

HMC-20

7” configurable Human Machine Controller

Revision: 23/01/2025



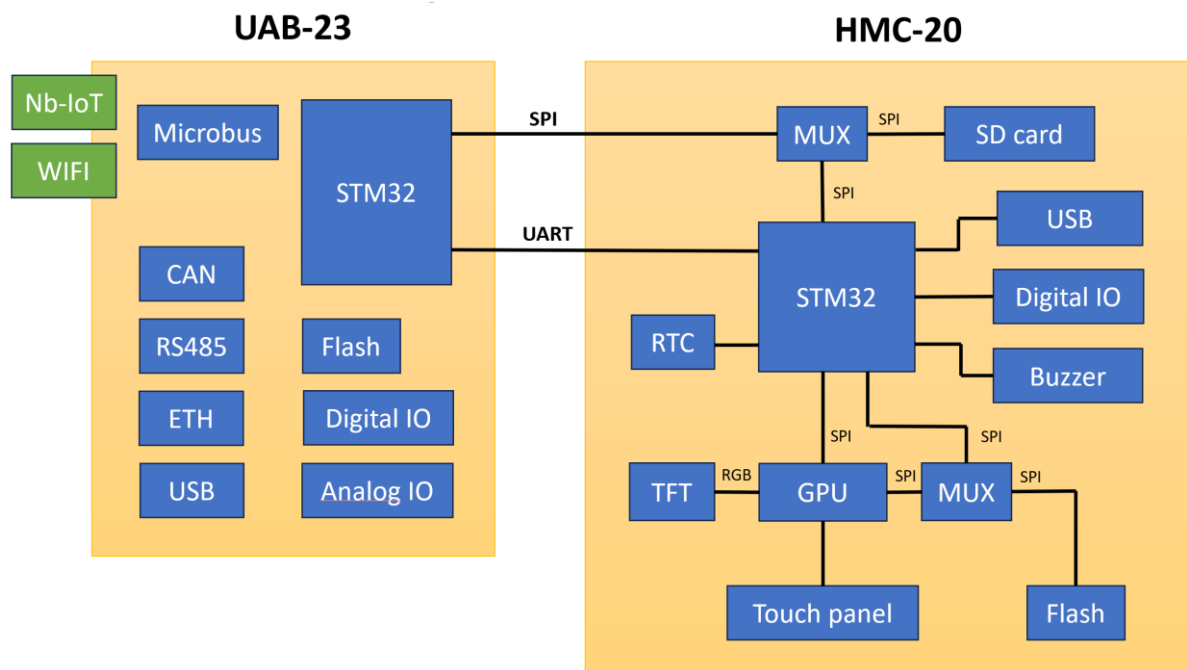
Content

1. Introduction.....	3
2. Application Domains.....	3
3. Hardware Features	4
4. Software Features	6
5. Specifications	6
5.1. Absolute Maximum Ratings	6
5.2. Electrical Characteristics.....	6
5.3. Optical Characteristics	7
5.4. Physical Characteristics.....	7
6. Connections	7
6.1. Power interface.....	7
6.2. IO interface	8
6.3. UA board interface	9
7. Real time interpreter commands	10
7.1. Visible widget parameters.....	12

7.2. Non visible widget parameters	14
7.3. General commands	14
8. User console	15
8.1. Clear full memory.....	15
8.2. Clear data for all applications.....	16
8.3. Dump current layout to json	16
8.4. Load application from memory	16
8.5. Store application from SD card	16
8.6. Set clock.....	17
8.7. Firmware updates.....	18
9. Json layout format specification	18
10. Certifications.....	19
11. Order codes.....	19

1. Introduction

The HMC-20 *Human Machine Controller* in combination with the UAB-23 *User Application Board* forms a multi-functional and fully configurable Human Machine Interface console that can be used for a wide range of applications. The split-up between a graphical and an application part makes it possible to use the console in different environments with a dedicated board with specific IO for user defined purposes. Both boards are available separately, and the UAB-23 design is open source for further customization. A case is available to mount both boards in a stackable way with invisible cabling. Available display sizes are 3,4” and 7”. There is auditory feedback that produces different sound effects when touching buttons. The HMC-20 contains 5 digital inputs, 5 digital outputs with selectable voltage, an SD-card interface, USB2.0 interface with virtual UART console, RTC with battery, power interface and a user application board connector. The TFT display has a capacitive or resistive touch interface and can be dimmed with PWM or put in sleep mode. The layout, interactions, images and widgets are configurable by the user by putting them on the SD-flash card as BMP/JPG images. Properties and UI-behavior is defined in a Json file. The application board connector contains a real time command interpreter to drive the graphical part with simple textual commands.

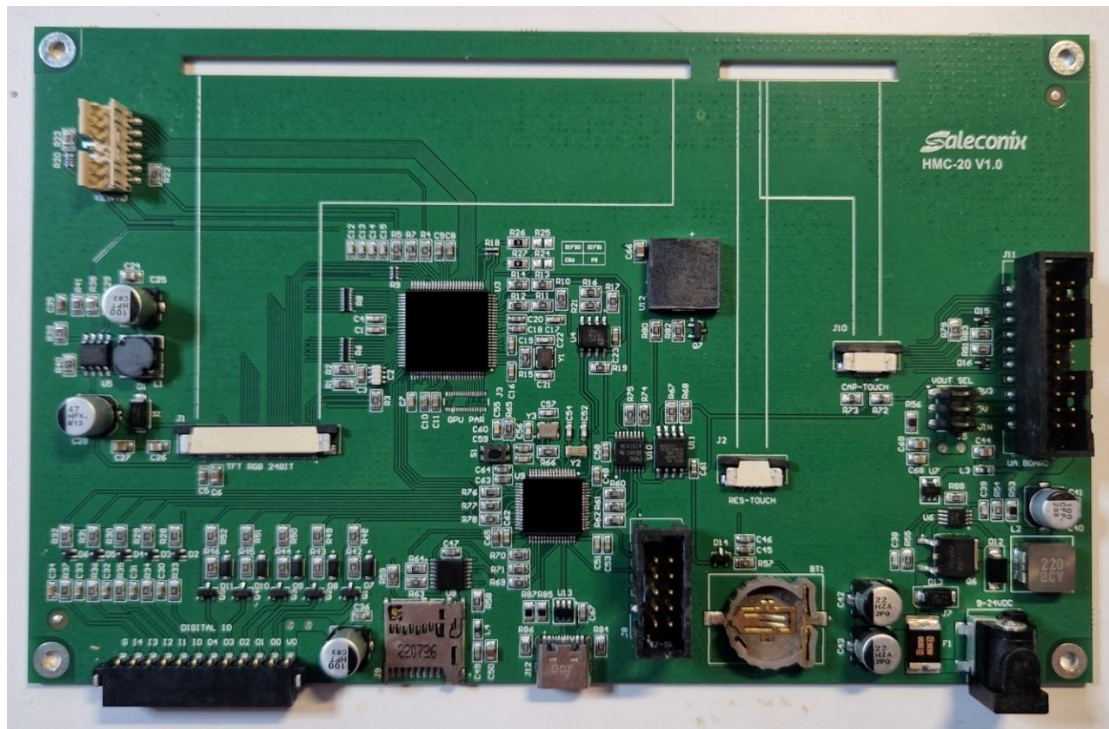


2. Application Domains

- General purpose HMI with static or animated graphics
- Agriculture (greenhouse, grow room)
- Buildings (home automation, liquid levels)
- Automotive (tank levels, battery level)
- Industrial applications

3. Hardware Features

HMC-20 board contains the graphical part that consists of the following parts (from left to right):



1. 7" 24bit 800x480pixels RGB TN TFT display, 1000cd/m2 with 60 degree viewing angle.
2. Keypad interface (J4) to attach a keyboard with up to 4x5 physical buttons in a matrix pattern to be used as an alternative for the touch panel interface.
3. PWM controlled LED backlight convertor generates 10V/300mA boost conversion for high brightness TFT display.
4. 5 digital Inputs and 5 digital Outputs with voltage selector. Internal pullup allows reading buttons or driving signals. Protected by diodes to drive relays or voltage driven LED-strips.
5. 24bit RGB FPC connector for TFT display, includes connections for backlight, video synchronization and resistive touch interface.
6. Graphics Processing Unit (GPU) allows putting graphical shapes, fonts and images on the display. 256/65K colors, up to 800x480 pixels. Internal and user defined font ROM 8x16 dots with enlargement and rotation function.
7. Microcontroller Processing Unit (MPU) with external clock, reset button, programming connector and high accuracy Real Time Clock (RTC). Acts as the central controller between all interfaces and GPU.

8. SD-card interface to import BMP/JPG images into board flash and read the layout from a json file. Once copied, the flash card can be removed. Firmware updates are also copied from the card with an internal bootloader.
9. EMC protected USB2.0 interface to access flash card as a drive¹ and to show a command menu over a virtual serial port to interact with the display.
10. Font ROM up to 15x16 dots², GB12345/BIG5 (Chinese) and JIS0208 (Japanese) character set. Language support for 150 countries including Latin, Cyril, Greek and Arabian.
11. 64Mb serial NOR flash to save images, layout and data with optimized index and HMAC (Hash-based message authentication code) for security purposes.
12. Piezo electric buzzer for auditory feedback allows the generation of different sound effects when pressing buttons.
13. Resistive and capacitive touch interface for touch control from the TFT panel. The capacitive touch data is read directly from the MPU. The touch controller is ready to support gesture and Hotknot. (capacitive transmission of data between screens)
14. Holder for 3V battery to keep real time clock running when no external power source is available.
15. User Application board interface connector (J11) to send serial commands to command interpreter in human readable format. Compatible with 3V3 FTDI serial to USB convertor cable. Contains access lines to SD flash card to store files from an external source.
16. Power supply that generates 5V at max 4A from a wide voltage input range 9-24VDC.

¹ Still in development

² Other font chips are possible up to 32x32 dots. Contact us for further support.

4. Software Features

- Predefined visible/non-visible widgets that are fully configurable and assignable to pages
- Real time command interpreter to send and receive sequences defined as widget events or from/to UART to modify properties and render pages
- Direct Memory Access (DMA) interface to store images from SD-card and save data on flash chip with indexing mechanism and HMAC to ensure data integrity
- Layout importing/exporting mechanism from Json file
- Calibration mechanism for resistive touch displays
- Initialization procedures
- Real time clock setting mechanism
- Rotation function: 0, 90, 180, 270°
- Firmware upgrades from a bin file and internal bootloader
- 3 demo apps: button, fogger and runlight with user application as a Python script

5. Specifications

5.1. Absolute Maximum Ratings

Parameter	Min	Typ	Max	Unit
Input Supply Voltage	-0.3	-	36	V
Operating Temperature	-20	-	70	°C
Storage Temperature	-30	-	85	°C
Storage Humidity	20	-	90	%RH
Digital Input Voltage	-36,7	-	40	V

5.2. Electrical Characteristics

Parameter	Min	Typ	Max	Unit
Input Supply Voltage	4,5	-	35	V
Maximum Power consumption ³	TBD	-	4	W
Digital Input Vth Low	-	-	1,254	V
Digital Input Vth High	1,85	-	-	
Digital Output Voltage ⁴	-	3,3/5/Vin	-	
Maximum output current	-	-	1,6	A
Maximum total output power @5V	16 ⁵	-	20	W
Maximum total output power @3V3	-	-	0,825 ⁶	
Serial port speed	-	67800	-	Bd

³ When digital output is not delivering power

⁴ Selectable with J5

⁵ When PWM display is at 100%

⁶ When PWM display is at 100%

5.3. Optical Characteristics

Parameter	Min	Typ	Max	Unit
Brightness	900	1000	-	cd/m ²
Horizontal Viewing Angle	60	70	-	degree
Vertical Viewing Angle Top	40	50	-	
Vertical Viewing Angle Bottom	60	70	-	
Contrast ratio	400	500	-	-
Uniformity	-	80 (If=270mA)	-	%
Life Time	-	50000 (If=270mA)	-	Hour
Response Time Rising	-	10	20	ms
Response Time Falling	-	15	30	
CF Color CIE 1931	-0,05	0,05	-	-
Backlight Color	White			
Colors	256			
Bit per pixel	8			

5.4. Physical Characteristics

Parameter	Min	Typ	Max	Unit
TFT size	-	7	-	Inch
Active display area	-	154,08(W)x85,92(H)	-	mm
Dot pitch	-	0,1926(W)x0,1790(H)	-	
Display dimension	-	164,9(W)x100(H)	-	
Board dimension inc. display	-	179,9(L)x111(W)x22(H)	-	
Mounting Hole Size	-	3	-	
Weight	-	TBD	-	g

6. Connections

6.1. Power interface

Pin	Function
Central pin	Positive power supply input
Border contact	Negative power supply input

6.2. IO interface

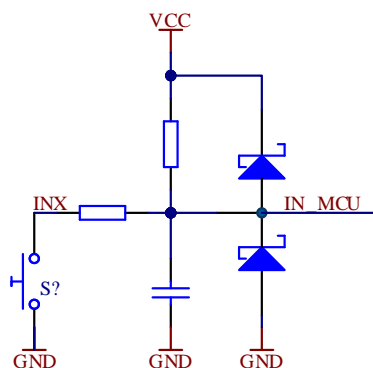
The IO interface (J6) contains 5 digital outputs and 5 digital inputs.

Pin number	1	2	3	4	5	6	7	8	9	10	11	12
Function	G	I4	I3	I2	I1	I0	O4	O3	O2	O1	O0	VO

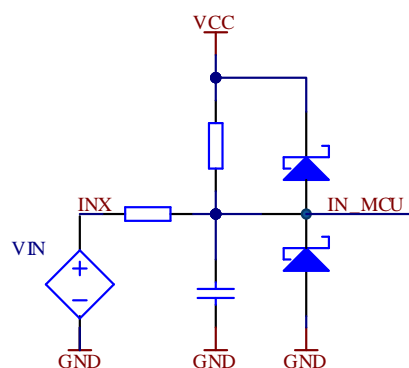
Function	Info	Function	Info
G	Ground	O4	Output4
I4	Input4	O3	Output3
I3	Input3	O2	Output2
I2	Input2	O1	Output1
I1	Input1	O0	Output0
I0	Input0	VO	Output voltage

Input configurations:

The inputs are flexible, debounced and protected against spikes and overvoltage. There are two different configurations that can be used:



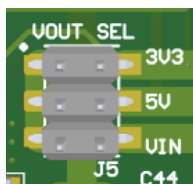
Input button configuration with internal pullup resistor (100k)



Active input voltage

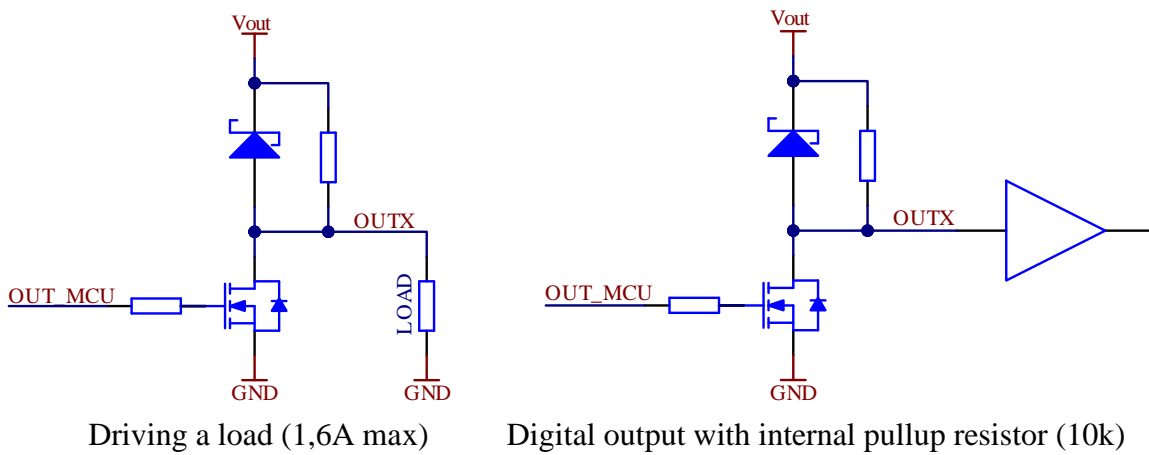
Output configurations:

The output voltage (VO) can be selected with jumper J5 as in following table:



1-2	3V3 (=Vcc)
3-4	5V
5-6	VIN

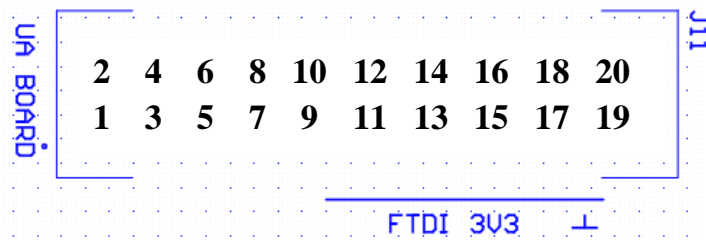
The outputs are protected with an internal diode against inductive loads and can be used in the following configurations:



Using higher Vout values allows reduction of the current to drive led strips, etc. The total power delivered by all outputs may not exceed 16W.

6.3. UA board interface

The user application board interface (J11) contains a serial port to send and receive commands to and from the command interpreter and SPI access to the SD flash card to store files from an external source.



Pin number	Function	Info	Pin number	Function	Info
1	Vin	Input Power voltage	2	Vin	Input Power voltage
3	5V	Regulated voltage	4	5V	Regulated voltage
5	3V3		6	3V3	
7	NC		8	NC	
9	NC	Command interface	10	NC	SD card interface
11	Rx		12	MOSI	
13	Tx		14	CLK	
15	NC		16	MISO	
17	NC		18	SDSEL	
19	GND		20	GND	

7. Real time interpreter commands

There are three formats for commands that the interpreter recognizes:

Property assignment for widget properties:

```
[pagename].[widget].[propertyname]=[property]\n
```

Property assignment for page properties:

```
[pagename].[propertyname]=[property]\n
```

Command for general actions:

```
[command] [argument]\n
```

Every command is ended with a newline character '\n'. This makes it also possible to enter the commands in command line for testing purposes. Some examples are:

```
"code.unlockkeypad.cmd='exe'\n" // unlock the keypad
"code.enter.act=true\n" // unlock the enter button
"code.crosshome.act=true\n" // unlock the cross-home button
"code.ok.vis=false\n" // Make the text 'ok' invisible
"code.pincodesaved.vis=false\n" // Make the text 'pincodesaved' invisible
"code.popup.vis=false\n" // Make the green popup window invisible
"page 'code'" // Redraw the code page
```

Properties and arguments can be of the type integer, Boolean or string. Strings are always quoted with single quotes. Embedded strings are quoted again in the deeper level of the command and are escaped by a double escape character '\\' belonging to each level. For example, the following command:

```
"code.pincod.pe='code.enter.tpe=\\'code.pincod.cmd=\\\\\\'save\\\\\\'\\'\\'\n"
```

Has the following meaning (read from left to right):

“Set the positive evaluation event (pe) of the pincod widget (pincod) on the code page (code) to update the touch press event (tpe) of the enter button (enter) on the code page (code) to save the pincod by executing a command (cmd) with argument ‘save’ on the pincod widget (pincod) on the code page (code).

A breakdown gives:

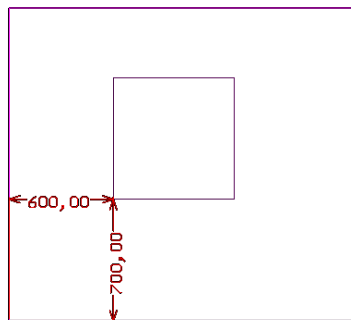
```
"code.pincod.pe='A'\n"           -> level 0 (no escapes)
A = code.enter.tpe= \\'B\\'      -> level 1 (double escapes)
B = code.pincod.cmd=\\\\\\'C\\\\\\' -> level 2 (quadruple escapes)
C = save
```

Widgets are divided into visible and non-visible types. Within the visible types, there is a subgroup called ‘*text types*’, which can contain readable text. Every widget must be assigned to a page and includes both common and specific parameters. Some parameters can be updated dynamically and made directly visible or updated in silent mode. In silent mode, the parameters are updated in the background and become visible only when a page is rendered or when the silent update mode is turned off.

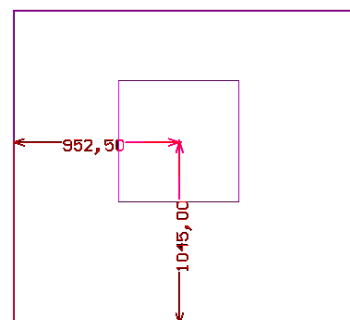
All widgets have a name that is unique to each page and acts as the reference handle.

Property	Info	Data type	Range	Json
Name	The name which acts as a unique identifier and is always mandatory	String	-	“name”

The position of the visual widgets is defined by coordinates x and y and a selected coordinate system. The **corner coordinate system** defines the coordinates of the widget in the left bottom corner of the widget canvas referenced to the left bottom of the screen. The **center coordinate system** defines the coordinates of the widget in the center point of the widget canvas references to the left bottom of the screen. Select the system that is most appropriate to perform your calculations.



Corner coordinate system



Center coordinate system

7.1. Visible widget parameters

Note: Json indicated with * are mandatory properties in the Json specification

Common visible type widget parameters:

Type	Property	Comment	Data type	Range	Default value	Command identifier	Json
Property	Background	Defines the status of the background. A solid button could be a confirmation button and an image button could be a touchable symbol.	String	'noback' 'contour' 'solid' 'image'	'noback'		"background"
	Background color ⁷	The color of the background when the type is 'solid'	Integer	16 bits	-	bco	"backcolor" *
	Image	When the type is 'image', the image can be selected by a unique name	String	-	""	img	"image"
	X	The coordinates of the widget using the defined coordinate system	Integer	16 bits	0		"Xloc"
	Y						"Yloc"
	Width	The size of the widget must be the same size of the background image	Integer	16 bits	0		"width"
	Height						"height"
	Extended width	The extended size defines the area that is touch sensitive	Integer	16 bits	0		"exwidth"
	Extended height						"exheight"
	Sound type	Define the sound type of the auditory interface when the widget is touch pressed					
	Clock part	Define the part of the clock when the widget is used to represent a time or a date	String	'noclock' 'YEAR_3' 'YEAR_2' 'YEAR_1' 'YEAR_0' 'MON_H' 'MON_L' 'DAY_H' 'DAY_L' 'HOUR_H' 'HOUR_L' 'MIN_H' 'MIN_L' 'SEC_H' 'SEC_L'	'noclock'		"clockpart"
Corner coordinate system	Define if the corner coordinate system is used instead of the center coordinate system	Bool	true/false	false		"ccordsys"	
Border color	Define the color of the widget canvas as an integer value	Integer	16 bits	-		"bordercolor" * when border = true	
Behavior	Active	When the widget is not active, it will not react on touch events	Bool Integer	true/false 0/1	true	act	"active"
	Border	Show the border canvas of the widget	Bool Integer	true/false 0/1	false		"border"
	Sound	Enable or disable the touch press sound	Bool Integer	true/false 0/1	true		"sound"
	Visibility	Make the widget visible or invisible on the screen	Bool Integer	true/false 0/1	true	vis	"visible"
	Image increment	When the image is used as a counter or an animation, the increment command renders the next image in the alphabetic sequence. The value is the maximum increment.	Integer	16 bits	-	imi	-
	Image decrement	When the image is used as a counter or an animation, the decrement command renders the previous image in the alphabetic sequence. The value is the minimum decrement.	Integer	16 bits	-	imd	-
	Offset	When the image is used as a counter or an animation, the offset command renders a specific image on an offset in the alphabetic sequence.	Integer	16 bits	0	ofs	-

⁷ The background color is calculated by the macro
`RGB(r,g,b) (((r<<8)&0xF800)|((g<<3)&0x07E0)|(b>>3))` where r,g,b are values from 0 to 0xFF

Event	Touch press	Define the touch press event	String	-	""	tpe	"e_press"
	Touch release	Define the touch release event	String	-	""		"e_release"
	Move up	Define the move up event	String	-	""		"e_moveup"
	Move down	Define the move down event	String	-	""		"e_movedown"
	Move left	Define the move left event	String	-	""		"e_moveleft"
	Move right	Define the move right event	String	-	""		"e_moveright"

Common text type visible type widget parameters:

Type	Property	Comment	Data type	Range	Default value	Command identifier	Json
Property	Text	Static text that will be used when the internal text is empty	String	-	""	-	"text"
	Internal text	Dynamic text that is changeable by commands and has precedence over the property 'Text'	String	-	"" "--" (textual)	txt	-
	Font color	The color of the text	Integer	16 bit	-		"fontcolor" *
	H Scale	Horizontal scale of the text	Integer	16 bit	1		"hscale"
	V Scale	Vertical scale of the text					"vscale"
Behavior	Soft Font ⁸	The font selected from the firmware	String	'BPG_Arial08x08_F' 'BPG_Arial08x08' 'BPG_Arial10x10' 'BPG_Arial10x12' 'BPG_Arial20x20' 'BPG_Arial29x32' 'BPG_Arial31x32' 'BPG_Arial63x63'	""		"softfont"
	Rom Font	The font selected from the font ROM chip	String	'norom' 'ascii_8x16' 'ascii_8x16b' 'ascii_12'	'norom'		"romfont"
	Internal Font	The font selected from the graphical chip	String	'ISO8859_1' 'ISO8859_2' 'ISO8859_3' 'ISO8859_4'	'ISO8859_1'		"intfont"

Specific text type visible type widget parameters:

Widget	Type	Property	Comment	Data type	Range	Default value	Command identifier	Json
Button	Property	Type	The type defines the shape of the button	String	'rectangle' 'ellipse' 'triangle_right' 'triangle_left'	'rectangle'		"type"
Textual	Behavior	V align	Vertical alignment of the text	String	'top' 'middle' 'bottom'	'middle'		"valign"
		H align	Horizontal alignment of the text	String	'left' 'center' 'right'	'center'		"halign"
		Textwrap	Wrap the text in the textual canvas	Bool	true/false	false		"textwrap"

⁸ The precedence rule is 1/ softfont – 2/ romfont – 3/ intfont

Specific visible type widget parameters:

Widget	Type	Property	Comment	Data type	Range	Default value	Command Identifier	Json
Page	Behavior	Silent update	When the silent update mode is on, the properties of the widgets attached to that page are updated in the background to avoid transition effects	Integer	0/1	0	sil	-
Led	-	-	-	-	-	-	-	-
Progress bar	Property	Value	The value of the progress bar	Integer	8 bits	0	val	“value”
	Behavior	Type	The type of the progress bar	String	‘up’ ‘down’ ‘left’ ‘right’	‘up’		“pbtype”

7.2. Non visible widget parameters

Widget	Type	Property	Comment	Data type	Range	Default Value	Command Identifier	Json
Timerblock	Property	Time	The time of the timer in ms unit	Integer	32 bits	0		“time”
	Behavior	Start/Stop	Start or stop the timer	Integer	0/1	0	en	-
	Event	Run out	Sequence to execute when the timer is running out	String	-	“”		“e_timer”
Pincode	Behavior	Command	Evaluate the pin code	String	‘eval’	-	cmd	-
			Safe the pincode defined by the attached widgets	String	‘save’			
	Event	Pass	Sequence to execute in the case of a pass evaluation	String	-	“”	pe	“e_pass”
		Fail	Sequence to execute in the case of a fail evaluation	String	-	“”		“e_fail”
		Saved successful	Sequence to execute when the pincode has been saved successfully	String	-	“”		“e_savedsuccess”
Function	Behavior	Execute	Execute the specified sequence	String	‘exe’	-	cmd	-
	Event	Execute	Sequence to execute when the function is called	String	-	“”		“e_execute”

7.3. General commands

Action	Comment	Data type	Range	Default value	Command Identifier
Page render	Render the page defined by a specific name	String	-	-	page
Send command	Send a command to the serial command output	String	-	-	get
Print screen	Save a print screen on the SD flash card. The data must be a dummy value and has no effect.	Integer	16 bits	-	pscr
Output	Set digital output X [0 - 4] on, off or toggle the output. When the data is an integer, use PWM output and it defines the percentage of the duty cycle.	String	‘on’ ‘off’ ‘toggle’	-	outX
		Integer	8 bit	-	

8. User console

When the USB-C cable is connected, a virtual serial port becomes available and can be opened at any baud rate. A convenient command-line tool, such as *SimplySerial*, can be used to automatically restore the connection when the cable is unplugged and plugged in.

Download the installer tool from: <https://github.com/fastddy516/SimplySerial>

And enter the command:

```
C:\windows\System32>ss -echo:ON
```

Then you get a menu like below:

```
<<< SimplySerial v0.9.2 connected via COM3 >>>
Settings : 9600 baud, no parity, 8 data bits, 1 stop bit, UTF-8 encoding, auto-connect
on, echo on, tx_linefeed:CR
Device   : VID:1F00 PID:2012
---
Use CTRL-X to exit.

HMI PANEL 1.0.0+0
-----
Build Jan 17 2025 20:07:28
Total flash size: 64Mb
Index[14,95%] | Storage[30,12%]
www.saleconix.be | info@saleconix.be
2024 All rights reserved.

Settings
-----
Time: 17/01/2025 20:04:51
Display size WxH: 800x480
Backlight brightness: 100%
Debug port speed: virtual
Command port speed: 120000Bd
LCD bits per pixel: 8
Current app selected: haut

Menu
----
s: Settings menu
i: Info menu
h: Help menu
1: Clear full memory
2: Clear data for all applications
3: Dump current layout to json
4: Load application from memory
5: Store application from SD card
6: Set clock
7: Firmware update
```

It shows the info menu and settings, and several user options can be entered.

8.1. Clear full memory

An index is stored in a reserved part of the memory to reference the address locations of images, data, and JSON apps. During the booting process, this index is read from memory. An **HMAC** (Hash-Based Message Authentication Code) is calculated over the index to ensure data integrity when it is loaded. This adds a layer of security to prevent unauthorized modifications to the flash memory.

The option to clear the full memory removes only the index, *not the actual flash content*. When new applications are copied from the SD card into memory, all old memory locations are overwritten.

8.2. Clear data for all applications

Some applications store data in memory, such as the PIN code for the pincode widget or other user settings. This option clears only the index that tracks this data, not the image or application sections. Note that the actual data is not deleted from the flash memory. You can use this option to reset user data to default values.

8.3. Dump current layout to json

When an application is loaded, an internal widget tree is built that contains all properties. The current state of this tree can be dumped to the screen in JSON format, which can then be copied and pasted into a JSON file. This format is exactly what will be accepted when the file is read from the SD card. Note that some properties may depend on the current state of the application.

8.4. Load application from memory




When applications are stored from the SD card into memory, one of them becomes active when selected in the menu using the option 'load application from memory.' This setting is stored, and the same application will be loaded automatically on the next boot.

8.5. Store application from SD card

Applications must be available on the SD card in a specific format. Each application must be in a folder that contains the appropriate images (bmp/jpg) and the corresponding JSON file. The name of the JSON file must match the name of the application.

Important remark: All images are shared between applications. This means that images can be reused across applications but must have different names in each application folder.

In the demo SD card, there are 3 folders:

-  fogger
-  fogger_bmp
-  haut

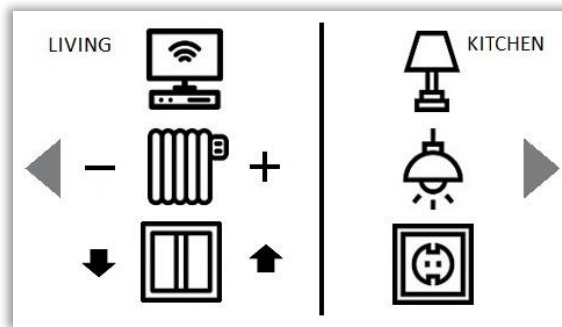
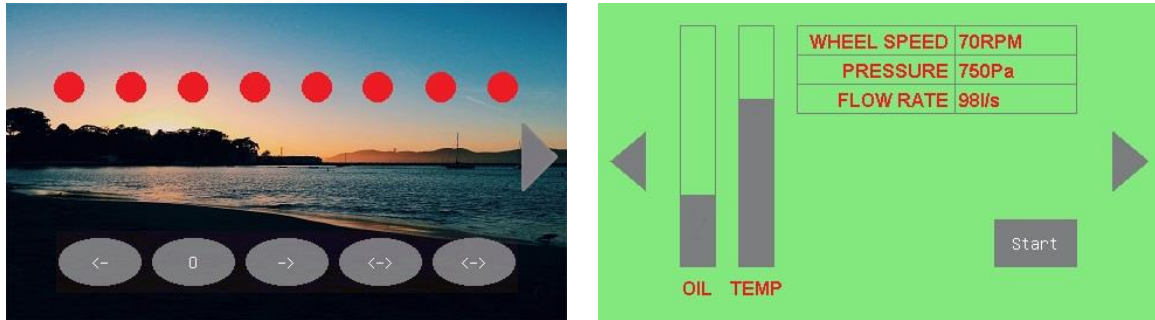
Fogger_bmp: This folder contains a demo layout of a *fogger application* that works autonomously. A fogger is a device that cleans the air at regular intervals. The demo includes screens for such an imaginary device. This folder contains the JSON file and the necessary bitmaps. Only bitmaps are used in this case for very fast interaction.

You can enter a PIN code for the first time, which will be saved if you press the ENTER button. Then, a menu will appear where you can access 'Device Info' and change the PIN code using the 'DEVICE PIN' button. On the main page, you will see a clock, which only works once it is set for the first time using the 'Set Clock' option.

Fogger: This is the same application, but some of the bitmaps have been replaced by textual widgets to create a more interactive interface, though this makes it slightly slower. Since some of the bitmaps are reused from the first application, you must load 'Fogger_bmp' first for this

to work. When the text is defined by a textual widget and a selected font, it can also be changed using one of the commands.

Haut: This is the *home automation application*. It contains three demo screens. The first screen is a signaling device for vans and trucks (automotive). The second screen displays progress bars and real-time data (industrial). The third screen shows a home automation system where you can switch lamps and plugs on/off, control a heating device, and operate a roller shutter. The last screen is an info page. It works together with the Python script included on the SD card.



1. Connect a 3V3 FTDI cable on the user application board connector
2. Setup the right COM ports in the Python script `demo_application.py`
3. Run the Python script with the following command:

```
C:\windows\System32>python demo_application.py cap
```

The option 'cap' indicates that a capacitive touch display is used. This option selects different COM port settings when both touch panel types are used on the same PC.

8.6. Set clock

This option is used to set the internal Real-Time Clock (RTC). You can set the year, month, date, and hour. The clock is used to drive widgets when they are linked to a part of the clock. Note that the clock must be set once before it is activated, and a 3V battery (type CR1220) must be installed to maintain the time when the power is disconnected.

8.7. Firmware updates

When firmware updates are available, they come in a signed and encrypted binary package. An internal bootloader copies the image from the SD card to the internal flash memory, performs a validation step, and swaps the old image with the new one. Once the image is accepted, it will be loaded after an automatic reboot. Wait at least 30 seconds after the image has been copied to ensure that the new one is loaded correctly. If the image is not accepted or the procedure is interrupted, the old image will be restored. Downgrading the firmware is not possible. On the SD card, there are test upgrade files containing the same firmware that was programmed during manufacturing.

Note: Check always the image version in the info menu

9. Json layout format specification

The JSON defines the layout of the application. When an application is loaded and active, it can be extracted by entering the command 'Dump current layout to JSON' in the user console over the USB port. The JSON follows a specific format to define the layouts.

The widgets are grouped into arrays by category. For instance, there is a category called 'buttons' that contains all the details for each button. Note that some properties are mandatory, such as 'backcolor' and 'name'. These are indicated in the table as mandatory.

```
{
  "buttons": [
    {
      "xloc": 72,
      "yloc": 80,
      "backcolor": 0,
      "background": "image",
      "fontcolor": 65535,
      "height": 17,
      "image": "s2.bmp",
      "name": "B0",
      "width": 15
    },
    ...
  ],
  "functions": [...],
  "pages": [...],
  "pincodes": [...]
}
```

The 'pages' category contains the properties for each page, along with all widgets that are part of the page, referenced by their name. This is a separate array where the order defines the visible layer level. For example:

```
...
"widgets": [
  [
    "B0",
    ...
    "unlockkeypad"
  ],
  [
    "popup"
  ]
]
```

```

    ],
    [
      "ok",
      "pincodesaved",
      "return",
      "wrongpincode"
    ]
  ],
  ...

```

This means that widgets like 'B0', etc., are visible on top of the 'popup' widget, and the 'popup' widget is visible on top of 'ok' and other widgets. Certain widgets can be reused across different pages (such as an OK button), but if they are not referenced by a page, they will never be rendered on the screen.

In the 'pincodes' category, there is a subcategory called 'widgets' that defines the widgets responsible for forming the entered PIN code. The order in the array is important, as it determines how the PIN code is interpreted correctly.

Check the examples from the demo applications to see the correct format.

10. Certifications

TBD

11. Order codes

Order code	Comment	Moq
HMC-20-C	Human machine controller PCB with capacitive touch panel	5
HMC-20-R	Human machine controller PCB with resistive touch panel	5
UAB-23	User application board with schematic and demo source code	5
HMC-CASE-3D38	Case to mount both PCB's 3D printed, height 38mm	1
HMC-CASE-38	Case to mount both PCB's with mold, height 38mm	500

Send a mail to info@saleconix.be for quotes, delivery times or more info.

HMC-20 Datasheet

Declaration

This documentation is the original work and copyrighted property of Saleconix (“SCX”). Reproduction in whole or in part must obtain the written approval of SCX and give clear acknowledgement to the copyright owner.

The information furnished by SCX is believed to be accurate and reliable. SCX reserves the right to make changes in circuit design and/or specifications at any time without notice. SCX does not assume any responsibility and liability for its use, nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SCX. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.

Third party licenses may be required to implement the solution/product. Customers shall be solely responsible to obtain all appropriately required third party licenses. SCX shall not be liable for any license fee or royalty due in respect of any required third-party license. SCX shall have no warranty, indemnity or other obligations with respect to matters covered under any third-party license.