BIS main electronic modules

- Oriented Linac4 -

Stéphane Gabourin
TE/MPE-EP

Workshop on Beam Interlock Systems
03-04 Jan 2015

Outline

BIC architecture
- BIS hardware: simplified layout
- BIS in LHC
- BIS in Linac4

Manager CIBM
- Critical code
- Monitoring code

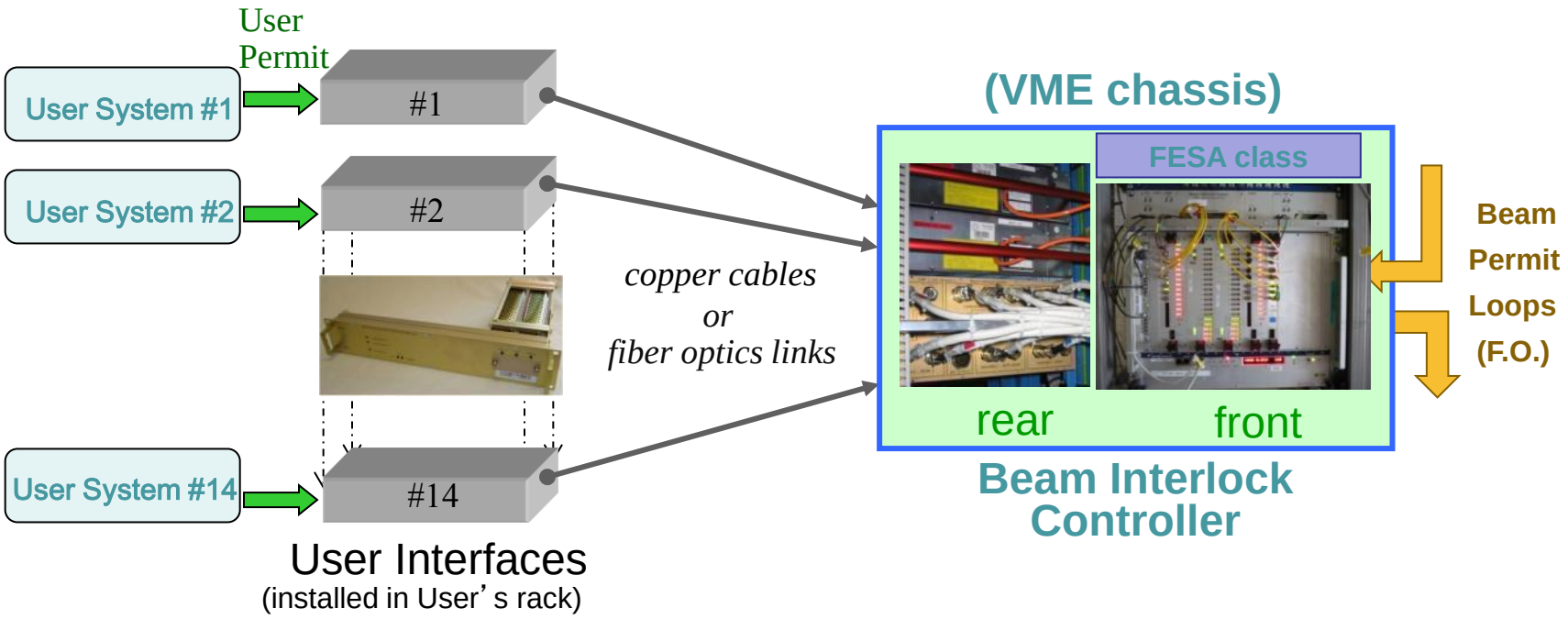
Master Matrix CIBX

Tester CIBT

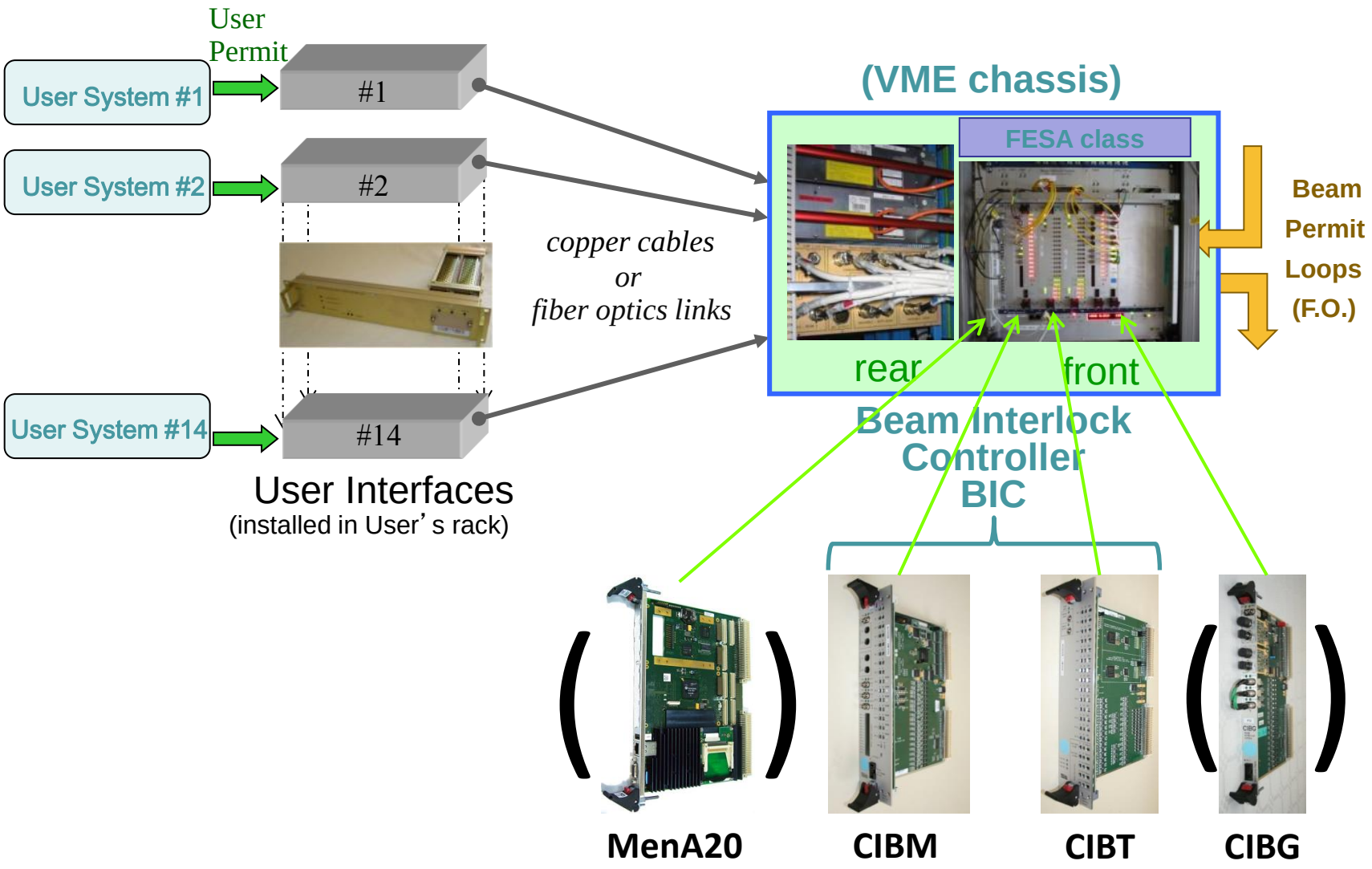
Interface with Target System CIBIT

BIS hardware simplified layout

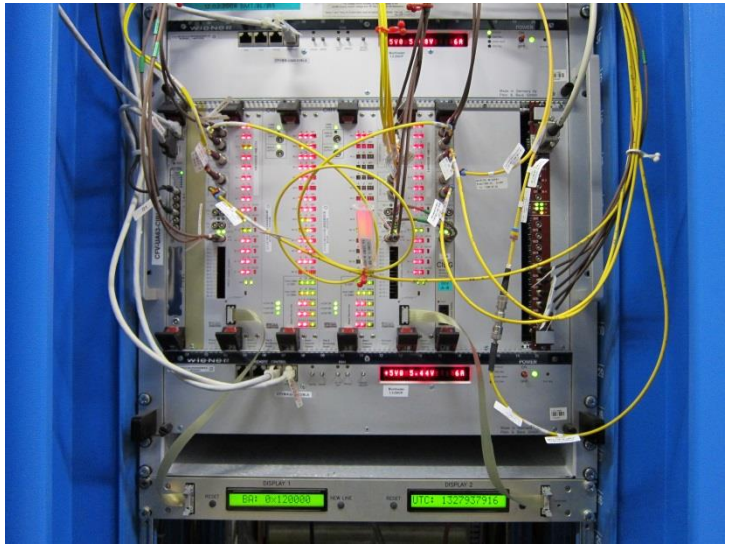
- The User Interfaces safely transmit User Permit signals from User Systems to the BIC
- The BIC generates a Local Permit from the User Permits
- The Local Permit is used to act on the Beam Permit
- To reach high safety level, all the chain is fully redundant (A and B permits).



BIS hardware simplified layout



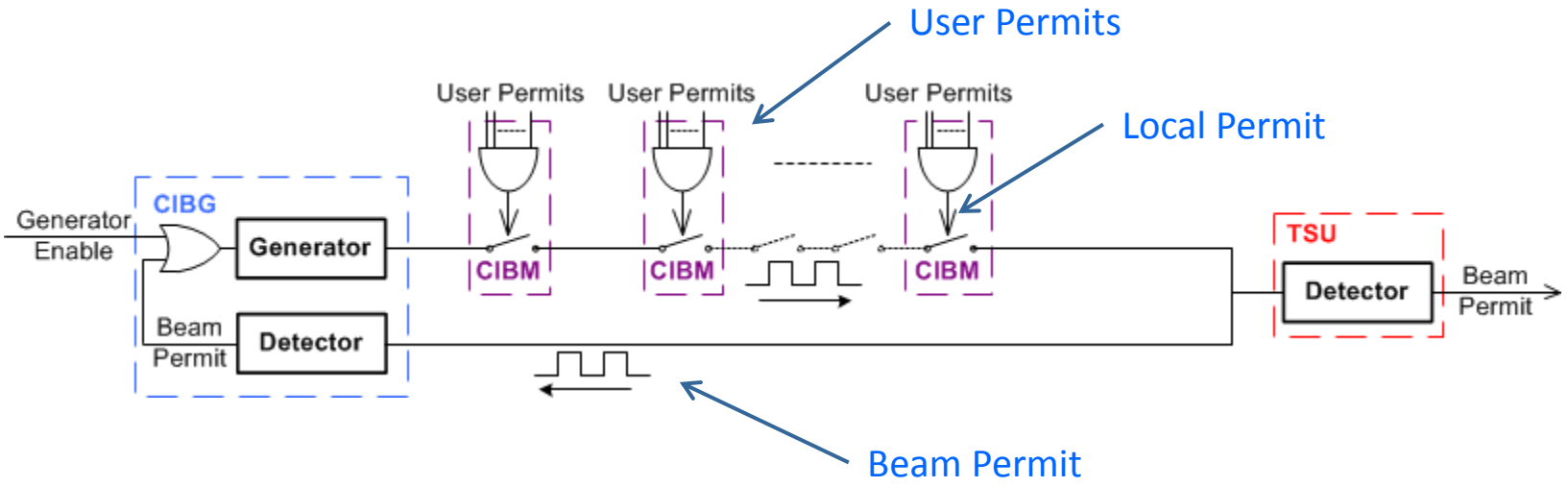
BIS in LHC



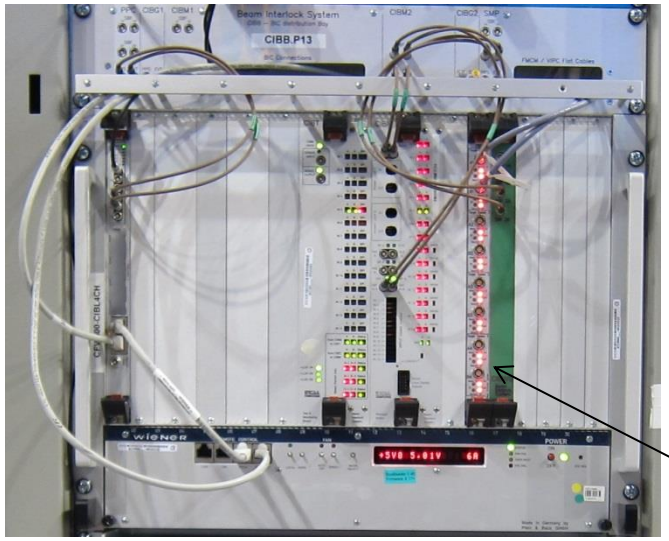
LHC BIC in UA63

LHC case (ring architecture):

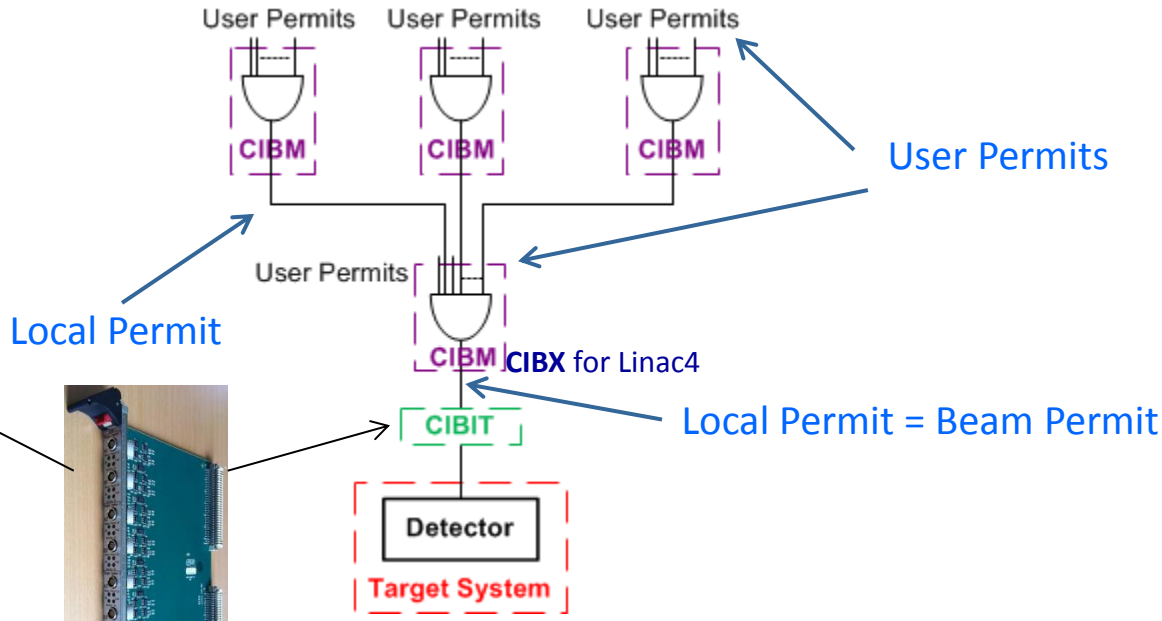
- Each **CIBM** gathers User Permits (up to 14) from User Systems
- The useful User Permits are ANDED to generates a Local Permit
- A Beam Permit, generated by the **CIBG**, circulates on a loop (optic fibers)
- Each Local Permit, when FALSE, "opens" the Beam Permit loop
- When Beam Permit becomes FALSE, the Detector triggers the Dumping System.



BIS in Linac4



Linac4 BIC



Linac4 case (tree architecture):

- Each **CIBM** gathers User Permits (up to 14) from User Systems or from other **CIBM**
- A logical AND of the useful User Permits generates a Local Permit
- The Local Permit of the **CIBM** (**CIBX**) at the end of the tree is used as Beam Permit
- The Beam Permit is transmitted to the Detector (electrical connection) by the **CIBIT**
- When Beam Permit becomes FALSE, the Detector triggers the Target System.

Outline

BIC architecture

- BIS hardware: simplified layout
- BIS in LHC
- BIS in Linac4

Manager CIBM

- Critical code
- Monitoring code

Master Matrix CIBX

Tester CIBT

Interface with Target System CIBIT

Manager CIBM

CIBM – Controls Interlocks Beam Manager

Critical Function

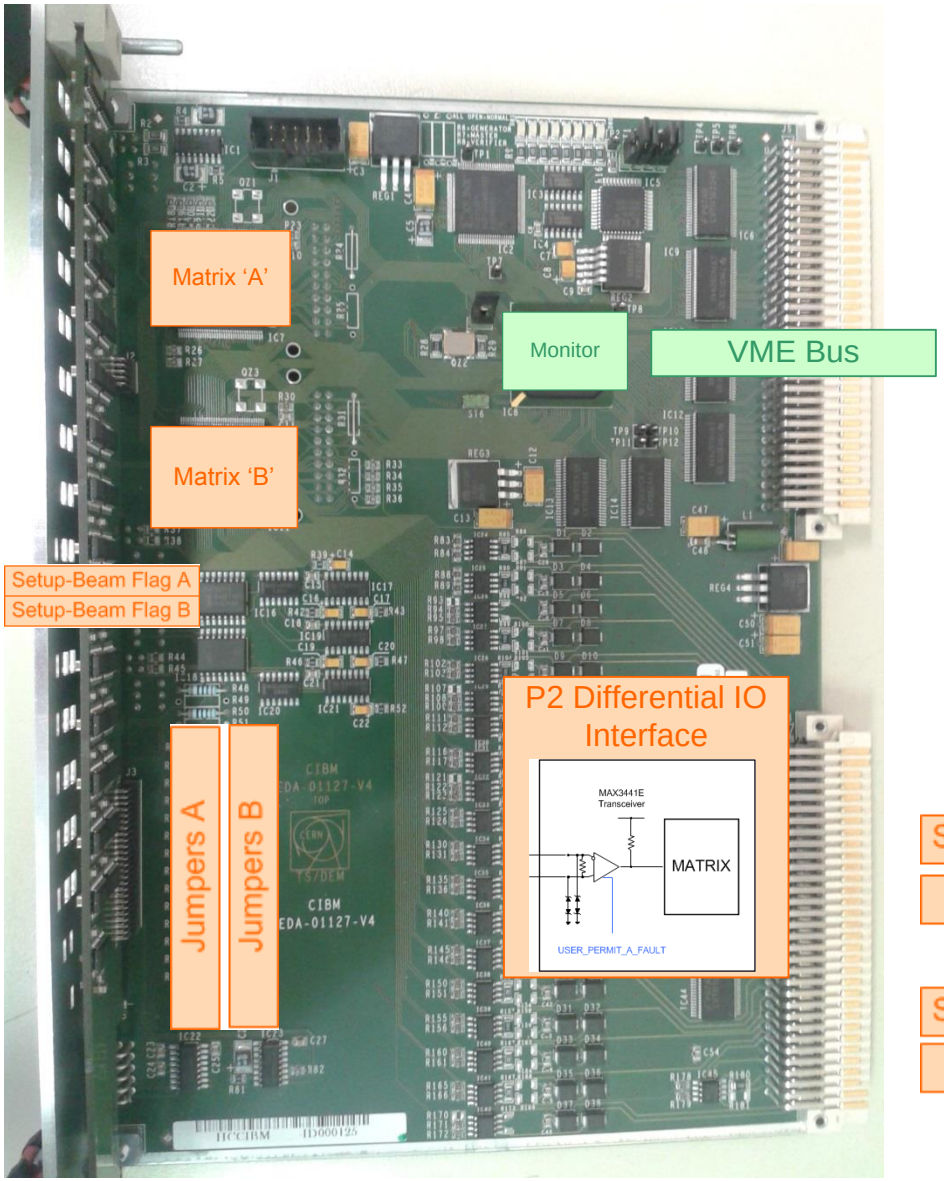
1.Perform local 'AND' to make a LOCAL_BEAM_PERMIT from all USER_PERMITs
2.Filter each USER_PERMIT to remove glitches
3.Apply DISABLE jumpers to switch off unused channels
4.Allow MASKABLE inputs to be ignored if SETUP_BEAM_FLAG is TRUE

Everything duplicated

Critical is in a different device to Non-Critical (monitoring)
Almost all data is only read by the monitoring (except software permit and mask)

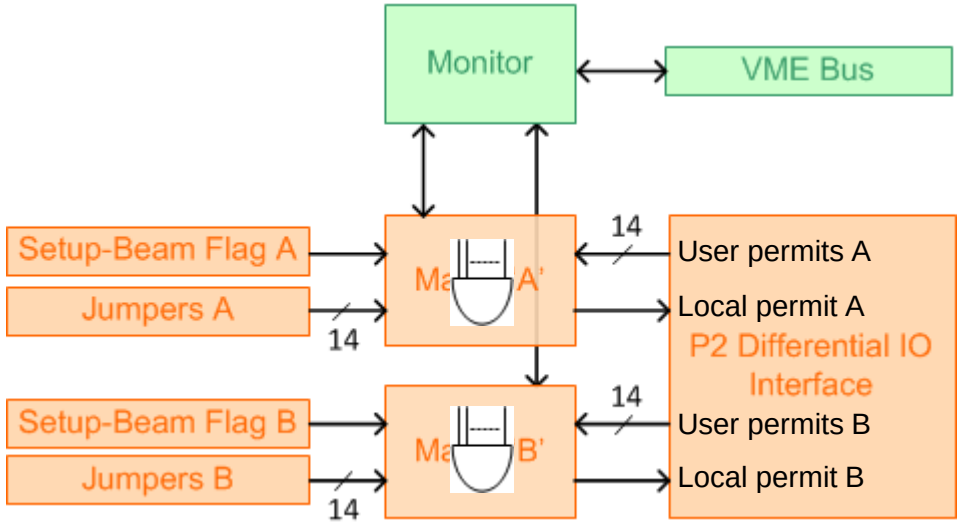
Fail Safe value applied if a user permit signal is faulty (or unplugged)

Manager CIBM

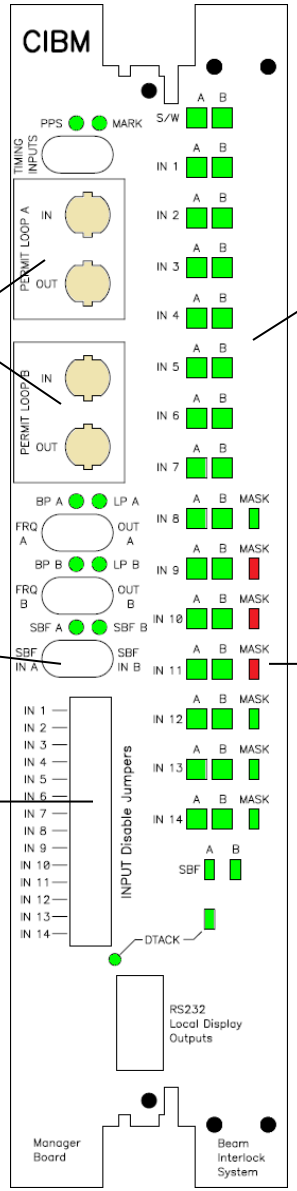


CIBM main characteristics:

- Max3440 transceivers for each user input
- 2 CPLD (Xilinx XC95288XL) for critical code
- 1 FPGA (Xilinx Spartan3) for monitoring
- VME interface connected on the monitoring
- Each channel can be disabled by hardware (2*14 jumpers)
- Up to 7 channels can be masked by software command



Manager CIBM



Beam Permit Loops for ring architecture

2*14 User Permits (A and B)

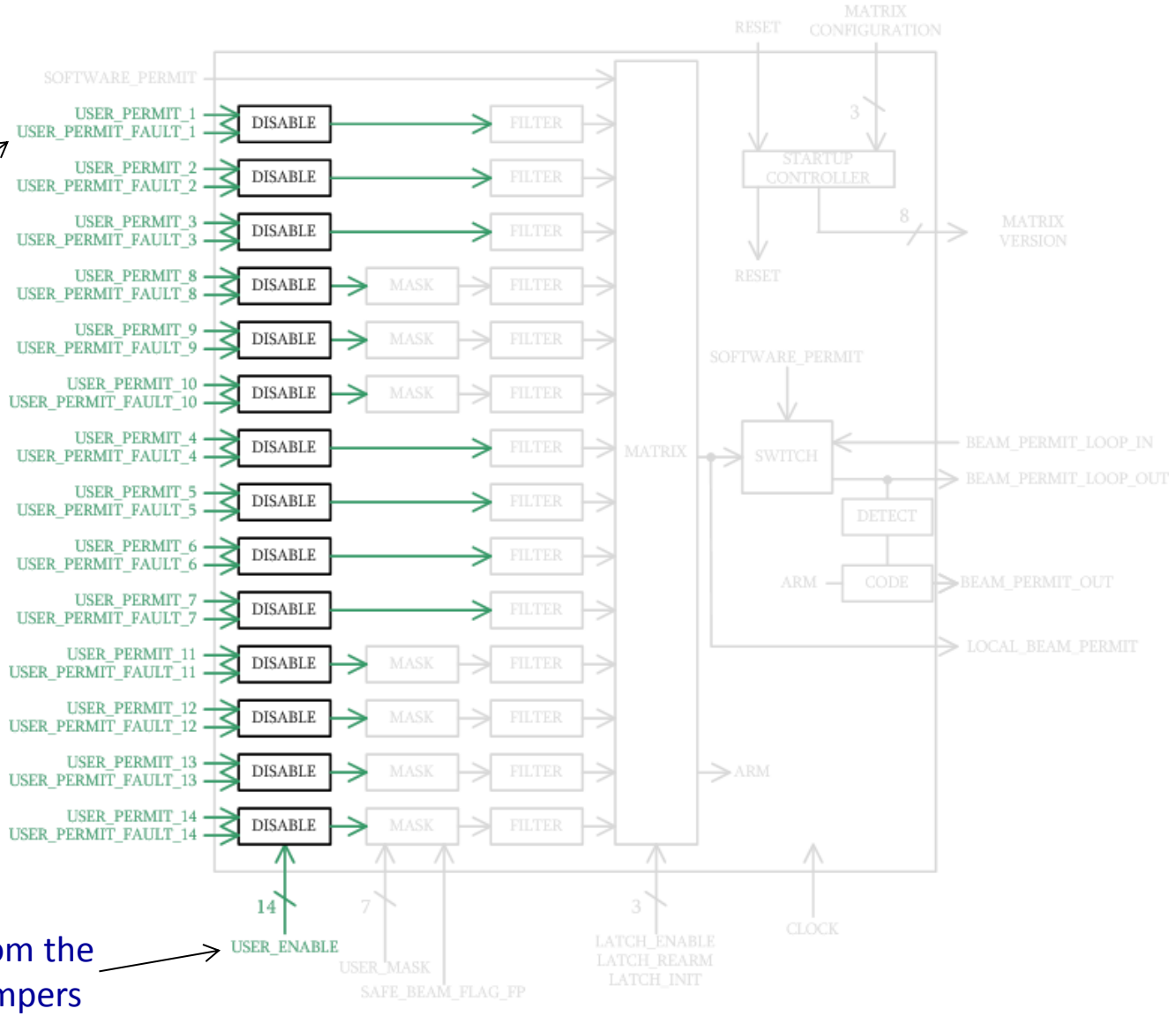
Setup Beam Flag input

Masks on the last 7 inputs

Jumpers

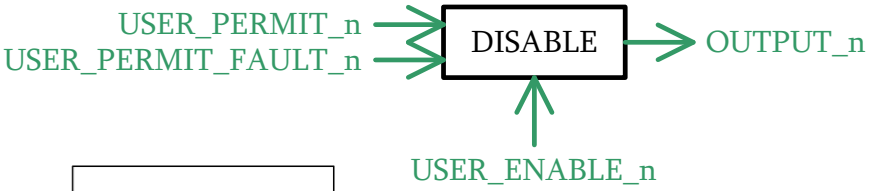
Manager CIBM – Critical code

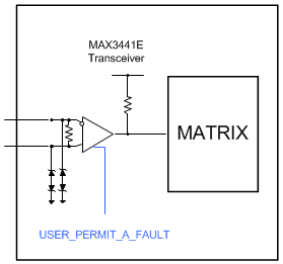
From the Max3440



From the Jumpers

Manager CIBM – Critical code



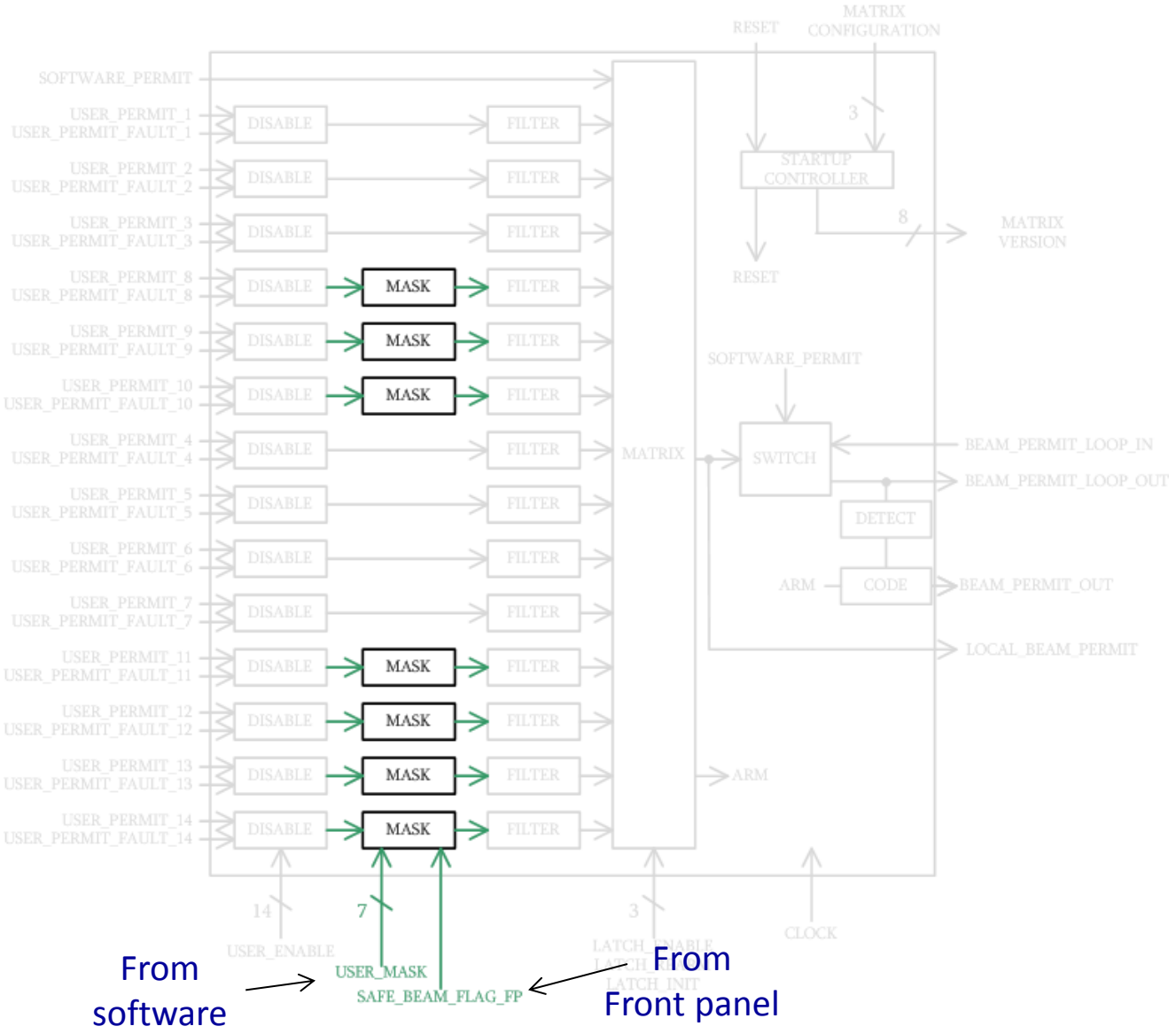


From user system | From the Max3440 | From jumpers | User Permit after enable

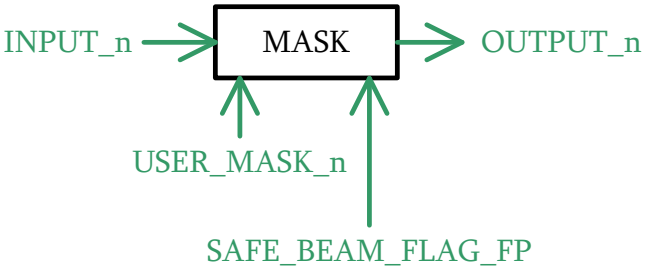
USER_PERMIT	USER_PERMIT_FAULT	USER_ENABLE	OUTPUT
TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
'x'	'x'	FALSE	TRUE

Real user permit

Manager CIBM – Critical code



Manager CIBM – Critical code



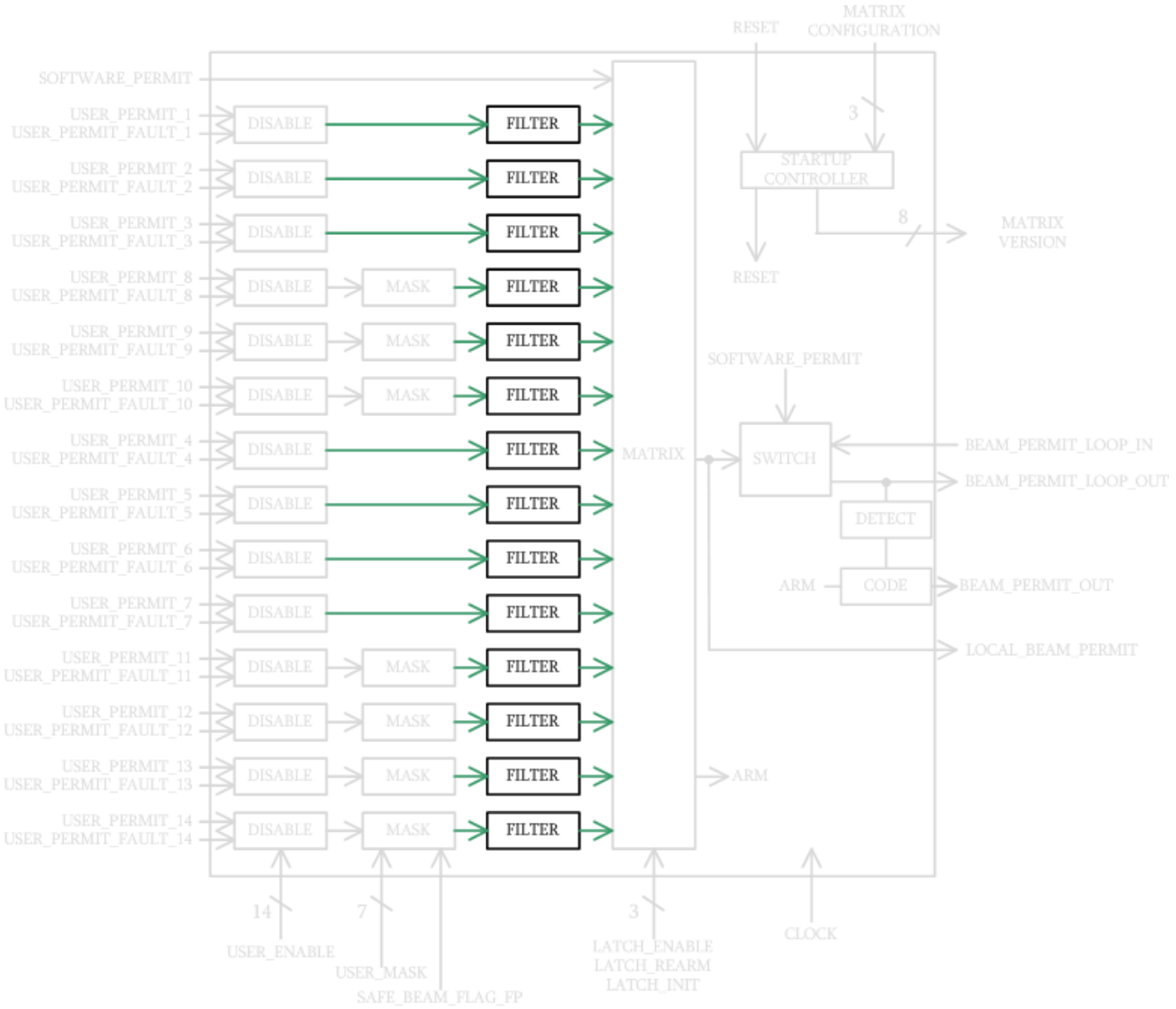
User Permit after enable | From software | From Front panel | User Permit after mask

USER_PERMIT	MASK	SAFE_BEAM_FLAG	OUTPUT
'x'	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

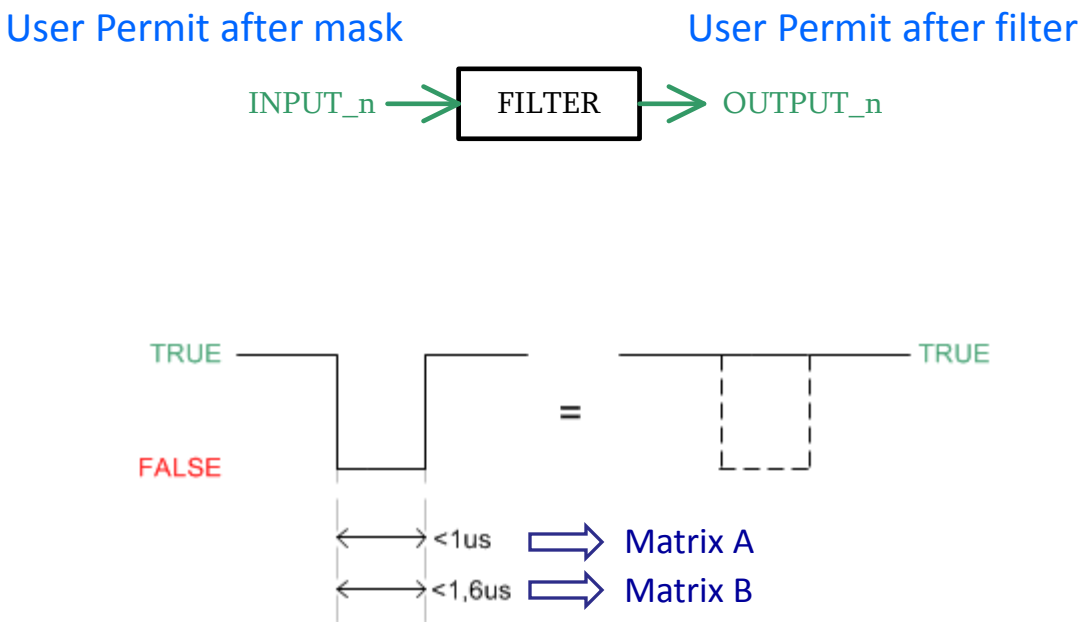
Real user permit

Only half of the inputs can be masked

Manager CIBM – Critical code

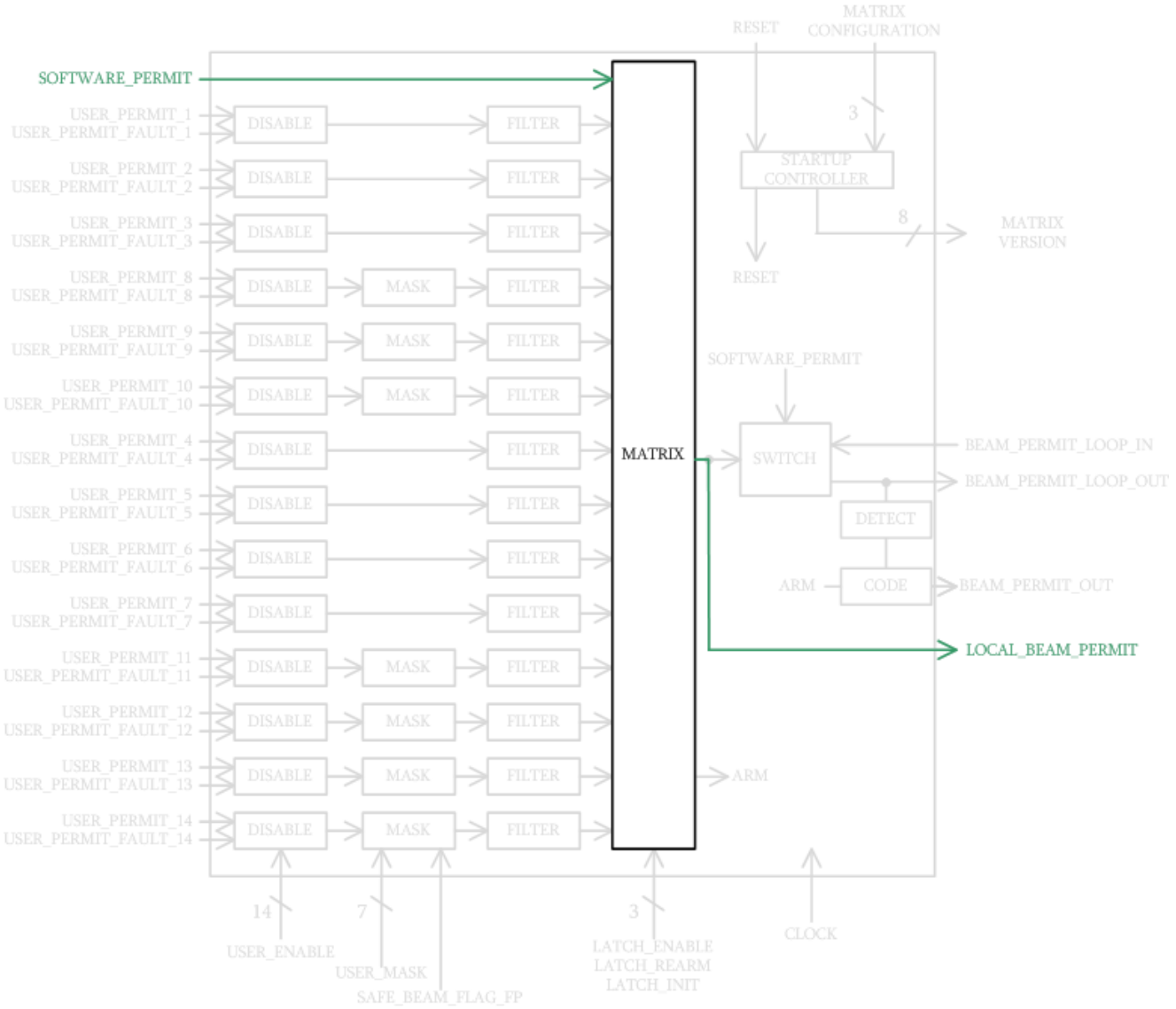


Manager CIBM – Critical code



- Probably the most critical code in the matrix => Implemented by 2 different programmers
- Perpetual filtering is possible if there is noise = counters monitored, logged and alarmed

Manager CIBM – Critical code



Manager CIBM – Critical code

User Permit after filter

If INPUT (14->1) = TRUE AND SOFTWARE_PERMIT = TRUE

then LOCAL_BEAM_PERMIT = TRUE

Manager CIBM – Monitoring code

- Gather a maximum of data from the Critical codes for monitoring (and debug) purpose.
- The memory is organized in 2 distinct parts:
  - Registers: to save statuses (permits, masks, disable, etc) - More than hundred 32-bits registers
  - History buffer: to keep a trace of the last 1024 “events”, mainly transitions (permits from TRUE to FALSE, etc). Used for diagnosis after a Post Mortem events.
- Simulates the behavior of the matrix
- Send the software permit and mask commands to the Critical Matrices

Outline

BIC architecture

- BIS hardware: simplified layout
- BIS in LHC
- BIS in Linac4

Manager CIBM

- Critical code
- Monitoring code

Master Matrix CIBX

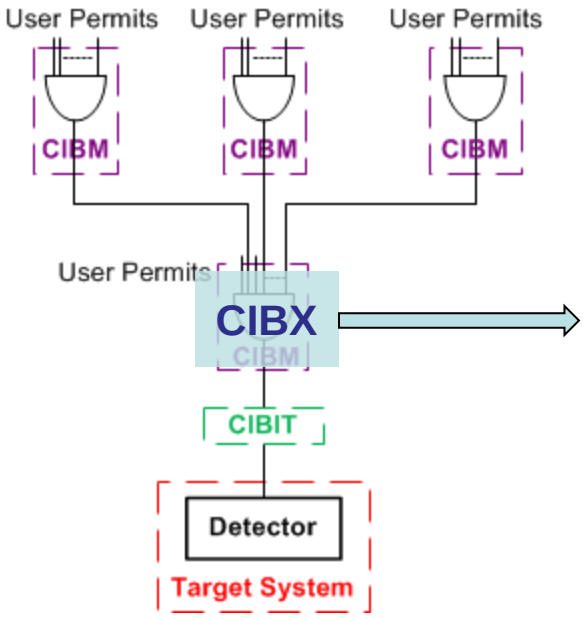
Tester CIBT

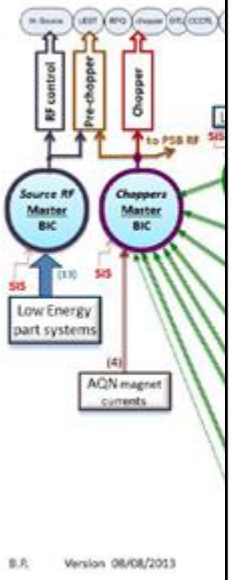
Interface with Target System CIBIT

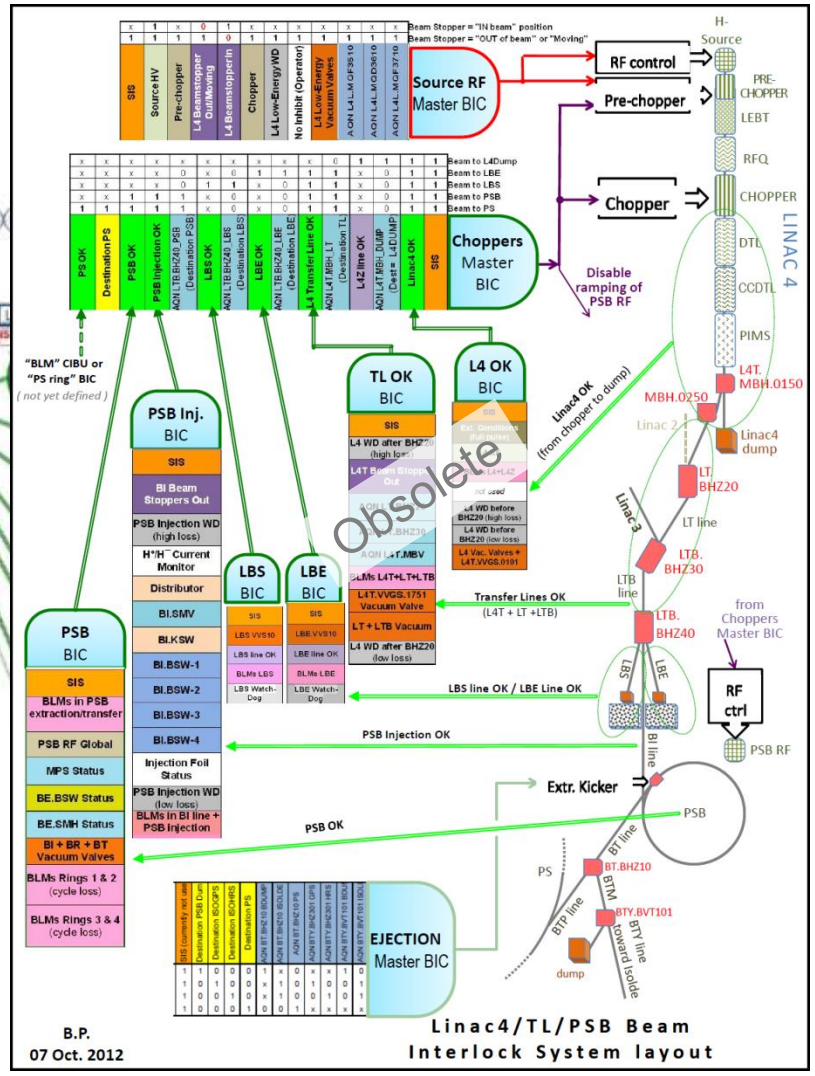
Master Matrix CIBX

CIBX – Controls Interlocks Beam Master MatriX

Real implementation in Linac4

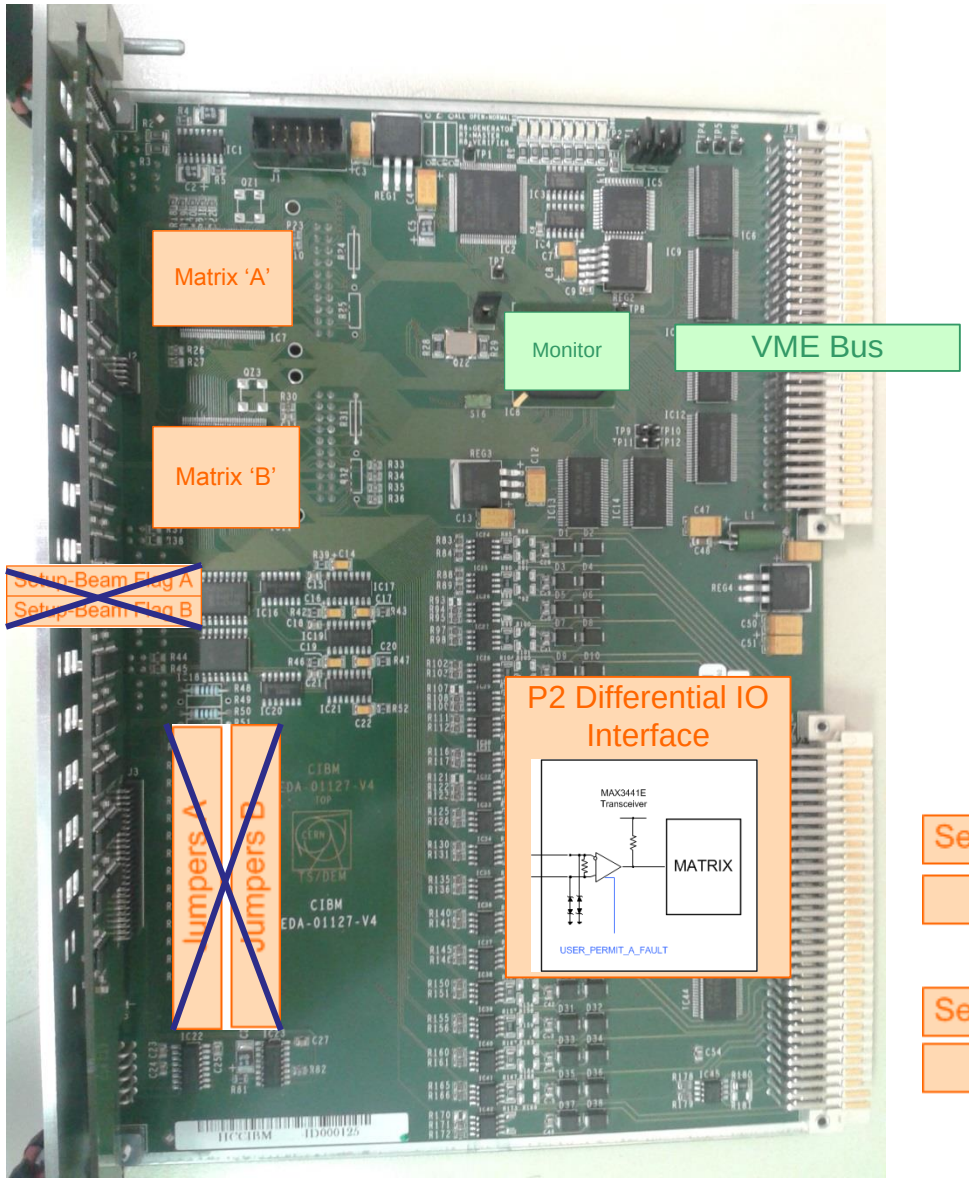






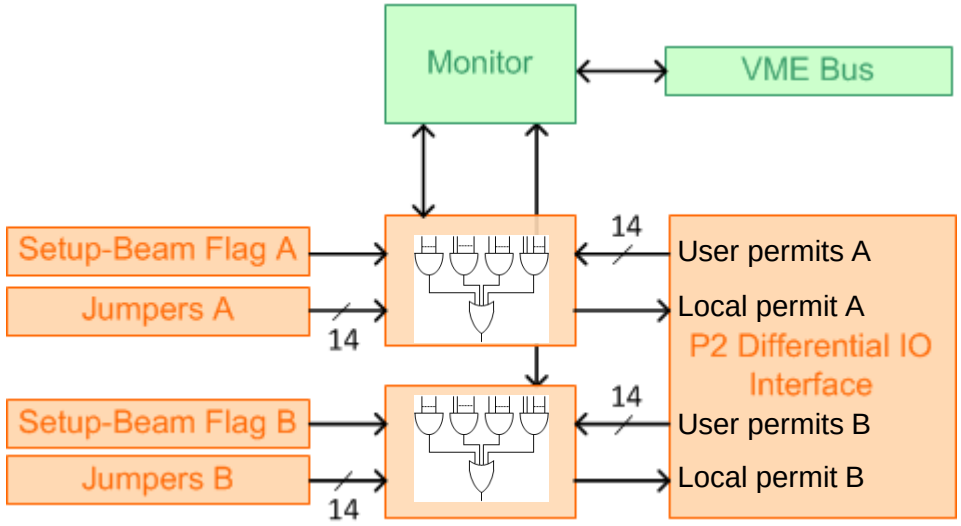
- The same hardware is used but the VHDL code changes
- 1 code for each type of **CIBX**

Master Matrix CIBX



CIBX main components:

- Max3440 transceivers for each user input
- 2 CPLD (Xilinx XC95288XL) for critical code
- 1 FPGA (Xilinx Spartan3) for monitoring
- VME interface connected on the monitoring
- ~~Each channel can be disabled by hardware (2*14 jumpers)~~
- ~~Up to 7 channels can be masked by software command~~



Master Matrix CIBX

Example of the Linac4 Source RF

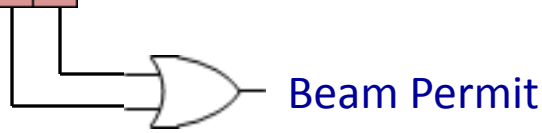
Linac4 Source RF			
0	SIS	1	1
1	Source Internal	1	1
2	Source HV	1	1
3	Pre-chopper	1	x
4	Source Beam Stoppers Out/Moving	1	0
5	Source Beam Stoppers In	0	1
6	Chopper	1	x
7	L4 Low-Energy Watchdog	1	x
8	L4 Low-Energy Vacuum Valves	1	x
9	AQN L4L.QUADS	1	x
10	RFQ	1	x
11	CCC Operator Switch	1	x
12	L4 Operator Switch	1	x
13	*not used*	x	x
14	*not used*	x	x
OUT	Beam Permit	1	1

SIS = Software permit

1 = User Permit must be TRUE to get the Beam Permit

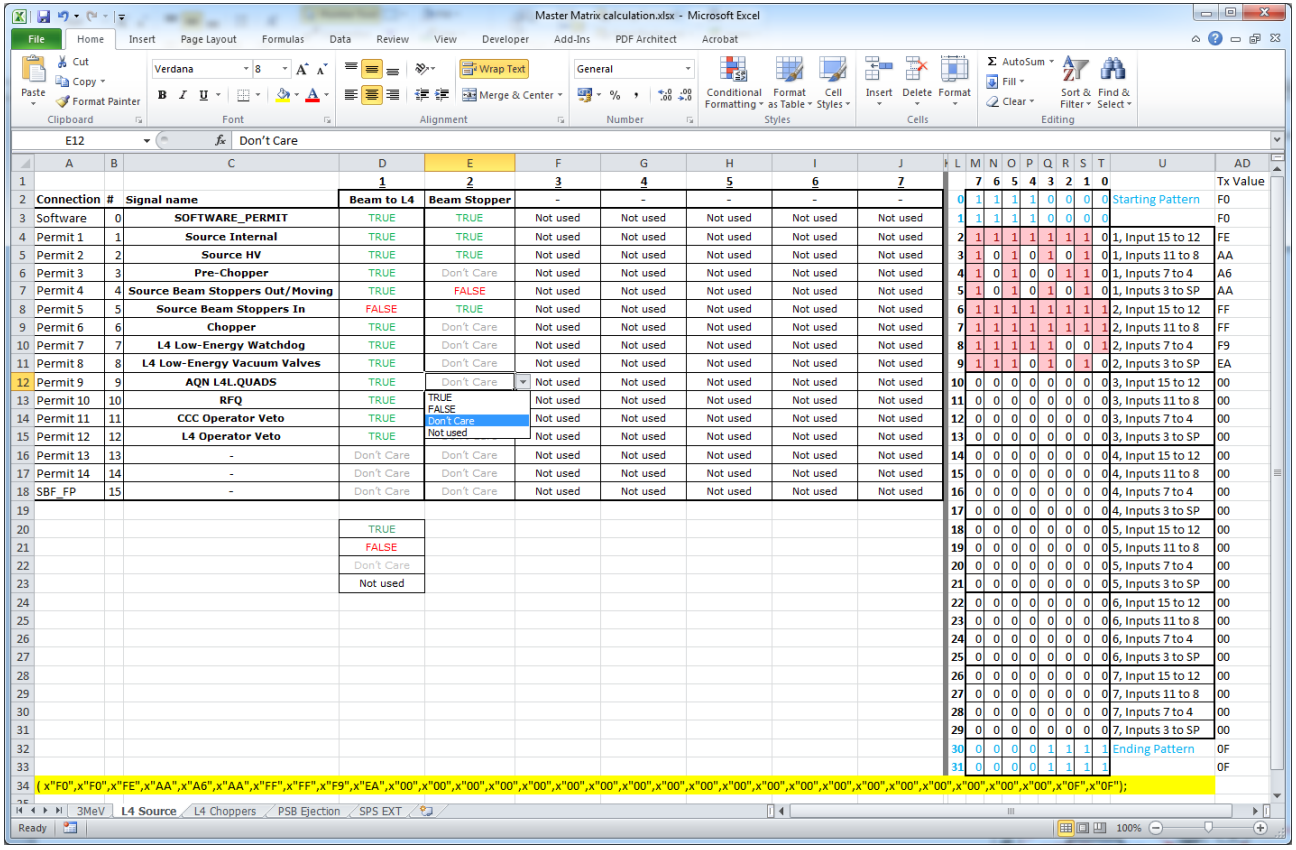
0 = User Permit must be FALSE to get the Beam Permit

x = User Permit is not taken into account



Master Matrix CIBX

- VHDL Critical code is implemented in a way that only little modification has to be done to modify the equations
- Use of a rom constant, directly generated from Excel…



- Monitoring code is the same that for the **CIBM** !

Outline

BIC architecture

- BIS hardware: simplified layout
- BIS in LHC
- BIS in Linac4

Manager CIBM

- Critical code
- Monitoring code

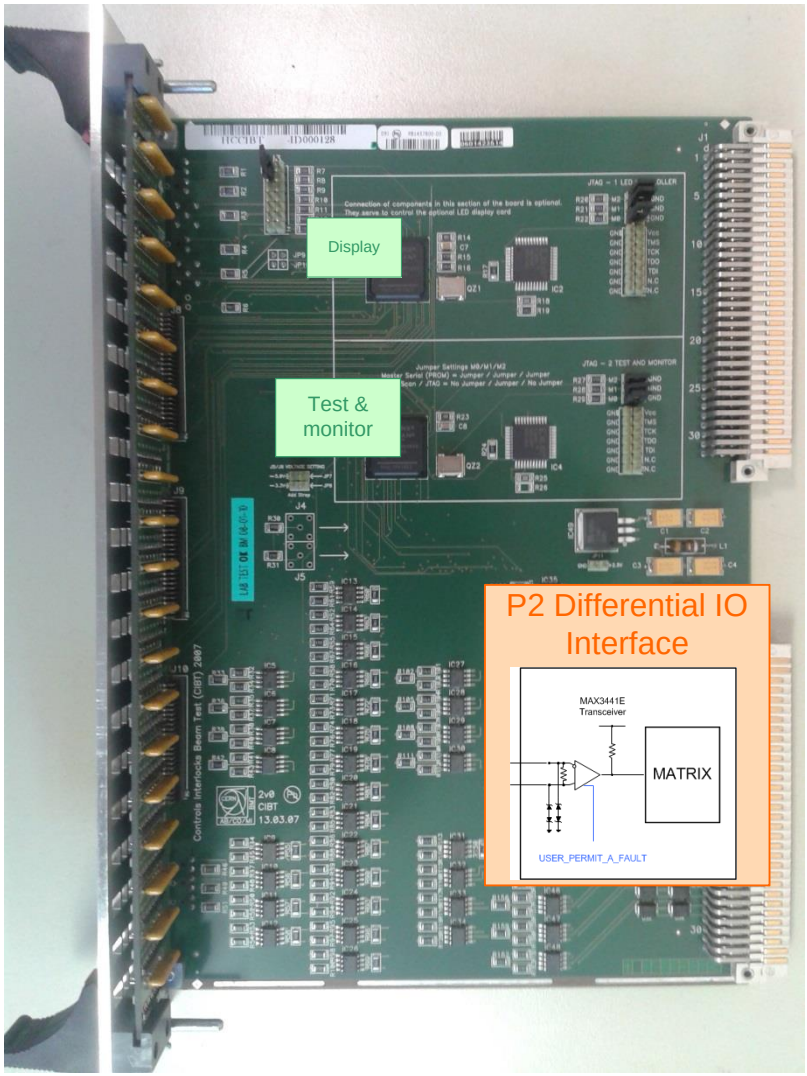
Master Matrix CIBX

Tester CIBT

Interface with Target System CIBIT

Tester CIBT

CIBT – Controls Interlocks Beam Tester



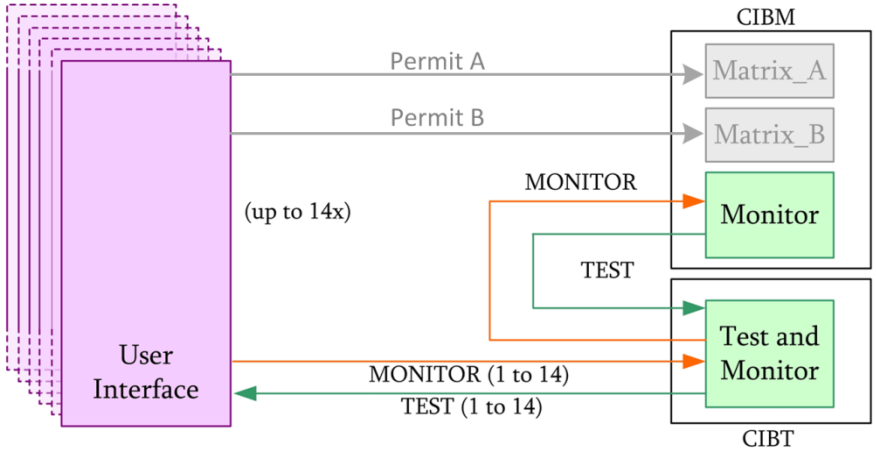
CIBT main components:

- Max3440 transceivers for each user interface input/output
- 1 FPGA (Spartan2) to test / monitor the User Interfaces (no redundancy)
- 1 FPGA (Spartan2) to control the display

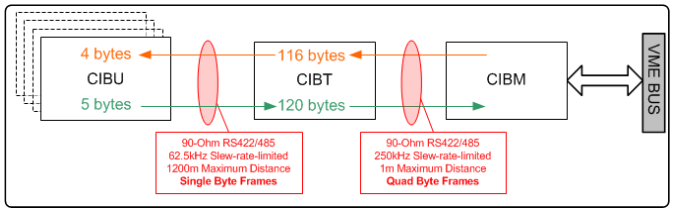
Tester CIBT

2 Functions

1.Test/Monitor the User Interfaces:



The **CIBT** is NOT connected to the VME bus

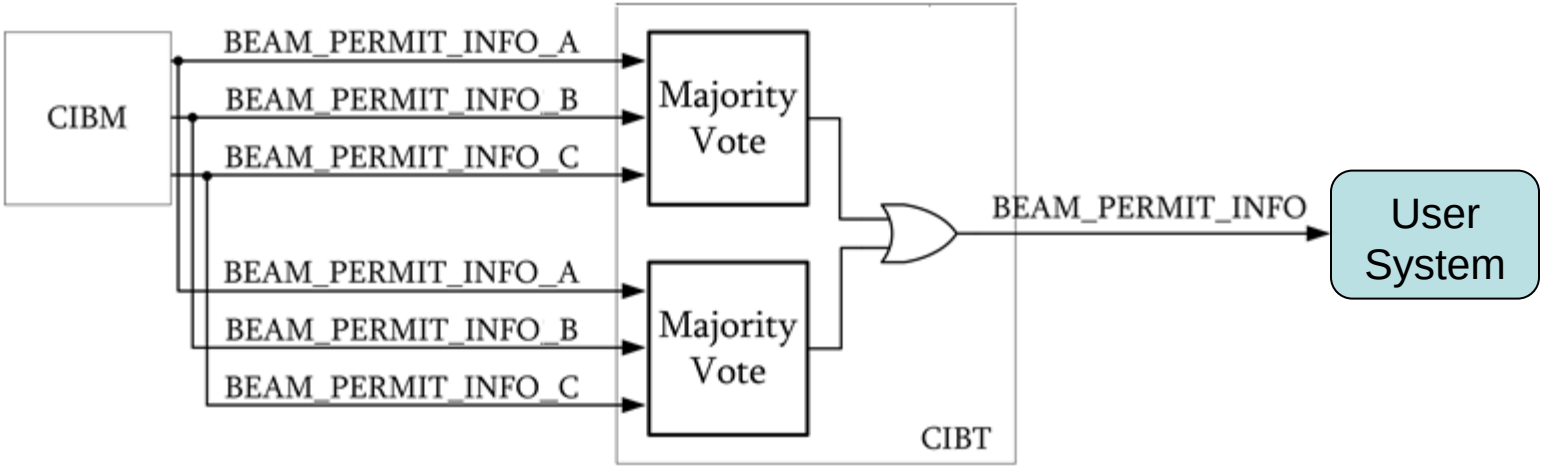


- The **CIBM/X** monitor writes a “test” buffer to the **CIBT**, which transmits this info to all User Interfaces
- The User Interfaces send a “monitoring” buffer to the CIBT, which save all the data and transmit them serially to the **CIBM/X**
- The monitoring data is stored in registers of the **CIBM/X**, then send to the supervision

Tester CIBT

2 Functions

2. Provide the Beam Permit Info (BPI) to the User Interface



- The BPI allows the User to know is there is beam in the machine
- This is not a critical signal, but it can be used by the User system to enter in test mode (when BPI is FALSE)
- Double triplication of the logic between the **CIBM** and **CIBT** to ensure integrity of the BPI

Interface Transmitter CIBIT

- The **CIBIT** recuperates the Local Permit of a **CIBM** to transmit it to the Target Systems by differential (2 wires) -> It becomes then the Beam Permit
- It is configured for the LEFT CIBM or Right CIBM
- It can send up to 4 Beam Permit to feed up to 4 Target Systems
- Each Beam Permit is duplicated (A and B) and send to the Target System by 1 cable
- Beam Permit A and Beam Permit B have each a monitoring included in the cable (2 more wires for the read back).

Discussion