



## **AIM**

### **Deliverable D3.1.1.2**

#### **EMD Interim prototype – Accompanying report**

Deliverable nature:	Prototype (P)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31 May 2009
Actual delivery date:	17 March 2009 (Accompanying report delivered on 15 June 2009)
Version:	1.0

---

#### ***Abstract***

This Deliverable is a prototype by nature and the intention of this document is report it as delivered and refer the deliverable D 3.1.1.1, EMD Design and specification report, as the document that should be consulted for more details about the EMD Interim Prototype.

---

---

### Disclaimer

---

This document contains material, which is the copyright of certain AIM consortium parties, and may not be reproduced or copied without permission.

All AIM consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the AIM consortium as a whole, nor a certain party of the AIM consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

### **Impressum**

[Full project title] A novel architecture for modelling, virtualising and managing the energy consumption of household appliances

[Short project title] AIM

[Number and title of work-package] WP3: Design of the power management architecture

[Document title] EMD Interim prototype – Accompanying report

[Editor: Name, company] Maria Barros, Eurescom GmbH

[Work-package leader: Name, company] Spyridon Tompros, Keletron

[Estimation of PM spent on the Deliverable] 42 MM (including development of the interim EMD prototype and creation of D3.1.1.1)

### **Copyright notice**

© 2008/2009 Participants in project AIM

## Table of Contents

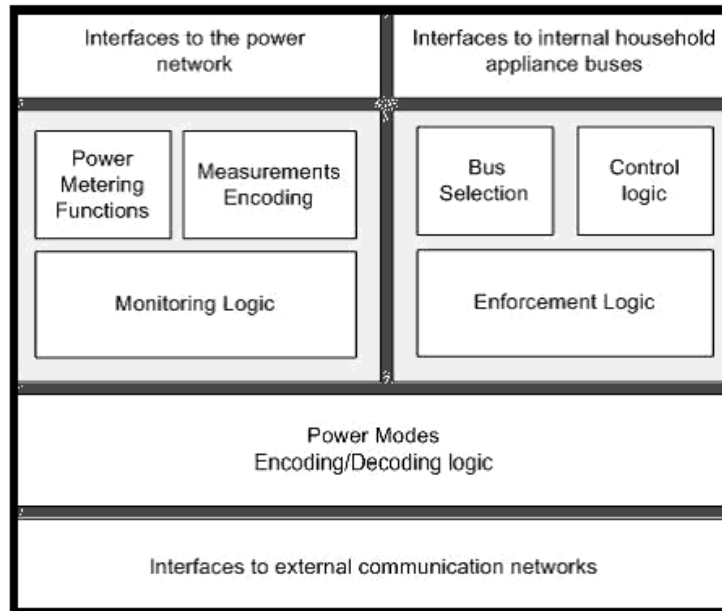
Table of Contents .....	2
1 Introduction .....	4
2 EMD generic architecture .....	5
3 The EMD hardware implementation .....	6
4 The EMD firmware roadmap .....	10

# **1 Introduction**

This document doesn't intend to describe the EMD Prototype but just present it as a finished deliverable. For more details the deliverable D 3.1.1.1, EMD Design and specification report, should be consulted.

## 2 EMD generic architecture

The EMD shall have a unified architecture, which will feature generic interfaces towards the household appliances, the power network and the home network. Due to its generic architecture, the system can be realized in differing forms, i.e. standalone external box or internal module. Concerning its buildings blocks, the system offers three generic-purpose interfaces; one towards home communications networks, one towards the mains power network and one for connecting to internal digital control buses of household appliances.



**Figure 1: Internal architecture of the Energy Management Device (EMD)**

With these three general-purpose interfaces, the system will be able to integrate with virtually any network environment or household appliance and will provide two types of power management logic:

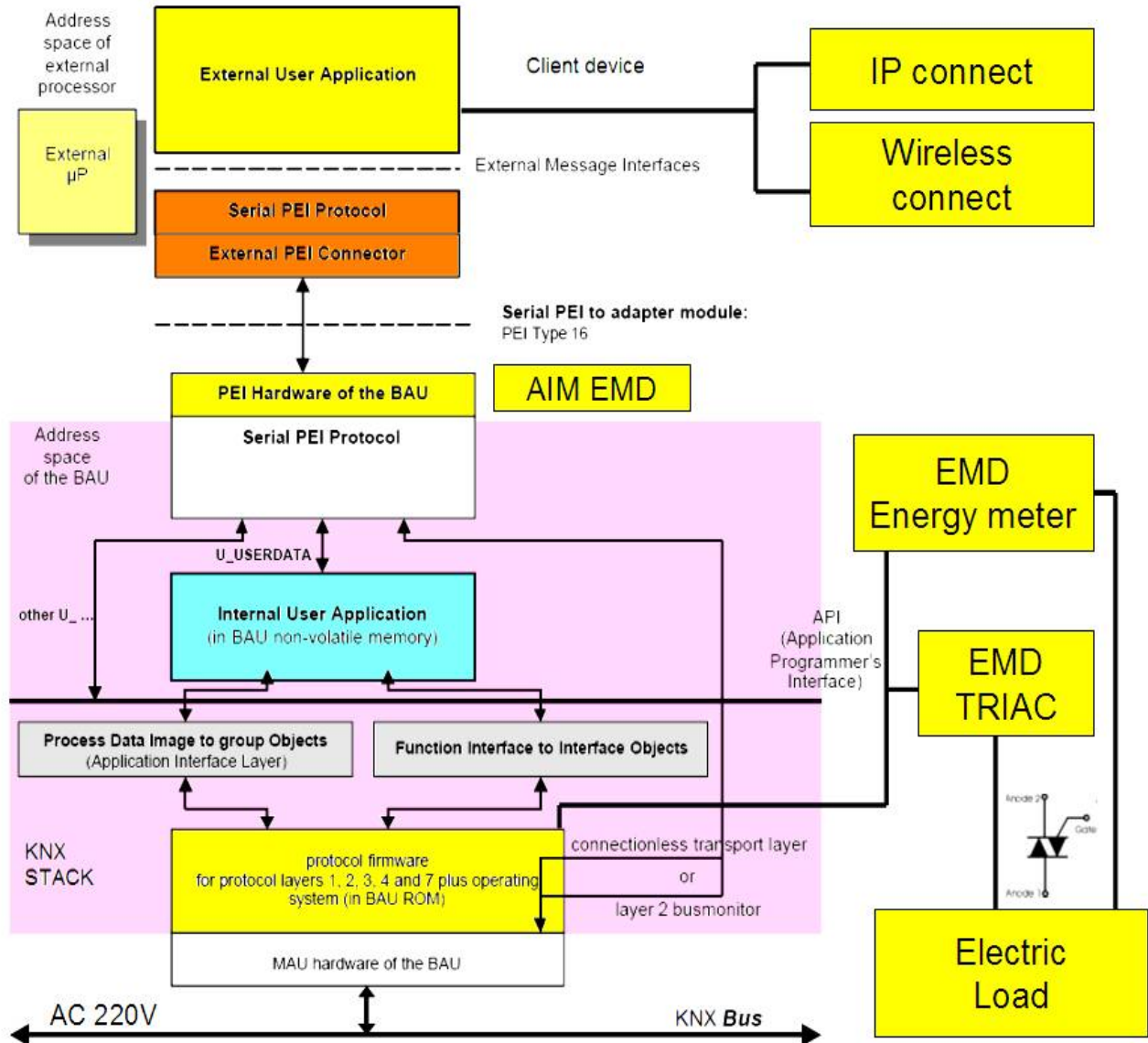
- **Energy monitoring:** power metering functions that are applied to power electronics of the household appliances, an encoding logic that turns measurement results into digital values and a monitoring logic that buffers the obtained measurements following user configuration commands.
- **Energy control:** control logic, taking into account the user commands as they have been decoded and submitted by the enforcement logic of a given appliance. Based on this information, the system performs selection of one of the several external interfaces to the household appliance.

EMDs shall be accessible either locally, through the AIM gateway or via external IP operator networks.

Only 3 heading levels appear in the table of contents.

### 3 The EMD hardware implementation

The architecture of AIM EMD is depicted bellow.



**Figure 1: EMD Hardware implementation diagram**

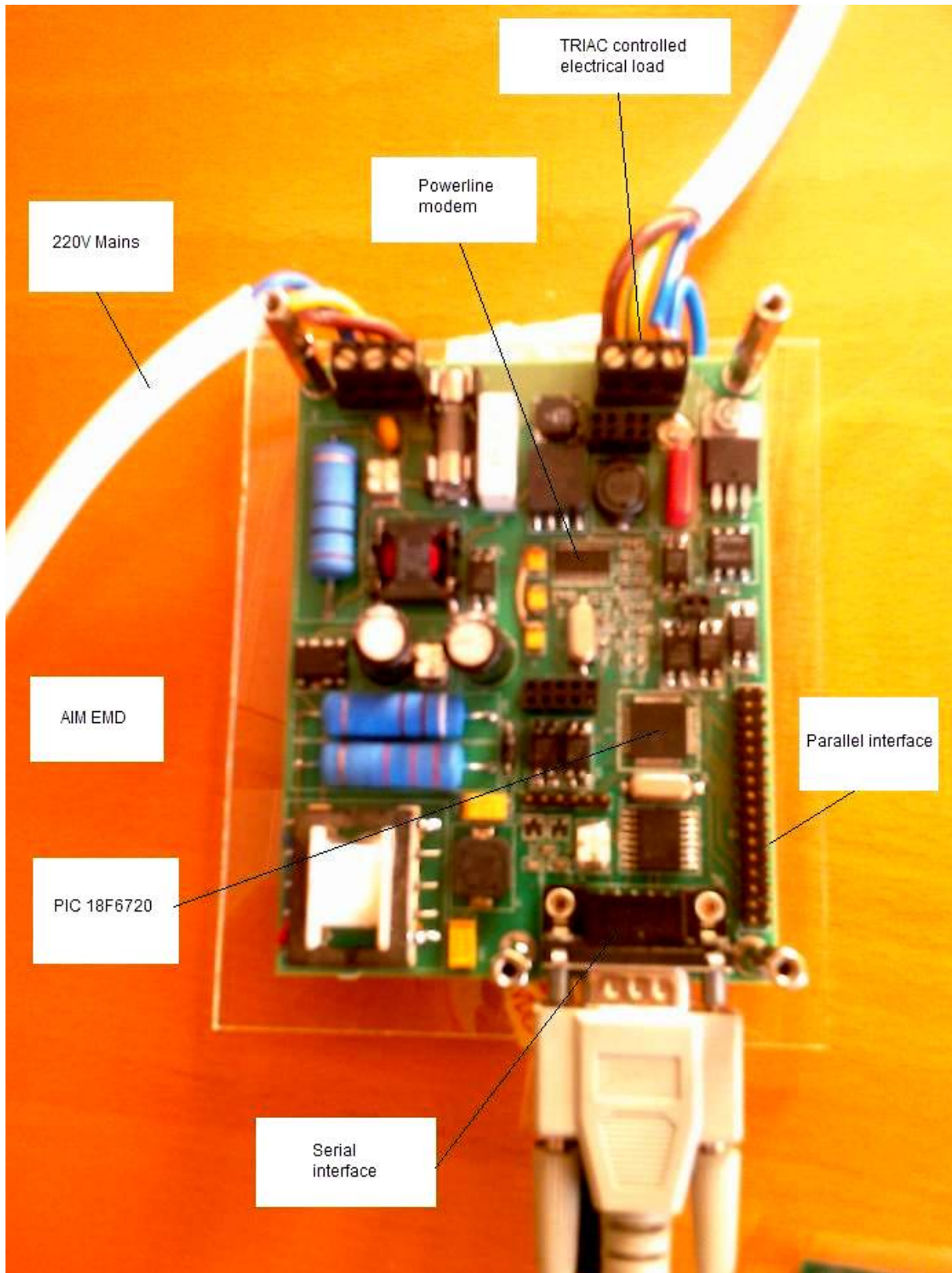
\* MAU (Mount Access Unit) in this case is the ST power line modem.

The EMD will possess two basic communication ports: one will be a USART channel (RS232) and the other will be constituted through a power-net link. The USART will allow us to bind an EMD to external hardware modules, which provide services such as wireless communication services or deliver IP connectivity (External User Application). A similar option will be available with a USB interface to IP and wireless modules.

Inside the EMD, External Message interface commands will be mapped into an internal format which is called 'Internal Message Interface'. This is the command message language that the KNX layers use with each other. Moving commands up and down the stack layers and on the KNX bus would be useless unless we could add persistence and reasoning to them.

The AIM EMD will be equipped with 1Kbyte of EEPROM which is used to store persistent KNX data (e.g. KNX network address ID, network groups etc – Bus Access Unit's non volatile memory). The KNX EMD must be able to process the user data (coming from the client device) into Group object data (in other words, standardized commands, aimed at possibly many recipients on the KNX net of specific Functional Blocks) and interface objects, working as conceptual information conduits between KNX devices and sensors, linked to one another. This virtual linking phase will be automatically handled at the net setup phase; a process which must be also supported by the command parser of the AIM EMD.

The API (Application programmers interface) refers to a KNX specified set of code functions, which the EMD micro-processor must support into order to be able to process commands coming from the client device. Most of the API calls work quite similar to the ANSI C standard library.



**Figure 2: AIM EMD**

The EMD receives an energy reading on its controlled device through an EMD energy meter circuit based on the STPM01 chip. It is also able to control the energy power that reaches the controlled electric load by using a TRIAC circuit. There is a hardware extension interface on the EMD, which

makes it possible to connect this device to other electronic boards, delivering complementary communication protocol functionality (e.g. TCP/IP) or enhanced control interfaces (e.g. infrared).

## 4 The EMD firmware roadmap

The AIM EMD will be a hardware device constituting a composition of a power line modem and an embedded micro-processor, exporting a serial interface and possessing 1Kbyte EEPROM with 3Kbytes of RAM. The process of hardware integration will not be addressed in this short description. Nevertheless, we can focus on the firmware design challenges that this project presented.

The basic development steps that the AIM EMD encompassed are the following:

1. **Abstracted KNX stack design (layer by layer)**

The definition on each layer's functionality is available in detail in the KNX spec documents.

2. **Generic finite state machine for KNX stack layers design**

Each KNX stack layer is a layered finite state machine/parser. We use 4 of these structures, for the Data link, Network, Transport and Application layer, respectively.

3. **KNX internal message pump in conjunction with a finite state machine**

The Transport and Application layer need an event handler on-top of their Finite State Machine. Their defined behaviour is quite complex and dependent on the client device's functionality state (idle, working, receiving information, etc). This means that their state machines must be aware of the external functional environment, which is guaranteed by the implementation of an event handler – this event handler is executed in the transport layer of KNX.

4. **EMI/IMI parameter parser for each KNX layer**

There is a clearly defined interface for communicating with the KNX stack. When commands reach the stack from the outside by a PEI16 (application-wise it is a RS232 serial port), the commands must be coded in EMI (KNX External Message Interface) format. These commands are translated into IMI (KNX Internal Message Interface), which actually controls the information flow among the KNX stack layers.

5. **PEI16 protocol implementation**

The PEI16 protocol is defined both physically (electrical signals) as well as abstractly by the KNX specs. For further information, the reader should refer to the KNX.org KNX PEI specification. It is conceptually a set of control codes communicated over RS232 between the client device and the AIM EMD.

6. **EEPROM and KNX Virtual RAM interface engineering**

Since the KNX is an open standard, (hardware platform independent), no hardware specs are provided. Instead, for all hardware related functions, such as the ones that deal with the persistency of a KNX device, the specs define abstract functions. These must be translated into concrete ones by our KNX PL implementation. Thus we need to define EEPROM virtual access points for the particular micro-processor base we are using. There is an area in the RAM which is used to store KNX object RAM data – thus a Virtual RAM interface has been incorporated in the stack to access this area in a uniform and KNX standard compliant manner.

7. **KNX API implementation**

The KNX API is a set of functions that the KNX micro-processor must implement in order to be able to process KNX messages. It is the standard function library of the KNX micro-processor world, irrespective of hardware base chosen.

8. **KNX object attribute translator**

The object constituents of KNX device data are not just arbitrary framed byte collections. In KNX topologies, the KNX net is used to communicate information between logical entities, called functional blocks, which are atomic operators in a KNX system (like an actuator, a push button or a sensor). All these come in predefined types, with standardized inputs and outputs. This format must be parsed both as the broadcast (message formulation) and the reception end; which is what the object attribute translator does.

### **9. Implementation of KNX net setup message parser**

This is a message conduit, aimed mainly to ‘personalize’ the KNX device, by registering its identity. The messages that are processed interact with the BAU’s EEPROM, by reading/writing or altering its contents. This functionality is used mainly during the setup phase of the network and has to do with placing a device in programmable mode, one at a time, making it perceptible to receiving personalization commands like receiving its KNX individual and domain address. In AIM we had to develop a setup procedure that could be automatically completed – similar to plug and play.

### **10. Low level engineering to interface and host the AIM EMD BAU on a PIC18F6720**

This process involved all the low-level implementation details that had to go into engineering the KNX EMD system; selecting portions of the code to go into specific memory areas, and developing the ST7540 driver over the PIC18F6720. The ST7540 is not equipped with firmware; it must be controlled by the PIC186720 using real time gate-logic. (FSK keying modulation of data, error correction, frame check sequencing, etc.).

### **11. Testing and Simulation in vitro**

The firmware codebase is unit and simulation tested before being ported into a micro-processor. It has to be immune to buffer overruns, not use dynamic memory allocation – de-allocation techniques, and it has to be crash immune under any circumstance and KNX traffic load. Since the KNX stack is a real-time system, without the benefit of being hosted on a Real Time Operating System, it is critical to verify that all atomic operations are executing in proper time frames, compatible with the KNX stack communication speed requests.

### **12. Integration with Energy Meter and TRIAC**

The completion of the EMD development coincides with porting the firmware code into a PIC186720, and interfacing it to an energy metering and TRIAC soft-switch module. This compound module should be connected to a PL KNX net topology and be operated remotely by the AIM Gateway. Operationally-wise, the EMD’s auto setup mode should be verified. The KNX defines a set of test-cases to which our KNX implementation should rigorously comply. Add -on boards like Zigbee, IP or 802.11 will be tested additionally to the power line net.