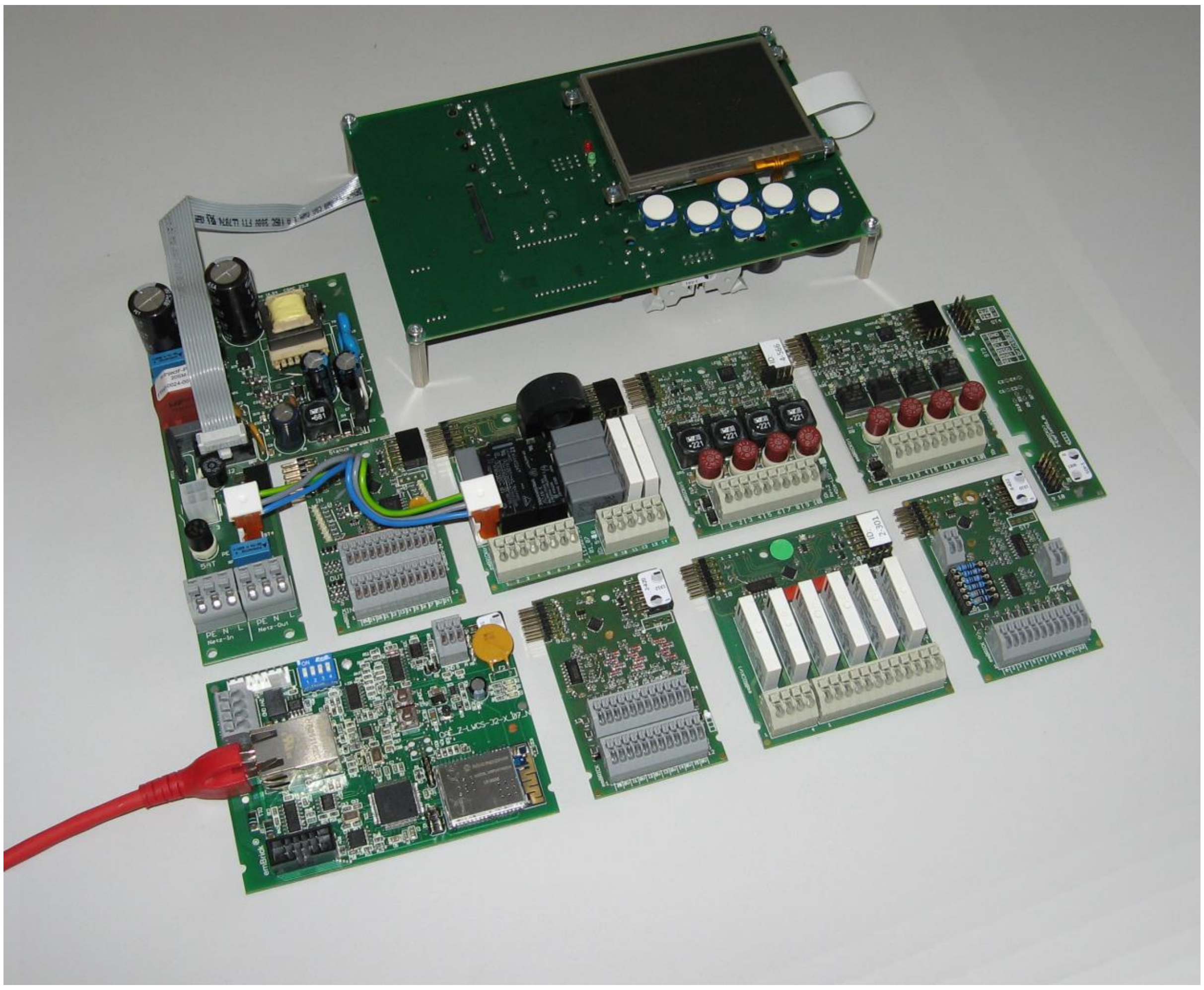
emBRICK

emBRICK® - EPC

Embedded Patch-board Controller



System Manual

Rev. 11

emBRICK® is an open-source project, developed and supported by

IMACS GmbH
Alfred-Nobel-Straße 2
D – 55411 Bingen am Rhein
www.imacs-gmbh.com
www.embrick.de

support@embrick.de
Hotline: +49 (0) 7154 80 83 - 15

You can live for yourself today
or help build tomorrow - for everyone
(Ronan Harris)

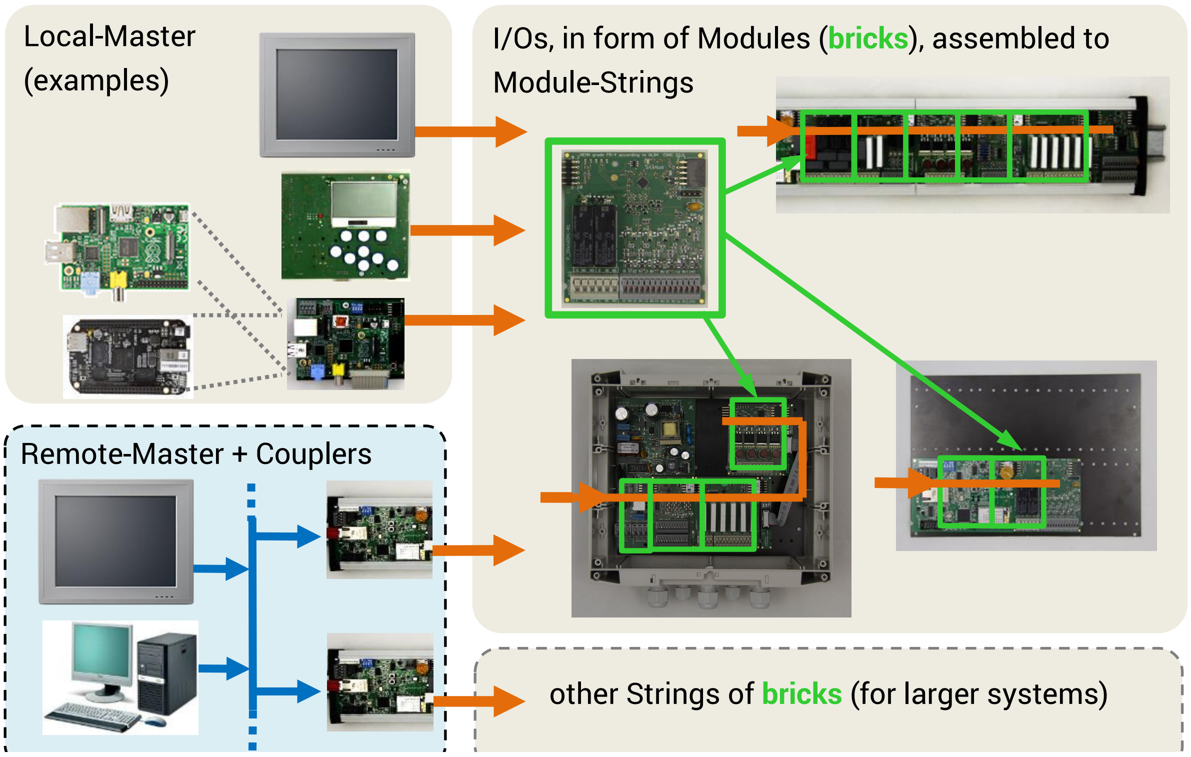
Content

1. The emBRICK® Mission

The mission of *emBRICK®* is an **open** and **free** I/O system to ...

build **compact** and **industrial suited** electronic **control systems** by **assembling** small **existing/own** embedded **boards** (**bricks**) ...

... via a SPI-based **local interface** and optional **remote buses** (LAN, WLAN, CAN, RSxxx, ...). We call this new class of controllers simple **EPC** (= **E**mbedded **P**atch-board **C**ontroller).


Local-Master
(examples)
I/Os, in form of Modules (bricks), assembled to Module-Strings
Remote-Master + Couplers
other Strings of bricks (for larger systems)

emBRICK® combines in a perfect way the **cost-efficient** and **tailored** characteristics of a dedicated embedded system with the **ready to use** and **flexibility** of a PLC system.

To ensure a high acceptance, it is an open and free system. I.e. besides buying existing devices, everyone can develop his own components to realize easily his individually tailored, cost-efficient and industrial-suited measure and control system.

1.1 Typical Applications

- Small, medium and large size **measure and control systems**
- **Sectoral purpose**, with direct sensor/actor interface
- **Autonomous single box** control solutions i.e. with HMI and communication interfaces
- **Rapid hardware prototyping** system for control and measuring applications
- **PLC replacement** (i.e. with a Soft-PLC, IPC or an embedded controller)
- Medium and large size **distributed IO-systems** (i.e. building automation)
- Physical front-end for **IoT** (Internet of Things)

For more details see *Product_Catalogue* and *Application_Manual*.

1.2 Basic Characteristics

- **free** - also for commercial use in own appliances (for pure EMS with a icense fee)
- **open** - supplying reference schematics, protocol source code, samples and starter kits
- **adaptable** to all systems, using common, low cost standard µCs/components
- **half ... third price** compared to common control systems (complete system view)
- scalable local and remote topologies, **1 ... >1000 I/Os**, up to **1ms update**, deterministic
- **low own power** consumption, average 50mW/slave module in operation (outputs inactive)
- global and sector specific modules for **direct connection** of various **sensors and actors**
- **easy installation**, no configuration necessary, simple plug modules together and use
- works with / programmable by **various established**, well known **platforms / languages**

1.3 Available Hardware Products

Beside own developments, currently the following components are available from IMACS:

Slave-Modules	> 50 different modules for the sectors: General Purpose, Building Automation, Process Control (Safety, Medical/Analytics planed)
Master boards	Core: Cortex-M3/4, ARM9/11, PIC24/32; HMI: 128x64 ... WVGA
Adaption boards	for LAN, WLAN, CAN, RSxxx, Raspberry Pi, Beaglebone Black
Appliances / Enclosures	ready Single Box Controller for and DIN-rail and wall mounting
Starterkits	for MSVC, CODESYS, Raspberry Pi, Beaglebone Black

1.4 Available Host Platforms, Connectivity

emBRICK® can be adapted to all platforms with almost every footprint/performance. For master units, currently the following system implementations are available (others planed):

Computer platforms	PC, Embedded-PC, Module-PC, Raspberry Pi, Beaglebone Black
µController platforms	ARM-Ax, ARM-Cortex-Mx, Microchip PIC24 / PIC32
Host Interfaces	Ethernet, CAN, RS232, RS485
Wireless Interfaces	WLAN

1.5 Available Programming Platforms

emBRICK® can be programmed by various systems, languages and IDEs (integrated development interface). Currently for master units the following systems are available (others planed):

OS / RTOS	Windows, Linux, FreeRTOS, proprietary
Programming languages	C, C++, IEC61131, Model-based (by implementing UML)
Model-based / Soft-PLC	CODESYS, radCASE, Enterprise Architect
C/C++ IDEs	MSVC, Cocox (GCC), MPLab (Microchip), Geany (Raspberry Pi), every other C/C++ IDE

2. Introduction

2.1 About this Manual

This manual contains basic system, architectural and topological information of *emBRICK®* and its communication technology *brickBUS®* with the focus on planners and users.

2.2 References / Manual Overview

For *emBRICK®* and *brickBUS®* the following documents are available. Before reading this document it is recommended to read them in the given order:

System Manual....................	(*embrick_System-Manual_#.pdf*) ... the basic system manual that contains the idea, the intention and the basic technical concept of *emBRICK®/ brickBUS®* like mechanics, electronics and communication protocol. It includes the glossary for all other documents.
Application Examples...........	(*emBRICK_Application-Examples_#.pdf*) ... overview of typical *emBRICK®* device configurations and sample constellations for different industrial applications. It gives an idea how to use *emBRICK®* as an alternative to a normal PLC or an individual PCB / embedded system.
Product Catalogue	(*emBRICK_Product-Catalogue_#.pdf*) ... contains the overviews and detailed datasheets of all IMACS-available *emBRICK®* components and products. This includes electrical and mechanical characteristics, terminal assignment and notes about their usage.
Programmers Manual	(*emBRICK_Programmers-Manual_#.pdf*) ... is the manual for application software programmers when using established programing systems like Embedded-IDEs, Soft-PLCs, CASE-Tools but also native C/C++-coding.
FAQ Manual..........................	(*emBRICK_FAQ-Manual_#.pdf*) ... contains answers to the most frequently asked questions about *emBRICK*® and its usage.
Developers Manual...............	is the manual for system developers, who like to create their own slave modules or master adaptions. It includes all technical details specifications of *brickBUS®* and also sample schematics and code samples of the software stacks. This document is only available on request from IMACS GmbH and needs the agreement on the *emBRICK®* free license conditions. Please contact support@embrick.de.

2.3 Homepage

The official website is www.embrick.de (coming soon). It will contain the upper named manuals, detailed datasheets and drawings, third party components, distributors, FAQs, schematics and source code, license condition for EMS.

2.4 Forum

http://forum.embrick.de

2.5 Roadmap

Currently the following products/enhancements are planed (partners are welcome):

- Wireless Connections
- PoE coupled master for more compact building solutions
- emBRICK-LE, a low energy version with different sleep-mode
- additional mechanical module formats like a box to clip on a DIN-rail

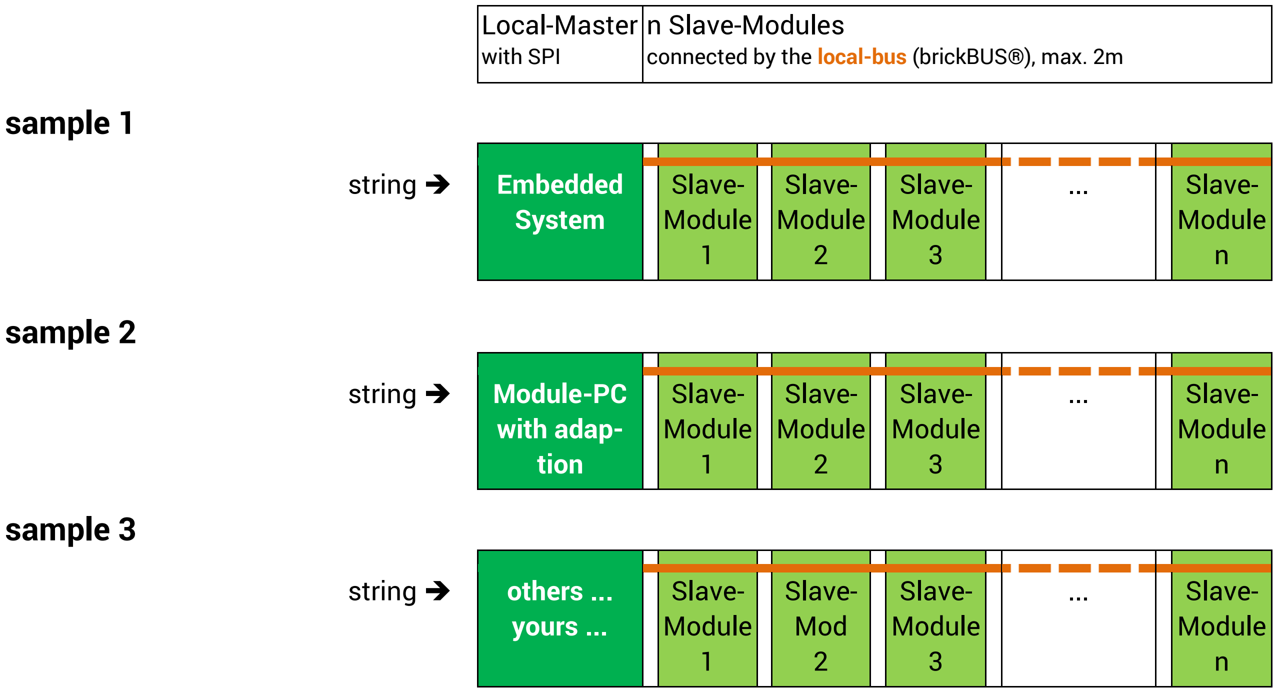
3. Concept and Topology

emBRICK® is a single master, multiple slave I/O-system. It comes with two different topologies, mostly depending on the number of I/Os and the physical size of the system.

- **Local-Mode** operation (1 local-master, n slaves)
- **Remote-Mode** operation (1 remote-master, m coupling-masters each with n slave-modules)

3.1 Local Mode Operation

This operation mode is used for small and local systems, mostly complete deceives. It contains of one ***string*** with one L***ocal-Master*** and multiple ***Slave-Modules***. The *local-master* here is mostly the (only) main CPU of the device that controls also HMI, external communication and application.



Each slave module has a male and a female bus connectors to plug them together in a row.

The **local-bus** (the kernel of *emBRICK®*) is the *brickBUS®*, that defines the physics and the data protocol. It is designed very simple to make it possible for every µC to be a communication partner. Physically it is a combined data/power bus, that contains a SPI-bus, a daisy-chain, 3.3V and optional 24V (for sensor/actor supply). The timing of the SPI is very undemanding and can also be realized with a pseudo SPI by "bit-banging".

In the standard version of the *brickBUS®* is realized by a 10-pin, 2 row, 2.54mm header connector. Reduced types and wireless are planned. The local-master is the only "intelligent" part (keeping the application software); it controls up to 32 local-bus slaves and supplies all slaves-modules with power. For higher power applications an additional supply of the single modules is possible.

Thus a simple module to module connection via a parallel SPI, the local-bus is a self-configuring bus and supports a random access. For this, during the initial sequence each module will be assigned a unique address by a daisy-chain. During normal operation this daisy-chain is "only" used to force a fast security "output-down".

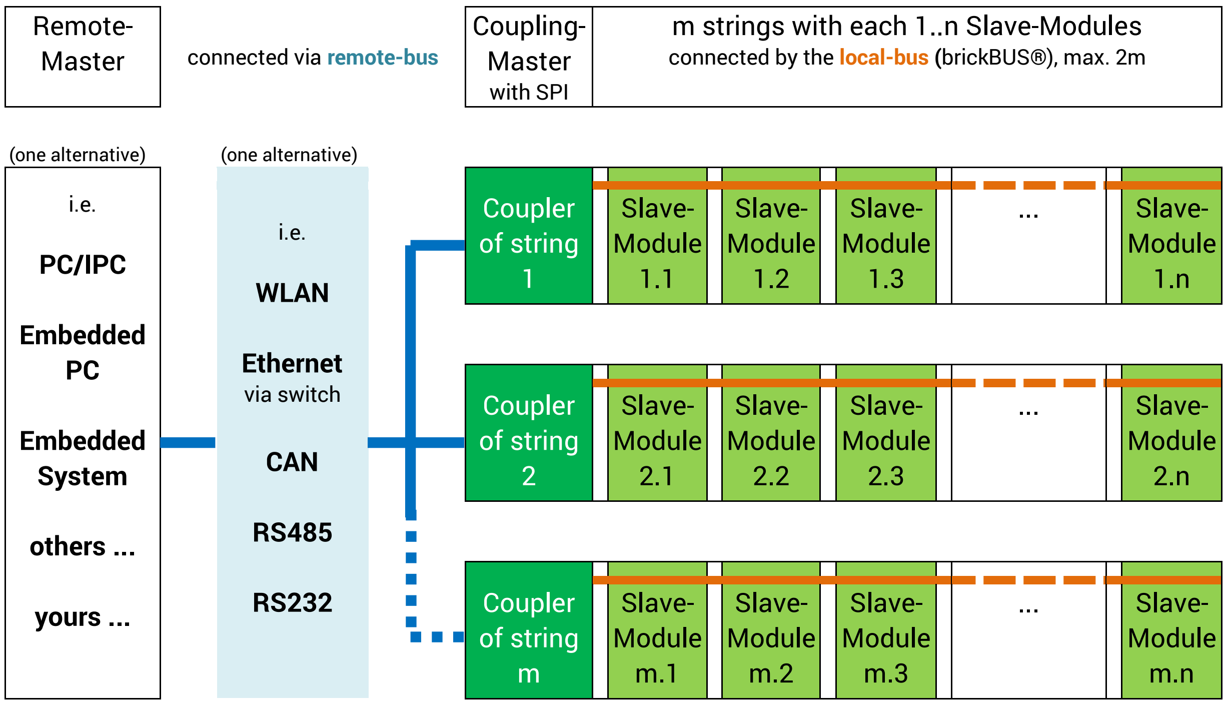
3.2 Remote Mode Operation

This operation mode is used for mid and large size systems i.e. with distributed devices/nodes. Other applications are the adaption of hosts that cannot support a real or pseudo SPI.

The remote-mode is an expansion of the local-mode in the way that it consists (in most cases) of multiple strings located over larger distances.

Contrary to the local operation the intelligent local-masters will be replaced by "non-intelligent" ***Coupling-Masters***. These coupling-masters will be controlled over the ***remote-bus*** by a ***Remote-Master*** who keeps the application intelligence of the whole system.

For this the remote-bus is based on various established communication/interface standards.



The remote-master is the only "intelligent" part (keeping the application software), controls a random number (limited by performance of remote-master and selected remote bus) of strings with up to 32 local-bus slave-modules.

Because the established communication methods used for the remote-bus do not support a well-defined order automatically, the coupling-masters of the remote-bus must have their own address that has to be assigned manually.

4. Technical Data

In future *emBRICK®* will support different versions concerning board format/size, coupling (electrical, data technical) and performance/speed. To separate this, the following three categories are defined:

BBF#	= brickBUS format: size and kind of usage
BBC#	= brickBUS coupling: type of connector and electrical layer
BBP#	= brickBUS performance: speed and communication

The digit "#" after the three letters indicates the respective type/level. Actually only the first level of all categories is realized, e.g. BBF1, BBC1, BBP1 (short form: "BB_{FCP} 1-1-1" or simple "1-1-1"). All documents will reference to these categories. The exact specifications of these categories and their types/levels are defined in the appendix of this document.

4.1 Local-Bus

Architecture	single master, multiple slaves
Bus topology	chain
Communication	SPI, synchronous, full duplex
Addressing	during initialization: via daisy chain; in running: via random access by self-configured addresses
Number of slaves	1...32 / string
Transportation, connectors	conform to BBTx, see 6.1
Performance, communication	conform to BBPx, see 0

For slave modules, actual Microchip PIC16/24 is used. Others (i.e. Cortex-M0) are planned.

4.2 Remote-Bus

Architecture	single master, multiple slaves
Bus topology	tree, star
Communication	several (Ethernet, WLAN, CAN, RS485, RS232, ...)
Addressing	manual during setup
Number of strings	no bus specific limit, depends on remote-master performance
Transportation, connectors	conform to BBTx, see 6.1
Performance, communication	conform to BBPx, see 0

4.3 Usability

Diagnostics	via LED on every slave-module see 6.4
Type/pricing of core unit	Slave: actually Microchip PIC16xxx / 24xxx, approx. 1€ / pc
Mounting	various possibilities, see 5

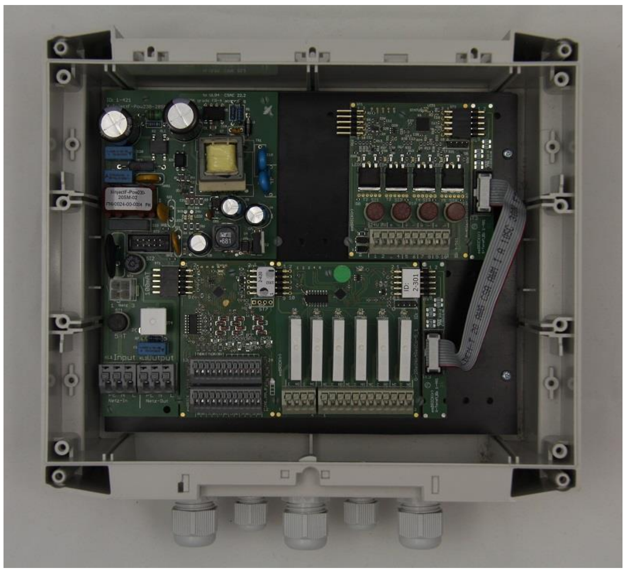
5. Mechanical Designs

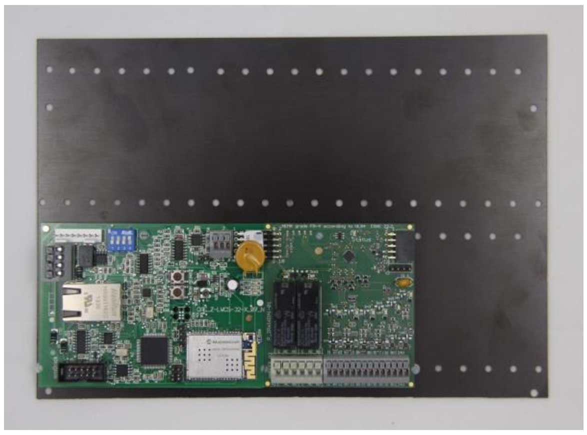
With emBRICK, the below listed ways of mounting/enclosure are possible:

"Patch-Board"	The modules will be plugged together in one or more rows and mounted flat with spacers onto a special carrier board with a defined grid of drills.
"Rail-Flat"	The modules will be shifted flat into a DIN-rail enclosure and fixed open frame onto a DIN-rail. A crystal-clear cover can be added as a touch protection.
"Rail-Box"	The modules (bigger format) will be mounted into a complete closed DIN-rail enclosure and contacted via DIN-rail bus connectors.
"Rack-Open" (planed)	The modules will be stacked and connected via a kind of backplane. In this case only one connector of the slave modules is used. Therefore the initialization works with a special mechanism.

5.1 Patch-Board Mounting

This mounting is typical for smaller and compact devices.

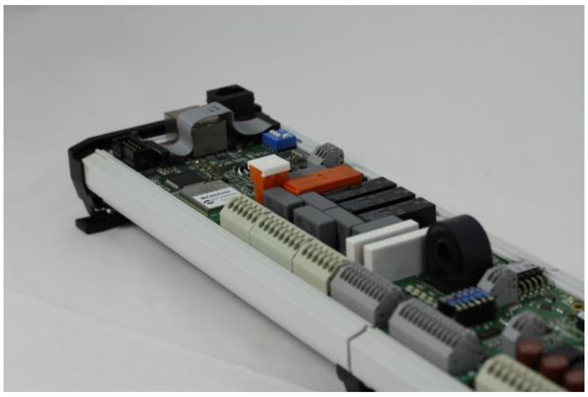


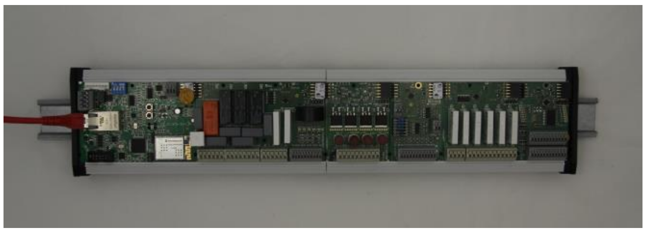


Format type	BBF1
Connection type	BBC1
Mounting	via screws, threated bolts or spacing bolts (metal, plastic) on a carrier board (different materials) with a hole grid.

5.2 Rail-Flat Mounting

This mounting type is used in bigger cabinet and wide area distributed systems like buildings.





Format type	BBF1
Connection type	BBC1
Mounting	by insertion into DIN-rail enclosures. For BBF1 use the DIN-rail enclosure with a board wide of 72mm, supplied by Bopla, OKW and Phoenix Contact, Elbag (series Mini Modular)

6. Appendix

To fulfill the different requirements concerning electrical/mechanical characteristics and performance/safety, the basic features of emBRICK are classified in *brickBUS*® types (BBx#) with different categories.

6.1 Board Format Types (BBF#)

The board format defines the mechanical conditions of a slave-/master-master module to guarantee their interoperability but also an easy and safe way of usage for the users.

Below the global features are listetd. For details please refer the developer manual "emBRICK_Developer.PDF"

BBF1:

Possible mounting	Patch-Board, Rail-Flat
Characteristic	open frame with optional angular cover plate

BBF2:

Possible mounting	Rail-Box
Characteristic	proteced box units to snap on to DIN-rail

BBF3:

Possible mounting	Front-Panel, 19"-rack
Characteristic	open frame with standard SubD-connector

BBF11:

Enhancement of the BBC1 with same outline but high speed communiaction

6.2 Bus Coupling Types (BBC#)

The bus coupling type defines the electrical boundary conditions of a slave-/master-master module to guarantee their interoperability but also an easy and safe way of usage for the users. Independent of the connection the same basic protocol is used for compatibility. Either a passive or an active adapter allows bridging. Beside the brickBUS connector, also other/separate connections/busses are used, i.e. to distribute 24Vdc or 230Vac power supply via different kind of cable/connectors.

Below the global features are listetd. For details please refer the developer manual "emBRICK_Developer.PDF"

BBC1:

Standard connection via 10pol. Pin connector support size bus and power for max. 2m string

BBC2: (planed)

Slow speed version of BBC1 with different connectors supports max. 10m string size.

BBC11: (under develpment)

High speed version of BBC1 to support the BBP11

6.3 Bus Performance Types (BBP#)

According the capability of the SPI interface of the slave module controllers, the slave modules are separated in different *speed classes* BBP# (= **b**rick**B**US **p**erformance). This allows e.g. to use also low cost µC without a SPI-FIFO and/or longer interrupt response time.

While initializing the bus, the master will automatically check the BBP type of each slave module and switch to the lowest BBP detected in the module-string. Therefore to reach a high BBP, all components in the string need to have at least the requested BBP.

The BBP of each module is printed in the identification table on the first page of the component description.

6.3.1 BBP1

Standard

bus clock / data rate	10...1,000 kBit/s
effective payload	≤ 40 kByte/s (approx. 320 kBit/s)
recommended update rate	20...1,000 Cycles / s (complete string)
error detection	1 Byte checksum / module
error correction	repetition of transfer

6.3.2 BBP2

Low speed

bus clock / data rate	10...100 kBit/s
effective payload	≤ 4 kByte/s (approx. 32 kBit/s)
recommended update rate	20...100 Cycles / s (complete string)
error detection	1 Byte checksum / module
error correction	repetition of transfer

6.3.3 BBP11 (under development)

High speed

Bus clock / data rate	1,000...25,000 kBit/s
effective payload	≤ 1,500 kByte/s (approx. 12,000 kBit/s)
recommended update rate	1000...10,000 Cycles / s (complete string)
error detection, correction	2 Byte checksum / module
error correction	repetition of transfer

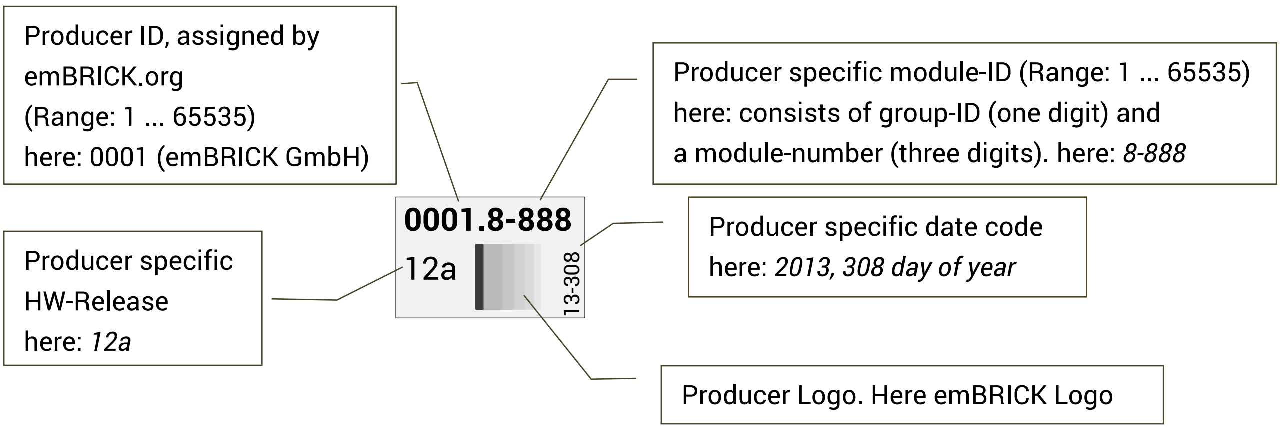
6.4 Bus State and Status-LED Blink Codes

A *brickBUS*® member (slave module) can have the following bus-states and signalize this with the corresponding LED-flash code:

uninitialized	Status-LED: continuous fast flashing approx. **5 Hz** = the slave module has not been initialized since last power on.
no_com	Status-LED: Morse code • **—** • = no running communication / communication breaks down. For >2s no messages received.
under_com	Status-LED: Morse code • • • = the slave is initialized and communication is running but not stable/to less messages (< 20 messages/s received)
disturbed_com	Status-LED: Morse code • • **—** = the slave is initialized and communication is running but to many error/wrong messages are received (> 1 com-error/s detected)
running	Status-LED: continuous flashing approx. **0.8 Hz** = correct operation with sufficient correct messages
defect	Status-LED: **no flashing** (continuous on or off lightning) = the module is defect. Operation of the BUS is no longer safe, please replace.

6.5 Component Labeling

The label of a component / module is placed on a bus connector and contains:


Producer ID, assigned by emBRICK.org (Range: 1 ... 65535) here: 0001 (emBRICK GmbH)
Producer specific module-ID (Range: 1 ... 65535) here: consists of group-ID (one digit) and a module-number (three digits). here: *8-888*
0001.8-888
12a
13-308
Producer specific date code here: *2013, 308 day of year*
Producer specific HW-Release here: *12a*
Producer Logo. Here emBRICK Logo

Note: M=2:1, real size of the label is 12.5mm x 8mm

6.6 Communication Cycle / Data Update Schema

The following schema describes the sequence of one data update cycle (BBP1) in a slave module, triggered by a bus-master communication.

1 wait for the sync gap (no communicated data for > $t_{SyncGap}$)
2 wait for a new message (header) from master
3 received header from master does not select this slave => continue with 1
(else - the slave is selected)
4 if exists: read **digital hardware inputs** and store into in-buffer
if exists: read **counter/capture hardware inputs** and store into in-buffer
5 data transfer with master, send in-buffer to master, get out-buffer from master
6 wait for the sync gap (no communicated data for > $t_{SyncGap}$)
7 write out-buffer content to **digital outputs**
write out-buffer content to **analog/PWM outputs**
8 if exists: read/convert **analog inputs** (one time of all channels) and store into in-buffer
9 continue with 2

Note:

The AD-conversion of the BBP1 slave-modules (for 8 ADC-channels) needs ...

- for slave-module with PIC16LF:	2.5ms (max. 3ms)
- for slave module with PIC24FJ:	250µs (max. 300µs)

Ensure in the master timing, that the time for the steps 8, 9, 1, 2, 3 in sequence exceeds this period.

The conversion timing of BBP11 modules will be <10µs (for 8 ADC-channels).

7. Glossary

Slave-Module (= Brick)	is a small I/O-board, working as the physical/electrical frontend of the *emBRICK®* system. It contains of a local-bus interface with typical 2 connectors (to be a member of a chain). A brick includes no application-functionality, e.g. it is only an interface with communication capabilities.
Local-Master	controls the local-bus and slave-modules and typically contains the functionality of the application.
Local-Bus	is the communication line that connects the slave modules together with a bus-master.
String	is the sum of the local-master with its slave-modules connected together.
Coupling-Master	is a special kind of a bus-master that couples a string of slave-modules (of a local-bus) to a more or less distant, non *brickBUS®* master unit (that is called the remote-master). The coupling-master is completely controlled by the remote-master; therefore it does not contain any application functionality.
Remote-Master	is a separate intelligent unit (i.e. a PC/IPC) that controls one or more strings via a coupling-master.
Remote-Bus	is the bus that connects the coupling-masters (of several strings) together with the remote-master. This bus connection can be realized by i.e. Ethernet, WLAN, CAN, RSxxx ore others established communication standards.
Bus-Master	A generic term for local-master, coupling-master and remote-master.

8. System Partners

company	Product	Producer	Appl. Develop	Tools	Solution.Dev.	Distri.
		(Prod-ID)	Applicationeer		Solutionieer	
IMACS	components	001	✓	✓	✓	✓
emBRICK	components	002	✓	✓	✓	✓
IMACS	radCASE		✓	✓	✓	✓
RST	GAMMA			✓		
Protos	eTrice			✓		
EuroSystems	components	101				
Laser & Co.	components	102				
Laser & Co.	SiSy			✓		
logi.cals	logi.CAD		✓	✓	✓	
IT-Cosmos			✓		✓	
emtrion	components	103				
coming			✓		✓	
3S	CODESYS			✓		
National Inst.	LabVIEW			✓		
Conrad	components					✓