Control and monitoring information is transmitted to all PLC's via serial data links (An extract of the plant control philosophy is given in Appendix B).

1.2 Statement of Problem

The general characteristics of industrial plants are, [1] :-

Each plant is specialised in that it incorporates specific technologies of mechanical, electro-mechanical and electronic systems, and the complex controls and instrumentation systems are sometimes designed for the particular process, and thus compensate and conceal faults.

Also, most high technology processes are complex in operation, have high throughput, long operating times and sequences, and serious consequences for a catastrophic accident.

This generally complicates the generalising of fault detection, diagnosis and recovery techniques.

If one is to consider fault "treatment" in general, [5], then there are four basic functions to perform, viz. detecting a fault, identify the fault, localise the fault, and then perform the correct fault action or recovery procedure. Identifying a fault involves finding the device or system detected, while localising the fault involves determining which part or sub-system the fault effects before taking corrective action.

An alarm is the effect of a failure. Nearly all control rooms have many alarms, too many for humans to handle. In manually monitored control and alarm systems, [8], the operator is required to make decisions in real-time, and must know the cause of the faults indicated by alarms to affect fault action. No action or the wrong action could lead to a major disaster and considerable financial loss.

1.3 Proposed Solution

The extent and effectiveness of fault recovery techniques in computer controllers are influenced by many factors, and this is to be investigated.

The following hypothesis is to be proven :-

1. Fault recovery techniques are essential, and the design of computer controlled plants must include fault recovery.
2. All critical conditions must be provided for, and all critical control paths and elements must be duplicated, or tripled if necessary.
3. Manual intervention or manual override facilities must be kept to a minimum, and must only be provided if absolutely necessary.
4. Corporate policies such as budget and manpower availability can restrict the fault recovery implementation.

This thesis can be divided into two main parts, viz. a detailed literature survey and the case study analysis.

The literature survey covers the fault diagnosis, fault recovery and system design concepts in sections 2 to 4. In section 2, the fault diagnostic principles will be detailed, viz. fault detection and fault analysis. Fault recovery strategies and fault-tolerance are dealt with in section 3 on fault recovery techniques. General design concepts are put forward in section 4, and hardware, software and complete system implementation considerations are detailed.

The second part will involve the analysis of system design used in the two case studies. The basis of this analysis will deal with the fault detection, fault recovery and system implementation.

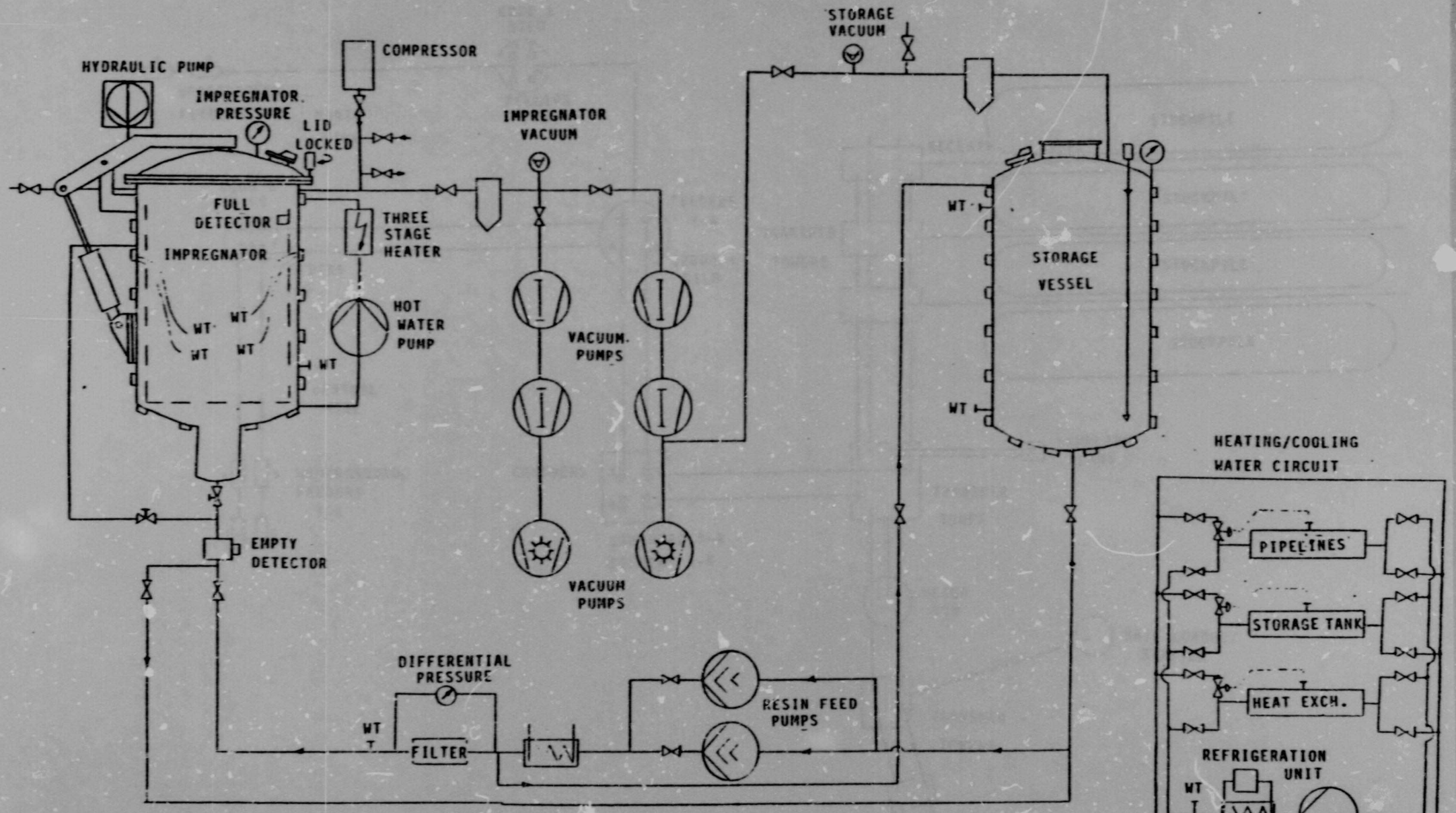

FIGURE 1.1 VACUUM PRESSURE IMPREGNATION PLANT DIAGRAM

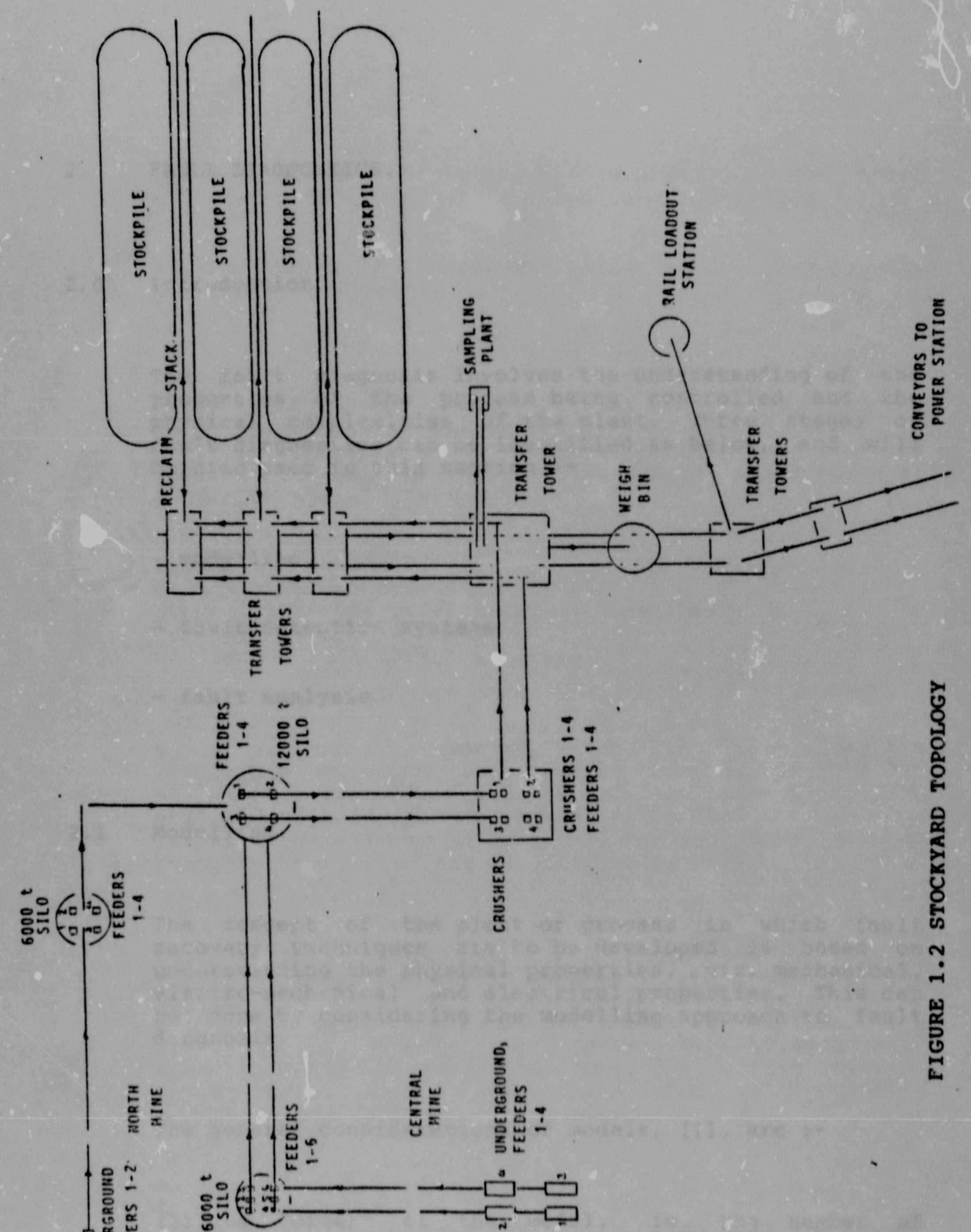

FIGURE 1.2 STOCKYARD TOPOLOGY

2. FAULT DIAGNOSTICS.

2.0 Introduction

The fault diagnosis involves the understanding of the properties of the process being controlled and the physical complexities of the plant. Three stages of fault diagnostics can be identified as below, and will be discussed in this section :-

- modelling
- fault detection systems
- fault analysis

2.1 Modelling

The concept of the plant or process in which fault recovery techniques are to be developed is based on understanding the physical properties, viz. mechanical, electro-mechanical and electrical properties. This can be done by considering the modelling approach to fault diagnosis.

The general considerations of models, [1], are :-

(1) The "order" of the model, ie. the number of independent functions required in the process. This divides the system into modules or sub-systems, which can then be handled independently in terms of control and fault diagnosis,and eventually fault recovery.

(2) The number of parameters involved in the model equations and conditions for each sub-system.

(3) The number of independent variables involved in the sub-system models.

Classification of models is best done as an aid to identifying the correct model type which one wishes to use. As there are many ways to describe and classify models, it seems best to compare models into groups of "opposite pairs" , viz.

Linear vs non-linear
Steady state vs non- steady state
Lumped parameter vs distributed parameter
Deterministic vs stochastic
Continuous vs discrete variable

A linear model is one which exhibits the important property of superposition, ie. the total effect of a set of stimuli is the algebraic sum of the individual effects of each stimulus taken individually. Also, a linear model is one in which all the equations used to describe the system are of first order only.

Steady state models are those which the values of dependent variables remain constant, ie. time-invariant or static models. Non-steady state models are those in which the dependent variables change with time, ie. transient or dynamic. Most systems exhibit both non-steady and steady state conditions, eg. during start-up of systems, (non-steady), and when an operating condition has been achieved, (steady).

A lumped-parameter model is one in which spatial variations are ignored, and where properties and states are regarded as homogenous throughout the system. Distributed-parameter models are ones in which spatial variations of properties and states are allowed for. As variations are generally small and the response of a process to stimuli is virtually instantaneous , the lumped-parameter model is used in most cases. An example is a mixing tank. In this, the engineer assumes a perfectly mixed tank, and so a lumped-parameter model is

used. In reality, there are regions of different concentrations and temperatures which is truly a distributed-parameter system.

A deterministic model is one in which a definite behaviour or pattern can be found to a variable or parameter according to varying conditions. A stochastic model is one in which a random input is inputted, to give a random output. This random input can be inputted after the output of a deterministic model output and still give a stochastic model result. A stochastic model should only be used when a good certainty of errors is possible, and in a truly random manner. If this is not truly random, then it becomes deterministic in that known behaviours can be deduced.

The distinction between continuous and discontinous variable models is that a continuous variable can assume any value within an interval, whereas a discrete variable can only take on one value for each interval of time. Continuous models can only be used in systems or sub-systems which have dedicated analogue controllers. All digital controllers are effectively discrete controllers, and would require discrete models.

The final consideration one must give to models is the physical method on which the models are based, be it physiochemical principles, population balance or emperical.

Physiochemical principles evolve from observations, physical equations and experience of conservation of mass, momentum and energy, and are the most generally applied.

Population balances are used where the number of entities is kept constant, ie. balanced. These are used particularly in chemical industries where chemical equations are based on molar and concentration formulae.

Empirical basis involves the simplification of a real, complex process performance in terms of subjective models describing the relationship between the variables and parameters used, and none of the other factors. This inherently leaves holes in the models which can cause "unaccountable" conditions, unless the model is expanded to accomodate them.

2.2 Fault Detection Systems

A typical process control system is implemented by devising a control flow chart for the hardware and software, which includes fault detection strategies. A set of software constructs for fault detection is given in Appendix C.

The two basic analytic tehcniques for fault detection and diagnosis are, [1] :-

(1) Estimation of variable and model parameters

(2) Pattern recognition

The basic principle involved in the state variable and parameter estimation involves the information flow required to do this, and is shown in Fig. 2.1 . The modelling system must be decided upon, and the relevant models and equations designed or developed. This involves understanding the physical basis of the model, and collecting experimental data for use in determining the parameters for the model, as discussed in section 2.1 .

Unknown data must be found by experimenting and observing effects to produce values for parameters used in the model.

Iteration and simulation of the system using the model must be done instead of actually testing for faults on a new process plant. This guards against doing physical damage to the equipment, and times for faults to appear can be shortened using simulation.

Once the system model has been developed as far as possible, final refinements must then be done on the actual plant during commissioning and testing of the process, ensuring safety features are used and monitored to be sure that these features are operating correctly. The system model and the software used in the implementation is then tested to detect all fault conditions, and any errors in the system model and fault detection system can be corrected.

For fault detection and diagnosis using pattern recognition there are three main principles used, viz. template matching , feature extraction and classification, and fault dictionaries.

Template matching involves the comparison of a sample or input with a stored set of prototypes. The sample is the collection of measurements, eg. temperatures, pressures and vacuums, etc. .

Feature extraction and classification is viewed as a three stage process, as shown in Fig. 2.2 . The measurements are made, from which the particular features required are selected, and then tested by an objective function. This is then classified and decided upon for fault recovery.

Fault dictionaries involve several samples taken at the same instant in time, and classifying them according to reference values as being too high (+), normal (0) and too low (-). The various combinations increase as powers of 3, eg. for 2 variables, there are 3 * 3 = 9 different combinations of (+),(0) and (-), for 8 variables, there are 3 to the power 8 = 6561 different combinations. Thus, this technique is very intensive and is seldom used even in large computer control systems.

Pattern recognition can also be applied to simple logical systems as implemented on digital controllers in a primitive manner, (see Appendix C.)

eg. a set of tripped motor inputs can be scanned to detect any one as being unhealthy using a template or overall mask to detect differences or faults, or using feature extraction by masking each individual input and matching this to a "template", ON for healthy, OFF for unhealthy, or vice versa.

It would certainly be good practise to incorporate the first two of the above techniques into the fault detection, diagnosis and recovery scheme. Time, effort and manpower requirements usually make this impractical, and this definitely excludes the use of fault dictionaries.

When designing an actual alarm or fault detection system the following should be considered, [8] :-

- the number of variables measured,
- which variables need alarms,
- the types of alarms to be handled,
- the alarm limits, and
- the alarm requirements during sequential operations.

For the types of alarms, and the handling of these in the fault recovery scheme, the alarms pertaining to variable measurement are, [8] :-

(1) absolute or static limits, eg. tank 10 % full, or 95 % full ;

(2) the deviation of the variable from a set point ;

(3) zero or full scale range limits, eg. tank empty or tank full ;

(4) rate-of-change limits, eg. temperature rising too fast, calculated and checked against a limit.

Typical alarms for logical systems include :-

(1) limit switches, eg. zero or full scale detection, valves open or closed;

(2) auxillary contacts, eg. devices stopped, tripped or even devices running when switched OFF, (short-circuit fault)

(3) and sequence conditions.

A good alarm system is one which assists the operator in detecting, diagnosing and correcting identified credible failure events, where the failure events have been determined using a Hazard and Operability study, (HAZOP). HAZOP is described in the next section of fault analysis.

All credible alarms are ones expected to occur frequently during the life of the system. The alarm system must detect failure events in all modes of operations to enhance the confidence of the operator of the system.

2.3 Fault Analysis

Each process control scheme has associated with it a unique set of faults which can occur, and should thus be analysed for these specific faults.

It has been shown in the previous section on fault detection that it is relatively simple to detect that a fault has indeed occurred, but finding the particular device or sub-system of the plant which is effected is usually lacking. A general strategy for analysing the faults is needed.

When one considers general alarm systems, [8], the whole process and it's components must be studied, i.e. all the modes of operation and events leading to fault conditions must be analysed. From this, the expected

responses of the plant and the complexity of detection of fault conditions can de determined, and the corresponding fault recovery schemes initiated. These findings should be checked, and revised if found to be incomplete.

Typical modes of failure can be defined, [24], viz.

Mode 0 - where the control system is working and protective system is either working or failed dangerously, which is undetectable, ie. denoted by (W,W) or (W,FD).

Mode 1 - when the protective system has detected that the plant failed and caused a normal trip, assuming the protective system is working, denoted by (F,W).

Mode 2 - a spurious trip occurs when the plant is working normally and the protective system has failed safe, denoted by (W,FS).

Mode 3 - a major fault or destructive hazard, where both the control and protective systems have failed, ie. (F,FD).

The distinction of the different control and protective systems is useful, but is usually done in the same software implementation, except where mechanical and hardwired safety facilities are provided. This causes the collapse of these modes into either mode 0 or mode 3 systems only. A more concise and functional method for fault analysis is therefore necessary.

Logical symbols and specific tests can be arranged into an information flow graph showing the cause and effect of any fault in a system, and this is termed a fault tree, [1]. The analysis of a plant starts with defining the undesired event, and working backwards from this to trace any equipment failures and operational or

procedural errors that may lead to the undesired event. The undesired event is usually drawn at the top of the fault tree, and is thus termed the top event.

Information flow from any equipment failure, operational or procedural error, ie. any fault, is a sequence of events leading towards the top event is called a fault path. These fault paths form the branches of the fault tree.

As an example, a typical chemical reactor is shown in Fig. 2.3 (a), and the associated fault tree is shown in Fig. 2.3 (b).

Once developed, the fault tree must be evaluated to find all unique modes of occurence which will cause the top event to occur, and these unique modes are termed minimal cut sets or critical paths. These are typically shown up as sub-system failures or critical device failures. A cut set is defined as a set of events occurring simultaneously causing a top event, and where a minimal cut set is the smallest set which must exist for top event to occur, ie. a critical cut set.

Fault paths may exist where events occur but do not cause the top event, and can maintain satisfactory performance even when an event occurs.

To perform fault tree analysis, the following steps must be taken, [1], and shown in Fig 2.4 :-

(1) define top event

(2) acquire working knowledge of the system to be analysed

(3) construct the fault tree

(4) validate the fault tree

(5) evaluate the fault tree

(6) provide recommendations and alternatives for informed decisions, ie. for fault recovery strategies.

To define the top event, one must specify the event which you wish to prevent from happening. When considering system safety, the top event should be defined as the critical hazard, for system reliability the top event is a system failure, and for operations analysis, the top event is defined as any fault.

To acquire a working knowledge of the system, the engineering methods must be understood, and the required control for the process must be studied. This involves considering models for the systems involved, (section 2.1).

The construction of the fault tree must follow some logical sequence, and should be easy to follow, (Keep It Simple Stupid is the best motto - KISS). This is perhaps best tackled by structuring the fault tree into sub-system fault trees and dealing with each smaller tree explicitly. This is analogous with structured programming techniques, and will be useful when converting the fault tree into software.

Validation and evaluation of the fault tree are needed to verify that the fault tree meets it's objectives and that all foreseeable faults conditions are catered for. The final realisation of the fault tree in software is limited by the microprocessor hardware and memory space in particular.

The final step is to provide recommendations and alternatives for informed decisions leading to specifying what to do if a particular fault has occurred. This is dealt with in detail in section 3 on fault recovery.

Other analysis schemes can be used, but these eventually form a psuedo-fault tree. Failure Mode and Effect Analysis (FMEA), [1], is one method where instead of backtracking from the top event, forward tracing from an event is done to define and evaluate different failure modes and their effects or consequences on other parts of the process.