JÜLICH
FORSCHUNGSZENTRUM

A Test System for the Front-End Electronics of the $\bar{P}$ANDA MVD

DPG-Frühjahrstagung 2016 - HK 29.3

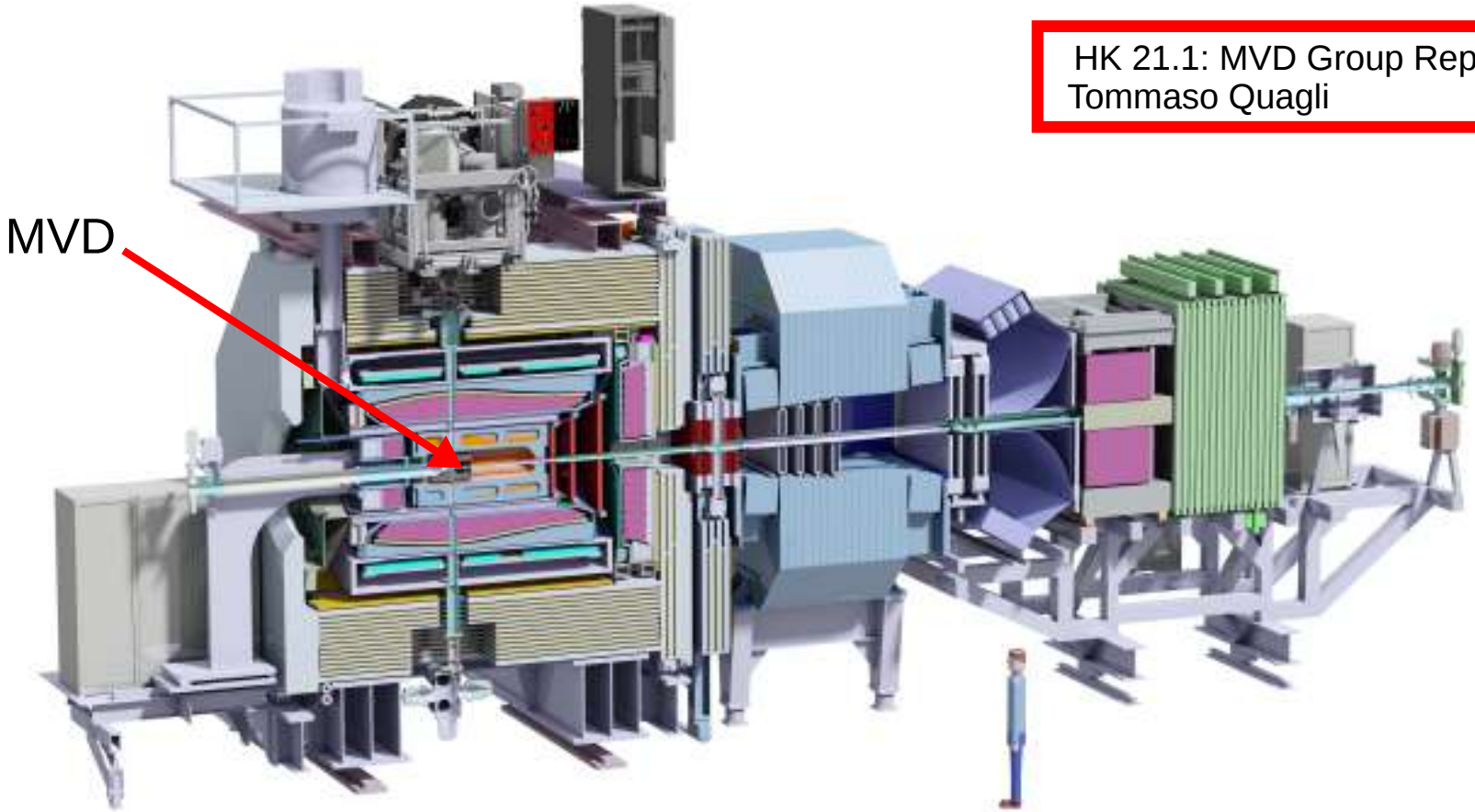
Alessandra Lai, Forschungszentrum Jülich, 15 March, 2016

The $\bar{P}$ANDA Experiment

Antiproton Annihilation at Darmstadt

JÜLICH FORSCHUNGSZENTRUM

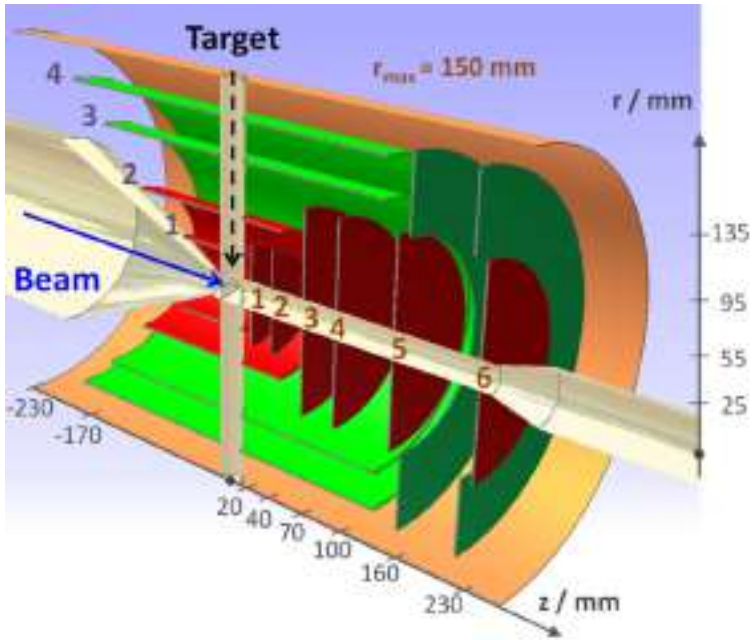
HK 21.1: MVD Group Report
Tommaso Quagli



- fixed p/N target
- $\bar{p}$ beam 1.5 - 15 GeV/c

MVD: Micro Vertex Detector

- event rate $\approx$ 20 MHz
- vertex resolution $<$ 100 μm



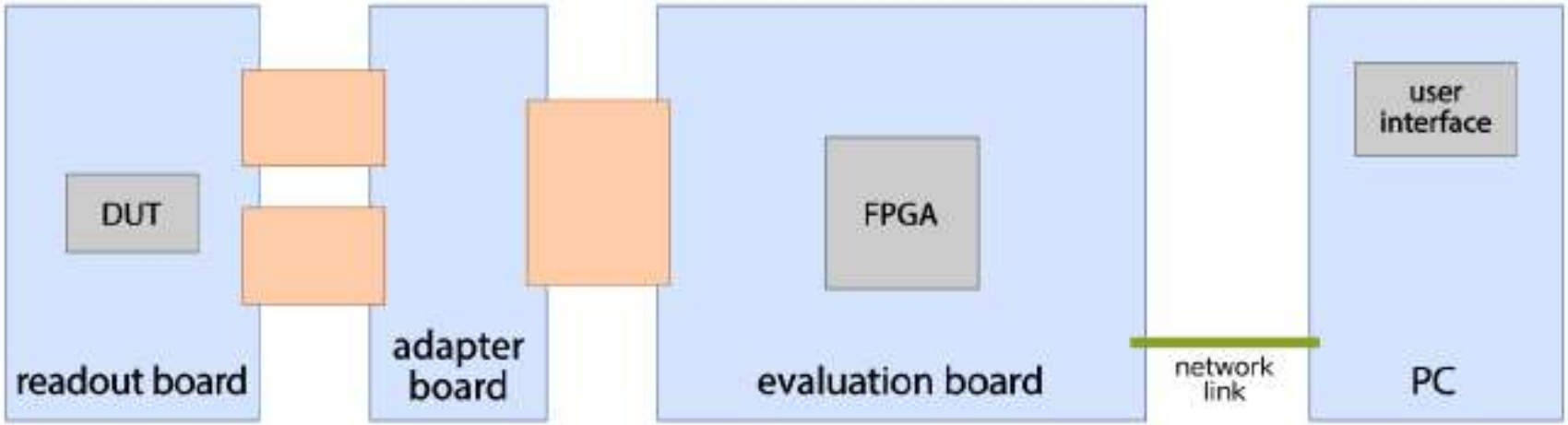
- time resolution $\approx$ 10 ns
- deposited energy information

- four barrel layers in the center
- six disk layers in the fw direction
- pixel detectors in the inner part
  $\longrightarrow$ front-end chip: ToPix
- double-sided strip detectors in the outer part
  $\longrightarrow$ front-end chip: PASTA [HK 7.2]

High performance and flexible test system needed for ToPix and PASTA

JDRS: Jülich Digital Readout System

The basic components



Data conversion & communication with PC:

- DUT: ToPix, PASTA
- evaluation board: Xilinx ML605 (Virtex-6 FPGA)
- firmware: VHDL

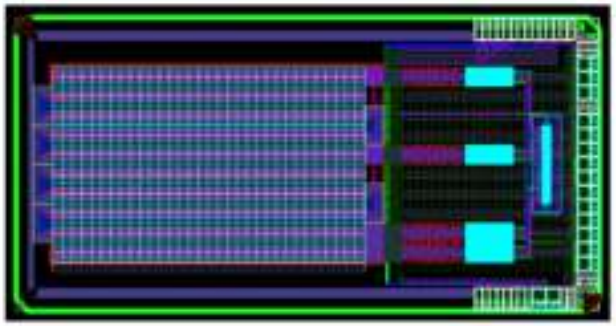
Configuration & data handling:

- PC
- software: C++
- **MVD readout framework (MRF)**
- **Qt-based GUI**

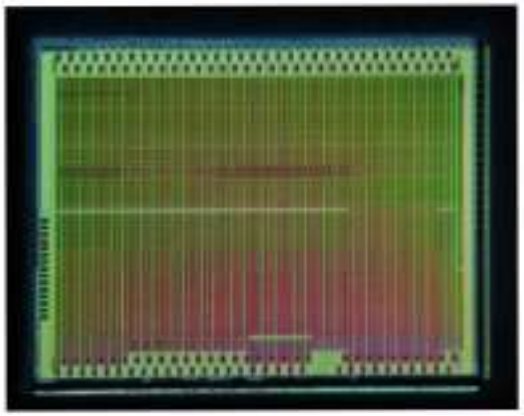
JDRS: Jülich Digital Readout System

The real components

ToPix



PASTA

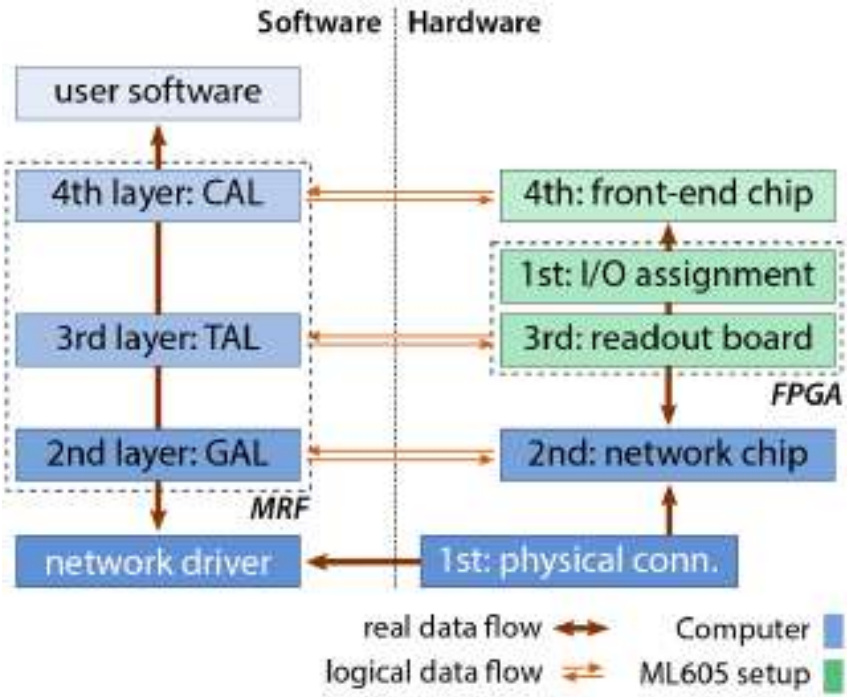


Xilinx ML605

MRF: MVD Readout Framework

Four abstraction layers isolate low level from higher level functions:

- physical layer
  $\longrightarrow$ ethernet connection between ML605 and PC
- generic access layer (**GAL**)
  $\longrightarrow$ data transfer and formatting
  e.g. open a connection, send and receive data packages, ...
- transport access layer (**TAL**)
  $\longrightarrow$ board-specific functions
  e.g. the clock generation, flush of buffers, ...
- chip access layer (**CAL**)
  $\longrightarrow$ DUT-specific functions
  e.g. configuration and data readout, ...



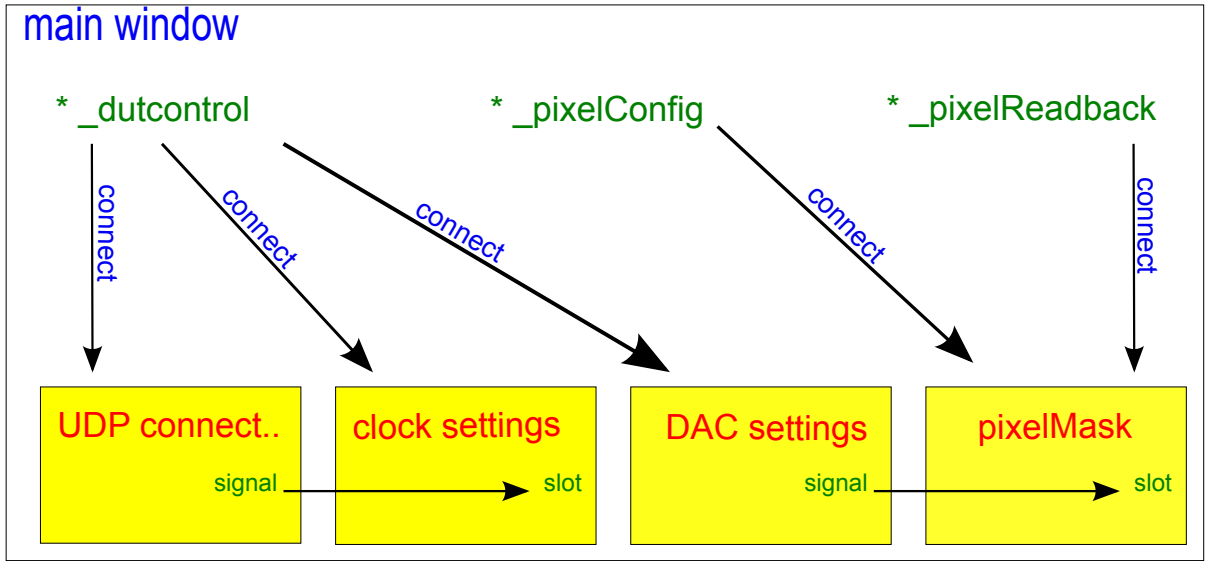
Qt-based GUI

Make it modular by separating the functionalities in independent projects.

Caveat

Comunication between projects is needed.

One main window that contains all the other tabs as sub-widgets.




```
connect(ui->pushButton_defaultMode, SIGNAL(clicked()),
        ui->widget_UDP, SLOT(connectUDP()));
```

Load and save settings

Settings (e.g. configuration data) are stored in *.json* files → Qt offers support for JSON
Advantages:

- easy programmatic access to the key-value pairs,
- human readable format.

```
"CommandCCR0"            : 32,
"CommandCCR1"            : 33,
"CommandCCR2"            : 34,
"CounterMode"            : 1,
"CounterEnable"          : 1,
"ReadoutCycleHalfSpeed"  : 1,
"FreezeStop"             : 4,
"Leak_P"                 : 1,
"SelectPol"              : 1,
"PreEmphasisTimeStamp"   : 1,
"PreEmphasisCommands"    : 1,
"CounterStopValue"       : 4095
```

```
"CalLevelDac"   : 5000,
"VCasIlc"       : 45580,
"VCasIfb"       : 40350,
"VRefBaseline"  : 37600,
"notUsed1"      : 0,
"notused2"      : 0,
"VRefD"         : 37300,
"VCasD"         : 32450
```

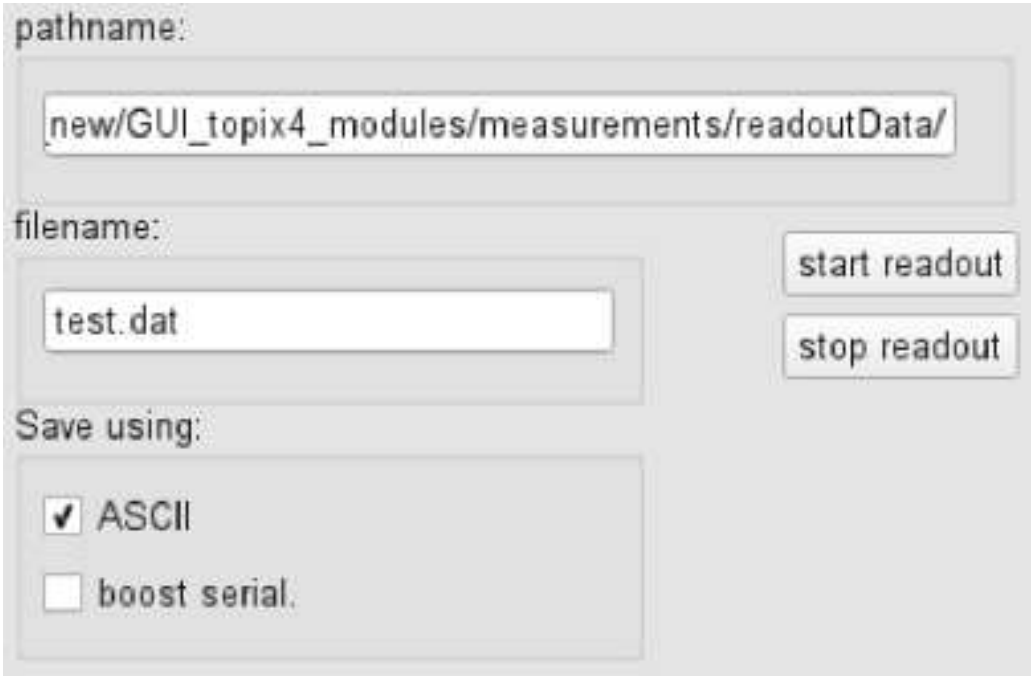
A similar format is available for the pixel configuration as well:

- the key is the row number ($nKeys = nRow = 20$),
- the value is a 32-entry array ($nCol = 32$).

Data handling

The data buffer is read (and emptied) when needed and the data is stored in a file.
Two possibilities for data storage exist:

- (direct) ASCII,
- boost serializer.



The interface allows to add new methods for storing the data
(e.g. *FairMQ*, useful for beam tests).

Measurements

JÜLICH FORSCHUNGSZENTRUM

One part of the GUI is dedicated to measurements $\rightarrow$ characterization of ToPix.

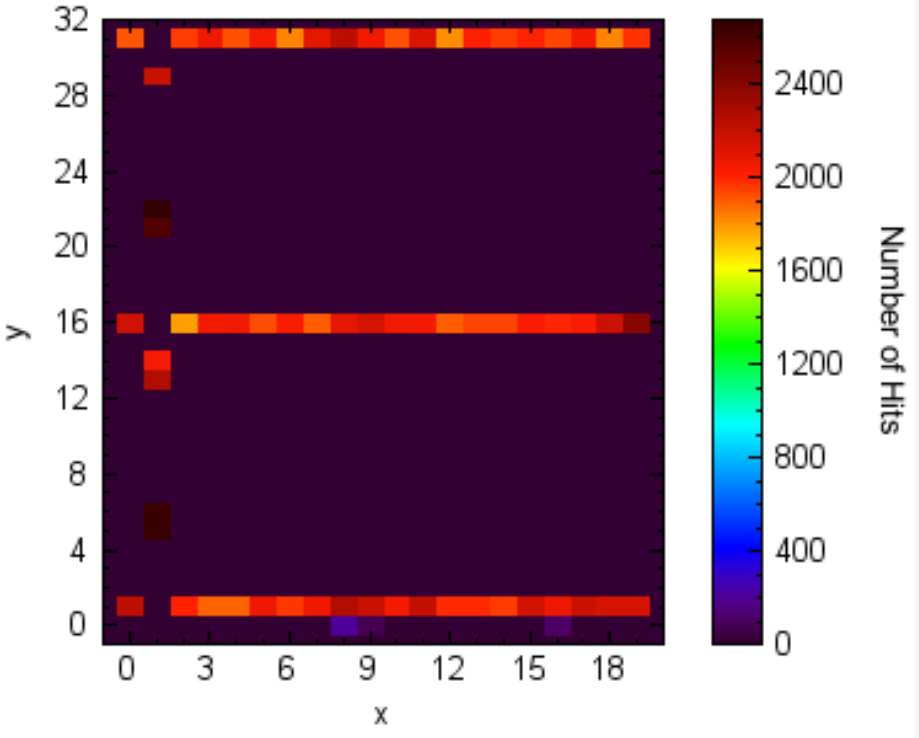
Each pixel has an internal circuit for the injection of a certain amount of charge.
Two modes of operation:

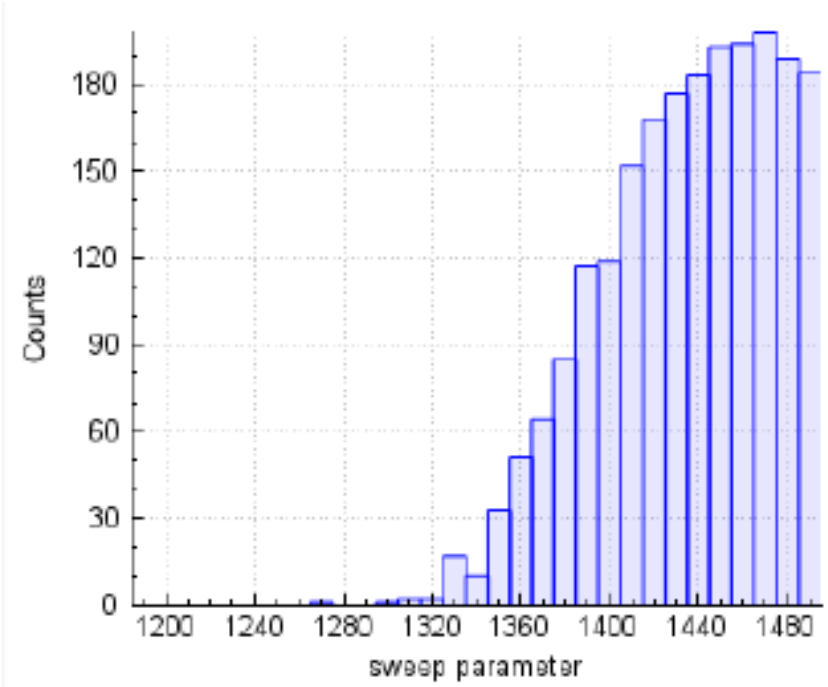
- inject once into enabled pixels
- enable certain pixels (pattern), inject and repeat the pattern shifted by one row for *n* times

Up to two parameters can be swept during the charge injection (e.g. threshold parameters, ...)

Measurements: online visualization

Row 1, 16, 31 enabled.





Git repository

The JDRS is under version control on Git (firmware and software).

- JDRS core (git submodule)
- JDRS ToPix

The core repository contains all the JDRS basic functionalities
(UDP connection, register access, chip configuration, ...).
$\Longrightarrow$ 100% reusable for PASTA.

The ToPix repository contains ToPix specific functionalities.
$\Longrightarrow$ partially reusable for PASTA (adaptations are required).

It is sufficient to checkout the repo and run the JDRS (config file included).
Dependencies:

- boost library
- root

Summary

- The PANDA MVD will use pixel and strip detectors.
- Two front-end chips: ToPix and PASTA.
- The JDRS is under development.

Requirements on the system: modular, maintainable, flexible and user friendly.
Achieved by:

- arranging the code to match the GUI structure,
- making the system as independent as possible from the environment,
- reducing the dependencies on external libraries/packages,
- optimizing data handling and storage.

Preparation for full characterization of the chip (at present ToPix, in a later stage PASTA) with automatic routines that can be handled from the GUI.

References

- $\bar{P}$ANDA and MVD: The PANDA Collaboration, Technical Design Report for the $\bar{P}$ANDA Micro Vertex Detector. (2011) [http://arxiv.org/abs/1207.6581]
- ToPix ASIC: INFN Torino
- PASTA ASIC: A. Zambanini, PhD thesis [http://hss-opus.ub.ruhr-uni-bochum.de/opus4/frontdoor/index/index/docId/466]
- ASICs pictures: courtesy of Daniela Calvo and Tommaso Quagli
- ML605: http://www.xilinx.com/
- MRF: M. Mertens, PhD thesis [http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MertensMariusC/]

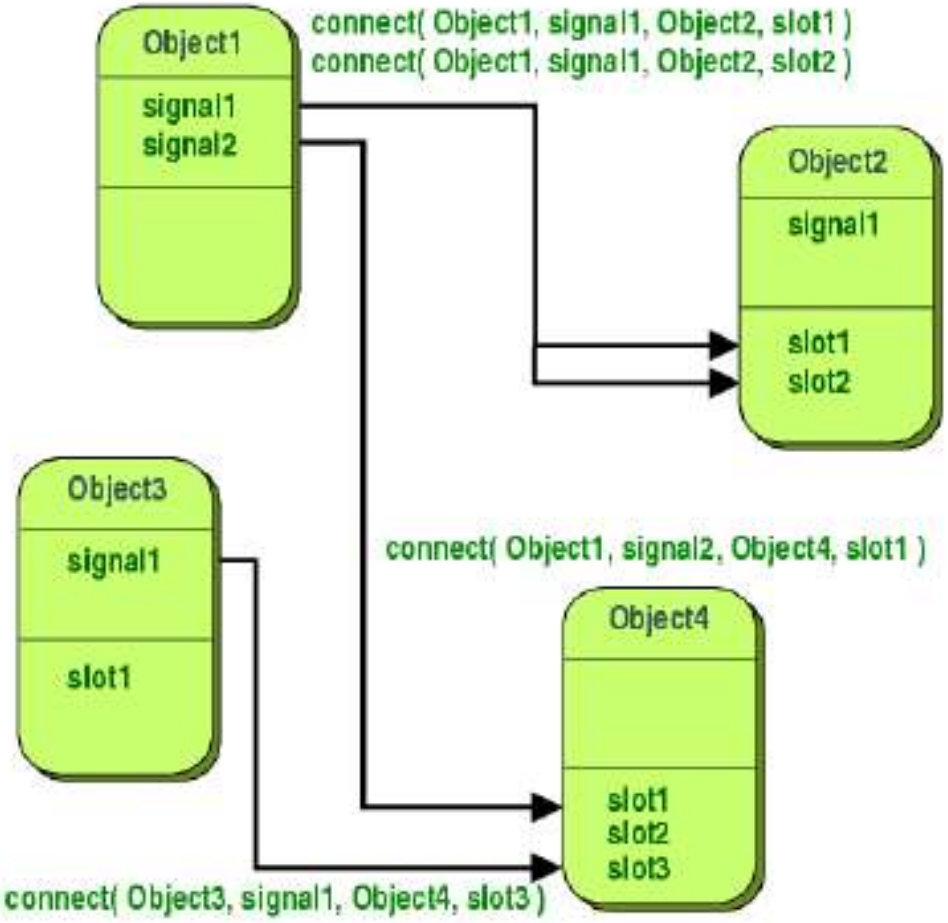
Qt - 'cute' framework

JÜLICH FORSCHUNGSZENTRUM

Qt is a widely used framework for developing application software with graphical user interfaces (GUIs) but not only (e.g. command-line tools).

- open source
- cross-platform (Linux, Windows, Android, Mac, ...)
- uses system resources (i.e. the app gets a native look)
- supports standard C++
- signals and slots mechanism (for event handling)
- supports several compilers (e.g. GCC, Visual studio)
- supports threading for parallel programming
- supports a designer for the layout of the UI
- ...

Signals and slots




```
connect(this, SIGNAL(sendTopix4CtrlPtr(TMrfCal_Topix4 *)),
                ui->widget_chipConfig, SLOT(recvTopix4CtrlPtr(TMrfCal_Top
connect(ui->pushButton_defaultMode, SIGNAL(clicked()),
                        ui->widget_UDP, SLOT(connectUDP()));
```

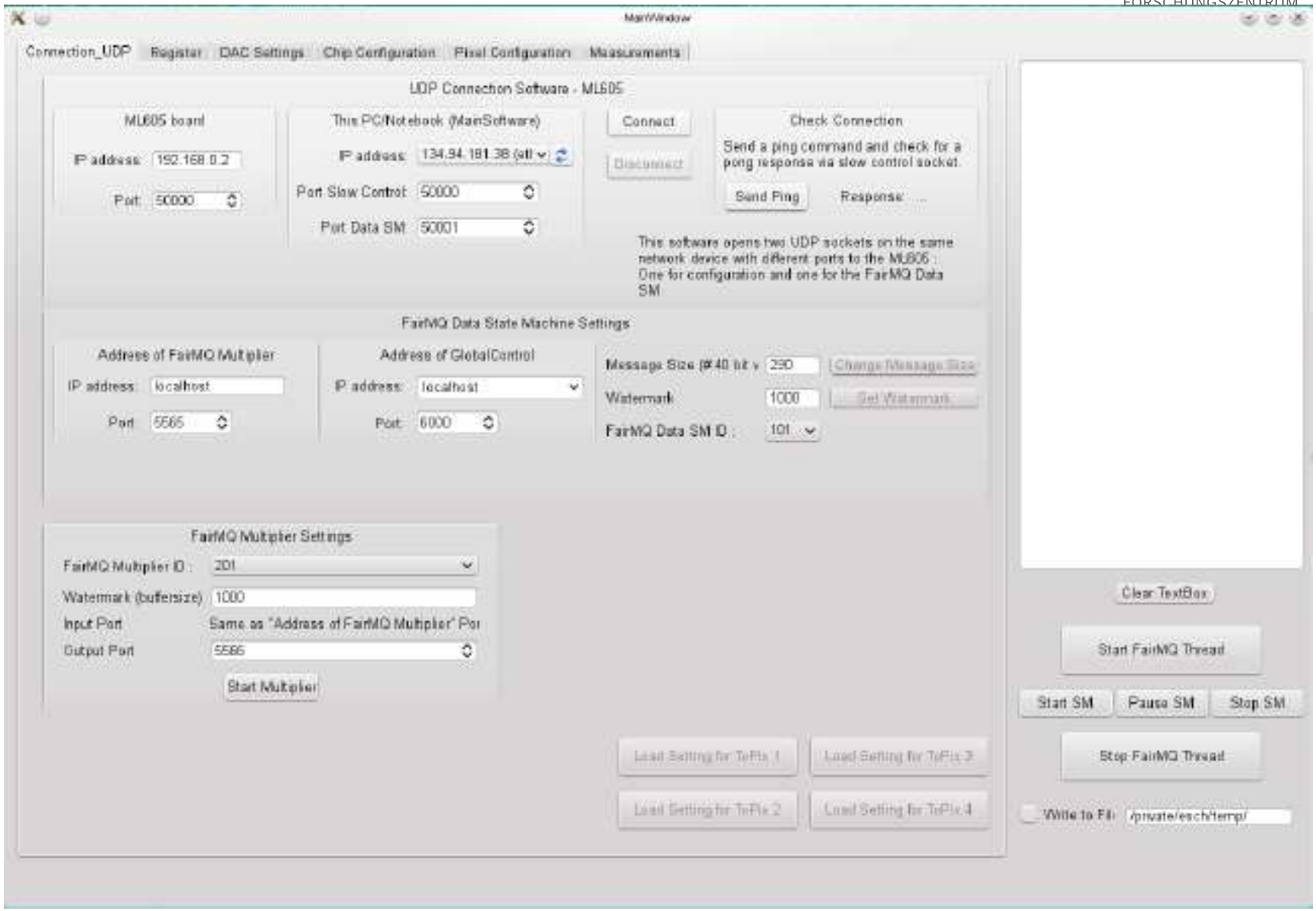
Former version

Originally designed for beam test measurements $\longrightarrow$ suffered from strict time constraints, quick implementation and workarounds:

- environment dependency
- lack of modularity and flexibility
- lack of structure
- hard coded settings and magic numbers in the code
- extreme sensitivity to external changes
- cumbersome for inexperienced users

- max reliable readout frequency $\simeq$ 50 MHz
- data from chip handled by *FairMQ* $\rightarrow$ cumbersome and not necessary for lab measurements

The code to generate the whole GUI in a single file $\approx$ 2000 lines!



Refactoring process: strategy

The idea is to make the existing framework modular by separating the functionalities in indipendent projects.
Each project consist of, at list:

- a *.pro file*
- a standard C++ class
- a form (i.e. the actual UI)

Rule of thumb: one project per tab.

Caveat

Comunication between projects is needed (e.g. an event that occurs in one tab might trigger an instruction in another tab).

One main window that contains all the other tabs as sub-widgets.

Main window and widgets

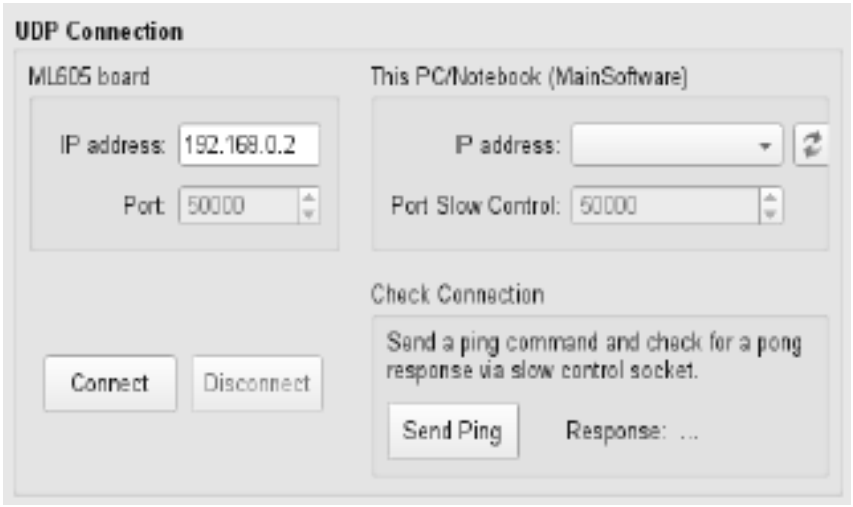
Each widget:

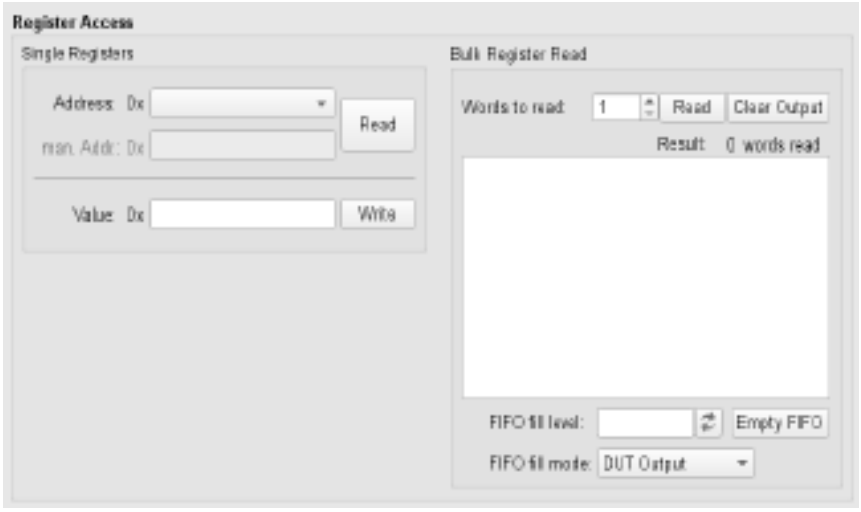
- is an indipendent project that can be executed standalone
- is included in the main window to build up the full gui
- is connected to the others, if needed, through the main window

The main window:

- holds the sub-widgets
- performs the connections between signals and slots
- initiates and distributes the global information

Widgets





The MRF

The MRF is now a library that can be built indipendently.
It has its own *.pro* file but no UI.
$\longrightarrow$ replace the files 'include' with the library include

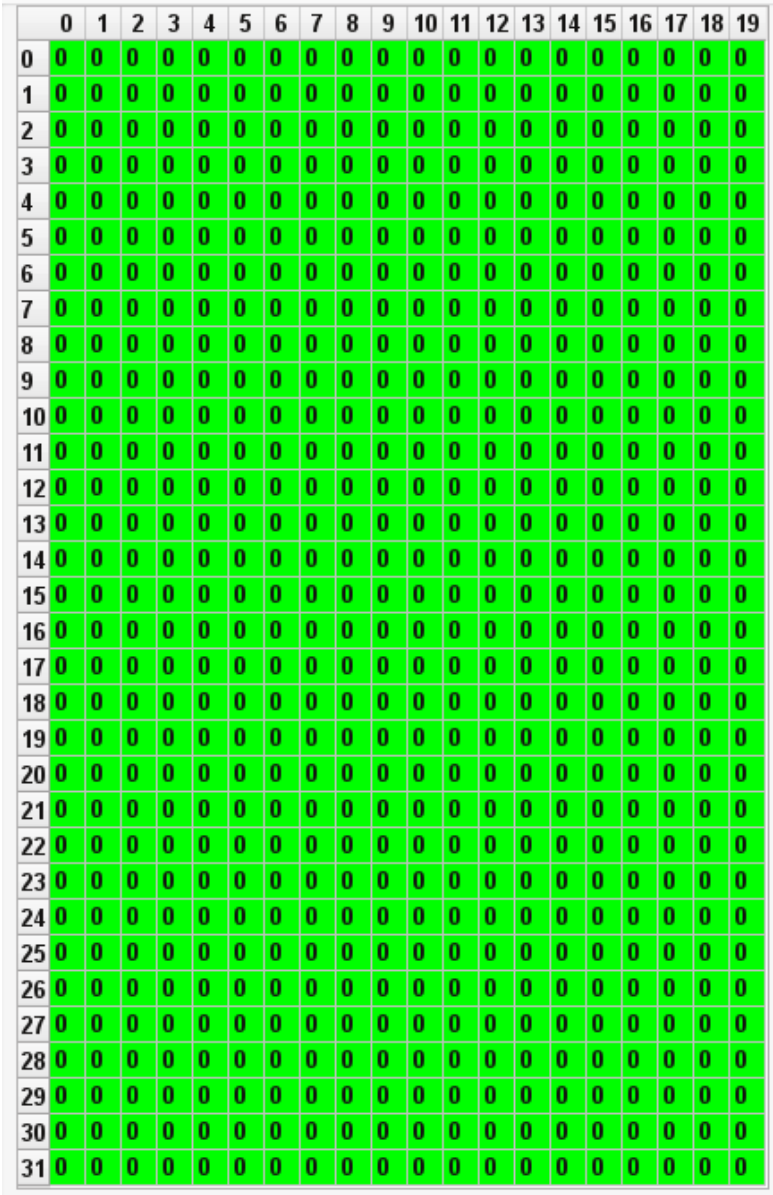
```
SOURCES += main.cpp\
        mainwindow.cpp \
        topix4_fairmq_readout.cpp \
         ../writetofile.cpp \
         ../../MRF/source/mrfdata_chain2ltc2604.cpp \
         ../../MRF/source/mrfdata_chainltc.cpp \
         ../../MRF/source/mrfdataadv2d.cpp \
         ../../MRF/source/mrfdataadvbase.cpp \
         ../../MRF/source/mrfcal_topix4.cpp \
         ../../MRF/source/mrfcal.cpp \          LIBS +=
         ../../MRF/source/mrftal_rbtopix4.cpp \    -L$${PROJECT
         ../../MRF/source/mrftal_rbbase_v6.cpp \
         ../../MRF/source/mrftal_rbbase.cpp \
         ../../MRF/source/mrfdataregaccess.cpp \
         ../../MRF/source/mrfdataadv1d.cpp \
         ...
HEADERS += mainwindow.h \
        topix4_fairmq_readout.h \
         ../writetofile.h \
        ...
```

Readback

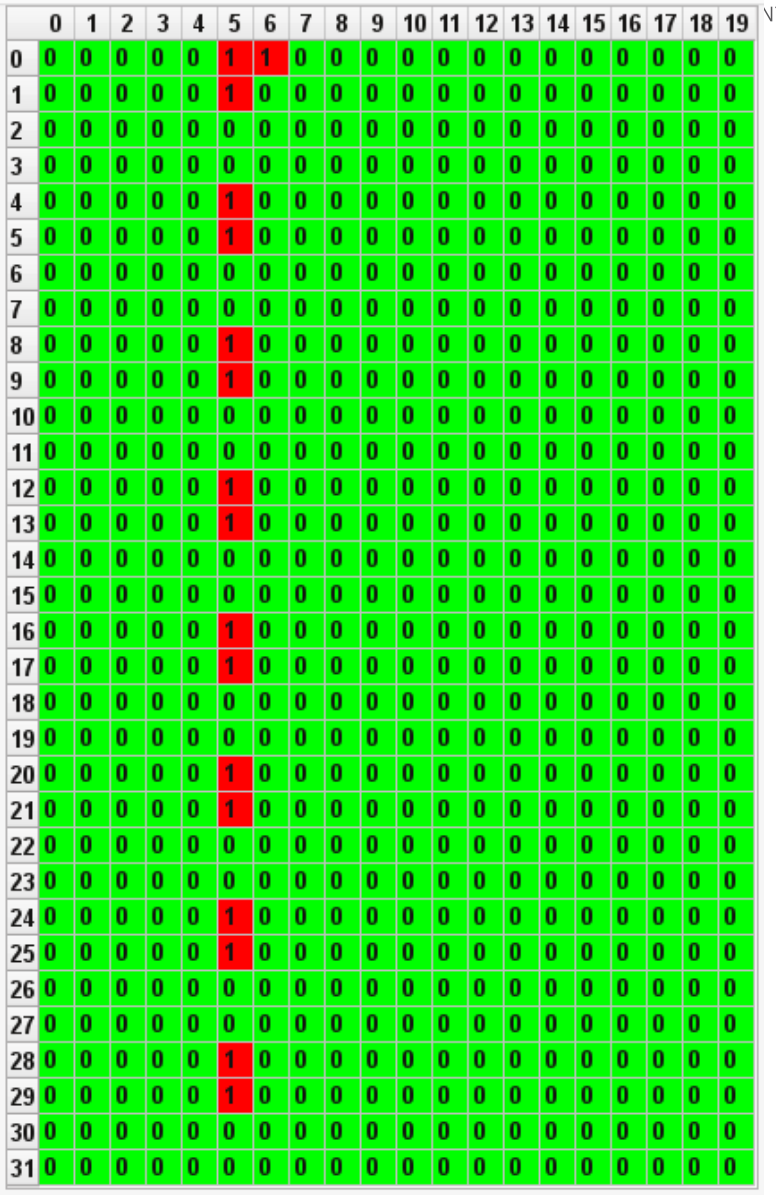
One way to check if the process of writing to the chip was successfull is to read back the data coming from it.

All the data that is written to the chip (configuration values, pixel status, etc) can be read back and visualized in the GUI.

For example the pixel mask status.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0