Development of the detector control system for the COMPASS detector at CERN

Christian Schuhegger
Technische Universität München

27th July 2001

Abstract

This document describes the implementation of system control software for the COMPASS [1] experiment at CERN. This work concentrates on the GEM and silicon detectors, but it also includes parts that are generally useful for all kinds of detectors. The only prerequisites were the PVSS II SCADA-system [87] and the JCOP [90] PVSS framework [91] distributed by ITCO [89] at CERN. To achieve the given aims there was work to do both on a C++ framework called SLiC [88] for hardware access and on top of the JCOP framework to customise it for the special needs of the GEM and silicon detectors.

Foreword

Because this is a diploma work in physics and my actual work did not have any obvious link to physics I decided to write an overview of the physics in the COMPASS experiment. After this decision I had to decide about the level of detail of the overview. As I had in the beginning some difficulties to find information of a student's level about the experiment and the involved theory I decided to do a student's level introduction to the physics in COMPASS. With this introduction to COMPASS I hope to help other students who will work for COMPASS in the future to find a starting point for further readings.

My whole report tries to be fluently readable for a final year student who has already some background in the area (either physics or computing), but who is not an expert. Therefore, this report might seem a bit too wordy and a bit of too much unnecessary detail for an expert. On the other hand the glossary tries to close the gap between my level of introduction and a reader who has only a little background in the area.

The glossary is to retain the readability for a student's level reader and therefore is a reference for concepts that would lead the text too far off the subject. The index on the other hand is a reference for concepts that were explained in text.

The schema behind where to put footnotes and where to do an in text explanation in parenthesis might seem without concept. An explanation might be that as stressed above I tried to do a fluently readable text (which is open to interpretation) and the concept I took to decide where to put footnote information, either in the text or as a real footnote, was my feeling about readability, which might not be the same for any other reader.

The final point I want to remark is that the amount of information given for some subject is by no means a measure for the importance of that subject. The amount of information I put into this work per subject is just a sign of how much information I had available for that subject!

Contents

List of Acronyms

ACE ..... Adaptive Communication Environment [73]
ACID ..... Atomicity Concurrency Isolation and Durability
ADC ..... Analogue to Digital Converter
API ..... Application Programmer Interface
BMS ..... Beam Momentum Station
CEDAR ..... Čerenkov Differential counter with Achromatic Ring focus (see page 30 and reference [50])
CERN ..... Organisation Européenne pour la Recherche Nucléaire
CHEOPS ..... Charm Experiment with Omni-Purpose Setup [2]
c.m. ..... centre-of-mass
COMPASS .... Common Muon and Proton Apparatus for Structure and Spectroscopy [1]
χPT ..... Chiral Perturbation Theory
CPU ..... Central Processing Unit
CVS ..... Concurrent Version System [81]
DCS ..... Detector Control System
DGLAP ..... Dokshitzer-Gribov-Lipatov-Altarelli-Parisi evolution equations
DIM ..... Distributed Information Management System [75]
DIS ..... deep-inelastic scattering
DISC ..... Directional Isochronous Self Collimating
DLL ..... Dynamic Linked Library
DNP ..... Dynamic Nuclear Polarisation
DNS ..... DIM Name Server
DP ..... PVSS data point
DPE ..... PVSS data point element
DPT ..... PVSS data point type
DTD ..... Design Type Definition
ECAL ..... Electromagnetic Calorimeter
EMC ..... European Muon Collaboration
FIFO ..... First In First Out – also called named pipe
FLT ..... First Level Trigger
FODO ..... Focusing-Defocusing quadrupoles
FSM ..... Finite State Machine
GEM ..... Gas Electron Multiplier
GUI ..... Graphical User Interface
HARP ..... Hadron Production for the Neutrino Factory and for the Atmospheric Neutrino Flux [7]
HCAL ..... Hadronic Calorimeter
HEP ..... High Energy Physics
HERA-B ..... An experiment to study CP violation in the B system using an internal target at the HERA proton ring [8]
HMC ..... Hadron Muon Collaboration [3]
HQET ..... Heavy Quark Effective Theory
HTML ..... Hyper Text Markup Language
IC ..... Integrated Circuit
IPC ..... Inter Process Communication
IT ..... Information Technology

ITCO ..... IT - Information Technology - COntrols Group [89]
JCOP ..... Joint Controls Project [90]
LAN ..... Local Area Network
LAS ..... Large Angle Spectrometer
LAT ..... Large Area Tracking
LHC ..... Large Hadron Collider
LO ..... leading order
Micromegas .... Micromesh Gaseous Chamber
MSGC ..... Micro-strip Gas Counter
MW ..... Muon Wall
MWPC ..... Multi-Wire Proportional Chamber
NIEL ..... Non Ionising Energy Loss
NLO ..... next to leading order
OLE ..... Object Linking and Embedding
OO ..... Object Oriented
OPC ..... Object Linking and Embedding (OLE) for Process Control
OS ..... Operating System
PCB ..... Printed Circuit Board
PGF ..... photon-gluon fusion process
PIT ..... Plastic Iarocci Tube [29]
PKA ..... Primary Knock on Atom
PLC ..... Programmable Logic Control(ler)
PVSS II ..... Prozessvisualisierungs- und Steuerungssystem [87]
PWA ..... Partial Wave Analysis
QCD ..... Quantum Chromo Dynamic
RD ..... Research & Development Projects at CERN
RICH ..... Ring Imaging Čerenkov Counter
RMS ..... Root Mean Square
SAT ..... Small Area Tracking
SC ..... Slow Control
SCADA ..... Supervisory Control and Data Acquisition
SLiC ..... SLiC stands for Slow Control
SM ..... Spectrometer Magnet
SMC ..... Spin Muon Collaboration [4]
SPS ..... Super Proton Synchrotron
UI ..... User Interface
XML ..... eXtensible Markup Language

1 Introduction

The COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy [1]) experiment was founded, as two different physics groups, the CHEOPS (Charm Experiment with Omni-Purpose Setup [2]) group and the HMC (Hadron Muon Collaboration [3]) group, with different physical aims recognised that the experimental set up to achieve these aims were very similar. Therefore, COMPASS is a successor of the experiments SMC (Spin Muon Collaboration [4]), WA89 [5] and WA102 [6]. The COMPASS experiment consists of two major physical programmes, the muon programme and the hadronic programme. Both programmes have several aspects. One important part of the muon programme is for example to analyse the "spin puzzle" and one important part of the hadronic programme is for example the study of hadronic states with charm flavour quantum number.

The COMPASS experiment is special, because no other fixed target experiment before had to cope with as high event rates as COMPASS proposes to do. On the one hand, these high event rates are necessary to achieve the set aims but on the other hand, these high event rates are also a challenging task on their own for experimental physics. The rates alone already impose some restrictions on the type of detectors, which can be used, and on the electronics to gather and process the gained results.

Nowadays physics experiments are not only a challenge for the technical realisation of the detectors, but also for the complexity management software like a DCS (Detector Control System) system (see glossary: Detector Control System). There is the high voltage system, the temperature measurement, the gas control and much more for every detector and for every cell of the detector. Therefore an experiment needs a sophisticated system for the detector control.

The subject of this work was to write software to integrate the GEM (Gas Electron Multiplier) and silicon detectors into the common system control framework. As COMPASS is one of the first experiments at CERN (Organisation Européenne pour la Recherche Nucléaire), which uses the PVSS II (Prozessvisualisierungs- und Steuerungssystem [87]) SCADA (Supervisory Control and Data Acquisition) system, it was not possible to use existing software for some major parts of the job to be done. For the C++ framework SLiC (SLiC stands for Slow Control) it was for example necessary to first produce a design and then to implement it.

The SCADA system already provides

1. an abstraction from the network accesses,
2. an archiving database,
3. a scripting language,
4. a user interface builder,
5. scalability,
6. user rights and user management.

On top of this SCADA system, the JCOP (Joint Controls Project [90]) project has already written a framework to support every detector team in building the control software for their detector The framework includes

1. a design of how to model detectors and sub devices,
2. an alarm handling scheme,
3. a user interface for managing and configuring the models of the detectors and
4. run-time user interfaces for common hardware elements.

Furthermore, the framework provides a set of tools, which implement missing functionality of the SCADA system, such as FSM (Finite State Machine) support. The framework uses interfaces to facilitate the integration of the different layers of the control system and to hide the underlying tools as much as possible from the users, thus reducing the amount of training and support required. The frameworks therefore allow the development of a coherent and homogeneous system by multiple and remote teams of developers.

The implementer of the control software of a certain detector now has to extend this framework for his special needs, which often just means that he has to assemble the different predefined parts of the framework to represent the detector in question.

Section two of this work describes the physical background of the experiment and the principles of the detectors that are used. It gives the reader a consistent overview of the experiment, the physics involved and the aims followed. It is therefore a collection of data out of several information sources in the context of COMPASS. Section three will outline the general principles of slow control. We follow the complete path from the hardware to the system supervisor in front of the system control screen. After a description of all requirements in section four, section five describes the readout process of the hardware connected to field-buses (see glossary: Field-bus). Section six then shows how this information is integrated in the common system control environment of COMPASS. Finally, section seven and eight give an overview of the achieved results in terms of efficiency and reliability and an outlook for future develpments.

2 The COMPASS Experiment

This section aims to provide a consistent, easy to follow and complete introduction to the physical background of the COMPASS experiment and its technical realisation.

2.1 Review of some theoretical aspects of deep-inelastic scattering

As this sub-section is a short version of section 2 ("Review of some theoretical aspects") in reference [9] the same equation numbers as in reference [9] are used. It describes the basics of deep-inelastic scattering (DIS) and structure functions.

2.1.1 Kinematics

In a typical fixed-target DIS experiment a lepton of energy E scatters from a nucleon or nuclear target at rest under an angle θ and with a final energy E'. Three Lorentz invariants can be constructed from these laboratory variables and the nucleon mass M.

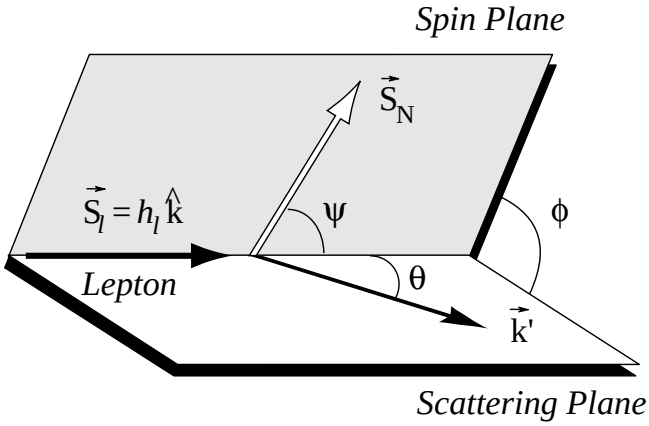

Figure 1: Kinematics of polarised deep-inelastic lepton-nucleon scattering.

$$q^2 = (k - k')^2 \stackrel{\text{lab}}{=} -4EE' sin^2 \frac{\theta}{2}, \tag{2.1}$$

$$P \cdot q \stackrel{\text{lab}}{=} M\nu \qquad = M(E - E'), \tag{2.2}$$

$$P \cdot k \stackrel{\text{lab}}{=} ME, \tag{2.3}$$

where k, k' and $P = (M, 0, 0, 0)$ are the four-momenta of the incoming and scattered lepton and of the target nucleon, respectively. In equation (2.1) the lepton mass was neglected as it will be throughout this review. The cross-section can then be written as a function of the dimensionless scaling variables $0 \leq x, y \leq 1$

$$x = \frac{-q^2}{2P \cdot q} \stackrel{\text{lab}}{=} \frac{Q^2}{2M\nu} \tag{2.4}$$

$$y = \frac{P \cdot q}{P \cdot k} \stackrel{\text{lab}}{=} \frac{\nu}{E}. \tag{2.5}$$

Other characteristic variables of the scattering process are the c.m. (centre-of-mass) energy, $\sqrt{s}$, and the c.m. energy of the hadronic final state, W,

$$s = (k + P)^2 = \frac{Q^2}{xy} + M^2 \tag{2.6}$$

$$W^2 = (q + P)^2 = \frac{1 - x}{x} Q^2 + M^2 \tag{2.7}$$

2.1.2 The deep-inelastic cross-section

Here only photon exchange is considered, as the energy in the experiment will be well below the W, Z masses. Then the BORN cross-section for inclusive (see glossary: Inclusive Measurement of Interactions) inelastic scattering of a charged lepton from a nucleon $\ell N \to \ell' X$, can be expressed as a product of a leptonic tensor, $l_{\mu\nu}$, and a hadronic tensor, $W^{\mu\nu}$,

$$\frac{\mathrm{d}^3\sigma}{\mathrm{d}x\mathrm{d}y\mathrm{d}\phi} = \frac{\alpha^2}{Q^4}\frac{y}{2} l_{\mu\nu} W^{\mu\nu}, \tag{2.8}$$

The tensors $l_{\mu\nu}$ and $W^{\mu\nu}$ involve the leptonic and hadronic electromagnetic currents, respectively (see reference [9] equations (2.10) to (2.16)). The general form of the hadronic tensor for a spin-$\frac{1}{2}$ target is

$$\begin{aligned}
\frac{1}{2}W^{\mu\nu(\mathrm{S})} &= -\left(g^{\mu\nu} - \frac{q^\mu q^\nu}{q^2}\right) F_1 + \left(P^\mu - \frac{P\cdot q}{q^2}q^\mu\right)\left(P^\nu - \frac{P\cdot q}{q^2}q^\nu\right)\frac{F_2}{P\cdot q} \\
\frac{1}{2}W^{\mu\nu(\mathrm{A})} &= -\epsilon^{\mu\nu\alpha\beta}q_\alpha\left(\frac{MS_\beta}{P\cdot q}(g_1+g_2) - \frac{M(S\cdot q)P_\beta}{P\cdot q}g_2\right).
\end{aligned} \tag{2.17}$$

Here the dimensionless *structure functions* F_1, F_2, g_1 and g_2 are in general functions of $P\cdot q$ and Q^2. S is the nucleon's polarisation vector.

2.1.3 Photoabsorption

As the scattering process of a lepton with the nucleon involves the radiation of a virtual photon off the lepton, which in turn is absorbed by the nucleon, the differential cross-section is related to the transverse and the scalar virtual-photon flux and the transverse and scalar virtual photoabsorption cross-section. Therefore, also the lepton and virtual-photon cross-section asymmetries are related (for a more detailed analysis see reference [9]).

2.1.4 Cross-section asymmetries

The observable spin-dependent effects in experiments are small and appear on top of the unpolarised cross-section. They must be determined from the differences of cross-sections, which are sensitive to small changes of the apparatus' acceptance. These systematic uncertainties largely cancel in the cross-section asymmetries.

$$A_\parallel(x,Q^2;E) = \frac{\Delta_\parallel\sigma}{\bar{\sigma}} = \frac{\sigma^{\overrightarrow{\Leftarrow}} - \sigma^{\overrightarrow{\Rightarrow}}}{\sigma^{\overrightarrow{\Leftarrow}} + \sigma^{\overrightarrow{\Rightarrow}}}, \tag{2.38}$$

$$A_\perp(x,Q^2;E) = \frac{\Delta_\perp\sigma}{\bar{\sigma}} = \frac{\mathcal{H}_\ell}{cos\phi}\cdot\frac{\sigma(\phi) - \sigma(\pi\pm\phi)}{\sigma(\phi) + \sigma(\pi\pm\phi)} \tag{2.39}$$

where $\to$ and $\Rightarrow$ indicate the beam and target polarisation, respectively and σ denotes the differential cross-section.

The parallel and perpendicular lepton asymmetries, $A_\parallel$ and $A_\perp$, do not have a straightforward physics interpretation and in addition they strongly depend on E or y. Hence, asymmetries obtained in experiments performed at different incident energies cannot be compared directly. Therefore it is customary to express the lepton asymmetries in terms of the virtual-photon asymmetries A_1 and A_2, which are functions of x and Q^2 only. From reference [14, 16] we see, that the SMC experiment measured A_2^{p} and A_2^{d} and they were found to be consistent with zero in their range of Q^2 and x.

Generally, one finds the relation between the asymmetries

$$\begin{pmatrix} A_\parallel/D \\ A_\perp/d \end{pmatrix} = \begin{pmatrix} 1 & \eta \\ -\xi & 1 \end{pmatrix}\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \tag{2.45}$$

where the depolarisation factor D and the three factors d, η and ξ are kinematic factors (an explicit expression can be found in reference [9] equations (2.43) and (2.44)).

The reason why this formula is cited here without any further explanation is, because it shows that the relation between the lepton asymmetries and the photon asymmetries is a simple linear equation. Therefore the lepton asymmetries and photon asymmetries can be used interchangeably. Furthermore we will use this equation later in sub-section 2.2.1 in equation 27, where we drop the A_2 dependency as justified above.

2.1.5 The quark parton model

The (QCD[1]-improved) quark parton model allows us to understand structure functions in terms of quarks and gluons. Due to the asymptotic freedom of QCD at high momentum transfers, Q^2, in DIS a hadron behaves like an incoherent superposition of free partons. If the hit parton, q, carries the fraction ξ of the hadron's 4-momentum, $p_q = \xi P$, then the $W^{\mu\nu}$ tensor for a free massless spin-$\frac{1}{2}$ parton inside the hadron can be calculated and one finds the structure functions

$$\hat{F_1} = \frac{1}{2} e_P^2 \delta(\xi - x), \quad \hat{F_2} = e_P^2 \xi \delta(\xi - x), \quad \hat{g}_1 = \lambda \frac{1}{2} e_P^2 \delta(\xi - x), \quad \hat{g}_2 = 0 \tag{2.58}$$

where e_P is the parton's charge. The factor $\lambda = \pm 1$ accounts for the fact that g_1 is defined using the hadron's spin orientation and therefore an additional minus sign is needed when the parton's spin is oriented opposite to the one of the hadron.

The probability to find inside a hadron a parton of a certain type carrying a momentum fraction, ξ, is parameterised by the parton distribution functions, $q_i^\lambda(\xi)$. For the quarks we use $q_i = u, d, s, \ldots$ and for the gluon $q_i = g$. For a longitudinally polarised hadron, parallel and antiparallel orientation of the parton spin with respect to the hadron spin are denoted by $\lambda = \pm 1$. For a transversely polarised hadron, parallel and antiparallel orientation of the parton spin with respect to the hadron spin are denoted by $\lambda = (\uparrow, \downarrow)$. These functions are number densities normalised to the total number of partons of the considered type in the hadron. Usually the $q_i(x)$ are understood as the sum of the distribution functions of quarks and antiquarks in both helicity states. For clarity we therefore introduce $\hat{q}$ for quarks and $\bar{q}$ for antiquarks and denote the difference of the helicity states by Δq,

$$q^\pm = \hat{q}^\pm + \bar{q}^\pm, \quad q = q^+ + q^-, \quad \Delta q = q^+ - q^- \tag{2.59}$$

The hadron structure functions are then given by

$$\mathcal{F}(x) = \sum_{i,\lambda} \int_0^1 \mathrm{d}\xi q_i^\lambda(\xi) \hat{\mathcal{F}}_i^\lambda(x, \xi), \tag{2.60}$$

where i runs over all quark flavours and $\mathcal{F} = F_1$, F_2, g_1 and g_2, yielding

$$F_1(x) = \frac{1}{2} \sum_i e_i^2 \left\{ q_i^+(x) + q_i^-(x) \right\} \tag{2.61}$$

$$F_2(x) = x \sum_i e_i^2 \left\{ q_i^+(x) + q_i^-(x) \right\} \tag{2.62}$$

$$g_1(x) = \frac{1}{2} \sum_i e_i^2 \left\{ q_i^+(x) - q_i^-(x) \right\} \tag{2.63}$$

$$g_2(x) = 0 \tag{2.65}$$

[1] Quantum Chromo Dynamic

In the limit $Q^2 \to \infty$, where the quark parton model is applicable, the structure functions do not depend on ν and Q^2 separately, but become functions of x only. Another consequence of the quark parton model is the CALLAN-GROSS *relation*

$$F_2(x) = 2xF_1(x) \tag{2.66}$$

2.1.6 Sum rules in the quark parton model

Before we start with the proper treatment of this subject, it is instructive to make a few remarks about an intuitive but wrong picture of the nucleon spin (see reference [10]). In the simple quark model the spin of the proton is carried by its three valence quarks, so that $\Delta\Sigma = \Delta u + \Delta d = 1$. In general terms one writes the spin equation for a nucleon as

$$\frac{1}{2} = \frac{1}{2}\Delta\Sigma + \Delta g + \langle L_z \rangle$$

where $\Delta\Sigma = \Delta u + \Delta d + \Delta s$ is the contribution of the quarks, Δg is the contribution of the gluons, and $\langle L_z \rangle$ is a possible contribution from the gluons' and quarks' angular momentum. In the simple quark model

$$\begin{aligned} u^+ &= \frac{5}{3},\ u^- = \frac{1}{3},\ \Delta u = \ \ \frac{4}{3},\ u = 2 \\ d^+ &= \frac{1}{3},\ d^- = \frac{2}{3},\ \Delta d = -\frac{1}{3},\ d = 1 \end{aligned}$$

the three valence quarks are in an S-state, so $\langle L_z \rangle = 0$, and the spin sum rule is satisfied by $\Delta\Sigma = 1$. Since the EMC (European Muon Collaboration) discovery (see below), we know that this picture does not correspond to reality, and that the contribution of the quarks to the spin of the nucleons is much smaller.

The ELLIS-JAFFE (reference [26]) and BJORKEN sum rules are sum rules for the first moment of g_1,

$$\Gamma_1 = \int_0^1 g_1(x)\mathrm{d}x.$$

They follow in the quark parton model immediately from equation (2.63). They are quoted here without explanation only to give the reader an idea of their analytical formulation.

$$\Gamma_1^p - \Gamma_1^n = \frac{1}{6}g_a \qquad \text{(BJORKEN } \textit{sum rule}\text{)}. \tag{2.82}$$

$$\Gamma_1^{p,n} = \frac{1}{12}g_a \left\{ \pm 1 + \frac{5}{3}\frac{3F/D - 1}{F/D + 1} \right\} \qquad \text{(ELLIS-JAFFE } \textit{sum rule}\text{)}. \tag{2.83}$$

For a further clarification of the meaning of the variables, see reference [9] equations (2.71) to (2.81).

In the quark parton model the total contributions of the quarks to the nucleon's spin is given by

$$\Delta\Sigma = \Delta u + \Delta d + \Delta s = a_0 \tag{2.84}$$

The value of a_0 and a_8 (used below) are linear combinations of the Δq_i. As explained above, naively one expects $\Delta\Sigma$=1. ELLIS and JAFFE assumed that the strange quarks are not polarised $\Delta s \equiv 0$, leading to $a_0 = \sqrt{3}a_8$ and thus $\Delta\Sigma = 0.579 \pm 0.026$. The value $\Delta\Sigma = 0.12 \pm 0.16$ found by the EMC ([23, 24]) experiment came as a big surprise. The latest result obtained by the HERMES collaboration [25] is $\Delta\Sigma = 0.30 \pm 0.04 \pm 0.09$.

2.2 The muon programme

The muon programme is four-fold. After it is now firmly established that the spin content of the nucleon is not entirely due to the valence quark spins, a decision between the competing models is needed. In the gluon interpretation, it is the polarised glue Δg (so-called axial anomaly [20, 21, 22], see also reference [9] equation (2.122)), which lowers the quark's contribution to the nucleon spin, whereas in an alternative model negatively polarised strange sea quarks are responsible (Skyrme model [15]). Therefore, the first two points in the muon programme are to decide between the two models

1. by measuring $\Delta g/g$ via the photon-gluon fusion process (PGF)

and

2. measuring the polarisation of strange quarks and/or antiquarks by measuring the longitudinal polarisation of Λ and $\bar{\Lambda}$ in the target and current fragmentation regions (see glossary: Fragmentation Region).

The other two points are

3. the measurement of the longitudinal spin distribution functions,

and

4. the measurement of the transverse spin distribution functions, which were up to now never measured.

The observation of $\Lambda(\bar{\Lambda})$ in the current fragmentation region also addresses the problem of spin dependent fragmentation functions (see glossary: Fragmentation Function), which are closely related to the spin structure functions (see *structure functions* in section 2.1.2 on page 9).

A total running of one and a half years with a ^{6}LiD target with 50% polarisation and of one year with a NH_3 target of 85% polarisation are planned. The running time will be split in 80% with longitudinal and 20% with transverse target polarisation.

2.2.1 $\Delta g/g$ via the photon-gluon fusion process

This section is a summary of reference [13] in the context of the preceding review of theoretical aspects.

After the discovery of the EMC that the Ellis-Jaffe sum rule (see equation (2.83)) was not obeyed for the proton and the neutron a direct measurement of the gluon polarisation $\Delta g/g$ is mandatory for a further clarification of the internal spin structure of the nucleon. The prediction of the Ellis-Jaffe sum rule (see equation (2.84)) are for $\Delta\Sigma = \sqrt{3}a_8 \approx 0.6$ and the measurements give us a value of $\Delta\Sigma \approx 0.2$. To repeat what was said above: If the gluon polarisation is large one model expects the axial anomaly to be responsible for the reduction of $\Delta\Sigma$. In the Skyrme model negatively polarised strange sea quarks are expected to produce the small values of $\Delta\Sigma$.

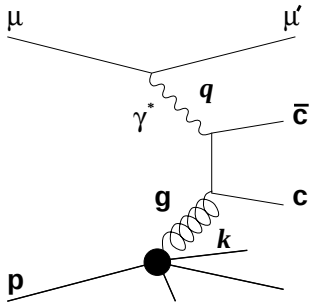

Figure 2: The PGF diagram.

Charm production in DIS is in leading order only due to the photon-gluon fusion process (PGF). Other processes only contribute in higher orders. Furthermore, since there is no or only a small intrinsic charmed quark distribution in the nucleon, diagrams with an incoming charmed quark do not contribute. Therefore, the PGF process is one of the cleanest processes that depend on the gluon distribution.

For real photons the cross-section of the subprocess $\gamma g \to c\bar{c}$ can be written as

$$\sigma^{\gamma g \to c\bar{c}} = \sigma(\hat{s}) + \lambda_\gamma \lambda_g \Delta\sigma(\hat{s}), \tag{24}$$

where $\hat{s} = (q+k)^2$ is the square of the energy and $\lambda_{\gamma,\,g}$ are the helicities in the photon-gluon c.m. system. The spin averaged part, $\sigma(\hat{s})$, and the spin dependent part, $\Delta\sigma(\hat{s})$, are given in leading order (LO) by equation (6.2) of reference [9]. Because of colour coherence of gluon couplings [17] it is expected that for $x \to 0$ the gluon polarisation behaves like $\Delta g/g \sim x$. Most proposed polarised gluon distributions exhibit such behaviour.

To obtain the photon-nucleon cross-section asymmetry, $A_{\gamma N}^{c\bar{c}}$, for the process $\gamma N \to c\bar{c}$ we must integrate over the c.m. energy, $\hat{s}$, from the threshold to the maximum available energy

$$A_{\gamma N}^{c\bar{c}}(\nu) = \frac{\Delta\sigma^{\gamma N \to c\bar{c}X}}{\sigma^{\gamma N \to c\bar{c}X}} = \frac{\int_{4m_c^2}^{2M_N\nu} \mathrm{d}\hat{s}\Delta\sigma(\hat{s})\Delta g(\eta,\hat{s})}{\int_{4m_c^2}^{2M_N\nu} \mathrm{d}\hat{s}\sigma(\hat{s}) g(\eta,\hat{s})}, \tag{25}$$

where η is the momentum fraction of the nucleon carried by the gluon ($\eta = \hat{s}/2M_N\nu$), M_N is the nucleon mass and ν is the photon energy. The experimentally observed asymmetry is

$$A^{\mathrm{exp}} = \frac{N^{\overrightarrow{\Leftarrow}} - N^{\overrightarrow{\Rightarrow}}}{N^{\overrightarrow{\Leftarrow}} + N^{\overrightarrow{\Rightarrow}}} = P_\mu P_t f A_{\mu N}^{c\bar{c}}, \tag{26}$$

where N is the number of charm-production events with antiparallel and parallel longitudinal polarisation of the muon and the target nucleon. The asymmetry is reduced by the beam and target polarisations, P_μ and P_t, and the *dilution factor* f[2], which is the fraction of polarisable nucleons in the target material. The muon-nucleon asymmetry $A_{\mu N}^{c\bar{c}}$ is related to the discussed above photon-nucleon asymmetry, $A_{\mu N}^{c\bar{c}}$, by

$$A_{\mu N}^{c\bar{c}}(E,y) = D A_{\gamma N}^{c\bar{c}} \approx \frac{1-(1-y)^2}{1+(1-y)^2} A_{\gamma N}^{c\bar{c}}(E,y). \tag{27}$$

The *depolarisation factor*, D, accounts for the lower polarisation of the virtual photon compared to the parent muon. As discussed above the complete relation is like equation (2.45) and here we dropped the A_2 dependence as justified by references [14, 16].

Now to obtain $\Delta g/g$ we need two additional ingredients, the so-called Dokshitzer-Gribov-Lipatov-Altarelli-Parisi evolution equations (DGLAP) (reference [18]) and the elementary photon-gluon cross-sections (reference [9] equations (6.2)). The parton distributions q_i are determined from a global fit to a wide range of scattering data. The basic procedure is to parameterise the $q_i(x,Q^2)$ at a low value of $Q^2 = Q_0^2$ such that the $q_i(q,Q^2)$ can be calculated at higher Q^2 by using next to leading order (NLO) DGLAP.

[2] $f(\mathrm{NH_3}) = 0.176, \quad f(^6\mathrm{LiD}) = 0.5$

Now it is possible to get $\Delta g/g$ by comparing different parameterisations to the measured data. COMPASS expects to measure $\Delta g/g$ to an accuracy (reference [1]) of:

$$\delta\left(\frac{\Delta g}{g}\right) \approx 0.11 \tag{28}$$

A short and comprehensive explanation (tagging and p_t cut) on how to obtain this estimate is given in reference [10] page 8.

2.2.2 Measurement of Λ and $\bar{\Lambda}$ polarisation

Above it is shown that by measuring the asymmetry in the production of open charm by PGF it is possible to distinguish between the polarised strange quark [15] and gluon interpretation of the EMC spin effect. Complementary information on the polarisation of strange quarks and/or antiquarks can be obtained by also measuring the longitudinal polarisation of Λ and $\bar{\Lambda}$ baryons.

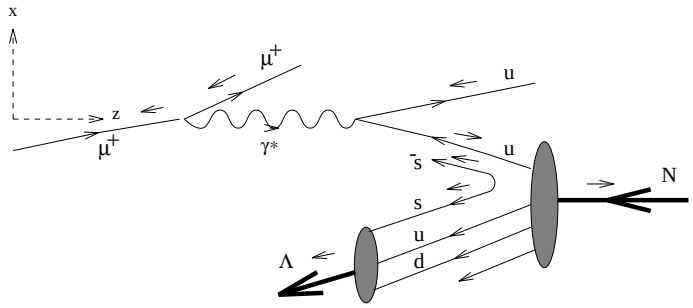

Figure 3: Λ polarisation according to the polarised $s\bar{s}$ sea model.

The basic idea of the longitudinal polarisation transfer from the polarised lepton to the unpolarised target fragments is that the polarised virtual boson will strike preferentially one quark polarisation state inside the target nucleon, and that the fragment left behind will contain some memory of the angular momentum removed. More specifically the mean polarisation of the remnant s or $\bar{s}$, and hence the Λ or $\bar{\Lambda}$, produced in the target fragmentation region , should be net negative, as diagrammatically shown in figure 3. This is due to the preference of the negative polarised virtual photon to interact with a u quark with positive helicity, which on average is aligned with the proton. The effect is large and can be detected in the COMPASS experiment.

2.2.3 Longitudinal spin distribution functions

In the view of the different explanations for the violation of the Ellis-Jaffe sum rule and the existence of various models for the spin structure of the nucleon it appears logical and tempting to decompose the nucleon spin into the valence and the sea components and to determine the spin distribution functions of the different flavours for the valence and the sea quarks. This can be achieved by semi-inclusive measurements (see glossary: Semi-Inclusive Measurement of Interactions) of deep inelastic scattering of polarised leptons on polarised proton and deuteron targets.

The measurement and the identification of the final state hadrons in the full range of momentum allows a detailed study of the fraction of the nucleon spin carried by the valence and sea quarks, respectively. In the current fragmentation region , semi-inclusive cross-sections factorise into the z independent quark spin distribution functions and the x independent quark fragmentation functions . The asymmetries of the spin dependent

virtual photon cross-sections for muoproduction of π^+, π^-, K^+, K^- and K° are given by the ratios of the different linear combinations of quark distribution functions from which then flavour separate distribution functions can be derived as a function of x (see reference [1] appendix A).

2.2.4 Transverse spin distribution functions

As shown by Jaffe and Ji [19], the momentum distributions $q(x)$, the helicity distributions $\Delta q(x)$, and the transverse spin distributions $\Delta_T q(x)$ completely specify the quark state inside the nucleon at the twist-two level. The functions $\Delta_T q(x)$ have never been measured up to now.

In all existing estimates $\Delta_T q(x)$ is nonzero at least for u quarks and it is different from $\Delta q(x)$.

COMPASS proposes to measure $\Delta_T q(x)$ in semi-inclusive DIS at the leading twist, by analysing the polarisation of the 'struck' quark.

2.3 The hadron programme

The hadron programme will address three main issues,

1. study of hadronic structure with virtual photons using PRIMAKOFF reactions,
2. study of gluonic systems and
3. study of charmed hadrons.

2.3.1 Hadronic structure with virtual photons

Because of the progress in the description of nucleon structure through non-perturbative QCD there are now predictions for quantities like polarisabilities and cross-sections available, which need to be confirmed by experimental measurements. Currently such studies are almost solely addressed at low-energy electron accelerators. COMPASS proposes to use high-energy pion, kaon and hyperon beams to provide complementary measurements using the PRIMAKOFF *reaction* mechanism (which is COMPTON scattering with virtual photons in inverse kinematics).

Rigorous predictions were made using an effective field theory, with a QCD chiral Lagrangian, which unambiguously follows from the assumption of spontaneously broken chiral symmetry. A low-energy expansion of the effective Lagrangian establishes unambiguous relationships between different processes. Now we need experimental investigations to determine how well chiral perturbation theory (χPT) works. Thereby COMPASS tests fundamental predictions of QCD like pion and kaon polarisabilities, and chiral axial anomaly amplitudes.

The PRIMAKOFF reaction relates processes involving real photon interactions to production cross-sections involving the exchange of virtual photons. Research on gamma-hadron interactions now plays an important role in studies of hadron structure. First of all, progress in experimental techniques now makes these experiments feasible and secondly, hadron-photon interactions supply information on the distribution of quark configurations in hadronic matter, via the photon interactions with the electric charges of quark fields. In general, PRIMAKOFF experiments require high Z targets, low mass tracking detectors, good charged-particle momentum resolution and high-resolution (spatial and energy) electromagnetic calorimetry.

In detail COMPASS will study hadron-photon interactions via the PRIMAKOFF reaction in the areas of hadron polarisabilities, hybrid meson production and radiative transitions of pion to a low mass two-pion system for a measure of the chiral anomaly. Therefore the COMPASS experiment uses a 50 to 280 GeV (μ,π,K,p)-beam and a virtual photon target.

Pion Polarisabilities [30]: The electric ($\bar{\alpha}$) and magnetic ($\bar{\beta}$) pion and kaon polarisabilities characterise the deformation of these particles in an electromagnetic field, as for example during $\gamma\pi$ or γ-kaon COMPTON-scattering. The polarisabilities depend on the rigidity of the particles' internal structures. Pion (kaon) polarisabilities can be studied via pion and kaon PRIMAKOFF reactions such as $\pi^-\gamma^* \to \pi^{-\prime}\gamma$. In pion-photon PRIMAKOFF scattering, a high energy pion scatters from a virtual photon in the COULOMB field of the target nucleus. The pion polarisabilities are determined by their effect on the shape of the measured $\gamma\pi$ COMPTON scattering angular distribution. The χPT (Chiral Perturbation Theory) effective Lagrangian then predicts the pion electric and magnetic polarisabilities. Because the available experimental data for pion polarisability has large uncertainties and kaon polarisability measurements have not been carried out, new high precision pion and kaon polarisability measurements are necessary to provide important new tests for QCD chiral dynamics.

Hybrid meson production [30],[32]: To understand confinement, it is of major importance to establish the existence of hybrid mesons and to study their structure. The hybrid ($q\bar{q}g$) mesons contain explicit glue as opposed to hidden glue in conventional hadrons. Because of difficulties in the mathematical treatment (PWA[3]) of the scattering results it is extremely important to have extra information from different hybrid production mechanisms where the physics is different and the ambiguities may look different. COMPASS may study PRIMAKOFF and diffractive production of non-strange light-quark hybrid mesons in the 1.4 to 3.0 GeV mass region.

Radiative transitions [32]: Radiative decay widths of mesons and baryons are powerful tools for understanding the structure of elementary particles and for constructing dynamical theories of hadronic systems. The small value of branching ratios of radiative decays makes them difficult to measure directly, because of the large background from strong decays. Studying the inverse reaction provides a relatively clean method for the determination of the radiative widths. Very good tracking resolution is needed (and available in COMPASS) to measure initial and final state momenta, and to thus exhibit the PRIMAKOFF signal at small four momentum transfer, where the electromagnetic processes dominate over the strong interaction. COMPASS will study radiative transitions of incident mesons to higher excited states. This will lead to new data for radiative transitions leading from the pion to the ρ, $a_1(1260)$, and $a_2(1320)$; and for the kaon to K^* and higher resonances.

Chiral Anomaly [33]: Another interesting radiative transition involves the chiral anomaly term of the effective chiral Lagrangian. The chiral anomaly component of the effective chiral Lagrangian predicts the $F_{3\pi}$ transition amplitude for the $\gamma \to 3\pi$ process. This amplitude was already measured, but only with low statistics and the result was not consistent with the $O(p^4)$ expectation. Therefore more precise measurements are needed, which COMPASS can provide.

[3]Partial Wave Analysis

		J^{PC}	$I=1$	$I=1$ $(n\bar{n})$	$I=1$ $(s\bar{s})$	Strange
L=0	S=0	0^{-+}	π	η	η'	K
	S=1	1^{--}	ρ	ω	ϕ	K^*
L=1	S=0	1^{+-}	b_1	h	h'	K_1
	S=1	0^{++}	a_0	f_0	f_0'	K_0
		1^{++}	a_1	f_1	f_1'	K_1
		2^{++}	a_2	f_2	f_2'	K_2^*
L=2	S=0	2^{-+}	π_2	η_2	η_2'	K_2
	S=1	1^{--}	ρ	ω	ϕ	K_1^*
		2^{--}	ρ_2	ω_2	ϕ_2	K_2
		3^{--}	ρ_3	ω_3	ϕ_3	K_3^*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: The quantum numbers and names of conventional $q\bar{q}$ mesons [34].

2.3.2 Study of gluonic systems

The following section is mainly an excerpt of reference [34] put into the context of the COMPASS experiment.

Because of its non-abelian character one fundamental prediction of QCD is the existence of states containing valence gluons. Due to confinement, only colour singlet objects can exist as physical hadrons. Coloured quarks form the fundamental triplet 3 representation of the SU(3) colour gauge group and antiquarks the conjugate anti-triplet $\bar{3}$ representation. In the constituent quark model, combining the spin and orbital angular momentum wave functions with the quark flavour wave functions, results in the meson states of table 1. States not fitting into this picture are considered "exotic". Thus, a meson with $J^{PC} = 1^{-+}$ would be forbidden in the constituent quark model as would be a doubly charged meson. However colour singlets can also be constructed with gluons g. Glueballs are hadrons with no valence quark content and hybrids are made up of valence quarks, antiquarks, and an explicit gluon degree of freedom.

The lightest glueball is found to be a 0^{++} state with a mass around 1500 MeV. Because the lowest glueballs have conventional quantum numbers with masses situated in dense background of conventional $q\bar{q}$ states, it is difficult to distinguish them from conventional mesons. The significant property of glueball decays is that one expects them to have flavour-symmetric couplings to final state hadrons. Measurement of electromagnetic couplings to glueball candidates would be extremely useful, because the radiative transition rates of a relatively pure glueball would be anomalous relative to the expectations for a conventional $q\bar{q}$ state and similarly, a glueball should have suppressed couplings to $\gamma\gamma$.

Additionally the simple picture above is complicated by mixing effects between the pure glueball and $q\bar{q}$ states with the same J^{PC} quantum numbers. With this motivation Lee and Weingarten [35, 36] approximated, that the $f_0(1710)$ glueball is $\sim 74\%$ glueball and the $f_0(1500)$ glueball is $\sim 98\%$ quarkonium, mainly $s\bar{s}$. Mixings can significantly alter the properties of the underlying states, which makes the interpretation of observed states difficult, and often controversial.

Given this discussion, the conventional wisdom is that it would be more fruitful to search for low mass hybrid mesons with exotic quantum numbers than to search for glueballs. Hybrids have the additional attraction that, unlike glueballs, they span complete flavour nonets and hence provide many possibilities for experimental detection. In addition, the lightest hybrid multiplet includes at least one J^{PC} exotic.

In the search for hybrids, there are two ways of distinguishing them from conventional

states. One approach is to look for an excess of observed states over the number predicted by the quark model. The drawback to this method is that it depends on a good understanding of hadron spectroscopy in a mass region that resisted an unambiguous understanding. Especially here COMPASS tries to improve the experimental data by also doing light meson spectroscopy to produce a proper classification of the scalar and pseudo scalar mesons to aid theory in finding an applicable model. The situation is further complicated by expected mixing between conventional $q\bar{q}$ states and hybrids with the same J^{PC} quantum numbers. The other approach is to search for quantum numbers, which cannot be accommodated in the quark model.

2.3.3 Studies of charmed hadrons

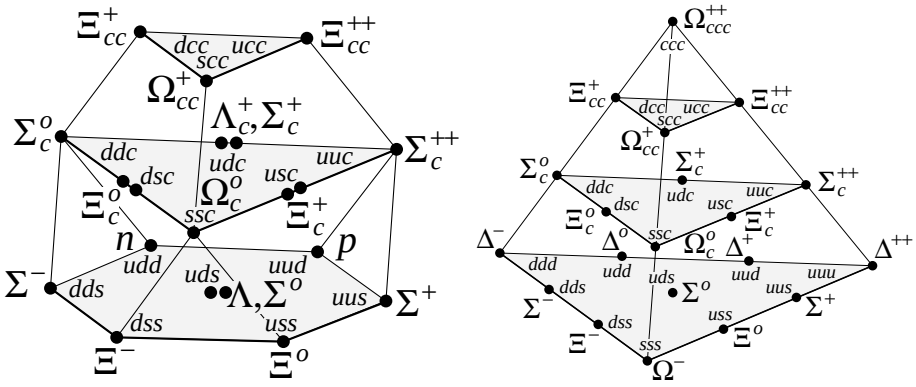

Figure 4: SU(4) representation of the ground state baryons.

Because up to date not even all $1/2^+$ ground states of charmed baryons have been observed and nothing is known about doubly charmed baryons, COMPASS tries to clarify the situation in the charmed hadrons sector.

The semi-leptonic decay widths provide the best test for our understanding of charmed baryon decays since precise theoretical predictions on rates and form factors are available. In particular, it is of interest to compare the q^2 dependence of form factors to calculations in the HQET (Heavy Quark Effective Theory) framework [38]. Although the charm quark mass is at the lower limit of applicability of the theory, corrections are believed to be of the order of only 20%.

Besides that the theoretical semi-leptonic decay width $\Gamma_{\rm sl}^{\rm th}$ of charmed baryons together with the experimental ratio of semi-leptonic to hadronic branching fractions $B_{\rm had}/B_{\rm sl}$ and the experimental measured lifetime τ allow a determination of hadronic partial widths $\Gamma_{\rm had} = \Gamma_{\rm sl}^{\rm th} B_{\rm had}/B_{\rm sl}$ and absolute hadronic branching ratios $B_{\rm had} = \Gamma_{\rm had}\tau$ for comparison with theory predictions. The absolute branching ratios are also needed in order to interpret beauty hadron decays.

Charmed hadron lifetimes are ideal for testing the understanding of the effects of the hadronic environment on the decay of the naked charm quark. These effects are about a factor 10 larger for charmed hadrons than for beauty hadrons but significantly smaller than in strange hadrons. To aid theory in its understanding of hadronic effects, precise measurements of the lifetime of all charmed baryons to better than 5% are required. The main difficulty lies in the short lifetimes of the charmed-strange baryons and their small production rate.

The production of doubly charmed quarks has not yet been addressed by any running experiment and it is, due to the very low cross-section and small branching ratios, an experimental challenge. The structure of doubly charmed baryons probably resembles

		muon programme		hadron programme	
top beam momentum		190	GeV/c	280	GeV/c
particles/spill		$2.0 \cdot 10^8$		10^8	
beam polarisation	P_B	0.80		0	
beam diameter	σ_x, σ_y	0.8	cm	0.3	cm

Table 2: Properties of the beam in the muon and hadron programme (see references [1, 44, 45]).

a heavy meson ("similar" to the H_2^+ molecule) where a light quark surrounds a heavy cc-diquark. Besides the spectroscopical interest, their lifetime offers insights into decay dynamics.

The process of hadroproduction of charmed particles can be subdivided into two different sub processes, the production of charmed quarks and the process of their hadronisation into charmed particles. The former defines the full cross-section of charm while the latter is responsible for the relative production of different types of charmed particles and their kinematic distributions. As the mass of the c-quark is not large enough we cannot limit ourselves to low order diagrams in perturbative QCD to calculate the full cross-section of charm. In addition, as the process of hadronisation is soft, the hadronisation is beyond the scope of perturbative QCD. Due to the very high beam rate the experiment proposed is capable of a systematic study of the charm production in a range of c.m. energies from 14 to 25 GeV.

D and D_s-mesons can be used to study rare processes like leptonic decays ($D \to \mu\nu$), which are the key to the determination of the charmed meson decay constants. For D-mesons the measurement of the leptonic decay is more difficult than for D_s-mesons, because the leptonic decay of the D mesons is CABIBBO suppressed and only appears with a branching ratio of about 10^{-4}. Other studies with D mesons include semi-leptonic decays, spectroscopy of orbitally and radially excited states and the search for rare or forbidden D-decays.

There will also be some investigations on charm exotics like singly charmed pentaquark and doubly charmed tetraquark systems.

2.4 The beam

The muon programme needs an incoming beam of 2×10^8 μ^+ per spill (see glossary: Spill) with an energy in the range of 90 to 200 GeV and a high longitudinal polarisation. The hadron programme uses unpolarised π, K, p and Σ beams with momenta of up to 300 GeV/c and preferably more. In table 2 you can find a summary of beam properties for both programmes.

All beams are generated in the M2 beam line using the T6 production target (see reference [46]). The target is made of beryllium. Typical target lengths are 200 to 500 mm (the interaction length (see glossary: Interaction Length) of beryllium is some 400 mm). A 400 GeV/c primary proton beam is extracted from the SPS (Super Proton Synchrotron) towards the north experimental area. A fraction of this beam, selected by two stages of septum magnets, is directed towards the primary target T6. The proton intensity incident on this target may be in the range between $2 \cdot 10^{12}$ and $1.2 \cdot 10^{13}$ protons per SPS cycle, where the upper limit is given by radiation protection limits. From the T6 target a secondary beam (positive or negative) at zero production angle is derived. Either this beam is transported directly to the experiment (in the case of the hadron beam), or tertiary muons or electrons are selected. A schematic layout of the M2 muon beam is shown in figure 5. A more detailed description of the M2 beam layout is available in the

form of a Beatch listing, indicating the exact positions of all magnets, collimators and detectors at `http://cern.ch/eagroup/beatch/m2yr2001.txt`. The hadron beams are a

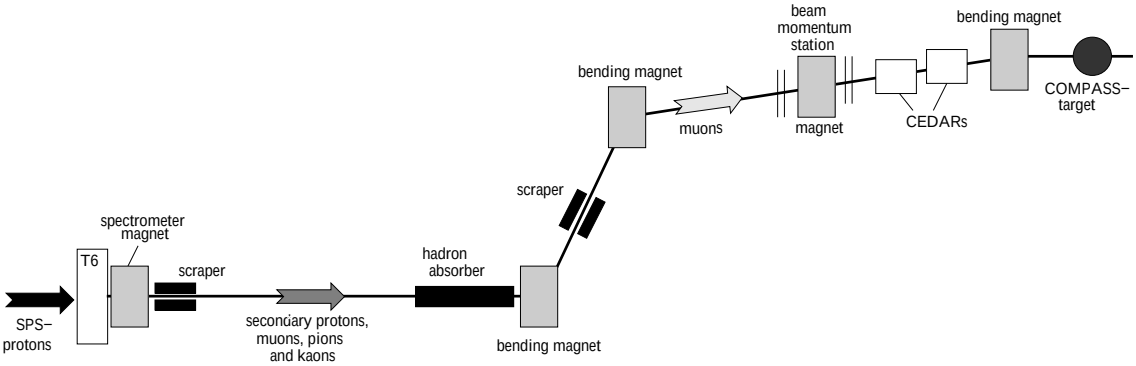

Figure 5: The M2 beam layout [12].

direct product of the incoming protons colliding with the beryllium atoms and producing secondary particles (mainly protons, pions and kaons). After the production target there is a Spectrometer Magnet, which selects particles with a defined momentum.

The following procedure depends on the type of particles needed for the experiment.

- Hadron beam: In this mode of operation the hadron absorbers are moved out of the beam and the secondary hadrons are transported directly from the primary target to the experiment. The maximum allowed fluxes are 10^8 hadrons per SPS cycle, limited by radioprotection guidelines. Typical spot sizes are of the order of 3 to 5 mm RMS (Root Mean Square). The beam composition as a function of beam momentum can be calculated with the ATHERTON *formula* [48].

- Muon beam: The M2 has historically been designed and operated as a muon beam. A large acceptance (see glossary: Acceptance), relatively wide-band ($\pm 10\% \Delta p/p$) pion beam, as well as the muons originating from pion decay, are transported through a 600 m long decay channel (FODO (Focusing-Defocusing quadrupoles) lattice). At the end of this channel, the muons are focused on a beryllium absorber (up to 9 units of 1.1 metres of beryllium each), which stops all the hadrons in the beam. The muons are picked up and transported through a second FODO channel. The muon momentum definition and cleaning is done in this section, too. At the end of this FODO array the beam is shaped in terms of spot size and divergence for the experiment. The maximum allowed flux is $2 \cdot 10^8$ muons per SPS cycle. Typical spot sizes at the target are 8 mm RMS, with a divergence of 0.5 mrad RMS in the horizontal plane and less than 1 mrad RMS in the vertical plane.

The muons are generated through pion decay (see figure 6). As the muon programme needs longitudinal polarised muons one uses the property of maximum parity violation [49] of the pion decay $\pi^+ \rightarrow \mu^+ + \nu_\mu$ to produce a mean muon polarisation of $P_B \sim 0.8$. Because neutrinos are only realised in nature as left handed objects, and because for massless objects left-handed chirality is directly coupled with negative helicity we know that, in the c.m. system of the reaction, the neutrino spin is oriented opposite to its velocity vector. And because of momentum and angular momentum conservation we also know that we have, in the c.m. system of the reaction, the muon with its spin vector oriented opposite to its velocity vector. Therefore now both particles have helicity $-\frac{1}{2}$ (that is the spin vector of the spin 1/2-particle points opposite to the direction of motion, see figure 6). If we start now with an incoming pion beam of known velocity we can at some later point in the beamline (after the decay) determine the polarisation of the muons

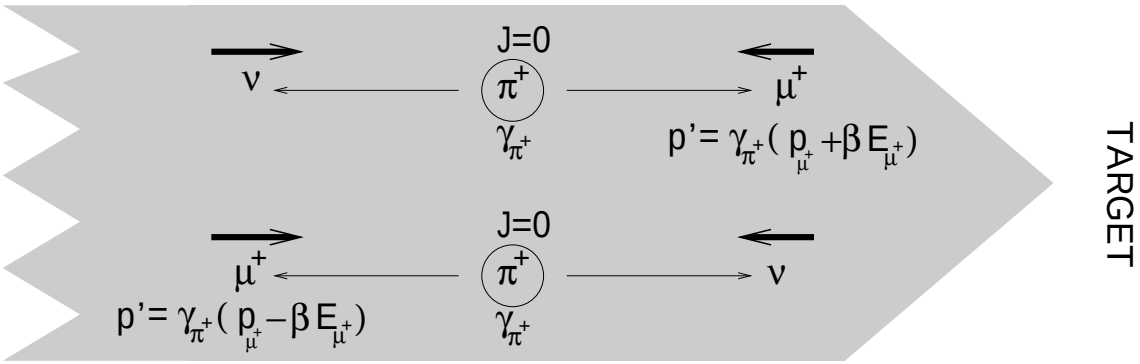

Figure 6: Pion decay [12].

by a second velocity measurement. The muons which decayed in the same direction as the beamline will be faster than the original incoming pion beam and they will have their spin opposite to the beamline and the muons which decayed in the opposite direction will be slower than the original pion beam and have their spin in the same direction as the beamline. All muons with a velocity vector not parallel to the beamline will be filtered out. But as we always have some divergence in the beam, we also have some uncertainty in the direction of polarisation of the pions even after the second measurement of velocities. Quantum mechanically, the spin direction corresponds to a two state system where we have to rotate the quantisation axis of the muons not pointing directly along the beamline into the beamline and recalculate the probability amplitudes. Therefore we will get some chance that the polarisation is along the beam line or opposite to it. Another point, which disturbs our simple reasoning above is the magnetic fields of the spectrometers, in which we get precession of the spin vector around the direction of the magnetic field lines.
For the hadron programme, we still need to identify the incoming particles. But as they all have the same momentum as selected by the SM (Spectrometer Magnet) after the production target it is sufficient to measure the velocity of the particles with a Čerenkov-Detector, a so-called CEDAR[4], to identify them. The CEDAR is a differential counter Čerenkov-Detector (see *Differential Čerenkov Counter* in section 2.5.4 on page 30) tuned to a certain Čerenkov angle. It can therefore identify exactly one type of particles. If you have two such devices you can identify two sorts of particles. It should also be pointed out that the CEDAR counters only work well up to hadron beam intensities of about 10^7. For COMPASS the CEDARs may be used for K tagging or rejection, where the expected K intensities are of the magnitude of 10^7.

2.5 The experimental apparatus

Previously the physics motivation for the COMPASS experiment was explained. To perform these measurements COMPASS proposes a new spectrometer with excellent particle identification and calorimetry, capable of standing beam intensities of up to $2 \cdot 10^8$ particles per spill and energies from 90 to 200 GeV.

Several parts of the physics programme need high intensities to produce the required statistics.

- The muon programme needs high rates, because the measured asymmetries in the polarised DIS are very small, and in addition the values of the asymmetries are distributed over different Bjorken x and Q^2.

[4] Čerenkov Differential counter with Achromatic Ring focus (see page 30 and reference [50])

- In the study of semi-leptonic charmed baryon decays the measurement of decay asymmetries can be used to compare measurement with predictions of the baryon form factor ratios from HQET. Up to now such comparisons of form factor ratios were done as average over q^2 because of small statistics.
- To measure the lifetimes of charmed-strange baryons to better than 5%, where the main difficulty lies in their short lifetimes and in their small production rates.
- The production of doubly charmed quarks is due to the very low cross-section and small branching ratios an experimental challenge.
- To be able to study the non-leptonic decays of D_s or even D mesons the high rates are vital.
- Due to the very high beam rates, COMPASS is capable of systematic studies of the charm production in a range of c.m. energies from 14 to 25 GeV.

The COMPASS experiment comprises many measurements of rather different natures, which require their own optimised set up. Nevertheless, the global structure of the different experiments bears many similarities. They use in common a modern, high rate forward spectrometer with two independent magnetic spectrometer stages, each equipped with tracking, particle identification, calorimetry and muon detection. A large fraction of the apparatus can be common to all experiments. This applies in particular for the section downstream from the first spectrometer magnet (SM). Due to the very different target set ups, the part upstream of the RICH (Ring Imaging Čerenkov Counter) has to be designed individually. This includes the use of different large angle SMs. Figure 7 shows the major experimental configuration as planned for the year 2002 run. For a more detailed detector map please have a look at [70].

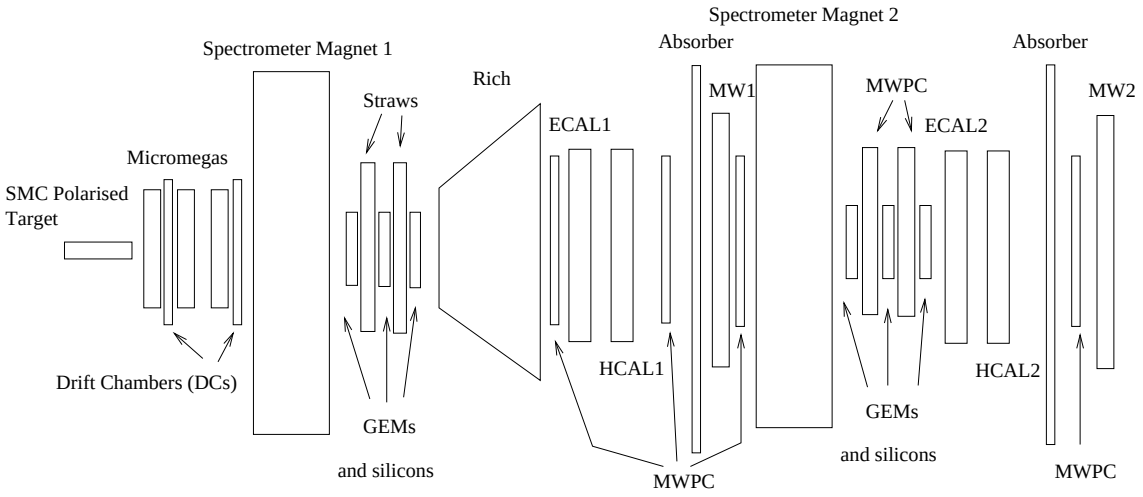

Figure 7: Schematic view of the COMPASS detector as foreseen for the year 2002 run.

The requirements for the various detectors are given by the maximum requirements for the different measurements. The most stringent ones are coming from the intensity of the muon beam (100 MHz) and the large interaction rate in the hadron beam (1 MHz). These large rates require particular care in the choice of detector materials as radiation damage in silicon detectors or calorimeter crystals can be severe. High demands on the speed of the detectors and the readout exist in particular in the hadron beam programme due to the requirements for a very fast and efficient trigger for charmed events. Therefore, fast front-end electronics, multi-buffering, and a large and fast storage of events is essential.

2.5.1 General set up

The detection of particles over a large acceptance range (see glossary: Acceptance) requires the use of a two-stage spectrometer. For the next year (2002) run the set up seen in figure 7 is planned and comprises the following detector stages (please refer to [70] for more detailed information):

- Polarised target,
- Micromesh Gaseous Chamber (Micromegas) and drift chambers,
- Spectrometer Magnet (SM) 1,
- Straws (small-diameter drift tubes), GEM, silicon micro-strips,
- RICH,
- Electromagnetic Calorimeter (ECAL) 1,
- Hadronic Calorimeter (HCAL) 1,
- Muon Wall (MW) 1,
- SM2,
- Multi-Wire Proportional Chamber (MWPC) with GEM and silicon detectors,
- ECAL2,
- HCAL2,
- Absorber and MWPC, MW2,
- Trigger hodoscope (see glossary: Hodoscope),
- Beam dump.

The first spectrometer part (LAS[5]) is responsible for large angles up to 200 mrad vertically and 250 mrad horizontally while the second spectrometer part is concerned with the detection of particles below 30 mrad. The COMPASS experiment has a so-called staggered tracking system, that is, for the different angles relative to the beam line, there are up to three different detector systems. For example between SM1 and RICH there are straws, GEM and silicon detectors (from large angles to small angles) with some overlap. With this set up one tries to keep the number of events per readout channel approximately constant. E.g. the silicon micro-strip detectors are directly in the beam and have a readout pitch of approximately 50 μm whereas the GEM detectors have a 400 μm pitch and are responsible for the area around the beam.

The next sections will describe the individual components of the COMPASS detector and the physics behind them. Because my work for the DCS (see glossary: Detector Control System) was concerned mainly with the GEM and silicon detectors the description of these detectors will be more detailed as other parts. Where the description of the physics would lead too far away from the description of the detectors you will find a reference to the glossary, where more details can be found. Most information in the next sections (and in the glossary, where the glossary explains physics) is derived out of references [1] and [51] to [62]. Before we start to investigate the different detector types in detail a general property of all COMPASS detectors should be mentioned. They all have a central deactivated area, where the beam will pass through, to prevent detector damage by the high beam rate. Only the silicon micro-strips don not have this feature, because they will be placed directly in the beam.

[5]Large Angle Spectrometer

2.5.2 Targets

Because COMPASS will start with the muon programme the muon target will be explained in more detail.

For the muon programme, it was planned to use the existing polarised target from the SMC collaboration [28], with a new and much larger super conducting solenoid/dipole magnet to allow for a sufficiently large acceptance (see glossary: Acceptance) (180 mrad in contrast to 65 mrad) of hadrons. But due to delivery problems of the magnet, COMPASS will start to run with the old SMC magnet until the new one arrives. The polarisation is done via DNP (Dynamic Nuclear Polarisation) in a homogeneous magnetic field of 2.5 T and at a temperature below 0.5 K. In the target material are little quantities of paramagnetic atoms, which react on irradiation of microwaves, slightly above or below the LARMOR frequency of the paramagnetic centres. As soon as the maximum polarisation is reached, the microwaves are turned off and the material goes over into "frozen spin" mode at a temperature below 50 mK. In this mode, the nuclear spin-lattice relaxation time is several hundred hours [1] if a holding field greater than 0.5 T is present.

The SMC target is divided into two cylindrical cells, 60 cm in length and 3 cm in diameter, which are polarised in opposite directions. This will help in eliminating systematic errors. By frequent reversals of the spin orientations, the systematic error from acceptance variations of the spectrometer with time can be greatly reduced. As target material, it is foreseen to use ^{6}LiD for measurements on deuteron and NH_3 for measurements on protons. With these materials, it is possible to achieve a level of polarisation as high as 50% for ^{6}LiD and 85% for NH_3.

For the hadron programme, there will be different targets for the different physics parts:

- a lead target for the studies using the PRIMAKOFF reactions,
- a liquid hydrogen target for the light meson spectroscopy and
- a target made up of slices of copper and silicon detectors for the studies of charmed hadrons.

The idea behind the sliced target is to be able to reconstruct the interaction point of the particles and therefore also being able to identify kinks in the particles trajectories which indicate a decay with an "invisible" neutrino.

2.5.3 Tracking detectors

Because the change of the direction of a particle in a magnetic field is a function of its momentum the tracking devices in conjunction with a Spectrometer Magnet (SM) serve for determining the particle momenta. Therefore one part of the tracking is clearly a SM to produce the magnetic field. As the deflection angle in a magnetic field depends on the ratio between the magnetic field strength and the particle's momentum there are two spectrometer magnets in COMPASS. The first magnet has a lower mangnetic field than the second one and is used to split up the tracks of particles in the lower momentum range of the experimental interest. High momentum particles pass the first spectrometer magnet nearly undeflected and the second magnet with a higher magnetic field is used to split up their tracks. In the following paragraphs we will have a closer look to the different tracking detectors from Large Area Tracking (LAT) devices towards Small Area Tracking (SAT) devices.

The straw drift tubes: Drift tubes are built as a stand-alone coaxial cylindrical drift chamber, made of a conducting-surface cylinder acting as cathode, and a sense wire stretched in the axis of the cylinder. Often, tubes are made of thin metalised foils, and arranged into densely packed layers or volumes. These can be used when short drift times are at a premium, like in high-rate environments. For a tube diameter of 4 mm, the maximum drift time (at the usual drift velocity) is 40 ns. Such small-diameter tubes are also called straws, and a collection of them a straw chamber.

The Multi-Wire Proportional Chamber (MWPC) detectors: A multi-wire chamber is a detector for charged particles, which essentially consists of thin parallel and equally spaced anode wires, symmetrically sandwiched between two cathode planes. The cathodes are at a negative voltage relative to the grounded anode wires. This creates a homogeneous electric field in most regions, with all field lines leading from the cathode to the anode wires. Around the anode wires, the field increases rapidly. If a particle passes through the detector it ionises the gas in the chamber, and the liberated electrons follow the electric field lines towards the anode wires. The strong field very close to the wire acts as a multiplication region. The energy of the electrons increases, and in turn, they ionise the gas, causing an avalanche of electrons to reach the anode wire.

The pulses are read from the anode wire. If the chamber is used in proportional mode (see glossary: Operational Modes of Gaseous Detectors), the pulse height is a measure of the energy loss of the particle in the gas. This can be used for particle or momentum identification. Simple multi-wire chambers are used as tracking chambers, with the anode wires only giving one bit of information for a passing particle. Multiple planes with different angles of inclination for the wires will then allow reconstruction of trajectories in space.

The Micromegas: The Micromesh Gaseous Chamber (Micromegas) is a parallel-plate gas detector that exploits the charge multiplication in uniform fields. The detector consists of a thin metal grid stretched at a very small distance, 50 to 100 μm, above a readout electrode. With a very high field applied across the gap, typically above 30 kV/ cm, electrons created by ionising particles in the upper drift region are collected and multiplied. Thanks to the small gap and high field, positive ions move very quickly, and most are collected by the cathode mesh. This prevents space-charge (see glossary: Space Charge) accumulation and induces very fast signals with only a small ion tail, 50 to 100 ns wide. The Micromegas can operate at very high particle fluxes.

Experimental data and theoretical considerations indicate that the maximum proportional gain in parallel-plate chambers is limited by the total amount of charge in the avalanche, around 10^7 to 10^8. Above this value, called the Raether *limit*, transition to a streamer occurs (see glossary: Operational Modes of Gaseous Detectors), followed by breakdown.

The COMPASS Micromegas were optimised for time resolution and are therefore used in front of the first spectrometer magnet, where the particle intensities are very high, instead of GEM detectors. Because the GEMs are near to the beam, one of their major design objectives for COMPASS was to use as little material as possible and to give rise to as little interactions with the beam as possible.

The Gas Electron Multiplier (GEM) detectors: The Gas Electron Multiplier, introduced by Sauli in 1996 [64], consists of one thin, metal-clad polymer foil (typical $\sim 50\,\mu$m Kapton$^{\text{TM}}$ plus 5-15 μm copper layers on both sides) chemically perforated by a high density of holes, typically 100/ mm^2. If one applies a potential difference between

the metal electrodes on both sides of the foil, it acts as an electron multiplier, which can be placed between the anode and the cathode of a parallel plate gas detector. On its way

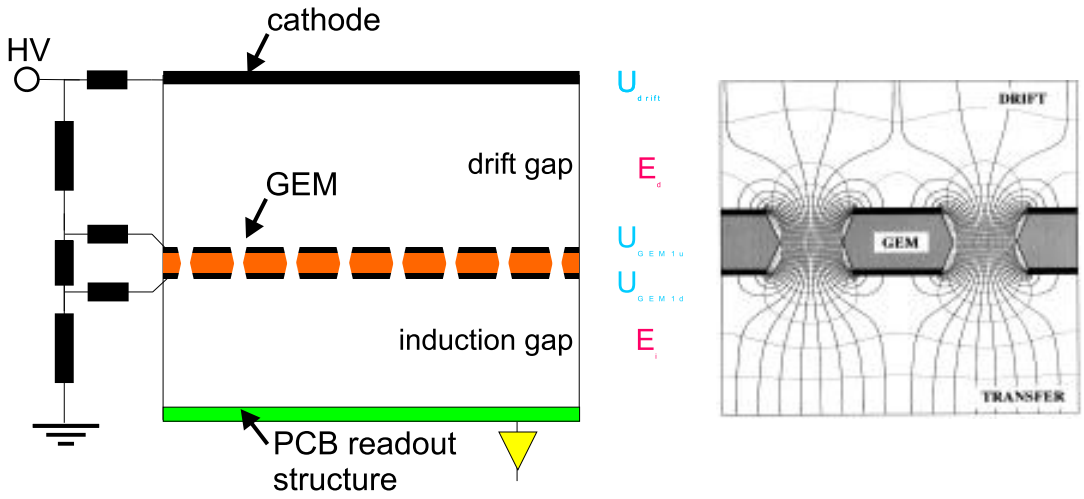

Figure 8: Set up of the GEM chamber on the left and the field line structure on the right [54].

through the gas volume, an ionising particle releases primary electrons, which loose their kinetic energy by liberating a number of secondary electrons within a very short range. Diffusion causes the development of a GAUSSIAN-shaped charge cloud that drifts in the homogeneous electric field towards the GEM. About 100 μm above the GEM, the parallel electric field lines are bunched into the GEM-holes (see figure 8 on the right), where gas amplification in the high electric field causes proportional charge multiplication. At a suitable choice of voltages the field lines below the GEM spread again and the charge cloud drifts towards the readout electrode where it is collected. For a more detailed discussion of the physical processes in the GEM detector, please refer to [54].

The GEM foil acts as a charge preamplifier, preserving the original ionisation pattern to a large extent. The first applications of the technology have been accomplished by combining the GEM amplifier with a standard MSGC (Micro-strip Gas Counter).

The gain of the GEM electrode depends on the thickness of the polymeric support, the diameter of the holes, the gas mixture (see glossary: Gas Mixtures in Gaseous Detectors), and the applied voltages. It is possible to achieve proportional gains up to 10^4, suitable for direct detection of ionisation on simple charge-collecting strip PCB[6]s (see figure 8 on the left). In this mode of operation, the signal detection on the strips is entirely due to the electrons collection, without a slow ion tail, and is typically a few tens of nanoseconds wide for a 1 mm wide conversion gap. With this simple readout structure spatial resolutions better than 40 μm can be reached. Important for its potential use as a position sensitive detector in a modern HEP (High Energy Physics) experiment is its efficiency. The efficiency in a gas detector is the ratio of the number of detected particles to the total number of ionising particles traversing the gas volume. As soon as the efficiency is $\approx 100\%$ no more gain is needed. Generally it is better to operate at lower voltages as this reduces the possibility of discharges and detector damages.

In the COMPASS experiment a modified version of the above described "simple" GEM detector, optimised for discharge prevention, is used. COMPASS uses triple GEM detectors with strip PCB readout (see figure 9). A triple GEM detector uses three GEM foils instead of a single one. The gain is divided over all three GEM foils to reduce the

[6]Printed Circuit Board

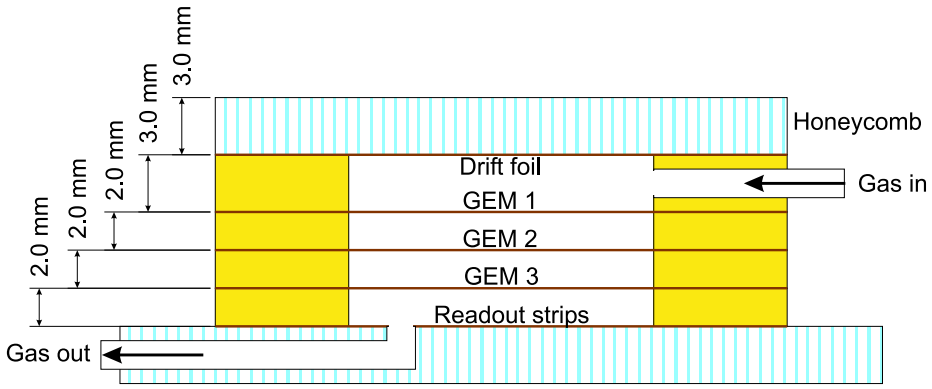

Figure 9: Cross section through a triple GEM detector [52].

field strengths between the two electrodes of each GEM foil. Furthermore the COMPASS GEM detectors do not just equally distribute the gain over the three foils, but they operate the "top" foil (the foil with the longest distance from the PCB readout) at about 10% higher voltage than the central foil and the "bottom" foil (the one which is closest to the readout) at about 10% lower voltage than the central foil. This technique is called asymmetric sharing of gain. Another improvement over the original design is the use of sectorised foils. By sectorising the foils one decreases the capacitatively stored energy in a GEM foil and in case of a discharge the amount of charge released in the discharge is smaller and the probability of propagating discharges is lower. Generally the triple GEM design, the use of sectorised foils and the asymmetric sharing of gain serve for the discharge prevention. A discharge in the gas volume between the bottom foil and the PCB board could destroy the highly sensitive readout electronics and damage the detector considerably.

The COMPASS GEM detectors use a two dimensional read out with 2×768 readout strips and a $400\,\mu$m pitch. By dividing the readout of the charge cloud between several readout strips, it is possible to calculate the centre-of-charge and to gain a resolution better than the width of a single readout strip. With this set up they can get up to a resolution of $54\,\mu$m.

The signals on both co-ordinates generated by a single physical event (Gaussian-shaped charge cloud) are because of the nature of their generation correlated in time and in signal amplitude. Therefore a GEM detector is able to give, besides the pure x and y information, also a probability of which x belongs to which y in the case where there are several events at the same time.

The GEM detectors for COMPASS use a Ar-CO_2 mixture in a relation of 70 : 30. Even if other gas mixtures (most notably organic mixtures) had a better behaviour in the sense of charge movement speed and build up of space charge (see glossary: Space Charge), for COMPASS it was decided to use Ar-CO_2 to prevent ageing effects (see glossary: Ageing).

The GEM detectors are grouped into stations, where each station consists of two GEM detectors. The two detectors are mounted back to back and rotated by an angle of 45°. One detector in a station measures the x/y co-ordinate and the other detector measures the u/v co-ordinate. In total there are 10 such stations in COMPASS.

The silicon micro-strip detectors: Semiconductor detectors have been used in high-energy physics applications in the form of pixel detectors, microstrip detectors and pads. These detectors are mainly made out of silicon, but GaAs or diamond is perhaps a future alternative to silicon. Recent progress in micro technology allows reliable large-scale production of detectors of sophisticated designs, at acceptable cost. Besides that one has the

advantage that one can build on a commercially available robust technology.

In the following explanation of the inner workings of a semiconductor detector, it is assumed that the basics of p-n junctions (diodes) are known. When an ionising particle penetrates the detector it produces electron-hole pairs along its track, where the number of pairs is a measure of the energy loss. An externally applied electric field separates the pairs before they recombine. Electrons drift towards the anode, holes to the cathode. The charge is collected by the electrodes. The drifting[7]/collected charge produces a current pulse on the electrode, whose integral equals the total charge generated by the incident particle, i.e. is a measure of the deposited energy. Silicon detectors are asymmetric p-n junctions. To work as a detector, the p^+n diode is reverse-biased by applying a positive voltage on the rear ohmic contact. At full depletion (the depletion zone spans the whole detector thickness), the electric field is a maximum in the junction and decreases to zero at the ohmic contact. In order to avoid losses in charge collection, the silicon detectors are over biased (below break-down voltage). The important concept here is the charge collection efficiency (see glossary: Charge Collection Efficiency).

Higher depletion voltages result in broader depletion zones and therefore lead to better signal to noise ratios (because of the lack of charge-recombination or thermal excitation in the depletion zone). As soon as full depletion is achieved a further increase of the voltage does not improve the behaviour of the detector any more. It is important to have a high ohmic resistivity to reduce the voltage of full depletion and to reduce the leakage currents (see glossary: Leakage Current), because as the detector material deteriorates due to radiation damage the material gets increasingly conductive and one has to increase the bias voltage to guarantee 100% depletion and to ensure the detectors correct operation.

The intrinsic energy resolution is related to the low-energy threshold: only 3.6 eV are necessary to produce an electron-hole pair, a low value compared to the ionisation energy in a gas (30 eV) or the approximately 300 eV necessary to generate an electron from a photocathode coupled to a plastic scintillator. The good spatial resolution comes from the high density of Si, from the little thickness and from the fact that there is no internal amplification like in gas detectors (no amplification effects corresponds to no additional statistical effects). On the other hand, the average energy loss in Si is high, about 390 eV/ μm, for a $\langle 111 \rangle$ oriented single crystal, and corresponds to about 110 e-h pairs. To limit the multiple Coulomb scattering, the detector thickness must be kept thin – the usual compromise thickness is 300 μm for optimum detection.

The silicon micro-strip detectors in COMPASS (see figure 10) use the same silicon design as the double sided micro-strip detectors for the high radiation environment in the HERA-B[8] experiment. The readout electronics was designed independently and is the same as for the GEM detectors. The silicon micro-strip detectors in COMPASS have

- a spatial resolution of 14 μm,
- a high ohmic resistivity to allow for bias voltages less than 100 V, and
- a time resolution of the order of nano seconds.

The double-sided readout means, that the detectors collect electrons on one side and holes on the other side and use two orthogonal readout structures to get two projections at once. In HERA-B one requirement for the silicon detectors was stand-alone track finding. Therefore, three or more independent projections were required. A stereo angle of 5° was chosen to realise four independent projections. The strips on both sides of a counter are orthogonal with respect to each other and tilted by 2.5° with respect to the detector edge.

[7] via influence

[8] An experiment to study CP violation in the B system using an internal target at the HERA proton ring [8]

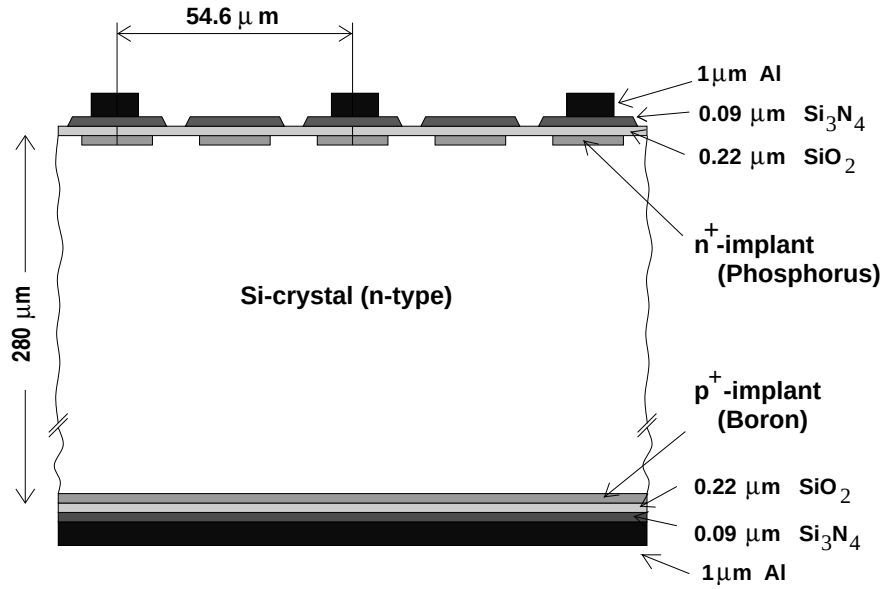

Figure 10: Schematic view of the silicon design used in HERA-B [57] (the same as in COMPASS).

By mounting two of these detectors back to back, one can obtain four views with 5° stereo angle as shown in figure 11.

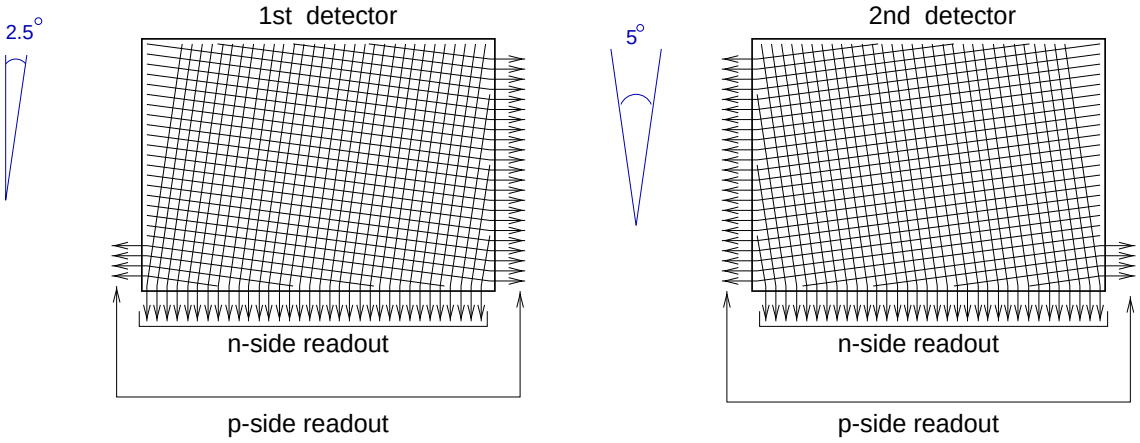

Figure 11: Orientation of the readout strips on the double sided detector. The right picture is a mirror image of the left one, representing the identical type of detector flipped around its shorter edge. Overlaying the two counters (in the same way as they are mounted, back to back) the 5° stereo angle is obtained [57].

One other important point for the COMPASS experiment is the radiation hardness (see glossary: Radiation Damage and Radiation Hardness). As the silicon detectors will be located in the direct beam region, the radiation damage can be severe. The main macroscopic effects of changes in the detector performance consist of a flux proportional increase in the leakage current, a dramatic change of the depletion voltage (needed to maintain the full sensitivity of the whole detector thickness) and the damage-related decrease of the charge collection efficiency. The HERA-B design of the silicon has already an elaborate multi guard-ring structure [57] to be able to operate the detector up to bias voltages of 300 V to 500 V. The guard ring structure shields the sensitive area from surface and edge leakage currents, but also provides a controlled, gradual drop of the potential

from the detector rim towards the potential of the undepleted substrate.

In COMPASS it was decided to cool the detector to cryogenic temperature (130 K) and to use the Lazarus effect (see *Lazarus effect* on page 76) to handle the radiation damage. The cooling has the disadvantage of introducing additional material in the beam line which might interact with the beam or with particles one wants to observe.

2.5.4 Particle identification

As described above the tracking detectors in conjunction with the spectrometer magnets provide the information to calculate the momentum of a particle. To identify the particle type one needs further information like the speed of the particle. Then it is possible to calculate the particle's mass and to identify the particle with the help of a mass table.

The RICH detector: ČERENKOV counters are generally detectors for charged particles using the light emitted by ČERENKOV radiation (see glossary: Čerenkov Radiation) to measure the particle's velocity $\beta = v/c$. Combined with the knowledge of the particle's momentum, β determines its mass. The index of refraction n of the ČERENKOV counters is carefully optimised for the particle masses and momentum range of the experiment in question.

Classification of ČERENKOV counters:

- *Threshold counters* record all light produced, thus providing a signal whenever β is above the threshold $\beta_t = 1/n$, at which point ČERENKOV radiation starts to be produced.
- *Differential counters* (like the CEDAR) accept light only in a narrow range of angles ($\delta \pm \Delta\delta$) i.e. in a narrow velocity interval. Resolutions of $\Delta\beta/\beta = 10^{-5}$ have been reached. As chromatic dispersion ($n = n(\delta)$) is the major source of error at high momenta, special achromatic counters, called DISC (Directional Isochronous Self Collimating) counters have been developed, which reach $\Delta\beta/\beta = 10^{-6}$ to 10^{-7}. Differential ČERENKOV counters suffer from the low acceptance both in angle and β.
- *Ring imaging* ČERENKOV *counters* (RICH): In the RICH particles pass through a radiator. The radiated photons are focused by a mirror onto a position-sensitive photon detector. The ČERENKOV radiation emitted at angle δ is focused onto a ring of radius r at the detector surface, and β can be determined by a measurement of r. For photon detection one uses thin photosensitive proportional chambers or drift chambers.

To achieve pions separation above three standard deviation level from kaons and protons between 3 and 120 GeV/c, COMPASS would like to uses two RICH detectors. Up to now only RICH1 is available and RICH2 is only a plan for the future. A schematic view of the available RICH detector can be seen in figure 12. The RICH is designed to cover the entire acceptance of the first spectrometer, providing hadron identification in the momentum range between 3 and 65 GeV/c. It uses C_4F_{10} as radiator gas and a MWPC with CsJ photo cathode and a pad readout to detect the photons.

2.5.5 Calorimeters (ECAL, HCAL)

Up to now we saw how to get the momentum and particle type with the tracking and particle identification detectors, but we still miss the energy of the particles. Therefore the calorimeters are used.

A calorimeter is a composite detector using total absorption of particles to measure the energy and position of incident particles or jets. In the process of absorption showers

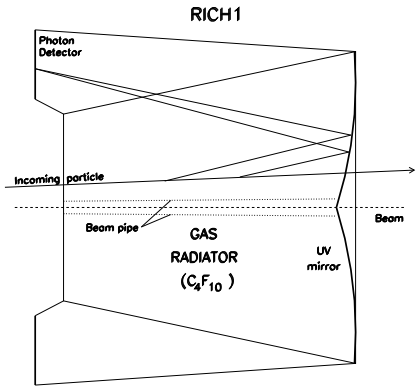

Figure 12: Schematic view of the RICH detector.

are generated by cascades of interactions, hence the name *shower counter* is occasionally used to describe a calorimeter. Characteristic interactions with matter (e.g. atomic excitation, ionisation) are used to generate a detectable effect, via particle charges. Each calorimeter is made of multiple cells, over whose volume the absorbed energy is integrated. Typically, incident electromagnetic particles, electrons and gammas, are fully absorbed in the Electromagnetic Calorimeter (ECAL), which comes before the HCAL.

Incident hadrons, on the other hand, may start their showering in the ECAL, but will nearly always be absorbed fully only in later layers, i.e. in the Hadronic Calorimeter (HCAL), built precisely for their containment.

The shower development is a statistical process. This explains why the relative accuracy of energy measurements in calorimeters improves with increasing energy, according to the empirical formula

$$\sigma_E/E \approx a/\sqrt{E} + \beta$$

where E is the energy of the incident particle, σ_E is the standard deviation of energy measurement, and a and β are constants depending on the detector type, e.g. the thickness and characteristics of active and passive layers.

From the construction point of view, one can distinguish between:

- Homogeneous Shower Counters: In homogeneous calorimeters the functions of passive particle absorption and active signal generation and readout are combined in a single material. Such materials are almost exclusively used for electromagnetic calorimeters, e.g. crystals (Crystal Calorimeter), composite materials (like lead glass, viz. PbO and SiO_2) or, usually for low energy, liquid noble gases.
- Heterogeneous Shower Counters (= Sampling Calorimeters): In sampling calorimeters, the functions of particle absorption and active signal readout are separated. This allows optimal choice of absorber materials and a certain freedom in signal treatment. Heterogeneous calorimeters are mostly built as sandwich counters, sheets of heavy-material absorber (e.g. lead, iron, uranium) alternating with layers of active material (e.g. liquid or solid scintillators, or proportional counters). Only the fraction of the shower energy absorbed in the active material is measured. Hadron calorimeters, needing considerable depth and width to create and absorb the shower, are necessarily of the sampling calorimeter type.

Good photon detection is mandatory to reconstruct final states, which include single photons and photons from hadron decays. COMPASS therefore uses a cellular lead-glass Electromagnetic Calorimeter with photo multiplier readout as ECAL1 – namely the

GAMS-4000 [43] that was in use in the WA102 experiment, which fulfils the requirements. Besides the GAMS calorimeter COMPASS also uses the WA89 lead glass calorimeter consisting of 650 blocks and the OLGA calorimeter consisting of 300 blocks. For ECAL2 it was planned to cover the central zone with a novel fine granulated calorimeter made of $PbWO_4$ heavy scintillating crystals, to cope with the heavy radiation load in the central zone of ECAL2. But because of funding problems COMPASS uses now a "pappardelle" *Pb*-scintillator sandwich calorimeter structure as central zone instead. Otherwise ECAL2 is identical to ECAL1.

The main task of HCAL1 will be the detection of neutrons from the decays of charmed baryons and triggering on them. It will be made of 25 mm thick Fe and 5 mm thick plastic scintillator sandwich cells. The total calorimeter thickness is 5 nuclear absorption lengths (see glossary: Absorption Length) for pions and 7 such lengths for protons. HCAL2 will be a compensated calorimeter with fine granulation in a heavily loaded central zone. It will be made of 16 mm thick Pb and 4 mm thick scintillator sandwich cells. Both, HCAL1 and HCAL2, will be used as part of the trigger for the PGF study. Because we expect hadronic products (especially *c*-quark content hadrons) the calorimeters will trigger after a certain threshold energy has been deposited in them.

2.5.6 Muon identification

COMPASS needs a large-area muon identifier for the muon programme and for the identification of muons from semi-leptonic decays in the hadron programme. This detector will make use of the higher penetration abilities of muons compared to hadrons and will be placed behind the Hadronic Calorimeters at each of the two stages of the spectrometer. The muons will pass about 50 cm of iron before the first Muon Wall (MW) and about two to three meters of concrete before MW2. In the case of MW1 Plastic Iarocci Tube [29] (PIT) detectors are used to get a position resolution of ≈ 1 mm. After the second MW stainless steel drift tubes are used with a diameter of 3 cm each.

2.5.7 Trigger

The trigger (more precisely: the First Level Trigger (FLT)) is a combination of signals from different detectors in the experiment. The trigger has to be adjusted to each physical process under study. The trigger is used to throw as much of the background as possible away and to only keep the interesting data. In the case of the photon-gluon fusion process for example one uses scintillator hodoscopes (see glossary: Hodoscope) to trigger on muons with a certain deflection angle off the beam line in conjunction (coincidence) with the above described hadron calorimeter threshold trigger condition and the information from a veto counter. A veto counter tells the system when a set of data should be thrown away. The veto counter for the muon programme for example is a halo (see glossary: Halo) veto counter in front of the target magnet to distinguish between muons interacting with the target and then being deflected or just muons that arrived already at a certain distance from the target and never interacted with it. The PRIMAKOFF reaction studies use a veto counter in the direct neighbourhood of the *Pb*-target, which distinguishes between PRIMAKOFF reactions and nuclear reactions.

Because the scintillator hodoscopes are at the end of the experimental set up and because of the processing time for the trigger condition the trigger signal will only be available after all the other detectors have read their values. To solve this problem one needs a signal delay for all the other detectors to bring the trigger and the arriving data from the other detectors into the right sequential time order. The general concept is that signals/data have to be stored as close as possible to the front-end electronics until the

trigger decision is made. Based on whether a good event was detected by the trigger detectors or not, the data of the other detectors will then be read and written into the data stream, or they will be dumped. For analog photomultiplier signals from calorimeters COMPASS uses the analog delay line method, where several hundred meters of cable are used for every readout channel to delay the signals. Other methods are electronic analog or digital pipelines. The APV25 chip (used by the GEM and silicon readout), for example, uses an 192 cell deep analog pipeline made up of an array of capacitors, where signals can be stored for $4\,\mu$s, before they are overwritten.

3 Slow Control in COMPASS

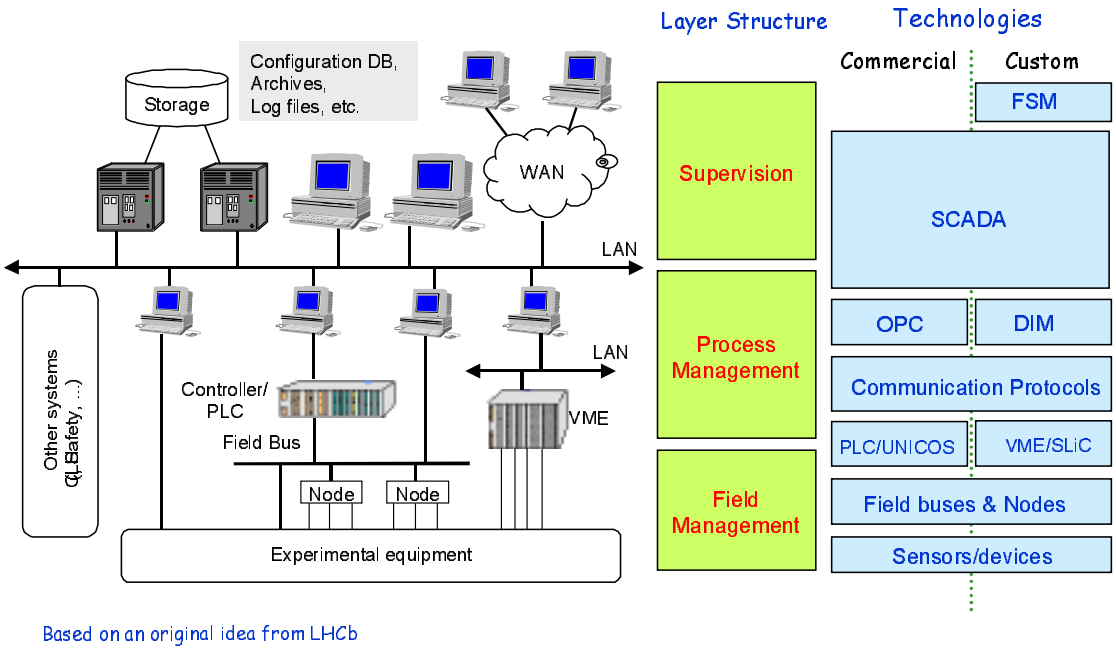

Figure 13: Technologies used in Slow Control.

In the previous section we saw the physics motivation of the COMPASS experiment. In order to enable the operation of the COMPASS detectors by just a few people on shift, it was decided to use a traditional two layer architecture for the Slow Control (see glossary: Slow Control, and figure 13):

- a supervision layer, using the SCADA-system (see glossary: SCADA System) PVSS II, and
- a front-end layer made up of VME (see glossary: VME), PLC (Programmable Logic Control(ler)) and field-bus components (see glossary: Field-bus).

Now before going into the details of what has to be done to implement the control software, a concepts overview is useful. Therefore, in this section, we will follow the complete path from the hardware to the operator in front of the control screen.

3.1 Motivation

The hardware is what we are interested in. The operators in front of their control screens want to *control* (switching devices on and off, changing voltage values, etc.) and to *monitor* (making sure that the hardware operates well) their hardware from a *centralised access point* (see glossary: Centralised Access Point). Therefore the operators are the top of the DCS hierarchy (see glossary: Detector Control System). They are the final point of decisions.

The whole slow control system is organised hierarchically (see figure 13), where the lower layers (like the field-bus layer) process the incoming data and pass higher level management information up to the next level in the slow control system. Only because of this abstraction process, where the system sorts out the most relevant information and passes it up one layer, the human operators are able to deal with the amount of data collected at the bottom layers. If the higher level layers want to communicate with the

lower ones they do this by sending *control commands*. This is also what an operator of the DCS does when he switches devices.

Besides the just described data processing, every layer in the slow control system can also be used for implementing procedures on how to act on certain exceptional circumstances, e.g. what to do when the gas sensor of a detector signals that the gas flow has stopped. These procedures, summarised under the term *loop back control*, can be implemented in every layer of the slow control system, depending on the degree of failure safety and speed needed.

As you can see in figure 13 the just described layers of the slow control system often correspond to specialised hardware, like PLCs or to dedicated computers running specialised software (in this report called "front-end application"). To enable the system to pass information between the different layers these specialised components have to be connected via a network. Therefore, the higher layers in the system "do not see" the real hardware, but only their data representation as they get it over the network. The point I want to clarify here is that most of the job of the slow control system is network communication and data processing and that the higher layers in the system only see an abstracted version of the real hardware.

Another important feature of the slow control system is the *data archiving* for later retrieval of the gathered values, so that the physicists can include the data in their analysis of the experiment. In the COMPASS set up the archiving is handled by the SCADA-system.

The whole slow control system allows a couple of people on shift to take care of the experimental set up *without the help of detector experts*.

3.2 Communication process

In figure 13 you can see the different hardware components involved in the communication process. The following discussion will be based on this figure.

The hardware can be divided into two groups: *sensors* (like an ADC (Analogue to Digital Converter)) and *actuators* (like a valve or a motor). The sensors convert the actual physical quantities like temperature, pressure, gas flow, etc. into voltage values. These voltage values in turn will be converted into digital values and either be transmitted over a network to a processing unit or directly accessed via the local bus of a processing unit. The processing unit is most likely a computer, which does time critical control jobs and basic control jobs like periodically reading the slave devices (see below: bus slave mode). One of its main purposes, from the SCADA-system perspective, is to publish the hardware values over another network (like ethernet) to the SCADA-system. In the SCADA-system these values are reported to the supervisor in front of the control screen. For the actuators the path is exactly opposite, e.g. the operator may decide to increase the gas flow in a detector and the SCADA-system sends the command all the way back to the valve, which controls the gas flow.

There are several types of networks used for data transmission between hardware and processing unit, which in general are called field-buses (see glossary: Field-bus). The type of field-bus and the type of field-bus protocol used in a certain application depend on several aspects like the speed, the real time behaviour, the configuration abilities and the reliability of the network. There exist field-buses that can be used in *bus master mode*, where the sensors in question report the change of values unsolicited to the processing unit, and there are field-buses that are used in *bus slave mode*, where the sensors report the change only when they are asked by the processing unit.

3.3 Communication example

Imagine an operator in front of our control system who wants to set the voltage level of a high voltage channel and who wants to watch how the actual voltage of the channel reaches its set point. In figure 14 you can see a schematic picture of an example communication

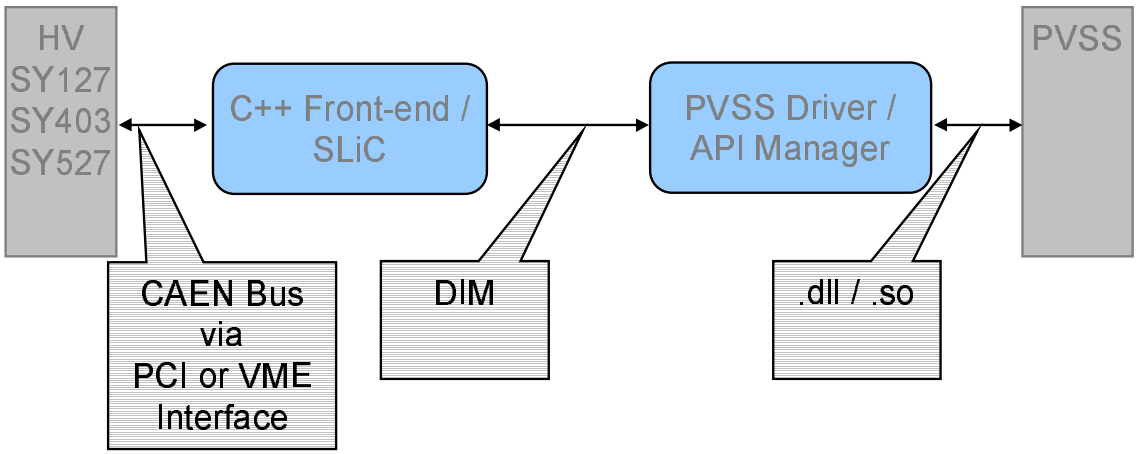

Figure 14: The jobs of the software in the system.

chain as it might also exist in the real slow control system. On the left you can see high voltage hardware (see glossary: "High" Voltage Crate) that communicates via a field-bus (CAENET field-bus) with the front-end application (in this example SLiC). On the right you can see the SCADA-system PVSS II which has a DIM[9]-protocol driver[10] and communicates over the DIM protocol (see glossary: DIM concepts overview) with the front-end application SLiC.

As soon as the operator triggers the change of the voltage level of the channel the communication process towards the hardware starts. The SCADA-system, which is responsible for the user interface, reads the value that the operator entered in some text field in the user interface and forwards it to a PVSS network driver. This network driver knows how to communicate with our front-end application and it forwards the value. The front-end application in turn knows about the field-buses it is responsible for and it forwards the value to the appropriate power supply. The power supply decodes the network communication and initiates the wanted action. As a result the voltage level of the voltage channel will start to change.

Because we use in our example a CAEN power supply, which is connected via a CAENET field-bus and which only operates in bus slave mode (see *bus slave mode* in section 3.2 on the preceding page) the change in the voltage level is not reported automatically. In this case, it is the job of the front-end application to ask the power supply periodically for the voltage levels of all the channels in the crate. After the front-end application has read the voltage values it forwards them (for efficiency reasons only the ones that have changed) to the appropriate SCADA-system network driver. The SCADA-system can then display these values on the screen for our operator.

3.4 Possible points of failure in the chain

In the outlined communication process above there are several possible points of failure:

- controlled hardware damage

[9] Distributed Information Management System [75]
[10] Actually this is a DIM API-manager that behaves like a driver.

- field-bus disconnect: The cable is accidentally unplugged or the cable has a problem.
- control unit:
  - power failure: The system might be accidentally switched off or there is a general power cut.
  - host OS (Operating System) bug: The OS of the control unit might have a bug.
  - control software bug: The control software, which is responsible for the readout process, might have a bug.
  - hardware failure: The CPU (Central Processing Unit) or some other IC (Integrated Circuit) of the control unit is damaged.
- LAN (Local Area Network) disconnect
- SCADA-system
  - power failure
  - host OS bug
  - SCADA-system bug
  - hardware failure

These possible failure conditions have to be taken into account at design-time of the whole system. The most critical hardware control jobs should be as close as possible to the actual hardware. If for example major damage can be caused by not reacting fast enough on failure conditions, then these control procedures should be implemented directly in hardware (e.g. no program and no CPU is involved at all). It is also a good idea to either publish a *heart beat* of all vital system components so that the higher layers can recognise a problem of a lower layer by monitoring this heart beat or to implement a top down check procedure where the higher layers ask the lower layers if they work correctly. In addition, if there is redundancy in the system then that redundant part can take over the tasks of the failing part.

3.5 Software in the system

Before we start to look at the different software pieces in the system it is instructive to talk about the reasoning, why we need all these software layers. No matter, into which engineer science we look, we find everywhere the concept of building blocks, like mass-produced screws, mass-produced ICs or mass-produced car engines. The main reasons for introducing building blocks are to hide the local complexity by defining an easier and clean interface to the higher layers and to shield the higher layers from changes in the lower layers. For example, to assemble a car we do not need to know how an engine works, we just have to follow the interface given by the engine designers to connect the engine to the rest of the car. On the other hand if the engine designers decide to improve the efficiency of the engine they can do so without interfering with the car assembly process simply by following the interface definition. So what we do in general is to introduce interfaces to allow different groups of people to develop independently on both sides of the interface.

We will start our investigation with the topmost (nearest to the user) piece of software, the SCADA-system itself. By using a SCADA-system we have an interface that shields the developers on top of it from

- the different host operating systems,
- the different GUI (Graphical User Interface) implementations,
- the data storage or archiving process,
- the networking details,

- the load balancing in distributed systems,
- the failure tolerance in redundant systems,

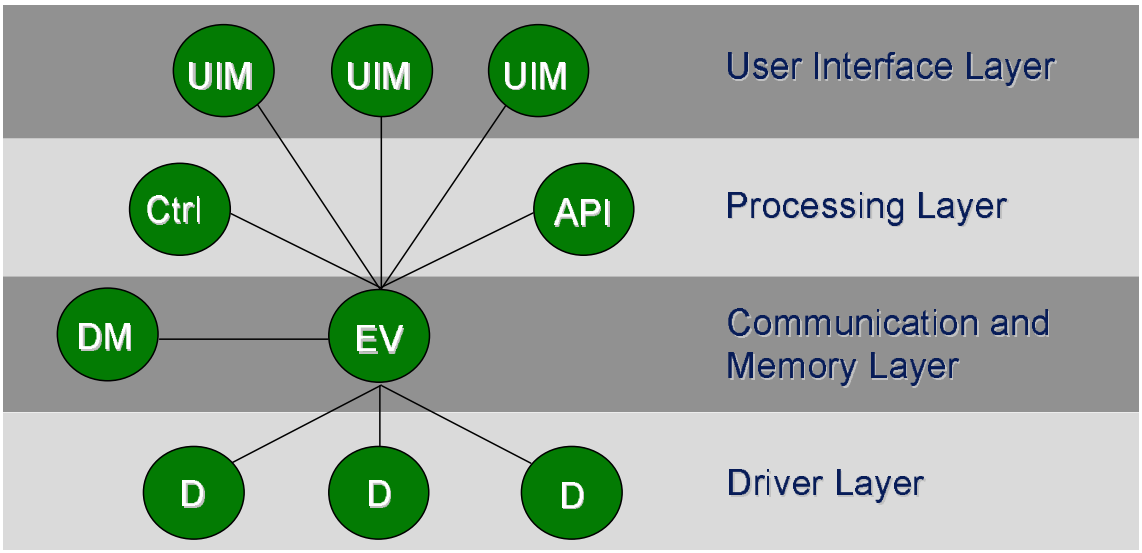

Figure 15: PVSS layers:

UIM:	User Interface Manager	Ctrl:	Control script language Manager
API:	API Manager	DM:	Data Manager
EV :	Event Manager	D :	Network Driver

In COMPASS we use the PVSS II SCADA-system. PVSS II is internally composed of several different managers. These managers are pieces of software with predefined jobs that communicate by messages. You can see the *internal architecture of PVSS II* in terms of these managers in figure 15. The heart of the system is the event manager. Here all messages from all other managers come together. At the bottom there are the drivers to communicate with the outside world, over OPC (Object Linking and Embedding (OLE) for Process Control) for example. The values gathered by the drivers arrive at the event manager, which then forwards them to the data manger. The data manager will log them in a database with a timestamp for further processing. On top of the event manager there is the User Interface (UI) manager and the Control manager. The UI manager is responsible for the display of UI elements like panels and it responds to user interactions by executing scripts on behalf of the user. The scripting language supported by PVSS is called Control (see *PVSS Control scripting language* in section 6.1.2 on page 55) and it is a C like interpreted language. The Control manager interprets standalone scripts that are not attached to user interface elements. All managers above the event manager only deal with PVSS data points (see *PVSS data point (DP)* in section 6.1.1 on page 55), where a data point is PVSS' concept of a basic data entity. This also means, that the whole job of a programmer on top of PVSS can be seen as only setting and getting PVSS data points with some data processing. Besides that PVSS provides the possibility to extend the overall system by writing API[11]-managers, which are basically DLL (Dynamic Linked Library) plug-ins.

The layered structure you see in figure 15 is also the basis for PVSS's distributed nature. Every manager may run on a different machine to distribute the workload on several computers. It is even possible to have a multi PVSS system interconnected together to spread workload on different PVSS systems.

[11] Application Programmer Interface

The only interface pointing downwards (towards the hardware) in the SCADA-system is the network driver layer to access the LAN. The SCADA-system may already provide some drivers for industry standard protocols for hardware access like OPC or such drivers may be added by the user of the SCADA-system. These drivers shield the SCADA-system from

- the way the data was retrieved and
- the networking details

and the driver layer

- provides the abstract concept of data points instead of network accesses[12].

The PVSS II SCADA-system already has support for OPC built in (OPC is only available on Windows platforms) and a driver for the DIM protocol was written to allow UNiX systems to communicate with PVSS II.

In general it was decided to use commercial software like OPC servers wherever possible. But if we have to deal with custom hardware, or legacy devices or if we have to implement loop back control in the front-end layer we have to implement the front-end software ourselves.

It would be possible to write a PVSS driver for every field-bus type and every piece of hardware connected to this field-bus type, but because there were more requirements than just the communication and to keep the communication layer independent of the SCADA-system another approach was chosen.

It was decided to build a framework called SLiC for custom front-end applications for hardware access and loop back control. This software was written to be easily extendable, easily configurable, reliable and efficient. There is also a front-end application that uses the framework to translate between the slow and low-level field-buses and the higher level DIM-protocol on top of a fast ethernet network, to allow PVSS II to access the hardware. This application is somewhat inconsistently also called SLiC and can be seen as an example for the capabilities of the SLiC framework. This SLiC application can be easily extended to implement time critical control jobs that have to run reliable without the help of the SCADA-system.
The SLiC application shields the SCADA-system from

- the field-bus type,
- the field-bus access mode (bus master or bus slave) and
- the differences in hardware of similar types.

To explain the last point let's take the example of the different high voltage crates either from the same company or from different companies. The different high voltage crates offer different functionality to the user, but the SCADA-system is generally only interested in a small subset of these features that are common to all the high voltage crate types. Therefore the SLiC-software takes the responsibility of gathering the required information from the crates and sending it in a uniform manner to the SCADA-system, which then can handle the high voltage hardware in a common way.

On the other hand the SLiC-software can be customised to use the additional functionality of some crate types internally (e.g. in loop back control) without the SCADA-system being aware of these features. In SLiC we reflect the difference in sensors and actuators by publishing the actuator elements as *WriteProperty* C++ objects (see *WriteProperty*

[12]Data points may or may not be connected to network accesses. Data points can and are also used solely for internal data storage.

in section 5.4 on page 47) and the sensor elements as *ReadProperty* C++ objects (see *ReadProperty* in section 5.4 on page 47).

In this report we do not investigate the details of what happens at the other end of the field-buses. This is mainly the domain of electronics engineering.

3.6 Loop back control and SCADA rules

It was already mentioned that the decision, where to implement the loop back control is a matter of the failure safety and speed needed. If your requirements are to react fast and/or reliably then you need to implement the loop back control procedures in the front-end or even better directly in the hardware. On the other hand if the speed and the reliability do not matter too much then it is easier to implement the loop back control in the higher levels of the control system, like in the SCADA-system. To implement the loop back control in the SCADA-system has the advantage that you have a central point of development (as opposed to many differently specialised front-end applications) and that the SCADA-system shields you from differences in the host OSes and GUIs.

Loop back control in hardware or in the front-end application is up to the hardware designer or the implementer of the software and therefore always a special case. If you use for example the SLiC framework to implement loop back control you have to follow the SLiC API and to write C++ code to implement the wanted behaviour.

The SCADA-system provides an API for creating rules and for creating GUI-panels. These *SCADA rules* can be seen as functions that map certain preconditions to defined post-conditions. Rules are set up by the control system designers or by detector experts and they are implemented in PVSS by programmers as Control scripts. If a script does not need to interact with the UI then the script execution can be done by a PVSS Control script manager in a standalone thread of execution. This is ideal for loop back control as the scripts can also run without the user interface. Otherwise, if a script is dependent on the user interface, the scripts are executed by the PVSS UI manager on behalf of the user. We can categorise the SCADA rules as automated rules and interactively executed rules.

We compared above a rule with a function. A function maps a set of input values onto a set of result values. The *automated rules* take their input values solely from PVSS data points[13]. As soon as a change occurs in the set of data points, for which the rule is responsible for, the change will be processed and the result will be reflected by a change of some other set of data point values and/or UI-elements. With this approach changes of values of PVSS data points may be used for loop back control or they may be connected to UI-elements. As an example for the connection to UI-elements imagine a high voltage channel. Such a channel only allows a certain amount of current to flow. If the current is too high an overcurrent alarm is signalled by the hardware and will be reflected as a change in some data point in PVSS. This hardware alarm is now mapped in the SCADA-system to an UI alarm signal. The state of the high voltage channel goes, visible for the user in front of the operating screen, from good to bad (e.g. a colour field changes from green to red). As a result the user might choose to switch off the channel, the high voltage crate or the whole detector that the channel belongs to.

The *interactively executed rules* are simple Control procedures, which are attached to UI gestures on UI elements – e.g. a button press event. Here the set of input values for the rule is a user gesture like a button press. The results will again be reflected by a change of some set of data point values and/or UI-elements.

[13] actually from DPEs (see *PVSS data point element (DPE)* in section 6.1.1 on page 55).

3.7 User interaction

The user interaction with the system is normally bound to the SCADA-level in the control system hierarchy. The SCADA API for GUI-panels allows the programmer to create UI panels in an OS and GUI independent way. As described above the UI elements of the panels then may be connected to rules to display the relevant information in a user friendly way and to trigger control commands (see *Control Command* in section 3.1 on page 35) in the SCADA-system. These panels are the interface between the PVSS data points and the operators in front of the control screens.

3.8 Conclusion

We saw that there is a lot of communication involved in the overall control process which ends in the PVSS data point concept. The loop back control can be put in any piece of software involved in the communication process. The reasons why to put it close to the hardware are the faster response time and the more fail-safe position. The reasons why to put it in the SCADA-system are the centralised position and the OS independent implementation. The archiving is solely done in PVSS by its internal archiving mechanisms. The slow control system can already handle a lot of problems according to rules which were set up by detector experts. But as software is always limited in its abilities the final point of decisions is the operator in front of the control screen. He interacts with the system with the UI provided by the SCADA system.

4 The Requirements

This work is concerned with the development of the slow control software for the GEM and silicon detectors of the COMPASS experiment and with the integration of this software into the overall COMPASS control system.

4.1 First level requirements

Both, GEM and silicon detector, need

- voltage control (see *Slow Control Term: Control* in section 3.1 on page 34),
- temperature monitoring (see *Slow Control Term: Monitor* in section 3.1 on page 34) and
- pressure monitoring.

Special for GEM is the

- gas monitoring

and special for silicon is the

- cryogenics control.

Both detector types use the CAEN SY527 (see glossary: "High" Voltage Crate) power supply [71], as it is able to produce low and high voltage output. The temperature will be measured with Pt100 sensors and the pressure will be measured with Baratron^{TM} pressure transducers.

All of these devices are connected to some sort of controller (e.g. a front-end application like SLiC or a PLC, etc.), which has access to that device via a dedicated hardware, like an ADC card or a network card. Most of the time the manufacturer of that dedicated hardware also supplies a driver for that hardware for the operating system of the controller, so that the programmer of the controller has access to it via the API that the operating system provides. If this driver is not available it has to be written. As the programmer uses the operating system API to interact with the real hardware he does not see any differences in the interaction with a networked device or with a measurement device, which is directly connected to the controller via an ADC. The procedure on how to get the data from the device clearly differs from driver to driver and it may also depend on the type of hardware. But the important point to note is that the driver layer adds another level of abstraction, which is able to hide the hardware details from the programmer, so that at an end the programmer only sees the API and not the hardware. From the viewpoint of a programmer a device can be summarised as a procedure he has to follow to get or set the values of the device and/or to initiate an action in the device.

As mentioned above this work is concerned with the development of the slow control software for the GEM and silicon detectors. As an example consider the GEM detector. Despite its complex internal structure and functionality a GEM detector is for the slow control system nothing more than just six voltage values and a gas flow value (the temperature and pressure will be taken globally for the whole experiment). The task of the slow control system is now to gather these values, to forward them to the relevant level of the slow control hierarchy and to process them according to predefined rules. Furthermore, the slow control system has to provide the user a convenient way to interact with the devices from a centralised access point.

4.2 Prerequisites

It is not necessary to start from scratch in order to implement the first level requirements. There is already a PVSS framework (the JCOP framework [91]) available which provides generic libraries, predefined PVSS data point types and data points, UI panels as well as guidelines for further development to aid the developers in the implementation of detector specific functionality. Therefore it is a quite straightforward job to link additional developments into the overall structure.

4.3 Derived requirements

Out of the first level requirements other requirements arise. As there was, at the beginning, nothing like the SLiC software available, SLiC had to be first designed and then implemented. Another obstacle inside PVSS was the lack of exception handling facilities in PVSS' scripting language Control. A convention had to be set up on how to handle exception conditions. There is also no documentation system or version control system integrated in PVSS, so we had to find out, which systems would be appropriate for the non standard Control script language. The creation or manipulation of lots of data points at once was another problem to be solved. As a first solution an experimental data point manipulation language was introduced to allow the DCS maintainer to easily perform update and maintenance tasks. Because the JCOP framework is a base for all the LHC (Large Hadron Collider) experiments it is too general to propose a top down system configuration procedure. The configuration procedure for the COMPASS experiment is in the moment experimental, as the COMPASS collaboration has not yet decided on a final solution.

5 The SLiC software

It was already mentioned in section three that SLiC is a framework for custom front-end applications for hardware access and loop back control. Based on this framework there is also the front-end application, which is somewhat inconsistently also called SLiC and which can be seen as an example on how to use the framework.

The SLiC framework was designed to support the bus master (see *bus master mode* in section 3.2 on page 35) and bus slave (see *bus slave mode* in section 3.2 on page 35) operation of devices. Furthermore, it is independent of the network-publishing method (like DIM) and it is independent of the configuration technology, like a database or an XML (eXtensible Markup Language) file. As a consequence every front-end application based on the SLiC framework can easily be extended to support several configuration methods and publishing schemes.

The front-end application uses the framework to translate between the slow and low-level field-buses and the higher level DIM-protocol (see glossary: DIM concepts overview) on top of a fast ethernet network, to allow PVSS II to access the hardware. It is also a place where to implement time critical control jobs that have to run reliably without the help of the SCADA-system. The reason why we use XML for configuration of the front-end application is because there is still no consensus on which configuration procedure to use for the slow control configuration and XML seemed neutral.

At the beginning of the software development two prototypes were implemented and later the ideas out of both branches were merged into the final SLiC product (both the framework and the application). This section will give you an insight in the challenges that had to be solved in the SLiC development and why certain decisions were taken over others, but it will not go into the very specific implementation details. For these details and for a user manual, please refer to the SLiC online documentation [88]. For C++ documentation please have a look at references [67] and [68].

In the following text of this section the descriptions are generally applicable to SLiC as framework and SLiC as front-end application. Therefore no differentiation between the two is made. The framework and the application were developed in parallel and share the same infrastructure and design principles.

5.1 Project infrastructure

When you get the source code distribution of SLiC all files belonging to the project are located below the folder named `./SLiC`. In this folder, there is a bash shell script called `./SLiC/env.sh`, which will set up environment variables. Before using this script it has to be adapted to the local situation, like where the required libraries are installed, where the C-compiler is located (if not in a standard position) and so on. After the command

```
source env.sh
```

inside of the `./SLiC` directory the make targets should work.

Below the top-level folder, there is the configuration file folder, `./SLiC/xml`, which contains example configuration files and the XML DTD (Design Type Definition).

There is the `./SLiC/perl` folder, which contains supporting perl scripts for the configuration process, for intelligent log file handling and for make file support.

There is the `./SLiC/src` folder, which contains all the make file files and source files.

There is the `./SLiC/doc` folder, which will contain the HTML[14] documentation after building the documentation with the doxygen tool [83].

[14] Hyper Text Markup Language

There is the `./SLiC/obj` folder, which will contain all the object, library and executable files after a build of the software. This folder will have a subfolder for each build configuration, e.g. build with debugging information (-g compiler option), build with insure++ [80] source instrumentation or fully optimised build (-O2 compiler flag). The advantage of this structure might be clarified by an example. Imagine you compiled the application with debugging information. After the application is fully debugged you might choose to fully optimise the code and turn off debugging information. After the delivery of the software a bugreport arrives and you have to go back to debug-mode of your application, repair the problems in a few files and compile the whole project again in full optimisation mode. As you can see, with this structure of object files you do not have to compile the whole project after switching between build configurations. It is only necessary to compile altered files. Therefore this approach has the clear advantage of saving a lot of compilation time, especially in the insure++ build configuration.

And finally there is the `./SLiC/product` folder, which must be set up to hold symbolic links to all pieces of software needed for the deployment of SLiC. This includes (besides other things) the DNS (DIM Name Server) (see glossary: DIM concepts overview), the DIM browser xdid and a perl script for intelligent logging.

The make files work recursively on the file system hierarchy and they support the following make targets:

- `make` will build all the files in a certain directory, e.g. the object files, the library files if any and the executable if any.
- `make all` will build all the files in the folder and all its sub folders as described under the `make` option.
- `make static` and `make staticall` will do the same as the `make` and the `make all` targets except that all final executables will be completely statically linked.
- `make debug` builds everything necessary and then starts the ddd [82] graphical debugger.
- `make deliver` builds everything necessary and then bundles the SLiC executable with all the files in the `./SLiC/product` folder as a `tar.gz` file.
- `make documentation` does what its name suggests. This target depends on the open source doxygen documentation tool [83].

For the doxygen tool to work properly some special comment blocks have to be added to the source files. After this preparation, doxygen is able to produce several output formats like HTML, PDF, PS, RTF...

Depending on the name of the main target of a make file the make file decides if to use or not to use the libtool [72] tool to simultaneously build static and shared libraries in a portable way. The main disadvantage of libtool is that it needs more time to produce both versions of a library. Its main advantage is that it is a simple to use "do the right thing" tool in a portable way.

All source files (please notice that this term is not limited to only C or C++ source files) are held in a CVS (Concurrent Version System [81]) repository to allow several programmers to work on the same files simultaneously, to keep track of different versions of the software and to allow for an easy comparison of old to new files.

After several tries with the open source tools libcwd [77] and libnjamd [78] we finally decided to use the commercial insure++ product [80] to check our software for memory leaks and memory corruption problems. This tool does source level instrumentation to detect all sorts of problems. Please refer to the manual for a complete description.

To easily navigate in and to analyse the source code we use the open source tool Source Navigator [86]. This tool has several browse modes which allow the developer to easily find his way through the numerous lines of code in the project.

5.2 Libraries used

In SLiC we build upon the functionality of the three libraries:

- libACE [73] is a portable general utility library with heavy multithread support,
- libxerces-c [74] is used to parse the XML configuration file and
- libdim [75] provides the DIM communication functionality.

Especially libACE is very useful because of its portable thread abstractions like the C++ class *Task* and the locking facilities. The *Task* class builds on top of the host OS' multi threading capabilities and makes spawning threads and communicating between threads easy. The locking facilities help the programmer to use inter thread locks, inter process locks and file locks in a uniform, exception safe and portable way.

5.3 Basic design decisions

Two basic design principles in computer programming, as defined in [65], are to introduce abstraction barriers (see glossary: Abstraction Barrier) and to use data-directed programming (see glossary: Data-Directed Programming) to achieve additivity (see *additivity of generic interfaces* on page 70). In OO (Object Oriented) programming this boils down to "programming towards interfaces" (see glossary: Programming Towards Interfaces). By programming towards interfaces we introduce "horizontal lines", which represent abstraction barriers that isolate different "levels" of the system. At the same time we also introduce "vertical lines" that isolate modules in each level of the system. The horizontal lines mean that different pieces of our software just use the interfaces of other objects to talk to them and therefore any dependencies on the implementation of these objects are avoided. The additivity in each level is achieved by implementing the interfaces and overwriting the virtual base class methods in the interfaces. In this way we can add more classes that adhere to a certain interface without interfering with other classes that implement the same interface. Because interfaces cannot be instantiated, we need so called factories (see glossary: Creational Patterns: Factory) to bring implementations of interfaces into life. In reference [66] one can find several "cook book recipes" on how to solve general problems in writing modular and extendable OO software. Personally, I think, one of the most important principles in software design is to avoid redundancy on all abstraction levels. I noticed that by doing so all other principles of good software design follow automagically. Up to now, to see the redundant parts of the code, is a matter of experience. But in the future there might be a theory, like the database normalisation theory, which gives an algorithm on how to remove the redundancy in code. Up to now, we can only rely on "cook book recipes" like design patterns, where people with a lot of experience collect there insights and write them down.

SLiC introduces a general base class *Object* from which nearly all other classes inherit. It then uses a global factory singleton (see [66]) to instantiate all classes that derive from *Object* and stores them in a global list. To advertise the creator objects to the global factory it seems at first sight a good idea to use static initialiser objects (see glossary: Static Initialiser Object). The advantage of this approach would be that all code that belongs to an implementation of an interface is separated from all other code and therefore the code dependencies are easy to figure out. But because such static initialisers are not referenced by any other piece of code in the system the linker does not link the code for the static initialiser into the final executable without some extra linker arguments. Because of this additional complexity, which might not be obvious for a developer, it was decided to use normal *initialiser functions* where each developer of an extension has to add a line of code in a well known file, which adds the creator objects to the global creator list.

5.4 Data publication strategies

At the moment SLiC publishes every logical entity as a separate DIM service or command (see glossary: DIM concepts overview), even if the actual hardware treats several such entities as groups. Figure 16 provides an example of the internal handling of read-back value groups from a high voltage crate. Please keep this picture in mind for the following discussion. There you can see how the "ReadStatus" operation, which is defined in the

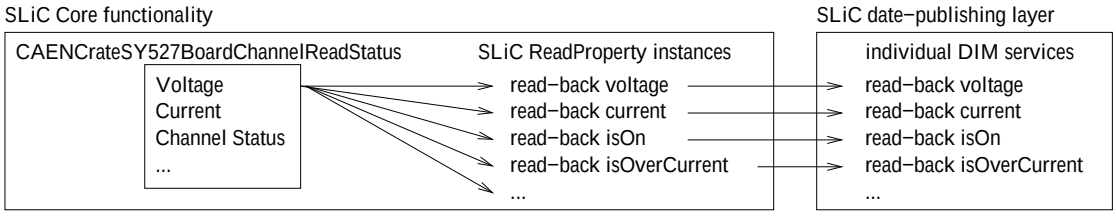

Figure 16: The logical entity publication strategy.

SY527 manual [71], causes a data packet to be sent from the SY527 crate to our software and how the packet is unpacked into SLiC-internal *ReadProperty* C++ class instances (see below for an explanation of the *ReadProperty* C++ class). Another example of such groupings for the SY527 high voltage crate are the bit states, like the on-off state, the over current state, the under current state, and other states, which are transmitted as a single 16 bit word. The decision on what data to package into groups is up to the hardware producers and the groupings are different even for hardware of similar type from the same producer. As you can see in figure 16 the elements of the groups are separated and published by SLiC as independent properties (the arrows from the left box to the right box).

Just to avoid any confusion it should be mentioned that this separation does not necessarily imply any inefficiencies in the read out process. Internally to SLiC we take care of property groups in an efficient way.

The reasoning behind the separation approach is to keep the client independent of the hardware. The splitting of the values that were read from the hardware into independent pieces of information introduces a new layer of abstraction on which the clients can build. Another advantage is that clients can decide which properties they are interested in so they can ignore the additional unwanted information. Internally to SLiC this separation into virtually independent values is always done (as you can see in the SLiC "Core functionality" box in both figures 16 and 17) and SLiC uses a *ReadProperty* and a *WriteProperty* C++ class to implement this separation. But one could also imagine, that instead of publishing these *Property* objects independently, one could put all these values on a queue (as indicated by the arrows out of the "Core functionality" box in figure 17) and a separate thread would unqueue the values, tag them and transmit them over a single transmission channel (indicated by the "channel" at the bottom of figure 17) to the client on a regularly timed bases (e.g. every 1 s). With this approach the network traffic could be reduced by sending compressed data packets instead of uncompressed ones. Because the single channel publication strategy approach produces bigger byte bundles compression would really help. The client would then either simply get all information that SLiC gathers from the hardware, independent of the fact if the client needs this information or not, or the client would have to follow some sort of initialisation protocol, to subscribe to the wanted values.

In my opinion both approaches, the logical entity publication strategy and the single channel publication strategy, are valid, because they do not brake with the idea of an

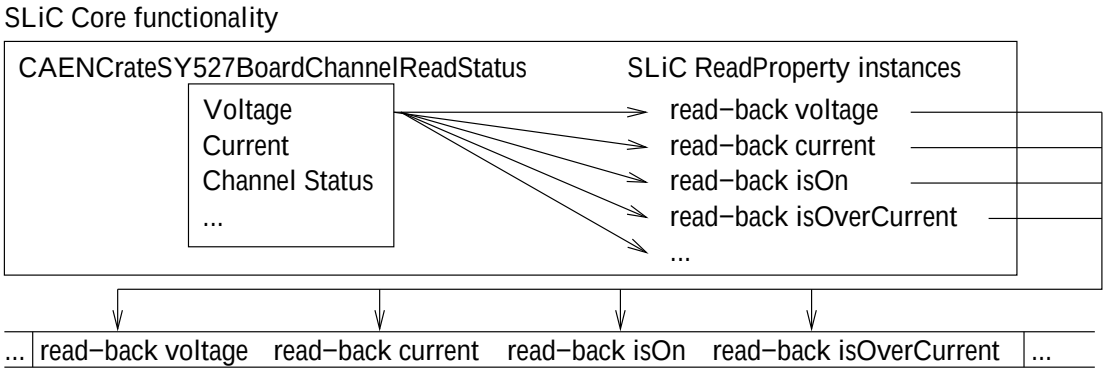

Figure 17: The single channel publication strategy.

additional abstraction layer between the hardware and the client of the SLiC software. Any approach in between, where for example the transmitted data packets are tailored to the hardware readout structures (which might be the most efficient approach) should not be taken into consideration, because this would introduce a lot of additional special cases, where, from a logical point of view, there are none. As an example consider again high voltage equipment. The CAEN SY127, SY403 and SY527 series of high voltage crates deliver the read-back values in different read-groups. Not to introduce this additional layer of abstraction would mean these crates would have to be implemented differently on the client side of the SLiC software (which might be the PVSS driver layer or the PVSS script layer), whereas with the current model the client side of SLiC is not aware of any differences in the internal workings of high voltage hardware. Additional code means additional maintenance efforts. Especially any change in the use of data structures in SLiC would imply the necessity of changing the code in the client side also.

5.5 Intelligent logging

It is important to follow the execution of the program in the production environment to guarantee the correct behaviour and to easily track down sources of problems. The general rule is, the more log output, the easier the analysis of problems. But because a verbose log file would just fill up the hard disk in a few days a more sophisticated logging mechanism is needed. In SLiC every class deriving from *Object* and every instance of a class derived from

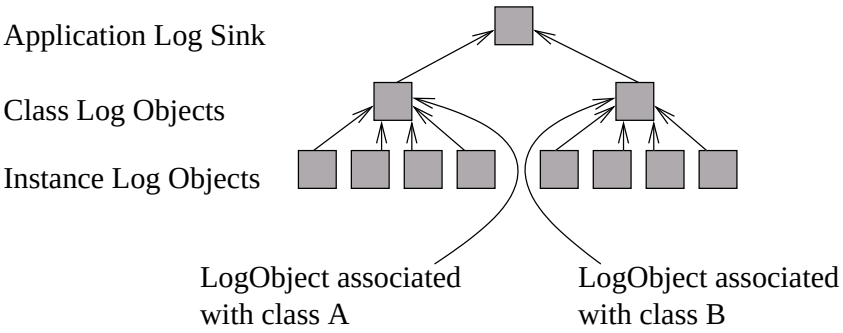

Figure 18: *LogObject* logging hierarchy.

Object has a *LogObject* associated with it. The result is a three level hierarchy of log objects (see figure 18), where every level forwards log messages to one level higher in the log object

tree. Every object that wants to print log messages attaches a log priority[15] to the message and calls one of the log methods on its *LogObject* instance. This per instance *LogObject* forwards the request to the per class unique *ClassLogger* instance, which in turn forwards the message to the final, per application, log sink. The final log sink might be stderr, a file or a network connection or any combination of them. Each level of logging has the possibility to reject log messages depending on its log priority. Therefore the log verbosity can be controlled on a per object basis, on a per class basis and on a per application basis. By implementing some IPC (Inter Process Communication) mechanisms into SLiC it will be possible to control the log verbosity at runtime. Internally the *LogObject* uses a log message queue. A separate thread then dequeues the messages and writes them to the final sink. This approach ensures that the application is not slowed down by slow system IO.

This behaviour is already quite flexible but improvements are still possible, e.g. it would be nice to get no log messages at all as long as the system is running fine. But in case of an error the last 100 lines of log messages preceding the error should also be printed to the log. To implement this idea a perl script is used. The perl script forks off another process, which starts SLiC with maximum verbosity and a Linux FIFO (First In First Out – also called named pipe) as log sink. On the other end of the FIFO the initial process of the perl script reads the log messages and stores them in a circular buffer. As soon as a log message with log priority LM_ERROR occurs the error message together with the circular buffer is printed to a log file and we get the error together with the preceding lines of normal debug output. Besides that it is possible to send the perl script a signal to force it to immediately switch into verbose mode so that the user can see the current activities of SLiC in full detail.

5.6 Multithreaded nature

As mentioned above one of the reasons why to produce SLiC was to translate from the slow field-buses to the DIM protocol on top of a fast ethernet connection. If we just read out all field-bus elements one after the other we would gain no speed improvements at all, because the system is only as fast as the slowest piece in the chain (the field-bus). Therefore we need to parallelise the accesses to field-bus interfaces[16] in order to report the hardware values as efficient as possible. To parallelise in SLiC means to use a different thread per hardware interface (ISA, PCI or VME card).

One of the first difficulties we were faced with was that because the GNU standard C++ library distributed with gcc [79] up to version 2.92.3 is not multithread safe we had to renounce on it. It was decided to use libACE instead. libACE is not quite a replacement for a standard C++ library, but it implements some containers which were sufficient for our purposes. Furthermore libACE provides some multi thread convenience classes like the already mentioned *Task* class, which makes working with different threads very easy.

Another difficulty was that libdim keeps mutexes locked in both situations, while a DIM command call-back (see *DIM Command* on page 71) is issued and while a DIM service (see *DIM Service* on page 71) is processed. But on the other hand we also need to keep locks to ensure that only one thread accesses a given hardware interface at a time. This situation lead to problems where in the case of a DIM service SLiC locks on a hardware interface first and then DIM locks inside its service handling code and in the case of a DIM command first libdim locks and then SLiC needs to lock. These two situations are schematically shown in figure 19. The time flow is from left to right. A dot indicates the process of acquiring a mutex, either in the DIM library (the upper line) or in the SLiC

[15] like LM_DEBUG, LM_INFO, LM_ERROR.

[16] Here the term interface means the hardware interface like a ISA, PCI or VME card.

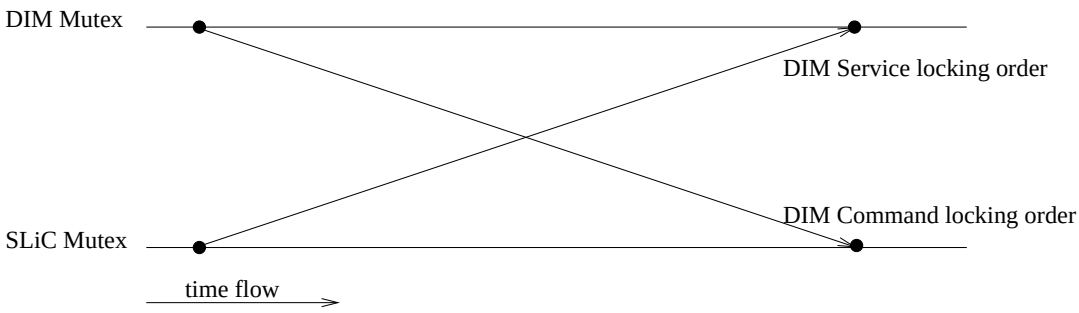

Figure 19: The deadlock condition.

code (the lower line). The crossing arrows show the deadlock condition. We solved the problem by enqueuing DIM command requests and processing them in a separate thread, so that we can keep the same locking order in both situations.

5.7 Common-devices

Although SLiC was designed as a framework it has been specialised to a fully functional application. For this application the CAEN line of high voltage power supplies, the SY127, the SY403 and the SY527 (see glossary: "High" Voltage Crate) and the access to the CAENET field-bus (see glossary: Field-bus) over a PC ISA card or a VME card have been implemented. These objects all build on top of a specialised framework class called *StructuredDevice*. This class' main objectives are to implement a field-bus request hierarchy and to implement a common user-friendly error-handling scheme. As the communication with the hardware is often done via a field-bus the request chain and the error handling scheme have to work hand in hand. The implementation of the *StructuredDevice* on top of the framework can be used as one example of how to specialise the framework to actual hardware requirements.

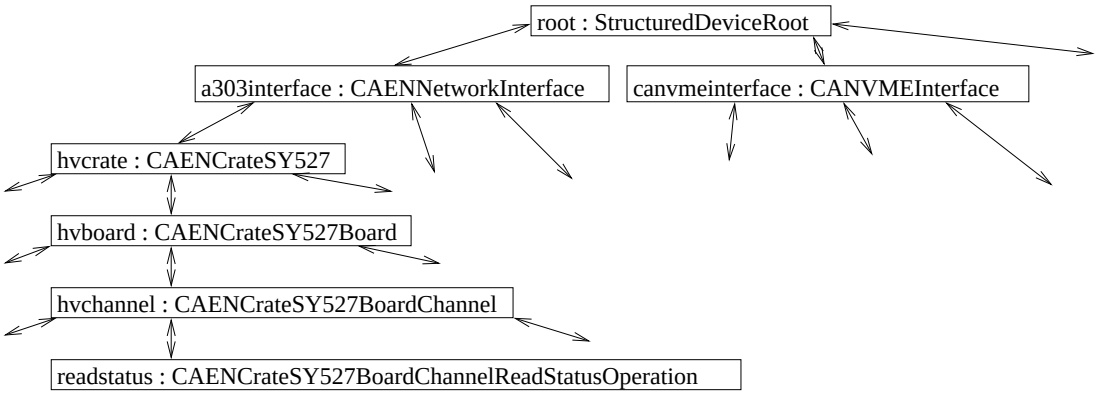

Figure 20: The *StructuredDevice* hierarchy.

The *StructuredDevice* instances are organised as a tree, which is rooted at the single *StructuredDeviceRoot* instance (see figure 20). Below the root element there are the field-bus hardware interface instances. These elements only know how to talk to the field-bus – how to send the request network frames (simple byte arrays) and how to receive the answer network frames. All leaves in this tree are of type *StructuredDeviceOperation*. An operation in the high voltage control part is for example to set the output voltage of a channel or as in figure 20 to read the status information (the status information for a

SY527 channel is composed of the voltage value, the current value, the on/off state and much more) of a channel.

As an example of a possible *StructuredDevice* hierarchy we consider the case of the CAEN SY527 power supply implementation (see also glossary: "High" Voltage Crate). In the implementation of the SY527 crate the leaf operations (like switching a channel on or off, etc.) would be located below a high voltage channel instance, which would be located below a high voltage board instance, which would be located below a high voltage crate instance which finally would be located below the field-bus interface. As you can see, the *StructuredDevice* hierarchy simply follows the physical layout of a SY527 crate. This structured implementation is not necessary for just communicating with the device and in fact the data representation as used to communicate with the crate over the network would at first sight suggest to implement one class with one method per operation. The real advantage of this structured approach lies in the possible error-handling scheme described below.

All requests to a device to set/get a value or to perform some action start at leaf elements and make their way up until they reach the field-bus interface, which does the request and returns the answer back to the leaf element.

The interesting case however is when problems appear. If the communication of a leaf element fails then the leaf element starts a check procedure, which works recursively on the *StructuredDevice* hierarchy. Every element in the *StructuredDevice* hierarchy implements a method that allows to determine if that element works fine or not. To clarify what was just said, consider the boards in the SY527 crate or the crate itself. The check procedure in the board element of the *StructuredDevice* hierarchy for the SY527 crate for example reads the high voltage boards characteristics, which only depend on the board and not on a specific channel[17] and we can say if the board is responding well or not. The check procedure in the crate element reads only the name of the crate. This operation is possible even if there is no single board in the crate and therefore we can decide if the crate is well connected and works fine or not.

The recursive check procedure starts at the leaf element where the problems appear initially. The leaf element asks its parent to check its state. If the parent element works fine it is obvious that the problem lies in the child and the software can tell the user the exact point of failure (or most often the point of misconfiguration). On the other hand, if the parent also detects a problem the check request is forwarded to the next parent, and the same procedure is repeated until a "good" element is found. The root element in the hierarchy is always in a good state, so that at an end the problem might be the network interface. If the problem is detected in the network interface the reason for the communication failure might be a missing driver on the front-end computer or a defective cable or a misconfigured device.

As we saw just in the last paragraph, the check procedure cannot decide if a problem is due to a defective cable or due to a misconfigured networking device, but nevertheless the check procedure is able to guide the user to the source of the problem. The check procedure can safe a lot of time in tracking down hardware problems or configuration mistakes and is therefore a valuable part of our software.

This description is only an outline of the rough structure. The details about the request mechanism and the error-handling scheme can be found in the SLiC documentation [88].

[17] I admit that this example is a bit artificial, because the board and the channels are not really separable as they share the same hardware, but it shows the idea behind the check procedure.

5.8 Surveys

If a field-bus supports several bus-masters then a device may work in bus-master mode and tell the SLiC software about read-back value changes by triggering a call-back function in SLiC. On the other hand, if a device only works in bus-slave mode then SLiC has to care for the periodic updates of the read-back values. These periodic updates are done by *Survey* instances.

A *Survey* has two signals as input, one that tells the *Survey* to start and one that tells it to stop. These signals might be periodically produced by a timer in the software or there might be any other trigger source like the start of spill (see glossary: Spill) and the end of spill signals in the COMPASS experiment. When a *Survey* starts its job it asks every element that the survey is responsible for, to update the read-back values. This request then triggers the field-bus request in the *StructuredDevice* instance, as described above. If everything goes well the next element in the chain is asked to update its values. If there was a problem detected by the *StructuredDevice* error-handling scheme the survey triggers a second request. If this request fails again the *StructuredDevice* element[18] is taken off the list of elements of the survey and it is put onto another list – the list of bad elements. After a survey finishes its job it checks if the end of survey signal has already arrived or not. If it did not arrive yet then the elements on the bad element list are treated one by one. If an element now responds correctly to the request, it is put on the good element list again, otherwise it is left on the bad list.

With this behaviour we ensure that the good elements keep their timing requirements as good as possible while the bad elements are treated at the end after all the good ones. The implemented strategy is to first ensure good communication (by checking a bad answer a second time) and only then ensure the timing requirements. So if the configured time interval for a survey is close to the required time to read every element under normal conditions, it may well happen, that in the case of problems the survey is stopped before all elements were handled. But at the next survey cycle the elements not handled will be treated first, so that all elements on the survey will be updated even if the timing is not kept.

Elements that are taken off the good element list will be automatically restored after they become available again. Even if the survey cycle stops before all the good elements were handled the *Survey* logic will check at least one bad element.

Consider an example: After a power cut the computer that runs SLiC starts automatically without any user interaction. SLiC then starts also and reads its configuration file. But the high voltage crates for example might need user interaction to be switched on again. As soon as a user switches the crates on again they are automatically reintroduced in the good element list in the surveys and the software takes care of them. The user in front of the DCS will be notified that a piece of hardware is not available by a colour code indicating the state of the piece of hardware.

5.9 The final product

The final product will run day and night for the whole duration of the experiment and only be stopped for configuration updates. Therefore it has to work as automated as possible and with as little user interaction as possible. After rebooting a processing unit the software should start automatically and should handle the internal configuration on its own. With all the above described software infrastructure we are able to run SLiC

[18] It is not a *StructuredDevice* element on the list of the survey, but some other element, which then refers to a *StructuredDevice* element. But as this other element type is not introduced in this document I also do not mention it here.

unattended as long as we do not change configuration files. Another important feature, especially in the long run, is the intelligent log handling, which ensures small log files, but also allows us to have full debug output and insight into the software's job in case of a problem.

6 PVSS and the JCOP framework in COMPASS

COMPASS decided to use the PVSS II SCADA-system. To reduce the amount of development work to implement the tailored control system for COMPASS in PVSS it was decided to use the JCOP (Joint Controls Project [90]) framework [91] with some modifications. The JCOP framework is developed by ITCO (IT - Information Technology - COntrols Group [89]) together with the LHC experiments at CERN as a common base for all the LHC experiments. Therefore it is the smallest common denominator which is useful for all the LHC experiments. For COMPASS it was necessary to extend the base to get a fully working control system. These extensions are bundled in the COMPASS framework, which was recently renamed into "Additional JCOP Framework Components" [92]. The frameworks add to the PVSS II base system new data point types, data points, panels and scripts as well as guidelines for further development (like a style-guide). Figure 21 visualises the relations between core PVSS functionality, the frameworks and the final DCS part on top of PVSS.

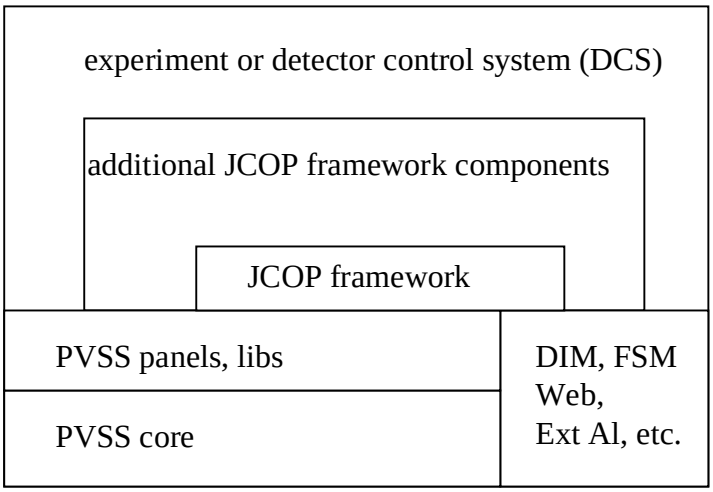

Figure 21: Layers of software on top of PVSS.

The functionality of the frameworks can be divided into a configuration part and a runtime part. The configuration part is used by the detector experts and DCS designers to modify the data representation of the experiment. The runtime part is used by the operators of the detectors during the run. Besides these frameworks a common and central system configuration mechanism is needed to configure the SCADA system and all other pieces of software that need configuration information. But up to now no common decision was met. Therefore we implemented a preliminary approach with a Filemaker data store and some Perl scripts to convert between different configuration file formats (see reference [93]).

6.1 Working with PVSS

Generally before starting with a software project one should have a clear idea of what the software is supposed to do. Therefore it is essential to talk with the later users of the software, the physicists and detector experts, to gather requirements. After this first step an experienced IT (Information Technology) professional should convert these ideas into strategies and rule sets that the software should implement.

The work with PVSS can be divided into three parts. The starting point is the design of a data representation of the detector and all the elements in it, no matter if the elements

are real hardware elements or just logical entities, which are used to facilitate the work with the detector, like groupings, trees, etc. The following section "The data point concept" will explain the technical details of the design step. After the design phase the next two steps are to implement all your strategies and rules that work on the data representation in the PVSS scripting language Control (see *PVSS Control scripting language* in section 6.1.2) and to create a user interface as interaction point between the user and your scripts.

6.1.1 The data point concept

A *PVSS data point (DP)* is a tree structure of *PVSS data point element (DPE)* objects. DPEs in turn are a collection of so called *configs*. These configs are a predefined and fixed set of interfaces that the DPE should support. The relation between the DPEs and the configs can be compared with multiple inheritance in an OO language like C++. The configs can be seen as a predefined set of classes, which you use to inherit from. A DPE is then the sum of all the capabilities implemented in the set of configs it "inherits" from. The grouping of DPEs into a hierarchy, which is then, called a data point is just used to keep things together that logically belong together (which is open to interpretation). All the DPs with the same hierarchy of DPEs are said to be of the same *PVSS data point type (DPT)*. Before you can instantiate a certain DP of a certain DPT you have to define its DPT first. In the current version of PVSS II it is possible to define DPTs dynamically at runtime by script.

Amongst the group of configs that are available there are

- the *value config*, which stores the value the DPE should represent,
- the *archiving config*, which controls archiving details and
- the *address config*, which enables the transparent network connection of the DPE[19].

PVSS stores all DP information with a timestamp inside its own database, which is optimised for data archiving. This also means that the later retrieval of values out of the database might be a slow process.

6.1.2 The Control scripting language

PVSS comes with an integrated C-like scripting language called *Control*. The user does not have to care about memory management as Control uses garbage collection. It supports basic data types like integers, floats and strings and dynamic-length arrays of these types. Control does not support any kind of structured data, even not multi dimensional (greater than two-dimensional) arrays. For these purposes you always have to introduce a new DPT that has the wanted structure. Control is a pure imperative language that builds on the use of functions like C does. It has no built in concept of exception handling, so that you have to revert to the same error handling techniques you also use in C. The language is either executed in the context of the PVSS UI manager or in a standalone thread of execution in the PVSS Control manager. PVSS already provides a set of built in functions in the areas:

- ActiveX functions,
- User administration,
- File functions,
- Database functions,
- DP functions,
- Dynamic arrays,

[19]Not every DPE is connected to a remote data entity over the network.

- Error functions,
- Functions for bit32 variables,
- Mathematical functions,
- Multilingual functionality,
- Message handling,
- GUI manipulation,
- Strings,
- Threads,
- Administration of managers, modules and panels,
- Time functions and
- Miscellaneous functions.

Not all of the provided functions are applicable in the UI manager and in the Control manager. For example ActiveX handling or GUI manipulation is obviously not possible in the Control manager, as there is no user interface attached to it.

As a positive side effect of the similarity between Control and C it is possible to use documentation systems like doc++ [84] that were intended for use with C or C++.

Control scripts can be either associated with a UI panel or they can be put into Control libraries. The functions in a Control library are accessible from all panels, whereas the functions in a panel are only accessible inside the user gesture handling code, which defines the function. Control enables the user to register global variables in the system with the Control script function `addGlobal()`, which then are accessible from every script.

All in all for a C programmer it should be straightforward to jump into PVSS script development.

6.1.3 The UI builder

If you have used one UI builder you know all UI builders (with some positive exceptions like the Visual Age series of IBM). Here PVSS is no exception and it allows the user to create panels with common UI elements like combo boxes, tables, buttons, text label etc. Besides that you can also do some drawings on the panel and illustrate its intended functionality in a graphical way. All these UI elements can be altered at runtime by script, but it is not possible to create UI elements dynamically at runtime. The most dynamic behaviour you can get is by defining some set of reference panels and then add them at runtime to a parent panel by script. A *reference panel* is a panel that may have unresolved parameters in it and serves as a pattern, which you can use as a building block to create more complex panels. These parameters must be resolved when the panel is inserted into another panel. You also have to use the UI builder to create user gesture handling scripts like button clicks. The disadvantage here is that you cannot use standard tools like a versioning system or the documentation tools mentioned above, because the script code will be stored in a PVSS internal format along with the graphics information for the panel. Therefore my personal approach is to just use the UI scripts for calling a library function and then doing all other coding inside of libraries. The disadvantage is that all libraries are held in memory and therefore more memory is permanently used. Also execution speed decreases a bit.

6.2 Below the User Interface

In this section we will have a look at the general functionality inside the frameworks below the user interface.

6.2.1 Internal data representation

How is a detector organised in terms of the PVSS DPs? The whole experiment is represented as a tree with cross-references (see figure 22). The actual data is kept in a strict tree, called the hardware tree. But to allow for different views of the same data there are also logical trees. These logical trees end at their leaf elements in pointers to real data elements inside the hardware tree.

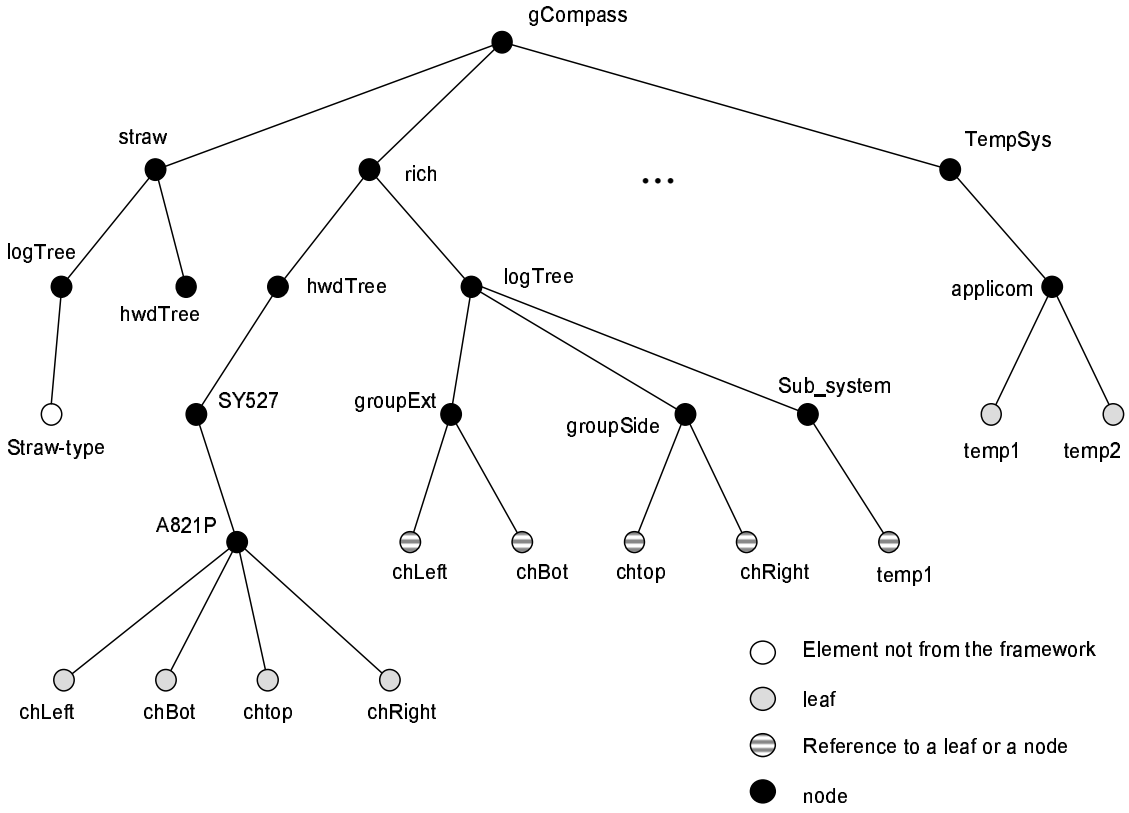

Figure 22: Data point hierarchy in the JCOP framework.

Why is this useful? In a experiment several detectors may share the same high voltage crate. Now to allow for the detector people to find their channels easily without browsing the whole hardware structure and selecting their channels out of the hundreds of channels in one crate it is easier to group these channels in a logical group, which is located in the detector tree. Besides that inside only one detector there is the need for grouping to allow all channels of a certain type of functionality to be switched on or off or to be set to a certain voltage level simultaneously. Another grouping might consist of all channels in one building block of a detector like a GEM cell out of the overall GEM detector. These logical trees can somehow be compared with views in databases. As in databases these views are generally useful, because people are interested in different details of the available information. Like in databases views might also be used for the access control of the hardware elements.

6.2.2 Creating elements

Creating elements, like a detector, a high voltage crate, a temperature sensor or any other detector component, means instantiating a DP of a certain DPT and initialising the DP to meaningful values. This initialisation process includes also the interconnection of parent and child elements in the data point hierarchies. To make this process more convenient

for the DCS designers there are the configuration panels to facilitate this job. With these panels and the underlying scripts it is possible to create the elements and initialise them to the wanted values. Furthermore in one of the future releases of the JCOP frameworks it will be possible to use such elements as patterns to create similar instances of the same DPT.

6.2.3 Network connections

As mentioned in section 3 (see *PVSS internal architecture* in section 3.5 on page 38) the network connections are handled transparently for the user through the PVSS driver layer. Each DPE that corresponds to some hardware value[20] has an address config (see *PVSS address config* in section 6.1.1 on page 55) associated with it that stores the information about which network interface to use. In the DIM protocol a unique name and the address of a DIM Name Server (DNS) identify for example every element.

6.2.4 Data archiving

The data archiving of values of DPEs in PVSS is configured by the archiving config. The archiving is independent of the fact if the DPE is connected to a real hardware element over an address config or not. Therefore all computed values can be put into an archive also. In the current version of PVSS II, which is in the moment 2.11.1, DPEs are associated to archiving groups. Then "a rectangle" of disk space is allocated where the values will be stored. Rectangle means that you have $n \cdot m$ "memory places" allocated at once, where n is the number of DPE values in the archiving group and m is the number of data elements that can be inserted until a new archiving rectangle is allocated. This procedure ensures a fast archiving behaviour but it also means that there might be a lot of disk space wasted in an archiving rectangle. If for example one of the values in the archiving group is updated every second and the others are only updated every minute then a new archive rectangle will already be instantiated as soon as the element that changes at a high frequency has changed m times and $(n-1) \cdot m \cdot (1-1/60)$ memory places are wasted. The configurer of the archiving has to take care of this by putting only elements into an archiving group that change at a similar frequency.

6.2.5 Alarm handling concept

As the different LHC experiments use different alarm handling schemes there is no common alarm handling functionality in the JCOP framework. The alarm handling scheme described here is an extension to the JCOP framework and can be found in the additional JCOP framework components (see glossary: Additional JCOP Framework Components: Alarm). The alarm handling is based on the data point hierarchy described above in section 6.2.1. There are two types of alarms – there are alarms on DPEs and there are summary alarms.

Initially you define the cases in which an alarm should be raised on a leaf data point's DPE. The DPE of leaf DPs can either be in one of 5 levels of alarm state or they can be invalid or they can be in good state (which means no alarm).

Now, for a fast diagnostic, it might be interesting to know in which branch of a tree an alarm was raised. Therefore, the additional component implementation of the JCOP framework allows the alarms to flow up to the top of the hierarchy by using the PVSS summary alarm mechanism. A node of a tree can have a summary alarm of all or of a part of its children. By configuring the summary alarm of all the nodes of a tree to be

[20] Please not that there are also DPEs that do not represent a hardware value!

the summary of their children alarms, any alarms will be propagated up to the top of the hierarchy.

The alarm state of an element, which is either a leaf DP or a summary alarm element, is visualised by colours. Under normal conditions the operator surveys just the root element of the hierarchy and as soon as this element indicates a problem the operator can locate the source of the problem in one of the sub-trees by following the "colour trace" of the problem. From there on he can either repair the problem himself or call a detector expert.

Besides the alarms described above there are two other flags,

- the masked flag (see glossary: Additional JCOP Framework Components: Mask an Alarm) and
- the disabled flag.

If an element is masked it still receives data from the hardware and does archiving but it does not propagate the alarms to the top anymore. This behaviour is useful for example, if you have a high voltage channel for which you have defined a set point and some tolerance band above and below this set point. If you now want to ramp up the voltage to a new value outside of this tolerance band then the voltage will leave the tolerance band and an alarm will be triggered. To avoid this "false alarm" the element is first masked before the ramping action takes place and then it is unmasked again after the ramping process has finished and the tolerance band has been recalculated. The disabled flag can be used to disable an element's network connection, so that its value is not updated anymore.

6.2.6 Access control

The procedure for access control (see glossary: Access Control) has still not been decided. But the direction seems to be a two level access control per detector. Either a user is supposed to be an expert of the detector, in which case he has full control over all settings. Or the user is in the normal user group, in which case he can just watch the read back values and is not allowed to change anything.

6.3 The User Interface

In the JCOP framework and in the additional framework components there are already a lot of panels defined that can be used either directly or they can be easily adapted to the special needs of a detector. As already mentioned above the panels are divided into configuration panels and runtime panels. The organisation of the panels for runtime and configuration panels follows the data point structure as outlined by the frameworks (a tree with cross references). The central access point in both cases is a panel called the detector explorer.

I should also mention for completeness that even if one decides to renounce on the use of the framework panels, like the detector explorer that the JCOP frameworks are still a rich source of reusable software components. The JCOP frameworks are so modular that even if you decide to not use their main user interface structure they still provide a set of components (reference panels, scripts, etc.) which can be generally useful and which can safe a lot of development time.

6.3.1 The detector explorer

The detector explorer panel is similar to the windows explorer, where we have on the left hand side a graphical representation of the hierarchy and on the right hand side the special panel to interact with the element we just investigate.

The detector explorer is the same for runtime and configuration panels. At runtime you explore the data gathered by the experiment and you can navigate around the whole experiment. At configuration time you explore the configuration for the experimental set up and you can create, modify, copy or delete elements from the detector's data representation. In the future it is foreseen to allow the user to duplicate elements to easily recreate similar data elements.

The approach of having just one interaction panel, the detector explorer, instead of having numerous independent panels that pop up new panels on demand is favourable, because it prevents the user interface from being cluttered with amounts of panels where the user has to find his way through. Besides that this approach imposes already a lot of design guidelines on the detector groups that develop their own panels in a natural way, so that a common style for the panels is kept, automatically. This common style is essential for the user to keep the overview in the DCS, which will be developed by a number of different people with different ideas about the look and feel of their panels.

6.3.2 Plugging new elements into the detector explorer

There is a general interface on how to plug in custom panels into the detector explorer. The exact procedure on how to do this is described in "Adding a device in the framework" [98], but to give you an impression on the complexity of this process I enumerate here the steps needed to do this. The process of adding a new device to the framework is based on panel templates and a Control script template. After receiving these templates and unpacking them the framework extender has to

- create a new data point type to represent the new device,
- allocate a new view[21] to store the new devices in,
- replace certain strings in the template files to give the elements meaningful names and avoid name clashes with other framework elements,
- modify the template panels to have the exact configuration and operation behaviour which is required, and
- extend certain `switch(){...}` statements in framework scripts to reflect the behaviour of the new data point type.

All in all it is a straightforward process where you also can build upon the many examples of elements provided by the frameworks.

[21] A view is a framework term, which is explained in the framework documentation [96].

7 The system in action

The development time for the frameworks is now nearly one year and for the SLiC software nearly half a year. During the development there was a permanent test set up in a lab to develop under lab conditions. Besides that there were several possibilities to try the software under real world conditions, too.

At the end of April there was a test beam for the GEM and silicons detectors. At this test beam the complete DCS chain was installed to get some user feedback and test results.

Since April the HARP[22] experiment has several tests with a number of high voltage channels running. They use five SY403 crates with a total of 275 channels and four SY527 crates with a total of 584 channels. The final number of channels will be slightly higher, as the system will be extended in the future. HARP uses an older version of the JCOP framework running on PVSS II version 2.10. But this set up gives already a first impression of the attended efficiency and reliability and it is a very good test ground.

Because COMPASS has not started yet there are no results about the behaviour of the software in the complete COMPASS set up.

As the results in the above test scenarios showed the performance bottle neck are the CAEN high voltage crates, where the response time over the CAENET network lies in the order of several milli seconds, whereas an estimation of the maximum transmission time of a 30 word network data packet (all transmissions are below 30 words) over the 1 MHz network would give a time value in the order of 0.5 ms. Therefore the runtime performance is as good as the crates can get.

There is still a problem with the start up time, which lies in the order of 20 minutes for 500 high voltage channels. This start up time is due to the registration of all the published DIM services and DIM commands at the DIM Name Server. If we start the software and do not publish the data over the network the software does still the same job, but the start up time is reduced to below one minute. Several solutions are under study today and it is not clear yet, which one is the preferred one.

The software uses roughly 50MB per 600 high voltage channels. This is mainly due to the heavy use of hash maps even where simple linked list would be sufficient also. This is the traditional memory to speed trade off. But because there is no need for so much speed (because the software is as fast as the hardware can get) there are possibilities to reduce the memory consumption.

Due to HARP's 24h test we can also expect a stable and reliable behaviour in the long run.

8 Conclusion

The development of the frameworks clearly facilitates the implementation of system control software for COMPASS and the frameworks provide a common structure that allows the development of a coherent and homogeneous system by multiple and remote teams. The frameworks try to hide the underlying tools as much as possible from the user, thus reducing the amount of training and support required. They are a source of as far as possible configured components (e.g. templates, standard elements and standard functionality), which are required for an experiment control system. Besides the benefits for COMPASS the frameworks also help other experiments (like HARP) to reduce the amount of development work for their control system.

[22] Hadron Production for the Neutrino Factory and for the Atmospheric Neutrino Flux [7]

Besides all the positive effects of the frameworks it is also important to remember that the frameworks are no answer to all problems. For example a solution for the access control mechanism has not yet been addressed at all and because the participating experiments in the JCOP project could not agree on a common configuration scheme there is no common top down configuration scheme integrated in the frameworks. Besides these two general problems there is still a lot of work for the specialisation of the framework components for all of the detector types in the COMPASS experiment.

Acknowledgements

My diploma thesis would not have been possible without the contributions and support of many friends and colleagues from Chair for Experimental Physics E18 at TU Munich and from IT - Information Technology - COntrols Group [89] at CERN.

I would first like to thank Mark Beharrell, Hervé Milcent and Lars Schmitt, who supported my work in all regards. Their professional and friendly way of helping me with answers to questions and suggestions helped me to find the appropriate solutions for the problems that appeared during the software development.

I would like to thank Prof. Stephan Paul who made this work possible in the first place. He unbureaucratically supported me where needed and introduced me to a great research group.

I would like to thank all the domain experts who repeatedly answered my questions concerning the set up of the COMPASS experiment and the physics behind it.

I am very grateful to the proofreaders for their efforts in making this text a bit more correct and readable.

A big thanks goes to my wife and my children for their love and trust in me during the past year. Without their support my work would not have been possible.

Appendix A Own contributions

Members of the JCOP working group and members of the ITCO group at CERN originated the initial ideas of the JCOP frameworks and the SLiC framework. The ideas on how to implement the specialised slow control software for the GEM and silicon detectors grew in an iterative process of implementing software and getting the feed back from the involved physics groups. Therefore, the requirements for my work were set from outside and I concentrated on the research on implementation strategies and the actual implementation work.

As I arrived at CERN it was assumed that my work would only take place on top of the PVSS II SCADA-system. Therefore, I started my development work on top of the by then available PVSS framework for COMPASS. I noticed that there was no means to handle large amounts of configuration data in that framework. There was only the inflexible PVSSascii manager (part of the PVSS distribution), which could dump PVSS databases and import dumped version of databases. I started to implement an experimental data point manipulation language, which could be used to configure the PVSS system database in a flexible way. It was designed to be extendable to cope with new requirements as they arise.

After some weeks it was recognised that a framework for device control on front-end computers was also needed I stopped my PVSS work and joined the SLiC development team (two people). The only remaining parts of my initial PVSS work in the actual JCOP frameworks are the exception handling scheme and the documentation system doc++ [84] for the Control script libraries. There is nothing comparable to this data point manipulation language in the latest release of the frameworks, but nevertheless my initial work on this subject showes that the approach to implement this language on top of the PVSS scripting language Control would be too slow to be usable. Therefore, if in the future one tries to build a configuration language the better approach would be to implement it as an extension to PVSS in C or C++.

The SLiC development started with a design and prototyping phase, where I implemented one prototype myself. Then the ideas from the prototypes were merged and the final SLiC design was produced. In the actual SLiC project I took the responsibility (initial implementation or set up and maintenance) for the CVS repository, for the build process structure, for the deployment of the improving versions of the front-end software, the make file implementation, the doxygen documentation tool and all the perl scripts used for make file support, intelligent logging and automated conversion of configuration data. Furthermore I did the evaluation for a thread safe replacement for libstdc++ and did the code change to exchange all the vulnerable containers with the new ACE (Adaptive Communication Environment [73]) containers. I also implemented a C++ template for a double linked list (*DLList*) from scratch. From the beginning of the project I promoted the use of debugging libraries to detect memory leaks and do intelligent debug output. After we found out that libcwd was not thread safe and libnjamd was using by far too much memory I did the evaluation of insure++ and the later integration of that tool into our project. For the intelligent debug output I implemented the quite powerful *LogObject* (see *LogObject* in section 5.5 on page 48) C++ class on top of basic ACE functionality and added the overall logging scheme to our project. With the help of insure++ I tracked down several memory leak problems (sometimes even not in our code, but in some external libraries) and an especially tricky problem, where our code used by far too much memory. I did the HARP bug report support and later bug fixing. Besides the just described work for the basic infrastructure of our project I also contributed to the actual functionality of SLiC. There, I did the implementations of the *StructuredDevice* (see *StructuredDevice* in section 5.7 on page 50) and all its general use subclasses plus the implementation of the

A303 ISA card CAENET access, the MOD V. 288 VME card CAENET access and the SY527 high voltage crate, board and channels implementations on top of these general use *StructuredDevice* subclasses. After we got our first external developer I did the necessary support to get him started with his development work on top of the SLiC framework and I wrote some code to make the integration of his work into the SLiC front-end application easier. Besides the pure development work, there were many hours of test runs with different hardware set ups to improve the SLiC front-end application to the level of a software that can be used in the final production environment (HARP, COMPASS, etc.).

After these tests were finished and SLiC entered into its stable state (as it is now) I again started development on top of PVSS and on top of the now available JCOP frameworks. This work is still going on and it consists mainly of writing specialisations to the JCOP framework components for the GEM and silicon detectors. After the specialisations will be finished they have to be integrate into the overall COMPASS DCS set up. One part of the specialisation work, which is a main change to the framework implementations, is to integrate user access control into the GEM and silicon components.

Glossary

Absorption Length The mean free path (see **Mean Free Path**) of a particle before undergoing a non-elastic interaction in a given medium. The relevant cross-section is $\sigma_{\text{tot}} - \sigma_{\text{el}}$. (see also **Collision Length** and **Interaction Length**)

Abstraction Barrier In general, the underlying idea of data abstraction is to identify for each type of data object a basic set of operations in terms of which all manipulations of data objects of that type will be expressed, and then to use only those operations in manipulating the data.

One can imagine abstraction barriers as "horizontal lines" in a program that isolate different "levels" of the system. At each level, the barrier separates the programs (above) that use the data abstraction from the programs (below) that implement the data abstraction.

Acceptance We define the acceptance a of an experiment as the average detection efficiency. Frequently, the word is also used in the more restricted sense of geometric acceptance defined below.

Let N be the total number of events that occurred, out of which n are observed. Then the expectation values of N and n are related by

$$E(n) = aE(N)$$

One may consider the acceptance as a function of one or more variables, or in a small region of phase space.

By this general definition, the acceptance includes all effects that cause losses of events: the finite size of detectors, the inefficiencies of detectors and of off-line event reconstruction, dead times, effects of veto counters, etc.

Let $x = (x_1, .., x_D)$ be the physical variables that describe an event, such as the momenta of the particles, positions of interaction vertices, and possibly also discrete variables like the number of particles, spin components, etc. These are random variables following a probability distribution

$$f(x)\mathrm{d}^D x = \frac{F(x)\mathrm{d}^D x}{\int_\Omega F(x)\mathrm{d}^D x}$$

Ω is the allowed region for x, and the integral includes a sum over discrete variables. The non-normalised density $F(x)$ is given by the experimental conditions, i.e. beam, target, etc., and is proportional to the differential cross-section. For a sufficiently small phase space region the differential cross-section is nearly constant and hence drops out from the normalised probability density $f(x)$.

Let $\epsilon(x)$ be the total detection efficiency for an event given its physical variables x. The acceptance is then the expectation value of $\epsilon(x)$,

$$a = \int_\Omega \epsilon(x) f(x) \mathrm{d}^D x.$$

If, to a sufficiently good approximation,

$$\epsilon(x) = \epsilon_g(x)\epsilon_d$$

where $\epsilon_g(x)$ is the purely geometric efficiency($\epsilon_g(x) = 1$ if the particles hit the detectors, $\epsilon_g(x) = 0$ otherwise) and ϵ_d is a constant detection efficiency, then

$$a = a_g \epsilon_d$$

$$a_g = \int_\Omega \epsilon_g(x) f(x) \mathrm{d}^D x.$$

a_g is the pure geometric acceptance.

Access Control In this report access control means limiting access to DCS resources only to authorised users, programs, processes, or other systems. Access control could be done on a per user basis, where you explicitly grant or restrict the rights of every individual user. But because such a system is a maintenance nightmare it is common practice to use user groups where rights are granted or restricted on a per group basis and subsequently the users are put into these groups.

Because access control is not implemented yet in the frameworks and up to now there is no decision about an access control strategy in COMPASS the initial step towards strong access control is just an additional layer on top of the SCADA-system which prevents accidental user mistakes.

Additional JCOP Framework Components: Alarm An alarm is a message which is generated by the control system when a piece of equipment deviates from the desired operation. Several levels and categories of alarm are possible. The PVSS term for alarm is alert.

Additional JCOP Framework Components: Acknowledge an Alarm This is an action issued from the user indicating to the control software that the user took in account the alarm, this does not always mean that he fixed the problem related to the alarm.

Additional JCOP Framework Components: Mask an Alarm To mask an alarm of a device is an action which stops the alarm evaluation for this device. The word "masked" applied to an alarm means the alarm is not evaluated. This does not prevent the data of the device to be refreshed. The PVSS masking action is the de-activation of the PVSS alert_hdl config.

Additional JCOP Framework Components: Un-Mask an Alarm To un-mask an alarm of a device is an action which starts the alarm evaluation for this device. The word "un-masked" applied to an alarm means the alarm is continuously evaluated. The PVSS un-masking action is the activation of the PVSS alert_hdl config

Additional JCOP Framework Components: Device A device is a piece of hardware or software equipment, e.g.: a SY127 CAEN crate is a device, a node of a hierarchy is a device, etc.

Ageing Ageing is generally a deterioration of detector properties like a deterioration of gain or a loss of resolution. Ageing in particular in gas detectors is often due to the use of organic gas mixtures. The organic gas mixtures tend to dissociate because of the incoming radiation. The remaining parts then either change the gas properties in the gas volume or they stick on conducting surfaces and change the electric fields. It is possible to improve the situation by using a gas current through the detector, so that the gas does not remain too long in the gas volume and has less chance to dissociate.

Annealing Annealing can be shortly summarised as the movement of defects in a crystal lattice. The mechanisms leading to annealing of defects can roughly be divided into migration, complex formation and dissociation processes. At a certain temperature the defects become mobile and migrate through the silicon lattice until they are stopped or a complex defect, composed of more than one constituent, dissociates into its components if the lattice vibrational energy is sufficient to overcome the binding energy. At least one of the constituents then migrates through the lattice.

BNC Components A family of components that include the BNC cable connector, BNC T connector, BNC barrel connector, and the BNC terminator. The origin of the acronym BNC is unclear. Names ascribed to these letters range from "British Naval Connector" to "Bayonet Neill-Councelman".

Centralised Access Point Centralised access point might be misunderstood by having only a single access point, a single computer. But what is meant here is that there is a software, that can run on as many computers as wanted and that can be used by as many operators (independently of each other: virtual single user mode) as wanted, which provides a logically centralised view of the system and hides the distributed nature of the DCS from the user. In database terms this might be summarised by the acronym ACID (Atomicity Concurrency Isolation and Durability).

Charge Collection Efficiency Is the ratio between the measured amount of charges and the deposited amount of charges (which is the result of a given deposited energy in the detector) in the detector. The charge collection efficiency is a combined effect of the semiconductor properties and the quality of the readout electronics.

Čerenkov Radiation ČERENKOV radiation is emitted whenever charged particles pass through matter with a velocity v exceeding the velocity of light in the medium,

$$v > v_t = c/n$$

with
n = refractive index of the medium,
c = velocity of light in vacuum,
v_t = threshold velocity.
The charged particles polarise the molecules, which then turn back rapidly to their ground state, emitting prompt radiation. The emitted light forms a coherent wavefront if $v > v_t$. ČERENKOV light is emitted under a constant ČERENKOV angle δ with the particle trajectory, given by

$$\cos\delta = v_t/v = c/(vn) = 1/(\beta n) = \frac{\beta_t}{\beta}.$$

and for the threshold
$\beta_t = v_t/c = 1/n$,
$\gamma_t = n/\sqrt{n^2-1}$.
The maximum emission angle is given by

$$\cos\delta_{\max} = 1/n = \beta_t \quad (\text{for } v = c).$$

The major problem of ČERENKOV radiation is the modest light output. The energy loss due to ionisation or excitation is two to three orders of magnitude higher than the energy lost in radiating ČERENKOV light in the energy range where photo multipliers can be used (a few eV, or about 400 nm wavelength). By its directionality, ČERENKOV light can, however, be separated from the background.

Collision Length The mean free path (see **Mean Free Path**) of a particle before undergoing a nuclear reaction, for a given particle in a given medium. The collision length (also known as the nuclear collision length) follows from the total nuclear cross-section σ_T by

$$\lambda_T = A/(\sigma_T N_A \rho)$$

(see **Mean Free Path** for an explanation of variables). The probability density function for distances between successive collisions is given by

$$\Phi(x)\mathrm{d}x = \frac{1}{\lambda_T} e^{-x/\lambda_T}\mathrm{d}x.$$

If one subtracts from the total cross-section the sum of elastic and quasi-elastic (diffractive) cross-sections, one obtains by the same formula the (nuclear) interaction length λ_1.

Some numerical values for λ_T and λ_1 are given in the following table.

Medium	λ_T $[cm]$	$\rho\lambda_T$ [g/ cm^2]	$\rho\lambda_1$ [g/ cm^2]
Fe	10.6	83.3	131.9
Al	26.1	70.6	106.4
Cu	9.6	85.6	134.9
Pb	10.2	116.2	193.7
Concrete	27.0	67.4	99.9

The numbers are from reference [51]. (see also **Absorption Length** and **Interaction Length**)

Creational Patterns: Factory Even if it does not match any definition of a creational pattern in [66] directly, I call our approach for creating objects in SLiC simply a factory. The purpose of a factory is to instantiate objects of a concrete sub class of an abstract type. The reasoning behind this approach is to keep the rest of the program unaware of the implementation details of concrete implementations of abstract types and therefore to reduces implementation dependencies between subsystems of the program. The core of our factory is a map of string representations of concrete object classes to a creator object that knows how to instantiate objects of that type. The factory object then is able to create objects by looking up the string representation of the object's class in the map and then forwarding the creation request to the creator object. The input to the factory object (the string representation of the object's type) is taken from a configuration source like a database or an XML file. The creator map is filled with content at program start-up by predefined initialiser functions (see *initialiser function* in section 5.3 on page 46). Every developer who implements another concrete sub class of an abstract type has to add a line to the initialiser function, which adds an appropriate creator object to the creator map.

Data-Directed Programming The general strategy of checking the type of a datum and calling an appropriate procedure is called dispatching on type. This is a powerful strategy for obtaining modularity in system design. Implementing the dispatch as `switch(){}` statements in dedicated procedures (generic interface procedure) has two significant weaknesses. One weakness is that the generic interface procedures must know about all the different representations. For instance, suppose we wanted to incorporate a new representation for complex numbers into a complex-number system. We would need to identify this new representation with a type, and then add a clause to each of the generic interface procedures to check for the new type and apply the appropriate data-access functions for that representation.

Another weakness of the `switch(){}` technique is that even though the individual representations can be designed separately, we must guarantee that no two data-access functions in the entire system have the same name.

The issue underlying both of these weaknesses is that the technique for implementing generic interfaces is not *additive*. The person implementing the generic selector procedures must modify those procedures each time a new representation is installed, and the people interfacing the individual representations must modify their code to avoid name conflicts. In each of these cases, the changes that must be made to the code are straightforward, but they must be made nonetheless, and this is a source of inconvenience and error.

What we need is a means for modularising the system design even further. This is provided by the programming technique known as *data-directed programming*. To understand how data-directed programming works, begin with the observation that whenever we deal with a set of generic operations that are common to a set of different types we are, in effect, dealing with a two-dimensional table that contains the possible operations on one axis and the possible types on the other axis. The entries in the table are the procedures that implement each operation for each type of argument presented.

Data-directed programming is the technique of designing programs to work with such a table directly. Previously, we would have implemented the mechanism as a set of procedures that each perform an explicit dispatch on type. Now we implement the interface as a single procedure that looks up the combination of the operation name and argument type in the table to find the correct procedure to apply, and then applies it to the contents of the argument. If we do this, then to add a new representation package to the system we need not change any existing procedures; we need only add new entries to the table (*additivity*).

If we use virtual methods, the just described technique of data-directed programming is done in C++ automatically by the compiler. The dispatch is done upon the data type of the object we call the method on. Therefore the technique in the above described way is no extraordinary gain. But if we have a closer look to creational design patterns in reference [66] then we find the same technique again. If, for example like in SLiC, we create all elements in the system from an xml file we have again the case of a two dimensional table, where we have on the one axis the string representation of the data type and on the other axis the create function/method. The implementation of the creation operation of all objects in our system as a table driven process ensures the additivity of development done by different groups.

But what we should still keep in mind from the above paragraphs is that `switch(){}` statements or extensive `if(){}else{}` chains always indicate a design weakness of our code, because they break additivity. Unless we are really sure that these statements just treat a fixed set of possibilities we should always use tables instead.

Detector Control System The Detector Control System is the slow control system of the whole experiment. With this term one refers to the whole system consisting of the supervision layer, the process management layer and the field management layer (see also figure 13 and **SCADA System**, **Slow Control**).

DIM concepts overview The basic concept in the DIM approach [75] is the concept of "service". Servers provide services to clients. A service is normally a set of data (of any type or size) and it is recognised by a name - "named services". The name space for services is free.

Services are normally requested by the client only once (at start-up) and they are subsequently automatically updated by the server either at regular time intervals or whenever the conditions change (according to the type of service requested by the client). The just described situation is implemented in the server (like SLiC is) as a *DIM Service*. Here the server code calls DIM library code to trigger the data transmission. On the other hand a client also has the possibility to pass information to the server by using a *DIM Command*. In this case the library code calls the server code to handle the command.

In order to allow for transparency (i.e, a client does not need to know where a server is running) as well as to allow for easy recovery from crashes and migration of servers, a name server was introduced. Servers "publish" their services and commands by registering them with the name server (normally once, at start-up). Clients "subscribe" to services/commands by asking the name server which server provides the service/command and then contacting the server directly.

Exclusive Measurement of Interactions A measurement of particle interactions in which all participating particles are identified and measured. Exclusive measurements including all particles of an interaction are usually possible only at relatively low laboratory energies and for simple interaction types like two- and three-body final states.

Field-bus A field-bus is simply a network. The only thing that distinguishes a field-bus from other networks is its use as a network in process control. E.g. in the future it is planed to use ethernet for process control and therefore it is possible to call ethernet a field-bus. Common field-buses are CAENET, CANbus or Profibus.

Fragmentation Function The probability density function of a characteristic variable describing the hadronisation of jets, e.g. longitudinal momenta of hadrons inside a quark or gluon jet. Typically, the variables used are $x_P = p_{\mathrm{had}}/p_{\mathrm{jet}}$ or $z = p_L/p_{\mathrm{jet}}$ with p_L being the hadron momentum along the jet axis. The extreme values for both variables are zero and one. The fragmentation functions measured in e^+e^- or $\bar{p}p$ interactions are characterised by a peak at zero and a fast experimental falloff towards higher values. The measured fragmentation functions show clear differences between quark and gluon jets: gluon jets have higher particle multiplicities, but their energy and energy fraction is lower: the fragmentation is softer.

Fragmentation Region The small-angle (c.m.) region of an interaction. Particles in the fragmentation region have momenta similar to the incident or target particle. Consequently one speaks about the beam fragmentation region, sometimes defined over a given range of rapidity (see glossary: Rapidity) like $y_{\max} - \Delta < y \leq y_{\max}$ where $y_{\max} = \log\sqrt{s/m}$ and $\Delta \approx 2$, or about the target fragmentation region (defined similarly for negative y).

Gas Mixtures in Gaseous Detectors Avalanche multiplication is essential in all gaseous detectors, in order to produce an electrical signal of sufficient amplitude. In principle, all gases can be used for generating electron avalanches, if the electric field near the (sense) wire is strong enough. However, depending on the mode of operation (see glossary: Operational Modes of Gaseous Detectors) and the intended use of the chambers, specific requirements towards, e.g. signal proportionality, high gain, good drift properties, or short recovery times, limit the choice of gases or gas mixtures. Multiplication occurs in noble gases at lower fields than in gases with complex molecules; the addition of other components increases the threshold voltage.

This suggests a noble gas as the main component of a chamber gas. Noble gases do not, however, allow operation at high enough gas gain without entering into a permanent discharge operation. The atoms excited during the avalanche process return to the ground state emitting photons at high enough energies to initiate a new avalanche in the gas or around the cathode. The latter may also be induced by the neutralisation of ions that travel to the cathode. This problem is solved by the addition of a *quenching gas* which absorbs energetic photons. Most organic compounds in the hydrocarbon and alcohol families are efficient in absorbing photons in the relevant energy ranges. The molecules dissipate the excess energy either by elastic collisions, or by dissociation into simpler radicals. Even a small amount of a polyatomic quencher added to a noble gas changes completely the operational characteristics of a chamber, and may allow gains in excess of 10^6 to be obtained before discharge.

A typical gas mixtures for proportional counters is P10 ($90\%\mathrm{Ar} + 10\%\mathrm{CH_4}$) and for MWPCs the "magic gas" mixture: $75\%\mathrm{Ar} + 24.5\%$ isobutane $+ 0.5\%$ freon.

Halo There is no consensus at present on a precise definition of the beam halo. Generally, the term halo describes the outer low-density edge of the beam in phase space that surrounds a dense central core.

Hodoscope The literal translation from the Greek is "pathviewer".

"High" Voltage Crate To give the reader an idea of what a power supply ("high" voltage crate) looks like and what its functionality is the CAEN SY527 power supply will be explained in more detail here.

The SY527 itself is a crate with ten empty slots for power supply boards. Before you can get voltage output from the crate you have to plug in a high voltage board, which provides several (in the order of ten) high voltage channels. To connect to such a channel you use typically a BNC connector (see **BNC Components**). The different types of boards have different characteristics concerning for example the allowed maximum voltage output or the allowed maximum current flow per channel. On the front of the crate there is a user interface for manual operation. Besides the manual operation the crate can either be operated via a text terminal or via a connection to the CAENET field-bus.

A typical field-bus action for the power supply might look like the following. A command, like "switch on channel one of board two", arrives at the crate over the CAENET. The crate processes the command and forwards it to the relevant board in the crate. The board finally does the wanted operation on the channel.

Inclusive Measurement of Interactions An inclusive measurement of a particle interaction is a partial measurement. Only a few produced particles, sometimes only one, are singled out for identification and measurement, ignoring the details of all other interaction products. Inclusive measurements dominate at high energies, where the separation of tracks and particle identification become difficult, even when using the most advanced detectors. Triggers in complex interactions are necessarily inclusive: the signature of interesting physics will be defined in terms of few phenomena, like high- leptons or jets, disregarding the rest of the interaction.

Interaction Length The mean free path (see **Mean Free Path**) of a particle before undergoing an interaction that is neither elastic nor quasi-elastic (diffractive), in a

given medium (designated as λ_1 in the glossary entry for Collision Length). The relevant cross-section is $\sigma_{\text{tot}} - \sigma_{\text{el}} - \sigma_{\text{diff}}$. (see also **Absorption Length** and **Collision Length**).

Leakage Current The current of a reverse biased diode is called leakage current or reverse current. While the reverse current of an ideal diode consists only of a diffusion current in reality impurities, contaminations and process induced defects in the semiconductor contribute to the current.

Mean Free Path The mean free path of a particle in a medium is a measure of its probability of undergoing interactions of a given kind. It is related to the cross-section corresponding to this type of interaction by the formula

$$\sigma\lambda = \Omega/N = A/(N_A\rho),$$

$$\begin{aligned} \sigma &= \text{cross-section } [\text{cm}^2], \\ \lambda &= \text{mean free path } [\text{cm}], \\ \Omega &= \text{volume of interaction,} \\ \text{with } N &= \text{number of target particles in } \Omega, \\ A &= \text{atomic weight [g/mole],} \\ N_A &= \text{Avogadro constant } (6.022 \cdot 10^{23}/\text{mole}), \\ \rho &= \text{density } [\text{g}/\text{cm}^3]. \end{aligned}$$

The mean free path is the average of a distribution of distances following an exponential law:

$$P(x)\mathrm{d}x = \frac{1}{\lambda} e^{-x/\lambda}\mathrm{d}x.$$

Tables often give the quantity $\lambda\rho = A/(\sigma N_A)$ (in g/cm^2) instead of λ (in cm). For numbers see **Collision Length**.

Operational Modes of Gaseous Detectors The charge collected by the anode of a chamber depends on the intensity of the electric field applied to the chamber. At some low voltage, the recombination of electrons and ions is overcome, but no gas multiplication occurs; a detector in this mode is insensitive to the voltage, and is called an ionisation chamber; the output signal is weak, and corresponds to the number of primary electrons.

As the voltage is increased, which happens in the fields of wire chambers, the primary ionisation electrons cause electron avalanches to form: the accelerating electric field is high enough to impart to the electrons, generated by the primary ionisation in the gas, an energy higher than the first ionisation potential of the gas. These electrons then produce ion-electron pairs while continuing along their path; the secondary electrons may, in turn, form further pairs, and the phenomenon is called gas multiplication. Eventually, the freed electrons drift towards the anode and produce an analogue signal that can be used for position and energy loss measurement. Most wire chambers work in this proportional mode, therefore the signals recorded by the detector are much higher and still proportional to the energy loss $\mathrm{d}E/\mathrm{d}x$ of the traversing particle. In most practical chambers, the electric field close to the thin (20 to 30 μm) anode wire has a high gradient, so that a multiplication factor of 105 to 106 is reached, with multiplication occurring mostly very close to the wire, where the field is strongest.

Strict proportionality assumes that space charge (due to the longer-lived positive ions) and induced effects remain negligible, compared to the external field. At higher

electric fields, or in a high flux of charged particles, the space charge (see **Space Charge**) effects alter the effective electric field, the chamber works in the mode of limited proportionality: the signal is no longer strictly proportional to the energy loss of the particle; the relation between collected charge and $\mathrm{d}E/\mathrm{d}x$ can still be put to use, though.

Further increase of the electric field eventually leads to electric breakdown of the gas. This takes place when the space charge inside the avalanche is strong enough to shield the external field. A recombination of ions then occurs, resulting in photon emission and in secondary ionisation with new avalanches beyond the initial one. If the process propagates (backwards, from the avalanche tail) until an ion column links anode and cathode, a spark discharge will eventually occur, and a chamber or counter is said to operate in the Geiger-Müller mode.

In the limited Geiger mode, this discharge is not allowed to happen, which can be achieved by adding quenching agents to the gas; output pulses at the anode are much higher in this mode than in the proportional mode. The process of spark discharge can also be stopped by manipulating the electric field: if only short (a few ns) pulses of high voltage are applied, short discharges develop from the ion trail of a crossing particle (streamers), and a track image can be obtained by photography (streamer chamber).

A similar effect as for the limited Geiger mode can be obtained using thick (50 to $100\,\mu$m, as opposed to the usual 20 to $30\,\mu$m) anode wires without using quenchers. This mode of operation, attractive because of its high mechanical reliability due to the thick wires, is called the limited streamer mode.

Programming Towards Interfaces It is important to understand the difference between an object's class and its type. An object's class defines how the object is implemented. The class defines the object's internal state and the implementation of its operations. In contrast, an object's type only refers to its interface – the set of requests to which it can respond. An object can have many types, and objects of different classes can have the same type.

C++ uses classes to specify both and object's type and its implementation. It's important to understand the difference between class inheritance and interface inheritance (or subtyping). Class inheritance defines an object's implementation in terms of another object's implementation. It's a mechanism for code and representation sharing. Interface inheritance describes when an object can be used in place of another.

The idea now is to program to an interface, not to an implementation. All subclasses can then respond to the requests in the interface of this abstract class, making them all subtypes of the abstract class. There are two benefits to manipulating objects solely in terms of the interface defined by abstract classes:

- Clients remain unaware of the specific types of objects they use, as long as the objects adhere to the interface that clients expect.
- Clients remain unaware of the classes that implement these objects. Clients only know about the abstract class(es) defining the interface.

This so greatly reduces implementation dependencies between subsystems that it leads to the following principle of reusable object-oriented design:

Program to an interface, not an implementation.

Don not declare variables to be instances of particular concrete classes. Instead, commit only on an interface defined by an abstract class. You have to instantiate classes somewhere in your system, of course, but therefore use the creational patterns described in reference [66].

Rapidity The rapidity is a variable frequently used to describe the behaviour of particles in inclusively measured reactions. It is defined by

$$y = \frac{1}{2} \log \frac{E + p_L}{E - p_L}$$

which corresponds to

$$\tanh(y) = p_L/E$$

y is the rapidity, p_L is the longitudinal momentum along the direction of the incident particle, E is the energy, both defined for a given particle. The accessible range of rapidities for a given interaction is determined by the available centre-of-mass energy and all participating particles' rest masses. One usually gives the limit for the incident particle, elastically scattered at zero angle:

$$|y_{\max}| = \log[(E + p)/m] = \log(\gamma + \gamma\beta)$$

with all variables referring to the through-going particle given in the desired frame of reference (e.g. in the centre-of-mass).

Note that $\partial y/\partial p_L = 1/E$. A Lorentz boost along the direction of the incident particle adds a constant, $\log(\gamma+\gamma\beta)$, to the rapidity. Rapidity differences, therefore, are invariant to a Lorentz boost.

Radiation Damage and Radiation Hardness The main macroscopic effects of changes in the detector performance consist of a flux proportional increase in the leakage current, a dramatic change of the depletion voltage (needed to maintain the full sensitivity of the whole detector thickness) and the damage-related decrease of the charge collection efficiency (see **Charge Collection Efficiency**).

The bulk damage produced in silicon particle detectors by hadrons or higher energetic leptons is caused primarily by displacing a Primary Knock on Atom (PKA) out of its lattice site resulting in a silicon interstitial and a left over vacancy (FRENKEL defect). Both can migrate through the silicon mono-crystalline lattice and may finally form complex point defects together with impurity atoms being resident in the silicon. However, the original PKA can only be displaced if its energy is higher than the binding energy of $\approx 25\,\mathrm{eV}$. The energy of a recoil silicon PKA or any other residual atom resulting from a nuclear reaction can of course be much higher. The further energy loss of these recoils consists of energy loss in ionisation and further displacements of other bulk atoms. At the end of any heavy recoil range the nonionising interactions are prevailing and a dense agglomeration of defects is formed. Only ionisation losses will not lead to any relevant changes in the silicon lattice. Hence the bulk damage depends exclusively on the Non Ionising Energy Loss (NIEL) and it has widely been verified that it is strictly proportional to this value (*NIEL scaling hypothesis*). It was found that there is a temperature and time dependence of the radiation damage effects like with the variation of the effective doping concentration N_{eff}. At room temperature an effect called annealing (see **Annealing**) takes place, where we divide between beneficial annealing, which "heals" partly the damage done by the

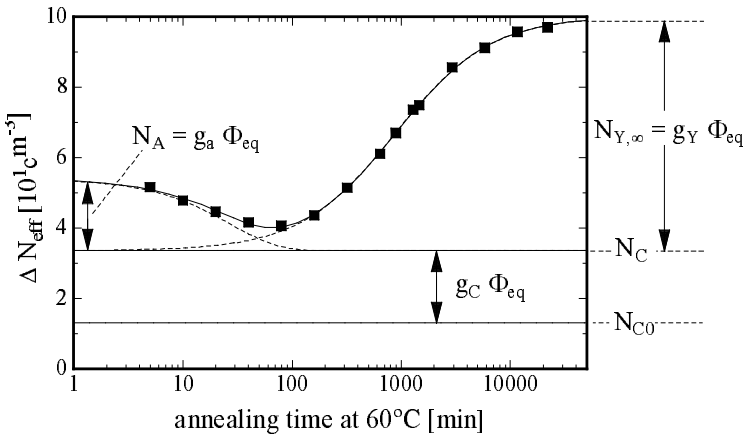

Figure 23: An example for the annealing behaviour of the radiation-induced change in the effective doping concentration ΔN_{eff} [61].

radiation, and reverse annealing, which increases the bad effects even after irradiating the material. The time dependence of the irradiation effect in N_{eff} is not only subject to beneficial annealing but also to the adverse effect, reverse annealing. An example of the whole complex behaviour is given in figure 23. Reverse annealing had only been observed after hadron irradiation. While in gamma-induced damage predominantly only point defects are being formed, hadronic interactions lead to cluster formation at the end of each high energetic PKA range. An obvious attempt for explaining the reverse annealing behaviour is therefore the assumption that these clusters will dissolve on a large time scale and the evading interstitials and/or vacancies would then migrate through the silicon lattice and form point defects responsible for the observed increase in the space charge (see **Space Charge**). But up to now no direct proof of such a correlation has been found.

Some possible improvements for radiation tolerance can be found in modifications of

- Process technology. The use of a quite elaborate guard ring structure had been shown to result in quite appreciable improvements. The guard ring structure shields the sensitive area from surface and edge leakage currents, but also provides a controlled, gradual drop of the potential from the detector rim towards the potential of the undepleted substrate. With this approach voltages of several hundred volts can safely be applied to achieve full depletion even in a radiation damaged detector.
- Operational conditions. Cooling of the detectors is essential to keep the reverse current at a tolerable level. For the ATLAS-SCT detectors an operating temperature of $-7°\text{C}$ is foreseen. This cooling has the additional beneficial effect, as the reverse annealing remains largely frozen. It was found that by cooling a detector even further to cryogenic temperature (e.g. 77 K) the radiation effects are frozen out. This effect has most recently been named *Lazarus effect* [58]. With decreasing temperature more and more defects are not able to re-emit the captured carriers back into the bands. The carriers are frozen in the defects and compensate the shallow doping. The free carrier concentration decreases and, if a voltage is applied, the observed absolute space charge density ($\propto$ depletion voltage) is smaller than at higher temperatures. Cooled to a cryogenic temperature the concentration of electron and hole traps is always high enough

to capture all free carriers. Consequently there is also no undepleted bulk and applying a voltage results in an electric field reaching through the whole detector. Thus the charge collection efficiency is increased compared to the high temperature operation of a not fully depleted detector.

- Choice of starting material. Several possibilities for increasing the radiation tolerance of silicon detectors have been proposed by using a specific type as starting material. These include the proper choice of resistivity (doping concentration), a modification of the carbon and oxygen concentration as being the main responsible impurities in damage kinetics or even to enrich silicon with Sn, which has been shown to act as vacancy trap. From all the experimental evidence [59] obtained so far as well as by theoretical considerations the oxygen content seems to be a key factor for an improved radiation hardness.

Besides improving the properties of silicon there is also research activity in finding other semi-conductor materials with initially better radiation hardness. The possible replacement candidates under investigation are germanium, gallium arsenide and diamond, where diamond has the most potential in replacing silicon. With a band gap of 5.5 eV diamond is an insulator with only a very small carrier concentration. Consequently diamond has not to be depleted and there is no need for a diode structure to build a detector. Furthermore it shows an excellent radiation hardness. However, the high energy needed to produce an e-h pair (13 eV) and the polycrystalline structure results in a relatively small signal compared to silicon. Furthermore the inhomogeneity of the material leads to non-uniformity of charge collection efficiency over the detector area which might limit the achievable resolution of position sensitive devices.

SCADA System SCADA stands for Supervisory Control And Data Acquisition. As the name indicates, it is not a full control system, but rather focuses on the supervisory level. As such, it is a pure software package that is positioned on top of hardware to which it is interfaced via PLCs for example.

Semi-Inclusive Measurement of Interactions A measurement of the scattered beam particle in coincidence with a secondary particle (see also **Inclusive Measurement of Interactions** and **Exclusive Measurement of Interactions**).

Slow Control The term refers to all the measurements and control actions that have to take place on a time scale of seconds to tens of seconds. If there exists the term Slow Control one also expects a "Fast Control". The fast control is the actual data taking of the experiment which happens on a time scale of μs.

Space Charge In a proportional or drift chamber, positive ions are released during the amplification process. While the negative charges (electrons) are collected on the anode wire, the ions drift slowly towards the cathode. In normal operational conditions, the charge density in the electron-ion avalanche is small compared to the charge density on the wire, and only small signal distortions occur. For chambers exposed to high-intensity beams, they may accumulate, and then produce space charge effects, in particular distortions of the effective electric field, causing inefficiencies of the chamber and thus influencing acceptance and calibration parameters of the chamber (e.g. causing a non-linear relation between the collected charge and $\mathrm{d}E/\mathrm{d}x$)

Spill The SPS does not produce a continuous beam, but packets. One such packet is called spill.

Static Initialiser Object A static initialiser object is a class declaration and an object instantiation of that type inside an anonymous namespace (to avoid name clashes with other code pieces). The sole purpose of this object is its constructor call at program start up (before the `main(){}` function is called). Inside this constructor it would be possible to add creator objects for certain implementations of interfaces into the creator list of the factory (see **Creational Patterns: Factory**) objects one uses.

VME VMEbus is a computer architecture. The term VME stands for VERSAmodule Eurocard and was first coined in 1980 by the group of manufacturers who defined it. The term bus is a generic term describing a computer data path, hence the name VMEbus.

Index

References

[1] G. Baum *et al.* [COMPASS Collaboration],
"COMPASS: A Proposal for a Common Muon and Proton Apparatus for Structure and Spectroscopy,"
CERN-SPSLC-96-14, `http://wwwcompass.cern.ch`

[2] Yu. A. Alexandrov *et al.*,
"CHEOPS: CHarm Experiment with Omni-Purpose Setup,"
CERN-SPSLC-95-22.

[3] E. Nappi *et al.*,
"Letter of Intent: Semi-inclusive Muon Scattering from a Polarized Target,"
CERN-SPSLC-95-27.

[4] V. W. Hughes *et al.* [Spin Muon Collaboration (SMC)],
"Measurement Of The Spin Dependent Structure Functions Of The Neutron And Proton,"
CERN-SPSC-88-47.

[5] A. Forino *et al.*,
"Proposal for a new hyperon beam experiment at the CERN SPS using the Omega facility,"
CERN-SPSC-87-43.

[6] J. P. Peigneux *et al.*,
"A Search for centrally produced non $q\ \bar{q}$ mesons in $p\ p$ interactions at $450\,\mathrm{GeV}/c$ using the CERN Omega spectrometer and GAMS-4000: Proposal (Extension of the WA91 and NA12/2 experiments),"
CERN-SPSLC-94-22.

[7] M. G. Catanesi *et al.*,
"Proposal to study hadron production for the neutrino factory and for the atmospheric neutrino flux,"
CERN-SPSC-99-35, `http://harp.web.cern.ch/harp`

[8] E. Hartouni *et al.*,
"HERA-B: An experiment to study CP violation in the B system using an internal target at the HERA proton ring. Design report,"
DESY-PRC-95-01.

[9] Gerhard K. Mallot,
"The spin structure of the nucleon,"
Habilitationsschrift, Johannes Gutenberg-Universität Mainz, August 1996.

[10] F. Bradamante,
"The gluon contribution to the nucleon spin and the COMPASS experiment at CERN,"
Prog. Part. Nucl. Phys. **44** (2000) 339.

[11] M. Schierloh,
"Einsatz programmierbarer Logikbausteine in der COMPASS-Ausleseelektronik,"
Diplomarbeit, Albert-Ludwigs-Universität Freiburg, December 1998.

[12] M. Niebuhr,
"Entwicklung eines 250- MHz-Zählers mit totzeitfreier Auslese für das COMPASS-Experiment,"
Diplomarbeit, Albert-Ludwigs-Universität Freiburg, November 2000.

[13] G. K. Mallot,
"Accessing the gluon polarisation in deep inelastic muon scattering,"
Prog. Part. Nucl. Phys. **36** (1996) 39.

[14] B. Adeva *et al.* [Spin Muon Collaboration],
"Spin asymmetries A_1 of the proton and the deuteron in the low x and low Q^2 region from polarized high energy muon scattering,"
Phys. Rev. **D 60** (1999) 072004.

[15] S. J. Brodsky, J. Ellis and M. Karliner,
"Chiral Symmetry And The Spin Of The Proton,"
Phys. Lett. B **206** (1988) 309.

[16] F. Bradamante,
"Highlights from the SMC and COMPASS experiments,"
`http://wwwcompass.cern.ch/compass/publications/ps/spin2000_franco.ps.gz`

[17] S. J. Brodsky, M. Burkardt and I. Schmidt,
"Perturbative QCD constraints on the shape of polarized quark and gluon distributions,"
Nucl. Phys. **B441** (1995) 197 [hep-ph/9401328].

[18] A. D. Martin, R. G. Roberts, W. J. Stirling and R. S. Thorne,
"Parton distributions: A new global analysis,"
Eur. Phys. J. **C4** (1998) 463 [hep-ph/9803445].

[19] R. L. Jaffe and X. Ji,
"Novel quark fragmentation functions and the nucleon's transversity distribution,"
Phys. Rev. Lett. **71** (1993) 2547 [hep-ph/9307329].

[20] A. V. Efremov and O. V. Teryaev,
JINR Dubna preprint E2-88-287 (unpublished) (1988).

[21] G. Altarelli and G. G. Ross,
"The Anomalous Gluon Contribution To Polarized Leptoproduction,"
Phys. Lett. B **212** (1988) 391.

[22] R. D. Carlitz, J. C. Collins and A. H. Mueller,
"The Role Of The Axial Anomaly In Measuring Spin Dependent Parton Distributions,"
Phys. Lett. B **214** (1988) 229.

[23] J. Ashman *et al.* [European Muon Collaboration],
"A measurement of the spin asymmetry and determination of the structure function $g(1)$ in deep inelastic muon proton scattering,"
Phys. Lett. B **206** (1988) 364.

[24] J. Ashman *et al.* [European Muon Collaboration],
"An investigation of the spin structure of the proton in deep inelastic scattering of

polarized muons on polarized protons,"
Nucl. Phys. B **328** (1989) 1.

[25] F. H. Heinsius [HERMES collaboration],
"Structure functions and the spin of the nucleon: From HERMES to COMPASS,"
hep-ex/0106021.

[26] J. Ellis and R. Jaffe,
"A Sum Rule For Deep Inelastic Electroproduction From Polarized Protons,"
Phys. Rev. D **9** (1974) 1444.

[27] D. Adams *et al.* [Spin Muon Collaboration],
"The polarized double cell target of the SMC,"
Nucl. Instrum. Meth. A **437** (1999) 23.

[28] D. Kramer [Spin Muon Collaboration],
"The SMC polarized target: Systems and operations,"
Prepared for 11th International Symposium on High-energy Spin Physics and the 8th International Symposium on Polarization Phenomena in Nuclear Physics (SPIN 94), Bloomington, Indiana, 15-22 Sep 1994.

[29] E. Iarocci,
"Plastic Streamer Tubes And Their Applications In High-Energy Physics,"
Nucl. Instrum. Meth. **217** (1983) 30.

[30] M. Moinester,
"Pion polarizabilities and hybrid meson structure at CERN COMPASS,"
hep-ex/0012063.

[31] M. Moinester and S. U. Chung,
"Hybrid meson structure at COMPASS,"
hep-ex/0003008.

[32] M. A. Moinester, V. Steiner and S. Prakhov,
"Hadron photon interactions in COMPASS,"
hep-ex/9903017.

[33] M. A. Moinester and V. Steiner,
"Pion and kaon polarizabilities and radiative transitions,"
hep-ex/9801008.

[34] S. Godfrey and J. Napolitano,
"Light meson spectroscopy,"
Rev. Mod. Phys. **71** (1999) 1411 [hep-ph/9811410].

[35] W. Lee and D. Weingarten,
"Scalar quarkonium masses and mixing with the lightest scalar glueball,"
[hep-lat/9805029].

[36] W. Lee and D. Weingarten,
"The scalar quarkonium spectrum and quarkonium glueball mixing,"
Nucl. Phys. Proc. Suppl. **63** (1998) 194 [hep-lat/9801013].

[37] M. A. Moinester and V. Steiner,
"Primakoff physics for CERN COMPASS hadron beam: Hadron polarizabilities, hybrid mesons, chiral anomaly, meson radiative transitions,"
[hep-ex/9801011].

[38] N. Isgur and M. B. Wise,
"Relationship between form factors in semileptonic $\overline{B}$ and D decays and exclusive rare $\overline{B}$-meson decays,"
Phys. Rev. D **42** (1990) 2388.

[39] C. Amsler *et al.* [Crystal Barrel Collaboration],
"High statistics study of $f_0(1500)$ decay into $\pi_0\pi_0$,"
Phys. Lett. B **342** (1995) 433.

[40] C. Amsler *et al.* [Crystal Barrel Collaboration],
"High statistics study of $f_0(1500)$ decay into $\eta\eta$,"
Phys. Lett. B **353** (1995) 571.

[41] C. Amsler *et al.* [Crystal Barrel Collaboration],
"$\eta\eta'$ threshold enhancement in $\bar{p}p$ annihilations into $\pi_0\eta\eta'$ at rest,"
Phys. Lett. B **340** (1994) 259.

[42] L. Schmitt,
<lschmitt@e18.physik.tu-muenchen.de>,
Private Communication, 2001.

[43] D. Alde *et al.*, "Acquisition System For The Hodoscope Spectrometer Gams-4000,"
Nucl. Instrum. Meth. A **240** (1985) 343.

[44] L. Gatignon,
"Overview of M2 comissioning in 1999,"
http://gatignon.home.cern.ch/gatignon/m2commissioning.html

[45] L. Gatignon,
"The modifications to the M2 beam for COMPASS,"
http://sl.web.cern.ch/SL/eagroup/NewM2/main.html

[46] L. Gatignon,
"User guide for the M2 beam,"
http://gatignon.home.cern.ch/gatignon/M2manual.html

[47] L. Gatignon,
<Lau.Gatignon@cern.ch>,
Private Communication, 2001.

[48] H. W. Atherton *et al.*,
"Precise Measurements Of Particle Production By 400-Gev/c Protons On Beryllium Targets,"
CERN-80-07.

[49] B. Povh et al.,
"Teilchen und Kerne,"
Springer 1999.

[50] C. Bovet, R. Maleyran, L. Piemontese, A. Placci and M. Placidi,
"The Cedar Counters For Particle Identification In The Sps Secondary Beams: A Description And An Operation Manual,"
CERN 82-13.

[51] R. K. Bock, A. Vasilescu,
"The Particle Detector BriefBook,"
http://rkb.home.cern.ch/rkb/titleD.html

[52] B. Ketzer *et al.*,
"Performance of triple GEM detectors in the COMPASS tracker,"
http://cern.web.cern.ch/CERN/Divisions/EP/TA-1/
GDD/publications.res/cern00116.pdf

[53] B. Ketzer,
<Bernhard.Ketzer@cern.ch>,
Private Communication, 2001.

[54] S. Kappler,
"Application of Mulit-GEM Detectors in X-Ray Imaging,"
Diploma Thesis, Institut für Experimentelle Kernphysik Karlsruhe, September 2000.

[55] M. Wiesmann,
<wiesmann@e18.physik.tu-muenchen.de>,
Private Communication, 2001.

[56] I. Konorov,
<Igor.Konorov@cern.ch>,
Private Communication, 2001.

[57] I. Abt, S. Masciocchi, B. Moshous, T. Perschke, K. Riechmann and W. Wagner,
"Double-sided microstrip detectors for the high radiation environment in the HERA-B experiment,"
Nucl. Instrum. Meth. A **439** (2000) 442.

[58] CERN RD39 Collaboration,
http://rd39.web.cern.ch/RD39

[59] The ROSE Collaboration,
http://rd48.web.cern.ch/RD48

[60] M. Moll,
"Radiation damage in silicon particle detectors: Microscopic defects and macroscopic properties,"
DESY-THESIS-1999-040.

[61] G. Lindstrom, M. Moll and E. Fretwurst,
"Radiation hardness of silicon detectors: A challenge from high-energy physics,"
Nucl. Instrum. Meth. A **426** (1999) 1.

[62] F. Sauli and A. Sharma,
"Micropattern gaseous detectors,"
Ann. Rev. Nucl. Part. Sci. **49** (1999) 341.

[63] G. Charpak and F. Sauli,
"The multistep avalanche chamber: A new high rate, high accuracy gaseous detector,"
Phys. Lett. B **78** (1978) 523.

[64] F. Sauli,
"GEM: A new concept for electron amplification in gas detectors,"
Nucl. Instrum. Meth. A **386** (1997) 531.

[65] Harold Abelson, Gerald Jay Sussman with Julie Sussman,
"Structure and Interpretation of Computer Programs,"
MIT Press, 1996, http://mitpress.mit.edu/sicp

[66] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides,
"Design Patterns, Elements of Reusable Object-Oriented Software,"
Addison-Wesley, 1994.

[67] Bruce Eckel,
"Thinking in C++ 2nd Edition", Free Electronic Book,
http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

[68] Danny Kalev,
"ANSI/ISO C++ Professional Programmer's Handbook,"
Que, 1999, http://ccfit.nsu.ru/programming/cpp&c/ansi_cpp_progr_handbook

[69] COMGEANT Monte-Carlo program,
http://valexakh.home.cern.ch/valexakh/wwwcomg

[70] COMGEANT COMPASS representation files,
/afs/cern.ch/compass/delivery/
simevt/testntfz63/detectors.63.dat
/afs/cern.ch/compass/delivery/
comgeant/0-0-6.06/data/geom/GPositions.06.txt

[71] CAEN SY527 high voltage crate documentation,
http://caenlab.caen.it/Catalogo.nsf/(AllDoc)/SY527

[72] GNU libtool,
A "do the right thing" tool for library creation,
http://www.gnu.org/software/libtool/libtool.html

[73] Adaptive Communication Environment,
http://www.cs.wustl.edu/~schmidt/ACE.html

[74] xerces-c,
The C/C++ XML parser from the Apache project,
http://xml.apache.org/xerces-c/index.html

[75] Distributed Information Management System,
http://delonline.cern.ch/d$onl/communications/dim/doc/www/dim.html

[76] A free implementation of the Standard C++ Library,
http://www.stlport.org

[77] libcwd,
An object oriented debugging library,
http://libcw.sourceforge.net/debugging/index.html

[78] libnjamd,
Not just another malloc debugger,
http://fscked.org/proj/njamd.shtml

[79] GCC,
GNU Compiler Collection,
http://gcc.gnu.org

[80] insure++,
A memory corruption and memory leak error detection tool,
http://www.parasoft.com/products/insure

[81] CVS,
An open source version control software,
http://www.cvshome.org

[82] DDD,
Data Display Debugger,
http://www.gnu.org/software/ddd

[83] doxygen,
A documentation generation tool,
http://www.stack.nl/~dimitri/doxygen

[84] doc++,
A documentation generation tool,
http://docpp.sourceforge.net

[85] Graphviz,
An open source graph drawing software,
http://www.research.att.com/sw/tools/graphviz

[86] Source-Navigator IDE,
http://sources.redhat.com/sourcenav

[87] PVSS II SCADA system,
http://www.pvss.com
http://www.etm.at

[88] SLiC,
A readout software for slow control,
http://itcofe.web.cern.ch/itcofe/Projects/SLiC/welcome.html

[89] ITCO,
IT - Information Technology - COntrols Group,
http://itcowww.cern.ch

[90] JCOP,
Joint COntrols Project,
http://itcowww.cern.ch/jcop/welcome.html

[91] JCOP Framework v1.0.1,
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/release.htm

[92] Additional JCOP Framework Components,
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/release.htm

[93] JCOP Framework configuration tool,
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
basic-fw/jcop-fe-sw-conf-1.0.1.zip

[94] H. Milcent,
"JCOP Basic Framework devices,"
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
basic-fw/jcop-basic-fw-1.0.1-doc-1.0.1.zip

[95] H. Milcent,
"JCOP Basic Framework utilties,"
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
basic-fw/jcop-basic-fw-1.0.1-doc-1.0.1.zip

[96] H. Milcent,
"Additional JCOP Framework Components,"
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
add-fw-pk/comp/jcop-comp-fw-1.0.1-doc-1.0.1.zip

[97] H. Milcent,
"Cook book on how to use the framework devices and utilities,"
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
add-fw-pk/comp/jcop-comp-fw-1.0.1-doc-1.0.1.zip

[98] O. Holme,
"Adding a device in the framework,"
http://itcowww.cern.ch/pvss2/FRAMEWORK/framework-1.0.1/
add-fw-pk/ext/add_device.pdf