A MULTI-PROCESSOR SYSTEM FOR THE DETECTION OF ELECTROCARDIOGRAM ARRHYTHMIAS

BRYAN THOMAS VICTOR WARTON

PhD THESIS

The University of Aston in Birmingham.
Department of Electrical Engineering

\- - - - - -

JANUARY 1979.

A MULTI-PROCESSOR SYSTEM FOR THE DETECTION OF ELECTROCARDIOGRAM ARRHYTHMIAS

B.T.V. WARTON

S U M M A R Y

This thesis describes the research work performed to produce a real-time distributed microprocessor system, for the diagnosis of cardiac arrhythmias. The distributed microprocessor approach developed, allowed the allocation of a microprocessor to each patient for the purpose of electrocardiogram analysis. The functions performed by these patient-dedicated processors include digital filtering of the electrocardiogram, the detection and measurement of each heart beat complex, and the diagnosis of arrhythmias by decision tables.

These patient processors then communicate the required patient analysis results and diagnoses, to a centralised operator-dedicated processor. This operator-dedicated processor allows the functions of overall system control, display and storage to be centralised.

INDEXING TERMS : Multi-microprocessor, Electrocardiogram, Arrhythmia, Real-Time.

ACKNOWLEDGMENTS

The author wishes to acknowledge the guidance and advice provided by his project supervisor, Professor H.A. Barker, and the information and opinions of a medical nature provided by Dr. R.E. Nagle of the Queen Elizabeth Hospital, Birmingham, and Dr. A.J. Camm, of St. Bartholomew's Hospital, London.

I would also like to thank the members of the academic and technical staff at Aston University, who have helped during my research.

I would also like to acknowledge the award of a Research Studentship provided by the Science Research Council.

Finally, I wish to thank Miss Jean Wilson for her help and encouragement during my research, and for typing this thesis.

Many thanks,

Bryan Warton

Bryan Warton.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1. General

In the early 1960s, Coronary Care Units (CCUs) were introduced into hospitals to provide centralised facilities for monitoring cardiac patients. The initial motivation for these units was the need for rapid detection and treatment of life-threatening events, such as cardiac arrest and ventricular fibrillation (disorganised heart activity).

The method used to monitor the cardiac condition of patients in CCUs, is via the amplification of the small potential differences associated with heart action, which occur on the surface of the chest. These electrical signals from the heart are referred to as electrocardiograms (ECGs).

Since the introduction of CCUs, continuous ECG monitoring in these units has shown that a high incidence of certain cardiac arrhythmias (abnormal changes in heart rate) preceed life-threatening events. The treatment of these arrhythmias with antiarrhythmic agents, proved to be successful in reducing the number of mortalities in CCUs.

Conventional ECG monitoring in CCUs is based upon heart rate meter alarms, and nurse observation of oscilloscope displaying ECG signals. Despite the relatively good results achieved with this type of monitoring, it was quite clear that a great deal of information was lost.

A study made comparing the results obtained from conventional ECG monitoring, and subsequent detailed analysis of the recorded ECGs made at the same time, showed that less than 20% of the serious arrhythmias had been detected (Ref. 1.).

In an effort to relieve the nursing staff of continuous visual ECG monitoring and also to provide more detailed arrhythmia monitoring results, various computer monitoring systems have been developed. The high cost of these computer monitoring systems, however, has precluded their widespread use in CCUs.

The advent of the microprocessor has now provided the possibility of producing a cheaper and more flexible monitoring system, not only by a direct reduction in the cost of information processing, but also by the possibility of distributing this processing throughout the system.

This report describes the development of such a distributed microprocessor monitoring system, which uses a 16 bit microprocessor, the TMS 9900, for real-time detection and diagnosis of ECG arrhythmias (Appendix E Published Work).

1.2. Electrocardiography and Arrhythmia Criteria

The various deflections which occur in the ECG signal indicate the operation of the heart. The diagnosis of arrhythmias from such signals is concerned with changes in the shape and frequency of these deflections.

The interpretation of the deflections in the ECG signal, and the monitoring diagnosis criteria used by the microprocessor system are described in Chapter 2. This Chapter also discusses the limitations and problems encountered with the initial monitoring criteria used, and suggests a possible set of alternative monitoring criteria.

1.3. Monitoring Systems

There are a large number of independent research and development groups producing different types of ECG monitoring systems. The different approaches used by these systems are briefly indicated in Chapter 3, and hence the reasons for the methods chosen for implementation in the microprocessor monitoring system are given.

1.4. System Functions and Structure

Having indicated the approach for implementation in the microprocessor system, Chapter 4 describes the individual functions performed by the monitoring system, and the ideal distribution of each function throughout the system. The performance of these functions in real-time, as obtained by actual implementations, are then examined in order to show that the ideal function distribution in the system can be implemented by the allocation of microprocessors on a one-per-patient basis. The overall system control is provided by a central operator-dedicated microprocessor.

1.5. Patient-dedicated Processor System Design

Those functions assigned to the patient-dedicated processor system are all implemented in the compact media of software, as described in detail in Chapter 5. This Chapter also describes how these real-time functions are controlled by the patient processor operating system developed, which contains a real-time task scheduler.

1.6. Operator-dedicated Processor System Design

Those functions assigned to the operator-dedicated processor system, and produced in software, are described in detail in Chapter 6. This Chapter also describes the inter-processor communications protocol that was developed for the system, which allows communications between processors to operate without interfering with the real-time processing being performed by each patient-dedicated processor.

1.7. System Hardware

As the system functions produced for the developmental system are all in the compact media of software, the resulting system hardware is predominantly the basic microprocessor system circuit elements. The design of microprocessor hardware systems is described in detail in the TMS 9900 microprocessor support literature; therefore, only the basic structure of the hardware system is described in Chapter 7. This Chapter then proceeds to discuss in more detail, the analog to digital converter, and inter-processor communications interfaces used in the developmental system.

1.8. System Performance and Results

As a result of producing the developmental system, it was possible to determine the real-time performance of the patient processor software, and hence the possibilities for expansion within the system, as discussed in Chapter 8.

Also examined in this chapter are the inter-processor communications performance results, obtained from studying a simulated three patient processor monitoring system. The results obtained indicate the effect of the current inter-processor communication performance, on the total number of patient processors which could be connected to the monitoring system.

CHAPTER 2

ELECTROCARDIOGRAPHY AND ARRHYTHMIA CRITERIA

2.1. The Electrocardiogram

The heart is the complex organ which pumps blood around the body. During the pumping action there is electrical activity within the heart, which can be detected by chest and limb lead electrodes (Ref. 2.). In all, there are 12 different lead arrangements used in clinical electrocardiogram diagnostic practice (Ref. 3.).

In order to observe the changing condition of patients in CCUs, one of the chest leads is continuously monitored. The type of signal obtained from a chest lead position, used in CCUs, is illustrated in Fig. 2.1. The P wave shown in Fig 2.1, corresponds to the contraction of the atrium (the two chambers which receive blood in the heart), and the QRS complex corresponds to the contraction of the ventricals (the two chambers which force the blood out of the heart).

The diagnosis of cardiac arrhythmias from such a signal is concerned with abnormal changes in the heart rhythm (Ref. 4.). When diagnosing the type of arrhythmia that has occurred, not only is the time interval between each beat (R-R interval) of importance, but also the shape of the QRS complex, and possibly its relationship to P and T waves, must be considered. This

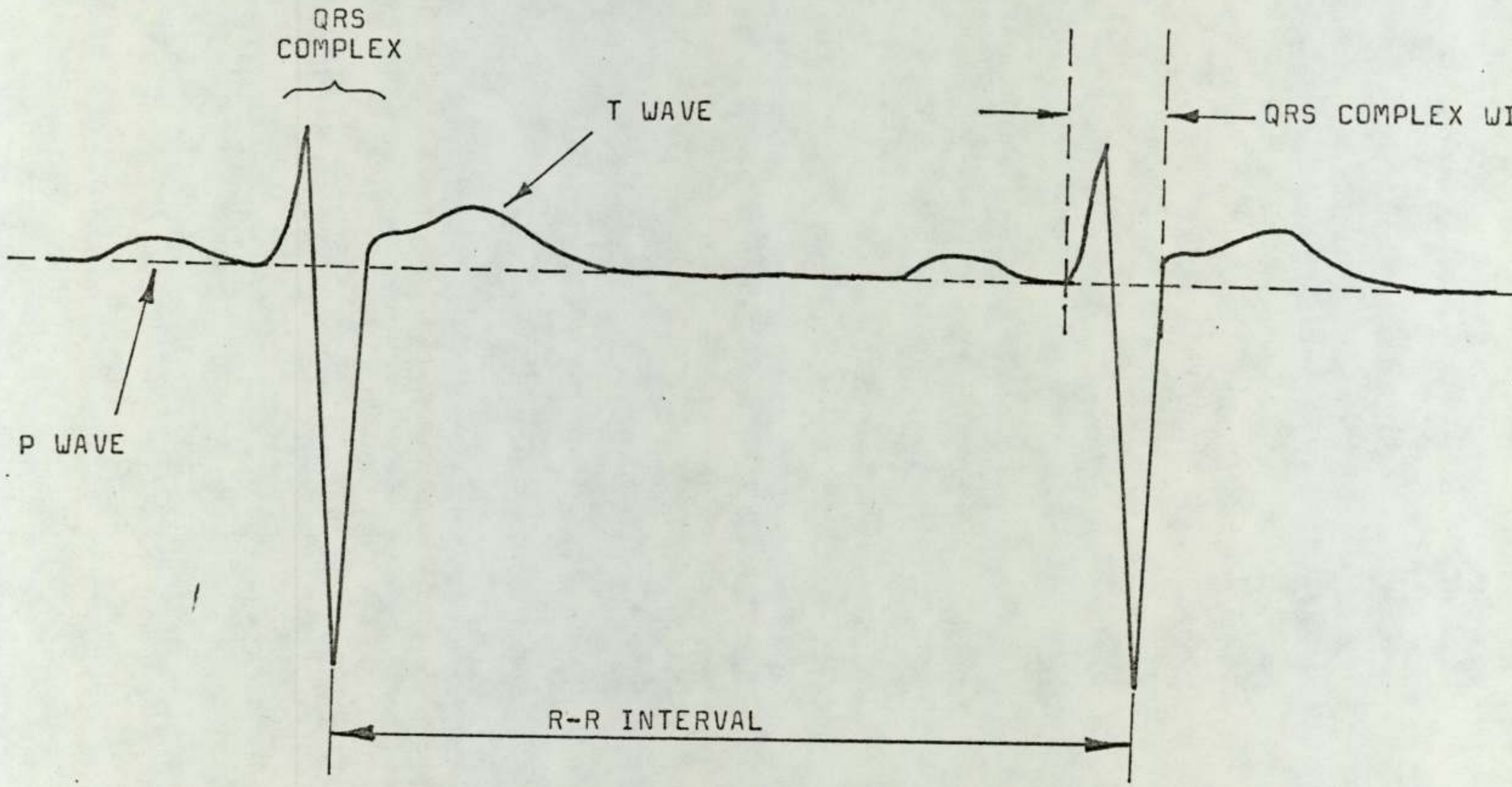


Fig 2.1. ECG CHARACTERISTICS

is because abnormal contractions of the heart can produce abnormal QRS complexes, and changes in heart rhythm.

2.2. Arrhythmia Criteria

The arrhythmia diagnosis approach adopted for the microprocessor monitoring system as discussed in Chapter 3, is that of pattern recognition. The diagnosis criteria used by real-time computer monitoring systems employing such an approach varies with each computer system as there is no agreed standard. As the criteria needed for the microprocessor system was initially required to enable the final structure of the system to be determined, it was decided that an existing set of criteria would be implemented. The requirements placed upon the criteria sought, ~~was~~ WERE that it should be as extensive as possible and representative of the type of criteria used in computer systems.

2.2.1. Initial Monitoring Criteria

The real-time computer monitoring system described by Rabin et. al. (Ref. 5.) provides a detailed list of arrhythmia diagnosis criteria, as reproduced in Table 2.1. From this list it can be seen that a total of 23 different arrhythmias or events are specified, the principal parameters involved being the R-R interval and QRS complex width.

As a result of discussions with cardiologists to determine their opinion of the diagnosis criteria shown in Table 2.1., the following points were noted :-

(1) The diagnosis of Escape beat defines the tolerance on the R-R interval (IER), used to classify them as long, short or normal, as

$$IER = \frac{ISAMP}{12} \text{ if pulse rate } > 120 \text{ beats/min.} \quad (2\cdot1)$$

or

$$IER = ISAMP\left(\frac{1}{8} - \frac{RATE}{2880}\right) \text{if pulse rate } \leqslant 120 \text{ beats/min.} \quad (2\cdot2)$$

No definition of ISAMP was given in the paper by Rabin et. al. After studying the above equations, it was concluded that the parameter ISAMP would be defined as, the average R-R interval measured in seconds. It was then possible to produce a graph showing the percentage tolerance allowed on the variation of the R-R interval, related to pulse rate, as shown in Fig. 2.2. From this graph it can be seen that the two equations above, give the same result at 120 beats/minute, and that as the pulse rate decreases from this point, the percentage tolerance increases.

It was then noted that in the opinion of cardiologists consulted, there was no need to relate the tolerance on the R-R interval to pulse rate. They considered that a constant percentage tolerance of 10 or 15 percent would be satisfactory.

(2) In several diagnoses, notably Supraventricular tachycardia and Bundle Branch Block, it was noted that a criteria of, 82% or more of QRS complexes

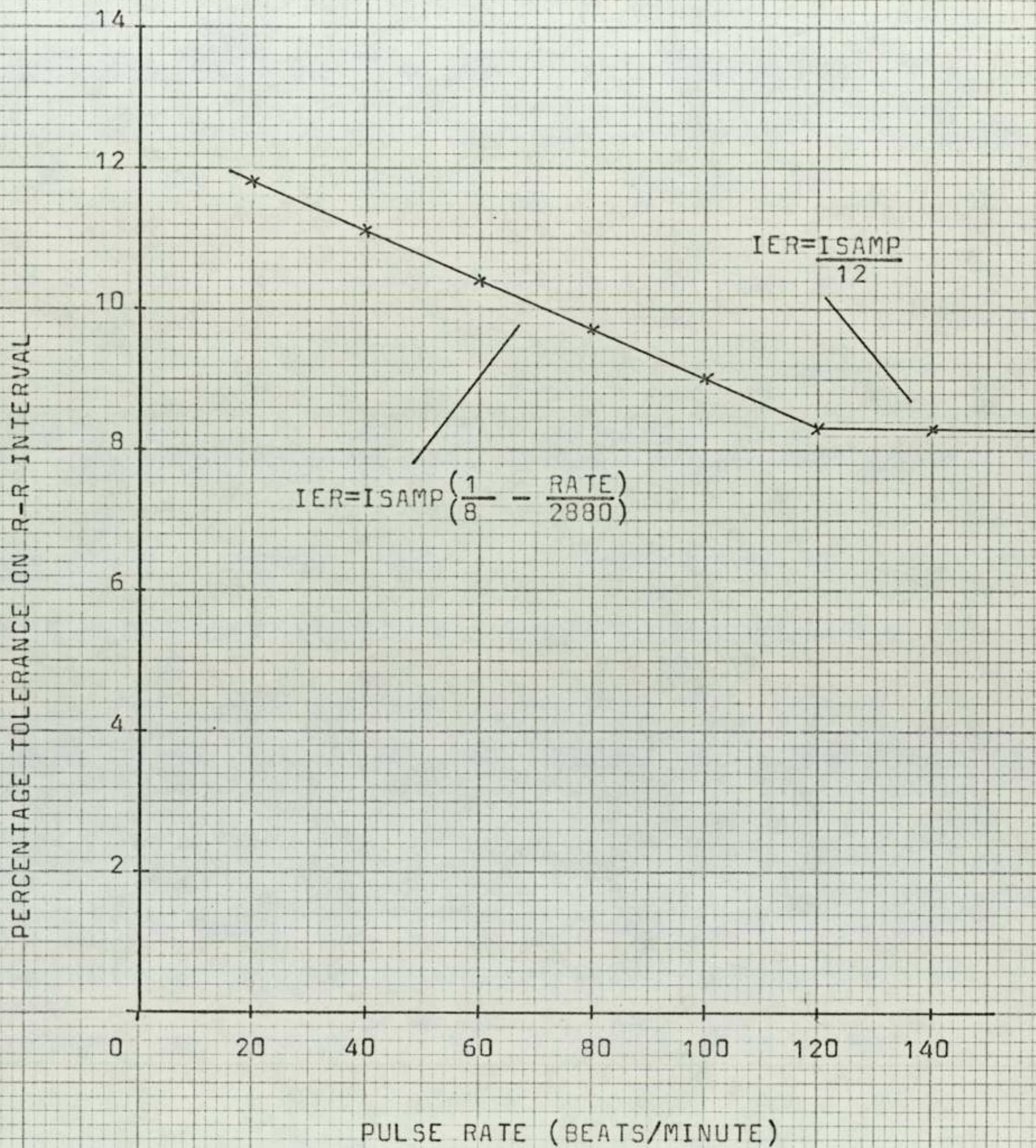


Fig 2.2. GRAPH SHOWING PERCENTAGE TOLERANCE ON R-R INTERVAL AGAINST PULSE RATE.

wide, had been specified. No indication was given as to how many beats this should be calculated from. Therefore, it was decided that this parameter would be calculated from between 200 and 400 of the most recent beats monitored. These values were chosen so that the parameter would not change too quickly nor too slowly.

The cardiologists also stated that there was no great medical significance to the value of 82%, and in fact this value could be made operator adjustable.

(3) It was indicated that the parameter of 112mS, used to classify QRS complexes as wide, could be increased to as much as 120mS, for the detection of abnormally wide QRS complexes.

(4) Doubt was also cast on the accuracy of diagnosing 23 different arrhythmias from only the QRS complex width, R-R interval duration and pulse rate. It was thought that for such a range of diagnoses, criteria relating to P waves should be included.

However, as this research project was not concerned with the development of new diagnostic criteria, it was decided that the criteria as given in Table 2.1., would be used as the initial system criteria, in order to enable the final distributed microprocessor system structure to be determined.

2.2.2. Alternative Monitoring Criteria

In an attempt to indicate an alternative set of diagnosis criteria, discussions were held with a Doctor at St. Bartholomew's Hospital in London, during the final year of research.

The approach used to produce the alternative monitoring criteria, shown in Table 2.2., was the division of the arrhythmias into two main groups. These two groups were classified as non-paroxysmal arrhythmias (i.e. arrhythmias arrived at slowly, such as a fast pulse rate reached gradually), and paroxysmal arrhythmias (i.e. arrhythmias which occur suddenly, such as a premature beat).

Within each of these groups there was then the further classification of fast (tachycardia) or slow (bradycardia) pulse rates, or early (premature) or late (pause) beats. This approach meant that the diagnoses produced were of a general nature, and in no way attempted some of the precise diagnoses given in Table 2.1.

It is not claimed here that the criteria given in Table 2.2. is better than that in Table 2.1., but simply an alternative. The field of computer monitoring diagnosis criteria, requires further medical research and detailed specification.

TABLE 2.1. INITIAL ARRHYTHMIA DIAGNOSIS CRITERIA

1. VT; Ventricular tachycardia. This diagnosis was made if the pulse rate was greater than 140, 82% or more of the QRS complexes were wide (a wide QRS was 0·112 seconds or longer in duration), and the preceding rhythm (50 beats) contained multiple PVC's (greater than 10% of the QRS complexes were PVC's).
2. SVT; Supraventricular tachycardia. For this diagnosis there must be a pulse rate greater than 110 but less than 140; or if the rate was greater than 140, less than 82% of the QRS complexes were wide.
3. BBB; Bundle Branch Block. More than 82% of QRS complexes were wide with a rate less than 140.
4. BRAD; Bradycardia. Simply any case when pulse rate was less than 60 but greater than 40.
5. IDIOVT; Idioventricular rhythm. An exceptionally slow pulse rate, less than 40 but greater than 20 was the only criteria.
6. C. ARST; Cardiac Arrest. A diagnosis of cardiac arrest was made when the heart rate was less than 80 and an R-R interval was not followed by a QRS complex for a period greater than or equal to 6 sec.
7. ASYT; Asystole. A diagnosis of asystole was made when the heart rate was less than 80 and an R-R interval was greater than or equal to 2 times the

Table 2.1. Cont...

average R-R interval but less than 6 sec. in duration.

8. ESCBT; Escape beat. A diagnosis of escape beat was made if the rate was less than 60 and a long R-R interval was followed immediately by a wide QRS.

Long (L), normal (N), and short (S) R-R intervals were defined according to whether a specific R-R interval varied from the average by more or less than an amount called IER. If it was within IER of the average, it was normal (N), more than IER greater than average it was long (L) and more than IER below average it was short (S). IER was also used in other diagnoses; it is critical in identifying a PVC or a PAC. It was set as follows : If rate > 120; IER=ISAMP/12. If rate $\leqslant 120$; IER=ISAMP(1/8-Rate/2880).

9. PVC; Premature ventricular contraction. A short (S) R-R interval, as defined in 8, followed by a wide QRS caused this diagnosis provided that less than 82% of QRS complexes were wide.

10. PVCRUN; A run of premature ventricular contractions. Two or more PVC's in succession caused this diagnosis.

11. VIBGY; A bigeminy rhythm produced by premature ventricular contractions.

The sequence of a short R-R interval (S) a wide QRS (WQRS), and then a not short R-R interval ($\bar{S}$) occurred 4 times in succession ($4(S-\bar{S})$). Note that an $\bar{S}$ R-R interval could be either normal or long. In this rhythm, it would usually be long because of

Table 2.1. Cont...

compensatory pause after a PVC.

12. VITRIGY; A trigeminy rhythm caused by premature ventricular contractions. The sequence of a short R-R interval (S) a wide QRS (WQRS) a not short R-R interval ($\overline{S}$) and a normal R-R interval (N) occurred 4 times in succession (4(S-$\overline{S}$-N)).

The next 6 diagnoses, 13 through 18, all had 3 criteria in common. These 3 criteria will be listed first :

A. Less than 82% of QRS complexes were wide.

B. No diagnosis of atrial fibrillation (AF).

C. The QRS complex following the R-R interval analysed was not wide.

13. PAC; Premature Atrial Contraction. In addition to the above 3 criteria a diagnosis of PAC was made on fulfilment of one of the following :

A. A short R-R interval (S) when the rate was greater than 90. If the rate was less than 90 the diagnosis could be made on encountering any one of the following sequences of R-R intervals.

B. S-$\overline{S}$-S; short, not short, short.

C. S-$\overline{S}$-N-$\overline{L}$; short, not short, normal, not long.

For example, a short R-R interval followed by normal R-R intervals would lead to the diagnosis.

14. PACRUN; A run of premature atrial contractions. Two or more PAC's (defined in 13) in succession

Table 2.1. Cont...

resulted in this diagnosis.

15. ABIGY; Bigeminy rhythm caused by premature atrial contractions. With the 3 criteria described prior to 13 there must be in addition a sequence of short R-R intervals followed by a not-short R-R interval ($S-\bar{S}$), which occurred 4 times in a row.
16. ATRIG; Atrial trigeminy. A trigeminy rhythm caused by contractions originating in the atrium. The sequence of a short R-R interval, a not short R-R interval, and a normal R-R interval ($S-\bar{S}-N$) must occur 4 times in succession.
17. ABCOND; Abnormal conduction. If the rate was greater than 110, a diagnosis of PACRUN had been made and the sequence of 3 short R-R intervals was followed by a wide QRS, this diagnosis was made.
18. SA; Sinus Arrhythmia. In addition to the 3 criteria described prior to 13 and a rate $\leqslant 90$, a diagnosis of SA was made on fulfilment of one of the following sequences of R-R intervals.
    A. $S-\bar{S}-L$, short, not short, long.
    B. $S-\bar{S}-N-L$, short, not short, normal, long.

    This ends the diagnoses which required the 3 criteria A-C described just before 13.
19. PC; Premature Contraction. Diagnosis of PVC or PAC with greater than 82% of QRS complexes wide would cause this diagnosis to be used rather than PVC or PAC.
20. AF; atrial fibrillation. No diagnosis of bigeminy or trigeminy rhythm. Less than 15% of QRS complexes

were PVC's. An irregularity indicator known as NODE(11) must be greater than a factor known as NODE(12). NODE(11) was the percentage of R-R intervals that differed by more than 4% from those immediately succeeding R-R intervals. The R-R intervals associated with PVC's were eliminated before making this calculation. NODE(12) was an arbitrary level described as 96-Rate/5. Thus, it was a linear function of rate which decreased as rate increased to recognise the fact that atrial fibrillation was more regular at fast rates than it was at slow rates.

Usually in atrial fibrillation 85 to 95% of R-R intervals differed by 4% or more from the next following R-R interval.

21. ACVTRT; Accelerated ventricular rate. If a diagnosis of BBB had been made and the rate was greater than 100, this diagnosis was designated instead of BBB. It was meant to indicate a more grave diagnosis than BBB. In some cases this could actually represent a ventricular tachycardia which was not diagnosed as ventricular tachycardia by the criteria, which required a pulse rate greater than 140.

22. SVTAC; Supraventricular tachycardia with abberant conduction. This diagnosis was made if the pulse rate was greater than 140, 82% or more of the QRS complexes were wide, and the preceding rhythm (50 beats) was SVT.

Table 2.1. Cont...

23. LS; Lost Signal. This diagnosis was made when the heart rate was greater than or equal to 80 and an R-R interval was not followed by a QRS complex for a period greater than or equal to 6 seconds.

TABLE 2.2. ALTERNATIVE ARRHYTHMIA DIAGNOSIS CRITERIA

NON-PAROXYSMAL ARRHYTHMIAS

(1) Non-Paroxysmal Tachycardia

This diagnosis is made when the average R-R interval (see Note 1) is less than 600mS, i.e. when the pulse rate is greater than 100 beats/minute. This diagnosis is not alarmed until the average R-R interval is less than 400mS (150 beats/minute), which should then produce a level 2 alarm (see Note 5).

(2) Non-Paroxysmal Bradycardia

This diagnosis is made when the average R-R interval exceeds 1 second, i.e. when the pulse rate falls below 60 beats/minute. This diagnosis is not alarmed until the average R-R interval exceeds 2 seconds (30 beats/minute), whereupon a level 2 alarm should be given.

PAROXYSMAL TACHYCARDIA

(3) Paroxysmal Tachycardia

This diagnosis is made when two or more short R-R intervals (see Note 2) occur in succession, and can be further characterised as having a "broad QRS" (see Note 3). This diagnosis should produce a level 2 alarm, unless the R-R interval becomes equal to or less than 300mS, (200 beats/minute) which should produce a level 1 alarm (see note 5).

(4) Premature Beat

This diagnosis is made if a short R-R interval

Table 2.2. Cont...

is followed by an interval which is not short. This diagnosis can be further specified as having a "broad QRS".

(5) Pause

This diagnosis is made if a very long R-R interval (see Note 4) occurs, and should result in a level 2 alarm being given.

(6) Asystole

This diagnosis is made if any R-R interval exceeds 5 seconds, and should result in a level 1 alarm being given.

NOTES

(1) During the monitoring start-up process, the first 16 R-R intervals detected are stored and used to calculate the initial value of the average R-R interval. Thereafter this store is continuously updated with the subsequent R-R intervals, provided that they have not been labelled as long or short (see Note 2).

(2) If an R-R interval measured, is found to be less than 85% of the average R-R interval time, it is labelled as short.

If an R-R interval measured, is found to be greater than 115% of the average R-R interval time, it is labelled as long.

Between these two limits the R-R interval is

Table 2.2. Cont...

considered to be normal, and can be used to update the average R-R interval.

(3) A QRS complex is classified as a "broad QRS" if greater than 120mS, and is measured from the points shown in Fig. 2.1.

(4) If an R-R interval is found to be greater than twice the average R-R interval time, it is labelled as very long.

(5) Alarm level 1 is the highest priority alarm and indicates an imperative diagnosed condition.
Alarm level 2 indicates diagnosed condition requiring immediate warning.

CHAPTER 3

MONITORING SYSTEMS

Since the introduction of coronary care units (CCUs) in the early 1960s, many independent groups have been developing arrhythmia monitoring systems. Initially, the only monitoring equipment used in CCUs, employed analog techniques to monitor pulse rate, which allowed very slow or fast heart rates to be alarmed.

Although equipment, such as the hybrid device described by Neilson (Ref. 6, 7, 8, 9.) which enable certain arrhythmias to be detected, and high speed mass ECG monitoring to be performed, have been developed. The need for ever increasing detailed and efficient analysis of ECG signals, has resulted in increasing efforts being made to produce computer systems capable of real-time arrhythmia monitoring.

Several different approaches, such as cross-correlation (Ref. 10-15.) and Fourier transform (Ref. 16) techniques have been employed in computer monitoring systems, but the most common approach used is that of feature or pattern recognition (Ref. 17-38.). This last approach is currently preferred as it consumes less processing time compared to other techniques, and therefore is more amenable to real-time implementation. The systems using this approach vary considerably in complexity, for example, the sample rate used by such systems ranges between 60 and

500 samples per second. The work performed by Wartak et. al. (Ref.39.) recommends that a sample rate of 250 samples per second, with a quantisation accuracy greater than 7 bits, should be used by monitoring systems.

Similarly, the approach predominantly used to detect QRS complexes; which is a delayed-difference threshold method, varies in its implementation between each system. The main variations of this approach are concerned with the time interval (number of sample periods) between the samples used to produce the difference signal, and the value of the detection threshold. The work described by Van Eyll et. al. (Ref.40.) provides the optimal values for this detection approach, which were obtained using a sample rate of 250Hz.

This detection approach is also sometimes extended to classify QRS complexes as abnormal, by monitoring the number of difference values that occur after the difference signal has passed through the detection threshold (Ref.17.). Other systems, prefer to perform direct measurements on the QRS complex to classify it as wide or abnormal. Thus the criteria used in each system to diagnose the type of arrhythmias which occur, differ with each system produced. These differences arise not only because of the different monitoring approaches adopted, but also because of the individual medical interpretations made of the parameters monitored by the particular system. There is currently no standard arrhythmia monitoring criteria, hence, each system produced employs and interprets the parameters monitored, in the manner preferred by those developing each system.

These developmental systems usually consist of one or possibly two computers such as the PDP-8 (Ref.19.), PDP-9 (Ref.22.), PDP-12 (Ref.27.), CDC 1700 (Ref.20.), NOVA 1210 (Ref.31.) and the HP 2100 (Ref.34.). Some of these developmental systems only monitor one patient (Ref.41.), but normally they are multi-patient monitoring systems, the number of patients monitored depending on the sample rate, monitoring complexity and speed of the computer used by each system.

In the past few years one of the main computer monitoring systems to appear on the market in the Hewlett-Packard HP 78220 arrhythmia monitoring system (Ref.42.). This system uses the HP 2108 processor, and costs over £40,000 for an 8 bed unit.

Since the introduction of the first microprocessor (Intel 4004) in 1971, the performance of such devices has continually improved. As their speed and sophistication increases, their use in ECG monitoring systems increases (Ref. 43-47.). These initial implementations use 8 bit microprocessors such as the Intel 8008, the Motorola M6800 and the Zilog Z-80.

The introduction of a 16 bit microprocessor the TMS 9900 (Appendix A), provided the possibility of producing a microprocessor monitoring system, which could perform real-time ECG monitoring, previously only possible with computer systems. Although the execution time of this microprocessor is still slower than those of computers used in monitoring systems, the cost of such devices would allow their distribution in a microprocessor monitoring system. In this way it was considered that any speed restriction problems would be overcome by this distributed microprocessor approach, and would also enable a more flexible and cheaper multi-patient monitoring system to be produced.

As the work by Wartak et. al. (Ref.39.) and Van Eyll et. al. (Ref.40.), mentioned above, were both concerned with a sample rate of 250Hz, their recommendations were chosen for use in the monitoring system to be developed. Similarly, the feature recognition approach was chosen for use in this system, because it is suited to real-time applications, hence the detailed criteria listed by Rabin et. al. (Ref. 5.) ~~was~~ WERE chosen, for initial use in the system, as discussed in Chapter 2.

CHAPTER 4

SYSTEM FUNCTIONS AND STRUCTURE

4.1. INTRODUCTION

The overall function of the multi-patient microprocessor system required, was that of arrhythmia detection and diagnosis. However, the system may be sub-divided into a number of distinctly different types of functions. These functions are broadly classified as follows.

4.2. System Functions

4.2.1. Signal Acquisition

The signal acquisition function is concerned with the amplification of the chest electrode potentials, to provide the ECG signals required by the system. It was decided that this function could be satisfactorily performed by the analog equipment in common "bedside" use in CCUs, which usually provide a 1 volt ECG signal, that could be used by the monitoring system.

This decision also has the advantages that the final system would be cheaper, by the exclusion of this function, and more likely to be acceptable for CCU use, as it would make greater use of existing CCU equipment.

4.2.2. Signal Conditioning

The signal conditioning function is concerned with the processing of the ECG signals applied to the system, to obtain a conditioned signal from which measurements may be taken with an appropriate degree of confidence. This is achieved by the detection of abnormal signal levels in the higher frequency components of the ECG signals, which would indicate the presence of artefact (unwanted muscle noise). Having detected the presence of noise, filtering of the ECG signal could then be performed. Also this function was to be responsible for the removal of very low frequency components, to remove d.c. and base-line drift from the signal.

As it is desirable to incorporate as many of the system functions as possible, in the more compact and flexible media of software, this function was implemented using digital filtering techniques (ref.48-53.), as described in section 5.2. The sampling rate of the ECG signal was set to 250Hz, as recommended by Wartak et. al. (ref.39.). Therefore, the signal conditioning function would have to be performed on each of these samples.

4.2.3. Signal Monitoring

The ECG signal monitoring function is concerned with the detection of each QRS complex, and the subsequent measurement and classification of features in the ECG signal which are of interest. The monitoring criteria to be initially used by the system, as described in Chapter 2, focuses attention principally on the length of the R-R interval, and the width of the QRS complex. The detection of each QRS complex is to be achieved using a delayed difference signal approach, using the optimal parameters suggested by Van Eyll et. al. (ref.40.), as described in Section 5.3. Using this approach it is necessary that the monitoring function QRS detection algorithm, be performed for each of the ECG samples.

Having detected a QRS complex, this function then performs the measurements required, and therefore this aspect of the monitoring function would have a frequency of operation dependent on the pulse rate of the monitored patients.

4.2.4. Diagnosis

Having performed the required measurements and observations by the monitoring function, it was the task of the diagnosis function to interpret the results, and provide a diagnosis. The approach used to implement this function was by the use of decision table techniques (ref.54.), as described in Section 5.4.

This function is only called upon, when the monitoring function has detected and measured the required parameters. Therefore, this function has a frequency of execution which is dependent upon the pulse rate of the patients being monitored at the time.

4.2.5. Display

The display function is concerned with the presentation of information, such as ECG data, parameter values, diagnoses and trends to the system operator. Therefore, there is the need for software to provide the data in its required form, to the appropriate system/operator interfaces, such as a VDU. This type of data would be of an intermittent nature, depending on such things as frequency of arrhythmia events.

4.2.6. Storage

The storage function is concerned with the retention of information such as trend data and stored arrhythmia events etc., which may be required for subsequent display to the system operator. This data would also be of an intermittent nature.

4.2.7. Communication and Control

In order for the system to perform as required by the operator, there is the need for a communication function to be present in the system. This function is concerned with the interpretation of input commands to the system, via a medium such as a VDU, and thus the activation of the system to perform the required task.

Also in order that the individual functions listed above should combine to produce the resulting system function of arrhythmia detection and diagnosis, there is the need for overall system control. The control function is thus the operating system (ref.56,61.) which allows the individual functions to interact to provide the system function required.

4.3. The Ideal System Function Structure

In order to determine a suitable structure for the system, it is necessary to examine the volume of information flowing between each individual function, as shown in Fig. 4.1.

As stated in section 4.2.1., only the signal acquisition function is by necessity distributed in the system, this being allocated on a one-per-patient basis. The remaining functions may therefore be distributed as required.

Since the highest rate of data transfer occurs permanently between the signal acquisition, conditioning and monitoring functions, the greatest benefits of a distributed processing network would be obtained if each of these functions were allocated on a one-per-patient basis.

If the diagnosis function was made a centralised function, the result would be a permanent flow of information, of low data rate converging on the centralised diagnosis function, from all the patient function nodes. If however, the diagnosis function was also allocated on a one-per-patient basis; it would result in the intermittent flow of information, of low data rate in the form of diagnoses, being the data sent to the centralised functions. This last configuration is the system structure which would result in the lowest volume of data flowing from the patient dedicated functions, to the centralised functions. Therefore,

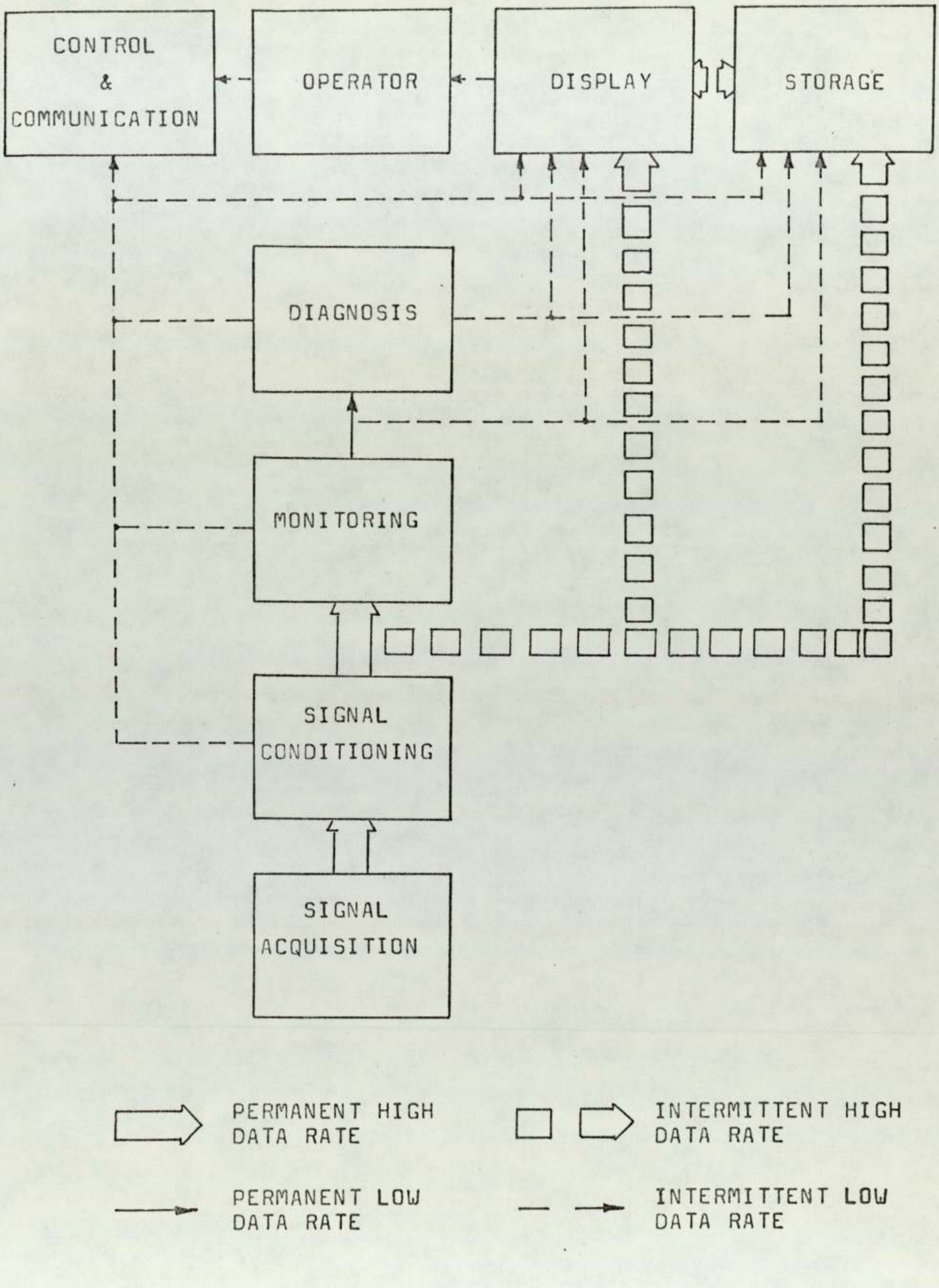


FIG. 4.1. SYSTEM FUNCTION AND INFORMATION FLOWS

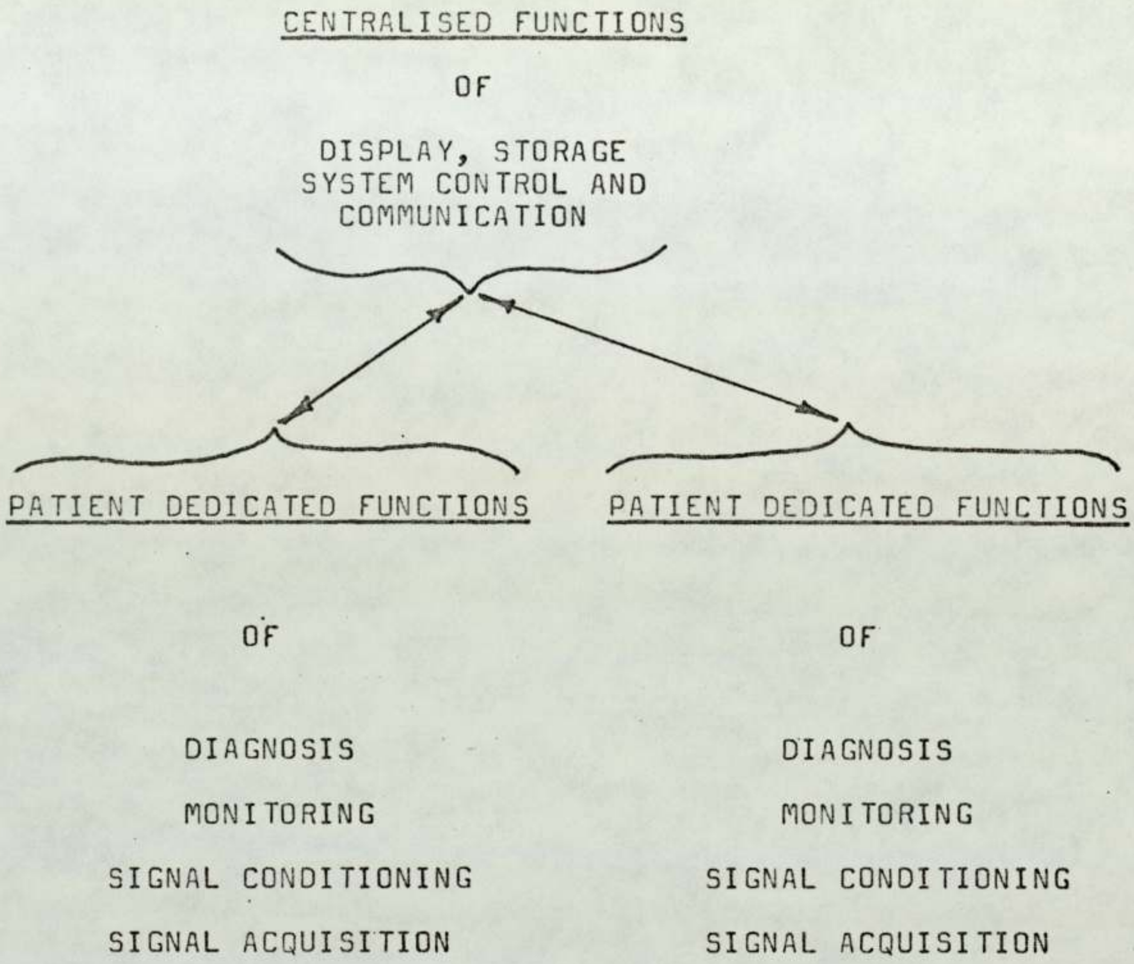
this structure, as illustrated in Fig. 4.2., was considered as being the ideal system function structure.

4.4. Performance of Functions in Real-Time

Having determined the most suitable function structure for the distributed system, the allocation of processors to functions, or vice-versa was considered. However, this procedure was not capable of simple or precise quantifications. It was therefore necessary to implement each function in turn, to determine the extent of processor utilisation in performing each function, and hence the final distribution of processors within the system.

As a result of producing the functions, which are described in detail in Chapters 5 and 6, the results as shown in Table 4.1. were obtained. From these results it was found that, for a single patient, approximately 42% of the processing time of a single TMS 9900 microprocessor is used in performing all the functions given in Table 4.1. Leaving the remaining 58% of unused processing time available for use by the storage, display and operator communication functions.

Therefore, this assessment showed that for a single patient, performance of all functions shown in Fig. 4.1. is well within the capability of a single TMS 9900 microprocessor. However, the aim was to produce a multi-patient monitoring system, with centralised facilities of system control, storage and display. These functions being centralised for ease of system management, and cost of peripheral devices.



4.2. IDEAL STRUCTURE OF DISTRIBUTED SYSTEM FUNCTIONS

FUNCTION	EXECUTION TIME	FREQUENCY OF EXECUTION	PERCENTAGE OF PROCESSING TIME CONSUMED BY FUNCTION PER SECOND
SIGNAL CONDITIONING	1 mS	250/Sec	25%
MONITORING	MIN 0·21mS	250/Sec	5·25%
	MAX 3·2mS	IF PULSE RATE=60 BPM THEN 1/Sec	0·32%
		IF PULSE RATE=180 BPM THEN 3/Sec	0·96%
DIAGNOSIS	0·2mS	IF PULSE RATE=60 BPM THEN 1/Sec	0·02%
		IF PULSE RATE=180 BPM THEN 3/Sec	0·06%
CONTROL	≃ 0·14mS	DEPENDENT ON FUNCTION ACTIVITY ≃ 750/Sec	10·5%

TABLE 4.1. FUNCTION EXECUTION TIME AND PERCENTAGE OF PROCESSING TIME CONSUMED PER SECOND

4.5. Distributed Processor System Structure

The final structure therefore, of the multi-patient distributed processing system, is the allocation of a single microprocessor to each patient. Each of these patient dedicated processors performs the functions of signal conditioning, monitoring and diagnosis, plus the functions of patient processor function control and inter-processor communication, as described in Chapters 5 and 6.

These patient dedicated processors then communicate the required information, such as arrhythmia diagnoses, to the centralised functions controlled by an operator-dedicated processor as illustrated in Fig. 4.3. Thus the ideal system function structure is in fact the final solution.

4.6. Conclusions

Having distributed the system as shown in Fig. 4.3., the additional advantage that has been gained, apart from those previously mentioned, is that of surplus processing time available at each of the patient dedicated processor nodes. This is because, the only demand made on the available surplus processing time of 58% at each patient processing node, is that made by inter-processor communication. Therefore, there is the availability of improving and expanding any of the patient dedicated functions.

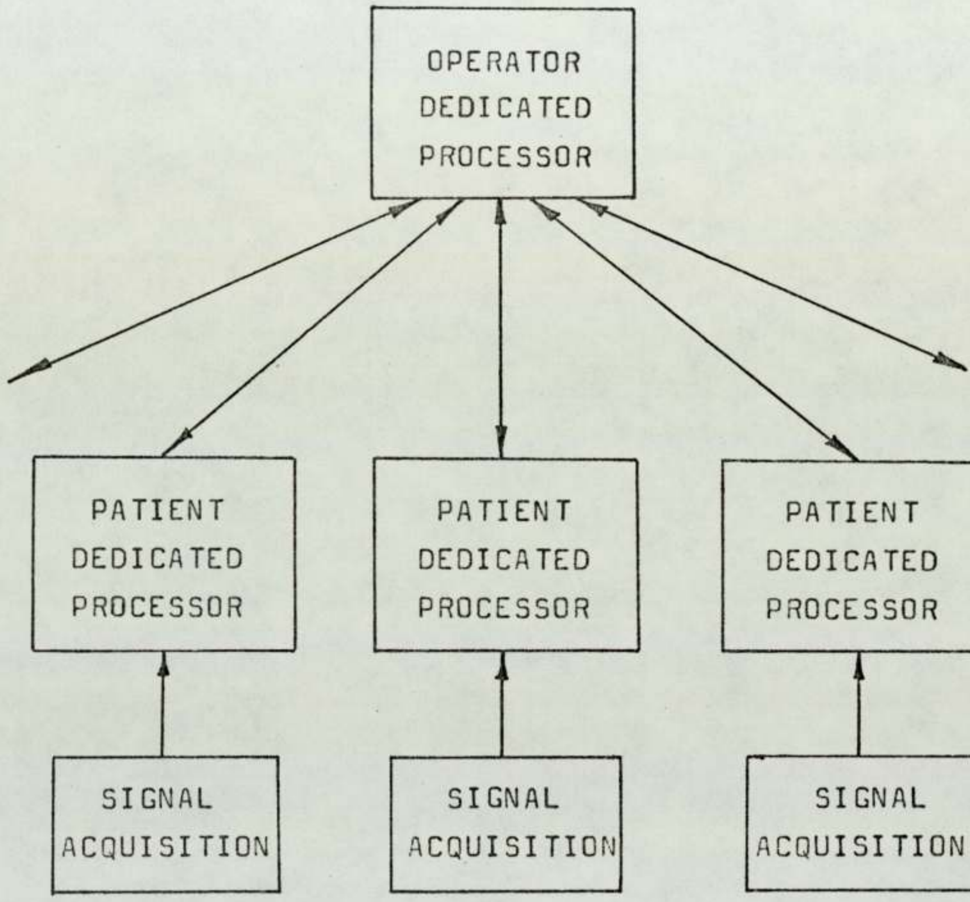


FIG. 4.3. STRUCTURE OF DISTRIBUTED PROCESSOR SYSTEM

CHAPTER 5

PATIENT-DEDICATED PROCESSOR SYSTEM DESIGN

5.1. INTRODUCTION

Each patient-dedicated processor system performs the functions of signal conditioning, monitoring, diagnosis and inter-processor communication, under the control of an operating system as shown in Fig 5.1. As can be seen from this diagram, the communication function consists of the three tasks of input, output and interpreter (microprocessor communication package), and the operating system is divided into the four modules of system initialisation, real-time task scheduler, ADC interrupt handler and UART interrupt handler.

In order to understand this system structure, let us consider each of the system tasks in turn, and hence the resulting control provided by the patient-dedicated operating system.

5.2. Signal Conditioning

The purpose of the signal conditioning function developed, is to improve the consistency of subsequent signal monitoring, by the removal of very low frequency components (d.c, and base-line drift), and the reduction, when present, of intermittent high frequency components (artefact), from the ECG signal.

A sketch of the normal ECG spectrum, as obtained by Golden et. al. (Ref.57.), is shown in Fig. 5.2. From

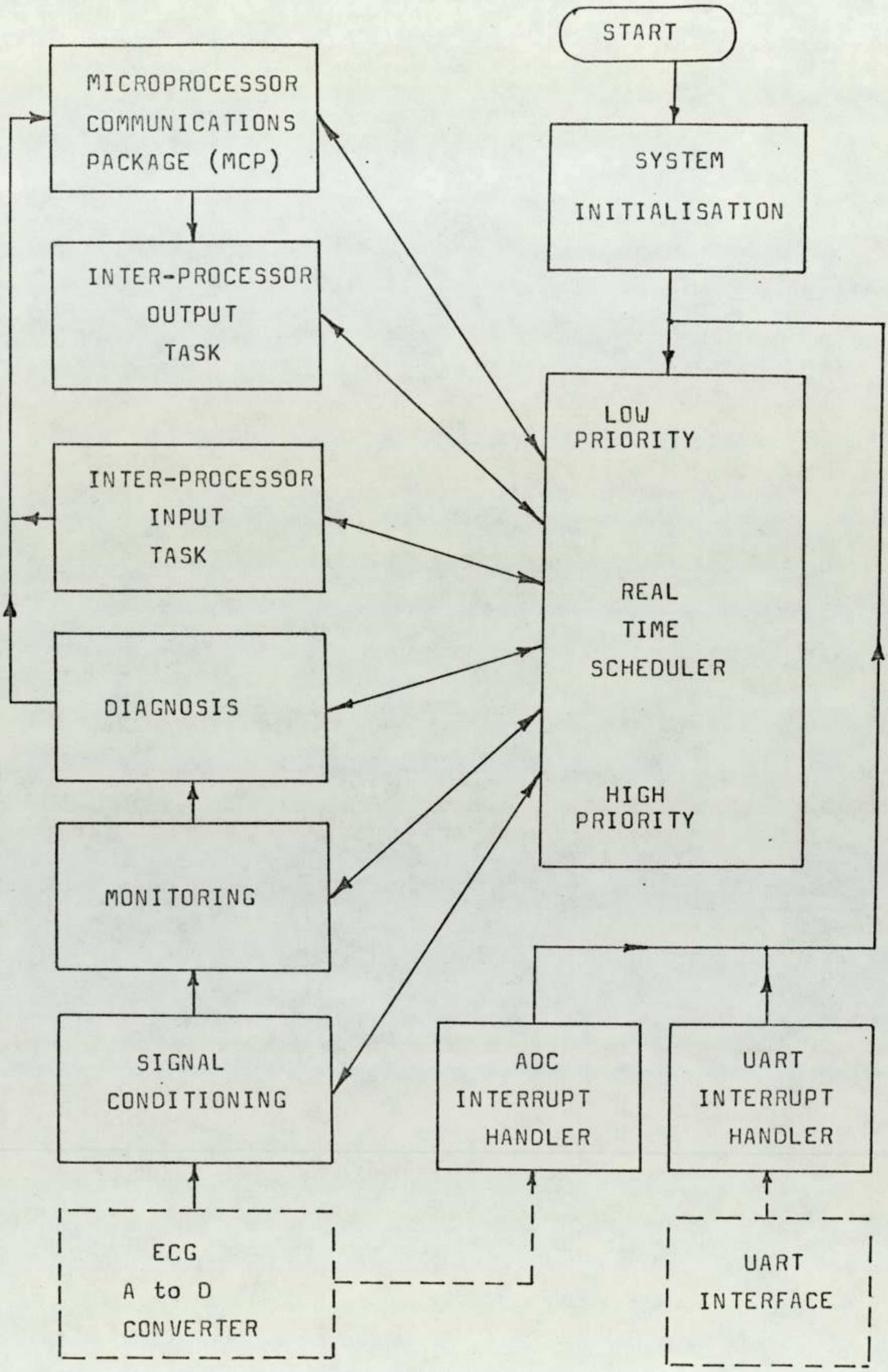


Fig. 5.1. SOFTWARE STRUCTURE OF PATIENT-DEDICATED PROCESSOR

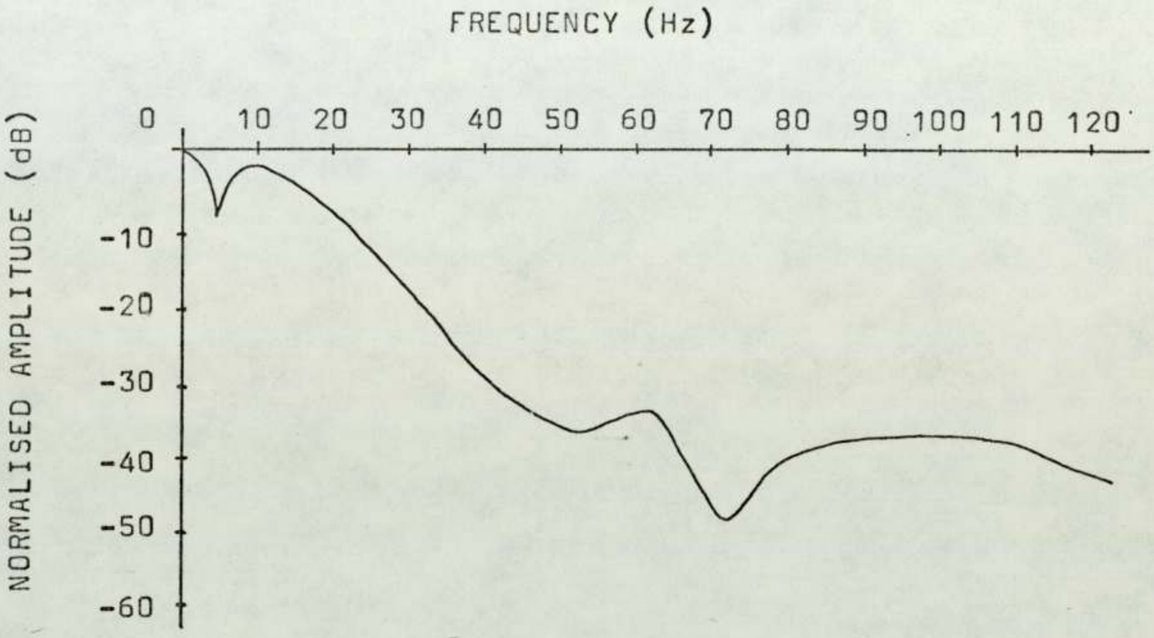


Fig. 5.2. SKETCH OF ECG SPECTRUM

this sketch it can be seen that the predominant portion of the ECG spectrum is below 60Hz.

The required signal conditioning is accomplished by an adaptive digital filtering algorithm, applied to the sampled ECG signal. The filtering algorithm developed contains three digital filters, the first of which is a high-pass filter with a very low cut-off frequency, used to remove the d.c. and base-line drift frequency components from the ECG signal. The remaining two digital filters perform the adaptive aspect of the filtering algorithm, concerned with the detection and reduction of artefact frequency components. The first of these two filters is a high pass filter used to monitor the frequency spectrum above the predominant portion of the ECG spectrum (Fig. 5.2.). It is this filter which enables artefact to be detected. The remaining filter is a low-pass filter, used to reduce the level of artefact in the ECG signal, when it has been detected. All of the filter cut-off frequencies were determined experimentally as described shortly.

As speed of operation is an important factor in the design of each function, it was decided that recursive digital filter techniques would be used, as this would reduce the number of time consuming multiplications to be performed by the algorithm.

The design of the digital filters used, was accomplished by transforming continuous Butterworth filters, into their discrete form using the Bilinear Transformation approach

(Ref. 48, 49, 50, 51, 52.). This approach states that continuous filter transfer functions in the form H(s), can be transformed into their discrete form of H(z) via the relationship

$$s \longrightarrow \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \qquad (5.1.)$$

However this transformation produces the effect known as "frequency warping" (Ref.49.) which results in the relationship between the analog filter cut-off frequency (w_A) and the resulting digital filter cut-off frequency (w_D) of

$$w_A = \frac{2}{T} \tan \frac{w_D T}{2} \qquad (5.2.)$$

As stated previously, the analog filter that was to be transformed by this approach was the Butterworth filter, which was chosen because it has a relatively simple transfer function that produces a maximally flat frequency response. A second order low-pass Butterworth filter has a transfer function given by

$$H_{LP}(s) = \frac{s_1 s_2}{s^2 + \sqrt{2}s + 1} \qquad (5.3.)$$

and a high-pass transfer function of the general form

$$H_{HP}(s) = \frac{s^2}{s^2 + \sqrt{2}s + 1} \qquad (5.4.)$$

Using the Bilinear Transformation equation 5.1., and the above equations provides the discrete forms of the filter transfer functions of

$$H_{LP}(z) = \frac{C_{LP}(1 + 2z^{-1} + z^{-2})}{1 + K_1 z^{-1} + K_2 z^{-2}} \qquad (5.5.)$$

and $$H_{HP}(z) = \frac{C_{HP}(1 - 2z^{-1} + z^{-2})}{1 + K_1 z^{-1} + K_2 z^{-2}} \qquad (5.6.)$$

where $$C_{LP} = \frac{w_A^2}{1 + \sqrt{2}w_A + w_A^2} \qquad (5.7.)$$

$$C_{HP} = \frac{1}{1 + \sqrt{2}w_A + w_A^2} \qquad (5.8.)$$

$$K_1 = \frac{2w_A^2 - 2}{1 + \sqrt{2}w_A + w_A^2} \qquad (5.9.)$$

$$K_2 = \frac{1 - \sqrt{2}w_A + w_A^2}{1 + \sqrt{2}w_A + w_A^2} \qquad (5.10.)$$

Using the Direct Form of block diagram construction (Ref.53.) on equations 5.5. and 5.6., it was possible to produce block diagrams of the digital filters (Fig. 5.3.) from which the software for the filters was produced.

The above theory was then used to produce filter coefficients for a range of filter cut-off frequencies, using the recommendations of Wartak et. al. (Ref.39.), that the ECG signal should be sampled at a rate of 250Hz (T=4mS) with 8 bits quantization resolution.

From the experimental work with these filters, it was found that the d.c. eliminating filter produced computational values which were so large that they could not be handled speedily by the processor. Hence, another form of the discrete transfer function was obtained as follows.

A high-pass continuous Butterworth filter, with a cut-off frequency of w_A, is given by

$$H_{HP}(s) = \frac{s^2}{s^2 + \sqrt{2}sw_A + w_A^2} \qquad (5.11.)$$

Now let the sampling frequency be rw_A i.e.,

$$T = \frac{1}{rw_A} \qquad (5.12.)$$

Substituting equation 5.12. into 5.1. gives,

$$s \longrightarrow 2rw_A \left(\frac{z-1}{z+1}\right) \qquad (5.13.)$$

Substituting equation 5.13. into 5.11. gives,

$$H_{HP}(z) = \frac{4r^2 w_A{}^2 \left(\frac{z-1}{z+1}\right)^2}{4r^2 w_A{}^2 \left(\frac{z-1}{z+1}\right)^2 + \sqrt{2}rw_A{}^2 2\left(\frac{z-1}{z+1}\right) + w_A{}^2} \qquad (5.14.)$$

which reduces to

$$H_{HP}(z) = \frac{\dfrac{4r^2}{4r^2 + 2\sqrt{2}r + 1}}{1 + \left(\dfrac{4r^2}{4r^2+2\sqrt{2}r+1}\right)\left(\dfrac{\sqrt{2}}{r(z-1)} + \dfrac{z}{(z-1)^2 r^2}\right)} \quad (5.15.)$$

Equation 5.15. is of the form

$$H(z) = \frac{K(z)}{1 + K(z)G(z)} \quad (5.16.)$$

which is of a standard feedback configuration, as shown in Fig. 5.4.

Using this alternative digital filter block diagram, the problem of large computational values was overcome, because the accumulations in the feedback path of the filter contain values of similar magnitude to that of the input to the filter.

The resulting filter produced using the alternative filtering approach was finally set to a cut-off frequency of 0·5Hz, with the resulting filter coefficients converted to 12 bits fractional accuracy as shown in Table 5.1.

In order to determine a suitable cut-off frequency for the filter to be used in the artefact detection section of the filtering algorithm, experiments were performed using a number of different cut-off frequencies and artefact detection threshold levels. From these experiments, it was observed that a filter cut-off

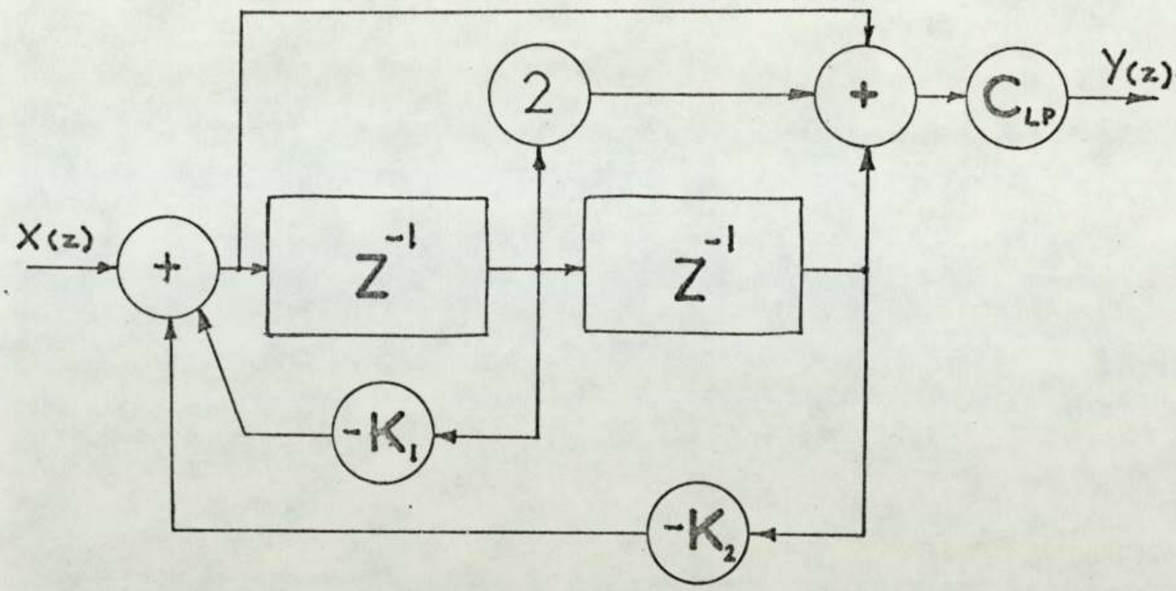


Fig. 5.3. LOW PASS DIGITAL FILTER BLOCK DIAGRAM

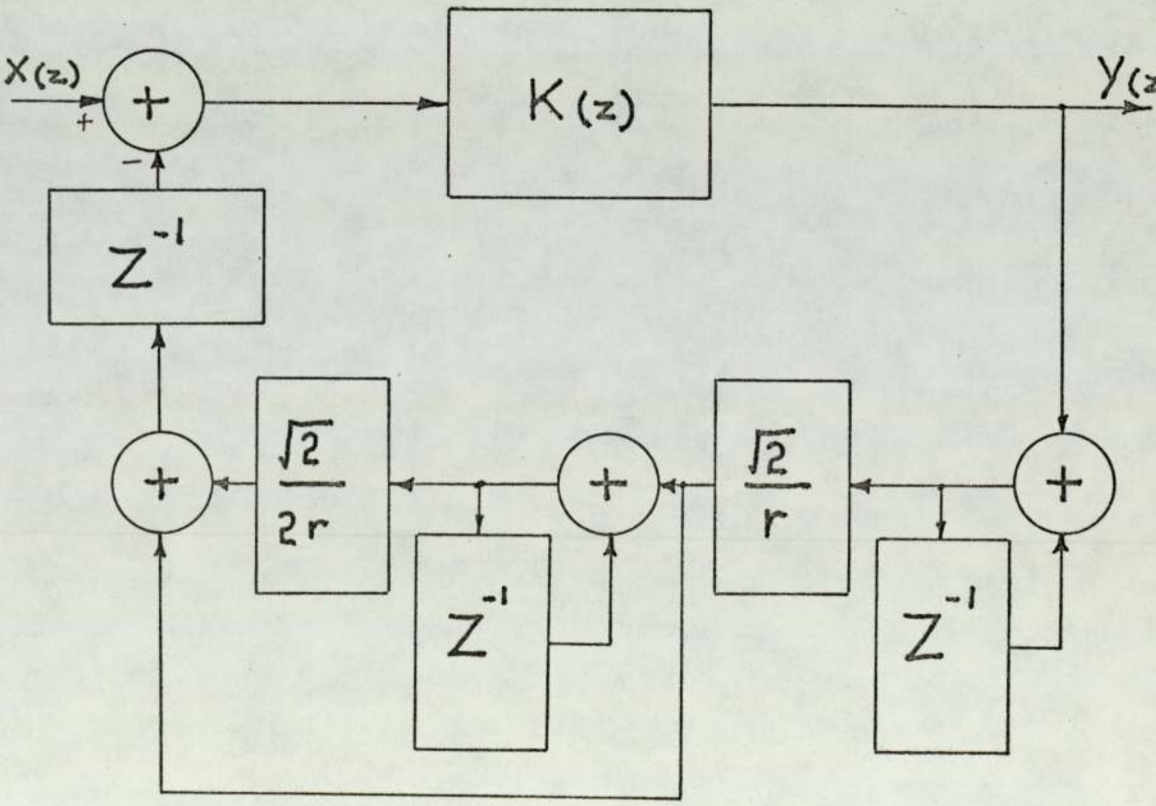


Fig. 5.4. STANDARD FEEDBACK CONFIGURATION OF

DIGITAL FILTER

frequency of 60Hz combined with a noise threshold level of 26 quantization units, produced an arrangement whereby artefact was detected, but the high frequency ECG components due to QRS complexes were not. As shown in Fig. 5.5. The resulting artefact detection filter was produced using the standard feedback configuration (Fig. 5.4.) approach as this allowed the filter coefficients to be converted to 12 bits fractional accuracy as shown in Table 5.1.

Two approaches were then adopted to determine a suitable cut-off frequency for the artefact reduction filter. The first of these approaches was a theoretical attempt to determine to what extent the QRS complex widths would be affected by low pass filtering. This was achieved by producing a program which performed subsequently lower cut-off frequency filtering on a stored example of a QRS complex. Each resulting filtered QRS complex was then measured by the program, and the percentage change in QRS width calculated. Also, in an attempt to vary the stored example of a QRS complex, this data was perturbated about the reference axis, in order to vary the stored QRS complex width. The resulting limits of QRS complex width variation, as produced by the program, are shown in Fig. 5.6., and as can be seen, below a 20Hz cut-off frequency, the percentage change in QRS complex width increases rapidly. Also, it is noted that not only can

D.C. FILTER COEFFICIENTS (CUT OFF FREQUENCY 0•5Hz)

$K(z)$	=	0•99115	=	$0FDB_{16}$
$\sqrt{2}/r$	=	0•01777	=	0048_{16}
$\sqrt{2}/2r$	=	0•00888	=	0024_{16}

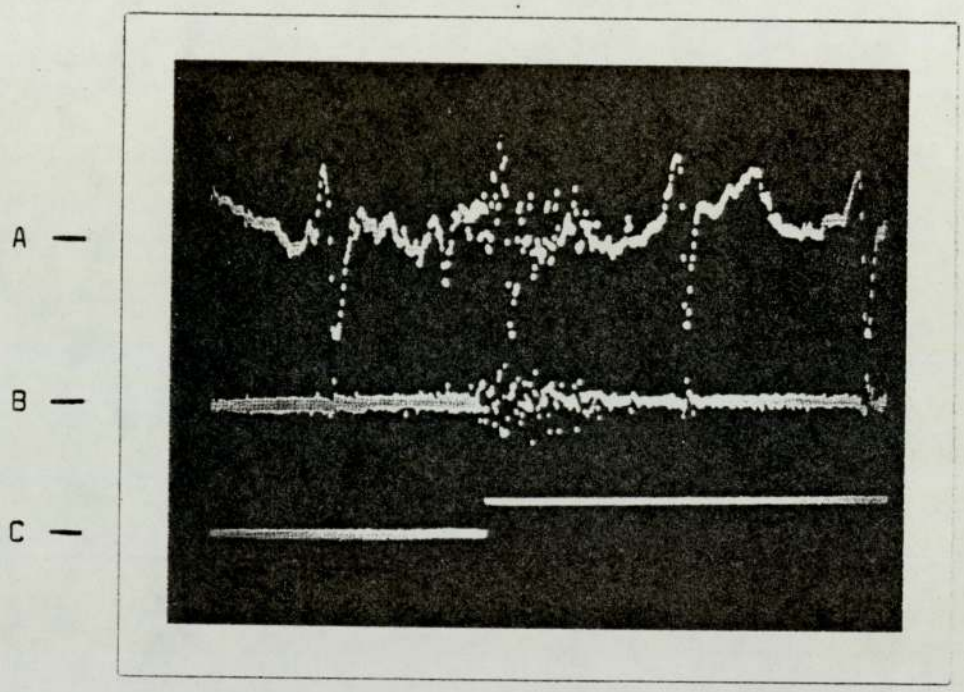
ARTEFACT DETECTION FILTER COEFFICIENTS (60Hz)

$K(z)$	=	0•311538	=	$04FC_{16}$
$\sqrt{2}/r$	=	2•656069	=	$2A7F_{16}$
$\sqrt{2}/2r$	=	1•328034	=	$153F_{16}$

ARTEFACT REDUCTION FILTER COEFFICIENTS (25Hz)

C_{LP}	=	0•067455	=	0011_{16}
K_1	=	1•142983	=	0124_{16}
K_2	=	-0•41280	=	$FF97_{16}$

TABLE 5.1. FILTER COEFFICIENTS



A SAMPLED ECG SIGNAL

B OUTPUT OF 60Hz HIGH-PASS FILTER

C NOISE DETECTION INDICATOR

Fig. 5.5. OUTPUT FROM HIGH PASS FILTER

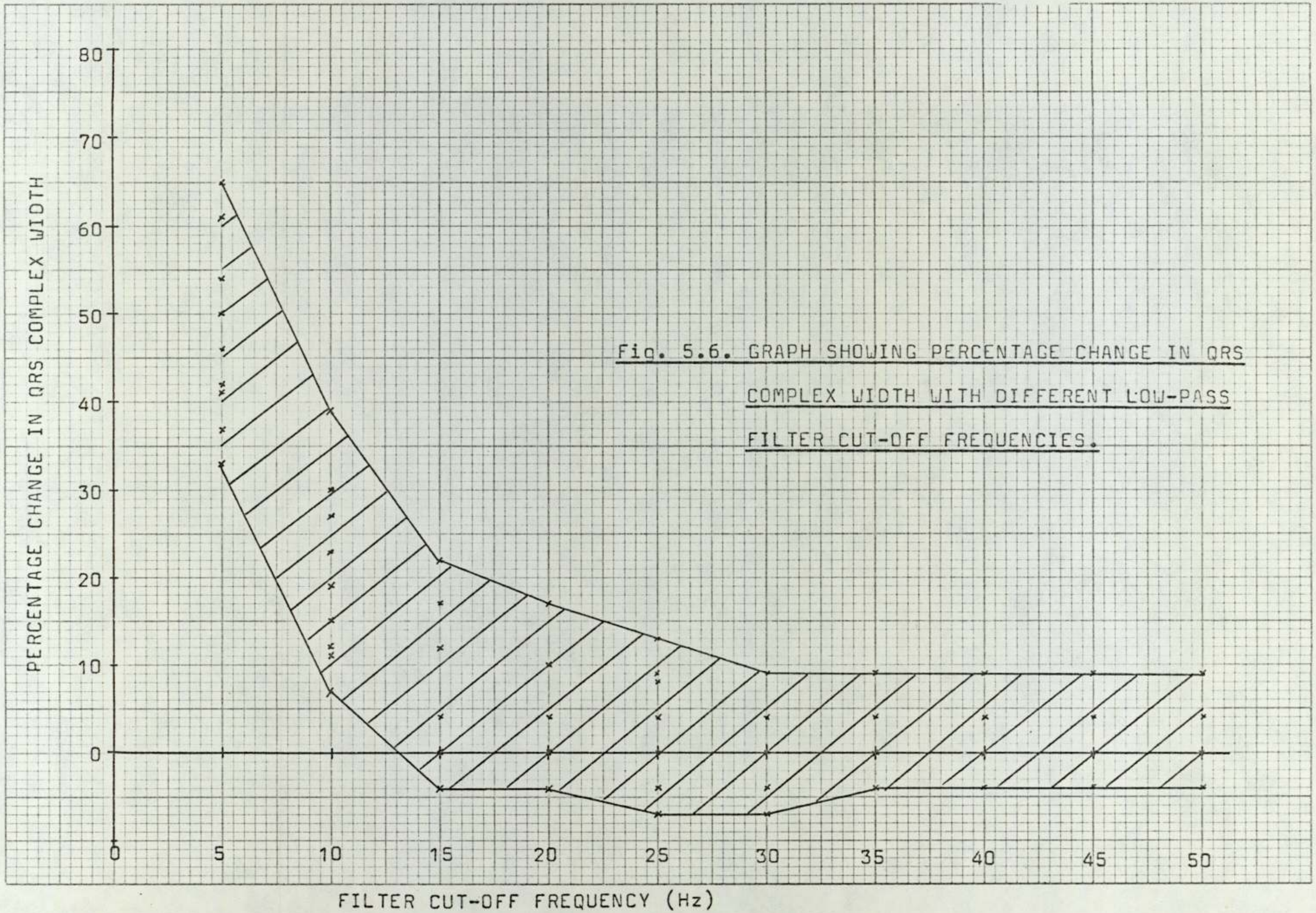


Fig. 5.6. GRAPH SHOWING PERCENTAGE CHANGE IN QRS COMPLEX WIDTH WITH DIFFERENT LOW-PASS FILTER CUT-OFF FREQUENCIES.

filtering increase the measured widths of QRS complexes, but as indicated by the occurrence of negative percentages, QRS complex widths measured can be reduced. This effect is due to the size of the samples at the extremes of the QRS complex. At these points the small sample values can easily change in sign as well as in amplitude, resulting in occasions where fewer sample periods are counted between the QRS width measurement points.

The second approach used to determine a suitable cut-off frequency for the artefact reduction filter, was the direct observation of the effect of reducing the filter cut-off frequency on actual ECG data. While performing this experiment observations were made of the effect of filtering on the QRS width, and the extent to which artefact was visibly reduced by each cut-off frequency.

As a result of this work, a cut-off frequency of 25Hz was chosen for the artefact reduction filter, because it provided good artefact level reduction, while not excessively increasing the QRS complex width. This particular filter was produced using the Direct Form of block diagram construction (Fig. 5.3.) and as a result of the larger computational values produced by such a filter, its coefficients were converted to 8 bits fractional accuracy, as shown in Table 5.1.

The resulting frequency characteristics of the three filters used in the filtering algorithm are shown in Fig. 5.7. Combining these filters in the

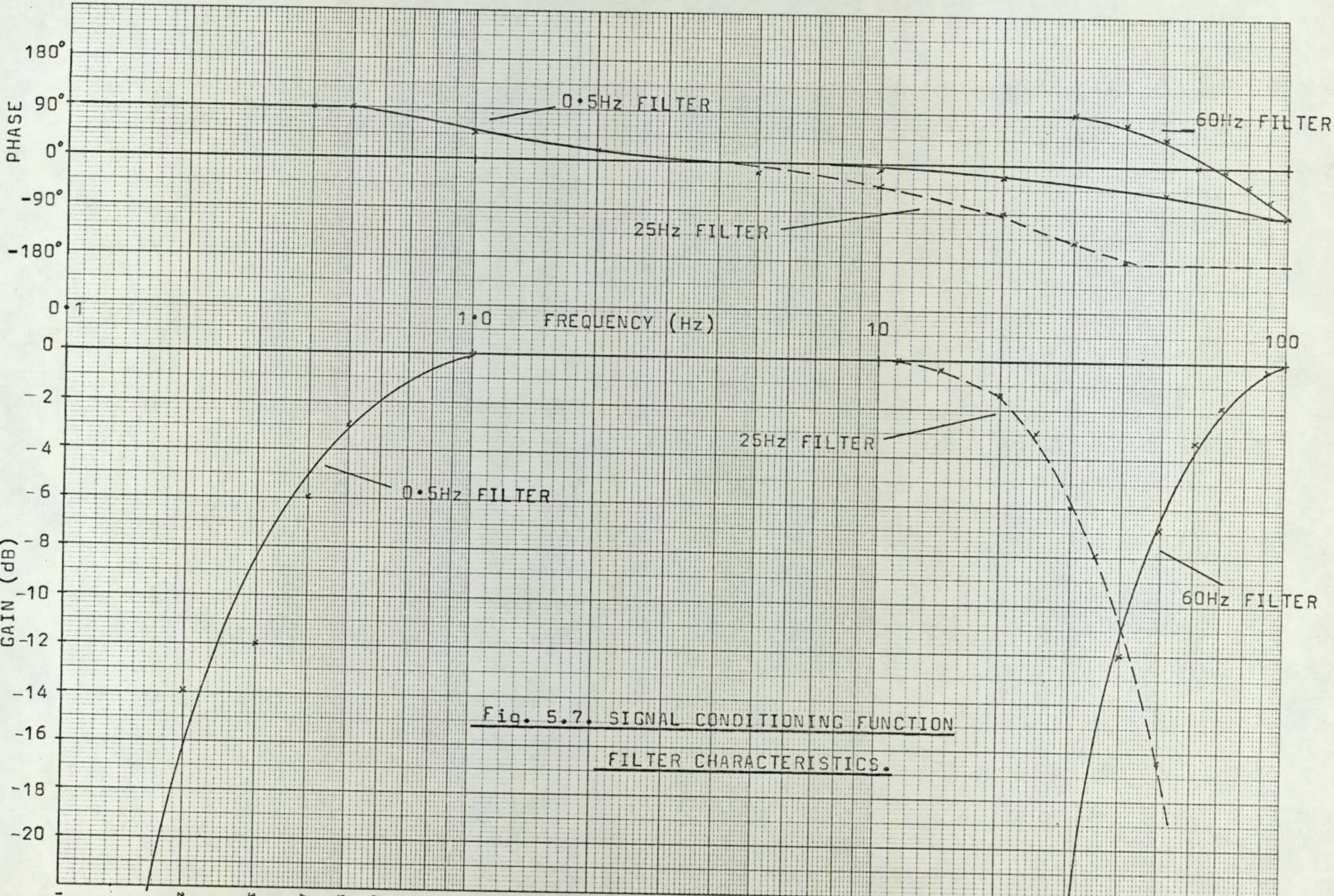


Fig. 5.7. SIGNAL CONDITIONING FUNCTION FILTER CHARACTERISTICS.

manner shown in Fig. 5.8. produces the required signal conditioning function. The delays of N and n sample periods, shown in Fig. 5.8., are chosen to equalise the total delays through the 25Hz low-pass filter and the direct path, and also to ensure that the filtered signal is introduced just prior to the occurrence of the artefact to be reduced. The delayed hold is introduced into the system, to ensure that the low-pass filter remains in continuous operation when rapid bursts of artefact are present in the ECG signal. Some examples of conditioned ECG signals are given in Fig. 5.9.

The execution time of this signal conditioning function software is lmS, which is well within the 4mS sampling period. The assembly program listing of this program can be found in the software appendix, under the program identifier IPDHTP.

5.3. Signal Monitoring

The first task of the monitoring function developed is the detection of each QRS complex. This is achieved using the optimal parameters suggested by Van Eyll et. al. (Ref.40.) for an on-line algorithm monitoring the QRS complex. This algorithm accomplishes the detection of the QRS complex by a delayed-difference threshold method. A signal is obtained by calculating the difference between samples of the ECG signal, which are separated by 6 sample periods (24mS). This signal is

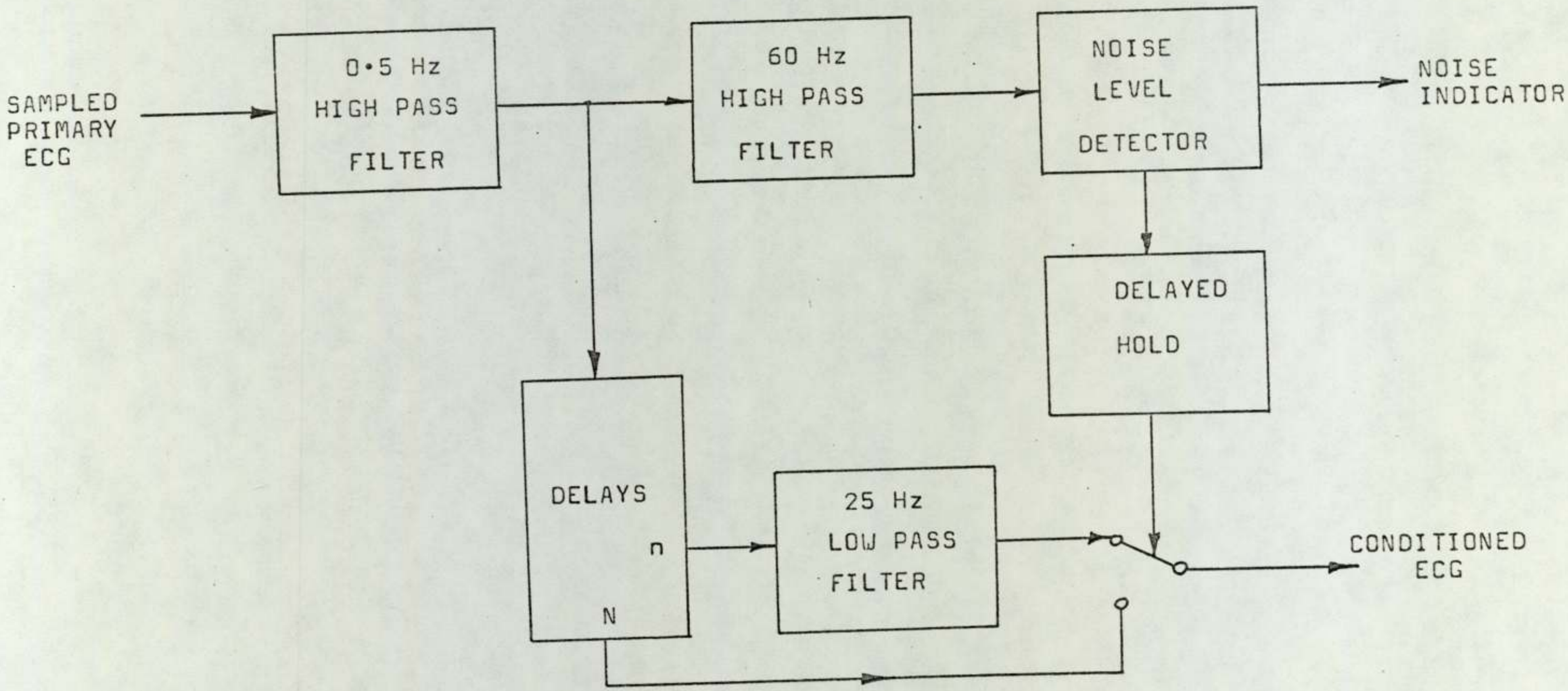
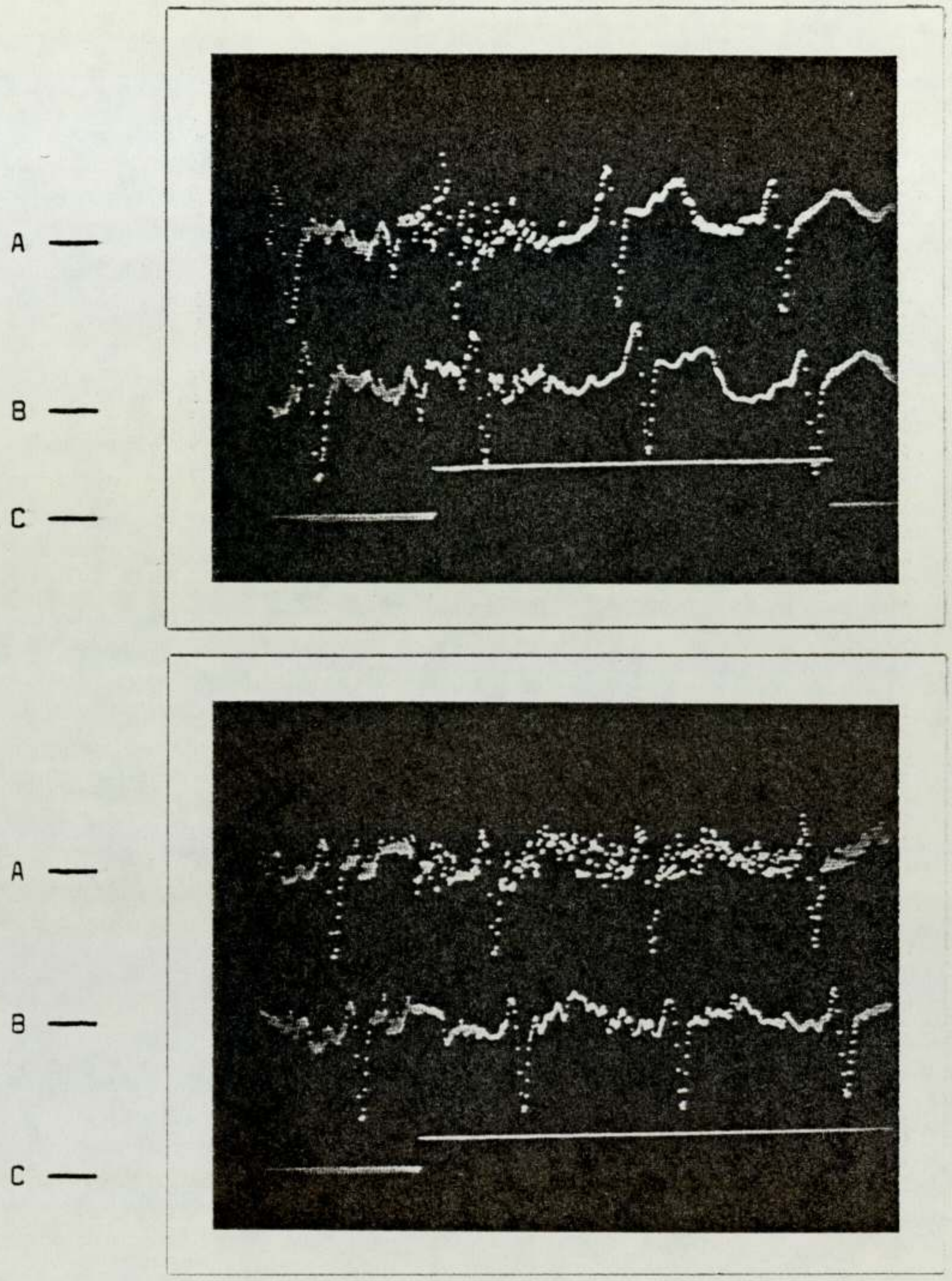


Fig. 5.8. BLOCK DIAGRAM OF ADAPTIVE FILTER



A SAMPLED ECG SIGNAL

B CONDITIONED ECG SIGNAL

C NOISE DETECTION INDICATOR

Fig. 5.9. EXAMPLES OF CONDITIONED ECG SIGNALS

then compared with a threshold (TD) which is 60% of the average minimal negative value (DMIN) of the delayed difference signal (Fig. 5.10.). When the delayed difference signal passes through this threshold the QRS complex is detected.

This detection method is adaptive, because the detection threshold (TD) is calculated from the average minimal negative value. The parameters used to calculate the average minimal negative value have also been optimised by Van Eyll et. al. (Ref.40.) as follows.

Every time a QRS is classified as normal the average minimal negative value is recalculated as :-

$$DMIN(NEW) = DMIN(OLD) \times 0{\cdot}8 + 0{\cdot}2 \times DMIN(CURRENT) \quad (5.17.)$$

and therefore the threshold (TD) is recalculated as

$$TD = 0{\cdot}6 \times DMIN(NEW) \quad (5.18.)$$

The approach by Van Eyll uses the number of samples below the detection threshold (TD) to classify the QRS complex as normal or abnormal, and hence whether or not to update the average minimal negative value. However, the approach adopted for the microprocessor system was that of updating the average minimal negative value only when the QRS width measurement algorithm (to be described), classified the QRS width as normal.

This detection method was adopted, because it is a method in common use in computer monitoring systems,

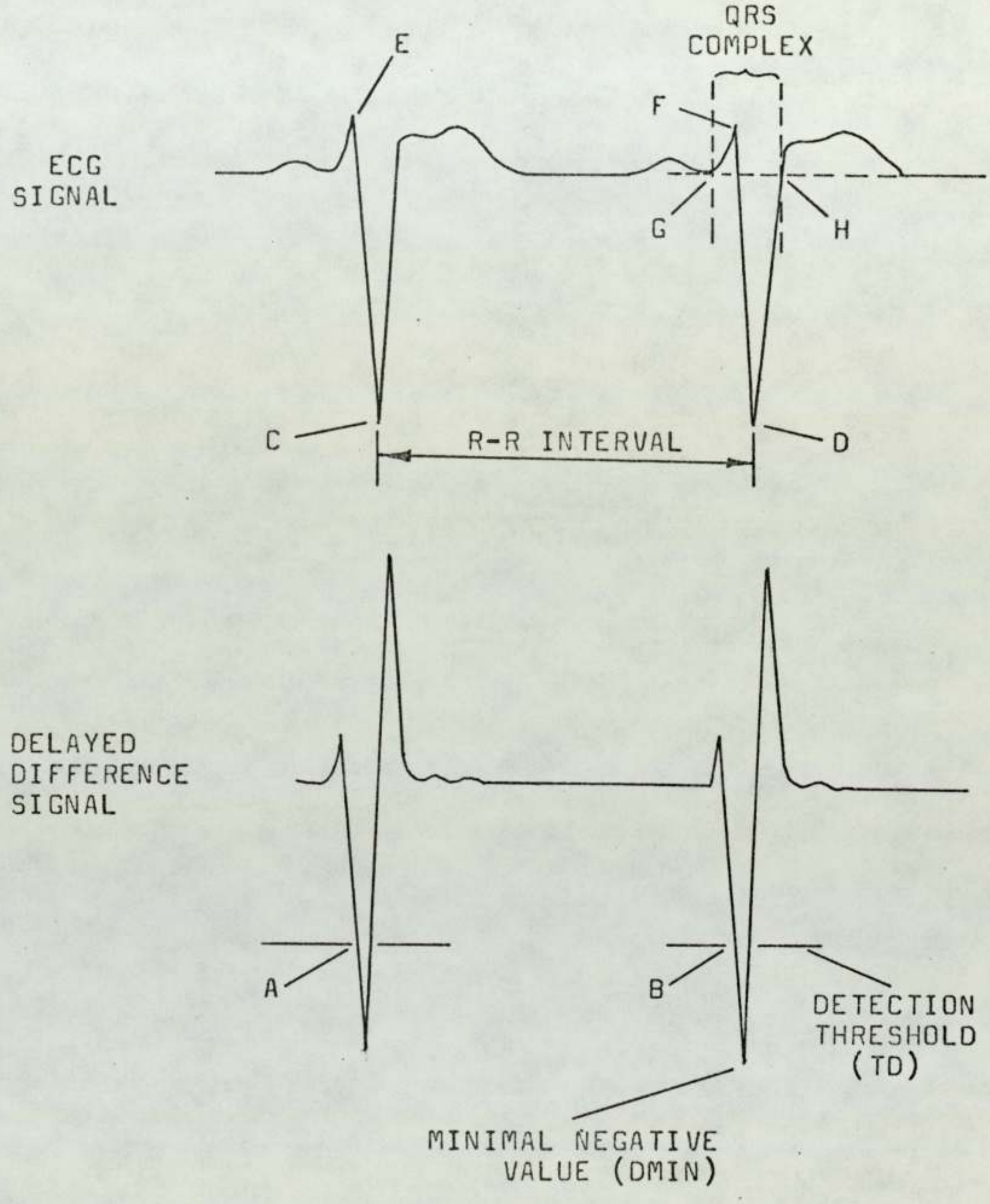


Fig. 5.10. DELAYED-DIFFERENCE DETECTION OF QRS COMPLEX

as it is relatively simple and therefore consumes as little processing time as possible. Also, the optimal values suggested by Van Eyll were developed for a sampling rate of 250Hz, which is that used by the signal conditioning function.

When the patient processor system is instructed to start the monitoring function, the first 2 seconds of ECG signal are used purely to determine a minimal negative value, which will be used initially by the detection software. This value, when found, is compared with a reference value to determine if it is large enough to be used as a minimal negative value. If the minimal negative value found, is too small, the monitoring function is stopped and a message, indicating that there has been a monitoring function start-up failure, is sent to the operator dedicated processor. If, however, the minimal negative value found is an allowable value, the detection threshold (TD) is calculated as 60% of this value, and a message is sent to the operator processor, indicating that the monitoring start-up procedure was successful.

Two consecutive QRS complexes are then detected and the R-R interval between them measured. This value is then used as the initial value for the average R-R interval (AVRR). Thereafter, any R-R interval which is classified as normal is used to up-date the average R-R interval.

After this initial starting-up process, the monitoring function will then perform the monitoring measurements and observations required. As indicated in section 2.5., two slightly different sets of monitoring criteria were eventually to be implemented. The first set of criteria used was that suggested by Rabin et. al. (Ref. 5.) , as given in Table 2.1.

5.3.1. R-R Interval Measurement

The measurement of the R-R interval as illustrated in Fig. 5.10., is defined as the time between the two negative peaks of successive QRS complexes. The monitoring function measures this interval by first counting the number of sample periods received (RRC), between the points where the delayed difference signal first passes through the detection threshold (points labelled A and B in Fig. 5.10.). This first measurement is only an estimate of the R-R interval, as points A and B in Fig. 5.10. do not correspond to the true measurement points C and D.

As will be described shortly, during the measurement of the QRS complex width, the position and amplitude of the minimum and maximum values in the QRS complex (C,D,E and F) are easily located. Therefore, the differences that exist between the positions A and C, and B and D can be calculated as each beat occurs. These parameters are defined as follows :-

$$\text{OFFSET OLD} = \text{OFSETO} = A - C \qquad (5.19.)$$

$$\text{OFFSET NEW} = \text{OFSETN} = B - D \qquad (5.20.)$$

Therefore, to calculate the true R-R interval, the following calculation is performed.

$$\text{TRUE R-R} = \text{RRC} + \text{OFSETO} - \text{OFSETN} \qquad (5.21)$$

As each new beat occurs, the previous OFSETN value becomes the OFSETO value.

This approach was adopted because the parameter RRC was available as it was required to perform the monitoring requirements that,

(1) If RRC $<$ 32 sample periods, no QRS complexes may be detected.

(2) If RRC $>$ Maximum R-R limit (such as 6 seconds = 1500 sample periods) is an alarm condition,

and also that the parameter OFSETN is easily obtained once the QRS width measurement is performed. Performing the calculation given by equation 5.21. is then quicker than a completely separate measurement of the R-R interval by additional software using the stored ECG signal, plus the fact that the ECG storage buffer would have to be long enough (6 seconds = 1500 bytes) to accommodate the longest allowable R-R interval.

5.3.2. QRS Complex Width Measurement

The measurement of the QRS complex width, as illustrated in Fig. 5.10., was initially defined as the interval between the zero cross over points indicated by G and H. However, it was found that in some of the recorded ECGs, the signal would approach these points but remain just above or below the cross-over point for a few extra sample periods. This resulted in a number of normal QRS complexes being labelled as wide. In an attempt to stop this situation, small thresholds were set at points G and H as shown in Fig. 5.11., to replace the zero crossover reference points. These thresholds were set to +5 quantization units for the positive threshold, and -5 units for the negative threshold, which in each case corresponds to approximately a 4% shift from the zero reference measurement points. The result of this new threshold approach was that QRS complexes previously incorrectly classified as wide were classified as normal. At the present time these threshold values are fixed, but if it was desirable they could easily be made operator adjustable parameters.

When a QRS complex has been detected, the monitoring function waits until the ECG signal passes through zero again, or until a counter estimating the QRS complex width exceeds the value for a wide complex. The algorithm which then measures the QRS complex width, functions as follows. First the position and amplitude of the minimum (MINP and MINV) and maximum (MAXP and MAXV) values within the QRS complex are found (Fig. 5.11.). A count is then made of the number of sample periods between MINP and the negative threshold, and similarly from MINP until the sample values pass down through the positive threshold. As this count is generated it is compared with the maximum count

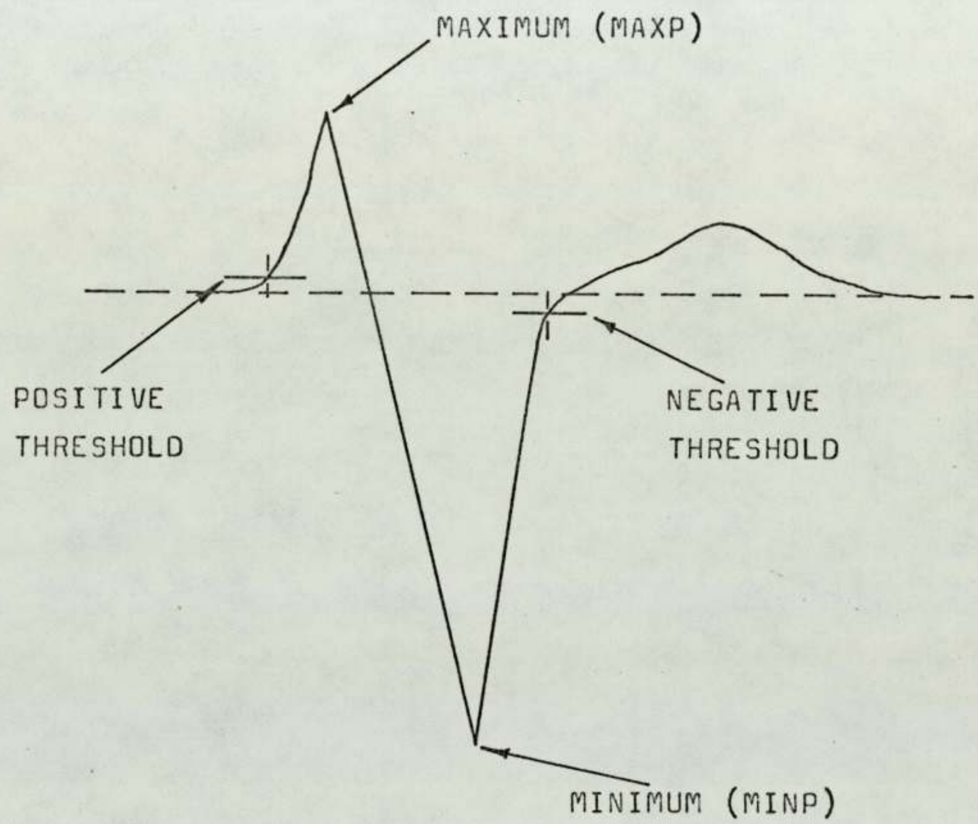


Fig. 5.11. THRESHOLD MEASUREMENT OF QRS COMPLEX WIDTH

required to classify the complex as wide (112mS=28). If this value is exceeded the width measurement algorithm finishes and labels the complex as wide.

5.3.3. Wide QRS Percentage Calculation

Having classified the current QRS complex as normal or wide, the next operation performed by the monitoring function is to determine the percentage of wide QRS complexes. No definition of how this should be calculated was given in the criteria given by Rabin et. al., therefore the approach adopted for the microprocessor system is as follows. In order that this percentage should be a reasonably sensitive parameter it was decided that it should be calculated from 200 to 400 of the most recent beats. This was accomplished by using two pairs of accumulators, T1 and W1, and T2 and W2; where the T accumulators count the beats as they occur, up to a maximum of 200 each, and the W accumulators count the number of wide beats contained in the corresponding T accumulator. Therefore, the procedure was to increment T1 as each beat occurred, at the same time W1 would be incremented if the beat was classified as wide. Then when T1 became greater than 200, T2 and W2 would be set to zero and subsequenty incremented as were T1 and W1. When T2 then exceeded 200, T1 and W1 would be set to zero and subsequently incremented, and so the process continues. Then as each beat occurred, the percentage of wide complexes would be calculated as

$$\frac{W1 + W2}{T1 + T2} \times 100 = \text{Percentage of wide complexes} \quad (5.22.)$$

5.3.4. R-R Interval Classification

The R-R interval classification algorithm is concerned with labelling each R-R interval as either very long, long, short or normal.

An R-R interval is classified as very long, if the current R-R interval is greater than twice the average value.

The classification of intervals as long or short is via the equations illustrated in Fig. 2.2. In the simple case when the pulse rate is greater than 120 beats per minute, the tolerance (IER) on the current R-R interval is calculated as

$$IER = \frac{ISAMP}{12} \quad (5.23.)$$

where ISAMP is the average R-R interval.

When the pulse rate is less than 120 beats per minute, the tolerance (IER) is calculated as

$$IER = ISAMP\left(\frac{1}{8} - \frac{RATE}{2880}\right) \quad (5.24.)$$

The RATE parameter in the above expression can be expressed in terms of the average R-R interval by

$$RATE = \frac{250 \times 60}{AVRR} = \frac{15000}{AVRR} \quad (5.25.)$$

where AVRR is the average number of sample periods in the R-R intervals.

Similarly ISAMP may be expressed as :-

$$ISAMP = \frac{AVRR}{250} \qquad (5.26.)$$

Combining these equations gives,

$$IER = \frac{AVRR}{250 \times 8} - \frac{15000}{2880 \times 250} \qquad (5.27.)$$

If IER is expressed in terms of the number of sample periods then,

$$IER = \frac{AVRR}{8} - 5{\cdot}2 \qquad (5.28.)$$

Approximating the coefficient in equation 5.28., to produce an expression easily and quickly performed by the microprocessor gives,

$$IER = \frac{AVRR}{8} - 5 \qquad (5.29.)$$

An R-R interval is then classified as long if,

$$RRC > AVRR + IER \qquad (5.30.)$$

or short if,

$$RRC < AVRR - IER \qquad (5.31.)$$

where RRC is the number of sample periods in the current R-R interval.

Between these two limits the R-R interval is classified as normal.

5.3.5. Sequence Pattern Recognition

At this stage in the monitoring program, three workspace registers contain sequence information as follows :-

RO = LONG R-R interval sequence bit flags
R1 = SHORT R-R interval sequence bit flags
R2 = WIDE QRS sequence bit flags

The information stored in each of these registers is obtained as follows. As each QRS complex is detected, all three registers are shifted left by one bit. Then, for example, if the QRS width algorithm finds the complex is wide, the least significant bit in register R2 is set to a one. The information stored in registers RO and R1 is generated in a similar manner by the R-R interval classification algorithm. There is no need for a register to indicate normal R-R intervals, as this is indicated by zeros occurring in corresponding bit positions of registers RO and R1.

The sequence pattern recognition algorithm, therefore uses these three registers to detect the sequences of long (L), short (S) or normal (N) R-R intervals, and wide QRS complexes (WQRS) that are specified in the diagnosis criteria of Table 2.1. A list of these sequences is given in Table 5.2. using a shorthand notation where, for example, a sequence consisting of a short R-R interval (S), followed by a wide QRS complex (WQRS), followed by an R-R interval which is not short ($\overline{S}$), all repeated 4 times, is written as, $4(S \rightarrow WQRS \rightarrow \overline{S})$.

The approach used to produce the sequence

SEQUENCES

(1) $4(S \rightarrow \bar{S})$

(2) $4(S \rightarrow WQRS \rightarrow \bar{S})$

(3) $4(S \rightarrow \bar{S} \rightarrow N)$

(4) $4(S \rightarrow WQRS \rightarrow \bar{S} \rightarrow N)$

(5) $S \rightarrow \bar{S} \rightarrow S$

(6) $S \rightarrow \bar{S} \rightarrow N \rightarrow \bar{L}$

(7) $3(S) \rightarrow WQRS$

(8) $S \rightarrow \bar{S} \rightarrow L$

(9) $S \rightarrow \bar{S} \rightarrow N \rightarrow L$

TABLE 5.2. Sequence of R-R Intervals and QRS Complex Width to be Detected.

recognition algorithm, was the use of assembly language instructions capable of detecting bit patterns in words (compare ones corresponding (COC) and Compare zeros corresponding (CZC)). (Ref.58.).

The algorithm is structured such that certain simple comparisons eliminate groups of sequences if not present, as illustrated by the basic flow diagram given in Fig. 5.12. The "X" shown in some of the decision boxes in Fig. 5.12., indicate that these intervals are not tested at that point in the algorithm.

5.3.6. Monitoring Function Output

The output from the monitoring function is in the form of two 16 bit words (MF1 and MF2). The individual bits of each word indicates the true (1) or false (0) results from the various monitoring algorithms. The condition indicated by each of these bits, is given in Table 5.3.; and it is these two words which are supplied to the diagnosis function.

The program modules which combine to give the monitoring results for the initial monitoring criteria (section 2.5.1.), have program module identifiers MONTDP, MONS1P and MONS2P, as supplied in the program appendix.

5.3.7. Alternative Monitoring Criteria

The monitoring function produced for the alternative monitoring criteria (section 2.5.2.) is an

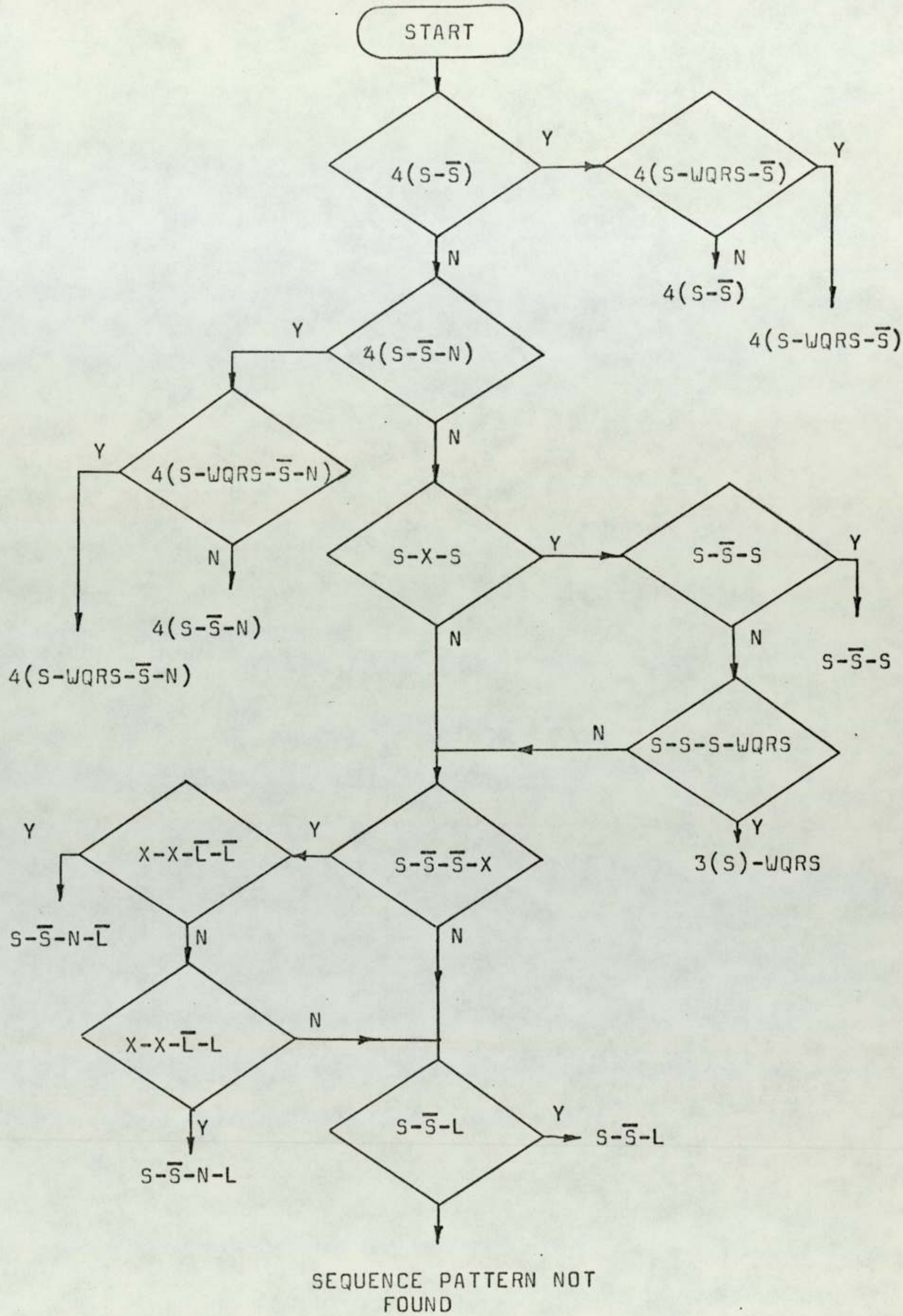


Fig. 5.12 BASIC FLOW DIAGRAM OF SEQUENCE DETECTION ALGORITHM

EVENT WORD MF1	WORD 1 SET BIT
PULSE RATE IS GREATER THAN 140 BEATS PER MINUTE	0 *
PULSE RATE IS GREATER THAN 110 BEATS PER MINUTE	1
PULSE RATE IS GREATER THAN 100 BEATS PER MINUTE	2
PULSE RATE IS GREATER THAN 90 BEATS PER MINUTE	3
PULSE RATE IS GREATER THAN 80 BEATS PER MINUTE	4
PULSE RATE IS GREATER THAN 60 BEATS PER MINUTE	5
PULSE RATE IS GREATER THAN 40 BEATS PER MINUTE	6
PULSE RATE IS GREATER THAN 20 BEATS PER MINUTE	7 *
	8 *
NO QRS COMPLEX DURING THE LAST 6 SECONDS	9
CURRENT R-R INTERVAL IS ≥ TWICE THE AVERAGE	10
CURRENT R-R INTERVAL IS LONG	11
CURRENT R-R INTERVAL IS SHORT	12
CURRENT QRS COMPLEX IS WIDE	13
MORE THAN 82% OF PREVIOUS QRS COMPLEXES ARE WIDE	14 *
	(MSB) 15 *

EVENT WORD MF2	WORD 2 SET BIT
A SEQUENCE 4(S→$\overline{S}$) HAS OCCURRED	0
A SEQUENCE 4(S→WQRS→$\overline{S}$) HAS OCCURRED	1
A SEQUENCE 4(S→$\overline{S}$→N) HAS OCCURRED	2
A SEQUENCE 4(S→WQRS→$\overline{S}$→N) HAS OCCURRED	3
A SEQUENCE S→$\overline{S}$→S HAS OCCURRED	4
A SEQUENCE S→$\overline{S}$→N→$\overline{L}$ HAS OCCURRED	5
A SEQUENCE 3(S)→WQRS HAS OCCURRED	6
A SEQUENCE S→$\overline{S}$→L HAS OCCURRED	7
A SEQUENCE S→$\overline{S}$→N→L HAS OCCURRED	8
	9 *
	10 *
	11
	12
	13
	14
	(MSB) 15

* These bits have a different meaning when the alternative monitoring criteria are used.

TABLE 5.3. TESTS PERFORMED IN ECG MONITORING FOR INITIAL DIAGNOSIS CRITERIA.

adaptation of the initial monitoring function.

Consider the numbered process boxes shown in the basic flow diagram of the monitoring function Fig. 5.13. The modifications that were made to these processes, to produce the alternative monitoring function were as follows.

Process 1. Initial Value For the Average R-R Interval

The initial value for the average R-R interval (AVRR), is calculated from the first 16 R-R intervals received, when the monitoring function is started. This replaces the original approach which was to use the first R-R interval measured, as the initial average value.

Process 2. Setting of Pulse Rate Flags in MF1

Certain new pulse rate flags were introduced into the monitoring result word MF1, as specified in Table 5.4.

Process 3. QRS Widths Measurement

The QRS complex width limit, used to classify complexes as wide or normal, was increased from 112mS (28 sample periods) to 120mS (30 sample periods).

Process 4. Percentage of QRS Complexes Wide

This process was not required for the alternative monitoring function, so it was removed.

Process 5. Calculation of True R-R Interval

This process remained unchanged.

Process 6. Classification of R-R Intervals

The classification of R-R intervals as long or

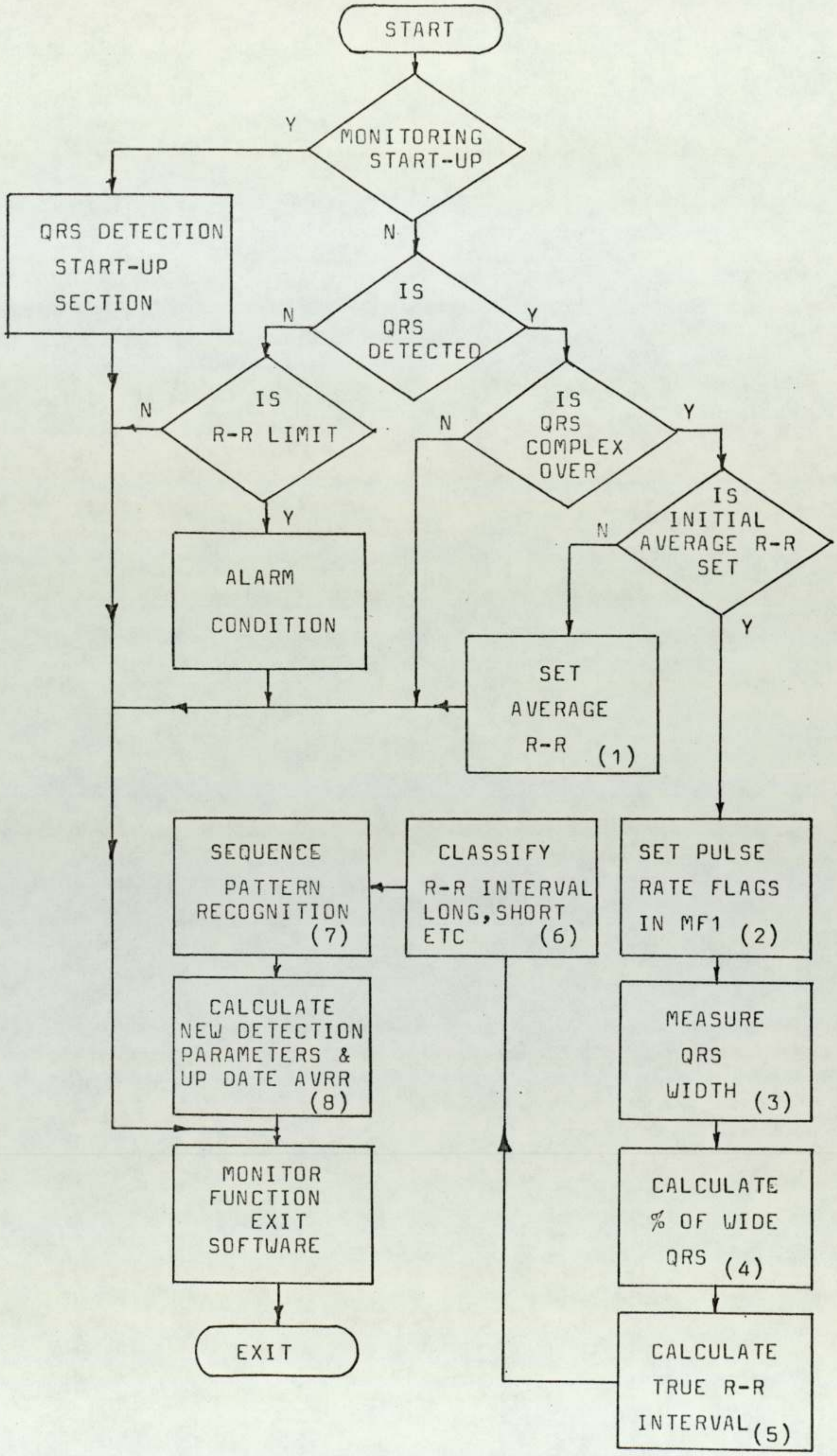


Fig. 5.13. BASIC FLOW DIAGRAM OF MONITORING FUNCTION.

short was changed, and in place of equations 5.23. and 5.29. previously used, a constant ± 15% tolerance on the average R-R interval was adopted.

Process 7. Sequence Pattern Recognition

The sequence pattern recognition section was expanded to include the new sequences of $(S - \tilde{S})$ and (S - S).

Process 8. Parameter Up-dating

The average R-R interval (AVRR) is calculated from the 16 most recent normal R-R intervals.

All modifications to the monitoring result words MF1 and MF2 for the alternative monitoring function are given in Table 5.4.

The program modules which combine to produce the alternative monitoring function, have program module identifiers of MONTDP, MONS3P and MONS4P as supplied in the program appendix.

5.4. Diagnosis Function

The diagnosis function is concerned with the recognition of specific arrhythmias from the bit patterns contained in the monitoring result words (MF1 and MF2).

The technique used to implement this function was a decision table type of approach (Ref.54.55.) A decision table is structured as shown in Table 5.5., where in the diagnosis function implementation, the condition stub contains the monitored events, such as "R-R interval long", and the condition body contains

EVENT WORD MF1	WORD 1 SET BIT
PULSE RATE IS GREATER THAN 150 BEATS PER MINUTE	0
PULSE RATE IS GREATER THAN 30 BEATS PER MINUTE	7
CURRENT R-R INTERVAL IS LESS THAN 600mS	8
NOT USED	14
PULSE RATE IS GREATER THAN 200 BEATS PER MINUTE	15
EVENT WORD MF2	**WORD 2 SET BIT**
A SEQUENCE $S - \overline{S}$ HAS OCCURRED	9
A SEQUENCE S - S HAS OCCURRED	10

TABLE 5.4. MODIFICATIONS TO MONITORING RESULT WORDS (TABLE 5.3.) FOR ALTERNATIVE MONITORING CRITERIA.

CONDITION STUB — CONDITION BODY

CONDITION 1	Y	Y	N	N
CONDITION 2	Y	N	-	N
ACTION 1	X			
ACTION 2		X		
ACTION 3			X	
ACTION 4				X

ACTION STUB — ACTION BODY

TABLE 5.5. EXAMPLE DECISION TABLE.

the condition requirements of yes (Y), no (N), or a dash which means either (Y or N). The action stub is then either a diagnosis or an instruction, which are matched to the conditions via the "X"s in the action body.

One of the most important rules, when using decision tables is that relating to "ambiguity" (Ref.54.). This rule simply states that no two sets of conditions, in the condition body, can be the same. For example, in Table 5.5., there is "ambiguity" between action 3 and 4, due to the dash in the condition body. This particular "ambiguity" could be resolved by either substituting a "Y" for the dash, or combining actions 3 and 4.

When trying to convert the initial arrhythmia criteria given in Table 2.1., into a decision table, it was found that the 23 different diagnoses given were not all mutually exclusive. Therefore, in terms of producing a single decision table, this would result in the type of "ambiguities" described previously. The approach adopted therefore was the production of a number of small decision tables that could be used, and which did not contain ambiguities. Also, as there was some doubt about the final diagnosis criteria that was to be used by the system, only 8 of the arrhythmia diagnoses were used to produce the first diagnosis program from decision tables. The remaining diagnoses would then have been incorporated into this diagnosis program, if this particular set of criteria was that chosen for use in the final system.

The resulting decision tables produced to diagnose 8 different arrhythmias are shown in Table 5.6. Using these tables, the first diagnosis program was produced, and is supplied in the program appendix under the program module identifier DIAT1P.

Although this program was produced from decision tables, it was apparent that the initial arrhythmia criteria did not lend themselves easily to decision table implementation. This is due mainly to the fact that the diagnoses were not specified so that they were mutually exclusive.

However, the alternative arrhythmia criteria given in Table 2.2., required only two decision tables for complete classification, as shown in Table 5.7. Using these decision tables, the alternative diagnosis program was produced much more easily, and is supplied in the program appendix under the program module identifier DIAT2P.

The output from the first diagnosis program is two 16 bit words (DF1 and DF2), the individual bits of which indicate, if set to a one, which arrhythmias have been diagnosed, as shown in Table 5.8. The output from the alternative diagnosis program is a single 16 bit word, the individual bits of which indicate the arrhythmias diagnosed and the corresponding alarm status etc. as shown in Table 5.9.

TABLE 1

IS PR 80 BPM	N	Y	-
NO QRS FOR 6 SEC	Y	Y	N
CARDIAC ARREST	X		
LOST SIGNAL		X	
GO TO TABLE 2			X
EXIT FUNCTION	X	X	

TABLE 2

IS PR 60 BPM	N	N	Y	-
IS PR 40 BPM	Y	N	Y	-
IS PR 20 BPM	Y	Y	Y	-
BRADYCARDIA	X			
IDIOVENTRICULAR		X		
GO TO TABLE 3			X	
GO TO TABLE 5	X	X		X

TABLE 3

IS PR 140 BPM	N	Y	-
IS PR 110 BPM	Y	Y	-
ARE 82% QRS WIDE	-	N	-
SUPERVENTRICULAR	X	X	
GO TO TABLE 4	X	X	X

TABLE 4

IS PR 140 BPM	N	-
ARE 82% QRS WIDE	Y	-
BUNDLE BRANCH BLOCK	X	
EXIT FUNCTION	X	X

TABLE 5

IS PR 80 BPM	N	-
IS RR 2.AVRR	Y	-
ASYSTOLE	X	
GO TO TABLE 6	X	X

TABLE 6

IS PR 60 BPM	N	-
WAS R-R LONG	Y	-
WAS QRS WIDE	Y	-
ESCAPE BEAT	X	
GO TO TABLE 4	X	X

TABLE. 5.6. DECISION TABLES USED TO PRODUCE INITIAL DIAGNOSIS FUNCTION.

TABLE 1

R-R SHORT	Y	Y	Y	Y	N	N	N	N	N	-	-
R-R LONG	N	N	N	N	N	N	Y	Y	Y	-	-
R-R VERY LONG	N	N	N	N	N	N	N	N	Y	-	-
SEQUENCE S-S	Y	Y	Y	Y	N	N	N	N	-	-	-
SEQUENCE $S-\overline{S}$	N	N	N	N	Y	Y	Y	Y	-	-	-
R-R 5 Sec.	N	N	N	N	N	N	N	N	N	Y	-
R-R 300mS	N	N	Y	Y	N	N	N	N	N	-	-
BROAD QRS (120mS)	N	Y	N	Y	N	Y	N	Y	-	-	-
PAROXYSMAL TACHYCARDIA	X	X	X	X							
PREMATURE BEAT					X	X	X	X			
PAUSE									X		
ASYSTOLE										X	
BROAD QRS		X		X		X		X			
ALARM 1 (HIGH)			X	X						X	
ALARM 2 (LOW)	X	X									
GO TO TABLE 2	X	X	X	X	X	X	X	X	X	X	X

TABLE 2

IS AVRR 400mS	N	Y	N	N	-
IS AVRR 600mS	Y	Y	N	N	-
IS AVRR 1 Sec.	Y	Y	N	N	-
IS AVRR 2 Sec.	Y	Y	Y	N	-
NON-PAROXYSMAL TACHYCARDIA	X	X			
NON-PAROXYSMAL BRADYCARDIA			X	X	
ALARM 2 (LOW)		X			
EXIT PROGRAM					X

TABLE 5.7. ALTERNATIVE DIAGNOSES DECISION TABLES.

BITS		DIAGNOSIS WORD DF1		DIAGNOSIS WORD DF2
0		VENTRICULAR TACHYCARDIA		PREMATURE ATRIAL CONTRACTION
1	*	SUPRAVENTRICULAR TACHYCARDIA		PAC RUN
2	*	BUNDLE BRANCH BLOCK		BIGEMNINY DUE TO PAC
3	*	BRADYCARDIA		ATRIAL TRIGEMINY
4	*	IDIOVENTRICULAR RHYTHM		ABNORMAL CONDUCTION
5	*	CARDIAC ARREST		SINUS ARRHYTHMIA
6	*	ASYSTOLE		PREMATURE CONTRACTION
7	*	ESCAPE BEAT		ATRIAL FIBRILLATION
8		PREMATURE VENTRICULAR CONTRACTION		ACCELERATED VENTRICULAR RATE
9		RUN OF PVC		SVT WITH ABBERANT CONDUCTION
10		BIGEMINY RHYTHM	*	LOST SIGNAL
11		TRIGEMINY RHYTHM		-
12		-		-
13		-		-
14		-		-
15		-		-

(* Currently implemented)

TABLE 5.8. DIAGNOSIS PROGRAM RESULT WORDS FOR INITIAL ARRHYTHMIA CRITERIA

BIT	DIAGNOSIS WORD DF1
0	NON PAROXYSMAL TACHYCARDIA
1	NON PAROXYSMAL BRADYCARDIA
2	PAROXYSMAL TACHYCARDIA
3	PREMATURE BEAT
4	PAUSE
5	ASYSTOLE
6	-
7	-
8	-
9	-
10	-
11	-
12	-
13	BROAD QRS
14	ALARM 1
15	ALARM 2

TABLE 5.9. ALTERNATIVE DIAGNOSIS PROGRAM RESULT WORD BIT ASSIGNMENT

5.5. Communications

The communications function is concerned with the transfer of information between the patient-dedicated processor, and the operator-dedicated processor.

In order to minimise the number of wires carrying information between processors, all of which would converge on the operator-dedicated processor, serial asynchronous data channels were adopted. The UARTs (Universal Asynchronous Receiver Transmitter) required for such a communications link, were implemented by dedicated TMS 9902 asynchronous communications controllers, specifically designed for use with the TMS 9900 microprocessor.

The inter-processor communications protocol developed for use in this distributed processor system is described in Chapter 6, and is designed to allow communications to operate without interfering with the real-time processing being performed by the patient-dedicated processors. The information which passes between processors comes in two form. The first is a single instruction, and the second is an instruction followed by 50 bytes of data, as described in Chapter 6.

Within each patient-dedicated processor system, the use of the UART communications channel is controlled by an operating system program, which is described in Section 5.6.

The remainder of the communications function is divided into the three tasks of input, output and interpreter (microprocessor communications package), as illustrated in Fig. 5.1.

5.5.1. Input Task

The Input task (program module identifier IPPTOP) is activated by the patient processor operating system, when normal communications protocol start-up procedures have been performed. Having been activated, the input task then proceeds to store the incoming instruction and data (when present) into an input file (IPFILE), transmitting protocol reply characters as required.

On completion of communications, the input task activates the microprocessor communications package task, and then deactivates itself. The basic flow diagram of the input task is shown in Fig. 5.14.

5.5.2. Microprocessor Communication Package

The microprocessor communications package (program module identifier MCPTOP) is concerned with the interpretation of the instructions placed in the communications input file, and the generation of output data for transmission from the patient processor system. For example, the instruction received by the input task and placed in the input file, may be interpreted by this task as being a request for pulse rate data, from the operator processor. Having decoded this instruction, the task then produces the correct reply instruction, and the requested pulse rate data, all of which are placed in an output file

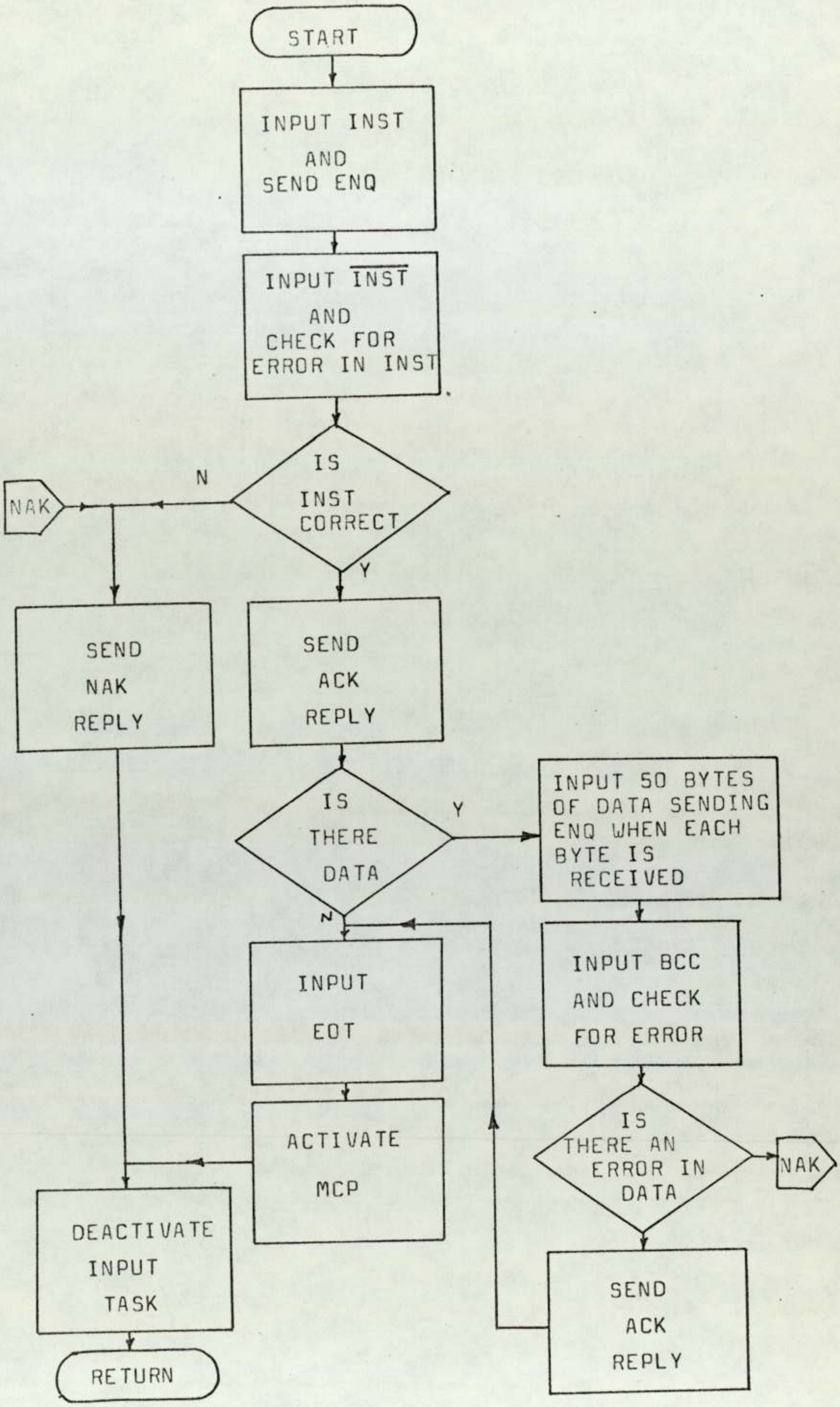


Fig. 5.14. BASIC FLOW DIAGRAM OF INPUT TASK.

(OPFILE). On completion of this operation, the task then goes to the operating system program which controls the use of the UART, so that communications can be established with the operator-dedicated processor, before activating the output task.

If the reply to the operator processor is a single instruction (i.e. no data), the address where the reply instruction is resident in PROM is supplied to the output task.

Before describing the operation of the output task, it should also be noted that the microprocessor communication package program, can also be activated by other system functions (monitoring and diagnosis). This allows messages and data associated with these functions to be sent to the operator processor. The basic flow diagram of the microprocessor communication package is shown in Fig. 5.15.

5.5.3. Output Task

When the output task (program module identifier OPPTOP) has been activated, it uses the address supplied to it, in order to locate the information to be transmitted. In the example given above, this address would be the location of the output file. As the transmission of information is performed by the output task, it waits for communication protocol replies from the operator processor, as described in Chapter 6.

The basic flow diagram of the output task is

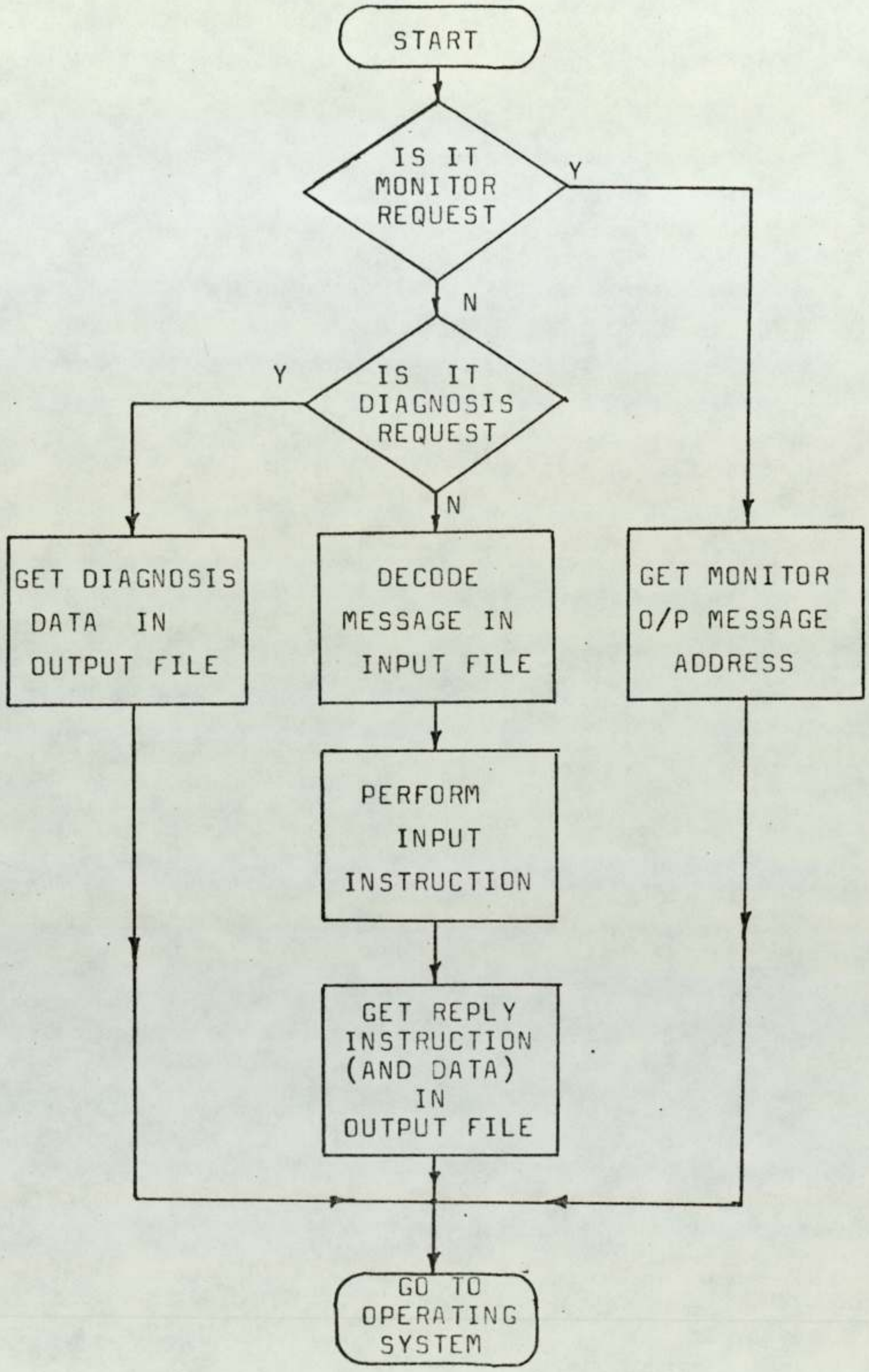


Fig. 5.15. BASIC FLOW DIAGRAM OF THE MICROPROCESSOR COMMUNICATION PACKAGE.

given in Fig. 5.16.

5.6. Control

As can be seen from Fig. 5.1., the operating system of the patient-dedicated processor consist of four programs.

The first of these programs is concerned with the patient processor system initialisation (program module identifier PPINIP), and is performed at system start-up time. The initialisation performed is that of memory (RAM) clearing and software environment setting, and the resetting and programming of hardware interfaces.

The second program module is concerned with handling the interrupt generated by the A to D converter (program module identifier PPCIHP). The function of this program is to determine which one of the 6 tasks, as shown in Fig. 5.1., was interrupted so that the software returns (WP, PC, and ST) (Ref.58.) can be saved, and the particular task concerned set suspended. Having done this the program then activates the signal conditioning task, and passes control to the real-time task scheduler program. Also during the operation of this program the time, since system start-up, is calculated using the 4mS interval of the ADC interrupt. The basic flow diagram of this program is shown in Fig. 5.17.

The real-time task scheduler program (program module identifier PPTHAP) is concerned with determining which of the 6 tasks should next be performed. The 6 tasks have different priorities, the highest being

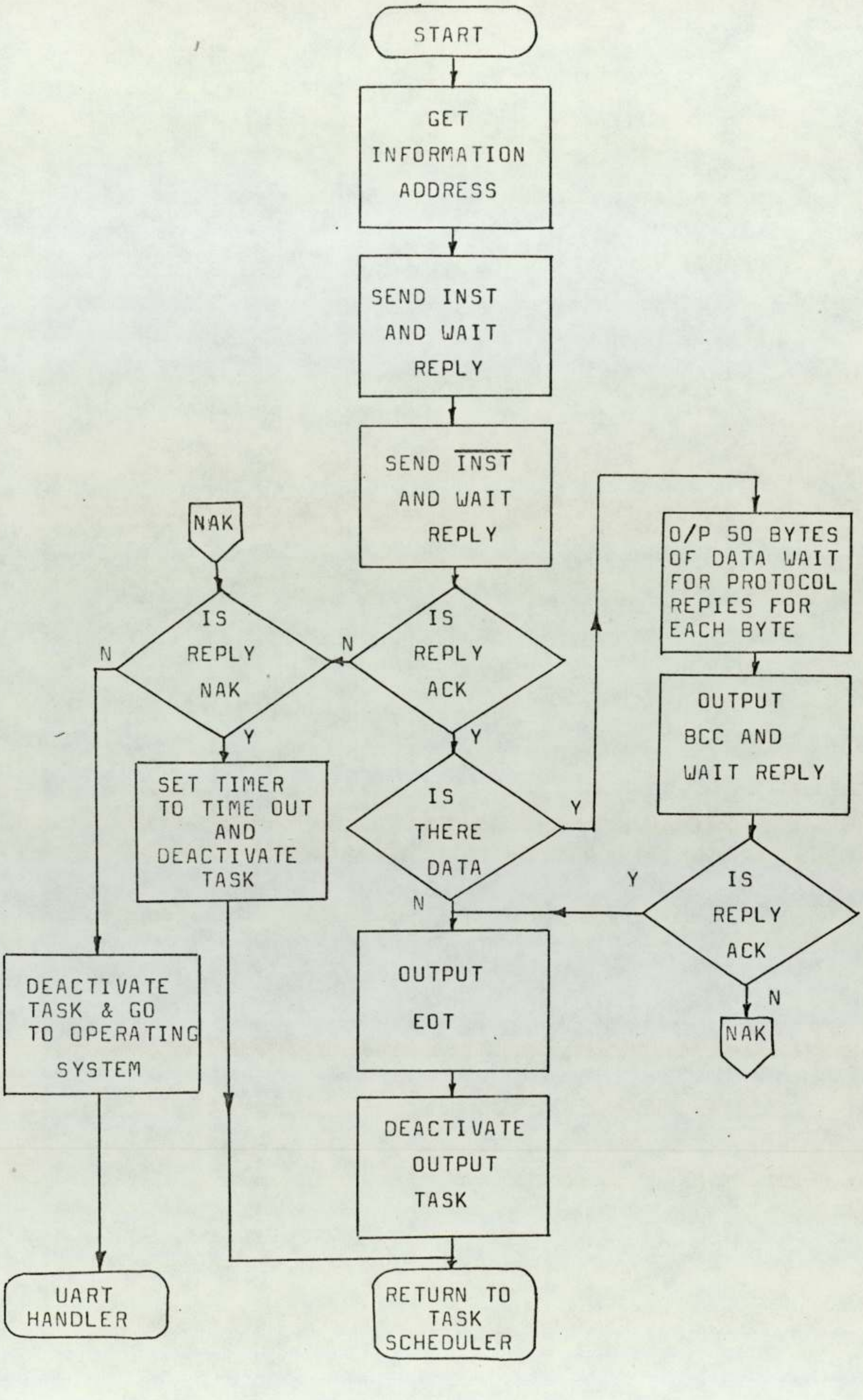


Fig. 5.16. BASIC FLOW DIAGRAM OF OUTPUT TASK

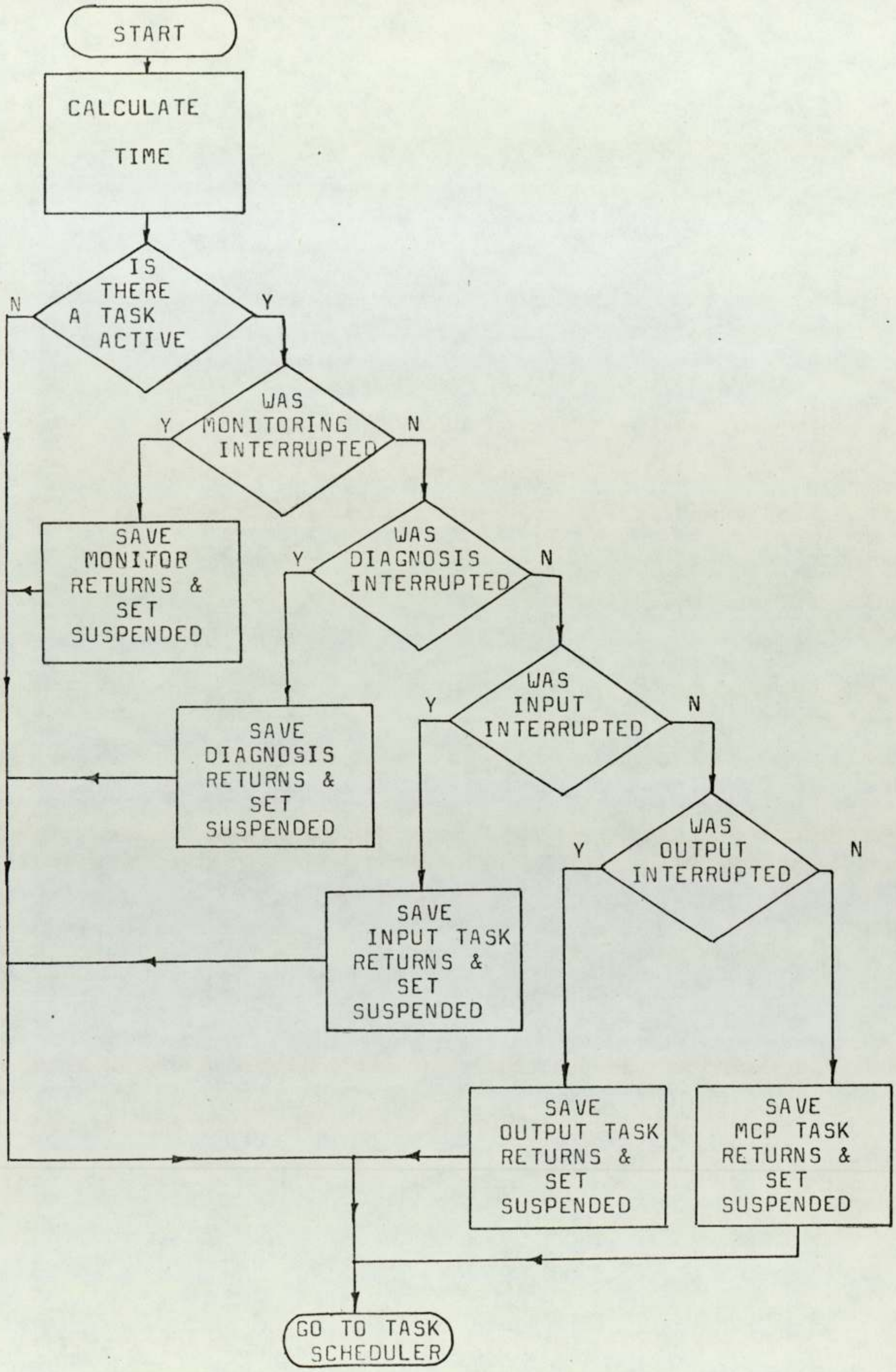


Fig. 5.17. BASIC FLOW DIAGRAM OF ADC INTERRUPT HANDLING PROGRAM.

signal conditioning as this consumes 25% of processing time (Table 4.1.). Having found an active task the scheduler program tests if the appropriate task is in a suspended state, having been previously interrupted, and passes control to the task accordingly. The basic flow diagram of the real-time task scheduler program is given in Fig. 5.18.

The fourth and final program which makes up the patient processor operating system, is the UART interrupt handling program (program module identifier PPUIHP). This program performs the complex functions of :-

(1) Establishing inter-processor communications.

(2) Activating input or output tasks as and when required.

(3) Deactivating input or output task, when an error occurs in the inter-processor communications.

(4) Handling the TMS 9902 interval timer, to detect the error of excessive delays during inter-processor communication.

(5) Handling output requests from the microprocessor communication package program.

(6) Activating suspended output requests from the microprocessor communication package program, which would have been suspended due to input communications.

The basic flow diagram of this program is given in Fig. 5.19, and details of the inter-processor communication protocol developed for use in this distributed processor system is given in Chapter 6.

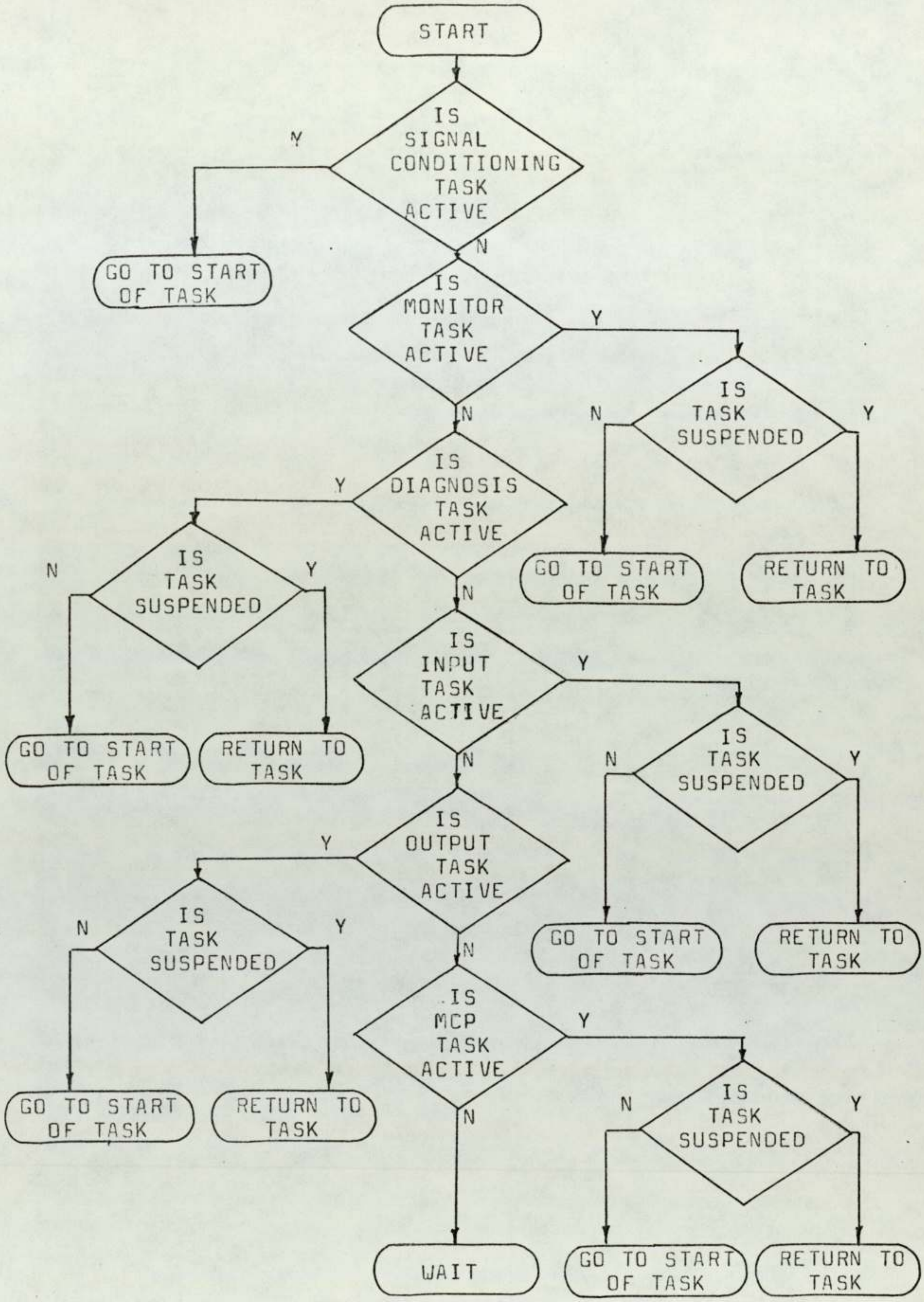


Fig. 5.18. BASIC FLOW DIAGRAM OF REAL-TIME TASK SCHEDULER PROGRAM.

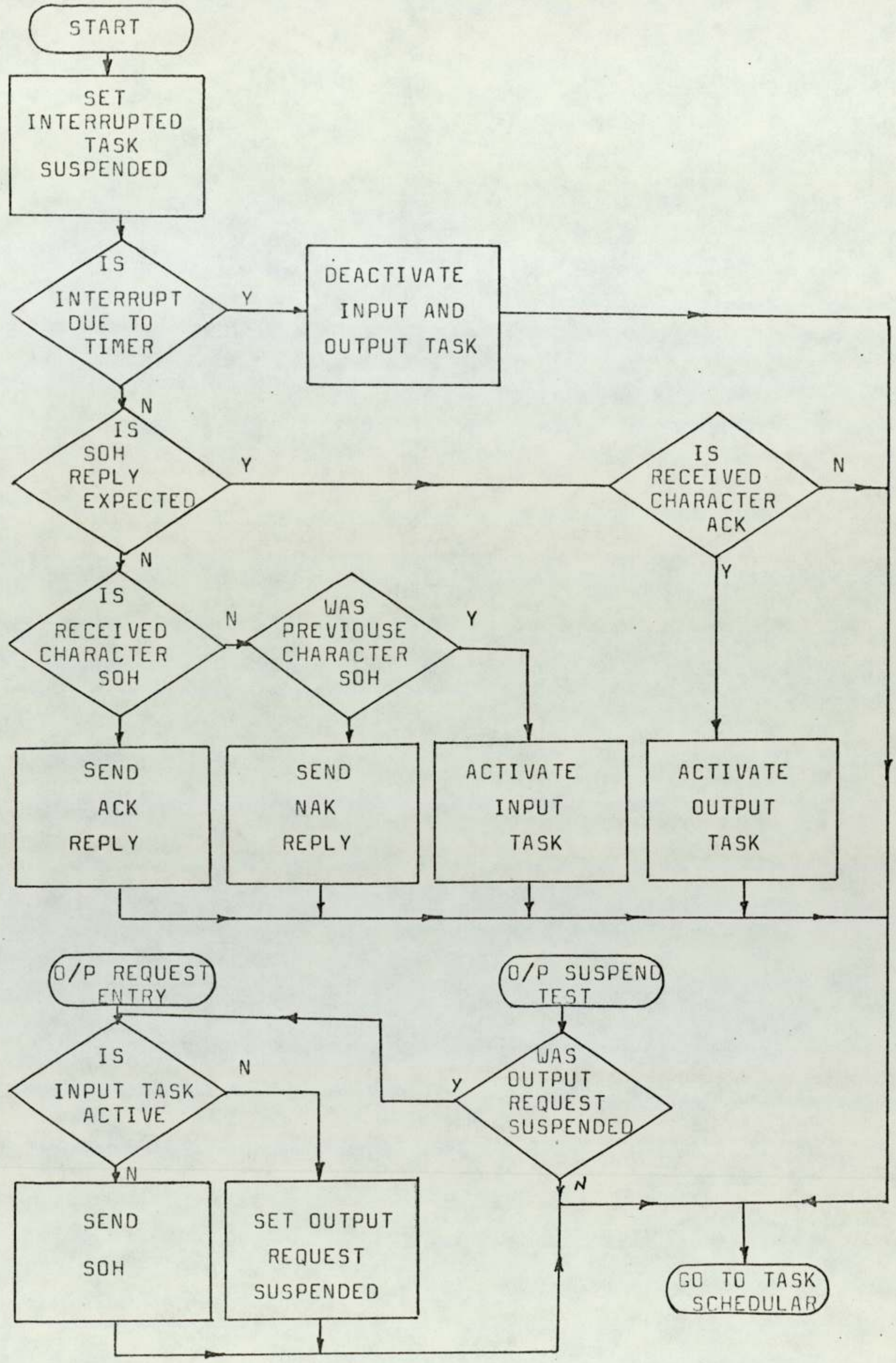


Fig. 5.19. BASIC FLOW DIAGRAM OF UART INTERRUPT HANDLING PROGRAM.

CHAPTER 6

Operator-dedicated Processor System Design

6.1. Introduction

The operator-dedicated processor (also referred to as the nurse station processor in the program appendix) is concerned with the centralised functions of display, storage, overall system control and system/operator communication. The software structure of the operator-dedicated processor system is given in Fig. 6.1., and as can be seen, it is similar to the patient-dedicated processor system. However, in this case the highest priority task is the inter-processor communications input task, and the lowest is the VDU peripheral controller task. The input task has the highest priority, because it is vital that information coming from patient-dedicated processors is handled quickly.

In order to understand the operation of this system, we must consider each of the system tasks in turn, and hence the resulting software control provided by the operator-dedicated processor operating system. However, an important feature which influences the operation of the system, is the inter-processor communication protocol, developed for use in this distributed processor system. Therefore, the inter-processor communications protocol developed will be described first.

6.2. Inter-processor Communications Protocol

The inter-processor communications are provided by

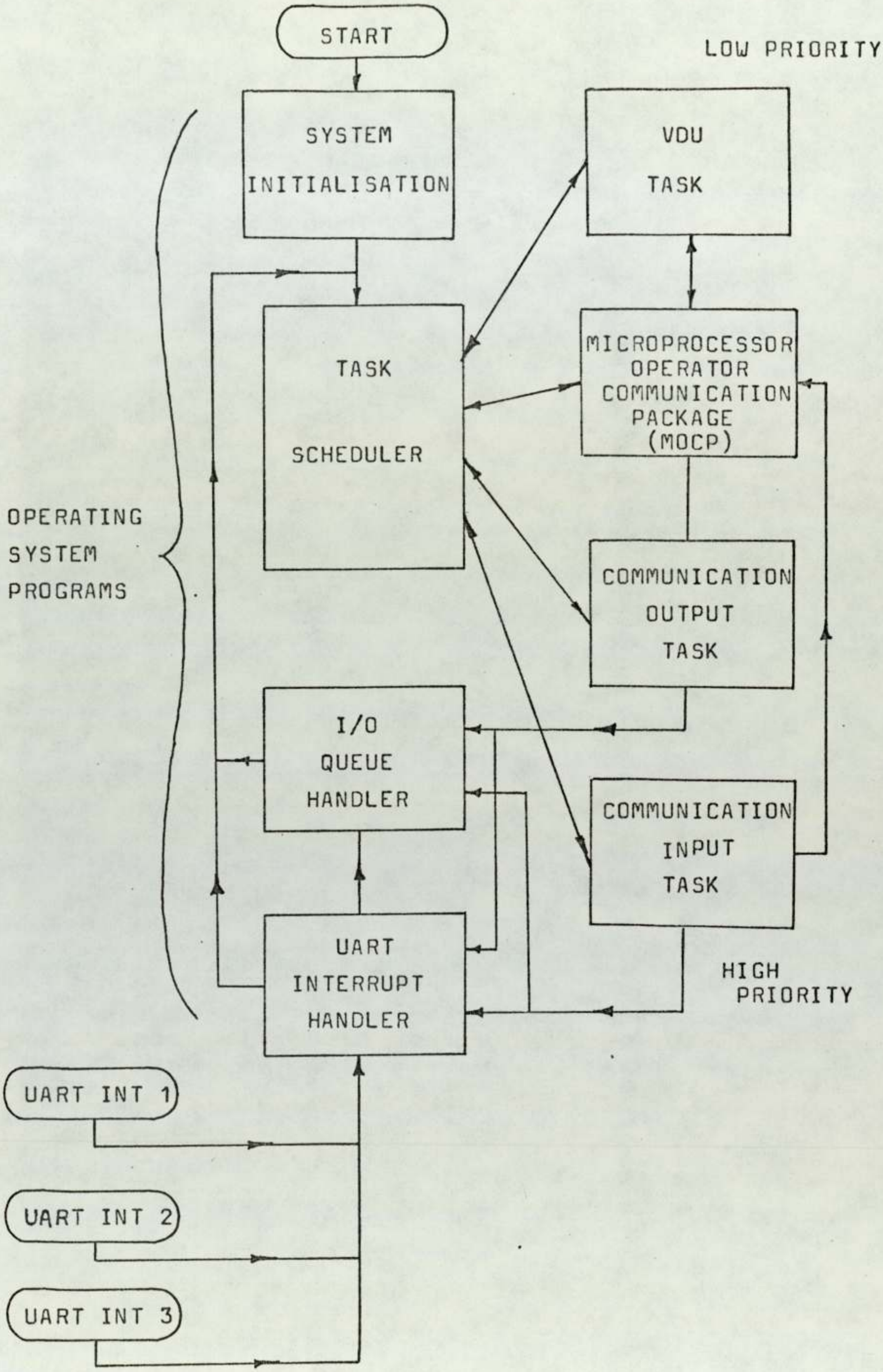


Fig. 6.1 SOFTWARE STRUCTURE OF OPERATOR-DEDICATED PROCESSOR SYSTEM.

serial data links in the form of UARTs (Universal Asyncronous Receiver Transmitters), which are used to transmit bytes of information between processors. This approach was chosen, to reduce the wiring converging on the operator system, and because of the availability of a device; the TMS 9902 asynchronous communication controller, which could provide the serial data links required, and which is specifically designed for use with the TMS 9900 microprocessor.

The aim therefore, was to provide a private serial communication link between each patient-dedicated processor and the operator-dedicated processor, as shown in Fig. 6.2. This structure and the inter-processor communication protocol developed, was influenced by the following important considerations.

(1) Inter-processor communications should not interfere in the normal real-time processing being performed by each patient-dedicated processor.

(2) The communications between processors should be as fast as possible. This is not only concerned with the bit rate used by the UARTs, but also with the speed with which the operator processor handles all of the communication channels at its disposal.

The first of these considerations was implemented by giving the patient processor communication function a lower priority than the real-time functions, as described in Chapter 5. However, by doing so one has produced the situation whereby the rate of data received or transmitted

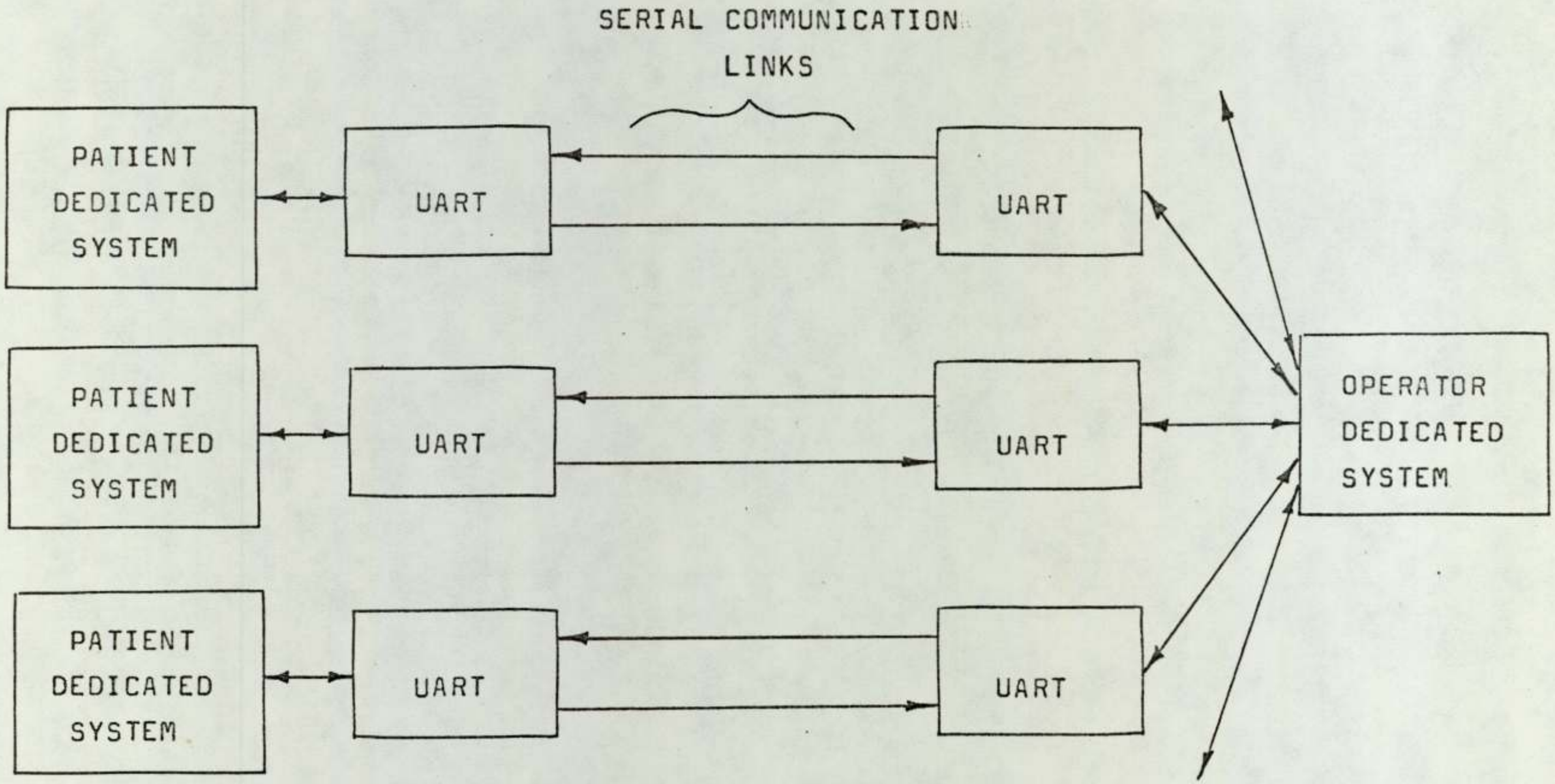


Fig. 6.2. DISTRIBUTED COMMUNICATION SYSTEM.

by the patient processor, is influenced by the real-time processing being performed by that processor. Therefore, for example, the operator processor could not transmit a continuous string of characters to the patient processor. This is because overrun errors would occur at the patient processor UART, while the patient processor was performing real-time functions.

The second consideration given above, is concerned with the manner in which the operator processor uses all the inter-processor communication channels. The approach chosen was to allow the operator processor to communicate simultaneously, if necessary, with all the patient-dedicated processors. The information passing between processors would then be handled as and when it occurred, and therefore this approach would make more efficient use of the processing time at the operator processor.

Therefore, the inter-processor communications protocol required, had to cater for all these considerations, and several approaches were considered (Ref. 59,60,61,62.). The final solution is shown diagramatically in Fig. 6.3. and Fig. 6.4. which indicate the appropriate programs that are active during communications, and the sequence of ASCII control characters (SOH,ACK,NAK,ENQ and EOT) and information characters (INST,$\overline{\text{INST}}$,DATA,BCC) sent, for correct inter-processor protocol. From Fig. 6.3. it will be seen that communications are established by the UART interrupt handler program of the sending processor, by the transmission of the "start of heading" (SOH)character,

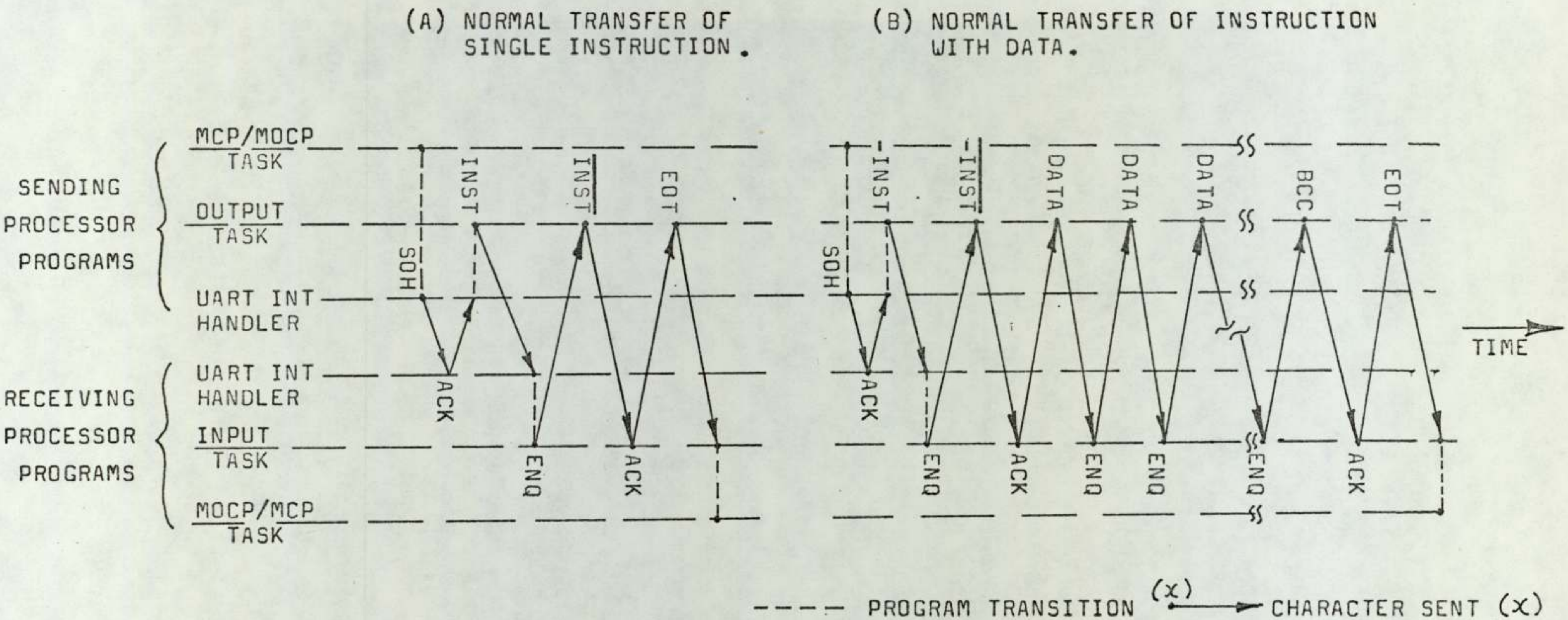


Fig. 6.3. PROTOCOL FOR NORMAL TRANSFER OF INFORMATION.

SOH :-START OFHEADING
ACK :-ACKNOWLEDGE
ENQ :-ENQUIRY
NAK :-NEGATIVE ACK

EOT :-END OF TRANSMISSION
INST :-INSTRUCTION
$\overline{\text{INST}}$;-INVERSE OF INST
BCC :-BLOCK CHECH CHAR

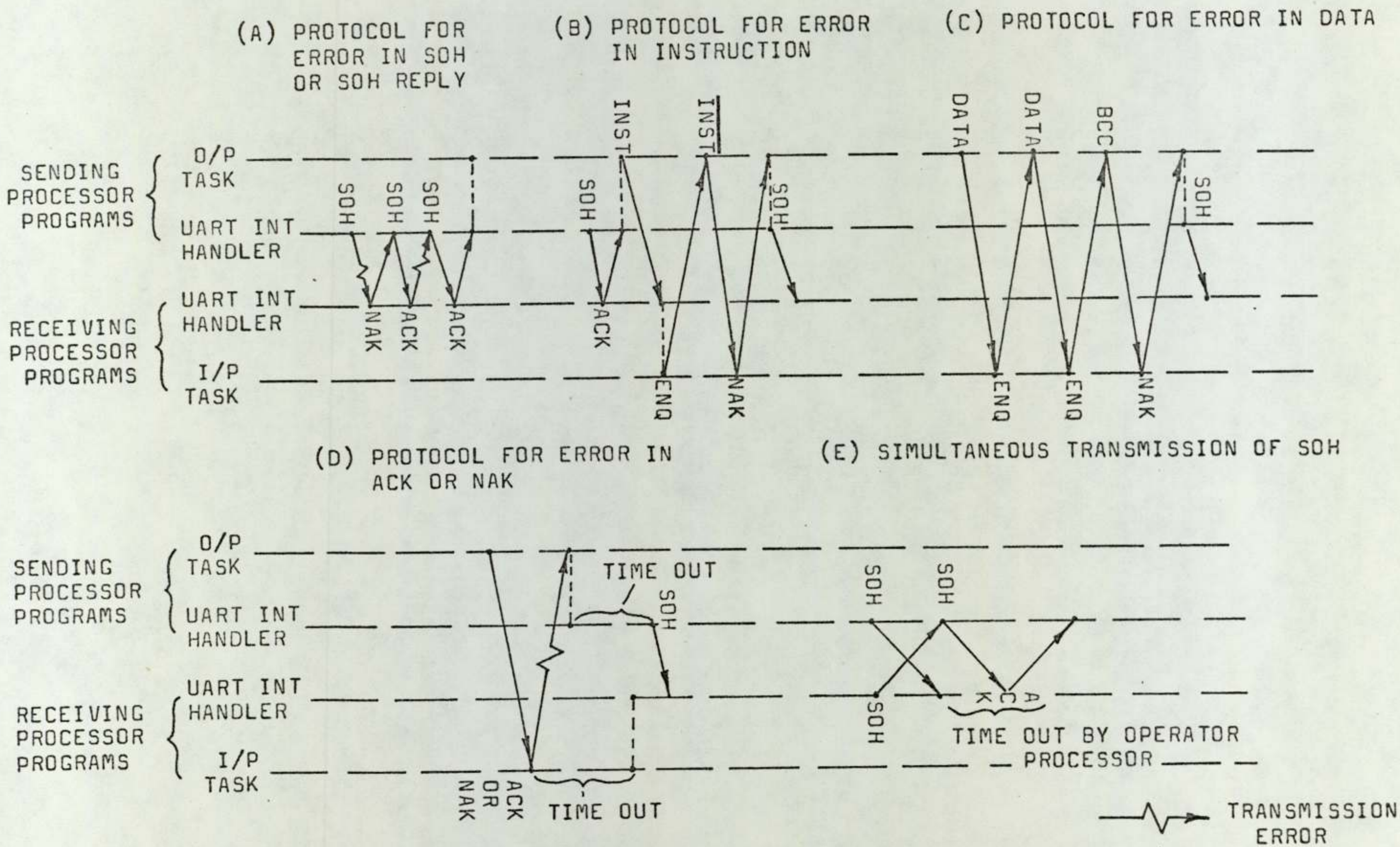


Fig. 6.4. COMMUNICATION PROTOCOL FOR ERROR CONDITIONS.

and the reception of the "acknowledge" (ACK) character. The instruction (INST) and its inverse ($\overline{\text{INST}}$) are then sent by the output task of the sending processor, and received and checked by the input task of the receiving processor. If the instruction is received correctly the input task replies with ACK (Fig. 6.3.), but if there is an error in the instruction the reply is "negative acknowledge" (NAK).

The instructions (INST) sent between processors are arranged such that a positive instruction character indicates that it is only this instruction being sent (Fig. 6.3.(A)). A negative instruction indicates that the instruction is followed by 50 bytes of data (Fig. 6.3.(B)).

On reception of the 50 bytes of data, a block check character (BCC) is sent, which is the sum of the 50 data bytes, without regard to overflow. If this character agrees with the total calculated at the receiving processor, the ACK reply is sent which results in the end of transmission (EOT) reply from the sending processor.

As can be seen from these diagrams, the inter-processor communications protocol requires that every character sent (except EOT) requires a reply character (ACK,NAK or ENQ). Using this approach is ensures that characters are only sent, when the previous character has been read in by the receiving processor.

In the event of an error occurring during communications, there are protocol procedures to enable communications to be re-established. Incorporated within this protocol is the use of the interval timer in each TMS 9902. These interval timers are used to detect excessive delays in replies which would indicate a communication failure, or to cause a "time out" operation in the other processor if a reply character was incorrect (Fig. 6.4.(D)).

The "time out" feature is also used by the inter-processor communications protocol, to resolve the simultaneous transmission of the SOH character between processors (Fig. 6.4.(E)).

6.3. Communications Function

In the structure of the operator software shown in Fig. 6.1., those programs which form the communications function are the input, output and microprocessor/operator communications package tasks, and the queue and interrupt handling operating system programs. The function of these operating system programs, is the controlled activation of the input and output tasks when there is information available in any of the inter-processor communication UARTs, as described in Section 6.6.

6.3.1. Input Task

The communications specification for the operator-system, was that it should be capable of simultaneous communication with all patient processors, to reduce patient processor waiting time and allow efficient use of operator-system processing time.

To provide this facility, the input task was produced as a multi-input re-entry program which receives information from patient processors. The important features which influenced the structure of this program, and its control by the operating system, was that of the workspace area concept, and the interrupt handling features of the TMS 9900 microprocessor (Ref.58.).

Each inter-processor communication UART connected to the operator-system has associated with it workspace areas and memory files, as shown in Fig. 6.5. The first of these workspace areas is used by the operating system UART interrupt handling program, and the second by the input or output task. It is the address of this second workspace area that is supplied to the input task, by the operating system queue handler program, when a character is present in the associated UART. The contents of this workspace area supplies all the information required by the input task, such as the CRU base address of the UART associated with that workspace area, and the state of the communications protocol for that channel (given by the branch pointer in R1).

A simplified flow diagram of the input task is given in Fig. 6.6. From this diagram it can be seen that the program is divided into 5 sections, each one related to a different aspect of the communication protocol. The branch pointer (R1) in the workspace area supplied to the input task, indicates which section is to be performed when removing the character from the

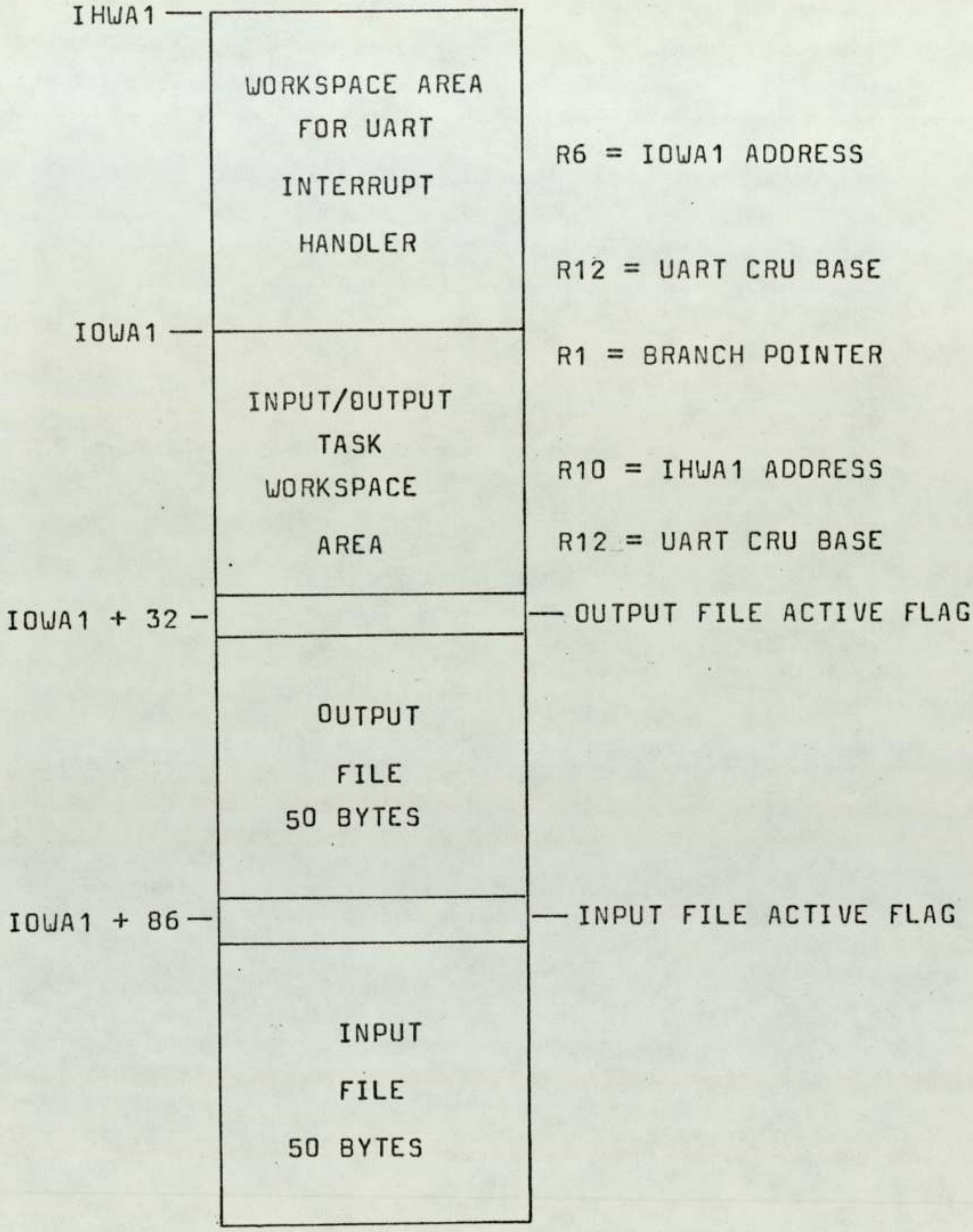


Fig. 6.5. WORKSPACE AREA & FILE STRUCTURE FOR EACH INTER-PROCESSOR COMMUNICATIONS CHANNEL.

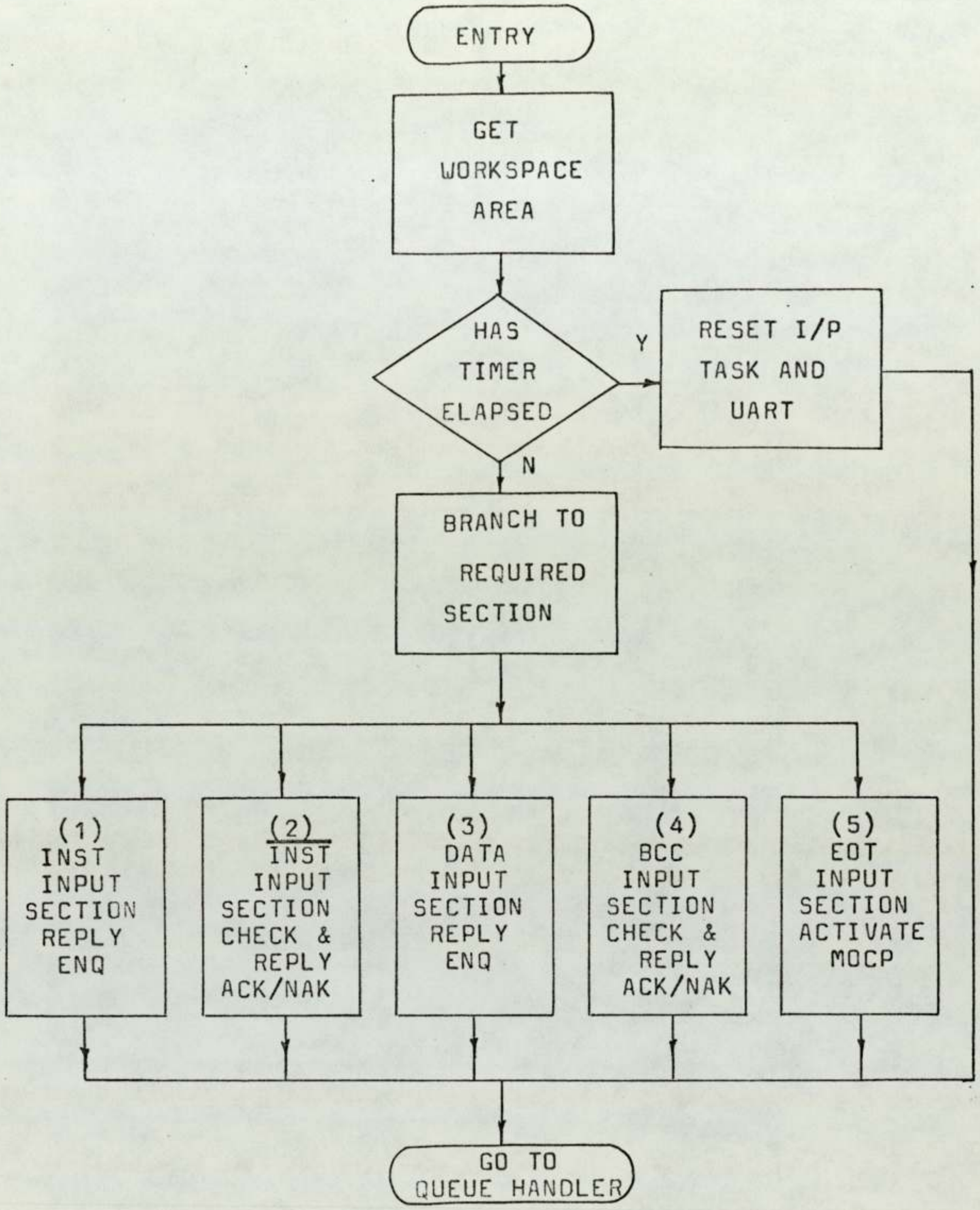


Fig. 6.6. BASIC FLOW DIAGRAM OF INPUT TASK.

appropriate UART, and placing it in the required input file. On completion of each section, the branch pointer is set to indicate which section is to be used, when the next character is received. In this way the input task can deal with each UART as the characters are received.

On successful completion of communications with a particular patient processor (Fig. 6.6. Section 5) the input task sets the corresponding input file active (Fig. 6.5.) and activates the microprocessor/operator communications package task (MOCP).

The execution time of this task, for one entry to it during the communications process, is approximately 0•2mS.

<u>6.3.2. Output Task</u>

The output task is concerned with the transmission of information from the operator processor to patient processors. As can be seen from Fig. 6.7., the structure of the output task is similar to that of the input task. In this case, however, there are six different sections associated with the transmission of information from output files (Fig. 6.5.).

Section 3 (Fig. 6.7.) is concerned with the reception of the ACK character (or NAK), and the transmission of the EOT character if no data is to be sent. If, however, data is to be transmitted, this is performed by sections 4 and 5, and the ACK character then received by Section 6 results in the EOT character

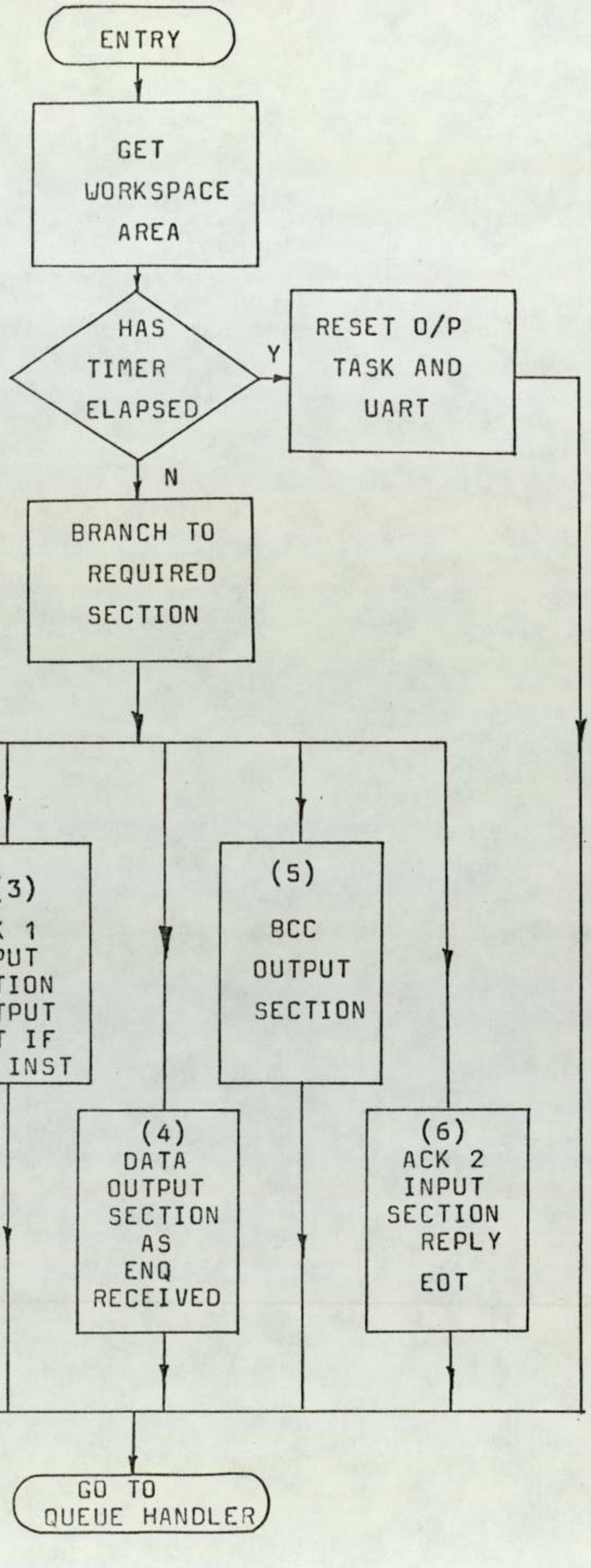


Fig. 6.7. BASIC FLOW DIAGRAM OF OUTPUT TASK.

being sent.

To indicate that the information contained in an output file has been sent, the output file active flag associated with it is reset; it having been set previously when data was placed in the file.

This approach results in the output task being active, only when a reply is present from a patient processor. This therefore, allows lower level tasks to be performed while the output task is awaiting replies.

As can be seen from Fig. 6.1., the output task has a lower priority than the input task. This is because the information coming from patient processors is real-time dependent, whereas output information from the operator-processor can be saved until an appropriate communication situation prevails.

The execution time of the output task, for a single character transmission is approximately 0·2mS.

6.3.3. Microprocessor/Operator Communications Package

The microprocessor/operator communications package (MOCP) task performs the following functions :-

(1) The interpretation of commands from the operator, and the execution of the required action. This is concerned with the issuing of instructions to the patient processors concerned.

(2) The interpretation of input information from patient processors. This information may be informing the operator-processor that a certain operation (such as

the successful initiation of the monitoring function) has been performed, or data which had been requested by the operator system.

The flow diagram in Fig. 6.8. shows the basic structure of the microprocessor/operator communications package task. As can be seen from this diagram, the task is divided into two sections concerned with the interpretation of operator or patient processor inputs. The mnemonic commands which the operator can give to the present system, via a VDU terminal, are given in Table 6.1. As can be seen, these instructions take the form of a two letter mnemonic command followed by the patient processor number concerned.

The inter-processor instruction codes (INST) which are currently in use, are given in Table 6.2. These codes were chosen from the chart shown in Table 6.3., which illustrates that the small region containing the ASCII codes used, has not been initially used to provide instruction codes. The remaining codes are divided into the two regions of positive (single instruction) codes, and negative (instructions with data) codes. It will also be seen that a pattern of codes have been eliminated for immediate use (those containing an X), leaving the remaining codes, which require a 2 bit change to give another allowed code, to be used first.

6.4. Display

The display function is concerned with the presentation

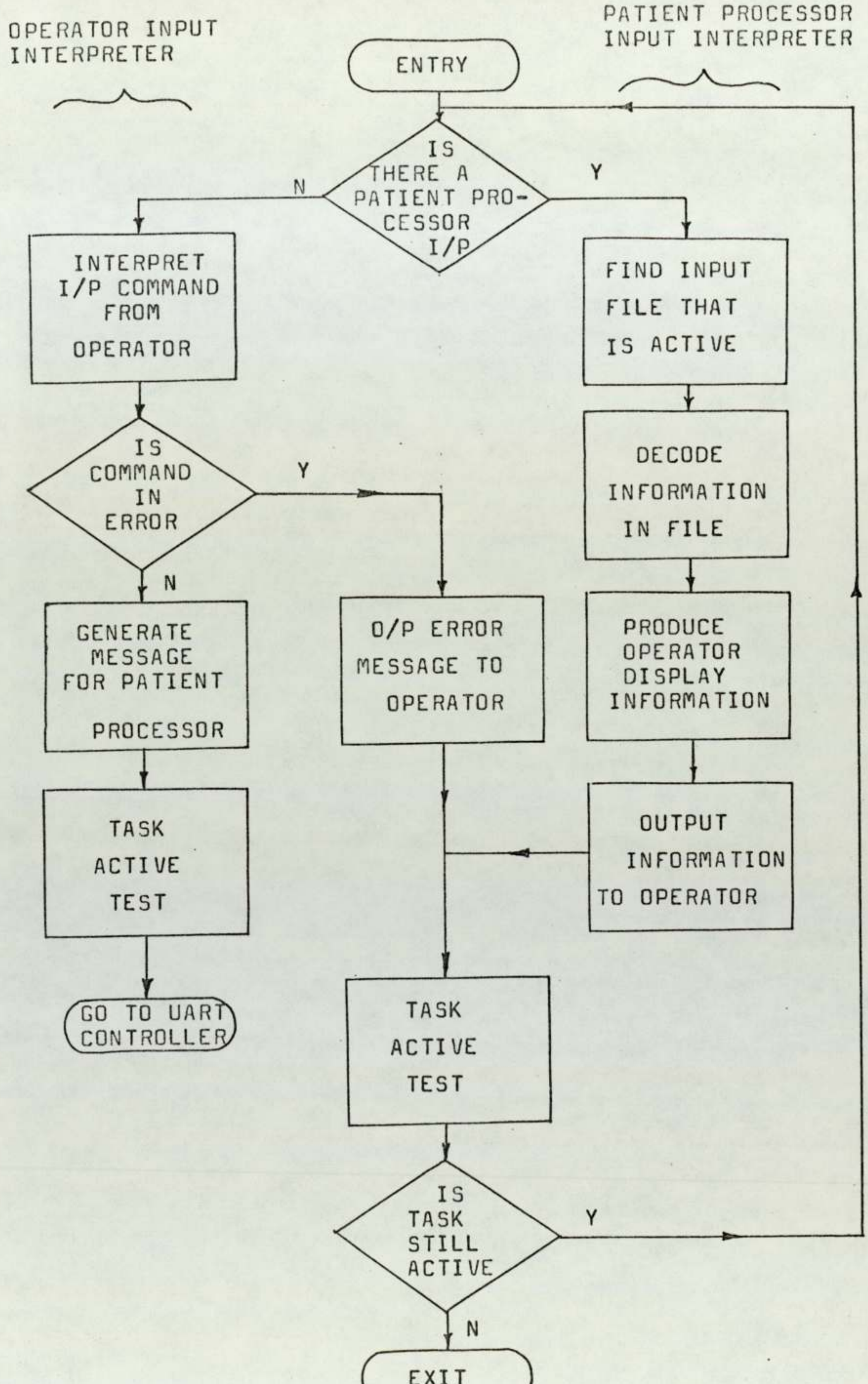


Fig. 6.8. BASIC FLOW DIAGRAM OF MICROPROCESSOR/ OPERATOR COMMUNICATIONS PACKAGE PROGRAM.

START MONITORING	SM1
END MONITORING	EM1
CONTINUE MONITORING	CM1
PATIENT STATUS	PS1
PULSE RATE	PR1

TABLE 6.1. OPERATOR INPUT COMMANDS

Instructions from Operator Processor to Patient Processor

		HEXADECIMAL CODES
OP1	START MONITORING	50
OP2	END MONITORING	53
OP3	CONTINUE MONITORING	55
OP4	SEND PATIENT STATUS DATA	56
OP5	SEND PULSE RATE	5C

Instructions from Patient Processors to Operator Processor

		HEXADECIMAL CODES	
		SINGLE INSTRUCTION	INSTRUCTION WITH DATA
PO1	MONITORING	21	
PO2	MONITORING START-UP OK	22	
PO3	MONITORING START-UP FAIL	24	
PO4	END OF MONITORING	27	
PO5	MONITORING AGAIN	28	
PO6	INSTRUCTION ERROR	2B	
PO7	MONITORING ALREADY	2D	
PO8	NOT MONITORING	2E	
PO9	PATIENT STATUS DATA		81
PO10	DIAGNOSIS		A3
PO11	PULSE RATE DATA		CO

TABLE 6.2. INTER-PROCESSOR INSTRUCTION CODES

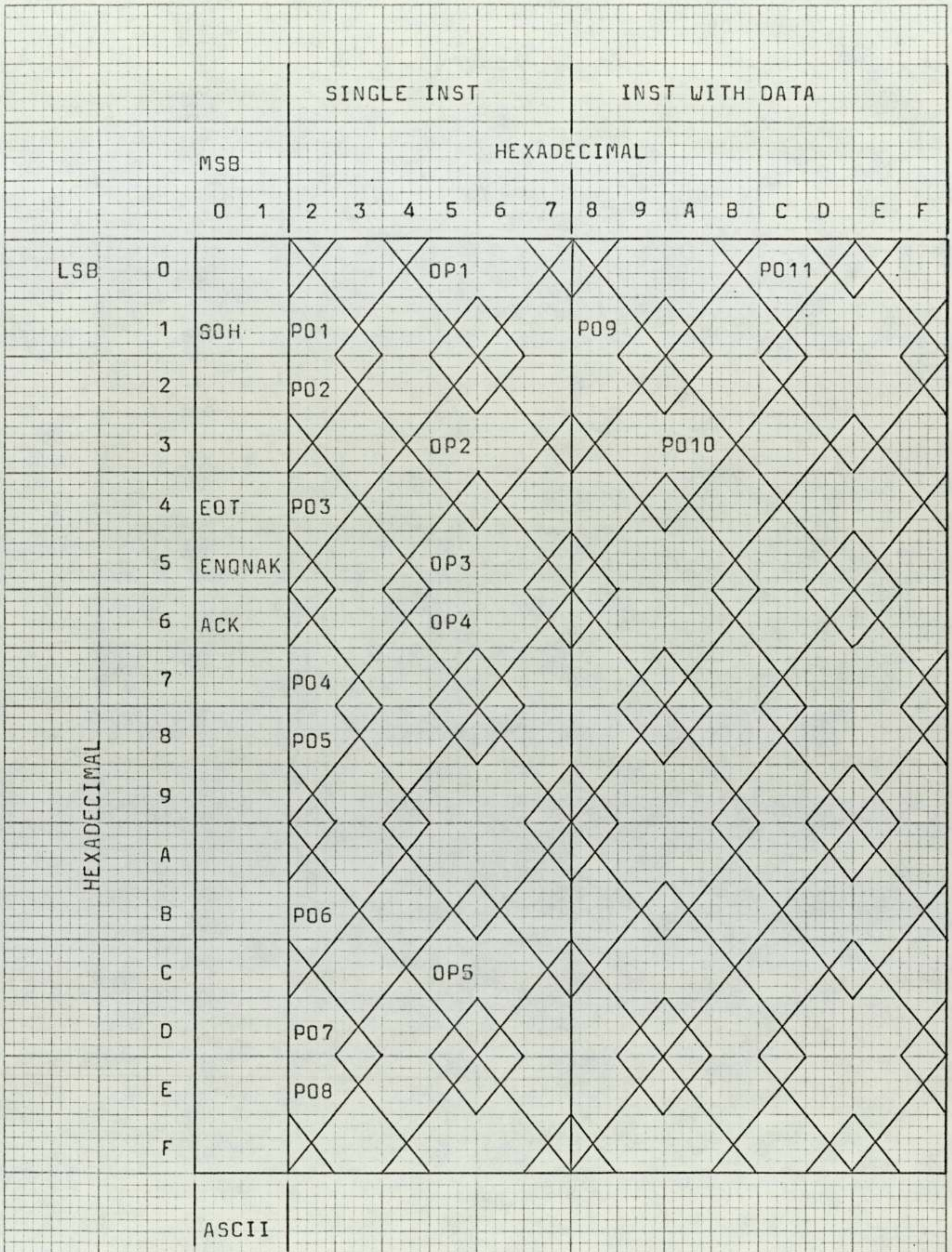


Table. 6.3. INSTRUCTION CODE TABLE.

of information to the operator, which in the present system is performed by a visual display unit (VDU). The software which controls this peripheral, is the VDU task (Fig. 6.1.). This task performs both the functions of outputting information to the VDU, and receiving input commands from the operator, via the VDU keyboard. The output section of the task has priority over the input section, thereby allowing the system to interrupt the operator to indicate a patient's condition. The basic flow diagram of the input and output sections of the VDU task, are shown in Fig. 6.9. and Fig. 6.10.

When there are no active tasks, the operating system passes control over to the input section of the VDU task. The characters received by this task are placed in an input file, and on reception of a carriage return character, the VDU task activates the MOCP task, so that the contents of this buffer may be decoded.

The output section of the VDU task may be entered in two ways, the first is by scheduling the task, and the second is by direct access to the task. The second of these approaches is mainly used in the system, as it allows the MOCP task to output information as it is generated.

The information supplied to the VDU task, comes in two forms (Fig. 6.11.). The first is a list of addresses indicating the files to be used. In this case the last character in each file is either negative; indicating that the next file address should be used, or the

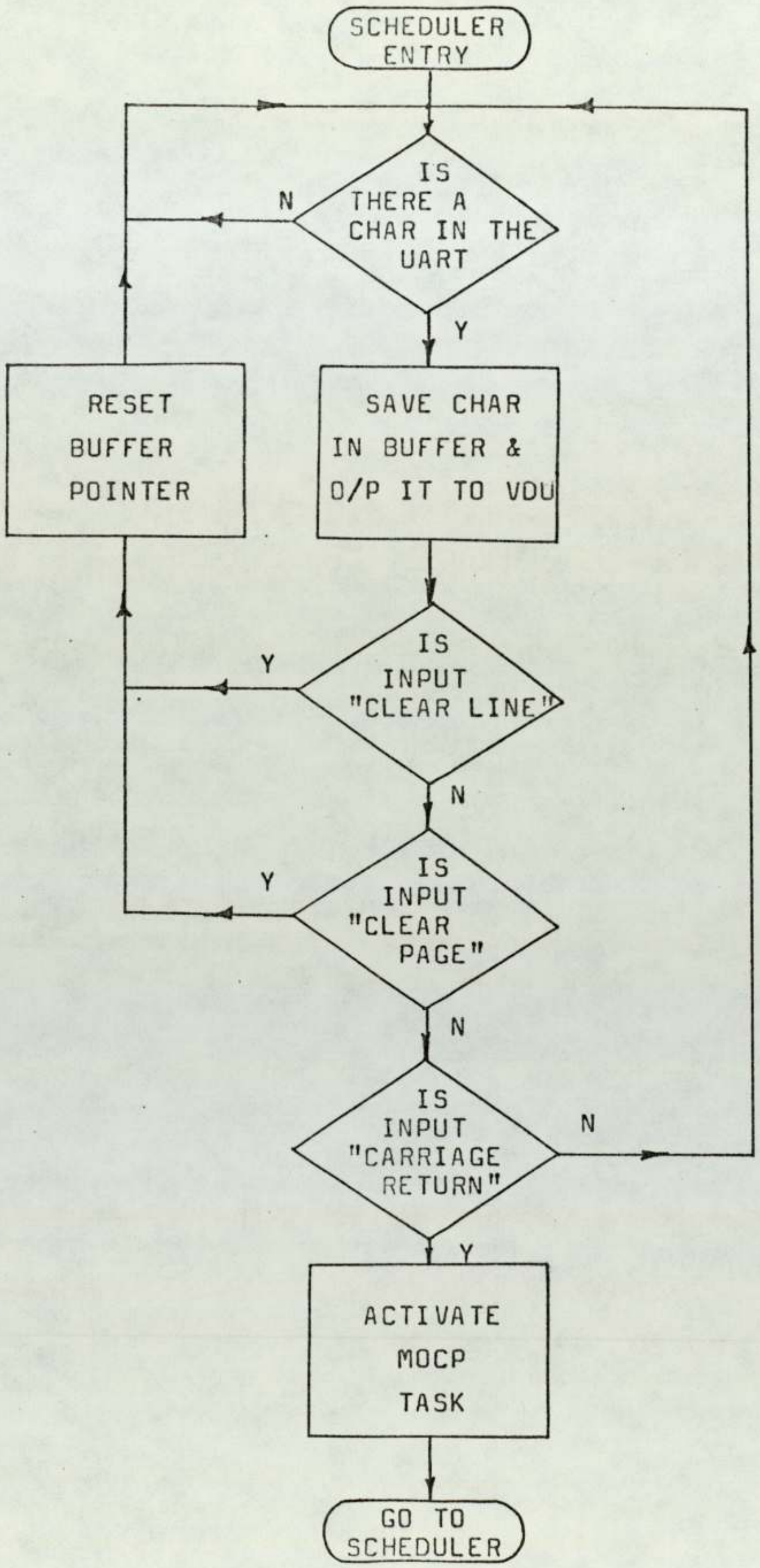


Fig. 6.9. BASIC FLOW DIAGRAM OF INPUT SECTION OF VDU TASK.

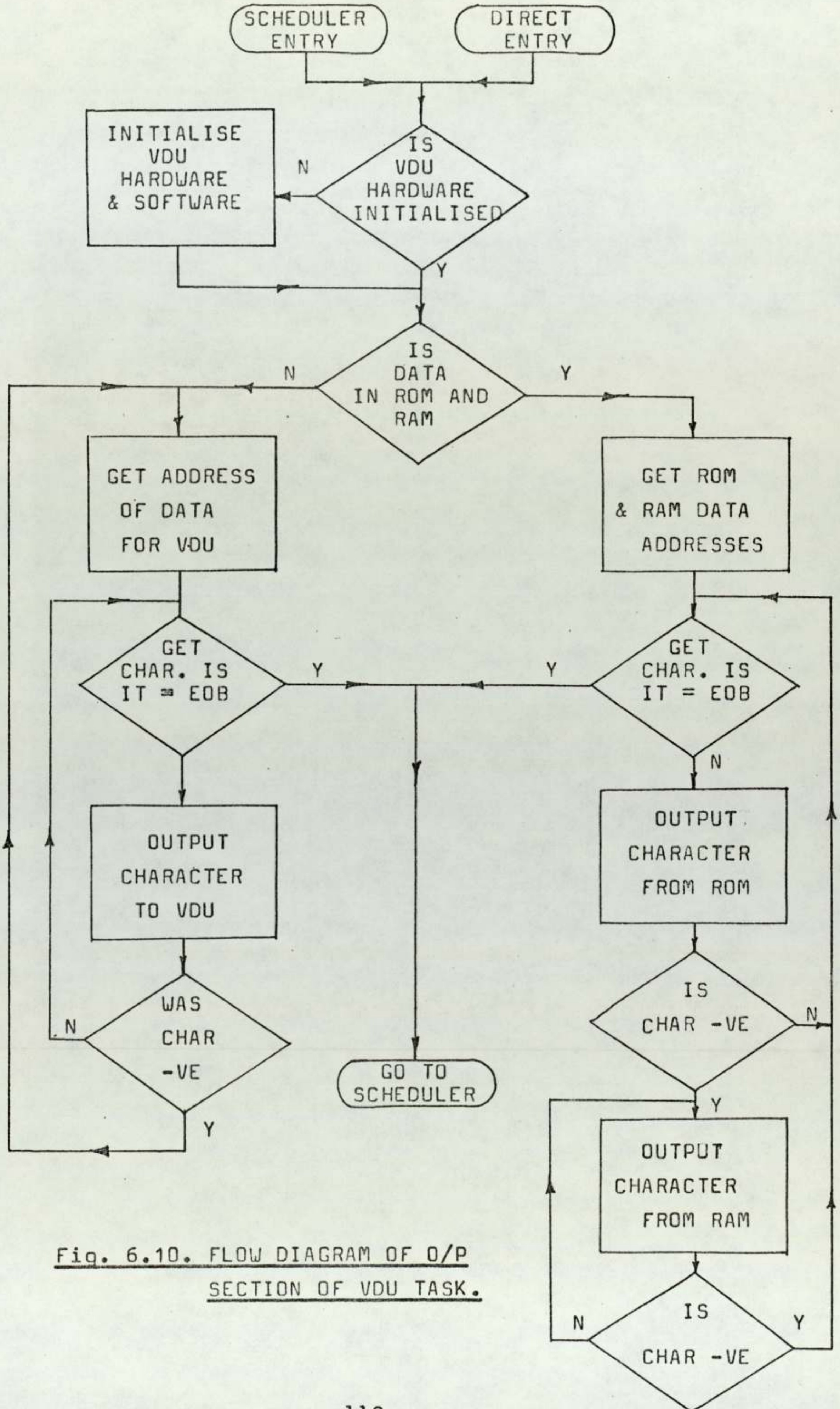


Fig. 6.10. FLOW DIAGRAM OF O/P SECTION OF VDU TASK.

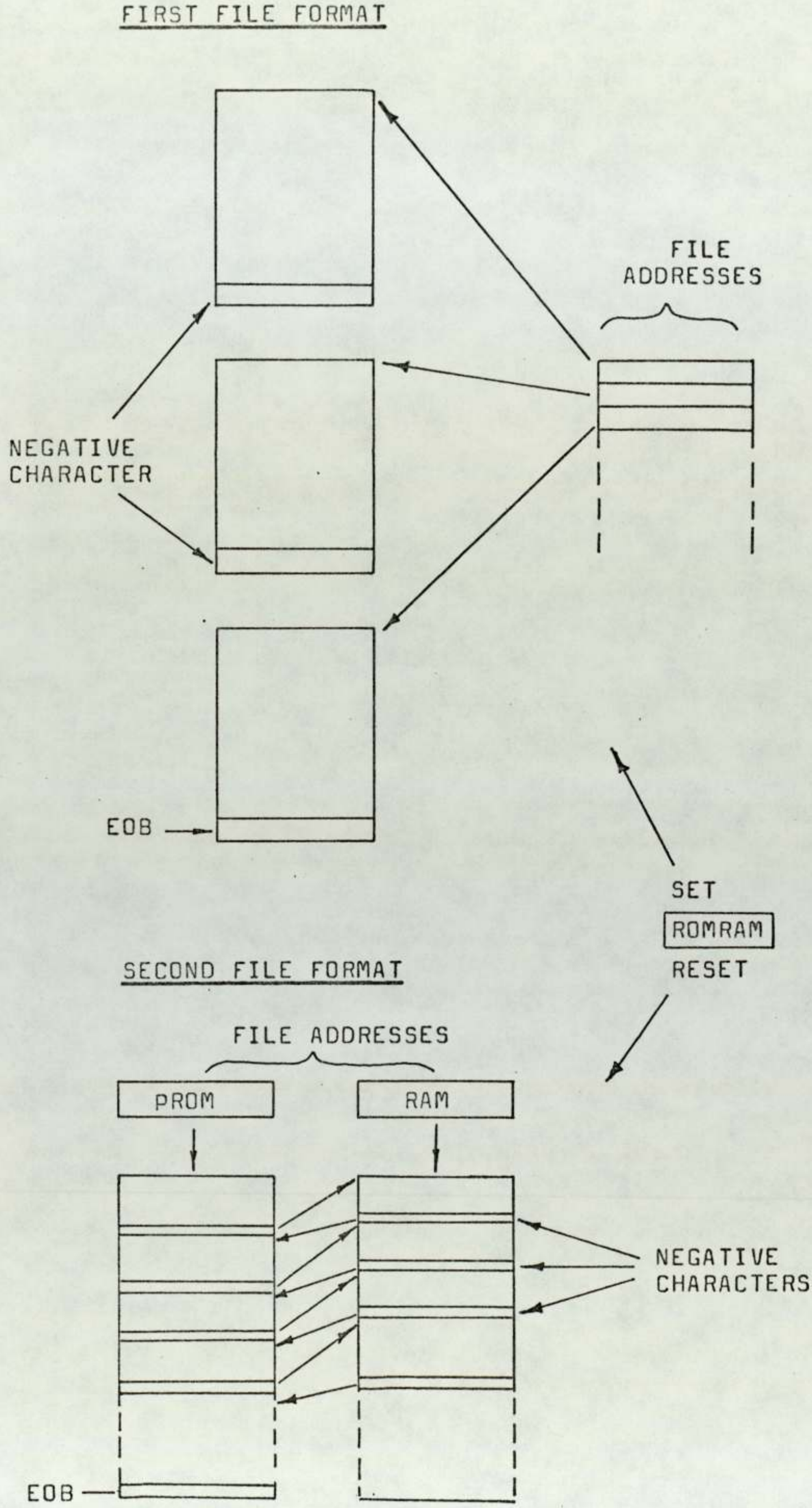


Fig. 6.11. OUTPUT FILE STRUCTURES FOR VDU TASK.

"end of buffer" (EOB) character, which indicates that there are no more file addresses.

However, it is often required that standard text and variable characters be mixed on the VDU display. In these situations, the standard text would be in "programable read only memory" (PROM), and the variable characters would be present in read/write memory (RAM). In order to cater specifically for this situation, an alternative file format structure was developed. In this case only the address of PROM information and RAM information is given to the VDU task. The PROM resident characters are structured such that, at the points where variable characters should be present on the display, the character is made negative. Similarly, in the RAM generated information, at the point where the VDU task should return to PROM information, the character is made negative. The operation of the VDU task with this file structure, is simply to jump from one file to the other, when a negative character is detected, or to terminate this output operation on the detection of the "end of buffer" character in the PROM file (Fig. 6.11.). To indicate which file format is in operation, a software flag (ROMRAM) is set (first format) or reset (second format) accordingly.

6.5. Storage

The storage function was not implemented in the experimental system, as no definite decision was made as to what data should be saved, and also because of the cost of peripheral equipment such as a disc store.

However, the type of data that could be stored by the system may take the form as follows.

(1) The storage of the first arrhythmia of each type diagnosed.

(2) The storage of one or more, of the most recent arrhythmias, of each type diagnosed.

(3) The storage of trend data such as the number of each type of arrhythmia detected, in perhaps 1 or 5 minute intervals over a number of hours.

(4) The storage of medication events.

6.6. Control

The operating system for the operator-dedicated processor software, as shown in Fig. 6.1., is similar to that of the patient processor system. The functions of the initialisation and task scheduler are the same in each system, the only differences being the task involved.

The major differences between the two operating systems is concerned with the UART interrupt handler and input/output queue handler programs.

6.6.1. UART Interrupt Handler Program

The UART interrupt handler program for the operator-dedicated processor system, is responsible for the control of all the inter-processor communication UARTs. It is therefore concerned with establishing or acknowledging, inter-processor communication start-up protocol procedures, and the subsequent supply of information to the I/O queue handler program, which is responsible for activating the input or output tasks.

As illustrated in Fig. 6.5., each inter-processor UART, has associated with it a workspace area, which is used by the UART interrupt handler program. It is these workspace areas; supplied to the program via interrupt vectors, which contain information that informs the program of the state of communications with the associated patient processor (i.e. whether the UART is being currently used for inputting or outputting of information etc.).

When a character is received by a UART, the interrupt produced causes the UART interrupt handler program (Fig. 6.12.) to be performed. If this program finds that the particular UART concerned is in either the input or output mode, it supplies the address of the I/O workspace area associated with that UART (Fig. 6.5.), to the I/O queue handler program. This workspace area is then supplied by the I/O queue handler program, to either the input or output task as required.

Also included within the UART interrupt handler program are the sections concerned with :-

(1) Acceptance of output requests from the MOCP task.

(2) Testing for suspended output requests from MOCP.

(3) The retransmission of the start-up protocol (SOH) due to an error detected by the output task.

(4) The activation of the "timing out" process due to

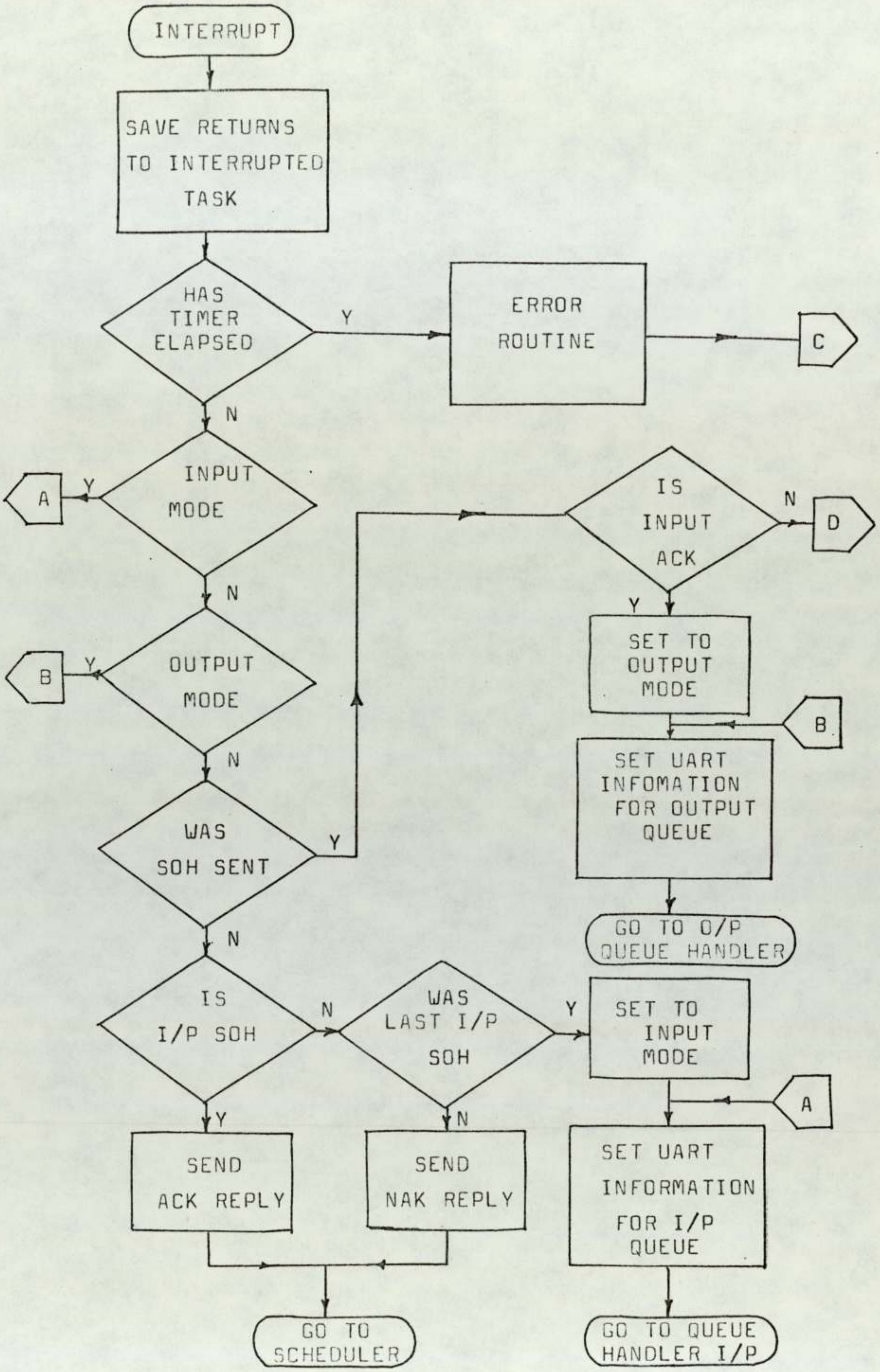


Fig. 6.12. BASIC FLOW DIAGRAM OF UART INTERRUPT HANDLER PROGRAM.

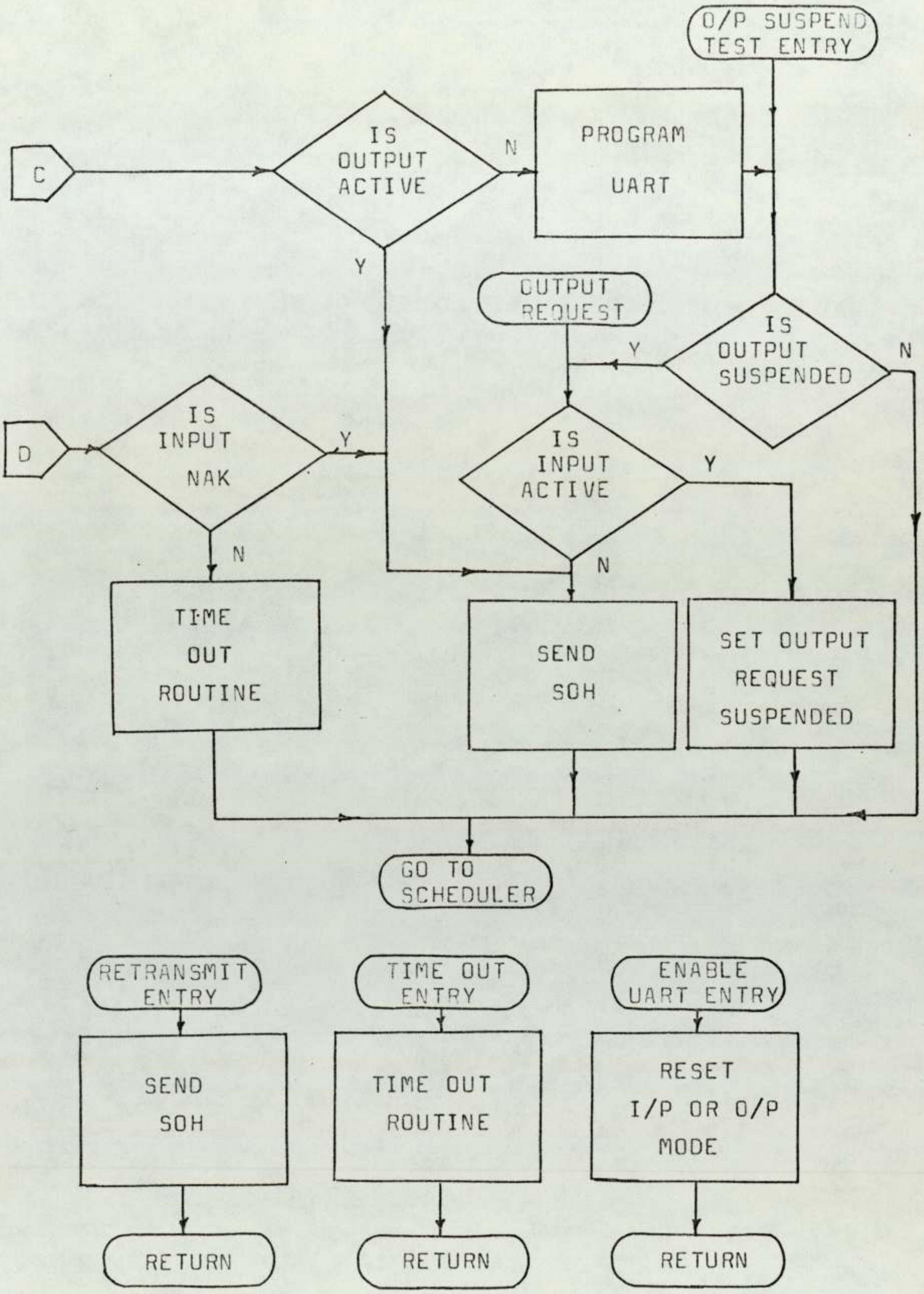


Fig. 6.12. continued.

BASIC FLOW DIAGRAM OF UART INTERRUPT HANDLER PROGRAM.

the reception of an illegal reply character detected by the output task.

(5) The enabling of the UART interrupt handler program on successful completion of input or output communications.

6.6.2. I/O Queue Handler Program

The I/O queue handler program is concerned with the activation of the input and output tasks, as UARTS receive characters for those tasks. Therefore, for example, if a character is received by a UART which is in the input mode, the UART interrupt handler program will supply the I/O workspace area address associated with that UART to the input queue handler routine. The queue handler program places this I/O workspace area address in a, first in first out (FIFO) input queue, and then ensures that the input task is active. If other UARTs are in the input mode and have received characters, the addresses of the I/O workspace areas associated with those UARTs are placed in the input queue. Subsequently, the input task is scheduled by the task scheduler program whereupon it uses the first of the I/O workspace area addresses given to it by the I/O queue handler program. On completion of the input task for that particular I/O workspace area, the input task returns to the queue handler program so that the next address of an I/O workspace area in the input queue, may be given to the task. If it is found by the queue handler program, that there are no more I/O workspace area addresses in the input queue, the input task is deactivated until such time as more characters have been received, for use by the input task.

When UARTs are in the output mode, the output queue and output task are handled in a similar manner by the output queue routines of the I/O queue handler program. The basic flow diagram of the I/O queue handler program is given in Fig. 6.13.

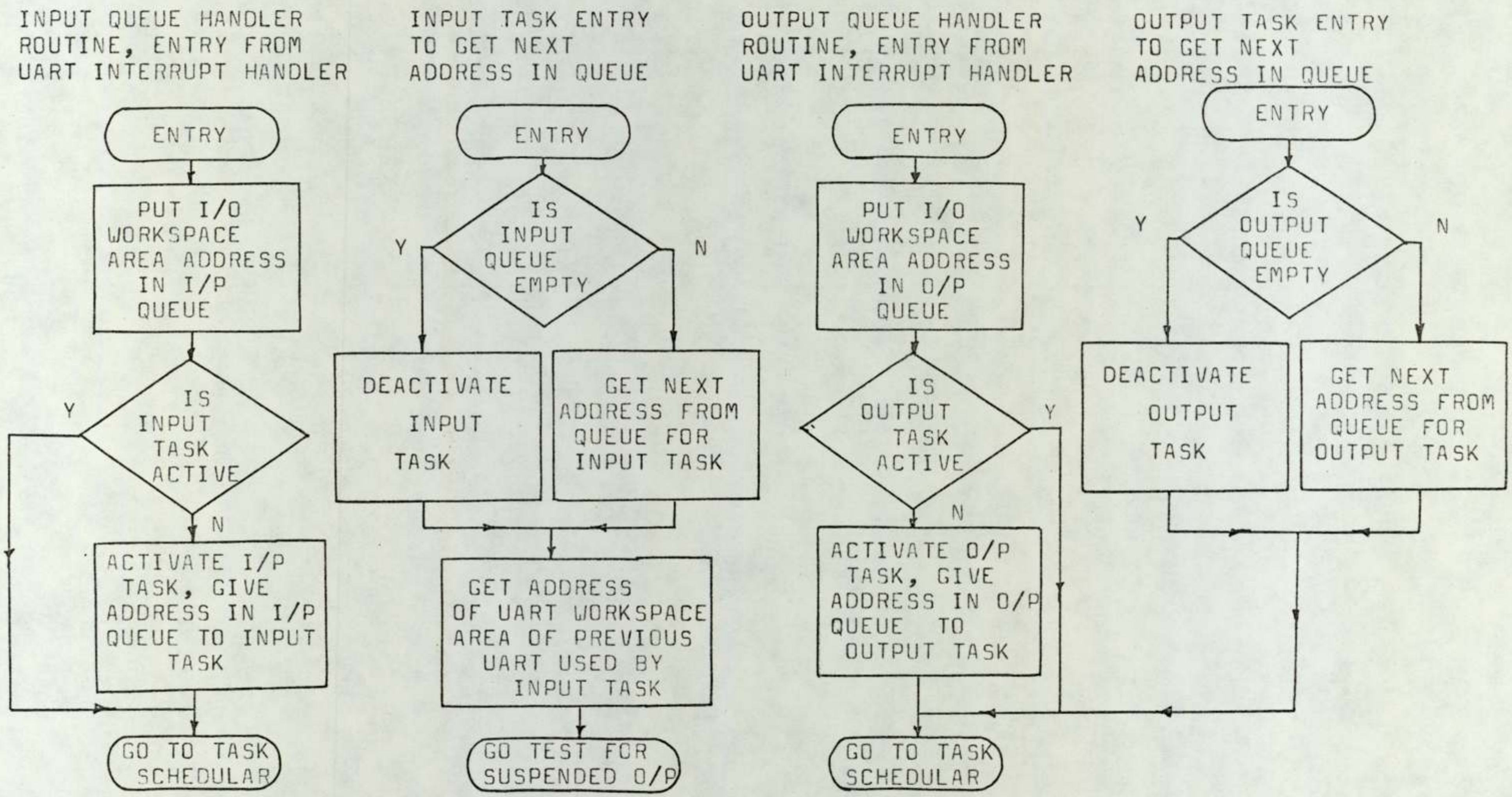


Fig. 6.13. BASIC FLOW DIAGRAM OF I/O QUEUE HANDLER PROGRAM.

CHAPTER 7

SYSTEM HARDWARE

7.1. General

When producing the various functions to be performed by the distributed system, the aim was to implement as much as possible in the compact form of software. As a result, it should therefore be possible to produce a single patient-dedicated processor system board, which would contain all the hardware requirements for a single patient. Such an approach would allow flexible system structure for different patient size CCUs, and would also aid mass production of the monitoring systems.

The basic block diagram of a microprocessor hardware system, is shown in Fig. 7.1., and a brief outline of the hardware and software features of the TMS 9900 microprocessor are given in the appendix. The only difference between the patient and operator dedicated processor hardware systems, is the size of memory (PROM and RAM) required, and the devices interfaced to each system.

In the present developmental system, the memory requirements for the patient system are met by 2K words of PROM and 2K words of RAM. The present operator system memory, does not exceed 2K words of PROM and 1K words of RAM, this being the memory required to implement the three patient (simulation) system. However, the memory requirements for the final operator system would be greatly increased, as more patient processors would be

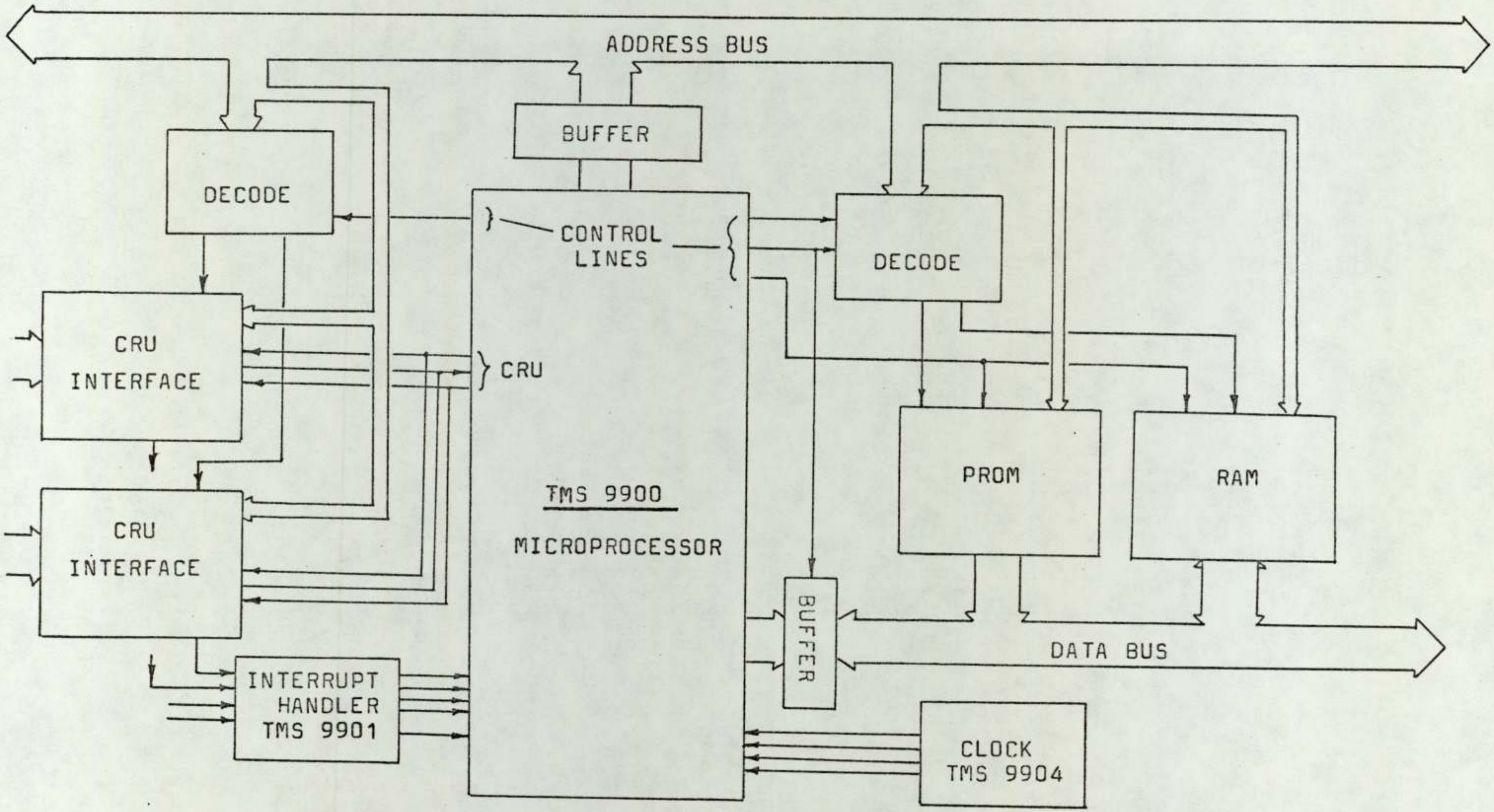


Fig. 7.1. BASIC HARDWARE BLOCK DIAGRAM OF A MICROPROCESSOR SYSTEM.

connected to the final system, and more software for the presentation of information would be incorporated into the system.

All the memory used in the system has an access time not in excess of 450nS, so that the microprocessors were never placed in a wait state, while waiting for data from memory.

As new components such as memory devices, and microprocessor support chips are continuously appearing on the market, no detailed circuit diagrams of such items will be given in this report. For information on such devices one should consult the latest microprocessor support components literature, and new devices specifications.

The following discussion will therefore be directed to the interface requirements for A to D conversion of the ECG signal, and inter-processor communication.

7.2. ADC Interface

The analog to digital converter (ADC), and the digital to analog converter (DAC), designed for use in the developmental system, are both interfaced to the patient processor system. In the final distributed system it is possible that the DAC would be dedicated to the operator system, allowing the display of stored arrhythmia events.

The requirements of the ADC interface are that it should convert the ECG signal to an 8 bit signed binary value, every 4mS. A basic block diagram of the ADC interface designed is given in Fig. 7.2. It operates independently from the microprocessor such that every 4mS the ECG signal would be automatically sampled and converted, and then the value found would be presented to the microprocessor, by the generation of an interrupt. Conversion of the signal was achieved by the use of an inexpensive 8 bit DAC, which produced a ramped output which was compared to the sampled analog signal.

Having produced the software for the patient processor system, it was found that there is scope for expanding the various functions. One way in which this surplus processing time could be put to use, is in order to reduce the component count of the ADC. This could be achieved by replacing the control in the present interface (Fig. 7.2.), with control from the microprocessor, producing the structure for the interface shown in Fig. 7.3. The microprocessor could then instruct the interface to convert the signal, or possibly use a successive approximation algorithm itself to convert the signal. Timing of the 4mS sample period could then be performed by the interval timer present in the TMS 9901 interrupt handler device (Fig. 7.1.), which is now available and would be present in the system.

7.3. Inter-processor Communication Interface

As indicated in Section 6.2. (Fig. 6.2.), the approach used for inter-processor communication is via serial data

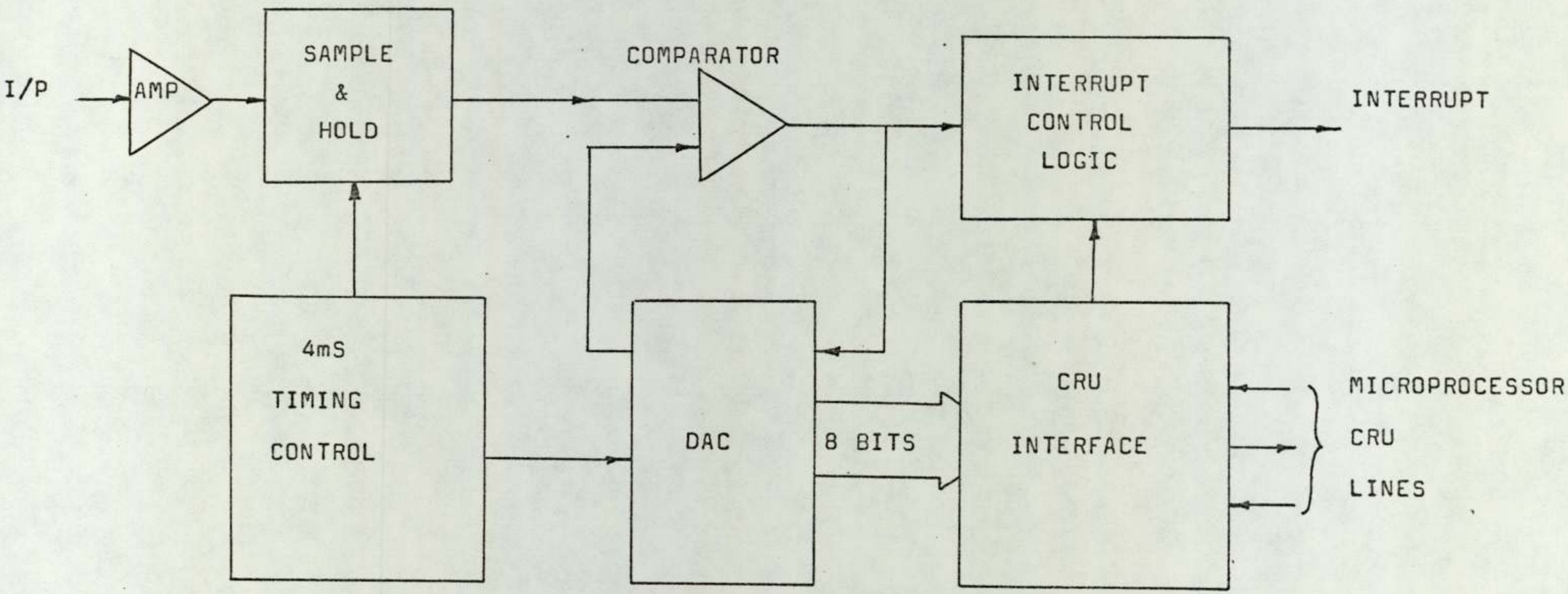


FIG. 7.2. BASIC STRUCTURE OF ADC USED IN DEVELOPMENTAL SYSTEM

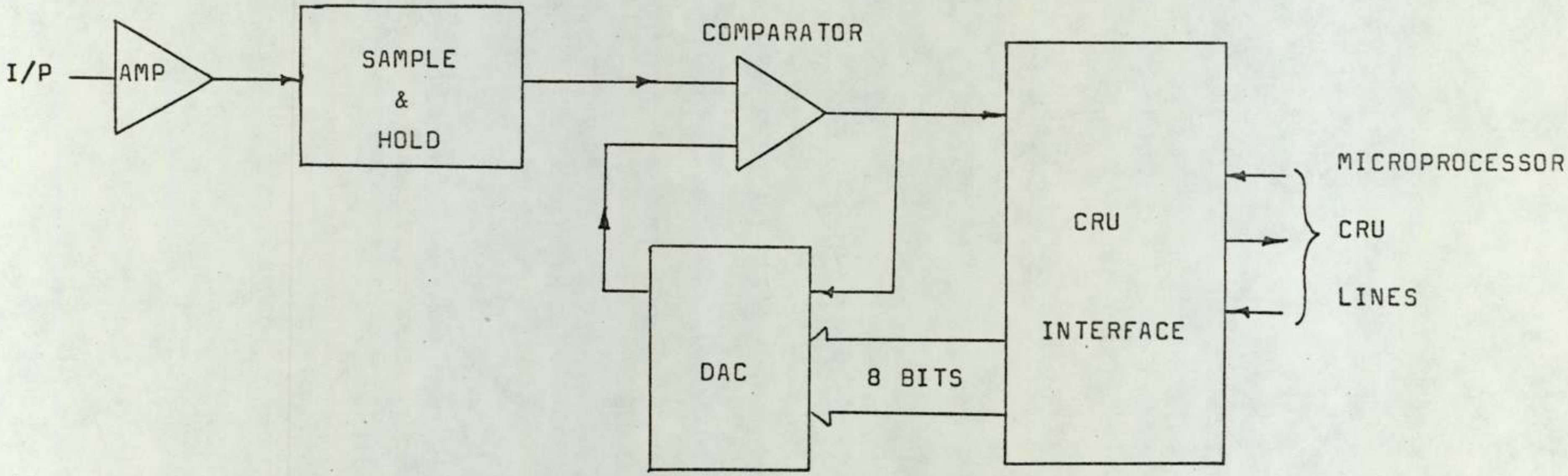


FIG. 7.3. ALTERNATIVE STRUCTURE OF ADC

links. The device used to provide the communication link is the TMS 9902 asynchronous communications controller, which is specifically designed for use with the TMS 9900 microprocessor.

The structure of the communications interface for the operator system is shown in Fig. 7.4. This diagram shows 3 communication channels but more TMS 9902's could easily be introduced into the interface, to provide communications with the final number of patient processors in the distributed system.

At the patient-dedicated processor systems, the communications interface is the same as in Fig. 7.4., but only one TMS 9902 is required.

The character transmission format, programmed into each TMS 9902 by the associated microprocessor at system start-up time, is one start bit, eight data bits, an even parity bit and two stop bits. The bit rate currently used in the system is 31250 bits/second, which corresponds to a character transmission time of 0·384mS.

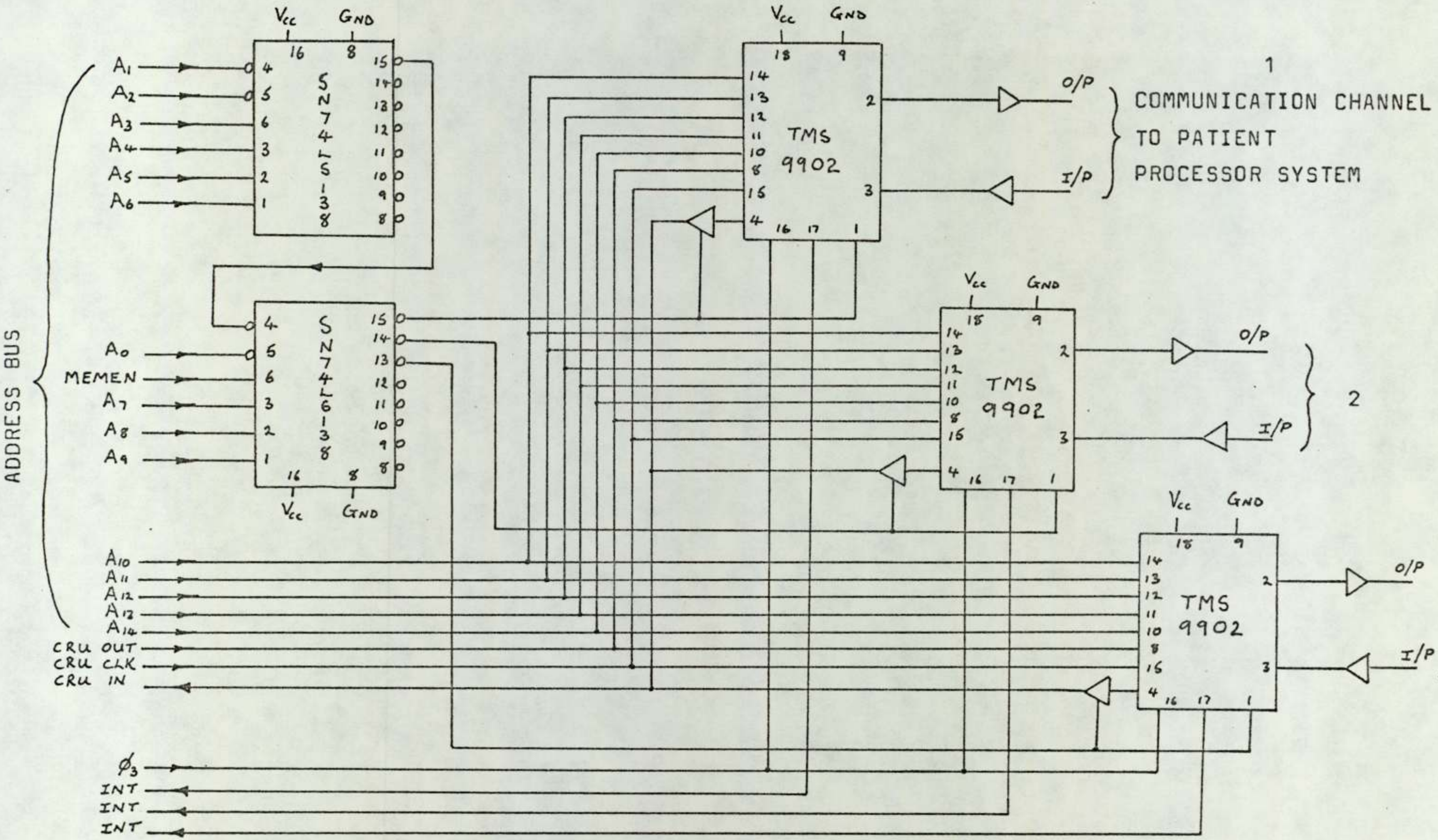


Fig. 7.4. CIRCUIT DIAGRAM OF OPERATOR SYSTEM COMMUNICATION INTERFACE.

CHAPTER 8

SYSTEM PERFORMANCE

8.1. Patient Processor Performance in Real-time

In the patient processor system, the real-time functions of signal conditioning, monitoring, diagnosis and control, consume approximately 42% of processing time (Table 4.1.). The remaining 58% of processing time is unused, except when communications between processors occurs. This surplus processing time therefore allows any of the patient processor functions to be expanded. However, the amount by which each function could be increased is dependent upon its frequency of execution.

If the execution time of the signal conditioning function (1mS) or the QRS detection algorithm (0·21mS) in the monitoring function were to be increased, the result would be a constant increase in the processing time consumed. This is because the frequency of execution of these algorithms is constant, at 250 per second. Therefore, for example, if both of these algorithms were increased by 50%, the processing time consumed would increase from 42% to approximately 56%.

If, however, the execution time of the monitoring function parameter measurement algorithms, or the diagnosis function were increased, the effect on processing time would be dependent on the pulse rate of the patient being monitored. For example, if these algorithms were increased by 10 or 20 times, the effect on the processing time consumed, related to the patient's heart rate, would be as shown in Fig. 8.1. As can be seen from this graph, even a substantial increase in these algorithms, and a very high pulse rate, does not cause an excessive increase in the processing time consumed.

Therefore, it should be apparent that the frequency of execution of each algorithm, is the factor which limits the amount by which each may be expanded, but within this limit there is still room for considerable improvement. However, the penalty paid for increasing the processing time consumed by these functions, is a reduction in the speed of information flow between processors, as discussed in Section 8.2.

8.2. Inter-processor Communications Performance

As only a single patient processor system was available for use in the experimental system; simulation of a three patient processor system was performed, to determine the performance of inter-processor communications.

The simulation of a three patient system was achieved by connecting the single patient processor system that was available to three inter-processor communication UARTs, as illustrated in Fig. 8.2. The outputs to the imaginary patient processors were unused, and the interrupts from the UARTs were arranged such that the real patient processor had the lowest interrupt level. This arrangement produced the worst case conditions for communications to three patient processors, because

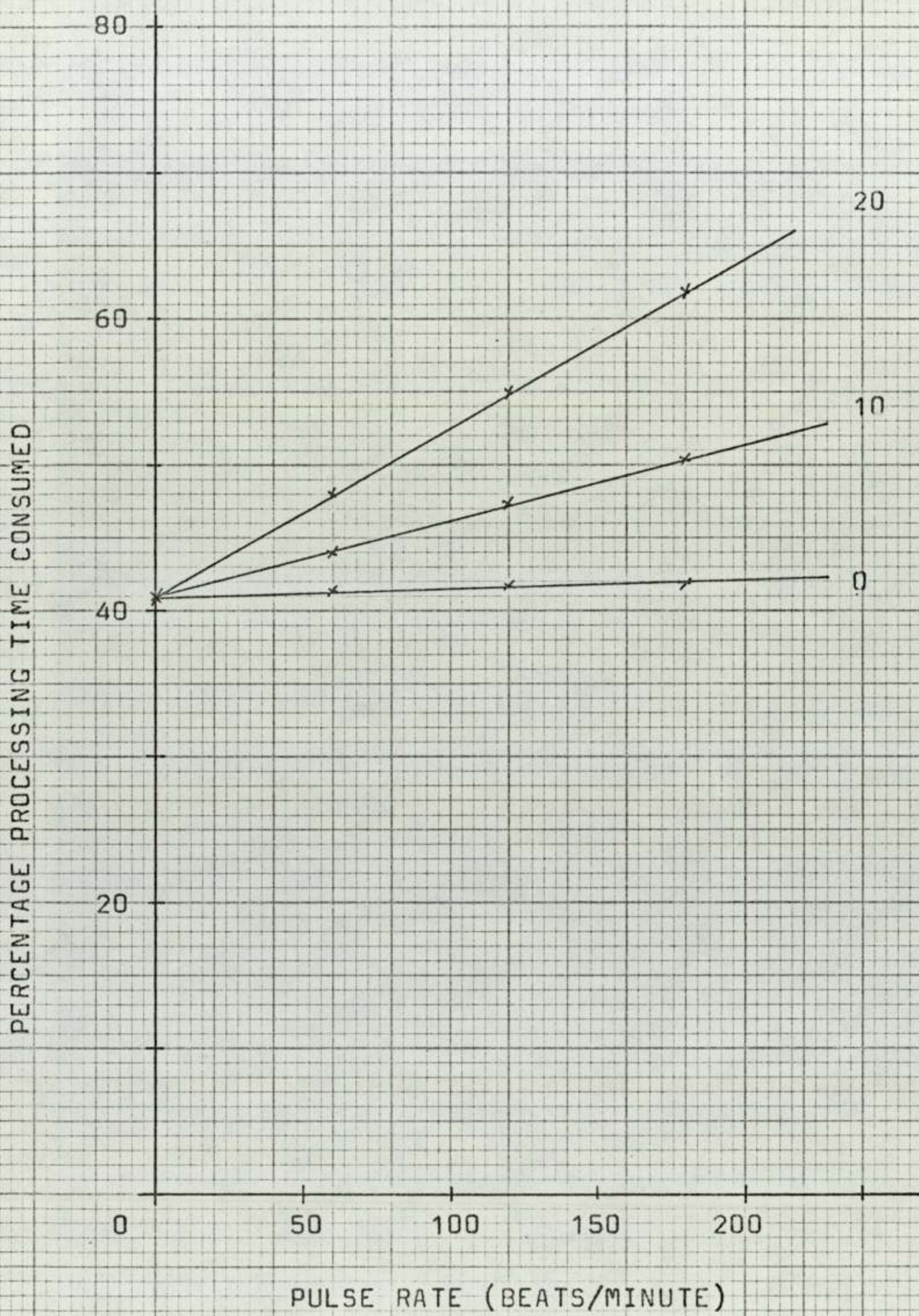


Fig. 8.1. GRAPH SHOWING EFFECT OF 10 & 20 TIMES INCREASE IN PULSE RATE RELATED ALGORITHMS.

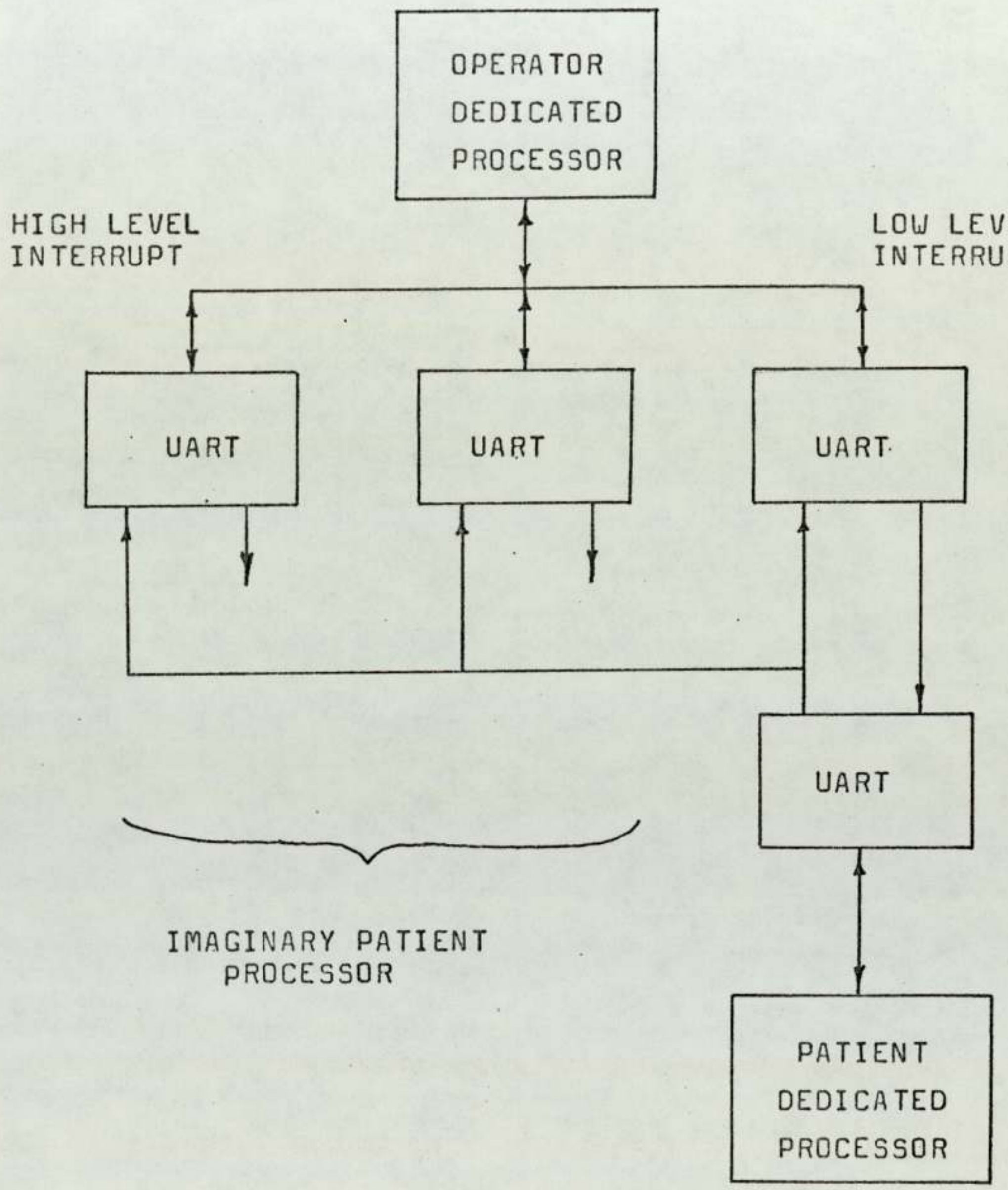


Fig. 8.2. BASIC STRUCTURE OF PATIENT PROCESSOR SIMULATION SYSTEM.

characters from the patient processor were received simultaneously in the UARTs at the operator system. This allowed the interrupt and queue handling facilities in the operator processor operating system to be tested, and measurements of communication rates for 1, 2 and 3 patient processors to be made.

The first experiments performed were to determine the effect of reducing the bit rate used by the UARTs, on the total transmission time of an instruction with 50 bytes of data (Fig. 6.3.(B)). This was performed with and without the monitoring function in operation in the patient processor system, and the results obtained are shown in Fig. 8.3. and Fig. 8.4. As can be seen from these graphs, some odd shaped curves were produced, which appeared to have discontinuities in them at regular intervals. These discontinuities were found to be due to the effect of the real-time processing performed by the patient processor, on the flow of information between processors. The introduction of the monitoring function (Fig. 8.4.) had the effect of increasing transmission times, but the points where the discontinuities occurred (70,110,220 mS) remain unchanged.

Consider the timing diagram given in Fig. 8.5., which shows the execution times of the signal conditioning and monitoring functions, and the transmission times of the characters transmitted and received by the patient processor. From this diagram it can be seen that the real-time processing performed by the patient processor,

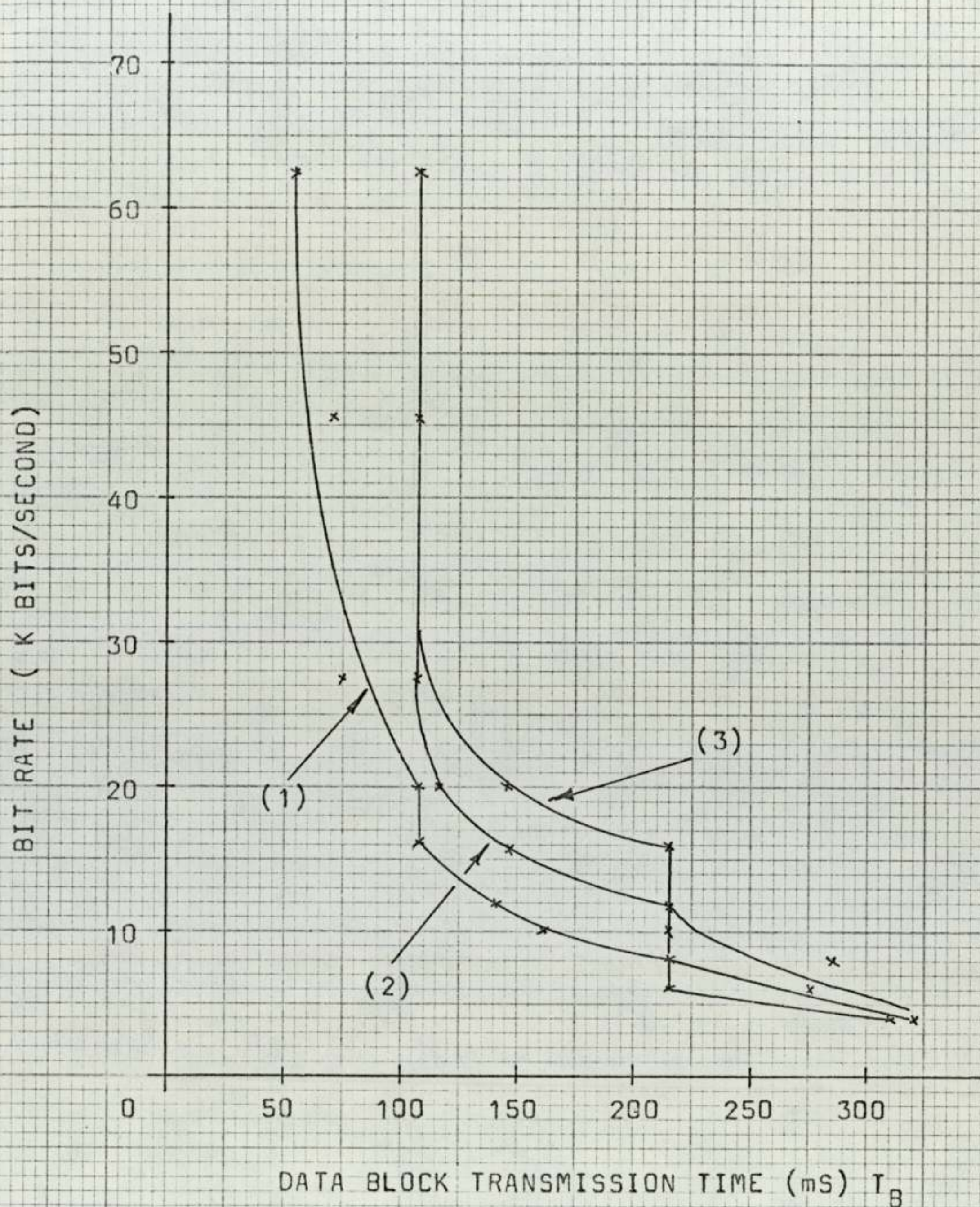


Fig. 8.3. GRAPH SHOWING EFFECT OF BIT RATE ON DATA BLOCK TRANSMISSION TIME FOR 1,2 & 3 PATIENT PROCESSORS (WITHOUT MONITORING FUNCTION OPERATING).

Fig. 8.4. GRAPH SWOWING EFFECT OF BIT RATE ON DATA BLOCK TRANSMISSION TIME FOR 1,2 & 3 PATIENT PROCESSORS (WITH MONITORING FUNCTION OPERATING).

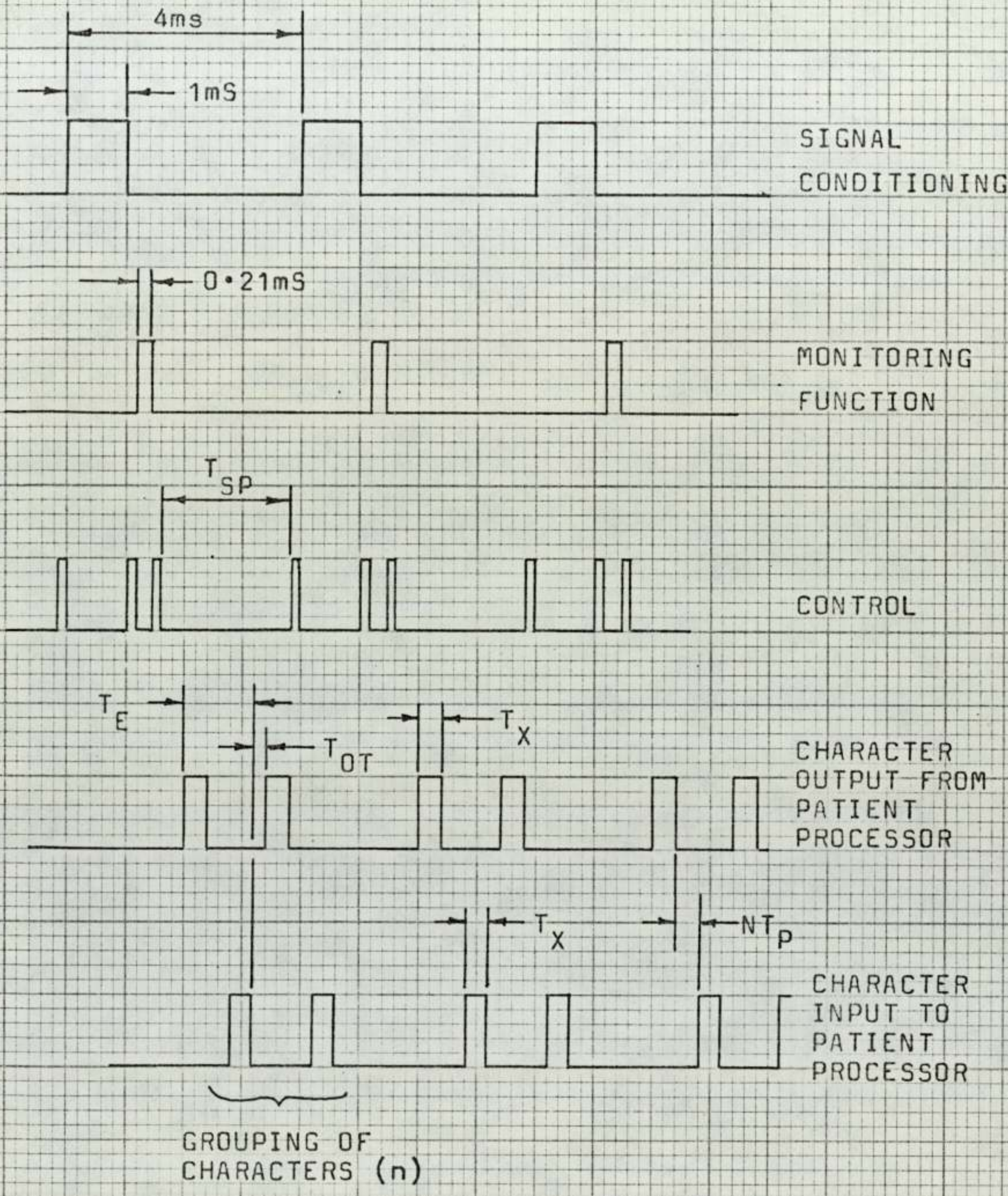


Fig. 8.5. INTER-PROCESSOR COMMUNICATION TIMING DIAGRAM.

causes the flow of characters between the processors to be grouped together. The discontinuities seen in the graphs (Fig. 8.3. and Fig. 8.4.) are due to this grouping effect, because although the bit rate may be reduced considerably, the number of characters sent in each available time interval (T_{SP}), remains constant.

As these time intervals are regular, and the number of characters in each interval remains constant over a range of bit rates, the result is a constant block transmission time, and thus the discontinuity in the graph. The three discontinuities seen in the graphs, correspond to the transmission of 3,2 and then 1 character in each time interval.

The intervals between the discontinuities are transition states where the number of characters in each interval varies.

In order to determine some analytical approach which would allow the transmission time for larger numbers of patient processors to be estimated; measurements were made of the time interval between the transmission of a character from the patient processor, and the reception of a reply character. This time interval was referred to as the transmission echo time (T_E) and the results obtained are shown in Fig. 8.6. As can be seen from these graphs, a linear relationship exists between the transmission echo time (T_E), the character transmission time (T_X) and the number of patient processors (N). This relationship

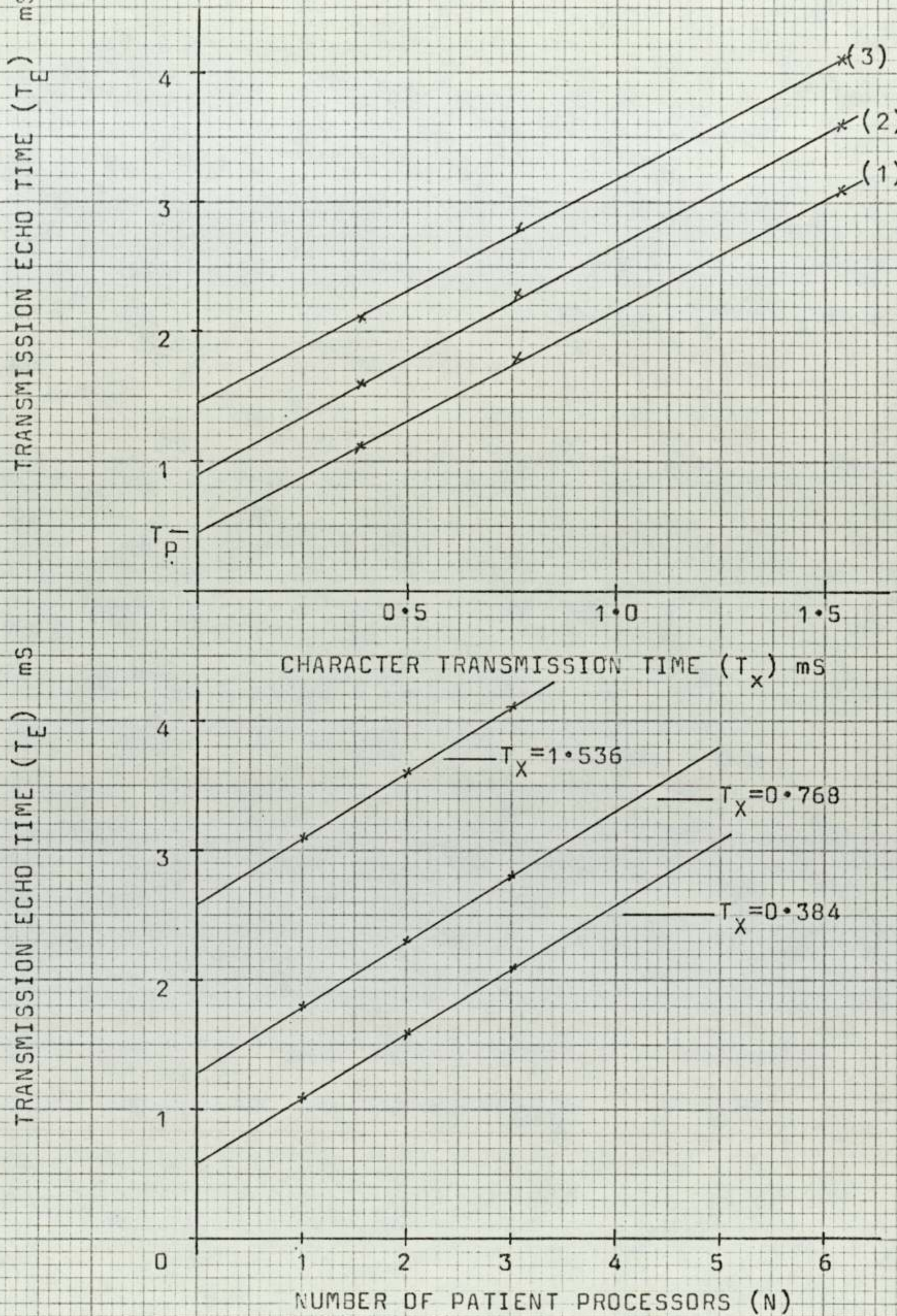


Fig. 8.6. EFFECT OF INCREASING T_X ON T_E (TOP GRAPH)

EFFECT OF INCREASING N ON T_E (BOTTOM GRAPH).

is therefore given by,

$$T_E = 2T_X + NT_P \qquad (8.1.)$$

The time T_P (Fig. 8.5.) is the constant processing time at the operator processor system, and was found to be equal to 0·45mS as shown in Fig. 8.6.

In order to estimate the block transmission time, the number of characters (n) that are sent during the surplus processing time (T_{SP}), at the patient processor system, must be known. The number of characters (n) may be estimated by,

$$n \simeq \frac{T_{SP}}{T_E + T_{OT}} \qquad (8.2.)$$

where T_{OT} is the response time of the patient processor output task (Fig. 8.5.), which was measured as being approximately 0·08mS.

Having obtained values for n, which correspond to the number of characters sent in a 4mS interval, it is then possible to estimate the block transmission time (T_B) from,

$$T_B \simeq \frac{4 \times 10^{-3} \times N_B}{n} \qquad (8.3.)$$

where N_B is the number of bytes sent in a block.

Using this approach it was possible to produce the graph as shown in Fig. 8.7., which is an approximation to the graph shown in Fig. 8.4. From this graph (Fig. 8.7.)

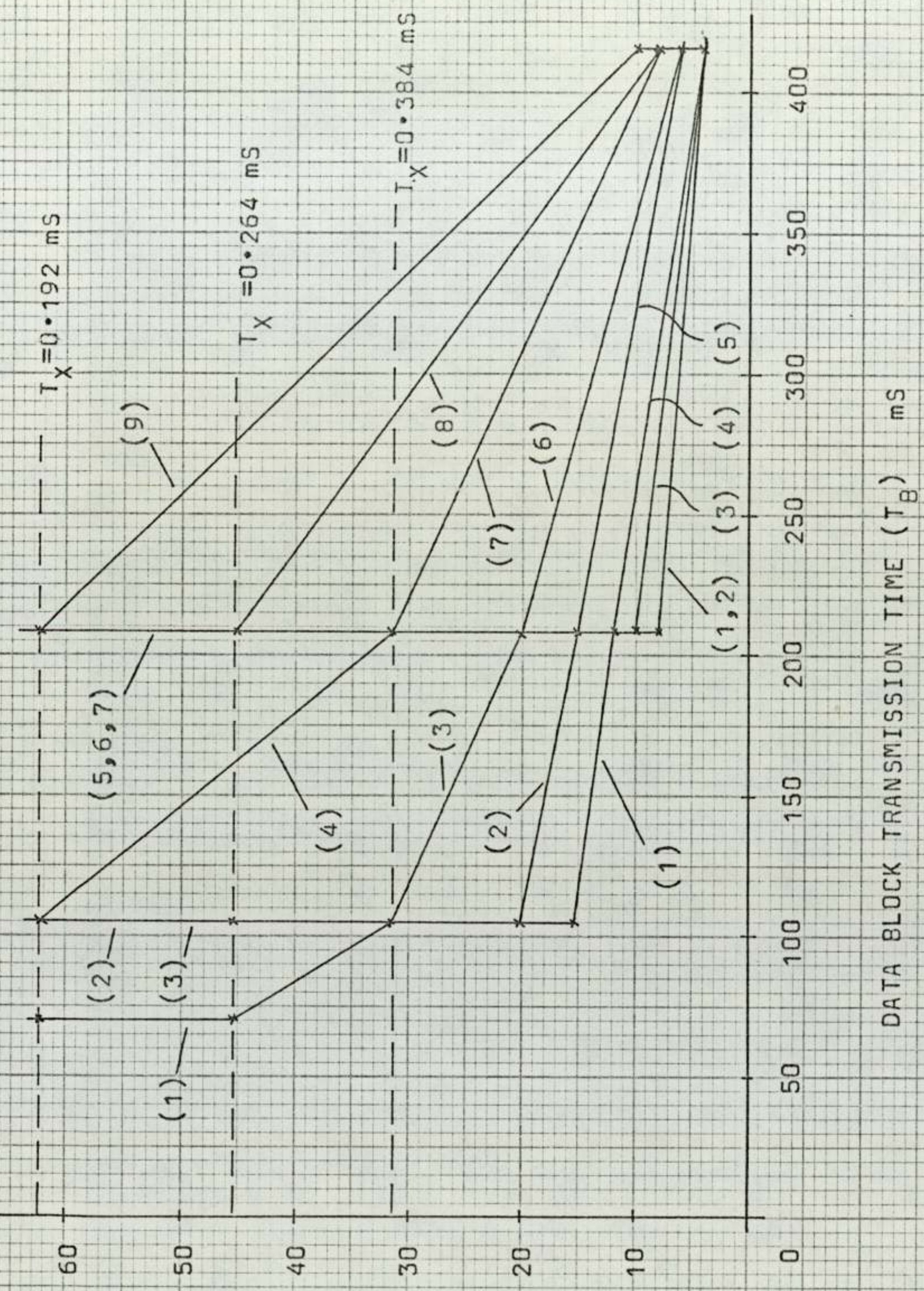


Fig. 8.7. GRAPH SHOWING CALCULATED EFFECT OF INCREASING THE NUMBER OF PATIENT PROCESSORS (N).

it was then possible to produce the curve shown in Fig. 8.8., which indicates the estimated maximum block transmission time for simultaneous communication with N patient processors. These results are for the worst case communication conditions, as they are calculated for simultaneous reception of all characters, as in the simulation system. In the normal system, however, if characters are received simultaneously, the operation of the operator processor operating system, causes the communications with each patient processor to become unsyncronised, and results in faster block transmission times.

It has been shown by the previous discussion that the real-time processing performed by each patient processor, is responsible for limiting the rate of information flow between processors. If the execution time of the signal conditioning function was increased, for example, the result would be a reduction in the surplus processing time (T_{SP}). This would result, as indicated by equations 8.2. and 8.3., in an increase in the block transmission times.

Within the present implementation of the inter-processor communications protocol, a time interval greater than 16mS between characters, indicates an error condition. With this arrangement, it allows the maximum block transmission time to be approximately 830mS, which would allow the operator system to communicate

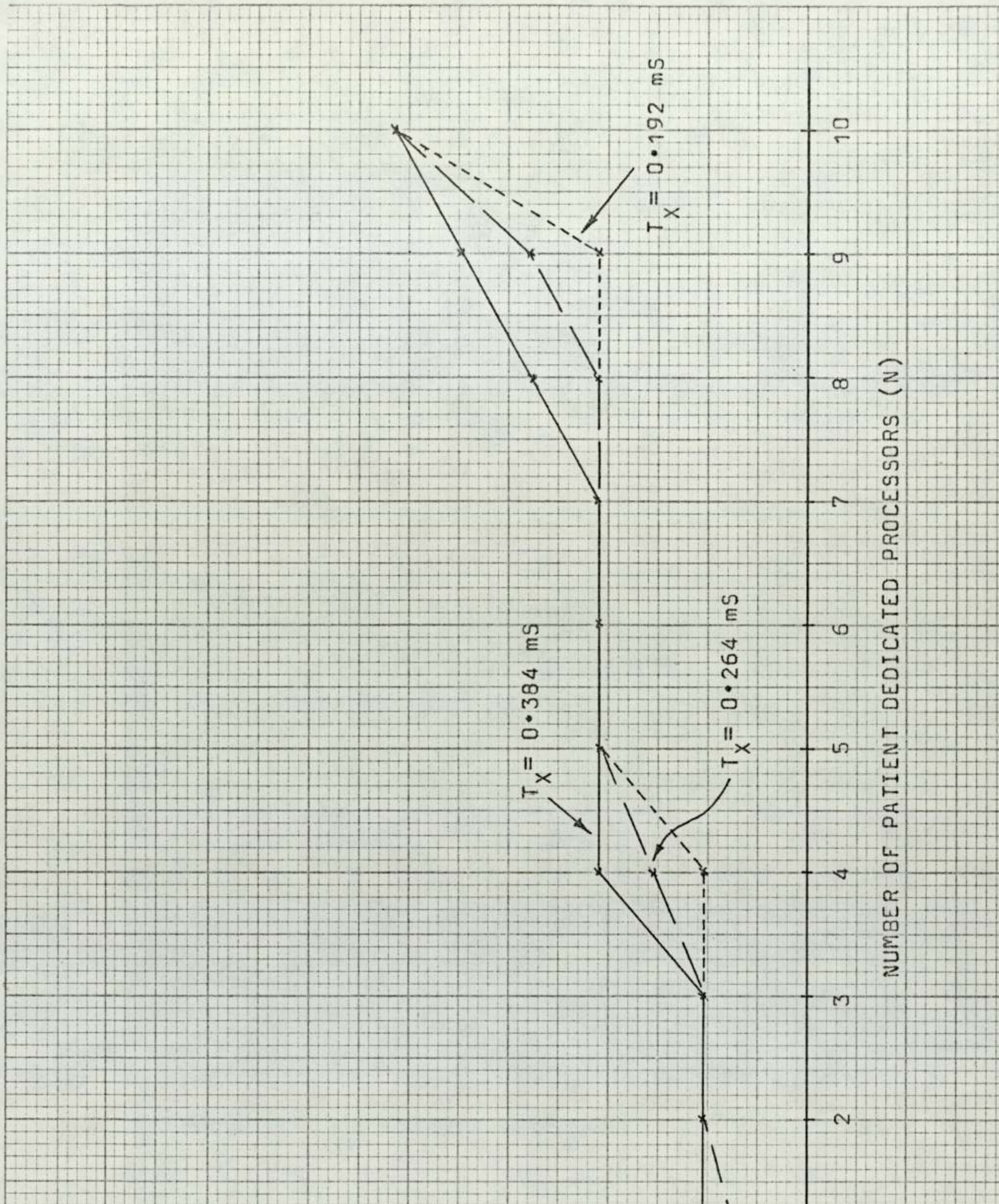


Fig. 8.8. MAXIMUM BLOCK TRANSMISSION TIME FOR SIMULTANIOUS COMMUNICATION WITH N PATIENT PROCESSORS.

simultaneously with approximately 30 patient processors without timing errors occurring. However, in a more practical system, it is likely that some other limit on the block transmission time would be specified. For example, it may be required that the block transmission time should not exceed the R-R time interval, when the pulse rate is 200 beats/minute (R-R interval = 300mS). With such a specification, it can be seen from Fig. 8.8., that the ideal number of patient processors in such a system would be eight.

Therefore, the number of patient processors in the final system will depend upon how much the existing real-time algorithms are expanded, and the maximum allowed block transmission time.

CHAPTER 9

CONCLUSIONS

9.1. General

The aim of this research project was to examine the possibility of using microprocessors, to produce an on-line real-time ECG arrhythmia monitoring system.

As a result of the research work performed, it has been demonstrated that although the execution times of current microprocessors are slower, compared to that of computers, their implementation in a distributed network enables a real-time ECG arrhythmia monitoring system to be produced. It has also been demonstrated that a suitable structure for a multi-patient distributed microprocessor system, is the allocation of a microprocessor to each patient, controlled by a centralised operator-dedicated microprocessor. The functions performed by these patient-dedicated processors are that of signal conditioning, monitoring and arrhythmia diagnosis.

This modular approach developed, results in a flexible system structure, which would allow mass produced systems to be tailored to each CCUs requirements.

The design of a system with this structure has been described, and the methods for realising each function have been examined in this report. The distributed structure of the monitoring system enabled the recommendations by Wartak et. al. regarding ECG sampling rate and Van Eyll et. al. regarding optimal parameters for QRS detection, to be implemented in this system. Other multi-patient computer monitoring systems, have used sampling rates less than that recommended by Wartak et. al., to enable multi-patient monitoring to be performed. Also the distributed microprocessor approach eliminates the need for an operating system capable of allocating processing time to each patient, as is required in multi-patient computer systems.

As a result of the problems encountered regarding the monitoring criteria to be used by the system, as discussed in Chapter 2, and also due to the lack of detailed diagnosed ECG recordings, no detailed evaluation of the systems diagnosis performance was undertaken. It is considered that further medical research is required to specify precisely diagnosis criteria for use by monitoring systems, and also that medical experts should be deeply involved in any system diagnosis performance evaluation.

As the monitoring and diagnosis functions are implemented in software, any new or improved monitoring criteria obtained from further work could be introduced easily into the system, as demonstrated by the implementation of the suggested alternative monitoring criteria described in Section 2.2.2.

It has also been shown that within the present performance of the system there is ample scope for expansion, before the processing time consumed becomes excessive. This system performance is a direct result of programming all software in assembly language. Although assembly language programming requires more programming effort, it enabled the execution times of the various algorithms to be made as short as possible. It is considered that had these programs been produced using a real-time high level language such as CORAL 66, the current opportunity for expansion would not have been so great.

It has been estimated that had these programs been produced in industry, the developmental cost of the software produced would have been £14,000. As this software is repeated and spread throughout the system, and if the patient-dedicated processor units were mass produced, the cost of software development per unit should not be excessive. The cost of a complete multi-microprocessor monitoring system, should therefore be far less than currently available computer monitoring systems.

Also, with the advent of microcomputers, such as the TMS 9940 (which is a single chip containing a CPU and currently limited ROM and RAM), there is the prospect of reducing the cost of the system still further.

9.2. Suggestions for Future Development

As it is apparent that there is the need for further medical research to be performed, to assess and develop new monitoring criteria, it is thought that a single microprocessor monitoring system could provide an inexpensive system for such studies.

Within the current patient processor system work could be performed to determine if QRS detection would be improved if the detection algorithm used only the low-pass filtered ECG signal, instead of the conditioned signal which may or may not have been filtered. Also the artefact indicator incorporated in the signal conditioning function, could be used to a greater extent, in order to inhibit certain monitoring operations, with a view to reducing the number of false positive diagnoses.

The introduction of a mass storage media into the system would enable trend data analysis to be performed, which could well be enhanced by some form of colour graphics display.

Other directions in which this project could be expanded, is by the introduction of another patient-dedicated processor, which could monitor the slower physiological signals, resulting in a total care monitoring system.

Alternatively, the system could be developed as a 12 lead monitoring system, enabling more sophisticated clinical diagnoses of individual patients to be obtained.

9.3. Final Remarks

There have been enquiries received from industry, relating to this research project, which have indicated that the system described in this report should have good commercial prospects. It is hoped that such interest will result in the benefits possible from a real-time arrhythmia monitoring system, being more widely available than at the present moment.

At this time when many people see the advent of microprocessors as producing social problems, such as unemployment due to automation, it is hoped that work such as this research project, indicates that such devices can also improve the quality of life.

APPENDIX A

THE TMS 9900 MICROPROCESSOR

HARDWARE FEATURES

The microprocessor used throughout the development of the multi-microprocessor arrhythmia monitoring system, is the TMS 9900. This microprocessor is a single-chip 16 bit central processing unit (CPU), produced using N-channel silicon-gate MOS technology. The CPU comes in a 64 pin chip, and is driven by a 3MHz four phase clock.

The processor employs a memory-to-memory form of architecture, whereby blocks of memory designated as workspace registers, replace the more common internal hardware registers. The basic microprocessor memory structure is shown in Fig. A.1, and as can be seen the first 32 words of memory are used for the 16 interrupt trap vectors. The next block of 32 words are then used for extended operation (XOP) instruction trap vectors. The last two memory words, $FFFC_{16}$ and $FFFE_{16}$, are used for the trap vectors of the LOAD signal. The remaining memory is then available for programmes, data and workspace registers. A total of 32,768 words of memory can be addressed by the processors 15 bit address bus, which is separate from the 16 bit data bus, thus simplifying the system design.

Within the processor there are three registers which are accessible by the user. These are the program counter (PC) which contains the address of the instruction following the current instruction being executed. The status register (ST) which contains the interrupt mask level and status information, relating to the instruction operation. The third and final register is the workspace pointer (WP) which contains the address of the first word in the current active workspace area. A workspace area consists of 16 consecutive memory words in the general memory area.

The workspace concept is particularly valuable during operations that require a context switch (i.e. a change from one program to another, or to a subroutine, for example when an interrupt occurs).

The processor can handle a total of 16 interrupt levels, which are serviced rapidly due to the memory-to-memory architecture.

Input and output data transfers to and from the processor are performed by a direct command-driven I/O Interface designated as the communications-register unit (CRU). The CRU provides up to 4096 directly addressable input and output bits. Both input and output bits can be addressed individually or in fields of 1 to 16 bits.

Software Features

The TMS 9900 microprocessor instruction set provides the same capabilities as those offered by full minicomputers. The instruction set provides 69 different instructions, which includes unsigned multiply and divide instructions.

The multiply instruction allows two unsigned 16 bit numbers to be multiplied together to produce a 32 bit answer. The divide instruction allows an unsigned 32 bit number to be divided by a 16 bit number, the answer being given as a 16 bit quotient and a 16 bit remainder.

Other instructions which were to be of particular use during the development of the arrhythmia monitoring system were the "compare ones corresponding" (COC) and "compare zeros corresponding" (CZC) instruction, which can be used to implement decision table programs.

The complete list of instruction mnemonics is given in Table A.1., along with the explanation of how to calculate individual instruction execution times. With a clock frequency of 3MHz, the average instruction execution time is approximately 10μS.

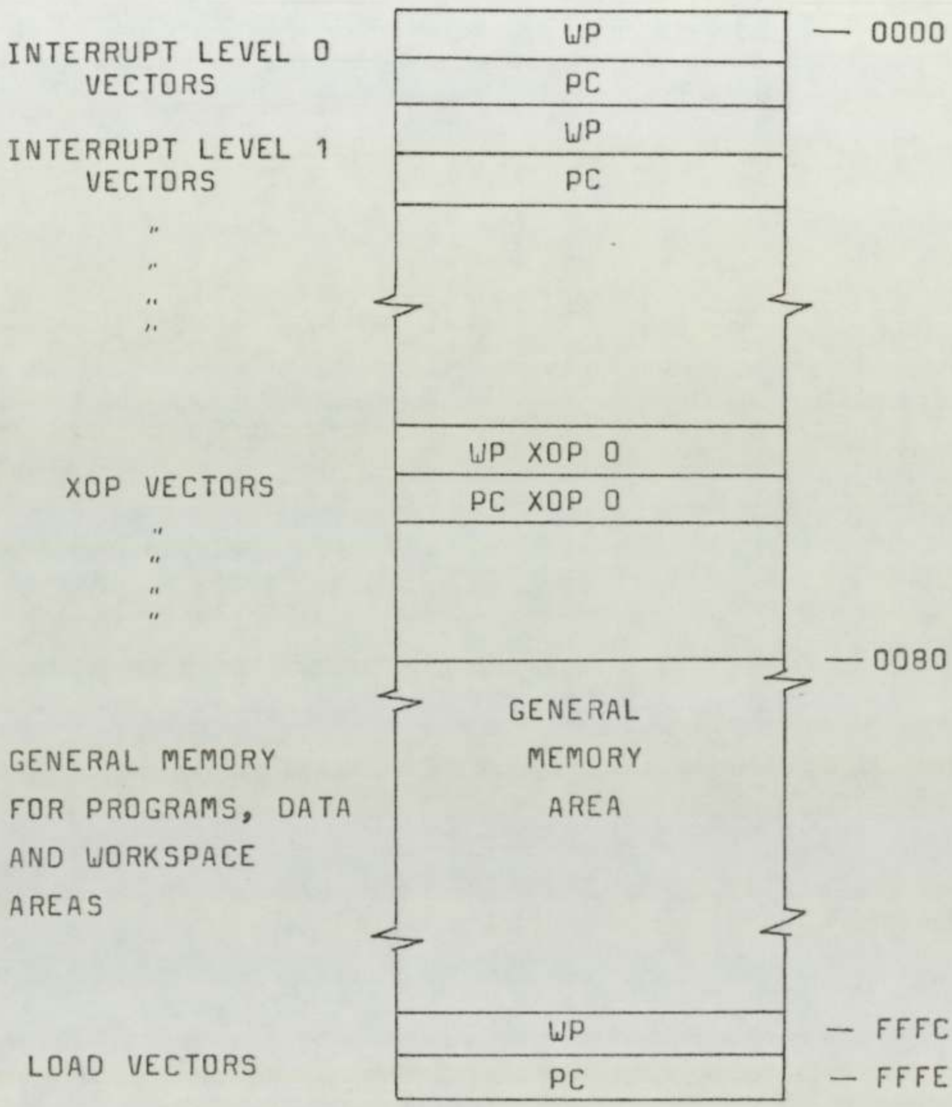


Fig. A.1. BASIC MEMORY STRUCTURE.

TMS 9900 INSTRUCTION EXECUTION TIMES

Instruction execution times for the TMS 9900 are a function of:

1) Clock cycle time, $t_{c(\phi)}$

2) Addressing mode used where operands have multiple addressing mode capability

3) Number of wait states required per memory access.

Table 3 lists the number of clock cycles and memory accesses required to execute each TMS 9900 instruction. For instructions with multiple addressing modes for either or both operands, the table lists the number of clock cycles and memory accesses with all operands addressed in the workspace-register mode. To determine the additional number of clock cycles and memory accesses required for modified addressing, add the appropriate values from the referenced tables. The total instruction-execution time for an instruction is:

$$T = t_{c(\phi)} \quad (C + W \cdot M)$$

where:

T = total instruction execution time;
$t_{c(\phi)}$ = clock cycle time;
C = number of clock cycles for instruction execution plus address modification;
W = number of required wait states per memory access for instruction execution plus address modification;
M = number of memory accesses.

TABLE 3
INSTRUCTION EXECUTION TIMES

INSTRUCTION	CLOCK CYCLES C	MEMORY ACCESS M	ADDRESS MODIFICATION†	
			SOURCE	DEST
A	14	4	A	A
AB	14	4	B	B
ABS (MSB = 0)	12	2	A	–
(MSB = 1)	14	3	A	–
AI	14	4	–	–
ANDI	14	4	–	–
B	8	2	A	–
BL	12	3	A	–
BLWP	26	6	A	–
C	14	3	A	A
CB	14	3	B	B
CI	14	3	–	–
CKOF	12	1	–	–
CKON	12	1	–	–
CLR	10	3	A	–
COC	14	3	A	–
CZC	14	3	A	–
DEC	10	3	A	–
DECT	10	3	A	–
DIV (ST4 is set)	16	3	A	–
DIV (ST4 is reset)*	92-124	6	A	–
IDLE	12	1	–	–
INC	10	3	A	–
INCT	10	3	A	–
INV	10	3	A	–
Jump (PC is changed)	10	1	–	–
(PC is not changed)	8	1	–	–
LDCR (C = 0)	52	3	A	–
($1 \leq C \leq 8$)	20+2C	3	B	–
($9 \leq C \leq 15$)	20+2C	3	A	–
LI	12	3	–	–
LIMI	16	3	–	–
LREX	12	1	–	–
RESET function	26	5	–	–
LOAD function	22	5	–	–
Interrupt context switch	22	5	–	–
LWPI	10	2	–	–
MOV	14	4	A	A
MOVB	14	4	B	B
MPY	52	5	A	–
NEG	12	3	A	–
ORI	14	4	–	–
RSET	12	1	–	–
RTWP	14	4	–	–
S	14	4	A	A
SB	14	4	B	B
SBO	12	2	–	–
SBZ	12	2	–	–
SETO	10	3	A	–
Shift (C ≠ 0)	12+2C	3	–	–
(C = 0, Bits 12–15 of WR0 = 0)	52	4	–	–
(C = 0, Bits 12–15 of WR0 = N ≠ 0)	20+2N	4	–	–
SOC	14	4	A	A
SOCB	14	4	B	B
STCR (C = 0)	60	4	A	–
($1 \leq C \leq 7$)	42	4	B	–
(C = 8)	44	4	B	–
($9 \leq C \leq 15$)	58	4	A	–
STST	8	2	–	–
STWP	8	2	–	–
SWPB	10	3	A	–
SZC	14	4	A	A
SZCB	14	4	B	B
TB	12	2	–	–
X**	8	2	A	–
XOP	36	8	A	–
XOR	14	4	A	–
Undefined op codes: 0000-01FF, 0320-033F, 0C00-0FFF, 0780-07FF	6	1	–	–

*Execution time is dependent upon the partial quotient after each clock cycle during execution.
**Execution time is added to the execution time of the instruction located at the source address.
†The letters A and B refer to the respective tables that follow.

ADDRESS MODIFICATION – TABLE A

ADDRESSING MODE	CLOCK CYCLES C	MEMORY ACCESSES M
WR (T_S or T_D = 00)	0	0
WR indirect (T_S or T_D = 01)	4	1
WR indirect auto-increment (T_S or T_D = 11)	8	2
Symbolic (T_S or T_D = 10, S or D = 0)	8	1
Indexed (T_S or T_D = 10, S or D ≠ 0)	8	2

ADDRESS MODIFICATION – TABLE B

ADDRESSING MODE	CLOCK CYCLES C	MEMORY ACCESSES M
WR (T_S or T_D = 00)	0	0
WR indirect (T_S or T_D = 01)	4	1
WR indirect auto-increment (T_S or T_D = 11)	6	2
Symbolic (T_S or T_D = 10, S or D = 0)	8	1
Indexed (T_S or T_D = 10, S or D ≠ 0)	8	2

As an example, the instruction MOVB is used in a system with $t_{c(\phi)} = 0.333\ \mu s$ and no wait states are required to access memory. Both operands are addressed in the workspace register mode.

$$T = t_{c(\phi)}\ (C + W \cdot M) = 0.333\ (14 + 0 \cdot 4)\ \mu s = 4.662\ \mu s.$$

If two wait states per memory access were required, the execution time is:

$$T = 0.333\ (14 + 2 \cdot 4)\ \mu s = 7.326\ \mu s.$$

If the source operand was addressed in the symbolic mode and two wait states were required:

$$T = t_{c(\phi)}\ (C + W \cdot M)$$
$$C = 14 + 8 = 22$$
$$M = 4 + 1 = 5$$
$$T = 0.333\ (22 + 2 \cdot 5)\ \mu s = 10.656\ \mu s.$$

4. TMS 9900 ELECTRICAL AND MECHANICAL SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1)	−0.3 to 20 V
Supply voltage, V_{DD} (see Note 1)	−0.3 to 20 V
Supply voltage, V_{SS} (see Note 1)	−0.3 to 20 V
All input voltages (see Note 1)	−0.3 to 20 V
Output voltage (with respect to V_{SS})	−2 V to 7 V
Continuous power dissipation	1.2 W
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−55°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1. Under absolute maximum ratings voltage values are with respect to the most negative supply, V_{BB} (substrate), unless otherwise noted. Throughout the remainder of this section, voltage values are with respect to V_{SS}.

TABLE A.1. INSTRUCTIONS

APPENDIX B

PATIENT PROCESSOR PROGRAM LISTING

All the programs and data modules required to produce the patient processor software system, are supplied in this appendix, in the order in which they appear in the link editor listing shown below.

PHASE 0, PATIENT ORIGIN = 0000 LENGTH = 1830

MODULE	NO	ORIGIN	LENGTH
PMPTVP	1	0000	00A4
PPINIP	2	00A4	00A0
PPTHAP	3	0144	00D6
PPCIHP	4	021A	00BA
PPUIHP	5	02D4	014C
IPDHTP	6	0420	01B6
MONTDP	7	05D6	018A
MONS1P	8	0760	0284
MONS2P	9	09E4	0142
DIAT1P	10	0B26	00DC
IPPTOP	11	0C02	00B0
OPPTOP	12	0CB2	0082
MCPTOP	13	0D34	0130
P0001D	14	0E64	0014
P0002D	15	0E78	000C
P0003D	16	0E84	002E
P0004D	17	0EB2	0032
P0005D	18	0EE4	0034
P0009D	19	0F18	0018
IPDHTD	20	0F30	00C1
MONTDD	21	0FF2	0020
G0016D	22	1012	007A
G0017D	23	108C	0024
G0003D	24	10B0	02EE
G0004D	25	139E	0362
G0005D	26	1700	003E
G0006D	27	173E	0024
G0018D>	28	1762	00CE

PATIENT PROCESSOR PROGRAM LABELS

D E F I N I T I O N S

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
A1	0E68	14	A1DC	0E74	14	A2	0E6A	14	A2DC	0E76	14
ACTNO	101C	22	ARST6	0EEC	18	ARTIF	109E	23	ASYT7	0EEE	18
AVRR	16FC	25	AVRRB	0E96	16	BBB3	0EE6	18	BEATS	0EA4	16
BPM100	0E8A	16	BPM110	0E88	16	BPM120	0E86	16	BPM140	0E84	16
BPM20	0E94	16	BPM40	0E92	16	BPM60	0E90	16	BPM80	0E8E	16
BPM90	0E8C	16	BRAD4	0EE8	18	BUFEND	0FEE	20	BUFFER	0FB4	20
BUFLEN	0E78	15	C	0E70	14	CLRWT	10AE	23	CM	0F23	19
CMS	10A0	23	CDC1	0EB2	17	CDC2	0EB6	17	CDC3	0EBC	17
CDC4	0EC0	17	CDC5	0EC8	17	CDC6	0ECA	17	COUNT	0E9E	16
CPCRET	104C	22	CSTIMC	1060	22	CSTRET	104E	22	CWPRET	104A	22
CZC1	0EB4	17	CZC2	0EB8	17	CZC3	0EBA	17	CZC4	0EBE	17
CZC5	0EC2	17	CZC6	0EC4	17	CZC7	0EC6	17	DARST0	0F0A	18
DARST1	0F08	18	DASYT0	0F0E	18	DASYT1	0F0C	18	DBBB0	0EFE	18
DBBB1	0EFC	18	DBP1	168E	25	DBRAD0	0F02	18	DBRAD1	0F00	18
DBUFL	0E7A	15	DESCB0	0F12	18	DESCB1	0F10	18	DF1	175E	27
DF2	1760	27	DFGO	10A2	23	DIAG	1054	22	DIAGM	0F1A	19
DIAINT	108E	23	DIAT11	0B26	10	DIAT1W	173E	27	DIDID0	0F06	18
DIDID1	0F04	18	DISF	1098	23	DLS1	0F14	18	DMINC	1738	26
DMINO	1736	26	DPCRET	103A	22	DRTWP	101E	22	DSTIMC	105E	22
DSTRET	103C	22	DSVTA0	0EF6	18	DSVTA1	0EF4	18	DSVTB0	0EFA	18
DSVTB1	0EF8	18	DWPRET	1038	22	EIGHT	0EE2	17	EM	0F22	19
ESCBT8	0EF0	18	FINISH	0732	7	FIVE	0EAA	16	HOURS	106A	22
HPOP	0FB0	20	HUND	0EA6	16	IBP1	139E	25	IBUF	10B0	24
IDH	1050	22	IDIDV5	0EEA	18	IPCRET	1040	22	IPDHT1	0420	6
IPDHTW	0F30	20	IPFILE	1782	28	IPINT	1090	23	IPPT01	0C02	11
IPPT02	0C8A	11	IPPT03	0CA0	11	IPPTOW	1762	28	IPRTWP	1024	22
IPTASK	1056	22	ISTRET	1042	22	IWPRET	103E	22	K1LP	0E6C	14
K2LP	0E6E	14	KDC	0E72	14	KHP	0E66	14	LEVEL	0E64	14
LPOP	0FB2	20	LS23	0EF2	18	MAXP	172A	26	MAXV	1726	26
MCP	105A	22	MCPINT	1094	23	MCPIP	109C	23	MCPT01	0D34	13
MCPTOW	180A	28	MCRTWP	1030	22	MF1	1700	26	MF2	1702	26
MFAIL	0F1F	19	MINL	0E7C	15	MINP	1728	26	MINUTE	1068	22
MINV	1724	26	MM1	0F25	19	MM2	0F26	19	MM3	0F27	19
MM4	0F28	19	MM5	0F29	19	MM6	0F2A	19	MONI	1052	22
MONINT	108C	23	MONS11	0760	8	MONS1W	1704	26	MONS21	09E4	9
MONSF	109A	23	MONTD1	05D6	7	MONTDW	0FF2	21	MONWR3	0FF8	21
MOTEXT	182C	28	MPCRET	1034	22	MRTWP	1018	22	MSEC	1064	22
MSTIMC	105C	22	MSTRET	1036	22	MSUDVE	0F1E	19	MWPRET	1032	22
NOORS	0B0E	9	NP	16F4	25	NORS6B	0EDE	17	OFSET0	173C	25
OPAC	1062	22	OPCRET	1046	22	OPDATA	182A	28	OPFILE	17D6	28
OPINT	1092	23	OPPT01	0CB2	12	OPPTOW	17B6	28	OPRTWP	102A	22
OPTASK	1058	22	OSTRET	1048	22	OWPRET	1044	22	P2POS	173A	26
PER82	0EAC	16	POINT	16FE	25	PPCIH1	021A	4	PPCIH2	025A	4
PPINI1	00A4	2	PPTHA1	0144	3	PPTHAW	1012	22	PPUIH1	02D4	5
PPUIH2	03B6	5	PPUIH3	03CC	5	PPUIH4	03E8	5	PPUIH5	0402	5
PPUIHW	106C	22	PR	0F20	19	PRGT60	0F16	18	PRM	0F18	19
PS	0F24	19	PSD	0F1C	19	PUART	040C	5	QRSNT	0EB1	16
QRSPT	0EB0	16	ORSWB	0EA2	16	QRSWID	0EA0	16	RAM	0F30	20
RAM2	0F32	20	RAMEND	1830	28	RRC	16F6	25	RRC32	0E7E	15
RRLB	0E98	16	RRMAXT	0EE0	17	RRSAVE	16F8	25	RRSB	0E9A	16
SDELAY	10A4	23	SEC	1066	22	SEQ1	0ECC	17	SEQ2	0ECE	17
SEQ3	0ED0	17	SEQ4	0ED2	17	SEQ5	0ED4	17	SEQ6	0ED6	17
SEQ7	0ED8	17	SEQ8	0EDA	17	SEQ9	0EDC	17	SEQB	0E9C	16
SETEND	10B0	23	SETNEG	10A2	23	SETONE	109C	23	SFP1	10A6	23
SIX	0E82	15	SIXTY	0F2E	19	SM	0F21	19	SR250	0F2C	19
SU	10A8	23	SU2	10AA	23	SU3	10AC	23	SVT2	0EE4	18
T1	172C	26	T2	172E	26	T3	1730	26	TDP1	16F2	25
TEN	0E80	15	TOG	1096	23	TWELVE	0EA8	16	VALUE	0FF0	20
W1	1732	26	W2	1734	26	WADC	0F90	20	WAHP	0F50	20
WALP	0F70	20	WID82S	182E	28	WIDE82	0EAE	16	WQRS	16FA	25

PATIENT MONITORING PROCESSOR TRANSFER VECTOR PROGRAM MODULE

1. DESCRIPTION

This program module defines the interrupt and XOP transfer vectors used by the Patient Monitoring Processor. Interrupt level 2 and XOP level 15 have been initiated to their appropriate values for the 990/4 Monitor software.

2. IDENTIFICATION

SUBROUTINE NAME	PMPTV
PROGRAM MODULE	PMPTVP
GENERAL DATA MODULE	G0016D

3. SIZE

PROGRAM MODULE	164	Bytes

4. CALLS FROM SUBROUTINE

INTERRUPT LEVEL	0	PPINI1
INTERRUPT LEVEL	3	PPCIH1
INTERRUPT LEVEL	4	PPUIH1

5. AUTHOR

B.T.V. WARTON.

```
SCR2.PROG.S.PMPTVP  16:46:30 TUESDAY, OCT 17, 1978.

       TITL 'PATIENT MONITORING PROCESSOR TRANSFER VECTOR PROGRAM
* PROGRAM MODULE IDENTIFIER    (PROG)
       IDT  'PMPTVP'
* CALLED PROGRAM MODULES
       REF  PPINIP,PPCIHP,PPUIHP
       REF  PPINI1,PPCIH1,PPUIH1
* LINKED DATA MODULES
       REF  G0016D,PPTHAW,PPUIHW
       RORG
       PSEG
* PROGRAM
       DATA PPTHAW,PPINI1              INTERRUPT 0 TRANSFER VECTORS
       DATA 0,0                            1
       DATA 0,>5D26                        2 REQUIRED BY MONITOR
       DATA PPTHAW,PPCIH1                  3
       DATA PPUIHW,PPUIH1                  4
       DATA 0,0,0,0,0,0,0,0                5 TO 8
       DATA 0,0,0,0,0,0,0,0,0,0            9 TO 13
       DATA 0,0,0,0                        14 & 15
       DATA 0,0,0,0,0,0,0,0,0,0        XOP TRANSFER VECTORS
       DATA 0,0,0,0,0,0,0,0,0,0
       DATA 0,0,0,0,0,0,0,0,0,0
       DATA >77D2,>604E                XOP 15 REQUIRED BY MONITOR
       BSS  32                         WORKSPACE REQUIRED BY MONITOR
ENTRY1 BLWP @0                         EMULATE INT 0
       END  ENTRY1
```

PATIENT PROCESSOR INITIALISATION PROGRAM

1. DESCRIPTION

This program runs at system start up time to initialise software and hardware as follows. First all RAM used is cleared then all software flags that should be set to +1 or -1 are initialised. All workspace registers requiring CRU base addresses and other parameters are then initialised, and finally the ADC/DAC and UART are reset and initialised.

On completion of this program control is passed to the Patient Processor Task Handler program.

2. IDENTIFICATION

PROGRAM MODULE	PPINIP
SPECIFIC DATA MODULE	IPDHTD
SPECIFIC DATA MODULE	MUNTDD
GENERAL DATA MODULE	G0001D
GENERAL DATA MODULE	G0002D
GENERAL DATA MODULE	G0003D
GENERAL DATA MODULE	G0008D

3. SIZE

PROGRAM MODULE 160 Bytes

4. CALLS TO SUBROUTINE

```
BLWP   @PPINI1
```

5. CALLS FROM SUBROUTINE

PROGRAM MODULE PPTHAP

```
BLWP   @PPTHA1
```

PROGRAM MODULE PPUIHP

```
BL     @PUART
```

6. AUTHOR

B.T.V. WARTON.

```
      TITL 'PATIENT PROCESSOR INITIALISATION PROGRAMME'
* PROGRAMME MODULE IDENTIFIER (TASK)
      IDT  'PPINIP'
* TRANSFER VECTORS OR ENTRIES
      DEF  PPINI1
* CALLED PROGRAM MODULES
      REF  PPTHAP,PPUIHP
      REF  PPTHA1,PUART
* LINKED DATA MODULES
      REF  IPDHTD,G0001D,G0002D,G0003D,G0008D,MONTDD
      REF  WAHP,WADC,INTHA1,BUFFER,WALP,IPDHTW,MONTDW
      REF  PPTHAW,PPUIHW,IPPTOW,OPPTOW
      REF  DFGO,IDH,MONI,DIAG,OCPF,IOTF
      REF  MONINT,DIAINT,OCPINT,IOTINT,IPINT
      REF  RAM,RAM2,RAMEND,SETONE,SETNEG,SETEND
      RORG
      PSEG
***********        P R O G R A M M E              **********
PPINI1 LWPI RAM
       LI   R0,RAM2
CLEAR  CLR  *R0+                  CLEAR FROM RAM TO RAMEND
       CI   R0,RAMEND
       JNE  CLEAR
       LI   R0,SETONE
SET    INC  *R0+                  INITIALIS  FLAGS TO +1
       CI   R0,SETNEG
       JNE  SET
SET1   SETO *R0+                  INITIALIS  FLAGS TO -1
       CI   R0,SETEND
       JNE  SET1
       CLR  @RAM
       LWPI WAHP
       LI   R0,BUFFER
       LI   R12,>100              9901 CRU BASE
       LI   R15,>0018             ENABLE INT 3&4
       LDCR R15,0
       LI   R12,>1100             CRU BASE
       LI   R15,BUFFER
       AI   R15,20                INITIALIS  TO BUFFER +20
       LWPI WALP
       LI   R0,BUFFER
       AI   R0,22                 INITIALIS  TO BUFFER +22
       LWPI WADC
       LI   R0,BUFFER             INITIALIS  TO BUFFER
       LI   R12,>1100             ADC CRU BASE
       LWPI IPDHTW
       LI   R12,>1100             CRU BASE ADC
       SBZ  8                     ADC RESET
       SBO  8
       CLR  R1                    BUF COUNT
       LWPI MONTDW
       CLR  R3
       LWPI PPTHAW
       LI   R12,>1100             ADC CRU BASE
       LWPI PPUIHW
       LI   R12,>1A80             UART CRU BASE
       BL   @PUART                GO PROGRAM UART
       LWPI IPPTOW
       LI   R12,>1A80             UART CRU BASE
       LWPI OPPTOW

       LI   R12,>1A80             UART CRU BASE
       LIMI 7                     INTERRUPT MASK
UP     BLWP @PPTHA1               GO TO TASK HANDLER
       JMP  UP
       END
```

PATIENT PROCESSOR TASK HANDLER

1. DESCRIPTION

This program is an operating system program conerned with scheduling system tasks in order of priority. Before this program passes control to the highest active task, it determines whether the task is in the suspended state. If the task is found to be suspended, control is passed to it at the point it was interrupted, otherwise control is passed to it at its starting point.

As an aid for observing the operation of the system, various CRU lines are set when a particular task is active, and reset on completion of that task.

This operating system program runs under the privileged interrupt mask of level zero, so that it cannot be interrupted.

2. IDENTIFICATION

SUBROUTINE NAME	PPTHA
PROGRAM MODULE	PPTHAP
GENERAL DATA MODULE	G0016D
GENERAL DATA MODULE	G0017D

3. SIZE

PROGRAM MODULE 214 Bytes.

4. CALLS TO SUBROUTINE

BLWP @PPTHA1

5. CALLS FROM SUBROUTINE

PROGRAM MODULE IPDHTP

BLWP @IPDHT1

PROGRAM MODULE MONTOP

BLWP @MONTD1

PROGRAM MODULE DIAT1P

BLWP @DIAT11

PROGRAM MODULE IPPTOP

BLWP @IPPTO1

PROGRAM MODULE OPPTOP

BLWP @OPPTOP

PROGRAM MODULE MCPTOP

BLWP @MCPTO1

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data to this program is the task active and interrupt flags listed below :-

IDH, MONI, DIAG, IPTASK, OPTASK, MCP,

MONINT, DIAINT, IPINT, OPINT, MCPINT.

7. EXTERNAL DATA TRANSFERS

PERIPHERAL CRU INTERFACE (UNUSED BITS IN ADC INTERFACE)

7.1. OUTPUT (TEST FACILITY)

CRU BASE >1000

CRU LINE	>B	MONITOR TASK ACTIVE BIT
CRU LINE	>C	DIAGNOSIS TASK ACTIVE BIT
CRU LINE	>D	INPUT TASK ACTIVE BIT
CRU LINE	>E	OUTPUT TASK ACTIVE BIT

8. TIMING

The maximum execution time of this program will not exceed 0.2µS

9. AUTHOR B.T.V. WARTON.

SCR2.TASK.S.PPTHAP   14:22:27 TUESDAY, OCT 17, 1978.

```
       TITL 'PATIENT PROCESSOR TASK HANDLER'
* PROGRAMME MODULE IDENT (TASK)
       IDT  'PPTHAP'
* TRANSFER VECTORS OR ENTRIES
       DEF  PPTHA1
* CALLED PROGRAM MODULES
       REF  IPDHTP,MONTDP,DIAT1P,OCPTOP,OPPTOP
       REF  IPDHT1,MONTD1,DIAT11,IPPT01,MCPT01,OPPT01
* LINKED DATA MODULES
       REF  G0016D,G0017D
       REF  PPTHAW,IDH,MONI,IPTASK,MCP,MCPINT,MCRTWP,OPINT
       REF  DIAG,MONINT,MRTWP,DIAINT,DRTWP,OPRTWP,OPTF
       REF  IPINT,IPRTWP,OPTASK
       RORG
       PSEG
*********     PROGRAMME     *****************
PPTHA1 DATA PPTHAW,STARTS
STARTS LIMI 2                  PRIVILEGE INT LEVEL
       LWPI PPTHAW
       MOV  @IDH,R0            TEST IDH FLAG
       JNE  BIDH               JUMP IF ACTIVE
       MOV  @MONI,R0           TEST MONI FLAG
       JNE  BMONI              JUMP IF ACTIVE
       MOV  @DIAG,R0
       JNE  @BDIAG
       MOV  @IPTASK,R0         TEST I/O FLAG
       JNE  BIPT               JUMP IF ACTIVE
       MOV  @OPTASK,R0         TEST OCP FLAG
       JNE  BOPT               JUMP IF ACTIVE
       MOV  @MCP,R0
       JNE  BMCP
       CLR  R5                 CLEAR TASK ACTIVE FLAG
       LIMI 7                  UNMASK ALL INTS
TOIDLE IDLE
       JMP  TOIDLE
BIDH   EQU  $
       LI   R5,1               SET TASK ACTIVE TO 1
       BLWP @IPDHT1            GO TO IDH TASK
       JMP  STARTS
BMONI  EQU  $
       LI   R5,2               SET TO TASK 2
       ABS  @MONINT            WAS MONI TASK SUSPENDED
       JGT  BM
       LWPI MRTWP              GET MONI TASK WP FOR RET
       RTWP
BM     LIMI 7                  UNMASK ALL INTS
       SBO  >B                 SET BIT TO INDICATE TASK ACTIVE
       BLWP @MONTD1            GO TO MONITOR TASK
       SBZ  >B                 RESET MONI TASK ACTIVE BIT
       JMP  STARTS
BDIAG  EQU  $
       LI   R5,3               SET TO TASK 3
       ABS  @DIAINT            WAS DIAG TASK SUSPENDED
       JGT  BD
       LWPI DRTWP              GET DIAG TASK WP FOR RET
       RTWP
BD     LIMI 7                  UNMASK ALL INTS
       SBO  >C                 SET DIAG TASK ACTIVE INDCATOR BIT
       BLWP @DIAT11            GO TO DIAGNOSIS TASK
       SBZ  >C                 RESET DIAG ACTIVE BIT
```

```
       JMP   STARTS
BIPT   EQU   $
       LI    R5,4          SET TO TASK 4
       ABS   @IPINT        WAS I/P TASK SUSPENDED
       JGT   BO
       LWPI  IPRTWP        GET I/P TASK WP FOR RET
       RTWP
BO     LIMI  7             UNMASK ALL INTS
       SBO   >D            SET I/P TASK ACTIVE BIT
       BLWP  @IPPT01       GO TO I/P TASK
       SBZ   >D            RESET I/P TASK ACTIVE BIT
       JMP   STARTS
BOPT   EQU   $
       LI    R5,5          SET TO TASK 5
       ABS   @OPINT        WAS O/P TASK SUSPENDED
       JGT   BI
       LWPI  OPRTWP        GET O/P TASK WP FOR RET
       RTWP
BI     LIMI  7             UNMASK ALL INTS
       SBO   >E            SET O/P TASK ACTIVE BIT
       BLWP  @OPPT01       GO TO O/P TASK
       SBZ   >E            RESET O/P TASK ACTIVE BIT
       JMP   STARTS
BMCP   EQU   $
       LI    R5,6          SET TO TASK 6
       ABS   @MCPINT       WAS MCP TASK SUSPENDED
       JGT   BMCP1
       LWPI  MCRTWP        GET MCP TASK WP FOR RET
       RTWP
BMCP1  LIMI  7             UNMASK ALL INTS
       BLWP  @MCPT01       GO TO MCP TASK
       JMP   STARTS
       END
```

PATIENT PROCESSOR ADC INTERRUPT HANDLER

1. DESCRIPTION

This program is an operating system program concerned with servicing the ADC interrupt. When the interrupt occurs this program determines which task, if any, was interrupted and will save the returns to that task and set it suspended. The signal conditioning task (IPDHTP) is then set active before this program branches to the task handling program (PPTHAP). Also incorporated in this program is a section of code concerned with calculating the time since system start up, in hours, minutes and seconds, from the 4mS interval ADC interrupt.

2. IDENTIFICATION

SUBROUTINE NAME	PPCIH
PROGRAM MODULE	PPCIHP
GENERAL DATA MODULES	G0016D
GENERAL DATA MODULES	G0017D
PERMANENT DATA MODULES	P0009D

3. SIZE

PROGRAM MODULE 186 Bytes

4. CALLS TO SUBROUTINE

INTERRUPT LEVEL	3	@PPCIH1
BL		@PPCIH2

5. CALLS FROM SUBROUTINE

PROGRAM MODULE	PPTHAP
BLWP	@PPTHAP

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data to this program is the 3 return vectors in registers R13, R14 and R15, of the workspace area used to service the interrupt (PPTHAW), and also the word containing the current task active number (ACTNO).

6.2. OUTPUT DATA

As a result of the execution of this program the word ACTNO is reset to zero and one of the group of 3 words

MWPRET	DWPRET	IWPRET	OWPRET	CWPRET
MPCRET	DPCRET	IPCRET	OPCRET	CPCRET
MSTRET	DSTRET	ISTRET	OSTRET	CSTRET

are loaded with the 3 returns in R13, R14 and R15, that are required when reactivating the currently interrupted task. To indicate that the task which has just been interrupted, is in the suspended state, its interrupt flag word is set to -1. The interrupt flags for the various tasks are :-

MONINT DIAINT IPINT OPINT MCPINT

Also during the execution of this program, the words MSEC, SEC, MINUTE , HOURS will be changed to indicate the time since the system was started.

7. AUTHOR

B.T.V. WARTON.

```
SCR2.TASK.S.PPCIHP  15:08:06 TUESDAY, OCT 17, 1978.
        TITL 'PATIENT PROCESSOR ADC INT HANDLER'
* PROGRAMME MODULE IDENT (TASK)
        IDT  'PPCIHP'
* TRANSFER VECTORS OR ENTRIES
        DEF  PPCIH1,PPCIH2
* CALLED PROGRAM MODULES
        REF  PPTHAP
        REF  PPTHA1
* LINKED DATA MODULES
        REF  G0016D,G0017D,P0009D
        REF  MWPRET,MPCRET,MSTRET,DWPRET,DPCRET,DSTRET
        REF  OWPRET,OPCRET,OSTRET,IWPRET,IPCRET,ISTRET
        REF  MONINT,DIAINT,IPINT,OPINT,MCPINT
        REF  RWPRET,RPCRET,RSTRET,CWPRET,CPCRET,CSTRET
        REF  MSEC,SR250,SIXTY,SEC,MINUTE,HOURS,ACTNO
        REF  IDH
        RORG
        PSEG
***********       P R O G R A M M E               ***********
PPCIH1 EQU  $
***********          T I M E           ****************
        INC  @MSEC              INC 4MS INTERVAL COUNTER
        C    @SR250,@MSEC       IS MSEC=250=1 SEC
        JNE  OUTIME
        CLR  @MSEC              RESET MSEC
        INC  @SEC               INC SECONDS COUNTER
        C    @SIXTY,@SEC        IS SEC=60=1 MINUTE
        JNE  OUTIME
        CLR  @SEC               RESET SECONDS COUNTER
        INC  @MINUTE            INC MINUTES COUNTER
        C    @SIXTY,@MINUTE     IS MINUTE=60=1 HOUR
        JNE  OUTIME
        CLR  @MINUTE            RESET MINUTE
        INC  @HOURS             INC HOURS COUNT
OUTIME EQU  $
        BL   @PPCIH2            GO SAVE RET OF INTERRUPTED TASK
        SETO @IDH               SET IDH TASK ACTIVE
        BLWP @PPTHA1            GO TO TASK HANDLER
PPCIH2 EQU  $
        MOV  @ACTNO,R5          GET ACTIVE TASK NUMBER
        JEQ  EXIT               IF=0 EXIT
        DECT R5                 IF MONI TASK GOTO MSAVE
        JEQ  MSAVE
        DEC  R5                 IF DIAG TASK GOTO DSAVE
        JEQ  DSAVE
        DEC  R5                 IF I/P TASK GOTO ISAVE
        JEQ  ISAVE
        DEC  R5                 IF O/P TASK GOTO OSAVE
        JEQ  OSAVE
        DEC  R5                 IF MCP TASK GOTO MCSAVE
        JEQ  MCSAVE
EXIT    CLR  @ACTNO             RESET TASK ACTIVE NUMBER
        RT
MSAVE   MOV  R13,@MWPRET        SAVE MONITOR WP RET
        MOV  R14,@MPCRET        SAVE MONITOR PC RET
        MOV  R15,@MSTRET        SAVE MONITOR ST RET
        SETO @MONINT            SET MONI INT FLAG
        JMP  EXIT
DSAVE   MOV  R13,@DWPRET        SAVE DIAGNOSIS WP RET
        MOV  R14,@DPCRET        SAVE DIAGNOSIS PC RET
```

SCR2.TASK.S.PPCIHP 15:08:06 TUESDAY, OCT 17, 1978.

```
       MOV  R15,@DSTRET          SAVE DIAGNOSIS ST RET
       SETO @DIAINT              SET DIAG INT FLAG
       JMP EXIT
ISAVE  MOV  R13,@IWPRET          SAVE I/P TASK WP RET
       MOV  R14,@IPCRET          SAVE I/P TASK PC RET
       MOV  R15,@ISTRET          SAVE I/P TASK ST RET
       SETO @IPINT               SET  IP INT FLAG
       JMP  EXIT
OSAVE  MOV  R13,@OWPRET          SAVE O/P TASK WP RET
       MOV  R14,@OPCRET          SAVE O/P TASK PC RET
       MOV  R15,@OSTRET          SAVE O/P TASK ST RET
       SETO @OPINT               SET O/P TASK INT FLAG
       JMP  EXIT
MCSAVE MOV  R13,@CWPRET          SAVE MCP TASK WP RET
       MOV  R14,@CPCRET          SAVE MCP TASK PC RET
       MOV  R15,@CSTRET          SAVE MCP TASK ST RET
       SETO @MCPINT              SET MCP TASK INT FLAG
       JMP  EXIT
       END
```

PATIENT PROCESSOR UART INTERRUPT HANDLER

1. DESCRIPTION

This program is an operating system program and is concerned with the interrupt generated by the UART (Universal Asynchronous Receiver Transmitter) which is the means by which the patient dedicated processor communicates to the operator dedicated processor.

The functions performed by this program are :-

(1) The testing of the UART interrupt to determine if it is due to a character being received, or the interval timer.

(2) The transmission or reception of the Start of Heading (SOH) character "Acknowledge" (ACK) or "Negative Acknowledge" (NAK) characters for the correct start up of communication between processors.

(3) The activation of either the Input Task or Output Task as required.

(4) The de-activation of successful or unsuccessful communication by the Input or Output tasks.

(5) The de-activation of either the Input or the Output task if the UART interval timer interrupt should occur, and the indication of the error to the operator if necessary.

(6) The programming of the UART for its correct operation at the required bit rate and transmission/reception format.

This operating system program is allowed the privilege of running under an interrupt mask of level 2.

2. IDENTIFICATION

SUBROUTINE	NAME	PPUIH
PROGRAM	MODULE	PPUIHP
GENERAL DATA MODULE		G0016D

3. SIZE

PROGRAM MODULE 332 BYTES

4. CALLS TO SUBROUTINE

INTERRUPT LEVEL	4	@PPUIH1
	B	@PPUIH2
	B	@PPUIH3
	B	@PPUIH4
	B	@PPUIH5
	BL	@PUART

5. CALLS FROM SUBROUTINE

PROGRAM MODULE	PPTHAP
BLWP	@PPTHA1
PROGRAM MODULE	PPCIHP
BL	@PPCIH2

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The three return vectors stored in R13, R14 and R15 of PPUIHW workspace area are saved by a BL @PPCIH2.

6.2. OUTPUT DATA

During the operation of this program, the software flags for activating or suspending the Input or Output task may be set or reset according to the communication situation at the time.

7. EXTERNAL DATA TRANSFER

PERIPHERAL TMS 9902 UART

7.1. INPUT

INTERRUPT LEVEL 4

CRU BASE >1A80

CRU BIT FUNCTIONS AS SPECIFIED
IN THE TMS 9902 DATA SHEETS.

7.2. OUTPUT

CRU BASE >1A80

CRU BIT FUNCTIONS AS SPECIFIED
IN THE TMS 9902 DATA SHEETS.

8. TIMING

Execution time for this program is of the order of 0.2mS in normal communication situations.

9. NOTES

The PPUIH1 entry point to this program is due to the UART interrupt which can be generated either by a character being received, or the UART interval timer elapsing.

The PPUIH2 entry point to the program is due to the Microprocessor Communication Package Task (MCP) requesting that a message be transmitted from the patient processor.

The PPUIH3 entry point to the program is due to the completion of the Input task, which allows the program to reset the UART software flags etc., and to test if output communication has been requested.

The PPUIH4 entry point to this program is due to the successful completion of the Output Task, and allows this program to reset UART software flags.

The PPUIH5 entry point to this program is due to the unsuccessful attempt by the Output task to communicate with the Operator dedicated processor. This section of program re-starts the inter-processor communication protocol by transmitting the "Start of Heading" (SOH) character.

10. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.PPUIHP 16:23:41 WEDNESDAY, OCT 18, 1978.

```
       TITL 'PATIENT PROCESSOR UART INT HANDLER'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'PPUIHP'
* TRANSFER VECTORS OR ENTRIES
       DEF  PPUIH1,PPUIH2,PPUIH3,PPUIH4,PPUIH5,PUART
* CALLED PROGRAMME MODULES
       REF  PPTHAP,PPCIHP
       REF  PPTHA1,PPCIH2
* LINKED DATA MODULES
       REF  G0016D
       REF  PPUIHW,OPTASK,OPAC,IPTASK,OPINT
       REF  IPINT
       RORG
       PSEG
********* P R O G R A M M E **************
PPUIH1 LIMI 2                       S.S INT LEVEL 0
       BL   @PPCIH2                 GO SUSPEND TASK
       LI   R12,>1A80               UART CRU BASE
       TB   19                      TIMINT
       JEQ  TIMELP                  GO TO TIME INT SECTION
       STCR R4,8                    GET CHAR FROM UART
       MOV  R0,R0                   TEST O/P RESPONCE FLAG(-1=Y)
       JNE  OPYES
       CI   R4,>0100                IS CHAR SOH
       JNE  SOHNO
       TB   9                       RCVERR
       JEQ  OPNAK
*
* OUTPUT ACK SECTION
OPACK  LI   R4,>0600                ACK
       BL   @OP                     O/P ACK
       SETO R1                      SOH FLAG
       SETO R2                      I/P FLAG
       CLR  R10                     ERR COUNT
       BL   @SETIME                 GO SET TIME
       BLWP @PPTHA1                 GO TO TASK HANDLER
*
*  CHAR OUTPUT SECTION
OP     SBO  18                      RIENB
       SBO  16                      TRAN ON
       LDCR R4,8                    LOAD UART WITH CHAR
       SBZ  16                      TRAN OFF
       RT
*
* IS RESPONCE TO SOH, ACK? SECTION
OPYES  CI   R4,>0600                IS CHAR ACK
       JNE  OPSOH
       SETO @OPTASK                 STIM O/P TASK
       CLR  R0                      RESET RESPONCE EXPECTED FLAG
       CLR  R10                     ERR COUNT=0
       BL   @SETIME                 GO SET TIME
       SBZ  18                      RESET RBRL & INHIBIT INT
       BLWP @PPTHA1                 GO TO TASK HANDLER
*
* IS CHAR INSTRUCTION SECTION
SOHNO  MOV  R1,R1                   TEST SOH FLAG
       JEQ  OPNAK                   JUMP IF SOH FLAG=0
       SETO @IPTASK                 STIM I/P TASK
       CLR  R1                      RESET SOH FLAG
       CLR  R10                     CLEAR ERROR COUNT
```

```
       BL   @SETIME               GO SET TIME
       SBZ  18                    INHIBIT INT
BOUT   BLWP @PPTHA1               GO TO TASK HANDLER
*
* OUTPUT NAK SECTION
OPNAK  CI   R4,>1500              WAS IT NAK
       JEQ  NAKER
       BL   @PUART                GO PROGRAM UART
       LI   R4,>1500              NAK
       CLR  R2                    RESET I/P FLAG
       BL   @OP                   O/P NAK
NAKER  BL   @ERROR                GO TO ERROR SECTION
       JMP  BOUT
*
* ERROR SECTION
ERROR  INC  R10                   INC ERROR COUNTER
       CI   R10,100               IS THERE 100 ERRORS
       JLT  RTOUT
* INDICATE ERROR
       LI   R12,>1BE0             SET ERROR CRU BASE
       SBO  0                     SET ERROR INDICATOR BIT
       LI   R12,>1A80             RET TO UART CRU BASE
RTOUT  RT
*
* SET UART TIMER SECTION
SETIME SBO  13                    LDIR
       LI   R5,>FA00              16 MS
       LDCR R5,8                  LOAD TIMER
       SBO  20                    ENABLE INTERVAL INT
       RT
*
* TIMER INTERRUPT HANDLER SECTION
TIMELP CLR  @IPTASK               DEACTIVATE IPTASK
       CLR  @OPTASK               DEACTIVATE OPTASK
       ABS  @IPINT                RESET I/P INT FLAG
       ABS  @OPINT                RESET O/P INT FLAG
       CLR  R0                    RESET RESP EXP FLAG
       CLR  R1                    RESET SOH FLAG
       CLR  R2                    RESET I/P FLAG
       BL   @ERROR                GO TO EROR SECTION
       MOV  @OPAC,@OPAC           TEST O/P ACTIVE FLAG
       JNE  OPSOH
       BL   @PUART                GO PROGRAM UART
       JMP  BOUT
* INDICATE ERROR
OPSOH  LI   R4,>0100              SOH
       BL   @OP                   O/P SOH
       SETO R0                    O/P RES EXPECTED
       BL   @SETIME               GO SET TIME
       BLWP @PPTHA1               GO TO TASK HANDLER
*
* O/P REQUEST ENTRY FROM MCP
PPUIH2 LIMI 2                     LOAD PRIVILEGE MASK
       SETO @OPAC                 SET O/P REQUEST ACTIVE
       LWPI PPUIHW
       CLR  R1                    SOH FLAG
       MOV  R2,R2                 I/P FLAG
       JEQ  OPSOH                 I/P NOT ACTIVE
       BLWP @PPTHA1               GO TO TASK HANDLER
*
```

```
* ENTRY FROM IPPTOP
PPUIH3 LIMI 2                   LOAD PRIVILEGE MASK
       CLR  @IPTASK             DEACTIVATE IPTASK
       LWPI PPUIHW
       CLR  R2                  RESET I/P FLAG
       MOV  @OPAC,@OPAC         IS OP ACTIVE
       JNE  OPSOH               SEND SOH
       SBO  18                  REC INT ON
       BLWP @PPTHA1             GO TO TASK HANDLER
*
* ENTRY FROM SUCCESFUL OPPTOP
PPUIH4 LIMI 2                   LOAD PRIVILEGE MASK
       CLR  @OPTASK             DEACTIVATE OPTASK
       LWPI PPUIHW
       CLR  R0                  RESET RESP FLAG
       CLR  R1                  RESET SOH FLAG
       CLR  @OPAC               RESET O/P ACTIVE TASK
       SBO  18                  ENABLE INT
       BLWP @PPTHA1             GO TO TASK HANDLER
*
* ENTRY FROM UNSUCCESFUL OPPTOP
PPUIH5 LIMI 2                   LOAD PRIVILEGE MASK
       CLR  @OPTASK             DEACTIVATE OPTASK
       JMP  OPSOH               GO TO O/P SOH
*
* PROGRAMME UART SECTION
PUART  SBO  31                  RESET
       LI   R7,>6300            8BITS EVEN PARIT 2STOP
       LDCR R7,8                LOAD UART
       SBZ  13                  TIMER NOT PROG
       LI   R7,>0010            LOAD UART
       LDCR R7,12               LOAD UART
       SBO  18                  ENABLE INT
       RT
       END
```

INPUT DATA HANDLER TASK

1. DESCRIPTION

This program is the highest priority task in the patient dedicated processor system, and performs the required signal conditioning on the sampled E C G. signal. It contains the three digital filters and noise detecting software as described in Chapter 5.

This program runs under the interrupt mask of level 2, and thus cannot be interrupted by the UART interrupt at level 4.

2. IDENTIFICATION

SUBROUTINE NAME	IPDHT
PROGRAM MODULE	IPDHTP
SPECIFIC DATA MODULE	IPDHTD
GENERAL DATA MODULE	G0002D
GENERAL DATA MODULE	G0003D
PERMANENT DATA MODULE	P0001D

3. SIZE

PROGRAM MODULE 438 BYTES

4. CALLS TO SUBROUTINE

```
BLWP      @IPDHT1
```

5. INTERNAL DATA TRANSFER

5.1. INPUT DATA

The DFGO software flag is tested but not changed by this program, to determine if it must activate the Monitoring Task.

5.2. OUTPUT DATA

The input samples obtained by this program from the AtoD Converter are conditioned by the digital filters, and the output data from this program is placed in the IBP1 Buffer.

5.3. ADDITIONAL DATA CHANGES

Software flags that may be changed by the program are :-

ARTIF which equals -1 if artefact is present in the input signal.

MONI which is set to -1 if Monitoring Task is activated, and

IDH which is set to 0 to de-activate this program.

6. EXTERNAL DATA TRANSFERS

6.1. INPUT

CRU BASE		>1100
CRU BITS	0 TO 7	ADC INPUT

6.2. OUTPUTS

CRU BASE		>1100
CRU BITS	0 TO 7	DAC OUTPUT
CRU BIT	8	INTERRUPT MASK OR RESET
CRU BIT	9	DAC OUTPUT HOLD

7. TIMING

Execution time 1 mS

8. NOTE

The digital filters implemented in this task are designed for a sample rate of 4mS (i.e. 250 sps).

9. AUTHOR

B.T.V. WARTON.

```
SCR2.PROG.S.IPDHTP  16:32:57 WEDNESDAY, OCT 18, 1978.

       TITL 'INPUT DATA HANDLER TASK (WITH DIGITAL FILTERS)'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'IPDHTP'
* TRANSFER VECTORS OR ENTRIES
       DEF  IPDHT1
* LINKED DATA MODULES
       REF  IPDHTD,G0002D,G0003D,P0001D
       REF  BUFFER,BUFEND,WAHP,WALP,WADC,HPOP
       REF  LPOP,LEVEL,KHP,A1,A2,K1LP
       REF  K2LP,KDC,A1DC,A2DC,C
       REF  IPDHTW,IBUF,IBP1,BUFLEN
       REF  DFGO,MONTDW,MSTIMC,MONI,IDH,VALUE
       REF  ARTIF
       RORG
       PSEG
**************** P R O G R A M M E ****************
IPDHT1 DATA IPDHTW,STARTI
STARTI STCR R0,8                 GET INPUT IN MSB
       MOVB R0,@IBUF(R1)         SAVE UNFILTERED DATA
*
*      D C  FILTER  (WADC)  ****************
STARTF LWPI WADC                 GET DC WA
       MOV  @IPDHTW,R1            GET I/P
       SRA  R1,8                 I/P<1>-1
       MOV  R1,R5                I/P TO ACC1
       S    R4,R5                ACC1-Z3
D11    MOV  R5,R9                GET ACC1
       MOV  @KDC,R10             GET CONSTANT
       BL   @MULT12              12 BIT MULT
       MOV  R9,*R0+              PUT IN BUFFER
       CI   R0,BUFEND            IS R0>BUFFER END
       JLE  D12
       LI   R0,BUFFER            RESET R0 TO BUFFER START
D12    MOV  R9,R6                O/P TO ACC2
       A    R2,R6                ACC2+Z1
D2     MOV  R6,R9                GET ACC2
       MOV  @A1DC,R10            GET A1
       BL   @MULT12              GO MULTIPLY
       MOV  R9,R7                ANS TO ACC3
       MOV  R9,R8                ANS TO ACC4
       A    R3,R7                ACC3+Z2
D3     MOV  R7,R9                GET ACC3
       MOV  @A2DC,R10            GET A2
       BL   @MULT12              GO MULTIPLY
       A    R9,R8                ADD TO ACC4
D4     MOV  R8,R4                ACC4 TO Z3
       MOV  R7,R3                ACC3 TO Z2
       MOV  R6,R2                ACC2 TO Z1
*
* LOW PASS (2ND ORDER ) FILTER
D1     LWPI WALP                 GET LOW PASS FILTER WA
       MOV  *R0+,R4              GET VALUE FROM BUFFER
       CI   R0,BUFEND            IS R0>BUFFER END
       JLE  D5
       LI   R0,BUFFER            RESET TO START OF BUFFER
D5     MOV  R2,R9                GET Z1
       MOV  @K1LP,R7             GET LP FILT K1 CONSTANT
       BL   @MULT8               8 BIT MULT
       A    R9,R4                ADD TO ACC1
       MOV  R3,R9                GET Z2
```

```
        MOV   @K2LP,R7             GET LP FILER K2 CONSTANT
        BL    @MULT8               GO MULTIPLY
        A     R9,R4                ADD TO ACC1
        MOV   R4,R5                ACC1 TO ACC2
        MOV   R2,R6                GET Z1
        SLA   R6,1                 X 2
        A     R6,R5                ADD TO ACC2
        A     R3,R5                ADD Z2 TO ACC2
        MOV   R5,R9
        MOV   @C,R7                GET LP FILTER C CONSTANT
        BL    @MULT8               O/P IN R9
        MOV   R2,R3                SAVE NEW Z2
        MOV   R4,R2                SAVE NEW Z1
        MOV   R9,@LPOP             SAVE O/P
*
* HIGH PASS  ( ARTIFACT DETECTION ) FILTER
        LWPI  WAHP
        MOV   *R0+,R1              GET VALUE FROM BUFFER
        CI    R0,BUFEND            IS R0> BUFFER END
        JLE   D6
        LI    R0,BUFFER            RESET TO START OF BUFFER
D6      MOV   R1,R5                I/P TO ACC1
        S     R4,R5                ACC1-Z3
        MOV   R5,R9                GET ACC1
        MOV   @KHP,R10             GET CONSTANT
        BL    @MULT12              12 BIT MULT
        MOV   R9,@HPOP             SAVE HP O/P
        MOV   R9,R6                O/P TO ACC2
        A     R2,R6                ACC2+Z1
        MOV   R6,R9                GET ACC2
        MOV   @A1,R10              GET A1
        BL    @MULT12              GO MULTIPLY
        MOV   R9,R7                ANS TO ACC3
        MOV   R9,R8                ANS TO ACC4
        A     R3,R7                ACC3+Z2
        MOV   R7,R9                GET ACC3
        MOV   @A2,R10              GET A2
        BL    @MULT12              GO MULTIPLY
        A     R9,R8                ADD TO ACC4
        MOV   R8,R4                ACC4 TO Z3
        MOV   R7,R3                ACC3 TO Z2
        MOV   R6,R2                ACC2 TO Z1
*
* LEVEL DETECTION SECTION
        MOV   @HPOP,R9             GET HP O/P
        ABS   R9                   MAKE VALUE +VE
        SRA   R13,1                Z/2
        A     R13,R9               R9+Z/2
        SRA   R13,1                R13/2
        A     R13,R9               R9+Z/4
        MOV   R9,R13               REPLACE Z
        MOV   *R15+,R10            GET UNFILTERED O/P
        CI    R15,BUFEND           IS R15>BUFFER END
        JLE   D7
        LI    R15,BUFFER           RESET R15 TO START OF BUFFER
D7      C     R9,@LEVEL            TEST LEVEL OF SIGNAL
        JLT   NOART
        LI    R14,250              1 SEC DELAY
        SETO  @ARTIF               SET ARTEFACT FLAG
        SBO   >A                   SET ARTIFACT CRU BIT
```

```
U2      DEC  R14              1 SEC DELAY COUNT
        JEQ  OVER
        MOV  @LPOP,R9         GET LP FILT O/P
        MOV  R9,R10           SAVE R9 IN R10
U1      SLA  R9,8             IS VALUE > 8 BITS ACCURACE
        JNO  DOK
        MOV  R10,R10          IS VALUE +VE
        JGT  P
        LI   R9,>8000         SET TO MIN 8 BIT -VE VALUE
        JMP  DOK
P       LI   R9,>7F00         SET TO MAX 8 BIT +VE VALUE
DOK     MOVB R9,@VALUE        SAVE R9 AT VALUE
        SBO  9                DAC HOLD
        LDCR R9,8             O/P
        SBZ  9                DAC SAMPLE
        SBZ  8                RESET
        SBO  8                RESET
        LWPI IPDHTW           RETURN TO IPDHTW
        MOVB @VALUE,@IBP1(R1) SAVE VALUE IN BUF
        MOV  @DFGO,R2         TEST DFGO
        JGT  GO
        CLR  R1               BUF COUNT
        LWPI MONTDW           GET MONITOR WA
        CLR  R3               RESET. COUNT
        LWPI IPDHTW
        JMP  RETNS            RET NO STIM
GO      INC  R1               INC BUF COUNT
        C    R1,@BUFLEN       IS R1>BUFFER LENGTH
        JLE  STIM
        CLR  R1               RESET COUNT
*
* ACTIVATION OF MONITOR TASK SECTION
STIM    INC  @MSTIMC          INC MONI STIM COUNT
        SETO @MONI            SET MONITOR TASK ACTIVE
*
* DEACTIVATION OF IPDHTP
RETNS   CLR  @IDH             RESET IPDHTP TASK ACTIVE FLAG
        RTWP                  RETURN TO TASK HANDLER
*
* DELAYED HOLD TIME OVER SECTION
OVER    MOV  R10,R9           GET UNFILTERED O/P IN R9
        ABS  @ARTIF           RESET ARTEFACT FLAG
        SBZ  >A               RESET ARTEFACT CRU BIT
        JMP  U1
*
* NO ARTIFACT DURING DELAY TEST SECTION
NOART   MOV  R14,R14          TEST 1 SEC DELAY
        JEQ  OVER
        JMP  U2
*
*       12 BIT MULT     *************
MULT12  ABS  R9               MAKE R9 +VE
        JGT  R9P              JUMP IF R9 +VE
        MPY  R10,R9           R10 X R9 ANSWER IN R9&R10
        SLA  R9,4             GET CORRECT BINARY POINT POSITION
        SRL  R10,12           GET CORRECT BINARY POINT POSITION
        A    R10,R9           MAKE ANSWER INTO SINGLE 16 BIT NUMBE
        NEG  R9               MAKE ANSWER -VE
        RT
R9P     MPY  R10,R9           R10 X R9 ANSWER IN R9&R10
```

SCR2.PROG.S.IPDHTP 16:32:57 WEDNESDAY, OCT 18, 1978.

```
       SLA   R9,4                   GET CORRECT BINARY POINT POSITION
       SRL   R10,12                 GET CORRECT BINARY POINT POSITION
       A     R10,R9                 GET ANSWER INTO SINGLE 16 BIT NUMBER
       RT
*
*      8 BIT MULT      ******************
MULT8  ABS   R9                     MAKE R9 +VE
       JGT   REG9P                  JUMP IF R9 +VE
       ABS   R7                     MAKE R7 +VE
       JGT   R9NR7P                 JUMP IF R7 +VE
R9PR7P MPY   R7,R9                  ANS IN R9&10
       JMP   ANS
REG9P  ABS   R7                     MAKE R7 +VE
       JGT   R9PR7P                 JUMP IF R7 +VE
R9NR7P MPY   R7,R9                  ANSWER IN R9&R10
       INV   R9                     MAKE R9 -VE
       NEG   R10                    MAKE R10 -VE
       JNE   ANS
       INC   R9                     R10 OVERFLOW ADD 1 TO R9
ANS    MOVB R10,R9                  GET INTO SINGLE 16 BIT WORD
       SWPB R9                      GET INTO SINGLE 16 BIT WORD
       RT
       END
```

MONITOR PROGRAM

1. DESCRIPTION

This first module of the monitoring task is concerned with the monitoring start up procedure and the subsequent detection of QRS Complexes. This module also has the exit section from the monitoring program.

2. IDENTIFICATION

SUBROUTINE NAME	MONTD
PROGRAM MODULE	MONTDP
SPECIFIC DATA MODULE	MONTDD
GENERAL DATA MODULE	G0002D
GENERAL DATA MODULE	G0003D
GENERAL DATA MODULE	G0004D
PERMANENT DATA MODULE	P0002D

3. SIZE

PROGRAM MODULE 394 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP     @MONTD1
  B      @FINISH
```

5. CALLS FROM SUBROUTINE

```
PROGRAM MODULE    MONS1P
         B        @MONS11

PROGRAM MODULE    MONS2P
         B        @NOQRS
```

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data for this program is found in IBP1 Buffer as supplied by the Signal Conditioning task (IPDHTP).

6.2. OUTPUT DATA

The output data from this program, which consists of the delayed difference signal, is placed in the buffer DBP1. Also supplied is a word labled POINT which indicates the position of the QRS complex in the IBP1 buffer, and a word labled RRC which is the estimate of the current R-R interval.

6.3. ADDITIONAL DATA CHANGES

Software flags which may change during the program operation are the start up flags SU, SU2 and SU3, and also the monitor MCP request flag. Also an address is placed in MOTEXT and the MCP task flags set when the Monitoring function wishes information to be transmitted to the operator dedicated processor.

7. TIMING

The normal execution time for this program is 0•21mS and occurs while a QRS complex has not been detected.

When a QRS complex is detected the maximum execution time of this program module combined with the other two (MONS1P and MONS2P) which make up the complete monitoring software package is 3•2mS.

For more information on software timing see Chapters 5 and 7.

8. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.MONTDP 16:56:30 WEDNESDAY, OCT 18, 1978.

```
      TITL 'MONITOR PROGRAM'
* PROGRAMME MODULE IDENT (PROG)
      IDT  'MONTDP'
* TRANSFER VECTORS OR ENTRIES
      DEF  MONTD1,FINISH
* CALLED PROGRAM MODULES
      REF  MONS1P,MONS2P
      REF  MONS11,NOQRS
* LINKED DATA MODULES
      REF  MONTDD,G0002D,G0003D,G0004D,P0002D
      REF  MONTDW,SDELAY,SFP1,SU,DBP1
      REF  TDP1,BUFLEN,DMINC
      REF  MINL,IBP1,SU2,RRC,RRC32,RRMAXT
      REF  TEN,SIX,MSTIMC,MONI,DBUFL,SU3,POINT
      REF  MSUDVE,MFAIL,MONSF,MOTEXT,CSTIMC,MCP
      REF  WAHP,MCP
      REF  DFGD,AVRR,CMS,MONS1W,NP,P2POS,QRSWID
      RORG
      PSEG
******************* P R O G R A M M E  ***************************
MONTD1 DATA MONTDW,SMONIT      BLWP VECTORS
SMONIT MOV  @SU,R0             IS START UP ACTIVE
       JGT  MON                JUMP & CONTINUE MONITORING
*
************         START UP SECTION       ****************
       MOV  @SDELAY,R0         IS START UP DELAY REQUIRED
       JGT  D1                 JUMP IF DELAY NOT REQUIRED
       INC  R3                 INC P2
       CI   R3,6               IS P2=6 I/P DELAYS
       JLT  RET1               JUMP<6
       ABS  @SDELAY            SET DELAY TO NOT REQUIRED
       CLR  R2                 CLR POSITION POINTER P1
       CLR  R4                 DIFF REG
       CLR  R5                 COUNTER 2
       CLR  R12                RESET DBPI COUNTER
RET1   B    @EXIT              THIS RET DOES NOT INC P1&P2
D1     MOVB @IBP1(R3),R4       GET VALUE AT P2
       SRA  R4,8               MSB TO LSB
       MOVB @IBP1(R2),R7       GET P1
       SRA  R7,8               MSB TO LSB
       S    R7,R4              P2-P1=R4
       MOV  R4,@DBP1(R12)      SAVE D IN DBP1
       MOV  R2,R2              TEST R2 (P1)
       JNE  D2                 JUMP IF R2NOT=0 (P1)
       MOV  @DBP1,R5           PLACE INMIN REG 5
       CLR  R6                 MIN POSITION
       B    @FINISH            GO TO FINISH SECTION
D2     C    R5,@DBP1(R12)      IS NEW DIFF<MIN(R5)
       JLT  D3                 JUMP IF NO
       MOV  @DBP1(R12),R5      SAVE NEW MIN
       MOV  R12,R6             SAVE POS OF NEW MIN
D3     CI   R2,500             HAVE 500 DIFF VALUES BEEN FOUND
       JGT  D4                 JUMP IF YES
       B    @FINISH            GO TO FINISH SECTION
D4     CI   R6,450             IS POS OF MIN>450
       JLT  D5                 JUMP IF NO
       CI   R2,550             HAVE 550 DIFFS BEEN FOUND
       JGT  D5                 JUMP IF YES
       B    @FINISH            GO TO FINISH SECTION
D5     C    @MINL,R5           IS MIN LIMIT <MIN
```

```
        JLT   BFAIL                 MIN>MINL SU FAIL $$$$$$$$
*
*********** THE NEXT SECTION FINDS TD FOR PAT 1
        MOV   R5,R9                 GET MIN VALUE
        ABS   R9                    MAKE +VE
        MPY   @SIX,R9               MIN X 6 ANS IN R10
        DIV   @TEN,R9               R9=ANS IE MIN X PARATD
        NEG   R9                    MAKE R9 -VE
        MOV   R9,@TDP1              SAVE TDP1
        ABS   @SU                   START UP OVER
        SETO  @SU2                  SET START UP 2 FLAG YES
        SETO  @SU3                  SET START UP 3  ''     ''
        SETO  @MONSF                SET MON OCP FLAG
        LI    R0,MSUOVE             GET O/P ADDRESS
        MOV   R0,@MOTEXT            LOAD MOTEXT WITH O/P TEXT ADDRESS
        INC   @CSTIMC               SU O/P TEXT STIM
        SETO  @MCP                  ACTIVATE MCP TAK
        B     @FINISH               GO TO FINISH SECTION
BFAIL   B     @FAIL                 GO TO START UP FAIL SECTION
******        END OF START UP SECTION
*
******        CONTINUOUS MONITORING SECTION    ***************
MON     MOVB @IBP1(R3),R4           GET VALUE AT P2
        SRA   R4,8                  MSB TO LSB
        MOVB @IBP1(R2),R7           GET P1
        SRA   R7,8                  MSB TO LSB
        S     R7,R4                 P2-P1=R4
        MOV   R4,@DBP1(R12)         SAVE DIFF VALUE IN DBP1
        MOV   @CMS,R0               IS CONTINUE MON FLAG SET -VE
        JLT   TESTTD
        INC   @RRC                  INC RR INTERVAL COUNT
        C     @RRC,@RRMAXT          IS RR INTERVAL>=6 SECONDS
        JGT   BNOQRS                JUMP IF YES
        C     @RRC,@RRC32           IS RR INT =>32SAMPLES
        JGT   TESTTD                JUMP IF YES
        B     @FINISH               GO TO FINISH SECTION
BNOQRS  B     @NOQRS                GO TO NO QRS SECTION
TESTTD  C     R4,@TDP1              IS DIFF>TD
        JGT   D10                   JUMP IF >LIMIT
        ABS   @SFP1                 SEARCH FLAG PAT 1
        INC   R1                    NP+1
        CI    R1,1                  IS NP=1
        JNE   D11                   JUMP IF NO
        MOV   R4,R5                 SAVE MIN VAL
        MOV   R3,R6                 SAVE POS OF MIN
        B     @FINISH               GO TO FINISH SECTION
D11     C     R5,R4                 IS THIS MIN<MIN SOFAR
        JLT   BFIN
        MOV   R4,R5                 SAVE NEW MIN
        MOV   R3,R6                 SAVE POSITION OF NEW MIN
        B     @FINISH               GO TO FINISH SECTION
D10     MOV   @SFP1,R0              TEST SEARCH FLAG
        JGT   BSEARC
        CLR   R1                    CLR NP COUNTER
        CLR   R10                   CLR SIGN CHANGE COUNT
BFIN    B     @FINISH               GO TO FINISH SECTION
BSEARC  MOV   R5,@DMINC             SAVE DMIN CURRENT
        ABS   @CMS                  TEST & RESET CONTINUE MON FLAG
        JGT   CMOK
        MOV   @AVRR,@RRC            MOVE AVERAGE RR INTO RR CURRENT
```

```
CMOK    MOV  R6,@POINT           SAVE POS OF MIN
        MOV  R1,@NP              SAVE NP
        INC  R10                 SHORT TERM QRS WIDTH COUNTER
        C    R10,@QRSWID         IS COUNT=QRSWID
        JEQ  MONSGO              GO MONI SEARCH IF QRS FOUND WIDE
        MOVB @IBP1(R3),R0        TEST FOR SIGN CHANGE
        JLT  FINISH
MONSGO  MOV  R3,@P2POS           SAVE P2 POSITION
        B    @MONS11
FAIL    SETO @MONSF              SET MONI MCP FLAG
        LI   R0,MFAIL            GET O/P ADDRESS
        MOV  R0,@MOTEXT          LOAD MOTEXT WITH ADDRESS OF O/P
        INC  @CSTIMC             STIM  MCP
        SETO @DFGO               SET DFGO SO IPDHTP NOT STIM MONTDP
        SETO @MCP                ACTIVATE MCP
*
*******         THE NEXT SECTION IS THE EXIT FROM MONITOR
FINISH LWPI MONTDW
        INC  R2                  P1+1
        C    R2,@BUFLEN          IS P1>BUFFER LENGTH
        JLE  D8
        CLR  R2                  RESET P1
D8      INC  R3                  P2+1
        C    R3,@BUFLEN          ISP2>BUFFER LENGTH
        JLE  DE
        CLR  R3                  RESET P2
DE      INCT R12                 DBP1 COUNT
        C    R12,@DBUFL          IS DBP1>BUFFER LENGTH
        JLE  EXIT
        CLR  R12                 RESET R12
*       THE NEXT LINES CONTROL MONI STIM COUNTER
EXIT    DEC  @MSTIMC             DEC MONI STIM COUNT
        JNE  EXIT1
        CLR  @MONI               DEACTIVATE MONI TASK
EXIT1   RTWP
        END
```

MONITOR SEARCH SECTION

1. DESCRIPTION

This program is the second module in the monitoring program, and is concerned with the monitoring start-up procedure for setting the initial value of the average R-R interval (AVRR). From then on this program is concerned with setting the test bits in the monitoring word MF1 as follows :-

(1) The setting of pulse rate flags in MF1.

(2) QRS complex width measurement.

(3) Calculating the percentage of wide QRS complexes.

(4) Calculating the true R-R interval.

(5) Classification of R-R intervals as long, short or normal.

2. IDENTIFICATION

SUBROUTINE NAME	MONS1
PROGRAM MODULE	MONS1P
GENERAL DATA MODULE	G0003D
GENERAL DATA MODULE	G0004D
GENERAL DATA MODULE	G0005D
PERMANENT DATA MODULE	P0003D

3. SIZE

PROGRAM MODULE 644 Bytes.

4. CALLS TO SUBROUTINE

B @MONS11

5. CALLS FROM SUBROUTINE

PROGRAM MODULE MONTDP

B @FINISH

PROGRAM MODULE MONS2P

B @MONS21

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data for this program is the output data from the program module MONTDP, this data being the current estimate of the R-R interval (RRC), and the position of the QRS complex (POINT) in the input buffer (IBP1).

6.2. OUTPUT DATA

The output from this program is the monitoring result word MF1, the individual bits of which are the TRUE/FALSE answers to the tests performed by the program. Also supplied by this program is the true R-R interval measurement which is placed in RRSAVE.

6.3. ADDITIONAL DATA CHANGES

The software flags that are effected by this program are SFP1, SU2 and SU3.

7. TIMING

This program is executed only when the MONTDP program has detected a QRS complex. The combined execution time of all the monitoring modules (MONTDP, MONS1P and MONS2P) is 3·2mS.

For more information on software timing see Chapters 5 and 7.

8. AUTHOR

B.T.V. WARTON.

```
SCR2.TASK.S.MONS1P  15:55:04 TUESDAY, OCT 17, 1978.

      TITL 'MONITOR SEARCH SECTION'
* PROGRAMME MODULE IDENT (TASK)
      IDT  'MONS1P'
* TRANSFER VECTORS OR ENTRIES
      DEF  MONS11
* CALLED PROGRAM MODULES
      REF  MONTDP,MONS2P
      REF  FINISH,MONS21
* LINKED DATA MODULES
      REF  G0003D,G0004D,G0005D,P0003D
      REF  SFP1,SU,DBP1
      REF  TDP1,BUFLEN,QRSPT,QRSNT
      REF  MINL,IBP1,POSIT,SU2,RRC
      REF  AVRR,MF1,MF2,BPM140,BPM110,BPM100,BPM90,BPM80
      REF  BPM60,BPM40,BPM20,BPM10,RRSAVE
      REF  AVRRB,RRSB,SEQB,RRLB,COUNT,MINV,MINP,QRSWID
      REF  QRSWB,TDG,CLRWT,BEATS,HUND,T1,W1,T2,W2,T3
      REF  PER82,WIDE82,WID82S,MAXV,MAXP,MONWR3
      REF  MONS1W,SU3,DMINC,DMINO,POINT
      REF  BPM120,TWELVE,FIVE,WQRS
      REF  P2POS,OFSETO
      RORG
      PSEG
****************** P R O G R A M M E ****************************
*                           REG ASSIGNMENT
*     R0=SEQL     R1=SEQS     R2=SWQRS  R8=MF1&MF2
*
******        SEARCH SECTION             *****************
MONS11 LWPI MONS1W
      SETO @SFP1                  RESET SEARCH FLAG
      MOV  @RRC,@RRSAVE           SAVE RR COUNT
      CLR  @RRC                   RESET RRC
      MOV  @SU2,R9                TEST SU2 FLAG
      JGT  SU2NO                  JUMP IF SU2=NO
      MOV  @SU3,R9                TEST SU3
      JGT  SU3NO                  JUMP IF SU3 NO
      ABS  @SU3                   RESET SU3
      B    @FINISH
SU3NO  MOV  @RRSAVE,@AVRR         MAKE RRSAVE=AVERAGE RR FOR START UP
      MOV  @DMINC,@DMINO          MAKE DMINO=DMINC FOR START UP
      ABS  @SU2                   RESET SU2
*
****  SETTING PULSE RATE FLAGS
*
SU2NO  CLR  R8                    MF1=R8
      C    @AVRR,@BPM80           IS PULSE >80 BPM
      JLE  GT80                   JUMP IF YES
      C    @AVRR,@BPM60           IS PULSE >60 BPM
      JGT  LE60                   JUMP IF NO
      AI   R8,>00E0               SET FLAGS >60
      JMP  SD1
LE60   C    @AVRR,@BPM40          IS PULSE >40
      JGT  LE40                   JUMP IF NO
      AI   R8,>00C0               SET FLAGS >40
      JMP  SD1
LE40   C    @AVRR,@BPM20          IS PULSE >20
      JGT  LE20                   JUMP IF NO
      AI   R8,>0080               SET FLAGS >20
      JMP  SD1
LE20   AI   R8,>0000              SET FLAG <20
```

```
       JMP SD1
GT80   C     @AVRR,@BPM90        IS PULSE >90
       JLE   GT90                JUMP IF YES
       AI    R8,>00F0            SET FLAGS >80
       JMP   SD1
GT90   C     @AVRR,@BPM100       IS PULSE >100
       JLE   GT100
       AI    R8,>00F8            SET FLAGS >90
       JMP   SD1
GT100  C     @AVRR,@BPM110       IS PULSE >110
       JLE   GT110
       AI    R8,>00FC            SET FLAGS>100
       JMP SD1
GT110  C     @AVRR,@BPM140       IS PULSE >140
       JLE   GT140
       AI    R8,>00FE            SET FLAGS >110
       JMP   SD1
GT140  AI    R8,>00FF            SET FLAGS >140
SD1    EQU   $
*
****   QRS WIDTH SECTION         **********************
*
WIDTH  EQU   $
       SLA   R2,1                SHIFT SEQ WIDE REG
       MOV   @POINT,R4           GET POINTER TO QRS POS
       CLR   R5                  MIN REG
       CLR   R9                  MAX
       S     @COUNT,R4
       JGT   INBUF
       JEQ   INBUF
       A     @BUFLEN,R4
INBUF  MOVB @IBP1(R4),R5         GET VALUE IN QRS
       MOV   R4,R6               SAVE POS
       CLR   R9                  MAX
       MOVB R5,R9
       MOV   R6,R10              MAX POS
       CLR   R7                  COUNTER
QU1    INC   R4                  INC POS POINT
       C R4,@BUFLEN              IS IT STILL INSIDE BUFFER
       JLE   QD3                 JUMP IF YES
       CLR   R4                  RESET
QD3    CB    R5,@IBP1(R4)        IS MIN<THIS VALUE
       JLT   QD1                 JUMP IF YES
       MOVB @IBP1(R4),R5         SAVE NEW MIN
       MOV   R4,R6               SAVE NEW POS
       JMP   Q1
QD1    CB    R9,@IBP1(R4)        IS VALUE IN R9>THAT IN IBP1
       JGT   Q1
       MOVB @IBP1(R4),R9         SAVE NEW VALUE
       MOV   R4,R10
Q1     C     R4,@MONWR3
       JNE   QU1
*
* AT THIS POINT R5=MIN @ R6=POS& R9=MAX@R10=POS
*
QD3A   SRA   R5,8                MAKE BYTE WORD
       MOV   R5,@MINV            SAVE MIN
       MOV   R6,@MINP            SAVE POS
       SRA   R9,8
       MOV   R9,@MAXV            SAVE MAX VALUE
```

```
       MOV   R10,@MAXP           SAVE MAX VALUE POSITION
       CLR   R3                  WIDTH REG
QU3    INC   R3                  WIDTH COUNT+1
       C     R3,@QRSWID          IS R3>QRS WIDE MEASUREMENT
       JGT   WIDE
       INC   R6                  INC POS
       C     R6,@BUFLEN          IS R6>BUFFER LENGTH
       JLE   QD5
       CLR   R6                  RESET POS
QD5    C     R6,@P2POS
       JEQ   QD5A
       CB    @IBP1(R6),@QRSNT    TEST IBP1 VALUE FOR SIGN CHANGE
       JLT   QU3                 JUMP IF STILL -VE
QD5A   MOV   @MINP,R6            GET MIN POS
QU4    INC   R3                  WIDTH+1
       C     R3,@QRSWID .
       JGT   WIDE
       DEC   R6                  DEC POS. BACKWARD SEARCH
       JGT   QD6
       JEQ   QD6
       MOV   @BUFLEN,R6
QD6    MOVB  @IBP1(R6),R9        TEST IBP1 FOR SIGN CHANGE
       JLT   QU4                 JUMP IF STILL -VE
QU5    INC   R3                  WIDTH+1
       C     R3,@QRSWID
       JGT   WIDE
       DEC   R6                  BACKWARD SEARCH
       JGT   QD7
       JEQ   QD7
       MOV   @BUFLEN,R6          SET POS TO BUFLEN
QD7    CB    @IBP1(R6),@QRSPT    TEST IBP1 FOR CHANGE IN SIGN
       JGT   QU5                 JUMP IF STILL +VE
*      AT THIS POINT R3=WIDTH OF QRS
QU6A   MOV   R3,@WQRS
       C     R3,@QRSWID          IS WIDTH NORMAL
       JLE   QRSN                JUMP IF NORMAL
WIDE   SOC   @QRSWB,R8           SET BIT WIDTH=1
       SOC   @SEQB,R2            SET SEQ BIT WIDE
*
****   82PER WIDE TEST SECTION ***************************
*
QRSN   MOV   @TOG,R9             TEST TOG FLAG
       JLT   T2W2
       ABS   @CLRWT              TEST  CLRWT FLAG
       JGT   COMP1
       CLR   @W1                 WIDE COUNT REG 1
       CLR   @T1                 TOTAL  //    //   //
COMP1  COC   @QRSWB,R8           IS QRSWB=1
       JNE   INCT1
       INC   @W1                 INC WIDE COUNT
INCT1  INC   @T1                 INC TOTAL  //
       C     @T1,@BEATS          IS T1=BEATS
       JNE   SUM
       SETO  @TOG                SET TOG FLAG
       SETO  @CLRWT              //   CLRWT //
       JMP   SUM
T2W2   ABS   @CLRWT              IS CLRWT SET
       JGT   COMP2
       CLR   @W2
       CLR   @T2
```

```
COMP2  COC  @QRSWB,R8           IS QRSWB=1
       JNE  INCT2
       INC  @W2                 INC WIDE COUNT
INCT2  INC  @T2                 //  TOTAL //
       C    @T2,@BEATS          IS T2=BEATS
       JNE  SUM
       ABS  @TOG                RESET FLAG
       SETO @CLRWT              SET FLAG
SUM    MOV  @W1,R9              GET W1
       A    @W2,R9              W1+W2=R9
       MPY  @HUND,R9            R9X100=R9/R10
       MOV  @T1,@T3             GET T1
       A    @T2,@T3              T1+T2=T3
       DIV  @T3,R9              R9/T3=R9 REMAINDER R10
       MOV  R9,@WID82S    T
       C    R9,@PER82           IS R9>=82 PERCENT
       JLT  CALRR
       SOC  @WIDE82,R8          SET 82 WIDE BIT FLAG YES (=1)
*
************  TRUE R-R SECTION .
*
CALRR  MOV  @P2POS,R13          GET P2 CROSS OVER POINT
       S    @MINP,R13
       JGT  DIFOK
       JEQ  DIFOK
       A    @BUFLEN,R13         ANS SHOULD BE +
DIFOK  MOV  @RRSAVE,R14
       A    @OFSETO,R14
       S    R13,R14             R14=RRC+OFSETO-OFSETN=TRR
       JLT  RRERRO
       MOV  R14,@RRSAVE
       MOV  R13@OFSETO
RRERRO EQU  $
*
****   START OF 2 X AVRR SECTION  ************
*
LT82   MOV  @AVRR,R9            GET AVRR
       SLA  R9,1                AVRR X 2
       C    @RRSAVE,R9          IS RRSAVE>=2 X AVRR
       JLT  RRLT                JUMP IF RRSAVE <
       SOC  @AVRRB,R8           SET AVRR BIT=1
*
****   START OF RR = LONG,SHORT OR NORMAL SECTION ******
*
RRLT   SLA  R0,1                SHIFT SEQ LONG REG
       SLA  R1,1                  //    // SHORT REG
       CLR  R9
       MOV  @AVRR,R10           GET AVRR
       C    R10,@BPM120
       JLE  GT120
       SRA  R10,3               1/8
       S    @FIVE,R10           R10 - 5
TESTLS MOV  @AVRR,R9            GET AVRR
       A    R10,R9              AVRR + IER
       C    @RRSAVE,R9
       JGT  RRLONG              JUMP IF RR IS LONG
       S    R10,R9              R9=AVRR
       S    R10,R9              AVRR- IER
       C    @RRSAVE,R9
       JHE  SAVEMF              JUMP IF NOT (IE NORMAL RR)
```

SCR2.TASK.S.MONS1P 15:55:04 TUESDAY, OCT 17, 1978.

```
       SOC   @RRSB,R8              SET SHORT BIT=1
       SOC   @SEQB,R1              SET SEQ SHORT BIT=1
       JMP   SAVEMF
GT120  DIV   @TWELVE,R9
       MOV   R9,R10
       JMP   TESTLS
RRLONG SOC   @RRLB,R8              SET LONG BIT=1
       SOC   @SEQB,R0              SET SEQ LONG BIT=1
*
*********    SAVE MF1         *************
*
SAVEMF EQU   $
       MOV   R8,@MF1               SAVE MF1
       B     @MONS21               GO TO MONS21 PROGRAMME
       END
```

MONITOR SEQUENCE SEARCH SECTION

1. DESCRIPTION

This program is concerned with testing for 9 different sequences of R-R intervals and QRS complex widths, and placing the result in monitoring word MF2.

At the end of this program is the updating procedures for the parameters DMIN, TDP1 and AVRR, and also the procedure for alarming the condition of "no QRS for 6 seconds".

2. IDENTIFICATION

SUBROUTINE NAME	MONS2
PROGRAM MODULE	MONS2P
GENERAL DATA MODULES	G0003D
GENERAL DATA MODULES	G0004D
GENERAL DATA MODULES	G0005D
PERMANENT DATA MODULES	P0004D

3. SIZE

PROGRAM MODULE 322 Bytes.

4. CALLS TO SUBROUTINE

```
B       @MONS21
B       @NOQRS
```

5. CALLS FROM SUBROUTINE

PROGRAM MODULE MONTDP

```
B       @FINISH
```

6. INTERNAL DATA TRANSFER

6.1. INPUT DATA

The input data for this program is supplied in register R0, R1 and R2 of workspace area MONS1W.

The contents of the registers are :-

(i) R0 = Long R-R interval sequence bits.

(ii) R1 = Short R-R interval sequence bits.

(iii) R2 = Wide QRS complex sequence bits.

Data also used by this program, are words MF1, DMINC, DMINO, TDP1, RRSAVE and AVRR.

6.2. OUTPUT DATA

The output from this program is the Monitoring result word MF2, the individual bits of which indicate which sequence of R-R intervals and QRS widths have been detected.

7. TIMING

This program is executed only when the MONTDP program has detected a QRS complex. The combined execution time of all the monitoring modules (MONTDP, MOND1P, MONS2P) is 3·2mS.

For more information on software timing see Chapters 5 and 7.

8. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.MONS2P 17:18:21 WEDNESDAY, OCT 18, 1978.

```
      TITL 'MONITOR SEQUENCE SEARCH SECTION '
* PROGRAMME MODULE IDENT
      IDT  'MONS2P'
* TRANSFER VECTORS OR ENTRIES
      DEF  MONS21,NOQRS
* CALLED PROGRAM MODULES
      REF  MONTDP
      REF  FINISH
* LINKED DATA MODULES
      REF  G0003D,G0004D,G0005D,P0004D
      REF  COC1,CZC1,COC2,CZC2,CZC3,COC3,CZC4,COC4,CZC5
      REF  CZC6,CZC7,COC5,COC6
      REF  SEQ1,SEQ2,SEQ3,SEQ4,SEQ5,SEQ6,SEQ7,SEQ8,SEQ9
      REF  MF2,DIAG,DMINC,DMIND,DSTIMC,EIGHT
      REF  MF1,NORS6B,QRSWB,SIX,TDP1,TEN
      REF  RRC,AVRR,RRSB,RRLB,RRSAVE
      RORG
      PSEG
*****************      PROGRAMME      ***********************
*     THIS NEXT SECTION SETS SEQUENCE FLAG BITS IN MF2
*                 N O T E
*     R0 = LONG SEQUENCE BIT FLAGS
*     R1 = SHORT  ''       ''   ''
*     R2 = WQRS   ''       ''   ''
*
MONS21 CLR  R8                 RESET MF1 REG
*
*******  SEQUENCES  4(S-NS) & 4(S(WQRS)-NS)
      COC  @COC1,R1            TEST SHORT SEQ S-S-S-S-
      JNE  SFAIL1
      CZC  @CZC1,R1            TEST NOT SHORT -NS-NS-NS-NS
      JNE  RETMF2
      COC  @COC1,R2            TEST WQRS W-W-W-W-
      JNE  FOUND1
      CZC  @CZC1,R2            TEST NOT WQRS -NW-NW-NW-NW
      JNE  FOUND1
      MOV  @SEQ2,R8            4(S(W)-NS) FLAG SET IN R8=MF2
      JMP  RETMF2
FOUND1 MOV  @SEQ1,R8           4(S-NS) FLAG SET IN R8=MF2
      JMP  RETMF2
*
*******  SEQUENCES  4(S-NS-N) & 4(S(WQRS)-NS-N)
SFAIL1 COC  @COC2,R1           TEST SHORT S--S--S--S--
      JNE  SFAIL2
      CZC  @CZC2,R1            TEST NOT SHORT -NSNS-NSNS-NSNS-NSNS
      JNE  SFAIL2
      CZC  @CZC3,R0            TEST NOT LONG --NL--NL--NL--NL
      JNE  SFAIL2
      COC  @COC2,R2            TEST WQRS W--W--W--W--
      JNE  FOUND2
      CZC  @CZC2,R2            TEST NOT WQRS -NWNW-NWNW-NWNW-NWNW
      JNE  FOUND2
      MOV  @SEQ4,R8            SET 4(S(W)-NS-N) FLAG BIT=1
      JMP  RETMF2
FOUND2 MOV  @SEQ3,R8           SET 4(S-NS-N) FLAG BIT=1
      JMP  RETMF2
*
*******  SEQUENCES  S-NS-S  &  S-S-S(WQRS)
SFAIL2 COC  @COC3,R1           TEST SHORT S-S
      JNE  SFAIL3
```

```
       CZC   @CZC4,R1           TEST NOT SHORT -NS-
       JNE   FAIL6
       MOV   @SEQ5,R8           SET S-NS-S FLAG BIT =1
       JMP   RETMF2
FAIL6  COC   @COC5,R2           TEST WQRS --W
       JNE   SFAIL3
       CZC   @CZC5,R2           TEST NOT WQRS NWNW-
       JNE   SFAIL3
       MOV   @SEQ7,R8           SET S-S-S(W) FLAG BIT=1
       JMP RETMF2
*
*******  SEQUENCES S-NS-N-NL  & S-NS-N-L
SFAIL3 COC   @COC4,R1           TEST SHORT S---
       JNE   SFAIL4
       CZC   @CZC5,R1           TEST NOT SHORT -NSNS-
       JNE   SFAIL4
       CZC   @CZC6,R0           TEST NOT LONG --NLNL
       JNE   SFAIL5
       MOV   @SEQ6,R8           SET S-NS-N-NL FLAG=1
       JMP   RETMF2
SFAIL5 CZC   @CZC7,R0           TEST NOT LONG --NL-
       JNE   SFAIL4
       COC   @COC5,R0           TEST LONG ---L
       JNE   SFAIL4
       MOV   @SEQ9,R8           SET S-NS-N-L FLAG BIT=1
       JMP   RETMF2
*
*******  SEQUENCE  S-NS-L
SFAIL4 COC   @COC6,R1           TEST SHORT S--
       JNE   RETMF2
       CZC   @CZC7,R1           TEST NOT SHORT -NS-
       JNE   RETMF2
       COC   @COC5,R0           TEST LONG --L
       JNE   RETMF2
       MOV   @SEQ8,R8           SET S-NS-L FLAG BIT=1
RETMF2 MOV   R8,@MF2            SAVE MF2
*
*******      FINDING NEW DMIN & TD IF NEEDED
       MOV   @MF1,R9            GET MF1
       COC   @QRSWB,R9          TEST WQRSIN MF1
       JEQ   AVRUD              NO UP DATE
       MOV   @DMINO,R9          GET DMIN OLD
       ABS   R9                 IS R9 +VE
       JLT   PD1
       JEQ   PD1
       BL    @DMINER            DMIN +VE GO TO ERROR SECTION
PD1    MPY   @EIGHT,R9          DMINO X 8=R9&R10
       DIV   @TEN,R9            R9&R10 / 10=DMINX0.8=R9
       MOV   R9,R8              R8=DMINO X 0.8
       MOV   @DMINC,R10         GET DMIN CURRENT
       ABS   R10                IS DMINC +VE
       JLT   PD2
       JEQ   PD2
       BL    @DMINER            DMINC +VE GO TO ERROR SECTION
PD2    SLA   R10,1              DMIN CURRENT X 2
       CLR   R9
       DIV   @TEN,R9            R9=DMIN CURRENT X0.2
       A     R9,R8              R8=NEW DMINO
       MOV   R8,R9
       MPY   @SIX,R9
```

```
        DIV  @TEN,R9              R9= +TD
        NEG  R9                   R9= -TD
        MOV  R9,@TDP1             LOAD NEW TD
        NEG  R8
        MOV  R8,@DMINO            LOAD NEW DMINO
*       AVRR UPDATE SECTION
AVRUD   MOV  @MF1,R9
        COC  @RRSB,R9             WAS RR SHORT
        JEQ  NUD
        COC  @RRLB,R9             WAS IT LONG
        JEQ  NUD
        MOV  @AVRR,R9             GET AVERAGE RR
        A    @RRSAVE,R9
        SRA  R9,1                 DIV BY 2 TO FIND NEW AVERAGE
        MOV  R9,@AVRR             SAVE NEW AVERAGE
NUD     JMP  STIM
DMINER RT
*       INSERT HERE ERROR SECTION FOR +DMIN'S IF REQIURED
*
*******       NO QRS FOR SIX SECONDS SECTION
NOQRS   MOV  @MF1,R9              GET MF1
        SOC  @NQRS6B,R9           SET NQRS6 BIT=1
        MOV  R9,@MF1              RET MF1
STIM    INC  @DSTIMC              INC DIAG STIM COUNT
        SETO @DIAG                SET DIAG F L A G
        B    @FINISH              GO TO FINISH SECTION
        END
```

DIAGNOSIS PROGRAM

1. DESCRIPTION

This program's function is to use the results from the Monitoring Task (MF1 and MF2), to determine if a diagnosis can be made. If a diagnosis is obtained, the diagnosis program activates the MCP task so that the diagnosis is communicated to the operator dedicated processor.

2. IDENTIFICATION

SUBROUTINE NAME	DIAT1
PROGRAM MODULE	DIAT1P
GENERAL DATA MODULE	G0002D
GENERAL DATA MODULE	G0005D
GENERAL DATA MODULE	G0006D
PERMANENT DATA MODULE	P0005D

3. SIZE

PROGRAM MODULE 220 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP      @DIAT11
```

5. INTERNAL DATA TRANSFERS

5.1. INPUT DATA

The input data to this program is the two words MF1 and MF2 which are generated by the monitoring task.

5.2. OUTPUT DATA

The output data from this program is the two words DF1 and DF2, the individual bits of which indicate (if equal to one) which diagnoses have been found.

5.3. ADDITIONAL DATA CHANGES

The software flags that may be effected by this program are those to activate the MCPTOP Task, i.e. flags DISF, CSTIMC and MCP, and also those flags to de-activate the diagnosis task, i.e. DSTIMC and DIAG.

6. TIMING

The execution time of this program is 0·2mS.

7. AUTHOR

B.T.V. WARTON.

```
        TITL 'DIAGNOSIS PROGRAMME'
* PROGRAMME MODULE IDENT (PROG)
        IDT   'DIAT1P'
* TRANSFER VECTORS OR ENTRIES
        DEF   DIAT11
* LINKED DATA MODULES
        REF   G0002D,G0005D,G0006D,P0005D
        REF   DIAT1W,MF1,MF2,DSTIMC,DIAG,DF1,DF2,MCP
        REF   DSVTA1,DSVTA0,DSVTB1,DSVTB0,DBBB1,DBBB0
        REF   DBRAD1,DBRAD0,DIDID1,DIDID0,DARST1,DARST0
        REF   DASYT1,DASYT0,DESCB1,DESCB0,DPVC1,DPVC0
        REF   DLS1,PRGT60,DISF
        REF   SVT2,BBB3,BRAD4,IDIOV5,ARST6,ASYT7
        REF   ESCBT9,LS23
        REF   CSTIMC,DFGO
        RORG
        PSEG
************* P R O G R A M M E ****************
DIAT11 DATA DIAT1W,SDIAGN
SDIAGN MOV   @MF1,R1             GET MF1
       MOV   @MF2,R2             GET MF2
       CLR   R3                  DF1
       CLR   R4                  DF2
*
*******        CARDIAC ARREST
       COC   @DARST1,R1          TEST FOR TRUE CONDITIONS
       JNE   LS
       CZC   @DARST0,R1          TEST FOR FALSE CONDITIONS
       JNE   LS
       SOC   @ARST6,R3           SET ARST DIAG BIT IN DF1
       B     @DIAGOP
*
*******        LOST SIGNAL
LS     COC   @DLS1,R1            TEST FOR TRUE CONDITIONS
       JNE   PRT
       SOC   @LS23,R4            SET LS DIAG BIT IN DF2
       SETO  @DFGO               HALT MONITORING
       B     @DIAGOP             GO TO DIAG O/P SECTION
*
PRT    COC   @PRGT60,R1          IS PR > 60
       JNE   PRLT60
*
*******        SUPER VENTRICULAR TACH
SVT    COC   @DSVTA1,R1          TEST FOR TRUE CONDITIONS
       JNE   SVTNO1
       CZC   @DSVTA0,R1          TEST FOR FALSE CONDITIONS
       JNE   DIAGOP
       SOC   @SVT2,R3            SET SVT DIAG BIT IN DF1
       JMP   BBB
SVTNO1 COC   @DSVTB1,R1          TEST FOR TRUE COND
       JNE   BBB
       CZC   @DSVTB0,R1          TEST FOR FALSE COND
       JNE   BBB
       SOC   @SVT2,R3            SET SVT DIAG BIT IN DF1
       JMP   BBB
*
*******        BRADYCARDIA
PRLT60 EQU   $
BRAD   COC   @DBRAD1,R1          TEST FOR TRUE COND
       JNE   IDIOVT
       CZC   @DBRAD0,R3          TEST FOR FALSE COND
       JNE   IDIOVT
```

```
       SOC   @BRAD4,R3             SET BRAD DIAG BIT IN DF1
       JMP   ASYT
*
*******        IDIOVENTRICULAR TACH
IDIOVT COC   @DIDIO1,R1            TEST FOR TRUE COND
       JNE   ASYT
       CZC   @DIDIO0,R1            TEST FOR FALSE COND
       JNE   ASYT
       SOC   @IDIOV5,R3            SET IDIOV DIAG BIT IN DF1
*
*******        ASYSTOLE
ASYT   COC   @DASYT1,R1            TEST FOR TRUE COND
       JNE   ESCBT
       CZC   @DASYT0,R1            TEST FOR FALSE COND
       JNE   ESCBT
       SOC   @ASYT7,R3             SET ASYT DIAG BIT IN DF1
*
*******        ESCAPED BEAT
ESCBT  COC   @DESCB1,R1            TEST FOR TRUE COND
       JNE   BBB
       CZC   @DESCB0,R1            TEST FOR FALSE COND
       JNE   BBB
       SOC   @ESCBT8,R3            SET ESCBT DIAG BIT IN DF1
*
*******        BUNDLE  BRANCH BLOCK
BBB    COC   @DBBB1,R1             TEST FOR TRUE COND
       JNE   DIAGOP
       CZC   @DBBB0,R1             TEST FOR FALSE COND
       JNE   DIAGOP
       SOC   @BBB3,R3              SET BBB DIAG BIT IN DF1
*******        DIAGNOSIS O/P
DIAGOP MOV   @DISF,R0              HAS LAST REQUEST BEEN DONE
       JLT   EXITA                 JUMP IF NO
       MOV   R3,@DF1               SAVE DF1
       MOV   R4,@DF2               SAVE DF2
       A     R3,R4                 IS DIAG <> 0
       JEQ   EXITA                 JUMP IIF NO DIAGNOSIS
       SETO  @DISF                 SET MCP DIAG FLAG
       INC   @CSTIMC               STIM MCP
       SETO  @MCP                  STIM MCP
EXITA  DEC   @DSTIMC               DEC DIAG STIM
       JNE   EXIT
       CLR   @DIAG                 CLR DIAG STIM
EXIT   RTWP                        RET TO TASK HANDLER
       END
```

INPUT DATA PROCESSOR TASK (PATIENT)

1. DESCRIPTION

The function of this program is to receive the instructions and data that are transmitted by the Output task of the operator dedicated processor. During this communication the input program must transmit to the operator processor, replies such as Acknowledge (ACK), Negative Acknowledge (NAK) and Enquiry (ENQ) as and when required by the interprocessor communication protocol.

This task also uses the UART interval timer to detect abnormal delays in the normal communication protocol.

2. IDENTIFICATION

SUBROUTINE NAME	IPPTO
PROGRAM MODULE	IPPTOP
GENERAL DATA MODULE	GOO16D
GENERAL DATA MODULE	GOO18D

3. SIZE

PROGRAM MODULE 176 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP    @IPPTO1
BL      @IPPTO2
BL      @IPPTO3
```

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	PPUIHP
B	@PPUIH3

6. INTERNAL DATA TRANSFER

6.1. OUTPUT

The data received by this program is placed in a file called IPFILE.

6.2. ADDITIONAL DATA CHANGES

The softward flags effected by this program are those to activate the MCP task i.e. CSTIMC, MCPIP and MCP flags.

7. EXTERNAL DATA TRANSFERS

PERIPHERAL TMS 9902 UART

7.1. INPUT

INTERRUPT LEVEL 4
CRU BASE >1A80
CRU BIT FUNCTIONS AS SPECIFIED IN THE
TMS 9902 DATA SHEETS.

7.2. OUTPUT

CRU BASE >1A80
CRU BIT FUNCTIONS AS SPECIFIED IN THE
TMS 9902 DATA SHEETS.

8. TIMING

The timing of this program is dependent on the data rate.

9. AUTHOR

B.T.V. WARTON.

```
       TITL 'INPUT DATA PROCESSOR TASK (PATIENT)'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'IPPTOP'
* TRANSFER VECTORS OR ENTRIES
       DEF  IPPT01,IPPT02,IPPT03
* CALLED PROGRAMME MODULES
       REF  PPUIH3
* LINKED DATA MODULES
       REF  G0016D,G0018D
       REF  IPPTOW,IPFILE,MCP,MCPIP,CSTIMC
       REF  PPTHAW
       RORG
       PSEG
********** P R O G R A M M E ***********
IPPT01 DATA IPPTOW,START
START  LI   R1,IPFILE            LOAD FILE ADDRESS
       TB   25                   TIMELP
       JEQ  EXIT                 ABANDON I/P
       BL   @INPUT               GO I/P CHAR
       MOVB R5,*R1               PUT IT IN I/P FILE
       LI   R4,>0500             ENQ
       BL   @IPPT03              O/P ENQ & WAIT
       SETO R5                   MAKE R5=-1
       BL   @INPUT               GO I/P CHAR
       AB   *R1,R5               LAST INST +THIS INST
       INC  R5                   +1 SHOULD=0 IF INST OK
       JNE  OPNAK
       LI   R4,>0600             ACK
       BL   @IPPT03              GO O/P ACK & WAIT
       MOVB *R1+,R5              IS INST +VE
       JLT  NEGINS
*
* END OF TRANSMISSION SECTION
EOTOK  INC  @CSTIMC              STIM MCP TASK
       SETO @MCP                 STIM MCP
       SETO @MCPIP               SET MCP FLAG
EXIT   EQU  $
       LWPI PPTHAW               GET WP
       SBZ  >D                   RESET I/P TASK CRU INDICATOR BIT
       B    @PPUIH3              GOTO PPUIHP ENTRY FROM IPPTOP
*
OPNAK  LI   R4,>1500             NAK
       BL   @IPPT02              O/P NAK
       JMP  EXIT                 EXIT PROGRAM
NEGINS LI   R9,50                COUNT FOR 50 BYTES OF DATA
       CLR  R8                   BCC
NEXT1  STCR R5,8                 GET CHAR
       SBZ  18                   RESET UART
       AB   R5,R8                BCC SUM
       MOVB R5,*R1+              STORE DATA
       DEC R9                    DEC BYTE COUNT
       LI   R4,>0500             ENQ
       BL   @IPPT03              O/P ENQ & WAIT
       MOV  R9,R9                HAS ALL DATA BEEN RECEIVED
       JNE  NEXT1
       STCR R5,8                 GET CHAR
       SBZ  18                   RESET UART
       CB   R5,R8                IS BCC = BCC
       JNE  OPNAK
       LI   R4,>0600             ACK
```

SCR2.PROG.S.IPPTOP 18:04:50 WEDNESDAY, OCT 18, 1978.

```
       BL   @IPPT03              O/P ACK& WAIT
       JMP  EOTOK
*
*******   I N P U T   C H A R
INPUT  SBZ  20                   TIMER INT OFF
       STCR R5,8                 STORE CHAR
       SBZ  18                   RESET UART
       RT
*
*******  O U T P U T   C H A R
IPPT02 SBZ  18                   RESET RBRL
       SBO  16                   TRANSMITTER ON
       LDCR R4,8                 LOAD UART
       SBZ  16                   TRAN OFF
       RT
*
******  S E T  T I M E R
SETIME SBO  13                   ENABLE TIMER COUNT IN UART
       LI   R5,>FA00             16 MS
       LDCR R5,8                 LOAD UART COUNT
       SBO  20                   ENABLE TIMER INT
       RT
*
******* OUTPUT CHAR & WAIT FOR ECHO
IPPT03 MOV  R11,R10              SAVE RET
       BL   @IPPT02              GO O/P CHAR
       BL   @SETIME              GO SET TIMER
RCVLP  TB   21                   RBRL
       JNE  RCVLP                JUMP IF CHAR NOT RECEIVED
       SBZ  20                   TURN OFF TIMER INT,CHAR RECEIVED
       B    *R10                 RETURN
       END
```

OUTPUT DATA PROCESSOR TASK (PATIENT)

1. DESCRIPTION

The function of this program is to transmit instructions and data to the input task of the operator dedicated processor. The information to be transmitted is supplied to this program by the MCP task. During the communication this output task must receive the replies transmitted from the operator processor, informing this task of normal (ACK) or abnormal (NAK) communication, or requesting the next character (ENQ).

This task also uses the UART interval timer to detect abnormal delays in the normal communication protocol.

2. IDENTIFICATION

SUBROUTINE NAME	OPPTO
PROGRAM MODULE	OPPTOP
GENERAL DATA MODULES	G0016D
GENERAL DATA MODULES	G0018D

3. SIZE

PROGRAM MODULE 130 BYTES.

4. CALLS TO SUBROUTINE

BLWP @OPPTO1

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	PPUIHP
B	@PPUIH4
B	@PPUIH5
SUBROUTINE NAME	IPPTOP
BL	@IPPTO2
BL	@IPPTO3

6. INTERNAL DATA TRANSFER

6.1. INPUT

The input data for this task is found at the address placed in word OPDATA. This allows PROM resident messages to be sent. If however data is to be transferred, the instruction and data are placed in the file called OPFILE.

6.2. ADDITIONAL DATA CHANGES

The only software flag effected by this task is the output task active flag (OPTASK), which is reset when the task is completed.

7. EXTERNAL DATA TRANSFERS

PERIPHERAL TMS 9902 UART

7.1. INPUT

INTERRUPT LEVEL 4
CRU BASE >1A80
CRU BIT ASSIGNMENT AS SPECIFIED IN
THE TMS 9902 DATA SHEETS.

7.2. OUTPUT

CRU BASE >1A80
CRU BIT ASSIGNMENT AS SPECIFIED IN
THE TMS 9902 DATA SHEETS.

8. TIMING

The timing of this program is dependent on the data rate.

9. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.OPPTOP 13:11:13 WEDNESDAY, OCT 18, 1978.

```
      TITL 'OUTPUT DATA PROCESSOR TASK (PATIENT)'
* PROGRAMME MODULE IDENT (PROG)
      IDT  'OPPTOP'
* TRANSFER VECTORS OR ENTRIES
      DEF  OPPT01
* CALLED PROGRAM MODULES
      REF  PPUIHP,IPPTOP
      REF  PPUIH4,PPUIH5,IPPT02,IPPT03
* LINKED DATA MODULES
      REF  OPPTOW,OPDATA,OPTASK
      REF  PPTHAW
      RORG
      PSEG
********** P R O G R A M M E ************
OPPT01 DATA OPPTOW,START
START  TB   25                    TIMELP
       JEQ  BADGO
       MOV  @OPDATA,R1            GET O/P TEXT ADDRESS
       MOVB *R1+,R4               GET INST
       BL   @IPPT03               O/P INST & SET TIMER & WAIT
       INV  R4                    INV INST
       BL   @IPPT03               O/P INV INS SET TIMER & WAIT
       STCR R5,8                  GET I/P
       SBZ  13
       CI   R5,>0600              IS IT ACK
       JNE  TRYNAK
       INV  R4                    IS INST POS
       JLT  NEGINS
OPEOT  LI   R4,>0400              EOT
       BL   @IPPT02               GO O/P CHAR
       LWPI PPTHAW                T
       SBZ  >E                    T
       B    @PPUIH4               GOTO PPUIHP ENTRY FROM OPPTOP
TRYNAK CI   R5,>1500              NAK
       JNE  TOUT
BADGO  EQU  $
       LWPI PPTHAW                T
       SBZ  >E                    T
       B    @PPUIH5               GOTO PPUIHP FROM UNSUCCESFUL OPPTOP
TOUT   BL   @SETIME
       CLR  @OPTASK
       RTWP
SETIME SBO  13
       LI   R5,>FA00              16 MS
       LDCR R5,8
       SBO  20
       RT
NEGINS LI   R9,50                 COUNT=50
       CLR  R8                    BCC
NEXT   MOVB *R1+,R4               GET DATA
       AB   R4,R8                 BCC=BCC+DATA
       DEC  R9                    COUNT=COUNT-1
       BL   @IPPT03
       MOV  R9,R9                 TEST COUNT
       JNE  NEXT
       MOVB R8,R4                 GET BCC
       BL   @IPPT03
       STCR R5,8                  GET I/P
       CI   R5,>0600              ACK
       JEQ  OPEOT
       JMP  TRYNAK
       END
```

MICROPROCESSOR COMMUNICATION PACKAGE

1. DESCRIPTION

This program's function is to generate or interpret the communication instructions and data which are passed between the patient and operator dedicated processors. The generation of instructions and data by this program is due to either a request from the Monitoring or Diagnosis tasks, or by a request for data from the operator processor. The instructions sent by the operator processor are interpreted by this program and the requested action performed.

2. IDENTIFICATION

SUBROUTINE NAME	MCPTO
PROGRAM MODULE	MCPTOP
GENERAL DATA MODULES	G0004D
GENERAL DATA MODULES	G0005D
GENERAL DATA MODULES	G0017D
GENERAL DATA MODULES	G0018D
PERMANENT DATA MODULES	P0009D

3. SIZE

PROGRAM MODULE 304 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP    @MCPTO1
```

5. CALLS FROM SUBROUTINE

```
SUBROUTINE NAME      PPUIHP
      B              @PPUIH2
```

6. INTERNAL DATA TRANSFER

6.1. INPUT

The input data to this program are the request

MONSF, DISF and MCPIP

and the data found at locations,

MOTEXT, IPFILE, DF1, DF2, MF1, MF2, and AVRR.

6.2. OUTPUT

The output from this program is used by the output task and is in the form of an address placed in location OPDATA, and instructions and data placed in the output file (OPFILE).

6.3. ADDITIONAL DATA CHANGES

As a result of instructions received by this program the following flags and data locations may be set or reset accordingly :-

MONSF, DISF, CSTIMC, MCP, SU, SU2, SU3,
T1, T2, T3, W1, W2, MONS1W, DFGO, SDELAY.

7. TIMING

The execution time of this program is dependent on the complexity of the operation it has been requested to perform, and also the number of times it is interrupted by higher priority tasks.

8. AUTHOR

B.T.V. WARTON.

```
       TITL 'MICROPROCESSOR COMMUNICATION PACKAGE'
* PROGRAMME MODULE IDENT (PROG)
       IDT   'MCPTOP'
* TRANSFER VECTORS OR ENTRIES
       DEF   MCPT01
* CALLED PROGRAMME MODULES
       REF   PPUIHP
       REF   PPUIH2
* LINKED DATA MODULES
       REF   G0004G,G0005D,G0017D,G0018D,P0009D
       REF   MCPTOW,MCPIP,MONSF,DISF,MOTEXT
       REF   SM,EM,PR,CM,OPDATA,CSTIMC,MCP
       REF   SU,SU2,SU3,RRC,T1,T2,T3,W1,W2,MONS1W
       REF   DFGO,MM3,MM1,OPFILE,PRM,AVRR,IPFILE,CMS,MM4
       REF   OPAC,DF1,DF2,DIAGM,MM2,SDELAY,MM5,MM6
       REF   PS,PSD,MF1,MF2
       RORG
       PSEG
********* P R O G R A M M E **********
MCPT01 DATA MCPTOW,START
START  ABS   @MCPIP               IS IT I/P FROM UP
       JLT   MINPUT
       ABS   @MONSF               IS IT MONI MCP REQUEST
       JLT   MIM
       MOV   @DISF,R7             IS IT DIAG MCP REQUEST
       JLT   DIM
       B     @OUT                 JUMP OUT
*
****** MONITOR TASK MESSAGE O/P REQUEST
MIM    MOV @MOTEXT,R2             GET MON O/P ADDRESS
       JMP   STIMOP
*
******* DIAGNOSIS TASK MESSAGE  O/P REQUEST
DIM    LI    R1,OPFILE            GET OPFILE ADDRESS
DOCLR  CLR   *R1+                 CLEAR OPFILE
       CI    R1,MCPTOW
       JNE   DOCLR
       LI    R1,OPFILE            GET OPFILE ADDRESS
       MOV   @DIAGM,*R1+          SAVE DIAG MESSAGE IN OPFILE
       MOV   @DF1,*R1+            SAVE DF1 IN OPFILE
       MOV   @DF2,*R1+            SAVE DF2 IN OPFILE
       LI    R2,OPFILE            GET OPFILE ADDRESS
       ABS   @DISF                RESET DIAG MESSAGE REQUEST
       JMP   STIMOP
*
****** INTER PROCESSOR MESSAGE REQUEST FROM I/P TASK
MINPUT LI    R4,IPFILE            GET ADDRESS OF I/P FILE
       CB    @SM,*R4              IS IT START MON INST
       JEQ   DOSM
       CB    @EM,*R4              IS IT END MON INST
       JEQ   DOEM
       CB    @PR,*R4              IS IT PULSE RATE INST
       JEQ   DOPR
       CB    @CM,*R4              IS IT CONTINUE MON INST
       JEQ   DOCM
       CB    @PS,*R4              IS IT PATIENT STATUSE REQUEST
       JEQ   DOPS
* ERROR IN INST TELL OPERATOR
       LI    R2,MM2               GET ERROR MESSAGE ADDRESS
*
```

```
****** ACTIVATION OF O/P TASK IF NOT ACTIVE
STIMOP MOV  @OPAC,R0            IS O/P ACTIVE
       JNE  STIMOP              WAIT FOR NOT ACTIVE *********
       MOV  R2,@OPDATA          GIVE O/P TASK O/P DATA ADDRESS
OUT    DEC  @CSTIMC             DEC MCP STIM COUNT
       JNE  EXIT
       CLR  @MCP                RESET MCP TASK ACTIVE FLAG
EXIT   B    @PPUIH2             GO TO UART INT HANDLER TO O/P SOH
*
******* START MONITORING SOFTWARE SECTION
DOSM   MOV  @DFGO,R7            IS IT MONITORING
       JGT  SMON
       SETO @SU                 SET MON START UP FLAGS
       SETO @SU2                SET MON START UP FLAGS
       SETO @SU3                SET MON START UP FLAGS
       SETO @SDELAY             SET MON START UP DELAY FLAG
       CLR  @RRC                CLEAR MON LOCATIONS
       CLR  @T1                 CLEAR QRS TOTAL 1
       CLR  @T2                 CLEAR QRS TOTAL 2
       CLR  @T3                 CLEAR QRS TOTAL 3
       CLR  @W1                 CLEAR QRS WIDE TOTAL 1
       CLR  @W2                 CLEAR QRS WIDE TOTAL 2
       LI   R5,MONS1W
       CLR  *R5+                CLEAR MON SEQUENCE REGS
       CLR  *R5+                CLEAR MON SEQUENCE REGS
       CLR  *R5                 CLEAR MON SEQUENCE REGS
       ABS  @DFGO               SET FLAG SO PPIDHP STIMS MON TASK
       LI   R2,MM3              MONITORING MESSAGE
       JMP  STIMOP
SMON   LI   R2,MM5              GET ALREADY MON MESSAGE
       JMP  STIMOP
*
****** END OF MONITORING SECTION
DOEM   MOV  @DFGO,R7            IS SYSTEM MONITORING
       JLT  EMNO                JUMP IF NO
       SETO @DFGO               SET DFGO SO IPDHTP DOSE NOT STIM MONI
       LI   R2,MM1              END MONITORING MESSAGE
       JMP  STIMOP
EMNO   LI   R2,MM6              GET NOT MONITORING MESSAGE
       JMP  STIMOP
*
****** PULSE RATE DATA TRANSMISSION SECTION
DOPR   LI   R3,OPFILE           GET OPFILE ADDRESS
       MOV  @PRM,*R3+           LOAD PR INST IN FILE
       MOV  @AVRR,*R3+          LOAD AVERAGE RR DATA IN FILE
       LI   R2,OPFILE           GET OPFILE ADDRESS
       JMP  STIMOP
*
****** DO CONTINUE MONITORING INSTRUCTION
DOCM   MOV  @SU,R7              IS SYSTEM MONITORING
       JLT  CMNO                JUMP IF NO
       ABS  @DFGO               RESET DFGO FLAG SO IPDHTP STIMS MONI
       SETO @CMS                SET CMS SO MONI KNOWES TO CONT MONI
       LI   R2,MM4              GET CONT MONI REPLY,MONITORING AGAIN
       JMP  STIMOP
CMNO   LI   R2,MM5              GET NOT MONITORING MESSAGE
       JMP  STIMOP
*
****** PATIENT STATUS DATA TRANSMISSION SECTION
DOPS   LI   R3,OPFILE           GET OPFILE ADDRESS
```

SCR2.PROG.S.MCPTOP 18:17:54 WEDNESDAY, OCT 18, 1978.

```
       MOV   @PSD,*R3+            LOAD OPFILE WITH PATIENT STATUS INST
       MOV   @AVRR,*R3+           LOAD OPFILE WITH AVERAGE RR
       MOV   @DF1,*R3+            LOAD OPFILE WITH DF1
       MOV   @DF2,*R3+            LOAD OPFILE WITH DF2
       MOV   @MF1,*R3+            LOAD OPFILE WITH MF1
       MOV   @MF2,*R3+            LOAD OPFILE WITH MF2
       LI    R2,OPFILE            GET OPFILE ADDRESS
       JMP   STIMOP
       END
```

PERMANENT DATA MODULE P0001D

1. DESCRIPTION

This data module contains the digital filter coefficients used in the signal conditioning program (IPDHTP).

2. IDENTIFICATION

PERMANENT DATA MODULE   P0001D

3. SIZE

20 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0001D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0001D'
       EVEN
       DEF  LEVEL,KHP,A1,A2,K1LP,K2LP,C
       DEF  KDC,A1DC,A2DC
       RORG
       PSEG
LEVEL  DATA >1A             NOISE THRESHOLD
KHP    DATA >04FC           60 HZ HP FILTER
A1     DATA >2A7F           ''
A2     DATA >153F           ''
K1LP   DATA >0124           25 HZ LP FILTER
K2LP   DATA >FF97           ''
C      DATA >11             ''
KDC    DATA >0FDB           DC FILTER
A1DC   DATA >0048           ''
A2DC   DATA >0024           ''
       END
```

PERMANENT DATA MODULE P0002D

1. DESCRIPTION

This data module contains the data specifying the length of the input ECG and difference buffers etc., used by the Monitoring Task.

2. IDENTIFICATION

PERMANENT DATA MODULE P0002D

3. SIZE

12 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0002D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0002D'
       EVEN
       DEF  BUFLEN,DBUFL,MINL,RRC32,TEN,SIX
       RORG
       PSEG
BUFLEN DATA 750             BUFFER LENGTH
DBUFL  DATA >60             DIFF BUFF LENGTH
MINL   DATA >FFF5           MIN LIMIT *******
RRC32  DATA 32              32 RR SAMPLES DELAY
TEN    DATA 10              CONSTANT
SIX    DATA 6               CONSTANT
       END
```

PERMANENT DATA MODULE P0003D

1. DESCRIPTION

This data module contains the information used by the Monitoring Task, such as pulse rate comparison words and normal QRS width limit etc.

2. IDENTIFICATION

PERMANENT DATA MODULE P0003D

3. SIZE

46 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0003D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0003D'
       EVEN
       DEF  BPM140,BPM120,BPM110,BPM100,BPM90
       DEF  BPM80,BPM60,BPM40,BPM20
       DEF  AVRRB,RRLB,RRSB,SEQB,COUNT,QRSWID
       DEF  QRSWID,QRSWB,BEATS,HUND,TWELVE,FIVE
       DEF  PER82,WIDE82,QRSPT,QRSNT
       RORG
       PSEG
BPM140 DATA 107             RR INT = 140 BPM
BPM120 DATA 125             RR INT = 120 BPM
BPM110 DATA 136             RR INT = 110 BPM
BPM100 DATA 150             RR INT = 100 BPM
BPM90  DATA 166             RR INT =  90 BPM
BPM80  DATA 187             RR INT =  80 BPM
BPM60  DATA 250             RR INT =  60 BPM
BPM40  DATA 375             RR INT =  40 BPM
BPM20  DATA 750             RR INT =  20 BPM
AVRRB  DATA >0400           RR<2XAVRR BIT
RRLB   DATA >0800           RR LONG BIT
RRSB   DATA >1000           RR SHORT BIT
SEQB   DATA >0001           SEQUENCE BIT
COUNT  DATA 20
QRSWID DATA 28              NORMAL WIDTH
QRSWB  DATA >2000           QRS WIDE BIT
BEATS  DATA 200             BEATS COUNT
HUND   DATA 100             CONST
TWELVE DATA 12              CONST
FIVE   DATA 5               CONST
PER82  DATA 82              82 PERCENT
WIDE82 DATA >4000           ''    ''     BIT
QRSPT  BYTE 5               QRS +VE THRESHOLD
QRSNT  BYTE -5              QRS -VE THRESHOLD
       END
```

PERMANENT DATA MODULE P0004D

1. DESCRIPTION

This data module contains the comparison words used by the Monitoring Task, to detect various sequences of R-R interval and QRS complex width.

2. IDENTIFICATION

PERMANENT DATA MODULE P0004D

3. SIZE

50 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0004D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0004D'
       EVEN
       DEF  CDC1,CDC2,CDC3,CDC4,CDC5,CDC6
       DEF  CZC1,CZC2,CZC3,CZC4,CZC5,CZC6,CZC7
       DEF  SEQ1,SEQ2,SEQ3,SEQ4,SEQ5,SEQ6,SEQ7
       DEF  SEQ8,SEQ9
       DEF  NQRS6B,EIGHT,RRMAXT
       RORG
       PSEG
CDC1   DATA >00AA          4(S-NS) & 4(S(W)-NS)
CZC1   DATA >0055          ''
CDC2   DATA >0924          4(S-NS-N) & 4(S(W)-NS-N)
CZC2   DATA >06DB          ''
CZC3   DATA >0249          ''
CDC3   DATA >0005          S-NS-S
CZC4   DATA >2             ''
CDC4   DATA >8             S-NS-N-NL
CZC5   DATA >6             ''
CZC6   DATA >3             ''
CZC7   DATA >2             S-NS-N-L
CDC5   DATA >1             ''
CDC6   DATA >4             S-NS-L
SEQ1   DATA >1             4(S-NS)
SEQ2   DATA >2             4(S(W)-NS)
SEQ3   DATA >4             4(S-NS-N)
SEQ4   DATA >8             4(S(W)-NS-N)
SEQ5   DATA >10            S-NS-S
SEQ6   DATA >20            S-NS-N-NL
SEQ7   DATA >40            3S-W
SEQ8   DATA >80            S-NS-L
SEQ9   DATA >100           S-NS-N-L
NQRS6B DATA >0200          NO QRS 6SEC BIT
RRMAXT DATA 1500           MAX TIME TO QRS 6SEC
EIGHT  DATA 8              CONSTANT
       END
```

PERMANENT DATA MODULE P0005D

1. DESCRIPTION

This data module contains the comparison words used by the diagnosis Task, to diagnose the different kinds of ECG arrhythmias. Also included in this module are the words used when setting the different diagnosis bits in DF1 and DF2.

2. IDENTIFICATION

PERMANENT DATA MODULE P0005D

3. SIZE

52 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
          TITL 'PERMANENT DATA MODULE P0005D'
* PERMANENT DATA MODULE IDENTIFIER
          IDT   'P0005D'
          EVEN
          DEF   SVT2,BBB3,BRAD4,IDIOV5,ARST6
          DEF   ASYT7,ESCBT8,LS23
          DEF   DSVTA1,DSVTA0,DSVTB1,DSVTB0,DBBB1,DBBB0
          DEF   DBRAD1,DBRAD0,DIDIO1,DIDIO0,DARST1,DARST0
          DEF   DASYT1,DASYT0,DESCB1,DESCB0,DLS1
          DEF   PRGT60
          RORG
          PSEG
SVT2      DATA >0002                SVT DIAG BIT SET
BBB3      DATA >0004                BBB DIAG BIT SET
BRAD4     DATA >0008                BRAD DIAG BIT SET
IDIOV5    DATA >0010                IDIOV DIAG BIT SET
ARST6     DATA >0020                ARST  DIAG BIT SET
ASYT7     DATA >0040                ASYT  DIAG BIT SET
ESCBT8    DATA >0080                ESCBT DIAG BIT SET
LS23      DATA >0040                LS    DIAG BIT SET
DSVTA1    DATA >00FF                SVT DIAG TEST
DSVTA0    DATA >4000                SVT DIAG TEST
DSVTB1    DATA >00FE                SVT DIAG TEST
DSVTB0    DATA >0001                SVT DIAG TEST
DBBB1     DATA >0400                BBB DIAG TEST
DBBB0     DATA >0001                BBB DIAG TEST
DBRAD1    DATA >00C0                BRAD DIAG TEST
DBRAD0    DATA >003F                BRAD DIAG TEST
DIDIO1    DATA >0080                IDIO DIAG TEST
DIDIO0    DATA >007F                IDIO DIAG TEST
DARST1    DATA >0200                ARST DIAG TEST
DARST0    DATA >001F                ARST DIAG TEST
DASYT1    DATA >0400                ASYT DIAG TEST
DASYT0    DATA >0A1F                ASYT DIAG TEST
DESCB1    DATA >2800                ESCBT DIAG TEST
DESCB0    DATA >103F                ESCBT DIAG TEST
DLS1      DATA >02F0                LOST SIG DIAG TEST
PRGT60    DATA >00E0                PR > 60 TEST
          END
```

PERMANENT DATA MODULE P0009D

1. DESCRIPTION

This data module contains the list of allowed instructions which can be received by the patient monitoring processor; and also the list of allowed messages which can be sent to the operator dedicated processor (MM1 TO MM6).

2. IDENTIFICATION

PERMANENT DATA MODULE P0009D

3. SIZE

24 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0009D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0009D'
       EVEN
       DEF  MSUOVE,MFAIL,PRM,PR,SM,EM,CM,DIAGM
       DEF  MM1,MM2,MM3,MM4,MM5,MM6,SR250,SIXTY
       DEF  PSD,PS
       RORG
       PSEG
PRM    DATA >C000           PR DATA MESSAGE
DIAGM  DATA >A300           DIAG ''  ''
PSD    DATA >8100           PATIENT STATUS DATA
MSUOVE BYTE >22             START UP OVER
MFAIL  BYTE >24             MONITOR FAIL
PR     BYTE >5C             SEND PR
SM     BYTE >50             START MON
EM     BYTE >53             END MON
CM     BYTE >55             CONTINUE MON
PS     BYTE >56             SEND PATIENT STATUS
MM1    BYTE >27             END OF MONITORING
MM2    BYTE >2B             ERROR MESSAGE
MM3    BYTE >21             MONITORING
MM4    BYTE >28             CONT MON
MM5    BYTE >2D             ALREADY MON
MM6    BYTE >2E             NOT MON
       EVEN
SR250  DATA 250             250 SAMPLES IN 1 SECOND
SIXTY  DATA 60              60 SEC 1 MIN
       END
```

SPECIFIC DATA MODULE IPDHTD

1. DESCRIPTION

This data module contains the workspace areas used for the different digital filters in program module IPDHTP, plus the locations where the output from the high and low pass filters are saved, as well as the output from the filtering network.

Also included in this module is the delay buffer present in the filtering network. It should be noted that this module must be the first in RAM as it contains the label RAM used by the initialisation program.

2. IDENTIFICATION

SPECIFIC DATA MODULE IPDHTD

3. SIZE

CLASS TWO	132	BYTES.
CLASS THREE	61	BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'INPUT DATA HANDLER SPECIFIC DATA MODULE'
* SPECIFIC DATA MODULE IDENTIFIER
       IDT  'IPDHTD'
       EVEN
* CLASS TWO
       DEF  RAM,RAM2
       DEF  IPDHTW,WAHP,WALP,WADC,HPOP,LPOP
       RORG
       PSEG
RAM    EQU  $                START OF RAM
RAM2   EQU  $+2              SECOND WORD IN RAM
IPDHTW BSS  32               IPDHT WORKSPACE AREA
WAHP   BSS  32               HIGH PASS FILT WORKSPACE AREA
WALP   BSS  32               LOW  PASS FILT WORKSPACE AREA
WADC   BSS  32               DC FILTER WORKSPACE AREA
HPOP   BSS  2                HIGH PASS FILTER O/P
LPOP   BSS  2                LOW PASS FILTER O/P
*
* CLASS THREE
       DEF  VALUE,BUFFER,BUFEND
BUFFER BSS  58               DELAY BUFFER IN IPDHTP
BUFEND BSS  2                END OF THIS BUFFER
VALUE  BSS  1                O/P FROM FILTER NETWORK
       END
```

SPECIFIC DATA MODULE MONTDD

1. DESCRIPTION

This data module contains the workspace area used by the monitoring task MONTDP.

2. IDENTIFICATION

SPECIFIC DATA MODULE MONTDD

3. SIZE

CLASS TWO 32 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'MONITOR SPECIFIC DATA MODULE'
* SPECIFIC DATA MODULE IDENTIFIER
       IDT  'MONTDD'
       EVEN
* CLASS TWO
       DEF  MONTDW,MONWR3
       RORG
       PSEG
MONTDW BSS  32                  MONITOR WORKSPACE AREA
MONWR3 EQU  MONTDW+6
       END
```

GENERAL DATA MODULE G0016D

1. DESCRIPTION

This data module contains the task handler workspace area, in which R5 is the task active number store. Also present in this module are the storage locations for the return vectors of suspended tasks; task active flags and counters, storage of time counters and the workspace area used by the PPUIHP program module.

2. IDENTIFICATION

GENERAL DATA MODULE G0016D

3. SIZE

CLASS TWO 122 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
        TITL 'GENERAL DATA MODULE G0016D'
* GENERAL DATA MODULE IDENTIFIER
        IDT   'G0016D'
        EVEN
* CLASS TWO
        DEF   PPTHAW,MRTWP,DRTWP,IPRTWP,OPRTWP,MCRTWP
        DEF   MWPRET,MPCRET,MSTRET,DWPRET,DPCRET,DSTRET
        DEF   OWPRET,OPCRET,OSTRET,IWPRET,IPCRET,ISTRET
        DEF   CWPRET,CPCRET,CSTRET
        DEF   IDH,MONI,DIAG,MSTIMC,DSTIMC,IPTASK,OPTASK
        DEF   CSTIMC,MCP,OPAC,ACTNO,MSEC,SEC,MINUTE
        DEF   HOURS,PPUIHW
        RORG
        PSEG
PPTHAW  BSS   6              TASK HANDLER WA START POINT
ACTNO   EQU   PPTHAW+10      R5 IN PPTHAW IS ACTIVE TASK STORE
MRTWP   BSS   6              ADDRESS FOR MONITOR RETURNS
DRTWP   BSS   6              ADDRESS FOR DIAGNOSIS RETURNS
IPRTWP  BSS   6              ADDRESS FOR I/P TASK RETURNS
OPRTWP  BSS   6              ADDRESS FOR O/P TASK RETURNS
MCRTWP  BSS   2              ADDRESS FOR MCP TASK RETURNS
MWPRET  BSS   2              MONITOR WP RET STORE
MPCRET  BSS   2              MONITOR PC RET STORE
MSTRET  BSS   2              MONITOR ST RET STORE
DWPRET  BSS   2              DIAGNOSIS WP RET STORE
DPCRET  BSS   2              DIAGNOSIS WP RET STORE
DSTRET  BSS   2              DIAGNOSIS WP RET STORE
IWPRET  BSS   2              I/P TASK WP RET STORE
IPCRET  BSS   2              I/P TASK PC RET STORE
ISTRET  BSS   2              I/P TASK ST RET STORE
OWPRET  BSS   2              O/P TASK WP RET STORE
OPCRET  BSS   2              O/P TASK PC RET STORE
OSTRET  BSS   2              O/P TASK ST RET STORE
CWPRET  BSS   2              MCP TASK WP RET STORE
CPCRET  BSS   2              MCP TASK PC RET STORE
CSTRET  BSS   2              MCP TASK ST RET STORE
IDH     BSS   2              IPDHTP ACTIVE FLAG(0=TASK NOT ACTIVE)
MONI    BSS   2              MONITOR TASK ACTIVE FLAG(-1=TASK ACTIVE)
DIAG    BSS   2              DIAGNOSIS TASK ACTIVE FLAG
IPTASK  BSS   2              I/P TASK ACTIVE FLAG
OPTASK  BSS   2              O/P TASK ACTIVE FLAG
MCP     BSS   2              MCP TASK ACTIVE FLAG
MSTIMC  BSS   2              MON STIM COUNT
DSTIMC  BSS   2              DIAG ''     ''
CSTIMC  BSS   2              MOCP   ''     ''
OPAC    BSS   2              UART O/P ACTIVE FLAG
MSEC    BSS   2              4MS COUNTER
SEC     BSS   2              SECONDS COUNTER
MINUTE  BSS   2              MINUTES COUNTER
HOURS   BSS   2              HOURS   COUNTER
PPUIHW  BSS   32             UART WORKSPACE AREA
        END
```

GENERAL DATA MODULE G0017D

1. DESCRIPTION

This data module contains software flags which are to be initialised to +1 or -1 by the initialisation program (PPINIP) at system start up time.

2. IDENTIFICATION

GENERAL DATA MODULE G0017D

3. SIZE

CLASS TWO 36 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0017D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0017D'
       EVEN
* CLASS TWO
       DEF  MONINT,DIAINT,IPINT
       DEF  DFGO,SDELAY,SFP1,SU,SU2,SU3,CMS
       DEF  TOG,CLRWT
       DEF  DISF,MONSF
       DEF  ARTIF,OPINT,MCPINT,MCPIP
       DEF  SETONE,SETNEG,SETEND
       RORG
       PSEG
SETONE EQU  $              THE FOLLOWING FLAGS ARE INITIALISED TO 1
MONINT BSS  2              MONITOR INTERRUPTED FLAG
DIAINT BSS  2              DIAGNOSIS INTERRUPTED FLAG
IPINT  BSS  2              I/P TASK INTERRUPTED FLAG
OPINT  BSS  2              O/P TASK INTERRUPTED FLAG
MCPINT BSS  2              MCP TASK INTERRUPTED FLAG
TOG    BSS  2              TOGGLE FLAG
DISF   BSS  2              MCP DIAGNOSIS REQUEST FLAG
MONSF  BSS  2              MCP MONITOR REQUEST FLAG
MCPIP  BSS  2              MCP IPTASK  REQUEST FLAG
ARTIF  BSS  2              ARTIFACT FLAG
CMS    BSS  2              +1=CONTINUE MONITORING
*
SETNEG EQU  $              THE FOLLOWING FLAGS ARE INITIALISED TO -1
DFGO   BSS  2              -1=NO MONITORING
SDELAY BSS  2              -1=DELAY
SFP1   BSS  2              -1=SEARCH
SU     BSS  2              -1=START UP
SU2    BSS  2               ''
SU3    BSS  2               ''
CLRWT  BSS  2              CLEAR QRS WIDE & TOTAL FLAG
SETEND EQU  $              THIS IS END OF INITIALISED FLAG SECTION
       END
```

GENERAL DATA MODULE G0003D

1. DESCRIPTION

This data module contains the input buffer, which can store 3 seconds of unfiltered ECG signal.

2. IDENTIFICATION

GENERAL DATA MODULE G0003D

3. SIZE

CLASS THREE 750 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0003D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0003D'
       EVEN
* CLASS THREE
       DEF  IBUF
       RORG
       PSEG
IBUF   BSS  750              INPUT BUFFER FOR 3 SECONDS
*                                   OF SAVED ECG SIGNAL
       END
```

GENERAL DATA MODULE   G0004D

1. DESCRIPTION

This data module contains, the conditioned signal input buffer (IBP1), the delayed difference signal buffer (DBP1), and the various locations where QRS complex detection and measurement parameters are saved.

2. IDENTIFICATION

GENERAL DATA MODULE G0004D

3. SIZE

CLASS TWO   866 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
      TITL 'GENERAL DATA MODULE G0004D'
* GENERAL DATA MODULE IDENTIFIER
      IDT  'G0004D'
      EVEN
* CLASS TWO
      DEF  IBP1,DBP1,TDP1,NP,RRC,RRSAVE,WQRS,AVRR
      DEF  POINT
      RORG
      PSEG
IBP1   BSS  752            I/P BUFF
DBP1   BSS  100            DIFF BUFF
TDP1   BSS  2              DIFFERENCE THRESHOLD
NP     BSS  2              NUMBER OF POINTS BELOW THRESHOLD
RRC    BSS  2              RR COUNT
RRSAVE BSS  2              RR SAVE LOCATION
WQRS   BSS  2              QRS WIDTH
AVRR   BSS  2              AVERAGE RR
POINT  BSS  2              ADDRESS POINTER TO QRS
      END
```

GENERAL DATA MODULE G0005D

1. DESCRIPTION

This data module contains the monitoring result words MF1 and MF2, monitor workspace area (MONS1W), plus the storage locations of the various parameters calculated by the monitoring task.

2. IDENTIFICATION

GENERAL DATA MODULE G0005D

3. SIZE

CLASS TWO 62 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0005D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0005D'
       EVEN
* CLASS TWO
       DEF  MF1,MF2,MONS1W,MINV,MAXV
       DEF  MINP,MAXP,T1,T2,T3,W1,W2,DMINO,DMINC
       DEF  P2POS,OFSETO
       RORG
       PSEG
MF1    BSS  2                MONITOR FLAG REG
MF2    BSS  2                ''
MONS1W BSS  32               MONITOR WORKSPACE AREA
MINV   BSS  2                MIN VALUE
MAXV   BSS  2                MAX  ''
MINP   BSS  2                MIN POSITION
MAXP   BSS  2                MAX    ''
T1     BSS  2                TOTAL 1
T2     BSS  2                 ''
T3     BSS  2                 ''
W1     BSS  2                WIDE TOTAL 1
W2     BSS  2                ''    ''    2
DMINO  BSS  2                DMIN OLD
DMINC  BSS  2                DMIN CURRENT
P2POS  BSS  2                P2 POSITION
OFSETO BSS  2                OFF SET OLD
       END
```

GENERAL DATA MODULE G0006D

1. DESCRIPTION

This data module contains the diagnosis workspace area, and diagnosis result words DF1 and DF2.

2. IDENTIFICATION

GENERAL DATA MODULE G0006D

3. SIZE

CLASS TWO 36 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0006D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0006D'
       EVEN
* CLASS TWO
       DEF  DIAT1W,DF1,DF2
       RORG
       PSEG
DIAT1W BSS  32            DIAGNOSIS WORKSPACE AREA
DF1    BSS  2             DIAGNOSIS FLAG
DF2    BSS  2             //
       END
```

GENERAL DATA MODULE G0018D

1. DESCRIPTION

This data module contains the workspace areas and files used by the Input, Output and MCP tasks.

It should be noted that this must be the last data module in RAM as it contains the end of RAM label RAMEND, used by the initialisation program.

2. IDENTIFICATION

GENERAL DATA MODULE G0018D

3. SIZE

CLASS TWO 206 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
      TITL 'GENERAL DATA MODULE G0018D'
* GENERAL DATA MODULE IDENTIFIER
      IDT  'G0018D'
      EVEN
* CLASS TWO
      DEF  IPPTOW,IPFILE,OPPTOW,OPFILE,MCPTOW
      DEF  OPDATA,MOTEXT,WID82S,RAMEND
      RORG
      PSEG
IPPTOW BSS  32             INPUT TASK WORKSPACE AREA
IPFILE BSS  52             INPUT FILE
OPPTOW BSS  32             OUTPUT TASK WORKSPACE AREA
OPFILE BSS  52             OUTPUT FILE
MCPTOW BSS  32             MCP WORKSPACE AREA
OPDATA BSS  2              O/P DATA ADDRESS
MOTEXT BSS  2              MONITOR O/P MESSAGE ADDRESS
WID82S BSS  2              % QRS WIDE
RAMEND EQU  $              THIS SHOULD BE END OF RAM
      END
```

APPENDIX C

ALTERNATIVE PATIENT PROCESSOR PROGRAM LISTING

The programs supplied in this appendix are those required to produce the alternative patient processor software system. They are indicated by the '*' in the following,

PHASE 0, PATIENT ORIGIN = 0000 LENGTH = 1824

	MODULE	NO	ORIGIN	LENGTH
	PMPTVP	1	0000	00A4
	PPINIP	2	00A4	00A0
	PPTHAP	3	0144	00D6
	PPCIHP	4	021A	00BA
	PPUIHP	5	02D4	014C
	IPDHTP	6	0420	01B6
	MONTDP	7	05D6	018A
*	MONS3P	8	0760	024E
*	MONS4P	9	09AE	015C
*	DIAT2P	10	0B0A	00C4
	IPPTOP	11	0BCE	00B0
	OPPTOP	12	0C7E	0082
	MCPTOP	13	0D00	0130
	P0001D	14	0E30	0014
	P0002D	15	0E44	000C
	P0003D	16	0E50	002E
	P0009D	17	0E7E	0018
*	P0010D	18	0E96	006A
	IPDHTD	19	0F00	00C1
	MONTDD	20	0FC2	0020
	G0016D	21	0FE2	007A
	G0017D	22	105C	0024
	G0003D	23	1080	02EE
	G0004D	24	136E	0362
	G0005D	25	16D0	003E
*	G0019D	26	170E	0048
	G0018D)	27	1756	00CE

ALTERNATIVE PATIENT PROCESSOR PROGRAM LABELS

D E F I N I T I O N S

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
A1	0E34	14	A1DC	0E40	14	A2	0E36	14	A2DC	0E42	14
ACTNO	0FEC	21	ALARM1	0EEE	18	ALARM2	0EF0	18	ARTIF	106E	22
ASYST	0EE2	18	AVRR	16CC	24	AVRRB	0E62	16	*BEATS	0E70	16
BPM100	0E56	16	BPM110	0E54	16	*BPM120	0E52	16	*BPM140	0E50	16
BPM150	0EFA	18	*BPM20	0E60	16	BPM200	0EFC	18	BPM30	0EF8	18
BPM40	0E5E	16	BPM60	0E5C	16	BPM80	0E5A	16	BPM90	0E58	16
BROAD	0EF2	18	BUFEND	0FBE	19	BUFFER	0F84	19	BUFLEN	0E44	15
C	0E3C	14	*CLRWT	107E	22	CM	0E89	17	CMS	1070	22
CDC1	0E96	18	CDC2	0E9A	18	CDC3	0EA0	18	CDC4	0EA4	18
CDC5	0EAC	18	CDC6	0EAE	18	COUNT	0E6A	16	CPCRET	101C	21
CSNS	0EF6	18	CSS	0EF4	18	CSTIMC	1030	21	CSTRET	101E	21
CWPRET	101A	21	CZC1	0E98	18	CZC2	0E9C	18	CZC3	0E9E	18
CZC4	0EA2	18	CZC5	0EA6	18	CZC6	0EA8	18	CZC7	0EAA	18
D0T1	0ECA	18	D0T2	0ECC	18	D1T1	0ECE	18	D1T2	0ED0	18
D2T1	0ED2	18	D2T2	0ED4	18	D2T3	0ED6	18	D2T4	0ED8	18
D3T1	0EDA	18	D3T2	0EDC	18	D4T1	0EDE	18	D5T1	0EE0	18
DBP1	165E	24	DBUFL	0E46	15	DF1	172E	26	DF2	1730	26
DFGO	1072	22	DIAG	1024	21	DIAGM	0E80	17	DIAINT	105E	22
DIAT11	0B0A	10	DIAT2W	170E	26	DISF	1068	22	DMINC	1708	25
DMINO	1706	25	DPCRET	100A	21	DRTWP	0FEE	21	DSTIMC	102E	21
DSTRET	100C	21	DWPRET	1008	21	EIGHT	0EC8	18	EM	0E88	17
FINISH	0732	7	*FIVE	0E76	16	HOURS	103A	21	HPOP	0F80	19
HUND	0E72	16	IBP1	136E	24	IBUF	1080	23	IDH	1020	21
IPCRET	1010	21	IPDHT1	0420	6	IPDHTW	0F00	19	IPFILE	1776	27
IPINT	1060	22	IPPT01	0BCE	11	IPPT02	0C56	11	IPPT03	0C6C	11
IPPTOW	1756	27	IPRTWP	0FF4	21	IPTASK	1026	21	ISTRET	1012	21
IWPRET	100E	21	K1LP	0E38	14	K2LP	0E3A	14	KDC	0E3E	14
KHP	0E32	14	LEVEL	0E30	14	LPOP	0F82	19	MAXP	16FA	25
MAXV	16F6	25	MCP	102A	21	MCPINT	1064	22	MCPIP	106C	22
MCPT01	0D00	13	MCPTOW	17FE	27	MCRTWP	1000	21	MF1	16D0	25
MF2	16D2	25	MFAIL	0E85	17	MINL	0E48	15	MINP	16F8	25
MINUTE	1038	21	MINV	16F4	25	MM1	0E8B	17	MM2	0E8C	17
MM3	0E8D	17	MM4	0E8E	17	MM5	0E8F	17	MM6	0E90	17
MONI	1022	21	MONINT	105C	22	MONS11	0760	8	MONS1W	16D4	25
MONS41	09AE	9	MONSF	106A	22	MONTD1	05D6	7	MONTDW	0FC2	20
MONWR3	0FC8	20	MOTEXT	1820	27	MPCRET	1004	21	MRTWP	0FE8	21
MSEC	1034	21	MSTIMC	102C	21	MSTRET	1006	21	MSUOVE	0E84	17
MWPRET	1002	21	NOORS	0AF2	9	NP	16C4	24	NPBRAD	0EEA	18
NPTACH	0EEC	18	NQRSB	0EC6	18	OFSETO	170C	25	OPAC	1032	21
OPCRET	1016	21	OPDATA	181E	27	OPFILE	17CA	27	OPINT	1062	22
OPPT01	0C7E	12	OPPTOW	17AA	27	OPRTWP	0FFA	21	OPTASK	1028	21
OSTRET	1018	21	OWPRET	1014	21	P2POS	170A	25	PAUSE	0EE4	18
PBEAT	0EE6	18	*PER82	0E78	16	PERCEN	1752	26	PERFLA	1754	26
POINT	16CE	24	PPCIH1	021A	4	PPCIH2	025A	4	PPINI1	00A4	2
PPTHA1	0144	3	PPTHAW	0FE2	21	PPUIH1	02D4	5	PPUIH2	03B6	5
PPUIH3	03CC	5	PPUIH4	03E8	5	PPUIH5	0402	5	PPUIHW	103C	21
PR	0E86	17	PRM	0E7E	17	PS	0E8A	17	PSD	0E82	17
PTACH	0EE8	18	PUART	040C	5	QRSNT	0E7D	16	QRSPT	0E7C	16
QRSWB	0E6E	16	QRSWID	0E6C	16	RAM	0F00	19	RAM2	0F02	19
RAMEND	1824	27	RRBAVR	0788	8	RRBEND	1750	26	RRBUF	1732	26
RRC	16C6	24	RRC32	0E4A	15	RRLB	0E64	16	RRMAXT	0EFE	18
RRSAVE	16C8	24	RRSB	0E66	16	SDELAY	1074	22	SEC	1036	21
SEQ1	0EB0	18	SEQ10	0EC2	18	SEQ11	0EC4	18	SEQ2	0EB2	18
SEQ3	0EB4	18	SEQ4	0EB6	18	SEQ5	0EB8	18	SEQ6	0EBA	18
SEQ7	0EBC	18	SEQ8	0EBE	18	SEQ9	0EC0	18	SEQB	0E68	16
SETEND	1080	22	SETNEG	1072	22	SETONE	105C	22	SFP1	1076	22
SIX	0E4E	15	SIXTY	0E94	17	SM	0E87	17	SR250	0E92	17
SU	1078	22	SU2	107A	22	SU3	107C	22	T1	16FC	25
T2	16FE	25	T3	1700	25	TDP1	16C2	24	TEN	0E4C	15
*TOG	1066	22	*TWELVE	0E74	16	VALUE	0FC0	19	W1	1702	25
W2	1704	25	WADC	0F60	19	WAHP	0F20	19	WALP	0F40	19
*WID82S	1822	27	*WIDE82	0E7A	16	WQRS	16CA	24			

MONITOR SEARCH SECTION SYSTEM 2

1. DESCRIPTION

This program module is an alternative program to program module MONS1P. The differences between these two programs are :-

(1) The start-up procedure of this program calculates the average R-R interval from 16 stored R-R intervals.

(2) This program does not calculate the percentage of wide QRS complexes.

(3) Various parameters have been changed to meet the new monitoring requirements.

2. IDENTIFICATION

SUBROUTINE NAME	MONS3
PROGRAM MODULE	MONS3P
GENERAL DATA MODULES	G0003D
GENERAL DATA MODULES	G0004D
GENERAL DATA MODULES	G0005D
GENERAL DATA MODULES	G0018D
PERMANENT DATA MODULES	P0003D
PERMANENT DATA MODULES	P0009D

3. SIZE

PROGRAM MODULE 590 BYTES.

4. CALLS TO SUBROUTINE

```
B       @MONS11
BL      @RRBAVR
```

5. CALLS FROM SUBROUTINE

PROGRAM MODULE	MONTDP
B	@FINISH
PROGRAM MODULE	MONS4P
B	@MONS41

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data for this program is the output data from the program module MONTDP; this data being the current estimate of the R-R interval (RRC), and the position of the QRS complex (POINT) in the input buffer (IBP1).

6.2. OUTPUT DATA

The output from this program is the monitoring result word MF1, the individual bits of which are the True/False answers to the tests performed by the program. Also supplied by this program is the true R-R interval measurement, which is placed in RRSAVE.

6.3. ADDITIONAL DATA CHANGES

The software flags that are effected by this program are SFP1, SU2 and SU3.

7. TIMING

The combined execution time of the monitoring modules (MONTDP, MONS3P and MONS4P) does not exceed 3·2mS.

8. AUTHOR

B.T.V. WARTON.

SCR2.TASK.S.MONS3P 16:46:41 THURSDAY, OCT 19, 1978.

```
       TITL 'MONITOR SEARCH SECTION SYSTEM 2'
* PROGRAMME MODULE IDENT (TASK)
       IDT  'MONS3P'
* TRANSFER VECTORS OR ENTRIES
       DEF  MONS11,RRBAVR
* CALLED PROGRAM MODULES
       REF  MONTDP,MONS4P
       REF  FINISH,MONS41
* LINKED DATA MODULES
       REF  G0003D,G0004D,G0005D,G0018D,G0019D,P0003D,P0009D
       REF  SFP1,SU,DBP1
       REF  TDP1,BUFLEN,QRSPT,QRSNT
       REF  MINL,IBP1,POSIT,SU2,RRC
       REF  AVRR,MF1,MF2,BPM140,BPM110,BPM100,BPM90,BPM80
       REF  BPM60,BPM40,BPM20,BPM10,RRSAVE
       REF  AVRRB,RRSB,SEQB,RRLB,COUNT,MINV,MINP,QRSWID
       REF  QRSWB,TOG,CLRWT,BEATS,HUND,T1,W1,T2,W2,T3
       REF  PER82,WIDE82,WID82S,MAXV,MAXP,MONWR3
       REF  MONS1W,SU3,DMINC,DMINO,POINT
       REF  BPM120,TWELVE,FIVE,WQRS
       REF  P2POS,OFSETO,RRBUF,RRBEND,RR6SEC,FIFTEE
       REF  BPM30,BPM150,BPM200,PERFLA,PERCEN
       REF  MSUOVE,MOTEXT,CSTIMC,MCP
       RORG
       PSEG
************************** PROGRAMME **************************
*                             REG ASSIGNMENT
*      R0=SEQL    R1=SEQS       R2=SWQRS  R8=MF1&MF2
******        SEARCH SECTION               ****************
MONS11 LWPI MONS1W
       SETO @SFP1                 RESET SEARCH FLAG
       MOV  @RRC,@RRSAVE          SAVE RR COUNT
       CLR  @RRC                  RESET RRC
       MOV  @SU2,R9               TEST SU2 FLAG
       JGT  SU2NO                 JUMP IF SU2=NO
* 16 BEATS START UP SECTION
       ABS  @SU3                  TEST SU3
       JGT  SU3NO                 JUMP IF SU3 NO
       CLR  R10
       LI   R15,RRBUF             GET RRBUF ADDRESS
BFIN   B    @FINISH               GO TO FINISH
*
*******          R-R AVERAGE SUB
RRBAVR LI   R9,RRBUF              GET RR BUFFER ADDRESS
       CLR  R10                   SUM REG
ADDRR  A    *R9+,R10              GET RR IN BUF ADD TO SUM
       CI   R9,RRBEND             IS IT THE END
       JLE  ADDRR                 JUMP IF NO
       SRA  R10,4                 SUM R-R/16
       MOV  R10,@AVRR             SAVE AVERAGE AT AVRR
       RT
*
****** START UP SECTION 3
SU3NO  MOV  @RRSAVE,*R15+         GET RR INTO BUF
       A    @DMINC,R10            SUM OF DMIN'S
       CI   R15,RRBEND            IS IT END OF BUFFER
       JLE  BFIN                  JUMP IF NO
       LI   R15,RRBUF             RESET R15 TO RRBUF START
       SRA  R10,4                 SUM DMIN/16
       MOV  R10,@DMINO            SAVE DMIN
```

```
        BL    @RRBAVR               GO CALCULAT AVERAGE RR
        LI    R9,MSUOVE             GET ADDRESS OF SU OVE MESSAGE
        MOV   R9,@MOTEXT            SET O/P ADDRESS
        INC   @CSTIMC               STIM MCP
        SETO  @MCP                  REQUEST MCP
        ABS   @SU2                  RESET START UP FLAG
*
****    SETTING PULSE RATE FLAGS
SU2NO   CLR   R8                    MF1=R8
        C     @AVRR,@BPM80          IS PULSE >80 BPM
        JLE   GT80                  JUMP IF YES
        C     @AVRR,@BPM60          IS PULSE >60 BPM
        JGT   LE60                  JUMP IF NO
        AI    R8,>00E0              SET FLAGS >60
        JMP   SD1
LE60    C     @AVRR,@BPM40          IS PULSE >40
        JGT   LE40                  JUMP IF NO
        AI    R8,>00C0              SET FLAGS >40
        JMP   SD1
LE40    EQU   $
        C     @AVRR,@BPM30          IS PULSE >30
        JGT   LE20                  JUMP IF NO
        AI    R8,>0080              SET FLAGS >20
        JMP   SD1
LE20    AI    R8,>0000              SET FLAG <20
        JMP   SD1
GT80    C     @AVRR,@BPM90          IS PULSE >90
        JLE   GT90                  JUMP IF YES
        AI    R8,>00F0              SET FLAGS >80
        JMP   SD1
GT90    C     @AVRR,@BPM100         IS PULSE >100
        JLE   GT100
        AI    R8,>00F8              SET FLAGS >90
        JMP   SD1
GT100   EQU   $
        C     @AVRR,@BPM110         IS PULSE >110
        JLE   GT110
        AI    R8,>00FC              SET FLAGS>100
        JMP   SD1
GT110   C     @AVRR,@BPM150         IS PULSE >150
        JLE   GT150
        AI    R8,>00FE              SET FLAGS >110
        JMP   SD1
GT150   C     @AVRR,@BPM200
        JLE   GT200
        AI    R8,>00FF              SET FLAGS >140
        JMP   SD1
GT200   AI    R8,>80FF
SD1     EQU   $
*
*******         QRS WIDTH SECTION       ************************
WIDTH   EQU   $
        SLA   R2,1                  SHIFT SEQ WIDE REG
        MOV   @POINT,R4             GET POINTER TO QRS POS
        CLR   R5                    MIN REG
        CLR   R9                    MAX
        S     @COUNT,R4             POINT-COUNT
        JGT   INBUF
        JEQ   INBUF
        A     @BUFLEN,R4            ADD BUFFER LENGH TO R4
```

SCR2.TASK.S.MONS3P 16:46:41 THURSDAY, OCT 19, 1978.

```
INBUF  MOVB @IBP1(R4),R5      GET VALUE IN QRS
       MOV  R4,R6             SAVE POS
                              MAX
       CLR  R9                PUT VALUE IN R9
       MOVB R5,R9             MAX POS
       MOV  R6,R10            COUNTER
       CLR  R7                INC POS POINT
QU1    INC  R4                IS IT STILL INSIDE BUFFER
       C R4,@BUFLEN           JUMP IF YES
       JLE  QD3               RESET
       CLR  R4                IS MIN<THIS VALUE
QD3    CB   R5,@IBP1(R4)      JUMP IF YES
       JLT  QD1               SAVE NEW MIN
       MOVB @IBP1(R4),R5      SAVE NEW POS
       MOV  R4,R6
       JMP  Q1                IS VALUE IN R9 > IN BUFFER
QD1    CB   R9,@IBP1(R4)
       JGT  Q1                PUT VALUE IN BUFFER IN R9
       MOVB @IBP1(R4),R9      SAVE POSITION
       MOV  R4,R10            COMPARE WITH CURRENT POS
Q1     C    R4,@MONWR3
       JNE  QU1
**** AT THIS POINT R5=MIN @ R6=POS& R9=MAX@R10=POS
QD3A   SRA  R5,8              MAKE BYTE WORD
       MOV  R5,@MINV          SAVE MIN
       MOV  R6,@MINP          SAVE POS
       SRA  R9,8              MAKE BYTE WORD
       MOV  R9,@MAXV          SAVE MAKE VALUE
       MOV  R10,@MAXP         SAVE MAKE POSITION
       CLR  R3                WIDTH REG
QU3    INC  R3                WIDTH COUNT+1
       C    R3,@QRSWID        IS R3 > QRS WIDE WIDTH
       JGT  WIDE
       INC  R6                INC POS
       C    R6,@BUFLEN        IS R6> BUFFER LENGTH
       JLE  QD5
       CLR  R6                RESET POS
QD5    C    R6,@P2POS         IS R6=CURRENT P2 POSITION
       JEQ  QD5A
       CB   @IBP1(R6),@QRSNT  TEST IBP1 VALUE FOR SIGN CHANGE
       JLT  QU3               JUMP IF STILL -VE
QD5A   MOV  @MINP,R6          GET MIN POS
QU4    INC  R3                WIDTH+1
       C    R3,@QRSWID        IS R3 >QRS WIDE WIDTH
       JGT  WIDE
       DEC  R6                DEC POS. BACKWARD SEARCH
       JGT  QD6
       JEQ  QD6
       MOV  @BUFLEN,R6        SET R6 TO BUFFER LENGTH
QD6    MOVB @IBP1(R6),R9      TEST IBP1 FOR SIGN CHANGE
       JLT  QU4               JUMP IF STILL -VE
QU5    INC  R3                WIDTH+1
       C    R3,@QRSWID        IS R3 >QRS WIDE WIDTH
       JGT  WIDE
       DEC  R6                BACKWARD SEARCH
       JGT  QD7
       JEQ  QD7
       MOV  @BUFLEN,R6        SET POS TO BUFLEN
QD7    CB   @IBP1(R6),@QRSPT  TEST IBP1 FOR CHANGE IN SIGN
       JGT  QU5               JUMP IF STILL +VE
*      AT THIS POINT R3=WIDTH OF QRS
```

```
QU6A   MOV  R3,@WQRS
       C    R3,@QRSWID         IS WIDTH NORMAL
       JLE  QRSN               JUMP IF NORMAL
WIDE   SOC  @QRSWB,R8          SET BIT WIDTH=1
       SOC  @SEQB,R2           SET SEQ BIT WIDE
QRSN   EQU  $
*
*******     TRUE R-R SECTION
CALRR  MOV  @P2POS,R13         GET P2 CROSS OVER POINT
       S    @MINP,R13          P2POS-MINP
       JGT  DIFOK
       JEQ  DIFOK
       A    @BUFLEN,R13        ANS SHOULD BE +
DIFOK  MOV  @RRSAVE,R14        GET RRSAVE
       A    @OFSETO,R14        RRSAVE+OFSETO
       S    R13,R14            R14=RRC+OFSETO-OFSETN=TRR
       JLT  RRERRO
       MOV  R14,@RRSAVE        REPLACE RRSAVE WITH TRUE RR
       MOV  R13@OFSETO         REPLACE OFSETO
RRERRO EQU  $
*
******* TEST FOR CURRENT R-R <600MS
       C    @RRSAVE,@BPM100    IS TRUE RR <600MS
       JGT  LT82
       ORI  R8,>100            SET <600MS BIT
*
*******    START OF 2 X AVRR SECTION   ************
LT82   MOV  @AVRR,R9           GET AVRR
       SLA  R9,1               AVRR X 2
       C    @RRSAVE,R9         IS RRSAVE>=2 X AVRR
       JLT  RRLT               JUMP IF RRSAVE <
       SOC  @AVRRB,R8          SET AVRR BIT=1
*
*******    START OF RR = LONG,SHORT OR NORMAL SECTION ******
RRLT   SLA  R0,1               SHIFT SEQ LONG REG
       SLA  R1,1                 //    // SHORT REG
       MOV  @PERFLA,@PERFLA    IS PERCENTAGE CHANGED
       JNE  PERSET
       LI   R10,15             15 PER
       MOV  R10,@PERCEN        SAVE %
       MOV  @AVRR,R9           GET AVRR
PERSET MPY  @PERCEN,R9         % X AVRR
       DIV  @HUND,R9           AVRR X %/100=R9
       MOV  @AVRR,R10          GET AVRR
       A    R9,R10             AVRR + IER=RR TOLERANCE
       C    @RRSAVE,R10        IS RR TOLERANCE>RRSAVE
       JGT  RRLONG
       S    R9,R10             R9=AVRR
       S    R9,R10             AVRR - IER=RR TOLERANCE
       C    @RRSAVE,R10        IS RRSAVE < RR TOLERANCE
       JHE  SAVEMF             JUMP IF NOT (IE NORMAL RR)
       SOC  @RRSB,R8           SET SHORT BIT=1
       SOC  @SEQB,R1           SET SEQ SHORT BIT=1
       JMP  SAVEMF
RRLONG SOC  @RRLB,R8           SET LONG BIT=1
       SOC  @SEQB,R0           SET SEQ LONG BIT=1
*
*******     SAVE MF1         *************
SAVEMF EQU  $
       MOV  R8,@MF1            SAVE MF1

       B    @MONS41            GO TO NEXT MONITORING MODULE
       END
```

MONITOR SEQUENCE SEARCH SECTION

SYSTEM 2

1. DESCRIPTION

This program is concerned with testing different sequences of R-R intervals and QRS complex widths, and placing the results in the monitoring word MF2.

At the end of this program is the updating procedures for the parameters DMIN, TDP1 and AVRR, and also the procedure for alarming the condition of "no QRS detected".

The differences between this program module, and program module MONS2P is the sequence patterns which are being monitored for, and the procedure for updating AVRR. (The sequence detected by MONS2P are retained in this module, but not used at present by the Diagnosis task (DIAT2P).

2. IDENTIFICATION

SUBROUTINE NAME	MONS4
PROGRAM MODULE	MONS4P
GENERAL DATA MODULE	G0003D
GENERAL DATA MODULE	G0004D
GENERAL DATA MODULE	G0005D
PERMANENT DATA MODULE	P0002D
PERMANENT DATA MODULE	P0010D

3. SIZE

PROGRAM MODULE 348 Bytes.

4. CALLS TO SUBROUTINE

```
B        @MONS41
B        @NOQRS
```

5. CALLS FROM SUBROUTINE

```
PROGRAM MODULE     MONTDP
    B              @FINISH
PROGRAM MODULE     MONS3P
    BL             @RRBAVR
```

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data for this program is supplied in registers R0, R1 and R2 of workspace area MONS1W. The contents of these registers are :-

(i) R0 = Long R-R interval sequence bits.

(ii) R1 = Short R-R interval sequence bits.

(iii) R2 = Wide QRS complex sequence bits.

6.2. OUTPUT DATA

The output from this program is the Monitoring result word MF2, the individual bits of which indicate which sequence of R-R intervals and QRS widths have been detected.

7. TIMING

The combined execution time of the monitoring modules (MONTDP, MONS3P and MONS4P) does not exceed 3·2mS.

8. AUTHOR

B.T.V. WARTON.

SCR2.TASK.S.MONS4P 17:02:05 THURSDAY, OCT 19, 1978.

```
       TITL 'MONITOR SEQUENCE SEARCH SECTION SYSTEM 2'
* PROGRAMME MODULE IDENT
       IDT  'MONS4P'
* TRANSFER VECTORS OR ENTRIES
       DEF  MONS41,NQQRS
* CALLED PROGRAM MODULES
       REF  MONTDP,MONS3P
       REF  FINISH,RRBAVR
* LINKED DATA MODULES
       REF  G0003D,G0004D,G0005D,P0002D,P0004D
       REF  COC1,CZC1,COC2,CZC2,CZC3,COC3,CZC4,COC4,CZC5
       REF  CZC6,CZC7,COC5,COC6
       REF  SEQ1,SEQ2,SEQ3,SEQ4,SEQ5,SEQ6,SEQ7,SEQ8,SEQ9
       REF  MF2,DIAG,DMINC,DMINO,DSTIMC,EIGHT
       REF  MF1,NQRSB,QRSWB,SIX,TDP1,TEN
       REF  RRC,AVRR,RRSB,RRLB,RRSAVE
       REF  RRBUF,RRBEND,CSNS,CSS,SEQ10,SEQ11
       RORG
       PSEG
*****************     PROGRAMME    **********************
*      THIS NEXT SECTION SETS SEQUENCE FLAG BITS IN MF2
*                   N O T E
*      R0 = LONG SEQUENCE BIT FLAGS
*      R1 = SHORT  ''        ''    ''
*      R2 = WQRS   ''        ''    ''
*********************************************************
*
MONS41 CLR  R8                    RESET MF1 REG
*
*******  SEQUENCE  S - S
       COC  @CSS,R1               COMPARE FOR S-S
       JNE  SFAILA
       MOV  @SEQ11,R8             SET SEQUENSE BIT FOUND
       JMP  SFAIL
*
*******  SEQUENCE S - NS
SFAILA COC  @CSNS,R1              COMPARE FOR S-NS
       JNE  SFAIL
       MOV  @SEQ10,R8             SET SEQUENCE BIT FOUND
*
*******  SEQUENCES  4(S-NS) & 4(S(WQRS)-NS)
SFAIL  COC  @COC1,R1              TEST SHORT SEQ S-S-S-S-
       JNE  SFAIL1
       CZC  @CZC1,R1              TEST NOT SHORT -NS-NS-NS-NS
       JNE  RETMF2
       COC  @COC1,R2              TEST WQRS W-W-W-W-
       JNE  FOUND1
       CZC  @CZC1,R2              TEST NOT WQRS -NW-NW-NW-NW
       JNE  FOUND1
       MOV  @SEQ2,R8              4(S(W)-NS) FLAG SET IN R8=MF2
       JMP  RETMF2
FOUND1 MOV  @SEQ1,R8              4(S-NS) FLAG SET IN R8=MF2
       JMP  RETMF2
*
*******   SEQUENCES  4(S-NS-N) & 4(S(WQRS)-NS-N)
SFAIL1 COC  @COC2,R1              TEST SHORT S--S--S--S--
       JNE  SFAIL2
       CZC  @CZC2,R1              TEST NOT SHORT -NSNS-NSNS-NSNS-NSN
       JNE  SFAIL2
       CZC  @CZC3,R0              TEST NOT LONG --NL--NL--NL--NL
```

```
        JNE   SFAIL2
        COC   @COC2,R2          TEST WQRS W--W--W--W--
        JNE   FOUND2
        CZC   @CZC2,R2          TEST NOT WQRS -NWNW-NWNW-NWNW-NWN
        JNE   FOUND2
        MOV   @SEQ4,R8          SET 4(S(W)-NS-N) FLAG BIT=1
        JMP   RETMF2
FOUND2  MOV   @SEQ3,R8          SET 4(S-NS-N) FLAG BIT=1
        JMP   RETMF2
*
*******     SEQUENCES   S-NS-S  &  S-S-S(WQRS)
SFAIL2  COC   @COC3,R1          TEST SHORT S-S
        JNE   SFAIL3
        CZC   @CZC4,R1          TEST NOT SHORT -NS-
        JNE   FAIL6
        MOV   @SEQ5,R8          SET S-NS-S FLAG BIT =1
        JMP   RETMF2
FAIL6   COC   @COC5,R2          TEST WQRS --W
        JNE   SFAIL3
        CZC   @CZC5,R2          TEST NOT WQRS NWNW-
        JNE   SFAIL3
        MOV   @SEQ7,R8          SET S-S-S(W) FLAG BIT=1
        JMP   RETMF2
*
*******     SEQUENCES  S-NS-N-NL  &  S-NS-N-L
SFAIL3  COC   @COC4,R1          TEST SHORT S---
        JNE   SFAIL4
        CZC   @CZC5,R1          TEST NOT SHORT -NSNS-
        JNE   SFAIL4
        CZC   @CZC6,R0          TEST NOT LONG --NLNL
        JNE   SFAIL5
        MOV   @SEQ6,R8          SET S-NS-N-NL FLAG=1
        JMP   RETMF2
SFAIL5  CZC   @CZC7,R0          TEST NOT LONG --NL-
        JNE   SFAIL4
        COC   @COC5,R0          TEST LONG ---L
        JNE   SFAIL4
        MOV   @SEQ9,R8          SET S-NS-N-L FLAG BIT=1
        JMP   RETMF2
*
*******     SEQUENCE   S-NS-L
SFAIL4  COC   @COC6,R1          TEST SHORT S--
        JNE   RETMF2
        CZC   @CZC7,R1          TEST NOT SHORT -NS-
        JNE   RETMF2
        COC   @COC5,R0          TEST LONG --L
        JNE   RETMF2
        MOV   @SEQ8,R8          SET S-NS-L FLAG BIT=1
RETMF2  MOV   R8,@MF2           SAVE MF2
*
*******       FINDING NEW DMIN & TD IF NEEDED
        MOV   @MF1,R9           GET MF1
        COC   @QRSWB,R9         TEST WQRSIN MF1
        JEQ   AVRUD             NO UP DATE
        MOV   @DMINO,R9         GET DMIN OLD
        ABS   R9                MAKE R9 +VE
        JLT   PD1
        JEQ   PD1
        BL    @DMINER           GO TO DMIN ERROR
PD1     MPY   @EIGHT,R9         DMINO X 8=R9&R10
```

SCR2.TASK.S.MONS4P 17:02:05 THURSDAY, OCT 19, 1978.

```
        DIV   @TEN,R9                R9&R10 / 10=DMINX0.8=R9
        MOV   R9,R8                  R8=DMINO X 0.8
        MOV   @DMINC,R10             GET DMIN CURRENT
        ABS   R10                    MAKE R10 +VE
        JLT   PD2
        JEQ   PD2
        BL    @DMINER                GO TO DMIN ERROR
PD2     SLA   R10,1                  DMIN CURRENT X 2
        CLR   R9
        DIV   @TEN,R9                R9=DMIN CURRENT X0.2
        A     R9,R8                  R8=NEW DMINO
        MOV   R8,R9
        MPY   @SIX,R9                R9 X 6
        DIV   @TEN,R9                R9= +TD
        NEG   R9                     R9= -TD
        MOV   R9,@TDP1               LOAD NEW TD
        NEG   R8                     MAKE R8 -VE
        MOV   R8,@DMINO              LOAD NEW DMINO
*
*******      AVRR UPDATE SECTION
AVRUD   MOV   @MF1,R9                GET MF1
        COC   @RRSB,R9               WAS RR SHORT
        JEQ   NUD
        COC   @RRLB,R9               WAS IT LONG
        JEQ   NUD
        MOV   @RRSAVE,*R15+          SAVE RRSAVE IN RR BUFFER
        CI    R15,RRBEND             IS IT THE END OF BUFFER
        JLE   GOSUM
        LI    R15,RRBUF              RESET TO START OF BUFF
GOSUM   BL    @RRBAVR                GO TO AVERAGE RR SUB
NUD     JMP   STIM
DMINER RT
*       INSERT HERE ERROR SECTION FOR +DMIN'S
*
*******      NO QRS  SECTION
NOQRS   MOV   @MF1,R9                GET MF1
        SOC   @NQRSB,R9              SET NOQRS BIT=1
        MOV   R9,@MF1                RET MF1
STIM    INC   @DSTIMC                INC DIAG STIM COUNT
        SETO @DIAG                   SET DIAG F L A G
        B     @FINISH                GO TO FINISH
        END
```

DIAGNOSIS PROGRAM SYSTEM 2

1. DESCRIPTION

This program is concerned with using an alternative approach to diagnose arrhythmias, using the results supplied by the monitoring modules MONS3P and MONS4P.

2. IDENTIFICATION

SUBROUTINE NAME	DIAT2
PROGRAM MODULE	DIAT2P
GENERAL DATA MODULE	G0002D
GENERAL DATA MODULE	G0005D
GENERAL DATA MODULE	G0006D
GENERAL DATA MODULE	G0019D
PERMANENT DATA MODULE	P0005D
PERMANENT DATA MODULE	P0010D

3. SIZE

PROGRAM MODULE 196 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP    @DIAT11
```

5. INTERNAL DATA TRANSFERS

5.1. INPUT DATA

The input to this program are the monitoring words MF1 and MF2.

5.2. OUTPUT DATA

The output data from this program is placed in locations DF1 and DF2.

5.3. ADDITIONAL DATA CHANGES

The flags that may be effected by the execution of this program are :-

DISF, MCP, CSTIMC, DSTIMC and DIAG.

6. TIMING

The execution time of this program will not exceed 0·2mS.

7. AUTHOR

B.T.V. WARTON.

```
      TITL 'DIAGNOSIS PROGRAMME SYSTEM 2'
* PROGRAMME MODULE IDENT (TASK)
      IDT  'DIAT2P'
* TRANSFER VECTORS OR ENTRIES
      DEF  DIAT11
* LINKED DATA MODULES
      REF  G0002D,G0005D,G0006D,G0019D,P0005D,P0010D
      REF  DIAT2W,MF1,MF2,DSTIMC,DIAG,DF1,DF2
      REF  DISF,CSTIMC,MCP
      REF  D0T1,D0T2,D1T1,D1T2,D1T3,D2T1,D2T2,D2T3,D2T4
      REF  D3T1,D3T2,D4T1,D5T1
      REF  ASYST,PAUSE,PBEAT,PTACH,NPBRAD,NPTACH
      REF  ALARM1,ALARM2,BROAD
*
************* P R O G R A M M E ****************
DIAT11 DATA DIAT2W,SDIAGN
SDIAGN MOV  @MF1,R1            GET MF1
       MOV  @MF2,R2            GET MF2
       CLR  R3                 DF1
       CLR  R4                 DF2
*
******* ASYSTOLE *******************
       COC  @D5T1,R1           COMPARE FOR TRUE CONDITIONS
       JNE  TD1
       SOC  @ASYST,R3          SET ASYST DIAG BIT IN DF1
       SOC  @ALARM1,R3         SET ALARM BIT
       JMP  TD4
*
*******   PAROXYSMAL TACHYCARDIA **********************
TD1    COC  @D2T1,R1           <600MS
       JNE  TD2
       COC  @D2T2,R2           S-S
       JNE  TD2
       SOC  @PTACH,R3          SET PTACH DIAG BIT IN DF1
       COC  @D2T3,R1           WIDE
       JNE  TD1NW
       SOC  @BROAD,R3          SET QRS BROAD BIT
TD1NW  CZC  @D2T4,R1           COMPARE FOR FALSE CONDITIONS
       JNE  SETA2
       SOC  @ALARM1,R3         SET ALARM BIT
       JMP  TD4
SETA2  SOC  @ALARM2,R3         SET ALARM BIT
       JMP  TD4
*
*******    PREMATURE BEAT ***************************
TD2    COC  @D3T1,R2           S-NS
       JNE  TD3
       SOC  @PBEAT,R3          SET PBEAT DIAG BIT
       COC  @D3T2,R1           WIDE
       JNE  TD3
       SOC  @BROAD,R3          SET BROAD QRS BIT
       JMP  TD4
*
*******    PAUSE ***************************
TD3    COC  @D4T1,R1           VERY LONG ?
       JNE  TD4
       SOC  @PAUSE,R3          SET PAUSE DIAG BIT
*
***************  SECONDARY TESTS ****************
*******   NON PAROXYSMAL TACHYCARDIA ******************
TD4    COC  @D0T1,R1           COMP TRUE CONDITIONS
       JNE  TD5
```

```
        SOC  @NPTACH,R3         SET NPTACH DIAG BIT
        CZC  @D0T2,R1           COMP FALSE COND
        JEQ  TD5
        SOC  @ALARM2,R3         SET ALARM BIT
        JMP  DIAGOP
*
********      NON PAROXYSMAL BRADYCARDIA ****************
TD5     COC  @D1T1,R1           COMP TRUE COND
        JNE  TD5A
        CZC  @D1T2,R1           COMP FALSE COND
        JNE  DIAGOP
TD5B    SOC  @NPBRAD,R3         SET NPBRAD DIAG BIT
        JMP  DIAGOP
TD5A    SOC  @ALARM2,R3         SET ALARM BIT
        JMP  TD5B
*
********       DIAGNOSIS O/P ********************
DIAGOP MOV   @DISF,R0           TEST DISF
        JLT  EXITA
        MOV  R3,@DF1            SAVE DF1
        MOV  R4,@DF2            SAVE DF2
        A    R3,R4              ARE DF1&DF2=0
        JEQ  EXITA
        SETO @DISF              SET DISF
        INC  @CSTIMC            STIM MCP
        SETO @MCP               STIM MCP
EXITA   DEC  @DSTIMC            DEC DIAG STIM COUNT
        JNE  EXIT
        CLR  @DIAG              CLR DIAG STIM
EXIT    RTWP                    RET TO TASK HANDLER
        END
```

PERMANENT DATA MODULE P0010D

1. DESCRIPTION

This data module contains the comparison words used by the monitoring module MONS4P, to detect various sequences of R-R interval and QRS complex widths. Also included in this module are the comparison words used by the diagnosis program DIAT2P, along with the words used when setting the different diagnosis bits etc. in DF1.

2. IDENTIFICATION

PERMANENT DATA MODULE P0010D

3. SIZE

106 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0010D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0010D'
       EVEN
       DEF  CDC1,CDC2,CDC3,CDC4,CDC5,CDC6
       DEF  CZC1,CZC2,CZC3,CZC4,CZC5,CZC6,CZC7
       DEF  SEQ1,SEQ2,SEQ3,SEQ4,SEQ5,SEQ6,SEQ7
       DEF  SEQ8,SEQ9,SEQ10,SEQ11
       DEF  NQRSB,EIGHT
       DEF  D0T1,D0T2,D1T1,D1T2
       DEF  D2T1,D2T2,D2T3,D2T4,D3T1,D3T2
       DEF  D4T1,D5T1
       DEF  ASYST,PAUSE,PBEAT,PTACH,NPBRAD,NPTACH
       DEF  ALARM1,ALARM2,BROAD
       DEF  BPM30,BPM150,BPM200,RRMAXT,CSS,CSNS
       RORG
       PSEG
CDC1   DATA >00AA          4(S-NS) & 4(S(W)-NS)
CZC1   DATA >0055          ''
CDC2   DATA >0924          4(S-NS-N) & 4(S(W)-NS-N)
CZC2   DATA >06DB          ''
CZC3   DATA >0249          ''
CDC3   DATA >0005          S-NS-S
CZC4   DATA >2             ''
CDC4   DATA >8             S-NS-N-NL
CZC5   DATA >6             ''
CZC6   DATA >3             ''
CZC7   DATA >2             S-NS-N-L
CDC5   DATA >1             ''
CDC6   DATA >4             S-NS-L
SEQ1   DATA >1             4(S-NS)
SEQ2   DATA >2             4(S(W)-NS)
SEQ3   DATA >4             4(S-NS-N)
SEQ4   DATA >8             4(S(W)-NS-N)
SEQ5   DATA >10            S-NS-S
SEQ6   DATA >20            S-NS-N-NL
SEQ7   DATA >40            3S-W
SEQ8   DATA >80            S-NS-L
SEQ9   DATA >100           S-NS-N-L
SEQ10  DATA >200           S-NS
SEQ11  DATA >400           S-S
NQRSB  DATA >0200          NO QRS BIT
EIGHT  DATA 8              CONSTANT
D0T1   DATA >00FC          DIAGNOSIS 0 TEST 1
D0T2   DATA >0001          DIAGNOSIS 0 TEST 2
D1T1   DATA >0080          DIAGNOSIS 1 TEST 1
D1T2   DATA >007F          DIAGNOSIS 1 TEST 2
D2T1   DATA >0100          DIAGNOSIS 2 TEST 1
D2T2   DATA >0400          DIAGNOSIS 2 TEST 2
D2T3   DATA >2000          DIAGNOSIS 2 TEST 3
D2T4   DATA >8000          DIAGNOSIS 2 TEST 4
D3T1   DATA >0200          DIAGNOSIS 3 TEST 1
D3T2   DATA >2000          DIAGNOSIS 3 TEST 2
D4T1   DATA >0400          DIAGNOSIS 4 TEST 1
D5T1   DATA >0200          DIAGNOSIS 5 TEST 1
ASYST  DATA >0020          ASYSTOLE
PAUSE  DATA >0010          PAUSE
PBEAT  DATA >0008          PREMATURE BEAT
PTACH  DATA >0004          PAROXYSMAL TACHYCARDIA
NPBRAD DATA >0002          NONE PAROXYSMAL BRADYCARDIA
NPTACH DATA >0001          NONE PAROXYSMAL TACHYCARDIA
ALARM1 DATA >8000          ALARM LEVEL 1 BIT
```

```
ALARM2 DATA >4000        ALARM LEVEL 2 BIT
BROAD  DATA >2000        BROAD QRS BIT
CSS    DATA 3            S-S
CSNS   DATA 2            S-NS
BPM30  DATA 500          30 BEATS / MIN
BPM150 DATA 100          150 BEATS / MIN
BPM200 DATA 75           200 BEATS / MIN
RRMAXT DATA 1250         RR INTERVAL MAX
       END
```

GENERAL DATA MODULE G0019D

1. DESCRIPTION

This data module contains the workspace area (DIAT2W) used by the diagnosis task, along with the locations where the diagnosis results words DF1 and DF2 are stored.

2. IDENTIFICATION

GENERAL DATA MODULE G0019D

3. SIZE

72 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0019D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0019D'
       EVEN
* CLASS TWO
       DEF  DIAT2W,DF1,DF2
       DEF  RRBUF,RRBEND,PERCEN,PERFLA
DIAT2W BSS  32             DIAGNOSIS TASK WA
DF1    BSS  2              DIAGNOSIS FILE WORD 1
DF2    BSS  2              DIAGNOSIS FILE WORD 2
RRBUF  BSS  30             RR BUFFER
RRBEND BSS  2              RR BUFFER END
PERCEN BSS  2              PERCENTAGE
PERFLA BSS  2              PERCENTAGE CHANGE FLAG
       END
```

APPENDIX D

OPERATOR/NURSE PROCESSOR PROGRAM LISTING

All the programs and data modules required to produce the operator/nurse processor software system, are supplied in this appendix, in the order in which they appear in the link editor listing shown below.

PHASE 0, NURSE   ORIGIN = 0000 LENGTH = 1034

MODULE	NO	ORIGIN	LENGTH
NSPTVP	1	0000	00A4
NSPIOP	2	00A4	009E
NSTHAP	3	0142	00BC
NIDQHP	4	01FE	00A2
NSUIHP	5	02A0	0202
NSPITP	6	04A2	0112
NSPOTP	7	05B4	011C
MOCPTP	8	06D0	0246
BIHEXP	9	0916	0040
ASCIIP	10	0956	0064
VDUIOP	11	09BA	00F8
P0007D	12	0AB2	000A
P0008D	13	0ABC	01B8
G0009D	14	0C74	004A
G0010D	15	0CBE	000E
G0011D	16	0CCC	0048
G0012D	17	0D14	01F8
G0013D	18	0F0C	0028
G0014D	19	0F34	003F
G0015D	20	0F74	00C0

OPERATOR/NURSE PROCESSOR PROGRAM LABELS

D E F I N I T I O N S

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
ASCDAT	0AD4	13	ASCII1	0956	10	ASCNUM	0AD0	13	BIHEX1	0916	9
BUFA	0FDC	20	CL	0ABA	12	CM	0AC0	13	CP	0ABB	12
CR	0AB9	12	CR1712	0AB2	12	EM	0ABE	13	EOB	0AB3	12
ERROR1	0AEF	13	FREE	1032	20	HEXONE	0AC6	13	HUND	0ACC	13
IHWA1	0D14	17	IHWA2	0DBC	17	IHWA3	0E64	17	INPUT	0A50	11
IOTF	1030	20	IOWA1	0D34	17	IOWA2	0DDC	17	IOWA3	0E84	17
IPFI1S	0D8A	17	IPFI2S	0E32	17	IPFI3S	0EDA	17	IPFIL1	0D88	17
IPFIL2	0E30	17	IPFIL3	0ED8	17	IPINT	0CC6	15	IPLIST	0AE4	13
IPPORT	0CAE	14	IPOS	0CEC	16	IPRTWP	0C92	14	IPSF	0CBC	14
IPSTIM	0CB2	14	IPSTRT	0CB0	14	IPTINT	0CBE	15	IPWPRT	0CAC	14
ITPORT	0C96	14	ITRTWP	0C7A	14	ITSTRT	0C98	14	ITWPRT	0C94	14
LINE	0AB6	12	MCPIP	0CBA	14	MINUS	0AD2	13	MOCINT	0CC2	15
MOCP	0CB6	14	MOCPSC	0CB8	14	MOCPT1	06D0	8	MOCPTW	0F34	19
MTPORT	0CA2	14	MTRTWP	0C86	14	MTSTRT	0CA4	14	MTWPRT	0CA0	14
NDF1	0F54	19	NDF11	0F5E	19	NDF2	0F56	19	NDF22	0F62	19
NEXTIP	0D10	16	NEXTOP	0D12	16	NIOQH1	01FE	4	NIOQH2	0256	4
NIOQH3	0226	4	NIOQH4	027C	4	NIOQHW	0CCC	16	NMF1	0F5A	19
NMF11	0F6B	19	NMF2	0F5C	19	NMF22	0F6F	19	NPR	0F58	19
NPR1	0F66	19	NSPI01	00A4	2	NSPIOW	0F0C	18	NSPIT1	04A2	6
NSPOT1	05B4	7	NSTHA1	0142	3	NSTHAW	0C74	14	NSUIH1	02A0	5
NSUIH2	03F0	5	NSUIH3	03FE	5	NSUIH4	041E	5	OPBYTE	058C	6
OPERR	0C5E	13	OPFIL1	0D54	17	OPINT	0CC4	15	OPM1	0B0F	13
OPM10	0C0B	13	OPM11	0C26	13	OPM12	0C4F	13	OPM2	0B1E	13
OPM3	0B34	13	OPM4	0B5C	13	OPM5	0B70	13	OPM6	0B83	13
OPM7	0B9B	13	OPM8	0BE3	13	OPM9	0BF9	13	OPPORT	0CA8	14
OPOS	0CFC	16	OPRAM	102C	20	OPROM	1025	20	OPRTWP	0C8C	14
OPSTIM	0CB4	14	OPSTRT	0CAA	14	OPTEXT	0B00	13	OPTINT	0CC0	15
OPWPRT	0CA6	14	OTPORT	0C9C	14	OTRTWP	0C80	14	OTSTRT	0C9E	14
OTWPRT	0C9A	14	PR	0AC2	13	PS	0AC4	13	PUART	040A	5
QIPN	0D0C	16	QOPN	0D0E	16	QUELEN	0AC8	13	RAM	0C74	14
RAM2	0C76	14	RAMEND	1034	20	RDR872	0AB4	12	ROMRAM	0CC8	15
SETEND	0CCC	15	SETIME	059E	6	SETNEG	0CC8	15	SETONE	0CBE	15
SETPC	0F2E	18	SETPC1	0F32	18	SETWP	0F2C	18	SETWP1	0F30	18
SM	0ABC	13	SPACE	0AD3	13	TASKAN	0C7E	14	TEN	0AC9	13
THOU	0ACE	13	VDUF	0CCA	15	VDUIO1	09BA	11	VDUIO2	09BE	11
VDUIOW	0F74	20	VDUOBA	0FB4	20	VDUOIW	0F94	20			

NURSE STATION PROCESSOR TRANSFER

VECTOR PROGRAM MODULE

1. DESCRIPTION

This program module defines the interrupt and XOP transfer vectors used by the Nurse Station Processor. Interrupt level 2 and XOP level 15 have been initialised to their appropriate values for the 990/4 Monitor Software.

2. IDENTIFICATION

SUBROUTINE NAME	NSPTV
PROGRAM MODULE	NSPTVP
GENERAL DATA MODULE	G0012D

3. SIZE

PROGRAM MODULE 164 Bytes.

4. CALLS FROM SUBROUTINE

INTERRUPT LEVEL	0	NSPI01
INTERRUPT LEVEL	3	NSUIH1
INTERRUPT LEVEL	4	NSUIH1
INTERRUPT LEVEL	5	NSUIH1

5. AUTHOR

B.T.V. WARTON.

```
      TITL 'NURSE STATION PROCESSOR TRANSFER VECTOR PROGRAM MODULE'
* PROGRAM MODULE IDENTIFIER
      IDT  'NSPTVP'
* CALLED PROGRAM MODULES
      REF  NSPI0P,NSUIHP
      REF  NSPI01,NSUIH1
* LINKED DATA MODULES
      REF  IHWA1,IHWA2,IHWA3
      RORG
      PSEG
********* P R O G R A M M E ***********
      DATA IHWA1,NSPI01             INTERRUPT 0
      DATA 0,0                                1
      DATA 0,>5D26                            2 REQUIRED BY MONITOR
      DATA IHWA3,NSUIH1                       3
      DATA IHWA2,NSUIH1                       4
      DATA IHWA1,NSUIH1                       5
      DATA 0,0,0,0,0,0,0,0,0,0                6 TO 10
      DATA 0,0,0,0,0,0,0,0,0,0                11 TO 15
      DATA 0,0,0,0,0,0,0,0,0,0      XOP TRANSFER VECTORS
      DATA 0,0,0,0,0,0,0,0,0,0      XOP TRANSFER VECTORS
      DATA 0,0,0,0,0,0,0,0,0,0      XOP TRANSFER VECTORS
      DATA >77D2,>604E              XOP 15 FOR MONITOR
      BSS  32                       WORKSPACE FOR MONITOR
ENTRY1 BLWP @0                      EMULATE INT 0
      END ENTRY1
```

NURSE STATION PROCESSOR INITIALISATION PROGRAM

1. DESCRIPTION

This program runs at system start-up time to initialise the contents of RAM. The initialisation that is performed is as follows :-

(1) Clearing of all RAM used.

(2) Initialisation of software flags to +1 or -1 as required.

(3) Initialisation of workspace register contents as required.

(4) Initialisation of hardware, i.e. programming of UART's etc.

2. IDENTIFICATION

SUBROUTINE NAME	NSPIO
PROGRAM MODULE	NSPIOP
GENERAL DATA MODULES	G0009D
GENERAL DATA MODULES	G0010D
GENERAL DATA MODULES	G0012D
GENERAL DATA MODULES	G0015D

3. SIZE

PROGRAM MODULE 158 Bytes.

4. CALLS TO SUBROUTINE

INTERRUPT LEVEL 0 NSPIO1

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSTHAP
BLWP	@NSTHA1

SUBROUTINE NAME	VDUIOP
BLWP	@VDUI01
SUBROUTINE NAME	NSUIHP
BL	@PUART

6. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.NSPIOP 12:26:00 FRIDAY, OCT 20, 1978.

```
      TITL 'NURSE STATION PROCESSOR INITIALISATION PROG'
* PROGRAMME MODULE IDENT
      IDT  'NSPIOP'
* TRANSFER VECTORS OR ENTRIES
      DEF  NSPI01
* CALLED PROGRAM MODULES
      REF  VDUIOP,NSTHAP,NSUIHP
      REF  VDUIO1,NSTHA1,PUART
* LINKED DATA MODULES
      REF  G0009D,G0010D,G0012D,G0015D
      REF  RAM,RAM2,RAMEND,SETONE,SETNEG,SETEND
      REF  IHWA1,IOWA1,OPTEXT,VDUOBA,NSTHAW
      REF  IHWA2,IOWA2
      REF  IHWA2,IOWA2
      REF  IHWA3,IOWA3
      RORG
      PSEG
********* P R O G R A M M E ************
NSPI01 LWPI RAM                  LOAD WP AT START OF RAM
       LI   R0,RAM2              GET ADDRESS OF SECOND RAM WORD
CLEAR  CLR  *R0+                 CLEAR RAM
       CI   R0,RAMEND            IS IT END OF RAM
       JNE  CLEAR
       LI   R0,SETONE            GET SET ONE FLAG ADDRESS
SET    INC  *R0+                 SET FLAGS TO +1
       CI   R0,SETNEG            IS IT END OF +1 FLAG SECTION
       JNE  SET
SET1   SETO *R0+                 SET FLAGS TO -1
       CI   R0,SETEND            IS IT END OF -1 FLAG SECTION
       JNE  SET1
******* UART 1 WA INITIALISATION
       LWPI IHWA1                LOAD INT HANDLER WP
       LI   R6,IOWA1             I/P O/P WA
       LI   R12,>1880            UART CRU BASE
       BL   @PUART               GO PROGRAM UART
       LWPI IOWA1                GET I/P O/P WP
       LI   R10,IHWA1            LOAD R10 WITH IHWA1 ADDRESS
       LI   R12,>1880            UART CRU BASE
******* UART 2 WA INITIALISATION
       LWPI IHWA2                LOAD INT HANDLER WP
       LI   R6,IOWA2             I/P O/P WA
       LI   R12,>1900            UART CRU BASE
       BL   @PUART               GO PROGRAM UART
       LWPI IOWA2                LOAD I/P O/P WP
       LI   R10,IHWA2            LOAD R10 WITH IHWA2 ADDRESS
       LI   R12,>1900            LOAD UART CRU BASE
******* UART 3 WA INITIALISATION
       LWPI IHWA3                LOAD INT HANDLER WP
       LI   R6,IOWA3             I/P O/P WA
       LI   R12,>1840            UART CRU BASE
       BL   @PUART               GO PROGRAM UART
       LWPI IOWA3                LOAD II/P O/P WP
       LI   R10,IHWA3            LOAD R10 WITH IHWA3 ADDRESS
       LI   R12,>1840            LOAD UART CRU BASE
******* O/P SYSTEM READY MESSAGE & INITIALISE HARDWARE
       LWPI NSTHAW               LOAD TASK HANDLER WP
       LI   R0,OPTEXT            GET O/P TEXT ADDRESS
       MOV  R0,VDUOBA            SAVE ADDRESS AT VDUOBA
       BLWP @VDUIO1              GOTO VDU PROG
       CLR  R0
```

```
SCR2.PROG.S.NSPIOP  12:26:00 FRIDAY, OCT 20, 1978.

       CLR  @RAM
       LI   R12,>100            LOAD 9901 CRU BASE
*******  ENABLING UART INTERRUPTS
       SBO  3                   ENABLE INT 3
       SBO  4                   ENABLE INT 4
       SBO  5                   ENABLE INT 5
UP     BLWP @NSTHA1             GO TO TASK HANDLER
       JMP  UP
       END
```

NURSE STATION PROCESSOR TASK HANDLER

1. DESCRIPTION

This program is concerned with scheduling active tasks in their order of priority. Before this program passes control to the highest priority active task, it tests whether the task is in the suspended state. If this is found to be the case, control is passed back to the task at the point at which it was interrupted. If the task is not in the suspended state, control is passed to it at its starting point.

2. IDENTIFICATION

SUBROUTINE NAME	NSTHA
PROGRAM MODULE	NSTHAP
GENERAL DATA MODULES	G0009D
GENERAL DATA MODULES	G0010D
GENERAL DATA MODULES	G0015D

3. SIZE

PROGRAM MODULE 188 Bytes.

4. CALLS TO SUBROUTINE

BLWP @NSTHA1.

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSPITP
BLWP	@NSPIT1
SUBROUTINE NAME	NSPOTP
BLWP	@NSPOT1
SUBROUTINE NAME	MOCPTP
BLWP	@MOCPT1

SUBROUTINE NAME	VDUIOP
BLWP	@VDUI01
BLWP	@INPUT

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data to this program is the task active and interrupt software flags,

IPSTIM, OPSTIM, MOCP, IOTF,
IPTINT, OPTINT, MOCINT, IPINT, OPINT.

6.2. OUTPUT DATA

The output data from this program is the task active number, which is placed in R5 of workspace area NSTHAW, and which can be referred to as TASKAN.

7. EXTERNAL DATA TRANSFERS

PERIPHERAL CRU INTERFACE

7.1. OUTPUT (TEST FACILITY)

CRU BASE ADDRESS		>1DE0
CRU LINE	0	INPUT TASK ACTIVE
CRU LINE	1	OUTPUT TASK ACTIVE
CRU LINE	2	MOCP TASK ACTIVE

8. TIMING

The execution time of this program is not in excess of 0·2mS.

9. AUTHOR

B.T.V. WARTON.

SCR2.TASK.S.NSTHAP 12:31:04 FRIDAY, OCT 20, 1978.

```
      TITL 'NURSE STATION PROCESSOR TASK HANDLER'
* PROGRAMME MODULE IDENT
      IDT  'NSTHAP'
* TRANSFER VECTORS OR ENTRIES
      DEF  NSTHA1
* CALLED PROGRAM MODULES
      REF  NSPITP,NSPOTP,MOCPTP,VDUIOP
      REF  NSPIT1,NSPOT1,MOCPT1,INPUT,VDUIO1
* LINKED DATA MODULES
      REF  G0009D,G0010D,G0015D
      REF  NSTHAW,ITRTWP,OTRTWP,MTRTWP,IPRTWP,OPRTWP
      REF  IPSTIM,OPSTIM,MOCP,IPTINT,OPTINT,MOCINT,IPINT
      REF  OPINT,IOTF
      RORG
      PSEG
*
******************* P R O G R A M M E ************
*
NSTHA1 DATA NSTHAW,START
START  CLR  R5                  RESET TASK NUMBER
       LIMI 7                   OPERATING SYSTEM TEST IF UART ACTIVE
       LIMI 0                   OPERATING SYSTEM PRIVILEGE INT MASK
       LWPI NSTHAW              LOAD TASK HANDLER WP
       LI   R12,>1DE0           LOAD TASK ACTIVATION CRU BASE
       MOV  @IPSTIM,R0          IS I/P TASK ACTIVE
       JNE  BIPT
       MOV  @OPSTIM,R0          IS O/P TASK ACTIVE
       JNE  BOPT
       MOV  @MOCP,R0            IS MOCP TASK ACTIVE
       JNE  BMOCP
       MOV  @IOTF,R0            VDU O/P ACTIVE
       JNE  BVDUOP
       LI   R5,5                SET TASK ACTIVE NUMBER TO 4
       ABS  @IPINT              WAS VDU I/P INTERRUPTED
       JGT  BIP
       LWPI IPRTWP              LOAD WP FOR VDU I/P TASK RETURNS
       RTWP
BIP    LIMI 7                   UNMASK ALL INTERRUPTS
       BLWP @INPUT              GO TO VDU I/P TASK
       JMP  START
BIPT   LI   R5,1                SET TO TASK 1
       ABS  @IPTINT             WAS I/P TASK INTERRUPTED
       JGT  BIPT1               JUMP IF NO
       LWPI ITRTWP              LOAD WP FOR INPUT TASK RETURNS
       RTWP
BIPT1  LIMI 7                   UNMASK ALL INTS
       SBO  0                   SET I/P TASK ACTIVE CRU BIT
       BLWP @NSPIT1             GO TO INPUT TASK
       SBZ  0                   RESET I/P TASK ACTIVE CRU BIT
       JMP  START
BOPT   LI   R5,2                SET TO TASK 2
       ABS  @OPTINT             WAS O/P TASK INTERRUPTED
       JGT  BOPT1
       LWPI OTRTWP              LOAD WP FOR OUTPUT TASK RETURNS
       RTWP                     RET TO INTERRUPTED TASK
BOPT1  LIMI 7                   UNMASK ALL INTS
       SBO  1                   SET O/P TASK ACTIVE CRU BIT
       BLWP @NSPOT1             GO TO OUTPUT TASK
       SBZ  1                   RESET O/P TASK ACTIVE CRU BIT.
       JMP  START
```

SCR2.TASK.S.NSTHAP 12:31:04 FRIDAY, OCT 20, 1978.

```
BMOCP   LI   R5,3          SET TO TASK 3
        ABS  @MOCINT       WAS MOCP INT
        JGT  BMOCP1
        LWPI MTRTWP        LOAD WP FOR MOCP TASK RETURNS
        RTWP               RET TO INTERRUPTED TASK
BMOCP1  LIMI 7             UNMASK ALL INTS
        SBO  2             SET MCP ACTIVE CRU BIT
        BLWP @MOCPT1       GO TO MOCP TASK
        SBZ  2             RESET MCP ACTIVE CRU BIT
        JMP  START
BVDUOP  LI   R5,4          SET TO TASK 4
        ABS  @OPINT        WAS VDU O/P INTERRUPTED
        JGT  BOP
        LWPI OPRTWP        LOAD WP FOR VDU O/P TASK RETURNS
        RTWP               RET TO INTERRUPTED TASK
BOP     LIMI 7             UNMASK ALL INTS
        BLWP @VDUIO1       GO TO VDU OUTPUT TASK
        JMP  START
        END
```

NURSE STATION I/O QUEUE HANDLER

1. DESCRIPTION

This program is concerned with placing the various inter-processor communications UARTs in input or output queues as required, and activating the input and output tasks accordingly. The program then when requested, supplies the next UART in the input queue to the input task or, the next UART in the output queue to the output task.

If the program finds that there are no more UARTs in the input queue, the input task is de-activated, and similarly if there are no more UARTs in the output queue, the output task is de-activated.

2. IDENTIFICATION

SUBROUTINE NAME	NIOQH
PROGRAM MODULE	NIOQHP
GENERAL DATA MODULE	G0009D
GENERAL DATA MODULE	G0011D
PERMANENT DATA MODULE	P0008D

3. SIZE

PROGRAM MODULE 162 Bytes.

4. CALLS TO SUBROUTINE

```
B       @NIOQH1
B       @NIOQH2
B       @NIOQH3
B       @NIOQH4
```

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSTHAP
BLWP	@NSTHA1

SUBROUTINE NAME NSUIHP

(SEE PROGRAM) NSUIH3

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data to this program is supplied in locations QIPN and QOPN, where :-

QIPN = workspace area address to be placed in the input queue, to be used later by the input task. Register R12 in this workspace contains the CRU base address for the UART to be used.

QOPN = workspace area address to be placed in output queue, to be used later by the output task. Register R12 in this workspace contains the CRU base address for the UART to be used.

6.2. OUTPUT DATA

The output data from this program is placed in locations NEXTIP and NEXTOP where :-

NEXTIP = the address of the next workspace area to be used by the Input Task.

NEXTOP = the address of the next workspace area to be used by the Output Task.

6.3. ADDITIONAL DATA CHANGES

The values supplied to this program at location QIPN and QOPN are placed in the queues IPQS and OPQS respectively.

The program also sets flags IPSTIM and OPSTIM to activate the Input Task or Output Task as required.

7. TIMING

The execution time of this program will not exceed 0·15mS.

8. AUTHOR

B.T.V. WARTON.

```
SCR2.PROG.S.NIOQHP  12:36:16 FRIDAY, OCT 20, 1978.

       TITL 'NURSE STATION I/O QUEUE HANDLER'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'NIOQHP'
* TRANSFER VECTORS OR ENTRIES
       DEF  NIOQH1,NIOQH2,NIOQH3,NIOQH4
* CALLED PROGRAM MODULES
       REF  NSTHAP,NSUIHP
       REF  NSTHA1,NSUIH3
* LINKED DATA MODULES
       REF  G0011D,G0009D,P0009D
       REF  NIOQHW,QIPN,IPQS,NEXTIP,QUELEN,IPSTIM
       REF  OPSTIM,QOPN,OPQS,NEXTOP
       RORG
       PSEG
********* P R O G R A M M E ************
****** INPUT QUEUE SECTION
NIOQH1 LWPI NIOQHW                I/O QUEUE HANDLER WA
       MOV  @QIPN,@IPQS(R2)       PUT NUMBER IN QUEUE
       INCT R2                    INC TO NEXT LOCATION IN QUEUE
       C    R2,@QUELEN            IS R2 = QUEUE LENGTH
       JLE  SETIP
       CLR  R2                    RESET POINTER
SETIP  MOV  @IPSTIM,R7            IS IPTASK ACTIVE
       JNE  OUT
       SETO @IPSTIM               STIM I/P TASK
       MOV  @IPQS(R1),@NEXTIP        GET NEXT I/P UART WA
OUT    BLWP @NSTHA1               GO TO YASK HANDLER
*
******* ENTRY TO QUEUE HANDLER TO GET NEXT I/P
NIOQH3 LWPI NIOQHW                LOAD QUEUE WP
       MOV  @IPQS(R1),R13         SAVE LAST IOWA
       INCT R1                    TO NEXT IN QUEUE
       C    R1,@QUELEN            IS R1=QUEUE LENGTH
       JLE  R1OK
       CLR  R1                    RESET POINTER
R1OK   C    R1,R2                 IS START=END
       JNE  GETIP
       CLR  @IPSTIM               DEACTIVATE I/P TASK
       CLR  @NEXTIP               RESET NEXT I/P REG
       JMP  ST
GETIP  MOV  @IPQS(R1),@NEXTIP GET NEXT I/P IN QUEUE
ST     AI   R13,-32               CAL UART WA
       LI   R14,NSUIH3            LOAD NSUIHP RET
       BLWP R13                   GOTO INT HANDLER TO TEST SUSPEND
*
******* OUTPUT QUEUE SECTION
NIOQH2 LWPI NIOQHW                LOAD QUEUE HANDLER WP
       MOV  @QOPN,@OPQS(R5)       PUT O/P IN QUEUE
       INCT R5                    NEXT LOCATION IN QUEUE
       C    R5,@QUELEN            IS R5 > QUEUE LENGTH
       JLE  SETOP
       CLR  R5                    RESET POINTER
SETOP  MOV  @OPSTIM,R7            IS I/P ACTIVE
       JNE  OUT
       MOV  @OPQS(R4),@NEXTOP        GET NEXT O/P
       SETO @OPSTIM               ACTIVATE O/P TASK
       JMP  OUT
*
******* ENTRY TO GET NEXT O/P ******
NIOQH4 LWPI NIOQHW                LOAD QUEUE WP
```

SCR2.PROG.S.NIOQHP 12:36:16 FRIDAY, OCT 20, 1978.

```
        INCT R4                       INC POINTER
        C    R4,@QUELEN               IS R4 > QUEUE LENGTH
        JLE  R4OK
        CLR  R4                       RESET POINTER
R4OK    C    R4,R5                    IS START=END OF QUEUE
        JNE  GETOP
        CLR  @OPSTIM                  DEACTIVATE O/P TASK
        CLR  @NEXTOP                  CLEAR NEXT O/P REG
        JMP  OUT
GETOP   MOV  @OPQS(R4),@NEXTOP GET NEXT O/P
        JMP  OUT
        END
```

NURSE STATION UART INTERRUPT HANDLER

1. DESCRIPTION

This program is an operating system program and is concerned with the interrupts generated by the UARTs used for inter-processor communication.

The functions performed by this program are :-

(1) The handling of the interrupts due to the UART interval timer.

(2) The transmission and reception of "Start of Heading" (SOH) characters "Acknowledge" (ACK) or "Negative Acknowledge" (NAK) characters for the correct start up of communication between processors.

(3) The furnishing of workspace area addresses to the Input/Output Queue Handler, for use by the Input or Output Tasks as required.

(4) The removal of unsuccessful communicating UARTs, from the input or output queues, and the possible indication of the errors to the operator if necessary.

(5) The programming of all UARTs.

This operating system program is allowed the privilege of running under the interrupt mask of level 0.

2. IDENTIFICATION

SUBROUTINE NAME	NSUIH
PROGRAM MODULE	NSUIHP
GENERAL DATA MODULES	G0009D

GENERAL DATA MODULES	G0010D
GENERAL DATA MODULES	G0011D
GENERAL DATA MODULES	G0015D
PERMANENT DATA MODULES	P0008D

3. SIZE

PROGRAM MODULE 514 Bytes.

4. CALLS TO SUBROUTINE

INTERRUPT LEVEL	3	@NSUIH1	IHWA3
INTERRUPT LEVEL	4	@NSUIH1	IHWA2
INTERRUPT LEVEL	5	@NSUIH1	IHWA1

B	@NSUIH2
B	@NSUIH3
B	@NSUIH4
BL	@PUART

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSTHAP
BLWP	@NSTHA1
SUBROUTINE NAME	NIOQHP
B	@NIOQH1
B	@NIOQH2

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The three return vectors stored in R13, R14 and R15 of the workspace area used, are stored in the required locations for the currently active task.

6.2. OUTPUT DATA

During the operation of this program, the software flag which indicates that the task has been interrupted (i.e. IPTINT, OPTINT, MOCINT, IPINT, OPINT) is set accordingly.

7. EXTERNAL DATA TRANSFERS

PERIPHERALS TMS 9902 UARTs.

7.1. INPUT

INTERRUPT LEVEL	3
CRU BASE	>1840
INTERRUPT LEVEL	4
CRU BASE	>1900
INTERRUPT LEVEL	5
CRU BASE	>1880

CRU BIT Functions as specified in the TMS 9902 Data Sheets.

7.2. OUTPUT

CRU BASES AS ABOVE.

8. TIMING

The execution time for this program is of the order of 0•2mS in normal communication situations.

9. AUTHOR

B.T.V. WARTON.

```
SCR2.TASK.S.NSUIHP  12:52:39 FRIDAY, OCT 20, 1978.

       TITL 'NURSE STATION UART INT HANDLER'
* PROGRAMME MODULE IDENT (TASK)
       IDT  'NSUIHP'
* TRANSFER VECTORS OR ENTRIES
       DEF  NSUIH1,NSUIH2,NSUIH3,NSUIH4,PUART
* CALLED PROGRAMME MODULES
       REF  NSTHAP,NIOQHP
       REF  NSTHA1,NIOQH1,NIOQH2
* LINKED DATA MODULES
       REF  G0009D,G0010D,G0011D,G0015D,P0009D
       REF  QOPN,QIPN,IOTF,OPERR,VDUOBA,ROMRAM
       REF  ITWPRT,ITPCRT,ITSTRT,OTWPRT,OTPCRT,OTSTRT
       REF  MTWPRT,MTPCRT,MTSTRT,IPWPRT,IPPCRT,IPSTRT
       REF  IPTINT,OPTINT,MOCINT,IPINT,NSTHAW,TASKAN
       REF  OPWPRT,OPPCRT,OPSTRT,OPINT
       REF  IPQS,OPQS,NEXTOP,NEXTIP,QUELEN
       RORG
       PSEG
*
SOHD   EQU  >0100
ACKD   EQU  >0600
NAKD   EQU  >1500
*
*****************  P R O G R A M M E  ***********
*
NSUIH1 LIMI 0                     OPERATING SYSTEM PROG
       BL   @STRR                 SUSPEND TASK ,SAVE RETS
       TB   19                    TIMINT
       JEQ  TIMELP                IF EQU GOTO TIMER ELAPS SECTION
       MOV  R7,R7                 I/P O/P MODE FLAG
       JLT  IPMODE                IF -VE = I/P MODE
       JGT  OPMODE                IF +VE = O/P MODE
       STCR R4,8                  GET DATA
       MOV  R0,R0                 IS O/P RESP EXPECTED
       JNE  OPYES
       CI   R4,SOHD               SOH ?
       JNE  SOHNO
       TB   9                     P,F,O ERROR ?
       JEQ  OPNAK
OPACK  LI   R4,ACKD               ACK
       BL   @OP                   O/P ACK
       SETO R1                    SET SOH RECEIVED FLAG
       SETO R2                    SET I/P FLAG
       CLR  R10                   ERROR COUNT
       BL   @SETIME               GO SET UART TIMER
BOUT   BLWP @NSTHA1               GO TO TASK HANDLER
*
****** OUTPUT CHAR SECTION
OP     SBO  18                    RIENB
       SBO  16                    TRANSMITTER ON
       LDCR R4,8                  OUTPUT CHAR
       SBZ  16                    TRANSMITTER OFF
       RT
*
TRYNAK CI   R4,NAKD               IS CHAR NAK ?
       JEQ  OPSOH
       BL   @NSUIH4               GO ENABLE NSUIHP PROG
       SETO R3                    SET O/P SUSPEND FLAG
       BL   @SETIME               GO SET UART TIMER
       JMP  BOUT
```

```
OPYES  CI   R4,ACKD               ACK ?
       JNE  TRYNAK
       CLR  R7                    RESET I/O MODE
       INC  R7                    SET O/P MODE
       CLR  R0                    RESET O/P RESP FLAG
       CLR  R10                   ERROR COUNT
OPMODE SBZ  18                    INHIBIT INT RBRL
       BL   @SETIME               GO SET UART TIMER
       SBZ  20                    INHIBIT TIMER INT
       MOV  R6,@QOPN              GIVE QUEUE NUMBER TO HANDLER
       B    @NIOQH2               GO TO PUT IN O/P QUEUE
SOHNO  MOV  R1,R1                 IS SOH FLAG SET
       JEQ  OPNAK
       SETO R7                    SET I/P MODE
       CLR  R1                    SOH FLAG
       CLR  R10                   CLEAR ERROR COUNT
*
****** INPUT MODE SECTION
* SET TIME TO TIME TILL CHAR IS USED BY IPTASK
IPMODE SBZ  18                    INHIBIT RBRL INT
       BL   @SETIME               GO SET UART TIMER
       SBZ  20                    INHIBIT TIMER INT
       MOV  R6,@QIPN              GIVE QUEUE NUMBER
       B    @NIOQH1               GO TO QUEUE HANDLER
OPNAK  CI   R4,NAKD               WAS CHAR NAK
       JEQ  NAKER                 DO NOT SEND NAK IF NAK RECEIVED
       BL   @PUART                GO PROGRAM UART
       LI   R4,NAKD               NAK
       CLR  R2                    RESET I/P FLAG
       BL   @OP                   O/P NAK
NAKER  BL   @ERROR                GO TO ERROR SECTION
       BL   @NSUIH4               GO ENABLE NSUIHP PROG
       JMP  BOUT
*
******  ERROR SECTION
ERROR  INC  R10                   ERROR COUNT
       CI   R10,100               IS THERE 100 ERRORS
       JLT  RETS
* INDICATE ERROR
       CLR  R10                   CLEAR ERROR COUNT
       MOV  @IOTF,@IOTF           IS VDU IN USE
       JNE  RETS
       LI   R10,OPERR             ERROR O/P ADDRESS
       MOV  R10,@VDUOBA           SAVE ADDRESS AT VDUOBA
       SETO @IOTF                 SET VDU ACTIVE
       SETO @ROMRAM               SET NOT MIXED
RETS   RT
*
******* INTERVAL TIMER INTERRUPT SECTION
TIMELP SBZ  20                    INHIBIT TIMER
       MOV  R7,R7                 TEST I/O MODE
       JEQ  IONO
       JGT  KILLOP
       LI   R8,IPQS               GET IPQS ADDRESS
KILL1  MOV  R8,R5
       A    @QUELEN,R5            CAL END
       JMP  KILL
KILLOP LI   R8,OPQS               GET OPQS ADDRESS
       JMP  KILL1
KILL   C    R6,*R8                IS IT THIS UART IN QUEUE
```

```
       JEQ  KILLIT
KILL2  INCT R8
       C    R8,R5              IS IT AT END
       JLE  KILL
       JMP  OVER
KILLIT CLR  *R8                REMOVE FROM QUEUE
       JMP  KILL2
OVER   C    R6,@NEXTOP         IS NEXT THIS UART
       JNE  TESTIP
       CLR  @NEXTOP            REMOVE FROM NEXTOP
TESTIP C    R6,@NEXTIP         IS NEXT THIS UART
       JNE  CLRWA
       CLR  @NEXTIP            REMOVE FROM NEXTIP
CLRWA  CLR  *R6                RESET BLIST POINTER
IONO   MOV  R0,R0              IS O/P RESPONCE EXPECTED
       JNE  OPSOHE
       MOV  R7,R7              WAS O/P ACTIVE
       JGT  OPSOH              GO O/P SOH IF YES
       BL   @NSUIH4            RESET FLAGS
       MOV  R3,R3              IS O/P SUSPENDED
       JNE  OPSOH
       BL   @PUART             PRAGRAM UART
       JMP  BOUT
OPSOHE BL   @ERROR             GO TO ERROR SECTION
       JMP  OPSOH
*
******* SET UART TIMER SECTION
SETIME SBO  13                 LDIR SET
       LI   R5,>FA00           16 MS
       LDCR R5,8
       SBO  20                 ENABLE TIMER INT
       RT
*
OPSOH  LI   R4,SOHD            SOH
       BL   @OP
       SETO R0                 SET O/P RESP
       CLR  R1                 RESET SOH
       CLR  R2                  ''   I/P FLAG
       CLR  R3                  ''   O/P ''
       CLR  R7                  ''   I/O MODE
       BL   @SETIME
       BLWP @NSTHA1            GO TO TASK HANDLER
* O/P REQUEST
NSUIH2 LIMI 0
       MOV  R2,R2              IS IT INPUTING
       JEQ  OPSOH
       SETO R3                 O/P SUSPENDED FLAG
       BLWP @NSTHA1            GO TO TASK HANDLER
*
******* O/P SUSPENDED TEST SECTION
NSUIH3 LIMI 0                  PRIVILEGE MASK
       MOV  R3,R3              IS O/P SUSPENDED
       JNE  NSUIH2             JUMP IF YES
       BLWP @NSTHA1            GO TO TASK HANDLER
*
********    PROGRAMMING UART SECTION
PUART  SBO  31                 RESET UART
       LI   R8,>6300           GET UART  CHAR FORMAT
       LDCR R8,8               LOAD FORMAT
       SBZ  13                 RESET TIMER
```

SCR2.TASK.S.NSUIHP 12:52:39 FRIDAY, OCT 20, 1978.

```
       LI   R8,>0010              GET RATE
       LDCR R8,12                 LOAD UART WITH RATE
       SBO  18                    ENABLE INT
       RT
*
******* ENABLE INT HANDLER SECTION
NSUIH4 CLR  R0                    O/P RESP
       CLR  R1                    SOH
       CLR  R2                    I/P
       CLR  R7                    I/O MODE
       RT
*
******* STORE RETURNS SECTION  *******************
STRR   MOV  @TASKAN,R5            GET TASK ACTIVE NUMBER
       CLR  @TASKAN
       MOV  R5,R5                 IS NO TASK ACTIVE
       JEQ  EXIT
       DEC  R5                    IS INPUT TASK ACTIVE
       JEQ  ITSAVE
       DEC  R5                    IS OUTPUT TASK ACTIVE
       JEQ  OTSAVE
       DEC  R5                    IS MOCP TASK ACTIVE
       JEQ  MTSAVE
       DEC  R5                    IS VDU O/P TASK ACTIVE
       JEQ  OPSAVE
       CLR  R5                    IS VDU I/P TASK ACTIVE
       JMP  IPSAVE
* SAVE INPUT TASK RETURNS
ITSAVE MOV  R13,@ITWPRT           I/P TASK WP RET
       MOV  R14,@ITPCRT           I/P TASK PC RET
       MOV  R15,@ITSTRT           I/P TASK ST RET
       SETO @IPTINT               SET I/P TASK INTERRUPTED FLAG
EXIT   RT
* SAVE OUTPUT TASK RETURNS
OTSAVE MOV  R13,@OTWPRT           O/P TASK WP RET
       MOV  R14,@OTPCRT           O/P TASK PC RET
       MOV  R15,@OTSTRT           O/P TASK ST RET
       SETO @OPTINT               SET O/P TASK INTERRUPTED FLAG
       RT
* SAVE MOCP TASK RETURNS
MTSAVE MOV  R13,@MTWPRT           MOCP TASK WP RET
       MOV  R14,@MTPCRT           MOCP TASK PC RET
       MOV  R15,@MTSTRT           MOCP TASK ST RET
       SETO @MOCINT               SET MOCP TASK INTERRUPTED FLAG
       RT
* SAVE VDU O/P TASK RETURNS
OPSAVE MOV  R13,@OPWPRT           VDU O/P TASK WP RET
       MOV  R14,@OPPCRT           VDU O/P TASK PC RET
       MOV  R15,@OPSTRT           VDU O/P TASK ST RET
       SETO @OPINT                SET VDU O/P INT FLAG
       RT
* SAVE VDU I/P TASK RETURNS
IPSAVE MOV  R13,@IPWPRT           VDU I/P TASK WP RET
       MOV  R14,@IPPCRT           VDU I/P TASK PC RET
       MOV  R15,@IPSTRT           VDU I/P TASK ST RET
       SETO @IPINT                SET VDU I/P TASK INT FLAG
       RT
       END
```

NURSE STATION PROCESSOR INPUT TASK

1. DESCRIPTION

This program has been written as a multi input re-entrant task, and is concerned with receiving characters from all of the inter-processor communicating UARTs. The task has six main sections, the first of which is concerned with obtaining the workspace area it is to use, which contains the required UART CRU base address, and the current state of the input communication protocol. The remaining five sections are concerned with :-

(1) Receiving the input instruction.

(2) Receiving the inverse of the input instruction.

(3) Receiving input data.

(4) Receiving the Block Check Character (BCC), and

(5) Receiving the End of Transmission Character (EOT).

2. IDENTIFICATION

SUBROUTINE NAME	NSPIT
PROGRAM MODULE	NSPITP
GENERAL DATA MODULE	G0009D
GENERAL DATA MODULE	G0011D
GENERAL DATA MODULE	G0013D

3. SIZE

PROGRAM MODULE 274 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP      @NSPIT1
```

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSUIHP
BL	@NSUIH4
SUBROUTINE NAME	NIOQHP
B	@NIOQH3

6. INTERNAL DATA TRANSFER

6.1. OUTPUT DATA

The characters received by this task are placed in the input files associated with each UART. The position of the input file is found by adding 86 to the address of the workspace pointer being used.

6.2. ADDITIONAL DATA CHANGES

Register RO in each of the input/output workspace areas is used as a Branch List Pointer, and is set to the next address to be used in the list when this task is next executed for that UART.

Having received the characters and placed them in an input file the task activates the MOCP task by setting flags MOCP and MCPIP, and incrementing the MOCP activation counter MOCPSC.

7. EXTERNAL DATA CHANGES

PERIPHERAL TMS 9902 UART

7.1. INPUT/OUTPUT

The CRU base address of the UART to be used is provided in the workspace area register R12 supplied to the Input Task.

8. TIMING

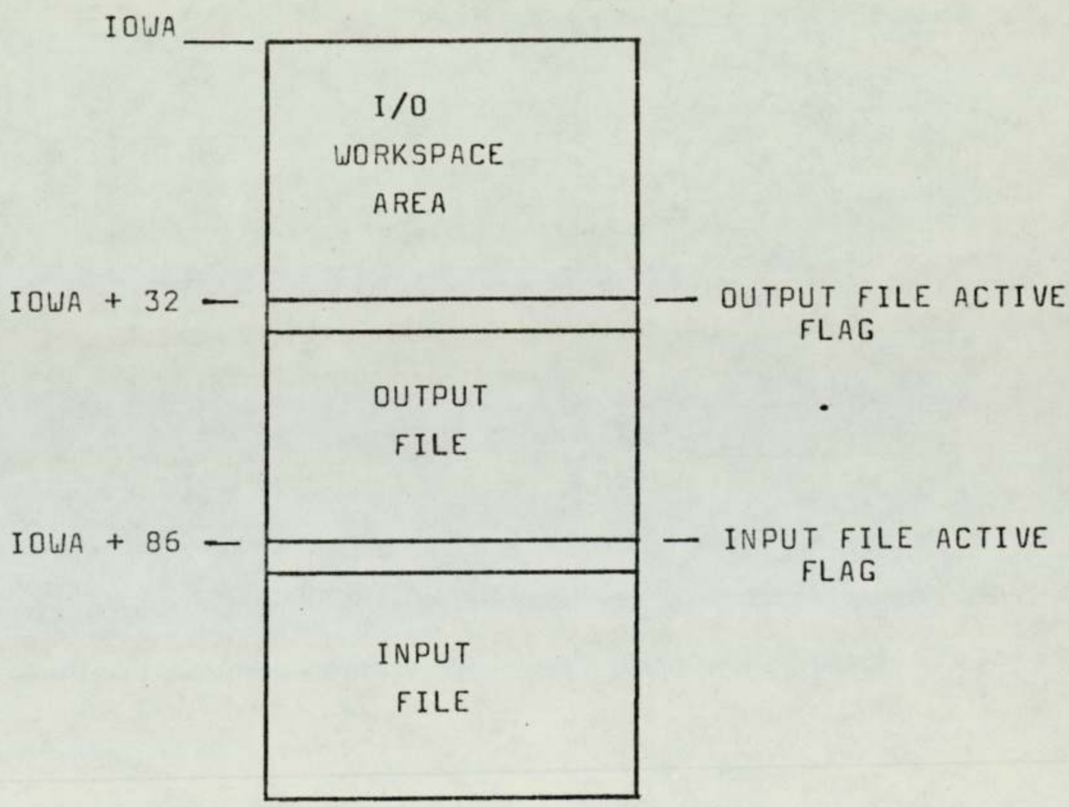
The execution time of this task is of the order of 0.2mS.

9. AUTHOR

B.T.V. WARTON.

10. NOTE

The file memory structure is an important feature of the operation of this program. The input/output file structure for each UART is shown below.



```
SCR2.PROG.S.NSPITP  13:14:14 FRIDAY, OCT 20, 1978.

       TITL 'NURSE STATION PROCESSOR I/P TASK'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'NSPITP'
* TRANSFER VECTORS OR ENTRIES
       DEF  NSPIT1,OPBYTE,SETIME
* CALLED PROGRAM MODULES
       REF  NSUIHP,NIOQHP
       REF  NSUIH4,NIOQH3
* LINKED DATA MODULES
       REF  G0013D,G0011D,G0009D
       REF  NSPIOW,NEXTIP,SETWP,SETPC,MOCP,MOCPSC,MCPIP
       REF  NSTHAW
       RORG
       PSEG
************* P R O G R A M M E ************
ACKOP  EQU  >0600
ENQ    EQU  >0500
NAKOP  EQU  >1500
*
NSPIT1 DATA NSPIOW,START
START  MOV  @NEXTIP,@SETWP     GO GET NEXT I/P UART
       JEQ  GONEXT             IF = 0 GO GET NEXT
       LI   R7,ADDRES          GET ADDRESS
       MOV  R7,@SETPC          MOV ADDRESS TO SETPC
       BLWP @SETWP             GET I/P WA
ADDRES TB   25                 TEST TIMELP
       JEQ  TIMER              JUMP IF TIME ELAPS
       LI   R1,BLIST           GET BRANCH LIST ADDRESS
       A    R0,R1              CAL BLIST POINTER
       MOV  *R1,R1             GET BRANCH FROM POINTER
       B    *R1                BRANCH
*
******* I/P INSTRUCTION SECTION
IPINST STWP R2
       AI   R2,86              GET I/P FILE ADDRESS
       STCR *R2,8              GET INST
       LI   R4,ENQ             LOAD ENQ CHAR
       BL   @OPBYTE            O/P ENQ
       BL   @SETIME            GO SET TIMER
       LI   R0,2               SET BLIST POINT TO IPINVI
       JMP  GONEXT
*
******* RECEIVE INV INSTRUCTION
IPINVI SETO R5                 MACK R5 =-1
       STCR R5,8               I/P CHAR
       AB   *R2,R5             ADD INST TO R5
       INC  R5                 INST+INVINST+1=0
       JNE  OPNAK              R5 SHOULD=0 FOR CORRECT INST
       LI   R4,ACKOP           LOAD ACK
       BL   @OPBYTE            O/P ACK
       BL   @SETIME            GO SET TIMER
       MOVB *R2+,R5            GET INST IS IT +VE
       JLT  NEGINS
       LI   R0,8               SET BLIST POINT TO EOT
       JMP GONEXT
NEGINS LI   R9,50              LOAD 50 BYTE COUNTER
       CLR  R8                 BCC SUM
       LI   R0,4               SET BLIST POINT TO IPDATA
       JMP  GONEXT
OPNAK  LI   R4,NAKOP           LOAD NAK
```

```
       LIMI 0                          *** MASK BEFOR LEAVIHG TASK **
       BL   @OPBYTE                    O/P NAK
       CLR  R0                         RESET BLIST POINT
       LI   R11,CHWP                   LOAD R11 WITH RET
       BLWP R10                        SET WA
CHWP   BL   @NSUIH4                    ENABLE INT HANDLER
       JMP  GONEXT
*
******** I/P DATA SECTION
IPDATA STCR R5,8                       STORE CHAR
       AB   R5,R8                      BCC SUM
       MOVB R5,*R2+                    PUT DATA IN I/P FILE
       LI   R4,ENQ                     LOAD ENQ CHAR
       BL   @OPBYTE                    O/P ENQ
       BL   @SETIME                    SET TIMER
       DEC  R9                         DEC BYTE COUNT
       JNE  GONEXT
       LI   R0,6                       SET BLIST POINT TO IPBCC
       JMP GONEXT
*
*******  I/P BCC & CHECK SECTION
IPBCC  STCR R5,8                       I/P CHAR
       CB   R5,R8                      ARE BCC SAME
       JNE  OPNAK
       LI   R4,ACKOP                   LOAD ACK CHAR
       BL   @OPBYTE                    O/P ACK
       BL   @SETIME                    SET TIMER
       LI   R0,8                       SET BLIST POINT TO IPEOT
       JMP  GONEXT
*
******* I/P EOT SECTION
IPEOT  SETO @MOCP                      STIM MOCP
       INC  @MOCPSC                    INC MOCP STIM COUNT
       SETO @MCPIP                     ACTIVATE MOCP TASK
       CLR  R0                         RESET BLIST
       STWP R2                         SAVE WP
       AI   R2,84                      WP+84=I/P FILE
       SETO *R2                        SET I/P FILE ACTIVE
       LI   R11,HERE                   SET R11 RET
       BLWP R10                        GET NEW WP
HERE   LIMI 0                          ****MASK BEFOR LEAVING****
       SBO  18                         ENABLE REC INT
       BL   @NSUIH4                    ENABL INT HANDLER
GONEXT EQU  $
       LWPI NSTHAW                     GET TASK HANDLER WP
       SBZ  0                          RESET TASK ACTIVE BIT
       B    @NIOQH3                    GO GET NEXT I/P FROM QUEUE
*
******* O/P CHAR SECTION
OPBYTE SBO  18                         RIENB
       SBO  16                         TRANSMITTER ON
       LDCR R4,8                       LOAD UART
       SBZ  16                         TRAN OFF
       RT
*
****** TIMELP HANDLER
TIMER  CLR  R0                         RESET BLIST
       LI   R11,HERE                   SET R11 RET
       BLWP R10                        ABANDON I/P
*
```

```
SCR2.PROG.S.NSPITP  13:14:14 FRIDAY, OCT 20, 1978.

******  SET UART TIMER SECTION
SETIME SBO  13                   ENABLE UART TIMER
       LI   R5,>FA00             LOAD R5 WITH COUNT
       LDCR R5,8                 LOAD UART WITH COUNT
       SBO  20                   ENABLE INT
       RT
*
****** BRANCH LIST VALUES
BLIST  DATA IPINST,IPINVI,IPDATA,IPBCC,IPEOT
       END
```

NURSE STATION PROCESSOR OUTPUT TASK

1. DESCRIPTION

This program has been written as a multi output re-entrant task, and is concerned with transmitting characters via the inter-processor communicating UARTs. The task has seven main sections, the first of which is concerned with obtaining the workspace area it is to use, which contains the UART CRU base address required, and the current state of the output communication protocol.

The remaining six sections are concerned with :-

(1) Transmitting instruction.

(2) Transmitting inverse of instruction.

(3) Testing that the "Acknowledge" character (ACK) is received, and transmission of EOT character.

(4) Transmitting data.

(5) Transmitting block check character (BCC).

(6) Testing that the "Acknowledge" character (ACK) is received.

2. IDENTIFICATION

SUBROUTINE NAME	NSPOT
PROGRAM MODULE	NSPOTP
GENERAL DATA MODULE	G0013D

3. SIZE

PROGRAM MODULE 284 Bytes.

4. CALLS TO SUBROUTINE

BLWP @NSPOT1

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NIOQHP
B	@NIOQH4
SUBROUTINE NAME	NSUIHP
BL	@NSUIH4
SUBROUTINE NAME	NSPITP
BL	@OPBYTE
BL	@SETIME

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The characters which are to be transmitted by this task are present in the Output files associated with each UART. The position of the output file is found by adding 34 to the address of the workspace pointer being used.

6.2. ADDITIONAL DATA CHANGES

Register R0 in each of the input/output workspace areas is used as a Branch list pointer, and is set to the next address to be used in the list, when this task is next executed for that UART.

7. EXTERNAL DATA CHANGES

PERIPHERAL TMS 9902 UART.

7.1. INPUT/OUTPUT

The CRU base address for the UARTs to be used is provided in the workspace area register R12 supplied to the Output Task.

8. TIMING

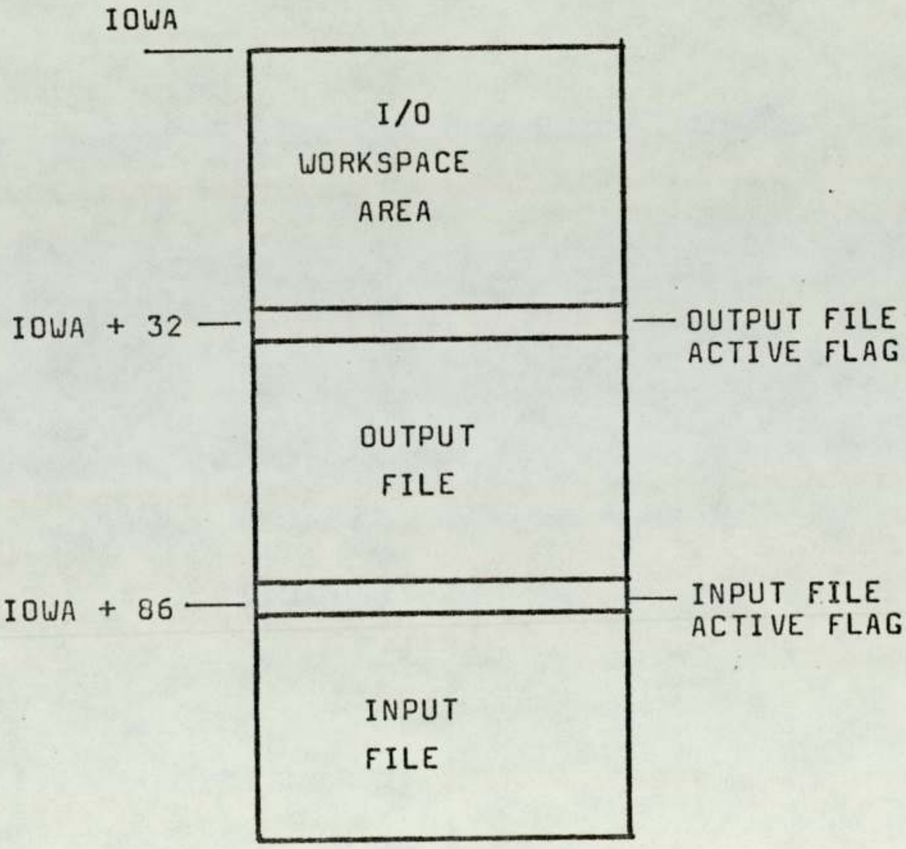
The execution time of this task is of the order of 0·2mS.

9. AUTHOR

B.T.V. WARTON.

10. NOTE

The file memory structure is an important feature of the operation of this program. The input/output file structure for each UART is shown below.



SCR2.PROG.S.NSPOTP 14:36:01 FRIDAY, OCT 20, 1978.

```
       TITL 'NURSE STATION PROCESSOR O/P TASK'
* PROGRAMME MODULE IDENT (PROG)
       IDT  'NSPOTP'
* TRANSFER VECTORS OR ENTRIES
       DEF  NSPOT1
* CALLED PROGRAM MODULES
       REF  NIOQHP,NSUIHP,NSPITP
       REF  NIOQH4,NSUIH4,OPBYTE,SETIME
* LINKED DATA MODULES
       REF  G0013D
       REF  NSPIOW,NEXTOP,SETWP1,SETPC1
       REF  NSTHAW
       RORG
       PSEG
************** P R O G R A M M E *************
EOT    EQU  >0400
ACK    EQU  >0600
NAK    EQU  >1500
*
NSPOT1 DATA NSPIOW,START
START  MOV  @NEXTOP,@SETWP1    GET WP
       JEQ  GONEXT
       LI   R7,LOCATE          GET PC
       MOV  R7,@SETPC1         SAVE PC RET
       BLWP @SETWP1            GET NEW WP
LOCATE TB   25                 TIMELP
       JEQ  TIMER              JUMP IF TIMER ELAPST
       LI   R1,BLIST1          GET BRANCH LIST ADDRESS
       A    R0,R1              CAL BLIST POINTER
       MOV  *R1,R1             GET BRANCH POINT
       B    *R1                BRANCH
*
****** O/P INSTRUCTION SECTION
OPINST STWP R2                 SAVE WP
       AI   R2,34              WP +34 =O/P FILE ADDRESS
       MOVB *R2+,R4            GET INST IN FILE
       BL   @OPBYTE            O/P INST
       BL   @SETIME            SET UART TIMER
       LI   R0,2               SET BLIST POINT TO OPINVI
       JMP  GONEXT
*
******* O/P INVERSE OF INSTRUCTION
OPINVI INV  R4                 INVERT INST
       BL   @OPBYTE            O/P INV INST
       BL   @SETIME            SET TIMER
       LI   R0,4               SET BLIST POINT TO ACK1
       JMP GONEXT
*
******* CHECK REPLY IS ACK
ACK1   STCR R5,8               I/P REPLY
       CI   R5,ACK             IS IT ACK
       JNE  TRYNAK             JUMP IF NO
       INV  R4                 IS INST +VE
       JLT  NEGINS             JUMP TO O/P DATA
OPEOT  LIMI 0                  **** MASK BEFOR EOT EXIT *****
       LI   R4,EOT             LOAD EOT CHAR
       BL   @OPBYTE            O/P EOT
       CLR  R0                 RESET BLIST POINTER
       STWP R2                 SAVE WP
       AI   R2,32              WP + 32 =O/P FILE FLAG
```

```
       CLR   *R2                        RESET O/P FILE ACTIVE FLAG
       LI    R11,CHWP1                  LOAD R11 RET
       BLWP  R10                        GET NEW WA
CHWP1  EQU   $
       BL    @NSUIH4                    GO ENABLE INT HANDLER
*
****** EXIT FROM O/P TASK TO QUEUE HANDLER
GONEXT EQU   $
       LWPI  NSTHAW                     GET TASK HANDLER WP
       SBZ   1                          RESET O/P TASK ACTIVE FLAG
       B     @NIOQH4                    GO TO QUEUE HANDLER
*
******* O/P OF DATA SECTION ( 50 BYTES)
NEGINS LI    R9,50                      SET BYTE COUNTER TO 50
       CLR   R8                         BCC
       MOVB  *R2,+R4                    GET DATA
       AB    R4,R8                      BCC SUM
       DEC R9                           DEC BYTE COUNTER
       BL    @OPBYTE                    O/P DATA
       BL    @SETIME                    SET TIMER
       LI    R0,6                       SET BLIST POINT TO OPDATA
       JMP   GONEXT
*
****** TEST FOR NAK REPLY
TRYNAK CLR   R0                         RESET BLIST POINT
       CI    R5,NAK                     IS REPLY NAK
       JNE   TOUT                       JUMP IF NO ?
       LI    R11,CHWP2                  LOAD R11 RET
       BLWP  R10                        GET NEW WP
CHWP2  LIMI  0                          ***MASK BEFOR EXIT*****
       BL    @NSUIH4                    GET INT HANDLER TO RETX SOH
       LI    R4,>0100                   SOH
       BL    @OPBYTE                    O/P SOH AGAIN
       BL    @SETIME                    SET TIMER
       SETO  R0                         SET RESP FLAG
       JMP   GONEXT
*
****** TIMING OUT ERROR SECTION
TOUT   LI    R11,CHWP3                  LOAD R11 RET
       BLWP  R10                        GET NEW WP
CHWP3  LIMI  0                          ***MASK BEFOR EXIT*****
       BL    @NSUIH4                    TIME OUT RESTART
       BL    @SETIME                    SET TIMING OUT TIME
       SETO  R3                         SET O/P SUSPENDED
       JMP   GONEXT
*
****** O/P DATA SECTION
OPDATA MOVB  *R2+,R4                    GET DATA
       AB    R4,R8                      BCC SUM
       BL    @OPBYTE                    O/P DATA
       BL    @SETIME                    SET UART TIMER
       DEC   R9                         DATA COUNT
       JNE   GONEXT
       LI    R0,8                       SET BLIST POINT TO OPBCC
       JMP   GONEXT
*
****** O/P BLOCK CHECK CHARACTER (BCC) SECTION
OPBCC  MOVB  R8,R4                      GET BCC
       BL    @OPBYTE                    O/P BCC
       BL    @SETIME                    SET TIMER
```

```
SCR2.PROG.S.NSPOTP   14:36:01 FRIDAY, OCT 20, 1978.
        LI    R0,10                    SET BLIST POINT TO ACK2
        JMP   GONEXT
*
****** RECEIVE ACK 2
ACK2    STCR R5,8                      I/P REPLY
        CI    R5,ACK                   IS IT ACK
        JNE   TRYNAK                   JUMP IF NOT
        JMP   OPEOT
*
****** UART TIMER ELAPS SECTION
TIMER   CLR   R0                       RESET BLIST POINTER
        LI    R11,CHWP2                LOAD R11 RET
        BLWP R10                       GO O/P SOH AGAIN
*
****** BRANCH LIST VALUES
BLIST1 DATA OPINST,OPINVI,ACK1,OPDATA,OPBCC,ACK2
        END
```

MICRO/OPERATOR COMMUNICATION PACKAGE

1. DESCRIPTION

This program is concerned with interpreting and/or generating the communication information which passes between processors or the operator dedicated processor and the operator. For example this task will interpret on input instruction from the operator (via the VDU), and generate the command to be sent to the required patient dedicated processor.

2. IDENTIFICATION

SUBROUTINE NAME	MOCPT
PROGRAM MODULE	MOCPTP
GENERAL DATA MODULE	G0008D
GENERAL DATA MODULE	G0009D
GENERAL DATA MODULE	G0012D
GENERAL DATA MODULE	G0014D
GENERAL DATA MODULE	G0015D
PERMANENT DATA MODULE	P0008D

3. SIZE

PROGRAM MODULE 582 Bytes.

4. CALLS TO SUBROUTINE

BLWP @MOCPT1.

5. CALLS FROM SUBROUTINE

SUBROUTINE NAME	NSUIHP
B	@NSUIH2
SUBROUTINE NAME	VDUIOP
BLWP	@VDUIO2

SUBROUTINE NAME	ASCIIP
BL	@ASCII1
SUBROUTINE NAME	BIHEXP
BL	@BIHEX1

6. INTERNAL DATA TRANSFERS

6.1. INPUT DATA

The input data to this program is supplied in the VDU input buffer (BUFA) or any of the UART input files (IPFI1S, IPFI2S, IPFI3S).

6.2. OUTPUT DATA

The output data from this program for the VDU is an address where the output to the VDU can be found (VDUOBA). The output data to be sent to patient processors is placed in the required UART output files (OPFIL1 etc.).

6.3. ADDITIONAL DATA CHANGES

The software flags which may be changed by the operation of this program are :-

MCPIP, IPSF, MOSPSC, MOCP, ROMRAM, IOTF, OPFIL1, IPFIL1, IPFIL2, IPFIL3.

7. TIMING

The execution time of this program is dependent on the input instruction which it has received.

8. AUTHOR

B.T.V. WARTON.

SCR2.TASK.S.MOCPTP 15:39:43 FRIDAY, OCT 20, 1978.

```
       TITL 'MICRO/OPERATOR COMMUNICATION PACKAGE'
* PROGRAMME MODULE IDENT (TASK)
       IDT  'MOCPTP'
* TRANSFER VECTORS OR ENTRIES
       DEF  MOCPT1
* CALLED PROGRAM MODULES
       REF  NSUIHP,VDUIOP,ASCIIP,BIHEXP
       REF  NSUIH2,VDUIO2,ASCII1,BIHEX1
* LINKED DATA MODULES
       REF  G0008D,G0009D,G0012D,G0014D,G0015D,P0008D
       REF  MOCPTW,MCPIP,IPSF,BUFA,SM,EM,CM
       REF  ERROR1,MOCPSC,MOCP,IPLIST,OPM1,OPM2,OPM3,OPM4
       REF  OPM5,OPM6,OPM7,OPM8,OPM9,OPM10,OPM11,OPM12
       REF  VDUOBA,ROMRAM,IOTF,HEXONE,IHWA1,OPFIL1
       REF  OPROM,OPRAM,NDF1,NDF11,NDF2,NDF22,NPR,NPR1
       REF  NMF1,NMF11,NMF2,NMF22,PR,PS
       REF  IPFIL1,IPFI1S
       REF  IPFIL2,IPFI2S
       REF  IPFIL3,IPFI3S
       RORG
       PSEG
************* P R O G R A M M E *************
MOCPT1 DATA MOCPTW,START
START  MOV  @MCPIP,R0           IS IT MICRO TO MICRO I/P
       JLT  MINPUT
TESTIP ABS  @IPSF               IS IT OPERATOR I/P
       JLT  OINPUT
       CLR  @MOCPSC             CLEAR MOCP STIM COUNT
       CLR  @MOCP               CLEAR MOCP ACTIVE FLAG
       RTWP
*
****** MICRO TO MICRO COMMUNICATION INTERPRETER
MINPUT MOV  @IPFIL1,R0          IS I/P FILE 1 ACTIVE
       JLT  IPA1
       MOV  @IPFIL2,R0          IS I/P FILE 2 ACTIVE
       JLT  IPA2
       MOV  @IPFIL3,R0          IS I/P FILE 3 ACTIVE
       JLT  IPA3
* EXTEND THIS IF MORE I/P FILES AVAILABLE
*
       CLR  @MCPIP
       JMP  TESTIP
*
****** GET FILE ADDRESS SECTION
IPA1   LI   R1,IPFI1S           GET START OF I/P FILE
       BL   @DECODE             DECODE MESSAGE
       CLR  @IPFIL1             CLEAR IPFIL1 FLAG
       JMP  OUT
IPA2   LI   R1,IPFI2S           GET START OF I/P FILE
       BL   @DECODE             DECODE MESSAGE
       CLR  @IPFIL2             CLEAR IPFIL2 FLAG
       JMP  OUT
IPA3   LI   R1,IPFI3S           GET START OF I/P FILE
       BL   @DECODE             DECODE MESSAGE
       CLR  @IPFIL3             CLEAR IPFIL3 FLAG
       JMP  OUT
* EXTEND THIS IF MORE I/P FILES AVAILABLE
*
****** OPERATOR I/P INTERPRETER
OINPUT LI   R7,BUFA             GET VDU BUFFER
```

```
        C     @SM,*R7              IS IT START MONITORING
        JEQ   DOSM
        C     @EM,*R7              END MONITORING ?
        JEQ   DOEM
        C     @CM,*R7              CONTINUE MONITORING
        JEQ   DOCM
        C     @PR,*R7              PULSE RATE
        JEQ   DOPR
        C     @PS,*R7              PATIENT STATUS
        JEQ   DOPS
ERRORM  LI    R2,ERROR1            I/P COMMAND ERROR
        BL    @VDUOP               O/P MESSAGE
****** EXIT FROM PROGRAMME IF NO STIMS SECTION
OUT     DEC   @MOCPSC              DEC MOCP STIM COUNT
        JNE   GO                   JUMP IF NOT = 0
        CLR   @MCPIP               RESET MICRO I/P FLAG
        CLR   @MOCP                RESET MOCP ACTIVE FLAG
        RTWP                       RETURN TO TASK HANDLER
GO      MOV   @MOCPSC,R2           GET MOCP STIM COUNT
        CI    R2,1                 IS IT = 1
        JNE   AGAIN                JUMP IF NO
        MOV   @IPSF,R2             IS OPERATOR FLAG SET
        JGT   AGAIN                JUMP IF IPSF +VE
        CLR   @MCPIP               RESET MICRO I/P FLAG
AGAIN   B     @START               GO TO START OF THIS TASK
*
****** MICRO TO MICRO MESSAGE GENERATION SECTION
DOSM    INCT  R7                   POINT R7 TO NEXT WORD IN BUFA
        BL    @WHICHU              GO FIND WHICH PATIENT UART
WA1     LI    R6,>5000             >50=BYTE CODE FOR START MONITOR
        MOVB  R6,*R8               PUT IN O/P FILE
        LI    R10,DSTIM            SET R10 FOR RET
        BLWP  R9                   GET IHWA REQUIRED
****** TASK STIM TEST
DSTIM   DEC   @MOCPSC              DEC MOCP STIM COUNT
        JNE   BRANCH
        CLR   @MOCP                DEACTIVATE TASK
        CLR   @MCPIP               MCP I/P FLAG
BRANCH  B     @NSUIH2              GO TO NSUIHP TO ACTIVATE O/P FROM UA
*
DOEM    INCT  R7                   POINT TO NEXT WORD IN BUFA
        BL    @WHICHU              WHICH PATIENT
        LI    R6,>5300             >53=END MONITORING
        MOVB  R6,*R8               SAVE IN O/P FILE
        LI    R10,DSTIM            SET R10 TO RET
        BLWP  R9
*
DOCM    INCT  R7                   INC BUFA POINTER
        BL    @WHICHU              WHICH PATIENT
        LI    R6,>5500             >55=CONTINUE MONITORING
        MOVB  R6,*R8               SAVE IN O/P FILE
        LI    R10,DSTIM            SET R10 TO RET
        BLWP  R9
*
DOPR    INCT  R7                   INC BUFA POINTER
        BL    @WHICHU              WHICH PATIENT
        LI    R6,>5C00             PR CODE
        MOVB  R6,*R8               PUT IN FILE
        LI    R10,DSTIM            SET R10 TO RET
        BLWP  R9
```

```
*
DOPS    INCT R7                         INC BUFA POINTER
        BL   @WHICHU                    WHICH PATIENT
        LI   R6,>5600                   PS CODE
        MOVB R6,*R8                     SAVE IN O/P FILE
        LI   R10,DSTIM                  SET R10 TO RET
        BLWP R9
*
****** MICRO TO MICRO MESSAGE DECODER
DECODE EQU  $
        MOV  R11,R12                    ******SAVE RET *********
        LI   R5,IPLIST                  GET ADDRESS OF COMMAND STRING
        CB   *R5+,*R1
        JEQ  MON                        JUMP IF MONITORING COMMAND
        CB   *R5+,*R1
        JEQ  MONOK                      MONITOR OK
        CB   *R5+,*R1
        JEQ  MONFAI                     MONITOR FAIL
        CB   *R5+,*R1
        JEQ  EMJMP                      MONITOR END
        CB   *R5+,*R1
        JEQ  CONMON                     CONTINUING MONITORING
        CB   *R5+,*R1
        JEQ  ERM                        ERROR MESSAGE
        CB   *R5+,*R1
        JEQ  ALREMO                     ALREADY MONITORING
        CB   *R5+,*R1
        JEQ  NOTMON                     NOT MONITORING
        CB   *R5+,*R1
        JEQ  DIAG                       DIAGNOSIS
        CB   *R5+,*R1
        JEQ  PRD                        PULSE RATE DATA
        CB   *R5+,*R1
        JEQ  PSD                        PATIENT STATUS DATA
* ETC   ( INSERT MORE COMMANDS HERE AS REQUIRED)
        B    *R12                       RET USING SAVED RET
MON     LI   R2,OPM1                    MONITORING
        JMP  STIMGO
MONOK   LI   R2,OPM2                    MONITOR START UP OK
        JMP  STIMGO
MONFAI  LI   R2,OPM3                    MON S.U FAIL
        JMP  STIMGO
EMJMP   LI   R2,OPM4                    END OF MONITORING
        JMP  STIMGO
CONMON  LI   R2,OPM5                    CONTINUE MON
        JMP  STIMGO
ERM     LI   R2,OPM6                    ERROR MESSAGE
        JMP  STIMGO
ALREMO  LI   R2,OPM8                    ALREADY MON MESSAGE
        JMP  STIMGO
NOTMON  LI   R2,OPM9                    NOT MON
STIMGO  BL   @VDUOP
        B    *R12                       RET USING SAVED RET
*
****** PROCESSING OF DIAGNOSIS DATA
DIAG    INCT R1                         INC FILE POINTER
        MOV  *R1+,@NDF1                 SAVE DF1
        MOV  *R1+,@NDF2                 SAVE DF2
        BL   @BIHEX1                    GO TO BINARY TO HEX CONVERTER
        DATA NDF1,NDF11,NDF2,NDF22
```

```
        DATA >FFFF                  END OF DATA MARKER
        LI   R2,OPM10               DIAGNOSIS MESSAGE
        LI   R3,NDF11               DIAGNOSIS MESSAGE
        BL   @MVDUOP                O/P MESSAGE
        B    *R12                   RET USING SAVE RET
*
* PROCESSING OF PULSE RATE DATA
PRD     INCT R1                     INC FILE POINTER
        MOV  *R1+,@NPR              GET PR DATA
        BL   @ASCII1                GO TO BINARY TO ASCII SUB
        DATA NPR,NPR1,>FFFF
        LI   R2,OPM12               PULSE RATE
        LI   R3,NPR1
        JMP  STIMRR
*
****** PROCESSING OF PATIENT STATUS DATA
PSD     INCT R1                     INC FILE POINTER
        MOV  *R1+,@NPR              GET PULSE RATE DATA
        MOV  *R1+,@NDF1             GET DIAGNOSIS WORD 1
        MOV  *R1+,@NDF2             GET DIAGNOSIS WORD 2
        MOV  *R1+,@NMF1             GET MONITORING WORD 1
        MOV  *R1+,@NMF2             GET MONITORING WORD 2
        BL   @BIHEX1                GO TO BINARY TO HEX CONVERTER
        DATA NPR,NPR1,NDF1,NDF11,NDF2,NDF22
        DATA NMF1,NMF11,NMF2,NMF22,>FFFF
        LI   R2,OPM11               O/P MESSAGE 11
        LI   R3,NDF11               O/P DATA
STIMRR  BL   @MVDUOP                GO TO VDU O/P
        B    *R12                   RET USING SAVED RET
* ETC
*
****** VDU NOT ROMRAM MIX OUTPUT
VDUOP   MOV  R2,@VDUOBA             GIVE OUT PUT ADDRESS TO VDU PROG
        SETO @ROMRAM                NOT MIXED
        BLWP @VDUIO2                GO DIRECT TO VDU TASK
        RT
*
****** VDU ROM/RAM MIXED O/P
MVDUOP  MOV  R2,@OPROM              SET PROM RESIDENT TEXT ADDRESS
        MOV  R3,@OPRAM              SET RAM RESIDENT TEXT ADDRESS
        ABS  @ROMRAM                MIXED DATA
        BLWP @VDUIO2                GO DIRECT TO VDU TASK
        RT
*
****** WHICH UART SECTION
WHICHU  C    *R7,@HEXONE            IS IT PATIENT ONE
        JEQ  UART1
* INSERT PROGRAM HERE AS MORE PATIENT PROCESSORS
* BECOME AVAILABLE
        B    @ERRORM
UART1   LI   R8,OPFIL1              LOAD R8 WITH OPFIL1 ADDRESS
        MOV  *R8,R0                 IS O/P FILE ACTIVE
        JEQ  FILEOK
        LI   R2,OPM7                TELL OPERATOR O/P ACTIVE
        BL   @STIMGO                O/P MESSAGE
        B    @OUT
FILEOK  SETO *R8+                   SET OPFILE ACTIVE
        LI   R9,IHWA1               LOAD R9 WITH IHWA1 TO BE USED
        RT
        END
```

BINARY TO HEX CONVERTER

1. DESCRIPTION

This program is concerned with converting a 16 bit binary word into the four ASCII character hexadecimal value of the word, needed for later transmission to a VDU terminal.

2. IDENTIFICATION

SUBROUTINE NAME	BIHEX
PROGRAM MODULE	BIHEXP
PERMANENT DATA MODULE	P0008D

3. SIZE

PROGRAM MODULE 64 Bytes.

4. CALLS TO SUBROUTINE

```
BL    @BIHEX1
```

5. INTERNAL DATA TRANSFERS

5.1. INPUT DATA

The input data for this subroutine is supplied after the BL @BIHEX1 statement by using the following statement,

```
DATA    a, b, a, b, >FFFF
```

where a = address of word to be converted

b = address where results are to be saved, and

>FFFF = end of data marker.

5.2. OUTPUT DATA

As stated above, the output results from this program are saved at the addresses specified by "b" words in the input data format.

5.3. ADDITIONAL DATA CHANGES

The last character of each of the group of 4 hexadecimal characters is made negative as an end of string indicator.

This program makes use of register R2, R3, R4, R5 and R11 in the currently active workspace area.

6. TIMING

The executing time of this program is dependent on the number of words to be converted.

7. AUTHOR

B.T.V. WARTON.

SCR2.TASK.S.BIHEXP 16:15:46 FRIDAY, OCT 20, 1978.

```
       TITL ' BINARY TO HEX CONVERTER '
* PROGRAMME MODULE IDENT (TASK)
       IDT  'BIHEXP'
* TRANSFER VECTORS OR ENTRIES
       DEF  BIHEX1
* LINKED DATA MODULES
       REF  P0008D
       REF  ASCDAT
       RORG
       PSEG
********* P R O G R A M M E ************
BIHEX1 MOV  *R11+,R2
       CI   R2,>FFFF            IS IT END OF DATA
       JEQ  OVER
       MOV  *R2,R2              GET DATA
       CLR  R3                  COUNT
       MOV  *R11+,R4            GET ADDRESS TO PUT DATA
BYTE   MOVB R2,R5               GET FIRST BYTE OF WORD
       SRL  R5,12               GET MOST SIG 4 BITS
       AI   R5,ASCDAT           R5 + ASCDAT ADDRESS
       MOVB *R5,*R4+            SAVE DATA
       MOVB R2,R5               GET FIRST BYTE AGAIN
       ANDI R5,>0F00            GET LS 4BITS MS BYTE
       SRL  R5,8                MAKE INTO WORD
       AI   R5,ASCDAT
       MOVB *R5,*R4+            SAVE DATA
       INCT R3                  COUNT
       CI   R3,4                IS WORD CONVERTED
       JEQ  NEG
       SWPB R2                  SWAP BYTE IN CONVERTION WORD
       JMP  BYTE
NEG    DEC  R4                  BACK 1 CHAR
       CLR  R2
       MOVB *R4,R2              GET CHAR
       NEG  R2                  MACK -VE
       MOVB R2,*R4              RET CHAR
       JMP  BIHEX1
OVER   RT
       END
```

BINARY TO ASCII DECIMAL

1. DESCRIPTION

This program is concerned with converting a 16 bit word into the ASCII characters required to display the decimal value of the word on a VDU.

2. IDENTIFICATION

SUBROUTINE NAME	ASCII
PROGRAM MODULE	ASCIIP
PERMANENT DATA MODULE	P0008D

3. SIZE

PROGRAM MODULE 100 Bytes.

4. CALLS TO SUBROUTINE

BL @ASCII1

5. INTERNAL DATA TRANSFERS

5.1. INPUT DATA

The input data for this subroutine is supplied after the BL @ASCII1 statement by using the following statement,

DATA a, b, a, b, ... >FFFF

where a = address of word to be converted

b = address where results are to be saved

and >FFFF = end of data marker.

5.2. OUTPUT DATA

As stated above the output results from this program are stored at the address specified by the "b" words in the input data format.

5.3. ADDITIONAL DATA CHANGES

The last ASCII character expressing the value of the word converted, is made negative as an end of string indicator.

This program makes use of registers R4, R5, R7, R8, R9 and R11 in the currently active workspace area.

6. TIMING

The executing time of this program is dependent on the number of words to be converted.

7. AUTHOR

B.T.V. WARTON.

```
SCR2.TASK.S.ASCIIP  16:18:58 FRIDAY, OCT 20, 1978.

       TITL ' BINARY TO ACSII DECIMAL '
* PROGRAMME MODULE IDENT (TASK)
       IDT  'ASCIIP'
* TRANSFER VECTORS OR ENTRIES
       DEF  ASCII1
* LINKED DATA MODULES
       REF  P0008D
       REF  MINUS,SPACE,THOU,HUND,TEN,ASCNUM
***********          P R O G R A M M E     ***************
ASCII1 MOV R11,R8                  SAVE DATA ADDRESS
NEXTV  CLR  R4
       MOV  *R8+,R5                GET DATA ADDRESS
       CI   R5,>FFFF               IS IT END OF DATA
       JEQ  OVER1
       MOV  *R8+,R9                GET DATA RET
       MOV  *R5,R5                 GET DATA
       ABS  R5                     IS IT NEG
       JGT  DOSPAC
       JEQ  DOSPAC
       MOVB @MINUS,*R9+            O/P MINUS
       JMP  DODIV
DOSPAC MOVB @SPACE,*R9+            INSERT A SPACE
DODIV  DIV  @THOU,R4               R4/1000
       CLR  R7                     ZERO SUP FLAG
       BL   @TZERO                 GO TEST FOR ZERO
       DIV  @HUND,R4               R4/100
       BL   @TZERO                 GO TEST FOR ZERO
       DIV  @TEN,R4                R4/10
       BL   @TZERO                 GO TEST FOR ZERO
       A    @ASCNUM,R5             R5+>30
       SWPB R5
       NEG  R5                     MAKE LAST CHAR -VE
       MOVB R5,*R9                 SAVE
       JMP  NEXTV
OVER1  B    *R8                    RETURN
TZERO  MOV  R7,R7                  TEST FLAG
       JNE  L1
       MOV  R4,R4                  TEST R4
       JNE  L1
       MOVB @SPACE,*R9+            INSERT A SPACE
       JMP  CONTA
L1     SETO R7                     SET FLAG
       A    @ASCNUM,R4             R4+>30
       SWPB R4
       MOVB R4,*R9+                SAVE
CONTA  CLR  R4                     RESET R4
       RT
       END
```

VDU INPUT/OUTPUT PROGRAM

1. DESCRIPTION

This is the lowest priority level task in the operator dedicated processor system. If the system has no other tasks active, control is passed to the VDU input routine. The output routine of the VDU program can be entered in two ways. The first is by activating the output routine as a task, which will be scheduled by the task handler program. The second method is direct entry to the VDU output routine, as is sometimes performed by the MOCP task. The second method is quicker, and does not require the de-activation of the MOCP task before the VDU output routine will run.

2. IDENTIFICATION

SUBROUTINE NAME	VDUIO
PROGRAM MODULE	VDUIOP
GENERAL DATA MODULE	G0008D
GENERAL DATA MODULE	G0013D
PERMANENT DATA MODULE	P0007D

3. SIZE

PROGRAM MODULE 248 Bytes.

4. CALLS TO SUBROUTINE

```
BLWP    @VDUIO1
BLWP    @VDUIO2
BLWP    @INPUT
```

5. INTERNAL DATA TRANSFERS

5.1. INPUT DATA

The input data to this program is the address where the output string of characters can be found, plus software flags indicating if the characters are in both ROM and RAM memory (flag ROMRAM). If the output string is resident in only RAM or ROM the address is supplied in location VDUOBA. If both types of memory contain the output string, the addresses of their locations are given at locations OPROM and OPRAM.

5.2. OUTPUT DATA

The characters received from the VDU are stored in buffer BUFA.

5.3. ADDITIONAL DATA CHANGES

On leaving this program, the task active flag IOTF is reset.

6. EXTERNAL DATA TRANSFERS

PERIPHERAL VDU

6.1. INPUT

CRU BASE >1000

TMS 9902 UART used in interface.

6.2. OUTPUT

AS ABOVE.

7. TIMING

The execution time of this program is dependent on the length of the character string to be output.

8. AUTHOR

B.T.V. WARTON.

SCR2.PROG.S.VDUIOP 16:22:27 FRIDAY, OCT 20, 1978.

```
       TITL ' VDU I/O PROGRAMME'
* PROGRAMME MODULE IDENT (PROG)
       IDT   'VDUIOP'
* TRANSFER VECTORS OR ENTRIES
       DEF   VDUIO1,VDUIO2,INPUT
* LINKED DATA MODULES
       REF   G0015D,G0008D,P0007D
       REF   VDUIOW,VDUF,CR1712,RDR872,VDUOBA,CR,BUFA
       REF   CL,CP,LINE,EOB,IOTF,OCPF,IPSF,VDUOIW
       REF   OSTIMC,EOS,FREE
       REF   ROMRAM,OPRAM,OPROM,MOCP,MOCPSC
       RORG
       PSEG
************            PROGRAMME          ****************
VDUIO1 DATA VDUIOW,START         ENTRY POINT FROM TASK HANDLER
VDUIO2 DATA VDUOIW,START         DIRECT ENTERY POINT
****** OUTPUT SECTION
START  ABS   @VDUF               TEST VDU FLAG
       JGT   LOADA
       LI    R12,>1000           LOAD CRU BASE
       SBO   31                  RESET COMMAND
       LDCR @CR1712,8            LOAD COUNT & RESET LDCTRL
       SBZ   20                  INHIBIT TIMER
       SBZ   13                  RESET LDIR
       LDCR @RDR872,11           LOAD RDR & RESET LRDR
       LDCR @RDR872,12           LOAD XDR & RESET LXDR
       LWPI VDUOIW               LOAD OTHER WP
       LI    R12,>1000           LOAD CRU BASE
       LWPI VDUIOW               RET TO FIRST WP
       CLR   R4                  CHAR REG
LOADA  EQU   $
       MOV   @ROMRAM,@ROMRAM     IS IT MIXED
       JGT   MIXED
       MOV   @VDUOBA,R0          GET ADDRESS OF O/P TEX
       CLR   R5
NEXT   CB    *R0,@EOB            IS CHAR END OF BUFFER MARKER
       JEQ   FOUND
       MOVB *R0+,R4              GET CHAR
       MOVB R4,R9                SAVR CHAR
       BL    @PRINT              O/P CHAR
       MOVB R9,R9                WAS CHAR -VE
       JGT   NEXT
       INCT R5
       MOV   @VDUOBA(R5),R0      GET NEXT ADDRESS
       JMP   NEXT
*
****** MIXED DATA SECTION
MIXED  MOV   @OPROM,R0           GET ROM TEXT ADDRESS
       MOV   @OPRAM,R1           GET RAM TEXT ADDRESS
MIXED1 CB    *R0,@EOB            IS CHAR END OF BUFFER
       JEQ   FOUND
       MOVB *R0+,R4              GET CHAR
       MOVB R4,R9                SAVE CHAR
       BL    @PRINT              O/P CHAR
       MOVB R9,R9                WAS CHAR -VE
       JGT   MIXED1
NEXT1  MOVB *R1+,R4              GET CHAR
       MOVB R4,R9                SAVE CHAR
       BL    @PRINT              O/P CHAR
       MOVB R9,R9                WAS CHAR -VE
```

```
       JGT   NEXT1
       JMP   MIXED1
*
****** OUTPUT CHAR SECTION
PRINT  SBO   16                  TURN ON TRANSMITTER
XBRE   TB    22                  WAIT FOR XBRE=1
       JNE   XBRE
       ABS   R4                  MAKE CHAR +VE
       LDCR  R4,8                LOAD CHAR
       SBZ   16                  TURN OFF TRAN
       RT
*
****** EXIT FROM TASK
FOUND  CLR   @IOTF               RESET IOTF FLAG
       RTWP
*
******* I N P U T   S E C T I O N
INPUT  DATA VDUIOW,INPUTS
INPUTS LWPI VDUIOW
       LIMI 7
       CLR   @FREE               FREE TIME COUNTER
       CLR   R2                  CHAR COUNTER
       LI    R3,BUFA             GET I/P BUFFER ADDRESS
RCVLP  INC   @FREE               INC FREE TIME COUNT
       TB 21                     TEST RCVLP
       JNE   RCVLP               JUMP IF CHAR NOT REC
       STCR  R4,8                STORE CHAR
       SBZ   18                  RESET REC
       BL    @PRINT              ECHO CHAR
       CB    R4,@CL              IS IT CLEAR LINE
       JEQ   INPUTS
       CB    R4,@CP              IS IT CLEAR PAGE
       JEQ   INPUTS
       MOVB  R4,*R3              PUT CHAR IN BUFFER
       INC   R2                  INC CHAR COUNT
       CB    R4,@CR              IS CHAR CR
       JEQ   EXITLF
       C     R2,@LINE            IS CHAR COUNT=LINE
       JEQ   EXIT
       INC   R3                  INC BUF ADDRESS
       JMP   RCVLP
EXIT   SETO  @MOCP               SET MOCP TASK FLAG
       INC   @MOCPSC             STIM MOCP COUNTER
       SETO  @IPSF               SET IPSF FLAG FOR MOCP
       CLR   @IOTF               RESET IOTF FLAG
       RTWP                      RET TO SSTH
* LINE FEED SECTION
EXITLF LI    R4,>0A00            =LF
       BL    @PRINT              O/P LF
       JMP   EXIT
       END
```

PERMANENT DATA MODULE P0007D

1. DESCRIPTION

This data module contains VDU, UART programming data and control characters.

2. IDENTIFICATION

PERMANENT DATA MODULE P0007D

3. SIZE

10 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'PERMANENT DATA MODULE P0007D'
* PERMANENT DATA MODULE IDENTIFIER
       IDT  'P0007D'
       EVEN
       DEF  CR1712,RDR872,LINE,EOB,CR,CL,CP
       RORG
       PSEG
CR1712 DATA >6200              CONTROL REG
RDR872 DATA >34                9600
LINE   DATA 80                 80 CHAR PER LINE
EOB    BYTE >80                END OF BUFFER
CR     BYTE >0D                CARRAGE RET
CL     BYTE >18                CLEAR LINE
CP     BYTE >0C                CLEAR PAGE
       END
```

PERMANENT DATA MODULE P0008D

1. DESCRIPTION

This data module contains lists of allowed operator input instructions, inter-processor instructions, and a list of output messages for the VDU.

2. IDENTIFICATION

PERMANENT DATA MODULE P0008D

3. SIZE

440 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
        TITL 'PERMANENT DATA MODULE P0008D'
* PERMANENT DATA MODULE IDENTIFIER
        IDT  'P0008D'
        EVEN
        DEF  SM,EM,CM,PR,PS,ERROR1,OPM1,OPM2,OPM3,OPM4
        DEF  OPM5,OPM6,OPM7,OPM8,OPM9,OPM10,OPM11,OPM12
        DEF  OPERR,IPLIST,HEXONE,OPTEXT,QUELEN
        DEF  ASCDAT,MINUS,SPACE,THOU,HUND,TEN,ASCNUM
        RORG
        PSEG
SM      DATA >534D          START MONITORING
EM      DATA >454D          END     ''
CM      DATA >434D          CONTIUE ''
PR      DATA >5052          PULSE RATE
PS      DATA >5053          PATIENT STATUS
HEXONE DATA >310D          1+ CR
QUELEN DATA 16             IP &O/P QUEUE LENGTH
TEN     DATA 10             CONSTANT
HUND    DATA 100            CONSTANT
THOU    DATA 1000           CONSTANT
ASCNUM DATA >30            CONSTANT
MINUS   BYTE >2D            MINUS SIGN
SPACE   BYTE >20            ASCII SPACE
ASCDAT TEXT '0123456789ABCDEF'
IPLIST BYTE >21            MON
        BYTE >22            MON OK
        BYTE >24            MON FAIL
        BYTE >27            END MON
        BYTE >28            CONT MON
        BYTE >2B            ERROR
        BYTE >2D            ALREADY MON
        BYTE >2E            NOT MON
        BYTE >A3            DIAG
        BYTE >C0            PR
        BYTE >81            PATIENT STATUS
*
****** OUTPUT TEXT    ****************
ERROR1 TEXT 'MNEMONIC ERROR'
        BYTE >0A,>0D,>80
OPTEXT TEXT 'SYSTEM READY'
        BYTE >0A,>0D,>80
OPM1    BYTE >07
        TEXT 'MONITORING'
        BYTE >07,>0A,>0D,>80
OPM2    TEXT 'MONITOR START UP OK'
        BYTE >0A,>0D,>80
OPM3    TEXT 'MONITOR START UP FAIL,RESTART MONITOR'
        BYTE >0A,>0D,>80
OPM4    TEXT 'END OF MONITORING'
        BYTE >0A,>0D,>80
OPM5    TEXT 'MONITORING AGAIN'
        BYTE >0A,>0D,>80
OPM6    TEXT 'INTER PROCESSOR ERROR'
        BYTE >0A,>0D,>80
OPM7    TEXT 'THE PREVIOUS COMMAND FOR THIS PATIENT IS '
        TEXT 'NOT COMPLETED; '
        BYTE >07
        TEXT 'TRY AAGAIN!'
        BYTE >07,>0A,>0D,>80
OPM8    TEXT 'ALREADY MONITORING!'
        BYTE >0A,>0D,>80
OPM9    TEXT 'NOT MONITORING!'
```

```
        BYTE >0A,>0D,>80
OPM10   TEXT -'DIAGNOSIS CODE DF1='
        TEXT -' DF2='
        BYTE >0A,>0D,>80
OPM11   TEXT -'PATIENT STATUS DF1='
        TEXT -' DF2='
        TEXT -' PR='
        TEXT -' MF1='
        TEXT -' MF2='
        BYTE >0A,>0D,>80
OPM12   TEXT -'PULSE RATE ='
        BYTE >0A,>0D,>80
OPERR   TEXT 'COMMUNICATION ERROR'
        BYTE >0A,>0D,>80
        END
```

GENERAL DATA MODULE G0009D

1. DESCRIPTION

This data module must be the first in RAM as it contains the label RAM used by the initialisation program (NSPIOP). It contains the Task Handler workspace area, the stack of storage locations for return vectors and task active flags.

2. IDENTIFICATION

GENERAL DATA MODULE G0009D

3. SIZE

CLASS TWO : 74 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
        TITL 'GENERAL DATA MODULE G0009D'
* GENERAL DATA MODULE IDENTIFIER
        IDT   'G0009D'
        EVEN
* CLASS TWO
        DEF   RAM,RAM2,NSTHAW,ITRTWP,OTRTWP,MTRTWP,IPRTWP
        DEF   ITWPRT,ITPCRT,ITSTRT,OTWPRT,OTPCRT,OTSTRT
        DEF   MTWPRT,MTPCRT,MTSTRT,IPWPRT,IPPCRT,IPSTRT
        DEF   IPSTIM,OPSTIM,MOCP,MOCPSC,MCPIP,IPSF,TASKAN
        DEF   OPRTWP,OPWPRT,OPPCRT,OPSTRT
        RORG
        PSEG
RAM     EQU   $                   START OF RAM
RAM2    EQU   $+2                 SECOND WORD OF RAM
TASKAN  EQU   $+10                TASK ACTIVE NUMBER
NSTHAW  BSS   6                   TASK HANDLER WA
ITRTWP  BSS   6                   I/P TASK WP FOR RET
OTRTWP  BSS   6                   O/P TASK WP FOR RET
MTRTWP  BSS   6                   MCP TASK WP FOR RET
OPRTWP  BSS   6                   VDU O/P TASK WP FOR RET
IPRTWP  BSS   2                   VDU I/P TASK WP FOR RET
* I/P TASK RETURNS
ITWPRT  BSS   2                   I/P TASK WP RET
ITPCRT  BSS   2                   I/P TASK PC RET
ITSTRT  BSS   2                   I/P TASK ST RET
* O/P TASK RETURNS
OTWPRT  BSS   2                   O/P TASK WP RET
OTPCRT  BSS   2                   O/P TASK PC RET
OTSTRT  BSS   2                   I/P TASK ST RET
* MOCP TASK RETURNS
MTWPRT  BSS   2                   MCP TASK WP RET
MTPCRT  BSS   2                   MCP TASK PC RET
MTSTRT  BSS   2                   MCP TASK ST RET
* VDU O/P TASK RETURNS
OPWPRT  BSS   2                   VDU O/P TASK WP RET
OPPCRT  BSS   2                   VDU O/P TASK PC RET
OPSTRT  BSS   2                   VDU O/P TASK ST RET
* VDU I/P TASK RETURNS
IPWPRT  BSS   2                   VDU I/P TASK WP RET
IPPCRT  BSS   2                   VDU I/P TASK PC RET
IPSTRT  BSS   2                   VDU I/P TASK ST RET
*
IPSTIM  BSS   2                   I/P TASK ACTIVE FLAG
OPSTIM  BSS   2                   O/P TASK ACTIVE FLAG
MOCP    BSS   2                   MCP TASK ACTIVE FLAG
MOCPSC  BSS   2                   MCP STIM COUNTER
MCPIP   BSS   2                   MICRO TO MICRO I/P MOCP FLAG
IPSF    BSS   2                   OPERATOR MOCP FLAG
        END
```

GENERAL DATA MODULE G0010D

1. DESCRIPTION

This data module contains software flags which must be initialised to +1 or -1, at system start up time.

2. IDENTIFICATION

GENERAL DATA MODULE G0010D

3. SIZE

CLASS TWO : 14 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
      TITL 'GENERAL DATA MODULE G0010D'
* GENERAL DATA MODULE IDENTIFIER
      IDT  'G0010D'
      EVEN
* CLASS TWO
      DEF  SETONE,SETNEG,SETEND,IPTINT,OPTINT,MOCINT
      DEF  IPINT,ROMRAM,VDUF,OPINT
      RORG
      PSEG
* FLAGS TO BE INITIALISED TO +1
SETONE EQU  $
IPTINT BSS  2              I/P TASK INT FLAG
OPTINT BSS  2              O/P TASK INT FLAG
MOCINT BSS  2              MOCP TASK INT FLAG
OPINT  BSS  2              VDU O/P TASK INT FLAG
IPINT  BSS  2              VDU I/P TASK INT FLAG
* FLAGS TO BE INITIALISED TO -1
SETNEG EQU  $
ROMRAM BSS  2              VDU ROM RAM MIX FLAG
VDUF   BSS  2              VDU FLAG
SETEND EQU  $
      END
```

GENERAL DATA MODULE G0011D

1. DESCRIPTION

This data module contains the Queue Handler program workspace area, input and output queues etc.

2. IDENTIFICATION

GENERAL DATA MODULE   G0011D.

3. SIZE

CLASS TWO :   72 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0011D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0011D'
       EVEN
* CLASS TWO
       DEF  NIOQHW,IPQS,OPQS,QIPN,QOPN,NEXTIP,NEXTOP
       RORG
       PSEG
NIOQHW BSS  32            QUEUE HANDLER WA
IPQS   BSS  16            I/P QUEUE
OPQS   BSS  16            O/P QUEUE
QIPN   BSS  2             QUEUE I/P VALUE
QOPN   BSS  2             QUEUE O/P VALUE
NEXTIP BSS  2             NEXT VALUE FOR I/P TASK
NEXTOP BSS  2             NEXT VALUE FOR O/P TASK
       END
```

GENERAL DATA MODULE G0012D

1. DESCRIPTION

This data module contains the Interrupt Handler workspace areas, Input/Output workspace areas, and input and output files, necessary for the number of inter-processor communications UARTs connected to the system (in this case 3).

2. IDENTIFICATION

GENERAL DATA MODULE G0012D

3. SIZE

CLASS TWO : 504 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0012D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0012D'
       EVEN
* CLASS TWO
       DEF  IHWA1,IOWA1,OPFIL1,IPFIL1,IPFI1S
       DEF  IHWA2,IOWA2,IPFIL2,IPFI2S
       DEF  IHWA3,IOWA3,IPFIL3,IPFI3S
       RORG
       PSEG
IHWA1  BSS  32                  UART CRU BASE 1880  INT 5
IOWA1  BSS  32                                ''    ''
OPFIL1 BSS  52                                ''    ''
IPFIL1 BSS  52                                ''    ''
IPFI1S EQU  IPFIL1+2                          ''    ''
*
IHWA2  BSS  32                  UART CRU BASE 1900  INT 4
IOWA2  BSS  32                                ''    ''
OPFIL2 BSS  52                                ''    ''
IPFIL2 BSS  52                                ''    ''
IPFI2S EQU  IPFIL2+2                          ''    ''
*
IHWA3  BSS  32                  UART CRU BASE 1840  INT 3
IOWA3  BSS  32                                ''    ''
OPFIL3 BSS  52                                ''    ''
IPFIL3 BSS  52                                ''    ''
IPFI3S EQU  IPFIL3+2                          ''    ''
       END
```

GENERAL DATA MODULE G0013D

1. DESCRIPTION

This data module contains the Input/Output task workspace area, and the locations used by these tasks when swopping to new workspace areas.

2. IDENTIFICATION

GENERAL DATA MODULE G0013D

3. SIZE

CLASS TWO : 40 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
       TITL 'GENERAL DATA MODULE G0013D'
* GENERAL DATA MODULE IDENTIFIER
       IDT  'G0013D'
       EVEN
* CLASS TWO
       DEF  NSPIOW,SETWP,SETPC,SETWP1,SETPC1
       RORG
       PSEG
NSPIOW BSS  32             I/P O/P TASK WA
SETWP  BSS  2              WP SETTING ADDRESS
SETPC  BSS  2              PC SETTING ADDRESS
SETWP1 BSS  2              WP1 SETTING ADDRESS
SETPC1 BSS  2              PC1 SETTING ADDRESS
       END
```

GENERAL DATA MODULE G0014D

1. DESCRIPTION

This data module contains the MOCP workspace area, and the locations used by the MOCP task to store received data.

2. IDENTIFICATION

GENERAL DATA MODULE G0014D

3. SIZE

CLASS TWO : 63 BYTES.

4. AUTHOR

B.T.V. WARTON.

```
        TITL 'GENERAL DATA MODULE G0014D'
* GENERAL DATA MODULE IDENTIFIER
        IDT  'G0014D'
        EVEN
* CLASS TWO
        DEF  MOCPTW,NDF1,NDF2,NPR,NDF11,NDF22
        DEF  NPR1,NMF1,NMF2,NMF11,NMF22
        RORG
        PSEG
MOCPTW  BSS  32                  MOCP TASK WORK SPACE
NDF1    BSS  2                   DF1 STORE
NDF2    BSS  2                   DF2 STORE
NPR     BSS  2                   PULSE RATE STORE
NMF1    BSS  2                   MF1 STORE
NMF2    BSS  2                   MF2 STORE
NDF11   BSS  4                   DF1 VDU O/P
NDF22   BSS  4                   DF2 VDU O/P
NPR1    BSS  5                   PR  VDU O/P
NMF11   BSS  4                   MF1 VDU O/P
NMF22   BSS  4                   MF2 VDU O/P
        END
```

GENERAL DATA MODULE G0015D

1. DESCRIPTION

This data module contains VDU workspace areas and buffers.

This must be the last data module in RAM as it contains the RAMEND label used by the system initialisation task.

2. IDENTIFICATION

GENERAL DATA MODULE G0015D

3. SIZE

CLASS TWO : 192 Bytes.

4. AUTHOR

B.T.V. WARTON.

```
      TITL 'GENERAL DATA MODULE G0015D'
* GENERAL DATA MODULE IDENTIFIER
      IDT  'G0015D'
      EVEN
* CLASS TWO
      DEF  VDUIOW,VDUOBA,BUFA,OPROM,OPRAM
      DEF  RAMEND,IOTF,FREE,VDUOIW
      RORG
      PSEG
VDUIOW BSS  32              VDU TASK HANDLER ENTRY WA
VDUOIW BSS  32              VDU TASK DIRECT ENTRY WA
VDUOBA BSS  40              VDU O/P BUFFER ADDRESSES
BUFA   BSS  80              I/P BUFFER
OPRAM  BSS  2               O/P RAM ADDRESS
OPROM  BSS  2               O/P ROM ADDRESS
IOTF   BSS  2               VDU TASK ACTIVE FLAG
FREE   BSS  2               FREE TIME COUNTER
RAMEND EQU  $               ****** END OF RAM *******
       END
```

APPENDIX E. PUBLISHED WORK

A DISTRIBUTED MICROPROCESSOR SYSTEM FOR THE DETECTION AND DIAGNOSIS OF CARDIAC ARRHYTHMIAS

H. A. Barker, TD, BSc, PhD, AFIMA, MInstMC, CEng, FIEE*

B. T. V. Warton, MSc, AMIEE*

Summary

The paper describes a system for the real-time monitoring of patients in coronary care units. A distributed approach allows the allocation of a microprocessor to each patient for the purposes of electrocardiogram analysis. The functions performed by these microprocessors include digital filtering of the electrocardiogram, detection and measurement of each heart beat complex, and diagnosis of arrhythmias by decision tables. The patient-connected microprocessor systems may be operated as stand-alone units, or may be connected to a further microprocessor system for the centralisation of information storage and display and operator control of the distributed system.

1. Introduction

In coronary care units (CCUs), the patients' heart rhythms are the principal indicators of their state of health. The detection and diagnosis of abnormalities in these rhythms, commonly known as arrhythmias, are therefore of prime importance in such units.

The most common method for measuring heart rhythms is by amplification of the small potential differences associated with heart action, which occur on the surface of the chest, to provide electrical signals in the form of electrocardiograms (ECGs). In the past, conventional computers have been used to assist in the detection and diagnosis of arrhythmias in ECG data (ref. 1-8). The high cost of on-line systems for this purpose, however, has precluded their widespread use in CCUs.

The advent of the microprocessor has now provided the possibility of providing a cheaper and more flexible system, not only by a direct reduction in the cost of information processing, but also by the concomitant possibility of distributing this processing throughout the system. The design and development of such a system is described here. Its object is to exploit the advantages which microprocessors can provide, and its results should benefit those whose dependence on the system may be literally a matter of life or death.

2. Analysis of system function

The overall function of the system may be sub-divided into a number of distinctly different types of function, broadly classified as follows:

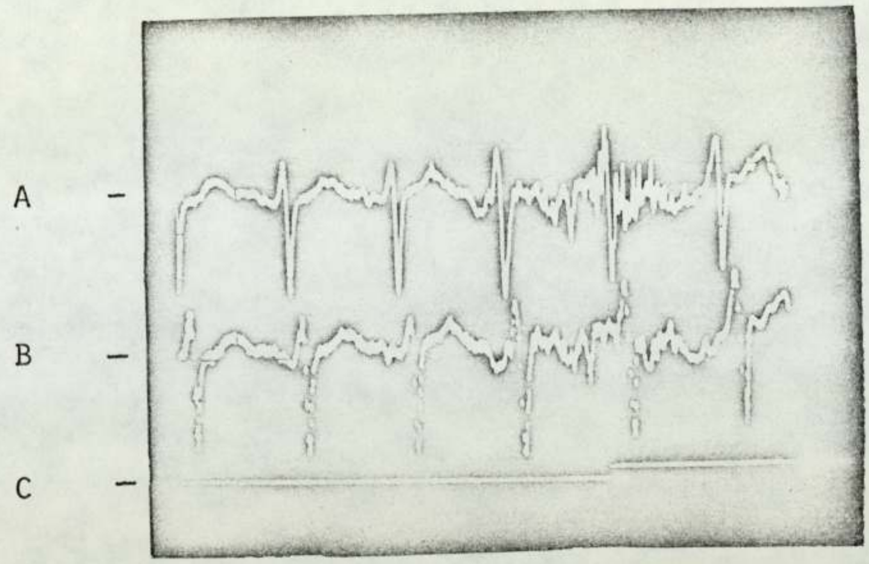
Signal acquisition: this function is concerned with the generation of a primary ECG signal (Fig. 1), by amplifying the chest electrode potentials. It is a function which is performed satisfactorily by analogue equipment in common 'bedside' use in CCUs, and will not be considered in detail here.

Signal conditioning: this function is concerned with the processing of a primary ECG signal, to obtain a conditioned signal (Fig. 1) from which measurements may be taken with an appropriate degree of confidence. The advantages of digital methods for this purpose are such that this function is best performed by a microprocessor, and the approach adopted here is described in Section 4.1.

---

* Department of Electrical Engineering, University of Aston in Birmingham

Published in IERE Conference Proceedings No. 41. Conference on Microprocessors in Automation and Communications, University of Kent at Canterbury, September 1978.



A PRIMARY ECG SIGNAL
B CONDITIONED ECG SIGNAL
C NOISE DETECTION INDICATOR

FIGURE 1 ECG SIGNALS

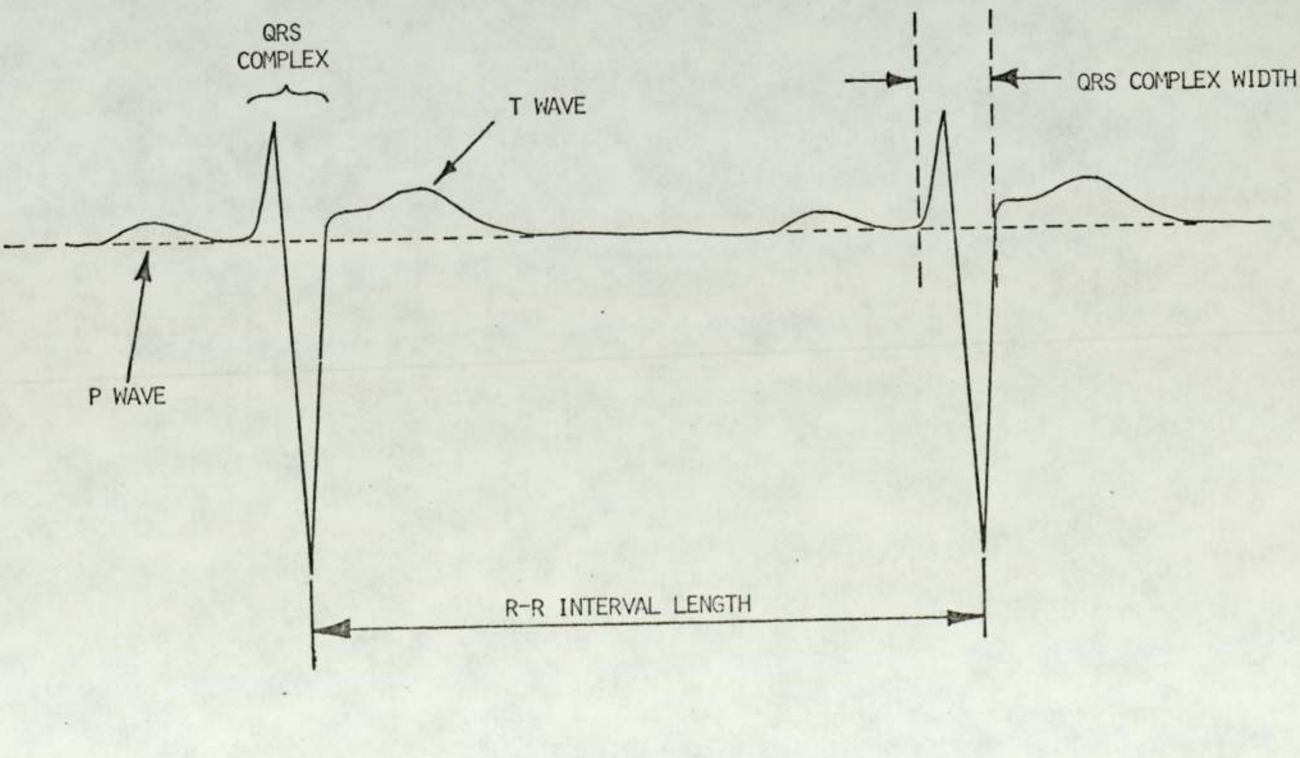


FIGURE 2 ECG CHARACTERISTICS

Signal monitoring: this function is concerned with the extraction of parameter measurements from a conditioned ECG signal. Attention here is focussed principally on the length of the R-R interval and the width of the QRS complex (Fig. 2), as described in Section 4.2.

Diagnosis: this function is concerned with the assessment of results obtained by monitoring. A decision table approach is used here, as described in Section 4.3.

Display: this function is concerned with the presentation of information, such as ECG data, parameter values, diagnoses and trends, to an operator.

Storage: this function is concerned with the retention of information concerned with display, using appropriate media.

In addition to these specific functions, the general functions of communication and control are pervasive throughout the system.

3. System structure

In order to determine a suitable structure for the system, it is necessary to examine the functions outlined in Section 2, and the information flows between them, as shown in Fig. 3. Only the signal acquisition function is of necessity distributed in the system, since this must be allocated on a one-per-patient basis. The remaining functions may be structured as required. Since the highest rates of data transfer occur permanently between the signal acquisition, conditioning and monitoring functions, the greatest benefits of distributed processing are obtained if each of these functions is allocated on a one-per-patient basis, and this arrangement is therefore adopted as the foundation of the system structure.

Within this arrangement, the allocation of processors to functions, or vice-versa, is not a procedure capable of simple or precise quantification. Some indication of an appropriate form of solution may however be obtained by assessing the degree of utilisation of a typical modern microprocessor in the performance of each function. With this approach, a particularly simple form of solution is obtained because the assessments show that, for a single patient, total performance of all the functions shown in Fig. 3 is well within the capability of a TMS 9900 16-bit microprocessor. These considerations therefore indicate that a system completely distributed on the basis of one such processor per patient is the most suitable structure for this application.

A structure in which all functions are distributed on a one-per-patient basis has the obvious advantage of flexibility, and for this reason each processor in the system is provided with the capability of operating in a stand-alone mode. There are, however, other factors which militate against complete distribution of the display and storage functions; these include the cost of peripheral devices associated with these functions and the management requirement for centralisation of the facilities which these functions provide.

In the final system structure, therefore, all functions, except those concerned with centralised display and storage, are distributed to patient-dedicated processor systems. These systems, although capable of stand-alone operation if required, normally perform only those functions upto and including diagnosis, and the information obtained from their operation is communicated to the centralised display and storage facilities under the control of an operator-dedicated processor system (Fig. 4).

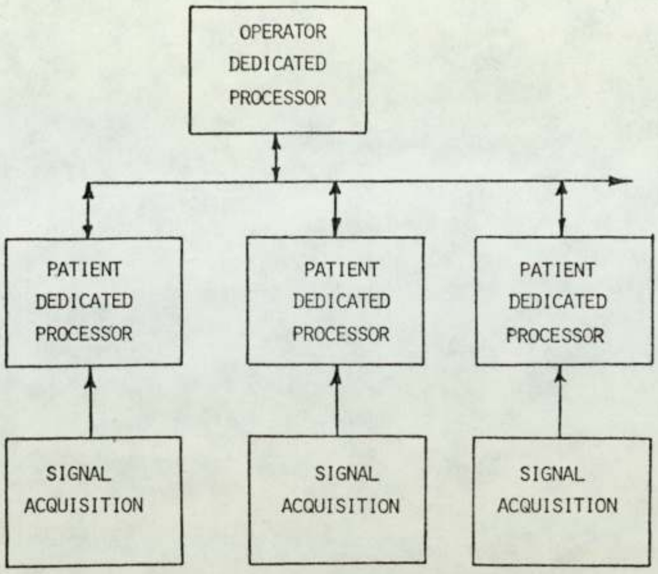


FIGURE 4 STRUCTURE OF DISTRIBUTED SYSTEM

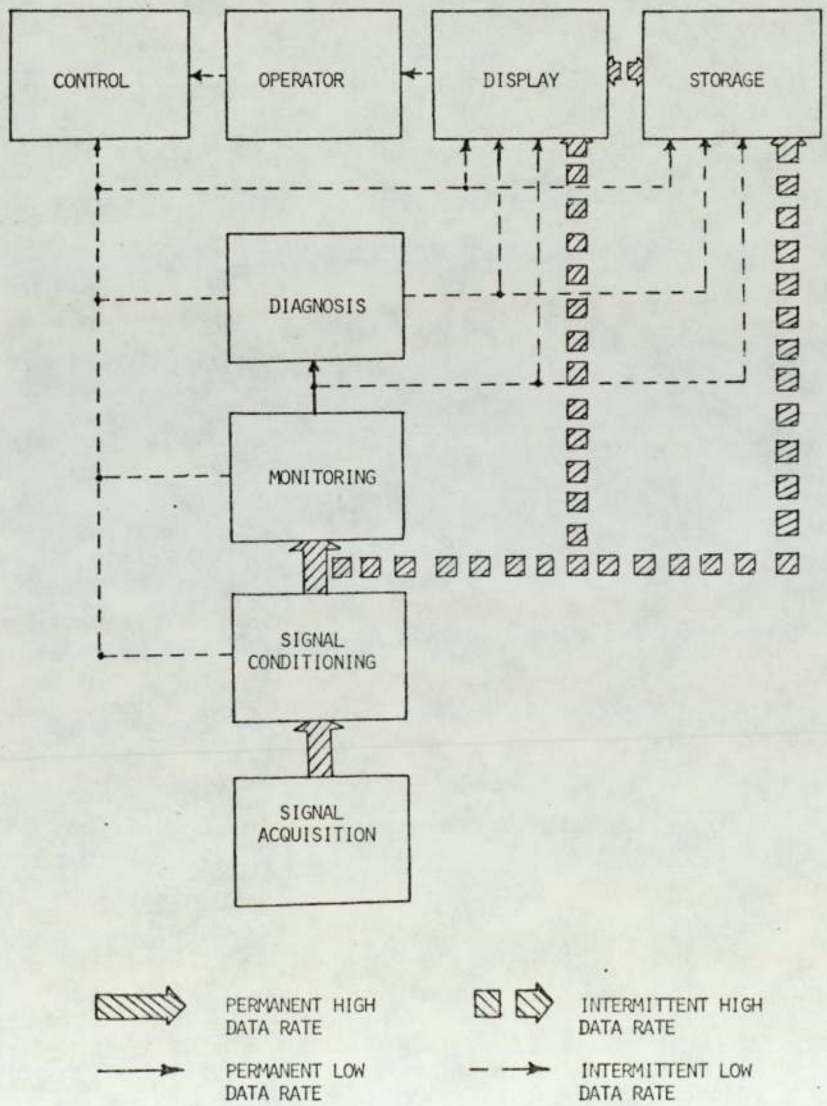


FIGURE 3 SYSTEM FUNCTIONS AND INFORMATION FLOWS

4. Patient-dedicated processor system design

Each patient-dedicated processor system performs the functions of signal conditioning, monitoring, diagnosis and communication, under the control of an operating system in the form of a real-time scheduler. The only function in any way dependent on the structure of the remainder of the system is that of communication; this is normally concerned with inter-processor communication, but must be concerned with operator communication when in the stand-alone mode.

4.1 Signal conditioning

The purpose of signal conditioning is to improve the consistency of subsequent signal monitoring, by the removal of very low frequency components (d.c. and base-line drift), and the reduction of intermittent high frequency components (artifact), from the primary ECG signal. A sketch of the normal ECG spectrum, as obtained by Golden et. al. (ref. 9) is shown in Fig. 5.

The required signal conditioning is accomplished by an adaptive digital filtering algorithm, applied to samples of the primary ECG signal which are obtained, as recommended by Wartak et. al. (ref. 10), at a rate of 250 Hz with 8-bit resolution. A block diagram of the digital filter is shown in Fig. 6. The d.c. and base-line drift is removed by a second-order high-pass filter with 0.5 Hz cut-off frequency, and the artifact is reduced by an adaptive arrangement in which the parameter settings have been determined experimentally. The presence of artifact is detected when the output of a second-order high-pass filter with 60 Hz cut-off frequency exceeds a threshold, in which case the output of a second-order low-pass filter with 25 Hz cut-off frequency is used in preference to an unfiltered signal. This approach allows the detection of artifact to be indicated (Fig. 1) through the diagnosis function to the operator, to show in a simple fashion that a temporary decrease in diagnosis confidence might be expected.

The delays of N and n samples, shown in Fig. 6, are chosen to equalise the total delays through the 25 Hz low-pass filter and the direct path, and also to ensure that the filtered signal is introduced just prior to the occurence of the artifact to be reduced. The delayed hold ensures that the low-pass filter remains in continuous operation when rapid bursts of artifact are present. The filtering algorithm execution time is 1 ms, which is well within the 4 ms sampling period. A short execution time, together with a relatively simple programme, are due mainly to the availability of a single instruction for a 16-bit multiplication operation with the TMS 9900 microprocessor.

4.2 Monitoring

Measurements are performed on the conditioned ECG signal after the detection of each QRS complex (Fig. 2), which is accomplished by a delayed-difference threshold method. A signal is obtained as the difference between samples which are separated by 6 sample periods (24 ms), and compared with a threshold which is 60% of the average minimal negative value of the signal (Fig. 7), as suggested by Van Eyll et.al. (ref. 11). The measurements which are then performed are those necessary for the subsequent diagnosis of certain arrhythmias, using the method of Rabin et.al. (ref. 8). Although their particular approach has been adopted here, the scope of the measurements could easily be extended to provide the data for any diagnostic method.

In this case, the primary measurements are the length of the R-R interval and the width of the QRS complex (Fig. 2). The R-R interval length is compared with the average R-R interval, relative to which it is classified

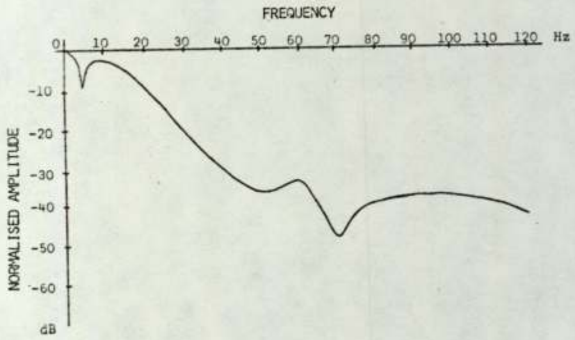


FIGURE 5 ECG SPECTRUM

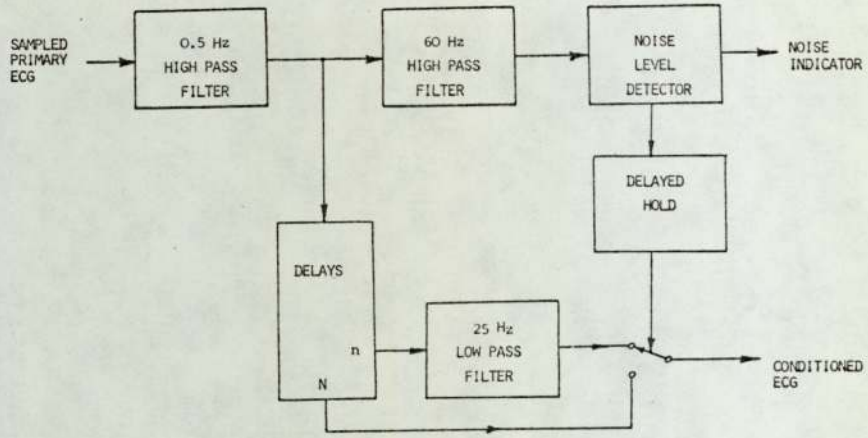


FIGURE 6 BLOCK DIAGRAM OF ADAPTIVE DIGITAL FILTER

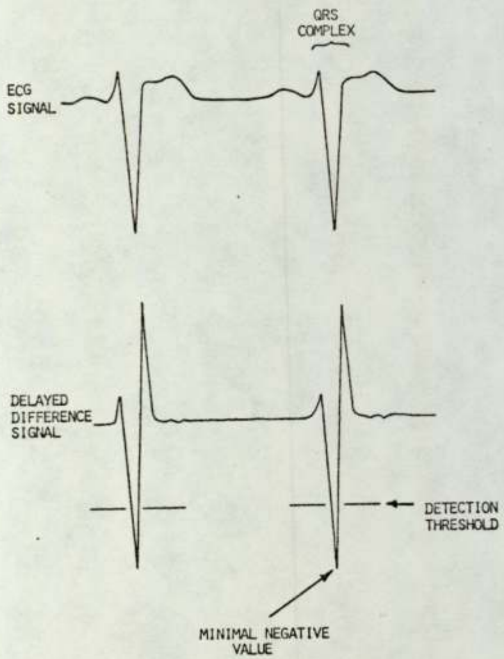


FIGURE 7 DELAYED-DIFFERENCE DETECTION OF QRS COMPLEX

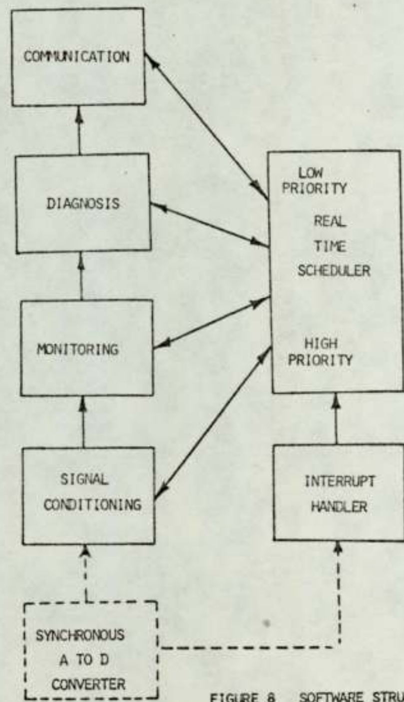


FIGURE 8 SOFTWARE STRUCTURE OF PATIENT DEDICATED PROCESSOR

as normal (N), short (S) or long (L). Normal R-R interval lengths are used to update the R-R interval average, which is compared with standard intervals corresponding to various pulse rates. The QRS complex width is compared with a standard interval of 120 ms, relative to which it is classified as wide (WQRS) or normal.

The measurements are used, together with previous measurements, in a number of tests, as shown in Table 1. Each test has either a true or false result, corresponding to which bits are set or reset in two 16-bit words for subsequent use in the diagnosis function. A shorthand notation is used to specify certain sequence tests; for example, a sequence consisting of an R-R interval which is short, followed by a QRS complex which is wide, followed by an R-R interval which is not short, all repeated 4 times, is written $4[S \rightarrow WQRS \rightarrow \bar{S}]$.

4.3 Diagnosis

Diagnosis is concerned with the recognition of specific arrhythmia conditions from the bit patterns of the words obtained by monitoring. The approach adopted here is to use decision tables (ref. 12), and an example of one of the tables used is shown in Table 2. The actual testing of bit patterns in a word is very simple with the TMS 9900 microprocessor, because of the availability of a single instruction (COC) for testing a word for a pattern of bits required to be set, and a single instruction (CZC) for testing a word for a pattern of bits required to be reset. To diagnose Bradycardia, a slow pulse rate condition, for example, the decision table in Table 2 defines the pattern required and it is only necessary to compare the first monitored word in Table 1 with CO_{16}, using the COC instruction, and with $3F_{16}$, using the CZC instruction, and if both tests are satisfied the diagnosis is established. On completion of a diagnosis such as this, other decision tables are then used to test for other possible diagnoses.

4.4 Communication

The communication function is concerned with the transfer of the information shown in Fig. 3, either to the operator-dedicated processor system in normal operation, or to the display and storage facilities directly in the stand-alone mode. In both cases an asynchronous serial data channel is used, and the receiver-transmitter is implemented by a dedicated TMS 9902 asynchronous communications controller specifically designed for use with the TMS 9900 microprocessor.

4.5 Control

All patient-dedicated functions operate under the control of a real-time scheduler. The operating priority of a function is the inverse of its position in the system hierarchy, and the signal conditioning function, which occupies the lowest position in the system hierarchy, therefore has the highest operating priority. This function is activated by an interrupt from the analogue-digital converter as each ECG sample is obtained, and each remaining function is then activated by the next highest priority function as the data flows through the system (Fig. 8). The real-time scheduler, however, suspends the operation of lower priority functions when a higher priority function is active, so each higher priority function is allowed to run to completion before the lower priority function is commenced.

EVENT	WORD 1 SET BIT
PULSE RATE IS GREATER THAN 140 BEATS PER MINUTE	0
PULSE RATE IS GREATER THAN 110 BEATS PER MINUTE	1
PULSE RATE IS GREATER THAN 100 BEATS PER MINUTE	2
PULSE RATE IS GREATER THAN 90 BEATS PER MINUTE	3
PULSE RATE IS GREATER THAN 80 BEATS PER MINUTE	4
PULSE RATE IS GREATER THAN 60 BEATS PER MINUTE	5
PULSE RATE IS GREATER THAN 40 BEATS PER MINUTE	6
PULSE RATE IS GREATER THAN 20 BEATS PER MINUTE	7
	8
A QRS COMPLEX HAS OCCURRED DURING THE LAST 6 SECONDS	9
CURRENT R-R INTERVAL IS NOT LESS THAN TWICE THE AVERAGE	10
CURRENT R-R INTERVAL IS LONG	11
CURRENT R-R INTERVAL IS SHORT	12
CURRENT QRS COMPLEX IS WIDE	13
MORE THAN 82% OF PREVIOUS QRS COMPLEXES ARE WIDE	14
	15

EVENT	WORD 2 SET BIT
A SEQUENCE $4[S \rightarrow \bar{S}]$ HAS OCCURRED	0
A SEQUENCE $4[S \rightarrow WQRS \rightarrow \bar{S}]$ HAS OCCURRED	1
A SEQUENCE. $4[S \rightarrow \bar{S} \rightarrow N]$ HAS OCCURRED	2
A SEQUENCE $4[S \rightarrow WQRS \rightarrow \bar{S} \rightarrow N]$ HAS OCCURRED	3
A SEQUENCE $S \rightarrow \bar{S} \rightarrow S$ HAS OCCURRED	4
A SEQUENCE $S \rightarrow \bar{S} \rightarrow N \rightarrow \bar{L}$ HAS OCCURRED	5
A SEQUENCE $3[S] \rightarrow WQRS$ HAS OCCURRED	6
A SEQUENCE $S \rightarrow \bar{S} \rightarrow L$ HAS OCCURRED	7
A SEQUENCE $S \rightarrow \bar{S} \rightarrow N \rightarrow L$ HAS OCCURRED	8
	9
	10
	11
	12
	13
	14
	15

TABLE 1 TESTS PERFORMED IN ECG MONITORING

EVENT	WORD 1 BIT	TRUE FALSE				ELSE
PULSE RATE 140	0	F	F	F	F	-
PULSE RATE 110	1	F	F	T	F	-
PULSE RATE 80	4	F	F	T	T	-
PULSE RATE 60	5	F	F	T	T	-
PULSE RATE 40	6	T	F	T	T	-
PULSE RATE 20	7	T	T	T	T	-
QRS IN LAST 6 SECONDS	9	-	-	-	F	-
BRADYCARDIA		X				
IDIOVENTRICULAR RHYTHM			X			
SUPRAVENTRICULAR RHYTHM				X		
LOST SIGNAL					X	
NEXT TABLE		X	X	X	X	X

TABLE 2 EXAMPLE OF DECISION TABLE DIAGNOSIS

5. Operator-dedicated processor system design

When the display and storage functions are centralised, an operator-dedicated processor system is required to act as a centre for communication and control between the patient-dedicated processor systems, and the display and storage facilities. As the majority of processing in the system is distributed, the data flows in the operator-dedicated system are low and consist mainly of routine information such as pulse rates and arrhythmia diagnoses. During normal operation, this information is simply repeated by the operator-dedicated system to a visual display unit, where it may be viewed by an operator.

There is, however, the requirement for information storage in the system, and although this function has not yet been implemented, its general characteristics may be described. A particular requirement is for the storage of those parts of the ECG signals which contain significant arrhythmia events. The data rates required for this are somewhat higher than those concerned with routine information, but the occurrance of such events is normally intermittent. Therefore a serial data transmission channel to the appropriate magnetic media, under the control of the operator-dedicated system, will be a suitable implementation for this function.

When the storage function is implemented, the control function will be extended to allow the recall of stored data to the visual display unit by an operator. At this stage it may be necessary to re-appraise the method of system management by the operator-dedicated system. It is not, however, envisaged that more than one TMS 9900 microprocessor will be required in the operator-dedicated system to service all the patient-dedicated systems in a normal CCU.

6. Conclusions

In an on-line system for the detection and diagnosis of arrhythmias in the ECGs of patients in CCUs, the use of microprocessors allows the data processing to be distributed as required. Examination of the functions and information flows within the system, in the context of modern microprocessor capability, indicates that in the preferred system structure the majority of functions are allocated to patient-dedicated 16-bit microprocessor systems, with the display and storage functions centralised in an operator-dedicated 16-bit microprocessor system.

The design of a system with this structure has been described in this paper. Methods for realising each function have been examined, and the nature of a microprocessor implementation discussed. In particular, the advantages of the powerful instruction set associated with a 16-bit microprocessor has been demonstrated.

The system is shown to be flexible, not only in respect of the hardware configuration, which allows each patient-dedicated system to operate on a stand-alone basis if required, but also in respect of the software functions. Although a particular diagnostic method has been adopted in the system described, the implementation of other diagnostic methods, involving further monitoring measurements, could be implemented in new software fairly easily.

7. Acknowledgments

The authors wish to acknowledge the advice on matters of a medical nature provided by Dr. R. E. Nagle, of the Queen Elizabeth Hospital, Birmingham, and Dr. A. J. Camm, of St. Bartholomew's Hospital, London.

Mr. Warton also wishes to acknowledge the award of a Research Studentship provided by the Science Research Council.

8. References

1. Hulting. J., and Nygards. M. E. Evaluation of a computer-based system for detecting ventricular arrhythmias. Acta. Med. Scand.Vol. 199, pp. 53-60, 1976.

2. Frost. D. A., Yanowitz. F. G., and Pryor. T. A. Evaluation of a computerized arrhythmia alarm system. Am. J. Cardiology, Vol. 39(4), pp. 583-587, 1977.

3. Burton. C. E., Portnoy. W. M., and Dirilten. M. An algorithm for on-line real-time computer detection of ECG changes. Int. J. Bio-Medical Computing (6), pp. 23-32, 1975.

4. Rabin. S. T., Haring. O. M., Lewis. F. J., Quinn. M., and Van Kirk. D. Digital Computer diagnosis of cardiac arrhythmias in a single-lead electrocardiogram. Int. J. Engng. Sci., Vol. 11, pp. 701-716, 1973.

5. Willems. J. L., and Pipberger. V. Arrhythmia detection by digital computer. Computers and Biomedical Research 5, pp. 263-278, 1972.

6. Haywood. L. J., Murthy. V. K., Harvey. G. A., and Saltzberg. S. On-line real-time computer algorithm for monitoring the ECG waveform. Computers and Biomedical Research 3, pp. 15-25, 1970.

7. Lewis. F. J., Dellor. S., Quinn. M., Lee. B., Will. R., and Raines. J. Continuous patient monitoring with a small digital computer. Computers and Biomedical Research 5, pp. 411-428, 1972.

8. Rabin. S. T., Haring. O. M., Lewis. F. J., Quinn. M., Dellar. S., and Van Kirk. D. A real-time computer system for the diagnosis of cardiac arrhythmias. Proc. San Diego. Biomed. Symp.,pp. 125-128, 1970.

9. Golden. D. P., Wolthuis. R. A., and Hoffler. G. W. A spectral analysis of the Normal Resting Electrocardiogram. IEEE Trans. on Biomed. Eng., pp. 366-372, Sept. 1973.

10. Wartak. J., Milliken. J. A., and Lywood. D. W. Theoretical and Practical aspects of analog-to-digital conversion of the electrocardiogram. Medical Research Engineering 9, pp. 21-23, 1970

11. Van Eyll. C ., Sterepenne. W., Lefevre. J., Bachy. J. L., Cosyrs. C. J., and Charlier. A. A. Optimal values for the Parameters of an on-line algorithm monitoring the QRS waveform. Meth. Inform. Med., Vol. 14 (4), pp. 202-207, Oct. 1975.

12. Humby. E. Programs from decision tables. Computer Monographs (19), Macdonald/American Elsevier. 1973.

REFERENCES

1. Romhilt. D.W., Bloomfield S.S., Chou. T.C., and Fowler. N.O. Unreliability of Conventional Electrocardiographic Monitoring for Arrhythmia Detection in Coronary Care Units. American Journal of Cardiology Vol. 31, pp. 457-461, April 1973.

2. Owen. S.G. Electrocardiography. The English University Press Ltd. 1973.

3. Lipman. B.S., Massie. E., Kleiger. R.E. Clinical Scalar Electrocardiography. Year Book Medical Publishers Incorporated 1973, 6th Edition.

4. Mangiola. S., Ritota. M.C. Cardiac Arrhythmias. Practical ECG Interpretation. J.B. Lippincott Company, 1974.

5. Rabin. S.T., Haring. O.M., Lewis. F.J., Quinn. M., Dellar. S., and Van Kirk. D. A Real-Time Computer System for the Diagnosis of Cardiac Arrhythmias. Proc. San Diego. Biomed. Symp., pp. 125-128, 1970.

6. Neilson. J.M., and Vellani. C.W. Computer Detection and Analysis of Ventricular Ectopic Rhythms. Quantitation in Cardiology. Leiden University Press, pp. 117-125, 1972.

7. Neilson. J.M. A Special Purpose Hybrid Computer For Analysis of ECG Arrhythmias. Computers for Analysis and Control in Medical and Biological Research. IEE Conference Publication No. 79. pp. 151-156, 1971.

8. Neilson. J.M. High Speed Analysis of Ventricular Arrhythmias from 24 hour recordings. Computers in Cardiology. IEE Catalog. No. 74CH0879-7C. pp. 55-59, 1974.

9. Neilson. J.M. An Adaptive Arrhythmia Monitoring. Computers in Cardiology. IEE Catalog. No. 74CH0879-7C. pp. 211-212, 1974.

10. Balm. G.J. Crosscorrelation Techniques applied to The Electrocardiogram Interpretation Problem. IEEE Transaction on Bio-medical Engineering. Vol. BME-14, No. 4., pp. 258-262, October 1967.

11. Haisty. W.K., Batchlor. C., Cornfield. J., and Pipberger. H.V. Discriminant Function Analysis of R-R Intervals: An Algorithm for On-line Arrhythmia Diagnosis. Computers and Biomedical Research 5, pp. 247-255, 1972.

12. Fieldman. C.L., Amazeen. P.G., Klien. M.D., and Lown. B. Computer Detection of Ventricular Ectopic Beats. Computers and Biomedical Research 3, pp. 666-674, 1971.

13. Wigertz. O., Blomqvist. P., Hulting. J., Matell. G., and Nygards. M.E. Evaluation and Further Development of a Computer-based System for Arrhythmia Detection. Proc. Conf. on Computers in Cardiology. pp. 215-216, 1974.

14. Arnold. J.M. Computavien II. An Advanced Approach to Arrhythmia Monitoring. Computers in Cardiology. IEE Catalog. No. 74CH0879-7C. pp. 231-232, 1974.

15. Specht. D.F. Gould 9500 Computerized Monitoring System. Computers in Cardiology. IEE Catalog. No. 74CH0879-7C, pp. 223-224, 1974.

16. Burton. C.E., Portnoy. W.M., and Dirilten. H. An Algorithm for On-line, Real-Time Computer Detection of ECG Changes. International Journal of Bio-Medical Computing 6, pp. 23-32, 1975.

17. Gerlings. E.D., Bower. D.L., and Rol. G.A. Detection of Abnormal Ventricular Activation in a Coronary Care Unit. Computers and Biomedical Research 5, pp. 14-24, 1972.

18. Frankel. P., Rothmeier. J., James. D., and Quaynor. N. A Computerized System for ECG Monitoring. Computers and Biomedical Research 8. pp. 560-567, 1975.

19. Fozzard. H. Computer Handling of Coronary Care Unit Data. Symposium on Coronary Heart Disease, Medical Clinics of North America. Vol. 57, No. 1, pp. 143-154, 1973.

20. Haywood. L.J., Harvey. G.A., and Kirk. W.L. On-line Digital Computer for Electrocardiogram Monitoring in a Coronary Care Unit. Journal of the Association for the Advancement of Medical Instrumentation. Vol. 3., No. 5. pp. 165-169, 1969.

21. Geddes. J.S., Warner. H.R. A PVC Detection Program. Computers and Biomedical Research 4, pp. 493-508, 1971.

22. Rabin. S.T., Haring. O.M., Lewis. F.J., Quinn. M., and Van Kirk. D. Digital Computer Diagnosis of Cardiac Arrhythmias in a single-lead electrocardiogram. Int. J. Engng. Sci., Vol. 11, pp. 701-716, 1973.

23. MacFarlane. P.W., and Lawrie. T.D.V. An Introduction to Automated Electrocardiogram Interpretation. Computers in Medicine Series. Butterworth Co. (Pub.) Ltd., 1974.

24. Yokoi. M., Watanabe. Y., Okamoto. N., Yasui. S., and Mizuno. Y. On-line Computer Diagnosis of Arrhythmias on ECG Using by Small Scale Digital Computer System. Japanese Circulation Journal, Vol. 33. pp. 129-138, 1969.

25. Oliver. G.C., Nolle. F.M., Wolff. G.A., Cox. J.R., and Ambos. H.D. Detection of Premature Ventricular Contractions with a Clinical System for Monitoring Electrocardiographic Rhythms. Computers and Biomedical Research 4, pp. 523-541, 1971.

26. Miller. A.C., Hoare. M.R., Rey. W., Laird. J.D., Arntzenius. A.C. and Hugenholtz. P.G. Two approaches to Arrhythmia Detection by Digital Computer in the Coronary Care Unit. Folia. Med. Neerl., 14. pp. 209-217, 1971.

27. Lewis. F.J., Deller. S., Quinn. M., Lee. B., Will. R., and Raines. J. Continuous Patient Monitoring with a Small Digital Computer. Computers and Biomedical Research 5, pp. 411-428, 1972.

28. Haywood. L.J., Murthy. V.K., Harvey. G.A. and Saltzberg. S. On-line Real time Computer Algorithm for Monitoring the ECG Waveform. Computers and Biomedical Research 3. pp. 15-25, 1970.

29. Harris. G.J. Cams-282. Computer Assisted Monitoring System. Proc. Conf. on Computers in Cardiology. pp. 225-227, 1974.

30. Dillman. R. The Hewlett-Packard 78220A Arrhythmia Monitoring System. Proc. Conf. on Computers in Cardiology. pp. 229-230, 1974.

31. Pryor. T.A. In-patient Arrhythmia Monitoring. Proc. Conf. on Computers in Cardiology. pp. 205-206, 1974.

32. Zeelenberg. C., Hoare. M.R., Engelse. W.A.H., Hagemiger. F., Hugenholtz. P.G. Arrhythmia Monitoring at the Thoraxcentrum, Rotterdam. Proc. Conf. on Computers in Cardiology. pp. 203-204, 1974.

33. Nolle. F.M., Cox. J.R. The Argus Arrhythmia Guard System. Proc. Conf. on Computers in Cardiology. pp. 201-202, 1974.

34. Sanders. W., Alderman. E., Tecklenberg. P., Harrison. D.C. The Stanford Computer-based Arrhythmia Monitoring System. Proc. Conf. on Computers in Cardiology. pp. 199-200, 1974.

35. Willems. J.L., and Pipberger. H.V. Arrhythmia Detection by Digital Computer. Computers and Biomedical Research 5, pp. 263-278, 1972.

36. Frost. D.A., Yarowitz. F.G., Pryor. T.A. Evaluation of a Computerized Arrhythmia Alarm System. Am. J. Cardiology, Vol. 39(4), pp. 583-587, 1977.

37. Hulting. J., and Nygards. M.E. Evaluation of a Computer-based System for Detecting Ventricular Arrhythmias. Acta. Med. Scand. Vol. 199, pp. 53-60, 1976.

38. Harrison. D.C., Sanders. W., Tecklenberg. P., and Alderman. E.L. State of the Art of Automated Arrhythmia Detectors - commercial systems. Computers in Cardiology National Institute of Health. pp. 11-14, 1974.

39. Wartak. J., Milliken. J.A., and Lywood. D.W. Theoretical and Practical Aspects of Anolog-to-digital Conversion of the Electrocardiogram. Medical Research Engineering 9, pp. 21-23, 1970.

40. Van Eyll. C., Strepenne. W., Lefevre. J., Bachy. J.L., Cosyns. C.J. and Charlier. A.A. Optimal Values for the Parameters of an On-line Algorithm Monitoring the QRS Waveform. Inform. Med., Vol. 14(4), pp. 202-207, Oct. 1975.

41. Fozzard. H., Kinias. P. Computers for recognition and Management of Arrhythmias. Symposium on Cardiac Rhythm Disturbances II. Medical Clinic of North America. Vol. 60. No. 2., pp. 291-298, 1976.

42. Hewlett-Packard Company publication. A Compendium on Automated Arrhythmia Detection. 5952-5333, 1976.

43. Kyle. D.G., Portnoy. W.M., and Burton C.E. A Recognition System for the Detection of Cardiac Arrhythmias. Proceedings of the Conference on the Application of Electronics in Medicine. IERE. pp. 223-228, 1976.

44. Diprose. M., Fitzgerald. M., Hanna. F.K., and Taylor. D.J.E. The Use of a Microprocessor in Routine Cardiac Assessment. IERE Conference on programmable instruments. pp. 195-202, 1976.

45. Keane. B., Dismukes. J., Singh. G., Bisehof. G. Arrhythmia Detection using a Low-power CMOS processor. Proceedings of Southeastcon 78, Region 3 Conference IEEE. pp. 19-20, 1978.

46. Cox. J.R., Murray. T.M. The Design and Development of a Microprocessor based system for ECG Analysis. Proceedings of Southeastcon 78, Region 3 Conference IEEE. pp. 538-540, 1978.

47. Sanders. C., Murray. T.M., Jones. W.T. The Development of a Real-time ECG Contour Analysis Program Utilising a Microprocessor System. Proceedings of Southeastcon 78, Region 3, Conference IEEE. pp. 541-544, 1978.

48. Rader. C.M., and Gold. B. Digital Filter Design Techniques in the Frequency Domain. Proc. IEEE. Vol. 55, No. 2, pp. 149-171, Feb. 1967.

49. Gold. B., and Rader. C.M. Digital Processing of Signals. McGraw-Hill 1969.

50. Temes. G.C., and Mitra. S.K. Modern Filter Theory and Design. John Wiley & Son, 1973.

51. Daniels. R.W. Approximation methods for Electronic Filter Design. McGraw-Hill, 1974.

52. Huelsman. L.P. Active Filters. McGraw-Hill, 1970.

53. Ackroyd. M.H. Digital Filters. Computers in Medicine Series. Butterworth, 1973.

54. Humby. E. Programs from Decision Tables. Computer Monographs (19), MacDonald/American Elsevier, 1973.

55. Pollack. S.L. Analysis of the Decision Rules in Decision Tables. U.S. Air Force Project RAND. RM-3669-PR. May, 1963.

56. Colin. A.J.T. Introduction to Operating Systems. Computer Monographs (17), MacDonald/American Elsevier, 1971.

57. Golden. D.P., Wolthuis. R.A., and Hoffler. G.W. A Spectral Analysis of the Normal Resting Electrocardiogram. IEEE Trans. on Biomedical Eng. pp. 366-372, Sept. 1973.

58. Texas Instruments. Assembly Language Programmer's Guide. ISBN. 0-904047-17-2.

59. Mears. F.C. Communication between Microprocessors. Seminar on "Microprocessorbased Systems - Construction and Development". Oct. 1975.

60. Hebditch. D.L. Data Communications, an Introductory Guide. Pub. Unwin Brothers Ltd., 1975.

61. Abramson. N., Kuo. F.F. Computer-Communication Networks. Prentice-Hall Inc., 1973.

62. Davies. D.W., Barber. D.L.A. Communication Networks for Computers. John Wiley and Son, 1973.