All over the world there are many research centers that are conducting researches with use of particle accelerators. Thanks to various experiments we could better understand surrounding world. But, there are still a lot of unknowns to explore which science needs better instruments. One of these tools are measurement systems. Unfortunately currently used solutions do not provide sufficient performance to satisfy growing needs. This implies the search for new solutions. One of such solution is a modern uTCA architecture. In this document an open source project of a base card (AFC -AMC to FMC carrier board) based on this standard has been described. This card is equipped with two FMC connectors, which allow to connect wide variety of extension cards. In combination with a powerful FPGA device this card is an universal base circuit for variety of projects. Among the others it allows to implement algorithms which are collecting data from fast ADCs and to process these data. Moreover the applied uTCA architecture provides fast communication interfaces which allow to transmit collected data.
The EuCARD-2 Enhanced European Coordination for Accelerator Research & Development project is co-funded by the partners and the European Commission under Capacities 7th Framework Programme, Grant Agreement 312453.

This work is part of EuCARD-2 Work Package 1: Management and Communication (MANCOM).

Warsaw University of Technology
Faculty of Electronics and Information Technology
Institute of Electronic Systems

MicroTCA based platform for advanced particle accelerators diagnostics

Authors:
Bartłomiej Juszczyk, MSc
Grzegorz Kasprowicz, Ph.D

Warsaw, 2014
MicroTCA based platform for advanced particle accelerators diagnostics

All over the world there are many research centers that are conducting researches with use of particle accelerators. Thanks to various experiments we could better understand surrounding world. But, there are still a lot of unknowns to explore which science needs better instruments. One of these tools are measurement systems. Unfortunately currently used solutions do not provide sufficient performance to satisfy growing needs. This implies the search for new solutions. One of such solution is a modern uTCA architecture.

In this document an open source project of a base card (AFC - AMC to FMC carrier board) based on this standard has been described. This card is equipped with two FMC connectors, which allow to connect wide variety of extension cards. In combination with a powerful FPGA device this card is an universal base circuit for variety of projects. Among the others it allows to implement algorithms which are collecting data from fast ADCs and to process these data. Moreover the applied uTCA architecture provides fast communication interfaces which allow to transmit collected data.

One of important issues is synchronization of measurements in the time domain. For this purpose direct connections between AFC cards through a backplane board were used. In order to make synchronization possible, a firmware for the FPGA device for the AFC triggering card was prepared.

In that kind of systems besides data acquisition channel it is also important to provide board diagnostics and management. This role is carried out by a microcontroller located also on the AFC board. It provides compatibility with the uTCA standard as it also delivers diagnostics data and controls other integrated circuits. A description of an application implemented in this microcontroller memory was also contained in this publication.

In order to provide tool for verification of the correctness of the printed circuit board and parts assembly it was also designed a specialized FMC module for testing purposes. Integrated circuits located on this board allow to verify voltage levels, electrical connections and fast links for data transmission.

The last chapter describes tests which verify the operation of this board, firmware and software. There were verified both basic parameters as also operation with external FMC modules. Moreover the microcontroller firmware was also examined.
# Contents

| Table of contents |  
|-------------------|---
| List of figures   |  
| List of tables    |  

## 1 The particle accelerators

1.1 Basics ........................................ 13
1.2 Meaning for the mankind ...................... 15
1.3 Methods for beam intensity and position measurements ........................................ 17

## 2 Integrated systems for data acquisition purposes

2.1 Architectures of currently used systems .......... 19
2.2 Available uTCA carriers .......................... 24

## 3 Concept of the system

3.1 The AFC carrier card ............................. 31
   3.1.1 The FPGA device selection .................. 33
   3.1.2 Diagnostic and management device .......... 34
   3.1.3 The power supply block ..................... 35
   3.1.4 The clock distribution section ............. 37
   3.1.5 Fast serial links for data transmission .... 38
   3.1.6 A card diagnostics ........................ 38
3.2 The FPGA firmware ................................ 39
   3.2.1 System triggers and interlocks ............. 40
3.3 The microcontroller firmware design ............ 42
   3.3.1 Intelligent platform management interface .... 42
   3.3.2 The diagnostic service .................... 47
   3.3.3 Miscellaneous tasks ....................... 51
3.4 The FMC HPC tester ............................ 52

---

5
4 The design implementation

4.1 The AFC card hardware design

4.1.1 The FPGA circuit

4.1.2 The microcontroller application

4.1.3 Power supply

4.1.4 The clock distribution unit

4.1.5 Fast serial links for data transmission

4.1.6 Diagnostic sensors

4.1.7 The AFC card layout design

4.2 Triggering cards firmware design

4.3 The microcontroller firmware implementation

4.3.1 The IPMI controller implementation

4.3.2 The diagnostic service implementation

4.3.3 The management service implementation

4.4 The FMC HPC tester hardware design

5 The design verification

5.1 Verification of basic parameters

5.2 The IPMI controller

5.3 The AFC card diagnostic

5.4 Triggering modules test

5.5 Tests with the FMC125 module

6 Conclusion

Bibliography

List of abbreviations
## List of Figures

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The LHC (Large Hadron Collider) tunnel and the tube (which contains among the others vacuum beam pipe and superconducting coils)</td>
<td>15</td>
</tr>
<tr>
<td>1.2</td>
<td>The Colin 6 medical accelerator developed at the National Centre for Nuclear Research - Świerk</td>
<td>16</td>
</tr>
<tr>
<td>2.1</td>
<td>VME card height 2U</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>VME chassis height 2U</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>uTCA architecture</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>The system architecture</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>The AFC main block diagram</td>
<td>32</td>
</tr>
<tr>
<td>3.3</td>
<td>The clock distribution switch simplified block diagram</td>
<td>37</td>
</tr>
<tr>
<td>3.4</td>
<td>The FPGA simplified firmware architecture</td>
<td>40</td>
</tr>
<tr>
<td>3.5</td>
<td>Trigger lines organisation in the uTCA architecture</td>
<td>41</td>
</tr>
<tr>
<td>3.6</td>
<td>Connections diagram between the MMC and the MCH (the IPMI interface)</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Architecture of the diagnostics facility</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>Diagnostic service states graph</td>
<td>50</td>
</tr>
<tr>
<td>3.9</td>
<td>The FMC HPC tester block diagram</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>The AFC v.3 card attached to the uRTM-8-SFP card via uRTM connector</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Top side of the AFC v.3.1 card</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>The FMC_DIO 32CH card</td>
<td>63</td>
</tr>
<tr>
<td>4.4</td>
<td>The FMC_DIO 32CH Wisbone registers</td>
<td>63</td>
</tr>
<tr>
<td>4.5</td>
<td>Clock connection three</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>The I2C bus tree</td>
<td>74</td>
</tr>
<tr>
<td>4.7</td>
<td>The FMC-HPC carrier tester top side</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>An example clock signal from internal 20 MHz oscillator (measured with an oscilloscope with 50 MHz bandwidth)</td>
<td>78</td>
</tr>
<tr>
<td>5.2</td>
<td>First tests in the uTCA crate</td>
<td>79</td>
</tr>
</tbody>
</table>
5.3 The modules population list .............................................. 80
5.4 An example of sensors reading from the Shelf Manager level .................. 80
5.5 Prepared workspace for the FMC-DIO modules tests purposes ................. 83
5.6 I2C and SPI architecture on the FMC125 module [6] .......................... 85
5.7 Firmware architecture for the FMC125 module .................................. 86
5.8 The prepared workspace for the FMC125 tests .................................. 87
5.9 An example of received data (test patterns) ...................................... 87
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The microcontroller pins and peripherals utilization</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Diagnostic sensors located on the AFC board</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>Data structure used to describe sensors</td>
<td>49</td>
</tr>
<tr>
<td>3.6</td>
<td>FPGA diagnostic memory map</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Sensors structure</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>DC/DC converters - voltage levels verification</td>
<td>78</td>
</tr>
<tr>
<td>5.2</td>
<td>INA 220 sensors read tests</td>
<td>82</td>
</tr>
</tbody>
</table>
Acknowledgements

Authors would like to express their deep gratitude to Andrzej Wojeński, Adrian Byszuk, Paweł Zienkiewicz and Tomasz Janicki from the PERG research group, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. Furthermore, authors would like to thank Mr. Wojciech Zabołotny for his valuable suggestions concerning article edition.
Chapter 1

The particle accelerators

Nowadays humanity has reached a lot. The progress is mainly possible thanks to the science. The uncountable numbers of inventions, technologies, improvements have its origin in various laboratories distributed around the world. Many of these inventions have found use in our everyday life, becoming an dispensable part of it. We have already achieved so much that it may seem that there is nothing more to discover. But thankfully this is not true. Humanity still moves forward and looks for newer and newer solutions. Science is still an irreplaceble way of achieving progress. However, science also needs more and more instruments that are much more accurate, faster, allowing to see more, to prove more. One of these instruments is a particle accelerator.

1.1 Basics

The simplest definition describing a particle accelerator suggests its name. It is a device used for acceleration an electrically-charged, elementary particles. The device uses electromagnetic field to propel protons, electrons or heavy ions to velocity approaching the speed of light. The electromagnetic field also forms particles in a well defined beams. We could define several methods of classifying accelerators. The basic division is due to the track shape and even so we can distinguish linacs (Linear Particle Accelerator) and circular accelerators. There are also some other ways of classifying, because of the:

- accelerating particles type
- accelerating methods ie. electric field or high frequency
- an energy of acceleration

Some of accelerators are used only as initial injectors for a bigger one. It is quite common that linac is injecting pre accelerated particles into circular track of the other accelerator.
So far as the definition is not too complicated, the construction of the accelerator already is. It requires scientists from different fields of science to make it works. The following points show, how many items an exemplary accelerator contains:

- "Particle source - provides the particles that will be accelerated
- Copper tube - the particle beam travels in a vacuum inside this tube
- Klystrons - microwave generators that make the waves on which the particles ride
- Electromagnets (conventional, superconducting) - keep the particles confined to a narrow beam while they are travelling in the vacuum, and also steer the beam when necessary
- Targets - what the accelerated particles collide with
- Detectors - devices that look at the pieces and radiation thrown out from the collision
- Vacuum systems - remove air and dust from the tube of the accelerator
- Cooling systems - remove the heat generated by the magnets
- Computer/electronic systems - control the operation of the accelerator and analyze the data from the experiments
- Shielding - protects the operators, technicians and public from the radiation generated by the experiments
- Monitoring systems - closed-circuit television and radiation detectors to see what happens inside the accelerator (for safety purposes)
- Electrical power system - provides electricity for the entire device
- Storage rings - store particle beams temporarily when not in use"

Accelerators not only need scientists from a various area of science but also in return give diverse research opportunities. That is why, there are many research centres across the entire world, engaged in experiments using accelerators. Laboratories like: CERN (European Organisation for Nuclear Research), DESY, GSI (Helmholtz Centre for Heavy Ion Research), SLAC or Brookhaven National Laboratory have very complex and expensive facilities. These usually consist of a large number of buildings and an underground tunnel with the pipe of an accelerator. The example of that kind of tunnel is shown in Figure 1.1.
1.2 Meaning for the mankind

A particle accelerators were originally build for studying structure of matter and space. It is certainly a milestone in this area of science. The invention of a particle accelerator has allowed to elicit a nuclear reaction, thereby allowing to perform researches on an elementary particle and course of this reaction. These researches have lead scientists to a better understanding of the entire universe. In large colliders, scientists can experimentally, by leading extremely high energy of 14 TeV, perform a quite similar process to the origin of the universe. Analysis of structure of materials also significantly benefits from particle accelerators. Like the majority of scientific inventions a particle accelerator has also found its application in industry, medicine and other researches.

One of the major examples is the semiconductor industry. The silicon doping with boron or phosphorus requires energies of high ranges. Linacs and tandem Van de Graffs are used for these purposes. Moreover, it is also contemplated to replace the photo-lithography process, which uses visible light, with the X-ray lithography to achieve better resolution. This allows to produce integrated circuits in a smaller size but also contains more components. The process called ion implantation is also used to produce elements with improved resistance on high external forces, required in example in an automotive industry.

It is also worth to mention that particle accelerators have potential to threat nuclear wastes [10]. Besides, it could become one of the most effective way of cleaning water or polluted gases [10]. These are also already applied to preserve food and to sterilize medical products.
A very important area is a health care. It has to be mentioned about radiotherapy as a method of fighting with cancer. In all medical facilities around the world there are a compact linacs prepared for this purpose. The precision of hitting in diseased cells by a beam of particles is so high, that the protons or ions deposit their whole energy directly into specified points in the patient’s body, practically without affecting the neighbouring cells. The accelerators are widely used not only in treatment but also in diagnostics. Using a radioisotopes technology, doctors are able to detect tumours, anomalies in blood flow, glucose metabolism, etc. Also the nuclear magnetic resonance imaging method is derivative of accelerators, because in both cases superconducting magnets are used. Finally, these are useful for pharmaceutical and DNA researches. Thanks to the powerful X-ray beams from synchrotron light sources, scientists can analyse protein structures or DNA structures.

Medical accelerators are also successfully designed in Poland. A series of Coline [8] accelerators were designed by the NCBJ Świerk (National Centre for Nuclear Research). Produced accelerators are modern devices equipped with portal imaging system and communication interface for management and verification system. These accelerators guarantee precise and stable irradiation which is necessary in treatment process. In Figure 1.2 the accelerator model Colin 6 is shown.

Figure 1.2: The Colin 6 medical accelerator developed at the National Centre for Nuclear Research - Świerk [2]
Recently popular in medicine are also accelerators for hadron radiotherapy. The ions of carbon are precisely controlled to deposit the energy to the cancer cells thereby giving very good results in treatment. Facility of this kind was also established in Poland, located in Cyclotron Centre Bronowice [9].

There are a lot of other examples of applications for particle accelerators not mentioned in this section. Quite a few were well described in an article [10] on this topic published by CERN. However, there would not be so many applications for accelerators without specialised methods allowing measure their parameters.

1.3 Methods for beam intensity and position measurements

Knowing the advantages of particle accelerators we also know that it is an area of science which is worth studying and worth investing. This is exactly the same phenomenon as earlier described in a regard to whole science. A particle accelerators already gave humanity a great knowledge and experience from different disciplines, but who knows what else they can bring. Therefore it is still worth to perform researches using them. However, as we also already know, the accelerator itself is not enough. We also need tools and methods to register occurring processes and to look into their nature.

From a number of methods designed for beam parameters measurement there is a group of solutions destined for position and intensity determining. Beam position monitors are the most frequent diagnostics used at almost all linacs, cyclotrons and synchrotrons. The signal is generated by the electromagnetic field produced by a passing beam. In the PhD thesis of Grzegorz Kasprowicz [11], author has concluded description and analysis of three commonly used methods from these groups. These are: Beam Current Transformer (BCT), Wall Current Monitor and electrostatic and electromagnetic position pickup (PU). They provide an electrical signal to the output. Thus there is a need to sample and process the signal in order to obtain information about the beam. For this purpose, it is necessary to use a measurement system. It has to provide analogue signals sampling, processing and storing and also to synchronize measurements in the time domain.
Chapter 2

Integrated systems for data acquisition purposes

One of the instruments that allows to look into processes which occurs in particle accelerators are measurement systems. All designed systems regardless of their destination are organised in a reasonable architecture, or at least they should be. It may seem, that the best solution is to build a system in a full custom way. This allows to adjust system practically freely. Nevertheless, despite all advantages of such approach, there is also one serious flaw. It costs much more time and work. Thus, it seems to be worth to at least look at available solutions.

2.1 Architectures of currently used systems

Nowadays there are a few standards commonly used to build integrated measurements systems. VME Standard (*Versa Module Europa*) is one of the oldest commonly in use. Despite the fact that it was developed in 1981 it is still widespread today. The VME standard is defined by numbers of requirements that have to be meet and which are related to mechanical, electrical or powering issues. Architecture is quite analogous to the Motorola 68000 standard, especially when we compare signals on a backplane board. There are separate data and address buses, both are 32-bits wide and these are organised as parallel buses. These buses are controlled by nine arbitration lines. Control over the arbitration bus is performed by a card in the slot number one of the chassis. The VME bus also provides seven interrupts levels. In case if interrupts at the same level occur in the same time the one closer to the first slot card will be handled first. It is called a daisy chain architecture. As regards to mechanical design the VME cards as well as the chassis are available in three standard heights. In Figure 2.1 and Figure 2.2 the vme card and
the VME chassis in the middle height are shown.

![VME card height 2U](image1)

Figure 2.1: VME card height 2U

![VME chassis height 2U](image2)

Figure 2.2: VME chassis height 2U

It is worth to be mentioned about special VME bus extension called VXI (VME eXtensions for Instruments). This standard gives system designers better flexibility by providing some special lines such as triggering, clocking or synchronization. There is still a limitation caused by shared address and data bus. A recent extension for the VXI bus specification the 2eVME was added in 2004, giving it a maximum data rate of 160 MB/s. Undoubtedly one of the advantage of this standards is availability of many different extensions card both in VME and VXI standard, some of them are also available on an open hardware license. Certainly all problems and errors that could appear are also well known. However, it looks rather that capabilities of this platforms were fully utilized and will slowly come out of use.

Significantly more recent and more powerful standard was developed by the National Instruments company. They provided PXI and PXI Express (PCI/PCI-E eXtensions for Instrumentation) platforms that are based on the PCI/PCI-E (Peripheral Component Interconnect/Express) electrical bus in combination with the modular Eurocard package. It also contains specialized signals dedicated for synchronization and system clocking. One of the biggest advantage of this system is the fact that a single PXI card could work either in an ordinary PC computer or in a specialized PXI case. Standard is available on the open industry license. Nevertheless developing a new card is limited for a few reasons. First of all because of restricted number of lines on a backplane board. Beyond the PCI or PCI-E bus there are only a few lines. This causes that flexibility in new hardware designs is not sufficient. In case of PXI there is also a limitation associated with the PCI shared bus. It is well known problem occurring also in the VME standard, resulting in necessity of sharing bandwidth between connected devices. This issue is not present in case of the PXI Express standard, because PCI-E bus is organised as point to
point connections. An introduction of the PXI Express standard also improved access to interfaces available in the system. In case of PXI platform it is possible to access only one interface at the same time. However, there is still a problem with drivers for hardware which are often very complex and advanced. Drivers also are proprietary and have to be delivered by a hardware developer. These are usually not available on other platforms than the MS Windows. Furthermore PXI and PXIe suffer lack of board to board connectivity other than PCI Express. This makes impossible to setup high speed low and deterministic latency link between the boards. Front panel connectors and cablink must be used instead to provide point to point links.

The most powerful and advanced systems which also give the user the most flexibility in a custom cards design is nowadays the solution based on the ATCA (Advanced Telecommunications Computing Architecture) and the uTCA (Micro Telecommunications Computing Architecture) platform. In both cases specifications are extensive and define parameters form different ranges. Starting from mechanical (shelf and card sizes) through electrical (electrical connections types), interfaces, buses and ending with shelf management. System accepts cards compatible with the AMC (Advanced Mezzanine Card) standard. Novelty in comparisons with previous standards is the possibility of replacing cards during system work, provided by the ”hot-swap” function. Connection between a single AMC card and a backplane board is established via the AMC connector. It contains four groups of signals:

- **Group 1 - modules management interface (IPMI, JTAG)**

- **Group 2 - communication interface lines (PCI-E, GbE, SerDes, etc.)**

- **Group 3 - powering lines**

- **Group 4 - triggers and clocks (uTCA.4 for physics extension)**

Usually modules are equipped with an interface destined to communicate with external devices and separate interface to communicate inside the shelf with other modules. Typically external interface role is fulfilled by the Ethernet port. In case of an internal interface there is a wide range to choose from. The most popular are PCI-Express, RapidIO, 10 Gigabit Ethernet or SerDes. There is also a possibility to design own interface using the differential electrical standard. Of course it is not possible to use all of mentioned interfaces at the same time. A designer is making a decision at the moment of backplane board specification selection. However it is still a very flexible solution what is a great convenience.
System management is performed by the "Shelf Manager". It is using the IPMI (Intelligent Platform Management Interface) for this purpose. A transmission medium is the I2C bus. Typically to the manager tasks belongs:

- Handling modules inserting and removing in the system
- Collecting information about active modules
- Supervising work parameters such as: temperatures, voltages, etc.
- Preventing system failures
- Managing power supply, fans speed
- Arranging connections on the backplane, between modules
- Other

The uTCA and ATCA standards are largely similar to each other. It is so because the uTCA is a special case of the ATCA, developed mainly to reduce costs and complexity of the system, but still providing high performance and data throughput. There is a few rather subtle differences in the architecture of this platform. The uTCA could have from 1 to 2 MCH (MicroTCA Carrier Hub), which combines functions of the ATCA main board and the Shelf Manager. The uTCA architecture sill has power supply unit, cooling unit and programmable network connections between system components. An example of the uTCA architecture is shown in Figure 2.3.

![Figure 2.3: uTCA architecture](image-url)
Chapter 2. Integrated systems for data acquisition purposes

The confirmation of the fact that the uTCA platform fulfills the requirements for trajectory measurement systems in particle accelerators is the emergence of a special extension for this architecture called uTCA for Physics. It mainly refers to a time distribution, triggering and synchronization issues. This extension suggests to use eight independent M-LVDS (Multipoint Low Voltage Differential Standard) lines for this purposes. This is important, not only for physics but also for other measurements systems, to know precisely when specified data sample was acquired during an experiment. It is also important that this extension does not violate the existing standard, because it uses available resources.

It is also worth to mention the FMC (FPGA Mezzanine Card) standard. It mainly contains requirements for card size and a connector applied. A designer has a great number of possibilities to use it according to his own needs. It is so because the FMC standard defines numbers of connections for general purposes, which could work both in a single ended or differential electrical standard. Moreover, there are also a few fast giga-bit links, clock and powering lines. A number of available lines depends on a connector selection. There are two types of connectors defined: the LPC (Low Pin Count - 160-lines) and the HPC (High Pin Count - 400-lines). The standard also requires EEPROM (Electrically Erasable Programmable Read-Only Memory) memory for the FMC module identification purpose. A great flexibility of the FMC standard caused that it becomes an ideal extension of the uTCA platform. Hence the idea of the AMC carrier card with the FMC connectors provided. It is possible because the FMC gives a great opportunity of control and data transmission. A large numbers of reconfigurable lines require a device that have equally large number of reconfigurable pins. Perfectly eligible for this is an FPGA (Field Programmable Gate Array) device. As a result the AMC card becomes a carrier card with no exchangeable parts providing interfaces implemented in an FPGA, power supply and compliance with the uTCA standard. On the other side there would be the FMC connectors allowing to attach a fully replaceable modules. The only one requirement for replaceable modules is to provide compliance with the FMC standard. This allows to implement variety of functionalities for the AMC card. It could be cards with an ADC (Analogue to Digital Converter), DAC (Digital to Analogue Converter), physical layer for communication interface and many many others.

The preceding architectures comparison shows that the greatest flexibility is offered by the FMC standard. However, it does not define neither communication protocol nor architecture of a whole system. Thus cause that an extension is required. In combination with the uTCA architecture this platform creates very powerful and sophisticated solution. A growing number of that kind of solutions, appearing on the market, confirms that the capabilities are vast and this is the proper direction of that kind of systems development.
2.2 Available uTCA carriers

Because of the fact that uTCA architecture with combination of the FMC standard has growing numbers of applications, in this section a review of available solutions was concluded. First of all attention was paid on advantages and limitations of analysed projects. Besides also a type of license for available products was important. The idea of use an open license solution is one of the goal of this project. Therefore considered carrier card should have open source. There are some reasons for use the open source license. The main benefits are design reuse and better collaboration with the industry. Moreover it gives a designer a very useful feedback about his project. An extensive set of open hardware projects is the web site of the OHWR project \[12\] (Open Hardware Repository). Projects posted there are widely commented by users. They reported problems, bugs, suggestions and also questions.

The AMC to the FMC carriers are produced among the others by the Vadatech company. They offer four products of this type \[13\]. All of these have only one FMC connector. The most powerful is the AMC515 model. Mostly all lines from the AMC slot are connected to an FPGA device on this board. That allows to implement protocols such as PCI-E, SRIO etc. The clock lines are all fixed, so there is no possibility to route these freely. This card has 2 GB of DDR-III memory provided. It is accessible from the FPGA directly through the AMC slot. Access is managed by a multicore communication processor (QorIQ P2040). Optionally it is possible to connect another external memory chips directly to the FPGA. Obviously all of these products presented by the Vadatech company are commercial solutions and their source files are not available.

Another solution is presented by the Struck Innovative Systeme company. The SIS8100 \[14\] card is a single width AMC card. It has single PCI-E lane over PLX bridge (PCI-E to local bus bridge). On a front panel there is an optical link provided or optionally the FMC connector. Unfortunately capabilities of this card are quite limited. This is because clock signals are fixed. Moreover, the PCI-E interface is the only one which is connected to the AMC slot.

Yet another AMC to FMC carrier has been developed by the Lodz University of Technology, DMCS (Department of Microelectronics and Computer Science). They presented universal FMC compliant module for xTCA systems \[15\]. This card is also the single width and has one FMC connector (HPC) like the previously described cards. The module is equipped with the Virtex-5 FPGA device. Also 72 Mb of external memory was provided. Authors claim that it is possible to implement four types of high-speed serial data communication links (PCI-E x4, Gigabit Ethernet, Custom RocketIO and SFP (Small form-
factor pluggable transceiver). Thus most ports from the AMC connector are connected. On the front panel there are SFP transceivers cages and trigger lines connectors. These are placed on the top side of this card just as the FMC connector. This could cause some limitations in relation to the FMC modules that are possible to use due to the mechanical design. Also the fact that the FMCs gigabit links were left unconnected is a restriction. Moreover, despite that it is possible to use various clock sources, nevertheless connections are fixed and not reconfigurable. It is also worth mentioning that the FPGA device can be configured in multiple ways (dedicated AMC pins, Xilinx platform cable, custom MMC firmware) which undoubtedly is an advantage.

The most interesting and advanced products were developed by the DESY research centre. Especially two cards: DAMC-FMC20 [16] and DAMC-FMC25 [16]. These are both equipped in two FMC connectors (DAMC-FMC 20 HPC and LPC, DAMC-FMC 25 both HPC). The DAMC-FMC25 is more efficient than the second one, therefore the description relates primarily to the DAMC-FMC25 board. Thus, unlike in the case of previously presented cards these are double width wide. It is also unusual that they are equipped with two FPGA devices. One of these (Virtex-5) was used to data processing whereas the second one (Spartan-6) is the board manager. Usage of two FPGA chips cause that some part of resources are required to be used for inter-chips communication. Both FPGA devices have external memory provided (Virtex-5 512 MB DDR2 and Spartan-6 128 MB DDR2). Invariably there is still the module management controller for diagnostics and power management purposes. Designers provide wide range of communication links to use. The great advantage is use of the uRTM (Micro Rear Transition Module) connector which allows to connect additional extension card. Partly there is also possible to reconfigure clock connections. However, this only applies to connections between FMCs and RTM. Moreover there is no solution provided which allows using the AMCs dedicated JTAG lines to program devices on these two boards.

Thus, summarising this review of the market one can notice that there are quite a few solutions of that kind. Some of them are more expanded and powerful and some are less. Thus, their abilities are varied and can be adapted to various projects. However, none of these mentioned is the open source project. Moreover, there are some limitations connected with clock distribution and card reconfiguration. Besides none of them has support for the White Rabbit time synchronization protocol [45]. There are also only a few cards equipped with the RTM connector. Thus, there is a lack of the open source solution that could cover all demands.
The motivation, goal and objectives for this project

The analysis of presented solutions showed that there is not any open hardware design that could be fitted to build trajectory measurement system in the uTCA architecture. The demand for such a solution has been reported by the ELHEP (Electronics for High Energy Physics) group in which this document was prepared. Moreover, there is a software developed for automatic configuration of measurement systems also designed by a member of PERG Photonics and Web Engineering Research Group group. For these reasons the design of that kind of carrier card with required components has become the goal of this publication.

The assumptions for this design were made with respect to the destination places for this platform. These will be the particle accelerator called Cryring and CBM Compressed Baryonic Matter) which is a part of FAIR [17] (Facility for Antiproton and Ion Research) facility located in the GSI research centre as well the accelerator called Sirius located in the LNLS [18] (Brazilian Synchrotron Light Laboratory). Moreover, also available solutions with their advantages and limitations were taken into consideration. As a result following general requirements were developed:

- The designed platform needs to be available on the open hardware license.

- Typically a single bunch detector has four outputs, this requires the ADC daughter board to have 4 channels with sampling frequency of at least 100 Msp s with 12-bits resolution at each channel. This requires HPC connectors to handle such number of digital lines. Moreover, the HPC connector provides better flexibility and more possibilities for potential system reuse.

- Wide bandwidth for data samples which requires high speed and capacity memory block to buffer these data.

- System reconfiguration during operation.
• Flexible, reconfigurable clock signals routing.

• Support for the White Rabbit time synchronization protocol.

• Possibility of system upgrade via JTAG interface is required.

• Low cost solution based on recent FPGA technology.

• Advanced diagnostic capabilities - all critical currents and voltages on the board as well as temperatures should be monitored.
Chapter 3

Concept of the system

The main goal is to develop an open license trajectory measurement system in a particle accelerator based on the uTCA architecture. For this purpose several components were designed. The first of these, was the AFC (AMC to FMC Card) carrier card which is a base card for all the FMC extension cards. This card provides compatibility with the uTCA architecture, communication interfaces and interfaces to the FMC modules handling. Furthermore various types of the FMC extension cards, implementing different functionalities, were designed or selected from available solutions. This applies to cards with the ADC converters, physical layer for input and output ports (triggering ports) and cards required for automatic hardware verification procedure. This document describes whole system in general, mainly detailing the tasks connected with the AFC card design, performed by the authors.

At the beginning of this chapter, the analysis of the system architecture was concluded. As it was mentioned, system is organised in the uTCA architecture. There is used chassis produced by the Vadatech corporation. It is the model VT811 [19], that provides 12 AMC slots, dual cooling unit, JTAG switch module and some other features. In Figure 3.1 a simplified block diagram of this system is shown. At the front panel there are analogue inputs which allow to connect analogue signals from detectors to ADC converters. Each detector typically has four outputs, therefore each card with ADC converter have four channels. Signals are sampled with a frequency of at least 100 Msp (Mega Samples Per Second) and with resolution of at least 12-bits. Then digitalized data stream is read out from ADC converters by an FPGA device placed on the AFC card.
The FPGA device also processes data and buffers it in a static RAM (Random Access Memory) memory block. The data is also time stamped in order to maintain the proper order. Measurements synchronization is provided by the FMC triggering module, one in each uTCA crate. Its task is to generate session start signal and one milliseconds ticks. These signals are transmitted to each AMC cards through the M-LVDS interface. Afterwards prepared data with time stamps are transmitted through the backplane board via PCI-E bus to the PCI-E switch.

This task is performed by a commercially produced by the Concurrent Technologies company module called the AM 90x/x1x [20]. It is a single board computer designed for the uTCA architecture. Reading and parsing the data from each AFC cards is managed by a software application running on this computer. All the informations together with configuration parameters are displayed on a graphical website application. An ordinary user has access to a whole system data and configuration parameters form any PC computer with the Ethernet port and a web browser.

Simultaneously with the measurements channel there are also available diagnostic and
management channels. In this area actions are carried out locally on each AFC board by a microprocessor and globally by the MCH (Micro Carrier Hub) crate controller. All AMC cards are connected with the MCH crate controller through the IPMI interface. Each interaction such as power allocation, fan speeds control, cards status are taken based on the information transmitted throughout this interface between the micro controller and the MCH. The MCH module is also commercially produced by the Vadatech company. It is model number UTC002 [21] developed for use in the uTCA chassis, which combines the Carrier Management Controller and the Shelf Manager in one device.

There are also other miscellaneous, modules in the chassis, such as power supplier, cooling fans module and status indicator. These are necessary components of the uTCA architecture. In this case, solutions offered by the chassis producer were applied.

As to the diagnostic part of this system, it should also be mentioned that all diagnostic informations from each AFC cards are send to the AM 90x/x1x module and also are displayed on the web site.

### 3.1 The AFC carrier card

The AFC card has to provide a few basic features. It was assumed that designed card will be the AMC double-width mid-size module, working in the uTCA architecture. Therefore, it was necessary to place the IPMI controller on it. Afterwards, because this controller has to report to the Shelf Manager card working conditions, diagnostic sensors were also placed. Obviously all integrated circuits require proper power supply. Because the backplane board provides only 12 V (Volt) and 3V3 (for card manager) positive DC power, there is a need to adjust voltage levels. It is also required to provide proper clock signals for sequential circuits. It is advantageous to retain the ability to freely configure sources and destinations of clock signals. The listed requirements are providing basics for a work.

Afterwards, there was a need to provide a possibility to connect extension cards which are implementing a proper functionality of a single AFC card. It was assumed to provide two FMC connectors on each AFC card in the HPC specification. It caused that there was a need for an efficient device that is able to handle this connectors. Furthermore because of the multitude of possible applications it was essential to use integrated circuit which can be configured in almost any way. Such a capabilities are offered by technology of an FPGA integrated circuit.

Moreover, throughout the AMC connector a single card is able to communicate with the other components via PCI-E bus. Thus it was necessary to connect also PCI-E lines.
to the FPGA and the AMC edge connector. Furthermore, it was also required to buffer data before sending. An external memory block is provided for the purpose. At last, also lines intended for external trigger signals are connected to the FPGA.

In order to meet the above assumptions the block diagram was prepared [3.2]. As it was mentioned, two separate FMC connectors are connected directly to the FPGA device. To allow data buffering a static RAM memory is used. The clock signals distribution is configured by the microcontroller. Moreover, this microprocessor also performs functions of the IPMI controller, power and temperature manager and other less important functions. In accordance with the uTCA for physics specification there are also eight M-LVDS lines provided for synchronization and clocking purposes.

![Figure 3.2: The AFC main block diagram](image-url)
3.1.1 The FPGA device selection

In order to select the FPGA device a few requirements were assumed. The FPGA device is connected to the FMC connectors in accordance with the connector specification. It is allowed to use lines in a differential or single ended electrical standard, depending on the needs. Thus, from each of the FMC connectors there are 160 user-defined pins connected. There are also four differential pairs of clock signals.

Afterwards, it was required to provide 2 GB of DDR3 SRAM memory with 32-bits wide data interface. This implied the usage of four memory chips with a capacity of 512 MB each. Whole memory interface therefore contains 32 data lines, 16 address lines and 24 other control lines (thus 72 lines in total).

Further, there was a need to connect also some communication interfaces, such as the SPI and the I2C to communicate with the microcontroller and optionally other chips placed on the AFC boards. It was also worth to provide UART to USB converter mainly for firmware development purposes. Moreover, eight M-LVDS triggering lines were also taken into consideration. As a result, we obtain requirements for about 430-440 general purposes lines in total.

It was also required to provide fast giga-bit links for the FMC links and the PCI-E bus. Two PCI-E x4 Gen2 buses were used. It provides theoretical throughput of 4 GB/s. Optionally two of these links are used for the White Rabbit time synchronization protocol implementation. To each of the FMC connectors there were connected four giga-bit links. As a result we obtain 16 GTP links required in total.

The estimation of required logic elements has been performed with respect to the documentations of the example designs found on the web pages of the Xilinx company [22], OpenCores project [23] and also from own projects reports. On this basis, it was determined that:

- DDR3 SRAM memory interface requires about 15 k logic cells,
- PCI-express x4 gen2 requires about 1 K logic cells,
- Wishbone master with two slave components and interface for four ADC channel with ChipScope analyser also attached - 2.5 K

These demands are fulfilled by the Artix-7 [24] in the FFG1156 package. It has up to 500 general purposes user pins, 16 GTP links, DDR3 Memory interface. Its giga-bit transceivers components are able to support up to PCI-E x4 Gen2 bus. It also has 215 K logic cells. Thus in relation to above mentioned example resources utilization, there is a large reserve remaining. Of course it depends on what algorithms will be implemented.
3.1.2 Diagnostic and management device

For the implementation of diagnostic and management services and also the IPMI controller, an ordinary microcontroller was the best choice. Therefore the requirements for this microcontroller were considered. One of the most important is a power voltage range. This microcontroller must work with a supply voltage of 3.3 V, because such a voltage is applied to the IPMI controller through the backplane board. There are no other requirements connected with power consumption, because device is powered from the grid. The next step was the analysis of needed resources such as general purposes pins, peripherals and so on. In Table 3.1 needed resources have been summarized.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Quantity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO</td>
<td>45</td>
<td>general purposes input/output lines required for: IPMI interface, power management, FPGA control lines, JTAG interface control, FMC monitoring</td>
</tr>
<tr>
<td>I2C</td>
<td>3</td>
<td>I2C peripheral bus controller required for: IPMI interface, digital temperature and power monitors reading, clock signals switch configuration, communication with the FMC modules other chips handling</td>
</tr>
<tr>
<td>SPI</td>
<td>3</td>
<td>Serial peripheral interface for: communication with an FPGA, SPI flash memory control and for power management</td>
</tr>
<tr>
<td>UART</td>
<td>1</td>
<td>Universal asynchronous receiver and transmitter for: debugging and diagnostics console</td>
</tr>
</tbody>
</table>

Table 3.1: The microcontroller pins and peripherals utilization

Summarizing the table 3.1 it was required to use a microcontroller with at least 65 pins. These requirements are met among the others by the NXP LPC1764 [25]. Obviously chosen microcontroller could be replaced by other available at the market. However this issue is not so important at this project and it was decided that this part is sufficient. Selected microcontroller could operate with voltages from 2.4 to 3.6 V. Besides it has up to 70 general purposes input output ports which is enough. It also has all the mentioned peripherals in a sufficient quantity. It is based on the Cortex-M3 [26] core, which is also an advantage because it is popular and well known architecture. There is of course possibility to reduce requirements for available peripherals because some of them could be shared. Nonetheless it requires to implement more complex firmware, due to the fact that
information from different sources have to be classified in somehow to take proper inter-
actions. Whereas this means that firmware implementation will take significantly more
time. Having regard to this and also that there is sufficient space on the AFC board it
was decided that no further changes are required.

3.1.3 The power supply block

As it was previously mentioned there is a need to adjust voltages to proper levels. After
card insertion there is only the IPMI controller powered from 3.3 V power source. Other
integrated chips could be powered after performing ”power negotiations” between the
microcontroller and the Shelf Manager. The input voltage from a backplane board is at
level of 12 V. Whereas on the AFC board there are following voltage levels required: 1.0 V,
1.2 V, 1.5 V, 1.8 V, 2.5 V (FMC adjustable voltage level, typically 2.5 V), 3.3 V, and also
12.0 V (also for the FMC). Moreover it is recommended to supervise all voltage levels and
current consumption of both FMC connectors. Those are voltages: VADJ(typically 2.5 V),
3.3 V and 12 V. Besides it is required to allow power on and off sequencing in any way. The
power sequencing and supervising are performed by the microcontroller. Afterwards, the
power requirements have been analysed to select sufficient power converters. The power
estimation at each voltage levels were listed below:

- 1.0 V - It is an internal FPGA supply voltage destined for: internal circuits, block
  RAM and GTP transmitters and receivers. Consumption at the level of 10.5 A is
  expected here.

- 1.2 V - It is also an internal FPGA supply voltage destined for the GTP transceivers
  termination circuits. Current draw of about 1.0 A is expected.

- 1.5 V - This is power a source for four DDR3 memories (1 A), FPGA memory
  interface (1 A - 2 banks), an also a few voltage level translators for the M-LVDS
  physical layer (8 x 50 mA). Thus, as a result the current draw from this source could
  be up to about 2.5 A.

- 1.8 V - An auxiliary supply voltage for the FPGA. Current draw at level of 2.4 A
  is expected here.

- 2.5 V (VADJ) - These are two independent channels for each FMC connector. Ac-
  cording to the FMC specification current draw on these lines could be at most
  4 A (+/- 5 %). Moreover, there was also included the power required for the FPGA
  banks connected with a particular FMC connector. Thus, it is required 4 A for each
FMC connector plus about 0.5 A for each FPGA bank (one FMC require 4 banks - 50 lines per bank). Finally, each of the two independent channels have to provide current of about 6 A.

- **3.3 V** - This voltage level is required for most of the AFC integrated circuits and also for the FMC connectors. Each of the FMC require at most 3 A. There are separate converters applied for each connector and also one more for integrated circuits on the AFC card. Regarding to the connectors supply, the case is simple. Specification says that as it was mentioned 3 A is required for each. In the case of the remaining converter the case is slightly more complicated, because one have to take into account more integrated chips. Thus, this converter has to provide power for clock distribution unit, JTAG switch module, M-LVDS transceivers and also a few less important integrated circuits. Thus, the total current consumption at this channel has been estimated for about 2 A.

- **12.0 V** - This voltage level is also required by the FMC standard. Of course it is connected directly from a backplane board. Nevertheless, it was mentioned here because it is important in the total power consumption estimation. Current draw for each FMC connector could reach at most 1 A at this line.

All the assumptions were made in regards to the Artix-7 documentation [24], FMC specification [27] and also other specifications of integrated chips used in this project. For the FPGA power estimation, the Xilinx power estimation spreadsheet [28] was also used. For this spreadsheet purposes estimated quantity of I/O lines, GTP links and DDR memory interface links were used. For each of the power lines was also adopted a safety margin of at least ten percent.

The total power estimation was also performed in this section because it is required by the Shelf Manager to specify card power requirements. Assuming 85% efficiency of each converters the result of about 100 W was obtained. It should be noted that assumptions relate to the maximum power requirements. Furthermore, it also contains a power reserve. Therefore, it may be expected that the real power consumption will be lower.
3.1.4 The clock distribution section

Providing a proper clock signal for a sequential logic devices is essential to ensure proper operation. Therefore, this section is very important. A clock signals have to be provided to the FPGA device as well as the FMC connectors. The source of these signals are able to be an internally produced signals on the AFC cards as well as signals from the backplane board and the FMC modules. By introducing the possibility of the clock signals connections reconfiguration, the whole AFC card configuration abilities are greatly increased. That sort of solution allows to connect any clock signal source to any receiver, thereby providing to a transceiver block any reference clock signal or synchronization with any clock source. In Figure 3.3 the simplified block diagram of this idea is shown.

Thus a firmware designer is able to chose source signals from the FMC dedicated clock signals, the backplane board or an internally produced clock signals. Both the FMC clock signals and the backplane clock signals come from the outside of the AFC board, therefore the board cannot control their parameters. However, it is possible to synchronize respective blocks to the corresponding clock signals. These may be for example signals from the ADC converter placed on an ordinary FMC board, used for data clocking or signals from the backplane board for the MGT transceivers. There are also a few clock sources available on the AFC board. These are sources among the others for the White Rabbit time synchronization protocol implementation, DDR memory, etc.

The clock connections were not specified exactly in this section, because hardware design assumes delaying this decision to the time of firmware design implementation. As it was mentioned, this solution increases possible applications of the AFC card.

![Figure 3.3: The clock distribution switch simplified block diagram](image-url)
3.1.5 Fast serial links for data transmission

Data acquisition systems usually generate significant amounts of information that have to be transmitted. Therefore the AFC card is equipped with easy method of gigabit links interconnects configuration. These are distributed between both FMC connectors, FPGA gigabit transceivers and uRTM extension connector. Layout of connections is determined by mounting SMD capacitors in appropriate locations. Hereby, a user can freely determine data flow through the gigabit links. This solution provides possibilities to attach various number of extension cards alike in the FMC standard and in the uRTM modules.

3.1.6 A card diagnostics

It is extremely important in that kind of designs to supervise parameters of key elements. It is so for two reasons. First of all, it is possible to prevent system failures or damages caused by over-temperature, over-voltage, etc. Moreover, knowledge about values of these parameters is necessary to verify for example collected measurements. Let us assume that in the case when during normal operation some unexpected events would have occurred (ie. exceeding the maximum or minimum rating for some integrated circuit), there would be a reasonable suspicion that measurements are not accurate.

For reasons mentioned above, several sensors were placed on the AFC board. Thus, there are five temperature sensors in the most critical points of this board. These sensors allow to measure temperature of the FPGA device, DDR3 memory and an ambient temperature of voltage converters and both FMC modules.

Moreover, there are also six sensors to monitor currents and voltages provided to the FMC modules. The following lines are supervised: 12 V, 3.3 V and V_ADJ. Current draws and voltage levels provided to the AFC card are also examined, but it is performed by the Shelf Manager.
3.2 The FPGA firmware

The FPGA firmware is designed in a hardware description language. In this part of the
document the architecture of firmware is discussed. Afterwards, the firmware for triggering
cards is described as it was also the author part of work.

Thus, independently of tasks performed by a particular module, there is always a need
to read or set some configuration parameters. There is a lot of settings in that sort of
systems and therefore a repeated rebuilding of the project is not an option. Thus it was
strongly recommended to provide some communication channel to make this configuration
possible.

A software designed for the FMC modules configuration was developed by A. Wojeński
as a part of his Master thesis [29]. He described the method of sending and receiving
various parameters between an ordinary PC computer and an FPGA device. This solution
moves the problem of configuration to the PC software called the FCS (FMC configuration
software). In the FPGA device the Wishbone bus has been used (bus description given
in [30]). It is organised in the master-slave architecture. The master component on this bus
is the wishbone module prepared to communicate with a PC computer and slave modules.
Slave components are also wishbone modules. Depending on their role it may be a set of
simple registers of interfaces. These components allow to configure some parameters or to
communicate with external devices. However, it makes no difference from the point of the
FCS software, because the communication boils down to a simple read or write operation
to a proper register with a proper value.

It is also worth to mention an open license application called wbgen2 [31], designed
to automatically generate Wishbone slave components. The input file for this software
contains list of required registers, access methods and simple description. As a result
a designer obtains: implementation of a slave module in the VHDL (Very High Speed
Integrated Circuits Hardware Description Language) language, header file with a map of
registers in C and also the HTML (HyperText Markup Language) documentation. These
methods have been applied in this design.

Thus, the diagram shown in Figure 3.4 describes the architecture mentioned above.
The Wishbone bus is used to set and read various parameters. Moreover, throughout this
bus it is possible to read diagnostic measurements stored in a block RAM memory. Afterwards,
it also allows to communicate with external integrated chips because it implements
Wishbone slave components with i.e. I2C bus or SPI bus bridge. A large collection of var-
ious Wishbone components is available on the OpenCores project web site [23] and also
on the OHWR web site [12]. Thus, in this firmware design a large number of implemented
and verified components have been used.

![Diagram](image.png)

**Figure 3.4: The FPGA simplified firmware architecture**

Channel for data transfer is a separate block. The data stream e.g. from an ADC converter, is transmitted through the interface block to an optional data processing component. Afterwards the data samples are captured by the data switch unit. Samples are alternately (from the FMC 1 and the FMC 2) organised in a proper order. Moreover they are also time stamped. Then they are stored in external memory chips. From the memory the data are subsequently read and transmitted through the PCI-E bus.

### 3.2.1 System triggers and interlocks

A trigger role in designed system is performed by the AFC carrier card with the FMC_DIO-32ch card [32] attached to the FMC slot. It is quite simple card equipped with four input/output buffers with configurable ports direction. Because these buffers are working in TTL electrical standard, there are also logic level translators to adjust voltage to the level of V_ADJ. There is also a power converter, that provides power for the TTL buffers. Moreover, an EEPROM memory (FMC compatibility) chip and a temperature sensor are also provided.
The firmware design and also some part of software driver was prepared in regards to the convention described above. Nevertheless it is slightly different in a few points. The architecture shown in Figure 3.4 relates more to a collection of data ie. from ADC converters. The data are then stored and transmitted.

In this case it is different, because a simple trigger signal needs to be transmitted to other AFC cards immediately. The M-LVDS lines are used for this purpose. The architecture of trigger signals in the system is shown in Figure 3.5. Therefore, there aren’t any blocks intended for data process and store in external memory. Similarly there are still the Wishbone bus and the diagnostic block used. Because there are a few parameters to set on the FMC_DIO card, thus there was a slave Wishbone component generated. This slave module is able to control lines direction, states and outputs. There is also a slave component for the I2C bus support. This one is used for the EEPROM memory and temperature sensor handling.

Afterwards, there was also prepared a file for the FCS software application. It contains a description of a main program loop. It is used for initial configuration of all FMC module ports direction and state. Moreover it also contain methods intended for verification of firmware in an FPGA and also in the FMC module.
3.3 The microcontroller firmware design

The description of the microcontroller firmware design was divided into two cooperating parts to make it easier to explain operating principles. The first part provides compatibility with the uTCA architecture, whereas the another part is engaged in reading diagnostic data from all sensors available on the AFC board, providing thereby information about card status. Moreover second part also manages major card sections, such as clock signals distribution block, power management, etc.

3.3.1 Intelligent platform management interface

In order to provide integrity with the uTCA architecture, it is required to implement the MMC (Module Management Controller) controller. It is implemented as specialized firmware executed by the LPC1754 microcontroller. It is a kind of a low level hardware management service based on the IPMI (Intelligent Platform Management Interface) interface. There are a lot of requirements that have to be met referred to IPMI and uTCA specifications. In this section of this publication only the conception of key implementation issues was described. Whereas operating principles of the IPMI interface is more precisely described in the IPMI specification.

Electrical connections between MMC and MCH are shown in Figure 3.6. As it could be noticed, they are not too complicated. This interface consists of: power line (MP), address determining lines (GA 0, GA 1, GA 2), data and clock lines (SCL and SDA - the I2C bus lines) and there are also two presence lines (PS 0, PS 1) which indicates module insertion.

The main idea and complexity of this interface is hidden in set of commands and functions that are required to be implemented. By using these commands the Shelf Manager and also the Module Manager are able to communicate and make specified interactions. This set of commands is included in the specification for the IPMI interface and it is extended by the uTCA specification. The described implementation implements mostly all of the mandatory commands as well as commands that were needed additionally for this system.

Each operation starts from a proper card registration in the Shelf Manager by the MMC controller. Typically it starts from card insertion process during which the Shelf Manager observes a change of presence pin state on a proper AMC slot. After that on the AMC card only the MMC controller should be powered properly and should be ready to communicate with the Shelf Manager. The MCH card then starts reading informations about the inserted card. These informations are defined by specification as the FRU (Field Replaceable Unit). It is used primarily to provide inventory informations about a
card. This consist of a few main fields, these are:

- **Common header** - This is a mandatory field. It contains basics information about firmware, IPMI version and also about offset values to other FRU fields.

- **Internal use area** - It is usually a private field and the content is implementation dependent.

- **Chassis info area** - This area is used to hold some informations about a system chassis.

- **Board info area** - This area provides information about the board.

- **Product info area** - A separate field for product info.

- **Multi record info area** - This is one of the most important field from the point of this system. It contains informations about current draws, backplane interconnections and also issues which could be implemented by extensions of this standard.

This implementation contains following fields: common header, board info area and multi record info area. As it was mentioned, the most important field is the multi record area. This field presents necessary informations about system configuration defined by the uTCA standard. All implemented records are organised in a special data structure prepared for this purpose. Names of this structure fields are consistent with the FRU [34] and the uTCA [5] specifications, in order to simplify later filling in. The FRU provides the MCH some basic information about firmware revision, supported IPMI version and also a little more complex data such as for example point to point connections at the backplane board.

These data are delivered to the Shelf Manager after card insertion. Afterwards the Shelf Manager is waiting for a message that a hot swap handle has been closed. The idea of this switch is explained later. After the hot swap handle was closed and the Shelf Manager receives information that this event occurred, the card activation sequence starts. During this process the MMC controller has to respond for a few more commands. Among the others the power negotiation procedure is carried out. At last, as a result of proper responses the MMC controller passes through the state transition diagram into the active state. A proper AFC card operation could start from here. From this moment, there is power allocated, connections are established and the AFC card is able to operate.
There is a requirement for live card removal and insertion. Each AMC module has a handle which activates the micro switch (hot swap). Its state needs to be under supervision of the MMC controller. Its use allows to provide cards insertions and extractions during system operation without power down and up cycle required of whole uTCA crate. The point is that, the 12 V on the power line for each of AMC modules could appear after that switch was closed and it will be disconnected otherwise. A simple hardware interrupt service is used for this purpose. The interrupt handler is called always after rising or falling edge occur on the hot swap switch port. The handler is not send information about a change immediately because there are no hardware debouncing circuits on the AFC card. Therefore a simple timer is used for a software switch debouncer. It simply waits a period of time from the last detected edge and afterwards it compares the latest stable handle state with the present state. If they are not equal it generates message to the Shelf Manager with a new hot swap switch state.

As it was already mentioned, the IPMI interface uses the I2C bus to communicate. The interface specification defines a large number of commands. Thankfully, not all of them are mandatory. Some commands are optional and are implementation dependent. However, at first it was necessary to implement a few services in the MMC controller in order to support the IPMI interface.

First of all it is required to provide data transmission channel and methods for this channel to transmit and receive this data. The I2C bus controller on the MMC has to be usually a slave device which receives data and sends acknowledgements when it is addressed by the Shelf Manager. However, it also sometimes has to enter into master mode.
in example when the MMC controller has to respond for a received inquiry. Solution of this issue boils down to setting appropriate bits in the I2C bus control register. A special procedures were prepared for this purpose. Data reading and writing operations are performed by the I2C controller interrupt handler. It was decided to use the hardware microcontroller interrupt service because it is necessary to take some interactions immediately when the need arise to provide correct data transfer and to avoid timeouts. Therefore the MMC controller has to go to the I2C communication handler regardless of the performed task each time when interrupt occurs. Thus this interrupt source has the highest priority from all possible. The interrupt handler is organised as a state machine which follows the steps according to I2C bus controller current state (read from the I2C status register).

Afterwards, in order to store received data as well as data to transmit, there were two separate buffers provided. Each message has its own header containing important information such as message length, checksum, functional group, command code, etc. Preliminary interactions with each of received message are able to be made, on the basis of this message header. Among the others a decision whether the message has been correctly received is taken. If for some reason the received message is not a proper one for example if it is too short or checksum is not correct, such message is deleted from the buffer and is not processed. Otherwise it is processed by a messages parser. This is the dedicated message parser for the IPMI messages. At first a received message is classified upon the functional group and afterwards upon the command code. If the received message is supported by the controller an appropriate response is created and transmitted.

Besides communication with the Shelf Manager, the MMC controller obviously has to cooperate with the hardware components placed on the AFC boards. In this case I mean components that are significant for the Shelf Manager. These are LED diodes, diagnostic sensors and hot swap switch. The hot swap switch was already described. So now we will focus on these diodes. There are three LED diodes on the AFC card front panel provided to indicate card state. These are hot swap state indicator (blue led), card activity led indicator (green led) and error led (the red one). Obviously according to the LED state an operator is able to quickly diagnose each card. Parameters of each diodes are stored in a special data structure. The shelf manger is capable to write informations into this structure through dedicated commands thereby setting the LED state remotely. Each of diodes may be in one of possible states: on, off and blinking with a custom period. It is also worth to mention that the blue led and the green led are mandatory and have well defined tasks. Whereas this red one is an optional diode and the microcontroller is capable to indicate also some other failures.
As regards to the diagnostic sensors, the reading is processed by a diagnostic service. However, the Shelf Manager is also capable to read this data throughout the IPMI interface thanks to the proper commands. Each time when the diagnostic service has updated the data from sensors, the dedicated IPMI diagnostic service (implemented on the MMC) checks new readings in order to compare them with sensors thresholds. In case of for example too high temperature a proper message is generated to inform the Shelf Manager about a particular event. All sensors available from the IPMI interface need to have the SDR [33] (Sensor Data Record) implemented. This record defines a few parameters for each sensor, such as: type, range, thresholds, name and so on. All these data are organised in the data structure. The Shelf Manager is capable to read these structures fields during operation.

The main program function is a simple infinite loop. Most of the microcontroller tasks are performed from hardware interrupts, therefore the main loop is very simple. It constantly checks if there are any incoming or outgoing messages available to process. It also verifies whether sensors have been updated and also if some interactions related to power supply are expected.

Afterwards, because there are a few functions that were planned to be executed periodically, one of the microcontroller timers is used to generate periodical interrupts. The timer interrupt handler checks whether a period of one of these functions has expired and executes it when necessary. It is not a part of the main loop. There are a few such functions and these have well defined tasks. Thus, one function for the IPMI messages retransmission has been planed. Moreover the other function simply counts system up time. Yet another checks if a whole module or the FPGA device restart timer has been set (it could be set by the Shelf Manager) and if so, it also counts a time to the restart and performs it at the appropriate time.

Furthermore a diagnostic IPMI serial console has been planned. It is a low level priority function which prints all messages and important events onto the diagnostic console throughout the serial interface. It is also performed by the hardware interrupt handler. Data is buffered and transmitted during the microcontroller idle time. A hardware FIFO queue (First In First Out) was used for this purpose and because of the limited capacity a software temporary buffer was also used. The use of such console is very important from the viewpoint of the firmware debugging and diagnosing issue. This is the main reason for which it was applied.
3.3.2 The diagnostic service

The description of firmware design has been divided intentionally into two sections, to emphasize the operating principles of the diagnostic service. As mentioned, diagnostics data is readable through the IPMI interface. However in the designed system it is not a major channel of the diagnostics data distribution. The Shelf Manager is able to read only essential data from the AFC card, sufficient to take appropriate interactions such as, for example adjustment the speed of cooling fans. This solution allows to limit amount of data sent to and read from the Shelf Manager. In this system the PCI-E bus is the main channel for information transfer. It is used not only for transmission of diagnostic data, but also for measurements data and so on. That means, that all data may be read and written from or to the one bus. It is much simpler to organise these information in a consistent and clear structures. Besides the PCI-E bus allows to read much more data in a shorter time without the MCH board duty cycles requiring. However, the PCI-E bus is provided by the FPGA firmware, so there was a need to implement a method of communication between the FPGA and the microcontroller. It is also one of the diagnostic service tasks. Architecture of the diagnostics facility is shown in Figure 3.7.

![Figure 3.7: Architecture of the diagnostics facility](image)

It was decided to use the microcontroller hardware interrupts in order to implement the diagnostics service. That was because it helped to save some idle processor state. This issue is important from the perspective of providing stable communication between the AFC card with the Shelf Manager implemented on the MCH through the IPMI interface. It is worth to notice that the Shelf Manager constantly checks if all inserted AMC cards are still present. If for some reason card stops to answer for the MCH request, it is classified as some kind of error. Therefore, during periodical work of the diagnostic service, there are no statements that cause the microcontroller idle waiting in example during communication with an external sensor which takes a while. Awaiting statement can prevent to answer for the MCH requests in a proper time so, as it was mentioned it causes unexpected
errors. It is obvious that one should avoid such a situation in order to improve the system reliability. Therefore it is natural to use the microcontroller hardware interrupts. It is also worth mentioning that proposed solution is resistant for any sensors failures. It is so because, in the worst case sensor failure can cause that data will not be a valid one. If awaiting statements have been used failure like this could cause whole system will hangs and will stops responding.

Because all chosen sensors have digital interfaces with the I2C bus provided, it was possible to connect all of them to the one microcontroller I2C bus in the master-slave architecture. A special care is required here during selecting sensors addresses on the I2C bus to avoid addresses conflicts. The usage of shared bus for all sensors have one significant advantage. Namely because only one type of transmission medium was used to communicate with sensors, it allows to simplify implementation of this part of the diagnostics service. Moreover, it is not required to read data extremely fast, the whole sensor list scanning could take a while (ie. 10 seconds). Therefore it is possible to use shared bus and to read data sequentially. The list of supported sensors with their assigned ids and the type of returned data is presented in Table 3.3.

<table>
<thead>
<tr>
<th>Sensor name</th>
<th>Measurement type</th>
<th>Unit</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX6642</td>
<td>FPGA temperature</td>
<td>°C</td>
<td>0x01</td>
</tr>
<tr>
<td>LM75AIM</td>
<td>FMC1 temperature</td>
<td>°C</td>
<td>0x02</td>
</tr>
<tr>
<td>LM75AIM</td>
<td>FMC2 temperature</td>
<td>°C</td>
<td>0x03</td>
</tr>
<tr>
<td>LM75AIM</td>
<td>DC/DC converters temperature</td>
<td>°C</td>
<td>0x04</td>
</tr>
<tr>
<td>LM75AIM</td>
<td>SDRAM memory temperature</td>
<td>°C</td>
<td>0x05</td>
</tr>
<tr>
<td>INA220</td>
<td>FMC1 - 12 V current draw/voltage level</td>
<td>A/V</td>
<td>0x06/0x0C</td>
</tr>
<tr>
<td>INA220</td>
<td>FMC1 - 3.3 V current draw/voltage level</td>
<td>A/V</td>
<td>0x07/0x0D</td>
</tr>
<tr>
<td>INA220</td>
<td>VADJ current draw/voltage level</td>
<td>A/V</td>
<td>0x08/0x0E</td>
</tr>
<tr>
<td>INA220</td>
<td>FMC2 - 12 V current draw/voltage level</td>
<td>A/V</td>
<td>0x09/0x0F</td>
</tr>
<tr>
<td>INA220</td>
<td>FMC2 - 3.3 V current draw/voltage level</td>
<td>A/V</td>
<td>0x0A/0x10</td>
</tr>
<tr>
<td>INA220</td>
<td>VADJ current draw/voltage level</td>
<td>A/V</td>
<td>0x0B/0x11</td>
</tr>
</tbody>
</table>

Table 3.3: Diagnostic sensors located on the AFC board

Each of these sensors are described by a few parameters gathered in a special table of structures prepared for this purpose. All fields in this structure with their data type format are presented in Table 3.5. The diagnostic service continuously and periodically reads measured parameters from all available sensors to keep it up to date. Each data reading
is initialized by a hardware timer interrupt handler by generating the start condition on the I2C bus. Furthermore the timer interrupts handler copies the information about currently updated sensor into an auxiliary structure to operate on this data during this sensor updating. This contains information such as: slave address, register addresses, how many bytes is required to read and how to handle raw data from sensor. Afterwards the diagnostic I2C bus handler takes actions to read data from the sensor.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slave I2C address</td>
<td>unsigned 1 byte</td>
<td>Sensor address on the I2C bus</td>
</tr>
<tr>
<td>measured value</td>
<td>floating 4 bytes</td>
<td>Sensor measured value</td>
</tr>
<tr>
<td>register address</td>
<td>unsigned 1 byte</td>
<td>Table with addresses of registers to read from sensor</td>
</tr>
<tr>
<td>raw data</td>
<td>unsigned 1 byte</td>
<td>Table with raw data read directly from sensor registers</td>
</tr>
<tr>
<td>number of registers</td>
<td>unsigned 1 byte</td>
<td>Indicates how many registers we want to read</td>
</tr>
<tr>
<td>function pointer</td>
<td>pointer to floating 4 bytes</td>
<td>Pointer to calculating function. This function is executed each time when the diagnostics service ends collecting raw data from the sensor registers. Afterwards the function calculates measured parameter and stores this value in the measured value field</td>
</tr>
<tr>
<td>bytes from register</td>
<td>unsigned 1 byte</td>
<td>Indicates how many bytes we want to read from each register e.g. if we want to read 1 byte from the register at the address 10 and then we want to read 3 bytes from register at the address 20 we have to fill register address field 0 with the value 10 and field 1 with the value 20. Then field bytes from register 0 with value 1 and field 1 with value 3. Finally, in field number of registers value 2 is expected.</td>
</tr>
</tbody>
</table>

Table 3.5: Data structure used to describe sensors

This handler was intentionally constructed similarly to the IPMI bus handler. This allows to largely reuse this code. Obviously nothing is perfect and there were also some
3.3. The microcontroller firmware design

changes required. The main difference is based on the fact that in this case the handler has to support only states possible for a master device. As previously, the diagnostic I2C bus interrupt handler takes specified actions depending on the I2C controller status. All the possible states for the master device are described in the microcontroller user manual \cite{35}. Obviously the scenario of the interactions was planned in a way that ensures data read. If an error occurs during communication process, the handler stops reading from this sensor at this cycle and will retry in the next cycle. After all measurements actualization is done, information are send to the FPGA device. Described dependences are shown in Figure 3.8.

![Diagram](image)

Figure 3.8: Diagnostic service states graph

The diagnostics data are sent to a block RAM (Random Access Memory) memory, implemented in the FPGA device. For communication purposes between the FPGA device and the microcontroller, an SPI interface was used. The microcontroller is a master device and it initializes connection each time. Data is organised in the 32 bits wide cells in a planned way. The RAM memory map is presented in Table 3.6. Obviously, first of all there are cells allocated for the data from the AFC board sensors. Moreover, there are also a few fields for unique AFC card ID, AMC slot number, IPMI I2C slave address and data valid indicator. Transferred data are verified by the microcontroller each time, by simple readback operation and comparison with the original information. During the data
transfer or in case of negative verification the data valid indicator is reset and of course it is set otherwise.

<table>
<thead>
<tr>
<th>Field number (hexadecimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0x03</td>
<td>unique AFC card ID number (16 bytes)</td>
</tr>
<tr>
<td>0x04</td>
<td>IPMB address (16 bits, MSB) and AMC slot number (16 bits, LSB)</td>
</tr>
<tr>
<td>0x05</td>
<td>data valid indicator</td>
</tr>
<tr>
<td>0x06 - 0xFF</td>
<td>sensor data fields, sensor identification number (8 bits, MSB) and measured value (24 bits, LSB) NOTICE: Sensor IDs are simple following numbers beginning with the value one. Order is the same as in Table 3.3</td>
</tr>
</tbody>
</table>

Table 3.6: FPGA diagnostic memory map

### 3.3.3 Miscellaneous tasks

The microcontroller also has a few tasks to do that are worth to mention but they cannot be classified neither to the IPMI nor to the diagnostic part of firmware. These could be rather called management tasks and are also important elements of this system, because they are providing power and clock signals.

The first of them is the clock management. On the AFC board there are a few clock sources available. Some of them, such as the fabric clock or the telecommunication clocks are generated by the MCH module and are delivered throughout the backplane board. Clock signals can be also delivered from the FMC card. At last, there are two VCOs (Voltage Control Oscillators) on the AFC board as well. One of them has fixed frequency but second one is programmable throughout the I2C bus by the FPGA device. All these clock signals are delivered directly to the ADN4604ASVZ cross point switch.

"The ADN4604 is a 16 x 16, buffered, asynchronous crosspoint switch that provides input equalization, output preemphasis and output level programming capabilities. The receivers integrate an equalizer that is optimized to compensate for typical backplane losses. The switch supports multicast and broadcast operation, allowing the ADN4604 to work in redundancy and port-replication applications. The part offers extensively programmable output levels and preemphasis settings." [36]
All the cross point switch parameters are controlled by the microcontroller firmware also through the I2C bus. Output signals are further delivered to all important system elements e.g. RAM memory, PCI-E controller and so on. Obviously this part of system is extremely important because proper clock distribution is essential to ensure correct operation of all sequential components. The cross point switch handle is not implemented with usage of the I2C bus interrupt handler as it was in the case of diagnostic sensors. Clock signals are rather set once after system start and then are required to be stable. Therefore, it is allowed to use blocking transfer type in order to configure it.

Power management is not very complex and it boils down to a simple microcontroller output ports controlling. According to a pin output state a proper converter is turned on or off. By default in this application all available voltage levels are required, therefore all voltage converters are switched on after backplane 12 V power comes on.

3.4 The FMC HPC tester

One of the extension cards is the FMC-HPC carrier tester (FMC High Pin Count Carrier Tester). It is an unusual card, because it does not extend the functionality of the AFC carrier card. As the name suggests it is designed for testing of carrier cards. It has to allow to verify correctness of all connections between the FPGA and the FMC HPC connector. Depending on the connection type there are checked: voltage level, electrical connection and layout quality (for fast giga-bit links).

The simplest way to explain the idea of this card is to present it on the block diagram. Thus in Figure 3.9 such a diagram was placed. This card is made as a standard single width FMC board with the high pin count connector provided. The I2C bus is used for diagnostic and management purposes. There are a few parameters to verify and to set.

We start a description from the voltage levels test method explanation. As it was already several times mentioned in this document, there are a few different voltages levels available in the FMC standard. All of these are connected to the ADC converter located on this tester. There are four independent sources of power provided by a carrier to a mezzanine card. Thus four channels of the ADC converter are required.

Afterwards, as we also already know there is about 160 general purposes user defined pins. All of these pins are treated as single ended signals in this case and are connected to the I2C input/output ports expanders. These allows to set or read a status of any individual line. Hereby, an electrical correctness of these lines connections is verified. A special test pattern will be designed for these purpose. It will be generated from a software application.
Thus, there is only verification of giga-bit links and clock signals remaining. In this case a slightly more sophisticated method is used. The FMC standard defines direction of these lines. Thus, all of these signals defined as the input signals for mezzanine module were connected to inputs of giga-bit switch. And also by analogy, output lines to the output ports of the same switch. In this way, a configurable loop-back was designed. Besides these FMC signals, there is also an internally produced clock signal source provided to this switch. An operator is able to verify as well an electrical connections as a layout quality, by sending and receiving some test patterns. There is also an external connector with one differential pair of transmission and receiving lines in order to build loop-backs with another tester module or to generate test patterns from the outside.

Figure 3.9: The FMC HPC tester block diagram
Chapter 4

The design implementation

This chapter describes all the important elements of the implementation process. There are descriptions of both hardware and firmware designs. Thus, it contains the description of the AFC board design as well as firmware design for the trigger card developed for the FPGA in the VHDL language. Moreover, the description of firmware designed for the microcontroller is also included. At last there is also the FMC-HPC carrier tester module presented.

4.1 The AFC card hardware design

The AFC card has been designed in the Altium Designer CAD tool. In this section the description of all blocks is presented.

4.1.1 The FPGA circuit

The Xilinx Artix-7 XC7A200T in the FFG1156 package is used. For both FMC connectors eight FPGA banks have been used to connect them. Those are the following banks: bank 16, bank 34, bank 35, bank 36 for the FMC 1 and bank 12, bank 13, bank 14, bank 15 for the FMC 2. In both cases signals were grouped in regards to the FMC specification and then connected to banks in these groups. Thanks to this all signals from the same group are in the same clock region of the FPGA. Hereby it decreases probability of problems with for example various signals propagation times caused by different lengths of lines. Moreover, the VREF (reference voltage) lines from the FMC connectors have been connected to the VREF pins from these FPGA blocks. Hereby, all signals and banks are working at the same electrical standard.

Remaining giga-bit lines have been connected to the giga-bit ports of the FPGA device. For each FMC connector four links (receiver and transmitter) were connected. Further
eight links were used as the PCI-E links. These links have been connected directly to the AMC edge connector at the other end (two FAT-PIPES from the AMC connector). All giga-bit lines have been AC coupled. As it was mentioned before, eight links were used for the PCI-E bus. However, only four of these are connected directly from the GTP transceivers to the AMC port called FAT-PIPE 2. The remaining four may also perform alternative function. Namely this is dependent on which capacitors are mounted (specific lines are connected or disconnected). Thus, these lines could be connected as well to the FAT-PIPE 1 as to the ports from 1 to 4 of the AMC connector. Port 1 and 2 are used for the White Rabbit time synchronization protocol implementation. Whereas ports 3 and 4 are connected to two SATA (Serial Advanced Technology Attachment) connectors. Hereby, it is possible to connect some external storage drive to the AFC card.

Banks 32 and 33 of the FPGA device were used to connect the SDRAM memory and M-LVDS lines in a following way. Thus, all address, command, clock and control lines for the external memory have been connected to the bank 32. Whereas, data lines and data strobes have been connected to the bank 33. Four Micron memory chips were used, model MT41J512M8RA-125:D [37]. Each memory chip has capacity of 512 MB, so as a result the memory capacity of each AFC card is 2 GB in total. The fly-by architecture was used to connect memory chips, which means that addresses and control lines are common for each memory (are connected in series) and also that lines are terminated at the lines end. The data lines are separate for each chip (parallel connection) and termination in the FPGA device is used. Remaining free lines from the bank 32 have been used for the M-LVDS, time synchronization signals. A physical layer for M-LVDS signals have been provided by the M-LVDS lines transceivers - SN65MLVD040RGT [38].

Moreover, there are also other miscellaneous signals that were also connected to the FPGA. Obviously the JTAG interface for a device configuration and debugging. All the JTAG lines has been connected to the backplane dedicated port throughout the AMC connector and also to an internal connector placed on the AFC board. However, because this interface is used both to the FPGA device and the FMC modules, the addressable JTAG bridge was applied. It is SCANSTA111 [39] integrated chip. It has one JTAG input port and three JTAG output ports, so it fits perfectly. To the input port signals from the AMC and internal connector were connected. Whereas, one of the output ports was used to the FPGA and other two ports to each FMC connectors. Address lines (of this switch) are controllable by the microcontroller ports. Moreover, there are also 128 Mbit of flash memory used (M25P128), with the SPI interface, to store FPGA configuration files.
4.1.2 The microcontroller application

The LPC1764FBD100 microcontroller with the Cortex-M3 core was selected. Nevertheless there is also a place for alternatively mounted ATxmega128A1U-AU. However, let us focus on the LPC1764 controller, because it is used by default in this design and the entire firmware design described further is dedicated for this controller.

Thus, for the IPMI interface as it was mentioned earlier there are a few lines required. There are four address lines to determine the geographical address in a uTCA crate. Three of these lines are used to read the address bits (GA0, GA1, GA2) and one (P1) to set or not to set pull-up for these three previous.

The process of determining an address is defined by the uTCA specification and it proceeds in two steps. The state of these three address lines is read in both cases when P1 is set high and when it is set low. Then, the proper address is set, depending on the result.

Furthermore there is the I2C interface required for communication purposes. A dedicated hardware controller is used for this called I2C 0. This bus is not shared with other components, it is used only for the IPMI interface.

Moreover, there are also three general purpose lines to manage led diodes states. Two of these diodes are required by the uTCA specification. These are as follows: "blue led" - which is used to provide feedback to an user of the Hot Swap state and "Led 1" - which indicates system failures. The remaining led diode is optional and it does not have any function defined by the specification. In this system it is a green led diode used to indicate card normal operation.

Afterwards, one of the requirements is to provide module handle switch on the front panel. This switch state is also monitored by an ordinary general purpose microcontroller pin. There is also a signal called ENABLE for the microcontroller, which comes from the Shelf Manger.

As it was already mentioned the microcontroller not only provides the IPMI interface, but it also performs other tasks. One of these is enabling and disabling power converters available on the AFC card. In this case it is provided by a simple setting or resetting of the appropriate general purpose I/O pin of the microcontroller.

Besides it also adjusts the V_ADJ voltage via digitally controlled variable resistor (AD8402ARZ1). It is controllable throughout the SPI interface. The SPI controller, implemented in the microcontroller, called the SPI 0 is used for this purpose.

Besides, there is yet another SPI interface in order to establish connection between the microcontroller and the FPGA device. Moreover there are also two I2C buses lines connected to each of the FMC connectors.
Furthermore, for the diagnostic purposes another I2C bus was used to communicate with all digital sensors placed on the AFC card. The serial port UART 3 is used as transmission channel for the diagnostic console.

There are also already mentioned lines form the JTAG bridge. These are the six address lines and one reset line. All of these are connected to general purpose pins. Finally, there are a few lines connected to the FPGA flash memory (SPI interface) and signals such as DONE which indicate if the FPGA device is already configured or not.

### 4.1.3 Power supply

In accordance with the assumptions made, a few voltage converters were used to provide proper voltage levels and current efficiency for all devices. All of these converters have enable port which allows to turn them on or off by the microcontroller. Thus, following devices were used:

- **1.0 V** - The TPS53353 [41] integrated chip was used. All of external components parameters were calculated from equations specified in the documentation. Thus, output voltage divider was set to produce 1 V and up to 14 A with switching frequency of 500 kHz. A typical application circuit was used.

- **1.2 V** - The NCP3170 [42] converter was used in a typical application circuit. Current draw was calculated for 2 A at maximum and the output divider was set for 1.2 V. The switching frequency is fixed for this integrated circuit and it is about 500 kHz.

- **1.5 V** - The TPS53126 [43] integrated DC/DC converter was used. It contains two channels. In order to produce 1.5 V level the channel 1 was used. The switching frequency was set to 350 kHz, over current limit is about 10 A, because it is mainly limited by external components. Obviously the output voltage has been set for 1.5 V.

- **1.8 V** - Once again the NCP3170 was used in a typical application circuit. Current draw was calculated for about 2.8 A at maximum and output divider was set for 1.8 V.

- **2.5 V (VADJ)** - The TPS53126 converter was used once more. It fits perfectly to these applications because 2 separate channels are required for both FMC connectors. In this case the output voltage level has to be adjustable in range from 1.8 V to 3.3 V and should be settable to 0 V to disable it. Therefore in output voltage resistor dividers digital potentiometers were used. Thus, to be more precisely the AD8402ARZ1 [40] chip was used. It is two channel digital potentiometer with
the SPI interface. It was connected to the microcontroller ports through this interface. As regards to remaining parameters, the switching frequency had been set for 350 kHz and the maximum output current is about 10 A.

- 3.3 V - In this case two DC/DC converters were used. As it was mentioned this voltage level is required by the FMC connectors and other miscellaneous components on the AFC board. Therefore, in order to supply power to the components placed on the AFC board, remaining channel of the TPS53126 converter was used (second channel, first one was used to produce 1.5 V). The other two power channels were also provided by the TPS53126 converter.

### 4.1.4 The clock distribution unit

The clock management unit is based on the fast digital switch ADN4604 [36]. It allows to connect up to sixteen differential signal sources and up to sixteen differential outputs. Thus, there were connected signals from the backplane board. These are five clock signals called telecommunication clock and one fabric clock. These signals allow to synchronize a receiver placed on the AFC card with other signal source placed in the uTCA crate.

There are also four clock signals from each of the FMC connectors. Four of these signals are defined as input clock lines. However, another four are bidirectional signals. Therefore these signals were also connected to output lines of this switch, thereby allowing use them also as output lines. The LVDS lines repeaters were used (FIN1101K8X [44]). These repeaters could be enabled or disabled, it depends on the status of the corresponding direction lines. Thus, if any line is used as output, the repeater is enabled and disabled otherwise.

Remaining three signals are coming from oscillators internally placed on the AFC card. Four different clock sources were used, according to the specification of the White Rabbit [15] time synchronization protocol. Nowadays it is the most accurate protocol available. It provides accuracy of sub-nanoseconds. As it was mentioned in order to provide a possibility to implement the White Rabbit node on the AFC card, four clock signals were required.

Thus, there is the VCTCXO (Voltage-Controlled Temperature Compensation Crystal Oscillator - part number VM53S3) which provides frequency of 25 MHz. It is voltage-controlled oscillator, thus there was also source of voltage required in order to control it. The AD5662 DAC (Digital To Analogue) converter was used. This converter is controllable by the FPGA device through the SPI interface. Signal from this oscillator was connected to a low-jitter frequency synthesizer (CDCM61004). The output frequency from
this synthesizer was set to 125 MHz and the LVDS output was chosen. This signal was connected to the clock switch input.

The another clock signal is also required by the White Rabbit protocol and is coming from the VCXO (Voltage-Controlled Crystal Oscillator). The LF VCXO026156 was used to generate frequency of 20 MHz. It is also controllable by the AD5662 DAC converter. This signal is also connected to the clock switch input.

There is yet another clock source remaining. It is a programmable Si570 XO (Crystal Oscillator). The output frequency may be selected in range form 10 MHz to 280 MHz, it is under control of the FPGA device through the I2C bus. This signal may be used for general purposes and it also is connected to the clock switch device.

All signal sources were already described, thus there are still possible receivers to discuss. As it was already mentioned, four of output signals are connected to the FMC bidirectional clock lines. Moreover there are also four telecommunication clock signals which are transmitted through the backplane board. These signals could be both sources or receivers. Afterwards, there is a clock signal connected to the SDRAM memory interface bank in the FPGA device. Moreover, five remaining output clock signals, were connected to the giga-bit receivers.

### 4.1.5 Fast serial links for data transmission

Selected assembly variant allows to use fast gigabit links between the FPGA and both uRTM and FMC cards. Obviously due to limited number of gigabit transceivers in the FPGA device there are only a few links of each connected. However it allows to connect both card types. In Figure 4.1 there is an example of uRTM module with 8 SFP cages connected to the AFC carrier. This card can be used for Gigabit Ethernet implementation or any other protocol with use of SFP transceivers.

### 4.1.6 Diagnostic sensors

As regards to the diagnostics of the AFC board a few sensors were placed. All of these sensors are connected through the I2C bus to the microcontroller. There are five temperature sensors. Four of these sensors are the LM75. These were located at the critical points of this board, in the neighbourhood of: voltage converters, both FMC modules and RAM memory integrated chips. There is also another temperature sensor for the FPGA device temperature reading. This sensor (MAX6642ATT90) is able to read temperature directly from the FPGA thorough dedicated lines.

Moreover there are also voltage and current draw monitors. These integrated circuits
are able to measure state of the following FMC lines (of both FMC connectors): 12 V, 3.3 V and V_ADJ. To measure the power consumption on mentioned power lines, special milli-ohm resistors were used. Six INA220 integrated chips were used for this power monitoring issue. These chips are able to directly measure voltage level and indirectly current draw. Afterwards, there are also two integrated circuits that were not mentioned earlier. Namely, these are: RTC (real-time clock, MCP79410) and EEPROM memory (AT24MAC602). Both of these integrated circuits are connected to the microcontroller throughout the I2C bus. The RTC device was used to provide an optionally possibility to store system-up time of each AFC card. Whereas, the EEPROM memory was used to provide unique identify number (128-bits wide) and also to store informations for the IPMI controller.

4.1.7 The AFC card layout design

The AFC carrier card was designed in the Altium Designer software application. It consists of twelve layers - 6 signal layers and 6 power planes. Layers stack was prepared to provide optimal tracks width and gaps between them, with maintaining an impedance for signal tracks of 50 ohms for single ended tracks and 100 ohms for differential pairs. The most important in this layout design was routing of signals for the DDR memory chips, FMC connectors (mainly giga-bit and clock signals) and also PCI-E signals. As it was already mentioned, DDR memory chips were connected in the fly-by topology. Signals were grouped in five groups. Address and control signals represent one group, whereas remaining four groups are the data lines of each memory chip. Thus, length of these signals were aligned (in each group) in order to avoid problems with various delays during the transmission. This was done similarly in case of giga-bit signals of each FMC.
connector and the PCI-E lines.

The shape of this carrier is constrained by the uTCA standard requirements. Obviously, this also had impact on the AMC connector placement. Moreover, placement of the FMC connectors is also forced by a mechanical issue. As a result, placement of all components is a combination of these specifications and logical arrangement in a way that allows to route signals in the simplest way. Thus, all DC/DC converters and M-LVDS transceivers were placed next to the AMC edge connector. The FPGA device was placed in the middle of the board. All DDR memory chips were placed as close as possible to the FPGA in order to ensure short signal paths. All remaining components were placed on the bottom side of this board.

The AFC board was designed and assembled. In Figure 4.2 there is the top side photography of this board.

Figure 4.2: Top side of the AFC v.3.1 card
4.2 Triggering cards firmware design

In accordance with the assumptions made during developing the concept, the FMC_DIO-32CH card was used (in Figure 4.3).

![Figure 4.3: The FMC_DIO 32CH card](image)

The wbgen2 application was used to generate the Wisbone slave component in order to handle the FMC_DIO-32CH card. The following registers were implemented [4.4]. All of these are 32-bits wide.

<table>
<thead>
<tr>
<th>H/W Address</th>
<th>Type</th>
<th>Name</th>
<th>VHDL/Verilog prefix</th>
<th>C prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>REG</td>
<td>FMC Status</td>
<td>wb_fmc_dio_32ch_csr_fmc_status</td>
<td>FMC_STATUS</td>
</tr>
<tr>
<td>0x1</td>
<td>REG</td>
<td>Control register</td>
<td>wb_fmc_dio_32ch_csr_dio_ctrl</td>
<td>DIO_CTRL</td>
</tr>
<tr>
<td>0x2</td>
<td>REG</td>
<td>Output register</td>
<td>wb_fmc_dio_32ch_csr_output_reg</td>
<td>OUTPUT_REG</td>
</tr>
<tr>
<td>0x3</td>
<td>REG</td>
<td>Input register</td>
<td>wb_fmc_dio_32ch_csr_input_reg</td>
<td>INPUT_REG</td>
</tr>
</tbody>
</table>

![Figure 4.4: The FMC_DIO 32CH Wisbone registers](image)

The FMC_STATUS register contains following fields:

- FMC card present (1-bit) - indicates if card is attached to the FMC slot
- Power good (1-bit) - the FMC power good line, not used in this case
- Patch connected (1-bit) - indicates if external is properly connected to the module
- FMC temperature alert (1-bit) - over-temperature alert from the temperature sensor
- LED1 state (1-bit) - indicates if led is on or off
- Firmware identification(27-bits) - version identification, it is a simple date (20130923)

The DIO_CTRL register contains following fields:
- Lines direction select (4-bits)
- Lines output enable (4-bits)
- Reserved fields(24-bits)

The OUTPUT_REG register has only one field:
- Output lines state (32-bits) - does not affect lines when direction is selected as input or if outputs are disabled

The INPUT_REG register also has only one field:
- Input lines state (32-bits) - always indicate current lines state, except when lines are disabled

Besides the above described component there was also the Wishbone slave to the I2C bus bridge used. It is a little bit modified version as compared to the OpenCores version. This change relates to the data buffering before the beginning of a communication. This helped to solve problem with timeout on the I2C bus generated by certain integrated chips.

Moreover, there are also Wishbone master component and address decoder. The wishbone master is used for communication between the FCS software application (on the PC computer) and all slave components. The address decoder block is used for addressing slave components on the Wishbone bus.

A driver for the FCS software was also prepared. It contains methods for: temperature reading, setting direction and state of lines and eeprom handling. These methods rely on read and write operations of the relevant Wishbone registers.
4.3 The microcontroller firmware implementation

Firmware for the microcontroller was designed in the LPCXpresso v.5.2.4 software development platform dedicated for NXP LPC microcontrollers. As it was previously mentioned the LPC1764 chip was used, thus firmware was prepared for this controller. The entire application is written in the C programming language. Following libraries were used: CMSIS v.2.0 [46] (for Cortex-M3 core) and also an open source peripheral drivers from CooCox [47] project web site.

At the beginning whole workspace with mentioned libraries included, was prepared. Afterwards initial configuration for the microcontroller was prepared. With usage of the CMSIS configuration wizard application [48], several parameters were configured. A main microcontroller PLL (Phase locked loop) was set to generate a clock frequency of 100 MHz from an external input 8 MHz reference clock signal. Moreover all required peripherals were turned on. The flash memory latency was set to 5 clock cycles with is default value for selected clock speed. After all necessary parameters were configured properly, an implementation of the IPMI controller functions has been started.

Firmware implements "normal" functions called from a main loop and also functions called from a hardware interrupt handlers. In the main function there are localized methods used mostly for initialization purposes and also some of the IPMI services. Whereas hardware interrupt handlers are used for functions which have to be executed immediately. There are following sources of interrupts: IPMI I2C bus controller, timer 0 (used for functions executed periodically), hot swap switch, debug and diagnostic console, diagnostics I2C bus and timer. Priorities have been established in the following order (starting with the highest): IPMI I2C bus, timer 0, hot swap, diagnostics I2C, diagnostics timer 3, serial console.

The timer 0 was used as general purpose periodical interrupts source. It was decided that the shortest period will last 100 usec, thus this timer has been set to generate interrupts after each 100 usec. From body of this handler various functions are called. This timer could not be classified neither to the IPMI service not to the diagnostics and management service, because it is used in both cases. Remaining sources of interrupts are described later when discussing their use.

4.3.1 The IPMI controller implementation

As is was already mentioned a main communication channel for this interface is provided by the I2C bus. The I2C-0 bus is used with a clock frequency of 100 kHz. In order to handle this bus several methods were implemented. The most important is the I2C bus
interrupt handler. The microcontroller I2C bus controller is by default a slave device, however in order to send response to the Shelf Manager it could become a master device. Therefore the I2C bus interrupt handler contains interactions for states possible in both cases. Thus, there are the following states implemented:

- in the master mode:
  - START_SENT - start condition has been transmitted, a slave device address (with write or read bit) will be transmitted in the next step
  - SLAVE_WRITE_NOT_ACK - a slave address with write bit has been transmitted, but the controller did not receive acknowledge, controller will enter a slave listen mode, the IPMI service will try to retransmit this message unless time out or maximum retransmissions attempts were exceeded
  - REPEATED_START_SENT - it is not supported state, the controller will enter in the slave listen mode
  - ARBITRATION_LOST - this means that controller has lost arbitration at the I2C bus, controller will enter slave listen mode and will wait a while for a retransmission
  - SLAVE_WRITE_ACK/DATA_SENT_ACK - controller sent address or data and received acknowledge, if there will be more data to send it will send it, otherwise it will generate stop condition and enter slave listen mode
  - DATA_SENT_NOT_ACK - data has been send but acknowledge was not received, controller will enter slave listen mode

- in the slave mode:
  - SLAVE+W_REC_ACK - the controller received own slave address with write bit and has returned the acknowledge, first data byte will be expected in the next state
  - SLAVE+R_REC_ACK - own slave address with read byte has been received, it is not supported to read data in the slave mode, thus controller will respond with the 0xFF value
  - SLAVE_DATA_SENT_ACK - as previous
  - SLAVE_DATA_REC_ACK - data has been received and acknowledge was transmitted,
- `ARB_LOST_SLAVE+W_REC_ACK` - arbitration lost during receiving data in the slave mode, controller will wait a while and then will return to the slave listen mode.

- `SLAVE_DATA_REC_NOT_ACK` - in this case the acknowledge was not transmitted, it means that master device will have to end transmission by generating stop condition.

In other cases the I2C bus controller will enter the slave listen mode.

In the main loop there is a function implemented which constantly checks if there are any received messages or those that are waiting to be sent. All received messages are verified if checksums of header and body are correct and also if received number of bytes is matching a value received in a message header. In case of positive verification a message is processed by the IPMI messages parser. All messages are specified in the IPMI documentation and all further interactions are taken on the principles defined by this document. If a message is not a proper one it is deleted.

In the case when there is a message waiting in the queue to be sent, the controller checks if it could enter to master transmitter mode. If it is able to enter this mode it loads a message to the transmitting buffer and generates start condition on the I2C bus. Afterwards the I2C bus interrupt handler proceeds this message. If the transmission is successful message is deleted from the queue, otherwise it is retransmitted (unless the retransmission limit has been reached).

Furthermore, several auxiliary methods were implemented for variable purposes. These are prepared for in example: calculating checksums, extracting specific fields from message (length, checksum, command code, etc.). Obviously, there is also the function that determines controller slave address.

The IPMI controller has to implement information called as the FRU. Following fields were implemented. At the beginning there is the common header which is mandatory. It contains information about offsets to other fields. There is also field called device id. It informs the Shelf Manager about supported IPMI version (v2.0 in this case) and also that device supports events on the IPMI bus and has sensors.

The most important is the multi record field. In this case it contains the point to point connectivity record and power requirements record. The first one informs the Shelf Manager that the AFC card have PCI-E bus provided. On the basis of submitted informations the Shelf Manager configures appropriate links as PCI-E links. Whereas, by using the power record the AFC card informs that require maximum current draw of 6 A (72 W). All of these areas of the FRU have been grouped in the data structures. All fields
names and lengths combined in these structures were determined by the specification of the FRU.

There are three commands to read or write the FRU. The Shelf Manager is capable to read the FRU by using GET_DEVICE_ID and READ_FRU_DATA functions. The first one is only used to read device ID field, whereas the second one allows to read whole FRU. The Shelf Manager needs to send a request with an offset value and an expected length of response in order to use READ_FRU_DATA function. There is also one function remaining called WRITE_FRU_DATA, which allows to write data to the FRU from the Shelf Manager (ie. for reconfiguration).

The hot swap switch as was assumed was implemented with usage of hardware interrupts. Each time when controller detects any edge, it generates interrupt from external source. This external source is obviously the hot swap port which is located at port 2 pin 13. This interrupt handler only sets a bit flag. This flag indicates that there is suspicion of the switch state change. Afterwards, this flag is noticed by the periodically running timer 0. This timer causes a little delay of about 10 ms. After this period of time the timer 0 interrupt handler checks whether the change really took place. If so it sets another flag for a function called form the main loop. This function prepares the IPMI message with the current state of switch and places it in the transmitting queue. Afterwards, a proper interactions are taken by the I2C bus controller and at last by the Shelf Manager.

The specialized data structure called the SDR has been implemented for the IPMI sensors. It consists of fields such these defined in the SDR specification [33]. It was decided that for the Shelf Manager only the hot swap switch and all temperature sensors will be available. Therefore, these sensors were described with the SDR structure.

Moreover, the IPMI diagnostic service has been implemented. It could be called from the main loop. This function checks if there are some changes that have to be processed (ie. hot swap closed, sensors update). Afterwards it makes further interactions. In case of diagnostics sensors this service compares measured value with the thresholds values. If it is necessary it also generates relevant information to the Shelf Manager (ie. exceeded threshold).

In case of led diodes there was also dedicated data structure prepared. It is quite simple because mainly it contains state of each led diode (period of blinking, etc.). The Shelf Manager is also capable to read and set parameters for these diodes states. The handling function is called from the main periodically timer (timer 0). Each time this function is executed it checks whether the indicated state is relevant. If no the state is changed.
It is also worth to mention the payload power management service. It was mainly implemented to monitor power lines from the backplane board. It is able to inform user when power comes on or off and also to take appropriate actions (such as turning on or off voltage converters). It also allows to perform tasks such as restart procedure, power sequencing and so on.

The last service are the debugger and diagnostic serial console, which can be used for various purposes. Nevertheless they are described in this section because in this firmware they mainly prints messages that are related to the IPMI service. Thus, following informations are possible: all received messages codes, informations about power and other events such as the hot swap handle state and others. The serial UART 3 (Universal Asynchronous Receiver and Transmitter) has been set in a following way: baudrate 115200 bps (bits per second), 8 data bits, 1 stop bits, no parity check and fifo empty signals as an interrupt source. As it was already mentioned the lowest level hardware interrupt was used for this console. This interrupt is generated each time when the hardware fifo queue becomes empty. Because this fifo queue is rather small (16 bytes deep) an auxiliary software circular buffer was also used. It has capacity of 256 bytes. The data to transmit is at first stored in the fifo and when fifo is not empty also in this circular buffer. The serial interface controller gets data to transmit from the fifo queue and when fifo becomes empty it generates an interrupt. The interrupt handler is loading new part of data to the fifo queue and these are transmitted.

4.3.2 The diagnostic service implementation

The diagnostic service was implemented with usage of the hardware interrupts. It has a few tasks to perform. Obviously, first of all it has to read data from sensors, afterwards it is also required to transmit these data to a FPGA and also to informs the IPMI diagnostics service that new data is available and could be updated. There were used two interrupt sources. The I2C 1 bus handler is used to communicate with sensors. Whereas the timer 3 handler initializes each sensor reading.

The implementation of this service consists of a few methods. Typically an operation of this service begins with the initialization process. During this process the following elements of this service are set: I2C bus, SPI interface, timer 3 and also the diagnostic structure is filled in.

Thus the I2C 1 bus is used with following configuration: the clock frequency is set to 400 kHz and the controller is switched to the interrupt mode. As it was already mentioned the interrupt handler was build with usage of the IPMI I2C bus handler. It has been a little bit modified. First of all, states related to the slave mode have been deleted, because
in this case the microcontroller is configured permanently as master device while all sensors are the slaves. Obviously the master states also needed to be a little bit modified. Nevertheless, in a large degree the idea and the code have been reused.

The SPI interface has been initialised with a clock frequency of 10 MHz. Because of the fact that transmission through this channel is quite rare, fast and also is not able to lead to the system suspension, it was decided not to use hardware interrupt service. Thus, after all sensors readings are updated, the microcontroller checks if the FPGA device is configured. If so it sends all diagnostic data and afterwards it reads it and verifies if there are no errors. Moreover it also sends information about verification results to the FPGA device. In the FPGA as it was assumed the memory block (dual port RAM - 256 cells x 32-bits) is implemented. Access to the memory is possible through the SPI interface (from the microcontroller) and the Wishbone bus (for further transmission process).

There is still timer 3, which remains to discuss. It is used to generate interrupts after every one second, it could be said that it is a kind of scheduler. Its interrupt handler has two tasks. At first it initiates communication process with each of the sensors. When all sensors are updated it initiates communication process with the FPGA device and also sets the flag for the IPMI diagnostic service to inform that new values are available.

There is also one more thing that is happening during the initialization process. Thus, the configuration process of the diagnostic structure. This structure was placed at code listing above.

```
typedef struct {
    uint32_t cardID [4];
    uint32_t AMC_NR;
    uint8_t sensorPtr;
    sensorStruct_T sensor [BOARD_SENSORS];
} SystemDiagnosticStruct_T;

typedef struct {
    float Val1;
    uint8_t rawData[MAX_FIELDS_TO_READ];
    uint8_t regAddr[MAX_FIELDS_TO_SEND];
    uint8_t bytesToSend;
    uint8_t bytesToReadFromReg[MAX_FIELDS_TO_SEND];
    ptrReadOutVal ReadOutFunc;
    uint8_t i2c_addr;
} sensorStruct_T;
```

Thus, there are a few fields to initialise. At first there is an unique AFC card identification number. It is 16 bytes wide. This number is read from the EEPROM memory by the microcontroller. Afterwards there is also the AMC card slot number in the uTCA
crate. The sensor pointer field indicates from which sensor the controller has to begin sensor scanning procedure (value 0 is entered here). This field also informs which sensor has been refreshed recently. Then there is the table of sensors data structures. Each of sensors located on the AFC board have been described with usage of this structure. The first value in this structure (Val1) is calculated from the raw data read from a sensor registers. Calculation of this value is performed by a function prepared for this purpose. Each sensor model has its own calculating function, thus there is the ReadOutFunc field to store a pointer to that function. All remaining parameters are required by the communication process. These fields contain addresses to device and to registers and numbers of bytes to read from these registers. In Table 4.1 is shown how this structure has been configured.

<table>
<thead>
<tr>
<th>Field name</th>
<th>MAX6642</th>
<th>LM75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val1</td>
<td>calculated value</td>
<td></td>
</tr>
<tr>
<td>rawData</td>
<td></td>
<td>read value</td>
</tr>
<tr>
<td>regAddr</td>
<td>external temp. (0x01); fractional part (0x10)</td>
<td>temp. register (0x00)</td>
</tr>
<tr>
<td>bytesToSend</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>bytesToReadFromReg</td>
<td>1; 1</td>
<td>2</td>
</tr>
<tr>
<td>ReadOutFunc</td>
<td>readFPGATemp(uint8_t *RawPtr)</td>
<td>get_LM_temp(uint8_t *RawPtr)</td>
</tr>
<tr>
<td>i2c_addr</td>
<td>0x48</td>
<td>0x4C; 0x4D; 0x4E; 0x4F</td>
</tr>
<tr>
<td>Val1</td>
<td>INA220 (voltage)</td>
<td>INA220 (current)</td>
</tr>
<tr>
<td>rawData</td>
<td></td>
<td>read value</td>
</tr>
<tr>
<td>regAddr</td>
<td>bus voltage (0x02)</td>
<td>shunt voltage (0x01)</td>
</tr>
<tr>
<td>bytesToSend</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bytesToReadFromReg</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ReadOutFunc</td>
<td>get_INNA_volt(uint8_t *RawPtr)</td>
<td>get_INNA_curr(uint8_t *RawPtr)</td>
</tr>
<tr>
<td>i2c_addr</td>
<td>0x40; 0x41; 0x42; 0x43; 0x44; 0x45</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Sensors structure

It should be explained that the INA220 sensors have been classified in software as a two separate sensors (voltage and current), despite the fact that it is the same sensor. It
is so because of two reasons. Firstly, because it was more convenient and easier to adjust to this data structure. However, the main reason was the fact that it is not possible to read data from bus voltage and shunt voltage registers at once. Moreover after each read operation a sensor needs a period of time before it is able to be read again. Therefore at first the bus voltage registers from all INA220 sensors are read and afterwards the shunt voltage registers.

4.3.3 The management service implementation

As regards to the microcontroller firmware design there is still one more service to describe. This consist of functions and methods prepared for configuration of the AFC card. Thus, it is called the management service. As it is already described, there are the following blocks: clock distribution unit, JTAG module and power management that required a preliminary configuration. For this purpose several functions have been implemented. This service does not contain interrupt handlers. Since it configures parameters that have to be constant during a single operation period of the AFC card, there were functions executed once after each power up event implemented. These are called from the main loop.

For the power management issue there was one function required. It is necessary to adjust V_ADJ voltage level with usage of the digital potentiometers. This configuration is performed by the microcontroller through the SPI 0 interface. Both V_ADJ channels have been set to produce 2.5 V.

In order to route clock signals the I2C 1 bus has been used. This bus is also used by the diagnostics sensors (using hardware interrupts). Thereby the clock configuration procedure has to switch settings for the duration of the configuration procedure. Obviously mainly it was about disabling interrupts. Thus, the configuration data is send in the blocking mode. In present configuration only two clock sources were used, in the way that is shown in Figure 4.3. The used clock connection switch is equipped in additional termination circuits and also have a few parameters in transmitter and receiver to set. The default values were used and termination circuits were turned off. Further possible changes could be applied after verification of signals quality.

The JTAG switch module has been set into normal operating mode. All of addresses lines were set to value 0. Thereby it will be possible to switch its address from a software application.
4.4 The FMC HPC tester hardware design

The layout and schematics design of this board was prepared in the Altium Designer software. For a communication purposes the I2C bus was used. Thus all used integrated chips also had to have this bus implemented.

For voltages levels supervising, the AD7997 [49] ADC converter was used. It has 8 analogue channels, 10-bits resolution for each of channels. To the analogue inputs, all specified in the FMC standard power lines were routed. At first voltage levels have been adjusted by an ordinary voltage resistors dividers to a proper analogue inputs range.

All of the user defined lines from the FMC connector have been routed to the MCP23017 [50]. These are 16-bits wide input or output ports expanders with serial interface. An operator is able to read or set ports state with their help. Because these user defined lines are tested only in terms of short circuits or open circuits, these were treated as a single ended lines to simplify board layout. There was also a need to use eleven such expanders to connect all required lines. Whereas these expanders have only three ports for setting a hardware address on the I2C bus. Thus it allows to set only eight different addresses. Therefore, in order to avoid addresses conflict, the bus has been extended by the PCA9547 [51] I2C bus multiplexer. The I2C bus organisation with integrated chips addresses has been shown in Figure 4.6. This bus division obviously results from addresses allocation but also from components placement on the PCB. All of the addresses were chosen in a way that prevents conflict regardless of the multiplexer settings.
With accordance to the assumptions the giga-bit lines has been connected to the digital crosspoint switch ADN4604 [36]. This integrated circuit allows to configure connections on these lines freely. It has sixteen differential inputs and sixteen differential outputs, so that is sufficient to provide connections for all signals. With usage of this switch there is a possibility to build some kind of a loop-back for these signals. Hereby it is possible to send some test patterns and then try to receive it. If the data rate is sufficiently large, an operator is able to detect potential problems with layout and obviously with electrical connections.

As regard to the PCB board, it was designed on four layers. It is worth to mention layers stack, because it is not typical. Thus, the most important, because of signal integrity is the bottom layer. All of the giga-bit links have been routed on this layer. These lines were routed in differential pairs topology, maintaining impedance of 100 ohms. Under this layer there is a reference ground plane. This is a uniform plane, which provides return paths for all signals. The other two layers were dedicated for remaining signals. Moreover, there are also power polygons on these layers. The PCB board design top side is shown in Figure 4.7.
Figure 4.7: The FMC-HPC carrier tester top side
Chapter 5

The design verification

This chapter describes the design verification and tests process. It consists of a few stages, which relates to both hardware and software design. During these stages following laboratory instruments were used: digital oscilloscopes, multimeters and power supplies. Moreover, during firmware and software implementation all applications designed for verification were really useful. Among the others these were: embedded debugging tool (included in the LPCXpresso application), ISE Simulator and ChipScope logic analyser. The first one has been used during development of the firmware for the microcontroller. Whereas two remaining have been applied for preparation of firmware for the FPGA device. The ISE simulator was used to simulate modules on the PC computer. Whereas the ChipScope logic analyser allows to read signals from the FPGA device.

5.1 Verification of basic parameters

On this stage all basic parameters have been verified. These are voltages levels and clock signals. Thus these are elements that are necessary for the further operating. In Table 5.2 below the theoretical and measured voltages levels were posted. All of these measurements were made with usage of the SANWA 5000A [52] digital multimeter. As it could be noticed all measured values match their theoretical values.

Afterwards, the verification of all clock signal sources has been performed. For this purpose a digital oscilloscope was used. With usage of this oscilloscope signal amplitudes and frequencies were examined. Moreover it has been also verified whether there are any undesirable effects connected with signal integrity issue, such as: undershoots, overshoots, ringbacks and others. All of these signals were verified positively. In Figure 5.1 an example shape of the 20 MHz clock signal source is shown.
## 5.1. Verification of basic parameters

### Table 5.1: DC/DC converters - voltage levels verification

<table>
<thead>
<tr>
<th>Theoretical voltage [V]</th>
<th>Measured voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.010</td>
</tr>
<tr>
<td>1.2</td>
<td>1.189</td>
</tr>
<tr>
<td>1.5</td>
<td>1.503</td>
</tr>
<tr>
<td>1.8</td>
<td>1.793</td>
</tr>
<tr>
<td>3.3</td>
<td>3.350</td>
</tr>
<tr>
<td><strong>FMC 1</strong></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.558</td>
</tr>
<tr>
<td>3.3</td>
<td>3.362</td>
</tr>
<tr>
<td><strong>FMC 2</strong></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.583</td>
</tr>
<tr>
<td>3.3</td>
<td>3.383</td>
</tr>
</tbody>
</table>

At these stage also configuration of the clock switch has been verified. Also with usage of the digital oscilloscope it has been examined whether appropriate clock sources are connected to appropriate receivers. It turned out that everything works properly. Moreover there were no problems with these signals quality, thus there were no further settings changes made.
5.2 The IPMI controller

At this section the examination process of the IPMI controller was described. The aim of this part of test was to check correctness of the communication between the Shelf Manager and the AFC card controller. Also data and commands interpretation has been verified. At this stage mainly the following tools there were used: embedded LPC microcontroller debugger (for data preview at the microcontroller), serial UART console (for IPMI commands printing etc.), the Shelf Manager web site and console (for data preview at the Shelf Manager).

Obviously, this test were initiated by insertion of the AFC card into the uTCA carrier. The photography of the uTCA carrier with the AFC cards inserted is shown in Figure 5.2.

After the AFC card insertion the Shelf Manager properly added it to the modules population list. Afterwards the hot swap switch has been closed and as a result the AFC carrier correctly went to the active state. The module manager controller also noticed that the power from the backplane board had came and it turned on dc/dc converters. Furthermore it was possible to verify whether the controller is also able to turn off the AFC card. For this purpose the hot swap switch has been opened and as a result the AFC carrier was turned off and went to the inactive state. In Figure 5.3 the modules list screenshot from the Shelf Manager web site were placed. The AFC module was underlined. As it could be noticed it was in active state. Moreover the card name has been read properly - AFC_AMC_8 this means that this is the AFC module and it was inserted into AMC slot number 8.
5.2. The IPMI controller

Afterwards it was possible to check other IPMI services. Thus, there were verified sensors reading and point to point connections. All data from sensors were read properly. These were compared with values stored in the microcontroller. In Figure 5.4 an example part of sensors reading was placed (from the Shelf Manager console). It contains information about the hot swap switch as also FPGA temperature sensor, current value and threshold status. As regards to the point to point connections, these settings were read from the Shelf Manager site and verified whether these are consistent with the assumed. The fact that these were properly chosen was ensured also by a team mate who was engaged in the PCI-E core implementation on the FPGA device. He successfully established a connection and was able to transmit data.

<table>
<thead>
<tr>
<th>Location</th>
<th>Entity ID/Inst</th>
<th>Device Name</th>
<th>Hotswap State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x82, FRU# 00</td>
<td>(0x02, 0x01)</td>
<td>UTCA CARRIER</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 01</td>
<td>(0x72, 0x61)</td>
<td>SH FRU DEV1</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 03</td>
<td>(0x02, 0x61)</td>
<td>UTCA MCH</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 05</td>
<td>(0x01, 0x61)</td>
<td>CCF AM 91X/K1x</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 12</td>
<td>(0x01, 0x69)</td>
<td>APC AMC 8</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 40</td>
<td>(0x1e, 0x61)</td>
<td>VT VT091</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 41</td>
<td>(0x1c, 0x62)</td>
<td>VT VT091</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 50</td>
<td>(0x0a, 0x61)</td>
<td>VT UTC017</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 79</td>
<td>(0x03, 0x60)</td>
<td>TELCO ALARM</td>
<td>M4 (Active)</td>
</tr>
<tr>
<td>0x82, FRU# 80</td>
<td>(0x0b, 0x61)</td>
<td>MCH DA INFO</td>
<td>M4 (Active)</td>
</tr>
</tbody>
</table>

Figure 5.3: The modules population list

Figure 5.4: An example of sensors reading from the Shelf Manager level
5.3 The AFC card diagnostic

At this stage the operation of the diagnostics service was verified. This examination applies to the main diagnostics channel which was referred earlier in this publication. Operating of the communication channel as also sensors reading were checked. At the code listing below there was placed the example diagnostics frame read from the FPGA memory by the FCS PC software through the Whisbone bus and the RS-232 interface. In the header there is a board type field and an unique identification number. There is also the location in the uTCA carrier as also the IPMI I2C bus address. Afterwards there are following diagnostics sensors located with their names, measurements and units.

```xml
<?xml version="1.0" encoding="utf-8"?>
<board type="AFC" unique_id="0000000015150b09534d08d64e0858c6"
position="0008" ipmi_i2c_addr="0080" comm_status="idle" >
<sensor id="01" meas_desc="FPGA/uni2423-/uni2423temp" value="34500" chip="MAX6642"
unit="mC" />
<sensor id="02" meas_desc="FMC1/uni2423-/uni2423temp" value="26000" chip="LM75"
unit="mC" />
<sensor id="03" meas_desc="FMC2/uni2423-/uni2423temp" value="31500" chip="LM75"
unit="mC" />
<sensor id="04" meas_desc="DC/DC/uni2423conv/uni2423temp" value="32000" chip="LM75"
unit="mC" />
<sensor id="05" meas_desc="SDRAM/uni2423temp" value="27000" chip="LM75"
unit="mC" />
<sensor id="06" meas_desc="FMC1/uni2423-12V/curr" value="0" chip="INA220"
unit="mA" />
<sensor id="07" meas_desc="FMC1/uni2423-3V3/curr" value="2" chip="INA220"
unit="mA" />
<sensor id="08" meas_desc="FMC1/adj/curr" value="0" chip="INA220"
unit="mA" />
<sensor id="09" meas_desc="FMC2/uni2423-12V/curr" value="1" chip="INA220"
unit="mA" />
<sensor id="0a" meas_desc="FMC2/uni2423-3V3/curr" value="0" chip="INA220"
unit="mA" />
<sensor id="0b" meas_desc="FMC2/adj/curr" value="-1" chip="INA220"
unit="mA" />
<sensor id="0c" meas_desc="FMC1/uni2423-12V" value="12192" chip="INA220"
unit="mV" />
<sensor id="0d" meas_desc="FMC1/uni2423-3V3" value="3350" chip="INA220"
unit="mV" />
<sensor id="0e" meas_desc="FMC1/adj" value="2558" chip="INA220"
unit="mV" />
<sensor id="0f" meas_desc="FMC2/uni2423-12V" value="12133" chip="INA220"
```
Moreover, the voltage and current measurements were also verified with and without load. Measurements were done using the digital multimeter (SANWA 5000a) and afterwards were compared with the AFC board sensors measurements. The FMC 1 connector worked without load whereas the FMC 2 worked with the FMC-DIO-32ch FMC module. Two cases were examined: when all output lines were disabled (at the FMC-DIO card) and when two buffers outputs were enabled (16 lines). The results of this test are presented in Table 5.2. As it could be noticed voltages levels are constant and correct. Whereas currents draws in case with no load and with disabled outputs oscillate around 0. In the case when two output buffers were enabled the current draws at the 12 V line and the 2.5 V line have raised. These results are consistent with the expected.

<table>
<thead>
<tr>
<th>Measured voltage [V]</th>
<th>Measured current [A]</th>
<th>Voltage (from sensor) [V]</th>
<th>Current (from sensor) [A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMC 1 - with no load</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.559</td>
<td>-</td>
<td>2.557</td>
<td>0.001</td>
</tr>
<tr>
<td>3.356</td>
<td>-</td>
<td>3.353</td>
<td>0.000</td>
</tr>
<tr>
<td>12.255</td>
<td>-</td>
<td>12.180</td>
<td>-0.002</td>
</tr>
<tr>
<td>FMC 2 - buffers outputs were disabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.585</td>
<td>0.005</td>
<td>2.572</td>
<td>0.002</td>
</tr>
<tr>
<td>3.392</td>
<td>0.003</td>
<td>3.380</td>
<td>0.000</td>
</tr>
<tr>
<td>12.135</td>
<td>0.018</td>
<td>12.135</td>
<td>0.021</td>
</tr>
<tr>
<td>FMC 2 - 2 buffers outputs were enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.585</td>
<td>0.032</td>
<td>2.572</td>
<td>0.030</td>
</tr>
<tr>
<td>3.392</td>
<td>0.003</td>
<td>3.380</td>
<td>0.000</td>
</tr>
<tr>
<td>12.120</td>
<td>0.365</td>
<td>12.118</td>
<td>0.359</td>
</tr>
</tbody>
</table>

Table 5.2: INA 220 sensors read tests


5.4 Triggering modules test

In order to verify triggering modules and firmware design the workspace presented in Figure 5.5 was prepared. Two FMC-DIO-32CH modules were connected to the AFC carrier card. The AFC board has been powered up from the power supply. External ports of the FMC-DIO modules have been connected to each other. Thereby forming a kind of loopback. Afterwards the FCS software was used to configure the card number one as an output card and the card number 2 as an input card. The testing procedure consisted off various configurations of the output lines and reading these from the input lines. In that way it has been verified whether all lines are working properly.

![Figure 5.5: Prepared workspace for the FMC-DIO modules tests purposes](image)

At the code listing below there was an example of the FCS console print placed. It shows the following stages. At first the FCS software established a connection with the FPGA device. Afterwards it has been checked whether there is a proper firmware in the FPGA. At last the lines have been set and read. There was also an operation of data write and read to the EEPROM memory performed as also a few temperature readings (from a temperature sensor).
FMC configuration software for the FMC DIO 32CH card AMC Artix7 Version
Author: Bartek Juszczyk

RS232_syscon: Init function - WB Master Component
RS232_syscon: RS-232 interface configuration in progress...
RS232_syscon: Reset function
RS232_syscon: Setting connection speed...
RS232_syscon: Wishbone component reset...
RS232_syscon: RS-232 initialization is done

============================================
Firmware identification
============================================
Firmware ID (FMC1): 1332c64(20130916)
Firmware ID (FMC2): 1332c64(20130916)

============================================
Input/output configuration FMC1
============================================
Output enable FMC1 (active high): f
Lines direction FMC1 (low - output, high - input): 0

============================================
Input/output configuration FMC2
============================================
Output enable FMC2 (active high): f
Lines direction FMC2 (low - output, high - input): f

============================================
Setting output lines
============================================
Output lines have been set to: 50505050

============================================
Reading input lines
============================================
Input lines status: 50505050
All done! All components on the FMC card had been configured and tested!
FMC card is ready to work!

Listing 5.2: An example of FCS software console print messages
5.5 Tests with the FMC125 module

This test was performed to check a few parameters. Because this module is producing high data samples rate it allows to detect potential limitations connected with layout design, FPGA device and also power supply. Obviously it also provides some information about communication between the FCS software and the FPGA device. Especially during communication with external integrated chips through the I2C bus.

The FMC125 is a commercially produced module by the 4DSP company. It has 4 channels of 8 bits analogue to digital converter. Each of these channels could operate in the independent mode with a data rate up to 1.25 Gsps or as one channel with a data rate up to 5 Gsps. Module is compliant with the FMC-HPC standard, thus it is possible to connect it to the AFC card.

At the beginning a few integrated chips had to be configured. There are following integrated chips placed on the FMC125 module: EEPROM memory (24LC02B), temperatures and voltages monitor (ADT7411), I2C to SPI bridge (SC18IS602B), clocks distributor (AD9517), fast ADC converter (EV8AQ160) and CPLD. All these chips, which require an initial configuration (the last three) are hidden behind the I2C bus to the SPI interface bridge. It is quite comfortable solution because all these chips require the same method for read and write operation. Obviously the only difference is different registers addresses and values, but procedure is the same. The discussed architecture is shown in Figure 5.6.

![I2C and SPI architecture on the FMC125 module](image)

Figure 5.6: I2C and SPI architecture on the FMC125 module

For testing purposes a specialized firmware for the FPGA device has been designed. Its architecture is quite similar to the earlier presented overall architecture. It is presented in Figure 5.7. It contains a few main components. These are:

- Wishbone master to RS-232 - It is master wishbone component. It was used for communication purposes between Wishbone slave components and PC computer. On the PC computer once again the FCS application was used.
- Wishbone control registers - It is a core generated with the wbgen2 application. It contains registers with configuration parameters for a data interface. These are in example delay values for input components.

- Wishbone to I2C bus - This core was used for communication with integrated circuits placed on the FMC125 module.

- Data interface - This block contains input logic blocks for each data lines from the ADC converter. It consists of Xilinx primitives such as: input delay blocks for data lines delay adjusting, input double data rate component for data reading on each clock edge and asynchronous FIFO queue for the data clock frequency slowing down.

- Logic analyser - It is also the Xilinx core. It was used in order to display received data.

![Figure 5.7: Firmware architecture for the FMC125 module](image)

After the FMC125 was correctly powered up and all voltage levels were verified with an ordinary digital multimeter, the initial configuration setting was begun. At first temperatures and unique ids of integrated circuits were read in order to verify a communication correctness. Afterwards an internal oscillator was set as a source of clock signal for the clock distribution device. Moreover it was selected no synchronization source and also clock distribution and ADC converter were set in a normal operation mode. All these functions were configured by the CPLD device. Then the clock distribution integrated circuit was configured to route an internal 2.5 GHz oscillator to the ADC converter clock input. And the ADC converter was set in a test mode with increasing counter on the digital outputs. Besides it was configured in four channels mode. This implies that each
Chapter 5. The design verification

Channel uses all sixteen lines for samples and on each clock edge there are two further samples available. In Figure 5.8 the prepared workspace is shown.

As a result of the carried work it was obtained a properly received data samples. In Figure 5.9 a part of received waveform is shown. In relation to this it is possible to conclude that samples are consistent with expected. Based on the last signal at this figure which is a 40 MHz reference clock, it is also possible to count samples rate. Thus, during one period of this clock there are four data frames in each channel. Each frame consists of eight samples. As a result there are 32 samples on each 40 MHz period, so that 1280 Msps in total for each channel. This is not exactly 1.25 Gsps because frame clock and reference clock are not source synchronous signals.

Figure 5.9: An example of received data (test patterns)
Chapter 6

Conclusion

This chapter is a short summary of this document. Developed AFC card meets all assumptions made. Thus, it has reconfigurable clock connections, provides high data throughput and it is the uTCA standard compliant. The device was manufactured, assembled, verified and tested.

This document describes hardware design at the functional level of the device. Moreover, also hardware startup and verification process is included. Furthermore, it contains some of firmware and software solutions used. The design verification chapter describes some of the capabilities and possible applications.

Developed AFC card is already used in a few scientific researches related to particle accelerators and high energy physics. Because of the fact that it allows to connect diverse FMC modules it is very flexible device which can be suited to any needs. Therefore it is difficult to point all possible applications. Some of them (already being developed) are listed below:

- particle accelerators
- GEM (Gas Electron Multiplier) detectors
- Noise Radars
- Software Defined Radios

At present platform is constantly evolving, new extension modules are created. Thus, possible application list is constantly growing. Authors hope that the developed platform will be able to meet variety of even the most stringent requirements for applications in industry and science.
Bibliography


# List of abbreviations

<table>
<thead>
<tr>
<th>A</th>
<th>Analogue to Digital Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analogue to Digital Converter</td>
</tr>
<tr>
<td>AFC</td>
<td>AMC to FMC carrier card</td>
</tr>
<tr>
<td>AMC</td>
<td>Advanced Mezzanine Card</td>
</tr>
<tr>
<td>ATCA</td>
<td>Advanced Telecommunications Computing Architecture</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>A circuit board containing sockets into which other circuit boards can be plugged in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backplane</td>
<td>A circuit board containing sockets into which other circuit boards can be plugged in</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>Compressed Baryonic Matter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBM</td>
<td>Compressed Baryonic Matter</td>
</tr>
<tr>
<td>CERN</td>
<td>European Organisation for Nuclear Research</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>Digital to Analogue Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC</td>
<td>Digital to Analogue Converter</td>
</tr>
<tr>
<td>DDR SRAM</td>
<td>Double Data Rate Synchronous Dynamic Random-Access Memory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E</th>
<th>Electrically Erasable Programmable Read-Only Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM</td>
<td>Electically Erasable Programmable Read-Only Memory</td>
</tr>
<tr>
<td>ELHEP</td>
<td>Electronics for High Energy Physics</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
<th>Facility for Antiproton and Ion Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIR</td>
<td>Facility for Antiproton and Ion Research</td>
</tr>
<tr>
<td>FCS</td>
<td>FPGA Configuration Software</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FMC</td>
<td>FPGA Mezzanine Card</td>
</tr>
<tr>
<td>FMC-HPC</td>
<td>FMC High Pin Count Carrier Tester</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FRU</td>
<td>Field Replaceable Unit</td>
</tr>
<tr>
<td>Character</td>
<td>Acronym</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>G</td>
<td>GbE</td>
</tr>
<tr>
<td></td>
<td>GPIO</td>
</tr>
<tr>
<td></td>
<td>GSI</td>
</tr>
<tr>
<td></td>
<td>GTP</td>
</tr>
<tr>
<td>H</td>
<td>HTML</td>
</tr>
<tr>
<td>I</td>
<td>I2C</td>
</tr>
<tr>
<td></td>
<td>IPMB</td>
</tr>
<tr>
<td></td>
<td>IPMI</td>
</tr>
<tr>
<td>J</td>
<td>JTAG</td>
</tr>
<tr>
<td>L</td>
<td>LHC</td>
</tr>
<tr>
<td></td>
<td>linac</td>
</tr>
<tr>
<td></td>
<td>LNLS</td>
</tr>
<tr>
<td></td>
<td>LVDS</td>
</tr>
<tr>
<td>M</td>
<td>MCH</td>
</tr>
<tr>
<td></td>
<td>M-LVDS</td>
</tr>
<tr>
<td></td>
<td>MMC</td>
</tr>
<tr>
<td></td>
<td>Msps</td>
</tr>
<tr>
<td>N</td>
<td>NCBJ</td>
</tr>
<tr>
<td>O</td>
<td>OHWR</td>
</tr>
<tr>
<td>P</td>
<td>PCI</td>
</tr>
<tr>
<td></td>
<td>PCI-E</td>
</tr>
<tr>
<td></td>
<td>PXI</td>
</tr>
<tr>
<td></td>
<td>PXI-E</td>
</tr>
<tr>
<td>R</td>
<td>RAM</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SATA</td>
<td>Serial Advanced Technology Attachment</td>
</tr>
<tr>
<td>SDR</td>
<td>Sensor Data Record</td>
</tr>
<tr>
<td>SerDes</td>
<td>Serializer/Deserializer</td>
</tr>
<tr>
<td>SFP</td>
<td>Small form factor pluggable</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SRIO</td>
<td>Serial Rapid IO</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver and Transmitter</td>
</tr>
<tr>
<td>uRTM</td>
<td>Mikro Rear Transition Module</td>
</tr>
<tr>
<td>uTCA</td>
<td>Micro Telecommunications Computing Architecture</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage-Controlled Oscillator</td>
</tr>
<tr>
<td>VHDL</td>
<td>Very High Speed Integrated Circuits Hardware Description Language</td>
</tr>
<tr>
<td>VME</td>
<td>Versa Module Europa</td>
</tr>
<tr>
<td>VXI</td>
<td>VME eXtensions for Instruments</td>
</tr>
</tbody>
</table>