SAFETY AND COMPUTER CONTROL

J. Love*

Safety related aspects of the computer control of batch processes are discussed. The principal features of an integrated safety environment, typical of most proprietary systems, are described. Safety issues in the project engineering of control systems are reviewed.
(Key words: safety, computer control, batch processes).

INTRODUCTION

There have been numerous incidents of the Flixborough/Bhopal type. These have all been due to process factors: lack of understanding, poor design, ineffective management, etc. Not a single major incident to date has been directly attributable to computer control per se. Given the complexity of modern control systems and their scope for error, both at the systems level and at the application level, this is an impressive record. Sustaining it is critically dependent upon two factors: firstly, the use of reliable, integrated safety environments based upon proven systems software and mature hardware. And secondly, the use of stringent quality assurance criteria, for applications software in particular, based upon the formulation of detailed functional specifications.

This is generally recognised within the process control community but has yet to be fully appreciated by the chemical engineering world.

* Chemical Engineering Department, University of Leeds, LS2 9JT

Alarms, Trips and Interlocks

It is perhaps useful, in the context of computer control, to discuss what is meant by these interrelated terms:

* Alarm: an indication that some abnormal condition or event has occurred. Alarms are normally triggered by the change in status of discrete input signals or by the value of analogue input signals going beyond some pre-defined limit.
  The syntax of an alarm is "if (status) then (display)"

* Trip: an action that is taken due to an alarm being initiated. The action taken is determined by logic and realised by software, which may be either configurable (e.g. logic blocks) and/or procedural (e.g. sequence coding).
  The syntax of a trip is "if (alarm) then (action)"

* Interlock: an action that is prevented until certain conditions are satisfied. Again, the logic is realised by means of software. The conditions tested may of course include alarms.
  The syntax of an interlock is "if (status) and (status) then (action)"

Standard Alarm Functions

The principal functions offered by most proprietary systems for handling alarms are as follows:

* configurable alarm settings (e.g. hihi/hi/lo/lolo)
* reserved display areas (banner lines)
* integration of alarms with faceplate displays
* interface with graphics displays
* use of colour coding/flashing
* annunciation/acknowledgement of alarms
* comprehensive alarm lists
* alarms included in event logs/records

Thus, for example, assuming appropriate alarm settings for a signal have been entered in to the database, once the alarm is initiated, it should automatically find its way through to the reserved display area, to all the relevant faceplate and graphics displays, onto the alarm list and into the event log.

Alarm lists typically provide for:

* grouping of alarms in user defined areas
* priority levels to be assigned to alarms
* listing of alarms in chronological order
* identification of alarm by both tag no. and name
* time and date stamping of alarms
* observation of current status of alarms
* multi-page listing of alarms

There is of course much more to an integrated safety environment than just being able to handle/display/log alarms. In the context of batch process control this is inextricably linked to the system's sequencing capability.

Structured Batch Control Language

It is perhaps also useful to review what is meant by a structured batch control language. The most common type is a high-level language with a process orientated instruction set of mnemonic and argument form. It provides a framework within which applications software may be developed. This framework has a hierarchial structure as shown in Figure 1, and discussed in detail in Ref 2. An important point to appreciate is that the provision of such a framework as standard and proven is fundamental to the flexible development of reliable and economical applications software.

Two terms in particular are worth further explanation: sequence and recipe.

Sequences contain generic information, and are not specific to any particular grade of product. They consist of a number of functionally related processing operations and stages which have to be executed in order during the course of a batch. Each stage consists of a number of events and actions. Sequence development consists of a pragmatic process of alignment of appropriate instructions against these events and actions. Sequence progression is determined by logic in accordance with the status of the plant and time, subject to any software constraints.

Recipes contain specific information and are used in conjunction with sequences to enable production of batches of particular grades of product. There are essentially four attributes to recipes:

* Formulation, that is which reagents are involved, where they are to come from and in what order, and the quantity of each required.

* Operating conditions, that is set points and ramp rates for control loops, settings/limits for alarms, trips, etc, conditional test data, time delays, and so on.

* Status information, that is flag settings for loop and sequence status, plant availability, operator access, etc, and initialisation of digital outputs;

* Recovery options, that is, criteria for handling the progression of abnormal batches, for example branch forward/backward within a sequence, jump into shutdown sequence, hold, advance in manual, etc.

The use of a structural batch control language is the key to applications diagnostics which, together with the standard alarm functions, enable a truly integrated safety environment.

Applications Diagnostics

This section is reproduced from Ref 3. At their lowest level, application diagnostics are embedded within the main control sequences and are simply used to activate alarms. More complex self-diagnostics are usually in the form of dedicated sequences, or sub-sequences, that run in parallel to the main control sequence, and typically initiate recovery options. These are well illustrated by consideration of a batch reactor.

For example, during the charging stage, having sent out a signal to close the drain valve, the sequence would wait a few seconds to allow for the dynamics of the valve, and then check the input signal generated by an associated proximity switch, the pair of discrete input/output signals being treated as an entity. If they were not consistent, an alarm would be initiated, otherwise sequence flow would progress to opening the vent and feed valves etc. Subsequently the sequence would monitor the level or weight of both the feed tank and the reactor. These of course should correspond. If they were not within limits an alarm would be generated.

For exothermic reactions, it is usually during the initial stages that the reactor is most unstable and control needs to be tightest. Any significant increase in batch temperature would almost certainly activate an alarm, if not trip a shut-down sequence. However, towards the end of the batch, once the exotherm is spent, the batch temperature may well be raised deliberately, say to drive the reaction to completion. That same alarm would be activated. If this were to happen repeatedly, it would become a nuisance and in due course would be ignored, which is inherently dangerous. In these circumstances it is appropriate for some embedded diagnostics to override the alarm, dependent on the stage of the batch.

Another related hazard arises from the continued addition of reagents to the reactor when, for some reason, the reaction goes too slowly or even fails. This can result in an exotherm beyond the capacity of the cooling system when the reaction does eventually get going. A diagnostics sequence can be used to calculate, in real-time, a dynamic heat balance across the reactor, taking into account the potential heat evolved from the weight of reagents added and, if necessary, closing the feed valves. This is inherently far safer than relying solely on a high temperature trip.

On a large integrated batch plant, in an emergency situation, many alarms and trips will occur quickly and it is unreasonable to expect the operator to be able to interpret the significance of them all. A parallel diagnostics sequence can be used to filter them, on a priority basis, and to focus the operator's attention on the most significant ones. In particular, the sequence can logically analyse certain combinations or patterns of alarms and automatically initiate recovery options or shut-down.

Applications diagnostics, as in the above examples, are practical and can be realised using the existing functional capability of computer control systems. The potential for such diagnostics is often not fully appreciated - they offer scope for substantial improvements in plant safety, for relatively little in additional costs.

In the context of batch process control, and also for the start-up and shut-down of continuous plant, applications diagnostics are perhaps the most significant contribution to safety that computer control can make.

Expert Systems

There is much enthusiasm about the potential for expert systems and there are various real-time artificial intelligence environments available, e.g. MUSE, G2, etc. In the context of process control, efforts have mostly been directed towards alarm handling, fault diagnosis and product quality, Ref 7.

In essence, the expert system sits alongside the proprietary control system and has access to its real-time data base. The control system functions as normal, the expert system operating in an on-line advisory role.

The experience gained in the RESCU project, for example, is of interest. This concerned the development of an expert system for an existing batch polymer plant. Despite considerable development costs, the operators are not (yet) relying on the systems recommendations, Ref 8.

The use of expert systems is in its infancy and is somewhat analogous to the early attempts at computer control. There is a lot of scope for spectacular failure! Pragmatism will lead to expert systems being applied to small and well-defined tasks, in which experience and confidence can be gained, and where some payback can be identified.

It will be a long time before the chemical industry, with its proper sensitivity to safety and environmental issues, has the confidence to close the loop around a real-time expert system in earnest.

Software based control and safety systems are critically dependent on their supporting hardware. There are three aspects to this of interest: reliability, which is discussed in detailed in Ref 3, architecture, and protection.

Architecture

A very obvious technology-driven trend has been towards systems with distributed architecture. High speed and robust serial communications links - referred to as data highways - have enabled processor power to be targeted very effectively. For example, remote input/output (i/o) signal processing, local control units and intelligent operator control stations, are all commonplace now.

In the context of batch control, however, the trend towards distribution is something of a contradiction. Batch plants are invariably multi-product, multi-stream or multi-purpose. They are usually topographically compact and often highly integrated. Thus large amounts of data about the status of the plant and its control system has to be gathered together to enable activities like sequence execution, contention handling and self diagnostics. This data concentration, and its potential for saturation of the communications links, is a very real constraint on distribution, Ref 9.

Current thinking is that the architecture of the control system needs to reflect that of the plant. Distributed systems may suit large continuous processes with relatively independent and geographically widespread plant units, but not necessarily batch processes, Ref 4.

Protection Systems

It is important to distinguish between safety functions and protection systems. In the normal course of events, providing the control system is functioning properly, all safety functions, including shut-down, are realised by means of the control system. However, in the event of the control system failing, or its safety functions being unable to cope with emergency situations, an independent high integrity protection system is activated. Protection systems are, in effect, the last resort regarding safety.

The hollowed practice, of course, is to use hard wired analogue circuits and devices for all the trips and interlocks of protection systems.

However, there are many circumstances in which digital devices offer significant advantages over their analogue counterparts. Current thinking is that, subject to various provisos, it is acceptable to use software based systems for protection purposes, often referred to as active safety systems.

The provisos include the following:

* the protection system is wholly independent of the control and safety systems in normal use;
* its design/configuration is properly considered and assessed;
* its software is thoroughly tested;
* its subsequent integrity is rigorously maintained.

Detailed guidance on the criteria for the design of software based protection systems is given in Ref 10.

Project Engineering

It does not matter a great deal what the control systems functional capabilities are if the project is not properly engineered.

Choosing an appropriate system is a crucial step in the project, and is fraught with prejudice. Factors to be considered in choosing a batch control system are discussed objectively in Ref 5.

Fundamental to choosing a system, developing the applications software, and testing it are the functional specifications. These provide the basis for all aspects of quality assurance, in particular of the applications software, and are discussed in detail in Ref 11. There are several stages:

1. User requirements specification. At this stage the users requirements are defined in sufficient detail to enable the suppliers to tender for the work. The user should specify the problem and leave the suppliers to work out the solutions.

2. Detailed functional specification. At this stage the chosen supplier and the user together sort out precisely what the system is required to do in all circumstances. This involves consideration of the safety requirements. For batch systems this includes developing sequence flow diagrams, or equivalent. The supplier and user must reach agreement on what functions are to be provided to meet the users requirements, and what resources will be required.

3. Software specification. At this stage the supplier specifies the detailed design of the applications software: this will consist of database tables to be operated upon by re-entrant routines, procedural coding to be interpreted in a real-time multi-tasking environment, etc. The supplier then develops the applications software to that design and tests it against the detailed functional specification.

4. Acceptance testing. At this stage, an agreed series of tests are carried out to demonstrate to the user that all the requirements of the detailed functional specification have been satisfied.

The amount of effort and commitment, from all parties, necessary to develop the specification and to produce the documentation in conformance with acceptable quality assurance criteria should not be under-estimated. Neither should the cost!

Sequence Flow Diagrams

In essence, these are a graphical representation of the process events and control actions, and of the logic interrelating them, organised in such a way as to depict the sequence progression. Various alternative representations are discussed in detail in Ref 12, but SFDs are the most common.

They are developed from a detailed analysis of process operations, as outlined in Ref 2.

Analysis occurs at two levels. At an overall level, operations are broken down into units and stages, either on a process or item basis, and sequential, parallel and common operations are identified. An example of an SFD at this overall level is shown in Figure 2.

These operations are then broken down at a detailed level into discrete events and actions.

Note in particular that:

* agreement must be reached on how every operation is to be carried out. This leads to a fuller understanding of the process/plant and often results in direct improvement/savings in itself;

* some 60-80% of the SFDs, and hence of the applications software, will be devoted to handling safety functions, i.e. alarms, trips, interlocks, diagnostics, etc, and to recovery options for handling abnormal conditions.

* the SFDs must be both correct and complete with regard to detail since the integrity of the system is involved.

HAZOP

The position of HAZOP studies in the software cycle is as indicated in Figure 3. An important point to appreciate is that it is the documentation, e.g. the SFDs alongside the P & IDs, which is subject to the HAZOP studies, and not the software itself!

In general, if there has been a change in the detailed functional specification, depending on the nature and scope of the change, it will be necessary to carry out further HAZOP studies before implementing the appropriate software changes. However, mistakes in the software revealed during commissioning can be corrected, without further HAZOP considerations, provided the functional specification has not been changed.

Software Change

One of the principal advantages of computer control is the increased flexibility that it offers, see Ref 1.

However, increased flexibility can be something of a liability. Thus it is easy to modify sequences, reconfigure loops, and even try out completely new strategies. All that is required, in theory, is to develop appropriate software using an on-line editor, and to activate it. In particular, the implementation can be made quickly compared with the pantomime that would be involved in making the same changes with an equivalent hard-wired system. This flexibility can lead to real improvements in plant performance, reliability, and safety.

But it is easy to lose track of what changes have been made, especially if they have been made regularly or by more than one person. Also, it is easy to make mistakes. Without careful and systematic development of software changes, coupled with thorough checking and testing, faults can be introduced that do not immediately manifest themselves. Unless care is taken in regulating software changes, its integrity is jeopardised. Current thinking is that access to software must be highly restricted and, in particular, that software changes must be treated just as seriously as changes to the plant. They must be subject to the same rigorous management procedure and safety assessment.

The procedure is discussed more fully, and a specimen modification control form presented in Ref 6.

CONCLUSION

In summary, computer control has a great deal to contribute to plant safety, especially for batch processes. However, it is critically dependent upon good project engineering which, to a large extent, is determined by the quality of the specifications and the documentation. These are very much a function of time and effort, and experience.

REMEMBER: **safe control is not cheap,**
cheap control is not safe.

REFERENCES

1. Love, J., Batch Process Control, Chemical Engineer, 437, pp 34-35, June 1987.

2. Love, J., Strategies for Batch Control, Chemical Engineer, 440, pp 29-31, September 1987.

3. Love, J., Confidence in Control, Chemical Engineer, 443, pp 36-38, December 1987.

4. Love, J., Trends and Issues in Batch Control, Chemical Engineer, 447, pp 24-26, April 1988.

5. Love, J., Horses for Courses before Carts, IEE Colloquium, London, December 1986.

6. Love, J., User Guide to Plant Commissioning, Chapt.6., I.Chem.E., to be published.

7. Shorter, D., IKBS in the Process Industries, I.Mech.E. Seminar, Manchester, September 1988.

8. Leitch, R., RESCU Retrospective Review, Heriot-Watt University, May 1987.

9. Benson, R., Process Systems Engineering: Past, Present and Future, I.Chem.E. Conference, Sydney, August 1988.

10. Programmable Electronic Systems in Safety Related Applications, HSE, 1987.

11. Guidelines for the Documentation of Software in Industrial Computer Systems, IEE, 1985.

12. Mallaband, S., The Specification of Batch Process Control Schemes by the use of Flow Charts, I.Chem.E. Conference, Leeds, September 1988.

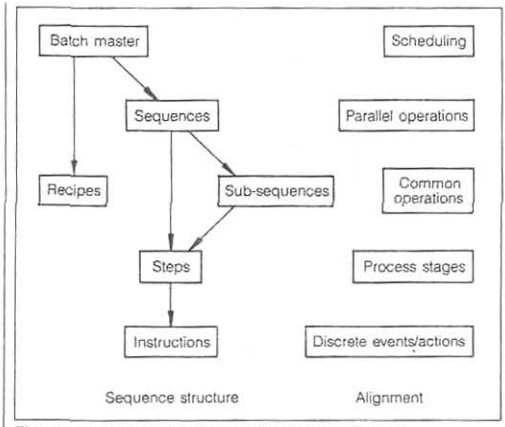

Figure 1: Framework for applications software

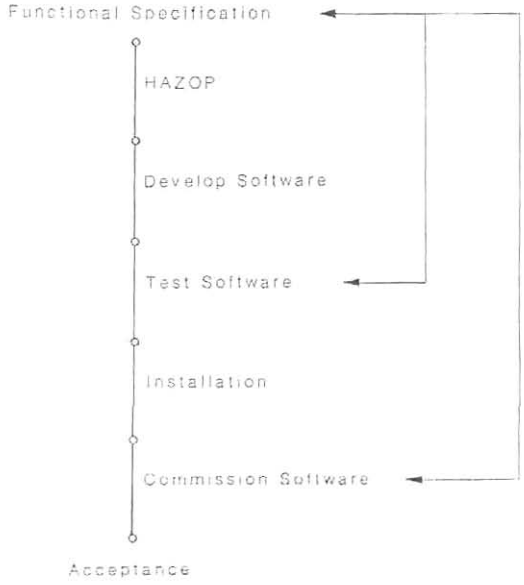

Figure 3

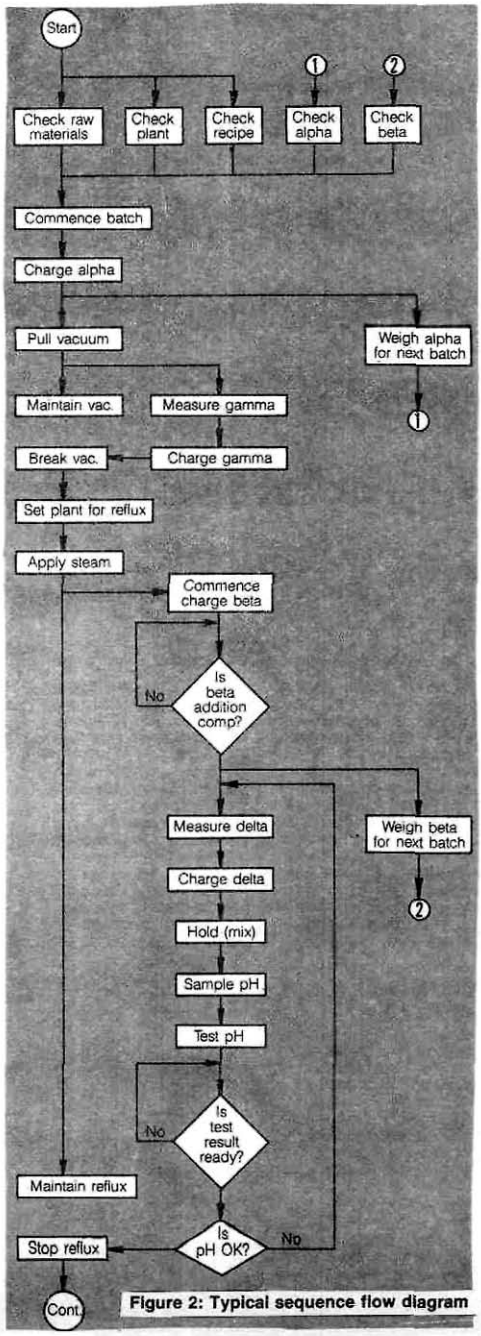

Figure 2: Typical sequence flow diagram