Master thesis

The Finite State Machine in the Detector Control System for the ALICE Muon Forward Tracker

Hiroshima University, Graduate School of Science
Department of Physical Science
Quark Physics Laboratory
M163823

Tomoki Kobayashi

Supervisor : Assoc. Prof. Kenta Shigaki
Chief Examiner : Assoc. Prof. Kenta Shigaki
Vice-Examiner : Prof. Kouji Kawabata

February, 26, 2018

Abstract

Heavy ion collisions at ultra-relativistic energy produce a hot and dense medium, called the Quark Gluon Plasma (QGP). In order to investigate and reveal the property of the QGP, we have been performing the ALICE experiment. For the third and the fourth experimental periods (Run3 and Run4) from 2021, we are proceeding with plans to upgrade the ALICE detector. Installing the Muon Forward Tracker (MFT) is one of the plans. The MFT will allow us to distinguish between prompt and non-prompt J/ψ, to measure heavy flavours to lower p_{T} , and to investigate Chiral symmetry breaking and restoration.

In order to enable to study these research subjects, I am engaged in the development of the MFT and I have contributed to the development of the Detector Control System (DCS). The DCS is a system for detector control and it is needed in the experiment. In the DCS, the Finite State Machine (FSM) is a software used to control/monitor experimental devices as an interface to supervise experimental equipment and it allows us to handle a lot of experimental equipment easily and simultaneously by simplification and automation of both control and monitoring. The FSM is composed of a tree like control hierarchy and the state transition on each node of the tree. In the state transition, the FSM concept is used. Behavior of each node is modeled into "states" of FSM. A state changes into another state by an external event. Device's condition is modeled and state transition is implemented using this concept so that simplification and automation of control/monitoring are enabled. I have designed and mostly implemented the FSM for the MFT control: the control hierarchy and FSM on each node. The control hierarchy of the MFT mainly consists of three sub-trees: **LVPS** (Low Voltage Power Supply) for controlling the CAEN power supply modules, **Cooling** for monitoring the cooling system, and **Detector** for controlling CAEN low voltage channels and DCDC converters supplying power to the devices, temperature sensors, and so on. I also have carried out actual machine tests of the FSM with CAEN power supply equipment and a temperature sensor, which are basic components for the MFT control. Through these tests simulating the real MFT control, I confirmed that turning channels ON/OFF by the FSM commands and state transition by the commands and/or device's condition work well. I have been expanding the hierarchy into the real MFT's one and it will be finalized in 2018.

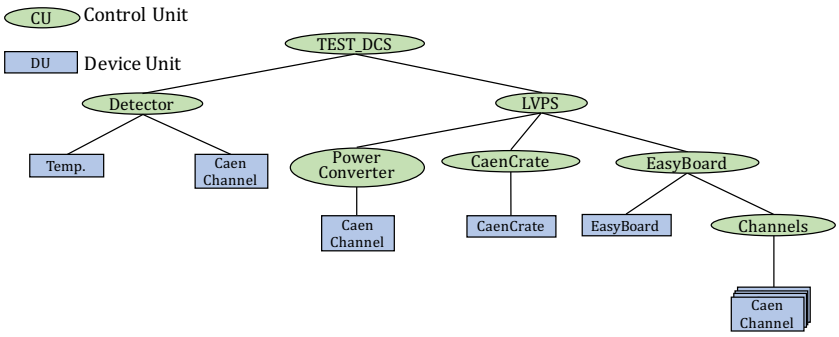

Figure 1: The control hierarchy of the FSM for the actual machine test

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Quark Gluon Plasma

Quantum Chromodynamics (QCD) explains quarks and gluons are bounded in hadrons and we cannot take them out alone from hadrons at low temperature and density. However, at extremely high temperature and/or density, QCD expects that they are released from the confinement and move freely. The hot and dense medium like this is a state of matter which is called the Quark Gluon Plasma (QGP).

1.2 High Energy Heavy Ion Collisions

The hot and dense matter can be made by high energy heavy ion collisions. At the LHC, which is the world's largest particle collider and operated from 2009 in European Organization for Nuclear Research (CERN), the hot and dense matter is made by colliding lead nuclei at 99.7% of light speed or faster and a lot of researchers are studying it.

1.3 The ALICE Experiment

The ALICE experiment we are taking part in stands for A Large Ion Collider Experiment. ALICE is aiming for heavy ion collisions only in the LHC experiments. When colliding heavy nuclei, thousands of particles are produced. The ALICE detector is designed to detect a lot of particles in wide transverse momentum range. The ALICE detector can be divided into three parts: (i) central barrel, which covers near the collision point ($|\eta| < 0.9$), (ii) muon arm, which detects muons at forward rapidity ($-4 < \eta < -2.5$), (iii) global detector, which characterizes of the collision [1, 2]. (Figure1.1)

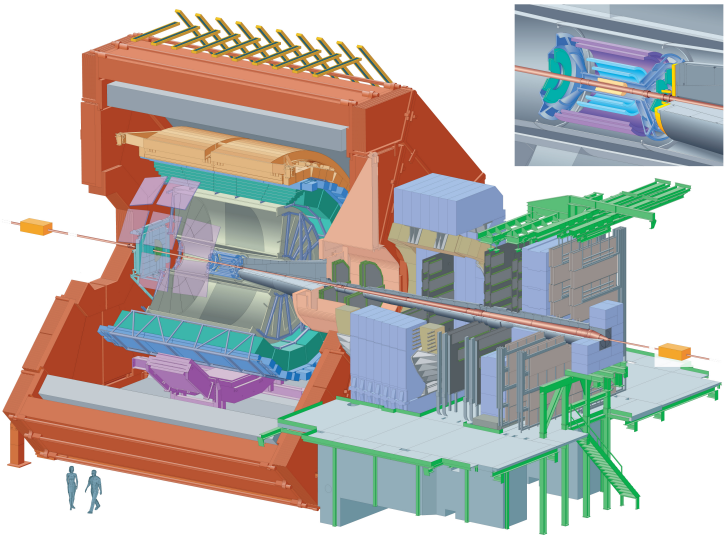

Figure 1.1: The ALICE detector [2]

1.4 Muon measurements

Photons and electrons are widely used as probes to study QGP. That's because photons and electrons have no strong interaction with QGP, so that they can take out the information of it. However, highly accurate analysis is difficult because there are a lot of background particles produced by pion decays due to their very light masses. Analysis with muons, which are leptons as same as electrons but have heavier masses, allows highly accurate analysis with very small background produced by pions.

1.5 The ALICE Upgrade Plans

The ALICE collaboration has plans to upgrade the ALICE detector and readout technologies during LHC Long Shutdown 2 (LS2, 2019-2020) [4, 3]. Even though a lot of physics data was accumulated and analyzed from the beginning of the LHC operation, there remains studies which can be hardly measured due to statistics and measurement accuracy. In addition, LHC is targeting to increase the luminosity up to $6 \times 10^{27} \mathrm{cm}^{-2} \mathrm{s}^{-1}$ and the collision rate up to 50 kHz in Pb-Pb collisions at the third experimental period (Run3) from 2021 [4] . However, current detectors's readout system and data taking method cannot cope with these. Therefore, we are going to upgrade the ALICE detector and introduce a new data acquisition method. The main upgrade plans are shown in following:

- ITS: The impact parameter resolution, tracking performance, momentum resolution, and readout rate will be improved. [5]
- TPC: The present MWPC based readout chambers will be replaced by GEM to improve readout rate. [6]
- TOF, TRD, Muon Spectrometer, ZDC, PHOS, EMCal, and HMPID: The readout electronics will be upgraded to cope with 50kHz Pb-Pb collisions. [4]
- MFT: The new muon detector will be added between the interaction point and the Muon Spectrometer's front absorber in order to improve tracking resolution. [7, 8]

- O^2: The estimated data amount of Run3 will be 3.4 TB/s. The O^2 system allows partial calibration and data reconstruction online and compresses the original raw data to reduce the data amount.[9]

1.6 The Muon Spectrometer and the Muon Forward Tracker

The Muon Spectrometer is currently operated as a muon detector in ALICE. It is located in front of the interaction point covering the pseud-rapidity region $-4 < \eta < -2.5$. It consists of the front absorber, the dipole magnet, the tracking system, the iron wall, and the trigger system. The front absorber, made of carbon, concrete, and steel, rejects all charged particles except muons coming from the interaction point. The dipole magnet provides a magnetic field. The tracking system consists of five tracking stations, each one is made of two cathode pad chambers with a spatial resolution of about 100 μm in the bending direction. The iron wall rejects the residual secondary particles. It is assumed that particles detected by the trigger system are muons since only muons can arrive at the trigger system through the absorbers [1].

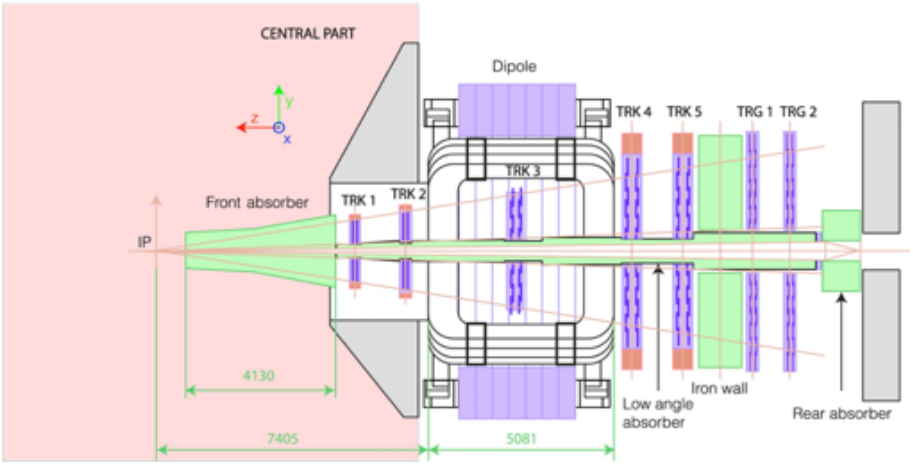

Figure 1.2: Schematic of the ALICE Muon Spectrometer [7]

Only muons can be selected by the front absorber, but the details of the vertex region are smeared out due to multiple scattering induced on the muon tracks in the absorber. Thus, for example, it is difficult to distinguish between prompt and non-prompt J/ψ and mass reconstruction since muons' generated points and the angle of the muon pair can be hardly measured. To resolve these problems, we are going to install a new detector, the MFT, stands for Muon Forward Tracker.

The main purpose of installing the MFT is to measure the interaction point accurately. For this purpose the MFT will be installed between the interaction point and the front absorber, so that it will become possible to measure muons' generated points by avoiding the multiple scattering in the absorber. Though the MFT detects all charged particles since it will be located in front of the absorber, only muon tracks are identified with requiring matching tracks measured by the MFT with tracks measured by the Muon Spectrometer. The MFT will cover the pseud-rapidity region $-3.6 < \eta < -2.5$. The MFT will enable the following measurements [7]:

– Heavy flavour

- Separating prompt J/ψ from non-prompt J/ψ (J/ψ coming from b-hadron feed-down). This allows us to measure prompt J/ψ R_{AA}, and to investigate quarkonium

production process and J/ψ excess at very low p_{T} in ultra-peripheral collisions.

- Separating between muons from heavy flavour and from pion and kaon. This allows measurement of heavy flavour at lower p_{T}.

– Light flavour

- Mass reconstruction with higher mass resolution. This allows us to investigate chiral symmetry breaking and restoration.

1.7 Hardware components of the MFT

The MFT detector consists of two Half MFTs: the upper Half MFT and the lower one, which have a same structure. A Half MFT is composed of five Half Disks and a Power Supply Unit (PSU). Each Half Disk can be divided into two Half Planes: the front Half Plane and the back one. The PSU can be divided into the front and the back plane as well. The Heat Exchanger (HE) is located between two Half Planes to cool the equipment. Ladders are located on each Half Plane, while DCDC converters, providing power for ALPIDEs, and GBT-SCAs are installed on the PSU. Ladders on each Half Plane is divided into four Zones. Power is supplied Zone by Zone. 2-5 ALPIDEs are installed on each Ladder. The ALPIDE, stands for ALICE pixel detector, is a new pixel sensor developed for the ALICE sub-detectors.

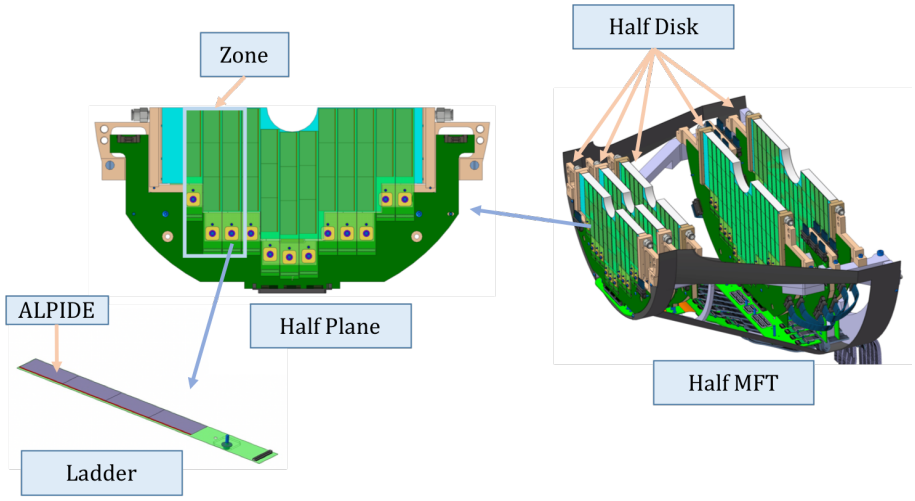

Figure 1.3: The MFT structure [8, 34, 35]

1.7.1 Low Voltage distribution

The power supply equipment used in the MFT mainly consists of the CAEN EASY (Embedded Assembly SYstem) system [18]. It is summarized in the following table (Table 1.1) and Schematic drawings (Figure 1.4). EASY BUS is used for communication between CAEN modules, and Ethernet is used to connect SY4527 with a computer.

Table 1.1: The CAEN supply devices used in the MFT

Model	Type	Description
SY4527	Crate	Universal multichannel power supply system
A1676A	Controller	Branch controller for the EASY system
A3486	Power Converter	48 V power supply for the EASY system
EASY3000	Easy Crate	Magnetic field and radiation tolerant crate for hostile area, the crate for the EASY system
A3006	Easy Board	6 channels ±(4-16)V/6A/90W power supply board, used for the ALPIDEs' Back-bias and GBT-SCA power source
A3009	Easy Board	12 channels 2-8V/9A/45W power supply board, used for the ALPIDEs LV power source
A3100	Easy Board	1 channel 2-8V/100A/600W power supply board, used for the RU board power source

The output voltages from the channels of A3009, A3006 and A3100 are changed by a DCDC converter or an inverter amplifier so that it is supplied to devices as following [36]:

- The power source for the ALPIDEs
  A3009 → DCDC converter → Zone
- The reverse substrate bias voltage of the ALPIDEs
  A3006 → Inverter amp → ALPIDEs
- The power source of the GBT-SCAs
  A3006 → DCDC converter → GBT-SCA
- The power source of the RU boards
  A3100 → DCDC converter → RU board

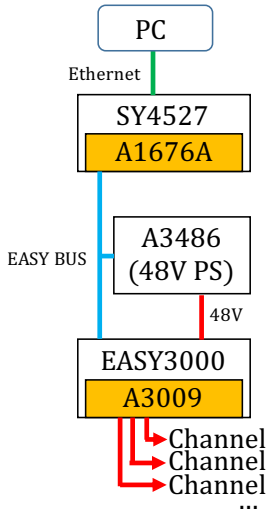

Figure 1.4: The schematic drawing of the CAEN system architecture

1.7.2 Readout electronics

GBT-SCA

GBT-SCA is an ASIC (application specific integrated circuit) for detector operation and/or the devices monitoring. It has 1 I^2C master, 1 SPI master, 1 JTAG master, 32 general I/O, 32 analog input, 4 analog output ports for connecting to the devices [21].

Readout Unit (RU)

The RU board has FPGAs, GBTXs (a flexible link interface chip and responsible for high speed bidirectional optical link) and a GBT-SCA to receive detector data including physics data and DCS data from the connected pixel sensors and control them [37].

1.7.3 Temperature sensors

The MFT has a lot of temperature sensors. These are important for both software and hardware safety interlock systems. A temperature sensor will install in each Half Plane and each DCDC converter cooling block, and each GBT-SCA and RU board has a on-board temperature sensor.

1.7.4 Cooling devices

MFT will be cooled by both water-cooling and air-cooling system. The FSM also reflects the system condition and monitor it. Concerning the cooling system, EN-CV Detector Cooling Section is in charge of the cooling system development and its FSM design and currently developing and designing by them. At the moment, therefore, temporary states (READY, NOT_READY, ERROR) have been designed in the FSM.

1.8 Purpose of this thesis

Physics that can be reached with the MFT from the LHC Run3 is very interesting and in order to enable to study them, I have decided to engage myself in the development of the MFT and I have contributed to the development of the Finite State Machine (FSM) in the Detector Control System (DCS), which is important to control the detector efficiently and carry out reliable and safety experiment.

Chapter 2

Detector Control System

2.1 Overview of the Detector Control System

The ALICE detector is located underground and we cannot control it in person not to be exposed to radiation and strong magnetic field around the detector. That's why we have to control and monitor it remotely. Moreover, each sub-detector consists of a lot of experimental equipment. In the case of the MFT, it consists of more than 1000 devices: 920 detection sensors, LV equipment, temperature sensors and more. Therefore a supervision system by which we can control them easily is needed. The DCS allows us to control a lot of equipment remotely and easily. For instance, we can configure voltages applied to detection sensors and monitor devices' condition by the DCS.

DCS consists of three layers: the supervision layer, the process management layer and the field management layer. The supervision layer is in the ALICE control room. We control and monitor detectors with the FSM and GUI control panels from there. The process management layer is in the counting room, which is located between the control room and the cavern, and it contains computing systems such as SCADA system and communication protocols, power supply systems, and so on. The field management layer is in the cavern. It consists of sensors, readout electronics, power supply systems, field busses, and so on.

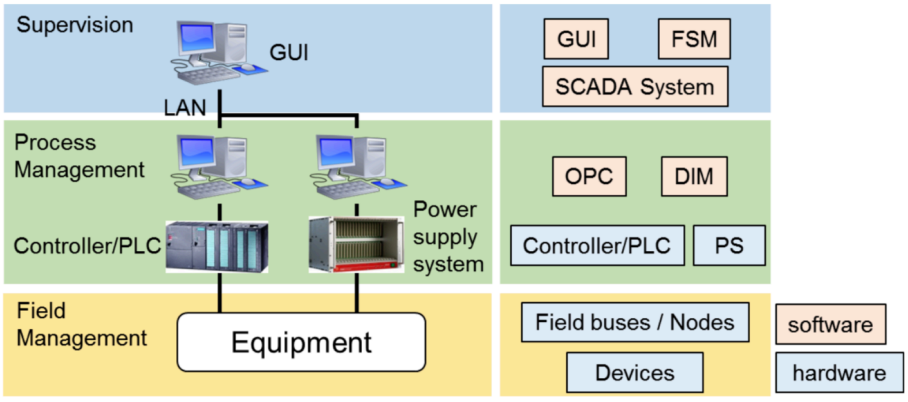

Figure 2.1: Architecture of the DCS [10]

2.2 DCS Software

2.2.1 SCADA system and WinCC OA

SCADA, stands for Supervisory Control and Data Acquisition, is one of the industrial control systems, which carries out supervision of a system and process control. In a factory, for instance, it allows us to manage conditions of power supply and production centrally, to operate equipment, and to handle alarms [11].

WinCC OA, stands for WinCC Open Architecture, is a SCADA application designed by ETM, a company of the Siemens group. WinCC OA provides following main components and tools [11]:

- A run time database

  Data coming from the devices is stored, and can be visualized and processed by accessing it.

- Archiving

  Data in the run-time database can be stored in long term storage, and users can retrieve it later by user interfaces or other processes.

- Alarm Generation & Handling

  Alarms can be generated by defining error situation in advance. Alarms are archived in an alarm database and can be displayed on an alarm display.

- A Graphical Editor

  Allows users to design their own user interface.

- A Scripting Language

  Allows users to handle data stored in a database from user interfaces and/or a background processes. The language is called CTRL(read control) scripts.

- A Graphical Parameterization tool

  Allows users to define structure of a database, which data should be archived and so on.

WinCC OA is composed of several processes, which are called Managers. WinCC OA is composed of the following Managers (Figure 2.2) [11]:

- The Event Manager (EVM) is responsible for all communications. It receives data from any Drivers (D) and stores it in a database.

- The Data Base Manager (DBM) provides us an interface to a database.

- User Interface Managers (UIM) can acquire device's data from a database and send data to a database which is to be sent to device. They can also request to keep an "open" connection to the database and be informed (for example to update the screen) when new data arrives from device.

- Ctrl Managers (Ctrl) provide for any data processing as "background" processes, by running a scripting language. The language is like "C" with extensions.

- API Managers (API) allow users to write their own programs in C++ using an API(Application Programing Interface) to access the data in the database.

- Drives (D) provide the interface to the devices to be controlled. These can be WinCC OA provided drivers like Profibus, OPC, etc.

WinCC OA can run on both Windows and Linux and it can be used as a distributed system by connecting machines by a network (Figure 2.3).

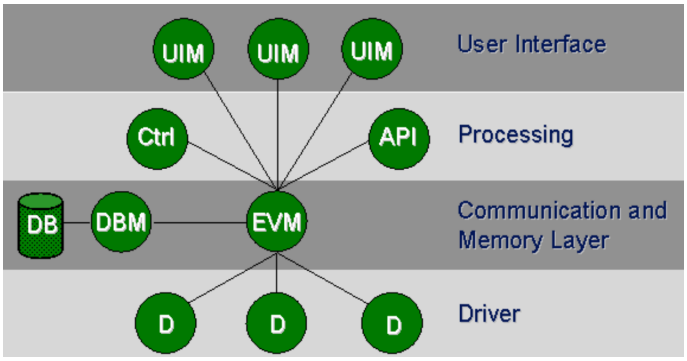

Figure 2.2: Architecture of WinCC OA [11]

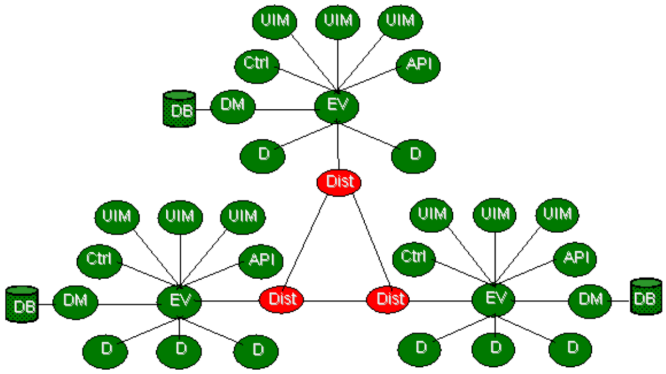

Figure 2.3: Distributed Systems on WinCC OA [11]

Data from devices is handled as Data Point Element (DPE), which is used to get values by the FSM and/or user interface panels and to configure devices. Data Point Type (DPT) describes a structure of device data. Users can define their own DPTs depending on devices they use. Here, Figure 2.4 is an example of HvChannel's DPT. Data Point (DP) is hierarchical structure which consists of DPTs. Figure 2.5 is an example of HaChannel's DP. This DP consists of two DPTs, which are Channel1 and Channel2. In this figure, DPEs are v0, state, currentLimit, status, vMon and iMon.

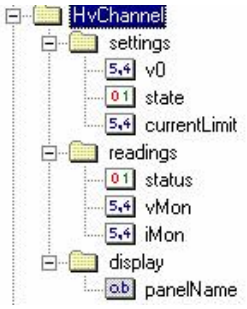

Figure 2.4: An example of DPT [11]

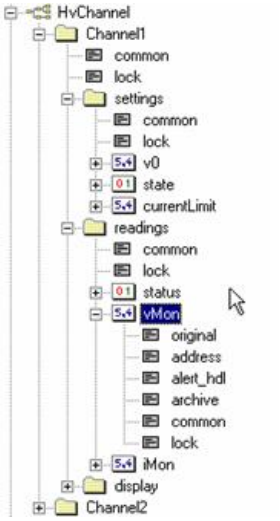

Figure 2.5: An example of DP [11]

2.2.2 JOint Control Project Framework

In order to reduce manpower and costs to develop the DCS, CERN's IT/CO group developed JOint Control Project Framework (JCOP, JCOP Fw) as a common framework to the four LHC experiment. This framework is based on WinCC OA and the purpose of this framework is to provide common guidelines and tools to develop the DCS. We can

control experimental equipment such as power supply and readout devices by using JCOP Fw [13].

2.2.3 OPC

OPC, stands for OLE for Process Control, is a standard specification for safe and reliable data exchange. OPC is developed and supported by OPC Foundation [14, 15]. An OPC client such as WinCC OA and an OPC server, which is developed and provided by the device maker, are needed to use OPC communication. An OPC client communicate with devices through an OPC server. In this thesis, OPC is used to connect with the CAEN power supply devices.

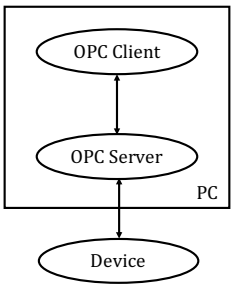

Figure 2.6: Architecture of OPC communication

2.2.4 Distributed Information Management system

Distributed Information Management system (DIM) is a communication protocol developed by CERN [16]. In this thesis, DIM is used to connect with ELMB.

2.3 The Finite State Machine

In the DCS, the Finite State Machine (FSM) is a software used to control/monitor detectors and experimental devices. I have designed and implemented the FSM in the MFT DCS development. The FSM is a interface to control/monitor experimental equipment and it allows us to control/monitor a lot of experimental equipment easily and simultaneously by simplification and automation of control/monitoring. The FSM is composed of the tree like control hierarchy and the state transition on each node of the tree hierarchy.

2.3.1 The Control Hierarchy

In the ALICE DCS, sub-detectors are controlled and monitored by using tree like control hierarchy modeling sub-systems and devices. This tree structure is composed of nodes and all the nodes except the top node have only one parent node. There are two kinds of nodes: Control Unit (CU) and Device Unit (DU) [23, 24, 25].

- Device Unit

  DUs have a role as an interface between software world and real world's devices. DUs have no children nodes. They do not implement logic behavior. They receive:

  - "commands" and act on the device
  - device data and translate it into a "state".

- Control Unit

  CUs are defined logically and they model and control the sub-trees below them. Their own state are determined by the children's states, and they can send commands to children depending the children's states. The logic behavior of a Control Unit is expressed in terms of Finite State Machines. State transition can be triggered by:

  - command reception(either from its parent of from an operator)
  - state changes of its children.

The number and kinds of DUs are decided by the number and kinds of controlled devices and the values of them. The hierarchy can have an arbitrary number of levels and CUs as necessary. An operator can control devices by sending commands by the top node. The commands are sent to DUs through CUs below the top node and devices are controlled. Conversely, DUs' states reflecting the devices' condition is propagated to the top node through CUs above the DUs. We can control the whole detector, recover from error situations, and automize operation by using this mechanism. The functions and the behavior are modeled and implemented by Finite State Machine (FSM) concept.

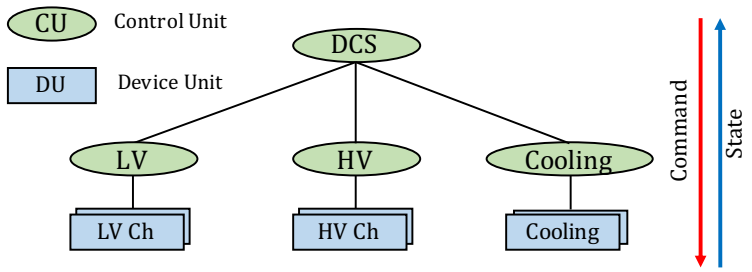

Figure 2.7: A Sample Tree

2.3.2 Finite State Machine concept

In controlling detectors, Finite State Machine (FSM) concept is used to control and monitor experimental equipment and the whole system by modeling them. FSM is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any time. In this concept, input values from real world's devices are modeled as states and one of the states are chosen. The FSM can change from one state to another in response to actions that executed by the operator and/or some external events. The change from one state to another is called transition.

The transition of states is visualized by means of state diagrams (Figure 2.8).

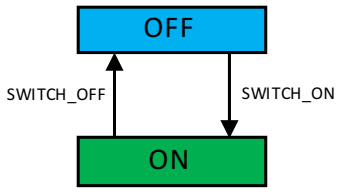

Figure 2.8: A Sample State Diagram

See Figure 2.8. In this sample state diagram, for example, the SWITCH_ON (SWITCH_OFF) command can be executed when the state is OFF (ON). The diagram shows that the FSM

changes the state between ON and OFF by commands reflecting the device's power condition. In the DCS, we define FSM on each node in the control hierarchy, and control and monitor the whole system by using the FSM.

By using FSM connecting with the devices, we can:

- control the devices (e.g. turn on/off the devices, recover from error situation)
- configure the devices (e.g. configure the voltage/current of the power supply and the thresholds of them)
- automize operation of the devices (e.g. automize the order of turning the devices on, turn the devices off automatically when error situation)
- monitor the devices (e.g. load the values of temperature sensors, translate the devices' condition into state)

As I explained above, the software for control/monitoring of the detector, consisting of the tree like control hierarchy and a state transition as FSM on each node is called the FSM.

2.3.3 Partitioning

It is possible to control and/or monitor a part of the whole system or a sub-system independently and concurrently by cutting off a sub-tree from the whole tree. Operators and experts can control a (sub-)tree by becoming the "owner". There are four partitioning modes in the JCOP FSM. (Figure 2.9)

- Included

  A child is fully controlled by its parent.

- Excluded

  A child is not controlled by its parent.

- Manual

  A parent does not send commands to its child.

- Ignored

  A parent ignores its child's state.

The owner operator has released ownership by excluding the sub-tree so that another operator can control the sub-tree. This is, for example, made good use of in the case of ERROR in a sub-tree. The expert can work with the sub-tree to handle the ERROR by excluding the sub-tree by the operator. After the work by the expert, the operator can include the sub-tree so that the experiment restarts with the whole system .

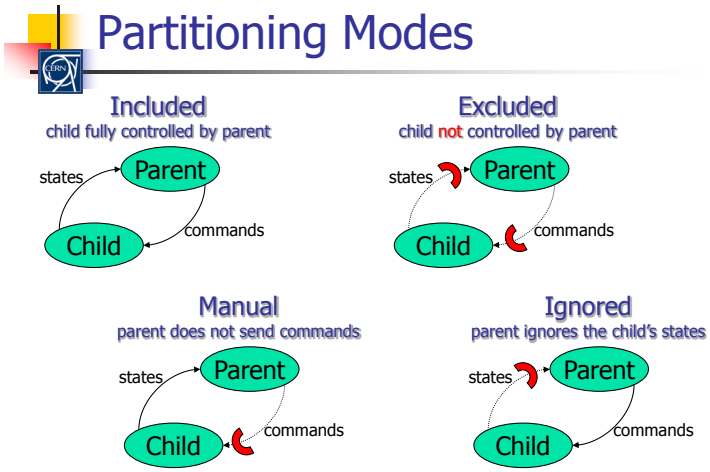

Figure 2.9: Partitioning modes available in the JCOP FSM [25]

2.3.4 State Management Interface

State Management Interface (SMI++) is a software framework to design and implement the distributed control system. This is based on the "State Manager" concept developed by the DELPHI experiment. In this concept, the real-world system is described as a collection of objects behaving as FSMs. The real-system is controlled by received commands that trigger actions and FSM's transitions on the SMI++ objects [29, 30, 31].

2.3.5 State Manager Language

State Manager Language (SML) is used to describe the object model of the experiment. SML allows for detailed specification design of the objects such as their states, actions and associated conditions. The main characteristics of this language are:

- Finite State Logic

  Objects are described as FSMs. The main attribute of an object is its state. Commands sent to an object trigger actions that can change its state.

- Sequencing

  An action performed by an abstract object is specified by a sequence of instructions which mainly consist of commands sent to other objects and logical tests on states of other objects.

- Asynchrounous

  Several actions proceed in parallel. A command sent by object A to objectB does not suspend the instruction sequence of object A. Only a test by object A on the state of object B suspend the instruction sequence of object A if object B is still in transition.

- AI-like rules

Each object can specify logical conditions based on states of other objects. When satisfied, these will trigger an action on the local object. This provides the mechanism for an object to respond to unsolicited state changes of its environment.

An example SML code is shown in the following:

```
Object : LVPS
    state : ON
            action: GO_OFF
                do GO_OFF $ALL$FwCaenChannel
                move_to MOVING_OFF
    ...
    state : MOVING_OFF
        when ( $ALL$FwCaenChannel in_state OFF ) move_to OFF
    ...
    state : OFF
            action: GO_ON
                do GO_ON $ALL$FwCaenChannel
                move_to MOVING_ON
    ...
    state : MOVING_ON
        when ( $ALL$FwCaenChannel in_state ON ) move_to ON
    ...
Object : FwCaenChannel
    state : ON
            action: GO_OFF
    ...
    state : OFF
            action: GO_ON
    ...
```

In this example, two objects are declared: LVPS and FwCaenChannel. LVPS is a CU that controls the CAEN low voltage channel at the supervisory layer, while FwCaenChannel is a DU representing the corresponding physical low voltage channel. In both objects, the list of available states and transition conditions are shown. For example, in object LVPS, the action "GO_OFF" can be executed when the state is ON. The action consists of sending the command "GO_OFF" to object FwCaenChannel and the state is changed into MOVING_OFF. In MOVING_OFF state, the state moves to OFF when all the objects FwCaenChannel have reached the OFF state [29, 30].

Chapter 3

The design of the Finite State Machine for the MFT

The FSM I have designed for MFT control is described in this chapter.

3.1 The Control Hierarchy

I have designed the control hierarchy of the MFT in reference to the hardware architecture (Figure 3.1) [10, 38], summarized by K. Yamakawa, a member of the Quark physics Laboratory. Figure 3.2 is the control hierarchy. A green elliptic node indicates Control Unit and a blue rectangular node indicates Device Unit. **MFT_DCS** is the top node of the hierarchy and it has two children: **Detector** and **Infrastructure**. **Infrastructure** has **LVPS** (Low Voltage Power Supply) for controlling the CAEN power supply modules and **Cooling** for monitoring the cooling system as children. Otherwise in the sub-tree below **Detector**, CAEN low voltage channels and DCDC converters supplying power to the devices, temperature sensors, etc. are controlled and monitored. The sub-tree structure follows the basic MFT hardware structure: Half MFTs, Half Disks, Half Planes, Zones, Ladders, and ALPIDEs. The nodes controlling the devices are added to the sub-tree with taking into account the area which they can affect so that detector operation and devices control becomes easy. For instance, GBT-SCA is installed into each Half Plane. If the GBT-SCA gets unusable due to extreme high temperature or damage, the Half Plane becomes unusable. In the FSM, each **Half_Plane** has one **GBT_SCA** node and if **GBT_SCA** moves to ERROR, **Half_Plane** moves to ERROR automatically, indicating that the Half Plane cannot be used, and turning off the power source to all the devices in the Half Plane can be implemented easily. The modules such as DCDC converters and low voltage channels are controlled in units of Half Planes, thus their CUs are implemented below **Half_Plane**s. Otherwise ALPIDEs power source can be controlled in units of Zones because each DCDC converter or each inverter amplifier output is split into 4 Zones and there are corresponding switches. Each Zone also has one RU board. Thus, **Zone** has their nodes as children and it allows us to control them in units of Zones.

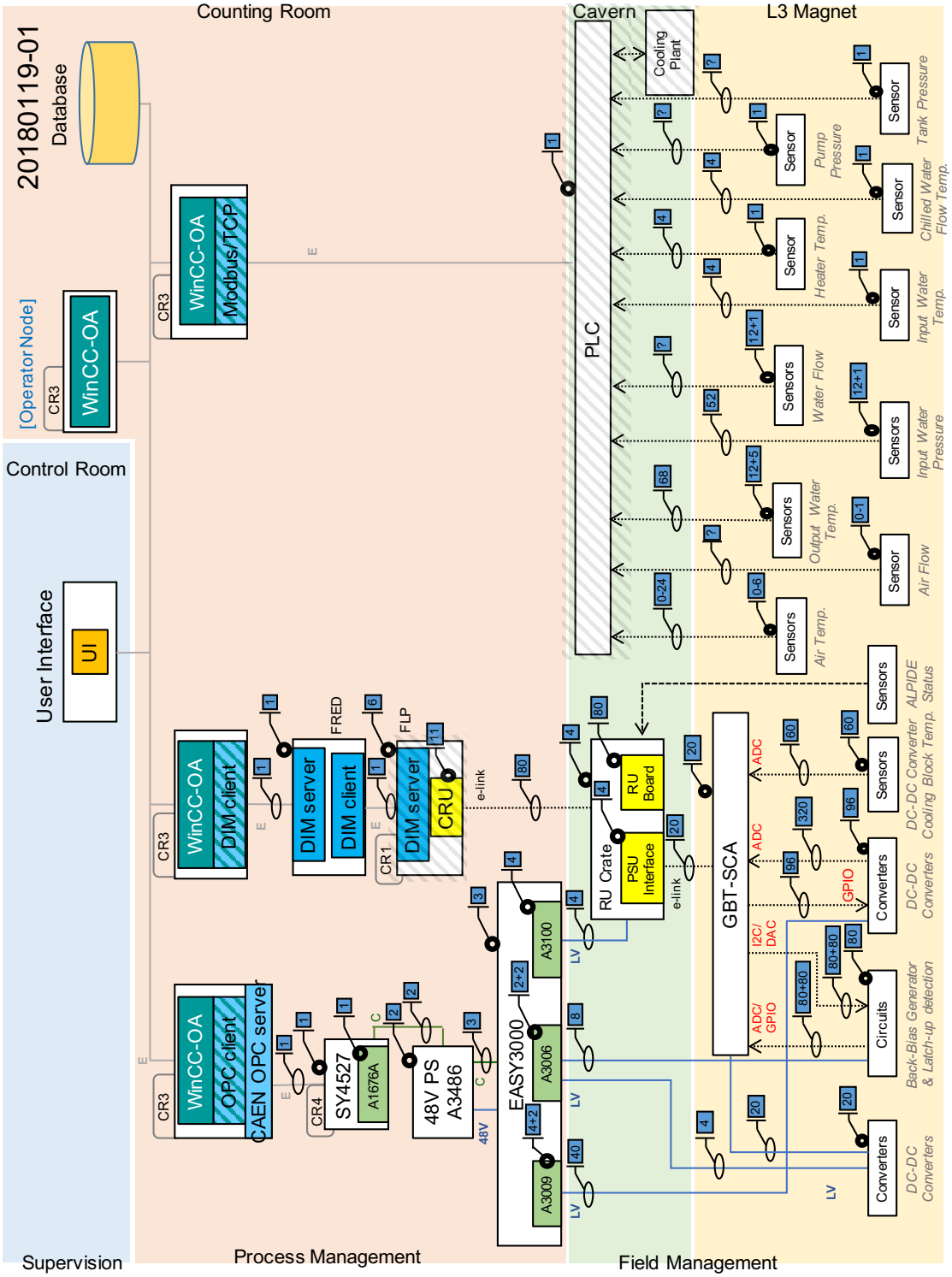

Figure 3.1: Hardware architecture of the MFT DCS [38]

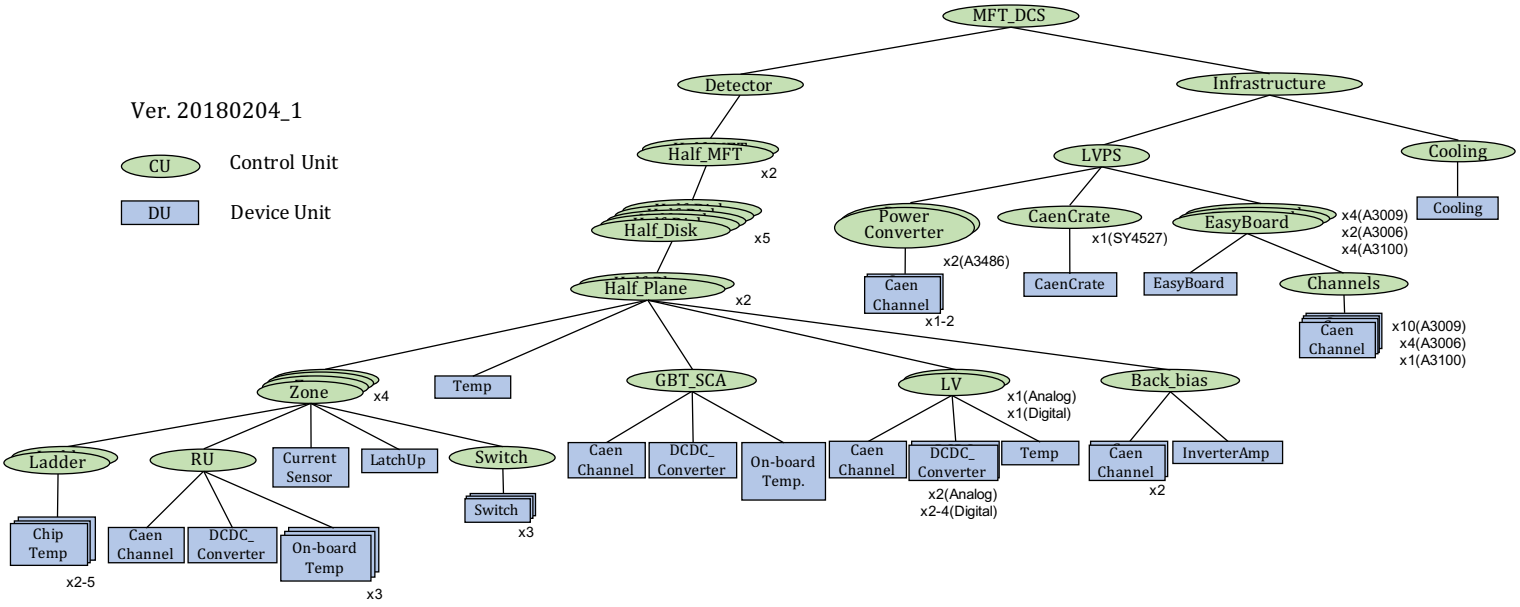

Figure 3.2: Control Hierarchy for the MFT

3.2 Implementation Strategy

In order to construct the complex control system, I designed the control hierarchy at first, and the possible states, their transition, transition conditions, and commands executing actions secondly. In the ALICE DCS, the top node FSMs of all the sub-detectors have to use common states based on guidelines provided by the ALICE controls coordination [27, 28], so that everyone can control the sub-detectors easily. On the other hand, the states in all the nodes except the top one can be determined freely since each sub-detector has specific equipment and requirements for operation. When designing the control hierarchy, I started it from the top level node to lowest level (DU), i.e. top-down approach, in order to come up with the overall tree-like structure. When designing the hierarchical FSMs, in contrast, I designed firstly possible states and their transitions of DUs taking into account their behaviour and values which are controlled/monitored, and secondly FSMs of CUs between DUs and the top node keeping consistency with the top level FSM, i.e. bottom-up approach. Thus, during implementation phase, a combination of bottom-up and top-down strategies have been used for improving system performance. To describe the MFT FSM, I started it from the top node, **MFT_DCS**, to lower level.

3.3 The State Diagram for the ALICE DCS

In the FSM of the ALICE DCS, the state name stands for the condition of the devices or a set of them, e.g. OFF, ON, ERROR, etc. To indicate the condition clearly, then, the states are shown with common coluours in the ALICE DCS when the state are shown in the state diagram or the control panel. The colours used here are grey, blue, green, yellow, orange, and red [22].

- DEAD(grey)

  The FSM is not running. Control is not available for this unit.

State example: DEAD

- OK_NOT_PHYSICS(blue)

  The unit is in a correct and stable state, but it is not ready for physics.

  State example: OFF, STANDBY

- OK_PHYSICS(green)

  The unit is ready for physics.

  State example: READY

- ALERT1(yellow)

  The unit is in a transition, normal state. It will change automatically to a stable state.

  State example: RAMPING_UP, MOVING_READY, MIXED

- ALERT2(orange)

  The unit or its tree is in a error or bad state. It can recover also automatically.

  State example: WARNING, HOT

- ALERT3(red)

  The unit or its tree is in a fatal error or bad state. It requires manual intervention for recovering.

  State example: ERROR, NO_CONTROL, TOO_HOT

3.4 MFT_DCS, The top node of the MFT

The MFT control system has been designed as a detector oriented hierarchy, i.e. based on the physical components of the detector. The top node of the hierarchy (**MFT_DCS**) is the main control unit (Figure 3.3). The commands are sent from this node to all the sub-trees below in parallel, and they report their states to the top node to reflect the overall state of the MFT. **MFT_DCS** has **Infrastructure** and **Detector** as children. **Infrastructure** consists of the external system, the devices controlled before or after detector operation, and the equipment whose operation can affect whole the detector. The sub-tree structure below **Detector** follows the hardware structure of the MFT, and it includes all the detector equipment, i.e. low voltage power supply modules, temperature sensors, and so on.

Figure 3.4 is the state diagram of **MFT_DCS**. It describes the possible states in **MFT_DCS** and the possible commands executing actions and/or triggering state transition. The states are divided into synchronous states used in normal operation (left) and asynchronous states appeared in error situations (center and right). The top node states follows the ALICE standard states [27]. The meaning of each synchronous state in **MFT_DCS** is following:

- OFF

  The CAEN power supply equipment is not ready. The cooling system condition is either ready or not ready.

- STANDBY

  The CAEN power supply equipment are ready for supplying power to the ALPIDEs and the readout boards. The cooling system are ready. The ALPIDEs and the readout boards power have not been supplied yet.

- DOWNLOADING

  All configuration data is being downloaded from database to the devices.

- STBY_CONFIGURED

  The devices are configured. The condition of equipment and systems are same as one in STANDBY.

- BEAM_TUNING

  A safe parking condition to face the LHC beam injection. The condition of equipment and systems are same as one in STANDBY.

- MOVING_READY

  The ALPIDEs and the readout boards power are ramping up.

- MOVING_STBY_CONF

  The ALPIDEs and the readout boards power are ramping down.

- READY

  The ALPIDEs and the readout boards power are supplied and the experiment can be started.

The meaning of each asynchronous state generally used in MFT is following:

- MIXED

  If the children are not all the same state (e.g. some low voltage channels are on. Others are off), then the parent goes to the MIXED state.

- ERROR

  If any children are ERROR or some error state, the parent goes to ERROR. If the parent is DU connecting a device and the device is in some error situations, the parent goes to ERROR.

- NO_CONTROL

  When connection between the device and the WinCC OA is lost, its DU goes to NO_CONTROL.

- INTERLOCK and INTERLOCK_WENT

  When hardware safety interlock system is activated, the device's DU goes to INTERLOCK. Once the interlock conditions are removed, the state automatically changes to INTERLOCK_WENT. In that state, RESET_INTERLOCK command can be executed and then it moves to OFF.

The correspondence between the parent's state transition and the children's is described in the synchronization table (Table 3.1). For example, the first state of **MFT_DCS** is OFF, then **Detector** is OFF and **Infrastructure** is NOT_READY. When **MFT_DCS** is OFF state, GO_READY command can be executed from **MFT_DCS** written in the diagram and it occurs a transition from NOT_READY to READY in **Infrastructure**, then **MFT_DCS** goes to STANDBY since **Infrastructure** is READY and **Detector** is OFF. State goes down from OFF to READY in the table in normal operation. At this time, the state having ▽ in its left side (e.g. DOWNLOADING and MOVING_READY) is appeared only during transition OFF to READY, and the state having ▲ (e.g. MOVING_STBY_CONF) is done during READY to OFF. The possible states in **MFT_DCS** are lined up in the most left line and the states in the children are in the right side lines of the thick line. The upper rows than the empty row (OFF to READY) are synchronous states and lower (MIXED and below) are asynchronous states. And the empty cell means that there can be any synchronous state. For example, when **Infrastructure** is NO_CONTROL, **MFT_DCS** goes to NO_CONTROL regardless of the state of **Detector**. I will describe FSM on each node by using state diagrams and synchronization tables.

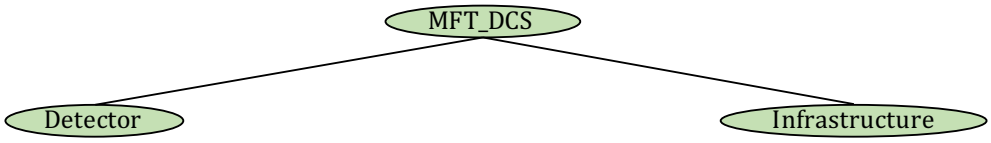

Figure 3.3: MFT top level FSM nodes

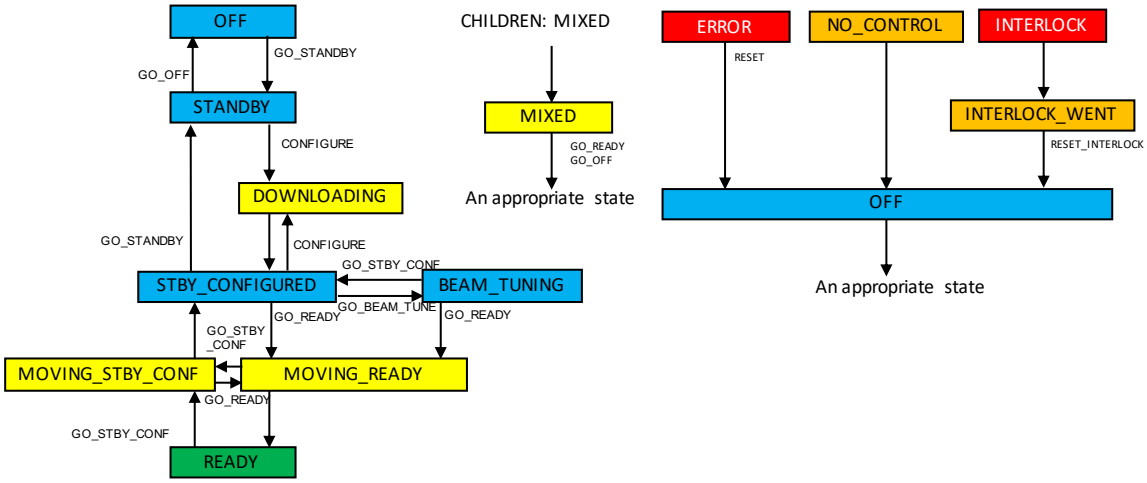

Figure 3.4: The State Diagram for **MFT_DCS**

	MFT_DCS	Detector	Infrastructure
	OFF	OFF	NOT_READY
	STANDBY	OFF	READY
▽	DOWNLOADING	DOWNLOADING	READY
	STBY_CONFIGURED	STBY_CONFIGURED	READY
	BEAM_TUNING	STBY_CONFIGURED	READY
▽	MOVING_READY	MOVING_READY	READY
▲	MOVING_STBY_CONF	MOVING_STBY_CONF	READY
	READY	READY	READY
	MIXED	MIXED	
	NO_CONTROL		NO_CONTROL
	INTERLOCK		INTERLOCK
	INTERLOCK_WENT		INTERLOCK_WENT
	ERROR		ERROR
	NO_CONTROL	NO_CONTROL	
	INTERLOCK	INTERLOCK	
	INTERLOCK_WENT	INTERLOCK_WENT	
	ERROR	ERROR	

Table 3.1: The synchronization table for **MFT_DCS**

3.5 Infrastructure and the sub-tree

In this section, FSMs on **Infrastructure** and below are described. Figure 3.5 shows the sub-tree. **Infrastructure** has **Cooling** and **LVPS** so that it reflects the cooling system and the LVPS conditions. Figure 3.6 is the state diagram of **Infrastructure** and Table 3.2 is the synchronization table. By sending GO_READY in NOT_READY, **LVPS** automatically executes SWITCH_ON triggering transition from NOT_READY to READY in **LVPS**. Only when **Cooling** is READY, **LVPS** can accept the GO_READY command from **Infrastructure** and send the SWITCH_ON command to children. **Infrastructure** goes to READY when both **Cooling** and **LVPS** are READY.

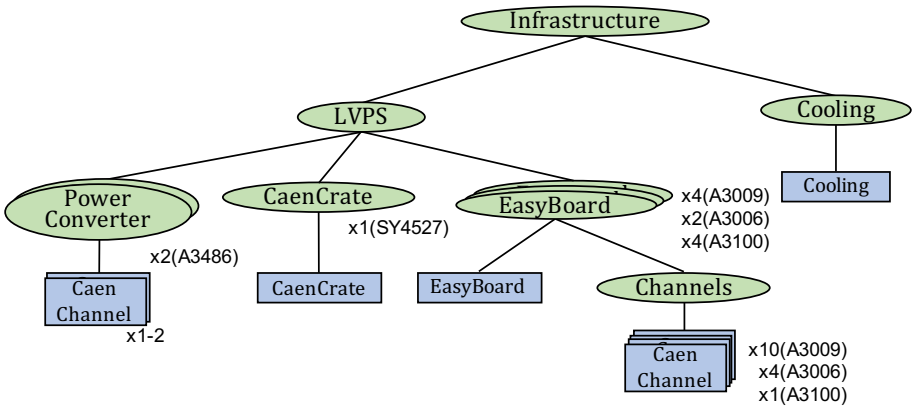

Figure 3.5: The Sub-tree structure below **Infrastructure**

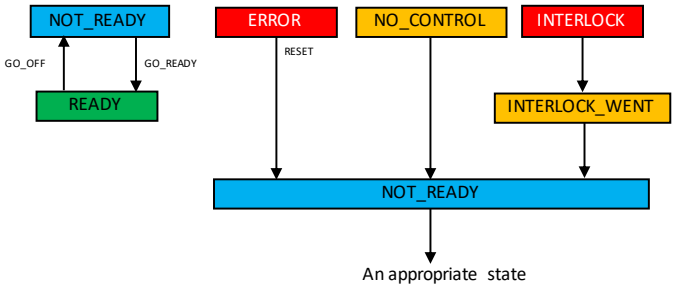

Figure 3.6: The State Diagram for **Infrastructure**

Infrastructure	LVPS	Cooling
NOT_READY	NOT_READY	NOT_READY
NOT_READY	NOT_READY	READY
NOT_READY	STANDBY	READY
NOT_READY	MOVING_READY	READY
READY	READY	READY
NO_CONTROL		NO_CONTROL
ERROR		ERROR
NO_CONTROL	NO_CONTROL	
INTERLOCK	INTERLOCK	
INTERLOCK_WENT	INTERLOCK_WENT	
ERROR	ERROR	

Table 3.2: The synchronization table for **Infrastructure**

3.5.1 Cooling

EN-CV Detector Cooling Section is in charge of the development of the cooling system including the design and implementation of the FSM. At this time, therefore, a simple FSM was designed and adopted temporary. The possible states are only READY, NOT_READY and ERROR, and this FSM allows only monitoring it since the cooling system of the MFT is shared to use with other sub-detectors and the MFT DCS will not control it. So far, NOT_READY supposes the condition which the cooling system have not worked yet or the detector is not sufficiently cooled. READY supposes that it is sufficiently cooled. ERROR supposes that the system is damaged or some kinds of error are detected.

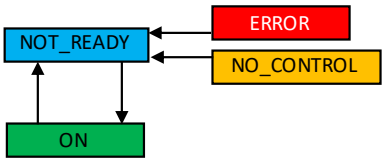

Figure 3.7: The State Diagram for **Cooling** CU and DU

Cooling (CU)	Cooling (DU)
NOT_READY	NOT_READY
READY	READY
NO_CONTROL	NO_CONTROL
ERROR	ERROR

Table 3.3: The synchronization table for **Cooling**

3.5.2 LVPS

The sub-tree of **LVPS** supervises the CAEN power supply devices: the main crate SY4527, EASY boards A3006, A3009, and A3100 supplying power to the ALPIDEs and the readout boards, and the power converter A3486 supplying 48 power to the EASY boards. The state diagram is Figure 3.8 and the synchronization table is Table 3.4. When GO_READY is sent from the parent to **LVPS** in NOT_READY, **LVPS** executes SWITCH_ON forward

PowerConverter. The command turns on the A3486's channels, and after that, **LVPS** automatically sends the CLEAR_ALARM to **CaenCrate** to clear alarms on the devices connected with SY4527 and moves to MOVING_READY. Then, EASY boards becomes ready for supplying power and **EasyBoards** goes to READY. Eventually, **LVPS** goes to READY.

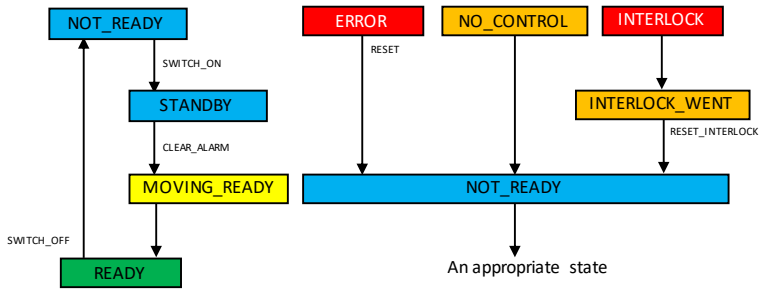

Figure 3.8: The State Diagram for **LVPS**

LVPS	PowerConverter	CaenCrate (CU)	EasyBoard (CU)
NOT_READY	STANDBY	READY	NOT_READY
NOT_READY	RAMPING_UP	READY	NOT_READY
NOT_READY	RAMPING_DOWN	READY	NOT_READY
STANDBY	READY	READY	NOT_READY
MOVIONG_READY	READY	ALARM_CLEARING	NOT_READY
READY	READY	READY	READY
NO_CONTROL			NO_CONTROL
ERROR			ERROR
NO_CONTROL		NO_CONTROL	
ERROR		ERROR	
NO_CONTROL	NO_CONTROL		
INTERLOCK	INTERLOCK		
INTERLOCK_WENT	INTERLOCK_WENT		
ERROR	ERROR		

Table 3.4: The synchronization table for **LVPS**

CaenCrate

CaenCrate (DU) controls the CAEN crate SY4527, and it monitors the connection status basically. If the connection is lost, it moves to NO_CONTROL. In addition, if alarms are generated in the devices connected with SY4527, they can be reset by CLEAR_ALARM command sent from this DU. The CU reflects the state of the DU directly. The state diagram of the CU and the DU is Figure 3.9 and Figure 3.10. The synchronization table is Table 3.5.

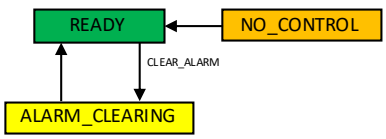

Figure 3.9: The State Diagram for **CaenCrate** CU

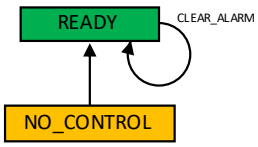

Figure 3.10: The State Diagram for **CaenCrate** DU

CaenCrate (CU)	CaenCrate (DU)
READY	READY
ALARM_CLEARING	READY
NO_CONTROL	NO_CONTROL

Table 3.5: The synchronization table for **CaenCrate**

PowerConverter

In **PowerConverter**, the power converter A3486s supplying 48V power to EASY boards are controlled and **PowerConverter** has **CaenChannel** DUs which control the low voltage channels of A3486. The state diagrams are Figure 3.11 and Figure 3.12. **CaenChanel** has OFF and ON states, and the low voltage channels supply power to EASY boards. **CaenChannel** moves to TRIPPED or OVERCURRENT when the voltage or current of the corresponding channel exceeds the threshold value. In addition to these, there are NO_CONTROL for connection lost and ERROR for other error situations. OVERCURRENT, TRIPPED, and ERROR are united into ERROR in **PowerConverter**.

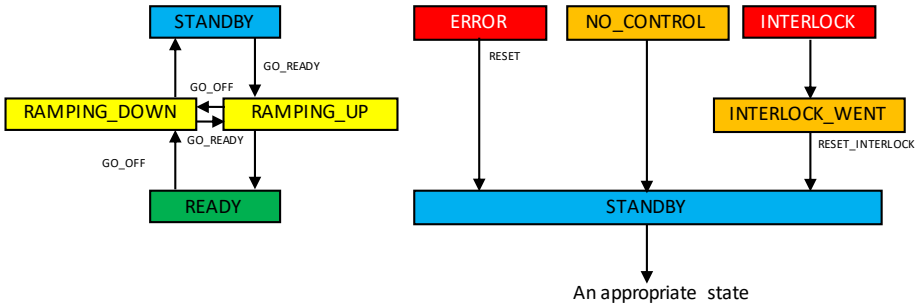

Figure 3.11: The State Diagram for **PowerConverter**

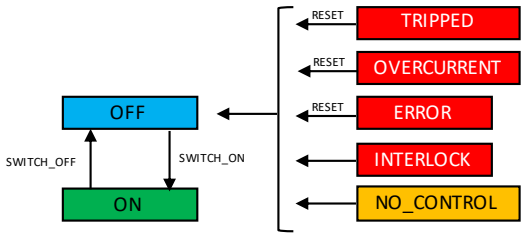

Figure 3.12: The State Diagram for **CaenChannel**

	PowerConverter	CaenChannel
	STANDBY	OFF
▽	RAMPING_UP	OFF
▲	RAMPING_DOWN	ON
	READY	ON
	NO_CONTROL	NO_CONTROL
	INTERLOCK	INTERLOCK
	INTERLOCK_WENT	INTERLOCK_WENT
	ERROR	ERROR
	ERROR	TRIPPED
	ERROR	OVERCURRENT

Table 3.6: The synchronization table for **PowerConverter**

EasyBoard

The state diagram of **EasyBoard** (CU) is Figure 3.13 and the synchronization table is Table 3.7. It has **EasyBoard** (DU) which monitors the condition of the EASY board and **Channels** for monitoring the conditions of its low voltage channels.

The state diagram of **EasyBoard** (DU) is Figure 3.14. It monitors the temperature sensor in the board and the condition of the 48V power from A3486. When 48V power is supplied and CLEAR_ALARM command are sent to SY4527, the state moves to READY from NOT_READY. If the value of the temperature sensor exceeds the threshold values (T_{hot}orT_{too_hot}), it moves to either HOT or TOO_HOT. In the most left column and the 7th row in the synchronization table, there is a blank cell. This means that even if **EasyBoard** (DU) goes to HOT, the CU does not reflect the state.

Channels has **CaenChannel**s as children to only monitor their states. The state diagram is Figure 3.15 and the synchronization table is Table 3.8. Here, **CaenChannel_FWMAJ** means the majority device unit of **CaenChannel**s. When a lot of DUs which have same parent (**CaenChannel**s in this case) moves to ERROR (or kinds of error states), the majority DU moves to MAJORITY_ERROR (if the number of the DUs in ERROR state exceeds the threshold number defined in advance) or MAJORITY_WARNING (if the number is between 1 and the threshold).

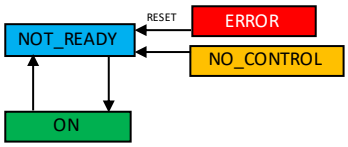

Figure 3.13: The State Diagram for **EasyBoard** (CU)

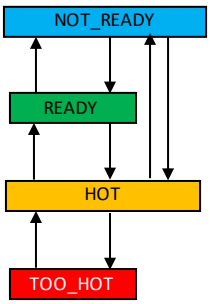

Figure 3.14: The State Diagram for **EasyBoard** (DU)

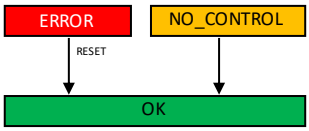

Figure 3.15: The State Diagram for **Channels**

EasyBoard (CU)	EasyBoard (CU)	Channels
NOT_READY	MF	OK
READY	READY	OK
NO_CONTROL		NO_CONTROL
ERROR		ERROR
	HOT	
ERROR	TOO_HOT	

Table 3.7: The synchronization table for **EasyBoard**

Channels	CaenChannel_FWMAJ	CaenChannel
OK	MAJORITY_OK	OFF
OK	MAJORITY_OK	ON
NO_CONTROL		NO_CONTROL
		INTERLOCK_WENT
	MAJORITY_WARNING	INTERLOCK(1≦n≦TH)
	MAJORITY_WARNING	ERROR(1≦n≦TH)
	MAJORITY_WARNING	TRIPPED(1≦n≦TH)
	MAJORITY_WARNING	OVERCURRENT(1≦n≦TH)
ERROR	MAJORITY_ERROR	INTERLOCK(TH<n)
ERROR	MAJORITY_ERROR	ERROR(TH<n)
ERROR	MAJORITY_ERROR	TRIPPED(TH<n)
ERROR	MAJORITY_ERROR	OVERCURRENT(TH<n)

Table 3.8: The synchronization table for **Channels**, TH is the threshold number and n is the number of the DUs in ERROR state

3.6 Detector and the sub-tree

The sub-tree below **Detector** is described here. The structure is Figure 3.16. It reflects the real MFT hardware structure: Half MFTs, Half Disks, Half Planes, Ladders, and the ALPIDEs, and low voltage distribution so that it enables us to operate each part of the MFT separately. GBT-SCA power, LV power of the ALPIDEs and Back-bias power are supplied by voltage transformation from the CAEN channel of the EASY boards using either DCDC converters or inverter amplifiers, and there is each of the transformers par one Half Plane. Thus, there are CUs controlling the power supply devices below every **Half_Plane**. In addition to this, **Half_Plane** has a DU of a temperature sensor since there is a temperature sensor on each Half Plane. On the other hand, the power is split into 4 Zones and there is switch on each line to turn it off Zone by Zone when either over-current or latch up is detected by current sensor or latch up detector. Therefore, a CU controlling the switches and DUs of the current sensor and the latch up detector exist below each **Zone**. **Zone** also has a CU of RU (Readout Unit) since there is one RU board for each Zone. The upper nodes than **Half_Plane** reflects the state of **Half_Plane** directly unless there is partitioning.

Figure 3.17 is the state diagram for **Detector**, **Half_MFT**, and **Half_Disk** in common. The state changes from OFF to STBY_CONFIGURED by the CONFIGURE command, and GO_READY makes it moves to READY. The synchronization table for these nodes are Table 3.9, 3.10, and 3.11. These tables describes the relation between the parent's state and the children's. In addition to these tables, the state moves to MIXED if some of children are READY and the others are OFF and/or STBY_CONFIGURED. When all of the children moves to the same state, parent goes to an appropriate state automatically. This MIXED state handling is common to all other nodes. If children moves to ERROR or NO_CONTROL, the parent reflects the state and after the children moves to normal state, the parent moves to an appropriate state via OFF state.

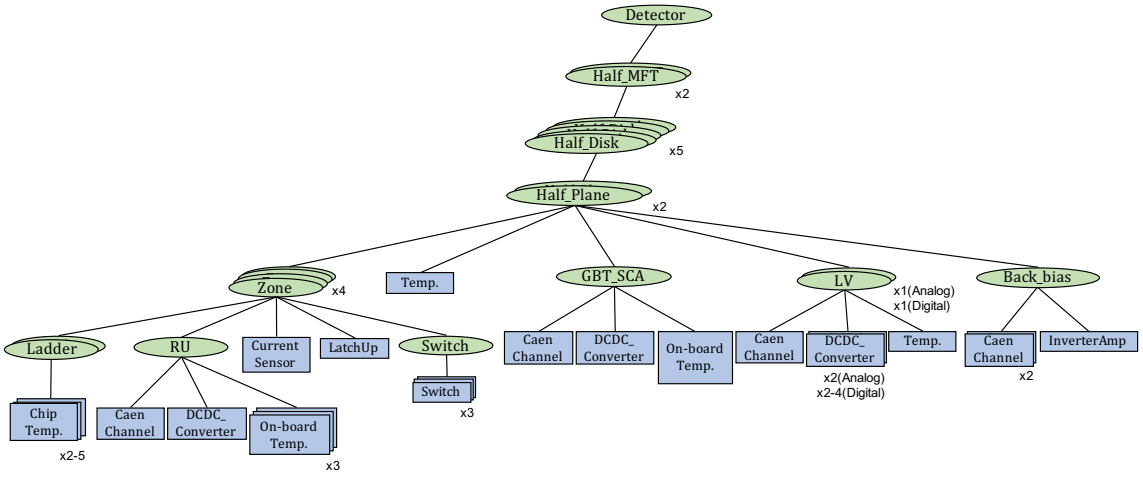

Figure 3.16: The Sub-tree structure below **Detector**

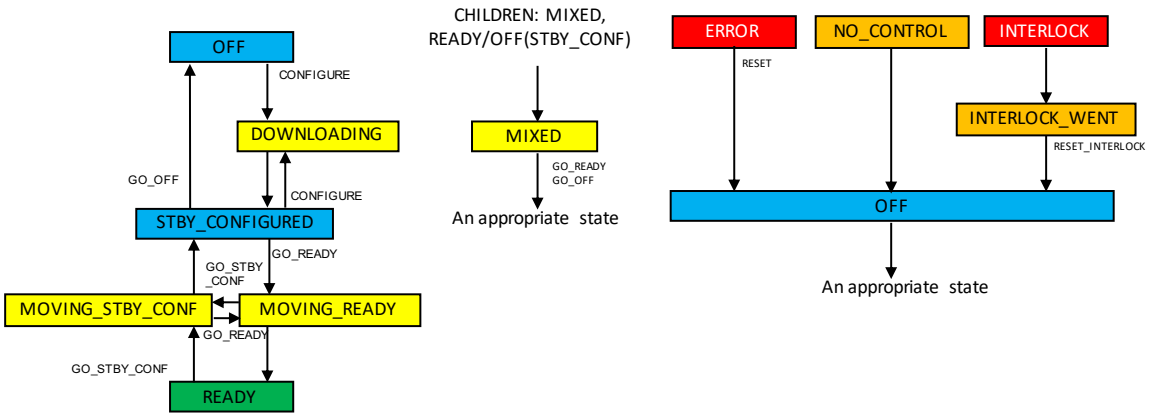

Figure 3.17: The State Diagram for **Detector**, **Half_MFT**, and **Half_Disk**

	Detector	Half_MFT
	OFF	OFF
▽	DOWNLOADING	DOWNLOADING
	STBY_CONFIGURED	STBY_CONFIGURED
▽	MOVING_READY	MOVING_READY
▲	MOVING_STBY_CONF	MOVING_STBY_CONF
	READY	READY
	MIXED	MIXED
	NO_CONTROL	NO_CONTROL
	INTERLOCK	INTERLOCK
	INTERLOCK_WENT	INTERLOCK_WENT
	ERROR	ERROR

Table 3.9: The synchronization table for **Detector**

	Half_MFT	Half_Disk
	OFF	OFF
▽	DOWNLOADING	DOWNLOADING
	STBY_CONFIGURED	STBY_CONFIGURED
▽	MOVING_READY	MOVING_READY
▲	MOVING_STBY_CONF	MOVING_STBY_CONF
	READY	READY
	MIXED	MIXED
	NO_CONTROL	NO_CONTROL
	INTERLOCK	INTERLOCK
	INTERLOCK_WENT	INTERLOCK_WENT
	ERROR	ERROR

Table 3.10: The synchronization table for **Half_MFT**

	Half_Disk	Half_Plane
	OFF	OFF
▽	DOWNLOADING	DOWNLOADING
	STBY_CONFIGURED	STBY_CONFIGURED
▽	MOVING_READY	MOVING_READY_READOUT
▽	MOVING_READY	MOVING_READY_LV
▲	MOVING_STBY_CONF	MOVING_OFF_READOUT
▲	MOVING_STBY_CONF	MOVING_OFF_LV
	READY	READY
	MIXED	MIXED
	NO_CONTROL	NO_CONTROL
	INTERLOCK	INTERLOCK
	INTERLOCK_WENT	INTERLOCK_WENT
	ERROR	ERROR

Table 3.11: The synchronization table for **Half_Disk**

3.6.1 Half_Plane and below

The sub-tree structure below **Half_Plane** is Figure 3.18. **Half_Plane** has a DU monitoring a temperature sensor and CUs of **GBT-SCA**, **LV**, **Back_bias**, and **Zone**

Figure 3.19 is the state diagram for **Half_Plane**. The state changes from OFF to STBY_CONFIGURED by CONFIGURE. GO_READY makes the power supply devices start supplying with the GBT-SCA's power and then the state moves to MOVING_READY_READOUT. After the GBT-SCA become ready, LV and Back-bias power supply are started and it moves to MOVING_READY_LV. When both **LV** and **Back_bias** are READY, it moves to READY.

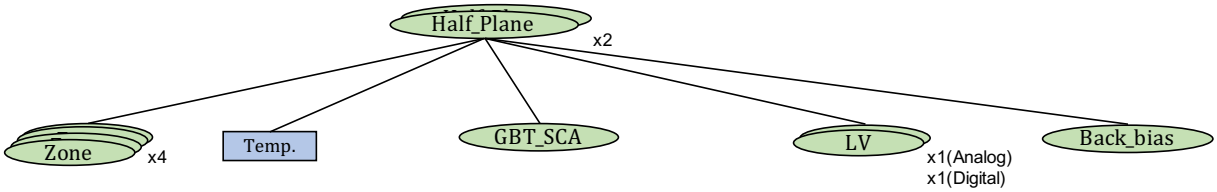

Figure 3.18: The Sub-tree structure of **Half Plane**

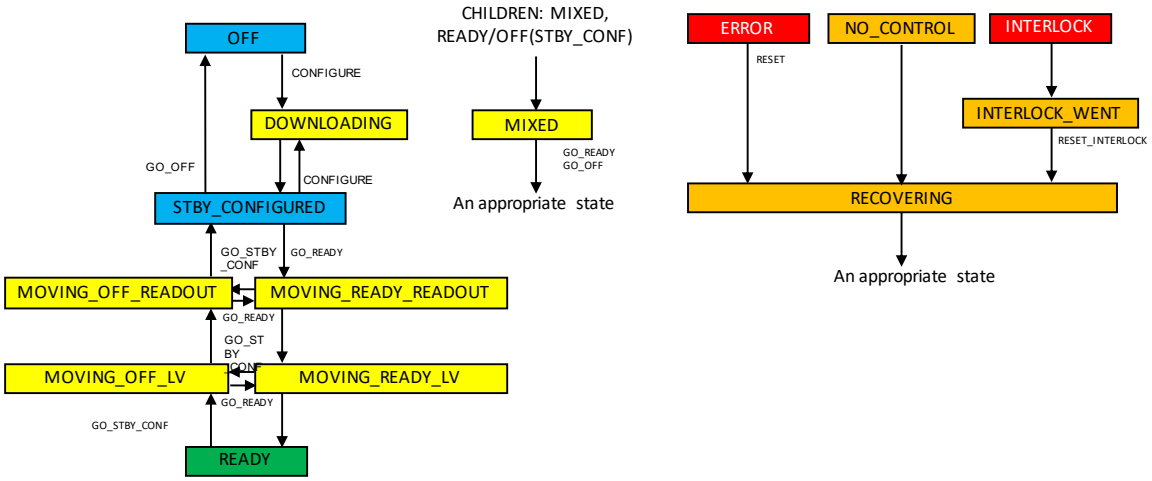

Figure 3.19: The State Diagram for **Half_Plane**

	Half_Plane	Zone	GBT_SCA	LV	Back_bias	Temp
	OFF	OFF	OFF	OFF	OFF	OK
▽	DOWNLOADING	DOWNLOADING	DOWNLOADING	DOWNLOADING	DOWNLOADING	OK
	STBY_CONFIGURED	STBY_CONFIGURED	STBY_CONFIGURED	STBY_CONFIGURED	STBY_CONFIGURED	OK
▽	MOVING_READY_READOUT	MOVING_READY	MOVING_READY	STBY_CONFIGURED	STBY_CONFIGURED	OK
▽	MOVING_READY_LV	RU_READY	READY	MOVING_READY	MOVING_READY	OK
▲	MOVING_OFF_READOUT	MOVING_STBY_CONF	MOVING_STBY_CONF	STBY_CONFIGURED	STBY_CONFIGURED	OK
▲	MOVING_OFF_LV	RU_READY	READY	MOVING_STBY_CONF	MOVING_STBY_CONF	OK
	READY	READY	READY	READY	READY	OK
						HOT
	ERROR					TOO_HOT
	NO_CONTROL				NO_CONTROL	
	INTERLOCK				INTERLOCK	
	INTERLOCK_WENT				INTERLOCK_WENT	
	ERROR				ERROR	
	MIXED			MIXED		
	NO_CONTROL			NO_CONTROL		
	INTERLOCK			INTERLOCK		
	INTERLOCK_WENT			INTERLOCK_WENT		
	ERROR			ERROR		
	NO_CONTROL		NO_CONTROL			
	INTERLOCK		INTERLOCK			
	INTERLOCK_WENT		INTERLOCK_WENT			
	ERROR		ERROR			
	MIXED	MIXED				
	NO_CONTROL	NO_CONTROL				
	INTERLOCK	INTERLOCK				
	INTERLOCK_WENT	INTERLOCK_WENT				
	ERROR	ERROR				

Table 3.12: The synchronization table for **Half_Plane**

GBT_SCA

GBT_SCA is a CU for supervising the power supply devices of GBT-SCA and the on-board temperature sensor. It has DUs for the CAEN low voltage channel, the DCDC converter, and the on-board temperature sensor. The state diagrams are Figure 3.12 (**Caen-Channel**), 3.21 (**DCDC_Converter**), 3.22 (**On-boardTemp**), and 3.20 (**GBT_SCA**). The synchronization table is Table 3.13.

DCDC_Converter has OFF, ON, and ERROR for error situations (e.g. the device is damaged). **On-boardTemp** reads the value which is measured by the GBT-SCA on-board temperature sensor and the state changes in READY, HOT and TOO_HOT depending on the temperature value. In addition to these states, it has NO_CONTROL for connection lost and ERROR for other error situations.The state diagram in Figure 3.20 is also used in CUs of **LV**, **Back_bias**, and **Zone** in common.

Then, **Configurator** is a DU which loads configuration recipe of the devices (e.g. setting values of the voltage and the current, threshold values of them) from the configuration database and inputs them into DPEs of the devices' DUs [32]. The setting values input into DPEs are applied to the devices via a communication protocol (e.g. OPC, DIM). The CU which has DUs that have setting values to be configured has a **Configurator** and it manages configuration of the DUs.

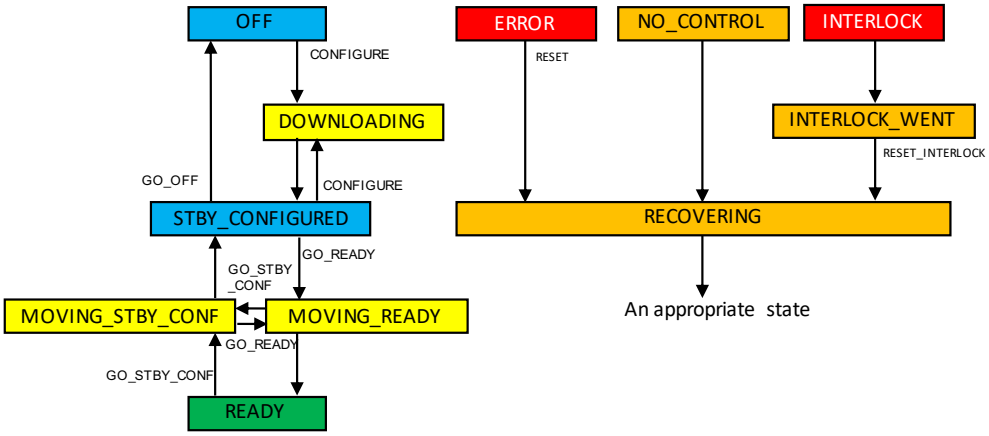

Figure 3.20: The State Diagram for **GBT_SCA**, **LV**, **Back_bias**, and **Zone**

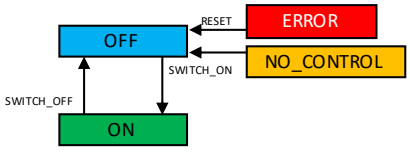

Figure 3.21: The State Diagram for DCDC converter

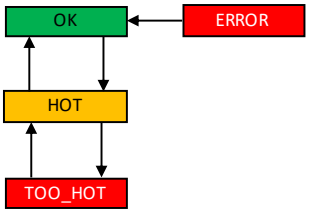

Figure 3.22: The State Diagram for On-board Temperature Sensor

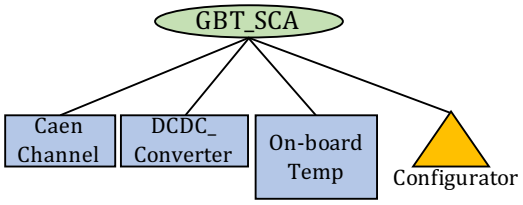

Figure 3.23: The Sub-tree structure of **GBT_SCA**

	GBT_SCA	CaenChannel	DCDC_Converter	On-boardTemp	Configurator
	OFF	OFF	OFF	OK	NOT_READY
▽	DOWNLOADING	OFF	OFF	OK	DOWNLOADING
	STBY_CONFIGURED	OFF	OFF	OK	READY
▽	MOVING_READY	OFF	OFF	OK	READY
▲	MOVING_STBY_CONF	ON	ON	OK	READY
	READY	ON	ON	OK	READY
				WARNING	
	ERROR			TOO_HOT	
	NO_CONTROL			NO_CONTROL	
	ERROR			ERROR	
	NO_CONTROL		NO_CONTROL		
	ERROR		ERROR		
	NO_CONTROL	NO_CONTROL			
	INTERLOCK	INTERLOCK			
	INTERLOCK_WENT	INTERLOCK_WENT			
	ERROR	ERROR			
	ERROR	TRIPPED			
	ERROR	OVERCURRENT			

Table 3.13: The synchronization table for **GBT_SCA**

LV

The power supply devices for Analog and Digital power source to the ALPIDEs are controlled in **LV**. It has DUs for CAEN low voltage channels (**CaenChannel**), DCDC converters (**DCDC_Converter**), and the temperature sensor for the DCDC converter's cooling block (**Temp**). The state diagrams are Figure 3.12 (**CaenChannel**), 3.21 (**DCDC_Converter**), 3.25 (**Temp**) The state diagram for **LV** is Figure 3.20 and the synchronization table is Table 3.14.

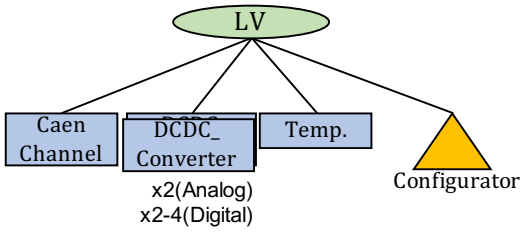

Figure 3.24: The Sub-tree structure of **LV**

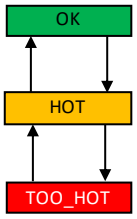

Figure 3.25: The State Diagram for **Temp**

	LV	CaenChannel	DCDC_Converter	Temp	Configurator
	OFF	OFF	OFF	OK	NOT_READY
▽	DOWNLOADING	OFF	OFF	OK	DOWNLOADING
	STBY_CONFIGURED	OFF	OFF	OK	READY
▽	MOVING_READY	OFF	OFF	OK	READY
▲	MOVING_STBY_CONF	ON	ON	OK	READY
	READY	ON	ON	OK	READY
				HOT	
	ERROR			TOO_HOT	
	NO_CONTROL		NO_CONTROL		
	ERROR		ERROR		
	NO_CONTROL	NO_CONTROL			
	INTERLOCK	INTERLOCK			
	INTERLOCK_WENT	INTERLOCK_WENT			
	ERROR	ERROR			
	ERROR	TRIPPED			
	ERROR	OVERCURRENT			

Table 3.14: The synchronization table for **LV**

Back_bias

The power supply devices for the Back-bias voltage of the ALPIDEs are controlled in **Back_bias**. It has DUs for the CAEN low voltage channels (**CaenChannel**) and the inverter amplifier (**InverterAmp**) used for voltage transformation. The state diagrams of **Back_bias** and **InverterAmp** are Figure 3.20 and 3.26. The synchronization table is Table 3.15.

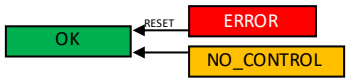

Figure 3.26: The State Diagram for **Inverter_amp**

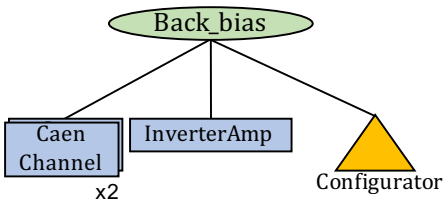

Figure 3.27: The Sub-tree structure of **Back_bias**

	Back_bias	CaenChannel	InverterAmp	Configurator
	OFF	OFF	OK	NOT_READY
▽	DOWNLOADING	OFF	OK	DOWNLOADING
	STBY_CONFIGURED	OFF	OK	READY
▽	MOVING_READY	OFF	OK	READY
▲	MOVING_STBY_CONF	ON	OK	READY
	READY	ON	OK	READY
	ERROR		ERROR	
	NO_CONTROL	NO_CONTROL		
	INTERLOCK	INTERLOCK		
	INTERLOCK_WENT	INTERLOCK_WENT		
	ERROR	ERROR		
	ERROR	TRIPPED		
	ERROR	OVERCURRENT		

Table 3.15: The synchronization table for **Back_bias**

Temp

There is a temperature sensor on each Half Plane and it is monitored in **Temp**. The State diagram is Figure 3.25. If the state is TOO_HOT, the state of **Half_Plane** becomes ERROR and the power supply devices below the Half Plane are automatically turned off.

Zone

The sub-tree structure below **Zone** is Figure 3.28. The state diagram of **Zone** is Figure 3.20 and the synchronization table is Table 3.16. It has CUs of **Ladder**, **RU**, **Switch** and DUs of **CurrentSensor**, **LatchUp**.

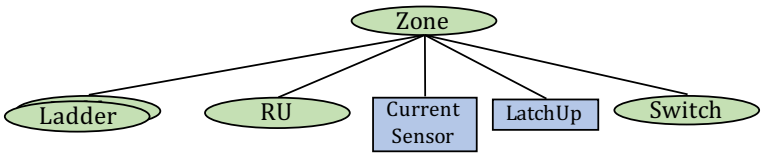

Figure 3.28: The Sub-tree structure of **Zone**

Zone	Ladder_FWMAJ	Ladder	RU	CurrentSensor	LatchUp	Switch (CU)
OFF	MAJORITY_OK	READY		OK	OK	OFF
OFF	MAJORITY_OK	READY	OFF	OK	OK	
▽ DOWNLOADING	MAJORITY_OK	READY	DOWNLOADING	OK	OK	ON
STBY_CONFIGURED	MAJORITY_OK	READY	STBY_CONFIGURED	OK	OK	ON
▽ MOVING_READY	MAJORITY_OK	READY	MOVING_READY	OK	OK	ON
▲ MOVING_STBY_CONF	MAJORITY_OK	READY	MOVING_STBY_CONF	OK	OK	ON
READY	MAJORITY_OK	READY	READY	OK	OK	ON
NO_CONTROL						NO_CONTROL
ERROR						ERROR
ERROR					ERROR	
ERROR				ERROR		
NO_CONTROL			NO_CONTROL			
INTERLOCK			INTERLOCK			
INTERLOCK_WENT			INTERLOCK_WENT			
ERROR			ERROR			
		WARNING(1≦n≦TH)				
ERROR	MAJORITY_WARNING	WARNING(TH<n)				
	MAJORITY_WARNING	ERROR(1≦n≦TH)				
ERROR	MAJORITY_ERROR	ERROR(TH<n)				

Table 3.16: The synchronization table for **Zone**

- **Ladder**

  There are **ChipTemp**s for the ALPIDEs on-board temperature sensors below **Ladder**. If the value of the temperature sensor exceeds the threshold values (T_{hot}orT_{too_hot}), it moves to either HOT or TOO_HOT. The state of **Ladder** changes depending on the state transition of **ChipTemp**. When the state of **Ladder** is ERROR, it moves back to READY if **ChipTemp** moves to OK in order to avoid a states oscillation around the threshold value of temperature. The state diagram of **ChipTemp** and **Ladder** are Figure 3.29 and Figure 3.30, and the synchronization table is Table 3.17

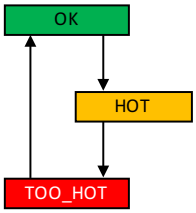

Figure 3.29: The State Diagram for **ChipTemp**

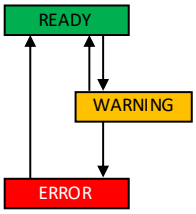

Figure 3.30: The State Diagram for **Ladder**

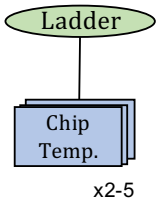

Figure 3.31: The Sub-tree structure below **Ladder**

Ladder	ChipTemp
READY	OK
WARNING	HOT
ERROR	TOO_HOT

Table 3.17: The synchronization table for

- **RU**

  The state diagram of **RU** is Figure 3.20, and it manages the power supply devices to the RU board and the RU on-board temperature sensor. **RU** has DUs for the CAEN

low voltage channel (**CaenChannel**), the DCDC converter (**DCDC_Converter**), and the on-board temperature sensor (**On-boardTemp**). The state diagrams are Figure 3.12 (**CaenChannel**), 3.21 (**DCDC_Converter**), 3.22 (**On-boardTemp**), and 3.20 (**RU**). The synchronization table is Table 3.18.

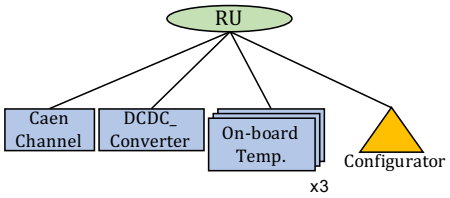

Figure 3.32: The Sub-tree structure below **RU**

	RU	CaenChannel	DCDC_Converter	On-boardTemp	Configurator
	OFF	OFF	OFF	OK	NOT_READY
▽	DOWNLOADING	OFF	OFF	OK	DOWNLOADING
	STBY_CONFIGURED	OFF	OFF	OK	READY
▽	MOVING_READY	OFF	OFF	OK	READY
▲	MOVING_STBY_CONF	ON	ON	OK	READY
	READY	ON	ON	OK	READY
				WARNING	
	ERROR			TOO_HOT	
	NO_CONTROL			NO_CONTROL	
	ERROR			ERROR	
	NO_CONTROL		NO_CONTROL		
	ERROR		ERROR		
	NO_CONTROL	NO_CONTROL			
	INTERLOCK	INTERLOCK			
	INTERLOCK_WENT	INTERLOCK_WENT			
	ERROR	ERROR			
	ERROR	TRIPPED			
	ERROR	OVERCURRENT			

Table 3.18: The synchronization table for **RU**

- **Switch**

  There are switches on power supply lines of LV and Back-bias split into 4 Zones. **Switch** DUs control them and **Switch** (CU) unites the DUs. The state diagram for **Switch** of both CU and DU is Figure 3.33.

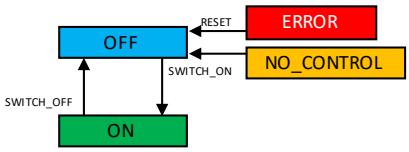

Figure 3.33: The State Diagram for Switches(DU, CU)

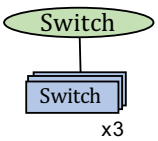

Figure 3.34: The Sub-tree structure below **Switch**

Switch (CU)	Switch (DU)
OFF	OFF
ON	ON
NO_CONTROL	NO_CONTROL
ERROR	ERROR

Table 3.19: The synchronization table for **Switch**

- **CurrentSensor**

  CurrentSensor is a DU for monitoring the current values of LV and Back-bias. If the value of the current sensor exceeds the threshold, hardware interlock turns OFF the switches of LV and Back-bias voltage lines automatically and the state moves to ERROR. The state diagram is Figure 3.35.

- **LatchUp**

  LatchUp is a DU for monitoring latch-up on the power supply circuits of LV and Back-bias. If latch-up occurs, hardware safety interlock turns off the switches of LV and Back-bias voltage lines automatically and the state moves to ERROR. The state diagram is Figure 3.36.

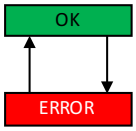

Figure 3.35: The State Diagram for Current Sensor

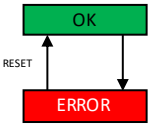

Figure 3.36: The State Diagram for Latch Up detector

Chapter 4

The FSM for the Actual Machine Tests

The actual machine tests have been carried out to test the FSM. In this chapter, the FSM for the tests is described.

4.1 The Architecture of the Actual Machine Tests

The hardware components used in the tests and the FSM are described in this section.

4.1.1 Hardware Components

The CAEN power supply equipment and a temperature sensor are used in the tests. The CAEN power supple system has been constructed by K. Yamakawa at CERN. SY4527 (CAEN crate), A1676A (crate controller), A3486 (power converter), EASY3000 (EASY crate), and A3009 (EASY board) are used in the tests, and the structure is same as the structure of power supply to the ALPIDEs in the real MFT design. The temperature sensor is a Pt100, a platinum resistance thermometer sensor, connected by the ELMB. The ELMB is an ASIC developed in ATLAS experiment and the purpose is same as the GBT-SCA. The GBT-SCA will be used in the MFT, however it has not been enough developed yet. Therefore, the ELMB is used in the actual machine tests. The resistance value of the Pt100 is measured with the ELMB, and the value is translated into temperature. The way to construct the CAEN system follows [19] and to connect with ELMB is described in [10].

Figure 4.1: The test bench of the CAEN power supply system for the MFT constructed by K. Yamakawa [39]

4.1.2 The Control Hierarchy of the Actual Machine Tests

The control hierarchy for the tests is Figure 4.2. This is a simplified hierarchy of the real MFT's: **Cooling** and the detailed structure below **Detector** are taken off. The structure below **LVPS** is same as the real MFT's. There are **CaenChannel** and **Temp** below **Detector**. It simulates the power source to the ALPIDEs and the ALPIDE's on-board temperature sensor. This sub-tree is designed to automatically turn the CAEN low voltage channels off when an abnormal rise in temperature is detected.

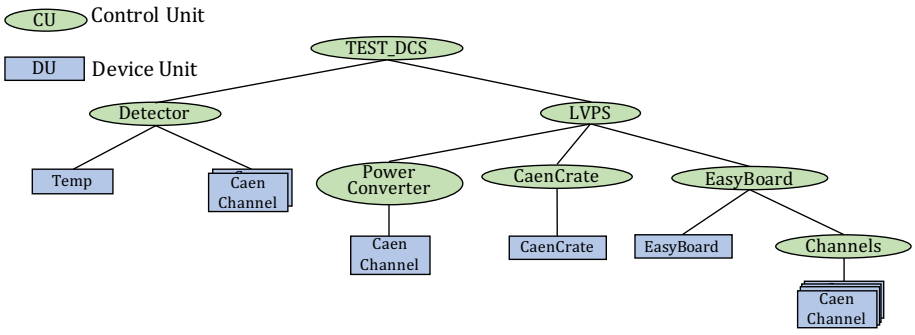

Figure 4.2: The Control Hierarchy of the FSM for the Actual Machine Test

4.2 Design of FSMs

Design of a state transition on each node is described in this section.

4.2.1 TEST_DCS for the Actual Machine Tests

The top node of the FSM is **TEST_DCS**. The state diagram is Figure 4.3 and the synchronization table is Figure 4.1. The state diagram of **TEST_DCS** is same as one of **MFT_DCS** except BEAM_TUNING since the devices condition is same as in STANDBY, and DOWNLOADING and STANDBY_CONFIGURED since the device configurations from the FSM have not been implemented yet.

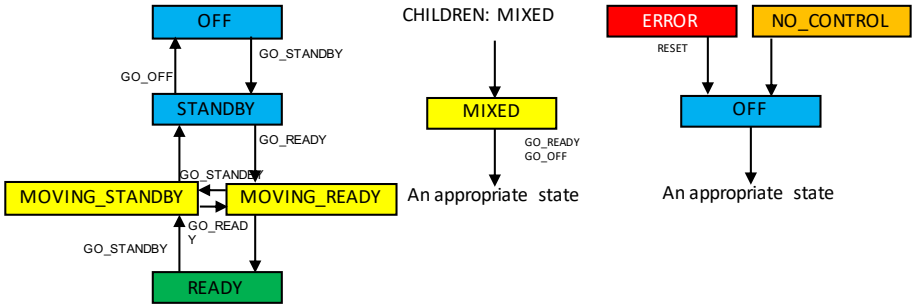

Figure 4.3: The State Diagram of **TEST_DCS**

	TEST_DCS	Detector	LVPS
	OFF	OFF	NOT_READY
	OFF	OFF	STANDBY
	OFF	OFF	MOVING_READY
	STANDBY	OFF	READY
▽	MOVING_READY	MOVING_READY	READY
▲	MOVING_STBY_CONF	MOVING_STBY_CONF	READY
	READY	READY	READY
	NO_CONTROL		NO_CONTROL
	ERROR		ERROR
		RECOVERING	
	NO_CONTROL	NO_CONTROL	
	ERROR	ERROR	

Table 4.1: The synchronization table for **TEST_DCS**

4.2.2 LVPS for the Actual Machine Tests

The structure and FSMs below **LVPS** are same as the MFT's. The detail is described in Section 3.5.2.

4.2.3 Detector for the Actual Machine Tests

The state diagram of **Detector** is Figure 3.17. **Detector** has DUs: **CaenChannel** and **Temp**. Each state diagram is same as the MFT's. The diagrams are Figure 3.12 (**CaenChannel**), 3.25 (**Temp**). OFF/READY in **Detector** corresponds to OFF/ON in **CaenChannel**. When the temperature exceeds the threshold values, the state of **Temp** moves to either HOT or TOO_HOT. If it moves to TOO_HOT, **Detector** goes to ERROR and then, **Detector** sends RESET command to **CaenChannel** to turn off the CAEN channels.

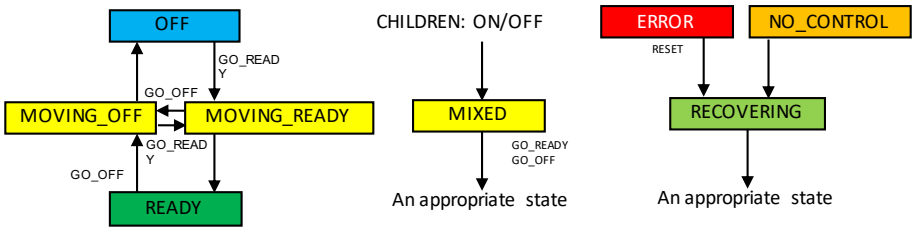

Figure 4.4: The State Diagram of **Detector**

4.3 Implementation of the FSM for the Actual Machine Tests

The FSM designed for the actual machine tests are implemented with JCOP. The tree structure implemented in JCOP is in Figure 4.5. The left panel shows only the CUs and the right panel shows all the CUs and the DUs only below **LVPS** and **Detector** due to restriction of the window size. The JCOP allows us to implement the FSM with simple GUI panels. Figure 4.6 is an example of the FSM configuration panel of CUs. In this panel, possible states of **LVPS** are written in "State list", and possible actions and state transition conditions of the corresponding state are in "Action list" and "When list". In this example, the NOT_READY state is selected, therefore possible actions and transition conditions when the state is NOT_READY are written in the corresponding lists. Then, the panel makes the SML code of **LVPS**. Figure 4.7 is an example of the FSM configuration panel of DUs. In this panel, the possible states of **CaenChannel** are written in "State list", and possible actions of the corresponding state are written in "Action list". The actions can be implemented in "Configure Device Actions" which is in the drop-down list of the "Configure Device" box by writing the SML code in which the actions are described. The state transition conditions also can be in "Configuration Device States" by writing one in which the transition conditions are described. The detailed information about implementation of the FSM can be reached in [23, 24, 26, 29, 30].

And then, the CAEN devices and the temperature sensor are connected and the functions of the FSM is tested.

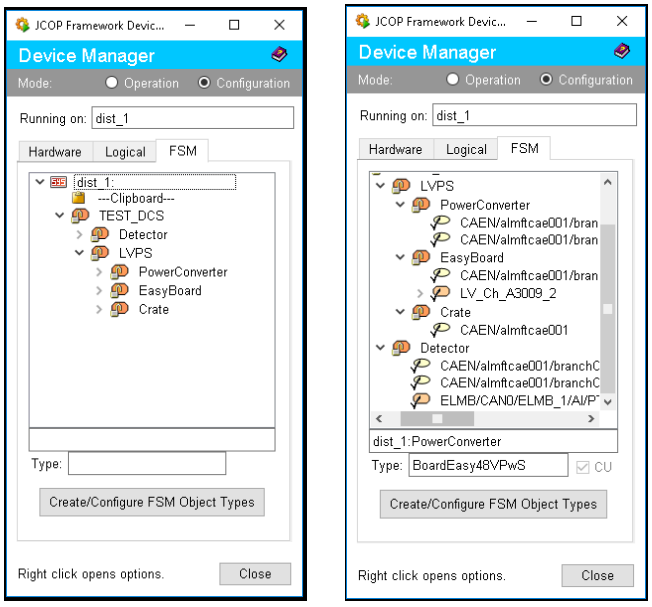

Figure 4.5: The FSM Tree for the Actual Machine test implemented in JCOP Fw. left: All CUs in the tree, Right: All DUs below **LVPS** and **Detector**

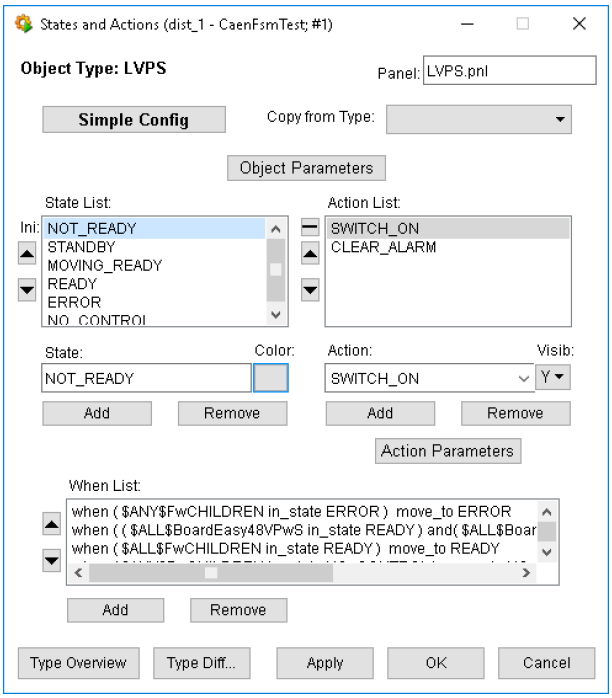

Figure 4.6: The FSM Configuration panel of **LVPS** in the JCOP

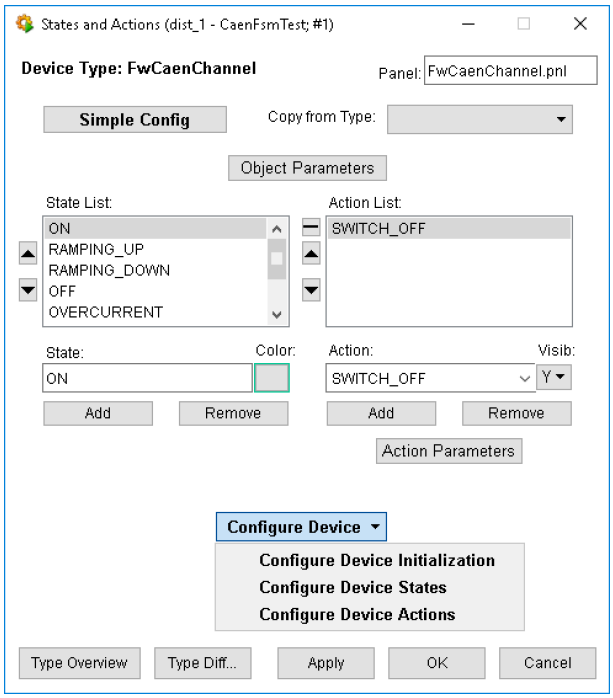

Figure 4.7: The FSM Configuration panel of **CaenChannel** in the JCOP

The SML codes of **Detector** and **CaenChannel** are shown as examples of a CU and a DU. The SML code of **Detector** is shown in the following. Possible states, codes and state transition conditions on each node in **Detector** are written in the code. Please refer the example in Section 2.3.5 to read this code.

The SML code of **Detector**

```
state : OFF
    when ( $ANY$FwCHILDREN in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,O_CONTROL} ) move_to ERROR
    when ( ( $ALL$FwCaenChannel in_state ON ) and ( $ALL$FwElmbAi in_state OK ) ) move_to READY
    when ( ( $ANY$FwCHILDREN in_state OFF ) and ( $ANY$FwCHILDREN in_state ON ) ) move_to MIXED
        action: GO_READY
            do SWITCH_ON $ALL$FwCaenChannel
            move_to MOVING_READY

state : MOVING_READY
    when ( $ANY$FwCHILDREN in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to ERROR
    when ( $ALL$FwCaenChannel in_state OFF ) move_to OFF
    when ( ( $ALL$FwCaenChannel in_state ON ) and ( $ALL$FwElmbAi in_state OK ) ) move_to READY
    when ( ( $ANY$FwCHILDREN in_state OFF ) and ( $ANY$FwCHILDREN in_state ON ) ) move_to MIXED
        action: GO_OFF
            do SWITCH_OFF $ALL$FwCaenChannel
            move_to MOVING_OFF
state : MOVING_OFF
    when ( $ANY$FwCHILDREN in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to ERROR
    when ( $ALL$FwCaenChannel in_state OFF ) move_to OFF
    when ( ( $ALL$FwCaenChannel in_state ON ) and ( $ALL$FwElmbAi in_state OK ) ) move_to READY
    when ( ( $ANY$FwCHILDREN in_state OFF ) and ( $ANY$FwCHILDREN in_state ON ) ) move_to MIXED
        action: GO_READY
            do SWITCH_ON $ALL$FwCaenChannel
            move_to MOVING_READY

state : READY
    when ( $ANY$FwCHILDREN in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to ERROR
    when ( $ALL$FwCaenChannel in_state OFF ) move_to OFF
    when ( ( $ANY$FwCHILDREN in_state OFF ) and ( $ANY$FwCHILDREN in_state ON ) ) move_to MIXED
        action: GO_OFF
            do SWITCH_OFF $ALL$FwCaenChannel
                move_to MOVING_OFF

state : ERROR
    when ( $ANY$FwCHILDREN not_in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to OFF
    when ( $ANY$FwElmbAi in_state TOO_HOT ) do RESET
        action: RESET
            do SWITCH_OFF $ALL$FwCaenChannel
            move_to RECOVERING

state : RECOVERING
    when ( $ANY$FwCHILDREN not_in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to OFF

state: MIXED
    when ( $ANY$FwCHILDREN in_state {ERROR,OVERCURRENT,TRIPPED,TOO_HOT,NO_CONTROL} ) move_to ERROR
    when ( $ALL$FwCaenChannel in_state OFF ) move_to OFF
    when ( $ALL$FwCHILDREN in_state ON, OK ) move_to READY
        action: GO_OFF
            do SWITCH_OFF $ALL$FwCaenChannel
        action: GO_READY
            do SWITCH_ON $ALL$FwCaenChannel
```

The SML code of **CaenChannel**'s actions

```
FwCaenChannel_doCommand(string domain, string device, string command)
{
    if (command == "SWITCH_OFF")
    {
        dpSet(device+".settings.onOff",0);
        fwDU_startTimeout(30, domain, device, "NO_CONTROL", "OFF");
    }
    if (command == "SWITCH_ON")
    {
        dpSet(device+".settings.onOff",1);
        fwDU_startTimeout(30, domain, device, "NO_CONTROL", "ON");
    }
    if (command == "RESET")
    {
        dpSet(device+".settings.onOff", 0);
        fwDU_startTimeout(30, domain, device, "NO_CONTROL", "ON");
    }
}
```

The possible actions of **CaenChannel** are defined in this SML code. For example, the if statement in line 3 defines the action in case the SWITCH_ON command is executed. In the case, "dpSet(device+".settings.onOff",0)" inputs 0 into "settings.onOff" which is a DPE for control of the device's power. It is designed by CAEN to turn it on if settings.onOff == 1 and to turn it off if settings.onOff == 0 [20]. Then, "fwDU_startTimeout(30, domain, device, "NO_CONTROL", "OFF")" defines that the state moves to NO_CONTROL if no reaction are shown from the devices in 30 seconds after the SWITCH_OFF command are executed and to OFF if the command works [33].

The SML code of **CaenChannel**'s states

```
FwCaenChannel_valueChanged(string domain, string device, int actual_dot_status, string &fwState)
{
    bool isOn               = getBit(actual_dot_status, 0);
    bool ramping_up         = getBit(actual_dot_status, 1);
    bool ramping_down       = getBit(actual_dot_status, 2);
    bool OvC                = getBit(actual_dot_status, 3);
    bool Trip               = (getBit(actual_dot_status, 8) || getBit(actual_dot_status, 9));
    bool NoCon              = getBit(actual_dot_status, 11);

    if(Trip){
        fwState = "TRIPPED";
    }
    else if(OvC){
        fwState = "OVERCURRENT";
    }
    else if(isOn){
        fwState = "ON";
    }
    else if (NoCon){
        fwState = "NO_CONTROL";
    }
    else if(actual_dot_status != 0){
        fwState = "ERROR";
    }
    else{
        fwState = "OFF";
    }
}
```

When the state of **CaenChannel** changes is described here. The "getBit" function acquires the Most Significant Bit (MSB) in the DPE value of "actual.status". For instance, "bool isOn = getBit(actual_dot_status, 0)" in line 3 determines if the MSB is 0. In this code, "else if(isOn)" determine if "isOn" is true, and in the case of true, the state moves to ON by "fwState = "ON".

4.4 Results of the Actual Machine Tests

In this section, the results of the tests are described. Control of the whole system from the top node (Normal Operation) and a safety interlock with the FSM in the case of an abnormal rise in temperature (Software Interlock) were confirmed.

4.4.1 Normal Operation

Figure 4.8 - 4.15 are the control panels of **TEST_DCS** in JCOP. The devices are controlled using the panel. The upper box shows the state of **TEST_DCS** and the two boxes below are children's states. By clicking the box of **TEST_DCS** state when the state is OFF (4.8), GO_STANDBY command can be selected in the pull-down menu (Figure 4.9). Firstly, Power Converter A3486's channel turns on and the EASY board A3009 become ready by GO_STANDBY. Reflecting the devices' condition, the state of **LVPS** moves from NOT_READY to READY via STANDBY (Figure 4.9 - 4.12). Then, **TEST_DCS** moves to STANDBY and GO_READY can be selected (Figure 4.13). Executing the command, GO_READY is sent to **Detector** and the state of **MFT_DCS** goes to MOVING_READY. (Figure 4.14) Then, **Detector** sent SWITCH_ON to **CaenChannel**. When the channel

turns on, **CaenChannel** moves to ON, then **Detector** and **TEST_DCS** go to READY (Figure 4.15). To turn off the devices, the commands should be executed in the reverse order. Through these tests, I have confirmed that the devices below **LVPS** and **Detector** have been controlled by only the commands from the top node.

4.4.2 Software Interlock

The CAEN channels are turned off when an abnormal rise in temperature is detected by this FSM implementation. Figure 4.16 - 4.19 are the panels. In the tests, the temperature sensor was warmed up with hands, and if the value exceeds the threshold of TOO_HOT (T_{too_hot} = 30°C at this time), **Temp** moves to TOO_HOT. In these panels, channel000 and channel001 are **CaenChannel**s and PT_4W_0_1 is **Temp**. In Figure 4.16, channels were on and the temperature value was room temperature (22.7 °C). Then, warming up the sensor made **Temp** moves to TOO_HOT, and **Detector** went to ERROR. The temperature at that time was 33.6°C due to the reading rate of the ELMB. Immediately after that, **Detector** sent the RESET command to **CaenChannel** automatically and the state moved to RECOVERING (Figure 4.18) so that ERROR state in **Detector** and the possible command in ERROR cannot seen but in actually the panel like Figure 4.17 existed in an moment and then moved to RECOVERING. When the temperature went below the threshold of HOT (T_{hot} = 25°C at this time), **Tmp** went back to OK, and **Detector** moved to READY. In the tests, I have confirmed that the software safety interlock by measured temperature value have been implemented.

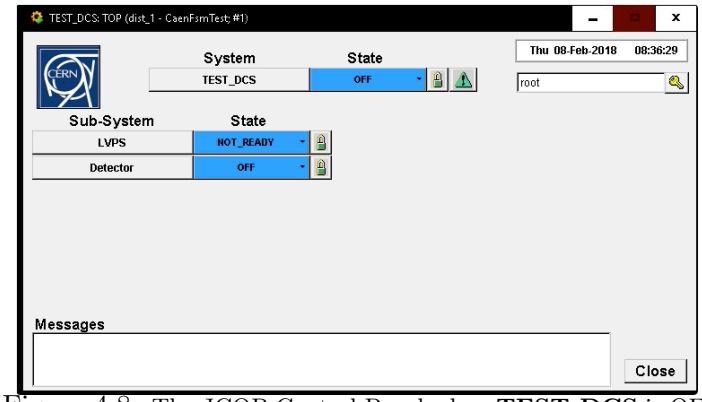

Figure 4.8: The JCOP Control Panel when **TEST_DCS** is OFF

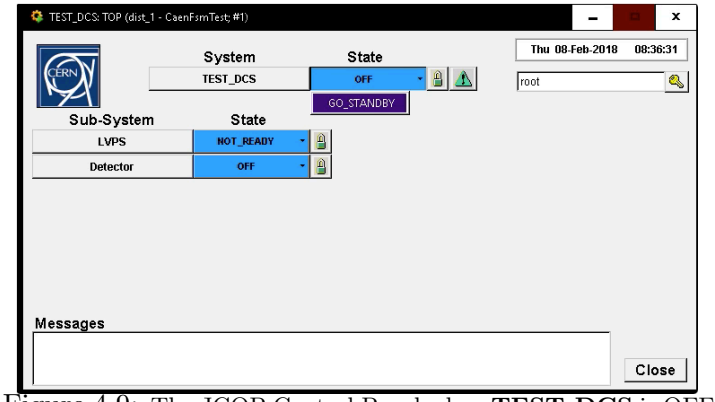

Figure 4.9: The JCOP Control Panel when **TEST_DCS** is OFF, GO_READY can be selected in the pull-down menu

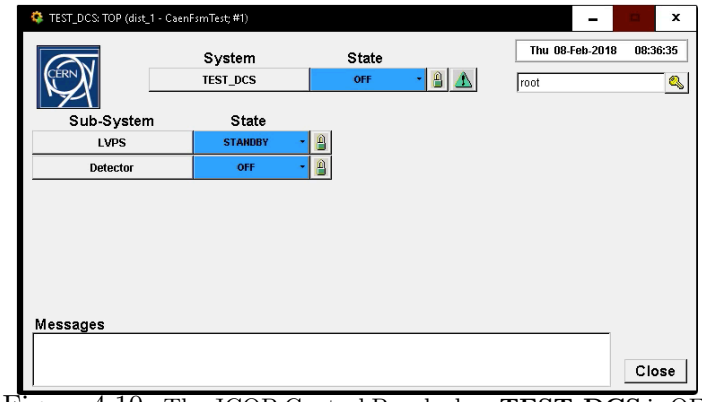

Figure 4.10: The JCOP Control Panel when **TEST_DCS** is OFF and **LVPS** is STANDBY

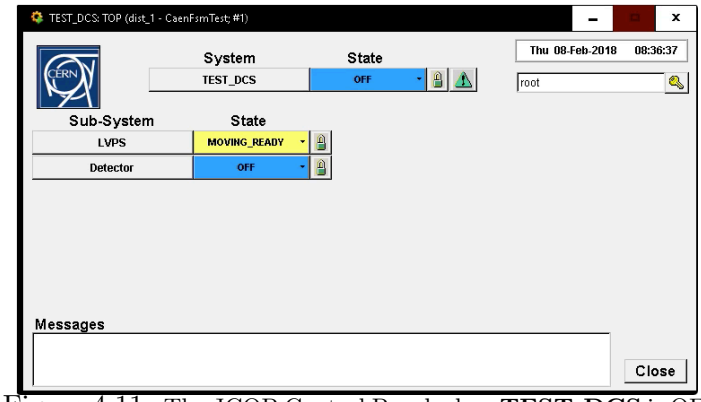

Figure 4.11: The JCOP Control Panel when **TEST_DCS** is OFF and **LVPS** is MOVING_READY

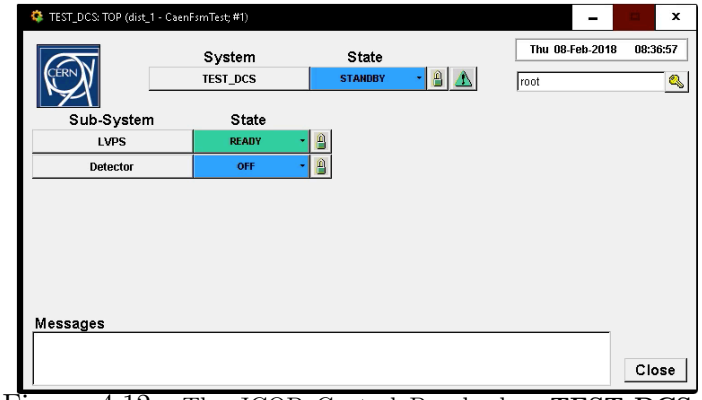

Figure 4.12: The JCOP Control Panel when **TEST_DCS** is STANDBY

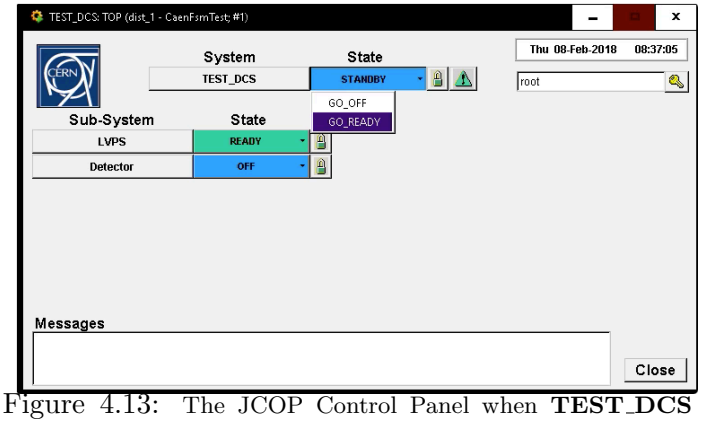

Figure 4.13: The JCOP Control Panel when **TEST_DCS** is STANDBY, GO_READY can be selected in the pull-down menu

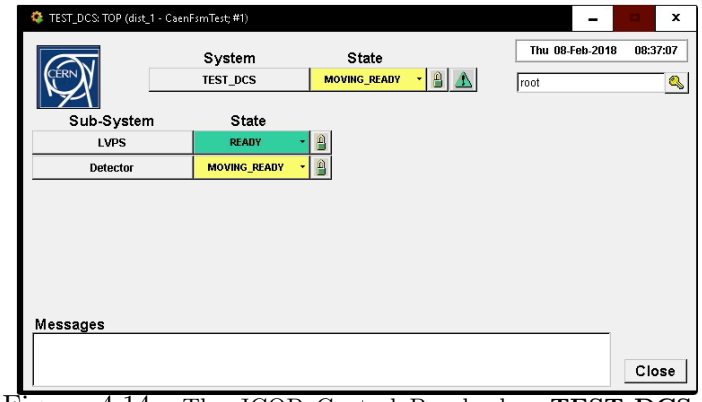

Figure 4.14: The JCOP Control Panel when **TEST_DCS** is MOVING_READY

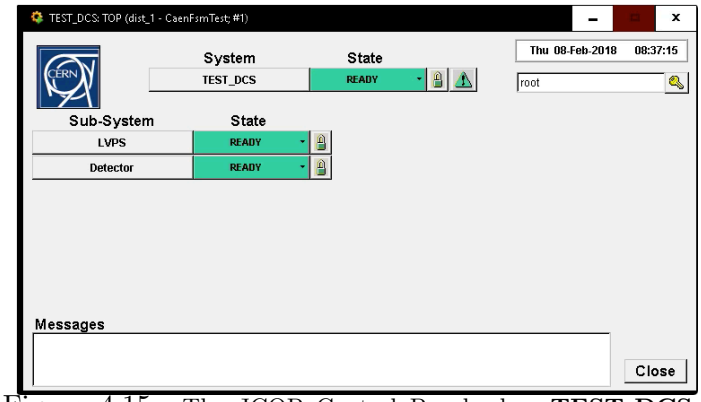

Figure 4.15: The JCOP Control Panel when **TEST_DCS** is READY

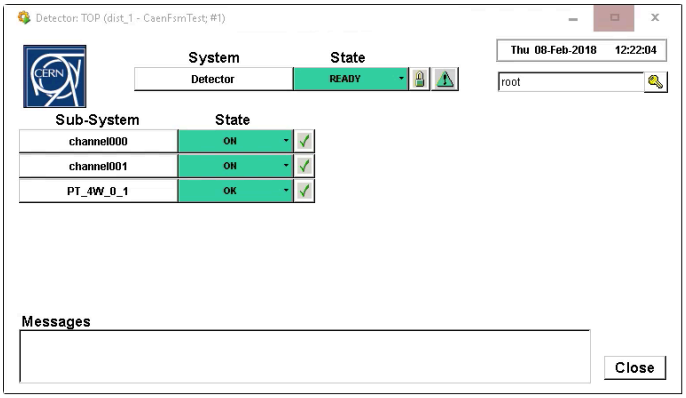

Figure 4.16: The JCOP Control Panel when **TEST_DCS** is READY

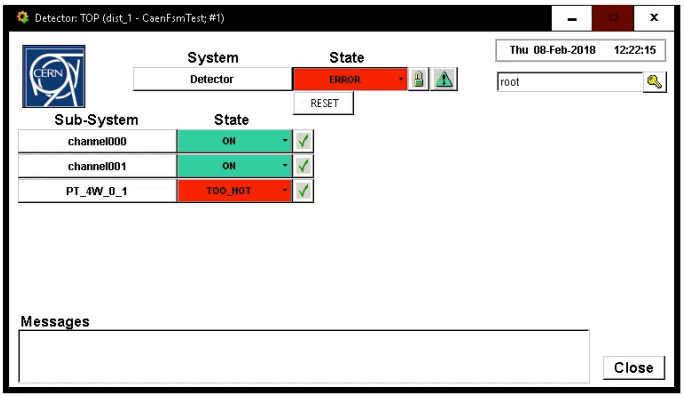

Figure 4.17: The JCOP Control Panel when **TEST_DCS** is ERROR

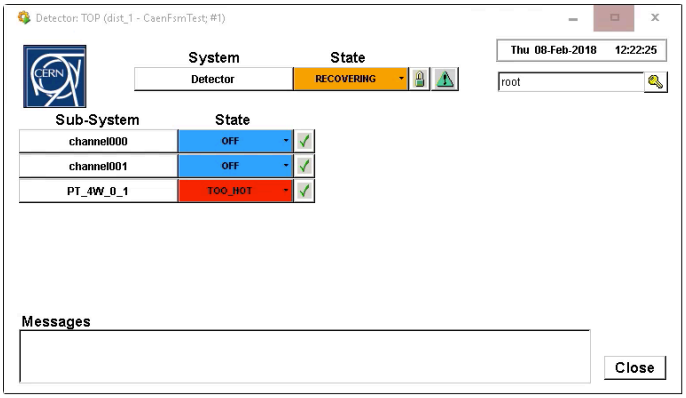

Figure 4.18: The JCOP Control Panel when **TEST_DCS** is RECOVERING

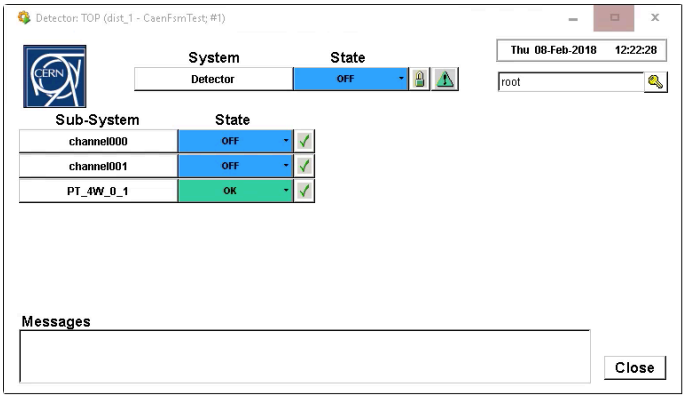

Figure 4.19: The JCOP Control Panel when **TEST_DCS** is OFF

4.5 Expanding the FSM for the Actual Machine Tests

The implementation of the FSM for the real MFT has been implementing by expanding the FSM for the actual machine tests. The part of control hierarchy which have been already implemented is Figure 4.20. There remains the devices whose hardware design has not been completed such as the GBT-SCA, the DCDC converter, etc and whose frameworks interfacing hardware and software have not been developed yet, so that either general analog or digital input is used in place of the devices and their frameworks to enable state transitions in the DUs. The device configuration from the FSM also has not been implemented yet. To complete implementation, the naming rule of equipment is needed. For example, there is **Half_Plane** in the control hierarchy but the MFT has 20 Half Planes in all since the MFT has 2 Half MFT, and each Half MFTs have 5 Half Disks, and each Half Disks have 2 Half Planes ($2 \times 5 \times 2 = 20$). Therefore each **Half_Plane** must have a specific name. The rule has been partly decided, but not completely [40]. After the completion, the specific names have to be decided following the rule. The FSM for the MFT has to be completed in 2018 including the remaining implementation.

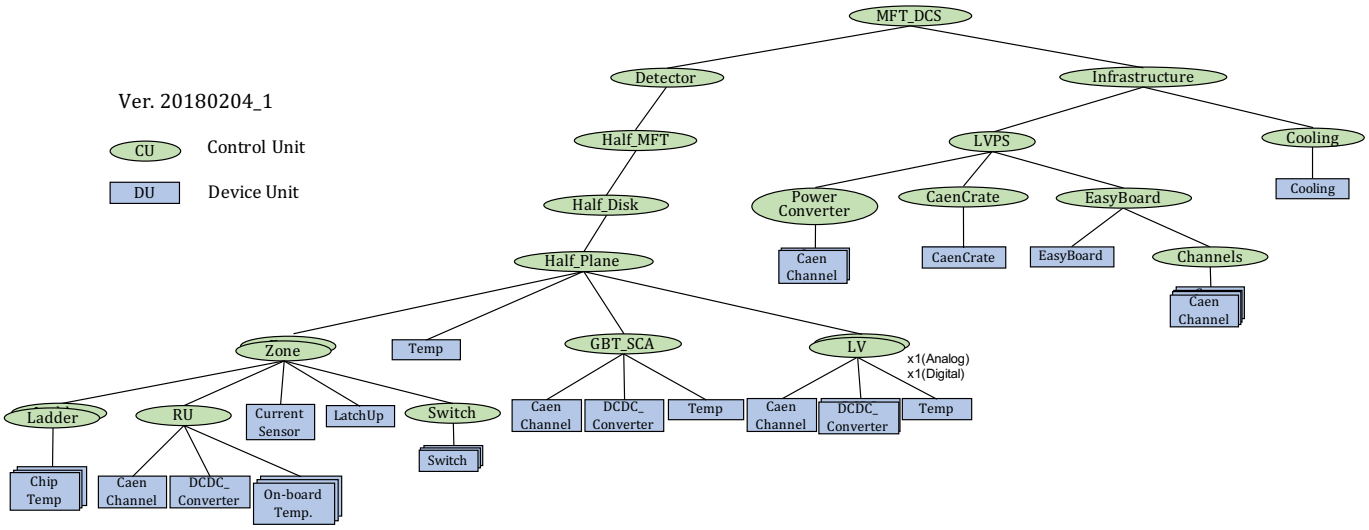

Figure 4.20: The Control Hierarchy already implemented

Chapter 5

Summary and Outlook

I have designed the FSM for the MFT: the control hierarchy and the state transitions. The design of the FSM has been completed. I have also carried out the actual machine tests of the FSM with the CAEN power supply equipment and a temperature sensor. Through the tests, I confirmed that the FSM works well. I have been implementing the FSM for the MFT by expanding the FSM for this test.

In the tests, the configuration of the devices have not been implemented. Moreover, there remain devices to be implemented and tested such as the GBT-SCA and the DCDC converters since their hardware and their JCOP frameworks are currently developing. The MFT will be installed in 2019, so that the MFT DCS including the FSM will be completed in 2018.

Acknowledgement

I would like to express my appreciate to supervisor Assoc. Prof. K. Shigaki. He gave me advices on the design of the FSM and supported my workshops participation. I would like to express my gratitude to staffs in Hiroshima University, Prof. T. Sugitate, Asst. Prof. K. Homma, and Asst. Prof. T. Miyoshi for their comments on my presentation. I would like to express my appreciate to K. Yamakawa. He gave me a lot of advices on the MFT FSM. I would like to thank O. Pinazza, P. Chochula, and the other members of the ALICE DCS group. They gave me a lot of advices and information for designing the FSM. I also would like to thank all of the members of the Quark Physics Laboratory for support my academic life.

Bibliography

[1] ALICE Collaboration, "The ALICE experiment at the CERN LHC", JINST 3 (2008) S08002

[2] ALICE, http://aliceinfo.cern.ch/Public/Welcome.html

[3] Longer term LHC schedule, https://lhc-commissioning.web.cern.ch/lhc-commissioning/schedule/LHC-long-term.htm

[4] ALICE Collaboration, "Upgrade of the ALICE Experiment: Letter of Intent", CERN-LHCC-2012-012 (2012)

[5] ALICE Collaboration, "Technical Design Report for the Upgrade of the ALICE Inner Tracking System", CERN-LHCC-2013-024 (2014)

[6] ALICE Collaboration, "Addendum to the Technical Design Report for the Upgrade of the ALICE Time Projection Chamber", CERN-LHCC-2015-002 (2015)

[7] ALICE Collaboration, "Addendum of the Letter of Intent for the upgrade of the ALICE experiment: The Muon Forward Tracker", CERN-LHCC-2013-014 (2013)

[8] ALICE Collaboration, "Technical Design Report for the Muon Forward Tracker", CERN-LHCC-2015-001 (2015)

[9] ALICE Collaboration, "Technical Design Report for the Upgrade of the Online-Offline Computing System", CERN-LHCC-2015-006 (2015)

[10] K. Yamakawa, "Development of the Detector Control System for the Muon Forward Tracker at ALICE", http://www.quark.hiroshima-u.ac.jp/thesis/master/17yamakawa_thesis.pdf

[11] Softech, SCADA system construction, http://www.softech.co.jp/scada_main.htm

[12] The LHCb collaboration, PVSS Introduction for Newcomers, https://lhcb-online.web.cern.ch/lhcb-online/ecs/PVSSIntro.htm

[13] About JCOP, http://jcop.web.cern.ch/

[14] OPC Foundation, https://opcfoundation.org/

[15] OPC Foundation (Japanese Page), https://jp.opcfoundation.org/

[16] C. Gaspar *et al.*, DIM, http://dim.web.cern.ch/dim/

[17] B. Hallgren *et al.*, The Embedded Local Monitor Board (ELMB) in the LHC Front-end I/O Control System, 7th Workshop on Electronics for LHC Experiments, Stockholm, Sweden, 10 - 14 Sep 2001, pp.325-330, CERN-2001-005

[18] CAEN, http://www.caen.it/csite

[19] CAEN, Technical Information Manual (A1676A), http://www.caen.it/servlet/checkCaenManualFile?Id=10067

[20] CAEN, Technical Information Manual (A3009), http://www.caen.it/servlet/checkCaenManualFile?Id=5402

[21] A. Caratelli *et al.*, "The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments", Journal of Instrumentation 10 (2015) C03034

[22] P. Chochula, "The ALICE DCS Training v.1.8", an internal document

[23] C. Gaspar, "JCOP Framework Hierarchical Controls Configuration & Operation", https://edms.cern.ch/ui/file/1035382/6/FSMConfig.pdf

[24] P.C.Burkimsher, "JCOP Framework Introductory demonstration for the Finite State Machine (FSM) component", https://edms.cern.ch/ui/file/1035382/6/fsmDemoInstructions.pdf

[25] O. Holme, S. Schmeing, "The FSM Toolkit General Usage", https://edms.cern.ch/ui/file/1035382/6/FSM_General_Usage.pdf

[26] O. Holme, "FSM Panel Reference Guide", https://edms.cern.ch/ui/file/1035382/6/FSM_Reference_Guide.pdf

[27] M. Boccioli, G. De Cataldo, "Alice DCS FSM integration guidelines", an internal document

[28] G. De Cataldo, *et al.*, " Finite State Machines for integration and control in ALICE ", Proc. ICALEPCS07, Knoxville, USA 650 (2007)

[29] SMI++ -State Manager Interface, http://smi.web.cern.ch/smi/index.html

[30] O. Holme, S. Schmeling, "SML Language & PVSS Libraries, "https://edms.cern.ch/ui/file/1035382/6/FSM_SML_and_PVSS_Libraries.pdf"

[31] C. Gaspar, "The Finite State Machine toolkit of the JCOP Framework", https://edms.cern.ch/file/1035382/6/FSM_Presentation.ppt

[32] F. Calheiros, "FSM Configuration DB Tool", https://edms.cern.ch/file/1035382/6/fwFSMConfDB.ppt

[33] A. Barriuso-Poy. "ATLAS DCS FSM Integration Guidelines", ATL/DQ/ON/0010

[34] MFT Working Package 3 (Disk), internal documents

[35] MFT Working Package 4 (Cone), internal documents

[36] A. Morreale, S. Martinez, "Low voltage powering and mapping of the MFT", internal documents (2018)

[37] K.M. Sielewicz *et al.*, "Prototype readout electronics for the upgraded ALICE Inner Tracking System", 2017 JINST 12 C01008

[38] K. Yamakawa, T. Kobayashi, and K. Shigaki, "MFT Users Requirements Document (URD_2018_02_00v1.pptx)", an internal document

[39] K. Yamakawa, private communications

[40] R. Tieulent, Naming and numbering convention, an internal document

Appendix A

A.1 The JCOP FSM Panel of LVPS in the actual machine tests

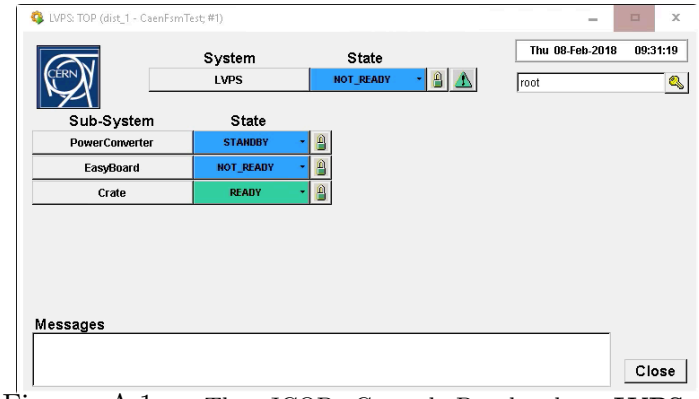

Figure A.1: The JCOP Control Panel when **LVPS** is NOT_READY

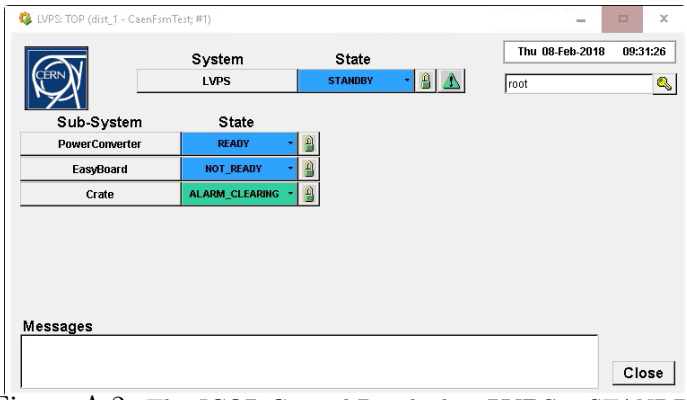

Figure A.2: The JCOP Control Panel when **LVPS** is STANDBY

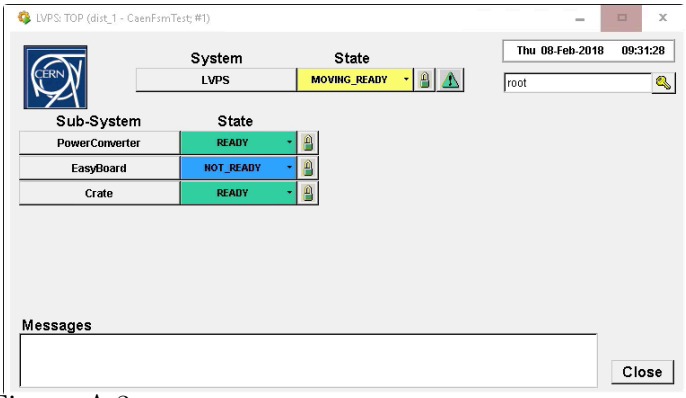

Figure A.3: The JCOP Control Panel when **LVPS** is MOVING_READY

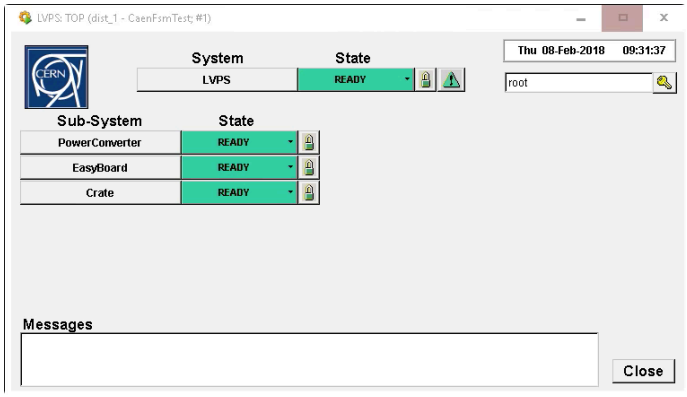

Figure A.4: The JCOP Control Panel when **LVPS** is READY

A.2 The JCOP FSM Panel of Detector in the actual machine tests

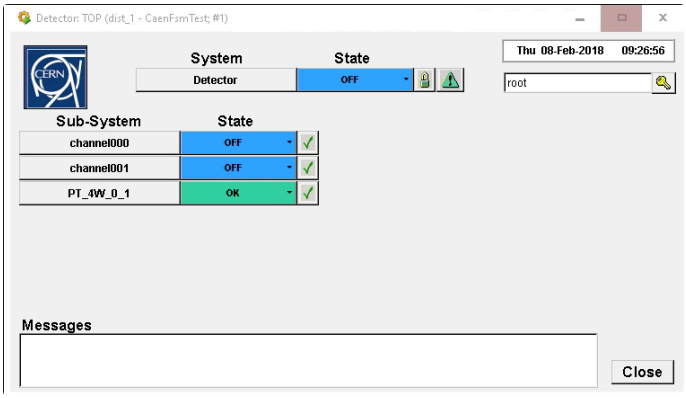

Figure A.5: The JCOP Control Panel when **Detector** is OFF

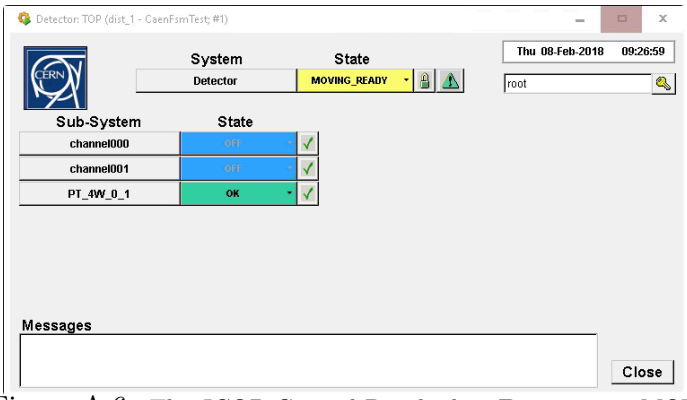

Figure A.6: The JCOP Control Panel when **Detector** is MOVING_READY

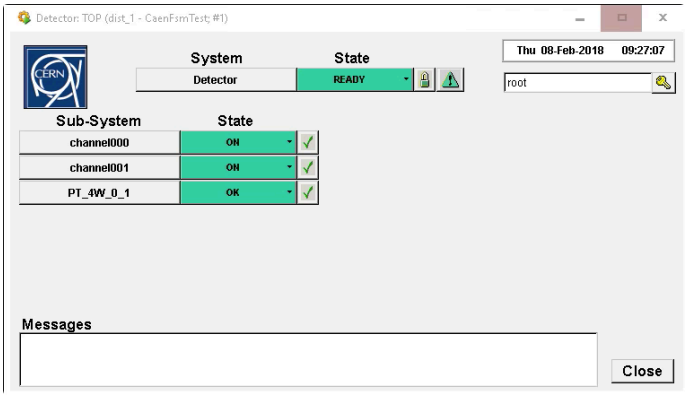

Figure A.7: The JCOP Control Panel when **LVPS** is READY