OPTIMISATION OF TEST TIMES OF ELECTRONIC EQUIPMENT BY THE APPLICATION OF STATISTICAL TEST PROCEDURES

Brian Philip Courtnage

A project report submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg, 1985

DECLARATION

I declare that this project report is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

(Signature of the candidate)

24 TH day of DECEMBER 1985

ABSTRACT

This project report investigates the feasibility of applying statistical test procedures (or sampling inspection) to reduce the time spent on inspecting electronic modules produced on the various production lines of Telephone Manufacturers (TM). The effect of these procedures on such factors as the average outgoing quality and the production yield, is examined.

Two inspection situations exist at TM but no information concerning the production process (distribution of defective products) is available, this data must therefore be simulated.

A number of sampling plans are chosen for evaluation. The relative performance of each of these sampling plans is then assessed using the simulated production process data. The sampling plan that produces the best results, is selected for implementation.

Before the sampling plans can be implemented at TM, the accuracy of the simulated production process must be considered. The entire evaluation process must be repeated, using collected production process data, if the simulation is inaccurate.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to all those who contributed to the preparation of the project report. Particular mention must be made of the following:

- My supervisor, Professor H.E. Hanrahan, Department of Electrical Engineering, University of the Witwatersrand, for his valuable comments, criticisms, and suggestions.

- Mr H. Watson and Mr J.A.W. Viljoen of Telephone Manufacturers of S.A. (Pty) Ltd, for their support and encouragement.

- Miss D.G. Prentice and Miss D.A. Prentice for proof reading the text and for typing the final manuscript.

The author also wishes to acknowledge the provision of facilities and financial support by the University of the Witwatersrand and the provision of financial support by Telephone Manufacturers of S.A. (Pty) Ltd.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

1.1 BACKGROUND TO THE PROBLEM

Consumer products, such as telephone instruments, are continuously becoming more complex in a functional sense. The main problem as far as inspecting is concerned is not the complexity of the product, but the time spent in inspection and fault diagnostics. As a consequence of high production volume, test equipment becomes a significant part of the production line and thereby has a direct influence on production yield.

An increase in production yield can be achieved by purchasing more test equipment. The new test equipment would supplement the existing test equipment at each test station, thus allowing more products to be inspected. Alternatively, we could attempt to reduce the time spent on inspecting each product. If the inspection time per product could be reduced, while maintaining a required quality level, one of the following situations would occur;

1. More products would be inspected using the existing test equipment, or

2. The same number of products would be inspected using less test equipment, allowing other products to be inspected using the remaining test equipment.

This would lead to the increased efficiency of the test equipment in both cases.

Various telecommunication products are manufactured by Telephone Manufacturers (TM). Each of these products contain a number of electronic modules, each of which must be inspected to ensure their correct operation. The existing inspection procedure used at TM is complete inspection,

otherwise refarred to as 100% inspection, of each electronic module. The tests include, the measurement of electrical parameters, which have to be within a specified range, and the checking of the various functions of the electronic module. This form of inspection is time consuming and expensive, as each test of every electronic module is checked, even if there is little or no chance that it will fail.

The object of this project report is to investigate the feasibility of applying statistical test procedures (or sampling inspection) to effectively inspect the electronic modules produced on the various TM production lines. The effect of these procedures on such factors as the average outgoing quality (the quality of the outgoing modules) and the production yield, will be examined.

1.2 OVERVIEW TO THE PROJECT REPORT

A review of sampling inspection is given in Chapter 2. This chapter introduces the reader to the terminology and general concepts of sampling inspection and may be overlooked by the reader who is familiar with the subject.

In Chapter 3, the inspection situation at Telephone Manufacturers (TM) is considered in detail. A number of sampling inspection plans are then chosen, each compatible to the inspection situation, for further evaluation.

Chapter 4 briefly describes how the sampling plans chosen in Chapter 3 can be integrated into the inspection situation at TM.

The general requirements of the sampling plan evaluation system are introduced in Chapter 5. A simulation of the production process at TM is proposed. The product which is inspected, is introduced and a method is

devised whereby it can be used to simulate the inspection situation found at TM.

Chapter 6 outlines the approach adopted when designing the evaluation system hardware. The system hardware is designed to fulfil the requirements laid down in Chapter 5, and includes; the external circuitry for the product under test (D.P.S. module), the interface unit circuitry and the automatic KEY operation circuitry.

Chapter 7 outlines the approach adopted when designing the evaluation system software. A top-down structured design procedure is adopted. The software is designed according to the requirements laid down in Chapter 5.

In Chapter 8, the accuracy of the results produced by the process curve simulator, is discussed. Two sampling plans are chosen, one for the inspection of long production runs of continuously produced products and the other for the inspection of short production runs of continuously produced products. This is done by comparing the results of a number of trials and selecting the sampling plan that ensures the best overall results (ie. the lowest average outgoing quality (or the least amount of defective products passed) for the least amount of time spent on inspection) in each case.

The concluding chapter discusses the limitations and possible extensions of the investigation into the use of sampling inspection to inspect the products manufactured by TM. These extensions relate to both the work that is in progress and the implementation, and evaluation, of alternative sampling plans.

2.0 SAMPLING INSPECTION : REVIEW

2.1 INTRODUCTION

The importance of sampling inspection and quality control procedures is very widely accepted, and there is a long history of applications to various branches of industry.

Now in industry it is sometimes necessary to defend inspection by samples against 100% inspection, and to explain why some procedures are reliable. Clearly there are some situations in which 100% inspection is desired rather than sampling inspection, but such situations are infrequent. The reasons why sample methods are preferred are as follows:

1. We never require absolutely accurate information about a batch or quality of goods to be sentenced. For the purpose of sentencing the batch, an estimate of the percentage of defective products is quite sufficient.

2. A point allied to (1) is that under the usual assumptions, the standard error of an estimate reduces as the number of observations increases, approximately as the reciprocal of the square root of the number of observations. Therefore in order to halve the standard error we must take four times as many observations. Beyond a certain point it is either impractical or not worth while achieving greater accuracy.

3. Even if the entire batch is inspected we still do not have an absolute accurate estimate of the percentage of defective products **unless inspection is perfect** . In industrial situations inspection is very rarely perfect and Hill (1962) quotes a probability of 0.9 as being "not unreasonable" for the probability of recognising defects by visual inspection. Some experiments have indicated that if inspectors

are faced with batches of 100% inspection, then the inspection tends to be less accurate than if sample methods are used.

4. In some cases inspection is very costly and 100% inspection is obviously ruled out. One case of this is destructive testing, as in testing of artillery shells. Another case of costly inspection is when complicated laboratory analyses are involved.

When sample methods are employed it is usually assumed that the sampling is random. Thus a sample should be taken in such a way that every item in the batch is equally likely to be taken. In practice this assumption is rarely satisfied and this has to be taken into account when drawing up an inspection plan.

Sometimes it is possible to stratify the items to be sentenced, and use this to draw up a more efficient sample procedure. For example, in the transport of bottled goods in cartons, the bottles next to the face of the carton are more likely to be damaged than those in the interior.

The attitude to the use of samples by the industrialist who is uninstructed in the subject (we may term it the naive attitude) is apt to take one of two extreme form. Some industrialists say that they cannot afford to risk making the mistakes that are inevitable if samples are used, that they can be content with nothing less than 100% inspection.

Doubtless such an attitude is sometimes justified, but such industrialists can sometimes be persuaded that 100% inspection can have its errors and their associated risks of mistaken decisions about the batch, that the risks associated with sampling can be calculated and controlled, and that it is often economical and justifiable to run calculated risks and use samples. At other extreme there is the uninstructed attitude of the industrialist who has based his decisions on samples for years (perhaps the destructiveness of the tests on the articles compels him to) and is unconscious that sampling errors exist. Such an industrialist will often use samples that are hopelessly inadequate in size; he must be persuaded

that sampling errors exist, and that the associated risks should be calculated and taken into account.

2.2 ACCEPTANCE SAMPLING : BASIC IDEAS

The purpose of acceptance sampling is to recommend a specific action; it is not an attempt to estimate quality or to control quality directly. The basic action recommended is to accept or reject the products represented by the sample. When planning a particular inspection plan, it is important to bear in mind the various possibilities for inspection, the type of inspection plan which is appropriate in the particular situation and the aims in view.

2.2.1 ADVANTAGES AND DISADVANTAGES OF SAMPLING

Compared with 100% inspection, sampling has some attractive advantages:

1. Economy due to inspecting only part of the manufactured products.

2. Less handling damage due to inspection.

3. Fewer inspectors, thereby simplifying the recruiting and training problem.

4. Applicability to destructive testing, with a qualified level of assurance of batch quality.

5. Rejections by vendors or shop departments of entire batches rather than merely returning of the defective products. This provides stronger motivation for improvement.

Sampling also has some inherent disadvantages:

1. There are risks of accepting "bad" batches and of rejecting "good" batches.

2. There is added planning and documentation.

3. The sample usually provides less information about the product than does 100% inspection.

2.2.2 OPERATING CHARACTERISTIC (OC) CURVES

A sampling plan specifies the sample size (n) and the associated number of defective products (c) that cannot be exceeded without rejecting the batch from which the sample was taken. The capability of the plan to discriminate between acceptable and unacceptable batches is revealed by its operating characteristic (OC) curve.

The horizontal axis of an OC curve indicates the percentage of defective products in the batch being sampled; the vertical axis shows the probability that the batch will accepted. A condition common to all OC curves is that a batch with no defectives will always be accepted. As the percentage of defective products increases, the probability that it will be rejected also increases.

Two ways to make OC curves more discriminating is to increase n while maintaining c and to decrease c while maintaining n. These recourses are intuitively logical; larger samples tend to more accurately represent the batch, and a decrease in the acceptance number tightens the restrictions.

The ideal OC curve can be obtained only by a 100% inspection of the entire batch made without inspection errors. It takes the form shown by the bold lines in figure 1.

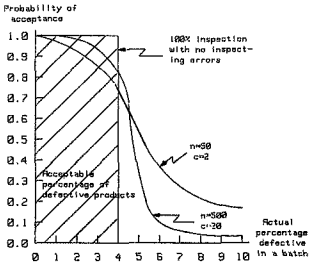


Figure 1. The Ideal OC Curve and Approaches to it.

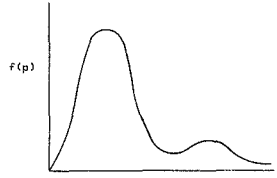
If the total number of defectives in the lot is more than the indicated 4% (shaded area), the lot is rejected without question.

The two curves in the figure show how increasing the sample size while maintaining the same acceptance proportion tends to make the OC curve come closer to the ideal Z-shaped curve. The best sample size to apply to each set of inspecting conditions is a compromise between the value of greater precision resulting from larger samples and the inspecting cost of collecting the larger sample.

2.2.3 PROCESS CURVE

The long run distribution of the quality of batches of items arriving at the inspection station is called the process curve. It is possible that a stochastic process of some kind governs the quality of incoming batches,

but this is usually ignored in batch inspection, partly on the grounds that it is very difficult in practice to obtain information on the process. With continuing production inspection, there is no meaning to the process curve without either arbitrarily batching it , or else bringing in the stochastic element. A typical process curve is shown in figure 2.



p = fraction of defective products
f = process curve = distribution density of p for products being submitted for inspection

Figure 2. Typical Process Curve for Percent Defective Products.

Published data on process curves is very scarce, but some data was collected by Ford (1951), part of which is quoted by Barnard (1954). Horsnell (1957) proposes some theoretical models for process curves which he says fits practical data quite well, but Barnard says, in the discussion following Horsnell's paper, that data he has seen does not bear the slightest resemblance to Horsnell's models. The scarcity of information is due to two reasons:

1. Such data is almost always regarded as industrial secrets.

2. Production conditions are sometimes not held constant for long enough to accumulate sufficient data.

Hamaker (1958) surveys the theoretical models assumed for process curves. The vital question is to know how accurately we need a knowledge of the process curve. The research to date indicates that our knowledge of the process curve does not need to be very precise (Pfanzagl, 1963; Wetherill, 1960; Wetherill and Campling, 1966; Hald, 1967b; Chiu, 1974) provided that the form of the curve chosen is reasonable.

It should be pointed out that in nearly all inspection situations some knowledge of the process curve is needed to arrive at a satisfactory sampling plan, although this knowledge is often used subjectively. We need to know (roughly) how likely it is that batches of any quality will occur, in order to decide how much protection we need at various quality levels. The importance of the process curve was pointed out by Mood (1943), who concluded that there is no point in sampling when the batch quality is stable. Sampling only makes sense with variable quality. We therefore need to take care about schemes worked out on a basis of stable production.

2.2.4 DEFINITIONS

Spotty Quality - In a recent paper, Horsnell (1957) assumes that what he calls **normal** production produces a binomial process curve, and, though this is not explicitly stated, it is evident from his article and the discussion following it that in addition he assumes the occurrence of **bad** batches with a much higher fraction of defective products. This high fraction of defective products will be referred to as **spotty quality** .

Average Outgoing Quality (AOQ) - Consider batches having a given percentage of defective products when presented for inspection this percentage is termed the "ingoing quality". For a given sampling scheme, a calculable portion of these will be accepted without change, the remainder will be rectified and will be passed forward with zero defective products.

The average percentage of defective products in these batches after inspection is the AOQ; and the AOQ is always better (corresponding to fewer defective products) than the ingoing quality.

It is not difficult to see that if, for a given scheme, the AOQ is calculated for a series of ingoing quality and the results are plotted, the graph will be something like the curve in figure 3.

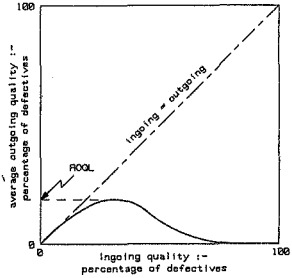


Figure 3. Average Ingoing and Outgoing Quality.

Average Outgoing Quality Limit (AOQL) - On referring to figure 3 we find that for low values of ingoing quality few batches are rectified and the AOQ is a little lower than the ingoing quality. For high values of ingoing quality a large proportion of the batches are rectified and made perfect and the AOQ becomes very low again. In between, the AOQ rises to a maximum known as the "average outgoing quality limit (AOQL)". For a given sampling scheme, the average percentage of defective products in outgoing batches cannot exceed the AOQL whatever the ingoing quality. Each

rectifying sampling scheme has its own AOQ curve just as it has its OC curve, and the AOQL is one criteria of the degree of control given by the scheme which has proved to meet the requirements of many industrialists.

Unrestricted Average Outgoing Quality Limit (UAOQL) - Certain properties of continuous sampling schemes are computed on the assumption that the probability that a unit is defective, p, is constant over time (ie. the ingoing quality is constant). While the possibility of runs of spotty quality has been mentioned in literature (Hamaker, 1958; Horsnell,1957), such parameters as the AOQL make no allowance for such runs. Lieberman (1953) recognised the doubt that could arise in the mind of a user of a continuous sampling scheme because of the questionable validity of the assumption. He shows that even without the assumption of a constant p, there exists a limit on the outgoing quality of the product. This limit has come to be called the "unrestricted average outgoing quality limit (UAOQL)".

2.2.5 CLASSIFICATION OF INSPECTION PLANS

Any system of classifying inspection plans is unsatisfactory in that borderline categories exist. Never the less it will be found useful to have some classification system. Different inspection situations are first listed and alternative sampling plans are given.

2.2.5.1 Inspection Situation

Batch Inspection or Continuous Inspection - Batch inspection occurs when we have items presented in, say, boxes, and it is desired to pass sentence on each box of items together, and not on each individual item. If on

the other hand we have a continuous nylon thread, or a production line of continuously produced small items such as chocolate bars and items are not treated in batches for sentencing, then we have continuous production inspection.

Rectifying Inspection or Acceptance Inspection - If batches of items are presented for sentencing and the possible decisions are, accept or reject, or accept or sell at a reduced price, etc., we have acceptance inspection. Rectification inspection occurs when one of the possible decisions is to sort out the bad items from a batch and adjust or rectify them, or else replace them. That is, with rectifying inspection, the proportion of defective items may be changed.

Inspection by Attribute or Inspection by Variables - Inspection by attributes occurs when items are classified simply as effective or defective. The opposite of this is inspection by variables when the result of the inspection is a measurement of length, voltage at which a voltage regulator works, etc. An intermediate classification between these is when items are graded.

2.2.5.2 Sampling Plans for Continuous Inspection

The Dodge Plans - In 1943, H.F.Dodge of Bell Telephone Laboratories published a continuous sampling plan, (Dodge, 1943). No formal provision for the adjustment of the machine was included and the plans have, therefore, been adjusted accordingly.

Normal Stage : Inspect a fraction (1/f) of the products. If a defective is found, the plan passes to the probation stage.

Probation Stage : Inspect 100% of the products. If i consecutive non-defective products are passed before k further defectives are found,

return to the normal stage. Otherwise, examine and, if necessary, adjust the machine.

After the machine is adjusted, the plan begins again at the probation stage in order to ensure that the adjustment has rectified the fault.

The plan is determined by the three quantities f, i and k. The quantity k is usually small and often equals 1. Dodge also considers the particular plan where i=4 as being appropriate for many cases.

Multi-Level Plans - An inconvenient feature of many continuous sampling plans is the abrupt change from partial inspection to 100% inspection. Not only is this feature inconvenient administratively, but also the introduction of this large amount of inspection is usually made at the first sign of trouble. It is felt that for many situations a smoother transition might be desirable and, Lieberman and Solomon (1954), published the theory for multi-level inspection plans. The introduction of several levels of inspection naturally complicates the inspection procedure to a degree which many will consider prohibitive. The plans, however, will be described briefly.

Normal Stage : Inspect a fraction f^k of the items, where $0<f<1$ and k is an integer greater than 1. When a defective product is found, proceed to the probation stage.

Probation Stage : The probation stage consists of several levels of inspection. At level j, $1 \leq j \leq k-1$, a fraction f^j of items is inspected. If a further defective product is found before i non-defective products have been inspected at this level proceed to level j-1. Otherwise, proceed to level j+1. The probation stage begins at level k-1 and returns to the normal stage whenever i consecutive non-defective products are found at this level. The lowest level, level 0 , consists of 100% inspection of products. If i consecutive non-defective products are found at this level the plan proceeds to level 1, but, if a defective product is found, the machine is adjusted. After adjustment, the subsequent production is inspected using the plan at level 0 of the probation stage.

This procedure may be tightened in several ways. When a defective product is found the plan could specify dropping several levels or all the way to level zero of the probation stage. These modifications, however, tend to destroy the smoothness of the transitions from the different levels of inspection.

Wald and Wolfowitz Plans - Wald and Wolfowitz (1945), Read and Beattie (1961) and Shahani (1979) gave plans of the same general type as the Dodge plans, but they were modified to fit various practical conditions. The inspection rate on the line is held constant and the product is artificially batched. Depending on the results, some batches are set aside for 100% inspection later. This plan forms a link between the Dodge type constant inspection plans and batch inspection plans.

Other Plans - Other plans do exist but most of them are variations of the plans discussed above.

2.2.5.3 Sampling Plans for Batch Inspection

Single Sampling Plans - Suppose we have batches of items presented, and the items are classified merely as effective or defective. A single sampling plan consists of selecting a fixed random sample of n items from each batch for inspection, and then sentencing each batch depending on the results. If the sentence is to be either accept or reject the batch, then each batch would be accepted if the number of defectives r found in the n items were less than or equal to the acceptance number c.

Multiple Sampling Plans - In this plan a first sample of n1 items is drawn, as a result of which we may either accept the batch, reject it, or else take a further sample of n2 items. If the second sample is taken, a decision to accept or reject the batch is taken upon the combined results.

Multiple Sampling Plans - In multiple sampling plans, one, or two, or several still smaller samples of n individual items are taken (usually truncated after some number of samples) until a decision to accept or reject is obtained.

Just as it had been found that double sampling requires less inspection than single sampling, it was found that multiple sampling, with an unlimited number of samples from a batch, generally requires a smaller average total inspection than double sampling. (Dodge, 1969; part I).

Sequential Sampling Plans - A further extension of the multiple sampling idea is the full sequential plan. In this plan, items are drawn from each batch one by one, and after each item a decision is taken as to whether to accept the batch, reject the batch or sample another item.

2.2.6 ECONOMIC ASPECTS OF SAMPLING

The choice of a sampling system for acceptance sampling has been controlled by two concepts ever since statisticians have been engaged in the field - risk and cost. Firstly, a given "amount of protection" is laid down, and then a sampling system is chosen from those available on the basis of achieving the minimum "amount of inspection" for the required amount of protection. At any rate that is the theory. When in practice, it is found that the desired amount of protection leads to an unacceptable amount of inspection even in the most favourable circumstances, the technical requirement is relaxed somewhat and a compromise is sought, which turns out to be more acceptable economically. In recent years various papers have been published on the subject, some of which are as follows : Anscombe (1950); Barnett (1974); Champernowne (1953); Chiu and Wetherill (1973); Hamaker (1951); Horsnell (1957); Satterwaite and Grad; Sittig (1951); Weibul (1951), and Wetherill and Chiu (1975).

Multiple Sampling Plans - In multiple sampling plans, one, or two, or several still smaller samples of n individual items are taken (usually truncated after some number of samples) until a decision to accept or reject is obtained.

Just as it had been found that double sampling requires less inspection than single sampling, it was found that multiple sampling, with an unlimited number of samples from a batch, generally requires a smaller average total inspection than double sampling. (Dodge, 1969; part I).

Sequential Sampling Plans - A further extension of the multiple sampling idea is the full sequential plan. In this plan, items are drawn from each batch one by one, and after each item a decision is taken as to whether to accept the batch, reject the batch or sample another item.

2.2.6 ECONOMIC ASPECTS OF SAMPLING

The choice of a sampling system for acceptance sampling has been controlled by two concepts ever since statisticians have been engaged in the field - risk and cost. Firstly, a given "amount of protection" is laid down, and then a sampling system is chosen from those available on the basis of achieving the minimum "amount of inspection" for the required amount of protection. At any rate that is the theory. When in practice, it is found that the desired amount of protection leads to an unacceptable amount of inspection even in the most favourable circumstances, the technical requirement is relaxed somewhat and a compromise is sought, which turns out to be more acceptable economically. In recent years various papers have been published on the subject, some of which are as follows : Anscombe (1950); Barnett (1974); Champernowne (1953); Chiu and Wetherill (1973); Hamaker (1951); Horsnell (1957); Satterwaite and Grad; Sittig (1951); Weibul (1951), and Wetherill and Chiu (1975).

The determination of a required amount of protection is a technical matter, yet behind the technical facade there is an economic element. The acceptance of sub-standard batches and the rejection of up-to-standard batches is an economic disadvantage for the parties concerned, and the determination of a certain "amount of protection" can therefore hardly be seen otherwise than as an - often enough intuitive - economic operation. From this it follows that a rational solution to the problem can be sought along the following lines :

1. Reduction of the risk ("lack of protection") of a sampling system to terms of cost.

2. Reduction of the amount of inspection involved in a sampling system to terms of cost.

3. Choice of a solution for which the sum of the costs under 1) and 2) is a minimum.

These total costs, which now have to be brought down to a minimum, are the "acceptance costs". They are costs resulting from the imperfection of the inspection method applied in practice. Acceptance costs therefore consist of :

1. Cost of inspection (a-costs)

2. Cost of defectives in accepted batches (s-costs)

3. Cost of effectives in rejected batches (g-costs)

All costs are expressed, not in a unit of coinage, but as a multiple of the value of the unit product in question.

1. Costs of inspection (a) include wages, travelling costs of inspectors and staff, reduction in the value of inspected units, depreciation of measuring instruments etc., and the share of the inspection department in the overheads enterprise. These costs are taken to be

proportional to the number of products inspected. The cost of inspection can be reduced by inspecting more products in unit time.

2. Costs of defectives passed on. These can arise through hold-ups in assembly, or through damage should the defective unit come into the hands of a consumer. Damage might vary from replacement of the defective unit to the loss of costly machinery and even lives. When loss of goodwill is involved, it will be particularly difficult to express the damage sustained (s) in terms of money. An approximation is often possible by assessment of the sum for which one might insure against such an eventuality.

3. Costs of effectives in rejected batches (g). They will, as a rule, be easier to assess than s-costs. When screening of the rejected batches is resorted to, they will equal the inspection costs (g=a). Screening will sometimes be impossible for technical (testing to destruction) or economic reasons. In these cases it might be that the rejected batch can be disposed of at a reduced price, or it might be that the lot is a total loss (g=1)

For a lot of given quality (fraction effectives=q), the average costs of acceptance are :

a-costs ... ax
s-costs ... P(1-q)Ns
g-costs ... (1-P)qNg

where x is the number of products inspected in each batch of N and where P is the probability of the batch being accepted under the given sampling system (Sittig, 1951), giving

$$K = ax+P(1-q)Ns+(1-P)qNg$$

2.3 DISCUSSION

The present review has looked at some fundamental concepts and principles of acceptance sampling plans. When planning any particular inspection plan, it is important to bear in mind the various possibilities for inspection. The type of inspection plan which is appropriate depends on the particular situation, and the aims in view.

3.0 CHOICE OF SAMPLING PLANS TO BE EVALUATED

3.1 REQUIREMENTS

In any realistic assessment of alternative sampling plans, the mechanics of the actual situations into which a sampling plan fits must be considered in some detail. In many papers important, even drastic, assumptions are made, both implicitly and explicitly, as to the manner in which a plan works. The implication of any assumptions must be carefully considered when choosing an inspection plan.

3.1.1 CURRENT INSPECTION SITUATION AT TM

The inspection stations are situated at critical points along each production line. The individual products move from one test station to another on a conveyor belt. As the product approaches the test station, it is manually placed in a test jig and 100% inspection is carried out. After being inspected, a working product is placed back on the conveyor belt, while a defective product is placed in a rack. The defective products are repaired and placed on the conveyor belt again. The defective products are therefore, effectively, replaced by working products.

3.1.2 CLASSIFICATION OF PRODUCTION AT TM

Each product has various tests which have to be carried out on them. These tests could be the measurement of an electrical parameter or a check that a particular function operates correctly.

It is assumed that **inspection by attributes** can be employed when inspecting a product. Inspection by attributes occurs when products are classified simply as **effective** or **defective** (Is the electrical parameter within a specified range or does the function operate correctly (yes/no) ?).

It was previously mentioned that all products at TM are manufactured on a continuous production line, but the length of the production runs vary and can be classified into two categories:

1. Long production runs - The products manufactured on these production lines are required in each end-product. We therefore have the continuous manufacture of one product on each of these production lines. This type of production will be referred to as **continuous production** .

2. Short production runs - The type of products manufactured on these production lines vary from time to time, depending on there demand. Between 500 and 700 products are manufactured during a short production run. Although the products are manufactured along a continuous production line, they can be thought of as batches of products. This type of production will be referred to as **batch production** .

3.1.3 CRITERIA FOR THE CHOICE OF INSPECTION PLANS

The inspection plans that are chosen must confirm to the following criteria:

1. Allow inspection by attributes.

2. Reduce the average time spent on inspecting each product.

3. Ensure a desired (or required) average output quality (AOQ).

4. Allow implementation with the minimum amount of disturbance of the existing inspection and production procedures.

All the criteria mentioned above applies to the sampling plans selected for both continuous and batch production.

3.2 CHOICE OF SAMPLING INSPECTION PLANS

It is evident from the criteria just discussed that two types of sampling plans are required. One sampling plan must be specifically chosen for continuous production, and will be referred to as a **continuous sampling plan** , while the other must be chosen for batch production, and will be referred to as a **batch sampling plan** . The batch production is however continuous in nature, so it will be impossible to apply conventional batch inspection plans (such as simple, double, multiple and sequential sampling) without interfering with the continuity of the current batch production process. Continuous sampling plans that allow the products to be artificially batched are therefore required for the inspection of batch production.

The sampling plans for continuous inspection, mentioned in chapter 2, all guarantee an average outgoing quality level (AOQL). This leads to the AOQ always being below the AOQL.

A number of continuous sampling plans will be reviewed. The sampling plans which can be implemented in the inspection situation at TM, are then chosen and studied in more detail.

3.2.1 DODGE PLANS

Five Dodge plans are considered, four of which are a variation of the continuous sampling plan first introduced as CSP-1 by Dodge (1943). The differences in the plans are evident in the rules they use to return to 100% inspection, after detecting a fault during sampling inspection. The five Dodge plans being considered are referred to as CSP-1, CSP-2, CSP-3, CSP-4, and CSP-5.

CSP-1 : This sampling plan was specifically designed for the inspection of continuous production (Dodge, 1943; Dodge, 1970 : Part IV), but may also be used for the inspection of batch production (Blackwell, 1977).

CSP-2 and CSP-3 : These plans can be used for the inspection of continuous production (Dodge and Torrey, 1951 ; Dodge 1970 : Part IV).

CSP-4 and CSP-5 : These plans were designed for the inspection of continuous production, but for an inspection situation that differs from that of TM (Derman, Johns and Lieberman, 1959).

When i products in succession are found with no defects, only a fraction (1/f) of the products are inspected. For CSP-4, when a defective product is found during the sampling, the remaining f-1 products in the segment are eliminated and 100% inspection begins from the next segment. For CSP-5, When a defective product is found during sampling, the remaining f-1 products in the segment are screened before 100% inspection is implemented.

Both these plans cannot be applied in the inspection situation found a TM. One product is randomly selected from a segment of f products. If the product is defective, a number of products which have already been passed to the next assembly stage, have to be returned for screening. This would disrupt the continuity of the production line.

3.2.2 MULTI-LEVEL PLANS

Lieberman and Solomon (1955) presented a system of continuous sampling plans, called multi-level inspection plans (MLP). These represent a modification to plan CSP-1 by providing several levels instead if a single level of sampling, and permitting greater economy of inspection effort if the incoming quality is consistently high. A criticism of this type of plan, when used with several levels, is that if the quality gradually, or even suddenly, deteriorates, the plan is slow to respond. Derman, Littauer and Solomon (1957) developed three additional sets of plans with modified rules for shifting back more quickly to a "tightened" inspection level.

All these multi-level inspection plans can be used for the inspection of continuous production. Two of these inspection plans will be selected for evaluation. These plans are are designated MLP (Lieberman and Solomon, 1954) and MLP-T, which is the tightest of the three plans developed by Derman, Littauer and Solomon (1957) (Dodge, 1970 : Part IV).

3.2.3 WALD AND WOLFOWITZ PLANS

Three constant sampling plans were developed by Wald and Woltowitz (1945), designated SPA, SPB, and SPC. Shahani (1979) developed four additional Wald and Wolfowitz plans, designated WSP-2, WSP-3, WSP-4, and WSP-5. The sampling plan WSP-1 refers to sampling plan SPC. All these plans are best suited for batch production because the requirement is that the products produced, on the production line, should be artificially batched.

SPA and SPB : Although these sampling plans optimised in the sense that they guarantee the desired AOQL with minimum inspection when the production process is in statistical control, they do not always behave very

favourably as far as local stability is concerned (ie. a deterioration of production process will cause a long segment in the sequence of outgoing products within which the relative frequency of defective products will be larger than the prescribed AOQL) (Wald and Wolfowitz, 1945).

SPC and WSP-1 : This sampling plan has been modified to achieve local stability. This plan can therefore be used for the inspection of batch production. This plan will be designated WSP-1 from now on.

WSP-2 : Each artificial batch consists of N products, made up of n segments each containing f products. When a defective product is found in a segment, all the previous segments of the batch are sent for 100% inspection and the count for the new batch of N products begins from the first product in the next segment. This would cause a disruption in the operation of the production line because all products, constituting the present batch, that have been passed must be returned for 100% inspection. This makes this inspection plan unsuitable for implementation on TM production lines.

WSP-3 : Once a certain number of defective products are found, the entire batch of N products is subject to 100% inspection. This will cause the continuous nature of the production line to be disrupted, making this inspection plan unsuitable for implementation at TM.

WSP-4 and WSP-5 : The operating rules of these two plans do not differ from WSP-1 and WSP-3, but an extra parameter is required to control the amount of 100% inspection. These two plans will not be considered because the batches are small, and it is therefore not considered necessary to control the amount of 100% inspection.

3.2.4 SUMMARY

The continuous inspection plans which must be evaluated are; the three Dodge plans CSP-1, CSP-2 and CSP-3 and the two multi-level plans MLP and MLP-T. There are two batch inspection plans that must be evaluated, they are; The Dodge plan CSP-F (CSP-1 for short production runs) and the Wald and Wolfowitz plan WSP-1.

3.3 CONTINUOUS INSPECTION PLANS : DETAIL

A detailed description is given of each of the selected continuous inspection plans. This will include information about assumptions made when the inspection plans were designed, and how the various parameters, required by the plans, are chosen.

3.3.1 PLAN CSP-1

Assumptions

In most of the theoretical treatments of CSP-1 the following three assumptions are made:

1. All defective products found during inspection are rectified or replaced by working products.

2. Inspection is perfect, ie. mistakes in identifying defective products are never made.

3. Theoretical calculations are made on the assumption that the process is producing defective products with a probability p, and that the probability that any product is defective is independent of the quality of the other products.

Assumption (1) is often realistic, but if it is not, account of this can be taken in the theory. Assumption (2) is unrealistic as we cannot ensure that the inspection is perfect. Assumption (3) effectively states that the process is in a steady state and provided that we realise the implications, it is realistic enough to proceed with the theory. A theoretical derivation of the inspection plan is given by Dodge (1943).

Procedure - See Figure 4

1. At the onset, inspect 100% of the products consecutively as produced and continue such inspection until i products in succession are found with no defects.

2. When i products in succession are found with no defects, discontinue 100% inspection, and inspect only a fraction f of the products, selecting individual products one at a time from the flow of products, in such a manner as to assure an unbiased sample.

3. If a sample product is found to be defective, revert immediately to 100% inspection of the succeeding products and continue until again i products in succession are found clear of defective products, as in paragraph (1).

4. Replace all defective products found with working products or, alternatively, correct the defective products.

General Features

If the quality is high, the sampling periods are relatively long and the portion of the total products inspected is small. On the other hand, if

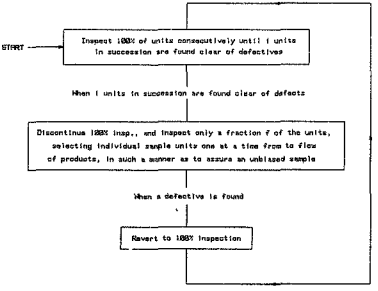
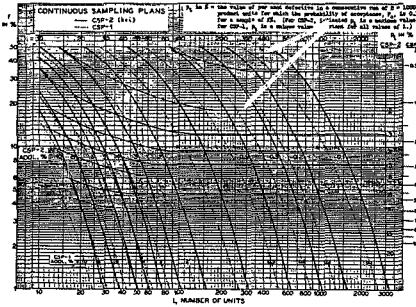


Figure 4. Procedure for Plan CSP-1.

the quality is poor, the sampling periods are relatively short, and the portion of products inspected is relatively large so that the level of defectivas is reduced.

The CSP-1 plan is defined by two constants, f and i, which can be changed at will, and for any particular combination of values of f and i there will be a specific value of AOQL. Certain considerations enter into the most advantageous choice of f and i. For example, it will be apparent that the use of too small a value of f will increase the chances of passing unnoticed a substantial run of low quality products. Some idea of the risk involved is shown by the right-hand scale of figure 5.

And again, if the total production is say only 10 products a day, it would hardly be reasonable to have a value of i=150. How the choice of f and of AOQL affects the statistical characteristic of a plan is indicated by the OC curves in figure 6 (Dodge and Torrey, 1951).



For plan CSP-1 and plan CSP-2, curves for determining values of f and i, for a given AOQL. On the right-handed vertical scale, p is the percentage of defective products in a consecutive run of N=1000 products for which the probability of acceptance P is 0.1 for a sample size equal to the corresponding value of "f in percent" (of N) on the left-hand vertical scale (Dodge, 1970 : Part IV).

Figure 5. AOQL Curves for CSP-1 and CSP-2 (k=i).

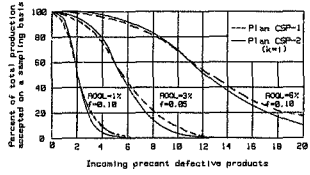


Figure 6. OC Curve of Three Corresponding CSP-1 and CSP-2 (k=i) Plans.

In his paper, Dodge (1943) studied the properties of this plan, and presented equations and charts for determining the AOQL as functions of the parameters f and i, under the assumption that the process is in a state of statistical control.

Lieberman (1953) showed that the Dodge plan guarantees an AOQL whether or not the process is in a state of statistical control. It is proved, without the assumption of control, that for a given f and i, an AOQL is guaranteed. This AOQL is given another definition, namely the unlimited average outgoing level (UAOQL). The formula for the UAOQL, developed by Lieberman (1953) is given as follows:

$$UAOQL = (k-1)/(k+i) \quad , \text{ where } k = 1/f$$

This result was obtained under the hypothesis of random sampling while on partial inspection. The same result was obtained by Derman, Johns and Lieberman (1959) under the hypothesis of probability sampling while on partial inspection. For a given f and i, the above value of the UAOQL is always higher than that obtained using Dodge's equations.

A formula for the UAOQL, under the assumption that the defective products are removed but not replaced with working products, is given by Banzhaf and Brugger (1970) as:

$$UAOQL = (1-f)/(f(i-1)+1)$$

3.2.2 PLAN CSP-2

Procedure - See Figure 7

1. At the onset, inspect 100% of the products consecutively as produced and continue such inspection until i products in succession are found with no defects.

2. When i products in succession are found with on defects, discontinue 100% inspection, and inspect only a fraction f of the products, selecting individual products one at a time from the flow of products, in such a manner as to assure an unbiased sample.

3. When a sample product is found defective, continue sampling, but keep count of the number of products inspected after finding the defective product.

   a. If a defective product is found in the next k or less sample products, revert to 100% inspection of the succeeding products as per paragraph (1).

   b. If no defective product is found in the next k sample products, continue sampling until the next defective product is found and then repeat procedure starting at the beginning of paragraph (3).

General Feature

Plan CSP-2 differs from plan CSP-1 in that, once sampling inspection is started, 100% inspection is not invoked when a defective product is found but is invoked only if a second defective product occurs in the next k or less sample products. The factor k is a general factor which may theoretically be assigned any value whatsoever. However, the use of k=i has certain advantages, including simplicity (Dodge, 1970 : Part IV).

For k=i, figure 5 gives curves for determining values of f and i for a given value of AOQL. Figure 6 shows a comparison of the OC curves for three CSP-2 plans and the corresponding CSP-1 plans, having the same AOQL and f values.

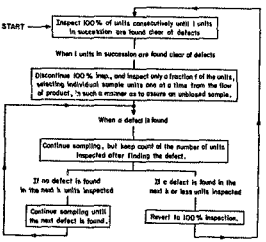


Figure 7. Procedure for Plan CSP-2.

Protection Against Spotty Quality

The p_t curves of figure 5 provides a guide with respect to the protection afforded against spotty quality in a continuous run of products. The p_t curves indicates that the protection against spotty quality falls very rapidly as f is reduced and that the inspection, considering low-quality segments of 1000 consecutive products each, becomes quite poor if f is less than 2% (Dodge and Torrey, 1951).

A formula for the UAOQL, under the assumption that the defective products are removed but not replaced with working products, is given by Banzhaf and Brugger (1970) as:

$$UAOQL = (2(1-f))/(1f+2(1-f))$$

3.3.3 PLAN CSP-3

Procedure - See Figure 8

1. At the onset, inspect 100% of the products consecutively as produced and continue such inspection until i products in succession are found with no defects.

2. When i products in succession are found with on defects, discontinue 100% inspection, and inspect only a fraction f of the products, selecting individual products one at a time from the flow of products, in such a manner as to assure an unbiased sample.

3. When a sample product is found to be defective, inspect the next 4 products. If a defective is found in the in the 4 products revert to 100% inspection of succeeding products as per paragraph (1). If no defective products are found in the 4 products, continue sampling, but keep count of the number for products inspected after finding a defective product.

   a. If a defective product is found in the next k or less sample products, revert to 100% inspection of the succeeding products as per paragraph (1).

   b. If no defective product is found in the next k sample products, continue sampling until the next defective product is found and then repeat procedure starting at the beginning of paragraph (3).

General Feature

In applying plan CSP-3, the curves of figure 5 may be used as an approximation for determining values for f and i for a given value of AOQL. The k=i values are equal to those for CSP-2 when AOQL is less then 2% and are

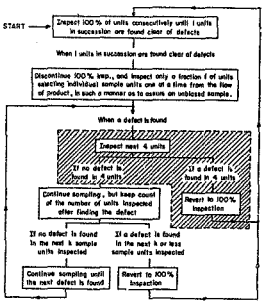


Figure 8. Procedure for Plan CSP-3.

less than those for CSP-2 by no more than 2 products when AOQL=10%. thus the effect of using figure 5 for CSP-3 for large values of AOQL is to give actual values of AOQL slightly smaller than the charted values.

Protection Against Spotty Quality

Plan CSP-3 can be considered a refinement of plan CSP-2 to provide extra protection against highly defective process quality. The extra protection against spotty quality is provided by calling for the inspection of four additional sample units whenever an allowed defective product is found during sampling, and for the immediate return to 100% inspection if one of the four products is found to be defective.

3.3.4 PLAN MLP

Procedure - See Figure 9

Let (f_0, ... , f_k) be a fixed vector of fractions (where $f_0 = 1$). Although it does not affect the mathematics, in practice, the plans will not be sensible unless $f_j > f_{j+1}$.

1. At the onset use f_0 (100%) inspection. Whenever f_0 inspection is used, if i successive products are found with no defects, discontinue f_0 inspection and begin f_1 inspection.

2. When using f_j inspection (j = 1, ... , k-1), if a defective product is found before i consecutive sample products are found with no defects revert immediately to f_{j-1} inspection. If i consecutive sample products are found with no defects discontinue f_j inspection and begin f_{j+1} inspection.

3. If on f_k inspection, continue until a defective product is found, then revert immediately to f_{k-1} inspection.

General Features

The plan that has been implemented uses k = 2, with f_0 denoting 100% inspection, f_1 denoting a sampling rate of f, and f_2 denoting a sampling rate of f^2. It can be seen that if k = 1 we have Dodge' CSP-1 sampling plan.

Figure 10 gives the curves from which the values of f and i are determined for a given value of AOQL, for k = 2.

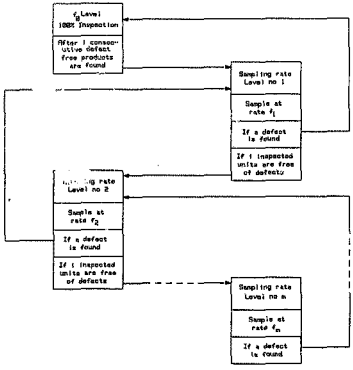


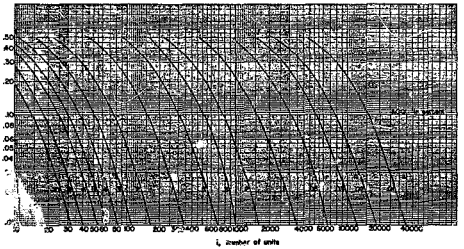
Figure 9. Procedure for Plan MLP.

Protection Against Spotty Quality

The value of k = 2 was chosen for two reasons:

1. With no limit on k, the plan is slow to respond, moving step by step from a looser to tighter levels of sampling, if the quality gradually, or even suddenly, deteriorates.

2. The curve for determining values of f and i for a given value of AOQL is given for k = 2 (Lieberman and Solomon, 1955)



Curves for determining values of f and i for a given value of AOQL; k = 2 (Lieberman and Solomon, 1955).

Figure 10. AOQL Curves for Plan MLP.

This low value of k = 2 allows the plan to respond rapidly to spotty quality, but also allows for a minimal amount of sampling when the quality of the ingoing products is high.

3.3.5 PLAN MLP-T

Procedure - See Figure 11

Let (f_0, ... ,f_k) be a fixed vector of fractions (where $f_0 = 1$). Although it does not affect the mathematics, in practice, the plans will not be sensible unless $f_j > f_{j+1}$.

1. At the onset use f_0 (100%) inspection. Whenever f_0 inspection is in use if i successive products are found with no defects, discontinue f_0 inspection and begin f_1 inspection.

2. When using f_j inspection (j = 1, ..., k-1), if a defective unit is found before i consecutive sample products are found with no defects revert immediately to f_0 inspection. If i consecutive products are found with no defects discontinue f_j inspection and begin f_{j+1} inspection.

3. If on f_k inspection, continue until a defective is found, then revert immediately to f_0 inspection.

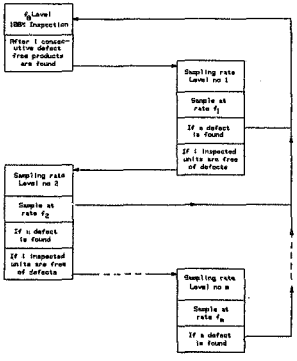


Figure 11. Procedure for Plan MLP-T.

General Features

The plan that has been implemented uses k = 2, with f_0 denoting 100% inspection, f_1 denoting a sampling rate of f, and f_2 denoting a sampling rate of f^2.

The curves for determining values of f and i for a given value of AOQL, with k = 2, for the MLP-T plan, were not obtained and it was assumed that the curves for the MLP plan (k = 2) could be used. Plan MLP-T is "tighter" than the MLP plan, so the AOQL used when determining the values of f and i, on figure 10 can most probably be slightly larger than the actual AOQL value required. As the exact nature of this relation is not known we will not take this into consideration.

Protection Against Spotty Quality

This plan is the tightest of the three plans developed by Derman, Littauer and Soloman (1957). It immediately reverts to 100% inspection when detecting a defect. With k = 2 this plan will respond quickly to an incoming stream of defective products.

3.4 BATCH INSPECTION PLANS : DETAIL

A detailed description of each of the selected batch inspection plans, is given. This will include information about assumptions made when the inspection plans were designed, and how the various parameters, required by the plans, are chosen.

3.4.1 PLAN CSP-F.

A modification of the CSP-1 plan was devised, which is the same as the CSP-1 plan in operation but is indexed by the two parameters i and n and also by the production run size. This plan is called CSP-F (Blackwell, 1977).

Procedure - See Figure 4.

Plan CSP-F operates in the same way as plan CSP-1. (refer to Section 3.3.1.).

General Features

The assumptions made by Dodge in his original paper of perfect inspection and non-varying quality are also made.

The choice of f, i, and N to give a required value of AOQL, is outlined in QSTAG 340 (1974). This reference was not obtainable. Two sets of values were however given by Blackwell (1977), for production runs of 500 - 1000 products.

For an AOQL value of 1.9%, i = 36 and f = 1/5.
For an AOQL value of 0.143%, i = 302 and f = 1/10.

3.4.2 PLAN WSP-1

Procedure

1. At the onset, inspect only a fraction , f, of the products, selecting individual sample products one at a time from the flow of the products, in such a manner so as to ensure an unbiased sample.

2. When a sample product is found to be defective, keep count of the number of defective products.

   a. If a total of M^* defects are found, where $M^* = f\ (AOQL)N/(1-f)$, revert to 100% inspection for the remainder of the batch.

   b. If the number of defects found thus far is less than M^*, continue sampling inspection starting at paragraph (1)

General Features

This plan limits the extent of 100% inspection to the remainder of the products in the batch. If the batch is made relatively large, eg. 5000 or more, little protection will be provided against a run of low quality products, whereas if the batch is small, the protection will be high but at the cost of more inspection. No specific guide to the choice of f and N (size of the batch) is provided.(Dodge, 1970 : part IV; Wald and Wolfowitz, 1945; Shahani, 1979)

In our case the batch is limited to between 500 and 700 products. The value of AOQL, given as a percentage , will be specified. The value of f ($0<f<1$) must be chosen as large as possible. During sampling inspection every 1/f product is inspected, the larger the value of f, the more the amount of products inspected.

Protection Against Spotty Products.

Wald and Wolfowitz (1945) provide mathematical proof that their plan will not allow the average outgoing quality (AOQ) to exceed the AOQL value, "even if Maxwell's demon of gas theory fame were to transfer his activities to the production process.". It is noted that, relative to the CSP-F plan, a fairly sizable price in increased inspection is paid to provide WSP-1's protection against batches of spotty quality. (Dodge, 1970 : Part IV.).

4.0 IMPLEMENTATION OF SAMPLING INSPECTION PLANS AT TM

4.1 GENERAL APPROACH

Various telecommunication products are manufactured by Telephone Manufactures (TM). Each product consists of a number of sub-assemblies, namely, the plastic casing and various electronic modules. The electronic modules are inspected, after being assembled, to ensure that they perform according to specification. The inspection of the modules is performed automatically using automatic test equipment. After a module is placed in a test jig, it is automatically inspected using pre-programmed test procedures.

A number of tests must be performed to completely inspect a module. The tests include; the measurement of electric parameters, which have to be within a specified range, and the checking of various functional operations. There are generally a large number of tests to be performed (\ Myklebust and Hammerboen (1983) estimate "about 60" tests).

When inspecting the module using sampling inspection, each test must be inspected using its own sampling inspection plan (the sampling inspection plans used for all the tests are usually the same). This is done to ensure that the tests that fail the least are inspected the least, and they should be separated from the tests that fail more often and require a greater amount of inspection.

4.2 CONSEQUENCES OF INSPECTION OF SEVERAL KINDS OF DEFECTS

4.2.1 EFFECT ON THE PROCESS AOQL

In his paper, Dodge (1943) outlined the general procedure for the inspection of several kinds defects of simultaneously : When applying a separate inspection plan to each test of the module, a separate value of AOQL must be established for each test. The various inspection plans, running parallel, are completely independent of one another, each insuring that the average outgoing quality (AOQ) does not exceed the value of AOQL.

A prescribed value of the process AOQL, designated $AOQL_p$, must be ensured. This $AOQL_p$ must be used to obtain a AOQL value for each test, designated $AOQL_t$, in such a way as to ensure that the process AOQ is always the $AOQL_p$ value.

The following approach has been adopted to calculate the $AOQL_t$ values from a given $AOQL_p$.

If there are N_t separate tests per module, and the values of $AOQL_t$ are equal for each test, the value of $AOQL_t$ can be calculated using the following equation :

$$AOQL_t = AOQL_p/N_t \quad \ldots\ldots \quad (1)$$

The proof for this equation is given in Appendix A.

If an $AOQL_p$ is required, and there are 50 tests on each module, the $AOQL_t$ can be calculated using equation (1), as being 0.02%. This $AOQL_t$ is very small and a large amount of inspection would be required to ensure this value. It is also not possible to obtain values of f and i from the standard AOQL curves (see figure 4) when employing Dodge sampling plans.

4.2.2 GROUPING OF TESTS

One method of solving the problem of too small an $AOQL_t$ value, is to reduce the number of tests for each module. This can be done by arranging the tests into groups, according to their seriousness. The chosen AOQL value for the group of tests, designated $AOQL_g$, now applies to all defects of the group collectively, and each group is always inspected for all the defects under consideration (Dodge, 1943).

Equation (1) now becomes :

$$AOQL_g = AOQL_p/N_g \quad \ldots\ldots \quad (2)$$

where N_g is the number of groups requiring separate inspection plans.

By limiting the maximum number of groups to ten, the value of $AOQL_g$ is 0.1% for an $AOQL_p$ of 1%. The values of f and i can now be determined using the standard AOQL curves (see figure 4 and figure 10) for the inspection plan used.

4.2.3 DEFINITION FOR PRODUCT AND TEST

The following definitions will be used in the remainder of the report :

1. **Product** - Product will refer to the article being inspected. The product may be an electric component, an electric module, or the final product.

2. **Test**- Each product has a number of electric parameters and functions which require inspection. When the number of electrical parameters and functions requiring inspection is small (< 10), a test will refer to the inspection of a single electrical parameter or function. When

a large number (> 10) of electrical parameters and functions must be inspected, they must first be arranged into groups of similar classes. Each of these groups will now be referred to as a **test** which must be inspected.

3. ***Sampling Plan*** - *Each* **product** *has a number of* **tests** *that must* be inspected. When inspecting a **product** individual inspection plans are used to inspect each **test** (ie. a number of inspection plans running in parallel, one for each **test**). A **sampling plan** will refer to this group of individual inspection plans (the inspection plans will all be the same type, and have the same $AOQL_t$.

Equations (1) and (2) lead to the following general equation:

$$AOQL_t = AOQL_p/N \quad \ldots\ldots \quad (3)$$

where N *is the number of* **tests** requiring inspection and $AOQL_t$ is the AOQL for each **test**.

4.3 DISCUSSION

The sampling inspection plans, used to inspect the tests, operate completely independently of each other. It is therefore possible to use a different sampling inspection plan for the inspection of each individual test.

When considering a particular product, we sometimes find that a certain test is critical to the operation of the product. By using 100% inspection *for this particular test and sampling* inspection for the other tests of the product, we could reduce the inspection time while ensuring that a completely defective product is not allowed to pass undetected.

This feature is not used by the system developed for this project ie. one type of sampling inspection plan is used to inspect all the tests on the product.

5.0 DESIGN OF THE SAMPLING PLAN EVALUATION SYSTEM

5.1 GENERAL APPROACH

One, of the five continuous inspection plans and one, of the two batch inspection plans must be selected for implementation at Telephone Manufacturers (TM). This is done by comparing the results of each trial, for a number of trials, and selecting the sampling plan that ensures the best overall results (ie. the lowest average outgoing quality (AOQ) for the least amount of time spent on inspection).

Each of the sampling plans being evaluated is used to inspect an identical production run (during each trial). The parameters required by each sampling plan are chosen so as to ensure an identical AOQL for each plan during a trial. This is to ensure that a fair comparison is made when selecting the sampling plan.

Identical production runs can be obtained by duplicating the number of defective products, the location of each defective product, and the test that failed on each defective product being inspected by each sampling plan during a trial. The production run is different for each trial.

There is no available information concerning the ingoing quality and the distribution of defective products (process curve) on the production lines of TM. The process curve used in the evaluation system must therefore be simulated.

5.2 SIMULATING THE PROCESS CURVE

5.2.1 CLASSIFICATION OF THE PROCESS CURVE

Before the process curve, for the production process at TM, can be simulated a number of assumptions must be made as to it's general shape and these assumptions must be substantiated.

The following assumptions are made which attempt to classify the process curve that might apply to the distribution of defective products on a production line at TM.

1. The distribution of ingoing defective products can be considered "normal" (ie. production in statistical control) for a large portion of the time. Horsnell (1954) assumed that "normal" production produces a binomial process curve. This section of the process curve is due to products having the following defects:

   a. The copper tracks on an electronic printed circuit board being accidentally joined after a soldering process, or

   b. a defective component that was not found during the initial inspection of the batch of components.

2. The occasional occurrence of a run of products with a high fraction of defective parts. This run of defective products could be due to the acceptance of a "bad" batch of components during the initial inspection.

Two types of process curves can therefore be used to represent the distribution of defective products in a production run. These two process curves are classified as TYPE-A and TYPE-B.

5.2.2 TYPE-A PROCESS CURVE

This process curve represents the distribution of defective products as a binomial distribution (ie. due to defective products being produced under assumption (1)) - see figure 12.

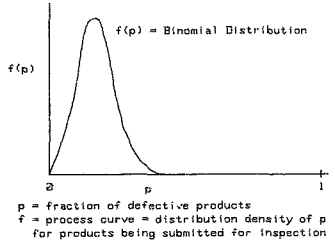


Figure 12. TYPE-A Process Curve.

A PDL description of the procedure used to generate a TYPE-A process curve is given in Appendix B1.

5.2.3 TYPE-B PROCESS CURVE

This process curve is represented by a two-humped distribution, the humps lying widely apart (ie. due to defective products being produced under *assumption (1) and assumption (2)*) - *see figure 13* (*Horsnell, 1954*). This type of process curve is referred to as a Horsnell distribution.

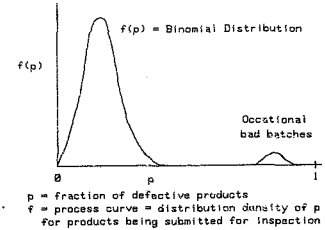


Figure 13. TYPE-B Process Curve.

The occasional bad batch gives rise to the spotty quality of a production run. A PDL description of the procedure used to produce a TYPE-A process curve is given in Appendix B2.

5.3 PRODUCT TO BE INSPECTED

5.3.1 BACKGROUND TO THE D.P.S. MODULE

The product chosen to be inspected forms part of the Disa Plan System (D.P. System). A brief description of where the product chosen (the D.P.S. module) fits into the D.P. System, and the basic operation of the system is given.

The D.P. 2/4 System is the Central Control Unit (CCU). Connected to the CCU are two exchange lines and four stations (for maximum configuration). A block diagram of the D.P. 2/4 System is given in figure 14.

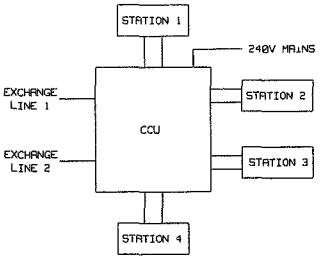


Figure 14. Block Diagram of the D.P. 2/4 System.

The Disa Plan 2/4 System offers the following facilities:

1. Outgoing calls to each exchange line by any station.

2. Receipt of calls from either exchange line by any station.

3. Intercom between any two stations without using the exchange lines.

4. Secretarial facilities eg. call transfer and call divert.

A more detailed description of the D.P. 2/4 System is given. A block diagram of the system is given in figure 15.

A station is a telephone instrument containing the normal telephone circuitry (speech circuitry) and the D.P.S. Module (digital circuitry).

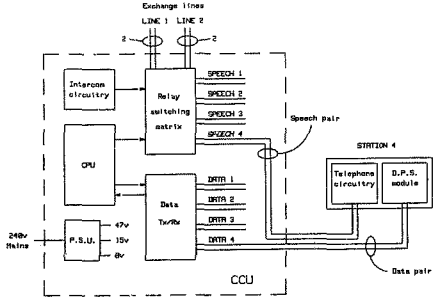


Figure 15. Detailed Block Diagram of the D.P. 2/4 System.

The CCU has two distinct sections; the normal telephone speech section and the digital section. The telephone speech section consists of three parts

1. The Relay Switching Matrix; this switching matrix is used to interconnect the station speech lines and the exchange lines under the control of the CPU on the CCU.

2. The intercom circuitry; controlling calls between stations without first going to the exchange.

3. The Telephone Circuit; this circuit is the normal speech circuit encountered on each telephone instrument:

The digital section consists of three parts:

1. The CPU; controls the activities of the various parts of the system.

2. The Data Tx/Rx Circuit; is used to send data to and receive data from the D.P.S. Module on each station.

3. The D.P.S. Module; is used to communicate the desired operations to the CCU and indicate the action taken, by the CCU, to the user.

The communication between the CCU and the stations takes place in a "round robin" fashion. The CCU first sends data to one station (the D.P.S. module on the station). This data may be "null data" ie. no output action is initiated. The station then responds with information concerning the operation of the KEY's that were pressed (if any). The "null data" is used to receive the action desired by the user when the CCU requires no output action from the D.P.S. module. This procedure is then repeated for each station. When the CCU has prompted all four stations, it starts again from the first one.

The serial communication between the CCU and the stations is illustrated in figure 16.

5.3.2 BASIC OPERATION OF THE D.P.S. MODULE

The D.P.S. module has no intelligence at all. The microprocessor on the D.P.S. module receives the serial data from the CCU (on a two wire communication bus or "data pair" - see figure 14). The data is used to determine which LED is to be turned on or off (indicating the action taken by the CCU to the user). The microprocessor then returns information concerning the KEYs that where pressed (indicating the action desired by the user to the CCU). A block diagram of the D.P.S. module is given in figure 17.

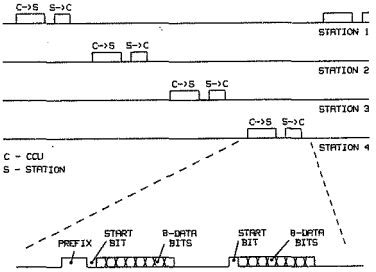


Figure 16. Serial Data Diagram for the D.P. 2/4 System.

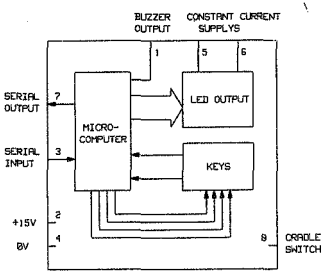


Figure 17. Block Diagram of the D.P.S. Module.

Each LED and KEY is associated with a specific function of the D.P.S. module. A table giving the bit assignments of the 8 bit data word is given below.

Bit Assignment Table		
Input Data	Output Data	Function
BITNO	KEYNO	
1	1	Line 1
2	2	Line 2
3	6	Divert
4	HK.SW	HK.SW
5	3	Station 1
6	4	Station 2
7	5	Station 3
8	7	Hold
5,6	3,4	Recall

5.3.3 DEFINING THE TESTS

A number of tests must be devised using the D.P.S. module. The complexity of the tests must be kept to a minimum, as the principle behind the sampling plans, and not the actual tests being performed, is being evaluated.

A number of simple functional checks can readily be implemented. The D.P.S. module has eight LED's each with a corresponding KEY (switch). Seven LED's can be instructed to switch on or off by sending the appropriate data (eight bit value in serial form) to the D.P.S. module.

Eight tests are devised. Seven tests will take the following form:

1. Send data to the D.P.S. module instructing an LED to turn on

2. Press the KEY corresponding to that appropriate LED.

3. Receive the result from the D.P.S. module (in serial form) and check that the correct KEY was pressed.

The additional test can be implemented by lighting up two LED's in which case the two KEY's corresponding to the LED's must be pressed.

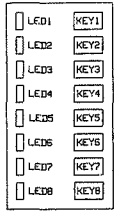
A failure of one of the tests is simulated using the following procedure:

1. Send data to the D.P.S. module instructing an LED (or two LED's) to turn on.

2. Do not press the correct KEY (or KEY's).

3. If the correct result is not received from the D.P.S. module within a predefined time period, the test is assumed to be faulty.

A table that summarises the required action of each of the eight tests is given below. LEDNO gives the number of the LED that turns on for the test and KEYNO gives the number of the KEY that must be pressed to ensure the "test pass" result.

TESTNO	LEDNO	KEYNO	D.P.S. MODULE
1	LED 1	KEY 1	
2	LED 2	KEY 2	
3	LED 6	KEY 6	
4	LED 3	KEY 3	
5	LED 4	KEY 4	
6	LED 5	KEY 5	
7	LED 8	KEY 7	
8	LED 1 & LED 2	KEY 1 & KEY 2	

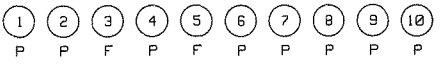
5.4 DESIGN OF THE EVALUATION PROCEDURE

5.4.1 SIMULATING A PRODUCTION RUN

A production run of 10000 products will be used in the evaluation of the continuous inspection plans, while a production run of between 500 and 700 will be used in the evaluation of the batch inspection plans. It is not economically feasible and is impractical to set up a production run of so many individual products.

One product, which is repeatedly inspected is used to simulate a production run. An example of how this can be achieved is given below.

Assume that a particular production run consists of ten products, and that two of the ten products are defective. This production run can be represented by the following schematic model (see figure 18).



P - PASS (Product in working order)
F - FAIL (Defective product)

Figure 18. Schematic Model of a Production Run.

This production run can be simulated using a single product and the following procedure.

```
REPEAT:
  IF ( the product must fail )
    THEN
      * set the product up so that it will fail
        when it is inspected *
  ...F:
  * inspect the product and adjust the sampling
    plan according to the test outcome *
UNTIL ( all the products have been simulated )
```

Each time a product is inspected, information regarding the failure of that particular product (in the production run) must first be used to ensure that the particular test on the product will fail if it is inspected. We therefore require some means of controlling the passing or failure of a particular test.

The information regarding the failure of a particular product is available in an array set up when simulating the process curves (see Appendix

B). This array contains a list of numbers of the products that must fail and the number of the test that must fail on each of these products.

An example of this array is given for the production run used in the previous example. There are two defective products in the production run of 10 products. Assume that test 1 of product number 3, and that test 3 of product number 5 must fail when they are inspected.

PRODUCTNO	TESTNO
3	1
5	3

Before inspecting the third product, first ensure that test number 1 will fail if it is inspected, before the sampling plan inspects the product. The same procedure is followed when inspecting product number 5. The other eight products contain no defects.When any of these products are inspected, ensure that every test on each product will work.

5.4.2 MEASUREMENT OF INSPECTION TIME

The time taken to inspect the products in a production run is one of the parameters used when selecting a sampling plan. The total time taken to inspect the products, in a production run, must therefore be recorded.

The D.P.S. module is the product being inspected. This product was arbitrarily chosen, and the tests designed for the D.P.S. module form only a small part of the total number of tests required to successfully inspect it. It is therefore not meaningful to measure the actual time (in seconds) spent on inspecting the D.P.S. module.

An arbitrary unit is introduced for the measurement of the time taken to inspect the product. This can be done because we are only interested in the performance of each sampling plan relative to the other sampling plans. The time to inspect a specific test on a product is categorised as follows:

1. 0 Units for a test that is not inspected (ie. NOTIME = 0 units).

2. 1 Unit for a test that is inspected and the result is correct (ie. PASSTIME = 1 unit).

3. 2 Units for a test that is inspected and fails (ie. FAILTIME = 2 units). The increase in inspection time for a test that fails can be attributed to the need to implement fault diagnostic procedures.

The time spent on inspecting a product is the sum of the time spent on inspecting the individual tests. The total time taken by a sampling plan to inspect the entire production run is obtained by summing the time spent on inspecting the individual products in the production run.

This method of time representation can only be used to compare the relative efficiencies of the sampling plans. For example : Plan 1 has taken a total time of 3000 units and Plan 2 has taken a total time of 4000 units when inspecting the same production run. If the required AOQL is not exceeded in both cases, Plan 1 is 25% more efficient than Plan 2.

5.4.3 THE EVALUATION PROCEDURE

The following procedure must be followed to ensure a fair comparison between the various inspection plans being evaluated.

1. Generate one process curve for each trial. A TYPE-A or TYPE-B process curve can be used. TYPE-B process curves will be used more frequently because they simulate spotty quality products in a production run.

2. Select the parameters required by each sampling plan so as to ensure an equal AOQL value for each trial.

   NOTE : A **sampling plan** was defined in chapter 4 as consisting of a number of inspection plans operating in parallel, one for each test on the product. To be able to ensure an AOQL value, the parameters of each inspection plan must be chosen to ensure an $AOQL_t$ value (AOQL for each test) for each test. The $AOQL_t$ value is calculated for a given $AOQL_p$ (AOQL for the production process) using equation (3) of chapter 4

   $$AOQL_t = AOQL_p \; / \; MAXTEST$$

   where: MAXTEST is the number of tests on the product (MAXTEST = 8 for the D.P.S. module).

3. Each sampling plan now inspects the production run simulated in step (1). One trial is complete when all the sampling plans being evaluated have completely inspected the common production run.

4. Repeat steps (1) to (3) for a number of trials, using a different process curve for each trial.

5. After completing a sufficient number of trials, the relative performance of each sampling plan can be assessed for each trial. The plan that performs the best over all the trials can be selected for implementation.

   The sampling plan that ensures the lowest AOQ with the least time spent on inspection is the sampling plan that performs the best for each trial.

5.5 SYSTEM REQUIREMENTS

The sampling plan evaluation system must be designed. There are a number of requirements that this system must meet. The sampling plan evaluation system is designed around an IBM Personal Computer (PC). The design of the system can therefore be split into two sections; the hardware design and the software design.

5.5.1 HARDWARE REQUIREMENTS

The D.P.S. module

The D.P.S. module requires a number of external components before it will operate. This circuitry must be designed and built. Once this external circuitry is built it is very simple to operate the module. There are two serial channels, one an input to and an output from, used to communicate between the D.P.S. module and the interface unit. The LED's can be controlled by sending the appropriate data, using the correct serial format, to the D.P.S. module. Information concerning the KEY being pressed can be received on the serial output.

The Interface Unit

The PC must be able to control the operation of the D.P.S. instrument. An interface between the PC and the D.P.S. module must therefore be designed and built. The interface unit must satisfy the following requirements:

1. The PC sends one byte of data to the interface unit. This parallel word must be converted to a serial word of the correct format and sent to the D.P.S. module. This conversion is done by the interface unit.

2. The D.P.S. module sends serial information to the interface unit. This information must be converted to parallel data before it can be sent to the PC.

3. The interface unit must have the means whereby the PC can control the tests to fail during the simulated production run.

4. The protocol used, for communicating between the PC and the interface unit, must be designed.

Automatic KEY Operation

When long test runs are carried out it is virtually impossible to press the correct KEY manually each time an LED turns on. Therefore a method must be devised that automatically presses the correct KEY.

5.5.2 SOFTWARE REQUIREMENTS

The program that is written must have the following features :

1. The software must be written following a modular approach. Each module can then be changed without affecting the operation of the remainder of the software.

2. The two process curves must be generated. A method which allows manual input of process curve data is also required. This will be used to enter data obtained from the production lines.

3. The parameters required by the various modules must be entered on the keyboard when an appropriate prompt is received.

4. The five continuous inspection plans and the two batch inspection plans must be implemented.

5. All the sampling plans must be able to use a common process curve.

6. The results of the various sampling plans can be monitored in two ways. They can be displayed on the video display unit or they can be printed.

7. The tests that must fail during the simulated production run, must be controlled using the system software.

5.6 DISCUSSION

The procedure that will be used to evaluate the various sampling plans has been outlined. This procedure will be used to chose a sampling plan that best suits the production process as defined by the two process curves.

The requirements of the sampling plan evaluation system have been listed. The detailed implementation of this system is discussed in Chapter 6 (System Hardware Development) and Chapter 7 (System Software Development).

6.0 SYSTEM HARDWARE DEVELOPMENT

6.1 EXTERNAL CIRCUITRY FOR THE D.P.S. MODULE

The D.P.S. module was introduced earlier (in chapter 5). The D.P.S. module requires external circuitry to enable it to operate. There are eight connections to the D.P.S. module, the function of each of these is listed below.

Connection	Function

1. Buzzer Output: When this output goes high the buzzer must sound.
2. +15 v Supply.
3. Serial Input: The serial input data must have the correct format and magnitude. The serial input data must swing between 0 and +15 v, for "0" and "1" respectively.
4. Ground (0 v).
5. Constant current supply of approximately 15 milliamperes to drive four LED's.
6. Constant current supply of approximately 15 milliamperes to drive the other four LED's.
7. Serial Output: The serial output signal also has a 15v swing.

8. Cradle Switch Control: This line must be pulled low, through a resistor, to represent the receiver "on hook".

The interface unit operates from a 5v TTL power supply. The serial input signal to the D.P.S. module must therefore be converted from a 5v TTL signal to the required 15v signal and the serial output signal must be converted from a 15v signal to a 5v TTL compatible one. The constant current supply is designed around the standard circuits used at TM, with the resistor values changed to ensure the correct current. An LED replaces the buzzer. The buzzer and cradle switch circuitry is also designed around the standard circuits used at TM.

A circuit diagram showing the external circuitry required by the D.P.S. module is given in Appendix C1 (Sheet 3). The external circuitry was built on vero-board (VERO 1 - see Appendix C1, sheet 7).

6.2 INTERFACE UNIT

6.2.1 GENERAL APPROACH

The format used to send serial data to the D.P.S. module is not standard (see figure 21). It is therefore not possible to use standard components (integrated circuits; USART's etc.)to transfer the serial data. It was decided to use a microprocessor to perform the task of sending the serial data to the D.P.S. module, and receiving the serial data from the D.P.S. module. The microprocessor will also control the exchange of data between the PC and the interface unit.

A block diagram of the system is given in figure 19.

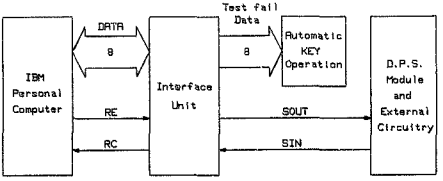


Figure 19. Block Diagram of the System.

The design and function of the Automatic KEY Operation block will be described later on in this chapter.

6.2.2 COMMUNICATION PROTOCOL

The communication between two electronic modules is based on a number of rules, or a Protocol. Two communication protocols are required by this system, one for the communication between the PC and the Interface Unit and the other for the communication between the Interface Unit and the D.P.S. module (and it's external circuitry). The operation of these protocols will now be discussed in more detail.

PC - Interface Unit Protocol

Both the PC and the Interface Unit have microprocessors on board. The *protocol must allow the two microprocessors to communicate to one another* whenever requires, without interrupting either microprocessor when it is executing a critical operation.

In this case, the PC will always initiate the communication. Once the PC has placed data in a buffer on the interface unit, it informs the microprocessor on the interface unit that the data is available. The PC now waits until the microprocessor on the interface unit has returned an answer before it proceeds with executing the program. Two control lines, RC and RE, are required to perform this task (see figure 19). A PDL description and a timing diagram of the protocol is given in figure 20.

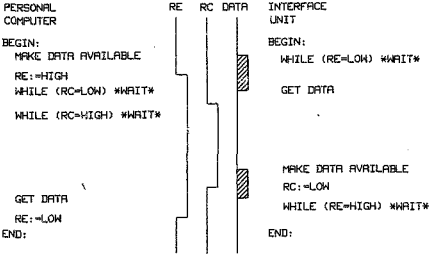


Figure 20. PC - Interface Unit Communication Protocol.

Interface Unit - D.P.S. Module Protocol

The protocol used for communication between the interface unit and the D.P.S. module (and external circuitry), is prescribed by the D.P.S. module operational requirements. The microprocessor on the interface unit must send the data (received from the PC) to the D.P.S. module in the correct format. The microprocessor then scans the serial input line (from the D.P.S. module) for a start bit. Once it has received the start bit,

the serial data (following the start bit) is read. A diagram showing the format and the timing requirements of the incoming and outgoing serial signal is given in figure 21.

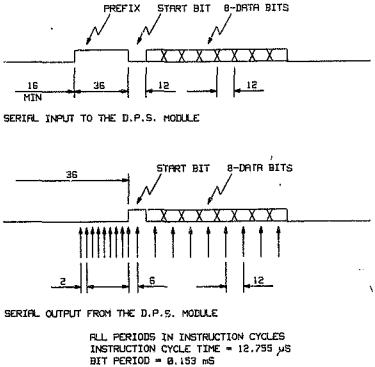


Figure 21. Interface Unit - D.P.S. Module Communication Protocol.

6.2.3 INTERFACE UNIT OPERATION

The interface unit is built on an IBM prototype card. A large area of the card is set aside for wirewrapping a user designed circuit. A D-shell connector is used to connect the prototype card to the D.P.S. module (and it's external circuitry). A block diagram of the existing circuitry,

on the prototype card, is given in figure 22. A detailed circuit diagram of the prototype card can be found in Appendix C1 (sheet 1).

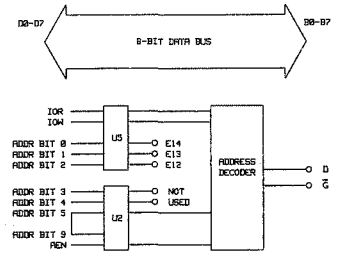


Figure 22. Block Diagram of the Prototype Card.

U2 and U5 are buffers. They buffer the address bus and control lines of the of the PC. The data bus buffer (U1) is not used. Address bits 0, 1, and 2 (E14, E13, and E12 respectively) are used by the interface unit.

A block diagram of the interface unit is given in figure 23. A detailed circuit diagram of the interface unit can be found in Appendix C1 (sheet 2) and the component layout is given in Appendix C1 (sheet 6).

U8 and U9 are eight bit latches (configured as a bye-directional latch), used to pass data between the PC and the microprocessor on the interface unit. U10 contains a pair of J/K flip-flops. These J/K flip-flops are connected in such a manner as to ensure that both the outputs (Q) toggle on a positive going edge of their respective control signals. The J/K flip-flops are cleared (outputs set low) when the system is manually reset or at power up. U10a generates the RE control line while U10b generates the RC control line. The status of the RE control line is read by

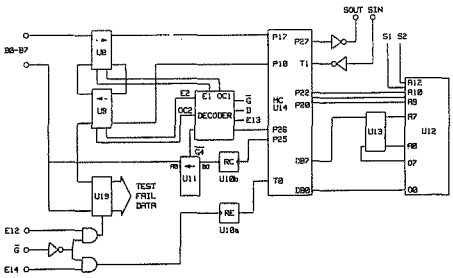


Figure 23. Block Diagram of the Interface Unit.

to T0 input of the microprocessor on the interface unit. The RC control line is buffered from the data bus of the PC using U11.

U14 is an 8035 microprocessor. The 8035 requires an external EPROM to store the program that determines it's operation. U13 is used to latch the lower byte of the address. Lines P20 - P22 (of port 2), together with the output of the two switches (S1 and S2) supply the additional address lines required by the EPROM. U12 is an 8 bit by 8K EPROM. S1 and S2 are used to select one of four 2K segments of the 8K EPROM. The 8035 can only address 2K of external memory. Four different programs therefore be stored on the EPROM, one in each of the 2K segments. S1 and S2 are used to select the program that will be executed.

U19 is a eight bit latch. This latch is used to hold the information regarding the test that must fail. The operation of this latch and the significance of the data it holds is discussed in detail when describing the operation and design of the Automatic KEY Operation Unit.

The serial information is sent to the D.P.S. module using bit 7 of port 2 (of the 8035 microprocessor). An inverting buffer is used to insulate this output (P27) from the D.P.S. module. The information received from the D.P.S. module is buffered (and inverted) before being read by the T1 input of the microprocessor.

The decoder controls the operating sequence of the various buffers and integrated circuit. The inputs to the decoder are controlled by the PC and the 8035. The detailed design of the decoder can be found in Appendix C3. A circuit diagram of the decoder is given in Appendix C1 (sheet4).

6.2.4 ADDRESSING THE INTERFACE UNIT

The prototype card is addressed using I/O mapped address space. The address space assigned to the prototype card ranges from 300H to 31FH (where H represents a Hexadecimal number). The address decoding circuit an the prototype card and the interface unit circuitry will be referred to as the interface unit in future.

It must be possible for the PC to initiate a number of operations on the interface unit. Two hypothetical functions are introduced to represent the communication between the PC and the interface unit. These functions are:

1. Read data from the interface unit.

   X = IN (address)

where "address" must be in the range 300H to 31FH and X is the byte of data read from that address.

2. Write data to the interface unit.

```
OUT address,X
```

where "address" must be in the range 300H to 31FH and X must be in the range 0 - 255.

The operations that the PC must perform are listed below. The list is derived from the communication protocol requirements introduced before (see figure 20).

1. Make Data Available - A byte of data (X) must be sent to the interface unit. This data is used by the interface unit to determine which test must be inspected. This operation is carried out using the following function:

```
OUT 300H,X
```

2. Set *RE* - After the system is reset, the output of the *RE flip-flop* (U10a) is low. The output of the RE flip-flop is toggled using the following function:

```
OUT 303H,X
```

X is a "dummy" data value which has no influence on the operation of the system. Each time this function is used the output of the RE flip-flop changes state.

3. Read RC - The value at the output of the RC flip-flop (U10b) is read using the following function:

```
X = IN ( 300H )
```

The least significant bit (bit 0) of the data value received (X) is the output of the RC flip-flop.

4. Get Data - Read the result of the test. This data is available on port 1 of the microprocessor on the interface unit. This operation is performed using the following function:

   X = IN (302H)

   X is the result of the test that was performed.

A short description of the procedure used when communicating to the interface unit is given below (see figure 20).

```
PROCEDURE: Interface Unit Process

INPUT : The data required to inspect a specific test is
        placed in TESTDAT
OUTPUT: The result of the test is output. If STATUS = FFH
        the test was successful. If STATUS = 00H, the test
        failed
CONSTANTS:
        MDA = 300H        RDRC = 300H
        SETRE = 303H      RDDAT = 302H

BEGIN:
  * make data available *
      OUT MDA,TESTDAT
  * set RE high *
      OUT SETRE,00H
  WHILE ( RC is low ) * wait *
      RC = (IN ( RDRC )).AND.01H
  ENDWHILE
  WHILE ( RC is high ) * wait *
      RC = (IN ( RDRC )).AND.01H
  ENDWHILE
```

```
  * read answer *
      STATUS = IN ( RDDAT )
  * set RE low *
      OUT SETRE,00H
END:

END PROCEDURE:
```

6.2.5 INTERFACE UNIT SOFTWARE

The microprocessor on the interface unit must have a program that it can execute to control the operation of the interface unit. This program is written in the form of a main program which calls two procedures, OUTP27 and SCANT1.

OUTP27 is a procedure that takes an eight bit word and converts it to the correct serial format (adding a prefix and a start bit to the serial version of the eight bit word). This serial information is sent to the D.P.S. module, output on P27 (bit 7 of port 2) of the microprocessor. The format and timing details of the serial output data is given in figure 21. SCANT1 is a procedure that receives serial information (on input T1) and converts it to an eight bit data word. The serial information is sent by the D.P.S. module. The procedure first waits for a start bit, it then samples the centre of each data serial data bit and uses this sampled information to build up an eight bit word. The format and timing details (and sampling points) of the serial input data, is given in figure 21.

The main program controls the action taken by the microprocessor. On power-up (or manual reset) the RC and RE flip-flops must be reset. This is done by producing a pulse on P24 (high-low-high), of sufficient duration, to reset the flip-flops. (See Appendix C1, sheet 2). The output of the RC flip-flop is toggled each time a pulse is outputted on P25.

A PDL description of the main program can be found in Appendix C4.1. The interface software was developed on a HP 64000 microprocessor development system. A listing of this software is given in Appendix C4.2.

6.3 AUTOMATIC KEY OPERATION

6.3.1 GENERAL APPROACH

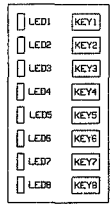
When performing a test, the PC instructs an LED to turn on. The KEY corresponding to this LED must be pressed to produce a "test pass" result. If the KEY is not pressed within a predetermined time interval, a "test fail" result is recorded. When long inspection runs are carried out out, it is virtually impossible to press the KEY manually each time a correct result is required. Additional hardware is designed allowing the automatic operation of the KEY corresponding to the LED that was turned on.

The eight tests were previously defined in Chapter 5. A new term, the **test-fail data** , is introduced. The test -fail data consists of an 8-bit word. Each bit in the word corresponds to a test. A table summarising the required action to be taken for each test and the test-fail data bit allocations to the test, is given on the following page.

When a particular test is required to fail, the bit corresponding to that test (in the test-fail data) is low. When the test must pass, the bit corresponding to that test is high.

The approach adopted when performing tests is outlined. The test-fail data for the first seven tests is first sent to the Automatic KEY Operation Unit. Test 1 to test 7 can now be carried out. The test-fail data for test 8 must first be sent to the Automatic KEY Operation Unit before test 8 is carried out.

TESTNO	LEDNO	KEYNO	TEST-FAIL DATA BITNO	D.P.S. MODULE
1	LED 1	KEY 1	1	
2	LED 2	KEY 2	2	
3	LED 6	KEY 6	6	
4	LED 3	KEY 3	3	
5	LED 4	KEY 4	4	
6	LED 5	KEY 5	5	
7	LED 8	KEY 7	8	
8	LED 1 & LED 2	KEY 1 & KEY 2	1 & 2	



Note:

1. Bit 1 of the test-fail data byte is the most significant bit while bit 8 is the least significant bit.

2. When LED 8 turns on KEY 7 must operate, and LED 7 is not used at all.

6.3.2 *AKO UNIT OPERATION*

A block diagram of the additional hardware, used to operate the KEY's automatically, is given in figure 24.

The test-fail data is held in a batch (U19). A light dependant resistor (LDR) is used to detect whether a LED is turned on or off. A resistor is placed in series with the LDR. When the LED is on, the resistance of

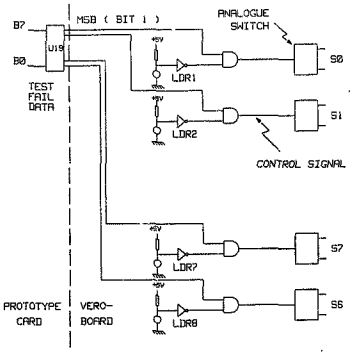


Figure 24. Block Diagram of the AKO Unit.

the LDR is low and the voltage drop across it is small. When the LED is off, the resistance of the LDR is high and the voltage drop across it is large. The resistor is chosen so as to ensure the correct voltage swing between the two extremes. This voltage swing drives a schmitt triggered inverter which produces an output signal with fast transition time.

The test-fail data and the output of the detection circuit is used to produce a signal to drive the analogue switches. The analogue switches are used to close the KEY circuit, simulating the pressing of the KEY. When the control signal is low, the resistance between the two switch contacts of the analogue switch is high, representing an open circuit. When the control signal is high the input resistance is low, representing a closed circuit.

A circuit diagram of the automatic KEY operation unit is given in Appendix C1 (sheet 5). The D.P.S. module requires a 15v supply. The analogue switches must also operate from the 15v supply. The control signals required by the analogue switches must be amplified from 5v to 15v to ensure their correct operation. The automatic KEY operation circuitry is built on vero-board. The LDR's and their resistors are mounted on VERO 3 while the remai .er of the circuit is divided between VERO 1 and VERO 2 (see Appendix C1, sheet 7 and sheet 8). The test-fail data is transferred from the interface unit to the external circuitry using the D-shell connector.

6.3.3 ADDRESSING THE AKO UNIT

The test-fail data latch is mounted on the interface unit. The test-fail data is sent to the interface unit using the following function.

```
OUT 304H,X
```

X is the test-fail data and 304H is the address corresponding to the test-fail data latch. The operation of this unit is completely separate from the operation of the interface unit, only the address decoding facilities of the interface unit (prototype card) is used.

7.0 SOFTWARE DEVELOPMENT

7.1 GENERAL APPROACH

A top-down structured design approach is used in the development of the Statistical Inspection System. The concepts of Data Flow and Task Management are use to describe the overall operation of the system. Program Description Language (PDL) is used to give a more detailed insight into the operation of each of the modules that constitute the system. A review of the Data Flow and Task Management technique is given in Appendix D. Information regarding the detailed use of PDL can be found in Walker (1984).

7.2 STATISTICAL INSPECTION SYSTEM SOFTWARE

7.2.1 STATISTICAL TEST PROCESS

The system was designed keeping user friendliness in mind. It is for this reason that the entire system is menu driven. Information required by the system is input using the keyboard after the user receives the appropriate prompt on the display. The Statistical Test Process is graphically represented by the process-resource diagram in figure 25. A PDL description of the operation of this process is given in Appendix E1.

The Interface Unit Resource is used to interface between the Personal Computer and the D.P.S module. A detailed description of the operation of this resource is given later on.

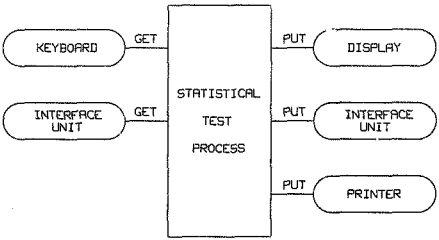


Figure 25. Statistical Test Process\

The Statistical Test Process can be divided into three activities.

- Data Input Process
- Sampling Process
- Data Output Process

Process - Resource Diagrams

These activities can be graphically represented using the following process-resource diagrams.

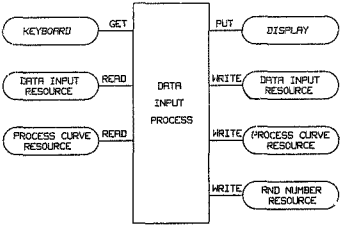


Figure 26. Data Input Process.

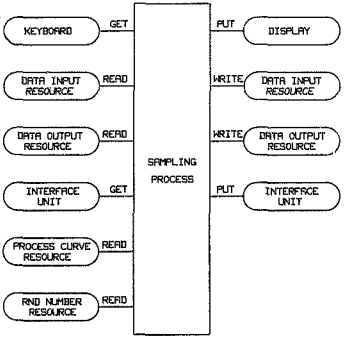


Figure 27. Sampling Process.

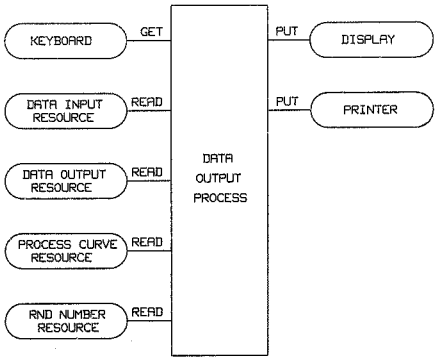


Figure 28. Data Output Process.

7.2.2 SYSTEM RESOURCES

Data Input Resource

The information required by each Sampling Plan regarding the type of algorithm to be implemented, The sample size and the filename (of the Data Output Resource) of the file in which the results are to be stored, is contained in a record. The Data Input Resource consists of up to 10 of these records. A description of the fields in the record is given below.

```
Record:
  SPLAN   : Character
  DOFN    : Character
  SSIZE   : Integer
  AOQL    : Real
  I       : Integer
  F       : Integer
  SSTATUS : Character
Endrecord:
```

where

SPLAN - Sampling plan required to run eg. 100% or CSP-1.

DOFN - Name of the random file in which the results of the specific sampling plan are stored.

SSIZE - Size of the sample to be run.

AOQL - Average outgoing quality level.

I and F - Sampling plan constants.

SSTATUS - This field is assigned FALSE if the test has not been run. After running the test a value of TRUE is assigned to SSTATUS.

Process Curve Resource

This resource contains a maximum of 10 records. Each record contains information concerning the type of process curve required, the name of the file in which the information is to be stored, the number of defective products and the probability of finding a defective products in a sample of 100 units. Each record consists of the following fields.

```
Record:
  PROCC   : Character
  PCFN    : Character
  PSIZE   : Integer
  PROB    : Real
Endrecord:
```

Data Output Resource

The results of the Sampling Plan are temporarily kept in the Data Output Resource before it is stored in a random file (of the same name) on the floppy. This resource is only 25 records long. When all 25 records are filled the information is then dumped to the floppy. The old information is then overwritten with more recent results. Each record consists of the following fields.

```
Record:
  TTYPE   : Character
  COUNT   : Integer
  TSTATUS : Character
  OUTCOME : Character
  TIME    : Integer
  TNUM    : Integer
  PSTATUS : Character
  PTIME   : Integer
  PFAIL   : Integer
Endrecord:
```

where

TTYPE - Type of testing being performed ie. PART or FULL.
COUNT - Keeps count of T's or F's.
TSTATUS - Should the specific test be run or not ie. RUN or NORUN.
OUTCOME - Has the specific test passed or not ie. PASS or FAIL.
TIME - Time taken to run the specific test.
TNUM - Number of the specific test.
PSTATUS - Has the specific product passed or not ie. PASS or FAIL.
PTIME - The time taken to run all tests in the product.
PFAIL - Number of the test that is to fail.

Random Number Resource

The Random Number Resource contains a list of product numbers and corresponding test numbers. The product numbers represent those products which

have a failure in one of there tests. The product numbers are randomly chosen from a range of 1 to SSIZE. The test numbers to fail are also randomly generated from a range of 1 to MAXTEST where MAXTEST is the number of tests per product. MAXTEST = 8 in this case. The information is stored on the floppy under the filename stored in the PCFN field of the Process Curve Resource. When this information is required it is loaded into the Random Number Resource. Each of the PSIZE records have the following fields.

```
Record:
    PNO     : Integer
    TNO     : Integer
Endrecord:
```

where

PNO - Product number and TNO - Test number

Interim Resource

The Interim Resource contains the information required by the Algorithm Process. This information is used to determine the type of operation (PART or FULL), the test status (RUN or NORUN), the present COUNT, COUNTK, FLG1, and FLG2 values, the outcome of the present test (PASS or FAIL) and the time taken to complete the test. The Interim Resource consists of MAXTEST or 8 records with the fields described below.

```
Record:
  TTYPE    : Character
  TSTATUS  : Character
  OUTCOME  : Character
  COUNT    : Integer
  COUNTK   : Integer
  FLG1     : Integer
  FLG2     : Integer
  TIME     : Integer
Endrecord:
```

7.2.3 DATA INPUT PROCESS

The Data Input Process is divided into a further five activities. A PDL description of the Data Input Process operation together with the PDL description of the five activities is given in Appendix E2. The five activities are listed below.

- Data Input Process A : Read values in the Data Input Resource.
- Data Input Process B : Read values in the Process Curve Resource.
- Data Input Process C : Input required values.
- Data Input Process D : Erase a record.
- Data Input Process E : Quit.

Process - Resource Diagrams

These activities can be graphically represented using the following process-resource diagrams.

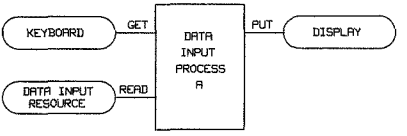


Figure 29. Read Values in the Data Input Resource.

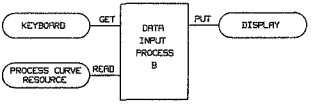


Figure 30. Read Values in the Process Curve Resource.

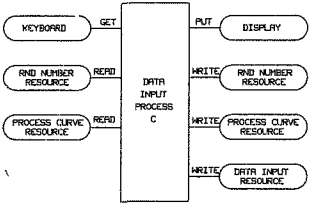


Figure 31. Input Required Values.

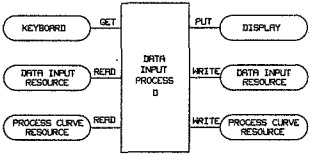


Figure 32. Erase a Record.

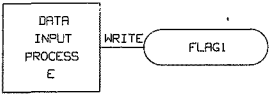


Figure 33. Quit.

Data Input Process A

This module prints the contents of the Data Input Resource on the Display. The displayed information is correctly formatted and lined up under the correct headings.

Data Input Process B

This module prints the contents of the Process Curve Resource on the display. The displayed information is correctly formatted and lined up under the correct headings.

Data Input Process C

This process is used to prompt the user for the information required by the various sampling plans. These values include the sample size, the type of sampling plan eg. 100% or CSP-1, the type of process curve to be used, the names of the files in which the results are to be stored, etc.

A module called CALC_P_C is called by this process. This module is used to set up the Random Number Resource. One of the following three options are used to produce the process curve:

- SELF: Input product numbers manually while randomly generating test numbers.

- TYPE-A: PSIZE product numbers are randomly generated according to a binomial distribution. A randomly selected test number is also generated.

- TYPE-B: PSIZE product numbers are randomly generated according to a binomial distribution. A randomly selected test number is generated for each product number. The system then prompts the user for the proportion of SSIZE and the distribution of spotty quality products. The spotty quality products are then generated for a single test on the product.

The product numbers and there corresponding test numbers are then sorted from smallest product number to largest product number and stored in a random file on the floppy. The name of this file corresponds to the name stored in the PCFN field of the Process Curve Resource.

Erase a Record

This process erases a record in both the Process Curve Resource and Data Input Resource corresponding to an ID supplied by the user.

Quit

This process provides the information required to return to the Statistical Test Process in the previous level.

7.2.4 SAMPLING PROCESS

The Sampling Process is divided into a further two activities. A PDL description of the Sampling Process operation together with the PDL descriptions of the two activities are given in Appendix E3. The two activities are listed below.

- Algorithm Process.

- Test Result Process.

These activities are graphically represented using the process-resource diagrams of figure 34 and figure 35 respectively.

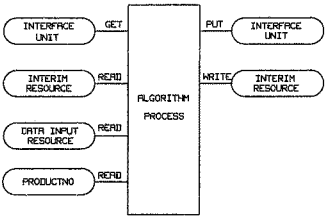


Figure 34. Algorithm Process.

Algorithm Process

This process executes the required sampling plan according to specific rules as specified by the algorithm. The sampling plan uses the information in the Interim Resource to run in the correct mode and also stores the results of this test in this resource. The results are used to determine the mode of operation for the next product. Seven algorithms have been implemented to date, they are the three Dodge continuous inspection plans (CSP-1, CSP-2, and CSP-3), the two Multi- level continuous inspection plans (MLP and MLP-T), and the two Batch inspection plans (CSP-F and WSP-1).

- Test Fail Process.

This module reads information form the Random Number Resource. If a fault is required to occur in the specific product the appropriate information is sent to the Interface Unit.

- Interface Unit Process.

This process is used tc pass information, about the test being performed, to the microprocessor in the interface unit. The result for the test is then is then received by this process.

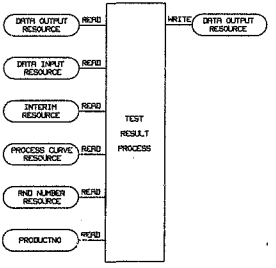


Figure 35. Test Result Process.

Test Result Process

This process is used to store the results of one specific test run and the overall results of the most recent product in the Data Output Resource. Once 25 results are available, the information is transferred to a random file on the floppy. The name of this file corresponds to the name stored in the DOFN field of the Data Input Resource. The total number of defective products detected and the total time taken to test the entire

batch are also calculated. These result are stored in a random file on the floppy.

7.2.5 DATA OUTPUT PROCESS

The Data Output Process is used to output the results of the tests to the user The results can either be output on the display or the printer. Four options are available to the user, they are:

- Output a datafile to the printer.
- Output a datafile to the display.
- Output a rnresource to the printer.
- Output a rnresource to the display.

The datafile refers to the results of the algorithm stored on the floppy. Rnresource refers to the list of product numbers and corrasponding test numbers stores on the floppy. A PDL description of this process can be found in Appendix E4.

7.3 LABELLING CONVENTIONS

Data labels or identifiers are represented as follows:

- Global or Module Global identifiers have a name which explains it's global use. A global identifier is on that is available to the entire program while a module global identifier is only used in a specific module and modules called by it.

- Local identifiers consist of three parts

  - The type of data ; I-integer, C-character, R-real.

  - A central part consisting of the first letter of the process in which it is used.

  - A number from 1 to n, where n is any positive integer number

  An example of such an identifier is gives as follows : The first identifier for a character in the Data Input Process is CDI1.

7.4 DISCUSSION

The system software was developed and operates on an IBM Personal Computer. The program is written in BASIC. The system stores all required information on the floppy. This ensures that the user does not have to input all the information each time the the program is run.

Because the arrays used by the system software are large, it was found that the PC ran out of memory if all seven sampling plans are combined into one program (BASIC is restricted to the use of 64K of the PC memory). The sampling plans are now distributed between three system programs.

1. Dodge Continuous Inspection Plans - This software is stored under the filename of DSYSTEM on the system floppy disc. Sampling plans 100%, CSP-1, CSP-2, and CSP-3 are available.

2. Multi-level Continuous Inspection Plans - This software is stored under the filename of MSYSTEM on the system floppy disc. Sampling plans 100%, MLP, and MLP-T(k=i) are available.

3. Batch Inspection Plans - This software is stored under the filename of BSYSTEM on the system floppy disc. Sampling plans 100%, CSP-F, and WSP-1 are available.

The three system programs have the same basic layout with the only difference being in the Sampling Process. A program listing of the entire Dodge continuous inspection plan software is given in Appendix G1. The program listings of the Sampling Process of the Multi-level continuous inspection plan software and the Batch inspection plan software can be found in Appendix G2 and Appendix G3 respectively.

8.0 RESULTS AND DISCUSSION

8.1 APPROACH ADOPTED WHEN EVALUATING THE SAMPLING PLANS

The evaluation procedure is based on the use of simulated production processes (or process curves). Before any decision can be taken involving the choice of a sampling plan, the accuracy with which the simulated process curves conform to the desired ones, must be determined.

The two results used when comparing the sampling plans are; the average outgoing quality (AOQ, which must remain below the limiting AOQL value) and the time taken (TOTALTIME) to inspect a run of produced products. The sampling plan which ensures the lowest AOQ for the least time spent on inspection, is assumed to be the best plan for that particular trial.

The AOQ and TOTALTIME values for each sampling plan, (for each trial) are multiplied together to produce a resultant. The resultant is a number which is used to measure the relative efficiency of the sampling plans. The sampling plan with the lowest resultant is assumed to be the most efficient plan for that particular trial. The sampling plan that performs the best over a number of trials, is selected for implementation.

8.2 PROCESS CURVE SIMULATION

8.2.1 GENERAL APPROACH

Two types of process curves can be generated using the system software; a binomial distribution (or TYPE-A) process curve and a Horsnell dis-

tribution (or TYPE-B) process curve. The TYPE-B process curve has a binomial distribution and an additional number of defective products following closely in succession (spotty quality). The accuracy of the simulation of TYPE-A process curves can therefore be determined by studying the binomial distribution portion of the TYPE-B process curves.

8.2.2 PRESENTING THE RESULTS

Before a TYPE-A process curve can be simulated, the following information is required :

1. The number of products in the production run (SSIZE)

2. The proportion of the products that must fail (PROB)

This information is used to generate a TYPE-A process curve using the procedure given in Appendix B1. When a TYPE-B process curve must be simulated, the binomial distribution portion is generated using the TYPE-A process curve procedure. The following additional information is required to generate the spotty quality product distribution.

1. The proportion of the products produced that must be defective (PROP)

2. The distribution of these defective products (DIST)

The procedure for generating the TYPE-B process curve is given in Appendix B2.

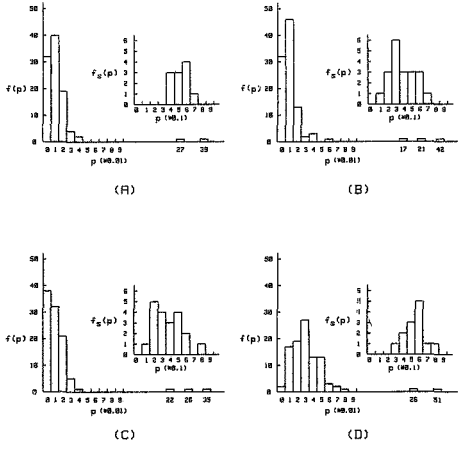
Four TYPE-B process curves are studied. The information used to simulate these process curves is given in the following table.

Table 1	Binomial		Spotty		Trial
Graph	SSIZE	PROB	PROP	DIST	No
A	10000	0.01	0.01	0.85	6
B	10000	0.01	0.01	0.50	7
C	10000	0.01	0.01	0.50	8
D	10000	0.01	0.01	0.70	9

The results of the four simulated TYPE-B process curves are given in figure 36. The four graphs of figure 36 were obtained using the following approach:

An array, the Random Number Resource, contains the product numbers and their corresponding test numbers, of each defective product. The information in this array is used to generate the graphs. Each production run has 10000 products. The production run is divided into one hundred segments of one hundred products each. The number of defective products in each segment is recorded. The results obtained are used to draw the graphs in figure 36. This procedure does not ensure an accurate representation of the spotty quality products. The distribution of these products can be determined by dividing the run of spotty quality products into segments, with ten products in each segment . The number of defective products in each segment is recorded. This information is used to generate the graph representing the distribution of spotty quality products in each case.

The horizontal axis gives the number of occurrences, p. (eg. 3 defective products in a segment of one hundred products would give p = 3/100 = 0.03). The vertical axis gives the probability distribution, f(p), or the number of segments that have n defective products (n = 1, 2, 3, ... , 100).



p = number of defective products in a sample
f(p) = process curve = distribution density of p for the production run
$f_s(p)$ = distribution density of p for the run of spotty quality products

Figure 36. Simulated Results of TYPE-B Process Curves.

8.2.3 DISCUSSION

Although the results are not entirely accurate, the general form of the Horsnell distribution (TYPE-B) process curve is evident. With the exception of the process curve in figure 36 C, the binomial distribution portion of the TYPE-B process curves, is accurate. TYPE-A process curves can therefore be used with confidence. Although the distribution of the spotty quality products is not correct (compared to the desired process curves), the existing distribution is quite sufficient to enable the evaluation of sampling plans using the simulated results for the TYPE-B process curves.

8.3 CONTINUOUS INSPECTION PLANS

8.3.1 GENERAL APPROACH

The results produced by each sampling plan have been recorded for a number of trials. The information required by the process curve simulation procedure was chosen in such a way so as to produce the widest range of process curves possible. The values used were chosen at random.

The production run size and the $AOQL_p$ (see chapter 4) values are kept constant throughout the evaluation procedure. This is done to reduce the number of variables used by the evaluation system. The curves in figure 5 and figure 10 (see chapter 3) are used to select the f and i values for a predetermined $AOQL_t$ value, for each sampling plan. The values of f and i are varied (still ensuring a constant $AOQL_t$) to determine their optimum setting. The same f value is used for all sampling plans, during each trial.

Note: The value of f required by the evaluation system (F) is the reciprocal of the value of f read off the AOQL curves (f) ie. F = 1/f.

8.3.2 SELECTING A SAMPLING PLAN

A table summarising the results obtained by each sampling plan, during the ten trials carried out, is given in Appendix F. A table containing the resultant (AOQ * TOTALTIME) for each sampling plan in each trial is given below.

Table 2	Resultant = AOQ * TOTALTIME					
Trail	MLP	MLP-T	CSP-1	CSP-2	CSP-3	Best
1	180	180	352	281	266	BOTH
2	271	271	422	392	422	BOTH
3	53	53	262	118	130	BOTH
4	160	160	296	342	296	BOTH
5	289	282	405	360	355	MLP-T
6	185	197	200	231	245	MLP
7	234	243	230	196	228	CSP-2
8	172	181	205	184	153	CSP-3
9	242	242	273	299	308	BOTH
10	96	100	95	117	100	CSP-1

BOTH = MLP and MLP-T

The last column of the table contains the sampling plan designation that produced the best result for each trial (ie. the sampling plan with the lowest resultant).

The Multi-Level sampling plans, designated MLP and MLP-T perform the best over ten trials. Three criteria can now be adopted to select the best of these two sampling plans.

1. The multi-level sampling plan that ensures the lowest AOQ value, during each trial, can be chosen (AOQ column in table 3 below).

2. The multi-level sampling plan that ensures the least time spent on the inspection of the production run in each case, can be chosen. (TIME column in table 3 below).

3. The multi-level sampling plan that achieves the lowest overall resultant for each trial, can be chosen (RESULTANT an in table 3 below).

A table summarising the outcome of the three above mentioned criteria for selecting the best multi-level plan, is given in the table below.

Table 3	Selection Criteria		
Trial	AOQ	TIME	RESULTANT
1	BOTH	BOTH	BOTH
2	BOTH	BOTH	BOTH
3	BOTH	BOTH	BOTH
4	BOTH	BOTH	BOTH
5	MLP-T	MLP	MLP-T
6	MLP	MLP	MLP
7	BOTH	MLP	MLP
8	BOTH	MLP	MLP
9	BOTH	BOTH	BOTH
10	MLP-T	MLP	MLP

BOTH = MLP and MLP-T

Sampling plan MLP-T performs the best when considering the AOQ criterion, while the MLP sampling plan produces the best results when considering the other two selection criteria.

Sampling Plan Selection

The MLP multi-level sampling plan is chosen because it produces the best overall results, when inspecting the simulated production process. (ie. it performs the best at two of the three criteria adopted for comparing the performances of the two multi-level sampling plans.)

DUCTION IN INSPECTION TIME

The time taken to successfully inspect a production run using 100% inspection and the MLP sampling plan is known. The performance of the MLP sampling plan relative to that of the 100% inspection can therefore be determined.

The reduction in inspection time, when using the MLP sampling plan, can be calculated by comparing the time taken to inspect a production run using the MLP sampling plan and 100% inspection.

A table (table 4) of the reduction in time, obtained using the MLP sampling plan over the 100% inspection, is given on the following page. The reduction in time is represented as a percentage of time spent on 100% inspection.

The average time reduction over ten trials is calculated as 30.4%. It is therefore possible to achieve an average reduction of approximately 30% in the time spent on inspection using the multi-level inspection plan (MLP) instead of employing 100% inspection.

Table 4	Time Reduction		F
Trial	Units	%	(1/f)
1	10876	13.6	5
2	15658	19.5	5
3	3928	4.9	10
4	10691	13.3	10
5	23502	29.3	3
6	37224	46.5	3
7	43549	54.3	3
8	41117	51.3	3
9	7145	8.9	5
10	49932	62.4	3

8.3.4 DISCUSSION

Certain considerations enter the most advantageous choice of f and i (f is the value represented on the curves in figure 5 and figure 10 of chapter 3). For example, it will be apparent that the use of too small a value of f (or too large a value of F, where F = 1/f) will increase the chances of passing unnoticed a substantial run of defective products. It is apparent, from the results represented in Appendix F1, that a value of f = 1/3 (or F = 3) produces the best overall results when inspecting the simulated production process using the MLP multi-level sampling plan. The time spent on inspecting a production run when using an MLP sampling plan with f = 1/3, is the lowest. The average reduction in inspection time over that achieved by 100% inspection can be calculated using information in table 4. There are 5 trials that used sampling plans of f = 1/3 (or

F ≈ 3). The average reduction in inspection time using results of these five trials is 48.8% over the time spent to totally inspect every product in the production run.

8.4 BATCH INSPECTION PLANS

8.4.1 GENERAL APPROACH

It is not possible to compare and select a batch inspection plan using the approach adopted for the selection of the continuous selection plan. This is due to the lack of information needed by sampling plan CSP-F. The two plans are assessed individually. A choice between the two plans may be possible depending on the outcome of the individual assessments.

8.4.2 ASSESSMENT OF PLAN CSP-F

The choice of f, i, and N to give a required value of AOQL is given in QSTAG 340 (1974). This reference was not obtainable. Two sets of values, for production runs of 500 - 1000 products, were given by Blackwell (1977).

For an $AOQL_t$ value of 1.9%; i = 36 and f =1/5 (F = 5).For an $AOQL_t$ value of 0.143%; i = 302 and f = 1/10 (F = 10).

The $AOQL_p$ value is 15.2% for the $AOQL_t$ of 1.9% and 1.144% for the $AOQL_t$ of 0.143% (See Chapter 4). It is highly unlikely that a customer would accept a batch of products that was inspected using an $AOQL_p$ of 15.2%. The only $AOQL_p$ value that can be used is therefore the 1.144% one.

A number of trials were carried out using an $AOQL_p$ of 1.144% ($AOQL_t$ = 0.143% ; i = 302, f = 1/10 (F = 10)). A table of the results obtained for the various trials, using a different production process for each trial, is given in Appendix F2.1.

When studying the results, it will be noticed that the $AOQL_p$ is exceeded in four of the twelve trials. Unless more promising results are obtained, using other values of f and i, this plan cannot be accepted.

8.4.3 ASSESSMENT OF PLAN WSP-1

The plan starts with partial inspection. If a total of M^* defects are found, where $M^* = (AOQL_t * SSIZE)/(F-1)$ and F = 1/f, the plan reverts to 100% inspection. When M^* is zero, the plan applies 100% inspection to the entire batch. The values of F and $AOQL_t$ (N is fixed) must therefore be chosen so as to ensure that M^* is one or larger. The graph in figure 37 is used when selecting values of $AOQL_p$ and F for each production run. Two curves are given on the graph; one for a batch of 500 products, the other for a batch of 700 products.

Two values of $AOQL_p$ are used during the assessment of this sampling plan, they are, 1.6% and 3.2%. The values of F for these values of $AOQL_p$ were chosen using the graph in figure 37. The values of F are 2 and 3 respectively. These values can be used for batches of 500, and of 700 products.

The tables summarising the results obtained using these values are available in Appendix F 2.2. These results are given in two tables, one for batches of 500 products and the other for batches of 700 products. The $AOQL_p$ was only exceeded in one of the twenty trials.

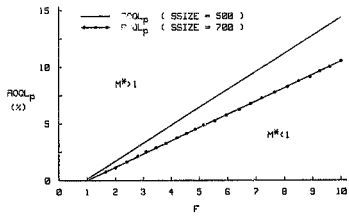


Figure 37. The Limitation on AOQL and F for Plan WSP-1.

Reduction in Inspection Time

The time spent on inspecting a batch with an $AOQL_p$ of 3.2% is lower than that of a batch using an $AOQL_p$ of 1.6%. The reduction in inspection time when using the WSP-1 sampling plan with an $AOQL_p$ of 1.6% is therefore considered. (this plan has the tightest specification). Two tables, one for a batch of 500 products and one for a batch of 700 products, is given on the following page (see table 5a and table 5b, respectively). These tables list the reduction in time spent on inspection using the sampling plan compared to the time spent when 100% inspection is applied.

The average time reduction, over five trials, for batches of 500 products is 33.2% and the average time reduction for batches of 700 products is 31.5%. It is therefore possible to achieve an average reduction in inspection time of approximately 30% when using the WSP-1 sampling plan, over the time taken to apply 100% inspection to the batch.

Table 5a	Reduction (500)	
Trial	Units	%
1	1410	35.4
2	1286	32.1
3	1633	40.7
4	978	24.3
5	1356	33.7
Average = 33.2%		

Table 5b	Reduction (700)	
Trial	Units	%
1	2234	39.8
2	1628	29.0
3	2475	44.1
4	1558	27.7
5	948	16.8
Average = 31.5%		

8.4.4 DISCUSSION

Plan CSP-F

Before Plan CSP-F is dismissed completely, an attempt can be made to obtain QSTAG 340 (1974). This paper contains further information concerning the choice of f and i for a given batch size. This information can be used to carry out a further assessment of the plan. If the CSP-F sampling plan still produces unacceptable results, it must be discarded.

Plan WSP-1

The values of M^* and F are integer values. The formula $M^* = (AOQL_t * SSIZE)/(F-1)$ is used to calculate the range of $AOQL_t$ for $M^* = 1$, F = 2 and a batch of 500 products (SSIZE = 500). This range is present because M^* is rounded down to an integer value when it is calculated (eg. if $M^* = 1.7$ it is rounded down to 1 but the $AOQL_t$ which gives $M^* =$

1.7 is 0.34% while the $AOQL_t$ which gives $M^* = 1$ is 0.2%, but the sampling plan gives the same protection in both cases.). With $M^* = 1$, F = 2 and SSIZE = 500, the WSP-1 sampling plan ensures an $AOQL_t$ in the range $0.2\% < AOQL_t < 0.4\%$ leading to the $AOQL_p$ range $1.6 < AOQL_p < 3.2\%$.

In the outcome of one trial the AOQ exceeded the desired $AOQL_p$ value. This trial was for a batch of 500 products, using an $AOQL_p$ of 1.6% and F = 2 (ie.$M^* = 1$). The AOQ in this case was 1.8%. If the effect of the rounding operation on M^* is taken into account, it can be seen that this AOQ lies within the $AOQL_p$ range given above. (ie. $1.6\% < AOQL_p < 3.2\%$). Although the $AOQL_p$ limit was exceeded in this case the WSP-1 sampling plan operates correctly.

Choice of the Batch Inspection Plan

If no further assessments of the plans were carried out, Plan WSP-1 would be chosen for implementation. This plan produces the best overall results when inspecting the simulated production process.

9.0 CONCLUSION

There is no available information concerning the ingoing quality and the distribution of defective products for the production lines of Telephone Manufacturers (TM). The production process data (process curves) used in the evaluation of the sampling plans was therefore simulated.

Two inspection situations were found to exits at TM, one for the inspection of long runs of continuously produced products (continuous production) and the other, for the inspection of short runs of continuously produced products (batch production). Two sampling plans were selected for implementation at TM, one for each of the inspection situations. Both the sampling plans ensure that the average outgoing quality (AOQ) of the outgoing products (ie. the average number of defective products not inspected) always remains below a prescribed limit. the average outgoing quality limit (AOQL). The parameters used by the sampling plans are selected so as to ensure the AOQL.

A reduction of 48.8% in the inspection time, over that of 100% inspection, is achieved using the multi-level continuous inspection plan, designated MLP (with $AOQL_p$ = 1%, F = 3 and i = 740), when inspecting the simulated continuous production data. A reduction of 31% in inspection time is achieved using plan WSP-1 (with $AOQL_p$ = 1.6% and F = 2) when inspecting the simulated batch production data. The reduction in inspection time depends on the choice of the AOQL value. The smaller the desired AOQL, the more inspection is required to ensure that the AOQ remains below this value. This leads to an increase in inspection time , and vice versa.

Before the sampling plans can be implemented at TM, the accuracy of the simulated process curve data (on which the sampling plan assessment was based) must be considered. If the simulated data compares favourable with the actual production process (process curve) data, the two sampling plans can be implemented with confidence. The entire evaluation procedure

must be repeated, using the collected production process data, if the simulated data is found to be inaccurate.

The efficiency of the test equipment will be increased with the introduction of sampling inspection procedures to inspect the products produced on the TM production lines. The existing test equipment are microprocessor based machines that need test programs to control their operation. The test programs contain the instructions required to inspect the various tests on the product. The test program can be modified to incorporate the use of the sampling plans during inspection.

The research reported in the project report has indicated a number of associated areas where further work is needed:

- Each product has a number of tests that must be inspected. The possibility of using a different sampling plan for each test can be investigated (ie. use 100% inspection on a test that is critical to the operation of the product being inspected, while sampling inspection is employed for the other tests).

- There is a need to convince people, who are dubious of the effectiveness of sampling inspection, of its advantages. A procedure that could be used to set these persons minds at rest, is given below.

  Use sampling inspection plans at all intermediate test stations on the production line, and 100% inspection at the final test station. The inspection time would be reduced and, if the inspection at the final test station is perfect, all defective products will be located before they leave the factory. To convince the dubious people of the effectiveness of sampling inspection, a sampling inspection plan could run parallel to the 100% inspection plan, and a comparison of the results made.

- A broader literature survey can be carried out in an attempt to locate alternative sampling plans that can be used in the inspection situations at TM.

APPENDIX A. CALCULATE AOQL FOR EACH TEST ON THE PRODUCT

Given

- Let $AOQL_t(I)$ (where I=1,...,N) be the AOQL value for test I.
- Let $AOQ_t(I)$ be the average outgoing quality (AOQ) for test I.
- Let $AOQL_p$ be the process AOQL.
- Let AOQ_p be the process AOQ.
- Let N be the total number of tests.

Assumptions

1. There are N separate tests per product.
2. The AOQL values for each test are equal.
3. $AOQ_t = AOQL_t$.

The total AOQ for the process is given as

$$AOQ_p = AOQ_t(1) + AOQ_t(2) + \ldots + AOQ_t(N)$$

by assumption (3), we have

$$AOQ_p = AOQL_t(1) + AOQL_t(2) + \ldots + AOQL_t(N)$$

Assumption (2) gives

$$AOQL_t = AOQL_t(1) = \ldots = AOQL_t(N)$$

leading to

$$AOQ_p = N * AOQL_t$$

Because the AOQ_t values for each test are assumed to be the limiting values ($AOQL_t$) we have that AOQ_p will also be the limiting value ($AOQL_p$) for the process

This gives

$$AOQL_p = N * AOQL_t$$

or

$$AOQL_t = AOQL_p / N$$

APPENDIX B. SIMULATING THE PROCESS CURVE

Definitions

Let

- SSIZE be the number of products in the production run
- PROB be the fraction of defective products
- NODEF be the number of defective products in the production run
- TESTNO be the number of the test that must fail on the product
- RND be the result of a random generator giving a number between 0 and 1
- PRODUCTNO be the number of the product that is defective
- RNRESOURCE be the array containing NODEF PRODUCTNO's with corresponding TESTNO's
- MAXTEST be the total number of tests on the product
- SSPOT be the fraction of spotty quality products
- NSPOT be the number of spotty quality products
- RSPOT be the range of product numbers within which the NSPOT defective products will fall
- FSPOT be the product number at which the RSPOT starts
- PSPOT be the probability of finding a defective product in the RSPOT of spotty quality products

B.1 TYPE-A PROCESS CURVE

Procedure : TYPE-A Process Curve (Binomial Distribution)

Generate an array of NODEF PRODUCTNO's and corresponding TESTNO's. This array represents the defective products in the production run and also contains the test, on the product, that is defective. Each product has the *probability PROB of being defective and each test on a* defective product has the probability of (1/MAXTEST) of being defective.

Begin :

* Calculate the number of defective products *
NODEF = INT(SSIZE * PROB)

* Generate NODEF PRODUCTNO's and corresponding TESTNO's. Each PRODUCTNO is chosen at random from the SSIZE products and the TESTNO is chosen at random from the MAXTEST tests on the product *

* Check for repeated PRODUCTNO's and generate new ones to replace one of the repeated numbers *

* Sort the array according to the PRODUCTNO's ; from smallest to largest. When a PRODUCTNO is swapped the corresponding TESTNO must also be swapped *

End :

B.2 TYPE-B PROCESS CURVE

Procedure : TYPE-B Process Curve (Horsnell Distribution)

First generate a binomial distribution as in TYPE-A process curve but do not sort the array. Then add the spotty quality products.

Begin :

```
* Calculate NODEF *
    NODEF = INT( SSIZE * PROB )

* Generate NODEF PRODUCTNO's and corresponding TESTNO's. Each
PRODUCTNO is chosen at random from the SSIZE products and the TESTNO
is chosen at random from the MAXTEST tests on the product *

* Check for repeated PRODUCTNO's and generate one new one to replace
one repeated number *

* Select a random TESTNO *
    TESTNO = INT( RND*MAXTEST )

* Calculate the constants required when generating the spotty quality
product numbers *
    NSPOT = INT( SSIZE * SSPOT )
    RSPOT = NSPOT / PSPOT
    Repeat:
        FSPOT = INT( SSIZE * RSPOT )
    until ( FSPOT < (SSIZE - RSPOT) )

* Generate NSPOT PRODUCTNO's. Each PRODUCTNO is chosen at random from
the range in which the spotty quality products must appear. The
PRODUCTNO and the constant TESTNO are stored in the array. NODEF is
incremented each time a spotty quality product is chosen *
```

```
* Sort the array according to PRODUCTNO's ; from smallest to largest.
When  a  PRODUCTNO  is  swapped the corresponding TESTNO must also be
swapped *

End :
```

APPENDIX C. SYSTEM HARDWARE DEVELOPMENT

C.1 - Circuit Diagrams

C.2 - Component List

C.3 - Decoder Logic Design

C.4 - Interface Unit Software

C.1 CIRCUIT DIAGRAMS

C.1.1 - Prototype Card

C.1.2 - Interface Unit

C.1.3 - External Circuitry to the D.P.S. Module

C.1.4 - Decoder

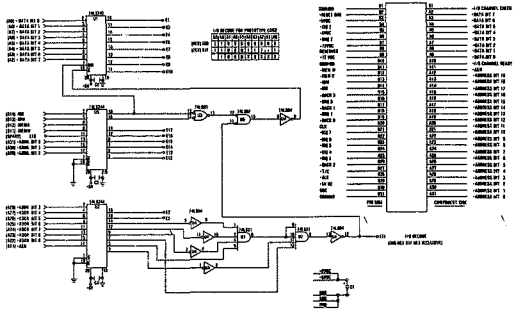
C.1.5 - Automatic KEY Operation

C.1.6 - Component Layout - Prototype Card

C.1.7 - Component Layout - VERO 1 and VERO 2

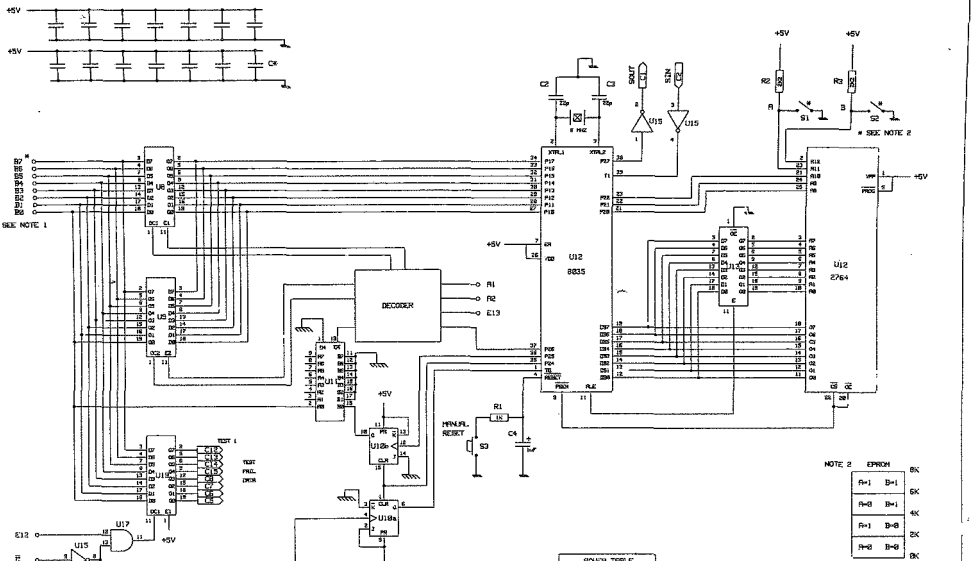
C.1.8 - Component Layout - VERO 3 and PCB 1

C.1.1 PROTOTYPE CARD



SHEET 1 OF 8

C.1.2 INTERFACE UNIT



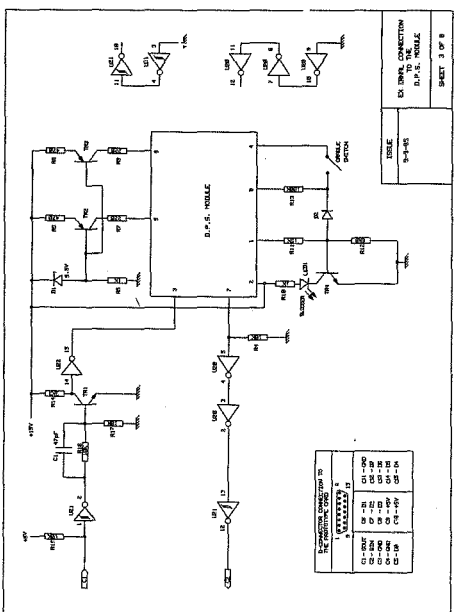
POWER TABLE		
+5V	IC	GND
20	U5	10
20	U6	10
16	U10	8
20	U11	10
28	U13	14
20	U13	10
40	U14	20
20	U20	10

*NOTE 1: U1 IS NOT USED

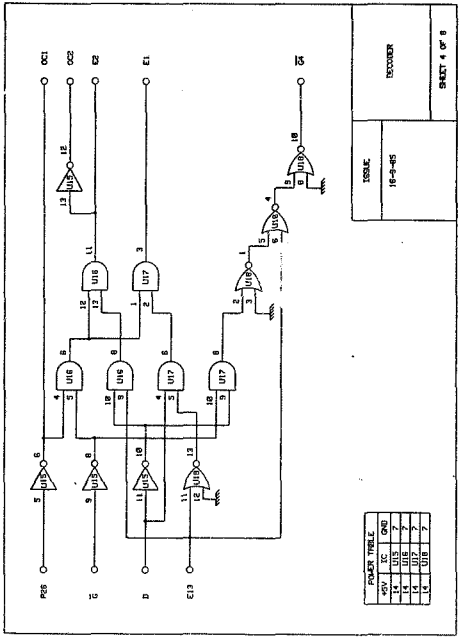
U1	TO	U1	TO
2	B0	7	B5
3	B1	8	B6
4	B2	9	B7
5	B3	1	A1
6	B4	19	A2

ISSUE	INTERFACE UNIT
12-9-85	
	SHEET 2 OF 9

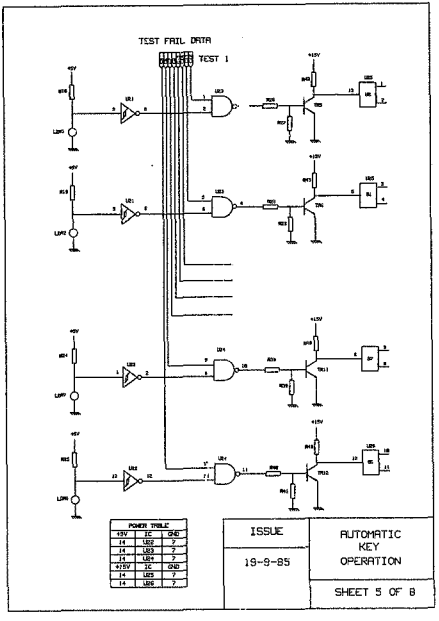
C.1.3 EXTERNAL CONNECTION TO THE D.P.S. MODULE



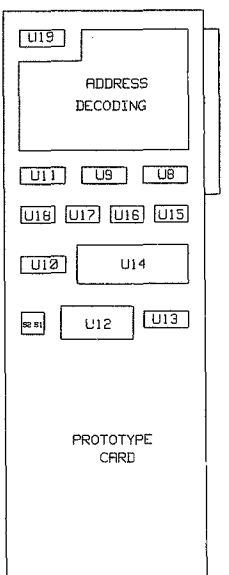
C.1.4 DECODER


DECODER
SHEET 4 OF 8
ISSUE
16-9-85
OC1
OC2
E2
E1
S2
POS
C
D
E13
POWER TABLE
+5V IC GND
14 U15 7
14 U16 7
14 U17 7
14 U18 7

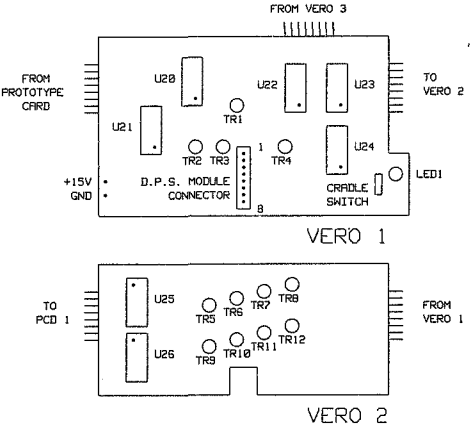
C.1.5 AUTOMATIC KEY OPERATION



C.1.6 COMPONENT LAYOUT - PROTOTYPE CARD

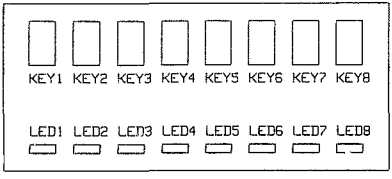


C.1.7 COMPONENT LAYOUT - VERO 1 AND VERO 2



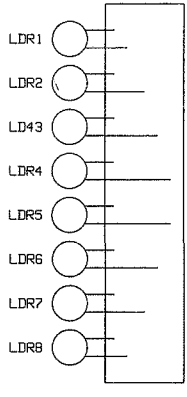
SHEET 7 OF 8

C.1.8 COMPONENT LAYOUT - VERO 3 AND PCB 1



PCB 1

FUNCT	KEY NO	SW NO
LINE 1	KEY1	S0
LINE 2	KEY2	S1
STATION1	KEY3	S2
STATION2	KEY4	S3
STATION3	KEY5	S4
DIVERT	KEY6	S5
HOLD	KEY7	S7
RECALL	KEY8	S6



SHEET 8 OF 8

VERO 3

C.2 COMPONENT LIST

Resistors (1/4 W unless specifically stated)

R1	- 1K	R11	- 15K
R2,R3	- 2K2	R12	- 6K8
R4	- 10K	R13	- 100K
R5	- 1K	R14	- 3K3
R6	- 470	R15,R16,R17	- 10K
R7	- 220	R18,...,R25	- 3K3
R8	- 470	R26,...,R41	- 10K
R9	- 220	R42,...,R49	- 2K2
R10	- 1K		

Capacitors

C1	- 47 pf (ceramic)
C2,C3	- 22 pf (ceramic)
C4	- 1 mfd (electrolytic)
C*	- 0.1 mfd (decoupling capacitors, ceramic)

Integrated Circuits

U1	- not used	U14	- 8035
U2	- 74LS244	U15	- 74LS04
U3	- 74LS08	U16	- 74LS08
U4	- 74LS04	U17	- 74LS08
U5	- 74LS244	U18	- 74LS02
U6	- 74LS02	U19	- 74LS373
U7	- 74LS21	U20	- 4049
U8	- 74LS373	U21	- 40106
U9	- 74LS373	U22	- 40106
U10	- 74LS109	U23	- 4011
U11	- 74LS245	U24	- 4011
U12	- 2764	U25	- 4016
U13	- 74LS373	U26	- 4016

Semi-conductors

TR1 - BC107b
TR2,TR3 - BCY71
TR4 - BC107b

D1 - 5.5v Zener diode
D2 - IN4148

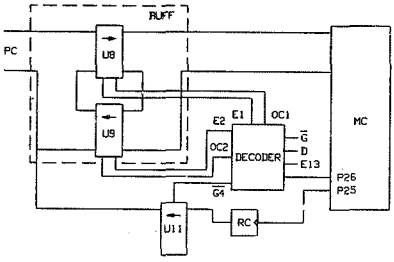
LED1 - 5 mm green LED

Miscellaneous

XTAL - 6 Mhz crystal
S1,S2 - 4 way DIL
C - 15 pin D-connector
LDR1,...,LDR8 - Light Dependant Resistors

C.3 DECODER LOGIC DESIGN

A block diagram of the system is given below



PC - Personal Computer
MC - Microprocessor
BUFF - Bye-directional data buffer and latch
E13 - Address bit 1 from the PC
P26 - Bit 6 of port 2 of the MC
D - DIR (U1, prototype card)
G - G (U1, prototype card)
E1 - Enable (U8, active high)
OC1 - Output Control (U8, active low)
E2 - Enable (U9, active high)
OC2 - Output Control (U9, active low)
G4 - Enable (U11, active low)

The decoder controls the operating sequence of the various chips. The inputs of the decoder are controlled by the PC (D, G, E13) and the microprocessor (MC) (P26). The following operations must be controlled:

1. MC read data from the BUFF.

2. PC write data to BUFF.

3. PC read data from MC.

4. PC read RC control line.

The outputs of the decoder are controlled according to the information at the decoder input. A truth table of the inputs and the corresponding outputs of the decoder is given below

Operation	Inputs				Outputs				
	D	$\overline{G}$	E13	P26	E1	E2	OC1	OC2	$\overline{G4}$
PC --> BUFF	1	0	0	0	1	0	1	1	1
BUFF --> MC	X	X	X	1	0	0	0	1	X
RC --> PC	0	0	0	X	0	0	X	1	0
MC --> PC	0	0	1	0	0	1	1	0	1
BUFF --> MC & RC --> PC	0	0	0	1	0	0	0	1	0
All others					0	0	1	1	0

A full truth table is given below

Inputs				Outputs				
D	$\overrightarrow{G}$	E13	P26	E1	E2	OC1	OC2	$\overrightarrow{G4}$
0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	1	0
0	0	1	0	0	1	1	0	1
0	0	1	1	0	0	0	1	X
0	1	0	0	0	0	1	1	1
0	1	0	1	0	0	0	1	X
0	1	1	0	0	0	1	1	1
0	1	1	1	0	0	0	1	X
1	0	0	0	1	0	1	1	1
1	0	0	1	0	0	0	1	X
1	0	1	0	0	0	1	1	1
1	0	1	1	0	0	0	1	X
1	1	0	0	0	0	1	1	1
1	1	0	1	0	0	0	1	X
1	1	1	0	0	0	1	1	1
1	1	1	1	0	0	0	1	X

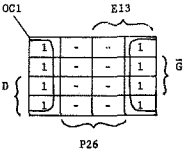
The expressions for the five outputs, in terms of the inputs, are minimised and are given as;

1. $E1 = \overrightarrow{D} \cdot \overrightarrow{\overrightarrow{G}} \cdot \overrightarrow{P26} \cdot \overrightarrow{E13}$

2. $E2 = D \cdot \overrightarrow{\overrightarrow{G}} \cdot \overrightarrow{P26} \cdot E13$

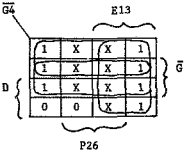
3. $OC2 = \overrightarrow{E2}$

4. OC1 is minimised using karnaugh maps.



$OC1 = \overline{P26}$

5. G4 is minimised using a karnaugh map.



$\overline{G4} = \overline{D} + \overline{G} + E13$ or $\overline{D \cdot G} + E13$

The decoder has been implemented using 2-Input NOR gates, 2-Input AND gates and Inverters. The circuit diagram of the decoder is given in Appendix C1 (Sheet 4).

C.4 INTERFACE UNIT SOFTWARE

C.4.1 A PDL DESCRIPTION

```
PROCEDURE: INTER2
BEGIN:
  * reset RE and RC *
  REPEAT
    WHILE ( RE = low )
      * wait *
    END WHILE:
    * read BUFF and store the data in register 0 ( R0 ) *
    * set RC = 1 *
    * start TIMER for 1 second *
    FLAG1 = FALSE
    REPEAT
      * send serial data *
        CALL OUTP27
      * receive serial answer *
        CALL SCANT1
      IF ( the answer is correct )
        THEN
          FLAG1 = TRUE
          R3 = FFH
        ELSE
          IF ( TIMER = 1 sec )
            THEN
              FLAG1 = TRUE
              R3 = 00H
          ENDIF:
      ENDIF:
    UNTIL ( FLAG1 = TRUE )
    * make data available to the PC *
```

```
      ( place result of R3 on port 1 )
    * switch LED off *
    * set RC = low *
    WHILE ( RE = high )
      * wait *
    END WHILE:
  UNTIL ( forever )
END:
```

PROCEDURE: OUTP27

This procedure takes the value of R0 and transmits to the D.P.S. module in the required serial format.

PROCEDURE: SCANT1

This procedure scans input T1 and receives the answer from the D.P.S. module. It converts the serial incoming information to an 8 bit word.

C.4.2 SOFTWARE LISTING

```
LOCATION OBJECT CODE LINE    SOURCE LINE
                       1  "8048"
                       2  ;#############################################################
                       3  ;THIS PROGRAM READS DATA FROM THE PC AND SENDS IT TO THE BOARD
                       4  ;UNDER TEST IN THE CORRECT FORMAT. THE ANSWER RECEIVED FROM THE
                       5  ;BOARD IS CHECKED. IF THE CORRECT ANSWER IS NOT RECEIVED IN 10
                       6  ;SECONDS , AN ERROR IS ASSUMED AND A ZERO IS SENT TO THE PC.
                       7  ;IF THE ANSWER IS CORRECT A ONE IS SENT TO THE PC.
                       8  ;
                       9
                                              ORG     00H
0000 C5                10 INIT:               SEL     RB0
0001 2300              11                     MOV     A,#00H          ;CLEAR RE AND RC
0003 3A                12                     OUTL    P2,A
0004 00                13                     NOP
0005 00                14                     NOP
0006 2310              15                     MOV     A,#10H
0008 3A                16                     OUTL    P2,A
0009 3609              17 MAINPR:             JT0     MAIN1           ;WHILE RE=LOW WAIT
000B 8409              18                     JMP     MAINPR
000D 2350              19 MAIN1:              MOV     A,#50H          ;READ BUFF
000F 3A                20                     OUTL    P2,A
0010 09                21                     IN      A,P1
0011 A8                22                     MOV     R0,A            ;STORE DATA IN R0
0012 2330              23 MAIN2:              MOV     A,#30H          ;SET RC=1
0014 3A                24                     OUTL    P2,A
0015 00                25                     NOP
0016 00                26                     NOP
0017 2310              27                     MOV     A,#10H
0019 3A                28                     OUTL    P2,A
001A BD0F              29 MAIN3:              MOV     R5,#0AFH        ;COUNTER 1 SECS
001C 1454              30 M32:                CALL    OUTP27          ;SEND SERIAL DATA
001E 148F              31                     CALL    SCAN11          ;RECEIVE ANSWER
0020 F9                32 MDECODER:           MOV     A,R1            ;DECODER
0021 D6                33                     XRL     A,R0
0022 C633              34                     JZ      K35
0024 2302              35                     MOV     A,#02
0026 97                36 M33:                DEC     A
0027 C62B              37                     JZ      M34
0029 0426              38                     JMP     M33
002B EB1C              39 M34:                DJNZ    R5,M32
002D B868              40                     MOV     R3,#68H         ;"NO" ANSWER
002F 0430              41                     JMP     MAIN4
0031 B8FF              42 M35:                MOV     R3,#0FFH        ;"YES" ANSWER
0033 23FF              43                     MOV     A,#0FFH         ;START DELAY
0035 97                44 M36:                DEC     A
0036 C636              45                     JZ      MAIN4
0038 0435              46                     JMP     M36
003A FB                47 MAIN4:              MOV     A,R3            ;MAKE DATA AVAIL
003B 39                48                     OUTL    P1,A
003C B860              49 MAIN5:              MOV     R0,#60H         ;SWITCH LED OFF
003E 1454              50 M51:                CALL    OUTP27
0040 148F              51                     CALL    SCAN11
0042 F9                52                     MOV     A,R1
0043 D6                53                     XRL     A,R0
0044 C648              54                     JZ      MAIN6
0046 043E              55                     JMP     M51
0048 2334              56 MAIN6:              MOV     A,#34H          ;SET RC=LOW
004A 3A                57                     OUTL    P2,A
```

```
LOCATION OBJECT CODE LINE   SOURCE LINE

  004B 00          58                 NOP
  004C 00          59                 NOP
  004D 2318        60                 MOV             A,#18H
  004F 3A          61                 OUTL            P2,A
  0050 2687        62 MAIN7:          INT0            MAINPR          ;WHILE RE=HIGH
  0052 8450        63                 JMP             MAIN7           ;WAIT
                   64 ;*************************************************************
                   65 ;OUTPUT THE DATA IN THE CORRECT SERIAL FORMAT
                   66 ;
  0054 BC09        67 OUTP27:         MOV             R4,#09          ;INITIALISE
  0056 2390        68                 MOV             A,#90H          ;P27=1
  0058 3A          69                 OUTL            P2,A
  0059 2325        70 OUT1:           MOV             A,#37           ;184 CYCLE DELAY
  005B 07          71 O1:             DEC             A
  005C C660        72                 JZ              OUT1E
  005E 0459        73                 JMP             O1
  0060 00          74 OUT1E:          NOP
  0061 00          75                 NOP
  0062 00          76                 NOP
  0063 2318        77                 MOV             A,#18H
  0065 3A          78                 OUTL            P2,A            ;P27=0
  0066 F8          79                 MOV             A,R0
  0067 A9          80                 MOV             R3,A
  0068 00          81                 NOP
  0069 00          82                 NOP
  006A 00          83                 NOP
  006B 00          84                 NOP
  006C 2309        85 OUT2:           MOV             A,#09H
  006E 07          86 O2:             DEC             A
  006F C673        87                 JZ              O2END
  0071 046E        88                 JMP             O2
  0073 97          89 O2END:          CLR             C
  0074 00          90                 NOP
  0075 00          91                 NOP
  0076 00          92                 NOP
  0077 FB          93                 MOV             A,R3
  0078 F280        94                 JB7             O21
  007A AB          95                 MOV             R3,A            ;P27=0
  007B 2310        96                 MOV             A,#10H
  007D 3A          97                 OUTL            P2,A
  007E 0486        98                 JMP             O22
  0080 AB          99 O21:            MOV             R3,A
  0081 2390       100                 MOV             A,#90H          ;P27=1
  0083 3A         101                 OUTL            P2,A
  0084 00         102                 NOP
  0085 00         103                 NOP
  0086 FB         104 O22:            MOV             A,R3
  0087 F7         105                 RLC             A
  0088 AB         106                 MOV             R3,A
  0089 E66C       107                 DJNZ            R4,OUT2
  008B 2318       108 OUT3:           MOV             A,#10H
  008D 3A         109                 OUTL            P2,A            ;P27=0
  008E 83         110                 RET
                  111 ;*************************************************************
                  112 ;READ SERIAL DATA ON T1
                  113 ;
  008F BC08       114 SCAN1:          MOV             R4,#08          ;INITIALISE
```

```
FILE: INTER2;BPC        HEWLETT-PACKARD: 8048 Assembler                    PAGE 3

LOCATION OBJECT CODE LINE     SOURCE LINE

                    115 SCAN1:      JT1             SCAN2           ;IF T1=1 GOTO SCAN2
0093 9491           116             JMP             SCAN1
0095 2386           117 SCAN2:      MOV             A,#86           ;30 CYCLE DELAY
0097 07             118 SC2:        DEC             A
0098 C69C           119             JZ              SC2END
009A 0497           120             JMP             SC2
009C 56A0           121 SC2END:     JT1             SCAN3           ;RECHECK T1=1
009E 9492           122             JMP             SCAN1
00A0 BB08           123 SCAN3:      MOV             R3,#08H         ;60 CYCLE DELAY
00A2 2301           124             MOV             A,#01H
00A4 07             125 SC3:        DEC             A
00A5 C6A9           126             JZ              SC3END
00A7 04A4           127             JMP             SC3
00A9 00             128 SC3END:     NOP
00AA 00             129             NOP
00AB 238A           130 SCAN4:      MOV             A,#8AH
00AD 07             131 SC4:        DEC             A
00AE C6B2           132             JZ              SC4END
00B0 04AD           133             JMP             SC4
00B2 00             134 SC4END:     NOP
00B3 00             135             NOP
00B4 56B8           136 SCAN5:      JT1             SCAN6           ;IF T1=1 GOTO SCAN6
00B6 FB             137             MOV             A,R3            ;T1=0
00B7 E7             138             RLC             A
00B8 AB             139             MOV             R3,A
00B9 04C0           140             JMP             SCAN7
00BB 1B             141 SCAN6:      INC             R3
00BC FB             142             MOV             A,R3
00BD F7             143             RLC             A
00BE AB             144             MOV             R3,A
00BF 00             145             NOP
00C0 ECAB           146 SCAN7:      DJNZ            R4,SCAN4
00C2 FB             147             MOV             A,R3
00C3 67             148             RRC             A
00C4 A9             149             MOV             R1,A
00C5 23A0           150 SCAN8:      MOV             A,#0A0H         ;DELAY AFTER SERIAL
00C7 07             151 SC8:        DEC             A               ;BYTE IS READ.
00C8 C8CC           152             JZ              SCAN9
00CA 04C7           153             JMP             SC8
00CC 83             154 SCAN9:      RET
                    155             END
                    ##################################################

Errors=  0
```

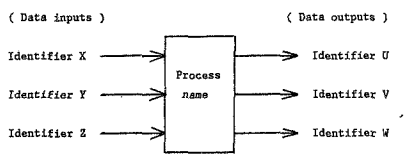
APPENDIX D. DATA FLOW AND TASK MANAGEMENT

An examination of the flow of data through an information processing system is fundamental to the design process. It is of major importance to be able to develop a clear overall view of how data transverses the system. Such an approach will then lead to a partitioning of the system into areas of processing activity, and as to how these activities are to be managed.

It is essential to distinguish graphically between the activities of data management and data transformation. The following conventions are introduced for this purpose

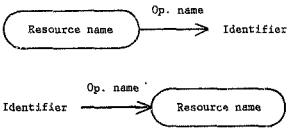
Graphical symbols for data flow diagrams

Symbols are introduced for processes (areas of data management) and for resources (areas of data transformation).



Graphics Symbol for a Process

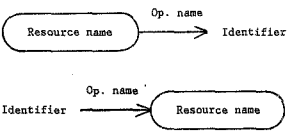
A process may command any number of data inputs and outputs.



Graphic Symbol for a Resource

The operator name is given immediately above the arrows pointing into or out of the resource. Other identifiers which are part of the procedure implementing the operation on the resource (ie. status, resource management information like pointers etc.) are not considered part of the data flow diagram.

A resource comprises an aggregate of structural components including *single data items*, *records* and *arrays* and a collection of operators (or procedures) which have exclusive access to the components of the data *aggregate*. User access to the resource will entirely be managed via the collection of operators. The operators taken together with the data aggregate uniquely characterise the behaviour of the resource.



Graphic Symbol for a Resource

The operator name is given immediately above the arrows pointing into or out of the resource. Other identifiers which are part of the procedure implementing the operation on the resource (ie. status, resource management information like pointers etc.) are not considered part of the data flow diagram.

A resource comprises an aggregate of structural components including single data items, records and arrays and a collection of operators (or procedures) which have exclusive access to the components of the data aggregate. User access to the resource will entirely be managed via the collection of operators. The operators taken together with the data aggregate uniquely characterise the behaviour of the resource.

APPENDIX E. PDL DESCRIPTION OF THE SYSTEM SOFTWARE

E.1 STATISTICAL TEST PROCESS

```
PROCEDURE: Statistical Test Process

INPUT : Keyboard, Interface Unit
OUTPUT: Display, Printer, Interface Unit

BEGIN:
  OPEN : Process Curve Resource and Data Input Resource
  IST1 = FALSE
  REPEAT
    PUT:
      DISPLAY: * clear screen *
      DISPLAY: STATISTICAL TEST PROCESS
      DISPLAY: 1 : Data Input Process
      DISPLAY: 2 : Sampling Process
      DISPLAY: 3 : Data Output Process
      DISPLAY: 4 : Quit
      Select the required option and press <enter>
    ENDPUT:
    GET:
      KEYBOARD: FUNCT
    ENDGET:
    * Check range of FUNCT *
    CASE FUNCT OF
      1 : CALL Data Input Process
      2 : CALL Sampling Process
      3 : CALL Data Output Process
      4 : * QUIT *
          * Save IDTOTAL *
          IST1 = TRUE
    ENDCASE:
  UNTIL (IST1=TRUE)
  * Close Random Files *
END:

END PROCEDURE: Statistical Test Process
```

E.2 DATA INPUT PROCESS

```
PROCEDURE: Data Input Process

INPUT : Keyboard, Data Input Resource, Process Curve Resource
OUTPUT: Display, Data Input Resource, Process Curve Resource,
        Random Number Resource

BEGIN:
  FLAG1 = FALSE
  REPEAT
    PUT:
      DISPLAY: DATA INPUT PROCESS
      DISPLAY: 1 : Read values in the data input resource
      DISPLAY: 2 : Read values in the process curve resource
      DISPLAY: 3 : Input required values
      DISPLAY: 4 : Erase a record
      DISPLAY: 5 : Quit
    ENDPUT:
    GET:
      KEYBOARD: IDI1
    ENDGET:
    CASE IDI1 OF
      1 : CALL Data Input Process A - See Appendix E2.1
      2 : CALL Data Input Process B - See Appendix E2.2
      3 : CALL Data Input Process C - See Appendix E2.3
      4 : CALL Data Input Process D - See Appendix E2.4
      5 : CALL Data Input Process E - See Appendix E2.5
    ENDCASE:
  UNTIL (FLAG1=TRUE)
END:

END PROCEDURE: Data Input Process
```

E.2.1 DATA INPUT PROCESS A

```
PROCEDURE: Data Input Process A

INPUT : Keyboard, Data Input Resource
OUTPUT: Display

BEGIN:
  IF IDTOTAL=0
    THEN
      PUT:
        DISPLAY: There are no records available
      ENDPUT:
    ELSE
      * Display headings *
      FOR IDA1 = 1 TO IDTOTAL
        * Display values *
      NEXT IDA1
  ENDIF:
END:

END PROCEDURE: Data Input Process A
```

E.2.2 DATA INPUT PROCESS B

```
PROCEDURE: Data Input Process B

INPUT : Keyboard, Process Curve Resource
OUTPUT: Display

BEGIN:
  IF IDTOTAL=0
    THEN
      PUT:
        DISPLAY: There are no records available
      ENDPUT:
    ELSE
      * Display headings *
      FOR IDB1 = 1 TO IDTOTAL
        * Display values *
      NEXT IDB1
  ENDIF:
END:

END PROCEDURE: Data Input Process B
```

E.2.3 DATA INPUT PROCESS C

```
PROCEDURE: Data Input Process C
```

```
INPUT : Keyboard, Random Number Resource, Process Curve Resource
OUTPUT: Display, Random Number Resource, Process Curve Resource,
        Data Input Resource

BEGIN:
  IF IDTOTAL=IDMAX
    THEN
      PUT:
        DISPLAY: Maximum number of records already exist
      ENDPUT:
    ELSE
      IDTOTAL = IDTOTAL+1
      ID = IDTOTAL
      * input the required data *
      * put the data in the random file buffer *
      GET:
        KEYBOARD: Use Existing or New process curve resource ; CDC1
      ENDGET:
      IDC1 = FALSE
      REPEAT
        IF CDC1=E
          THEN
            * input the required data *
            * copy the appropriate random number resource and store
              it under the correct filename *
            IDC1 = TRUE
          ELSE
            IF CDC1=N
              THEN
                * input the required data *
                * put data in the random file buffer *
                CALL CALC_P_C
                IDC1 = TRUE
            ENDIF:
        ENDIF:
      UNTIL (IDC1=TRUE)
  ENDIF:
END:


MODULE: CALC_P_C

BEGIN:
  CDCC1 = PROCC  * process curve name *
  CASE CDCC1 OF:
    SELF: Manual Input
      NODEF = 0
      IDCC2 = FALSE
      REPEAT
        * input the number of the product to fail *
        * generate a random test number *
        * save the values *
        NODEF = NODEF+1
        GET:
          KEYBOARD: Last number (Y/N) ; CDCC2
        ENDGET:
        IF CDCC2=Y
```

```
          THEN
            IDCC2 = TRUE
          ELSE
            * display the amount of products already input *
          ENDIF:
        UNTIL (IDCC2=TRUE)
        * sort the array according to product numbers; from smallest to
          largest.  When a product number is swapped  the corresponding
          test number must also be swapped. *
        * calculate and save PSIZE and PROB *
      TYPE-A: Binomial Distribution
        See Appendix B1
      TYPE-B: Horsnell Distribution
        See Appendix B2
    ENDCASE:
END:
END MODULE: CALC_P_C

END PROCEDURE: Data Input Process C
```

E.2.4 DATA INPUT PROCESS D

```
PROCEDURE: Data Input Process D

INPUT : Keyboard, Data Input Resource, Process Curve Resource
OUTPUT: Display, Data Input Resource, Process Curve Resource

BEGIN:
  IF IDTOTAL = 0
    THEN
      PUT:
        DISPLAY: There are no records available
      ENDPUT:
    ELSE
      GET:
        KEYBOARD: Input the ID of the record to be displayed; ID
      ENDGET:
      IDD1 = ID
      IDD2 = FALSE
      REPEAT
        IF ID = IDTOTAL
          THEN
            * Fill the record with blanks *
            IDD2 = TRUE
          ELSE
            IDD3 = ID
            ID = ID+1
            * GET record ID : PUT record IDD3 *
        ENDIF:
      UNTIL (IDD2=TRUE)
      PUT:
        DISPLAY: Record IDD1 has been erased
      ENDPUT:
  ENDIF:
```

```
END:

END PROCEDURE: Data Input Process D
```

E.2.5 DATA INPUT PROCESS E

```
PROCEDURE: Data Input Process E

INPUT : none
OUTPUT: FLAG1

BEGIN:
  FLAG1 = TRUE
  * save all the information required by the system, on floppy *
END:

END PROCEDURE: Data Input Process E
```

E.3 SAMPLING PROCESS

```
PROCEDURE: Sampling Process

INPUT : Keyboard, Random Number Resource, Process Curve Resource,
        Interface Unit, Data Input Resource, Data Output Resource
OUTPUT: Display,  Data Input Resource,  Data Output Resource,
        Interface Unit

BEGIN:
  PUT:
    DISPLAY: SAMPLING PROCESS
    DISPLAY: 1 : Run all test processes
    DISPLAY: 2 : Run individual test processes
    DISPLAY: 3 : Quit
    DISPLAY: Select the required option and press <enter>
  ENDPUT:
  GET:
    KEYBOARD: IS1
  ENDGET:
  CASE IS1 OF
    1 : * All tests *
      FOR ID = 1 TO IDTOTAL
        * Copy random number resource into an array *
        CALL Sampling Plans
      NEXT ID
    2 : * Individual Tests *
      GET:
        KEYBOARD: Enter the ID of the record ; ID
      ENDGET:
      * Copy random number resource to an array *
      CALL Sampling Plans
    3 : Quit
  ENDCASE:
END:


MODULE: Sampling Plans

BEGIN:
  IS3 = SSIZE
  CASE SPLAN OF
    100% : * Full Inspection *
      FOR PRODUCTNO = 1 TO IS3
        CALL Algorithm_A Process - See Appendix E3.1
        CALL Test Result Process - See Appendix E3.5
        PUT:
          DISPLAY: Record ID : Completed PRODUCTNO of IS3
        ENDPUT:
      NEXT PRODUCTNO
    CSP-1 :
      FLAG1 = FALSE
      FOR PROCUCTNO = 1 TO IS3
        CALL Algorithm_B Process - See Appendices E3.2, E3.3, E3.4
        CALL Test Result Process - See Appendix E3.5
        PUT:
```

```
          DISPLAY: Record ID : Completed PRODUCTNO of IS3
        ENDPUT:
  END CASE:
  SSTATUS = TRUE
END:
END MODULE: Sampling Plans

END PROCEDURE: Sampling Process
```

E.3.1 ALGORITHM_A PROCESS (100% TESTING)

NOTATION - The *Interim Resource* (*INTERRES*) is a two dimensional array. Two subscripts are required to reference any component in the array. One subscript refers to a specific record, in this case each test has its own record. The other subscript refers to a field in the record. There are eight fields in each record (TTYPE, TSTATUS, OUTCOME, COUNT, COUNTK, FLG1, FLG2, TIME) - See Chapter 7. A specific field (say TTYPE) in a specific record (say TESTNO) is referenced using the following expression.

```
INTERRES(TESTNO,TTYPE)
```

The notation used for this expression, for the following PDL description, is simply TTYPE.

```
TTYPE = INTERRES(TESTNO,TTYPE)
```

```
PROCEDURE: Algorithm_A Process

INPUT : Interim Resource, Interface Unit, Data Input Resource, PRODUCTNO
OUTPUT: Interim Resource, Interface Unit

CONSTANTS:
      FAILTIME = 2
      PASSTIME = 1
      NOTIME = 0

BEGIN:
  FLAG1=FALSE
  FOR TESTNO = 1 TO MAXTEST
    TTYPE = FULL
    COUNT = 0
    TSTATUS = RUN
    TIME = 0
    CALL Test Fail Process - See Appendix E3.6
    CALL Interface Unit Process - See Appendix E3.6
    IF STATUS=TRUE
      THEN
        OUTCOME = PASS
        TIME = PASSTIME
      ELSE
        IF STATUS=FALSE
          THEN
            OUTCOME = FAIL
```

```
            TIME = FAILTIME
          ELSE
            OUTCOME = UNKNOWN
        ENDIF:
    ENDIF:
  NEXT TESTNO
END:

END PROCEDURE: Algorithm_A  Process
```

E.3.2 ALGORITHM_B PROCESS (CSP SAMPLING)

NOTATION - The Interim Resource (INTERRES) is a two dimensional array. Two subscripts are required to reference any component in the array. One subscript refers to a specific record, in this case each test has its own record. The other subscript refers to a field in the record. There are eight fields in each record (TTYPE, TSTATUS, OUTCOME, COUNT, COUNTX, FLG1, FLG2, TIME) - See Chapter 7. A specific field (say TTYPE) in a specific record (say TESTNO) is referenced using the following expression.

```
INTERRES(TESTNO,TTYPE)
```

The notation used for this expression, for the following PDL description, is simply TTYPE.

```
TTYPE = INTERRES(TESTNO,TTYPE)
```

```
PROCEDURE: Algorithm_B Process ( CSP Sampling )

INPUT : Interim Resource, Interface Unit, Data Input Resource, PRODUCTNO
OUTPUT: Interim Resource, Interface Unit

CONSTANTS:
      FAILTIME = 2
      PASSTIME = 1
      NOTIME = 0

BEGIN:
  FLAG1 = FALSE
  IF FLAG2=FALSE
    THEN
      * initialise the Interim Resource *
      FOR TESTNO = 1 TO MAXTEST
        TTYPE = FULL
        COUNT = 0
        TSTATUS = RUN
        TIME = 0
        CALL Test Fail Process - See Appendix E3.6
        CALL Interface Unit Process - See Appendix E3.6
        IF STATUS=TRUE
          THEN
            OUTCOME = PASS
            TIME = PASSTIME
```

```
        ELSE
          IF STATUS=FALSE
            THEN
              OUTCOME = FAIL
              TIME = FAILTIME
            ELSE
              OUTCOME = UNKNOWN
          ENDIF:
        ENDIF:
      NEXT TESTNO
      FLAG2 = TRUE
    ELSE
      FOR TESTNO = 1 TO MAXTEST
        CALL Test Fail Resource - See Appendix E3.6
        CASE SPLAN OF:   * type of sampling plan *
          CSP-1 :
            IF OUTCOME=PASS
              THEN
                IF COUNT=1
                  THEN
                    IF TTYPE=FULL
                      THEN
                        TTYPE = PART
                        TSTATUS = NORUN
                        COUNT = F-1     * F = 1/f *
                      ELSE
                        TSTATUS = RUN
                        COUNT = F       * F = 1/f *
                    ENDIF:
                  ELSE
                    IF TTYPE=FULL
                      THEN
                        TSTATUS = RUN
                      ELSE
                        TSTATUS = NORUN
                    ENDIF:
                    COUNT = COUNT-1
                ENDIF:
              ENDIF:
              IF OUTCOME=FAIL
                THEN
                  COUNT = I     * I = i *
                  TTYPE = FULL
                  TSTATUS = RUN
              ENDIF:
              IF OUTCOME=UNKNOWN
                THEN
                  COUNT = COUNT-1
              ENDIF: ( end of CSP-1 )
            CSP-2 :
              IF OUTCOME=PASS
                THEN
                  IF TTYPE=FULL
                    THEN
                      IF COUNT=1
                        THEN
                          TTYPE = PART
```

```
                    TSTATUS = NORUN
                    COUNT = F-1     * F = 1/f *
                    FLG1 = FALSE
                  ELSE
                    TSTATUS = RUN
                    COUNT = COUNT-1
                ENDIF:
              ELSE
                IF COUNT=1
                  THEN
                    TSTATUS = RUN
                    COUNT = F     * F = 1/f *
                    IF FLAG1=TRUE
                      THEN
                        IF COUNTK=1
                          THEN
                            FLG1 = FALSE
                          ELSE
                            COUNTK = COUNTK-1
                        ENDIF:
                    ENDIF:
                  ELSE
                    TSTATUS = NORUN
                    COUNT = COUNT-1
                ENDIF:
            ENDIF:
          ELSE
            IF OUTCOME=FAIL
              THEN
                IF TTYPE=PART
                  THEN
                    IF FLG1=FALSE
                      THEN
                        COUNTK = I     * I = i *
                        COUNT = F-1    * F = 1/f *
                        TSTATUS = NORUN
                        FLG1 = TRUE
                      ELSE
                        TSTATUS = RUN
                        TTYPE = FULL
                        COUNT = I     * I = i *
                    ENDIF:
                  ELSE
                    TSTATUS = RUN
                    COUNT = I     * I = i *
                ENDIF:
              ELSE
                * OUTCOME = UNKNOWN *
                COUNT = COUNT-1
            ENDIF:
        ENDIF: ( end of CSP-2 )
      CSP-3 :
        IF OUTCOME=PASS
          THEN
            IF TTYPE=PART
              THEN
                IF FLG2=FALSE
```

```
                THEN
                  IF COUNT=1
                    THEN
                      TSTATUS = RUN
                      COUNT = F     * F = 1/f *
                    ELSE
                      TSTATUS = NORUN
                      COUNT = COUNT-1
                  ENDIF:
                ELSE
                  IF COUNT=1
                    THEN
                      TSTATUS = NORUN
                      COUNT = F-1     * F = 1/f *
                      FLG2 = FALSE
                    ELSE
                      TSTATUS = RUN
                      COUNT = COUNT-1
                  ENDIF:
              ENDIF:
              IF TSTATUS=RUN
                THEN
                  IF FLG1=TRUE
                    THEN
                      IF COUNTK=1
                        THEN
                          IF FLG2=FALSE
                            THEN
                              FLG1=FALSE
                          ENDIF:
                        ELSE
                          COUNTK = COUNTK-1
                      ENDIF:
                  ENDIF:
              ENDIF:
            ELSE
              IF COUNT=1
                THEN
                  TSTATUS = NORUN
                  COUNT = F-1     * F = 1/f *
                  TTYPE = PART
                  FLG1 = FALSE
                  FLG2 = FALSE
                ELSE
                  TSTATUS = RUN
                  COUNT = COUNT-1
              ENDIF:
          ENDIF:
        ELSE
          IF OUTCOME=FAIL
            THEN
              IF TTYPE=PART
                THEN
                  IF FLG1=FALSE
                    THEN
                      TSTATUS = RUN
                      TTYPE = PART
```

```
                        COUNTK = I     * I = i *
                        COUNT = 4
                        FLG1 = TRUE
                        FLG2 = TRUE
                      ELSE
                        TSTATUS = RUN
                        COUNT = I     * I = i *
                        TTYPE = FULL
                      ENDIF:
                    ELSE
                      TSTATUS = RUN
                      COUNT = I     * I = i *
                      TTYPE = FULL
                    ENDIF:
                    OUTCOME = UNKNOWN *
                  COUNT = COUNT-1
                ENDIF:
              ENDIF: ( end of CSP-3 )
            ENDCASE:
            IF STATUS=RUN
              THEN
                CALL Interface Unit Process - See Appendix E3.6
                IF STATUS=TRUE
                  THEN
                    OUTCOME = PASS
                    TIME = PASSTIME
                  ELSE
                    IF STATUS=FALSE
                      THEN
                        OUTCOME = FAIL
                        TIME = FAILTIME
                      ELSE
                        OUTCOME = UNKNOWN
                      ENDIF:
                ENDIF:
              ELSE
                OUTCOME = PASS
                TIME = NOTIME
            ENDIF:
          NEXT TESTNO
        ENDIF:
      END:

      END PROCEDURE: Algorithm_B  Process ( CSP Sampling )
```

E.3.3 ALGORITHM_B PROCESS (MLP SAMPLING)

PROCEDURE: Algorithm_B Process (MLP Sampling)

Only the CASE function differs from the PDL description of the Algorithe_B Process in Appendix E3.2 (Algorithm_B Process - CSP Sampling). The CASE function which follows replaces that of Appendix E3.2 when the multi-level continuous inspection plans are executed.

```
CASE SPLAN OF:   * Sampling Plan *
  MLP :
    IF OUTCOME=PASS
      THEN
        IF TTYPE=FULL
          THEN
            IF COUNT=1
              THEN
                TTYPE = PART
                TSTATUS = NORUN
                COUNT = P-1     * P = 1/f *
                COUNTK = I      * I = i *
                FLG1 = 1
              ELSE
                TSTATUS = RUN
                COUNT = COUNT-1
            ENDIF:
          ELSE
            IF COUNT=1
              THEN
                IF COUNTK=1
                  THEN
                    IF FLG1<2
                      THEN
                        FLG1 = FLG1+1
                    ENDIF:
                    TSTATUS = RUN
                    COUNTK = I       * I = i *
                    COUNT = P**FLG1 * P = 1/f *
                  ELSE
                    TSTATUS = RUN
                    COUNTK = COUNTK-1
                    COUNT = P**FLG1 * P = 1/f *
                ENDIF:
              ELSE
                TSTATUS = NORUN
                COUNT = COUNT-1
            ENDIF:
        ENDIF:
      ELSE
        IF OUTCOME=FAIL
          THEN
            IF TTYPE=FULL
              THEN
                TSTATUS = RUN
                COUNT = I     * I = i *
                FLG1 = 0
              ELSE
                IF FLG1=1
                  THEN
                    TSTATUS = RUN
                    TTYPE = FULL
                    COUNT = I     * I = i *
                    FLG1 = 0
                  ELSE
                    TSTATUS = NORUN
```

```
                    COUNTK = I    * I = i *
                    FLG1 = FLG1-1
                    COUNT = (F**FLG1)-1
                  ENDIF:
              ENDIF:
          ENDIF:
      ENDIF: ( end of MLP sampling plan )
    MLP-T :
      IF OUTCOME=PASS
        THEN
          IF TTYPE=FULL
            THEN
              IF COUNT=1
                THEN
                  TTYPE = PART
                  TSTATUS = NORUN
                  COUNT = F-1     * F = 1/f *
                  COUNTK = I      * I = i *
                  FLG1 = 1
                ELSE
                  TSTATUS = RUN
                  COUNT = COUNT-1
              ENDIF:
            ELSE
              IF COUNT=1
                THEN
                  IF COUNTK=1
                    THEN
                      IF FLG1<2
                        THEN
                          FLG1 = FLG1+1
                      ENDIF:
                      TSTATUS = RUN
                      COUNTK = I        * I = i *
                      COUNT = F**FLG1 * F = 1/f *
                    ELSE
                      TSTATUS = RUN
                      COUNTK = COUNTK-1
                      COUNT = F**FLG1 * F = 1/f *
                  ENDIF:
                ELSE
                  TSTATUS = NORUN
                  COUNT = COUNT-1
              ENDIF:
          ENDIF:
        ELSE
          IF OUTCOME=FAIL
            THEN
              TSTATUS = RUN
              TTYPE = FULL
              COUNT = I     * I = i *
              FLG1 = 0
          ENDIF:
      ENDIF: ( end of MLP-T sampling plan )
  ENDCASE:

  END PROCEDURE: Algorithm_B Process ( MLP Sampling )
```

E.3.4 ALGORITHM_B PROCESS (CSP AND WSP SAMPLING)

NOTATION - The Interim Resource (INTERRES) is a two dimensional array. Two subscripts are required to reference any component in the array. One subscript refers to a specific record, in this case each test has its own record. The other subscript refers to a field in the record. There are eight fields in each record (TTYPE, TSTATUS, OUTCOME, COUNT, COUNTK, FLG1, FLG2, TIME) - See Chapter 7. A specific field (say TTYPE) in a specific record (say TESTNO) is referenced using the following expression.

```
INTERRES(TESTNO,TTYPE)
```

The notation used for this expression, for the following PDL description, is simply TTYPE.

```
TTYPE = INTERRES(TESTNO,TTYPE)
```

```
PROCEDURE: Algorithm_B Process ( CSP and WSP Sampling )

INPUT : Interim Resource, Interface Unit, Data Input Resource, PRODUCTNO
OUTPUT: Interim Resource, Interface Unit

CONSTANTS:
       FAILTIME = 2
       PASSTIME = 1
       NOTIME = 0

BEGIN:
  FLAG1 = FALSE
  IF FLAG2=FALSE
    THEN
      * initialise the Interim Resource *
      FOR TESTNO = 1 TO MAXTEST
        IF SPLAN=WSP-1
          THEN
            AOQL! = AOQL/MAXTEST
            AOQL! = (AOQL!*SSIZE)/(F-1)
            IF AOQL!<1
              THEN
                TTYPE = FULL
                TSTATUS = RUN
              ELSE
                TTYPE = PART
                TSTATUS = RUN
                COUNT = F     * F = 1/f *
                COUNTK = INT( AOQL! )
            ENDIF:
          ELSE  * SPLAN = CSP-F *
            TTYPE = FULL
            TSTATUS = RUN
            COUNT = I     * I = i *
            TIME = 0
        ENDIF:
        CALL Test Fail Process - See Appendix E3.6
        CALL Interface Unit Process - See Appendix E3.6
```

```
        IF STATUS=TRUE
          THEN
            OUTCOME = PASS
            TIME = PASSTIME
          ELSE
            IF STATUS=FALSE
              THEN
                OUTCOME = FAIL
                TIME = FAILTIME
              ELSE
                OUTCOME = UNKNOWN
              ENDIF:
        ENDIF:
      NEXT TESTNO
      FLAG2 = TRUE
    ELSE
      FOR TESTNO = 1 TO MAXTEST
        CALL Test Fail Process - See Appendix E3.6
        CASE SPLAN OF:   * Sampling Plan *
          CSP-F :
            NOTE : Plan CSP-F has the same operation as Plan CSP-1
          WSP-1 :
            IF TTYPE=PART
              THEN
                IF OUTCOME=PASS
                  THEN
                    IF COUNT=1
                      THEN
                        COUNT = F     * F = 1/f *
                        TSTATUS = RUN
                      ELSE
                        COUNT = COUNT-1
                        TSTATUS = NORUN
                    ENDIF:
                  ELSE
                    IF OUTCOME=FAIL
                      THEN
                        IF COUNTK=1
                          THEN
                            TTYPE = FULL
                            TSTATUS = RUN
                          ELSE
                            TSTATUS = NORUN
                            COUNTK = COUNTK-1
                        ENDIF:
                    ENDIF:
                ENDIF:
            ENDIF:
        ENDCASE:
        IF STATUS=RUN
          THEN
            CALL Interface Unit Process - See Appendix E3.6
            IF STATUS=TRUE
              THEN
                OUTCOME = PASS
                TIME = PASSTIME
              ELSE
```

```
                IF STATUS=FALSE
                  THEN
                    OUTCOME = FAIL
                    TIME = FAILTIME
                  ELSE
                    OUTCOME = UNKNOWN
                ENDIF:
            ENDIF:
          ELSE
            OUTCOME = PASS
            TIME = NOTIME
        ENDIF:
      NEXT TESTNO
  ENDIF:
END:

END PROCEDURE: Algorithm_B  Process ( CSP and WSP Sampling )
```

E.3.5 TEST RESULT PROCESS

```
PROCEDURE: Test Result Process

INPUT : Interim Resource, Data Output Resource, Process Curve Resource,
        Random Number Resource, Data Input Resource, PRODUCTNO
OUTPUT: Data Output Resource

BEGIN:
  * Copy the results of the Interim Resource to the Data Output Resource
  *
  TOTALTIME = 0
  FOR TESTNO = 1 TO MAXTEST
    * Check for FAIL OUTCOME's and calculate DEFNUMBER *
    TOTALTIME = TOTALTIME+TIME
  NEXT TESTNO
  * Save TOTALTIME in the Data Output Resource *
  * Calculate RUNTIME and store it on the floppy *
  * If there are 25 records store them on the floppy and start again *
END:

END PROCEDURE: Test Result Process
```

E.3.6 MODULES FOR ALGORITHM PROCESS

```
MODULE: Test Fail Process

INPUT : TESTNO
OUTPUT: Interres Unit Resource

BEGIN:
  IF TESTNO <> 8   * test 8 *
    THEN
```

```
      IF FLAG1=FALSE
        THEN
          NODEF = PSIZE
          IF VAR<=NODEF
            THEN
              IF PRODUCTNO < RNRES(TESTNO,PNO)
                THEN
                  TEST=0
                  KEYNO = FFH
                ELSE
                  TEST = RNRES(TESTNO,TNO)
                  CASE TEST OF
                    1 : KEYNO = FEH
                    2 : KEYNO = FDH
                    3 : KEYNO = DFH
                    4 : KEYNO = FBH
                    5 : KEYNO = F7H
                    6 : KEYNO = EFH
                    7 : KEYNO = 7FH
                    8 : KEYNO = FFH
                  ENDCASE:
                  VAR = VAR+1
              ENDIF:
            ELSE
              TEST=0
              KEYNO = FFH
          ENDIF:
          PUT:
            OUT 304H,KEYNO
          ENDPUT:
          FLAG1 = TRUE
        ENDIF:
      ELSE
        IF TEST=8
          THEN
            KEYNO = FCH
          ELSE
            KEYNO = FFH
        ENDIF:
        PUT:
          OUT 304H,KEYNO
        ENDPUT:
        FLAG1=FALSE
    ENDIF:
  END:
  END MODULE: Test Fail Process


  MODULE: Interface Unit Process

  INPUTS : TESTNO
  OUTPUTS: STATUS

  CONSTANTS:
         MDA=300H          RDRC=300H
         SETRE=303H        RDDAT=302H
         TESTDAT(1)=80H  TESTDAT(5)=04H
```

```
      TESTDAT(2)=40H  TESTDAT(6)=02H
      TESTDAT(3)=20H  TESTDAT(7)=01H
      TESTDAT(4)=08H  TESTDAT(8)=C0H

BEGIN:
  * make data available *
    OUT MDA,TESTDAT(TESTNO)
  * set RE high *
    OUT SETRE,00H
  WHILE ( RC is low ) *wait*
    RC=IN (RDRC.AND.01H)
  ENDWHILE:
  WHILE (RC is high ) *wait*
    RC=IN (RDRC.AND.01H)
  ENDWHILE:
  * get data *
    STATUS=IN (RDDAT)
  * set RE low *
    OUT SETRE,00H
END:
END MODULE    terface Unit Process
```

E.4 DATA OUTPUT PROCESS

```
PROCEDURE: Data Output Process

INPUT : Keyboard, Data Input Resource, Data Output Resource,
        Process Curve Resource, Random Number Resource
OUTPUT: Display, Printer

BEGIN:
  IDO4 = FALSE
  REPEAT
    PUT:
      DISPLAY: DATA OUTPUT PROCESS
      DISPLAY: 1 : Output a datafile to the printer
      DISPLAY: 2 : Output a datafile to the screen
      DISPLAY: 3 : Output a rnresource to the printer
      DISPLAY: 4 : Output a rnresource to the screen
      DISPLAY: 5 : Quit
    ENDPUT:
    GET:
      KEYBOARD: IDO1
    ENDGET:
    CASE IDO1 OF
      1 : * Print the appropriate headings and results *
      2 : * Display the appropriate headings and results *
      3 : * Print the appropriate headings and results *
      4 : * Display the appropriate headings and results *
      5 : * Quit *
          IDO4 = TRUE
    ENDCASE:
  UNTIL (IDO4=TRUE)
END:

END PROCEDURE: Data Output Process
```

APPENDIX F. EVALUATION RESULTS

F.1 CONTINUOUS INSPECTION PLAN RESULTS

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
1a	100%	10000	0.02	200	0	0	0	0	80200	TYPE-A
1b	MLP	"	"	"	1000	5	0.00125 / 0.01	0.0026	69324	"
1c	MLP-T	"	"	"	1000	5	"	0.0026	69324	"
1d	CSP-1	"	"	"	700	5	"	0.0066	53261	"
1e	CSP-2	"	"	"	1000	5	"	0.0045	62507	"
1f	CSP-3	"	"	"	1000	5	"	0.0042	63344	"
2a	100%	10000	0.02	215	0	0	0	0	80215	TYPE-B 0.0025 0.65
2b	MLP	"	"	"	1000	5	0.00125 / 0.01	0.0042	64557	"
2c	MLP-T	"	"	"	1000	5	"	0.0042	64557	"
2d	CSP-1	"	"	"	700	5	"	0.0082	51441	"
2e	CSP-2	"	"	"	1000	5	"	0.0073	53715	"
2f	CSP-3	"	"	"	1000	5	"	0.0082	51441	"
3a	100%	10000	0.02	200	0	0	0	0	80200	TYPE-A
3b	MLP	"	"	"	1300	10	0.00125 / 0.01	0.0007	76272	"
3c	MLP-T	"	"	"	1300	10	"	0.0007	76272	"
3d	CSP-1	"	"	"	1000	10	"	0.0042	62289	"
3e	CSP-2	"	"	"	1300	10	"	0.0016	74035	"
3f	CSP-3	"	"	"	1300	10	"	0.0018	72485	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
4a	100%	10000	0.02	214	0	0	0	0	80214	TYPE-B 0.0025 0.85
4b	MLP	"	"	"	1300	10	0.00125 / 0.01	0.0023	69524	"
4c	MLP-T	"	"	"	1300	10	"	0.0023	69524	"
4d	CSP-1	"	"	"	1000	10	"	0.0049	60352	"
4e	CSP-2	"	"	"	1300	10	"	0.0057	59992	"
4f	CSP-3	"	"	"	1300	10	"	0.0049	60353	"
5a	100%	10000	0.02	215	0	0	0	0	80215	TYPE-B 0.0025 0.85
5b	MLP	"	"	"	740	3	0.00125 / 0.01	0.0051	56713	"
5c	MLP-T	"	"	"	740	3	"	0.0048	58909	"
5d	CSP-1	"	"	"	410	3	"	0.0083	48808	"
5e	CSP-2	"	"	"	660	3	"	0.0068	52929	"
5f	CSP-3	"	"	"	660	3	"	0.0069	51474	"
6a	100%	10000	0.01	164	0	0	0	0	80164	TYPE-B 0.01 0.85
6b	MLP	"	"	"	740	3	0.00125 / 0.01	0.0043	42920	"
6c	MLP-T	"	"	"	740	3	"	0.0045	43697	"
6d	CSP-1	"	"	"	410	3	"	0.0050	40023	"
6e	CSP-2	"	"	"	660	3	"	0.0062	37270	"
6f	CSP-3	"	"	"	660	3	"	0.0065	37635	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
7a	100%	10000	0.01	176	0	0	0	0	80176	TYPE-B 0.01 0.50
7b	MLP	"	"	"	740	3	0.00125 / 0.01	0.0064	36627	"
7c	MLP-T	"	"	"	740	3	"	0.0066	38013	"
7d	CSP-1	"	"	"	410	3	"	0.0061	37690	"
7e	CSP-2	"	"	"	660	3	"	0.0050	39144	"
7f	CSP-3	"	"	"	660	3	"	0.0062	33773	"
8a	100%	10000	0.01	176	0	0	0	0	80176	TYPE-B 0.01 0.50
8b	MLP	"	"	"	740	3	0.00125 / 0.01	0.0044	39059	"
8c	MLP-T	"	"	"	740	3	"	0.0044	41034	"
8d	CSP-1	"	"	"	410	3	"	0.0054	37907	"
8e	CSP-2	"	"	"	660	3	"	0.0045	40867	"
8f	CSP-3	"	"	"	660	3	"	0.0037	41229	"
9a	100%	10000	0.03	372	0	0	0	0	80372	TYPE-B 0.01 0.70
9b	MLP	"	"	"	1000	5	0.00125 / 0.01	0.0033	73227	"
9c	MLP-T	"	"	"	1000	5	"	0.0033	73227	"
9d	CSP-1	"	"	"	700	5	"	0.0053	64998	"
9e	CSP-2	"	"	"	1000	5	"	0.0042	71110	"
9f	CSP-3	"	"	"	1000	5	"	0.0044	69932	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AvQ	Total Time	PC Type
10a	100%	10000	0.005	66	0	0	0	0	80066	TYPE-B 0.002 0.40
10b	MLP	"	"	"	740	3	0.00125 / 0.01	0.0032	30134	"
10c	MLP-T	"	"	"	740	3	"	0.0031	32227	"
10d	CSP-1	"	"	"	410	3	"	0.0027	35197	"
10e	CSP-2	"	"	"	660	3	"	0.0035	33366	"
10f	CSP-3	"	"	"	660	3	"	0.0030	33454	"

F.2 BATCH INSPECTION PLAN RESULTS

F.2.1 PLAN CSP-F RESULTS

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
1a	100%	700	0.02	14	0	0	0	0	5614	TYPE-A
1b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0086	3534	"
2a	100%	700	0.02	15	0	0	0	0	5615	TYPE-B 0.0025 0.85
2b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0086	3434	"
3a	100%	700	0.02	21	0	0	0	0	5621	TYPE-B 0.01 0.50
3b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0157	3634	"
4a	100%	500	0.02	10	0	0	0	0	4010	TYPE-A
4b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0020	3373	"
5a	100%	500	0.02	11	0	0	0	0	4011	TYPE-B 0.0025 0.85
5b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0020	3187	"
6a	100%	500	0.02	15	0	0	0	0	4015	TYPE-B 0.01 0.50
6b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0012	3313	"
7a	100%	500	0.01	8	0	0	0	0	4008	TYPE-B 0.01 0.50
7b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0080	3018	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
8a	100%	500	0.01	9	0	0	0	0	4009	TYPE-B 0.015 0.70
8b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0080	2836	"
9a	100%	500	0.03	17	0	0	0	0	4017	TYPE-B 0.015 0.70
9b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0080	3442	"
10a	100%	700	0.01	13	0	0	0	0	5613	TYPE-B 0.01 0.50
10b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0043	3238	"
11a	100%	700	0.01	11	0	0	0	0	5611	TYPE-B 0.01 0.70
11b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.01143	2934	"
12a	100%	700	0.03	27	0	0	0	0	5627	TYPE-B 0.015 0.70
12b	CSP-F	"	"	"	302	10	0.00143 / 0.0114	0.0214	3736	"

F.2.2 PLAN WSP-1 RESULTS

WSP-1 results for a batch of 500 products.

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
1a	100%	500	0.02	10	0	0	0	0	4010	TYPE-A
1b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0060	2606	"
2a	100%	500	0.02	11	0	0	0	0	4011	TYPE-B 0.005 0.85
2b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0040	2725	"
3a	100%	500	0.02	11	0	0	0	0	4011	TYPE-B 0.015 0.70
3b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0040	2378	"
4a	100%	500	0.05	25	0	0	0	0	4025	TYPE-A
4b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0180	3047	"
5a	100%	500	0.03	19	0	0	0	0	4019	TYPE-B 0.01 0.85
5b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0080	2663	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
6a	100%	500	0.02	12	0	0	0	0	4012	TYPE-B 0.005 0.85
6b	WSP-1	"	"	"	N/A	3	0.004 / 0.034	0.0060	2003	"
7a	100%	500	0.01	12	0	0	0	0	4012	TYPE-B 0.015 0.70
7b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0140	1542	"
8a	100%	500	0.03	16	0	0	0	0	4018	TYPE-B 0.01 0.85
8b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0160	2511	"
9a	100%	500	0.03	20	0	0	0	0	4020	TYPE-B 0.01 0.70
9b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0240	1799	"
10a	100%	500	0.02	14	0	0	0	0	4014	TYPE-B 0.01 0.85
10a	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0160	1945	"

WSP-1 results for a batch of 700 products.

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
1a	100%	700	0.02	14	0	0	0	0	5614	TYPE-A
1b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0043	3380	"
2a	100%	700	0.02	16	0	0	0	0	5616	TYPE-B 0.005 0.85
2b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0086	3988	"
3a	100%	700	0.01	16	0	0	0	0	5616	TYPE-B 0.015 0.70
3b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0086	3141	"
4a	100%	700	0.03	26	0	0	0	0	5626	TYPE-B 0.01 0.85
4b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0100	4068	"
5a	100%	700	0.05	35	0	0	0	0	5635	TYPE-A
5b	WSP-1	"	"	"	N/A	2	0.002 / 0.016	0.0086	4687	"

Trial No	Test Type	Sample Size	PC Prob	PC Size	I	F	AOQL t/p	AOQ	Total Time	PC Type
6a	100%	700	0.02	17	0	0	0	0	5617	TYPE-B 0.005 0.85
6b	WSP-1	"	"	"	N/A	3	0.004 / 0.034	0.0029	2753	"
7a	100%	700	0.01	13	0	0	0	0	5613	TYPE-B 0.015 0.70
7b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0029	2683	"
8a	100%	700	0.03	27	0	0	0	0	5627	TYPE-B 0.01 0.85
8b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0143	3485	"
9a	100%	700	0.05	35	0	0	0	0	5635	TYPE-A
9b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0229	3537	"
10a	100%	700	0.04	26	0	0	0	0	5626	TYPE-B 0.01 0.70
10b	WSP-1	"	"	"	N/A	3	0.004 / 0.032	0.0129	3129	"

APPENDIX G. SOFTWARE LISTING

G.1 CONTINUOUS INSPECTION PLAN SOFTWARE LISTING

```
10 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
20 REM Statistical Test Process Initialize Variables
30 '   # = I - integer , C - character , R - real
40 '   ? = 1,2,3,.....
50     DEFINT I    : DEFSTR C    : DEFSNG R
60     RANDOMIZE TIMER : OPTION BASE 1
70 '   constant:
80 '     global:
90 '      integer:
100           IDMAX=10     : MAXTEST%=8   : TRUE=&HFF    : FALSE=&H0
110 '         * Data Output Resource *
120           TTYPE%=1     : TSTATUS%=2   : OUTCOME%=3   : COUNT%=1
130           TIME%=2      : PSTATUS%=4   : TNUM%=3      : PTIME%=4
140           PFAIL%=5     : PRODNO%=1    : COUNTK%=3    : FLG1%=4
              FLG2%=5
150 '         * Random Number Resource *
160           PNO%=1       : TNO%=2
170 '  variables:
180 '    local:
190 '      #ST?
200 '      integer:
210 '        FUNCT%
220 '    global:
230 '      integer:
240           ID=0         : IDTOTAL=0
250 '         KEYNO%       : TEST%         : TESTNO%      : PRODUCTNO%
260 '      array:
270 '        * Data Output Resource *
280           DIM DORES$(25,4)          : DIM DORES%(25,5)
290 '        * Random Number Resource *
300           DIM RNRES%(500,2)
310 '      real:
320 '        AOQ!
330 '  Data Input Process:
340 '    variable:
350 '      local:
360 '        #DT?
370 '        module:
380 '          D_I_PRO_A:  D_I_PRO_B:  D_I_PRO_C: D_I_PRO_D:
390 '            #DA?         #DB?          #DC?         #DD?
400 '      module global:
410 '        integer:
420 '          FLAG1% , NODEF%
430 '  Sampling Process:
440 '    constant:
450 '      module:
460 '        Interfsce Unit Proce:s
470 '          module:
480 '            Interface
490 '              array:
500 '                * Test Data Resource *
510                  DIM TESTDAT%(8)
520 '              integer:
530                  TESTDAT%(1)=&H80     : TESTDAT%(5)=&H4
540                  TESTDAT%(2)=&H40     : TESTDAT%(6)=&H2
550                  TESTDAT%(3)=&H20     : TESTDAT%(7)=&H1
560                  TESTDAT%(4)=&H8      : TESTDAT%(8)=&HC0
```

```
570                 MDA%=&H300            : SETRE%=&H303
580                 RDRC%=&H300           : RDDAT%=&H302
590 '       Test Fail Process
600 '         integer:
610             TFAIL%=&H304
620 '   varible:
630 '     local:
640 '     module:
650 '     module global:
660 '       integer:
670           NOTIME%=0  : PASSTIME%=1  : FAILTIME%=2  ' STATUS%
680           REF%=1
690 '     module:
700 '       integer:
710           VAR%=1     : FLAG1%=FALSE  : FLAG2%=FALSE  : DEFNUMBER%=0
720 '       array:
730 '         * Interim Resource *
740           DIM INTERRES$(8,3)          : DIM INTERRES%(8,5)
750 '     module:
760 '       Test Fail Process
770 '         #TF?
780 '       Interface Unit Process
790 '         module:
800 '           Interface
810 '             integer:
820 '               IN%
830 ' Result Output Process:
840 '   varible:
850 '     local:
860 '       #RO?
870 ' Initialise IDTOTAL
880     OPEN "initial" AS #5 LEN=20
890     FIELD #5, 4 AS IDTOTAL$, 12 AS RUNTIME$, 4 AS DEFNUMBER$
900     GET #5,1 : IDTOTAL=CVI(IDTOTAL$)
910 REM End of Initialise Variables
920 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
930 REM ************************************************************
940 REM Statistical Test Process
950 'begin:
960     OPEN "P_C_RES" AS #1 LEN=36
970     FIELD #1, 8 AS PROCC$ , 8 AS PROB$ , 10 AS PCFN$ , 4 AS PSIZE$
980     OPEN "D_I_RES" AS #2 LEN=43
990     FIELD #2, 6 AS SPLAN$ , 10 AS DOFN$ , 6 AS SSIZE$ , 8 AS AOQL$ ,
                4 AS I$ , 4 AS F$ , 5 AS SSTATUS$
1000    IST1 = FALSE
1010 '  repeat
1020 '    put:
1030        CLS
1040        PRINT " "
1050        PRINT "  STATISTICAL TEST PROCESS "
1060        PRINT " "
1070        PRINT "  1 : Data Input Process "
1080        PRINT "  2 : Sampling Process "
1090        PRINT "  3 : Data Output Process "
1100        PRINT "  4 : Quit "
1110        PRINT "  Select the required option and press <enter> "
1120        PRINT " "
```

```
1130 '    end put:
1140 '    get:
1150        INPUT FUNCT%
1160 '    end get:
1170      IF FUNCT% < 1 OR FUNCT% > 4 THEN GOTO 1020
1180 '    case FUNCT% of
1190      ON FUNCT% GOTO 1200,1220,1240,1260
1200 '      1 : call DATA INPUT PROCESS ( D_I_PRO )
1210            GOSUB 1370  : GOTO 1290
1220 '      2 : call SAMPLING PROCESS ( SAMPLE_PRO )
1230            GOSUB 5200  : GOTO 1290
1240 '      3 : call DATA OUTPUT PROCESS ( D_O_PRO )
1250            GOSUB 10000 : GOTO 1290
1260 '      4 : quit
1270            LSET IDTOTAL$ = MKI$(IDTOTAL)
1280            PUT #5,1 : IST1 = TRUE
1290 '    end case:
1300 '  until (IST1 = TRUE)
1310    IF IST1 = FALSE THEN GOTO 1010
1320    CLOSE
1330 'end:
1340 REM End of Statistical Test Procedure
1350 REM ************************************************************************
1360 END
1370 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1380 REM Data Input Process  ( D_I_PRO )
1390 'begin:
1400    FLAG1%=FALSE
1410 '  repeat
1420 '    put:
1430        PRINT " "
1440        PRINT "  DATA INPUT PROCESS "
1450        PRINT " "
1460        PRINT "  1 : Read values in the data input resource "
1470        PRINT "  2 : Read values in the process curve resource "
1480        PRINT "  3 : Input required values "
1490        PRINT "  4 : Erase a record "
1500        PRINT "  5 : Quit "
1510        PRINT "  Select the required option and press <enter> "
1520        PRINT " "
1530 '    end put:
1540 '    get:
1550        INPUT IDI1
1560 '    end get:
1570      IF IDI1 < 1 OR IDi1 > 5 THEN GOTO 1420
1580 '    case IDI1 of
1590      ON IDI1 GOSUB 1710 , 2060 , 2360 , 4500 , 4960
1600 '      1 : call D_I_PRO_A
1610 '      2 : call D_I_PRO_B
1620 '      3 : call D_I_PRO_C
1630 '      4 : call D_I_PRO_D
1640 '      5 : call D_I_PRO_E
1650 '    end case:
1660 '  until (FLAG1%=TRUE)
1670    IF FLAG1% = FALSE THEN GOTO 1410
1680 'end:
1690    RETURN
```

```
1700 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
1710 REM Data Input Process A ( D_I_PRO_A )
1720 'begin:
1730    IF IDTOTAL = 0
          THEN GOTO 1740 ELSE GOTO 1790
1740 '      put:
1750          PRINT " "
1760          PRINT "  There are no records available "
1770 '      endput:
1780        GOTO 2010
1790 '    else
1800 '      put:
1810          PRINT " "
1820          PRINT "  ID    Sample   Data-O/P   Sample   ";
1830          PRINT "  AOQL     I      F     Sample "
1840          PRINT "          Plan     File NM    Size    ";
1850          PRINT "                          Status "
1860          PRINT " "
1870 '      endput:
1880        FOR IDA1 = 1 TO IDTOTAL
1890          ID = IDA1 : GET #2,IDA1
1900 '        put:
1910            PRINT USING "#####     ";ID ;
1920            PRINT USING "\     \   ";SPLAN$ ;
1930            PRINT USING "\     \ ";DOFN$ ;
1940            PRINT USING "    ##### ";CVI(SSIZE$) ;
1950            PRINT USING " #.##### ";CVS(AOQL$) ;
1960            PRINT USING "####  ";CVI(I$) ;
1970            PRINT USING "####     ";CVI(F$) ;
1980            PRINT USING "&";SSTATUS$
1990 '        endput:
2000        NEXT IDA1
2010 '  endif:
2020 'end:
2030  RETURN
2040 REM End of Data Input Process A ( D_I_PRO_A )
2050 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
2060 REM Data Input Process B ( D_I_PRO_B )
2070 'begin:
2080    IF IDTOTAL = 0
          THEN GOTO 2090 ELSE GOTO 2140
2090 '      put:
2100          PRINT " "
2110          PRINT "  There are no records available "
2120 '      endput:
2130        GOTO 2310
2140 '    else
2150 '      put:
2160          PRINT " "
2170          PRINT "  ID    Process   Pro-Curve     Prob      Pro-Curve"
2180          PRINT "        Curve     File NM                 Size "
2190          PRINT " "
2200 '      endput:
2210        FOR IDB1 = 1 TO IDTOTAL
2220          ID = IDB1 : GET #1,IDB1
2230 '        put:
2240            PRINT USING "#####   ";ID ;
```

```
2250              PRINT USING " \     \    ";PROCC$ ;
2260              PRINT USING " \     \    ";PCFN$ ;
2270              PRINT USING "     #.#####";CVS(PRCB$);
2280              PRINT USING "      #####";CVI(PSIZE$)
2290 '          endput:
2300          NEXT IDB1
2310 '    endif:
2320 'end:
2330  RETURN
2340 REM End of Data Input Process B ( D_I_PRO_B )
2350 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
2360 REM Data Input Process C ( D_I_PRO_C )
2370 'begin:
2380     IF IDTOTAL = IDMAX
           THEN GOTO 2390 ELSE GOTO 2430
2390 '       put:
2400           PRINT "  Maximum number of records already exist "
2410 '       endput:
2420         GOTO 3230
2430 '     else
2440         IDTOTAL = IDTOTAL+1 : ID = IDTOTAL
2450 '       put:
2460           PRINT " " : PRINT "  Supply the following data : "
2470 '       endput:
2480 '       get:
2490           INPUT "  Sample plan ";CDC1
2500           INPUT "  Data output filename ";CDC2
2510           INPUT "  Sample size ";IDC1
2520 '       endget:
2530         IF CDC1="100%" THEN IDC2 = 1 : GOTO 2560
2540         IF CDC1="CSP-1" OR CDC1="csp-1" OR CDC1="CSP-2"
              CDC1="csp-2" OR CDC1="CSP-3" OR CDC1="csp-3"
              THEN IDC2 = 2 : GOTO 2560
2550         GOTO 2450
2560 '       case idc2 of
2570         ON IDC2 GOTO 2580,2610
2580 '         1 : 100% Testing
2590                 RDC1 = 0 : IDC3 = 0 : IDC4 = 0
2600               GOTO 2680
2610 '         2 : CSP  Testing
2620 '             get:
2630                 INPUT "  AOQL ";RDC1
2640                 INPUT "  I ";IDC3
2650                 INPUT "  F ";IDC4
2660 '             endget:
2670               GOTO 2680
2680 '       endcase:
2690         LSET SPLAN$ = CDC1        : LSET DOFN$ = CDC2
2700         LSET SSIZE$ = MKI$(IDC1)  : LSET AOQL$ = MKS$(RDC1)
2710         LSET I$ = MKI$(IDC3)      : LSET F$ = MKI$(IDC4)
2720         LSET SSTATUS$ = "FALSE"
2730         PUT #2,ID
2740 '       get:
2750           PRINT " "
2760           INPUT "  Use Existing or New process curve resource ";CDC1
2770 '       endget:
2780         IDC1 = FALSE
```

```
2790 '      repeat
2800          IF CDC1 = "e" OR CDC1 = "E"
                THEN GOTO 2810 ELSE GOTO 3040
2810 '            get:
2820                PRINT " "
2830                INPUT "  Enter the ID of the curve to be used ";IDC2
2840                IF IDC2 < 1 OR IDC2 > IDTOTAL  THEN GOTO 2810
2850                INPUT "  Process curve filename ";CDC2
2860 '            endget:
2870              GET #1,IDC2
2880              OPEN PCFN$ AS #3 LEN=9
2890              FIELD #3, 6 AS PNO$, 3 AS TNO$
2900                FOR IDCC3 = 1 TO CVI(PSIZE$)
2910                  GET #3,IDCC3 : RNRES%(IDCC3,PNO%) = CVI(PNO$)
2920                  RNRES%(IDCC3,TNO%) = CVI(TNO$) : PUT #3,IDCC3
2930                NEXT IDCC3
2940              CLOSE #3
2950              LSET PCFN$ = CDC2 : PUT #1,ID : GET #1,ID
2960              OPEN PCFN$ AS #3 LEN=9
2970              FIELD #3, 6 AS PNO$, 3 AS TNO$
2980                FOR IDCC3 = 1 TO CVI(PSIZE$)
2990                  LSET PNO$ = MKI$(RNRES%(IDCC3,PNO%))
3000                  LSET TNO$ = MKI$(RNRES%(IDCC3,TNO%))
3010                  PUT #3,IDCC3
3020                NEXT IDCC3
3030              CLOSE #3 : IDC1 = TRUE : GOTO 3200
3040 '          else
3050              IF CDC1 = "n" OR CDC1 = "N"
                    THEN GOTO 3060 ELSE GOTO 3190
3060 '                put:
3070                    PRINT " "
3080                    PRINT "  Supply the following data : "
3090 '                endput:
3100 '                get:
3110                    INPUT "  Process curve ";CDC2
3120                    INPUT "  Process curve filename ";CDC3
3130                    INPUT "  Prob ";RDC
3140 '                endget:
3150                  LSET PROCC$ = CDC2       : LSET PCFN$ = CDC3
3160                  LSET PROB$ = MKS$(RDC1) : PUT #1,ID
3170 '                call CALC_P_C
3180                  GOSUB 3270 : IDC1 = TRUE
3190 '              endif:
3200 '        endif:
3210 '      until (IDC1 = TRUE)
3220        IF IDC1 = FALSE THEN GOTO 2740
3230 '  endif:
3240 'end:
3250  RETURN
3260 REM ##############################################################################
3270 REM CALC_P_C module
3280 'begin:
3290    OPEN PCFN$ AS #3 LEN=9
3300    FIELD #3, 6 AS PNO$, 3 AS TNO$
3310    GET #1,ID : GET #2,ID
3320    CDCC1 = PROCC$
3330 '  case CDCC1 of
```

```
3340    IF CDCC1 = "SELF     " OR CDCC1 = "self     " THEN IDCC1 = 1
3350    IF CDCC1 = "TYPE-A   " OR CDCC1 = "type-a   " THEN IDCC1 = 2
3360    IF CDCC1 = "TYPE-B   " OR CDCC1 = "type-b   " THEN IDCC1 = 3
3370    ON IDCC1 GOTO 3380,3650,3870
3380 '    SELF   : Manual input
3390            NODEF% = 0 : IDCC2 = FALSE
3400 '          repeat
3410 '            get:
3420                PRINT " "
3430                INPUT "  Enter the number of the product to fail ";
                      IDC3
3440 '            endget:
3450              IF IDCC3 > CVI(SSIZE$) THEN PRINT "  Too large "
                    : GOTO 3410
3460              NODEF% = NODEF%+1
3470              RNRES%(NODEF%,PNO%) = IDCC3
3480              RNRES%(NODEF%,TNO%) = INT(RND*9)
3490              IF RNRES%(NODEF%,TNO%) = 0 OR RNRES%(NODEF%,TNO%) = 9
                    THEN GOTO 3480
3500 '            get:
3510                INPUT "  Last number (Y/N) ";CDCC2
3520 '            endget:
3530              IF CDCC2 = "Y" OR CDCC2 = "y"
                    THEN GOTO 3540 ELSE GOTO 3550
3540                  IDCC2 = TRUE : GOTO 3590
3550 '              else
3560 '                put:
3570                    PRINT "  ";N  ...."   ils already "
3580 '                endput:
3590 '              endif:
3600 '          until (IDCC2 = TRUE)
3610            IF IDCC2 = FALSE THEN GOTO 3400
3620            LSET PSIZE$ = MKI$(NODEF%)
3630            LSET PROB$ = MKS$(NODEF%/CVI(SSIZE$))
3640            PUT #1,ID : GOTO 4280
3650 '    TYPE-A : Binomial Distribution
3660            NODEF% = INT(CVI(SSIZE$)*CVS(PROB$))
3670            LSET PSIZE$ = MKI$(NODEF%) : PUT #1,ID
3680 '          Generate random numbers
3690            FOR IDCC2 = 1 TO NODEF%
3700              RNRES%(IDCC2,PNO%) = INT(RND*CVI(SSIZE$))
3710              IF RNRES%(IDCC2,PNO%) = 0 THEN GOTO 3700
3720              RNRES%(IDCC2,TNO%) = INT(RND*9)
3730              IF RNRES%(IDCC2,TNO%) = 0 OR RNRES%(IDCC2,TNO%) = 9
                    THEN GOTO 3720
3740            NEXT IDCC2
3750 '          Check for repeated PNO's
3760            IF NODEF% = 1 THEN GOTO 4380
3770            FOR IDCC3 = 1 TO NODEF%
3780              FOR IDCC4 = (IDCC3+1) TO NODEF%
3790                IF RNRES%(IDCC3,PNO%) = RNRES%(IDCC4,PNO%)
                      THEN GOTO 3800 ELSE GOTO 3830
3800                    RNRES%(IDCC4,PNO%) = INT(RND*CVI(SSIZE$))
3810                    IF RNRES%(IDCC4,PNO%) = 0 THEN GOTO 3800
3820                    GOTO 3770
3830 '              endif:
3840              NEXT IDCC4
```

```
3850            NEXT IDCC3
3860            GOTO 4280
3870 '      TYPE-B : Horsnell Distribution
3880            NODEF% = INT(CVI(SSIZE$)*CVS(PROB$))
3890 '          Generate random numbers
3900            FOR IDCC2 = 1 TO NODEF%
3910              RNRES%(IDCC2,PNO%) = INT(RND*CVI(SSIZE$))
3920              IF RNRES%(IDCC2,PNO%) = 0 THEN GOTO 3910
3930              RNRES%(IDCC2,TNO%) = INT(RND*9)
3940              IF RNRES%(IDCC2,TNO%) = 0 OR RNRES%(IDCC2,TNO%) = 9
                    THEN GOTO 3930
3950            NEXT IDCC2
3960 '          Check for repeated PNO's
3970            IF NODEF% = 1 THEN GOTO 4070
3980            FOR IDCC3 = 1 TO NODEF%
3990              FOR IDCC4 = (IDCC3+1) TO NODEF%
4000                IF RNRES%(IDCC3,PNO%) = RNRES%(IDCC4,PNO%)
                      THEN GOTO 4010 ELSE GOTO 4040
4010                    RNRES%(IDCC4,PNO%) = INT(RND*CVI(SSIZE$))
4020                    IF RNRES%(IDCC4,PNO%) = 0 THEN GOTO 4010
4030                    GOTO 3980
4040 '              endif:
4050              NEXT IDCC4
4060            NEXT IDCC3
4070 '          get:
4080              INPUT "  Number of spotty products (prop. of SSIZE) ";
                    RDCC1
4090              INPUT "  Probability of spotty products ";RDCC2
4100 '          endget:
4110            TESTNO% = INT(RND*9)
4120            IF TESTNO% = 0 OR TESTNO% = 9 THEN GOTO 4110
4130            IDCC2 = INT(RDCC1*CVI(SSIZE$)) 'No of spotty products
4140            IDCC3 = IDCC2/RDCC2            'Range of spotty products
4150            IDCC4 = INT(RND*CVI(SSIZE$))   'Spotty products start
4160            IF IDCC4 = 0 OR IDCC4 > (CVI(SSIZE$)-IDCC3)
                  THEN GOTO 4150
4170            FOR IDCC5 = 1 TO IDCC2
4180              IDCC6 = IDCC4+INT(RND*IDCC3)
4190              FOR IDCC7 = 1 TO NODEF%
4200                IF IDCC6 = RNRES%(IDCC7,PNO%) THEN GOTO 4250
4210              NEXT IDCC7
4220              NODEF%=NODEF%+1
4230              RNRES%(NODEF%,PNO%) = IDCC6
4240              RNRES%(NODEF%,TNO%) = TESTNO%
4250            NEXT IDCC5
4260            LSET PSIZE$ = MKI$(NODEF%) : PUT #1,ID
4270            GOTO 4280
4280 '  endcase:
4290 '  Sort the array
4300    FOR IDCC2 = 1 TO NODEF%
4310      FOR IDCC3 = (IDCC2+1) TO NODEF%
4320        IF RNRES%(IDCC2,PNO%) > RNRES%(IDCC3,PNO%)
              THEN GOTO 4330 ELSE GOTO 4350
4330              SWAP RNRES%(IDCC2,PNO%),RNRES%(IDCC3,PNO%)
4340              SWAP RNRES%(IDCC2,TNO%),RNRES%(IDCC3,TNO%)
4350 '        endif:
4360      NEXT IDCC3
```

```
4370     NEXT IDCC2
4380     FOR IDCC2 = 1 TO NODEF%
4390       LSET PNO$ = MKI$(RNRES%(IDCC2,PNO%))
4400       LSET TNO$ = MKI$(RNRES%(IDCC2,TNO%))
4410       PUT #3,IDCC2
4420     NEXT IDCC2
4430     CLOSE #3
4440 'end:
4450   RETURN
4460 REM End of CALC_P_C Module
4470 REM ##########################################################################
4480 REM End of Data Input Process C ( D_I_PRO_C )
4490 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
4500 REM Data Input Process D ( D_I_PRO_D )
4510 'begin:
4520     IF IDTOTAL = 0
         THEN GOTO 4530 ELSE GOTO 4580
4530 '     put:
4540         PRINT " "
4550         PRINT "  There are no records available "
4560 '     endput:
4570       GOTO 4910
4580 '   else
4590 '     get:
4600         PRINT " "
4610         INPUT "  Input the ID of the record to be erased ";ID
4620 '     endget:
4630       IF ID < 1 OR ID > IDTOTAL THEN GOTO 4590
4640       IDD1 = ID : IDD2 = FALSE
4650       GET #1,ID : KILL PCFN$
4660       GET #2,ID : IF SSTATUS$ <> "FALSE" THEN KILL DOFN$
4670 '     repeat
4680         IF ID = IDTOTAL
             THEN GOTO 4690 ELSE GOTO 4800
4690 '           Place blanks in the required record
4700             LSET SFLAN$ = " "       : LSET I$ = MKI$(0)
4710             LSET DOFN$ = " "        : LSET F$ = MKI$(0)
4720             LSET SSIZE$ = MKI$(0) : LSET SSTATUS$ = "FALSE"
4730             LSET AOQL$ = MKS$(0)   : PUT #2,ID
4740             LSET PROCC$ = " "       : LSET PROB$ = MKS$(0)
4750             LSET PCFN$ = " "        : LSET PSIZE$ = MKI$(0)
4760             PUT #1,ID
4770             LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKI$(0)
4780             PUT #5,ID
4790             IDTOTAL = IDTOTAL-1 : IDD2 = TRUE : GOTO 4850
4800 '           else
4810               IDD3 = ID : ID = ID+1
4820               GET #2,ID : PUT #2,IDD3
4830               GET #1,ID : PUT #1,IDD3
4840               GET #5,ID : PUT #5,IDD3
4850 '           endif:
4860 '     until (IDD2 = TRUE):
4870       IF IDD2 = FALSE THEN GOTO 4670
4880 '     put:
4890         PRINT " " : PRINT "  Record ";IDD1;" has been erased "
4900 '     endput:
4910 '   endif:
```

```
4920 'end:
4930  RETURN
4940 REM End of Data Input Process D ( D_I_PRO_D )
4950 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
4960 REM Data Input Process E ( D_I_PRO_E )
4970 'begin:
4980     PUT #1,ID : CLOSE #1
4990     PUT #2,ID : CLOSE #2
5000     LSET IDTOTAL$ = MKI$(IDTOTAL)
5010     PUT #5,1  : CLOSE #5
5020     OPEN "P_C_RES" AS #1 LEN=30
5030     FIELD #1, 8 AS PROCC$ , 8 AS PROB$ , 10 AS PCFN$ , 4 AS PSIZE$
5040     OPEN "D_I_RES" AS #2 LEN=43
5050     FIELD #2, 6 AS SPLAN$ , 10 AS DOFN$ , 6 AS SSIZE$ , 8 AS AOQL$ ,
                  4 AS I$ , 4 AS F$ , 5 AS SSTATUS$
5060     OPEN "initial" AS #5 LEN=20
5070     FIELD #5, 4 AS IDTOTAL$, 12 AS RUNTIME$, 4 AS DEFNUMBER$
5080     FLAG1% = TRUE
5090 'end:
5100  RETURN
5110 REM End of Data Input Process E ( D_I_PRO_E )
5120 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
5130 REM End of Data Input Process ( D_I_PRO )
5200 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5210 REM Sampling Process ( SAMPLING_PROCESS )
5220 'begin:
5230 '   put:
5240       PRINT " "
5250       PRINT "  SAMPLING PROCESS "
5260       PRINT " "
5270       PRINT "  1 : Run all test processes "
5280       PRINT "  2 : Run an individual test process "
5290       PRINT "  3 : Quit "
5300       PRINT "  Select the required option and press <enter> "
5310       PRINT " "
5320 '   endput:
5330 '   get:
5340       INPUT IS1
5350 '   endget:
5360     IF IS1 < 1 OR IS1 > 3 THEN GOTO 5330
5370     IF IS1 = 3 THEN GOTO 5420
5380 '   get:
5390       INPUT "  Save results (Y/N) " ; SAVER$
5400 '   endget:
5410     IF SAVER$ = "Y" OR SAVER$ = "y" OR SAVER$ = "N" OR SAVER$ = "n"
           THEN GOTO 5420 ELSE GOTO 5380
5420 '   case IS1 of
5430     ON IS1 GOTO 5440,5620,5810
5440 '     1 : *all tests*
5450         FOR ID = 1 TO IDTOTAL
5460           GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5470           LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
                 : PUT #5,ID
5480 '         initialise R_N_RES%
5490           OPEN PCFN$ AS #3 LEN=9
5500           FIELD #3, 6 AS PNO$, 3 AS TNO$
5510             IDC3 = CVI(PSIZE$)
```

```
5520              FOR IDC4 = 1 TO IDC3
5530                GET #3,IDC4
5540                RNRES%(IDC4,PNO%) = CVI(PNO$)
5550                RNRES%(IDC4,TNO%) = CVI(TNO$)
5560              NEXT IDC4
5570            CLOSE #3
5580 '          call SAMPLING PLANS
5590            GOSUB 5850
5600          NEXT ID
5610          GOTO 5820
5620 '      2 : *individual test*
5630 '        get:
5640            INPUT "  Enter the ID of the record ";ID
5650 '        endget:
5660          IF ID < 1 OR ID > IDTOTAL THEN GOTO 5630
5670            GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5680            LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
                  : PUT #5,ID
5690 '          initialise R_N_RES%
5700            OPEN PCFN$ AS #3 LEN=9
5710            FIELD #3, 6 AS PNO$, 3 AS TNO$
5720              IDC3 = CVI(PSIZE$)
5730              FOR IDC4 = 1 TO IDC3
5740                GET #3,IDC4
5750                RNRES%(IDC4,PNO%) = CVI(PNO$)
5760                RNRES%(IDC4,TNO%) = CVI(TNO$)
5770              NEXT IDC4
5780              CLOSE #3
5790 '          call SAMPLING PLANS
5800            GOSUB 5850 : GOTO 5820
5810 '      3 : *quit*
5820 '      endcase:
5830 'end:
5840 RETURN
5850 'Subroutine SAMPLING PLAN
5860 'begin:
5870    OPEN DOFN$ AS #4 LEN=39
5880    FIELD #4, 5 AS PFAIL$, 5 AS PSTATUS$, 3 AS PTIME$, 7 AS OUTCOME$
        , 6 AS TTYPE$, 4 AS COUNT$, 5 AS TSTATUS$, 2 AS TNUM$, 2 AS TTIME$
5890    VAR% = 1 : CNT% = 1 : REF% = 1 : FLAG1% = FALSE
5900    CS1 = SPLAN$ : IS3 = CVI(SSIZE$)
5910    IF CS1="100%  " THEN IS2=1 : GOTO 5950
5920    IF CS1="CSP-1 " OR CS1="csp-1 "
          THEN IS2=2 : SAMPLE%=1 : GOTO 5950
5930    IF CS1="CSP-2 " OR CS1="csp-2 "
          THEN IS2=2 : SAMPLE%=2 : GOTO 5950
5940    IF CS1="CSP-3 " OR CS1="csp-3 "
          THEN IS2=2 : SAMPLE%=3 : GOTO 5950
5950 '  case is2 of
5960    ON IS2 GOTO 5970,6100
5970 '    1 : *100% Sampling*
5980        FOR PRODUCTN% = 1 TO IS3
5990 '       call ALGORITHM_A PROCESS ( ALGM_A_PRO )
6000          GOSUB 6290
6010 '       call TEST RESULT PROCESS ( T_RES_PRO )
6020          GOSUB 9490
6030 '       put:
```

```
6040             PRINT "  Record "; USING "##";ID;
6050             PRINT " : Completed "; USING "#####";PRODUCTNO%;
6060             PRINT " of "; USING "#####";IS3
6070 '         endput:
6080          NEXT PRODUCTNO%
6090          GOTO 6230
6100 '      2 : *CSP Sampling*
6110          FLAG2% = FALSE
6120          FOR PRODUCTNO% = 1 TO IS3
6130 '         call ALGORTHM_B PROCESS ( ALGM_B_PRO )
6140            GOSUB 6590
6150 '         call TEST RESULT PROCESS ( T_RES_PRO )
6160            GOSUB 9490
6170 '         put:
6180             PRINT "  Record "; USING "##";ID;
6190             PRINT " : Completed "; USING "#####";PRODUCTNO%;
6200             PRINT " of "; USING "#####";IS3
6210 '         endput:
6220          NEXT PRODUCTNO%
6230 '    endcase:
6240      LSET SSTATUS$ = "TRUE" : PUT #2,ID : CLOSE #4
6250 'end:
6260  RETURN
6270 REM End of Sampling Plans
6280 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
6290 REM Algorithm_A ( 100% Testing )
6300 'begin:
6310      FLAG1% = FALSE
6320      FOR TESTNO% = 1 TO MAXTEST%
6330        INTERRES$(TESTNO%,TTYPE%) = "FULL"
6340        INTERRES%(TESTNO%,COUNT%) = 0
6350        INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6360        INTERRES%(TESTNO%,TIME%) = 0
6370 '     call TEST_FAIL
6380        GOSUB 8870
6390 '     call INTER_UNIT
6400        GOSUB 9340
6410        IF STATUS% = TRUE
              THEN GOTO 6420 ELSE GOTO 6450
6420            INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6430            INTERRES%(TESTNO%,TIME%) = PASSTIME%
6440            GOTO 6530
6450 '       else
6460            IF STATUS% = FALSE
                  THEN GOTO 6470 ELSE GOTO 6500
6470                INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6480                INTERRES%(TESTNO%,TIME%) = FAILTIME%
6490                GOTO 6520
6500 '           else
6510                INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
6520 '         endif:
6530 '     endif:
6540      NEXT TESTNO%
6550 'end:
6560  RETURN
6570 REM End of Algorithm_A ( 100% Testing )
6580 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
6590 REM Algorithm_B ( CSP Sampling )
6600 'begin:
6610    FLAG1% = FALSE
6620    IF FLAG2% = FALSE
          THEN GOTO 6630 ELSE GOTO 6880
6630 '      *Initialise*
6640        FOR TESTNO% = 1 TO MAXTEST%
6650          INTERRES$(TESTNO%,TTYPE%) = "FULL"
6660          INTERRES%(TESTNO%,COUNT%) = CVI(I$)
6670          INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6680          INTERRES%(TESTNO%,TIME%) = 0
6690 '        call TEST_FAIL
6700          GOSUB 8870
6710 '        call INTER_UNIT
6720          GOSUB 9340
6730          IF STATUS% = TRUE
                THEN GOTO 6740 ELSE GOTO 6770
6740              INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6750              INTERRES%(TESTNO%,TIME%) = PASSTIME%
6760              GOTO 6850
6770 '          else
6780              IF STATUS% = FALSE
                    THEN GOTO 6790 ELSE GOTO 6820
6790                  INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6800                  INTERRES%(TESTNO%,TIME%) = FAILTIME%
6810                  GOTO 6840
6820 '                else
6830                  INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
6840 '              endif:
6850 '        endif:
6860        NEXT TESTNO%
6870        FLAG2% = TRUE : GOTO 8810
6880 '    else
6890        FOR TESTNO% = 1 TO MAXTEST%
6900 '        call TEST_FAIL
6910          GOSUB 8870
6920 '        case sample% of
6930          ON SAMPLE% GOTO 6940,7260,7820
6940 '          1 : *CSP-1*
6950              IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                    THEN GOTO 6960 ELSE GOTO 7160
6960                  IF INTERRES%(TESTNO%,COUNT%) = 1
                        THEN GOTO 6970 ELSE GOTO 7070
6970                      IF INTERRES$(TESTNO%,TTYPE%) = "FULL"
                            THEN GOTO 6980 ELSE GOTO 7020
6980                          INTERRES$(TESTNO%,TTYPE%) = "PART"
6990                          INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7000                          INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7010                          GOTO 7050
7020 '                      else:
7030                          INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7040                          INTERRES%(TESTNO%,COUNT%) = CVI(F$)
7050 '                    endif:
7060                      GOTO 7150
7070 '                  else
7080                      INTERRES$(TESTNO%,TTYPE%) = "FULL"
```

```
                    THEN GOTO 7090 ELSE GOTO 7110
7090                  INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7100                  GOTO 7130
7110 '              else
7120                  INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7130 '            endif:
7140              INTERRES%(TESTNO%,COUNT%) =
                          INTERRES%(TESTNO%,COUNT%)-1
7150 '        endif:
7160 '      endif:
7170        IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
              THEN GOTO 7180 ELSE GOTO 7210
7180            INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7190            INTERRES$(TESTNO%,TTYPE%) = "FULL"
7200            INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7210 '      endif:
7220        IF INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
              THEN GOTO 7230 ELSE GOTO 7240
7230            INTERRES%(TESTNO%,COUNT%) =
                          INTERRES%(TESTNO%,COUNT%)-1
7240 '      endif:
7250        GOTO 8580
7260 '    2 : *CSP-2*
7270        IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
              THEN GOTO 7280 ELSE GOTO 7580
7280            INTERRES$(TESTNO%,TTYPE%) = "FULL"
                  THEN GOTO 7290 ELSE GOTO 7390
7290                IF INTERRES%(TESTNO%,COUNT%) = 1
                      THEN GOTO 7300 ELSE GOTO 7340
7300                    INTERRES$(TESTNO%,TTYPE%) = "PART"
7310                    INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7320                    INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7330                    INTERRES%(TESTNO%,FLG1%) = FALSE
                            : GOTO 7370
7340 '                  else
7350                    INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7360                    INTERRES%(TESTNO%,COUNT%) =
                                INTERRES%(TESTNO%,COUNT%)-1
7370 '                  endif:
7380                    GOTO 7560
7390 '                else
7400                  IF INTERRES%(TESTNO%,COUNT%) = 1

                        THEN GOTO 7410 ELSE GOTO 7520
7410                      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7420                      INTERRES%(TESTNO%,COUNT%) = CVI(F$)
7430                      IF INTERRES%(TESTNO%,FLG1%) = TRUE
                            THEN GOTO 7440 ELSE GOTO 7500
7440                          IF INTERRES%(TESTNO%,COUNT%) = 1
                                THEN GOTO 7450 ELSE GOTO 7470
7450                              INTERRES%(TESTNO%,FLG1%) = FALSE
7460                              GOTO 7490
7470 '                          else
7480                              INTERRES%(TESTNO%,COUNT%) =
                                      INTERRES%(TESTNO%,COUNT%)-1
7490 '                        endif:
7500 '                    endif:
7510                      GOTO 7550
```

```
7520 '                    else
7530                         INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7540                         INTERRES%(TESTNO%,COUNT%) =
                                      INTERRES%(TESTNO%,COUNT%)-1
7550 '                   endif:
7560 '                 endif:
7570                   GOTO 7800
7580 '               else
7590                   IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
                         THEN GOTO 7600 ELSE GOTO 7770
7600                       IF INTERRES$(TESTNO%,TTYPE%) = "PART"
                             THEN GOTO 7610 ELSE GOTO 7720
7610                           IF INTERRES%(TESTNO%,FLG1%) = 1
                                 THEN GOTO 7620 ELSE GOTO 7660
7620                               INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7630                               INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7640                               INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7650                               INTERRES%(TESTNO%,FLG1%) = TRUE
                                     : GOTO 7700
7660 '                           else
7670                               INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7680                               INTERRES$(TESTNO%,TTYPE%) = "FULL"
7690                               INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7700 '                         endif:
7710                           GOTO 7750
7720 '                     else
7730                         INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7740                         INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7750 '                   endif:
7760                     GOTO 7790
7770 '                 else       \
7780                     INTERRES%(TESTNO%,COUNT%) =
                                  INTERRES%(TESTNO%,COUNT%)-1
7790 '             endif:
7800 '           endif:
7810             GOTO 8580
7820 '         3 : *CSP-3*
7830             IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                   THEN GOTO 7840 ELSE GOTO 8300
7840                 IF INTERRES$(TESTNO%,TTYPE%) = "PART"
                       THEN GOTO 7850 ELSE GOTO 8160
7850                   IF INTERRES%(TESTNO%,FLG2%) = FALSE
                         THEN GOTO 7860 ELSE GOTO 7950
7860                     IF INTERRES%(TESTNO%,COUNT%) = 1
                           THEN GOTO 7870 ELSE GOTO 7900
7870                       INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7880                       INTERRES%(TESTNO%,COUNT%) = CVI(F$)
7890                       GOTO 7930
7900 '                   else
7910                       INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7920                       INTERRES%(TESTNO%,COUNT%) =
                                      INTERRES%(TESTNO%,COUNT%)-1
7930 '                 endif:
7940                   GOTO 8040
7950 '               else
7960                   IF INTERRES%(TESTNO%,COUNT%) = 1
```

```
                    THEN GOTO 7970 ELSE GOTO 8000
7970                  INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7980                  INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7990                  INTERRES%(TESTNO%,FLG2%) = FALSE
                        : GOTO 8030
8000 '                else
8010                    INTERRES$(TESTNO%,TSTATUS%) = "RUN"
8020                    INTERRES%(TESTNO%,COUNT%) =
                                INTERRES%(TESTNO%,COUNT%)-1
8030 '                endif:
8040 '              endif:
8050                IF INTERRES$(TESTNO%,TSTATUS%) = "RUN"
                      THEN GOTO 8060 ELSE GOTO 8140
8060                    IF INTERRES%(TESTNO%,FLG1%) = 1
                          THEN GOTO 8070 ELSE GOTO 8130
8070                      IF INTERRES%(TESTNO%,COUNT%) = 1
                            THEN GOTO 8080 ELSE GOTO 8100
8080                          IF INTERRES%(TESTNO%,FLG2%) = FALSE
                                THEN
                                  INTERRES%(TESTNO%,FLG1%) = FALSE
8090                            GOTO 8120
8100 '                        else
8110                            INTERRES%(TESTNO%,COUNT%) =
                                      INTERRES%(TESTNO%,COUNT%)-1
8120 '                        endif:
8130 '                    endif:
8140 '              endif:
8150                GOTO 8280
8160 '            else
8170                IF INTERRES%(TESTNO%,COUNT%) = 1
                      THEN GOTO 8180 ELSE GOTO 8240
8180                    INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
8190                    INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
8200                    INTERRES$(TESTNO%,TTYPE%) = "PART"
8210                    INTERRES%(TESTNO%,FLG1%) = FALSE
8220                    INTERRES%(TESTNO%,FLG2%) = FALSE
8230                    GOTO 8270
8240 '                  else
8250                    INTERRES$(TESTNO%,TSTATUS%) = "RUN"
8260                    INTERRES%(TESTNO%,COUNT%) =
                                INTERRES%(TESTNO%,COUNT%)-1
8270 '                endif:
8280 '            endif:
8290              GOTO 8560
8300 '          else
8310              IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
                    THEN GOTO 8320 ELSE GOTO 8530
8320                  IF INTERRES$(TESTNO%,TTYPE%) = "PART"
                        THEN GOTO 8330 ELSE GOTO 8470
8330                    IF INTERRES%(TESTNO%,FLG1%) = FALSE
                          THEN GOTO 8340 ELSE GOTO 8410
8340                      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
8350                      INTERRES$(TESTNO%,TTYPE%) = "PART"
8360                      INTERRES%(TESTNO%,COUNT%) = CVI(I$)
8370                      INTERRES%(TESTNO%,COUNT%) = 4
8380                      INTERRES%(TESTNO%,FLG1%) = TRUE
8390                      INTERRES%(TESTNO%,FLG2%) = TRUE
```

```
8400                         GOTO 8450
8410 '                     else
8420                         INTERRES$(TESTNO%,TSTATUS%) = "RUN"
8430                         INTERRES%(TESTNO%,COUNT%) = CVI(I$)
8440                         INTERRES$(TESTNO%,TTYPE%) = "FULL"
8450 '                   endif:
8460                     GOTO 8510
8470 '                 else
8480                     INTERRES$(TESTNO%,TSTATUS%) = "RUN"
8490                     INTERRES%(TESTNO%,COUNT%) = CVI(I$)
8500                     INTERRES$(TESTNO%,TTYPE%) = "FULL"
8510 '               endif:
8520                 GOTO 8550
8530 '             else
8540                 INTERRES%(TESTNO%,COUNT%) =
                              INTERRES%(TESTNO%,COUNT%)-1
8550 '           endif:
8560 '         endif:
8570           GOTO 8580
8580 '       endcase:
8590         IF INTERRES$(TESTNO%,TSTATUS%) = "RUN"
               THEN GOTO 8600 ELSE GOTO 8760
8600 '           call INTER_UNIT
8610             GOSUB 9340
8620             IF STATUS% = TRUE
                   THEN GOTO 8630 ELSE GOTO 8660
8630                 INTERRES$(TESTNO%,OUTCOME%) = "PASS"
8640                 INTERRES%(TESTNO%,TIME%) = PASSTIME%
8650                 GOTO 8740
8660 '             else
8670                 IF STATUS% = TRUE
                       THEN GOTO 8680 ELSE GOTO 8710
8680                     INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
8690                     INTERRES%(TESTNO%,TIME%) = FAILTIME%
8700                     GOTO 8730
8710 '                 else
8720                     INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
8730 '               endif:
8740 '           endif:
8750             GOTO 8790
8760 '         else
8770             INTERRES$(TESTNO%,OUTCOME%) = "PASS"
8780             INTERRES%(TESTNO%,TIME%) = NOTIME%
8790 '       endif:
8800       NEXT TESTNO%
8810 '  endif:
8820 'end:
8830  RETURN
8840 REM End of Algorithm_B ( CSP and MLP Sampling )
8850 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
8860 REM ##############################################################
8870 REM Test Fail Process ( TEST_FAIL )
8880 'begin:
8890    IF TESTNO% <> 8
          THEN GOTO 8900 ELSE GOTO 9190
8900        IF FLAG1% = FALSE
```

```
                THEN GOTO 8910 ELSE GOTO 9170
8910              NODEF% = CVI(PSIZE$)
8920              IF VAR% <= NODEF%
                    THEN GOTO 8930 ELSE GOTO 9100
8930                  IF PRODUCTNO% < RNRES%(VAR%,PNO%)
                        THEN GOTO 8940 ELSE GOTO 8950
8940                      TEST% = 0 : KEYNO% = &HFF : GOTO 9080
8950 '                  else
8960                      TEST% = RNRES%(VAR%,TNO%)
8970 '                    * Keyno does not correspond to test *
8980                      ON TEST% GOTO 8990,9000,9010,9020,9030,9040,
                                        9050,9060
8990                        KEYNO% = &HFE : GOTO 9070
9000                        KEYNO% = &HFD : GOTO 9070
9010                        KEYNO% = &HDF : GOTO 9070
9020                        KEYNO% = &HFB : GOTO 9070
9030                        KEYNO% = &HF7 : GOTO 9070
9040                        KEYNO% = &HEF : GOTO 9070
9050                        KEYNO% = &H7F : GOTO 9070
9060                        KEYNO% = &HFF
9070                      VAR% = VAR%+1
9080 '                  endif:
9090                    GOTO 9120
9100 '                else
9110                    TEST% = 0 : KEYNO% = &HFF
9120 '              endif:
9130 '              put:
9140                  OUT TFAIL%,KEYNO%  ' Interface Unit
9150 '              endput:
9160                FLAG1% = TRUE
9170 '          endif:
9180            GOTO 9290
9190 '        else
9200            IF TEST% = 8
                 THEN GOTO 9210 ELSE GOTO 9220
9210                KEYNO% = &HFC : GOTO 9240
9220 '           else
9230                KEYNO% = &HFF
9240 '          endif:
9250 '          put:
9260              OUT TFAIL%,KEYNO%  ' Interface Unit
9270 '          endput:
9280            FLAG1% = FALSE
9290 '      endif:
9300 'end:
9310  RETURN
9320 REM End of Test Fail Process ( TEST_FAIL )
9330 REM ##################################################################
9340 REM Interface Unit Process ( INTER_UNIT )
9350 'begin:
9360 OUT MDA%,TESTDAT%(TESTNO%)
9370 OUT SETRE%,&H0
9380 IN% = INP(RDRC%) AND &H1
9390 IF IN% <> &H1 THEN GOTO 9380
9400 IN% = INP(RDRC%) AND &H1
9410 IF IN% <> &H0 THEN GOTO 9400
9420 STATUS% = INP(RDDAT%)
```

```
9430 OUT SETRE%,&H0
9440 'end:
9450  RETURN
9460 REM End of Interface Unit Process ( INTER_UNIT )
9470 REM ##########################################################################
9480 REM End of Algorithm Process
9490 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
9500 REM Test Result Process ( TEST_RES_PRO )
9510 'begin:
9520    IF SAVER$ = "Y" OR SAVER$ = "y" THEN GOTO 9530 ELSE GOTO 9620
9530    TESTNO% = 1
9540    DORES$(REF%,PSTATUS%) = "PASS"
9550    DORES$(REF%,TTYPE%) = INTERRES$(TESTNO%,TTYPE%)
9560    DORES%(RE.%,COUNT%) = INTERRES%(TESTNO%,COUNT%)
9570    DORES$(REF%,TSTATUS%) = INTERRES$(TESTNO%,TSTATUS%)
9580    DORES$(REF%,OUTCOME%) = INTERRES$(TESTNO%,OUTCOME%)
9590    DORES%(REF%,TIME%) = INTERRES%(TESTNO%,TIME%)
9600    DORES%(REF%,PFAIL%) = TEST%
9610    DORES%(REF%,TNUM%) = TESTNO%
9620    TOTALTIME% = 0
9630    FOR TESTNO% = 1 TO MAXTEST%
9640      GET #5,ID
9650      IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
            THEN GOTO 9660 ELSE GOTO 9690
9660          DORES$(REF%,PSTATUS%) = "FAIL"
9670          ITR1 = CVI(DEFNUMBER$)+1
9680          LSET DEFNUMBER$ = MKI$(ITR1) : PUT #5,ID
9690 '    endif:
9700      TOTALTIME% = TOTALTIME%+INTERRES%(TESTNO%,TIME%)
9710    NEXT TESTNO%
9720    DORES%(REF%,PTIME%) = TOTALTIME%
9730    GET #5,ID
9740    ITR3# = CVD(RUNTIME$)+TOTALTIME%
9750    LSET RUNTIME$ = MKD$(ITR3#) : PUT #5,ID
9760    IF SAVER$ = "Y" OR SAVER$ ="y" THEN GOTO 9770 ELSE GOTO 9940
9770    REF% = REF%+1
9780    IF PRODUCTNO% = CNT%*25 OR PRODUCTNO% = CVI(SSIZE$)
          THEN GOTO 9790 ELSE GOTO 9930
9790 '      * Copy Data Output Resource to Random File #4 *
9800        FOR ITR2 = 1 TO (REF%-1)
9810          LSET TTYPE$ = DORES$(ITR2,TTYPE%)
9820          LSET COUNT$ = MKI$(DORES%(ITR2,COUNT%))
9830          LSET TSTATUS$ = DORES$(ITR2,TSTATUS%)
9840          LSET OUTCOME$ =DORES$(ITR2,OUTCOME%)
9850          LSET TTIME$ = MKI$(DORES%(ITR2,TIME%))
9860          LSET TNUM$ = MKI$(DORES%(ITR2,TNUM%))
9870          LSET PFAIL$ = MKI$(DORES%(ITR2,PFAIL%))
9880          LSET PSTATUS$ = DORES$(ITR2,PSTATUS%)
9890          LSET PTIME$ = MKI$(DORES%(ITR2,PTIME%))
9900          PUT #4,((CNT%-1)*25+ITR2)
9910        NEXT ITR2
9920        CNT% = CNT%+1 : REF% = 1
9930 '  endif:
9940 'end:
9950  RETURN
9960 REM End of Test Result Process ( TEST_RES_PRO )
9970 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
9980 REM End of Sampling Process ( SAMPLING_PROCESS )
10000 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10010 REM Data Output Process ( D_O_PROCESS )
10020 'begin:
10030   ON ERROR GOTO 12060
10040   ID04 = FALSE
10050 '  repeat
10060 '   put:
10070       PRINT " "
10080       PRINT "  DATA OUTPUT PROCESS "
10090       PRINT " "
10100       PRINT "  1 : Output a datafile to the printer "
10110       PRINT "  2 : Output a datafile to the screen "
10120       PRINT "  3 : Output a rnresource to the printer "
10130       PRINT "  4 : Output a rnresource to the screen "
10140       PRINT "  5 : Quit "
10150       PRINT "  Select the required option and press <enter> "
10160       PRINT " "
10170 '   endput:
10180 '   get:
10190       INPUT ID01
10200 '   endget:
10210     IF ID01 < 1 OR ID01 > 5 THEN GOTO 10060
10220 '   case ID01 of
10230     ON ID01 GOTO 10240,10830,11420,11700,11980
10240 '     1 : Output a datafile to the printer
10250         IF IDTOTAL = 0
            THEN PRINT "  There are no records available ":GOTO 12000
10260 '       get:
10270           INPUT "  Input the ID of the record ";ID
10280 '       endget:
10290         IF ID < 1 OR ID > IDTOTAL THEN GOTO 10260
10300         GET #1,ID : GET #2,ID : GET #5,ID
10310         IF SSTATUS$ = "FALSE"
            THEN PRINT "  The test must still be run " : GOTO 12000
10320         OPEN DOFN$ AS #4 LEN=39
10330        FIELD #4,5 AS PFAIL$,5 as PSTATUS$,3 AS PTIME$,7 AS OUTCOME$
           ,6 AS TTYPR$,4 AS COUNT$,5 AS TSTATUS$,2 AS TNUM$,2 AS TTIME$
10340 '       * Calculate no of def prod not found *
10350         ID02 = CVI(PSIZE$)-CVI(DEFNUMBER$)
10360 '       * Calculate AOQ *
10370         AOQ! = ID02/CVI(SSIZE$)
10380 '       put:
10390           LPRINT "  ID                    : ";USING "#####";ID
10400           LPRINT "  Sampling plan         : ";USING "&";SPLAN$
10410           LPRINT "  Filename              : ";USING "&";DOFN$
10420           LPRINT "  Sample size           : ";USING "#####";
              CVI(SSIZE$)
10430           LPRINT "  AOQL                  : ";USING "#.#####";
              CVS(AOQL$)
10440           LPRINT "  I                     : ";USING "#####";CVI(I$)
10450           LPRINT "  F                     : ";USING "#####";CVI(F$)
10460           LPRINT "  Process curve         : ";USING "&";PROCC$
10470           LPRINT "  Process size          : ";USING "#####";
              CVI(PSIZE$)
10480           LPRINT "  Prob                  : ";USING "#.#####";
```

```
          CVS(PROB$)
10490     LPRINT " "
10500     LPRINT "  Defective found      : ";USING "#####";
            CVI(DEFNUMBER$)
10510     LPRINT "  Defectives not found : ";USING "#####";IDO2
10520     LPRINT "  AOQ                 : ";USING "#.#####";AOQ!
10530     LPRINT "  Total time for run   : ";USING "#######";
            CVD(RUNTIME$)
10540     LPRINT " ":LPRINT " ":LPRINT " ":LPRINT " ":LPRINT " "
10550 '   endput:
10560       IF SAVER$ = "Y" OR SAVER$ = "y"
              THEN GOTO 10570 ELSE GOTO 10820
10570       INPUT "  List results (Y/N) ";CDO2
10580       IF CDO2 = "y" OR CDO2 = "Y"
              THEN GOTO 10600 ELSE GOTO 10820
10590 '   put:
10600       LPRINT " "
10610       LPRINT "   Prod Prod Prod    Prod Test Test Count Test    ";
10620       LPRINT "Outcome Test "
10630       LPRINT "   No   Fail Status Time No   Type       Status ";
10640       LPRINT "        Time "
10650       LPRINT " "
10660     endput:
10670     FOR IDO3 = 1 TO CVI(SSIZE$)
10680       GET #4,IDO3
10690 '     put:
10700         LPRINT USING "  #####";IDO3;
10710         LPRINT USING "   ##   ";CVI(PFAIL$);
10720         LPRINT USING " \   \  ";PSTATUS$;
10730         LPRINT USING "####   ";CVI(PTIME$);
10740         LPRINT USING "### ";CVI(TNUM$);
10750         LPRINT USING " \   \ ";TTYPE$;
10760         LPRINT USING "####   ";CVI(COUNT$);
10770         LPRINT USING " \    \  ";TSTATUS$;
10780         LPRINT USING " \     \";OUTCOME$;
10790         LPRINT USING "###";CVI(TTIME$)
10800 '     endput:
10810     NEXT IDO3
10820     GOTO 12000
10830 '  2 : Output a datafile to the screen
10840     IF IDTOTAL = 0
            THEN PRINT "  There are no records available ":GOTO 11410
10850 '   get:
10860       INPUT "  Input the ID of the record ";ID
10870 '   endget:
10880     IF ID < 1 OR ID > IDTOTAL THEN GOTO 10850
10890     GET #1,ID : GET #2,ID : GET #5,ID
10900     IF SSTATUS$ = "FALSE"
            THEN PRINT "  The test must still be run " : GOTO 12000
10910     OPEN DCFN$ AS #4 LEN=39
10920    FIELD #4,5 AS PFAIL$,5 AS PSTATUS$,# AS PTIME$,7 AS OUTCOME$
         ,6 AS TTYPE$,4 AS COUNT$,5 AS TSTATUS$,2 AS TNUM$,2 AS TTIME$
10930 '   * Calculate no of def prod not found *
10940     IDO2 = CVI(PSIZE$)-CVI(DEFNUMBER$)
10950 '   * Calculate AOQ *
10960     AOQ! = IDO2/CVI(SSIZE$)
10970 '   put:
```

```
10980         PRINT "  ID                          : ";USING "#####";ID
10990         PRINT "  Sampling plan               : ";USING "&";SPLAN$
11000         PRINT "  Filename                    : ";USING "&";DOFN$
11010         PRINT "  Sample size                 : ";USING "#####";
                CVI(SSIZE$)
11020         PRINT "  AOQL                        : ";USING "#.#####";
                CVS(AOQL$)
11030         PRINT "  I                           : ";USING "#####";CVI(I$)
11040         PRINT "  F                           : ";USING "#####";CVI(F$)
11050         PRINT "  Process curve               : ";USING "&";PROCC$
 060          PRINT "  Process size                : ";USING "#####";
                CVI(PSIZE$)
11070         PRINT "  Prob                        : ";USING "#.#####";
                CVS(PROB$)
11080         PRINT " "
11090         PRINT "  Defective found             : ";USING "#####";
                CVI(DEFNUMBER$)
11100         PRINT "  Defectives not found        : ";USING "#####";IDO2
11110         PRINT "  AOQ                         : ";USING "#.#####";AOQ!
11120         PRINT "  Total time for run          : ";USING "#######";
                CVD(RUNTIME$)
11130         PRINT " "
11140 '     endput:
11150         IF SAVER$ = "Y" OR SAVER$ = "y"
                THEN GOTO 11160 ELSE GOTO 11410
11160         INPUT " List results (Y/N) ";CDO2
11170         IF CDO2 = "y" OR CDO2 = "Y"
                THEN GOTO 11190 ELSE GOTO 11410
11180 '     put:
11190         PRINT " "
11200         PRINT "   Prod  Prod  Prod     Prod  Test  Test  Count ";
11210         PRINT "  Test    Outcome  Test "
11220         PRINT "    No    Fail  Status  Time  No    Type        ";
11230         PRINT "  Status           Time "
11240         PRINT " "
11250 '     endput:
11260         FOR IDO3 = 1 TO CVI(SSIZE$)
11270           GET #4,IDO3
11280 '         put:
11290             PRINT USING "  ##### ";IDO3;
11300             PRINT USING "  ##    ";CVI(PFAIL$);
11310             PRINT USING " \  \   ";PSTATUS$;
11320             PRINT USING "####    ";CVI(PTIME$);
11330             PRINT USING "###  ";CVI(TNUM$);
11340             PRINT USING " \  \   ";TTYPE$;
11350             PRINT USING "####    ";CVI(COUNT$);
11360             PRINT USING " \    \  ";TSTATUS$;
11370             PRINT USING "  \    \ ";OUTCOME$;
11380             PRINT USING "###";CVI(TTIME$)
11390 '         endput:
11400         NEXT IDO3
11410         GOTO 12000
11420 '   3 : Output a rnresource to the printer
11430         IF IDTOTAL = 0
                THEN PRINT "  There are no records available ":GOTO 12000
11440 '       get:
11450           INPUT "  Input the ID of the record ";ID
```

```
11460 '       endget:
11470         IF ID < 1 OR ID > IDTOTAL THEN GOTO 11440
11480         GET #1,ID
11490         OPEN PCFN$ AS #3 LEN=9
11500         FIELD #3, 6 AS PNO$, 3 AS TNO$
11510 '       put:
11520           LPRINT "  ID                    : ";USING "#####";ID
11530           LPRINT "  Process curve         : ";USING "&";PROCC$
11540           LPRINT "  Filename              : ";USING "&";PCFN$
11550           LPRINT "  Process size          : ";USING "#####";CVI(PSIZE$)
11560           LPRINT "  Prob                  : ";USING "#.#####";
                  CVS(PROB$)
11570           LPRINT "  "
11580           LPRINT "  No   Product No    Test No "
11590           LPRINT "  "
11600 '       endput:
11610         FOR ID02 = 1 TO CVI(PSIZE$)
11620           GET #3,ID02
11630 '         put:
11640             LPRINT USING " ### ";ID02;
11650             LPRINT USING "   #####      ";CVI(PNO$);
11660             LPRINT USING "   ###";CVI(TNO$)
11670 '           endput:
11680         NEXT ID02
11690         GOTO 12000
11700 '     4 : Output a rnresource to the screen
11710         IF IDTOTAL = 0
                THEN PRINT "  There are no records available ":GOTO 12000
11720 '       get:
11730           INPUT "  Input the ID of the record ";ID
11740 '       endget:                    \
11750         IF ID < 1 OR ID > IDTOTAL THEN YTO 11720
11760         GET #1,ID
11770         OPEN PCFN$ AS #3 LEN=9
11780         FIELD #3, 6 AS PNO$, 3 AS TNO$
11790 '       put:
11800           PRINT "  ID                    : ";USING "#####";ID
11810           PRINT "  Process curve         : ";USING "&";PROCC$
11820           PRINT "  Filename              : ";USING "&";PCFN$
11830           PRINT "  Process size          : ";USING "#####";CVI(PSIZE$)
11840           PRINT "  Prob                  : ";USING "#.#####";
                  CVS(PROB$)
11850           PRINT "  "
11860           PRINT "  No   Product No    Test No "
11870           PRINT "  "
11880 '       endput:
11890         FOR ID02 = 1 TO CVI(PSIZE$)
11900         GET #3,ID02
11910 '         put:
11920             PRINT USING " ### ";ID02;
11930             PRINT USING "   #####      ";CVI(PNO$);
11940             PRINT USING "   ###";CVI(TNO$)
11950 '           endput:
11960         NEXT ID02
11970         GOTO 12000
11980 '     5 : QUIT
11990         ID04 = TRUE
```

```
12000 '  endcase:
12010    CLOSE #3 : CLOSE #4
12020 ' until (ID04 = true):
12030   IF ID04 = FALSE THEN GOTO 10050
12040 'end:
12050  RETURN
12060 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
12070 REM Error Routine
12080 'begin:
12090    IF ERR = 27 OR ERR = 24
           THEN GOTO 12100 ELSE GOTO 12110
12100        PRINT "  Check Printer " : RESUME
12110 '  endif:
12120 REM End of Error Routine
12130 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
12140 REM End of Data Output Process
12150 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12160 RESUME
```

G.2 MULTI-LEVEL INSPECTION PLAN SOFTWARE LISTING

```
5140 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5150 REM Sampling Process ( SAMPLING_PROCESS )
5160 'begin:
5170 '  put:
5180       PRINT " "
5190       PRINT "  SAMPLING PROCESS "
5200       PRINT " "
5210       PRINT "  1 : Run all test processes "
5220       PRINT "  2 : Run an individual test process "
5230       PRINT "  3 : Quit "
5240       PRINT "  Select the required option and press <enter> "
5250       PRINT " "
5260 '  endput:
5270 '  get:
5280       INPUT IS1
5290 '  endget:
5300     IF IS1 < 1 OR IS1 > 3 THEN GOTO 5270
5310     IF IS1 = 3 THEN GOTO 5360
5320 '  get:
5330       INPUT "  Save results (Y/N) " ; SAVER$
5340 '  endget:
5350     IF SAVER$ = "Y" OR SAVER$ = "y" OR SAVER$ = "N" OR SAVER$ = "n".
           THEN GOTO 5360 ELSE GOTO 5320
5360 '  case IS1 of
5370     ON IS1 GOTO 5380,5560,5750
5380 '    1 : *all tests*
5390         FOR ID = 1 TO IDTOTAL
5400           GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5410           LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
                 : PUT #5,ID
5420 '         initialise R_N_RES%
5430           OPEN PCFN$ AS #3 LEN=9
5440           FIELD #3, 6 AS PNO$, 3 AS TNO$
5450             IDC3 = CVI(PSIZE$)
5460             FOR IDC4 = 1 TO IDC3
5470               GET #3,IDC4
5480               RNRES%(IDC4,PNO%) = CVI(PNO$)
5490               RNRES%(IDC4,TNO%) = CVI(TNO$)
5500             NEXT IDC4
5510           CLOSE #3
5520 '         call SAMPLING PLANS
5530           GOSUB 5790
5540         NEXT ID
5550         GOTO 5760
5560 '    2 : *individual test*
5570 '      get:
5580           INPUT "  Enter the ID of the record ";ID
5590 '      endget:
5600         IF ID < 1 OR ID > IDTOTAL THEN GOTO 5570
5610           GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5620           LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
                 : PUT #5,ID
5630 '         initialise R_N_RES%
5640           OPEN PCFN$ AS #3 LEN=9
5650           FIELD #3, 6 AS PNO$, 3 AS TNO$
5660             IDC3 = CVI(PSIZE$)
5670             FOR IDC4 = 1 TO IDC3
```

```
5680              GET #3,IDC4
5690              RNRES%(IDC4,PNO%) = CVI(PNO$)
5700              RNRES%(IDC4,TNO%) = CVI(TNO$)
5710            NEXT IDC4
5720            CLOSE #3
5730 '        call SAMPLING PLANS
5740          GOSUB 5790 : GOTO 5760
5750 '      3 : *quit*
5760 '    endcase:
5770 'end:
5780  RETURN
5790 'Subroutine SAMPLING PLAN
5800 'begin:
5810    OPEN DOFN$ AS #4 LEN=39
5820    FIELD #4, 5 AS PFAIL$, 5 AS PSTATUS$, 3 AS PTIME$, 7 AS OUTCOME$
        ,6 AS TTYPE$, 4 AS COUNT$, 5 AS TSTATUS$, 2 AS TNUM$, 2 AS TTIME$
5830    VAR% = 1 : CNT% = 1 : REF% = 1 : FLAG1% = FALSE
5840    CS1 = SPLAN$ : IS3 = CVI(SSIZE$)
5850    IF CS1="100%  " THEN IS2=1 : GOTO 5880
5860    IF CS1="MLP   " OR CS1="mlp   "
          THEN IS2=2 : SAMPLE%=1 : GOTO 5880
5870    IF CS1="MLP-T " OR CS1="mlp-t "
          THEN IS2=2 : SAMPLE%=2 : GOTO 5880
5880 '  case IS2 of
5890    ON IS2 GOTO 5900,6030
5900 '    1 : *100% Sampling*
5910        FOR PRODUCTNO% = 1 TO IS3
5920 '        call ALGORITHM_A PROCESS ( ALGM_A_PRO )
5930          GOSUB 6220
5940 '        call TEST RESULT PROCESS ( T_RES_PRO )
5950          GOSUB 8850
5960 '        put:
5970            PRINT "  Record "; USING "##";ID;
5980            PRINT " : Completed "; USING "#####";PRODUCTNO%;
5990            PRINT " of "; USING "#####";IS3
6000 '        endput:
6010        NEXT PRODUCTNO%
6020        GOTO 6160
6030 '    2 : *CSP Sampling*
6040        FLAG2% = FALSE
6050        FOR PRODUCTNO% = 1 TO IS3
6060 '        call ALGORITHM_B PROCESS ( ALGM_B_PRO )
6070          GOSUB 6520
6080 '        call TEST RESULT PROCESS ( T_RES_PRO )
6090          GOSUB 8850
6100 '        put:
6110            PRINT "  Record "; USING "##";ID;
6120            PRINT " : Completed "; USING "#####";PRODUCTNO%;
6130            PRINT " of "; USING "#####";IS3
6140 '        endput:
6150        NEXT PRODUCTNO%
6160 '  endcase:
6170    LSET@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
6180 'end:
6190  RETURN
6200 REM End of Sampling Plans
6210 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
6220 REM Algorithm_A ( 100% Testing )
6230 'begin:
6240    FLAG1% = FALSE
6250    FOR TESTNO% = 1 TO MAXTEST%
6260      INTERRES$(TESTNO%,TTYPE%) = "FULL"
6270      INTERRES%(TESTNO%,COUNT%) = 0
6280      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6290      INTERRES%(TESTNO%,TIME%) = 0
6300 '    call TEST_FAIL
6310      GOSUB 8230
6320 '    call INTER_UNIT
6330      GOSUB 8700
6340      IF STATUS% = TRUE
            THEN GOTO 6350 ELSE GOTO 6380
6350          INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6360          INTERRES%(TESTNO%,TIME%) = PASSTIME%
6370          GOTO 6460
6380 '      else
6390          IF STATUS% = FALSE
                THEN GOTO 6400 ELSE GOTO 6430
6400              INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6410              INTERRES%(TESTNO%,TIME%) = FAILTIME%
6420              GOTO 6450
6430 '          else
6440              INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
6450 '        endif:
6460 '    endif:
6470    NEXT TESTNO%
6480 'end:
6490  RETURN
6500 REM End of Algorithm_A ( 100% Testing )
6510 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
6520 REM Algorithm_B ( MLP Sampling )
6530 'begin:
6540    FLAG1% = FALSE
6550    IF FLAG2% = FALSE
          THEN GOTO 6560 ELSE GOTO 6810
6560 '      *Initialise*
6570        FOR TESTNO% = 1 TO MAXTEST%
6580          INTERRES$(TESTNO%,TTYPE%) = "FULL"
6590          INTERRES%(TESTNO%,COUNT%) = CVI(I$)
6600          INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6610          INTERRES%(TESTNO%,TIME%) = 0
6620 '        call TEST_FAIL
6630          GOSUB 8230
6640 '        call INTER_UNIT
6650          GOSUB 8700
6660          IF STATUS% = TRUE
                THEN GOTO 6670 ELSE GOTO 6700
6670              INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6680              INTERRES%(TESTNO%,TIME%) = PASSTIME%
6690              GOTO 6780
6700 '          else
6710              IF STATUS% = FALSE
                    THEN GOTO 6720 ELSE GOTO 6750
6720                  INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6730                  INTERRES%(TESTNO%,TIME%) = FAILTIME%
```

```
6740                    GOTO 6770
6750 '                 else
6760                      INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
6770 '                 endif:
6780 '               endif:
6790               NEXT TESTNO%
6800               FLAG2% = TRUE : GOTO 8170
6810 '           else
6820               FOR TESTNO% = 1 TO MAXTEST%
6830 '               call TEST_FAIL
6840                 GOSUB 8230
6850 '               case sample% of
6860                 ON SAMPLE% GOTO 6870,7480
6870 '               1 : *MLP*
6880                   IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                       THEN GOTO 6890 ELSE GOTO 7240
6890                     IF INTERRES$(TESTNO%,TTYPE%) = "FULL"
                           THEN GOTO 6900 ELSE GOTO 7020
6900                         IF INTERRES%(TESTNO%,COUNT%) = 1

                               THEN GOTO 6910 ELSE GOTO 6970
6910                             INTERRES$(TESTNO%,TTYPE%) = "PART"
6920                             INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
6930                             INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
6940                             INTERRES%(TESTNO%,COUNTK%) = CVI(I$)
6950                             INTERRES%(TESTNO%,FLG1%) = 1
6960                             GOTO 7000
6970 '                         else
6980                             INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6990                             INTERRES%(TESTNO%,COUNT%) =
                                       INTERRES%(TESTNO%,COUNT%)-1
7000 '                       endif:
7010                         GOTO 7220
7020 '                     else
7030                         IF INTERRES%(TESTNO%,COUNT%) = 1
                               THEN GOTO 7040 ELSE GOTO 7180
7040                             IF INTERRES%(TESTNO%,COUNTK%) = 1
                                   THEN GOTO 7050 ELSE GOTO 7120
7050                               IF INTERRES%(TESTNO%,FLG1%) < 2
                                     THEN GOTO 7060 ELSE GOTO 7070
7060                                   INTERRES%(TESTNO%,FLG1%) =
                                             INTERRES%(TESTNO%,FLG1%)+1
7070 '                             endif:
7080                               INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7090                               INTERRES%(TESTNO%,COUNTK%) = CVI(I$)
7100                               INTERRES%(TESTNO%,COUNT%) =
                                         CVI(F$)**INTERRES%(TESTNO%,FLG1%)
7110                               GOTO 7160
7120 '                           else
7130                               INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7140                               INTERRES%(TESTNO%,COUNTK%) =
                                         INTERRES%(TESTNO%,COUNTK%)-1
7150                               INTERRES%(TESTNO%,COUNT%) =
                                         CVI(F$)**INTERRES%(TESTNO%,FLG1%)
7160 '                           endif:
7170                             GOTO 7210
7180 '                         else
```

```
7190                        INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7200                        INTERRES%(TESTNO%,COUNT%) =
                                    INTERRES%(TESTNO%,COUNT%)-1
7210 '                    endif:
7220 '                  endif:
7230                    GOTO 7460
7240 '                else
7250                    IF INTERRES$(TESTNO%,OUTCOME%) ="FAIL"
                          THEN GOTO 7260 ELSE GOTO 7450
7260                        IF INTERRES$(TESTNO%,TTYPE%) = "FULL"

                              THEN GOTO 7270 ELSE GOTO 7310
7270                            INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7280                            INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7290                            INTERRES%(TESTNO%,FLG1%) = 0
7300                            GOTO 7440
7310 '                        else
7320                            IF INTERRES%(TESTNO%,FLG1%) = 1
                                  THEN GOTO 7330 ELSE GOTO 7380
7330                                INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7340                                INTERRES$(TESTNO%,TTYPE%) = "FULL"
7350                                INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7360                                INTERRES%(TESTNO%,FLG1%) = 0
7370                                GOTO 7430
7380 '                            else
7390                                INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7400                                INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7410                                INTERRES%(TESTNO%,FLG1%) =
                                            INTERRES%(TESTNO%,FLG1%)-1
7420                                INTERRES%(TESTNO%,COUNT%) =
                                        CVI(F$)**INTERRES%(TESTNO%,FLG1%)-1
7430 '                            endif:
7440 '                        endif:
7450 '                    endif:
7460 '                endif:
7470                    GOTO 7940
7480 '                2 : *HLP-T*
7490                    IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                          THEN GOTO 7500 ELSE GOTO 7850
7500                        IF INTERRES$(TESTNO%,TTYPE%) = "FULL"
                              THEN GOTO 7510 ELSE GOTO 7630
7510                            IF INTERRES%(TESTNO%,COUNT%) = 1
                                  THEN GOTO 7520 ELSE GOTO 7580
7520                                INTERRES$(TESTNO%,TTYPE%) = "PART"
7530                                INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7540                                INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7550                                INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7560                                INTERRES%(TESTNO%,FLG1%) = 1
7570                                GOTO 7610
7580 '                            else
7590                                INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7600                                INTERRES%(TESTNO%,COUNT%) =
                                            INTERRES%(TESTNO%,COUNT%)-1
7610 '                          endif:
7620                            GOTO 7830
7630 '                        else
7640                            IF INTERRES%(TESTNO%,COUNT%) = 1
```

```
                    THEN GOTO 7650 ELSE GOTO 7790
7650                  IF INTERRES%(TESTNO%,COUNTK%) = 1
                        THEN GOTO 7660 ELSE GOTO 7730
7660                      IF INTERRES%(TESTNO%,FLG1%) < 2
                            THEN GOTO 7670 ELSE GOTO 7680
7670                          INTERRES%(TESTNO%,FLG1%) =
                                        INTERRES%(TESTNO%,FLG1%)+1
7680 '                      endif:
7690                        INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7700                        INTERRES%(TESTNO%,COUNTK%) = CVI(I$)
7710                        INTERRES%(TESTNO%,COUNT%) =
                                CVI(F$)**INTERRES%(TESTNO%,FLG1%)
7720                        GOTO 7770
7730 '                    else
7740                        INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7750                        INTERRES%(TESTNO%,COUNTK%) =
                                        INTERRES%(TESTNO%,COUNTK%)-1
7760                        INTERRES%(TESTNO%,COUNT%) =
                                CVI(F$)**INTERRES%(TESTNO%,FLG1%)
7770 '                    endif:
7780                      GOTO 7820
7790 '                  else
7800                      INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7810                      INTERRES%(TESTNO%,COUNT%) =
                                        INTERRES%(TESTNO%,COUNT%)-1
7820 '                endif:
7830 '              endif:
7840                GOTO 7920
7850 '            else
7860                IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
                      THEN GOTO 7870 ELSE GOTO 7910
7870                    INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7880                    INTERRES$(TESTNO%,TTYPE%) = "FULL"
7890                    INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7900                    INTERRES%(TESTNO%,FLG1%) = 0
7910 '              endif:
7920 '            endif:
7930              GOTO 7940
7940 '          endcase:
7950            IF INTERRES$(TESTNO%,TSTATUS%) = "RUN"
                  THEN GOTO 7960 ELSE GOTO 8120
7960 '              call INTER_UNIT
7970                GOSUB 8700
7980                IF STATUS% = TRUE
                      THEN GOTO 7990 ELSE GOTO 8020
7990                    INTERRES$(TESTNO%,OUTCOME%) = "PASS"
8000                    INTERRES%(TESTNO%,TIME%) = PASSTIME%
8010                    GOTO 8100
8020 '                else
8030                    IF STATUS% = FALSE
                          THEN GOTO 8040 ELSE GOTO 8070
8040                        INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
8050                        INTERRES%(TESTNO%,TIME%) = FAILTIME%
8060                        GOTO 8090
8070 '                    else
8080                        INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
8090 '                  endif:
```

```
8100 '              endif:
8110                GOTO 8150
8120 '            else
8130                INTERRES$(TESTNO%,OUTCOME%) = "PASS"
8140                INTERRES%(TESTNO%,TIME%) = NOTIME%
8150 '            endif:
8160          NEXT TESTNO%
8170 '  endif:
8180 'end:
8190  RETURN
8200 REM End of Algorithm_B ( MLP Sampling )
8210 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
8220 REM ################################################################################
8230 REM Test Fail Process ( TEST_FAIL )
8240 'begin:
8250    IF TESTNO% <> 8
          THEN GOTO 8260 ELSE GOTO 8550
8260       IF FLAG1% = FALSE
             THEN GOTO 8270 ELSE GOTO 8530
8270           NODEF% = CVI(PSIZE$)
8280           IF VAR% <= NODEF%
                 THEN GOTO 8290 ELSE GOTO 8460
8290              IF PRODUCTNO% < RNRES%(VAR%,PNO%)
                    THEN GOTO 8300 ELSE GOTO 8310
8300                  TEST% = 0 : KEYNO% = &HFF : GOTO 8440
8310 '                else
8320                  TEST% = RNRES%(VAR%,TNO%)
8330 '                * Keyno does not correspond to test *
8340                  ON TEST% GOTO 8350,8360,8370,8380,8390,8400,
                                    8410,8420
8350                    KEYNO% = &HFE : GOTO 8430
8360                    KEYNO% = &HFD : GOTO 8430
8370                    KEYNO% = &HDF : GOTO 8430
8380                    KEYNO% = &HFB : GOTO 8430
8390                    KEYNO% = &HF7 : GOTO 8430
8400                    KEYNO% = &HEF : GOTO 8430
8410                    KEYNO% = &H7F : GOTO 8430
8420                    KEYNO% = &HFF
8430                    VAR% = VAR%+1
8440 '                endif:
8450                  GOTO 8480
8460 '              else
8470                  TEST% = 0 : KEYNO% = &HFF
8480 '            endif:
8490 '            put:
8500                OUT TFAIL%,KEYNO%   ' Interface Unit
8510 '            endput:
8520              FLAG1% = TRUE
8530 '       endif:
8540        GOTO 8650
8550 '    else
8560         IF TEST% = 8
              THEN GOTO 8570 ELSE GOTO 8580
8570            KEYNO% = &HFC : GOTO 8600
8580 '        else
8590            KEYNO% = &HFF
8600 '       endif:
```

```
8610 '       put:
8620           OUT TFAIL%,KEYNO%   ' Interface Unit
8630 '       endput:
8640         FLAG1% = FALSE
8650 '    endif:
8660 'end:
8670  RETURN
8680 REM End of Test Fail Process ( TEST_FAIL )
8690 REM ##########################################################################
8700 REM Interface Unit Process  ( INTER_UNIT )
8710 'begin:
8720 OUT MDA%,TESTDAT%(TESTNO%)
8730 OUT SETRE%,&H0
8740 IN% = INP(RDRC%) AND &H1
8750 IF IN% <> &H1 THEN GOTO 8740
8760 IN% = INP(RDRC%) AND &H1
8770 IF IN% <> &H0 THEN GOTO 8760
8780 STATUS% = INP(RDDAT%)
8790 OUT SETRE%,&H0
8800 'end:
8810  RETURN
8820 REM End of Interface Unit Process ( INTER_UNIT )
8830 REM ##########################################################################
8840 REM End of Algorithm Process
8850 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
8860 REM Test Result Process ( TEST_RES_PRO )
8870 'begin:
8880    IF SAVER$ = "Y" OR SAVER$ = "y" THEN GOTO 8890 ELSE GOTO 8980
8890    TESTNO% = 1
8900    DORES$(REF%,PSTATUS%) = "PASS"
8910    DORES$(REF%,TTYPE%) = INTERRES$(TESTNO%,TTYPE%)
8920    DORES%(REF%,COUNT%) = INTERRES%(TESTNO%,COUNT%)
8930    DORES$(REF%,TSTATUS%) = INTERRES$(TESTNO%,TSTATUS%)
8940    DORES$(REF%,OUTCOME%) = INTERRES$(TESTNO%,OUTCOME%)
8950    DORES%(REF%,TIME%) = INTERRES%(TESTNO%,TIME%)
8960    DORES%(REF%,PFAIL%) = TEST%
8970    DORES%(REF%,TNUM%) = TESTNO%
8980    TOTALTIME% = 0
8990    FOR TESTNO% = 1 TO MAXTEST%
9000      GET #5,ID
9010      IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
            THEN GOTO 9020 ELSE GOTO 9050
9020          DORES$(REF%,PSTATUS%) = "FAIL"
9030          ITR1 = CVI(DEFNUMBER$)+1
9040          LSET DEFNUMBER$ = MKI$(ITR1) : PUT #5,ID
9050 '    endif:
9060      TOTALTIME% = TOTALTIME%+INTERRES%(TESTNO%,TIME%)
9070    NEXT TESTNO%
9080    DORES%(REF%,PTIME%) = TOTALTIME%
9090    GET #5,ID
9100    ITR3# = CVD(RUNTIME$)+TOTALTIME%
9110    LSET RUNTIME$ = MKD$(ITR3#) : PUT #5,ID
9120    IF SAVER$ = "Y" OR SAVER$ ="y" THEN GOTO 9130 ELSE GOTO 9300
9130    REF% = REF%+1
9140    IF PRODUCTNO% = CNT%*25 OR PRODUCTNO% = CVI(SSIZE$)
          THEN GOTO 9150 ELSE GOTO 9290
9150 '     * Copy Data Output Resource to Random File #4 *
```

```
9160        FOR ITR2 = 1 TO (REF%-1)
9170          LSET TTYPE$ = DORES$(ITR2,TTYPE%)
9180          LSET COUNT$ = MKI$(DORES%(ITR2,COUNT%))
9190          LSET TSTATUS$ = DORES$(ITR2,TSTATUS%)
9200          LSET OUTCOME$ =DORES$(ITR2,OUTCOME%)
9210          LSET TTIME$ = MKI$(DORES%(ITR2,TIME%))
9220          LSET TNUM$ = MKI$(DORES%(ITR2,TNUM%))
9230          LSET PFAIL$ = MKI$(DORES%(ITR2,PFAIL%))
9240          LSET PSTATUS$ = DORES$(ITR2,PSTATUS%)
9250          LSET PTIME$ = MKI$(DORES%(ITR2,PTIME%))
9260          PUT #4,((CNT%-1)*25+ITR2)
9270        NEXT ITR2
9280        CNT% = CNT%+1 : REF% = 1
9290 '  endif:
9300 'end:
9310  RETURN
9320 REM End of Test Result Process ( TEST_RES_PRO )
9330 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
9340 REM End of Sampling Process ( SAMPLING_PROCESS )
10000 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

G.3 BATCH INSPECTION PLAN SOFTWARE LISTING

```
5200 REM ************************************************************
5210 REM Sampling Process ( Sampling_Process )
5220 'begin:
5230 '  put:
5240      PRINT " "
5250      PRINT "  SAMPLING PROCESS "
5260      PRINT " "
5270      PRINT "  1 : Run all test processes "
5280      PRINT "  2 : Run an individual test process "
5290      PRINT "  3 : Quit "
5300      PRINT "  Select the required option and press <enter> "
5310      PRINT " "
5320 '  endput:
5330 '  get:
5340      INPUT IS1
5350 '  endget:
5360    IF IS1 < 1 OR IS1 > 3 THEN GOTO 5330
5370    IF IS1 = 3 THEN GOTO 5420
5380 '  get:
5390      INPUT "  Save results (Y/N) " ; SAVER$
5400 '  endget:
5410    IF SAVER$ = "Y" OR SAVER$ = "y" OR SAVER$ = "N" OR SAVER$ = "n"
        THEN GOTO 5420 ELSE GOTO 5380
5420 '  case IS1 of
5430    ON IS1 GOTO 5440,5620,5810
5440 '    1 : *all tests*
5450        FOR ID = 1 TO IDTOTAL
5460          GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5470          LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
              :PUT #5,ID
5480 '        initialise R_N_RES%
5490          OPEN PCFN$ AS #3 LEN=9
5500          FIELD #3, 6 AS PNO$, 3 AS TNO$
5510            IDC3 = CVI(PSIZE$)
5520            FOR IDC4 = 1 TO IDC3
5530              GET #3,IDC4
5540              RNRES%(IDC4,PNO%) = CVI(PNO$)
5550              RNRES%(IDC4,TNO%) = CVI(TNO$)
5560            NEXT IDC4
5570          CLOSE #3
5580 '        call SAMPLING PLANS
5590          GOSUB 5850
5600        NEXT ID
5610        GOTO 5820
5620 '    2 : *individual test*
5630 '      get:
5640          INPUT "  Enter the ID of the record ";ID
5650 '      endget:
5660        IF ID < 1 OR ID > IDTOTAL THEN GOTO 5630
5670          GET #1,ID : GET #2,ID : GET #5,ID : CNT% = 1
5680          LSET DEFNUMBER$ = MKI$(0) : LSET RUNTIME$ = MKD$(0)
              :PUT #5,ID
5690 '        initialise R_N_RES%
5700          OPEN PCFN$ AS #3 LEN=9
5710          FIELD #3, 6 AS PNO$, 3 AS TNO$
5720            IDC3 = CVI(PSIZE$)
5730            FOR IDC4 = 1 TO IDC3
```

```
5740              GET #3,IDC4
5750              RNRES%(IDC4,PNO%) = CVI(PNO$)
5760              RNRES%(IDC4,TNO%) = CVI(TNO$)
5770            NEXT IDC4
5780            CLOSE #3
5790 '       call SAMPLING PLANS
5800         GOSUB 5850 : GOTO 5820
5810 '     3 : *quit*
5820 '   endcase:
5830 'end:
5840  RETURN
5850 'Subroutine SAMPLING PLAN
5860 'begin:
5870    OPEN DOFN$ AS #4 LEN=39
5880    FIELD #4, 5 AS PFAIL$, 5 AS PSTATUS$, 3 AS PTIME$, 7 AS OUTCOME$
      , 6 AS TTYPE$, 4 AS COUNT$, 5 AS TSTATUS$, 2 AS TNUM$, 2 AS TTIME$
5890    VAR% = 1 : CNT% = 1 : REF% = 1 : FLAG1% = FALSE
5900    CS1 = $PLAN$ : IS3 = CVI(SSIZE$)
5910    IF CS1="100%  " THEN IS2=1 : GOTO 5970
5920    IF CS1="CSP-F " OR CS1="csp-f "
          THEN IS2=2 : SAMPLE%=1 : GOTO 5970
5930    IF CS1="WSP-1 " OR CS1="wsp-1 "
          THEN IS2=2 : SAMPLE%=2 : GOTO 5970
5940    REM
5950    REM
5960    REM
5970 '  case IS2 of
5980    ON IS2 GOTO 5990,6120
5990 '    1 : *100% Sampling*
6000        FOR PRODUCTNO% = 1 TO IS3
6010 '        call ALGORITHM_A PROCESS ( ALGM_A_PRO )
6020          GOSUB 6310
6030 '        call TEST RESULT PROCESS ( T_RES_PRO )
6040          GOSUB 8610
6050 '        put:
6060            PRINT "  Record "; USING "##";ID;
6070            PRINT " : Completed "; USING "#####";PRODUCTNO%;
6080            PRINT " of "; USING "#####";IS3
6090 '          endput:
6100        NEXT PRODUCTNO%
6110        GOTO 6250
6120 '    2 : *CSP and WSP Sampling*
6130        FLAG2% = FALSE
6140        FOR PRODUCTNO% = 1 TO IS3
6150 '        call ALGORTHM_B PROCESS ( ALGM_B_PRO )
6160          GOSUB 6610
6170 '        call TEST RESULT PROCESS ( T_RES_PRO )
6180          GOSUB 8610
6190 '        put:
6200            PRINT "  Record "; USING "##";ID;
6210            PRINT " : Completed "; USING "#####";PRODUCTNO%;
6220            PRINT " of "; USING "#####";IS3
6230 '          endput:
6240        NEXT PRODUCTNO%
6250 '  endcase:
6260    LSET SSTATUS$ = "TRUE" : PUT #2,ID :CLOSE #4
6270 'end:
```

```
6280  RETURN
6290 REM End of Sampling Plans
6300 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
6310 REM Algorithm_A ( 100% Testing )
6320 'begin:
6330    FLAG1% = FALSE
6340    FOR TESTNO% = 1 TO MAXTEST%
6350      INTERRES$(TESTNO%,TTYPE%) = "FULL"
6360      INTERRES%(TESTNO%,COUNT%) = 0
6370      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6380      INTERRES%(TESTNO%,TIME%) = 0
6390 '    call TEST_FAIL
6400      GOSUB 7990
6410 '    call INTER_UNIT
6420      GOSUB 8460
6430      IF STATUS% = TRUE
            THEN GOTO 6440 ELSE GOTO 6470
6440          INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6450          INTERRES%(TESTNO%,TIME%) = PASSTIME%
6460          GOTO 6550
6470 '      else
6480          IF STATUS% = FALSE
                THEN GOTO 6490 ELSE GOTO 6520
6490              INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6500              INTERRES%(TESTNO%,TIME%) = FAILTIME%
6510              GOTO 6540
6520 '          else
6530              INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
6540 '        endif:
6550 '    endif:
6560    NEXT TESTNO%
6570 'end:
6580  RETURN
6590 REM End of Algorithm_A ( 100% Testing )
6600 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
6610 REM Algorithm_B ( CSP and WSP Sampling )
6620 'begin:
6630    FLAG1% = FALSE
6640    IF FLAG2% = FALSE
          THEN GOTO 6650 ELSE GOTO 7060
6650 '      *Initialise*
6660        FOR TESTNO% = 1 TO MAXTEST%
6670          IF IS2 = 2 AND SAMPLE% >= 2
                THEN GOTO 6680 ELSE GOTO 6810
6680              AOQL! = CVS(AOQL$)/MAXTEST%
6690              AOQL! = (AOQL!*CVI(SSIZE$)), .VI(F$)-1)
6700              IF AOQL! < 1
                    THEN GOTO 6710 ELSE GOTO 6740
6710                  INTERRES$(TESTNO%,TTYPE%) = "FULL"
6720                  INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6730                  GOTO 6790
6740 '              else
6750                  INTERRES$(TESTNO%,TTYPE%) = "PART"
6760                  INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6770                  INTERRES%(TESTNO%,COUNT%) = CVI(F$)
6780                  INTERRES%(TESTNO%,COUNTK%) = INT(AOQL!)
6790 '            endif:
```

```
6800                GOTO 6860
6810 '            else
6820                INTERRES$(TESTNO%,TTYPE%) = "FULL"
6830                INTERRES%(TESTNO%,COUNT%) = CVI(I$)
6840                INTERRES$(TESTNO%,TSTATUS%) = "RUN"
6850                INTERRES%(TESTNO%,TIME%) = 0
6860 '          endif:
6870 '          call TEST_FAIL
6880            GOSUB 7990
6890 '          call INTER_UNIT
6900            GOSUB 8460
6910            IF STATUS% = TRUE
                  THEN GOTO 6920 ELSE GOTO 6950
6920                INTERRES$(TESTNO%,OUTCOME%) = "PASS"
6930                INTERRES%(TESTNO%,TIME%) = PASSTIME%
6940                GOTO 7030
6950 '            else
6960                IF STATUS% = FALSE
                      THEN GOTO 6970 ELSE GOTO 7000
6970                  INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
6980                  INTERRES%(TESTNO%,TIME%) = FAILTIME%
6990                  GOTO 7020
7000 '              else
7010                  INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
7020 '              endif:
7030 '          endif:
7040          NEXT TESTNO%
7050          FLAG2% = TRUE : GOTO 7930
7060 '      else
7070          FOR TESTNO% = 1 TO MAXTEST%
7080 '          call TEST_FAIL
7090            GOSUB 7990
7100 '          case sample% of
7110            ON SAMPLE% GOTO 7120,7440
7120 '            1 : *CSP-F*
7130                IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                      THEN GOTO 7140 ELSE GOTO 7340
7140                  IF INTERRES%(TESTNO%,COUNT%) = 1
                        THEN GOTO 7150 ELSE GOTO 7250
7150                    IF INTERRES$(TESTNO%,TTYPE%) = "FULL"
                          THEN GOTO 7160 ELSE GOTO 7200
7160                      INTERRES$(TESTNO%,TTYPE%) = "PART"
7170                      INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7180                      INTERRES%(TESTNO%,COUNT%) = CVI(F$)-1
7190                      GOTO 7230
7200 '                  else:
7210                      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7220                      INTERRES%(TESTNO%,COUNT%) = CVI(F$)
7230 '                  endif:
7240                    GOTO 7330
7250 '                else
7260                    IF INTERRES$(TESTNO%,TTYPE%) = "FULL"
                          THEN GOTO 7270 ELSE GOTO 7290
7270                      INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7280                      GOTO 7310
7290 '                  else
7300                      INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
```

```
7310 '                    endif:
7320                      INTERRES%(TESTNO%,COUNT%) =
                                INTERRES%(TESTNO%,COUNT%)-1
7330 '                 endif:
7340 '              endif:
7350                IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
                      THEN GOTO 7360 ELSE GOTO 7390
7360                    INTERRES%(TESTNO%,COUNT%) = CVI(I$)
7370                    INTERRES$(TESTNO%,TTYPE%) = "FULL"
7380                    INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7390 '              endif:
7400                IF INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
                      THEN GOTO 7410 ELSE GOTO 7420
7410                    INTERRES%(TESTNO%,COUNT%) =
                              INTERRES%(TESTNO%,COUNT%)-1
7420 '              endif:
7430                GOTO 7700
7440 '            2 : *WSP-1*
7450                IF INTERRES$(TESTNO%,TTYPE%) = "PART"
                      THEN GOTO 7460 ELSE GOTO 7680
7460                    IF INTERRES$(TESTNO%,OUTCOME%) = "PASS"
                          THEN GOTO 7470 ELSE GOTO 7560
7470                        IF INTERRES%(TESTNO%,COUNT%) = 1
                              THEN GOTO 7480 ELSE GOTO 7510
7480                            INTERRES%(TESTNO%,COUNT%) = CVI(F$)
7490                            INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7500                            GOTO 7540
7510 '                      else
7520                            INTERRES%(TESTNO%,COUNT%) =
                                      INTERRES%(TESTNO%,COUNT%)-1
7530                            INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7540 '                      endif:
7550                        GOTO 7670
7560 '                  else
7570                      IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
                            THEN GOTO 7580 ELSE GOTO 7660
7580                          IF INTERRES%(TESTNO%,COUNT%) = 1
                                THEN GOTO 7590 ELSE GOTO 7620
7590                              INTERRES$(TESTNO%,TTYPE%) = "FULL"
7600                              INTERRES$(TESTNO%,TSTATUS%) = "RUN"
7610                              GOTO 7650
7620 '                        else
7630                              INTERRES$(TESTNO%,TSTATUS%) = "NORUN"
7640                              INTERRES%(TESTNO%,COUNT%) =
                                        INTERRES%(TESTNO%,COUNT%)-1
7650 '                        endif:
7660 '                    endif:
7670 '                endif:
7680 '            endif:
7690              GOTO 7700
7700 '          endcase:
7710            IF INTERRES$(TESTNO%,TSTATUS%) = "RUN"
                  THEN GOTO 7720 ELSE GOTO 7880
7720 '              call INTER_UNIT
7730                GOSUB 8460
7740                IF STATUS% = TRUE
```

```
                     THEN GOTO 7750 ELSE GOTO 7780
7750                   INTERRES$(TESTNO%,OUTCOME%) = "PASS"
7760                   INTERRES%(TESTNO%,TIME%) = PASSTIME%
7770                   GOTO 7860
7780 '               else
7790                   IF STATUS% = FALSE
                         THEN GOTO 7800 ELSE GOTO 7830
7800                       INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
7810                       INTERRES%(TESTNO%,TIME%) = FAILTIME%
7820                       GOTO 7850
7830 '                   else
7840                       INTERRES$(TESTNO%,OUTCOME%) = "UNKNOWN"
7850 '                 endif:
7860 '             endif:
7870               GOTO 7910
7880 '           else
7890               INTERRES$(TESTNO%,OUTCOME%) = "PASS"
7900               INTERRES%(TESTNO%,TIME%) = NOTIME%
7910 '         endif:
7920         NEXT TESTNO%
7930 '   endif:
7940 'end:
7950  RETURN
7960 REM End of Algorithm_B ( CSP and WSP Sampling )
7970 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
7980 REM ############################################################
7990 REM Test Fail Process ( TEST_FAIL )
8000 'begin:
8010     IF TESTNO% <> 8
           THEN GOTO 8020 ELSE GOTO 8310
8020         IF FLAG1% = FALSE
               THEN GOTO 8030 ELSE GOTO 8290
8030             NODEF% = CVI(PSIZE$)
8040             IF VAR% <= NODEF%
                   THEN GOTO 8050 ELSE GOTO 8220
8050                 IF PRODUCTNO% < RNRES%(VAR%,PNO%)
                       THEN GOTO 8060 ELSE GOTO 8070
8060                     TEST% = 0 : KEYNO% = &HFF : GOTO 8200
8070 '                 else
8080                     TEST% = RNRES%(VAR%,TNO%)
8090 '                   * Keyno does not correspond to test *
8100                     ON TEST% GOTO 8110,8120,8130,8140,8150,8160,
                                       8170,8180
8110                       KEYNO% = &HFE : GOTO 8190
8120                       KEYNO% = &HFD : GOTO 8190
8130                       KEYNO% = &HDF : GOTO 8190
8140                       KEYNO% = &HFB : GOTO 8190
8150                       KEYNO% = &HF7 : GOTO 8190
8160                       KEYNO% = &HEF : GOTO 8190
8170                       KEYNO% = &H7F : GOTO 8190
8180                       KEYNO% = &HFF
8190                     VAR% = VAR%+1
8200 '               endif:
8210                 GOTO 8240
8220 '             else
8230                 TEST% = 0 : KEYNO% = &HFF
8240 '           endif:
```

```
8250 '          put:
8260              OUT TFAIL%,KEYNO%  ' Interface Unit
8270 '          endput:
8280            FLAG1% = TRUE
8290 '      endif:
8300        GOTO 8410
8310 '    else
8320        IF TEST% = 8
             THEN GOTO 8330 ELSE GOTO 8340
8330           KEYNO% = &HFC : GOTO 8360
8340 '         else
8350             KEYNO% = &HFF
8360 '      endif:
8370 '      put:
8380          OUT TFAIL%,KEYNO%  ' Interface Unit
8390 '      endput:
8400        FLAG1% = FALSE
8410 '  endif:
8420 'end:
8430  RETURN
8440 REM End of Test Fail Process ( TEST_FAIL )
8450 REM ##########################################################################
8460 REM Interface Unit Process ( INTER_UNIT )
8470 'begin:
8480 OUT MDA%,TESTDAT%(TESTNO%)
8490 OUT SETRE%,&H0
8500 IN% = INP(RDRC%) AND &H1
8510 IF IN% <> &H1 THEN GOTO 8500
8520 IN% = INP(RDRC%) AND &H1
8530 IF IN% <> &H0 THEN GOTO 8520
8540 STATUS% = INP(RDDAT%)
8550 OUT SETRE%,&H0
8560 'end:
8570  RETURN
8580 REM End of Interface Unit Process ( INTER_UNIT )
8590 REM ##########################################################################
8600 REM End of Algorithm Process
8610 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
8620 REM Test Result Process ( TEST_RES_PRO )
8630 'begin:
8640    IF SAVER$ = "Y" OR SAVER$ = "y" THEN GOTO 8650 ELSE GOTO 8740
8650    TESTNO% = 1
8660    DORES$(REF%,PSTATUS%) = "PASS"
8670    DORES$(REF%,TTYPE%) = INTERRES$(TESTNO%,TTYPE%)
8680    DORES%(REF%,COUNT%) = INTERRES%(TESTNO%,COUNT%)
8690    DORES$(REF%,TSTATUS%) = INTERRES$(TESTNO%,TSTATUS%)
8700    DORES$(REF%,OUTCOME%) = INTERRES$(TESTNO%,OUTCOME%)
8710    DORES%(REF%,TIME%) = INTERRES%(TESTNO%,TIME%)
8720    DORES%(REF%,PFAIL%) = TEST%
8730    DORES%(REF%,TNUM%) = TESTNO%
8740    TOTALTIME% = 0
8750    FOR TESTNO% = 1 TO MAXTEST%
8760      GET #5,ID
8770      IF INTERRES$(TESTNO%,OUTCOME%) = "FAIL"
            THEN GOTO 8780 ELSE GOTO 8810
8780          DORES$(REF%,PSTATUS%) = "FAIL"
              ITR1 = CVI(DEFNUMBER$)+1
```

```
8800            LSET DEFNUMBER$ = MKI$(ITR1) : PUT #5,ID
8810 '       endif:
8820         TOTALTIME% = TOTALTIME%+INTERRES%(TESTNO%,TIME%)
8830      NEXT TESTNO%
8840      DORES%(REF%,PTIME%) = TOTALTIME%
8850      GET #5,ID
8860      ITR3# = CVD(RUNTIME$)+TOTALTIME%
8870      LSET RUNTIME$ = MKD$(ITR3#) : PUT #5,ID
8880      IF SAVER$ = "Y" OR SAVER$ ="y" THEN GOTO 8890 ELSE GOTO 9060
8890      REF% = REF%+1
8900      IF PRODUCTNO% = CNT%*25 OR PRODUCTNO% = CVI(SSIZE$)
            THEN GOTO 8910 ELSE GOTO 9050
8910 '        * Copy Data Output Resource to Random File #4 *
8920          FOR ITR2 = 1 TO (REF%-1)
8930            LSET TTYPE$ = DORES$(ITR2,TTYPE%)
8940            LSET COUNT$ = MKI$(DORES%(ITR2,COUNT%))
8950            LSET TSTATUS$ = DORES$(ITR2,TSTATUS%)
8960            LSET OUTCOME$ =DORES$(ITR2,OUTCOME%)
8970            LSET TTIME$ = MKI$(DORES%(ITR2,TIME%))
8980            LSET TNUM$ = MKI$(DORES%(ITR2,TNUM%))
8990            LSET PFAIL$ = MKI$(DORES%(ITR2,PFAIL%))
9000            LSET PSTATUS$ = DORES$(ITR2,PSTATUS%)
9010            LSET PTIME$ = MKI$(DORES%(ITR2,PTIME%))
9020            PUT #4,((CNT%-1)*25+ITR2)
9030          NEXT ITR2
9040          CNT% = CNT%+1 : REF% = 1
9050 '   endif:
9060 'end:
9070  RETURN
9080 REM End of Test Result Process ( TEST_RES_PRO )
9090 REM @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
9100 REM End of Sampling Process ( SAMPLING_PROCESS )
10000 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REFERENCES AND BIBLIOGRAPHY

REFERENCES

1. Anscombe,F.J. (1950) "The cost of inspection", in the volume STATISTICAL METHODS IN INDUSTRIAL PROTECTION published by the Royal Statistical Society and containing the papers of the *Sheffield* Conference on 1950.

2. Banzaf,R.A., and Brugger,R.M. (1970) "MIL-STD-1235 (ORD) single and multi-level continuous sampling procedures and tables for inspection by attributes", J. QUAL. TECHNOLOGY, 2, pp 41-53.

3. Barnard,G.A. (1954) "Sampling inspection and statistical decisions", J.R. STATIST. SOC., B, 16, pp 151-174.

4. Barnett,V. (1974) "Economic choice of sample size for sampling inspection plans", APPLIED STATISTICS, 23, pp 149-157.

5. Blackwell,M.T.R. (1977) "The effect of short production runs on CSP-1", TECHNOMETRICS, 19, pp 259-263.

6. Champernowne,D.G. (1953) "The economics of sequential sampling procedures for defectives", APPLIED STATISTICS, 2, pp 118-130.

7. Chiu,W.K. (1974) "A new prior distribution for attributes sampling", TECHNOMETRICS, 16, pp 93-102.

8. Chiu,W.K., and Wetherill,G.B. (1975) "The economic design of continuous inspection procedures: a review paper", INST. STAT. REV., 41, pp 357-373.

9. Derman,C., Littauer,S., and Solomon,H. (1957) "Tightened multi- level continuous sampling plans", ANN. MATH. STATIST., 28, pp 395-404.

10. Derman.C., Johns,M.V.,Jnr, and Lieberman,G.J. (1959) "Continuous sampling procedures without control", ANN. MATH. STATIST., 30, pp 1175-1191.

11. Dodge,H.F. (1943) "A sampling inspection plan for continuous production", ANN. MATH. STATIST., 14, pp 264-279.

12. Dodge,H.F. (1969) "Notes on the evolution of acceptance sampling plans : Part I", J. QUAL. TECHNOLOGY, 1, pp 77-88.

13. Dodge,H.F. (1970) "Notes on the evolution acceptance sampling plans : Part IV", J. QUAL. TECHNOLOGY, 2, pp

14. Dodge,H.F., and Torrey,M.N. (1951) "Additional continuous sampling inspection plans", INDUSTRIAL QUALITY CONTROL, 7, pp 5-9.

15. Ford,J.H. (1951) "Examples of the process curve", D.I.C. Thesis, Imperial College.

16. *Hald,A. (196/b) "Asymptatic properties of Bayesian single sampling* plans", J.R.STATIST. SOC., B, 29, pp 162-173.

17. Hamaker,H.C. (1951) "Economic principles in industrial sampling problems : A general introduction", BUL. INTERNATIONAL STATISTICAL INSTITUTE, 33, pp 105-122.

18. Hamaker,H.C. (1958) "Some basic principles of sampling inspection by attributes", APPLIED STATISTICS., 7, pp 149-159.

19. Horsnell,G. (1957) "Economical acceptance sampling schemes", J.R.STATIST. SOC., A, 120, pp 148-201.

20. Juran,J.M. (1974) "Quality Control Handbook", Third Edition, McGRAW-HILL, Chapter 24, pp 24-1 - 24-44.

21. Lieberman,G.J. (1953) "A note on Dodge's continuous sampling inspection plan", ANN. MATH. STATIST., 24, pp 480-484.

22. Lieberman,G.J. and Solomon,H. (1955) "Multi-level continuous sampling plans", ANN. MATH. STATIST., 26, 686-704.

23. Mood,A.M. (1943) "On the dependence of sampling inspection plans upon population distribution", ANN. MATH. STATIST,. 14, pp 415-425.

24. Myklebust,B. and Hammerboen,K. (1983) "'Home made" ATE for high volume production of telephone sets", EB Communications, A/S Elektrisk Bureau, AIE 83, session 2, pp 66-78.

25. Pfanzagl,J. (1963) "Sampling procedures based on p[illegible] distributions and costs", TECHNOMETRICS, 5, pp 47-61.

26. Read,D.R., and Beattie,D.W. (1961) "The variable lot-size acceptance sampling plan for continuous production", APPLIED STATISTICS, 10, pp 147-156.

27. QSTAG 340 (1974) "Single and multi-level continuous sampling procedures and tables for inspection by attributes", Draft Document, U.S. Army.

28. Satterthwaite,F.E. and Grad,B. "The choice of lot inspection plans on the basis of cost", Report issued by the General Electric Company, Production Device Division, Bridgeport, Connecticut, U.S.A..

29. Shahani,A.K. (1979) "Wald-Wolfowitz type sampling plans for continuous production", TECHNOMETRICS, 21, pp 21-31.

30. Sittig,J. (1951) "The economic choice of sampling system in acceptance sampling", BULL. INTERNATIONAL STATISTICAL INSTITUTE, 33, pp 51-84.

31. Wald,A., and Wolfowitz,J. (1945) "Sampling inspection plans for continuous production which insures a prescribed limit on the outgoing quality", ANN. MATH. STATIST., 16, pp 30-49.

32. Walker,A.J. (1984) "Structured Information Processing System Design", Revision 1.2, Department of Electrical Engineering, University of the Witwatersrand, Johannesburg, South Africa.

33. Weibull,I. (1951) "A method of determining inspection plans on an economic basis", BULL. INTERNATIONAL STATISTICAL INSTITUTE, 33, pp 85-104.

34. Wetherill,G.B. (1977) "Sampling Inspection and Quality Control", *Second Edition*, CHAPMAN and HALL.

35. Wetherill,G.B. and Campling,G.E.G. (1966) "The decision theory approach to sampling inspection", J.R.STATIST. SOC., B, 28, pp 381-416.

36. Wetherill,G.B. and Chiu,W.K. (1975) "A review of acceptance sampling schemes with emphasis on the economic aspect", INT. STAT. REV., 43, pp 191-209.

BIBLIOGRAPHY

1. Anscombe,F.J. (1958) "Rectifying inspection of continuous output", J. AMER. STATIST. ASS., 53, pp 702-719.

2. Dodge,H.F. (1947) "Sampling plans for continuous production", INDUSTRIAL QUALITY CONTROL, 4, pp 5-9.

3. Dodge,H.F. (1969) "Notes on the *evolution of acceptance* sampling plans : Part II", J. QUAL. TECHNOLOGY, 1, pp 155-162.

4. Dodge,H.F. (1969) "Notes on the evolution of acceptance sampling plans : Part III", J. QUAL. TECHNOLOGY, 1, pp 225-232.

5. Dodge,H.F., and Romig,H.G. (1929) "A method of sampling inspection", BELL SYSTEM TECHNICAL JOURNAL, 8, pp 613-631.

6. Gregory,G. (1957) "Statistical quality control: A review of continuous sampling plans", JOURNAL OF THE TEXTILE INSTITUTE, 48, pp 467-481.

7. Hald,A. (1967a) "The determination of single *sampling attribute* plans with given producer's and consumer's risk", TECHNOMETRICS, 9, pp 401-415.

8. Hamaker,H.C. (1950) "The practical application of sampling inspection plans and tables", PHILIPS TECHNICAL REVIEW, 11, pp 260-270.

9. Heikes,R.G., and Montgomrey.D.C. (1981) "Productivity is enhanced with statistical quality control", *IE, May 81, pp 52-75.*

10. Hill,I.D. (1962) "Sampling inspection and defence specification *DEF-131*", J.R.STATIST. SOC., A, 125, pp 31-87.

11. Hillier,F.S. (1964) "New criteria for selecting continuous *sampling* plans", TECHNOMETRICS, 6, pp 161-178.

12. Lowe,C.W. (1968a) "*Industrial Statistics*", Volume 1, BUSINESS BOOKS LIMITED.

13. Lowe,C.W. (1968b) "Industrial Statistics", Volume 2, BUSINES LIMITED.

14. Mood,A.M. (1940) "The distribution theory of runs", ANN. MATH. STATIST., 11, pp 367-392.

15. Murphy,R.B. (1958) "A criterion to limit inspection effort in continuous sampling plans", BELL SYSTEM TECHNICAL JOURNAL, 37, pp 115-134.

16. Page,E.S. (1954) "Continuous sampling schemes", BIOMETRIKA, 41, pp 100-115.

17. Phillips,M.J. (1969) "A survey of sampling procedures for continuous production", J.R.STATIST. SOC., A, 132, pp 205-228.

18. Prairie,R.R., and Zimmer,W.J. (1970) "Continuous sampling plans based on cumulative sums", APPLIED STATISTICS, 19, pp 222-230.

19. Riggs,J.L. (1981) "Production Systems : Planing, Analysis and Control", Third Edition, WILEY, chapter 15, pp 544-546.

20. Sackrowitz,H. (1972) "Alternative multi-level continuous sampling plans", TECHNOMETRICS, 14, pp 645-652.

21. Savage,I.R. (1959) "A production model and continuous sampling plan", J.AMER. STATIST. ASS., 54, pp 231-247.

22. Tippett,L.H.C. (1958) "A guide to acceptance sampling", APPLIED STATIST., 7, pp 133-148.

23. Wetherill,G.B. and Chiu,W.K. (1974) "A simplified attribute sampling scheme", APPLIED STATISTICS, 23, pp 143-148.

24. White,L.S. (1965) "Markovian decision models for the evaluation of a large class of continuous sampling inspection plans", ANN. MATH. STATIST., 36, pp 1408-1420.