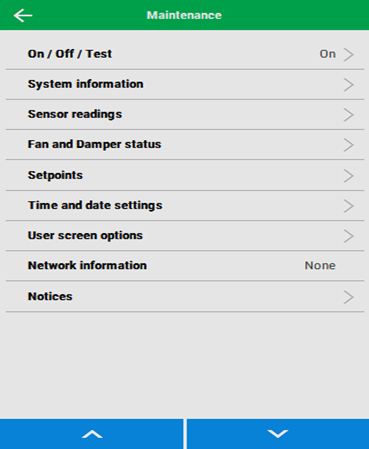
|  |  |  |  |
| --- | --- | --- | --- |
| ***Maintenance Menu*** |  |  |  |
| On/Off/Test> | ***On/Off/Test Menu*** |  |  |
|  | On |  |  |
|  | Off |  |  |
|  |  | Off Screen |  |
|  | Test Sequence> |  |  |
|  |  | ***Test Confirmation Menu*** |  |
|  |  | Cancel |  |
|  |  | Confirm | ***Test1*** |
|  |  |  | Cancel |
|  |  |  | ***Test2*** |
|  |  |  | Cancel |
|  |  |  | ***Test3*** |
|  |  |  | Cancel |
| System Info > | ***System Info Menu*** |  |  |
|  | Unit Status: |  |  |
|  | Mode: |  |  |
|  | Unit Activity: |  |  |
|  | Configuration: |  |  |
|  | About | ***About Information Screen*** |  |
|  |  | Project No.: |  |
|  |  | Setup Date: |  |
|  |  | NVHR Model: |  |
|  |  | Hardware Version: |  |
|  |  | Firmware Version: |  |
|  |  | Parent S/N: |  |
|  |  | Child S/N: |  |
|  |  | Network ID: |  |
|  |  | Protocol: |  |
|  |  | Slave Address: |  |
|  |  | Read Me> | ***Read Me Screen*** |
| Sensor Readings> | ***Sensor Readings Screen*** |  |  |
|  | External Temp: |  |  |
|  | Room Temp: |  |  |
|  | Room CO2: |  |  |
|  | Delivery Temp: |  |  |
|  | Exhaust Temp: |  |  |
| Fan & Damper Status> | ***Fan & Damper Menu*** |  |  |
|  | Fans> | ***Fan Information screen*** |  |
|  |  | Parent Fresh: |  |
|  |  | Child Fresh: |  |
|  | Dampers> | ***Damper Info Screen*** |  |
|  |  | Fresh Damper: |  |
| Set Points> | ***Set Point Menu*** |  |  |
|  | Vent On Room Temp: |  |  |
|  | Winter Sum Xover: |  |  |
|  | Vent On Room CO2 |  |  |
|  | Winter Offset |  |  |
|  | Temp Trigger: |  |  |
| Time & Date Settings | ***Time & Date Menu*** |  |  |
|  | Time > | ***Time Adj Screen*** |  |
|  |  | Hour |  |
|  |  | Min |  |
|  | Date > | ***Date Adj Screen*** |  |
|  |  | Year YYYY (+/-) |  |
|  |  | Month MM (+/-) |  |
|  |  | Day DD (+/-) |  |
|  | User Screen Options> | ***User Screen Options Menu*** |  |
|  |  | Allow Max Vent |  |
|  |  | All Room Override |  |
|  |  | Allow Room Temp Adj |  |
|  | Network Information > | ***BACnet Info Screen*** |  |
|  |  | Baud Rate |  |
|  |  | MAC Address |  |
|  |  | ***Modbus Info Screen*** |  |
|  |  | Slave Address |  |
|  |  | Baud Rate |  |
|  |  | Parity |  |
|  |  | Stop Bit |  |
|  | Notices > | ***Notices Menu*** |  |
|  |  | Fans> | ***Fan Error Screen*** |
|  |  | Dampers> | ***Damper Error Screen*** |
|  |  | Sensors > | ***Sensor Screen*** |
|  |  | Contact Details> | ***Contact Details Screen*** |
| ***Setup Menu*** |  |  |  |
| On/Off/Test | ***On/Off/Test Menu*** |  |  |
|  | Same as Maintenance |  |  |
| System Info > | ***System Info Menu*** |  |  |
|  | Unit Status: |  |  |
|  | Mode: |  |  |
|  | Unit Activity: |  |  |
|  | About > |  |  |
|  |  |  |  |
| Sensor Readings > | Same as Maintenance |  |  |
| Fan & Damper Status > | Same as Maintenance |  |  |
| Set Points > | Same as Maintenance |  |  |
|  | + Fan Speeds > | ***Fan Speed Set Points*** |  |
|  |  | Set Point F1 > | ***Fan Speed Adj Screen*** |
|  |  | Set Point F2 > | Parent Fresh Adj (+/-) |
|  |  | Set Point F3 > | Child Fresh Adj (+/-) |
|  |  | Set Point F4 > |  |
|  |  | Set Point F5 > |  |
|  |  | Set Point F6 > |  |
|  |  | Set Point F7 > |  |
|  |  | Set Point F8> |  |
|  | + Override Times | ***Set Override Durations*** |  |
|  |  | Duration 1 | ***Override Adj Screen*** |
|  |  | Duration 2 | hh adj (+/-) |
|  |  | Duration 3 | mm Adj (+/-) |
|  |  | Duration 4 / End of Day | End of Day Adj (Yes/No) |
| Time & Date Settings > | Same as Maintenance |  |  |
|  | + Occupied Days | ***Occupied Days Screen*** |  |
|  | +Night Cool Period | ***Night Cool Adj Screen*** |  |
|  | +HMI Screen Controls | ***HMI Screen Options Menu*** |  |
|  | +Network Information | ***BACnet Info screen*** |  |
|  |  | ***Modbus info screen*** |  |
|  | Notices > | Same as Maintenance |  |
|  | Reset | ***Reset to Factory Settings*** |  |

# Appendix B – NVHR HMI Screenshots

The following pages depict the screen shots captured from NVHR HMI application.







1. Source to be made available in maintenance / commissioning screens [↑](#footnote-ref-1)
2. In future update, with night cool cancellable, Auto is not highlighted and clickable to cancel (for a given period) [↑](#footnote-ref-2)