

To the Graduate Council:

I am submitting herewith a thesis written by Richard Odell Eason entitled "Development and Comparison of Algorithms for Generating a Scan Sequence for a Random Access Scanner." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Engineering, with a major in Electrical Engineering.

Donald W. Bouldin

Donald W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance:

Michael G. Thomason

R. C. Gonzalez

Accepted for the Council:

L. E. Evans

Vice Chancellor
Graduate Studies and Research

Thesis
80
E 386
cop. 2

DEVELOPMENT AND COMPARISON
OF ALGORITHMS FOR GENERATING A SCAN SEQUENCE
FOR A RANDOM ACCESS SCANNER

A Thesis
Presented for the
Master of Engineering
Degree
The University of Tennessee, Knoxville

Richard Odell Eason
June 1980

3041750

ACKNOWLEDGMENTS

The author is very grateful for the invaluable guidance, patience and helpful advice of Dr. Donald W. Bouldin, Committee chairman, as well as committee members Dr. R. C. Gonzalez and Dr. Michael G. Thomason. Appreciation is also extended to Mr. J. M. Jansen of Oak Ridge National Laboratory for his direction and advice.

Further appreciation is expressed to my parents for their support and encouragement throughout my education and to my wife, Mary Anne, for her patience and understanding during the time spent on this study.

The author also wishes to express his thanks to Amanda DeBord and Pamela Hrushanyk for their typing assistance.

The research for this project was sponsored by the United States Department of Energy under Contract No. W-7405-eng-26 with the Union Carbide Corporation.

ABSTRACT

Many data acquisition systems incorporate high speed scanners to convert analog signals into digital format for further processing. Some systems multiplex many channels into a single scanner. A random access scanner whose scan sequence is specified by a table in random access memory will permit different scan rates on different channels. Generation of this scan table can be a tedious manual task when there are a large number of channels (e.g. 50), more than a few scan rates (e.g. 5), and/or the ratio of the highest scan rate to the lowest scan rate becomes large (e.g. 100:1). This study develops an algorithm which will generate these scan sequences for the random access scanner and implements the algorithm on a digital computer.

Application of number theory to the mathematical statement of the problem led to development of several algorithms which were implemented in FORTRAN. The most efficient of these algorithms operate by partitioning the problem into a set of subproblems. Through recursion they solve each subproblem by partitioning it repeatedly into even smaller parts, continuing until a set of simple problems is created. From this process, a pictorial representation or "wheel diagram" of the problem can be constructed. From the wheel diagram and a description of the original problem, a scan table can be constructed. In addition, the wheel diagram can be used as a method of storing the scan sequence in a smaller amount of memory.

The most efficient partitioning algorithm solved most scan table problems in less than a second of CPU time. Some types of problems, however, required as much as a few minutes of CPU time.

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
II. THE SCANNING PROBLEM	3
III. MATHEMATICAL FORMULATION OF THE PROBLEM	9
Initial Computations	11
A Mathematical Formulation	19
Partitioning the Problem - The Wheel Approach	28
IV. ALGORITHM DEVELOPMENT	48
Algorithm Based on First Occurrence of Each Channel	48
ZAP - A Wheel Approach	55
ZIP	79
SCAN	82
V. RESULTS AND COMPARISON OF ALGORITHMS	85
VI. CONCLUSIONS	92
LIST OF REFERENCES	94
APPENDICES	96
APPENDIX I. PROGRAM LISTING	97
APPENDIX II. SAMPLE RUNS OF ZAP2	144
APPENDIX III. FILES USED IN TESTING THE ALGORITHMS	155
VITA	163

LIST OF TABLES

TABLE		PAGE
3.1	Condensed Scan Table for Example 1	22
3.2	Condensed Scan Table for Example 2	25

LIST OF FIGURES

FIGURE	PAGE
2.1 A Simple Scan Table	5
2.2 Scan Tables Containing Dummy Channels	7
3.1 A Scan Table for Example 1	23
3.2 Equation 6 Applied to All Channel Pairs for Example 1	24
3.3 A Scan Table for Example 2	26
3.4 Application of Equation 6 to Channel Pairs of Example 2 . . .	27
3.5 Dividing a Scan Table for Example 1 into Smaller Tables . . .	30
3.6 Mathematics for Partitioning a Problem	33
3.7 A Simpler Representation of Figure 3.5b	36
3.8 The First Stage in Forming a Wheel Diagram for Example 1 . .	37
3.9 An Almost-complete Wheel Diagram for Example 1	38
3.10 The Complete Wheel Diagram for Example 1	41
3.11 Storing a Scan Sequence in Wheel Form	44
3.12 Reducing Dummy Wheels	47
4.1 Flowchart for Computing Starting Locations	49
4.2 Sample Run of Example 2 Using the First Location Algorithm .	51
4.3 Basic Program Flow for the Wheel Approaches	56
4.4 The SOLVE Routine for ZAP	58
4.5 Wheel Form of Solution to the Fourth Subproblem	69
4.6 Wheel Form of Solution to the Fifth Subproblem	70
4.7 Solution of the Subproblems Combined to Form a Solution of the Original Problem	71
4.8 Wheel Form Corresponding to the Given String Form	75
4.9 The SOLVE Routine for ZIP	80

FIGURE	PAGE
4.10 The SOLVE Routine for SCAN	83
5.1 Files Used in Comparison	86
5.2 Comparison of Execution Times on Some of the Files	87

CHAPTER I

INTRODUCTION

In many data acquisition systems, there exists a requirement for scanning a large number of analog input channels at varying rates. Most systems employ a single high-speed analog scanner for conversion of analog signals to digital format. All of the channels are multiplexed into this single scanner. Many scanning systems are equipped with a sequential access scanning mode which permits all signals to be scanned at the same rate. Other systems are equipped with a random access scanner whose channel scan sequence is specified by a table in random access memory. This table scanning method provides a means for permitting different scan rates on different channels. However, as the number of channels becomes large (e.g. 50), the number of scan rates are more than a few (e.g. 5), and/or the ratio of the highest scan rate to the lowest scan rate becomes large (e.g. 100:1), then the scan table becomes quite large and unmanageable. Producing this scan table can often be a tedious manual task. As requirements change, a new scan table must again be developed. Hence, the development of an automated procedure for "on the fly" scan table production for the random access scanner becomes an attractive method if not a necessity. The recovery of the information also depends on utilization of the scan table and an appropriate algorithm to retrieve the data for a specific channel.

This study develops the mathematics for representation of the scan table and develops and implements in FORTRAN an algorithm which generates

these scan tables given the scan frequency of each channel. A machine search of the literature was performed and no other work on this subject was found.

Chapter II contains a background of scan table fundamentals and defines a few basic scan table parameters. Chapter III describes two mathematical formulations of the scan table problem. Chapter IV covers development of various algorithms based on the mathematics presented in Chapter III. In Chapter V the algorithms are compared and a brief history of their development is given. Conclusions are given in Chapter VI.

CHAPTER II

THE SCANNING PROBLEM

As an example of a scan table problem, consider the following. Seven channels A_1 , B_1 , B_2 , C_1 , C_2 , D_1 , and D_2 are required to be scanned at the following rates.

<u>Channel</u>	<u>Frequency</u>
A_1	6Hz
B_1	3Hz
B_2	3Hz
C_1	2Hz
C_2	2Hz
D_1	1Hz
D_2	1Hz

Note that in this problem each channel is represented by a subscripted letter. It will be the convention throughout this report to represent the channels in this manner. Different letters will be applied to different frequencies for a given problem. In practice, however, numbers are generally used for channel identification.

An alternate representation for a scan table problem is to group the channels according to frequency and simply indicate the number of channels at each frequency.

<u>Label</u>	<u>Number</u>	<u>Frequency</u>
A	1	6Hz
B	2	3Hz
C	2	2Hz
D	2	1Hz

This second representation will generally be used in this report. So n_i refers to the number of channels in group number i , and f_i refers to the frequency of channels in group number i . It will be clearly stated when the first representation is being used in which case a reference to f_i indicates the frequency of channel number i .

A scan table which satisfies the scan requirements of this problem is given in Figure 2.1. This scan table indicates that channel A_1 is sampled on the first cycle and every third cycle thereafter. Channel B_1 is sampled on the second cycle and every sixth cycle thereafter. Channel C_1 is sampled on the third cycle and every ninth cycle thereafter. Similarly channels B_2 , C_2 , D_1 , and D_2 are listed in the scan table on the cycles on which they are to be sampled. The scan table is a circular list, so it is cycled through repeatedly.

It is important to note that the scan sequence for a given problem is a periodic function continuing indefinitely. The scan table is a window of this function which is exactly one period long.

By defining the system clock frequency, f_s , as the frequency at which the scan table is to be accessed, one finds that here the system clock frequency is 18 Hz. Furthermore, the divide rate, r_i , for frequency i , may be defined as the repetition rate that a channel at that frequency appears in the scan table. In this example, channel A_1 has a divide rate of three

A_1
B_1
C_1
A_1
B_2
C_2
A_1
B_1
D_1
A_1
B_2
C_1
A_1
B_1
C_2
A_1
B_2
D_2

Figure 2.1. A simple scan table.

since it appears exactly once every three entries. Channels B_1 and B_2 have divide rates of six since they appear exactly once every six entries, and so forth. It is essential that this divide rate be constant for each channel when digital filtering is to be used.

The relationship between channel frequency, f_i , channel divide rate, r_i , and system clock frequency, f_s , is given by

$$f_i = f_s / r_i \quad (1)$$

The scan table solution given in Figure 2.1 is by no means unique. In general, a problem has a multitude of scan tables all which serve as solutions. For instance, the assignment for channels B_1 and B_2 could be interchanged, and likewise for channels C_1 and C_2 , and channels D_1 and D_2 . Also the last entry could be shifted to the first location and all other entries shifted down one space. In addition consider that by adding "dummy channels" an infinite number of scan tables are possible. Figure 2.2 depicts two such scan tables for this problem, each generated by adding six dummy channels and increasing the system clock frequency to 24 Hz. The spaces marked "dummy" represent dummy channels, or rather time slots when none of the original channels are sampled. The first table is identical to the original table of Figure 2.1 except that a dummy channel was inserted after every third entry. The second example shows that it is not necessary that the dummy channels appear at equally spaced intervals as might be assumed from the first example.

Due to the increase in clock frequency it is usually not desirable to add these dummy channels, increasing the length of the scan table, as this higher rate may be difficult to implement. A higher clock frequency means

(a)

A ₁
B ₁
C ₁
dummy
A ₁
B ₂
C ₂
dummy
A ₁
B ₁
D ₁
dummy
A ₁
B ₂
C ₁
dummy
A ₁
B ₁
C ₂
dummy
A ₁
B ₂
D ₂
dummy

(b)

A ₁
B ₁
C ₁
C ₂
A ₁
B ₂
dummy
D ₁
A ₁
B ₁
dummy
D ₂
A ₁
B ₂
C ₁
C ₂
A ₁
B ₁
dummy
dummy
A ₁
B ₂
dummy
dummy

Figure 2.2. Scan tables containing dummy channels.

the system must run at a faster speed. In some cases, however, the opposite is true. Adding dummy channels may bring the system clock frequency up to a frequency that can be implemented when the lower frequency cannot be implemented with the given hardware. For instance a system clock frequency of 7025 Hz may not be available while a frequency of 8000 Hz is. In general, however, it is the minimum size scan table and consequently the lowest clock frequency which is desired.

The original example has a scan table length of 18 entries and a system clock frequency of 18 Hz. As a result the scan table is cycled through exactly once each second. This is not necessarily the case in the general problem. For example consider a problem identical to the first except that each of the seven channels requires a sample rate of twice its rate in the original problem. The original scan table would still be a valid solution, but the system clock frequency would be doubled (36 Hz). For such a problem the scan table would be cycled through once each half second.

Note that a property of the scan table is that it is accessed at a constant frequency f_s . The time separation between successive samples is $1/f_s$. Each of the time slots separated by this constant time interval has a single channel associated with it which is to be sampled at that time. By placing sample/hold circuits on the input channels or by allowing an unconstant system sample rate, these constraints could be lifted, providing for additional flexibility in the solution. However, these things will not be considered in this report as they result in considerable complication of problem mathematics, scan table format, and hardware required for implementation. For instance, in the case of using sample/hold circuits, droop can be a real problem at some of the lower sample frequencies.

CHAPTER III

MATHEMATICAL FORMULATION OF THE PROBLEM

This chapter describes the mathematics involved in scan table generation. The chapter begins by outlining a few conventions and divides the rest of the mathematics into three sections. In the first section formulas for basic system parameters are derived such as system clock frequency, scan table length, and divide rate for each channel. The second section describes for the scan table a mathematical model and the resulting requirements for a solution. In the third section a simplification procedure which can be applied to most problems is discussed and the corresponding mathematics are presented.

The first step in developing the mathematics for this problem is to determine what restrictions if any are to be placed on the frequencies used in the problem. To examine this, first recall Equation 1 and note that r_i is necessarily a finite positive integer.

$$f_i = f_s / r_i \quad i = 1, 2, \dots, n \quad (1)$$

Taking the ratio between any two frequencies f_i and f_j yields

$$\frac{f_i}{f_j} = \frac{(f_s / r_i)}{(f_s / r_j)} = \frac{r_j}{r_i} \quad \begin{array}{l} i = 1, 2, \dots, n \\ j = 1, 2, \dots, n \end{array}$$

Since both r_i and r_j are integers, it immediately follows that the ratio between any two frequencies must be a rational number in order to obtain a solution using a conventional scan table.

Now by scaling the frequencies it is possible to represent the given set of frequencies by a set of integer frequencies. Scaling is accomplished

by multiplying each frequency by the same constant k . k is chosen to have a value such that this product results in an integer for each frequency. The solution obtained by working a problem which has been modified in such a manner is valid in every way except that the system clock frequency has also been scaled by a factor of this constant k . The desired system clock frequency is found by dividing by k . For this reason integer frequencies may be assumed in the following mathematics without loss of generality.

It is desired that the frequencies represented in a scan table problem be integers for several reasons. One reason is that the mathematics are somewhat simplified if the frequencies as well as the divide rates are integers. Another reason for using integer frequencies is that most computers can work with integers with absolute precision while decimal real numbers are subject to truncation error.

In order to clarify the mathematics in the rest of the chapter, the following definitions are given. All numbers are integers.

f_s = system clock frequency. Equals the number of samples, including dummy channels, to be taken every second.

L_s = length of the scan table.

For $i = 1, 2, \dots, n$

f_i = frequency at which the channels in group i are to be sampled.

n_i = number of channels in group i .

r_i = divide rate for channels in group i .

For $i = 1, 2, \dots, k$

f_i = sample frequency for channel i .

r_i = divide rate for channel i .

s_i = scan table location of the first entry of channel i .

$LCM_i(v_i)$ = Lowest common multiple of the set of integers $\{v_1, v_2, \dots, v_k\}$. i.e., the lowest number which is evenly divisible by all numbers in the set.

$GCF_i(v_i)$ = Greatest common factor of the set of integers $\{v_1, v_2, \dots, v_k\}$. i.e., the largest number which divides evenly into all integers in the set.

Initial Computations

This section describes the initial computation for a scan table problem which produces information such as scan table length, system clock frequency, and divide rate for each channel. The equations will be given first and an explanation and proof will follow.

$$\frac{f_s}{LCM_i(f_i)} = \text{integer} \quad \begin{matrix} i = 1, 2, \dots, n \\ n_i \neq 0 \end{matrix} \quad (2)$$

$$f_s \geq \sum_{i=1}^n n_i \times f_i \quad (3)$$

$$L_s = LCM_i(r_i) = \frac{f_s}{GCF_i(f_i)} \quad (4)$$

Equation 2 states that the system clock frequency is an even multiple of $LCM_i(f_i)$ where f_i is the frequency for any group that contains at least one channel. This equation can be proved by rewriting Equation 1 as

$$r_i = f_s / f_i$$

Since each divide rate is an integer then this says the system clock frequency must be evenly divisible by each frequency. By definition of the lowest common multiple, if a set of numbers divides evenly into a number

then the lowest common multiple of that set of numbers divides evenly into that number [1]. Equation 2 immediately follows.

Equation 3 is another expression which helps to determine the system clock frequency. For a group of n_i channels to each be sampled f_i times a second, it is required that the system devote $n_i \times f_i$ sampling intervals each second to that group. This is true of all frequency groups, therefore the total number of samples required each second is at least $\sum n_i \times f_i$ for all i . The value of f_s represents the number of samples taken by the system each second, so f_s must be at least this value. The inequality in this equation is a result of the first equation. Dummy channels may be added to increase f_s from this minimum number of samples per second to a value which is a multiple of $\text{LCM}_i(f_i)$ if necessary. As was stated in the last chapter, a solution with the smallest system clock is usually the most desired. From Equations 2 and 3, the smallest system clock frequency is the smallest integer greater than or equal to $\sum n_i \times f_i$ which is a multiple of $\text{LCM}_i(f_i)$. In addition if a scan table with a given system clock is not realizable for some other reason which will be dealt with later, then the system clock is increased in amounts of $\text{LCM}_i(f_i)$ by adding additional dummy channels. The subject of how to add these dummy channels will be addressed following a proof of Equation 4.

Equation 4 defines the length of the scan table as the lowest common multiple of the divide rates represented. This equation comes from the definition of lowest common multiple. The scan table length, L_s , is exactly equal to one period of the scan sequence. Each divide rate must divide evenly into this length. Therefore, the scan table length must be equal to the lowest common multiple of the divide rates present. The second equality in this equation is derived from a property of lowest

common multiples and greatest common factors. Since the two sets of integers $\{r_1, r_2 \dots r_n\}$ and $\{f_1, f_2 \dots f_n\}$ are related by the equation $r_i x f_i = f_s$ for all i , then the product of $\text{LCM}_i(r_i)$ times $\text{GCF}_i(f_i)$ is also equal to f_s . This can also be written as $\text{LCM}_i(r_i) = f_s / \text{GCF}_i(f_i)$. This produces the second form of Equation 4.

Some methods of solution for the scan table which will be described later require treating the dummy channels as if they were regular channels. The frequency assigned to each dummy channel may be determined by knowing its divide rate. Assuming each empty slot in the scan table is represented by a unique dummy channel, the divide rate for each dummy channel is then the scan table length, L_s , equal to $\text{LCM}_i(r_i)$. From the previous paragraph it is seen that the corresponding frequency for this divide rate is given by $\text{GCF}_i(f_i)$. Another possibility concerning these dummy channels arises when in the scan table solution, some or all of the dummy channels appear at regularly spaced intervals. If such is the case then the dummy slots in the scan table could be represented by a fewer number of dummy channels at higher frequencies than this $\text{GCF}_i(f_i)$. However, by assigning the dummy channels to these higher frequencies in the problem specification, the solution set is restricted to those solutions with the dummy slots appearing at the related equally spaced intervals. The general solutions are therefore given when there is no restriction on the spacing of the dummy channels and consequently when the dummy channels are assigned a frequency of $\text{GCF}_i(f_i)$.

In view of the discussion presented thus far in this chapter and Equations 1, 2, 3, and 4, the initial calculations for a scan table problem are as follows. Write the frequency set as a set of integer frequencies by scaling. Calculate the minimum possible system clock frequency using

Equations 2 and 3. Compute the divide rates for each channel or frequency group using Equation 1. Either form of Equation 4 can now be used to determine the scan table length. If the method of solution requires the dummy channels to be included in the calculation, add an additional group of channels at a frequency of $GCF_i(f_i)$. The number of dummy channels in this new group is such that when multiplied by its frequency $GCF_i(f_i)$, and added to $\sum n_i x f_i$, a multiple of $LCM_i(f_i)$ is obtained.

It should be noted at this point that a solution is not always guaranteed with this procedure. Sometimes an even larger number of dummy channels is required to increase the system clock frequency to a higher multiple of $LCM_i(f_i)$. A problem of this type will be shown in the third example.

The following examples are given to demonstrate the mathematical representations presented thus far. The examples will be continued in the remaining sections of this chapter. Note that these examples are relatively simple and are used to demonstrate the concepts. Their scan tables may be easy to generate manually. This is not necessarily true in a more complex problem.

Example 1

<u>Label</u>	<u>n_i</u>	<u>f_i</u>
A	1	30
B	3	15
C	4	10
D	4	5

Initial calculations proceed as follows.

$$\sum_{i=1}^n n_i \times f_i = n_1 f_1 + n_2 f_2 + n_3 f_3 + n_4 f_4$$

$$= 1 \times 30 + 3 \times 15 + 4 \times 10 + 4 \times 5 = 135$$

$$\text{LCM}_i(f_i) = \text{LCM}(30, 15, 10, 5) = 30$$

Now 150 is the smallest multiple of 30 which is greater than or equal to 135. Therefore, f_s equals 150 Hz. Dummy channels will fill in the gap between 135 and 150. This means an additional 15 samples a second will be devoted to the dummy channels. The frequency of the dummy channels is

$$\text{GCF}_i(f_i) = \text{GCF}(30, 15, 10, 5) = 5 \text{ Hz}$$

The number of dummy channels at this frequency would then be three, since three channels at 5 Hz yield the desired 15 samples a second. If the method of solution directs that these dummy channels be included, they may be placed in group E.

With f_s now known the r_i 's may be determined using Equation 1. The divide rates for each frequency group are given below:

<u>Label</u>	<u>n_i</u>	<u>f_i</u>	<u>r_i</u>
A	1	30	5
B	3	15	10
C	4	10	15
D	4	5	30
E	3	5	30

The scan table length is given by

$$L_s = \text{LCM}_i(r_i) = \text{LCM}(5, 10, 15, 30, 30) = 30$$

The second form of Equation 4 could have been used yielding the same result.

$$L_s = f_s / \text{GCF}_i(f_i) = 150 / 5 = 30$$

Note that the dummy channels are at the same frequency as the channels in group D. The three dummy channels could just as well have been added to this group. After a solution is found then three of the D channels could then be reassigned as dummy channels. A scan table solution for this problem will be shown in the next section.

Example 2

<u>Label</u>	<u>n_i</u>	<u>f_i</u>
A	2	50
B	1	30
C	4	20
D	3	10

$$\begin{aligned} \sum_{i=1}^n n_i \times f_i &= n_1 \times f_1 + n_2 \times f_2 + n_3 \times f_3 + n_4 \times f_4 \\ &= 2 \times 50 + 1 \times 30 + 4 \times 20 + 3 \times 10 = 240 \end{aligned}$$

$$\text{LCM}_i(f_i) = \text{LCM}(50, 30, 20, 10) = 300$$

In this case the smallest multiple of 300 which is greater than or equal to 240 is 300. Therefore f_s equals 300 Hz. The dummy channels must therefore consume 60 samples per second. The frequency of the dummy channels is

$$\text{GCF}_i(f_i) = \text{GCF}(50, 30, 20, 10) = 10 \text{ Hz}$$

The number of dummies is then $60 \div 10 = 6$. They may be assigned to group D if desired. The divide rate for each frequency group is given in the table below.

<u>Label</u>	<u>n_i</u>	<u>f_i</u>	<u>r_i</u>
A	2	50	6
B	1	30	10
C	4	20	15
D	9	10	30

The scan table length is

$$L_s = \text{LCM}_i(r_i) = \text{LCM}(6, 10, 15, 30) = 30$$

A scan table solution for this problem is given in the next section.

Example 3

<u>Label</u>	<u>n_i</u>	<u>f_i</u>
A	1	3
B	1	2

$$\sum_{i=1}^n n_i \times f_i = n_1 \times f_1 + n_2 \times f_2 = 1 \times 3 + 1 \times 2 = 5$$

$$\text{LCM}_i(f_i) = \text{LCM}(3, 2) = 6$$

$$f_s = 6 \text{ Hz}$$

$$\text{GCF}_i(f_i) = \text{GCF}(3, 2) = 1$$

From the above equations one finds one dummy channel is required at this frequency of 1 Hz. With this channel create a new frequency group, C.

The divide rates are

<u>Label</u>	<u>n_i</u>	<u>f_i</u>	<u>r_i</u>
A	1	3	2
B	1	2	3
C	1	1	6

The scan table length is then

$$L_s = \text{LCM}_i(r_i) = \text{LCM}(2, 3, 6) = 6$$

As will be demonstrated in the next section, no scan table exists which contains divide rates of both 2 and 3. To remedy the situation more dummy channels should be added, still at a frequency of 1 Hz. The next higher multiple of $\text{LCM}_i(f_i)$ is 12, therefore, a total of 7 dummy channels

at a frequency of 1 Hz are required. The new divide rates are

<u>Label</u>	$\underline{n}_i$	$\underline{f}_i$	$\underline{r}_i$
A	1	3	4
B	1	2	6
C	7	1	12

The new scan table length is

$$L_s = \text{LCM}_i(r_i) = \text{LCM}(4,6,12) = 12$$

The next section will demonstrate that these new divide rates are acceptable. A scan table solution is now possible.

A Mathematical Formulation

In this section a mathematical representation of the problem will be presented. The channels are not grouped according to frequency in this representation and consequently a separate f_i is assigned to each channel.

If the divide rate for a channel is known as well as one of its locations in the scan table, then the remaining locations are given by adding and subtracting multiples of the divide rate from the known location. For this reason a scan table may be completely described by its length, the divide rate for each channel, and the first occurrence of each channel in the scan table. The divide rate for each channel may be determined using methods of the previous section so all that remains in finding a scan table solution is to find the first occurrence in the scan table of each channel.

The first location for each channel is represented by the integer s_i . Using the convention that the first scan table location is numbered 1, then the range of allowable values for s_i are given by,

$$1 \leq s_i \leq r_i \quad (5)$$

The upper limit restriction on s_i is imposed by the fact that each channel must appear exactly once every r_i entries. In addition to this restriction, each s_i must be such that in the scan table represented no two channels are assigned to the same location. For example if two channels have divide rates of 4 and 6, then first locations of 3 and 1 respectively would not be possible because location 7 would then be assigned to both channels.

In view of the above constraint, the set of first locations $\{s_1, s_2, \dots, s_k\}$ is acceptable as a solution if and only if the following equation is satisfied for all non-equivalent values of i and j .

$$\frac{s_i - s_j}{\text{GCF}(r_i, r_j)} \neq \text{integer} \quad \begin{array}{l} i = 1, 2, \dots, k \\ j = 1, 2, \dots, k \\ i \neq j \end{array} \quad (6)$$

This equation is based on a theorem [1] for equations of the form $ax+by=n$ where x , y , and n are integer constants, x and y are not both zero, and a and b are integers to be determined. The theorem states that there is a solution to this equation if and only if n is evenly divisible by $\text{GCF}(x, y)$. Another way of stating this is that there are no solutions to the equation if and only if n is not evenly divisible by $\text{GCF}(x, y)$.

In the case of the scan table, the locations occupied by channel i and channel j are given by $s_i + (a \times r_i)$ and $s_j + (b \times r_j)$ respectively, where

a and b represent arbitrary integers. What is required is that no two channels occupy the same scan table location. This is the same as requiring for the given values of s_i , r_i , s_j , and r_j , that the equation

$$s_i + (a \times r_i) = s_j + (b \times r_j)$$

has no solution for any values of a and b. This equation can be rewritten in the form

$$(a \times r_i) - (b \times r_j) = s_j - s_i$$

Since a and b represent arbitrary integers the minus sign on the left may be changed to plus.

$$(a \times r_i) + (b \times r_j) = s_j - s_i$$

Again what is necessary is that there be no solution to this equation for any i and j. The theorem stated previously says there is no solution if and only if the number $s_i - s_j$ is not evenly divisible by $GCF(r_i, r_j)$. This completes the proof of Equation 6.

One important implication of Equation 6 is that each pair of divide rates in the scan table must have a greatest common factor other than 1. If the greatest common factor of a pair of divide rates is equal to 1, then the left hand side of Equation 6 must be an integer, for any values of s_i and s_j . Therefore no scan table solution is possible using such divide rates. To obtain a valid solution, it is then necessary to add dummy channels in the manner described in the previous section. This would increase the system clock frequency and consequently the divide rates would increase to an acceptable value. Such was the case of Example 3 in the last section.

To this point a condensed version for a scan table has been presented along with an equation to establish its validity. Chapter IV will present algorithms to create scan tables based on these mathematics.

Continuing Example 1 from the previous section, Table 3.1 presents the condensed version of a scan table by including the first occurrence in the scan table of each channel. Note that the dummy channels need not be specified in the condensed scan table.

Table 3.1
Condensed Scan Table for Example 1

Label	f_i	r_i	s_i
A_1	30	5	1
B_1	15	10	2
B_2	15	10	7
B_3	15	10	4
C_1	10	15	3
C_2	10	15	8
C_3	10	15	13
C_4	10	15	5
D_1	5	30	9
D_2	5	30	19
D_3	5	30	29
D_4	5	30	10

The corresponding scan table is given in Figure 3.1. The table is numbered to assist the reader in convincing himself that the two scan table forms are equivalent.

1	A ₁
2	B ₁
3	C ₁
4	B ₃
5	C ₄
6	A ₁
7	B ₂
8	C ₂
9	D ₁
10	D ₄
11	A ₁
12	B ₁
13	C ₃
14	B ₃
15	D ₆
16	A ₁
17	B ₂
18	C ₁
19	D ₂
20	C ₄
21	A ₁
22	B ₁
23	C ₂
24	B ₃
25	D ₅
26	A ₁
27	B ₂
28	C ₃
29	D ₃
30	D ₇

Figure 3.1. A scan table for Example 1.

Figure 3.2 illustrates how Equation 6 is satisfied for all channel pairs. Each entry in the table is a fraction. The numerator is the absolute value of the difference in first occurrences of the two channels. The denominator is the greatest common factor of the divide rates. Note that in no case does the fraction reduce to an integer.

B ₁	$\frac{1}{5}$											
B ₂	$\frac{6}{5}$	$\frac{5}{10}$										
B ₃	$\frac{3}{5}$	$\frac{2}{10}$	$\frac{3}{10}$									
C ₁	$\frac{2}{5}$	$\frac{1}{5}$	$\frac{4}{5}$	$\frac{1}{5}$								
C ₂	$\frac{7}{5}$	$\frac{6}{5}$	$\frac{1}{5}$	$\frac{4}{5}$	$\frac{5}{15}$							
C ₃	$\frac{12}{5}$	$\frac{11}{5}$	$\frac{6}{5}$	$\frac{9}{5}$	$\frac{10}{15}$	$\frac{5}{15}$						
C ₄	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{1}{5}$	$\frac{2}{15}$	$\frac{3}{15}$	$\frac{8}{15}$					
D ₁	$\frac{8}{5}$	$\frac{7}{10}$	$\frac{2}{10}$	$\frac{5}{10}$	$\frac{6}{15}$	$\frac{1}{15}$	$\frac{4}{15}$	$\frac{4}{15}$				
D ₂	$\frac{18}{5}$	$\frac{17}{10}$	$\frac{12}{10}$	$\frac{15}{10}$	$\frac{16}{15}$	$\frac{11}{15}$	$\frac{6}{15}$	$\frac{14}{15}$	$\frac{10}{30}$			
D ₃	$\frac{28}{5}$	$\frac{27}{10}$	$\frac{22}{10}$	$\frac{25}{10}$	$\frac{26}{15}$	$\frac{21}{15}$	$\frac{16}{15}$	$\frac{24}{15}$	$\frac{20}{30}$	$\frac{10}{30}$		
D ₄	$\frac{9}{5}$	$\frac{8}{10}$	$\frac{3}{10}$	$\frac{6}{10}$	$\frac{7}{15}$	$\frac{2}{15}$	$\frac{3}{15}$	$\frac{5}{15}$	$\frac{1}{30}$	$\frac{9}{30}$	$\frac{19}{30}$	
	A ₁	B ₁	B ₂	B ₃	C ₁	C ₂	C ₃	C ₄	D ₁	D ₂	D ₃	

Figure 3.2. Equation 6 applied to all channel pairs for Example 1.

A condensed scan table for the second example of the previous section is given in Table 3.2 and the corresponding scan table is in Figure 3.3. Figure 3.4 shows for this example how the absolute value of the difference in first occurrences is never divisible by $\text{GCF}(r_i, r_j)$.

The third example of the previous section demonstrates that the greatest common factor of two divide rates cannot be equal to one. The original clock rate computed resulted in divide rates of 3 and 2. A scan

Table 3.2
Condensed Scan Table for Example 2

Label	f_i	r_i	s_i
A_1	50	6	1
A_2	50	6	3
B_1	30	10	2
C_1	20	15	5
C_2	20	15	8
C_3	20	15	11
C_4	20	15	14
D_1	10	30	4
D_2	10	30	6
D_3	10	30	10

1	A_1
2	B_1
3	A_2
4	D_1
5	C_1
6	D_2
7	A_1
8	C_2
9	A_2
10	D_3
11	C_3
12	B_1
13	A_1
14	C_4
15	A_2
16	dummy
17	dummy
18	dummy
19	A_1
20	C_1
21	A_2
22	B_1
23	C_2
24	dummy
25	A_1
26	C_3
27	A_2
28	dummy
29	C_4
30	dummy

Figure 3.3. A scan table for Example 2.

A_2	$\frac{2}{6}$									
B_1	$\frac{1}{2}$	$\frac{1}{2}$								
C_1	$\frac{4}{3}$	$\frac{2}{3}$	$\frac{3}{5}$							
C_2	$\frac{7}{3}$	$\frac{5}{3}$	$\frac{6}{5}$	$\frac{3}{15}$						
C_3	$\frac{10}{3}$	$\frac{8}{3}$	$\frac{9}{5}$	$\frac{6}{15}$	$\frac{3}{15}$					
C_4	$\frac{13}{3}$	$\frac{11}{3}$	$\frac{12}{5}$	$\frac{9}{15}$	$\frac{6}{15}$	$\frac{3}{15}$				
D_1	$\frac{3}{6}$	$\frac{1}{6}$	$\frac{2}{10}$	$\frac{1}{15}$	$\frac{4}{15}$	$\frac{7}{15}$	$\frac{10}{15}$			
D_2	$\frac{5}{6}$	$\frac{3}{6}$	$\frac{4}{10}$	$\frac{1}{15}$	$\frac{2}{15}$	$\frac{5}{15}$	$\frac{8}{15}$	$\frac{2}{30}$		
D_3	$\frac{9}{6}$	$\frac{7}{6}$	$\frac{8}{10}$	$\frac{5}{15}$	$\frac{2}{15}$	$\frac{1}{15}$	$\frac{4}{15}$	$\frac{6}{30}$	$\frac{4}{30}$	
	A_1	A_2	B_1	C_1	C_2	C_3	C_4	D_1	D_2	

Figure 3.4. Application of Equation 6 to channel pairs of Example 2.

table containing such divide rates cannot be implemented, therefore dummy channels were added. This increased the system clock frequency to twice what it was before, causing the divide rates to also double. The new divide rates had a common factor of two making a solution possible. Because a solution to this problem is simple, one will not be given here. A solution to this problem may be found in Appendix II.

Partitioning the Problem - The Wheel Approach

In this section a simplification that can be applied to most problems will be discussed. The section will include the mathematics for such a procedure. Wheel diagrams will be introduced as a graphical method for representing scan tables solved in such a manner.

In the fields of science and mathematics many problems may be solved by breaking the problem up into smaller parts and then solving this set of "subproblems." If these subproblems cannot immediately be solved then they too may be broken into smaller parts. By continuing this process it is possible to solve large problems by using only a few rules for partitioning the problem and a few solutions to the simple problems eventually created. Fortunately most scan table problems may be treated in such a manner.

The procedure for dividing a scan table into smaller parts is a type of partitioning. It will first be demonstrated graphically using Example 1 and will be followed by a mathematical treatment of the process. The concept of a wheel diagram will evolve from this process.

For Example 1, methods shown at the beginning of this chapter allow the determination of the system clock frequency, scan table length and divide rate for each channel. The number, frequency, and divide rate for each group of frequencies are shown again in the following table. The dummy channels are included in group D since the D channels have the same divide rate as the dummy channels. This is not necessary but it simplifies the problem. The channels may be separated later by relabeling three of the D channels as dummy channels.

<u>Label</u>	<u>n_i</u>	<u>f_i</u>	<u>r_i</u>
A	1	30	5 = 5 X 1
B	3	15	10 = 5 X 2
C	4	10	15 = 5 X 3
D	7	5	30 = 5 X 6

Figure 3.5 shows how this scan table may be divided into a set of five smaller scan tables each of one-fifth the original size. The original scan table is shown in Figure 3.5a with the entries numbered periodically from 1 through 5. Figure 3.5b shows five smaller "mini scan tables" each composed of entries corresponding to a given number in the original scan table. For instance all the entries in the original scan table which are numbered 2 are placed in the mini scan table numbered 2. The entries within each mini scan table are in the same order that they appear in the original table. These five tables are linked together by a common "wheel" with five "spokes," one spoke leading to each mini scan table.

Associated with the original scan table of Figure 3.5a is a pointer indicating which entry is to be sampled on a particular clock cycle. The pointer is incremented (i.e., moved down in the table one entry) for each system clock cycle. A pointer is also associated with each mini scan table of Figure 3.5b. In addition the wheel at the center has a pointer associated with it. The pointer on this wheel indicates which mini scan table is to be accessed on a particular clock cycle. This pointer is rotated one "spoke" clockwise for each system clock cycle. The pointers on the mini scan tables are incremented only on cycles when that particular mini scan table is accessed. In this case the mini scan tables are cycled

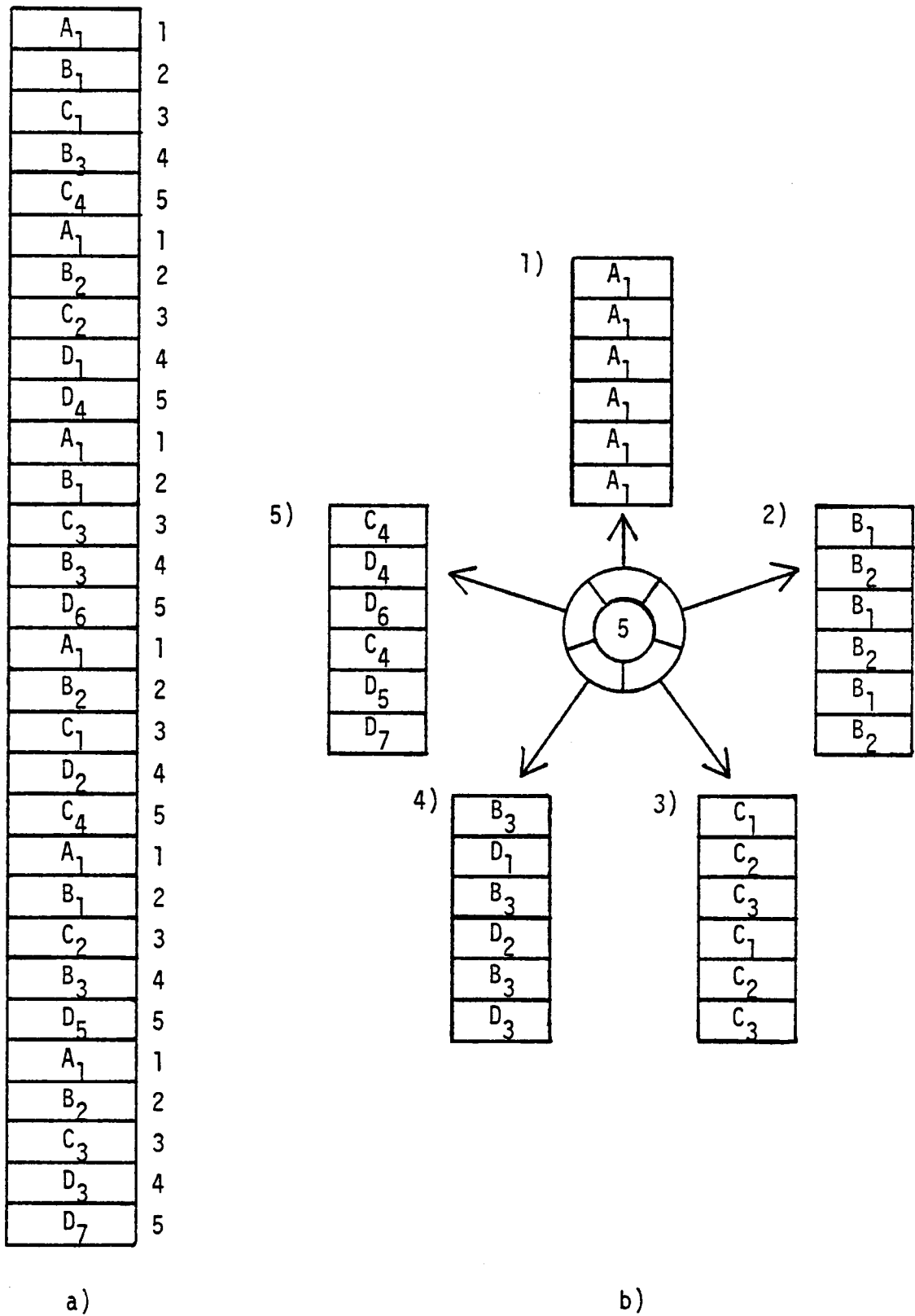


Figure 3.5. Dividing a scan table for Example 1 into smaller tables. a) The original table. b) The five resulting mini scan tables.

through at one-fifth the system clock frequency. By initializing the pointer on the wheel to the first mini scan table, a scan sequence identical to the one of the original table will be generated.

A significant property of the mini scan tables shown in Figure 3.5b is that each channel appears in one and only one of these mini scan tables. In this figure then there are essentially five separate problems each containing a unique set of channels. Each of these subproblems has its own scan table. The subproblems are alternately selected at the system clock frequency. Therefore each mini scan table is cycled through at one-fifth the frequency that the original scan table was cycled through. To compensate for this, each channel's divide rate within its mini scan table is one-fifth its divide rate in the original scan table. Therefore the equation

$$f_i = f_s / r_i$$

still holds if f_s is now defined as the frequency that the mini scan table is cycled through and r_i is the channel's divide rate in the mini scan table.

The reason that this problem may be divided into five subproblems, each containing a unique set of frequencies, is because of the common factor in the original divide rates. The greatest common factor of the original divide rates is five. Since each channel's divide rate is a multiple of the number five, then numbering the scan table periodically from one to five as in Figure 3.5a results in a channel always occurring with the same number. Channels associated with each of the five numbers may now be separated into five separate scan table problems.

There was nothing special about the number five in this example. All that was important is that there was a common factor greater than one. If

the greatest common factor of the divide rates was equal to one then nothing could be achieved by partitioning. In this case the original problem would result in a single subproblem, and this subproblem would be identical to the original problem.

Another important point is that if the greatest common factor of the divide rates is not prime then one of its factors could be used as the number of pieces in which to partition a problem. The end result, however, would be the same. For example, if $GCF_i(r_i)$ is equal to six, then the problem could be partitioned into three subproblems instead of six. Each channel's new divide rate in the mini scan tables would be divided by three. The divide rates within each subproblem must still have a common factor of two. Therefore each of the three subproblems could further be divided into two subproblems. At this point there would be six subproblems, each containing one-sixth of the original problem. It would have been just as well to have divided the original problem into its six pieces in the first place.

As mentioned at the beginning of this section, each subproblem may be broken down just as if it were an original problem. By recursively subdividing each problem in this way, eventually the subproblems created must consist of a set of channels at a single frequency and divide rate. The solution to a problem containing a single frequency group is trivial. Its scan table consists of a list of the channels in any order, where each channel appears only once. The mathematics depicted in Figure 3.6 summarizes the procedure for breaking down a problem into its subproblems.

The original problem is described by:

$\bar{N}_0 = (n_1, n_2, \dots, n_n)$ the number in each frequency group

$\bar{F}_0 = (f_1, f_2, \dots, f_n)$ the frequency of each group

$fs_0 = \bar{N}_0 \cdot \bar{F}_0$ the frequency the original table is cycled through at

$\bar{R}_0 = (r_1, r_2, \dots, r_n)$ the divide rate for channels in a frequency group. Note $r_i = fs_0/f_i$

$g_0 = \text{GCF}_i(r_i) \quad i=1,2,\dots,n$
and $n_i \neq 0$ the greatest common factor of the divide rates used. g_0 must be greater than one.

The problem may be partitioned into g_0 subproblems.

For subproblem j ($j=1,2,\dots,g_0$)

$\bar{N}_j = (n_{j1}, n_{j2}, \dots, n_{jn})$ the number in each frequency group

$\bar{F}_j = \bar{F}_0$ the frequency of each group is unchanged

$fs_j = fs_0/g_0$ the frequency each subproblem's scan table is cycled through is independent of subproblem

$\bar{R}_j = (r_{j1}, r_{j2}, \dots, r_{jn})$ the divide rate for each frequency group

$r_{ji} = r_i/g_0$ the divide rate is independent of the subproblem

Note that in addition the following must be true of the N_j 's.

Criteria 1) $\sum_{j=1}^{g_0} \bar{N}_j = \bar{N}_0$ this is so all the numbers add up to the original problem

Criteria 2) $\bar{N}_j \cdot \bar{F}_0 = fs_j = fs_0/g_0 \quad j = 1,2,\dots,n$ Similar to Equation 3

Figure 3.6. Mathematics for partitioning a problem. Each number is a non-negative integer.

These mathematics provide much information about the subproblems. What they do not indicate, however, is just how the channels are distributed among the subproblems. Criteria 1 and 2 narrow down the possibilities for this partitioning somewhat. Also note that many subproblems arrived at, due to their nature, may have no solutions at all. Therefore an algorithm for partitioning problems must be able to backtrack. Backtracking involves removing an unsolvable subproblem from the list and continuing the search for new subproblems. The next chapter will discuss algorithms for recursively partitioning problems to achieve scan table solutions.

The partitioning described for Example 1 in Figure 3.5b was arrived at through the use of scan table. In the general case the scan table is not known and the partitioning must develop through the mathematics. In Example 1 to achieve the partition of Figure 3.5b, the mathematics will proceed as follows.

$$\overline{N}_0 = (1, 3, 4, 7)$$

$$\overline{F}_0 = (30, 15, 10, 5)$$

$$fs_0 = \overline{N}_0 \cdot \overline{F}_0 = 150$$

$$\overline{R}_0 = (5, 10, 15, 30)$$

$$g_0 = \text{GCF}(5, 10, 15, 30) = 5$$

The problem is partitioned into five parts. The following partition is valid.

$$\bar{N}_1 = (1,0,0,0)$$

$$\bar{N}_2 = (0,2,0,0)$$

$$\bar{N}_3 = (0,0,3,0)$$

$$\bar{N}_4 = (0,1,0,3)$$

$$\bar{N}_5 = (0,0,1,4)$$

$$\bar{F}_j = \bar{F}_0 = (30,15,10,5)$$

$$fs_j = fs_0/g_0 = 30$$

$$\bar{R}_j = (1,2,3,6)$$

In addition note that this set of N_j 's satisfies the following.

$$\sum_{j=1}^5 \bar{N}_j = \bar{N}_0 = (1,3,4,7)$$

and

$$\bar{N}_j \cdot \bar{F}_0 = fs_j = 30 \quad \text{for all } j = 1,2,3,4,5$$

An easy way to keep track of a partitioning is to combine the $\bar{N}_j$ vectors to form a matrix. Writing each vector as a row of the matrix produces a matrix with a width equal to the number of frequency groups and length equal to g_0 . Now the requirements on the matrix are as follows. Each element must be a non-negative integer. The sum of the numbers in each column must be equal to the corresponding element of $\bar{N}_0$. The dot product of each row with $\bar{F}_0$ should equal fs_0/g_0 .

The $\bar{N}_j$ vectors which are determined at this point contain no information about how the channels are distributed among the subproblems. They tell only how many channels of each frequency group are in each subproblem. In solving scan table problems what is generally done is to work only with the number of channels at each frequency in each subproblem.

After a solution is found, then each channel is assigned to a particular spot in the solution.

In the representation for Example 1 in Figure 3.5b, page 30, it is seen that the mini scan tables shown are not all in their optimum form. In each case the scan table length should be equal to the lowest common multiple of all the divide rates represented. Since it is not for mini scan tables 1,2, and 3 there is a repetition in each of these tables. A simpler version of Figure 3.5b is given in Figure 3.7.

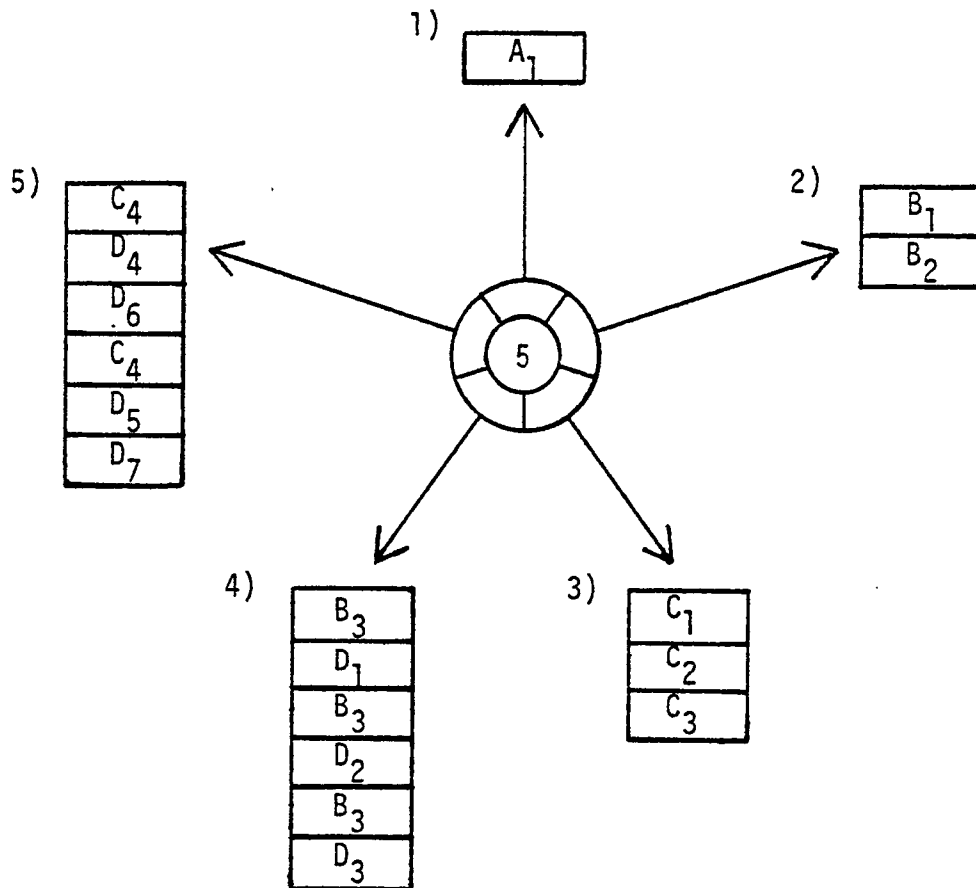


Figure 3.7. A simpler representation of Figure 3.5b.

To continue with this example and develop further the concept of the wheel diagram, it is noted that the first mini scan table has only a single element A_1 . In an actual wheel diagram, this scan table of length one would be removed and the channel A_1 would be placed directly on the center wheel. In addition the mini scan tables numbered 2 and 3 may both be immediately solved. Each table contains channels of a single frequency. These two mini scan tables can then each be drawn as wheels with entries directly on the wheel. These wheels are still like the mini scan tables. They each have their own pointers which are incremented (rotated one slot clockwise) each time that wheel is addressed by the pointer on the "main wheel" in the center. The pointer for the main wheel still rotates one slot for each period of the system clock cycle. Figure 3.8 shows the wheel diagram as it is developed to this point.

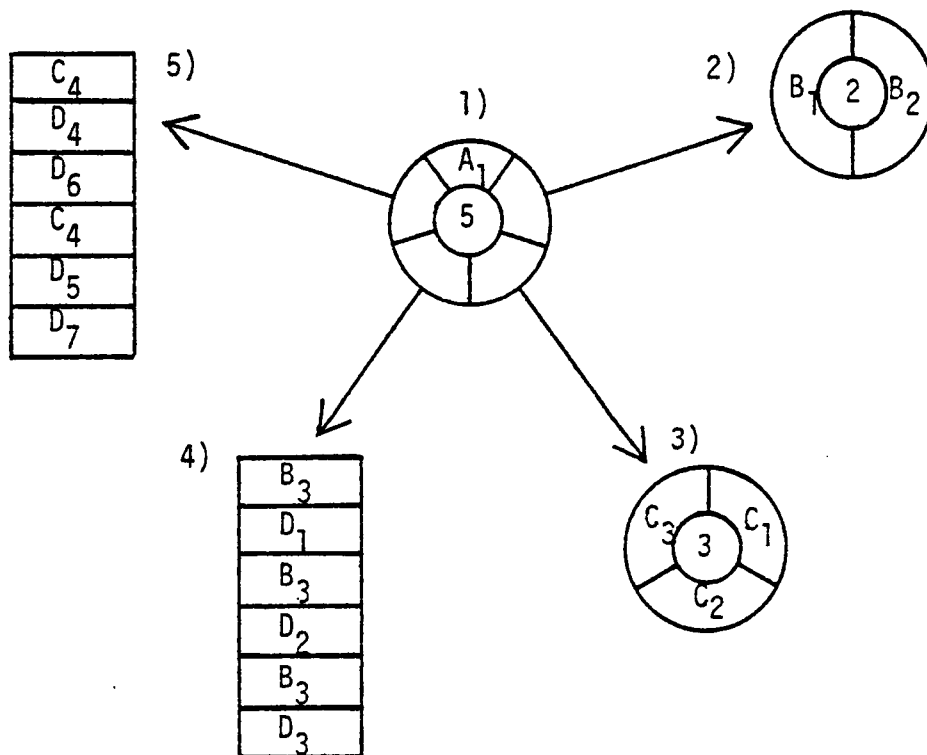


Figure 3.8. The first stage in forming a wheel diagram for Example 1.

To continue this procedure to obtain a complete wheel diagram, each of the mini scan tables must be partitioned in the same manner as the original problem. Each of the divide rates in mini scan table number 4 has a common factor of two. Therefore the entries in this table are numbered periodically from one to two. The entries numbered 1 are placed in a scan table and the entries numbered 2 are placed in a separate scan table. The two tables are addressed by a "2-wheel" which is in the location of the original mini scan table. Similarly the divide rates of the entries in the mini scan table number 5 have a common factor of 3. They are therefore placed in three mini scan tables with a common "3-wheel." Figure 3.9 shows the almost-complete wheel diagram.

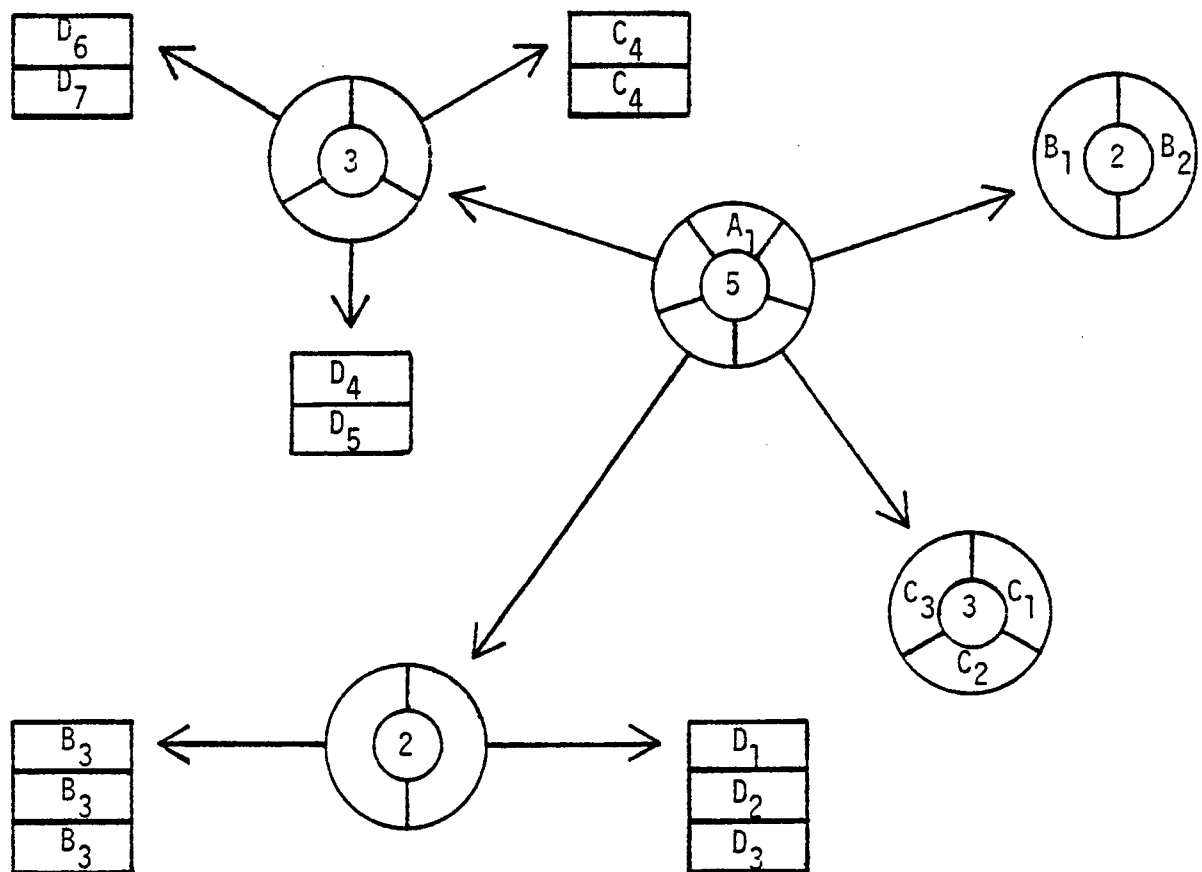


Figure 3.9. An almost-complete wheel diagram for Example 1.

The mathematics corresponding to the partitioning of subproblems number 4 and 5 are as follows.

For subproblem 4

$$\overline{N}_0 = (0,1,0,3)$$

$$\overline{F}_0 = (30,15,10,5)$$

$$fs_0 = \overline{N}_0 \cdot \overline{F}_0 = 30$$

$$\overline{R}_0 = (1,2,3,6)$$

$$g_0 = \text{GCF}_i(r_i) \quad \begin{matrix} i = 1,2,3,4 \\ n_i \neq 0 \end{matrix}$$

$$= \text{GCF}(2,6) = 2$$

The subproblems are

$$\overline{N}_1 = (0,1,0,0)$$

$$\overline{N}_2 = (0,0,0,3)$$

$$\overline{F}_j = \overline{F}_0 = (30,15,10,5)$$

$$fs_j = fs_0/g_0 = 15$$

$$\overline{R}_j = (,1, ,3)$$

For subproblem 5

$$\overline{N}_0 = (0,0,1,4)$$

$$\overline{F}_0 = (30,15,10,5)$$

$$fs_0 = \overline{N}_0 \cdot \overline{F}_0 = 30$$

$$\overline{R}_0 = (1,2,3,6)$$

$$g_0 = \text{GCF}_i(r_i) \quad \begin{matrix} i = 1,2,3,4 \\ n_i \neq 0 \end{matrix}$$

$$= \text{GCF}(3,6) = 3$$

The subproblems are

$$\overline{N}_1 = (0,0,1,0)$$

$$\overline{N}_2 = (0,0,1,2)$$

$$\overline{N}_3 = (0,0,1,2)$$

$$\overline{F}_j = \overline{F}_0 = (30,15,10,5)$$

$$fs_j = fs_0/g_0 = 10$$

$$\overline{R}_j = (, ,1,2)$$

Note that each problem included all frequency groups in its specification; however, not all frequency groups had channels in it.

To complete the wheel diagram for Example 1, all that remains is to further reduce the mini scan tables to wheels as before, since each contains channels of a single frequency. The complete wheel diagram is now shown in Figure 3.10.

The dots on this figure indicate the initial pointer locations for each wheel. Now, the operation of the complete wheel diagram is as follows. The main wheel is the original 5-wheel at the center. The pointer on this wheel is incremented (rotated one slot clockwise) after each cycle of the system clock. On each clock period the main wheel pointer either indicates a channel to be sampled or another wheel to be addressed. Likewise, each slot on each subwheel also indicates either a channel to be sampled or another wheel to be addressed. Following each cycle when any subwheel is addressed, the pointer on that subwheel is incremented. As a result, on each system clock cycle exactly one channel is addressed, with pointers on all intermediate wheels being updated.

In this example A_1 is sampled initially and the main wheel pointer is updated. On the next cycle the 2-wheel is addressed. This pointer points

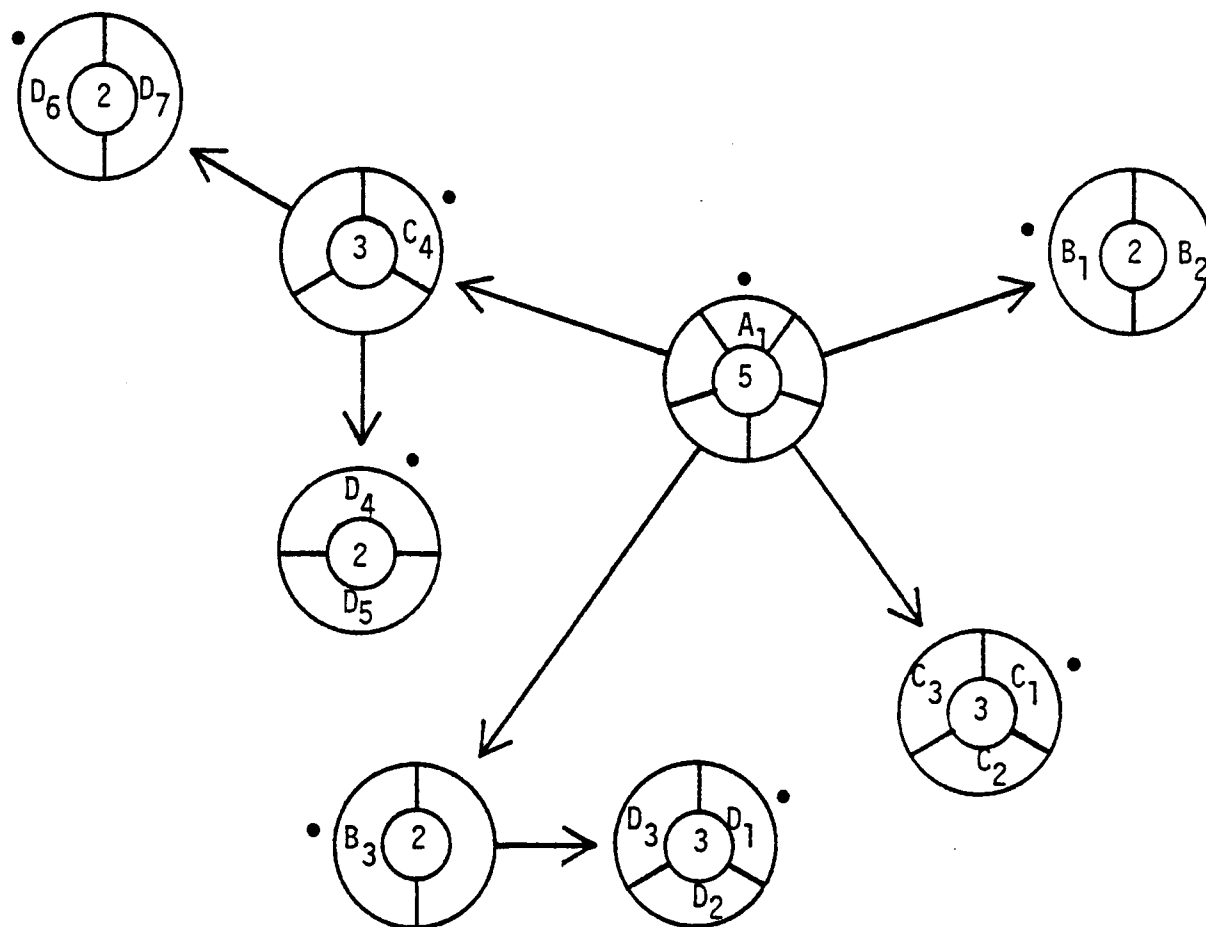


Figure 3.10. The complete wheel diagram for Example 1.

to channel B_1 so that channel is sampled and the pointer on that wheel is incremented along with the main wheel pointer at the end of the cycle. Similarly on the third cycle, C_1 is sampled and that pointer is incremented. On the fourth cycle B_3 is sampled. The pointer on that wheel is incremented and now points to the 3-wheel. Continuing in this fashion the reader should convince himself that the original scan sequence shown in Figure 3.5a, page 30, will be generated.

At this point note that a channel's divide rate in the original problem is reflected in the wheel diagram by multiplying together all wheel sizes between the channel that wheel is on and the main wheel, inclusive. For example channel D_6 is on a 2-wheel (wheel of size two) coming from a 3-wheel which comes from the main wheel (size 5). The divide rate for this channel is $2 \times 3 \times 5 = 30$.

In trying to partition Example 2 the following is true.

$$\overline{N}_0 = (2, 1, 4, 9)$$

$$\overline{F}_0 = (50, 30, 20, 10)$$

$$fs_0 = \overline{N}_0 \cdot \overline{F}_0 = 300$$

$$\overline{R}_0 = (6, 10, 15, 30)$$

$$g_0 = GCF_i(r_i) = GCF(6, 10, 15, 30) = 1$$

The greatest common factor of the divide rates is 1 in this case. Therefore partitioning is not possible with this problem.

One of the reasons for using a wheel based approach in solving scan tables is that for most problems the scan sequence may be stored in less memory if it is stored in wheel form rather than in scan table form. The scan table form may require a lot of memory because each channel may appear

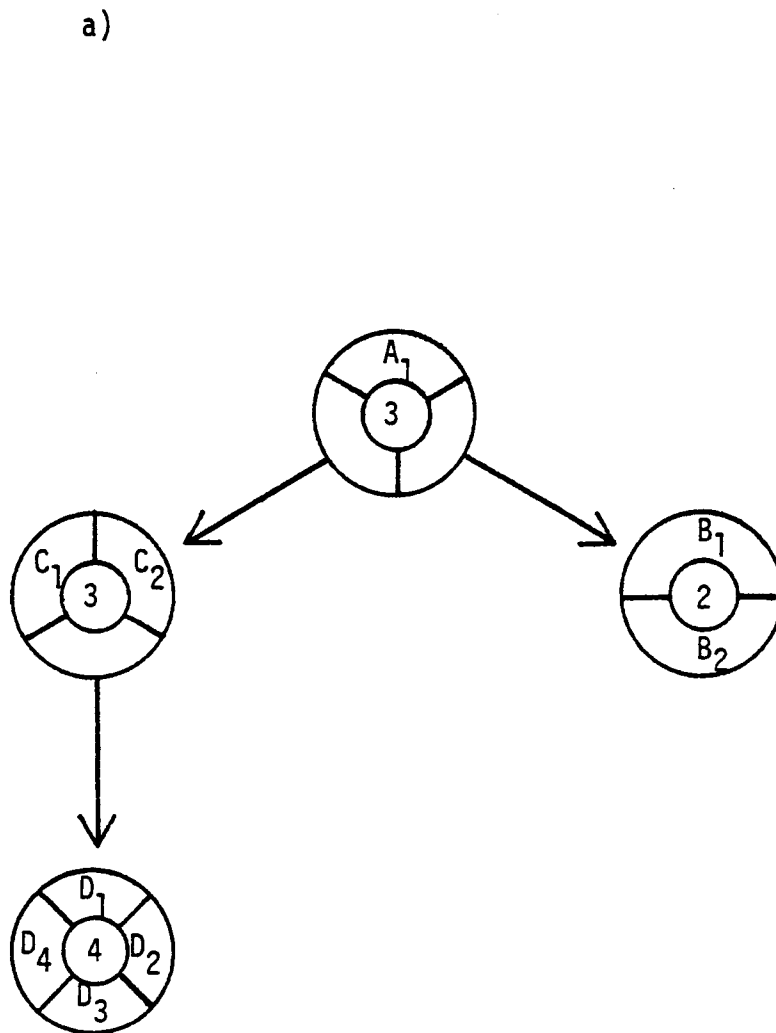
many times in the table. Storing a wheel diagram, however, involves only one location for each channel (each channel appears only once on the wheel diagram) and a few additional locations for parameters of each wheel. The scan sequence represented by a wheel diagram in memory can be produced by simulating the operation of the wheel diagram. Figure 3.11 shows an example of a wheel representation in memory.

In this representation each wheel has two memory locations associated with it. The first is an entry for the current pointer position. The next is the wheel size. In Figure 3.11b the main wheel pointer is in location 1 and the main wheel size is in location 2. The 2-wheel has its pointer and wheel size in locations 6 and 7 respectively, and so forth. Immediately following the wheel size for each wheel is a list of the entries on that wheel. The wheel pointer indicates one of these entries. Each entry is either a channel to be sampled or an indirect address to another wheel's pointer. An indirect address could be flagged by a single bit in each byte.

The scan sequence can be obtained from the wheel representation in memory just as it was from the actual wheel diagram. On each cycle the main wheel is addressed. Each time a wheel is addressed, its pointer is incremented. If the pointer value is greater than the wheel size, the pointer is reset to a value of one. The pointer then indicates either the channel to be sampled or another wheel to be addressed.

There are two major considerations in storing the scan sequence in wheel form. The first is how much memory is saved. The second is how much time is involved in generating the scan sequence from the wheel form.

The amount of memory required using the scan table form is equal to the scan table length plus one memory location for a pointer and one for



b)

1	pointer
2	3
3	A ₁
4	I 6
5	I 10
6	pointer
7	2
8	B ₁
9	B ₂
10	pointer
11	3
12	C ₁
13	C ₂
14	I 15
15	pointer
16	4
17	D ₁
18	D ₂
19	D ₃
20	D ₄

Figure 3.11. Storing a scan sequence in wheel form. a) A wheel diagram. b) The wheel diagram in memory.

the scan table length. The scan table length is $LCM_i(r_i)$, so the total number of locations required to store the scan table, pointer, and scan table length is $LCM_i(r_i) + 2$. A scan table form of Figure 3.11 would therefore require 38 locations. The wheel representation however, requires only 20 memory locations (just over half as many).

In general the number of memory locations required for a wheel representation of a scan table of C channels and W wheels is

$$\text{memory size} = C + 3 \times W - 1$$

Each channel is listed only once so one location is required for each channel. In addition each wheel involves three locations. The first two are the pointer value and the wheel size. The third location is accounted for by the indirect address which selects that wheel. This third location is involved with every wheel except the main wheel. Therefore the above equation is valid in determining the amount of memory required to store a scan sequence in wheel form.

A major factor in the time required to generate a scan sequence from the wheel representation is the number of memory accesses required on each cycle. Each time a wheel is addressed three or four memory accesses are required. The pointer value must be read from memory, requiring one memory access. If the pointer is decremented instead of incremented then the wheel size is needed only as a reset value when the pointer decrements to zero. This involves a memory access only every once in a while. The next memory access required every time a wheel is addressed involves storing the wheel's new pointer value. The final required memory access is reading the entry on the wheel indicated by the pointer. Therefore, there are three and sometimes four memory accesses required to retrieve a channel for each level down it is from the main wheel. For example if a wheel

diagram has a wheel five levels down, then 15 to 20 memory accesses are required each time a channel on that wheel is addressed.

The gain in representing the scan sequence in wheel form is generally a reduction in memory size. The drawback however, is that a fast memory and a fast processing unit are required to generate the sequence when a high system clock frequency is used.

If the scan sequence is to be stored in wheel form then the amount of memory can be reduced even further and the number of levels also reduced if the "dummy wheels" are reduced. If a wheel has all dummy channels on it, then the indirect address to that wheel can be replaced with a dummy channel. Figure 3.12 shows an example of this process. Figure 3.12a depicts a segment of a wheel diagram which contains four dummy wheels or wheels containing all dummy channels. In Figure 3.12b each of these dummy wheels has been removed and the indirect address pointing to that wheel is replaced with a dummy channel. Note that there are now four dummy channels, but they are each at a lower divide rate and consequently a higher frequency. As a result the wheel diagram is stored in less memory and fewer indirects are involved in its operation.

If there are no other channels at the same frequency as the dummy channels then this reduction is simple. Just reduce all wheels where the channels are at the divide rate of the dummy channels. However, if other channels exist at a frequency of the dummy channels then it is a non-trivial matter to optimally decide which wheels to assign as dummy wheels and reduce. A technique for accomplishing this will be described in the next chapter.

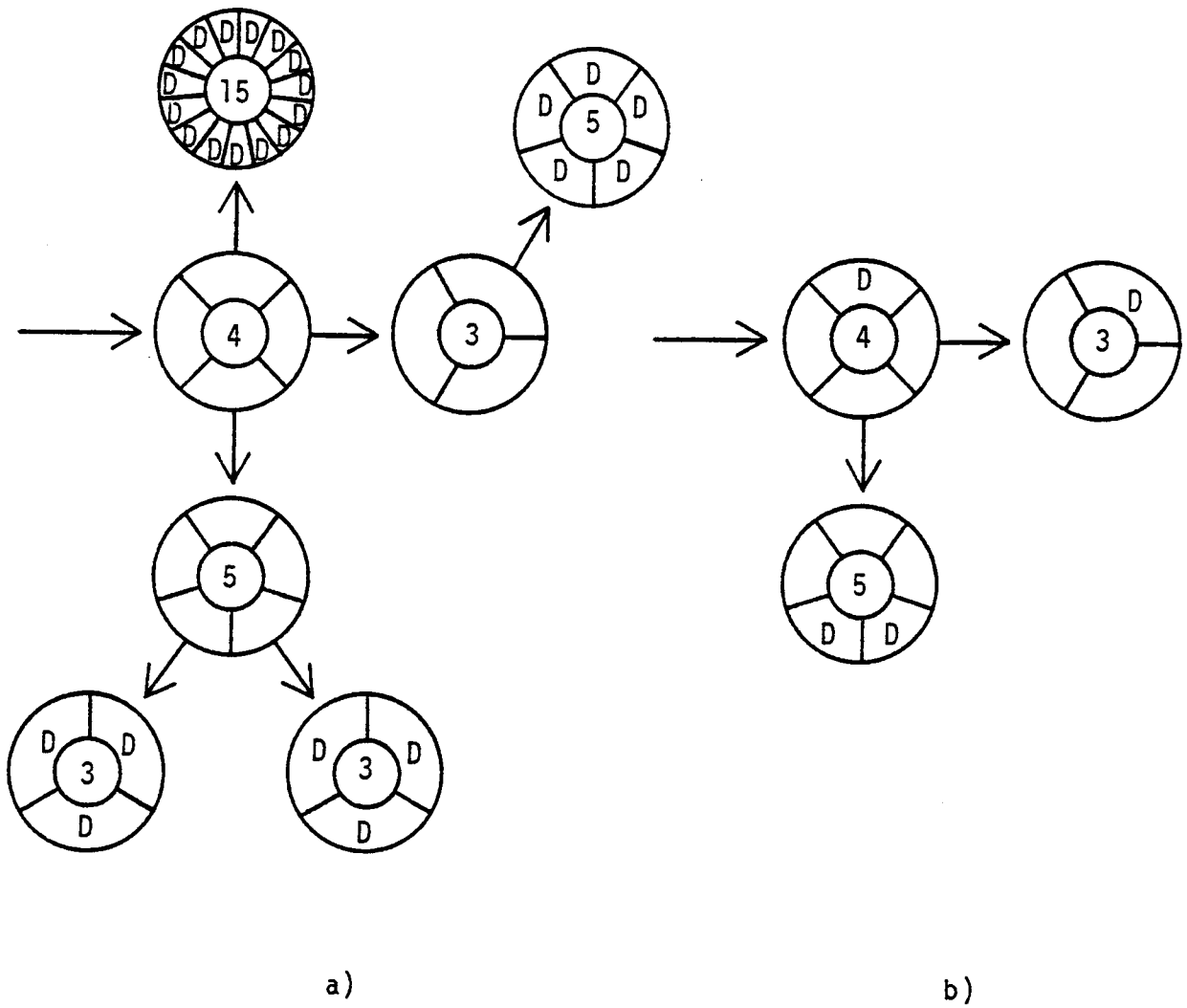


Figure 3.12. Reducing dummy wheels. a) A segment of the original wheel diagram. b) The segment with reduced dummy wheels.

CHAPTER IV

ALGORITHM DEVELOPMENT

The last chapter developed the mathematics which apply to the scan table. In this chapter a few algorithms will be discussed which generate scan tables based on these mathematical formulas. Each different approach is discussed in a separate section. The next chapter will focus on the test results.

Algorithm Based on First Occurrence of Each Channel

In view of the developments presented in the previous chapter, the most straightforward approach to generating a scan table appears to be one which determines the first occurrence of each channel in the scan table using Equations 5 and 6. The simplification procedure using wheel diagrams requires a recursive process and is therefore more difficult to implement. It will be the subject of the remaining sections in this chapter.

The flowchart of Figure 4.1 indicates a procedure for determining the initial location of each channel. With this algorithm, the dummy channels do not need to be included in the representation of the problem. The first part of the algorithm computes the divide rate for each channel, along with scan table length and system clock frequency. It then produces a condensed scan table which contains the initial location and divide rate for each channel.

The algorithm operates just as if it were filling in the scan table one channel at a time. The first channel is always assigned to location one. The remaining channels are taken one at a time and assigned the first starting location which is not used by any of the previous channels. After

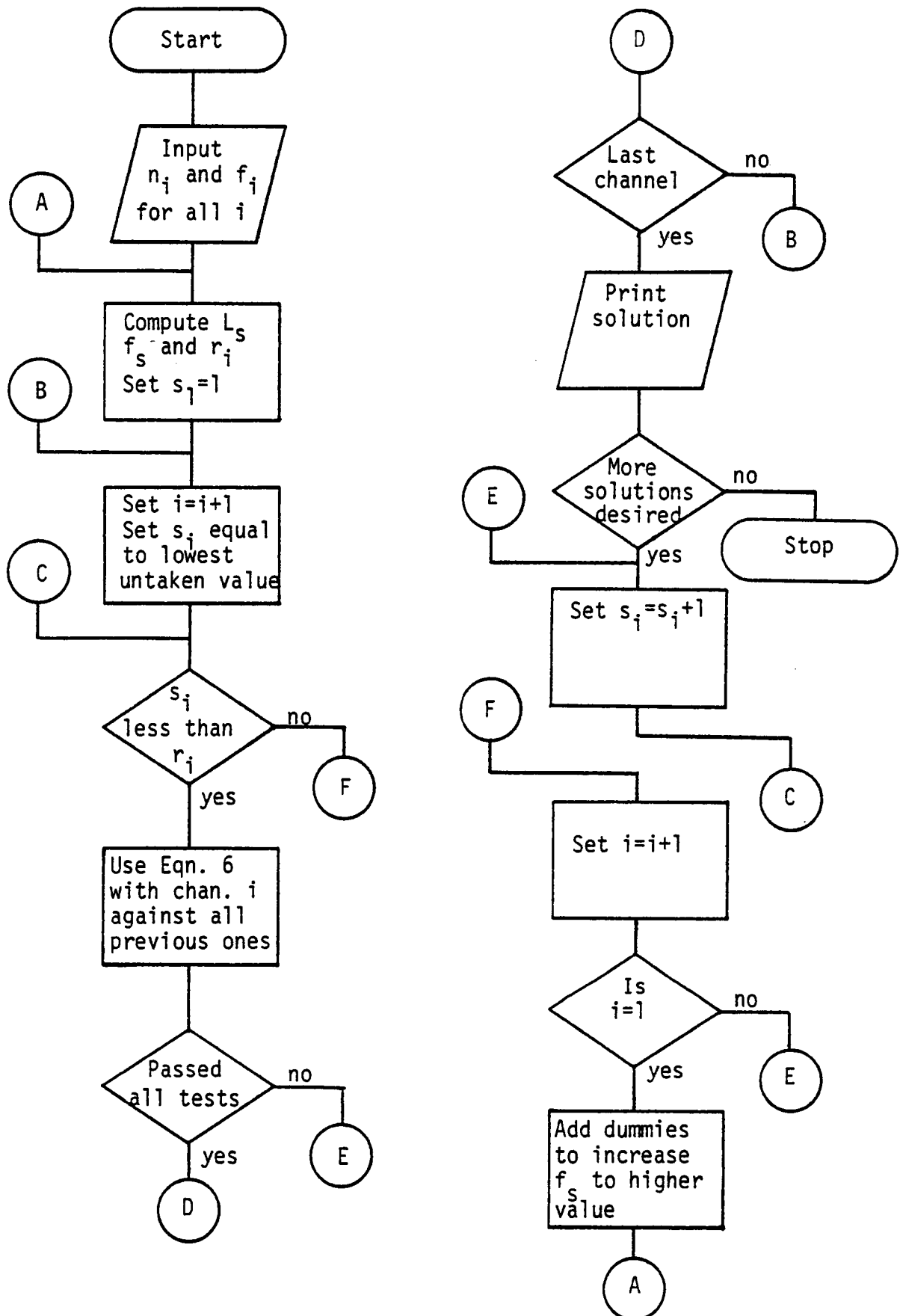


Figure 4.1. Flowchart for computing starting locations.

the starting location is assigned, the algorithm makes a check to see if this channel produces conflicting assignments with the previous channels using Equation 6. If this test fails then the next higher starting location is chosen and the test is repeated. If a starting location less than or equal to the divide rate is not available, then the algorithm backs up a step. It assumes the starting location for the previous channel was invalid and sets it to the next higher value and then continues as before. If all channels are assigned then the solution has been found. If the algorithm backs up far enough to try changing the starting location for the first channel, then the system clock frequency is increased by an amount $LCM_i(f_i)$ since all possibilities of the original problem have been tried. This has the effect of increasing each channel's divide rate, permitting more possibilities for assignment of starting locations. The algorithm then repeats.

The execution of this algorithm on Example 2 of the last chapter is depicted in Figure 4.2. In this figure each column indicates the first location assigned to each channel on a particular step. The circled entry in each column is the channel which has just been given a new starting location. At the bottom of each column is a space which indicates if the new channel assignment passed the test using Equation 6 on all previous channels. If the space is blank it indicates the channel passed all tests. In this case the next channel is added on the next step. If the space is not blank it indicates the first channel with which it would produce conflicting assignments as proved by Equation 6. For instance on step 3 channel B_1 is assigned a starting location of 3. However, using Equation 6 on channels A_1 and B_1 it is indicated channel B_1 must be moved. Note that by step 10 channel B_1 has been given all possible starting locations less

		STEP NUMBER																												
SCAN TABLE LOCATION		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
	1	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	A ₁	
	2		A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂		B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	B ₁	
	3			B ₁								A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	A ₂	
	4				B ₁									C ₁											D ₁	D ₁	D ₁	D ₁	D ₁	D ₁
	5					B ₁									C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	C ₁	
	6						B ₁									C ₂										D ₂	D ₂	D ₂	D ₂	D ₂
	7							B ₁									C ₂										D ₃			
	8								B ₁									C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂	C ₂
	9									B ₁									C ₃									D ₃		
	10										B ₁									C ₃									D ₃	
	11																				C ₃	C ₃	C ₃	C ₃	C ₃	C ₃	C ₃	C ₃	C ₃	C ₃
	12																					C ₄								
	13																						C ₄							
	14																							C ₄	C ₄	C ₄	C ₄	C ₄	C ₄	C ₄
	15																													
		FIRST CONFLICTING CHANNEL																												
			A ₁	A ₂	A ₁	A ₂	A ₁	A ₂	A ₁	A ₂			A ₁		A ₂	A ₁		A ₂	A ₁		A ₂	A ₁					A ₁	A ₂		

Figure 4.2. Sample run on Example 2 using the first location algorithm.

than its divide rate where channels A_1 and A_2 are as shown. Therefore, on the next step the algorithm backs up and searches for a new starting location for channel A_2 . After assigning channel A_2 to starting location 3, channel B_1 can be placed in location 2 and the algorithm continues. On the last step the last channel has been placed and its test of Equation 6 on all previous channels passed. The starting locations assigned at this point are therefore all valid. This solution found here is identical to the sample solution given for Example 2 in the last chapter.

A FORTRAN coding of this algorithm is given in Appendix I. The program has three distinct sections. The main program is used to input the data and control the calling of the other subroutines. A subroutine called SETUP is used to compute the system clock frequency, scan table length, divide rate for each channel and to initialize some other variables. A final subroutine, SOLVE, is the workhorse of the program. It generates the starting locations for each channel and prints the solution. An additional function routine, KGCF, is used to compute greatest common factors.

The main program in this approach is very similar to the main program used for the wheel approaches to be described subsequently. As the program begins execution it asks three questions: (1) Input file name, (2) Output file name, and (3) Number of solutions desired. The input file is the file from which the data is to be read. This file may be composed of any number of problems, each separated by a blank line of data. If no input file name is given then the program requests the user to enter the data from the keyboard. The output file is the file to which the solutions are written. If no output file name is given then the output is directed to the terminal instead of a file. For the number of solutions desired, the program accepts two variables representing the range of solutions to be printed for

each problem. For example, if the range is given as "5,10" then the program looks for ten solutions to each problem but prints only solutions number five through ten. If the second number is not given or if it is less than the first, then all solutions up to the first number are printed.

After the data has been entered for a problem, it is reordered in order of decreasing frequency. In general this provides for a shorter execution time. Before calling the other routines, the number of channels at each frequency is sent to the output file.

The main program makes a subroutine call to SETUP to calculate divide rates and initialize parameters. Next SOLVE is called. SOLVE prints the desired solutions as they are found and returns when this is done or when no more solutions can be found with this system clock frequency. If SOLVE returns without finding the required number of solutions then the system clock frequency is increased, and the divide rates are recomputed. This results in more dummy channels. SOLVE is then called again to find the remaining number of solutions. When SOLVE returns and the desired number of solutions has been found, then the next problem is read from the file and the procedure is repeated. If there are no more problems in the file or if the data was entered from the keyboard, then all files are closed and the program stops.

As was mentioned previously, subroutine SETUP computes the system clock frequency and divide rate for each channel. Its calculations are based on Equations 1 through 4. In addition to divide rate, each channel has two other values assigned to it in this subroutine. The first is in an array IVAL which indicates which frequency group the channel belongs to. The second is in an array NMAX which indicates the highest starting location a channel can have. These arrays are used to speed execution.

They are discussed in detail in the next paragraph. Another array which is calculated in this subroutine is MGCF. This array contains the greatest common factor of the divide rates for each pair of frequency groups. This array is not absolutely necessary but its use will also speed execution. As this array is produced each greatest common factor is checked. As discussed in Chapter III, the greatest common factor for each pair of divide rates must be greater than one in order for the problem to work. If it is one for any pair, then the system clock frequency is increased and the program loops back to recompute the divide rates and other variables.

The SOLVE routine implements the major part of the algorithm by computing a starting location for each channel such that Equations 5 and 6 are satisfied. The arrays IVAL, NMAX, and MGCF are used for faster execution. Instead of having to recompute greatest common factors each time in testing Equation 6, IVAL and MGCF are used. IVAL indicates to which frequency group each channel belongs. Its values therefore range from one to the number of frequency groups. MGCF is the array of greatest common factors for each pair of divide rates. Using this array the greatest common factor for a pair of divide rates is found by addressing the proper element. NMAX is used for faster execution as follows. In the operation of this program the channels are grouped by frequency. In addition each channel's starting location is assigned a higher value than the one assigned to the previous channel from that frequency group. Because of Equation 5, each channel in a frequency group must have a starting location less than or equal to the divide rate. No two channels may have the same starting location. Therefore, the highest starting location a channel can have is fixed by the divide rate and the number of channels in that group yet to be assigned. This maximum value for each

channel is stored in NMAX so it won't have to be recomputed each time. Other than the addition of these three arrays, the subroutine follows the flowchart of Figure 4.1, page 49. It prints each solution as it is found and returns when all desired solutions are found or when it can find no more solutions using the given divide rates.

The program is simple in its design and operation, but as will be shown in Chapter V, it requires too much execution time for most problems with very many channels. The algorithms discussed next use the wheel approach. Their operation proved much more satisfactory.

ZAP - A Wheel Approach

During the course of this project there were three major approaches developed to solve scan tables using wheel diagrams. The FORTRAN implementations of these are in programs ZAP, ZIP, and SCAN. The SCAN programs were developed first and proved unsatisfactory. They required an excessive amount of execution time and memory. The ZIP algorithms were developed next and showed a marked improvement in speed in addition to requiring less memory. ZAP was developed last and proved to be superior to the ZIP algorithms in solving the average problem.

All of these approaches have the same basic program flow as shown in Figure 4.3. The difference in the approaches lies mainly in the routine which solves the problem.

This section describes the ZAP approach to solving a scan table problem. This is a wheel approach based on the simplification procedure discussed in Chapter III. The ZIP approach is very similar. Its differences will be described in the next section.

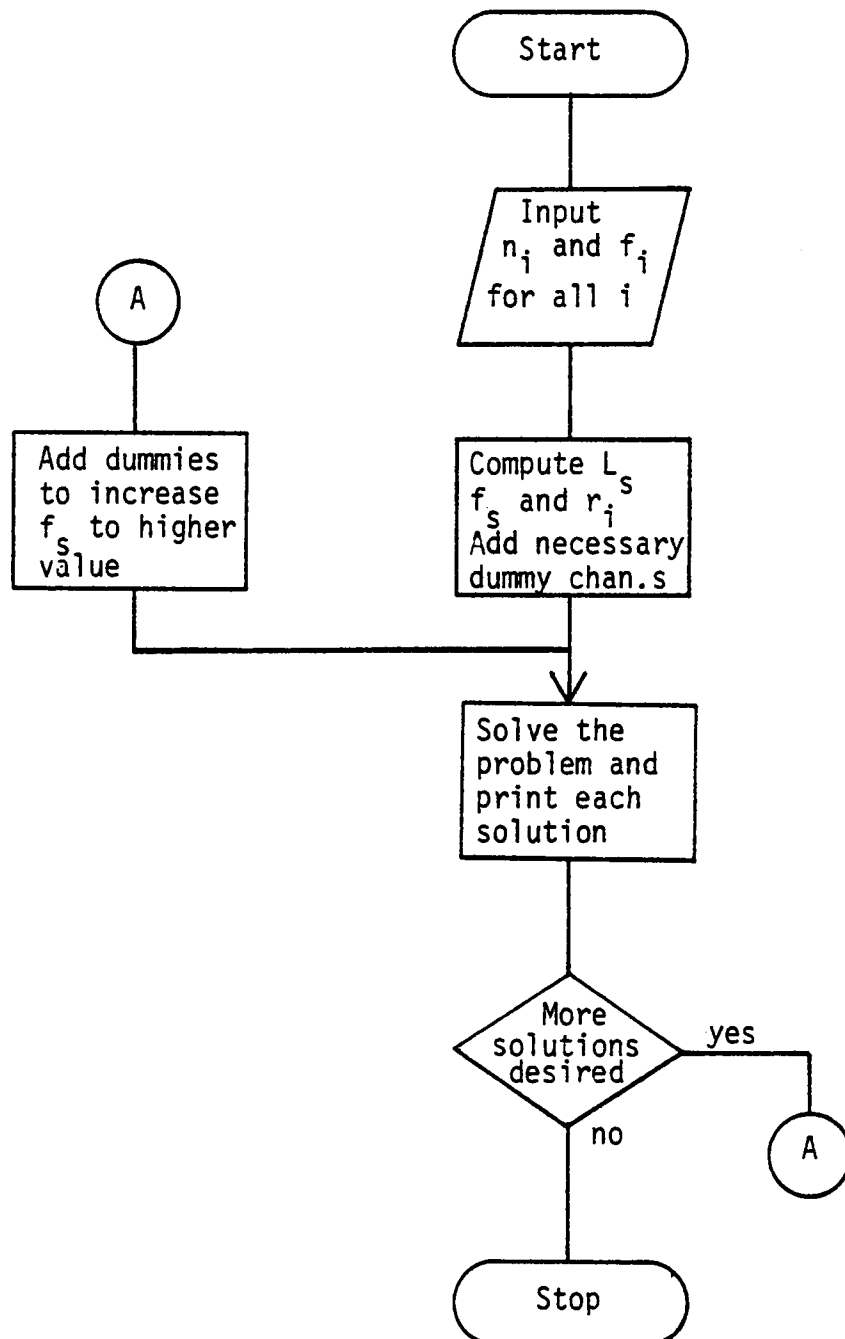


Figure 4.3. Basic program flow for the wheel approaches.

In view of the discussion on wheel based approaches in Chapter III, it is evident that a recursive routine is required to solve a problem which must be partitioned an arbitrary number of times. Given a problem that it cannot immediately solve, such a routine must be capable of dividing the problem into subproblems, storing all necessary information, and then the routine must call itself to solve each subproblem. Each subproblem is then treated as if it were the original problem. In addition if a deadend is encountered, then the routine must be able to backtrack by withdrawing a previous decision and continuing in a different direction. Because backtracking is necessary, the program flow and decision making must be systematic to avoid getting stuck in an infinite loop.

Due to the recursive nature of the routine, one must pay close attention to the use of the words problem, subproblem, and original problem in the following discussion. Generally the discussion follows one level of recursion. The word problem refers to the problem currently being partitioned. The subproblems are the subproblems of this problem.

The flowchart for the problem solving routine using the ZAP approach is shown in Figure 4.4. In this approach as with the other wheel approaches, the channels are grouped according to frequency. The internal representation of the problem and its subproblems includes primarily the number of channels in each frequency group and the frequency and divide rates of the frequency group. The channels are assigned to slots on the wheel diagram after the wheel diagram is found.

When this routine is given a problem containing more than one frequency group, it first divides the problem into a set of subproblems, possibly some of which do not have solutions. For the problem the divide rates are

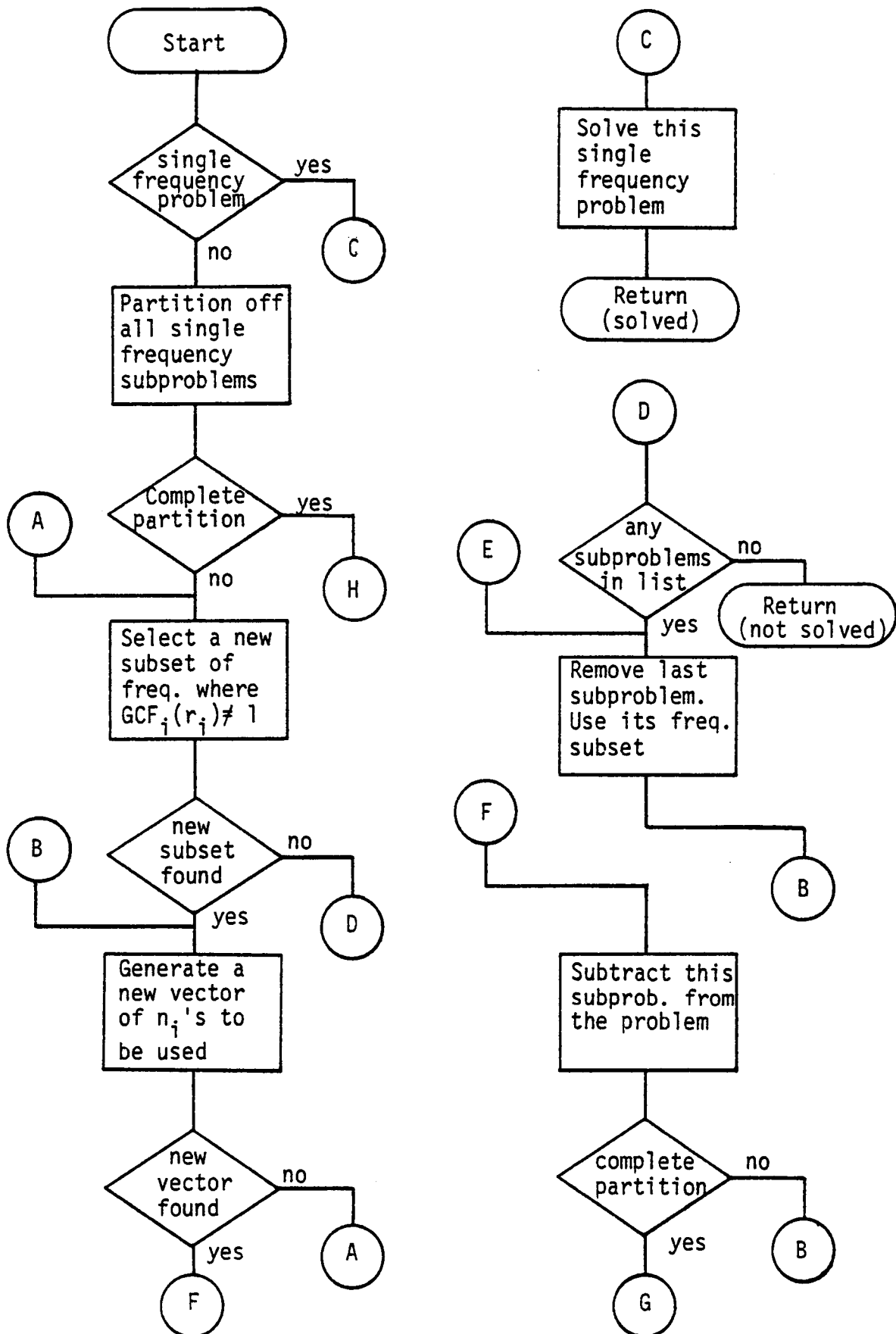


Figure 4.4. The SOLVE routine for ZAP.

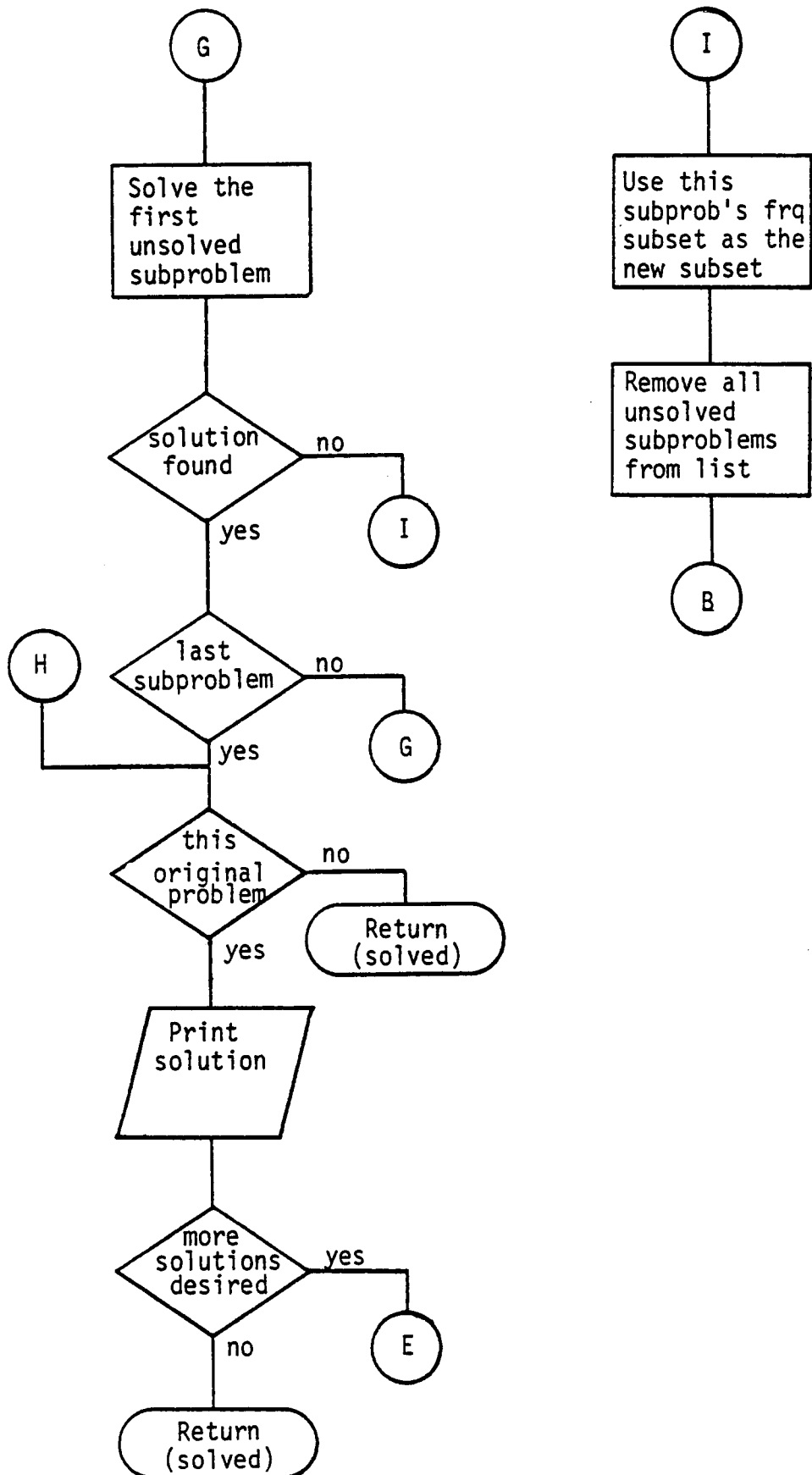


Figure 4.4. (Continued).

known and the number of subproblems is the greatest common factor of these divide rates. The routine later solves the subproblems by calling itself.

In selecting a set of subproblems, those subproblems containing a single frequency group are tried first. As each subproblem is determined it is subtracted from the problem at hand. The problem then represents what further subproblems are required. If the problem cannot be partitioned using single frequency subproblems, then subproblems containing channels of two, three, etc. frequency groups are searched for until the problem has been completely partitioned. Again as each subproblem is determined, it is subtracted from the original problem. When the given problem is reduced to zero, then its set of subproblems is complete and the routine may begin solving each subproblem.

Referring to Figure 4.4 the first thing the problem solving routine does is compute single frequency subproblems. This is a relatively simple task. From the outset the divide rates of all the channels are known. The greatest common factor of these is the number of subproblems. Dividing each divide rate by this greatest common factor yields the channel's divide rate in any subproblems. The wheel representation of single frequency subproblems consists of a single wheel of the proper size. All that is required to create such a subproblem using a given frequency group is that there be enough channels at that frequency in the problem to completely fill that wheel. Thus in this section of the problem solving routine, the number of channels in each frequency group is compared to the corresponding subproblem divide rate. If the number of channels is greater, then a subproblem is created using that frequency group. The number of channels in the subproblem is equal to the wheel size. This is done for each frequency group as many times as possible. Each time such a subproblem is

created, the number of channels in that frequency group is subtracted from the given problem. After all such single frequency subproblems are used, the algorithm checks to see if the problem is completely partitioned. If so it begins to solve these simple subproblems. Otherwise the problem is not fully partitioned and the algorithm begins looking for multiple frequency subproblems.

The next section of the routine will generate a new subset of frequencies to be used in a potential subproblem. This search for a subset is done in a systematic manner following a few simple rules. A convenient way to represent the subset of frequencies used in a subproblem is to assign a binary digit to each frequency group. This digit is one if that frequency group is used in a subproblem and zero if it is not. By stringing the digits together a binary number is formed with a number of digits equal to the number of frequency groups. Now the rules for determining the next subset of frequencies can be explained as follows. Given the binary number representing a subset of frequencies, then the next smaller binary number containing the same number of nonzero digits represents the next subset of frequencies. If no such binary number exists then the number of frequencies in the subset is increased by one. Consequently the next subset of frequencies is represented by a binary number with an additional nonzero digit. This new number is chosen to be the highest binary number with this number of nonzero digits. In many cases the representation of a problem may contain frequency groups that have no channels in them. A general algorithm may still include these frequency groups in its representation of the problem if these groups are excluded from all frequency subsets. Thus in determining the next binary number, all binary numbers with a one for the digit corresponding to these

frequencies are skipped. If no new subset of frequencies can be found, then the program must backtrack. Otherwise the algorithm moves on to the next phase. Backtracking involves going back and redefining the previous subproblem. In redefining this subproblem it keeps its same subset of frequencies. Only the number of channels at each frequency are changed. This will be described later.

Once a subset of frequencies is found it is determined if a subproblem with that subset of frequencies can be further partitioned. This is done by taking the greatest common factor of the divide rates for the subproblem. As discussed in the last chapter, this number must be greater than one in order for the subproblem to be further partitioned. If it is equal to one then the search is resumed for a new subset of frequencies. Otherwise the algorithm begins the next task of determining the number of channels to be included at each frequency of the subproblem.

The next section of the routine generates the number of channels in each frequency group for the subproblem. This set of numbers must satisfy two constraints. The first is that the set must satisfy Criteria 2 in Figure 3.6, page 33. The second requirement is that the number of channels in a frequency group of the subproblem must be less than or equal to the number of channels remaining in the same frequency group of the problem being partitioned.

This section of the routine has two entry points. The first is used when the subset of frequencies is used for the first time. The other entry point is used when the previous subproblem contained the same frequency subset. In the latter case the routine starts with the number of channels in each frequency group for the previous subproblem and generates another combination of numbers. To avoid choosing the same combination twice, the

selection proceeds systematically by determining the numbers one at a time in some order. As each new number is chosen then the product of it times the corresponding frequency is added to a total yielding the partial dot product. After all numbers are determined then this total should equal to fs_0/g_0 as given by Criteria 2 in Figure 3.6, page 33. As each new number is determined it is given the maximum value possible so that it does not exceed the number of channels in that frequency group in the given problem and it does not push the partial dot product to a value greater than fs_0/g_0 . Because all of the frequency groups in this subset must be used in this subproblem, it is required that each number chosen must be at least one. If at any point the number chosen for a frequency group is zero then the previous number chosen is decreased by one, the partial dot product is readjusted and a new value is determined for the new number. As the last number is chosen the dot product should be fs_0/g_0 . If it is not then again the previous number is decreased by one, the partial dot product is readjusted and a new value is determined for the last number. This continues until the dot product equals the desired number or the possible choices run out. If no combination of numbers is found the routine branches back to determine a new subset of frequencies. If a combination is found then the numbers represent a subproblem.

Once a subproblem is found it is subtracted from the original problem as many times as possible. Since the number of channels in each frequency group for the subproblem was chosen less than or equal to the number of channels in the given problem then the subproblem should subtract from the problem at least once. However, it is possible that the subproblem can be subtracted more than once in which case it is used as a subproblem that number of times. If the given problem has not been reduced to zero at this

point then the algorithm loops back to find another subproblem using this same subset of frequencies. If the given problem has reduced to zero then it has been completely partitioned and the routine begins solving each of the subproblems.

Because of the recursive nature of the routine, the subproblems may be solved by the routine calling itself, now treating the subproblem as the original problem. First, however, parameters for the subproblem such as divide rate for each frequency group and the system clock frequency for the subproblem are computed. If the routine returns with a solution then the next subproblem is tried. If all subproblems were solved then the given problem as a whole was solved and the routine prints the solution if it was the original problem or returns otherwise. If the subproblem was not solved then the routine must backtrack to that point. All subproblems from there on are added together to reconstruct the given problem at that point. All previous subproblems are kept along with their solutions. The routine then branches back to generate a new subproblem using the same frequency subset as in the unsolved subproblem. From there the routine goes on by redefining the remaining subproblems. After all the subproblems are redefined then the routine begins at the first unsolved subproblem and continues solving as before.

Example 1 is repeated below to demonstrate this algorithm. The dummy channels are already added before entering the routine. The system clock frequency and divide rates are already known.

$$\overline{N}_0 = (1,3,4,7)$$

$$\overline{F}_0 = (30,15,10,5)$$

$$\overline{R}_0 = (5,10,15,30)$$

$$fs_0 = 150 \text{ Hz}$$

Compute:

$$g_0 = \text{GCF}_i(r_i) = 5 \quad \begin{array}{l} i = 1, 2, 3, 4 \\ n_i \neq 0 \end{array}$$

$$fs_j = fs_0 / g_0 = 150 / 5 = 30$$

$$\bar{R}_j = (1, 2, 3, 6)$$

This says there are five subproblems, each of which has a system clock frequency of 30 Hz. The divide rate for each frequency group in the subproblems are one-fifth the group's original divide rate. Now the algorithm first searches for single frequency subproblems. These are possible when $n_{oi} \geq r_{ji}$. This is true for the first frequency group and the following subproblem is formed.

$$\bar{N}_1 = (1, 0, 0, 0) \quad \bar{R}_1 = (1, , ,) \quad fs_1 = 30 \text{ Hz}$$

Subtracting this from the given problem one gets

$$\bar{N}_0 = (0, 3, 4, 7) \quad \bar{R}_0 = (5, 10, 15, 30) \quad fs_0 = 150 \text{ Hz}$$

Another single frequency subproblem exists using the second frequency group. This subproblem is

$$\bar{N}_2 = (0, 2, 0, 0) \quad \bar{F}_2 = (, 2, ,) \quad fs_2 = 30 \text{ Hz}$$

Now this is subtracted from the given problem yielding

$$\bar{N}_0 = (0, 1, 4, 7) \quad \bar{R}_0 = (5, 10, 15, 30) \quad fs_0 = 150 \text{ Hz}$$

A third single frequency subproblem can be obtained from the third frequency group. It is as follows:

$$\bar{N}_3 = (0, 0, 3, 0) \quad \bar{R}_3 = (, , 3,) \quad fs_3 = 30 \text{ Hz}$$

This is subtracted from the given problem yielding

$$\bar{N}_0 = (0, 1, 1, 7) \quad \bar{R}_0 = (5, 10, 15, 30) \quad fs_0 = 120 \text{ Hz}$$

Finally, a fourth single frequency subproblem can be obtained from the fourth frequency group. It is

$$\bar{N}_4 = (0,0,0,6) \quad \bar{R}_4 = (, , ,6) \quad fs_4 = 30 \text{ Hz}$$

Subtracting this from the given problem yields

$$\bar{N}_0 = (0,1,1,1) \quad \bar{R}_0 = (5,10,15,30) \quad fs_0 = 120 \text{ Hz}$$

What remains of the original problem must now form the final subproblem. As the algorithm proceeds it finds that this fifth subproblem has no solution. This is because the greatest common factor of the divide rates included is equal to one (i.e. $GCF(2,3,6)=1$). Therefore the algorithm will backtrack and remove the fourth subproblem from the list. The original problem is again described as

$$\bar{N}_0 = (0,1,1,7) \quad \bar{R}_0 = (5,10,15,30) \quad fs_0 = 120 \text{ Hz}$$

Now there are still two subproblems to be found. Since all single frequency subproblems were found, the algorithm now begins looking for a two frequency subproblem. In this case a subset of frequencies with two elements is represented by a four digit binary number containing exactly two ones. To begin the highest such number is tried first. It is 1100. This number and also the next lower two such numbers, 1010 and 1001, all require a channel from the first frequency group. At this point there are no channels in the first frequency group, so these three numbers are skipped. The next lowest binary number with two ones is 0110. Both frequency groups represented contain channels, so the subset represented is acceptable so far. Next the greatest common factor of the divide rates is checked. In the created subproblem, the divide rate for the second

frequency group is 2. The divide rate for the third frequency group is 3. The greatest common factor of these is one. This is not acceptable in a subproblem so this subset of frequencies is not used and the search is continued for a new subset. The next lowest binary number is 0101. The divide rates for the represented frequency groups are 2 and 6. The greatest common factor of these is 2 so the subset is valid and the algorithm now searches for a number of channels to be used at each frequency.

As described before, the dot product formed by the number of channels with the frequency of each should equal to fs_j in each subproblem. In addition the maximum number of channels which can be in the second and fourth frequency groups are 1 and 7 respectively.

From these two requirements the maximum number of channels in the second frequency group is one. The partial dot product at this point is then $1 \times 15 = 15$. The maximum value of the number of channels in the fourth frequency group would be three. This number is less than the maximum of seven and it brings the dot product to the desired value of 30.

$$\bar{N}_4 \cdot \bar{F}_4 = 1 \times 15 + 3 \times 5 = 30 = fs_j$$

Therefore the fourth subproblem is defined as

$$\bar{N}_4 = (0, 1, 0, 3) \quad \bar{R}_4 = (, 2, , 6) \quad fs_4 = 30$$

Subtracting this from the given problem yields

$$\bar{N}_0 = (0, 0, 1, 4) \quad \bar{R}_0 = (5, 10, 15, 30) \quad fs_0 = 150 \text{ Hz}$$

One final subproblem is yet to be found and it must consist of the remainder of the problem. However the algorithm must proceed in a systematic manner to assure that the subproblem has not already been tried in the event that multiple solutions are desired. The algorithm proceeds

by searching for a new set of numbers at each frequency using the frequency subset of the fourth subproblem. Since the second frequency group was in this subset and it now contains no channels then no combination is possible using this subset. The algorithm then searches for a new frequency subset. The binary representation for this subset was 0101. The next subset would be represented by 0011. This is a valid subset as the problem still has channels in these frequency groups. The divide rates for these frequency groups in the subproblem are 3 and 6. The greatest common factor of these is three therefore the subset is still valid and the program determines the number of channels used in each frequency group in the subproblem. Following the same procedure used for the fourth subproblem, the fifth subproblem becomes

$$\overline{N}_5 = (0,0,1,4) \quad \overline{R}_5 = (, ,3,6) \quad fs_5 = 30 \text{ Hz}$$

The original problem reduces to

$$\overline{N}_0 = (0,0,0,0) \quad \overline{R}_0 = (5,10,15,30) \quad fs_0 = 150 \text{ Hz}$$

and as a result it has been completely partitioned. Now the routine begins solving the subproblems. The first three subproblems contain a single frequency and their solution is already known. The fourth subproblem is now solved by recursion in the algorithm. This subproblem is treated as was the original problem and is described as

$$\overline{N}_0 = (0,1,0,3) \quad \overline{R}_0 = (,2, ,6) \quad fs_0 = 30\text{Hz}$$

$$g_0 = \text{GCF}_i(r_i) \quad \begin{matrix} i = 1,2,3,4 \\ n_i \neq 0 \end{matrix}$$

$$= \text{GCF}(2,6) = 2$$

This problem is partitioned into two subproblems. The divide rates for the frequency groups in the subproblems and the system clock frequency are then

$$\bar{R}_j = (, 1, , 3) \quad fs_j = 15\text{Hz}$$

Taking single frequency subproblems first, it is found the second frequency group is a candidate, as the number of channels is greater than or equal to the divide rate in the subproblem. The first subproblem becomes

$$\bar{N}_1 = (0, 1, 0, 0) \quad \bar{R}_1 = (, 1, ,) \quad fs_1 = 15\text{Hz}$$

Subtracting this from the given problem yields

$$\bar{N}_0 = (0, 0, 0, 3) \quad \bar{R}_0 = (, 2, , 6) \quad fs_0 = 30\text{Hz}$$

The second subproblem also contains a single frequency.

$$\bar{N}_2 = (0, 0, 0, 3) \quad \bar{R}_2 = (, , , 3) \quad fs_2 = 15\text{Hz}$$

Subtracting this from the problem yields

$$\bar{N}_0 = (0, 0, 0, 0) \quad \bar{R}_0 = (, 2, , 6) \quad fs_0 = 30\text{Hz}$$

and the problem is fully partitioned. Since the subproblems each contain a single frequency, their solution is immediately known. The solution to the problem is then known. In wheel form this solution is shown in Figure 4.5.

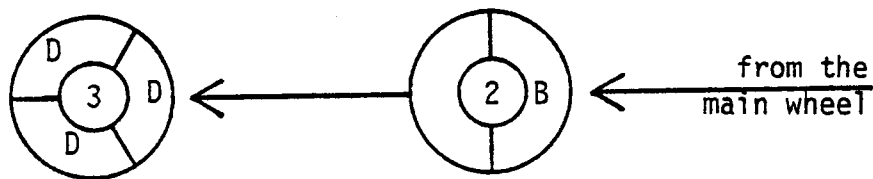


Figure 4.5. Wheel form of solution to the fourth subproblem.

The algorithm now returns with this fourth subproblem solved. It then solves the fifth subproblem in a similar manner yielding Figure 4.6 for its solution.

These five subproblems are then combined to form the complete solution for the original problem as shown in Figure 4.7.

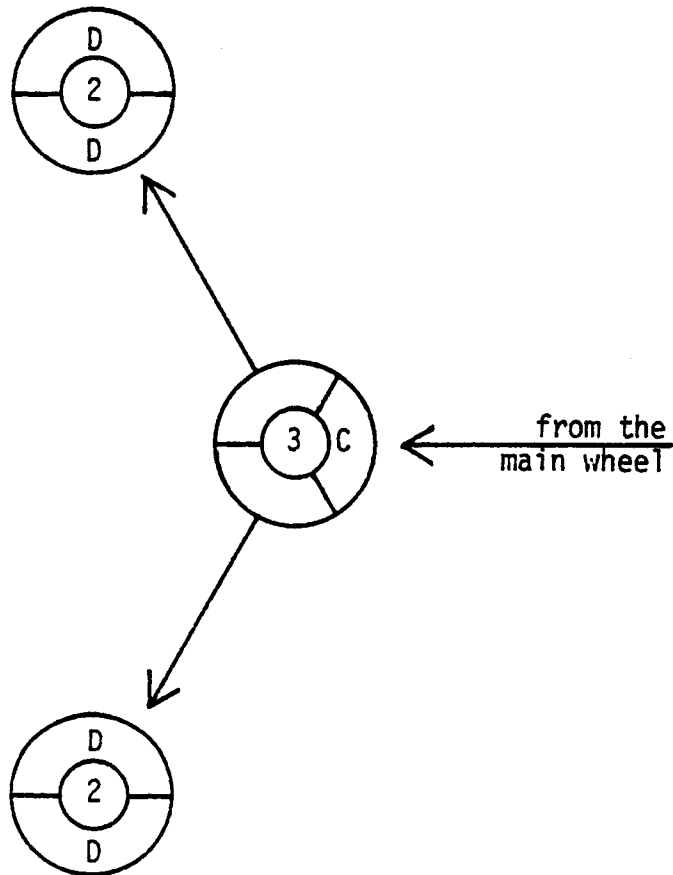


Figure 4.6. Wheel form of solution to the fifth subproblem.

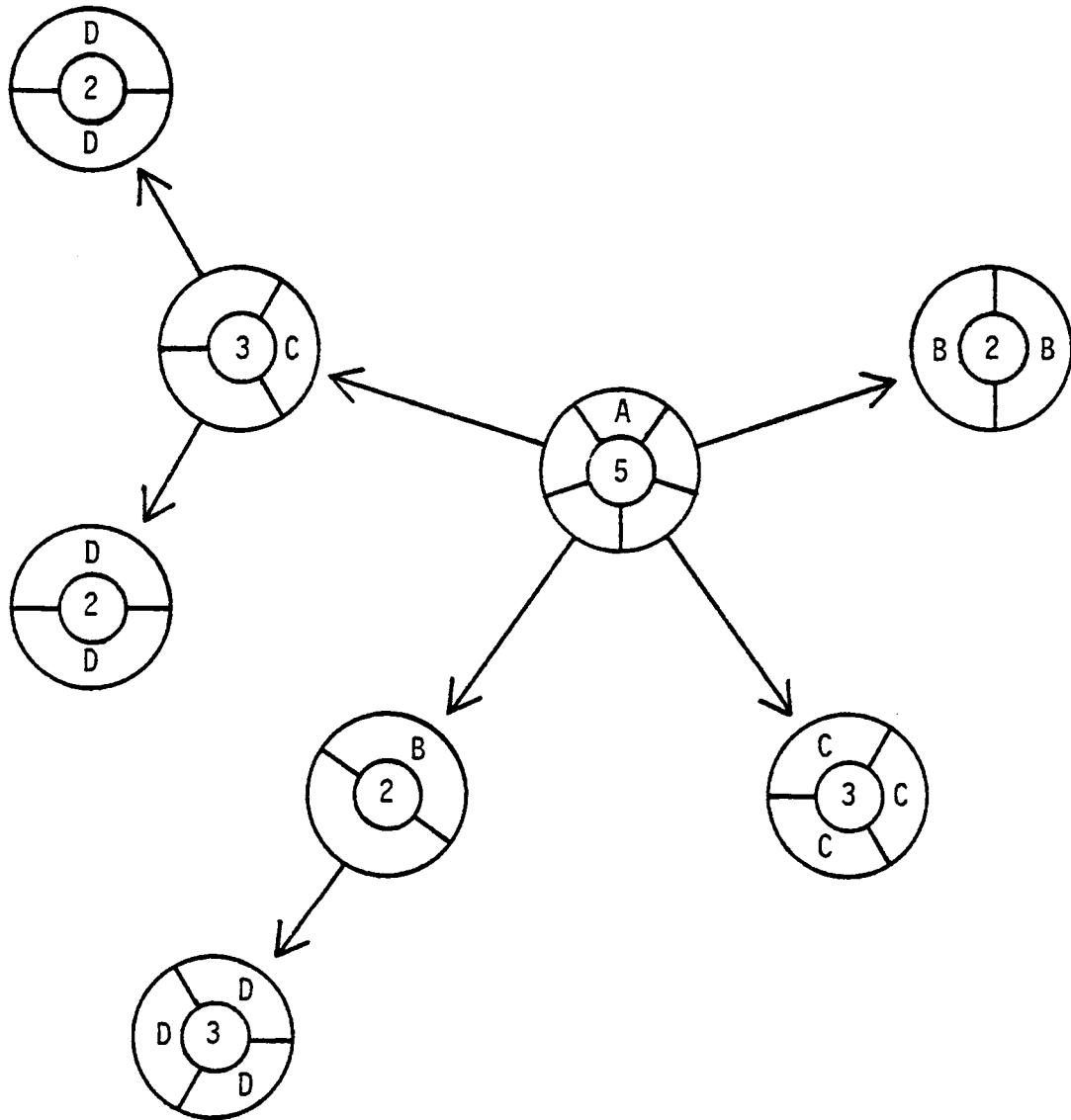


Figure 4.7. Solution of the subproblems combined to form a solution of the original problem.

At this time the channels of each frequency group may be assigned to an appropriate slot on the wheel diagram. In this way the solution of Figure 3.10, page 41, may be obtained.

Appendix I includes a FORTRAN implementation of this algorithm. A brief discussion of this implementation will now be given.

The main program and set up routine are almost identical to the ones used in the FORTRAN versions of the SIMP routine. The arrays used however are slightly different. Since the routine used to solve the problem is recursive in nature, it is best to store all the data and problem parameters in several stacks.

The arrays N, IDIV, and ITOT are used for the purpose of storing problem parameters. N is the array which stores the number of channels at each frequency in a subproblem. IDIV is the divide rate for each channel of a subproblem. Finally ITOT indicates the system clock frequency for each subproblem. NPS is a stack pointer which indicates the level of subproblems currently being worked on.

In addition to the problem stacks there are also data stacks which hold the solutions to the subproblems. The major ones of these are LSP and ISTN. LSP is a long stack whose width is equal to the number of frequency groups. Each row of LSP contains the number of channels in each frequency group for some subproblem. The rows are grouped according to subproblem. Each time a new subproblem is started its N vectors for its subproblems are added to the bottom of the stack. To keep track of this grouping an array LSTR is used. LSTR is actually part of the problem stack. Each element of LSTR indicates which row of LSP is the beginning of the N vectors for the subproblems of a problem. Thus for problem I, the N vectors for its subproblems are stored in rows of LSP, where the rows range from LSTR(I) to

LSTR(I+1)-1. Another array, ICUR, tells which subproblem in each group is currently being solved.

The solutions for a problem represented by a row of LSP are stored in an array ISTN. ISTN is a long array with a width of three. It represents a string form of the tree-like wheel diagram for a subproblem. Each row of ISTN describes a pointer to one or more wheels. The first element of the three tells how many times that subproblem is used. The second indicates the wheel size of the subproblem. This is the same as the number of subproblems into which the subproblem is partitioned. The third element equals the number of channels which lie directly on the wheel. This is the same as the number of subproblems with a single channel of divide rate equal to one. If there are spokes on the wheel other than these channels directly on the wheel then they are described by the following rows of ISTN. As ISTN is scanned from start to finish, each row describes a spoke of the most recent wheel which has not yet been completely described. An example of ISTN is given by the following string.

```

1,  4,  1
    2,  6,  3
        2,  2,  2
            1,  3,  3
                1,  8,  5
                    3,  3,  0
                        2,  3,  3
                            1,  2,  2

```

The rows of this string have been staggered to aid in understanding which rows are a part of a previous row. The wheel form of the output of ZAP is also staggered. Staggering the rows however is not essential in its interpretation.

The above string is read as follows. To begin with there is one 4-wheel with one channel directly on the wheel. Therefore there are yet three spokes to determine. The next row begins description of two of these subproblems. It says there are two 6-wheels with three channels on each. The two 6-wheels have identical subproblems. Three of the subproblems are yet unspecified. The third row indicates that from each 6-wheel are two 2-wheels with two channels on each. The 2-wheels are completely specified so the next row again describes a spoke of the 6-wheel. The next row indicates a single 3-wheel with three channels on it. The 3-wheel is completely described. This wheel also completes the description of the two 6-wheels. To this point the description is a single 4-wheel with one channel on it and two 6-wheels. Each of the 6-wheels has three channels on it in addition to two 2-wheels and one 3-wheel. The 4-wheel which was started with still has another spoke to be described. The fifth row begins its description. This fourth spoke is a single 8-wheel with five channels on it. The next row says the 8-wheel has three 3-wheels on it, each of which has no channels directly on the wheel. These 3-wheels still need to be described. The last two lines describe the spokes of each 3-wheel as being two 3-wheels with three channels on each, and one 2-wheel with two channels on it. Now all spokes on all wheels have been described. The wheel form of the string is given in Figure 4.8.

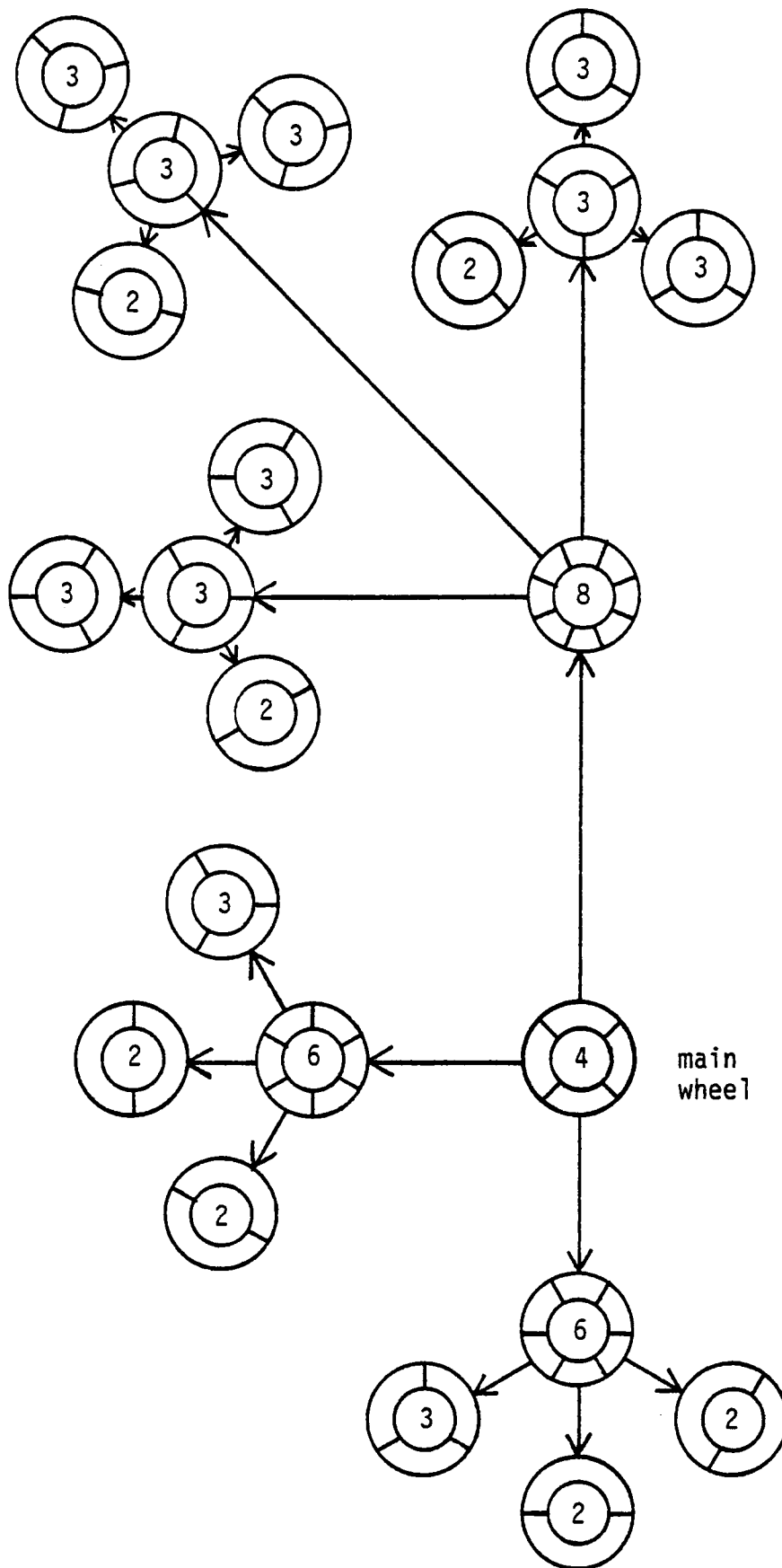


Figure 4.8. Wheel form corresponding to the given string form.

Note that a property of this string form is that the subproblems appear in the order that they are solved. As a solution to each subproblem is found, it is placed on the end of this array. Thus the array builds as the subproblems are found without requiring any internal juggling.

As mentioned previously ISTN contains a string form of a solution to a subproblem represented by a row of LSP. The array IST contains one element for each row of LSP. This element indicates which row of ISTN begins the solution to the row of LSP. Thus for row K of LSP, the subproblem represented has a solution in ISTN where the solution's row numbers range from $IST(K)$ to $IST(K+1)-1$.

In addition to IST, each row of LSP also has two other variables associated with it. These variables are in the arrays MA and NTIMS. MA indicates for the subproblem the size of the wheel it represents. NTIMS indicates how many times a subproblem is used for a problem. This is used so a subproblem will have to be solved only once even though it is used many times for a problem.

One other array used in this program is MASK. This is a logical array that is used to indicate which frequency groups are used in the current problem being solved.

For the most part the program is identical to the algorithm described in this section. However, a few modifications were made to shorten execution time. One such modification was a special section to be used when the last subproblem of a group was being determined. In this case instead of determining the last subproblem by searching, this special section is used. All the section has to do is determine if the remainder of the problem is a valid subproblem. It also determines if the subproblem is one that has already been tried. If it has been tried then the program

rejects the subproblem because the subproblems must come in a definite order.

Another section that was added to increase speed involves backtracking. If in the process of backtracking the routine removes a subproblem containing a single channel at a single frequency, then the routine returns with no solution. This is because subproblems with a single channel must be used if possible. If a single channel in a problem fulfills the requirements for a subproblem then there is no possibility of using the channel any other way. Because of the order in which subproblems are created, single subproblems are defined first and therefore removed by backtracking last. If the routine backtracks far enough to remove such a subproblem, no further solutions exist and the routine exits.

One major source of improvement in execution time came from a prechecking section. As the SOLVE routine begins with a problem or subproblem it makes a simple check which indicates for some problems that they have no solution. Consider the following as an example.

$$\bar{N}_0 = (1,4,1) \quad \bar{F}_0 = (3,2,1) \quad fs_0 = 12 \quad \bar{R}_0 = (4,6,12)$$

The problem may seem to be a valid subproblem, but as simple as it is, it has no solution. For the problem, $GCF_i(r_i) = GCF(4,6,12) = 2$. There are therefore two subproblems and the divide rates of the channels in the subproblems are 2, 3, and 6. In terms of wheel diagrams this means there is a 2-wheel in the center and the channels must have further divide rates of 2, 3, and 6. Consider the single channel in the last frequency group. It could be on a 6-wheel attached directly to the 2-wheel in the center, but that would require five other channels filling up the remaining slots on the 6-wheel. This channel could be on a 3-wheel attached to a 2-wheel

which is attached to the 2-wheel in the center, but again that would require an additional two channels to fill the empty slots on the 3-wheel. Finally it could be on a 2-wheel attached to a 3-wheel attached to the 2-wheel in the center. Still this would leave one empty slot on the 2-wheel that channel is on. Therefore there is no place the channel in the third frequency group can be placed on the wheel diagram without leaving empty slots on a wheel diagram. Since the problem is now completely specified, this means other channels will be left without a place to be put. Therefore the following general statement may be made. The number of channels in each frequency group must be at least as big as the smallest wheel the channel can be placed on; however, if other wheels can come off of that wheel then the restriction does not apply.

The wheel size of a frequency group when used in conjunction with another frequency group is a constant throughout a problem. For a channel of frequency group i to be used in conjunction with a channel of frequency group j , the following wheel size is required for the i group.

$$r_i / \text{GCF}(r_i, r_j)$$

Note that if channels of group j are on a wheel coming off the wheel channels of group i are on, then r_j is a multiple of r_i , and this value is one.

The smallest wheel size available for frequency group i is found by taking the minimum value of the above expression for all values of j where frequency group j is used in the problem or subproblem. If any channels in group i are included in a problem or subproblem then their number must be

at least this minimum wheel size. This is the check that was added to the beginning of the SOLVE routine.

One additional feature of this program is that it will reduce all the dummy wheels as described in the previous chapter. This is easy to do when no other channels exist at the divide rate of the dummy channels. Otherwise an optimum assignment of dummy wheels is very difficult to achieve. To get around this problem the program always assigns the dummy channels a frequency not assigned to any other frequency group. For reasons discussed earlier the dummy channels are normally given a frequency of $GCF_i(f_i)$ and a divide rate $LCM_i(r_i)$. The difficulty arises when another frequency group has a frequency $GCF_j(f_j)$. Since frequency times divide rate equals the constant f_s , then assigning the dummy channels a frequency which is a submultiple of $GCF_j(f_j)$ will result in the divide rate being a multiple of $LCM_j(r_j)$. This would be an acceptable assignment if a submultiple exists. To assure that a submultiple of $GCF_j(f_j)$ exists, the program internally doubles each frequency as it is entered. This does not change the solution any except the system clock frequency is also doubled. As a result $GCF_j(f_j)$ must now be divisible by two since all frequencies were initially integers. Dummy channels can now be assigned an internal frequency of 1 Hz and no other channel will have this frequency. As a solution is found, all wheels with the corresponding divide rate are reduced and the system clock frequency is divided by two.

ZIP

The ZIP method is almost identical to ZAP except for a change in the order of execution. Again ZIP uses the same basic program flow shown in Figure 4.3, page 56. A flowchart for the SOLVE routine of ZIP is shown in Figure 4.9. Comparing this figure with the SOLVE routine of ZAP reveals

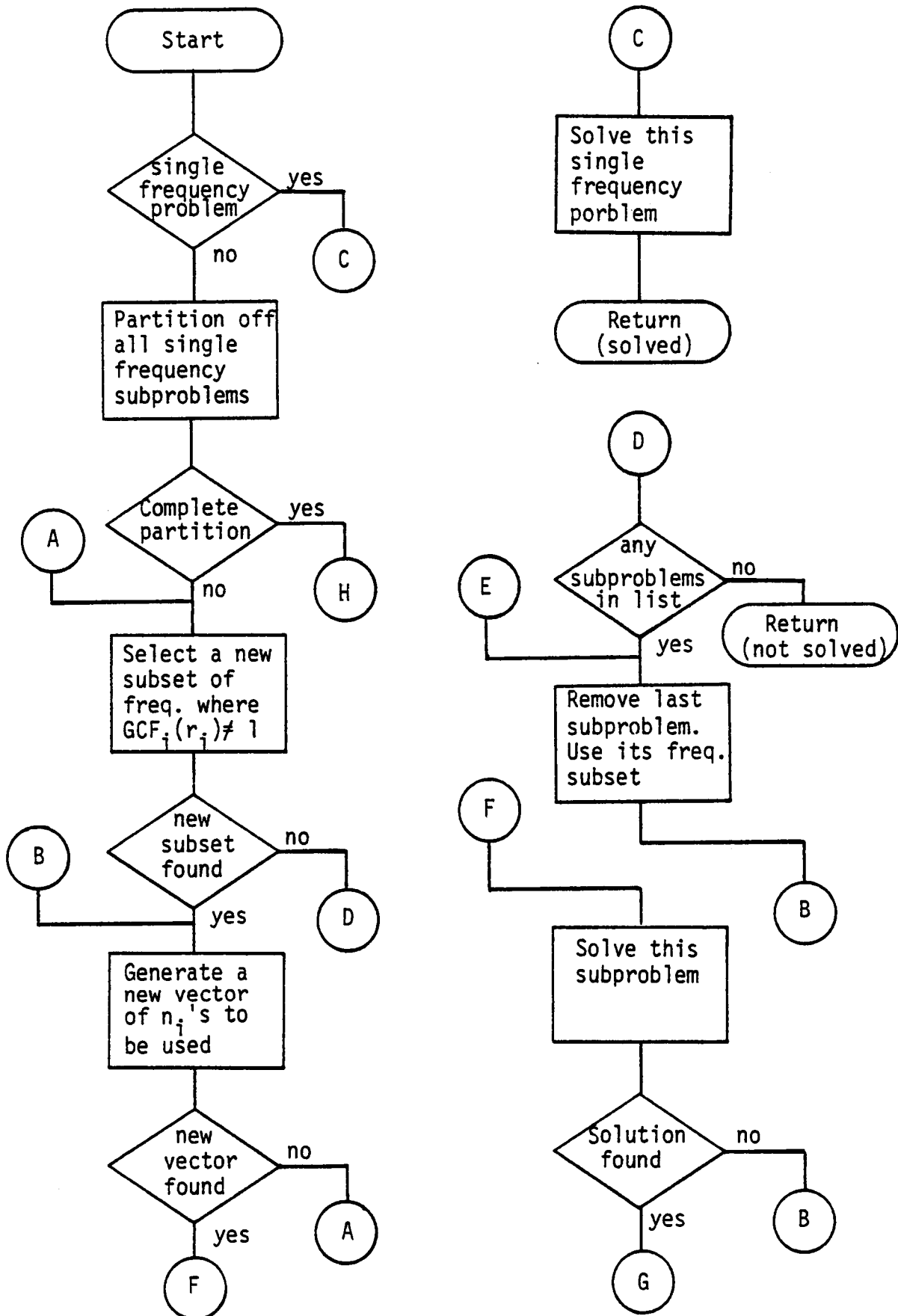


Figure 4.9. The SOLVE routine for ZIP.

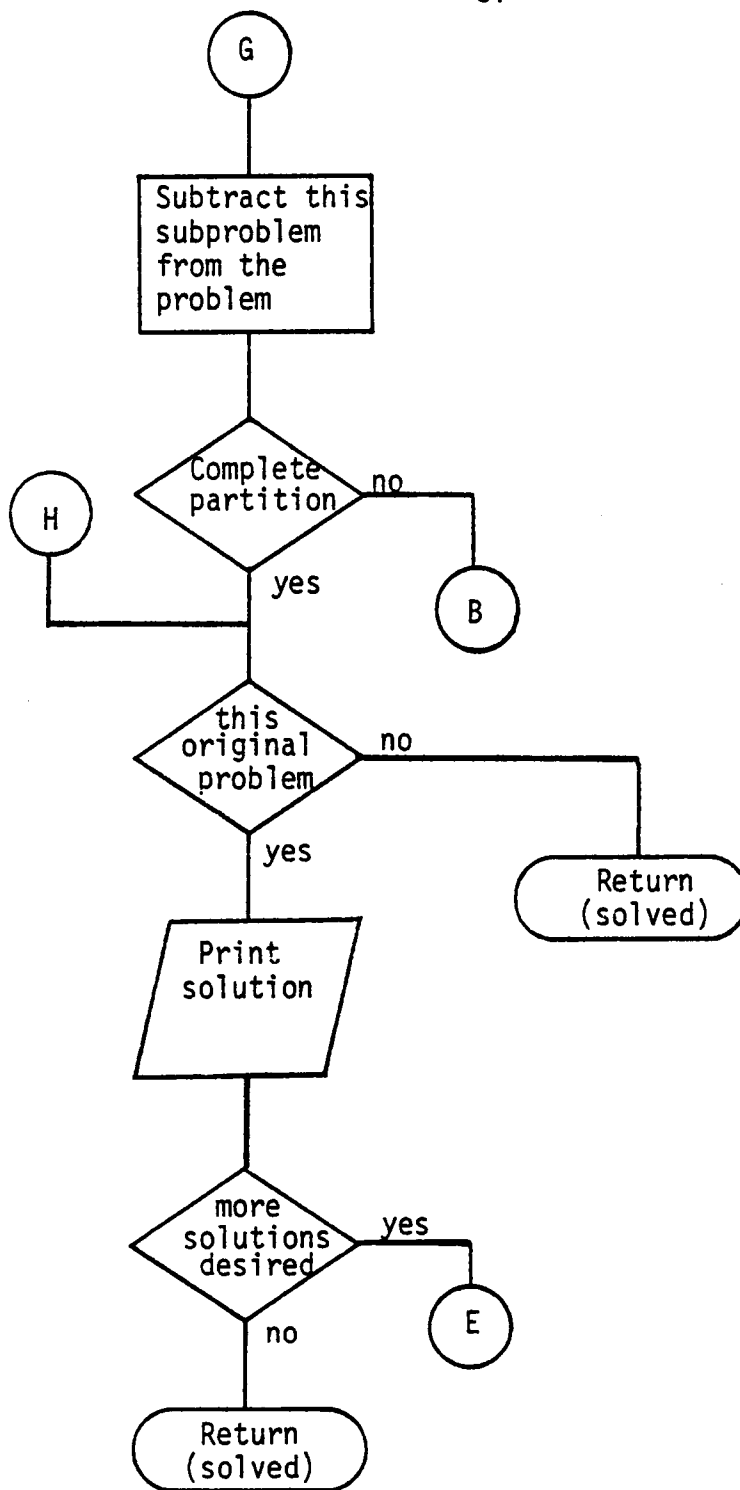


Figure 4.9. (Continued).

only one difference. The difference is that ZIP solves each subproblem as it is found before determining the remaining subproblems for a problem. ZAP on the other hand, finds the whole set of subproblems before attempting to solve any of them. Consequently, ZIP uses a depth-first search for solutions while ZAP uses a breadth-first search.

In ZIP each section of the algorithm has an identical function as the corresponding section of the ZAP algorithm. Consequently the FORTRAN implementations are very similar. In addition the speed improvements described for the ZAP algorithm are also used in ZIP with no differences.

SCAN

The SCAN programs constitute the initial attempts at creating wheel diagrams. They were very slow in execution; however, they laid the framework for later programs. As with the ZIP and ZAP programs, Figure 4.3, page 56, represents the basic program flow for the SCAN programs. Again the routine which solves the problem is recursive. The feature of SCAN that sets it apart from ZIP and ZAP is that the SOLVE routine, when called, returns with all solutions to the problem instead of just one.

Figure 4.10 depicts the basic flowchart for the solving routine. As the routine is entered it is given a subset of frequencies to be used in the problem. In addition the divide rate of each frequency is known. The number of channels at each frequency is not specified. The routine uses the number of channels at each frequency in the original problem only as maximum allowable values in the subproblem. Before returning, the routine creates a list of all possible problems which can be created using the given set of frequencies, their divide rates, and the maximum number of channels to be used in each. The creation of this list is a two step

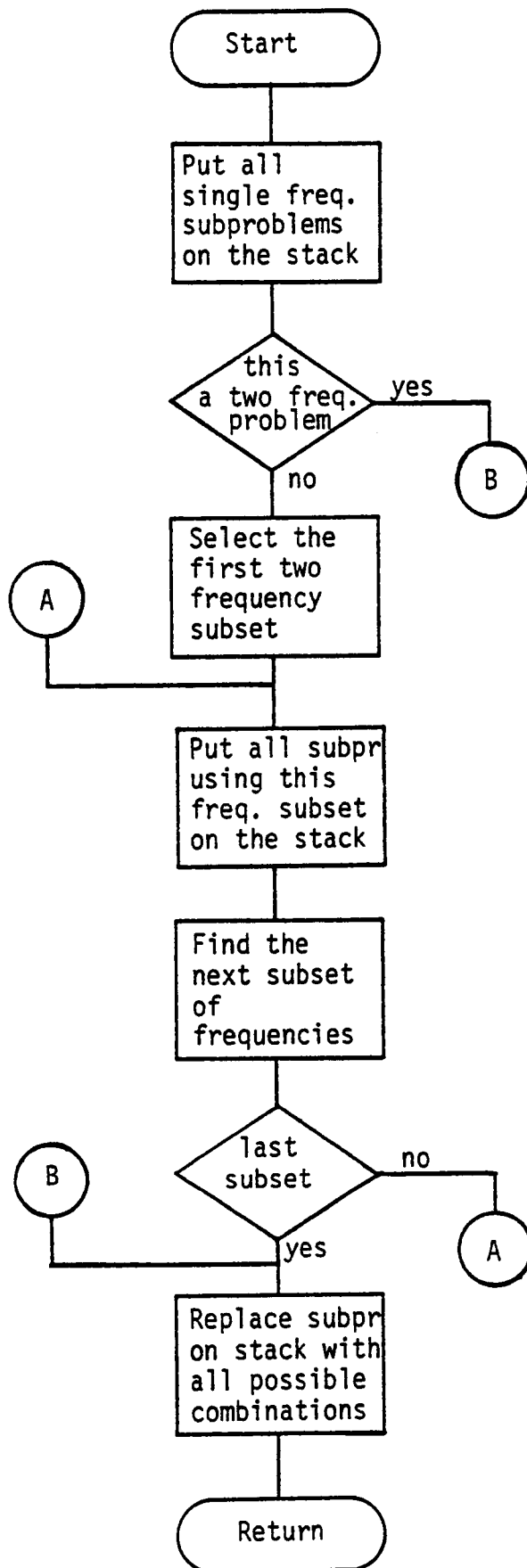


Figure 4.10. The SOLVE routine for SCAN.

process. First all possible subproblems using these frequencies and divide rates are created. Since the routine is recursive, the subproblems are found by the routine calling itself. After all possible subproblems are generated then all possible combinations of the subproblems are found. Each combination constitutes another solution to the problem.

In the FORTRAN implementation of this algorithm, a slight change was made in the execution of the SOLVE routine in the event that the problem was the original problem. In this case the routine would not wait until all possible subproblems were found before searching for a combination of subproblems to satisfy the problem. Instead it would search for a combination following each addition of subproblems to its list. This allowed some solutions to be printed before all possible subproblems were found.

CHAPTER V

RESULTS AND COMPARISON OF ALGORITHMS

Each of the algorithms presented in this report offers a method for producing scan tables for a given problem. As mentioned previously, the wheel method algorithms were developed in the order SCAN, ZIP, then ZAP. In fact the development of the SCAN and ZIP programs actually each involved about a half dozen versions. As each major improvement or alteration was added to a program, the new version was stored under a new file name. Some changes consisted only of differences in input and output format while others contained true speed improvements. This chapter presents a comparison of these methods by following a history of their development.

Individual files used in this comparison are given in Figure 5.1. An additional file used is BIGFIL which is a collection of sixty-five problems. This file is listed in Appendix III along with numerous other files used to test the algorithms. A table comparing execution times of the major algorithms is given in Figure 5.2. Unless indicated otherwise, the times given are the number of CPU seconds required to find a single solution to the problem. The times given for BIGFIL, however, are CPU seconds required to find five solutions to each of the sixty-five problems. The FORTRAN algorithms were executed on the DEC-10 computer at the University of Tennessee.

The SCAN programs were all very slow. The last of these, SCAN6, was the fastest. As was described in the last chapter the solving routine for this program finds all possible subproblems involving a given frequency subset. This method has the advantage in that there is no backtracking necessary. All possible solutions to a problem can be found without ever

Problem 1

$\underline{n}_i$	$\underline{f}_i$
7	10
14	50
12	60
5	200
11	300

Problem 2

$\underline{n}_i$	$\underline{f}_i$
36	50
6	15
4	20
3	30

Problem 3

$\underline{n}_i$	$\underline{f}_i$
30	20
30	10
5	50
10	25
5	100

Problem 4

$\underline{n}_i$	$\underline{f}_i$
20	10000
15	5000
21	1000
25	500
100	25
500	10

Problem 5

$\underline{n}_i$	$\underline{f}_i$
51	500
15	250
9	200
25	25
149	50
813	20
483	10
19	1000

Problem 6

$\underline{n}_i$	$\underline{f}_i$
1	50
10	3
20	20
25	4
20	1
1	300

Problem 7

$\underline{n}_i$	$\underline{f}_i$
35	50
19	25
99	20
49	10

Problem 8

$\underline{n}_i$	$\underline{f}_i$
7	10000
15	5000
25	2500
50	1000
50	100
19	50
800	20
100	60
25	10
325	5

Problem 9

$\underline{n}_i$	$\underline{f}_i$
1	50
10	3
5	20
100	4
20	1
1	300

Problem 10

$\underline{n}_i$	$\underline{f}_i$
1	50
10	3
20	20
100	4

Figure 5.1. Files used in comparison.

File	SCAN6	ZIP2	ZIP3	ZIP4	ZIP7	ZAP	SIMP
Problem 1	0.64	0.15	0.16	0.15	0.18	0.17	>90.00 [*]
Problem 2	0.27	0.14	0.15	0.14	0.15	0.15	>75.00 [*]
Problem 3	13.89	0.13	0.13	0.13	0.14	0.14	>90.00 [*]
Problem 4	**	0.17	0.18	0.17	0.18	0.18	51.83
Problem 5	>8 min [*]	0.25	0.25	0.26	0.27	0.27	>90.00 [*]
Problem 6	**	47.22	25.51	11.89	11.97	1.34	>90.00 [*]
Problem 7	>60.00 [*]	>60.00 [*]	0.27	0.26	0.24	0.24	>90.00 [*]
Problem 8	**	0.27	0.29	0.22	0.25	0.25	>90.00 [*]
Problem 9	**			28.33	28.45	1.65	>2 min [*]
Problem 10	**				> 1 hr [*]	167.72	>5 min [*]
BIGFIL***		56.50	53.99	16.04	13.82	11.51	

^{*}These files did not produce a solution in the time shown.

^{**}On each of these runs with SCAN6 the problem would not solve because the stacks used to store subproblems would overflow.

^{***}This is a collection of 65 problems (given in Appendix III). Five solutions of each were produced in these runs.

Figure 5.2. Comparison of execution times on some of the files. Times are CPU seconds unless given otherwise.

having to throw away any of the results produced. It was not realized at the time this program was written however that the number of solutions generated could easily be very large. The routine worked fine as long as simple problems were tried. However, problems with many choices for possible subproblems and problems requiring a large number of subproblems took an excessive amount of CPU time to execute. They often overflowed the stacks used to store the subproblems. For these reasons another approach was desired.

The ZIP algorithm was developed next and showed a marked improvement over the SCAN programs. At the expense of adding backtracking to the algorithm, the SOLVE routine of the ZIP version needs only to find a single solution to each problem it is given. For most of the problems tried the improvement resulted in execution times on the order of a few tenths of a second of CPU time while the SCAN versions did not produce a solution for the same problem even after several minutes of CPU time.

Figure 5.2 includes a comparison of execution times between SCAN6 and ZIP2 on a few files. Note from the figure that half of the problems shown would not execute at all with SCAN6 because they overflowed the stacks used to store the subproblems. BIGFIL would not execute with any of the SCAN programs either.

Although ZIP2 produced solutions for most problems in a few tenths of a second, there were a few problems for which it still took a long time to execute. Problem 7 is one such problem which was studied. It appeared that if the SOLVE routine were given a problem which had solutions, then a solution would be found fairly quickly. However, if the problem had no solutions without increasing the system clock frequency (adding more dummy channels), then a great deal of time would be spent in the SOLVE routine

searching all the possibilities before returning with no solution and then adding the dummy channels. For example the system clock frequency initially calculated for Problem 7 was 4700 Hz. No solutions exist for this problem with that clock frequency, but the algorithm had to try all possibilities before discovering this and increasing the system clock frequency to 4800 Hz. For this reason the precheck was added which indicated for some problems that they had no solution. This lookahead was placed in the SOLVE routine so it would be applied to each subproblem as well as the actual problem. Therefore there was question as to whether more time would be spent in prechecking each subproblem than the time saved in not trying the problems with no solutions.

ZIP3 was identical to ZIP2 except ZIP3 included this precheck. Figure 5.2 shows the effect of this change in solving some typical problems. On most problems there was overall improvement in execution time. For the problems such as Problem 7 which failed the precheck on the main problem the improvement was excellent. For example Problem 7 was not solved with ZIP2 but was solved in a few tenths of a second with ZIP3.

The other two speed improvements mentioned in the last chapter were also added about this time. One improvement was the check during backtracking to see if a subproblem being removed contained a single channel. The other was the special section of the SOLVE routine to speed execution when the last subproblem of a set was being determined. These two speed improvements were added in a separate version, ZIP4, for comparison purposes. This version also contains the other speed improvement in ZIP3. Again Figure 5.2 shows the effect of these improvements on some typical problems. Note that ZIP4 found five solutions to BIGFIL in much less time than ZIP3 did.

The last version of the ZIP algorithm, ZIP7, was the only other one of major importance in this comparison. In addition to some changes in printout format, ZIP7 contains the added feature of reducing the dummy wheels. Because the method of reducing the dummy wheels sometimes involves adding an additional frequency group, it was necessary to determine the effect on execution time. Execution times using ZIP7 are also included in Figure 5.2. It is seen that for most of the problems reducing the dummy channels had no significant effect on execution times. It even speeded the execution of BIGFIL somewhat.

Even with the added speed improvements there were still a few problems that did not find a solution even after many minutes of CPU time. Problem 10 was one such problem. ZIP7 was allowed to run for one hour of CPU time on Problem 10 and still no solution was found. As in Problem 7 this problem has no solution with the system clock frequency initially calculated. Unlike Problem 7, however, the fact that it has no solution is not detected by the precheck, so all possible partitionings are tried anyway. Problems 6 and 9 are slight modifications of Problem 10 which greatly reduce the number of possible partitions. Both of these simpler problems would execute with the ZIP routines although they took many seconds of CPU time. In each of these problems it was noted that a lot of time was spent solving a subproblem when it could be determined beforehand that the rest of the problem could not be further partitioned. For this reason the ZAP approach was developed. ZAP first divides the entire problem into subproblems before spending any time on the subproblems. The new approach proved significantly faster on each of these problems which were not solved quickly with ZIP7. Figure 5.2 includes this comparison. An improvement in the execution of BIGFIL is also seen.

The final algorithm compared in Figure 5.2 is SIMP. This algorithm will solve problems with a small number of channels (listed in Appendix III) in a fairly short time. However problems with a large number of channels take very long to solve. In Figure 5.2 only Problem 4 was solved in less than a minute of CPU time. The fact that this problem was solved at all in this time reflects the extreme dependency SIMP has on the ordering of the input data. The reason the problem was solved in this amount of time was that the order in which the channels were input was very near the order in which they can first appear in the scan table. Changing the order of the channels can drastically effect execution time.

Each of the algorithms makes an exhaustive search for solutions. With the exception of the SCAN programs, the algorithms which partition the problem were much faster than those that did not. In addition, if a solution exists at all with the system clock frequency initially calculated, one is usually found quickly with these routines.

CHAPTER VI

CONCLUSIONS

Several algorithms for generating scan tables for a random access scanner have been described. Each uses exhaustive search in finding solutions. The SIMP approach is very straightforward and is easy to implement. However, for scan table problems with a large number of channels, it is very inefficient. The wheel based approaches provide a means for recursively dividing a problem into smaller subproblems. The most efficient of these is the ZAP approach. Through breadth-first search, ZAP partitions a problem into a set of subproblems, then solves each subproblem by recursively calling itself. A wheel diagram evolves from the execution of this algorithm. From the wheel diagram and a description of the initial problem, a scan table can be constructed.

In most cases a scan sequence may be stored in less memory if it is stored in wheel form rather than in scan table form (e.g. one-tenth as much in some typical problems). However, due to the number of levels encountered in generating a scan sequence from the wheel form, a much faster system is often needed in this approach. A faster memory, which is generally more expensive, may therefore be required. However, systems requiring a small memory and having processing time available between samples may benefit from this storage technique. So in implementing one approach or the other, the cost/speed tradeoffs must be considered.

There remain many related topics for research. In using the wheel form for storing and generating scan sequences, a way to minimize the number of wheels and consequently the number of memory locations is still needed. Further speed improvements could be made to the existing routines

to reduce execution time even further for some classes of problems. In addition SIMP could be combined with ZAP to provide solutions to some problems and subproblems which have solutions but cannot be partitioned. Another important topic for study is to derive a procedure for determining an optimal assignment of channels to scanners when more than one scanner is used.

It is possible this study may be applied to other areas besides scan sequence generation for data input to random access scanners. It may be viewed as a form of queuing theory. Additionally, it may find a use in time division multiplexing such as that used in satellite communications and other information systems.

LIST OF REFERENCES

LIST OF REFERENCES

1. N. H. McCoy, The Theory of Numbers, The Macmillan Company, New York, 1965.
2. USA Standard FORTRAN, ANSI X3.9-1966, American National Standards Institute, New York, 1966.
3. American National Standard Flowchart Symbols and Their Usage in Information Processing, ANSI X3.5-1970, American National Standards Institute, New York, 1966.
4. S. K. Morgan, How to Configure a Data Acquisition System, EMR Telemetry, Sarasota, Florida, 1977.

APPENDICES

APPENDIX I
PROGRAM LISTING

```

00100 C
00200 C
00300 C
00400 C
00500 C
00600 C
00700 C
00800 C
00900 C
01000 C
01100 C
01200 C
01300 C
01400 C
01500 C
01600 C
01700 C
01800 C
01900 C
02000 C
02100 C
02200 C
02300 C
02400 C
02500 C
02600 C
02700 C
02800 C
02900 C
03000 C
03100 C
03200 C
03300 C
03400 C
03500 C
03600 C
03700 C
03800 C
03900 C
04000 C
04100 C
04200 C
04300 C
04400 C
04500 C
04600 C
04700 C
04800 C
04900 C
05000 C
05100 C
05200 C
05300 C
05400 C
05500 C
05600 C
05700 C
05800 C
05900 C
06000 C

                                ZAP2.FOR

                                MAIN PROGRAM

THE MAIN PROGRAM IS USED TO INPUT THE DATA FROM A FILE
OR FROM THE KEYBOARD. DATA IS INPUT AS NUMBER OF
FREQUENCIES, FREQUENCY. THE DATA IS STORED IN TWO ARRAYS,
N(1,I) AND IFREQ(I). THESE TWO ARRAYS ALONG WITH THE NUMBER
OF FREQUENCIES, NF, ARE PASSED TO THE REST OF THE PROGRAM
THROUGH THE COMMON STATEMENT. FREQUENCIES ARE AUTOMATICALLY
REORDERED FROM HIGHEST TO LOWEST. THE MAIN PROGRAM
ALSO CONTROLS THE CALLING OF THE OTHER SUBROUTINES.

COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100
),
2 ICUR(18),NTIMS(100),MA(100)
COMMON/TIME/ MTOT,MPROB,MSOLN
COMMON/NEW/ NS1,NS2
COMMON/SCAN/LID(2000),LFRQ(2000),LDIV(2000),LBGN(2000)
DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),
1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
LOGICAL MASK,ISOL
DOUBLE PRECISION FILE,FILOUT

FROM HERE TO LINE 65 THE DATA IS INPUT FROM A FILE. IF NO
FILE IS INDICATED THEN DATA IS INPUT FROM THE KEYBOARD.
OUTPUT FILE IS ASKED FOR, AND AGAIN IF NO OUTPUT FILE IS
GIVEN THEN SOLUTIONS ARE OUTPUT TO THE TERMINAL. THE NUMBER
OF SOLUTIONS DESIRED IS ALSO ASKED FOR.
CALL JIFFY(MTOT,III)
IFLN=1
IODEV=5
TYPE 30
30 FORMAT (5X,'TYPE INPUT FILNAM.EXT> ',)$)
READ (5,40) FILE
40 FORMAT (A10)
TYPE 41
41 FORMAT (5X,'TYPE OUTPUT FILNAM.EXT> ',)$)
READ (5,40) FILOUT
IF (FILOUT.EQ.' ') GO TO 42
IODEV=1
OPEN (UNIT=11,DEVICE='DSK',ACCESS='SEQOUT',FILE=FILOUT)
42 CONTINUE
TYPE 45
45 FORMAT (' # SOLUTIONS DESIRED? > ',)$)
READ (5,60) NS1,NS2
IF (NS2.NE.0) GO TO 46
NS2=NS1
NS1=1
46 CONTINUE
IF (FILE.EQ.' ') GO TO 100
OPEN (UNIT=10,DEVICE='DSK',ACCESS='SEQIN',FILE=FILE)
K=10
C THIS LOOP (TO LINE 65) DOES THE ACTUAL DATA INPUT.
NF=1
NCH=1
N(1,1)=1
50 READ (K,60,END=65) LID(NCH),LFRQ(NCH)

```

```

06100      IFREQ(NF)=LFRQ(NCH)*2
06200      IF (LID(NCH).LE.0) GO TO 65
06300  53    NCH=NCH+1
06400      IF (NF.EQ.1) GO TO 62
06500      DO 55 I1=1,NF-1
06600      IF (IFREQ(NF).NE.IFREQ(I1)) GO TO 55
06700      N(1,I1)=N(1,I1)+1
06800      GO TO 50
06900  55    CONTINUE
07000  60    FORMAT (2I)
07100  62    NF=NF+1
07200      N(1,NF)=1
07300      IF (NF.LE.18) GO TO 50
07400  65    CALL JIFFY(MPROB,III)
07500      MSOLN=MPROB
07600  C    THIS LOOP TO LINE 68 WILL USE A BUBBLE SORT TO ORDER THE
07700  C    FREQUENCIES FROM HIGHEST TO LOWEST FOR FASTER EXECUTION.
07800      DO 68 I1=1,NF-2
07900      DO 68 I2=I1+1,NF-1
08000      IF (IFREQ(I1).GT.IFREQ(I2)) GO TO 68
08100      I3=IFREQ(I1)
08200      IFREQ(I1)=IFREQ(I2)
08300      IFREQ(I2)=I3
08400      I3=N(1,I1)
08500      N(1,I1)=N(1,I2)
08600      N(1,I2)=I3
08700  68    CONTINUE
08800  C    FILE NUMBER AND NUMBER OF FREQUENCIES INPUT ARE TYPED ON
08900  C    THE TTY AND THE PROGRAM JUMPS TO LINE 200 TO BEGIN SOLVING.
09000  70    NF=NF-1
09100      NCH=NCH-1
09200      NFORG=NF
09300      DO 72 I=1,NF
09400  72    IFOUT(I)=IFREQ(I)/2
09500      NSOL=0
09600      IF (IODEV.EQ.11) TYPE 78,IFLN
09700  78    FORMAT (2X,'FILE',I5)
09800      WRITE (IODEV,79) IFLN
09900  79    FORMAT (/, 'FILE',I5)
10000      WRITE (IODEV,80) NF,NCH
10100  80    FORMAT(/, ' INPUT: ',I6,' FREQUENCIES, ',I8,' CHANNELS')
10200      I2=0
10300  85    I1=I2+1
10400      I2=I2+7
10500      IF (I2.GT.NF) I2=NF
10600      WRITE (IODEV,90) (N(1,I),I=I1,I2)
10700  90    FORMAT (/, ' NUMBER:',3X,I8)
10800      WRITE (IODEV,95) (IFOUT(I),I=I1,I2)
10900  95    FORMAT (' FREQUENCY:',I8)
11000      IF (I2.LT.NF) GO TO 85
11100      GO TO 200
11200  C    NEXT 5 LINES SET UP PARAMETERS FOR KEYBOARD INPUT.
11300  100    K=5
11400      NF=1
11500      NCH=1
11600      N(1,1)=1
11700      TYPE 110
11800  110    FORMAT (5X,'TYPE DATA IN AS CHANNEL ID, FREQUENCY IN 2I
11900  C    GO TO 50
12000  C    SETUP IS CALLED TO INITIALIZE PARAMETERS AND ARRAYS.

```

```

12100 200      CALL SETUP
12200 C SOLVE IS CALLED TO OBTAIN AND PRINT OUT DESIRED SOLUTIONS.
12300      CALL SOLVE
12400 C IF NOT ENOUGH SOLUTIONS WERE FOUND THEN ADD FREQUENCIES
12500 C AT THE GCF AND LOOK FOR MORE SOLUTIONS.
12600      IF (NSOL.GE.NS2) GO TO 210
12700      IF (NF.EQ.NFORG) N(1,NF+1)=0
12800      NF=NFORG+1
12900      IFREQ(NF)=1
13000      N(1,NF)=N(1,NF)+ITOT(1)
13100      GO TO 200
13200 C STOP IF DATA READ FROM KEYBOARD.
13300 210      IF (K.EQ.5) STOP
13400 C IF DATA IS READ FROM A FILE THEN SEE IF THERE IS MORE
13500 C TO READ AND IF SO THEN BEGIN READING IT AND JUMP TO 53.
13600 C IF END OF FILE THEN CLOSE THE FILE AND STOP.
13700      NF=1
13800      NCH=1
13900      N(1,1)=1
14000 220      READ (K,60,END=300) LID(1),LFRQ(1)
14100      IFREQ(1)=LFRQ(1)*2
14200      IF (LID(1).LE.0) GO TO 220
14300      IFLN=IFLN+1
14400      GO TO 53
14500 300      CLOSE (UNIT=10,FILE=FILE)
14600      IF (IODEV.EQ.11) CLOSE (UNIT=11,FILE=FILOUT)
14700      STOP
14800      END
14900 C
15000 C
15100      SUBROUTINE SETUP
15200 C
15300 C THIS SUBROUTINE WILL INITIALIZE A FEW PROBLEM
15400 C PARAMETERS AND ARRAYS. THE GREATEST COMMON FACTOR, NGCF,
15500 C AND LOWEST COMMON MULTIPLE, LCM, OF THE FREQUENCIES
15600 C IS COMPUTED IN ADDITION TO ITOTL, THE SUM OF N(I)*IFREQ(I).
15700 C IT IS DETERMINED IF ADDITIONAL CHANNELS NEED TO BE ADDED
15800 C AND IF SO THEY ARE ADDED AT A FREQUENCY OF NGCF. CHANNELS
15900 C ARE ADDED TO BRING UP ITOTL TO A MULTIPLE OF LCM (AND AT
16000 C LEAST TWICE LCM). THE MAIN WHEEL SIZE (MAIN) IS COMPUTED
16100 C AS THIS RATIO OF ITOTL/LCM.
16200 C
16300 C OTHER PARAMETERS THAT ARE INITIALIZED ARE ON THE PROBLEM
16400 C STACK. THE PROBLEM STACK INVOLVES THE FOLLOWING. N(I,J) IS
16500 C THE NUMBER OF CHANNELS AT A FREQUENCY IFREQ(J) FOR PROBLEM I.
16600 C N(I,J) WAS SET IN THE MAIN PROGRAM. IM TELLS HOW MANY
16700 C FREQUENCIES ARE USED IN PROBLEM. IT IS SET TO NF FOR THE
16800 C INITIAL PROBLEM. IDIV(I,J) IS THE DIVIDE RATE FOR A
16900 C FREQUENCY IFREQ(J) USED IN PROBLEM I. IT IS SET FOR THE
17000 C INITIAL PROBLEM AS LCM/IFREQ(J). ITCT(I) IS A PARAMETER
17100 C WHICH INDICATES WHAT THE SUM OF N(I,J)*IFREQ(J) OVER J
17200 C SHOULD BE FOR A SPOKE OF A WHEEL IN PROBLEM I.
17300 C ITOT(1) IS SET TO LCM. THE NUMBER OF CHANNELS USED IN EACH
17400 C SPOKE OF A PROBLEM IS STORED IN THE ARRAY LSP. LSTR(I)
17500 C IS AN ARRAY OF POINTERS THAT TELLS WHICH ROW OF LSP BEGINS
17600 C THE SPOKES OF PROBLEM I. THUS FOR PROBLEM I, THE SPOKES IN
17700 C LSP(IP,J) RANGE FOR VALUES OF IP FROM LSTR(I) TO LSTR(I+1)-1,
17800 C WHERE LSP(IP,J) INDICATES THE NUMBER OF CHANNELS AT
17900 C FREQUENCY IFREQ(J) WHICH COME OFF THAT SPOKE. ISTR(IB,K) IS
18000 C A STRING FORM OF A SOLUTION TO EACH SPOKE. K HAS A VALUE

```

```

18100 C OF 1, 2, OR 3. ISTN(IB,1) IS THE NUMBER OF TIMES THAT SPOKE
18200 C IS USED. ISTN(IB,2) TELLS THE SIZE OF THE WHEEL LEADING
18300 C DIRECTLY FROM THAT SPOKE. ISTN(IB,3)=ISTN(IB,2) FOR TERMINAL
18400 C WHEELS, OTHERWISE IT'S ZERO. ANOTHER ARRAY, IST(IP) IS AN
18500 C ARRAY OF POINTERS WHICH TELLS WHERE THE STRING SOLUTION
18600 C STARTS IN ISTN FOR SPOKE IP. THUS THE STRING SOLUTION
18700 C OF SPOKE IP HAS IB VALUES RANGING FROM IST(IP) TO
18800 C IST(IP+1)-1 IN THE ARRAY ISTN(IB,K). THE POINTERS NPS,
18900 C LSTR(1), AND IST(1) ARE ALL INITIALIZED AT 1.
19000 C
19100 C ANOTHER ARRAY USED IS MASK(J), A LOGICAL ARRAY THAT IS NOT IN
19200 C STACKED FORM BUT RATHER RECOMPUTED FOR EACH PROBLEM. A
19300 C TRUE MASK(J) INDICATES THAT FREQUENCY IFREQ(J) IS USED IN
19400 C THE CURRENT PROBLEM. MASK(J) IS SET TO ALL TRUE INITIALLY.
19500 C ANOTHER LOGICAL VARIABLE USED IS ISOL. THIS VARIABLE
19600 C INDICATES UPON RETURNING FROM SOLVE IF THE SUBROUTINE FOUND
19700 C A SOLUTION TO THE GIVEN PROBLEM. THE NUMBER OF SOLUTIONS TO
19800 C THE MAIN PROBLEM IS CARRIED ALONG IN NSOL (INITIALLY SET AT
19900 C ZERO) AND THE NUMBER OF CHANNELS IS CARRIED IN NCH.
20000 C ONE LAST ARRAY COMPUTED HERE IS MIN(I,J). THIS
20100 C ARRAY IS DESCRIBED IN SUBROUTINE SOLVE.
20200 C
20300 C      COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
20400 C      1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NCUT(100),IFOUT(100
20500 C      ),
20600 C      2 ICUR(18),NTIMS(100),MA(100)
20700 C      COMMON/NEW2/ MIN(18,18)
20800 C      DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),
20900 C      1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
21000 C      LOGICAL MASK,ISOL
21100 C FIND GCF OF FREQUENCIES.
21200 C      NGCF=IFREQ(1)
21300 C      DO 10 I=2,NFORG
21400 C      NGCF=KGCF(NGCF,IFREQ(I))
21500 C FIND SUM OF (N(1,I)*IFREQ(I)) AND LCM OF FREQUENCIES.
21600 C      ITOTL=0
21700 C      LCM=NGCF
21800 C      DO 20 I=1,NF
21900 C      LCM=LCM*IFREQ(I)/KGCF(LCM,IFREQ(I))
22000 C      ITOTL=ITOTL+N(1,I)*IFREQ(I)
22100 C      ITOT(1)=LCM
22200 C DETERMINE IF CHANNELS MUST BE ADDED. IF NOT GO TO 40.
22300 C      MAIN=ITOTL/LCM
22400 C      K=LCM*MAIN
22500 C      IF (K.EQ.ITOTL.AND.K.NE.LCM) GO TO 40
22600 C FROM HERE TO 40 ADD CHANNELS. MAKE SURE MAIN WHEEL IS OF AT
22700 C LEAST SIZE 2. CHANNELS ARE ADDED AT FREQUENCY NGCF.
22800 C RECOMPUTE ITOTL, NCH, AND MAIN WHEEL SIZE.
22900 C      IF (ITOTL.LE.LCM) K=LCM
23000 C      NF=NF+1
23100 C      IFREQ(NF)=1
23200 C      N(1,NF)=K+LCM-ITOTL
23300 C      ITOTL=K+LCM
23400 C      MAIN=ITOTL/LCM
23500 C      CONTINUE
23600 C COMPUTE DIVIDE RATES AND SET MASK TO ALL TRUE.
23700 C      DO 50 I=1,NF
23800 C      IDIV(1,I)=LCM/IFREQ(I)
23900 C      MASK(I)=.TRUE.
24000 C INITIALIZE A FEW MORE VARIABLES.
24100 C      DO 60 I=1,NF

```

```

24100      DO 60 J=1,NF
24200 60    MIN(I,J)=IDIV(1,I)/KGCF(IDIV(1,J),IDIV(1,I))
24300      NPS=1
24400      IM=NF
24500      IST(1)=1
24600      LSTR(1)=1
24700      RETURN
24800      END
24900  C
25000  C
25100      FUNCTION KGCF(I,J)
25200  C
25300  C THIS FUNCTION ROUTINE WILL COMPUTE THE GREATEST COMMON FACTOR
25400  C OF TWO INTEGERS. I AND J ARE NOT CHANGED.
25500  C
25600      K=I
25700      KGCF=J
25800 10    CONTINUE
25900      IF (K.EQ.0) RETURN
26000      L=KGCF-(KGCF/K)*K
26100      KGCF=K
26200      K=L
26300      GO TO 10
26400      END
26500  C
26600  C
26700      SUBROUTINE SOLVE
26800  C
26900  C THIS SUBROUTINE WILL SOLVE THE PROBLEM ON TOP OF THE PROBLEM
27000  C STACK. ISOL IS RETURNED AS TRUE IF A SOLUTION WAS FOUND.
27100  C IF THE PROBLEM IS THE MAIN PROBLEM THEN THE DESIRED
27200  C SOLUTIONS ARE PRINTED AS THEY ARE FOUND.
27300  C
27400      COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
27500 1  IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100)
27600 2  ICUR(18),NTIMS(100),MA(100)
27700      COMMON/TIME/ MTOT,MPROB,MSOLN
27800      COMMON/NEW/ NS1,NS2
27900      COMMON/NEW2/ MIN(18,18)
28000      COMMON/SCAN/LID(2000),LFRQ(2000),LDIV(2000),LBGN(2000)
28100      DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),
28200 1  IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
28300      DIMENSION IN(18),IX(18),IA(18)
28400      DIMENSION IPNT(18),IPNTS(18),IPNTF(18)
28500      LOGICAL MASK,ISOL
28600      ISOL=.FALSE.
28700  C THE LOOP FROM HERE TO LINE 105 DOES A TEST ON THE INPUT
28800  C DATA. IT IS A PRECHECK WHICH IMMEDIATELY TELLS FOR SOME
28900  C PROBLEMS THAT THEY HAVE NO SOLUTION. MIN(I,J) IS AN ARRAY
29000  C WHERE MIN(I,J)=IDIV(1,I)/KGCF(IDIV(1,I),IDIV(1,J)). THIS
29100  C MEANS MIN(I,J) IS THE MINIMUM WHEEL SIZE AVAILABLE FOR
29200  C FREQUENCY IFREQ(I) WHEN USED ON A SPOKE WITH FREQUENCY
29300  C IFREQ(J). TAKING THE MINIMUM MIN(I,J) OVER ALL J FOR
29400  C FREQUENCIES USED IN A PROBLEM THEN GIVES THE MINIMUM
29500  C NUMBER OF CHANNELS AT FREQUENCY IFREQ(I) WHICH MUST BE
29600  C AVAILABLE IF ANY ARE USED AT ALL.
29700      DO 105 I=1,NF
29800      IF (.NOT.MASK(I)) GO TO 105
29900      K=ITOT(1)
30000      DO 100 J=1,NF

```

```

30100      IF (.NOT.MASK(J)) GO TO 100
30200      IF (I.EQ.J) GO TO 100
30300      IF (K.GT.MIN(I,J)) K=MIN(I,J)
30400 100    CONTINUE
30500      IF (K.LE.N(NPS,I)) GO TO 105
30600      NPS=NPS+1
30700      RETURN
30800 105    CONTINUE
30900  C THE LOOP FROM HERE TO LINE 120 TAKES ALL SINGLE FREQUENCY
31000  C SPOKES WHICH MAY BE USED IN THE PROBLEM AND PUTS THEIR
31100  C SOLUTION IN LSP AND ISTN. THE PROBLEM IS THEN UPDATED.
31200      LSTR(NPS+1)=LSTR(NPS)
31300      DO 120 I=1,NF
31400      IF (.NOT.MASK(I)) GO TO 120
31500      IF (N(NPS,I).LT.IDIV(NPS,I)) GO TO 120
31600      IP=LSTR(NPS+1)
31700      DO 110 J=1,NF
31800 110    LSP(IP,J)=0
31900      LSP(IP,I)=IDIV(NPS,I)
32000      IB=IST(IP)
32100      ISTN(IB,1)=N(NPS,I)/IDIV(NPS,I)
32200      NTIMS(IP)=ISTN(IB,1)
32300      N(NPS,I)=N(NPS,I)-ISTN(IB,1)*IDIV(NPS,I)
32400      ISTN(IB,2)=IDIV(NPS,I)
32500      ISTN(IB,3)=IDIV(NPS,I)
32600      MA(IP)=ISTN(IB,2)
32700      IST(IP+1)=IST(IP)+1
32800      LSTR(NPS+1)=LSTR(NPS+1)+1
32900 120    CONTINUE
33000      ICUR(NPS)=LSTR(NPS+1)
33100      NPS=NPS+1
33200      IM=1
33300  C A CHECK IS MADE TO SEE IF THE PROBLEM WAS SOLVED WITH SINGLE
33400  C FREQUENCY SPOKES. IF SO GO TO 600. IF NOT GO TO 300 TO
33500  C DEVELOPE THE FIRST MASK.
33600      DO 130 I=1,NF
33700      IF (N(NPS-1,I).NE.0) GO TO 300
33800 130    CONTINUE
33900      GO TO 600
34000  C FROM HERE TO LINE 310 IS A SECTION WHICH FINDS THE NEXT MASK.
34100 200    J=IM
34200      DO 210 I=1,NF
34300      IF (.NOT.MASK(I)) GO TO 210
34400      IF (N(NPS-1,I).EQ.0) GO TO 260
34500      J=J-1
34600 210    CONTINUE
34700      I=NF+1
34800      J=0
34900 220    I=I-1
35000      IF (N(NPS-1,I).EQ.0) GO TO 220
35100      IF (MASK(I)) GO TO 230
35200      J=-1
35300      GO TO 220
35400 230    J=J+1
35500      MASK(I)=.FALSE.
35600      IF (J.GE.IM) GO TO 300
35700      IF (J.GT.0) GO TO 220
35800      K=0
35900      DO 240 J=1,NF
36000 240    IF (MASK(J)) K=K+1

```

```

36100      K=IM-K
36200      DO 250 J=I+1,NF
36300      IF (N(NPS-1,J).EQ.O) GO TO 250
36400      MASK(J)=K.GT.O
36500      K=K-1
36600  250    CONTINUE
36700      GO TO 400
36800  260    MASK(I)=.FALSE.
36900  270    I=I+1
37000      IF (I.GT.NF) GO TO 280
37100      MASK(I)=N(NPS-1,I).NE.O.AND.J.GT.O
37200      IF (MASK(I)) J=J-1
37300      GO TO 270
37400  280    IF (J.LE.O) GO TO 400
37500      K=O
37600  290    I=I-1
37700      IF (I.EQ.O) GO TO 300
37800      IF (N(NPS-1,I).EQ.O) GO TO 290
37900      K=K+1
38000      IF (.NOT.MASK(I)) GO TO 290
38100      J=J+1
38200      IF (K.LE.J) GO TO 290
38300      MASK(I)=.FALSE.
38400      GO TO 270
38500  C FROM HERE TO LINE 310 A NEW MASK IS FOUND FOR TIMES WHEN THE
38600  C MASK SIZE MUST BE INCREASED BY ONE.
38700  300    IM=IM+1
38800      J=IM
38900      DO 310 I=1,NF
39000      MASK(I)=J.GT.O.AND.N(NPS-1,I).NE.O
39100      IF (MASK(I)) J=J-1
39200  310    CONTINUE
39300      IF (J.LE.O) GO TO 400
39400  C THIS SECTION WILL BACKTRACK BY STRIPPING THE LAST SPOKE
39500  C FROM THE WHEEL AND UPDATING THE PROBLEM. THE MASK IS SET
39600  C ACCORDING TO THIS STRIPPED SPOKE AND PARAMETERS ARE SET TO
39700  C SEARCH FOR A NEW COMBINATION OF NUMBERS USING THE SAME MASK.
39800  320    IF (LSTR(NPS).EQ.LSTR(NPS-1)) RETURN
39900      IP=LSTR(NPS)-1
40000      NTIMS(IP)=NTIMS(IP)-1
40100      IM=O
40200      DO 350 I=1,NF
40300      MASK(I)=LSP(IP,I).GT.O
40400      IF (MASK(I)) IM=IM+1
40500  350    N(NPS-1,I)=N(NPS-1,I)+LSP(IP,I)
40600      IF (IM.GT.1) GO TO 352
40700      DO 351 I=1,NF
40800  351    IF (MASK(I)) J=I
40900      IF (LSP(IP,J).NE.1) GO TO 352
41000      N(NPS-1,J)=N(NPS-1,J)+NTIMS(IP)
41100      RETURN
41200  352    CONTINUE
41300      IF (NTIMS(IP).EQ.O) LSTR(NPS)=LSTR(NPS)-1
41400      IF (ICUR(NPS-1).GT.LSTR(NPS)) ICUR(NPS-1)=LSTR(NPS)
41500  C FROM HERE TO THE LINE BEFORE LINE 400 THE INITIALIZATION
41600  C IS MADE TO FIND A NEW COMBINATION OF NUMBERS.
41700  355    J=O
41800      IF (IM.EQ.1) GO TO 200
41900      IB=LSTR(NPS)
42000      MA(IB)=MA(IP)

```

```

42100      ID=0
42200      DO 360 I=1,NF
42300      IF (.NOT.MASK(I)) GO TO 360
42400      IF (N(NPS-1,I).EQ.0) GO TO 200
42500      J=J+1
42600      IX(J)=IFREQ(I)
42700      IN(J)=N(NPS-1,I)
42800      IA(J)=LSP(IP,I)
42900      IF (IA(J).LE.IN(J)) GO TO 360
43000      ID=ID+(IA(J)-IN(J))*IX(J)
43100      IA(J)=IN(J)
43200 360    CONTINUE
43300      I=IM
43400      IF (ID.EQ.0) GO TO 430
43500      ID=ID+IA(I)*IX(I)
43600      GO TO 450
43700  C FROM HERE TO THE LINE BEFORE LINE 430 THE MAIN WHEEL IS
43800  C FOUND FOR THE NEW MASK. IF THE MAIN WHEEL IS ONE THEN THE
43900  C PROGRAM JUMPS TO 200 TO LOOK FOR ANOTHER MASK. NEW DIVIDE
44000  C RATES ARE FOUND AND THE SET UP IS MADE TO LOOK FOR A
44100  C COMBINATION OF NUMBERS USING THAT MASK.
44200 400    MN=ITOT(1)
44300      DO 410 I=1,NF
44400      IF (.NOT.MASK(I)) GO TO 410
44500      MN=KGCF(MN, IDIV(NPS-1,I))
44600 410    CONTINUE
44700      IF(MN.EQ.1) GO TO 200
44800      IP=LSTR(NPS)
44900      MA(IP)=MN
45000      J=1
45100      DO 420 I=1,NF
45200      IF (.NOT.MASK(I)) GO TO 420
45300      IX(J)=IFREQ(I)
45400      IN(J)=N(NPS-1,I)
45500      J=J+1
45600 420    CONTINUE
45700      ID=ITOT(NPS-1)
45800      I=1
45900      GO TO 450
46000  C THE SECTION FROM HERE TO LINE 500 WILL FIND THE NEXT
46100  C COMBINATION OF NUMBERS WHICH SATISFY THE EQUATION
46200  C  $SUM(N(I) * F(I)) = K$ . IA(I) IS AN ARRAY OF THE NUMBER OF
46300  C CHANNELS (IA IS NUMBERS TO BE DETERMINED). IX(I) IS AN ARRAY
46400  C OF THE CORRESPONDING FREQUENCIES. IN(I) IS AN ARRAY OF
46500  C THE NUMBER AVAILABLE AT EACH FREQUENCY (MAXIMUM ALLOWABLE
46600  C VALUES ON IA(I)). A JUMP TO FIND A NEW MASK IS MADE IF NO
46700  C NEW COMBINATION EXISTS.
46800 430    ID=ID+IA(I)*IX(I)
46900      I=I-1
47000      IF (I.EQ.0) GO TO 200
47100      IF (IA(I).EQ.1) GO TO 430
47200      IF (IN(I+1).GT.IA(I+1)) GO TO 435
47300      IK=I+1
47400 433    IK=IK+1
47500      IF (IK.GT.IM) GO TO 430
47600      IF (IN(IK).LE.IA(IK)) GO TO 433
47700 435    ID=ID+IX(I)
47800      IA(I)=IA(I)-1
47900 440    I=I+1
48000 450    IA(I)=ID/IX(I)

```

```

48100      IF (IA(I).GT.IN(I)) IA(I)=IN(I)
48200      IF (IA(I).EQ.O) GO TO 430
48300      ID=ID-IA(I)*IX(I)
48400      IF (I.NE.IM) GO TO 440
48500      IF (ID.NE.O) GO TO 430
48600      IP=LSTR(NPS)
48700      J=1
48800      C THIS LOOP COPIES THE NEW NUMBERS INTO A SPOKE OF LSP AND
48900      C SETS NTIMS(IP) EQUAL TO THE MOST NUMBER OF TIMES THE NEW
49000      C SPOKE IS USED.
49100      NTIMS(IP)=ITOT(NPS-1)
49200      DO 500 K=1,NF
49300      LSP(IP,K)=O
49400      IF (.NOT.MASK(K)) GO TO 500
49500      LSP(IP,K)=IA(J)
49600      IZ=N(NPS-1,K)/LSP(IP,K)
49700      IF (NTIMS(IP).GT.IZ) NTIMS(IP)=IZ
49800      J=J+1
49900      500 CONTINUE
50000      LSTR(NPS)=LSTR(NPS)+1
50100      IZ=O
50200      C THIS LOOP UPDATES THE PROBLEM AND SETS IZ TO TEST IF ONE
50300      C OR ZERO SPOKES ARE LEFT.
50400      DO 520 I=1,NF
50500      N(NPS-1,I)=N(NPS-1,I)-NTIMS(IP)*LSP(IP,I)
50600      520 IZ=IZ+N(NPS-1,I)*IFREQ(I)
50700      C IF MORE THAN ONE SPOKE LEFT THEN JUMP TO 355 TO FIND A NEW
50800      C COMBINATION USING THE SAME MASK.
50900      IF (IZ.GT.ITOT(NPS-1)) GO TO 355
51000      C IF NO SPOKES LEFT THEN DONE SO GO TO SOLVE SPOKES.
51100      IF (IZ.EQ.O) GO TO 545
51200      C THIS IS A SPECIAL SECTION TO SPEED UP EXECUTION WHEN JUST
51300      C ONE SPOKE IS LEFT.
51400      MN=ITOT(1)
51500      J=O
51600      K=O
51700      ID=O
51800      IK=O
51900      IZ=O
52000      C IN THIS LOOP THE MASK IS SET TO THE REMAINING FREQUENCIES,
52100      C THE MAIN WHEEL IS FOUND, AND TESTS ARE MADE TO DETERMINE IF
52200      C THE REMAINING NUMBERS MAKE UP A COMBINATION THAT HAS ALREADY
52300      C BEEN TRIED (BY COMPARING TO THE LAST COMBINATION).
52400      DO 530 I=1,NF
52500      MASK(I)=N(NPS-1,I).GT.O
52600      IF (LSP(IP,I).NE.O) K=K+1
52700      IF (.NOT.MASK(I)) GO TO 525
52800      J=J+1
52900      MN=KGCF(MN,IDIV(NPS-1,I))
53000      IF (IZ.NE.O) GO TO 525
53100      IF (LSP(IP,I).GT.N(NPS-1,I)) IZ=1
53200      IF (LSP(IP,I).LT.N(NPS-1,I)) IZ=-1
53300      525 IF (J.GT.K.AND.ID.EQ.O) IK=1
53400      530 IF (J.NE.K) ID=1
53500      C THESE SIX LINES MAKE THE TESTS AND JUMP TO BACKTRACK IF THE
53600      C MASK DIDN'T PASS.
53700      IF (MN.EQ.1) GO TO 320
53800      IF (J.LT.K) GO TO 320
53900      IF (J.GT.K) GO TO 535
54000      IF (IK.NE.O) GO TO 320

```

```

54100      IF (ID.NE.O) GO TO 535
54200      IF (IZ.LE.O) GO TO 320
54300      C THE MASK WAS OK SO SET LSP AND DIVIDE RATES.
54400      535 IP=IP+1
54500          MA(IP)=MN
54600          NTIMS(IP)=1
54700          DO 540 I=1,NF
54800          LSP(IP,I)=N(NPS-1,I)
54900          N(NPS-1,I)=O
55000      540 CONTINUE
55100          LSTR(NPS)=LSTR(NPS)+1
55200      C SOLVE THE SUBPROBLEMS.
55300      545 IF (ICUR(NPS-1).EQ.LSTR(NPS-1)) GO TO 550
55400          IP=ICUR(NPS-1)-1
55500          IB=IST(IP)
55600          ISTN(IB,1)=NTIMS(IP)
55700      550 IP=ICUR(NPS-1)
55800          IB=IST(IP)
55900          ISTN(IB,1)=NTIMS(IP)
56000          ISTN(IB,2)=MA(IP)
56100          ISTN(IB,3)=MA(IP)
56200          IM=O
56300          DO 555 J=1,NF
56400          MASK(J)=LSP(IP,J).GT.O
56500          IF (MASK(J)) IM=IM+1
56600      555 CONTINUE
56700          CALL SAME
56800          IF (ISOL) GO TO 580
56900          NPS=NPS-1
57000          I=LSTR(NPS)
57100      560 I=I-1
57200          DO 570 J=1,NF
57300      570 N(NPS-1,J)=N(NPS-1,J)+LSP(I,J)*NTIMS(I)
57400          IF (I.GT.ICUR(NPS-1)) GO TO 560
57500          LSTR(NPS)=ICUR(NPS-1)
57600          IP=LSTR(NPS)
57700          IM=O
57800          DO 575 I=1,NF
57900          MASK(I)=LSP(IP,I).GT.O
58000          IF (MASK(I)) IM=IM+1
58100      575 CONTINUE
58200          GO TO 355
58300      580 ICUR(NPS-1)=ICUR(NPS-1)+1
58400          IF (ICUR(NPS-1).LT.LSTR(NPS)) GO TO 550
58500      C PROGRAM JUMPS HERE WHEN A SOLUTION IS FOUND. GO TO 700
58600      C IF IT'S A SOLUTION TO A SUBPROBLEM. OTHERWISE PRINTOUT
58700      C THE SOLUTION AND STATISTICS. ALSO DUMP OUT A CONDENSED
58800      C SCAN TABLE TO FILE FOR22.DAT. A PROGRAM CALLED TABLE
58900      C CAN CREATE A COMPLETE SCAN TABLE FROM THIS FILE.
59000      600 IF (NPS.NE.2) GO TO 700
59100          IRS=MSOLN
59200          CALL JIFFY(MSOLN,III)
59300          NSOL=NSOL+1
59400          IF (NSOL.LT.NS1) GO TO 320
59500          WRITE (IODEV,620) NSOL
59600      620 FORMAT (//,3X,'SOLUTION NUMBER ',I5,/)
59700          IK=O
59800          IF (ISTN(1,2).EQ.1) IK=ISTN(1,1)
59900          IK1=IK
60000          IKK=O

```

```

60100      ID=MAIN
60200      625  IF (ID.EQ.0) GO TO 627
60300      IKK=IKK+1
60400      ID=ID-ISTN(IKK,1)
60500      IF (ISTN(IKK,2).NE.ISTN(IKK,3)) GO TO 627
60600      IF (ISTN(IKK,2).NE.ITOT(1)) GO TO 625
60700      IK=IK+ISTN(IKK,1)
60800      627  CONTINUE
60900      WRITE (IODEV,630) MAIN,IK
61000      630  FORMAT (8X,'1','I5','I5)
61100      NCHO=IK
61200      IZ=1
61300      IP=LSTR(2)
61400      J=1
61500      DO 632 I=1,NCH
61600      LBGN(I)=0
61700      632  LDIV(I)=0
61800      IF (IK.EQ.0) GO TO 638
61900      LB=0
62000      DO 636 I=1,IK
62100      634  LB=LB+1
62200      IF (LB.GT.NCH) GO TO 638
62300      IF (LFRQ(LB).NE.(ITOT(1)/2)) GO TO 634
62400      LBGN(LB)=I
62500      LDIV(LB)=MAIN
62600      636  CONTINUE
62700      638  CONTINUE
62800      IPNTP(1)=IK-1
62900      IN(1)=MAIN
63000      IX(1)=1
63100      IA(1)=1
63200      NFK=NFORG+1
63300      DO 640 I=1,NFORG
63400      IFOUT(I)=IFREQ(I)/2
63500      640  NOUT(I)=0
63600      IFOUT(NFK)=0
63700      NOUT(1)=IK1
63800      DO 670 I=1,IST(IP)-1
63900      IF (ISTN(I,2).EQ.1) GO TO 663
64000      IA(J+1)=IA(J)*ISTN(I,2)
64100      IF (IA(J+1).NE.ITOT(1)) GO TO 650
64200      IF (J.NE.1) GO TO 663
64300      ID=ISTN(I,1)
64400      IFR=ITOT(1)/2
64500      IKK=0
64600      GO TO 661
64700      650  WRITE (IODEV,652)
64800      652  FORMAT (' ','$)
64900      DO 654 K=1,J
65000      WRITE (IODEV,655)
65100      655  FORMAT ('+' ,'$)
65200      IK=ISTN(I,3)
65300      IF (ISTN(I+1,2).EQ.1.AND.I.NE.IST(IP)-1) IK=ISTN(I+1,1)
65400      IKK=I
65500      ID=ISTN(I,2)
65600      IF (ISTN(I,2).EQ.ISTN(I,3)) GO TO 657
65700      656  IF (ID.EQ.0) GO TO 657
65800      IKK=IKK+1
65900      ID=ID-ISTN(IKK,1)
66000      IF (ISTN(IKK,2).NE.ISTN(IKK,3)) GO TO 657

```

```

66100      IF (IA(J+1)*ISTN(IKK,2).NE.ITOT(1)) GO TO 656
66200      IK=IK+ISTN(IKK,1)
66300      657 CONTINUE
66400      WRITE (IODEV,660) (ISTN(I,K),K=1,2),IK
66500      660 FORMAT ('+',I5,',',I5,',',I5)
66600      IX(J+1)=IX(J)*ISTN(I,1)
66700      IZ=IZ+IX(J+1)
66800      IPNTS(J)=IPNTF(J)+1
66900      IPNTF(J)=IPNTF(J)+ISTN(I,1)
67000      IPNTS(J+1)=0
67100      IPNTF(J+1)=IK-1
67200      IF (IK.EQ.0) GO TO 663
67300      ID=IX(J+1)*IK
67400      NCHO=NCHO+ID
67500      IFR=ITOT(1)/IA(J+1)/2
67600      DO 6601 LA=1,J+1
67700      6601 IPNT(LA)=IPNTS(LA)
67800      LB1=0
67900      DO 6608 LA=1,ID
68000      LB=LB1
68100      6602 LB=LB+1
68200      IF (LB.GT.NCH) GO TO 6609
68300      IF (LFRQ(LB).NE.IFR.OR.LBGN(LB).NE.0) GO TO 6602
68400      LB1=LB
68500      LC=0
68600      DO 6604 LB=1,J
68700      6604 LC=LC+IPNT(LB+1)*IA(LB)
68800      LBGN(LB1)=LC*MAIN+IPNT(1)+1
68900      LDIV(LB1)=IA(J+1)*MAIN
69000      LB=J+1
69100      6606 IPNT(LB)=IPNT(LB)+1
69200      IF (IPNT(LB).LE.IPNTF(LB)) GO TO 6608
69300      IPNT(LB)=IPNTS(LB)
69400      LB=LB-1
69500      IF (LB.GT.0) GO TO 6606
69600      6608 CONTINUE
69700      6609 CONTINUE
69800      IKK=0
69900      661 IKK=IKK+1
70000      IF (IFR.LT.IFOUT(IKK)) GO TO 661
70100      IF (IFR.NE.IFOUT(IKK)) GO TO 662
70200      NOUT(IKK)=NOUT(IKK)+ID
70300      GO TO 663
70400      662 DO 6625 I2=IKK,NFK
70500      IK=IKK+NFK-I2
70600      IFOUT(IK+1)=IFOUT(IK)
70700      6625 NOUT(IK+1)=NOUT(IK)
70800      IFOUT(IKK)=IFR
70900      NOUT(IKK)=ID
71000      NFK=NFK+1
71100      IF (NFK.LE.99) GO TO 663
71200      NFK=NFK-1
71300      TYPE 6626
71400      6626 FORMAT (' ERROR -- OFLO ON # FREQUENCIES')
71500      663 IN(J)=IN(J)-ISTN(I,1)
71600      IF (ISTN(I,2).EQ.ISTN(I,3)) GO TO 665
71700      J=J+1
71800      IN(J)=ISTN(I,2)
71900      GO TO 670
72000      665 IF (IN(J).NE.0) GO TO 670

```

```

72100      J=J-1
72200      IF (J.NE.0) GO TO 665
72300 670    CONTINUE
72400      NFK=NFK-1
72500      ID=ITOT(1)*MAIN/2
72600      WRITE (IODEV,672) NFK, ID
72700 672    FORMAT (/, ' FREQUENCIES USED:', I6, 20X, 'CLOCK RATE --', I1
0)
72800      ID=0
72900 674    IK=ID+1
73000      ID=ID+7
73100      IF (ID.GT.NFK) ID=NFK
73200      WRITE (IODEV,676) (NOUT(I), I=IK, ID)
73300 676    FORMAT (/, ' NUMBER:', 3X, 7I8)
73400      WRITE (IODEV,678) (IFOUT(I), I=IK, ID)
73500 678    FORMAT ( ' FREQUENCY:', 7I8)
73600      IF (ID.LT.NFK) GO TO 674
73700      ID=NCHO+3*IZ-1
73800      WRITE (IODEV,680) NCHO, IZ, ID
73900 680    FORMAT (/, I8, ' CHANNELS', I8, ' WHEELS', I8,
74000 1 ' MEMORY WORDS')
74100      IZ=ITOT(1)*MAIN/NGCF+2
74200      R=FLOAT(ID)/FLOAT(IZ)*100.
74300      WRITE (IODEV,690) IZ, R
74400 690    FORMAT ( ' 1 WHEEL', I8, ' WORDS -- ', F5.1, '% (RATIO)')
74500      IZ=IZ-2
74600      WRITE (IODEV,6901) IZ
74700 6901    FORMAT (/, ' CONDENSED SCAN TABLE', 10X,
74800 1 ' SCAN TABLE LENGTH --', I12, /)
74900      WRITE (22, 1111) NCH, IZ
75000      WRITE (22, 1111) (LID(I), LBGN(I), LDIV(I), I=1, NCH)
75100 1111    FORMAT (3I10)
75200      ID=0
75300 6902    IK=ID+1
75400      ID=ID+10
75500      IF (ID.GT.NCH) ID=NCH
75600      WRITE (IODEV,6903) (LID(I), I=IK, ID)
75700 6903    FORMAT ( ' CHANNEL ID:', 10I6)
75800      WRITE (IODEV,6904) (LFRQ(I), I=IK, ID)
75900 6904    FORMAT ( ' FREQUENCY:', 10I6)
76000      WRITE (IODEV,6905) (LBGN(I), I=IK, ID)
76100 6905    FORMAT ( ' FIRST LOC:', 10I6)
76200      WRITE (IODEV,6906) (LDIV(I), I=IK, ID)
76300 6906    FORMAT ( ' REP RATE:', 10I6, /)
76400      IF (ID.LT.NCH) GO TO 6902
76500      CALL JIFFY(MSOLN, III)
76600      RS=FLOAT(MSOLN-IRS)/60.
76700      RP=FLOAT(MSOLN-MPROB)/60.
76800      RT=FLOAT(MSOLN-MTOT)/60.
76900      WRITE (IODEV,695) RT, RP, RS
77000 695    FORMAT (/, ' CPU TIMES (TOTAL, PROBLEM, SOLUTION) ', 3F10.2,
/)
77100      IF (NSOL.LT.NS2) GO TO 320
77200      ISOL=.TRUE.
77300      RETURN
77400  C SOLUTION TO SUBPROBLEM WAS FOUND SO ADJUST A FEW NUMBERS AND
77500  C RETURN.
77600 700    NPS=NPS-1
77700      IP=ICUR(NPS-1)
77800      IB=IST(IP)
77900      ISTN(IB, 3)=0
78000      J=LSTR(NPS+1)

```

```

78100      IST(IP+1)=IST(J)
78200      ISOL=.TRUE.
78300      RETURN
78400      END
78500  C
78600  C
78700      SUBROUTINE SAME
78800  C
78900  C THIS SUBROUTINE IS CALLED FROM SOLVE AND IS USED TO SETUP
79000  C A SUBPROBLEM AND CALL SOLVE BACK TO SOLVE THIS SUBPROBLEM.
79100  C
79200      COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
79300      1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100
79400      ),
79500      2 ICUR(18),NTIMS(100),MA(100)
79600      DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),
79700      1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
79800      LOGICAL MASK,ISOL
79900      IP=ICUR(NPS-1)
80000      IF (IM.EQ.1) GO TO 200
80100      IB=LSTR(NPS)
80200      IST(IB)=IST(IP)+1
80300      ITOT(NPS)=ITOT(NPS-1)/MA(IP)
80400      DO 100 I=1,NF
80500      IDIV(NPS,I)=IDIV(NPS-1,I)/MA(IP)
80600      100 N(NPS,I)=LSP(IP,I)
80700      CALL SOLVE
80800      RETURN
80900  C SINGLE FREQUENCY PROBLEMS ARE SOLVED HERE AND A RETURN
81000  C IS MADE.
81100      200 ISOL=.TRUE.
81200      IST(IP+1)=IST(IP)+1
81300      RETURN
81400      END

```

JIFFY.MAC

```
TITLE    JIFFY - RETURN CPU TIME IN JIFFYS
ENTRY JIFFY
JIFFY:  HRROI  1,4          ;TABLE 4 - MY JOB
        GETTAB 1,          ;ASK MONITOR
        SETZ   1,          ;IF IT FAILS RETURN 0
        MOVEM  1,@O(16)    ;STORE IT
        POPJ   17,         ;AND RETURN
        END
```

```

00100 C
00200 C
00300 C
00400 C
00500 C
00600 C THIS PROGRAM WILL PRODUCE A SCAN TABLE FROM A CONDENSED
00700 C VERSION. THE INPUT FILE MUST BEGIN WITH THE NUMBER OF
00800 C CHANNELS USED AND THE LENGTH OF THE SCAN TABLE. THESE
00900 C TWO VALUES ARE FOLLOWED BY A LIST OF CHANNEL ID, STARTING
01000 C LOCATION, AND DIVIDE RATE FOR EACH CHANNEL.
01100 C THE PROGRAM ZAP2 WILL AUTOMATICALLY PRODUCE A FILE FOR INPUT
01200 C TO THIS PROGRAM.
01300
01400
01500 DIMENSION LID(2000),LBGN(2000),LDIV(2000),LSC(2000)
01600 DOUBLE PRECISION FILEIN,FILOUT
01700 TYPE 10
01800 10 FORMAT (' TYPE INPUT FILNAM.EXT> ',S)
01900 READ (5,20) FILEIN
02000 20 FORMAT (A10)
02100 TYPE 30
02200 30 FORMAT (' TYPE OUTPUT FILNAM.EXT> ',S)
02300 READ (5,20) FILOUT
02400 OPEN (UNIT=10,DEVICE='DSK',ACCESS='SEQIN',FILE=FILEIN)
02500 OPEN (UNIT=11,DEVICE='DSK',ACCESS='SEQOUT',FILE=FILOUT)
02600 READ (10,*) NCH,LS
02700 READ (10,*) (LID(I),LBGN(I),LDIV(I),I=1,NCH)
02800 LEND=0
02900 100 LEND=LEND+2000
03000 IF (LEND.GT.LS) LEND=LS
03100 DO 110 I=1,2000
03200 110 LSC(I)=0
03300 DO 120 I=1,NCH
03400 IF (LBGN(I).GT.LEND) GO TO 120
03500 115 K=LBGN(I)
03600 IF (LSC(K).NE.0) GO TO 1000
03700 LSC(K)=LID(I)
03800 LBGN(I)=LBGN(I)+LDIV(I)
03900 IF (LBGN(I).LE.LEND) GO TO 115
04000 120 CONTINUE
04100 WRITE (11,130) (LSC(I),I=1,LS)
04200 130 FORMAT (10I7)
04300 IF (LEND.LT.LS) GO TO 100
04400 140 CONTINUE
04500 CLOSE (UNIT=10)
04600 CLOSE (UNIT=11)
04700 STOP
04800 1000 TYPE 1010
04900 1010 FORMAT (' ERROR')
05000 GO TO 140
05100 END

```

```

00100 C
00200 C
00300 C
00400 C
00500 C
00600 C
00700 C
00800 C
00900 C
01000 C
01100 C
01200 C
01300 C
01400 C
01500 C
01600 C
01700 C
01800 C
01900 C
02000 C
02100 C
02200 C
02300 C
02400 C
02500 C
02600 C
02700 C
02800 C
02900 C
03000 C
03100 C
03200 C
03300 C
03400 C
03500 C
03600 C
03700 C
03800 C
03900 C
04000 C
04100 C
04200 C
04300 C
04400 C
04500 C
04600 C
04700 C
04800 C
04900 C
05000 C
05100 C
05200 C
05300 C
05400 C
05500 C
05600 C
05700 C
05800 C
05900 C
06000 C

                                ZIP7.FOR

                                MAIN PROGRAM

THE MAIN PROGRAM IS USED TO INPUT THE DATA FROM A FILE
OR FROM THE KEYBOARD. DATA IS INPUT AS NUMBER OF
FREQUENCIES, FREQUENCY. THE DATA IS STORED IN TWO ARRAYS,
N(1,I) AND IFREQ(I). THESE TWO ARRAYS ALONG WITH THE NUMBER
OF FREQUENCIES, NF, ARE PASSED TO THE REST OF THE PROGRAM
THROUGH THE COMMON STATEMENT. FREQUENCIES ARE AUTOMATICALLY
REORDERED FROM HIGHEST TO LOWEST. THE MAIN PROGRAM
ALSO CONTROLS THE CALLING OF THE OTHER SUBROUTINES.

COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100)
)
COMMON/TIME/ MTOT,MPROB,MSOLN
COMMON/NEW/ NS1,NS2
DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),IM(18),
1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
LOGICAL MASK,ISOL
DOUBLE PRECISION FILE,FILOUT
FROM HERE TO LINE 65 THE DATA IS INPUT FROM A FILE. IF NO
FILE IS INDICATED THEN DATA IS INPUT FROM THE KEYBOARD.
OUTPUT FILE IS ASKED FOR, AND AGAIN IF NO OUTPUT FILE IS
GIVEN THEN SOLUTIONS ARE OUTPUT TO THE TERMINAL. THE NUMBER
OF SOLUTIONS DESIRED IS ALSO ASKED FOR.
CALL JIFFY(MTOT,III)
IFLN=1
NCH=0
IODEV=5
TYPE 30
30 FORMAT (5X,'TYPE INPUT FILNAM.EXT> ',S)
READ (5,40) FILE
40 FORMAT (A10)
TYPE 41
41 FORMAT (5X,'TYPE OUTPUT FILNAM.EXT> ',S)
READ (5,40) FILOUT
IF (FILOUT.EQ.' ') GO TO 42
IODEV=11
OPEN (UNIT=11,DEVICE='DSK',ACCESS='SEQOUT',FILE=FILOUT)
42 CONTINUE
TYPE 45
45 FORMAT (' # SOLUTIONS DESIRED? > ',S)
READ (5,60) NS1,NS2
IF (NS2.NE.0) GO TO 46
NS2=NS1
NS1=1
46 CONTINUE
IF (FILE.EQ.' ') GO TO 100
OPEN (UNIT=10,DEVICE='DSK',ACCESS='SEQIN',FILE=FILE)
K=10
THIS LOOP (TO LINE 65) DOES THE ACTUAL DATA INPUT.
NF=1
50 READ (K,60,END=65) N(1,NF),IFREQ(NF)
IFREQ(NF)=IFREQ(NF)*2
IF (N(1,NF).LE.0) GO TO 65
53 NCH=NCH+N(1,NF)

```

```

06100      IF (NF.EQ.1) GO TO 62
06200      DO 55 I1=1,NF-1
06300      IF (IFREQ(NF).NE.IFREQ(I1)) GO TO 55
06400      N(1,I1)=N(1,I1)+N(1,NF)
06500      GO TO 50
06600  55    CONTINUE
06700  60    FORMAT (2I)
06800  62    NF=NF+1
06900      IF (NF.LE.18) GO TO 50
07000  65    CALL JIFFY(MPROB,III)
07100      MSOLN=MPROB
07200  C    THIS LOOP TO LINE 68 WILL USE A BUBBLE SORT TO ORDER THE
07300  C    FREQUENCIES FROM HIGHEST TO LOWEST FOR FASTER EXECUTION.
07400      DO 68 I1=1,NF-2
07500      DO 68 I2=I1+1,NF-1
07600      IF (IFREQ(I1).GT.IFREQ(I2)) GO TO 68
07700      I3=IFREQ(I1)
07800      IFREQ(I1)=IFREQ(I2)
07900      IFREQ(I2)=I3
08000      I3=N(1,I1)
08100      N(1,I1)=N(1,I2)
08200      N(1,I2)=I3
08300  68    CONTINUE
08400  C    FILE NUMBER AND NUMBER OF FREQUENCIES INPUT ARE TYPED ON
08500  C    THE TTY AND THE PROGRAM JUMPS TO LINE 200 TO BEGIN SOLVING.
08600  70    NF=NF-1
08700      NFORG=NF
08800      DO 72 I=1,NF
08900  72    IFOUT(I)=IFREQ(I)/2
09000      NSOL=0
09100      IF (IODEV.EQ.11) TYPE 78,IFLN
09200  78    FORMAT (2X,'FILE',I5)
09300      WRITE (IODEV,79) IFLN
09400  79    FORMAT (/, ' FILE',I5)
09500      WRITE (IODEV,80) NF,NCH
09600  80    FORMAT(/, ' INPUT: ',I6, ' FREQUENCIES, ',I8, ' CHANNELS')
09700      I2=0
09800  85    I1=I2+1
09900      I2=I2+7
10000      IF (I2.GT.NF) I2=NF
10100      WRITE (IODEV,90) (N(1,I),I=I1,I2)
10200  90    FORMAT (/, ' NUMBER:',3X,I8)
10300      WRITE (IODEV,95) (IFOUT(I),I=I1,I2)
10400  95    FORMAT ( ' FREQUENCY:',I8)
10500      IF (I2.LT.NF) GO TO 85
10600      GO TO 200
10700  C    NEXT 5 LINES SET UP PARAMETERS FOR KEYBOARD INPUT.
10800  100    K=5
10900      NF=1
11000      TYPE 110
11100  110    FORMAT (5X,'TYPE DATA IN AS NUMBER FREQUENCY IN 2I FORMA
11200      GO TO 50
11300  C    SETUP IS CALLED TO INITIALIZE PARAMETERS AND ARRAYS.
11400  200    CALL SETUP
11500  C    SOLVE IS CALLED TO OBTAIN AND PRINT OUT DESIRED SOLUTIONS.
11600      CALL SOLVE
11700  C    IF NOT ENOUGH SOLUTIONS WERE FOUND THEN ADD FREQUENCIES
11800  C    AT THE GCF AND LOOK FOR MORE SOLUTIONS.
11900      IF (ISOL) GO TO 210
12000      IF (NF.EQ.NFORG) N(1,NF+1)=0

```

```

12100      NF=NFORG+1
12200      IFREQ(NF)=1
12300      N(1,NF)=N(1,NF)+ITOT(1)
12400      GO TO 200
12500  C   STOP IF DATA READ FROM KEYBOARD.
12600  210   IF (K.EQ.5) STOP
12700  C   IF DATA IS READ FROM A FILE THEN SEE IF THERE IS MORE
12800  C   TO READ AND IF SO THEN BEGIN READING IT AND JUMP TO 53.
12900  C   IF END OF FILE THEN CLOSE THE FILE AND STOP.
13000      NF=1
13100      NCH=0
13200  220   READ (K,60,END=300) N(1,NF),IFREQ(NF)
13300      IFREQ(NF)=IFREQ(NF)*2
13400      IF (N(1,NF).LE.0) GO TO 220
13500      IFLN=IFLN+1
13600      GO TO 53
13700  300   CLOSE (UNIT=10,FILE=FILE)
13800      IF (IODEV.EQ.11) CLOSE (UNIT=11,FILE=FILOUT)
13900      STOP
14000      END
14100  C
14200  C
14300      SUBROUTINE SETUP
14400  C
14500  C   THIS SUBROUTINE WILL INITIALIZE A FEW PROBLEM
14600  C   PARAMETERS AND ARRAYS.  THE GREATEST COMMON FACTOR, NGCF,
14700  C   AND LOWEST COMMON MULTIPLE, LCM, OF THE FREQUENCIES
14800  C   IS COMPUTED IN ADDITION TO ITOTL, THE SUM OF N(I)*IFREQ(I).
14900  C   IT IS DETERMINED IF ADDITIONAL CHANNELS NEED TO BE ADDED
15000  C   AND IF SO THEY ARE ADDED AT A FREQUENCY OF NGCF.  CHANNELS
15100  C   ARE ADDED TO BRING UP ITOTL TO A MULTIPLE OF LCM (AND AT
15200  C   LEAST TWICE LCM).  THE MAIN WHEEL SIZE (MAIN) IS COMPUTED
15300  C   AS THIS RATIO OF ITOTL/LCM.
15400  C
15500  C   OTHER PARAMETERS THAT ARE INITIALIZED ARE ON THE PROBLEM
15600  C   STACK.  THE PROBLEM STACK INVOLVES THE FOLLOWING.  N(I,J) IS
15700  C   THE NUMBER OF CHANNELS AT A FREQUENCY IFREQ(J) FOR PROBLEM I.
15800  C   N(I,J) WAS SET IN THE MAIN PROGRAM.  IM(I) TELLS HOW MANY
15900  C   FREQUENCIES ARE USED IN PROBLEM I.  IT IS SET TO NF FOR THE
16000  C   INITIAL PROBLEM.  IDIV(I,J) IS THE DIVIDE RATE FOR A
16100  C   FREQUENCY IFREQ(J) USED IN PROBLEM I.  IT IS SET FOR THE
16200  C   INITIAL PROBLEM AS LCM/IFREQ(J).  ITOT(I) IS A PARAMETER
16300  C   WHICH INDICATES WHAT THE SUM OF N(I,J)*IFREQ(J) OVER J
16400  C   SHOULD BE FOR A SPOKE OF A WHEEL IN PROBLEM I.
16500  C   ITOT(1) IS SET TO LCM.  THE NUMBER OF CHANNELS USED IN EACH
16600  C   SPOKE OF A PROBLEM IS STORED IN THE ARRAY LSP.  LSTR(I)
16700  C   IS AN ARRAY OF POINTERS THAT TELLS WHICH ROW OF LSP BEGINS
16800  C   THE SPOKES OF PROBLEM I.  THUS FOR PROBLEM I, THE SPOKES IN
16900  C   LSP(IP,J) RANGE FOR VALUES OF IP FROM LSTR(I) TO LSTR(I+1)-1,
17000  C   WHERE LSP(IP,J) INDICATES THE NUMBER OF CHANNELS AT
17100  C   FREQUENCY IFREQ(J) WHICH COME OFF THAT SPOKE.  ISTN(IB,K) IS
17200  C   A STRING FORM OF A SOLUTION TO EACH SPOKE.  K HAS A VALUE
17300  C   OF 1, 2, OR 3.  ISTN(IB,1) IS THE NUMBER OF TIMES THAT SPOKE
17400  C   IS USED.  ISTN(IB,2) TELLS THE SIZE OF THE WHEEL LEADING
17500  C   DIRECTLY FROM THAT SPOKE.  ISTN(IB,3)=ISTN(IB,2) FOR TERMINAL
17600  C   WHEELS, OTHERWISE IT'S ZERO.  ANOTHER ARRAY, IST(IP) IS AN
17700  C   ARRAY OF POINTERS WHICH TELLS WHERE THE STRING SOLUTION
17800  C   STARTS IN ISTN FOR SPOKE IP.  THUS THE STRING SOLUTION
17900  C   OF SPOKE IP HAS IB VALUES RANGING FROM IST(IP) TO
18000  C   IST(IP+1)-1 IN THE ARRAY ISTN(IB,K).  THE POINTERS NPS,

```

```

18100 C LSTR(1), AND IST(1) ARE ALL INITIALIZED AT 1.
18200 C
18300 C ANOTHER ARRAY USED IS MASK(J), A LOGICAL ARRAY THAT IS NOT IN
18400 C STACKED FORM BUT RATHER RECOMPUTED FOR EACH PROBLEM. A
18500 C TRUE MASK(J) INDICATES THAT FREQUENCY IFREQ(J) IS USED IN
18600 C THE CURRENT PROBLEM. MASK(J) IS SET TO ALL TRUE INITIALLY.
18700 C ANOTHER LOGICAL VARIABLE USED IS ISOL. THIS VARIABLE
18800 C INDICATES UPON RETURNING FROM SOLVE IF THE SUBROUTINE FOUND
18900 C A SOLUTION TO THE GIVEN PROBLEM. THE NUMBER OF SOLUTIONS TO
19000 C THE MAIN PROBLEM IS CARRIED ALONG IN NSOL (INITIALLY SET AT
19100 C ZERO) AND THE NUMBER OF CHANNELS IS CARRIED IN NCH.
19200 C ONE LAST ARRAY COMPUTED HERE IS MIN(I,J). THIS
19300 C ARRAY IS DESCRIBED IN SUBROUTINE SOLVE.
19400 C
19500 C      COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
19600 C      1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100
19700 C      )
19800 C      COMMON/NEW2/ MIN(18,18)
19900 C      DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),IM(18),
20000 C      1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
20100 C      LOGICAL MASK,ISOL
20200 C      FIND GCF OF FREQUENCIES.
20300 C      NGCF=IFREQ(1)
20400 C      DO 10 I=2,NFORG
20500 C      NGCF=KGCF(NGCF,IFREQ(I))
20600 C      FIND SUM OF (N(1,I)*IFREQ(I)) AND LCM OF FREQUENCIES.
20700 C      ITOTL=0
20800 C      LCM=NGCF
20900 C      DO 20 I=1,NF
21000 C      LCM=LCM*IFREQ(I)/KGCF(LCM,IFREQ(I))
21100 C      ITOTL=ITOTL+N(1,I)*IFREQ(I)
21200 C      ITOT(1)=LCM
21300 C      DETERMINE IF CHANNELS MUST BE ADDED. IF NOT GO TO 40.
21400 C      MAIN=ITOTL/LCM
21500 C      K=LCM*MAIN
21600 C      IF (K.EQ.ITOTL.AND.K.NE.LCM) GO TO 40
21700 C      FROM HERE TO 40 ADD CHANNELS. MAKE SURE MAIN WHEEL IS OF AT
21800 C      LEAST SIZE 2. CHANNELS ARE ADDED AT FREQUENCY NGCF.
21900 C      RECOMPUTE ITOTL, NCH, AND MAIN WHEEL SIZE.
22000 C      IF (ITOTL.LE.LCM) K=LCM
22100 C      NF=NF+1
22200 C      IFREQ(NF)=1
22300 C      N(1,NF)=K+LCM-ITOTL
22400 C      ITOTL=K+LCM
22500 C      MAIN=ITOTL/LCM
22600 C      40 CONTINUE
22700 C      COMPUTE DIVIDE RATES AND SET MASK TO ALL TRUE.
22800 C      DO 50 I=1,NF
22900 C      IDIV(1,I)=LCM/IFREQ(I)
23000 C      MASK(I)=.TRUE.
23100 C      INITIALIZE A FEW MORE VARIABLES.
23200 C      DO 60 I=1,NF
23300 C      DO 60 J=1,NF
23400 C      MIN(I,J)=IDIV(1,I)/KGCF(IDIV(1,J),IDIV(1,I))
23500 C      NPS=1
23600 C      IM(1)=NF
23700 C      IST(1)=1
23800 C      LSTR(1)=1
23900 C      RETURN
24000 C      END

```

```

24100 C
24200 FUNCTION KGCF(I,J)
24300 C
24400 C THIS FUNCTION ROUTINE WILL COMPUTE THE GREATEST COMMON FACTOR
24500 C OF TWO INTEGERS. I AND J ARE NOT CHANGED.
24600 C
24700 K=I
24800 KGCF=J
24900 10 CONTINUE
25000 IF (K.EQ.O) RETURN
25100 L=KGCF-(KGCF/K)*K
25200 KGCF=K
25300 K=L
25400 GO TO 10
25500 END
25600 C
25700 C
25800 SUBROUTINE SOLVE
25900 C
26000 C THIS SUBROUTINE WILL SOLVE THE PROBLEM ON TOP OF THE PROBLEM
26100 C STACK. ISOL IS RETURNED AS TRUE IF A SOLUTION WAS FOUND.
26200 C IF THE PROBLEM IS THE MAIN PROBLEM THEN THE DESIRED
26300 C SOLUTIONS ARE PRINTED AS THEY ARE FOUND.
26400 C
26500 COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
26600 1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100)
26700 COMMON/TIME/ MTOT,MPROB,MSOLN
26800 COMMON/NEW/ NS1,NS2
26900 COMMON/NEW2/ MIN(18,18)
27000 DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),IM(18),
27100 1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
27200 DIMENSION IN(18),IX(18),IA(18)
27300 LOGICAL MASK,ISOL
27400 ISOL=.FALSE.
27500 C THE LOOP FROM HERE TO LINE 105 DOES A TEST ON THE INPUT
27600 C DATA. IT IS A PRECHECK WHICH IMMEDIATELY TELLS FOR SOME
27700 C PROBLEMS THAT THEY HAVE NO SOLUTION. MIN(I,J) IS AN ARRAY
27800 C WHERE MIN(I,J)=IDIV(1,I)/KGCF(IDIV(1,I),IDIV(1,J)). THIS
27900 C MEANS MIN(I,J) IS THE MINIMUM WHEEL SIZE AVAILABLE FOR
28000 C FREQUENCY IFREQ(I) WHEN USED ON A SPOKE WITH FREQUENCY
28100 C IFREQ(J). TAKING THE MINIMUM MIN(I,J) OVER ALL J FOR
28200 C FREQUENCIES USED IN A PROBLEM THEN GIVES THE MINIMUM
28300 C NUMBER OF CHANNELS AT FREQUENCY IFREQ(I) WHICH MUST BE
28400 C AVAILABLE IF ANY ARE USED AT ALL.
28500 DO 105 I=1,NF
28600 IF (.NOT.MASK(I)) GO TO 105
28700 K=ITOT(1)
28800 DO 100 J=1,NF
28900 IF (.NOT.MASK(J)) GO TO 100
29000 IF (I.EQ.J) GO TO 100
29100 IF (K.GT.MIN(I,J)) K=MIN(I,J)
29200 100 CONTINUE
29300 IF (K.LE.N(NPS,I)) GO TO 105
29400 NPS=NPS+1
29500 RETURN
29600 105 CONTINUE
29700 C THE LOOP FROM HERE TO LINE 120 TAKES ALL SINGLE FREQUENCY
29800 C SPOKES WHICH MAY BE USED IN THE PROBLEM AND PUTS THEIR
29900 C SOLUTION IN LSP AND ISTN. THE PROBLEM IS THEN UPDATED.
30000 LSTR(NPS+1)=LSTR(NPS)

```

```

30100      DO 120 I=1,NF
30200      IF (.NOT.MASK(I)) GO TO 120
30300      IF (N(NPS,I).LT.IDIV(NPS,I)) GO TO 120
30400      IP=LSTR(NPS+1)
30500      DO 110 J=1,NF
30600 110    LSP(IP,J)=0
30700      LSP(IP,I)=IDIV(NPS,I)
30800      IB=IST(IP)
30900      ISTN(IB,1)=N(NPS,I)/IDIV(NPS,I)
31000      N(NPS,I)=N(NPS,I)-ISTN(IB,1)*IDIV(NPS,I)
31100      ISTN(IB,2)=IDIV(NPS,I)
31200      ISTN(IB,3)=IDIV(NPS,I)
31300      IST(IP+1)=IST(IP)+1
31400      LSTR(NPS+1)=LSTR(NPS+1)+1
31500 120    CONTINUE
31600      NPS=NPS+1
31700      IM(NPS)=1
31800      C A CHECK IS MADE TO SEE IF THE PROBLEM WAS SOLVED WITH SINGLE
31900      C FREQUENCY SPOKES. IF SO GO TO 600. IF NOT GO TO 300 TO
32000      C DEVELOPE THE FIRST MASK.
32100      DO 130 I=1,NF
32200      IF (N(NPS-1,I).NE.0) GO TO 300
32300 130    CONTINUE
32400      GO TO 600
32500      C FROM HERE TO LINE 310 IS A SECTION WHICH FINDS THE NEXT MASK.
32600 200    J=IM(NPS)
32700      DO 210 I=1,NF
32800      IF (.NOT.MASK(I)) GO TO 210
32900      IF (N(NPS-1,I).EQ.0) GO TO 260
33000      J=J-1
33100 210    CONTINUE
33200      I=NF+1
33300      J=0
33400 220    I=I-1
33500      IF (N(NPS-1,I).EQ.0) GO TO 220
33600      IF (MASK(I)) GO TO 230
33700      J=-1
33800      GO TO 220
33900 230    J=J+1
34000      MASK(I)=.FALSE.
34100      IF (J.GE.IM(NPS)) GO TO 300
34200      IF (J.GT.0) GO TO 220
34300      K=0
34400      DO 240 J=1,NF
34500 240    IF (MASK(J)) K=K+1
34600      K=IM(NPS)-K
34700      DO 250 J=I+1,NF
34800      IF (N(NPS-1,J).EQ.0) GO TO 250
34900      MASK(J)=K.GT.0
35000      K=K-1
35100 250    CONTINUE
35200      GO TO 400
35300 260    MASK(I)=.FALSE.
35400 270    I=I+1
35500      IF (I.GT.NF) GO TO 280
35600      MASK(I)=N(NPS-1,I).NE.0.AND.J.GT.0
35700      IF (MASK(I)) J=J-1
35800      GO TO 270
35900 280    IF (J.LE.0) GO TO 400
36000      K=0

```

```

36100      290      I=I-1
36200              IF (I.EQ.0) GO TO 300
36300              IF (N(NPS-1,I).EQ.0) GO TO 290
36400              K=K+1
36500              IF (.NOT.MASK(I)) GO TO 290
36600              J=J+1
36700              IF (K.LE.J) GO TO 290
36800              MASK(I)=.FALSE.
36900              GO TO 270
37000      C FROM HERE TO LINE 310 A NEW MASK IS FOUND FOR TIMES WHEN THE
37100      C MASK SIZE MUST BE INCREASED BY ONE.
37200      300      IM(NPS)=IM(NPS)+1
37300              J=IM(NPS)
37400              DO 310 I=1,NF
37500              MASK(I)=J.GT.0.AND.N(NPS-1,I).NE.0
37600              IF (MASK(I)) J=J-1
37700      310      CONTINUE
37800              IF (J.LE.0) GO TO 400
37900      C THIS SECTION WILL BACKTRACK BY STRIPPING THE LAST SPOKE
38000      C FROM THE WHEEL AND UPDATING THE PROBLEM. THE MASK IS SET
38100      C ACCORDING TO THIS STRIPPED SPOKE AND PARAMETERS ARE SET TO
38200      C SEARCH FOR A NEW COMBINATION OF NUMBERS USING THE SAME MASK.
38300      320      IF (LSTR(NPS).EQ.LSTR(NPS-1)) RETURN
38400              IP=LSTR(NPS)-1
38500              IB=IST(IP)
38600              ISTN(IB,1)=ISTN(IB,1)-1
38700              K=ISTN(IB,2)
38800              IM(NPS)=0
38900              DO 350 I=1,NF
39000              MASK(I)=LSP(IP,I).GT.0
39100              IF (MASK(I)) IM(NPS)=IM(NPS)+1
39200      350      N(NPS-1,I)=N(NPS-1,I)+LSP(IP,I)
39300              IF (IM(NPS).GT.1) GO TO 352
39400              DO 351 I=1,NF
39500      351      IF (MASK(I)) J=I
39600              IF (LSP(IP,J).NE.1) GO TO 352
39700              N(NPS-1,J)=N(NPS-1,J)+ISTN(IB,1)
39800              RETURN
39900      352      CONTINUE
40000              IF (ISTN(IB,1).EQ.0) LSTR(NPS)=LSTR(NPS)-1
40100              GO TO 355
40200      353      LSTR(NPS)=LSTR(NPS)-1
40300              DO 354 I=1,NF
40400      354      MASK(I)=LSP(IP,I).GT.0
40500      C FROM HERE TO THE LINE BEFORE LINE 400 THE INITIALIZATION
40600      C IS MADE TO FIND A NEW COMBINATION OF NUMBERS.
40700      355      J=0
40800              IF (IM(NPS).EQ.1) GO TO 200
40900              IB=LSTR(NPS)
41000              IB=IST(IB)
41100              ISTN(IB,2)=K
41200              ID=0
41300              DO 360 I=1,NF
41400              IF (.NOT.MASK(I)) GO TO 360
41500              IF (N(NPS-1,I).EQ.0) GO TO 200
41600              J=J+1
41700              IX(J)=IFREQ(I)
41800              IN(J)=N(NPS-1,I)
41900              IA(J)=LSP(IP,I)
42000              IDIV(NPS,I)=IDIV(NPS-1,I)/K

```

```

42100      IF (IA(J).LE.IN(J)) GO TO 360
42200      ID=ID+(IA(J)-IN(J))*IX(J)
42300      IA(J)=IN(J)
42400 360    CONTINUE
42500      I=IM(NPS)
42600      IF (ID.EQ.0) GO TO 430
42700      ID=ID+IA(I)*IX(I)
42800      GO TO 450
42900  C FROM HERE TO THE LINE BEFORE LINE 430 THE MAIN WHEEL IS
43000  C FOUND FOR THE NEW MASK. IF THE MAIN WHEEL IS ONE THEN THE
43100  C PROGRAM JUMPS TO 200 TO LOOK FOR ANOTHER MASK. NEW DIVIDE
43200  C RATES ARE FOUND AND THE SET UP IS MADE TO LOOK FOR A
43300  C COMBINATION OF NUMBERS USING THAT MASK.
43400 400    MN=ITOT(1)
43500      DO 410 I=1,NF
43600      IF (.NOT.MASK(I)) GO TO 410
43700      MN=KGCF(MN,IDIV(NPS-1,I))
43800 410    CONTINUE
43900      IF(MN.EQ.1) GO TO 200
44000      IP=LSTR(NPS)
44100      IB=IST(IP)
44200      ISTN(IB,2)=MN
44300      J=1
44400      DO 420 I=1,NF
44500      IF (.NOT.MASK(I)) GO TO 420
44600      IDIV(NPS,I)=IDIV(NPS-1,I)/MN
44700      IX(J)=IFREQ(I)
44800      IN(J)=N(NPS-1,I)
44900      J=J+1
45000 420    CONTINUE
45100      ID=ITOT(NPS-1)
45200      I=1
45300      GO TO 450
45400  C THE SECTION FROM HERE TO LINE 500 WILL FIND THE NEXT
45500  C COMBINATION OF NUMBERS WHICH SATISFY THE EQUATION
45600  C  $\sum (N(I) * F(I)) = K$ . IA(I) IS AN ARRAY OF THE NUMBER OF
45700  C CHANNELS (IA IS NUMBERS TO BE DETERMINED). IX(I) IS AN ARRAY
45800  C OF THE CORRESPONDING FREQUENCIES. IN(I) IS AN ARRAY OF
45900  C THE NUMBER AVAILABLE AT EACH FREQUENCY (MAXIMUM ALLOWABLE
46000  C VALUES ON IA(I)). A JUMP TO FIND A NEW MASK IS MADE IF NO
46100  C NEW COMBINATION EXISTS.
46200 430    ID=ID+IA(I)*IX(I)
46300      I=I-1
46400      IF (I.EQ.0) GO TO 200
46500      IF (IA(I).EQ.1) GO TO 430
46600      IF (IN(I+1).GT.IA(I+1)) GO TO 435
46700      IK=I+1
46800 433    IK=IK+1
46900      IF (IK.GT.IM(NPS)) GO TO 430
47000      IF (IN(IK).LE.IA(IK)) GO TO 433
47100 435    ID=ID+IX(I)
47200      IA(I)=IA(I)-1
47300 440    I=I+1
47400 450    IA(I)=ID/IX(I)
47500      IF (IA(I).GT.IN(I)) IA(I)=IN(I)
47600      IF (IA(I).EQ.0) GO TO 430
47700      ID=ID-IA(I)*IX(I)
47800      IF (I.NE.IM(NPS)) GO TO 440
47900      IF (ID.NE.0) GO TO 430
48000      IP=LSTR(NPS)

```

```

48100      J=1
48200      C THIS LOOP COPIES THE NEW NUMBERS INTO A SPOKE OF LSP.
48300      DO 500 K=1,NF
48400      LSP(IP,K)=0
48500      IF (.NOT.MASK(K)) GO TO 500
48600      LSP(IP,K)=IA(J)
48700      J=J+1
48800      500 CONTINUE
48900      LSTR(NPS)=LSTR(NPS)+1
49000      C THIS SPOKE IS SOLVED BY CALLING SAME.
49100      CALL SAME
49200      IF (.NOT.ISOL) NPS=NPS-1
49300      505 IP=LSTR(NPS)-1
49400      IB=IST(IP)
49500      K=ISTN(IB,2)
49600      C IF NO SOLUTION THEN JUMP TO 353 TO FIND A NEW COMBINATION
49700      C USING THE SAME MASK.
49800      IF (.NOT.ISOL) GO TO 353
49900      ISOL=.FALSE.
50000      ISTN(IB,1)=ITOT(NPS-1)
50100      C THIS LOOP FINDS THE MASK AGAIN AND SETS IZ EQUAL TO THE MOST
50200      C NUMBER OF TIMES THE NEW SPOKE CAN BE USED.
50300      DO 510 I=1,NF
50400      MASK(I)=LSP(IP,I).GT.0
50500      IF (.NOT.MASK(I)) GO TO 510
50600      IZ=N(NPS-1,I)/LSP(IP,I)
50700      IF (ISTN(IB,1).GT.IZ) ISTN(IB,1)=IZ
50800      510 CONTINUE
50900      IZ=0
51000      C THIS LOOP UPDATES THE PROBLEM AND SETS IZ TO TEST IF ONE
51100      C OR ZERO SPOKES ARE LEFT.
51200      DO 520 I=1,NF
51300      N(NPS-1,I)=N(NPS-1,I)-ISTN(IB,1)*LSP(IP,I)
51400      520 IZ=IZ+N(NPS-1,I)*IFREQ(I)
51500      C IF MORE THAN ONE SPOKE LEFT THEN JUMP TO 353 TO FIND A NEW
51600      C COMBINATION USING THE SAME MASK.
51700      IF (IZ.GT.ITOT(NPS-1)) GO TO 355
51800      C IF NO SPOKES LEFT THEN DONE SO GO TO 600.
51900      IF (IZ.EQ.0) GO TO 600
52000      C THIS IS A SPECIAL SECTION TO SPEED UP EXECUTION WHEN JUST
52100      C ONE SPOKE IS LEFT.
52200      MN=ITOT(1)
52300      J=0
52400      K=0
52500      ID=0
52600      IK=0
52700      IZ=0
52800      C IN THIS LOOP THE MASK IS SET TO THE REMAINING FREQUENCIES,
52900      C THE MAIN WHEEL IS FOUND, AND TESTS ARE MADE TO DETERMINE IF
53000      C THE REMAINING NUMBERS MAKE UP A COMBINATION THAT HAS ALREADY
53100      C BEEN TRIED (BY COMPARING TO THE LAST COMBINATION).
53200      DO 530 I=1,NF
53300      MASK(I)=N(NPS-1,I).GT.0
53400      IF (LSP(IP,I).NE.0) K=K+1
53500      IF (.NOT.MASK(I)) GO TO 525
53600      J=J+1
53700      MN=KGCF(MN,IDIV(NPS-1,I))
53800      IF (IZ.NE.0) GO TO 525
53900      IF (LSP(IP,I).GT.N(NPS-1,I)) IZ=1
54000      IF (LSP(IP,I).LT.N(NPS-1,I)) IZ=-1

```

```

54100 525 IF (J.GT.K.AND.ID.EQ.O) IK=1
54200 530 IF (J.NE.K) ID=1
54300 C THESE SIX LINES MAKE THE TESTS AND JUMP TO BACKTRACK IF THE
54400 C MASK DIDN'T PASS.
54500 IF (MN.EQ.1) GO TO 320
54600 IF (J.LT.K) GO TO 320
54700 IF (J.GT.K) GO TO 535
54800 IF (IK.NE.O) GO TO 320
54900 IF (ID.NE.O) GO TO 535
55000 IF (IZ.LE.O) GO TO 320
55100 C THE MASK WAS OK SO SET LSP AND DIVIDE RATES.
55200 535 IP=IP+1
55300 IB=IST(IP)
55400 ISTN(IB,2)=MN
55500 DO 540 I=1,NF
55600 LSP(IP,I)=O
55700 IF (.NOT.MASK(I)) GO TO 540
55800 LSP(IP,I)=N(NPS-1,I)
55900 IDIV(NPS,I)=IDIV(NPS-1,I)/MN
56000 540 CONTINUE
56100 LSTR(NPS)=LSTR(NPS)+1
56200 C SOLVE THE SUBPROBLEM.
56300 CALL SAME
56400 C IF A SOLUTION WAS FOUND THEN RESUME EXECUTION AT 505.
56500 C OTHERWISE GO TO BACKTRACK.
56600 IF (ISOL) GO TO 505
56700 NPS=NPS-1
56800 LSTR(NPS)=LSTR(NPS)-1
56900 GO TO 320
57000 C PROGRAM JUMPS HERE WHEN A SOLUTION IS FOUND. GO TO 700
57100 C IF IT'S A SOLUTION TO A SUBPROBLEM. OTHERWISE PRINTOUT
57200 C THE SOLUTION AND STATISTICS.
57300 600 IF (NPS.NE.2) GO TO 700
57400 IRS=MSOLN
57500 CALL JIFFY(MSOLN,III)
57600 NSOL=NSOL+1
57700 IF (NSOL.LT.NS1) GO TO 320
57800 WRITE (IODEV,620) NSOL
57900 620 FORMAT (//,3X,'SOLUTION NUMBER ',I5,/)
58000 IK=O
58100 IF (ISTN(1,2).EQ.1) IK=ISTN(1,1)
58200 IK1=IK
58300 IKK=O
58400 ID=MAIN
58500 625 IF (ID.EQ.O) GO TO 627
58600 IKK=IKK+1
58700 ID=ID-ISTN(IKK,1)
58800 IF (ISTN(IKK,2).NE.ISTN(IKK,3)) GO TO 627
58900 IF (ISTN(IKK,2).NE.ITOT(1)) GO TO 625
59000 IK=IK+ISTN(IKK,1)
59100 627 CONTINUE
59200 WRITE (IODEV,630) MAIN,IK
59300 630 FORMAT (8X,'1',' ',I5,' ',' ',I5)
59400 NCH=IK
59500 IZ=1
59600 IP=LSTR(2)
59700 J=1
59800 IN(1)=MAIN
59900 IX(1)=1
60000 IA(1)=1

```

```

60100      NFK=NFORG+1
60200      DO 640 I=1,NFORG
60300      IFOUT(I)=IFREQ(I)/2
60400      640      NOUT(I)=0
60500      IFOUT(NFK)=0
60600      NOUT(1)=IK1
60700      DO 670 I=1,IST(IP)-1
60800      IF (ISTN(I,2).EQ.1) GO TO 663
60900      IA(J+1)=IA(J)*ISTN(I,2)
61000      IF (IA(J+1).NE.ITOT(1)) GO TO 650
61100      IF (J.NE.1) GO TO 663
61200      ID=ISTN(I,1)
61300      IFR=ITOT(1)/2
61400      IKK=0
61500      GO TO 661
61600      650      WRITE (IODEV,652)
61700      652      FORMAT (' ',,$)
61800      DO 654 K=1,J
61900      654      WRITE (IODEV,655)
62000      655      FORMAT ('+',,$)
62100      IK=ISTN(I,3)
62200      IF (ISTN(I+1,2).EQ.1.AND.I.NE.IST(IP)-1) IK=ISTN(I+1,1)
62300      IKK=I
62400      ID=ISTN(I,2)
62500      IF (ISTN(I,2).EQ.ISTN(I,3)) GO TO 657
62600      656      IF (ID.EQ.0) GO TO 657
62700      IKK=IKK+1
62800      ID=ID-ISTN(IKK,1)
62900      IF (ISTN(IKK,2).NE.ISTN(IKK,3)) GO TO 657
63000      IF (IA(J+1)*ISTN(IKK,2).NE.ITOT(1)) GO TO 656
63100      IK=IK+ISTN(IKK,1)
63200      657      CONTINUE
63300      WRITE (IODEV,660) (ISTN(I,K),K=1,2),IK
63400      660      FORMAT ('+',I5,',',I5,',',I5)
63500      IX(J+1)=IX(J)*ISTN(I,1)
63600      IZ=IZ+IX(J+1)
63700      IF (IK.EQ.0) GO TO 663
63800      ID=IX(J+1)*IK
63900      NCH=NCH+ID
64000      IFR=ITOT(1)/IA(J+1)/2
64100      IKK=0
64200      661      IKK=IKK+1
64300      IF (IFR.LT.IFOUT(IKK)) GO TO 661
64400      IF (IFR.NE.IFOUT(IKK)) GO TO 662
64500      NOUT(IKK)=NOUT(IKK)+ID
64600      GO TO 663
64700      662      DO 6625 I2=IKK,NFK
64800      IK=IKK+NFK-I2
64900      IFOUT(IK+1)=IFOUT(IK)
65000      6625      NOUT(IK+1)=NOUT(IK)
65100      IFOUT(IKK)=IFR
65200      NOUT(IKK)=ID
65300      NFK=NFK+1
65400      IF (NFK.LE.99) GO TO 663
65500      NFK=NFK-1
65600      TYPE 6626
65700      6626      FORMAT (' ERROR -- OFLO ON # FREQUENCIES')
65800      663      IN(J)=IN(J)-ISTN(I,1)
65900      IF (ISTN(I,2).EQ.ISTN(I,3)) GO TO 665
66000      J=J+1

```

```

66100      IN(J)=ISTN(I,2)
66200      GO TO 670
66300  665  IF (IN(J).NE.0) GO TO 670
66400      J=J-1
66500      IF (J.NE.0) GO TO 665
66600  670  CONTINUE
66700      NFK=NFK-1
66800      ID=ITOT(1)*MAIN/2
66900      WRITE (IODEV,672) NFK, ID
67000  672  FORMAT (/,' FREQUENCIES USED:',I6,20X,'CLOCK RATE --',I1
67100      0)
67100      ID=0
67200  674  IK=ID+1
67300      ID=ID+7
67400      IF (ID.GT.NFK) ID=NFK
67500      WRITE (IODEV,676) (NOUT(I),I=IK,ID)
67600  676  FORMAT (/,' NUMBER:',3X,7I8)
67700      WRITE (IODEV,678) (IFOUT(I),I=IK,ID)
67800  678  FORMAT (' FREQUENCY:',7I8)
67900      IF (ID.LT.NFK) GO TO 674
68000      ID=NCH+3*IZ-1
68100      WRITE (IODEV,680) NCH,IZ,ID
68200  680  FORMAT (/,' CHANNELS:',I8,' WHEELS:',I8,
68300      1 ' MEMORY WORDS')
68400      IZ=ITCT(1)*MAIN/NGCF+2
68500      R=FLOAT(ID)/FLOAT(IZ)*100.
68600      WRITE (IODEV,690) IZ,R
68700  690  FORMAT (' 1 WHEEL, ',I8,' WORDS -- ',F5.1,'% (RATIO)')
68800      CALL JIFFY(MSOLN,III)
68900      RS=FLOAT(MSOLN-IRS)/60.
69000      RP=FLOAT(MSOLN-MPROB)/60.
69100      RT=FLOAT(MSOLN-MTOT)/60.
69200      WRITE (IODEV,695) RT,RP,RS
69300  695  FORMAT (' CPU TIMES (TOTAL,PROBLEM,SOLUTION) ',3F10.2,/)
69400      IF (NSOL.LT.NS2) GO TO 320
69500      ISOL=.TRUE.
69600      RETURN
69700  C  SOLUTION TO SUBPROBLEM WAS FOUND SO ADJUST A FEW NUMBERS AND
69800  C  RETURN.
69900  700  NPS=NPS-1
70000      IP=LSTR(NPS)-1
70100      IB=IST(IP)
70200      ISTN(IB,3)=0
70300      J=LSTR(NPS+1)
70400      IST(IP+1)=IST(J)
70500      ISOL=.TRUE.
70600      RETURN
70700      END
70800  C
70900  C
71000      SUBROUTINE SAME
71100  C
71200  C  THIS SUBROUTINE IS CALLED FROM SOLVE AND IS USED TO SETUP
71300  C  A SUBPROBLEM AND CALL SOLVE BACK TO SOLVE THIS SUBPROBLEM.
71400  C
71500      COMMON NF,N,IFREQ,NGCF,LSTR,MASK,IM,NPS,IDIV,MAIN,IODEV,
71600      1 IST,ITOT,NSOL,ISOL,LSP,ISTN,NCH,NFORG,NOUT(100),IFOUT(100)
71700      DIMENSION N(18,18),IFREQ(18),LSTR(18),MASK(18),IM(18),
71800      1 IDIV(18,18),IST(100),ITOT(18),LSP(100,18),ISTN(500,3)
71900      LOGICAL MASK,ISOL
72000      IP=LSTR(NPS)-1

```

```

72100      IB=IST(IP)
72200      IST(IP+1)=IST(IP)+1
72300      ITOT(NPS)=ITOT(NPS-1)/ISTN(IB,2)
72400      DO 100 I=1,NF
72500 100    N(NPS,I)=LSP(IP,I)
72600      IF (IM(NPS).EQ.1) GO TO 200
72700      CALL SOLVE
72800      RETURN
72900  C  SINGLE FREQUENCY PROBLEMS ARE SOLVED HERE AND A RETURN
73000  & IS MADE.
73100 200    ISOL=.TRUE.
73200      ISTN(IB,3)=ISTN(IB,2)
73300      RETURN
73400      END

```

```

00100 C
00200 C                               SCAN6.FOR
00300 C
00400 C
00500 C SCAN6 IS DIFFERENT FROM SCAN5 IN THE FOLLOWING WAYS.
00600 C SUBROUTINE TRIAL WAS CHANGED FOR QUICKER EXECUTION.
00700 C SUBROUTINES SETUP AND SAME WERE CHANGED TO USE A GREATEST
00800 C COMMON FACTOR ROUTINE FOR MORE STRAIGHT FORWARD EXECUTION
00900 C (THIS ALSO TAKES AWAY THE NEED FOR PRIME FACTOR ARRAYS).
01000 C
01100 C
01200 C          MAIN PROGRAM
01300 C
01400 C THE MAIN PROGRAM CALLS SUBROUTINES INPUT, SETUP,
01500 C LIST3, AND INC
01600 C
01700 C          DOUBLE PRECISION ANS
01800 C          CALL INPUT
01900 5          CALL SETUP
02000 C          CALL LIST3
02100 C          TYPE 10
02200 10         FORMAT (//, 5X, 'INCREASE MAIN WHEEL?', $)
02300 C          READ (5, 20) ANS
02400 20         FORMAT (A1)
02500 C          IF (ANS.NE.'Y') STOP
02600 C          CALL INC
02700 C          GO TO 5
02800 C          END
02900 C
03000 C
03100 C          SUBROUTINE INPUT
03200 C
03300 C THIS SUBROUTINE IS USED TO INPUT THE DATA FROM A FILE
03400 C OR FROM THE KEYBOARD. DATA IS INPUT AS NUMBER OF
03500 C FREQUENCIES, FREQUENCY. THE DATA IS STORED IN TWO ARRAYS,
03600 C N(I) AND IFREQ(I). THESE TWO ARRAYS ALONG WITH THE NUMBER OF
03700 C FREQUENCIES, NF, ARE PASSED TO THE REST OF THE PROGRAM
03800 C THROUGH THE COMMON STATEMENT.
03900 C
04000 C          COMMON NF, N, IFREQ, LCM, NGCF, LEND, MASK, IM, MAW,
04100 C          1 NPS, LSTSZ, IDIV, LST3, LISTR, ISTR
04200 C          DIMENSION N(15), IFREQ(15), MASK(15, 15), IM(15), MAW(15),
04300 C          1 LSTSZ(15), IDIV(15, 15), LST3(1000, 15), LISTR(2, 5000), ISTR(10
04400 C          00)
04500 C          LOGICAL MASK
04600 C          DOUBLE PRECISION FILE
04700 C          TYPE 10
04800 10         FORMAT (5X, 'IS DATA TO BE READ FROM A FILE?', $)
04900 20         READ (5, 20) J
05000 C          IF (J.NE.'Y') GO TO 100
05100 C DATA READ FROM FILE
05200 C          TYPE 30
05300 30         FORMAT (5X, 'TYPE INPUT FILNAM.EXT> ', $)
05400 C          READ (5, 40) FILE
05500 40         FORMAT (A10)
05600 C          OPEN (UNIT=10, DEVICE='DSK', ACCESS='SEQIN', FILE=FILE)
05700 C          K=10
05800 C          NF=1
05900 50         READ (K, 60, END=70) N(NF), IFREQ(NF)
06000 C          IF (N(NF).EQ.0) GO TO 70

```

```

06100 60      FORMAT (2I)
06200      NF=NF+1
06300      IF (NF.LE.15) GO TO 50
06400 70      NF=NF-1
06500      TYPE 80,NF
06600 80      FORMAT (///,2X,I4,'  FREQUENCIES INPUT',//)
06700      IF (K.EQ.10) CLOSE (UNIT=10,ACCESS='SEQIN',FILE=FILE)
06800      RETURN
06900 C DATA READ FROM KEYBOARD
07000 100     K=5
07100      NF=1
07200      TYPE 110
07300 110    FORMAT (5X,'TYPE DATA IN AS NUMBER FREQUENCY IN 2I FORMA
T')
07400      GO TO 50
07500      END
07600 C
07700 C
07800      SUBROUTINE SETUP
07900 C
08000 C THIS SUBROUTINE IS CALLED TO SET UP THE PROBLEM.
08100 C THE GREATEST COMMON FACTOR AND LOWEST COMMON MULTIPLE
08200 C OF THE FREQUENCIES ARE DETERMINED.
08300 C
08400 C THIS ROUTINE IS USED TO CREATE THE INITIAL PROBLEM AND
08500 C PUT ITS PARAMETERS ON THE PROBLEM STACK. THE PROBLEM STACK
08600 C INVOLVES THE FOLLOWING. IDIV(I,J) EQUALS THE DIVIDE RATE
08700 C FOR FREQUENCY J ON PROBLEM I. MASK(I,J) IS A LOGICAL ARRAY
08800 C WHOSE (I,J) ELEMENT IS TRUE WHEN FREQUENCY J IS USED IN
08900 C PROBLEM NUMBER I. IT IS SET TO ALL TRUE HERE FOR PROBLEM
09000 C ONE AS PROBLEM ONE INVOLVES ALL FREQUENCIES. IM(1) IS THE
09100 C NUMBER OF FREQUENCIES USED IN PROBLEM I. IM(1) IS SET
09200 C TO NF. MAW(1) EQUALS THE SIZE OF THE MAIN WHEEL FOR
09300 C PROBLEM I. MAW(1) IS DETERMINED. ISTR(1) IS SET TO ONE.
09400 C ARRAY ISTR AND ARRAY LSTSZ ARE DESCRIBED IN SUBROUTINE
09500 C LIST3.
09600 C
09700 C THE STACK POINTER FOR THE PROBLEM STACK, NPS, IS
09800 C INITIALIZED AT 1. IT IS DETERMINED IF ADDITIONAL FREQUENCIES
09900 C ARE NEEDED AND IF SO THEY ARE ADDED AT FREQUENCY NGCF.
10000 C LEND IS A POINTER TO THE ENTRY AFTER THE LAST ENTRY IN
10100 C THE STACK LST3. LEND IS SET TO 1. SUBROUTINE TRIAL IS
10200 C CALLED TO INITIALIZE SOME PARAMETERS THERE. AT THE END OF
10300 C THIS ROUTINE THE SYSTEM CLOCK FREQUENCY AND FREQUENCIES USED
10400 C ARE PRINTED OUT ON THE TTY.
10500 C
10600      COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
10700      1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
10800      DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
10900      1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
00)
11000      LOGICAL MASK
11100 C FIND GCF OF FREQUENCIES.
11200      NGCF=IFREQ(1)
11300      DO 10 I=2,NF
11400 10      NGCF=KGCF(NGCF,IFREQ(I))
11500 C FIND SUM OF (N(I)*IFREQ(I)) AND LCM OF FREQUENCIES.
11600      ITOTL=0
11700      LCM=NGCF
11800      IFRG=0
11900      DO 20 I=1,NF
12000      LCM=LCM*IFREQ(I)/KGCF(LCM,IFREQ(I))

```

```

12100      IF (IFREQ(I).EQ.NGCF) IFRG=I
12200      ITOTL=ITOTL+N(I)*IFREQ(I)
12300      C DETERMINE IF CHANNELS MUST BE ADDED. IF NOT GO TO 40.
12400      MAW(1)=ITOTL/LCM
12500      K=LCM*MAW(1)
12600      IF (K.EQ.ITOTL.AND.K.NE.LCM) GO TO 40
12700      C FROM HERE TO 40 ADD CHANNELS. MAKE SURE MAIN WHEEL IS OF AT
12800      C LEAST SIZE 2. CHANNELS ARE ADDED AT FREQUENCY NGCF.
12900      C RECOMPUTE ITOTL AND MAIN WHEEL SIZE.
13000      IF (ITOTL.LE.LCM) K=LCM
13100      IF (IFRG.NE.O) GO TO 30
13200      NF=NF+1
13300      IFRG=NF
13400      N(NF)=O
13500      IFREQ(NF)=NGCF
13600      30      IDF=(K+LCM-ITOTL)/NGCF
13700      N(IFRG)=N(IFRG)+IDF
13800      ITOTL=K+LCM
13900      MAW(1)=ITOTL/LCM
14000      40      CONTINUE
14100      C PUT GCF FREQUENCY FIRST FOR FASTER EXECUTION.
14200      IF (IFRG.EQ.O) GO TO 45
14300      T=N(1)
14400      N(1)=N(IFRG)
14500      N(IFRG)=T
14600      T=IFREQ(1)
14700      IFREQ(1)=IFREQ(IFRG)
14800      IFREQ(IFRG)=T
14900      45      CONTINUE
15000      C COMPUTE DIVIDE RATES AND SET MASK TO ALL TRUE.
15100      DO 50 I=1,NF
15200      IDIV(1,I)=LCM/IFREQ(I)
15300      50      MASK(1,I)=.TRUE.
15400      C INITIALIZE A FEW MORE VARIABLES.
15500      NPS=1
15600      IM(1)=NF
15700      CALL TRIAL
15800      LEND=1
15900      ISTR(1)=1
16000      C PRINT OUT A FEW STASTICS.
16100      TYPE 500,ITOTL
16200      500      FORMAT (///,2X,'SYSTEM CLOCK FREQUENCY --'I7,/,/,
16300      1 2X,'FREQUENCIES USED')
16400      DO 700 I=1,NF
16500      TYPE 600,N(I),IFREQ(I)
16600      600      FORMAT (2X,2I8)
16700      700      CONTINUE
16800      RETURN
16900      END
17000      C
17100      C
17200      FUNCTION KGCF(I,J)
17300      C
17400      C THIS FUNCTION ROUTINE WILL COMPUTE THE GREATEST COMMON FACTOR
17500      C OF TWO INTEGERS. I AND J ARE NOT CHANGED.
17600      C
17700      K=I
17800      KGCF=J
17900      10      CONTINUE
18000      IF (K.EQ.O) RETURN

```

```

18100      L=KGCF-(KGCF/K)*K
18200      KGCF=K
18300      K=L
18400      GO TO 10
18500      END
18600  C
18700  C
18800      SUBROUTINE LIST3
18900  C
19000  C THIS SUBROUTINE WILL FIND ALL SOLUTIONS TO THE PROBLEM WHICH
19100  C IS ON TOP OF THE PROBLEM STACK. A SOLUTION IS A SET OF
19200  C WHEELS WHICH CONTAINS AT LEAST ONE OF EACH FREQUENCY DESIRED
19300  C AND NONE OF THE OTHER FREQUENCIES. A RETURN WILL NOT BE MADE
19400  C UNTIL ALL SOLUTIONS ARE PLACED IN LST3 AND LISTR. LST3
19500  C IS A STACK ARRAY WHERE EACH ROW CONTAINS THE NUMBER USED AT
19600  C EACH FREQUENCY FOR THAT PARTICULAR SOLUTION. LEND IS A
19700  C POINTER TO THE ENTRY AFTER THE LAST ENTRY IN THIS STACK.
19800  C LSTSZ(I) IS THE NUMBER OF ENTRIES IN LST3 WHICH ARE
19900  C SOLUTIONS OF PROBLEM I. THEREFORE FOR A PROBLEM ON TOP OF
20000  C THE PROBLEM STACK, ITS SOLUTIONS RANGE FROM LEND-LSTSZ(NPS)
20100  C TO LEND-1 IN LST3. CORRESPONDING TO EACH ROW ENTRY IN
20200  C LST3 THERE IS A STRING WHICH TELLS HOW THE WHEELS ARE
20300  C GENERATED FOR THIS SOLUTION. THESE STRINGS ARE PLACED IN THE
20400  C STACK LISTR(I,J). ISTR IS AN ARRAY OF POINTERS WHICH
20500  C TELLS WHERE STRING I BEGINS. THUS FOR ROW ENTRY I IN
20600  C LST3, THE CORRESPONDING STRING IS IN LISTR(K,L) WHERE L
20700  C RANGES FROM ISTR(I) TO ISTR(I+1)-1 (K HAS VALUES 1 AND 2).
20800  C
20900  C FOR A PROBLEM INPUT TO THIS ROUTINE IT FIRST FINDS WHAT
21000  C ALL THE POSSIBLE SPOKES ARE. IT THEN CALLS COMB WHICH
21100  C TAKES ALL COMBINATIONS OF THESE SPOKES WHICH SATISFY THE
21200  C CONSTRAINTS (MAIN WHEEL SIZE, USE OF ALL THE GIVEN
21300  C FREQUENCIES). THE NUMBER OF FREQUENCIES USED MUST BE AT
21400  C LEAST TWO FOR THIS SUBROUTINE (ONE FREQUENCY IS A TRIVIAL
21500  C CASE). IF THE NUMBER OF FREQUENCIES IS TWO FOR THE PROBLEM
21600  C ON TOP OF THE STACK, THE SOLUTION IS EASY AND THE ROUTINE
21700  C SOLVES IT AND RETURNS.
21800  C
21900  C FOR A THREE FREQUENCY PROBLEM THE SUBROUTINE BEGINS BY
22000  C PLACING ALL THE SINGLE FREQUENCY SOLUTIONS FOR SPOKES ON THE
22100  C STACK. NEXT ALL COMBINATIONS OF TWO FREQUENCIES ARE
22200  C DETERMINED. FOR EACH FREQUENCY SET THE ROUTINE PUTS THESE
22300  C ON THE PROBLEM STACK AND FINDS ALL SPOKE SOLUTIONS USING
22400  C THESE TWO BY CALLING SUBROUTINE SAME. SAME IS A ROUTINE
22500  C WHICH FIGURES THE PROBLEM PARAMETERS (E.G. MAIN WHEEL SIZE)
22600  C AND THEN CALLS LIST3 TO SOLVE THE PROBLEM (LIST3 IN EFFECT
22700  C CALLED ITSELF TO SOLVE THIS SUBPROBLEM). THE TWO
22800  C FREQUENCY PROBLEM SOLUTIONS ARE NOW POSSIBLE SPOKES FOR
22900  C THE THREE FREQUENCY PROBLEM. AFTER ALL TWO FREQUENCY
23000  C SOLUTIONS ARE SOLVED, THE ROUTINE CALLS COMB TO COMBINE
23100  C THESE SPOKES TO FORM SOLUTIONS OF THE THREE FREQUENCY
23200  C PROBLEM. THE ROUTINE THEN RETURNS.
23300  C
23400  C IN THIS MANNER, FOR A PROBLEM OF N FREQUENCIES, THIS ROUTINE
23500  C WILL FIRST RECORD POSSIBLE SPOKES USING SINGLE FREQUENCIES,
23600  C THEN POSSIBLE SPOKES USING ALL COMBINATIONS OF TWO
23700  C FREQUENCIES, AND SO ON UP TO N-1 FREQUENCIES. IT THEN TAKES
23800  C ALL COMBINATIONS OF THESE SPOKES BY CALLING COMB TO FORM
23900  C SOLUTIONS. IN SOLVING ANY OF THESE SPOKE PROBLEMS OF
24000  C GREATER THAN TWO FREQUENCIES IT CREATES THE PROBLEM, PUTS

```

```

24100 C IT ON THE PROBLEM STACK AND CALLS ITSELF.
24200 C
24300 COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
24400 1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
24500 DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
24600 1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
00)
24700 LOGICAL MASK
24800 C INITIALIZE THE PROBLEM LIST SIZE TO ZERO.
24900 LSTSZ(NPS)=0
25000 C FROM HERE TO 10 - FIND SINGLE FREQUENCY SPOKES (USING MASK)
25100 C AND PUT THESE SOLUTIONS IN LST3 AND LISTR. UPDATE THE
25200 C STACK POINTERS.
25300 DO 10 I=1,NF
25400 IF (.NOT.MASK(NPS,I).OR.IDIV(NPS,I).GT.N(I)) GO TO 10
25500 DO 5 J=1,NF
25600 5 LST3(LEND,J)=0
25700 LST3(LEND,I)=IDIV(NPS,I)
25800 IZ=ISTR(LEND)
25900 IF (IZ.GT.5000) CALL ER(2)
26000 LISTR(1,IZ)=LST3(LEND,I)
26100 LISTR(2,IZ)=LST3(LEND,I)
26200 LEND=LEND+1
26300 IF (LEND.GT.1000) CALL ER(1)
26400 LSTSZ(NPS)=LSTSZ(NPS)+1
26500 ISTR(LEND)=ISTR(LEND-1)+1
26600 10 CONTINUE
26700 C IF ITS A TWO FREQUENCY PROBLEM THEN CALL COMB AND RETURN.
26800 IF (IM(NPS).NE.2) GO TO 20
26900 CALL COMB
27000 RETURN
27100 C FROM HERE TO 30 - CREATE A TWO FREQUENCY PROBLEM USING THE
27200 C FIRST TWO FREQUENCIES OF THE CURRENT PROBLEM. PUSH THIS ON
27300 C THE PROBLEM STACK.
27400 20 NPS=NPS+1
27500 IM(NPS)=2
27600 25 J=IM(NPS)
27700 DO 30 I=1,NF
27800 MASK(NPS,I)=MASK(NPS-1,I).AND.J.GT.0
27900 IF (MASK(NPS,I)) J=J-1
28000 30 CONTINUE
28100 C SOLVE THIS NEW PROBLEM.
28200 35 CALL SAME
28300 C IF THE PROBLEM JUST SOLVED YIELDED POSSIBLE SPOKES TO THE
28400 C MAIN WHEEL OF THE ORIGINAL PROBLEM THEN CALL TRIAL. TRIAL
28500 C IS A LOOK AHEAD SUBROUTINE SIMILAR TO COMB IN FUNCTION.
28600 C TRIAL HOWEVER WILL GO AHEAD AND TRY COMBINATIONS OF SPOKES
28700 C EVEN THOUGH ITS LIST OF SPOKES IS NOT COMPLETE. IT LEAVES
28800 C THE SPOKES AS THEY ARE AND ONLY OUTPUTS THE NEW SOLUTIONS.
28900 IF (NPS.EQ.2) CALL TRIAL
29000 C FROM HERE TO 60 THE MASK FOR THE PROBLEM JUST SOLVED IS
29100 C ALTERED SO IT INCLUDES THE NEXT COMBINATION OF N FREQUENCIES
29200 C USING FREQUENCIES FROM THE SUPER PROBLEM (ONE DOWN ON THE
29300 C PROBLEM STACK). I.E. IT FINDS THE NEXT N FREQUENCY PROBLEM
29400 C USING ITS SUPERPROBLEMS' FREQUENCIES. IF A NEW COMBINATION
29500 C IS FOUND, THE PROGRAM SOLVES IT BY BRANCHING TO 35. IF ALL
29600 C COMBINATIONS FOR THAT N HAVE BEEN TRIED THEN THE PROGRAM
29700 C BRANCHES TO 70.
29800 I=NF+1
29900 J=0
30000 40 I=I-1

```

```

30100      IF (.NOT.MASK(NPS-1,I)) GO TO 40
30200      IF (MASK(NPS,I)) GO TO 50
30300      J=-1
30400      GO TO 40
30500  50     J=J+1
30600      MASK(NPS,I)=.FALSE.
30700      IF (J.GE.IM(NPS)) GO TO 70
30800      IF (J.GT.O) GO TO 40
30900      K=0
31000      DO 55 J=1,NF
31100  55     IF (MASK(NPS,J)) K=K+1
31200      K=IM(NPS)-K
31300      DO 60 J=I+1,NF
31400      IF (.NOT.MASK(NPS-1,J)) GO TO 60
31500      MASK(NPS,J)=K.GT.O
31600      K=K-1
31700  60     CONTINUE
31800      GO TO 35
31900  C THE PROGRAM BRANCHES HERE IF SOLUTIONS TO ALL PROBLEMS
32000  C OF THAT NUMBER OF FREQUENCIES HAVE BEEN ADDED TO THE LIST OF
32100  C SPOKES FOR THE SUPER PROBLEM. IF THIS COMPLETES THE LIST
32200  C OF SPOKES FOR THE SUPER PROBLEM THEN GO TO 80 TO COMBINE
32300  C THEM. IF NOT THEN ADD ONE TO THE NUMBER OF FREQUENCIES USED
32400  C AND GO TO 25 TO FIND THE FIRST SET USING THAT NUMBER
32500  C OF FREQUENCIES.
32600  70     CONTINUE
32700      IF (IM(NPS).EQ.IM(NPS-1)-1) GO TO 80
32800      IM(NPS)=IM(NPS)+1
32900      GO TO 25
33000  C TAKE THE LAST PROBLEM OFF THE STACK (ALREADY SOLVED). THEN
33100  C CALL COMB TO COMBINE THE SPOKES AND SOLVE THE PROBLEM
33200  C GIVEN TO THE SUBROUTINE. THEN RETURN.
33300  80     NPS=NPS-1
33400      IF (NPS.NE.1) CALL COMB
33500      RETURN
33600      END
33700  C
33800  C
33900      SUBROUTINE SAME
34000  C
34100  C THIS SUBROUTINE IS CALLED BY LIST3. IT SETS UP THE
34200  C PROBLEM AND THEN CALLS LIST3 BACK. MAW(NPS), THE MAIN
34300  C WHEEL SIZE, IS FOUND (GCF OF DIVIDE RATES) AND PUT ON THE
34400  C PROBLEM STACK. ALSO THE SUBPROBLEMS' DIVIDE RATES ARE
34500  C FOUND AND PUT ON THE STACK AS IDIV(I,J) (EQUAL TO THE
34600  C DIVIDE RATE OF FREQUENCY J FOR PROBLEM I).
34700  C
34800      COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
34900      1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
35000      DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
35100      1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
35200      00)
35200      LOGICAL MASK
35300  C FROM HERE TO 10 DETERMINE MAIN WHEEL SIZE FOR SUBPROBLEM
35400  C AS THE GREATEST COMMON FACTOR OF THE FREQUENCIES USED.
35500      MAW(NPS)=LCM
35600      DO 10 I=1,NF
35700      IF (.NOT.MASK(NPS,I)) GO TO 10
35800      MAW(NPS)=KGCF(MAW(NPS),IDIV(NPS-1,I))
35900  10     CONTINUE
36000  C RETURN IF MAIN WHEEL IS OF SIZE 1. OTHERWISE COMPUTE

```

```

36100 C DIVIDE RATES.
36200 IF (MAW(NPS).EQ.1) RETURN
36300 DO 20 I=1,NF
36400 IDIV(NPS,I)=IDIV(NPS-1,I)/MAW(NPS)
36500 C CALL LIST3 TO SOLVE THIS SUBPROBLEM AND THEN RETURN.
36600 CALL LIST3
36700 RETURN
36800 END
36900 C
37000 C
37100 SUBROUTINE COMB
37200 C
37300 C THIS SUBROUTINE WILL USE ALL THE SOLUTIONS FOR SPOKES FOR
37400 C A PROBLEM AND WILL COMBINE THEM TO FORM ALL POSSIBLE
37500 C SOLUTIONS FOR THE PROBLEM. THE PROBLEM USES LSTSZ(NPS)
37600 C TO DEFINE HOW MANY ENTRIES OF LST3 ARE SPOKES FOR THAT
37700 C PROBLEM. IT TAKES THESE ENTRIES (ALONG WITH THEIR STRING
37800 C FORM IN LISTR(I,J)) AND PUTS THESE IN TEMPORARY ARRAYS.
37900 C USING THESE ARRAYS, THE COMBINATIONS ARE TAKEN AND THE
38000 C SOLUTIONS PUT BACK IN LST3 AND LISTR. EACH TIME A SOLUTION
38100 C IS FOUND LSTSZ FOR THE PREVIOUS PROBLEM IS INCREASED BY ONE
38200 C BECAUSE EACH SOLUTION TO THIS PROBLEM IS A SPOKE TO THE
38300 C PREVIOUS PROBLEM. THE PROGRAM RETURNS WHEN ALL COMBINATIONS
38400 C HAVE BEEN TRIED. EXTRA ARRAYS USED ARE AS FOLLOWS. ITEMP1
38500 C IS A TEMPORARY ARRAY FOR STORING THE PROBLEMS' SECTION OF
38600 C LST3. ITEMP2 IS A TEMPORARY ARRAY OF THE CORRESPONDING
38700 C STRINGS FROM LISTR. ITEMP3 IS COPIED FROM ISTR AS THE
38800 C POINTERS FOR THE STRINGS IN ITEMP2. MPNT IS A SET OF
38900 C POINTERS (ONE FOR EACH SLOT ON THE MAIN WHEEL) WHICH
39000 C POINT TO WHICH ENTRY FROM THE LIST OF SPOKES IN ITEMP1 IS
39100 C USED TO FILL THAT SLOT.
39200 C
39300 COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
39400 1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
39500 DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
39600 1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
39700 LOGICAL MASK
39800 DIMENSION ITEMP1(500,15),ITEMP2(2,2000),ITEMP3(500),MPNT
39900 (500)
40000 C RETURN IF THERE WERE NO SPOKE SOLUTIONS.
40100 IF (LSTSZ(NPS).EQ.0) RETURN
40200 C NEW LIST END IS FOUND.
40300 LEND=LEND-LSTSZ(NPS)
40400 LENOLD=LEND
40500 C FROM HERE TO 20 ITEMP1 AND ITEMP3 ARE COPIED FROM APPROPRIATE
40600 C SECTIONS OF LST3 AND ISTR RESPECTIVELY.
40700 IF (LSTSZ(NPS).GT.500) CALL ER(3)
40800 DO 20 I=1,LSTSZ(NPS)
40900 DO 10 J=1,NF
41000 ITEMP1(I,J)=LST3(LEND+I-1,J)
41100 ITEMP3(I)=ISTR(LEND+I-1)-ISTR(LEND)+1
41200 C CONTINUE
41300 C ISTR AND LIKEWISE ITEMP3 CONTAIN A FINAL ENTRY OF WHERE THE
41400 C NEXT STRING SHOULD START IF ONE WERE ADDED.
41500 I=LSTSZ(NPS)+1
41600 ITEMP3(I)=ISTR(LEND+I-1)-ISTR(LEND)+1
41700 C IF (ITEMP3(I).GT.2000) CALL ER(4)
41800 C THIS LOOP COPIES ITEMP2 FROM LISTR.
41900 DO 30 J=1,ITEMP3(I)-1
42000 K=ISTR(LEND)+J-1
ITEMP2(1,J)=LISTR(1,K)

```

```

42100 30      ITEMP2(2,J)=LISTR(2,K)
42200 C      J IS THE NUMBER OF SPOKES WHICH ARE CURRENTLY FILLED.
42300 C      INITIALLY USE ONE SPOKE, IT BEING THE FIRST ENTRY FROM
42400 C      THE LIST.
42500      J=1
42600      MPNT(1)=1
42700 C      FROM HERE TO 60 THE CURRENT SELECTION OF SPOKES IS TESTED
42800 C      TO SEE IF ANY FREQUENCY OCCURS MORE THAN N(I) FOR THAT
42900 C      FREQUENCY. IF SO GO TO 70. NALL IS SET TO 1 IF ANY
43000 C      FREQUENCY IN THE PROBLEM IS NOT USED.
43100 40      NALL=0
43200      DO 60 K=1,NF
43300      IF (.NOT.MASK(NPS,K)) GO TO 60
43400      ICNT=0
43500      DO 50 L=1,J
43600      IP=MPNT(L)
43700      ICNT=ICNT+ITEMP1(IP,K)
43800      IF (ICNT.GT.N(K)) GO TO 70
43900 50      CONTINUE
44000      IF (ICNT.EQ.0) NALL =1
44100 60      CONTINUE
44200 C      AT THIS POINT NO FREQUENCY WAS USED TOO OFTEN SO SEE IF THE
44300 C      NUMBER OF SPOKES EQUALS THE MAIN WHEEL SIZE. IF SO GO TO
44400 C      80. OTHERWISE ADD A NEW SPOKE WITH THIS SPOKE POINTING TO
44500 C      THE SAME SPOKE AS THE PREVIOUS ONE. I.E. ADD A NEW SPOKE
44600 C      LIKE THE ONE BEFORE IT. THEN BRANCH TO 40 TO TEST THIS
44700 C      NEW SELECTION OF SPOKES.
44800      IF (J.EQ.MAW(NPS)) GO TO 80
44900      J=J+1
45000      MPNT(J)=MPNT(J-1)
45100      GO TO 40
45200 C      THE PROGRAM BRANCHES HERE FROM ABOVE WHEN THE CURRENT
45300 C      SELECTION OF SPOKES RESULTS IN A FREQUENCY OCCURRING TOO
45400 C      OFTEN. SO CHANGE THE LAST SPOKE ON THE LIST IF THERE IS
45500 C      SOMETHING LEFT TO CHANGE IT TO AND GO TO 40 TO TEST THIS
45600 C      SELECTION. OTHERWISE TAKE OFF THE LAST SPOKE AND BRANCH
45700 C      BACK TO 70 TO TRY AND CHANGE THE PREVIOUS SPOKE.
45800 70      MPNT(J)=MPNT(J)+1
45900      IF (MPNT(J).LE.LSTSZ(NPS)) GO TO 40
46000      J=J-1
46100      IF (J.GT.0) GO TO 70
46200 C      IF THERE IS NOTHING LEFT TO TRY THEN ADJUST PREVIOUS
46300 C      PROBLEMS' LIST SIZE AND RETURN.
46400      LSTSZ(NPS)=LEND-LENOLD
46500      IF (NPS.NE.1) LSTSZ(NPS-1)=LSTSZ(NPS-1)+LSTSZ(NPS)
46600      RETURN
46700 C      PROGRAM BRANCHES HERE IF ALL SPOKES ARE FILLES PROPERLY.
46800 C      TRY AGAIN IF NOT ALL FREQUENCIES WERE USED.
46900 80      IF (NALL.EQ.1) GO TO 70
47000 C      GOOD SOLUTION RESULTED. THE REST OF THIS ROUTINE RECORDS
47100 C      IT IN LST3, LISTR, AND ISTR. UPDATE POINTERS.
47200      LEND1=LEND+1
47300      IF (LEND1.GT.1000) CALL ER(1)
47400      ISTR(LEND1)=ISTR(LEND)
47500      IS=ISTR(LEND1)
47600      IF (IS.GT.5000) CALL ER(2)
47700 C      BEFORE TYING STRINGS TOGETHER IN LISTR, PRECEED THEM WITH
47800 C      THEIR MAIN WHEEL SIZE (ASSUME NO FREQUENCIES LIE ON
47900 C      THAT WHEEL).
48000      LISTR(1,IS)=MAW(NPS)

```

```

48100          LISTR(2,IS)=0
48200          IR=IS
48300          IS=IS+1
48400      C   THIS LOOP (TO 90) ENTERS THE SOLUTION IN LST3 (BY SUMMING
48500      C   LST3 ELEMENTS FROM THE SPOKES USED).
48600          DO 90 K=1,NF
48700          LST3(LEND,K)=0
48800          DO 90 L=1,J
48900          IP=MPNT(L)
49000      90   LST3(LEND,K)=LST3(LEND,K)+ITEMP1(IP,K)
49100      C   THIS LOOP (TO 100) COPIES THE STRINGS FOR EACH SPOKE INTO
49200      C   LISTR FROM ITEMP2.
49300          DO 100 L=1,J
49400          L1=MPNT(L)
49500          L2=ITEMP3(L1)
49600          L3=ITEMP3(L1+1)-1
49700      C   IN THESE THREE LINES, A SPOKE THAT CONTAINS A SINGLE
49800      C   FREQUENCY ON THE MAIN WHEEL IS SKIPPED IN COPYING BUT
49900      C   THE NUMBER OF FREQUENCIES ON THE MAIN WHEEL IS INCREMENTED.
50000          IF (ITEMP2(1,L2).NE.1) GO TO 95
50100          LISTR(2,IR)=LISTR(2,IR)+1
50200          GO TO 100
50300      95   CONTINUE
50400      C   THIS LOOP (TO 98) ACTUALLY COPIES THE STRING.
50500          DO 98 K=L2,L3
50600          LISTR(1,IS)=ITEMP2(1,K)
50700          LISTR(2,IS)=ITEMP2(2,K)
50800      98   IS=IS+1
50900          IF (IS.GT.5000) CALL ER(2)
51000      100  CONTINUE
51100          ISTR(LEND+1)=IS
51200      C   THE NEXT LINE ONLY APPLIES TO A PROBLEM WHERE THE ORIGINAL
51300      C   PROBLEM IS JUST TWO FREQUENCIES.
51400          IF (NPS.EQ.1) CALL OUTPUT
51500          LEND=LEND+1
51600          GO TO 70
51700      END
51800      C
51900      C
52000          SUBROUTINE OUTPUT
52100      C
52200      C   SUBROUTINE USED TO OUTPUT SOLUTIONS OF MAIN PROBLEMS WITH
52300      C   JUST TWO FREQUENCIES. (ROUTINE USED MORE WITH EARLIER
52400      C   VERSIONS OF SCAN).
52500      C
52600          COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
52700          1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
52800          DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
52900          1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
00)
53000          LOGICAL MASK
53100          I=LEND
53200          TYPE 25,I
53300      25   FORMAT (///,5X,'SOLUTION NUMBER',I4,/)
53400          DO 40 J=ISTR(I),ISTR(I+1)-1
53500          TYPE 30,(LISTR(K,J),K=1,2)
53600          30   FORMAT (5X,2(I5,' ','I5))
53700          40   CONTINUE
53800          RETURN
53900          END
54000      C

```

```

54100 C
54200 SUBROUTINE TRIAL
54300 C
54400 C THIS SUBROUTINE HAS THE SAME FUNCTION AS SUBROUTINE COMB
54500 C ONLY IT IS CALLED AS A LOOK AHEAD AND DOES NOT CHANGE
54600 C ANY OF LIST3. IT IS CALLED EACH TIME A NEW SET OF SPOKE
54700 C SOLUTIONS ARE FOUND FOR THE MAIN WHEEL (NPS=2). COMBINATIONS
54800 C OF THESE SPOKES ARE TRIED TO FIND NEW SOLUTIONS TO THE MAIN
54900 C PROBLEM. TEMPORARY ARRAYS ARE NOT NEEDED AS NO LISTS
55000 C ARE CHANGED. HERE AGAIN MPNT IS AN ARRAY OF POINTERS TO
55100 C SPOKES USED FOR THE SOLUTIONS, ONLY POINTERS ALWAYS
55200 C START FROM THE END SO THE ROUTINE CAN STOP BEFORE DUPLICATE
55300 C SOLUTIONS ARE FOUND. AN ARRAY ICNT IS USED TO STORE LST3
55400 C TOTALS USED FOR A SOLUTION.
55500 C
55600 COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
55700 1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
55800 DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
55900 1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
56000 LOGICAL MASK
56100 DIMENSION MPNT(500),ICNT(15)
56200 DOUBLE PRECISION JA
56300 C WHEN THE SUBROUTINE IS CALLED FROM SETUP NPS IS 1 SO THESE
56400 C VARIABLES ARE INITIALIZED.
56500 IF (NPS.NE.1) GO TO 10
56600 LO=0
56700 NSOL=0
56800 RETURN
56900 10 CONTINUE
57000 C IF NOTHING NEW WAS ADDED TO THE LIST THEN RETURN.
57100 IF (LSTSZ(1).EQ.LO) RETURN
57200 C POINT THE FIRST SPOKE AT THE LAST ENTRY.
57300 MPNT(1)=LSTSZ(1)
57400 J=1
57500 C INITIALIZE TOTALS TO ZERO.
57600 DO 20 K=1,NF
57700 20 ICNT(K)=0
57800 40 CONTINUE
57900 IP=MPNT(J)
58000 C ADD SPOKE TOTALS FOR PROBLEM. BRANCH OUT IF A NUMBER
58100 C IS EXCEEDED.
58200 DO 50 K=1,NF
58300 ICNT(K)=ICNT(K)+LST3(IP,K)
58400 IF (ICNT(K).GT.N(K)) GO TO 60
58500 50 CONTINUE
58600 C TOTALS ARE OK SO FAR. BRANCH TO 80 IF ENOUGH WERE USED.
58700 IF (J.EQ.MAW(1)) GO TO 80
58800 C ADD ANOTHER SPOKE AT LAST SPOKES VALUE AND REEVALUATE.
58900 J=J+1
59000 MPNT(J)=MPNT(J-1)
59100 GO TO 40
59200 55 IP=MPNT(J)
59300 K=NF
59400 C SPOKE ENTRY OVERFLOWED A TOTAL (TOO MANY CHANNELS AT SOME
59500 C FREQUENCY) - SO SUBTRACT IT FROM TOTALS.
59600 60 CONTINUE
59700 DO 65 L=1,K
59800 65 ICNT(L)=ICNT(L)-LST3(IP,L)
59900 C SET SPOKE POINTER TO NEXT ENTRY IF POSSIBLE AND REEVALUATE.
60000 C NOTE LIST IS GONE THROUGH BACKWARDS.

```

```

60100      MPNT(J)=MPNT(J)-1
60200      IF (MPNT(J).GT.O.AND.MPNT(1).GT.LO) GO TO 40
60300  C   OTHERWISE ADJUST PREVIOUS POINTER IF POSSIBLE.
60400      J=J-1
60500      IF (J.GT.O) GO TO 55
60600  C   IF NO POINTERS LEFT TO ADJUST THEN RETURN.
60700      LO=LSTSZ(1)
60800      RETURN
60900  C   PROGRAM BRANCHES HERE IF THE SOLUTION IS VALID (NO TOTAL
61000  C   CAME TO MORE THAN THE NUMBER FOR THAT FREQUENCY).
61100  C   FROM HERE TO THE END OF THE ROUTINE THE SOLUTION
61200  C   IS OUTPUT.
61300  80      NSOL=NSOL+1
61400          TYPE 90,NSOL
61500  90      FORMAT (///,5X,'SOLUTION NUMBER',I4,/)
61600  C   FROM HERE TO 100 SET MN EQUAL TO THE NUMBER OF SPOKES WHICH
61700  C   INDUCE AN ENTRY ON THE MAIN WHEEL.
61800      MN=O
61900      DO 100 L=1,J
62000          L1=MPNT(L)
62100          L2=ISTR(L1)
62200  100      IF (LISTR(1,L2).EQ.1) MN=MN+1
62300  C   OUTPUT MAIN WHEEL STATISTICS.
62400          TYPE 110,MAW(1),MN
62500  110      FORMAT (5X,I5,',',I5)
62600  C   THIS LOOP (TO 200) OUTPUTS THE STRINGS FOR EACH SPOKE.
62700      DO 200 L=1,J
62800          L1=MPNT(L)
62900          L2=ISTR(L1)
63000          L3=ISTR(L1+1)-1
63100          IF (LISTR(1,L2).EQ.1) GO TO 200
63200          DO 150 K=L2,L3
63300          TYPE 110,LISTR(1,K),LISTR(2,K)
63400  150      CONTINUE
63500  200      CONTINUE
63600  C   CONTINUE THE SEARCH IF THE OPERATOR SO DESIRES.
63700          TYPE 300
63800  300      FORMAT (/ ,5X,'CONTINUE SEARCH?',S)
63900          READ (5,310) JA
64000  310      FORMAT (A1)
64100          IF (JA.NE.'Y') STOP
64200          GO TO 55
64300      END
64400  C
64500  C
64600      SUBROUTINE INC
64700  C
64800  C   THIS SUBROUTINE IS USED TO INCREASE THE NUMBER OF CHANNELS
64900  C   AT THE FREQUENCY GCF IN THE EVENT THAT NO SOLUTIONS EXIST FOR
65000  C   THE PROBLEM AS IS.  THIS SUBROUTINE IS CALLED FROM THE
65100  C   MAIN PROGRAM.
65200  C
65300      COMMON NF,N,IFREQ,LCM,NGCF,LEND,MASK,IM,MAW,
65400      1 NPS,LSTSZ,IDIV,LST3,LISTR,ISTR
65500      DIMENSION N(15),IFREQ(15),MASK(15,15),IM(15),MAW(15),
65600      1 LSTSZ(15),IDIV(15,15),LST3(1000,15),LISTR(2,5000),ISTR(10
65700      00)
65700      LOGICAL MASK
65800  C   FIND IF ANY FREQUENCY IS AT FREQUENCY NGCF.
65900      DO 10 I=1,NF
66000      IF (IFREQ(I).EQ.NGCF) GO TO 20

```

```

66100 10      CONTINUE
66200 C  IF FREQUENCY DOESN'T EXIST THEN CREATE IT.
66300      NF=NF+1
66400      IFREQ(NF)=NGCF
66500      N(NF)=LCM/NGCF
66600      RETURN
66700 C  ADD CHANNELS TO THE EXISTING FREQUENCY.
66800 20      N(I)=N(I)+LCM/NGCF
66900      RETURN
67000      END
67100 C
67200 C
67300      SUBROUTINE ER(N)
67400 C
67500 C  SUBROUTINE CALLED WHEN AN OVERFLOW IS DETECTED ON ONE OF
67600 C  THE STACKS.  HALTS PROGRAM AND OUTPUT MESSAGE.
67700 C
67800      GO TO (10,20,30,40),N
67900 10      TYPE 15
68000 15      FORMAT (//,2X,'OFLO ON LST3')
68100      STOP
68200 20      TYPE 25
68300 25      FORMAT (//,2X,'OFLO ON LISTR')
68400      STOP
68500 30      TYPE 35
68600 35      FORMAT (//,2X,'OFLO ON ITEMP1')
68700      STOP
68800 40      TYPE 45
68900 45      FORMAT (//,2X,'OFLO ON ITEMP2')
69000      STOP
69100      END

```

```

00100 C
00200 C
00300 C
00400 C
00500 C
00600 C
00700 C
00800 C
00900 C
01000 C
01100 C
01200 C
01300 C
01400 C
01500 C
01600 C
01700 C
01800 C
01900 C
02000 C
02100 C
02200 C
02300 C
02400 C
02500 C
02600 C
02700 C
02800 C
02900 C
03000 C
03100 C
03200 C
03300 C
03400 C
03500 C
03600 C
03700 C
03800 C
03900 C
04000 C
04100 C
04200 C
04300 C
04400 C
04500 C
04600 C
04700 C
04800 C
04900 C
05000 C
05100 C
05200 C
05300 C
05400 C
05500 C
05600 C
05700 C
05800 C
05900 C
06000 C

                                SIMP.FOR

                                MAIN PROGRAM

THE MAIN PROGRAM IS USED TO INPUT THE DATA FROM A FILE
OR FROM THE KEYBOARD. DATA IS INPUT AS NUMBER OF
FREQUENCIES, FREQUENCY. THE DATA IS STORED IN TWO ARRAYS,
N(I) AND IFREQ(I). THESE TWO ARRAYS ALONG WITH THE NUMBER
OF FREQUENCIES, NF, ARE PASSED TO THE REST OF THE PROGRAM
THROUGH THE COMMON STATEMENT. FREQUENCIES ARE AUTOMATICALLY
REORDERED FROM HIGHEST TO LOWEST. THE MAIN PROGRAM
ALSO CONTROLS THE CALLING OF THE OTHER SUBROUTINES.

COMMON NF,N,IFREQ,NGCF,IDIV,IODEV,NSOL,NCH,MGCF,LCM,IVAL
,ITOTL,
1 NMAX(2000)
COMMON/TIME/ MTOT,MPROB,MSOLN
COMMON/NEW/ NS1,NS2
DIMENSION N(18),IFREQ(18),IDIV(2000),IVAL(2000),MGCF(18,
18)
DOUBLE PRECISION FILE,FILOUT
FROM HERE TO LINE 65 THE DATA IS INPUT FROM A FILE. IF NO
FILE IS INDICATED THEN DATA IS INPUT FROM THE KEYBOARD.
C OUTPUT FILE IS ASKED FOR, AND AGAIN IF NO OUTPUT FILE IS
GIVEN THEN SOLUTIONS ARE OUTPUT TO THE TERMINAL. THE NUMBER
OF SOLUTIONS DESIRED IS ALSO ASKED FOR.
CALL JIFFY(MTOT,III)
IFLN=1
NCH=0
IODEV=5
TYPE 30
30 FORMAT (5X,'TYPE INPUT FILNAM.EXT> ',S)
READ (5,40) FILE
30 FORMAT (A10)
TYPE 41
41 FORMAT (5X,'TYPE OUTPUT FILNAM.EXT> ',S)
READ (5,40) FILOUT
IF (FILOUT.EQ.' ') GO TO 42
IODEV=11
OPEN (UNIT=11,DEVICE='DSK',ACCESS='SEQOUT',FILE=FILOUT)
42 CONTINUE
TYPE 45
45 FORMAT (' # SOLUTIONS DESIRED? > ',S)
READ (5,60) NS1,NS2
IF (NS2.NE.0) GO TO 46
NS2=NS1
NS1=1
IF (NS2.EQ.0) NS2=1
46 CONTINUE
IF (FILE.EQ.' ') GO TO 100
OPEN (UNIT=10,DEVICE='DSK',ACCESS='SEQIN',FILE=FILE)
K=10
C THIS LOOP (TO LINE 65) DOES THE ACTUAL DATA INPUT.
NF=1
50 READ (K,60,END=65) N(NF),IFREQ(NF)
IF (N(NF).LE.0) GO TO 65
NCH=NCH+N(NF)
53 IF (NF.EQ.1) GO TO 62

```

```

06100      DO 55 I1=1,NF-1
06200      IF (IFREQ(NF).NE.IFREQ(I1)) GO TO 55
06300      N(I1)=N(I1)+N(NF)
06400      GO TO 50
06500  55    CONTINUE
06600  60    FORMAT (2I)
06700  62    NF=NF+1
06800      IF (NF.LE.18) GO TO 50
06900  65    CALL JIFFY(MPROB,III)
07000      MSOLN=MPROB
07100  C    THIS LOOP TO LINE 68 WILL USE A BUBBLE SORT TO ORDER THE
07200  C    FREQUENCIES FROM HIGHEST TO LOWEST FOR FASTER EXECUTION.
07300      DO 68 I1=1,NF-2
07400      DO 68 I2=I1+1,NF-1
07500      IF (IFREQ(I1).GT.IFREQ(I2)) GO TO 68
07600      I3=IFREQ(I1)
07700      IFREQ(I1)=IFREQ(I2)
07800      IFREQ(I2)=I3
07900      I3=N(I1)
08000      N(I1)=N(I2)
08100      N(I2)=I3
08200  68    CONTINUE
08300  C    FILE NUMBER AND NUMBER OF FREQUENCIES INPUT ARE TYPED ON
08400  C    THE TTY AND THE PROGRAM JUMPS TO LINE 200 TO BEGIN SOLVING.
08500  70    NF=NF-1
08600      NSOL=0
08700      IF (IODEV.EQ.11) TYPE 78,IFLN
08800  78    FORMAT (2X,'FILE',I5)
08900      WRITE (IODEV,79) IFLN
09000  79    FORMAT (/, ' FILE',I5)
09100      WRITE (IODEV,80) NF,NCH
09200  80    FORMAT(/, ' INPUT: ',I6, ' FREQUENCIES, ',I8, ' CHANNELS')
09300      I2=0
09400  85    I1=I2+1
09500      I2=I2+7
09600      IF (I2.GT.NF) I2=NF
09700      WRITE (IODEV,90) (N(I),I=I1,I2)
09800  90    FORMAT (/, ' NUMBER:',3X,I8)
09900      WRITE (IODEV,95) (IFREQ(I),I=I1,I2)
10000  95    FORMAT (' FREQUENCY:',7I8)
10100      IF (I2.LT.NF) GO TO 85
10200      GO TO 200
10300  C    NEXT 5 LINES SET UP PARAMETERS FOR KEYBOARD INPUT.
10400  100   K=5
10500      NF=1
10600      TYPE 110
10700  110   FORMAT (5X,'TYPE DATA IN AS NUMBER FREQUENCY IN 2I FORMA
T')
10800      GO TO 50
10900  C    SETUP IS CALLED TO INITIALIZE PARAMETERS AND ARRAYS.
11000  200   CALL SETUP
11100  C    SOLVE IS CALLED TO OBTAIN AND PRINT OUT DESIRED SOLUTIONS.
11200  201   CALL SOLVE
11300      IF (NSOL.GE.NS2) GO TO 210
11400      ITOTL=ITOTL+LCM
11500      I1=1
11600      L=ITOTL/IFREQ(1)
11700      J=N(1)
11800      DO 205 I=1,NCH
11900      J=J-1
12000      IF (J.GE.0) GO TO 204

```

```

12100      I1=I1+1
12200      L=ITOTL/IFREQ(I1)
12300      J=N(I1)-1
12400  204    IDIV(I)=L
12500  205    NMAX(I)=L-J
12600      DO 207 I=1,NF
12700      II=ITOTL/IFREQ(I)
12800      DO 207 J=1,NF
12900      JJ=ITOTL/IFREQ(J)
13000  207    MGCF(I,J)=KGCF(II,JJ)
13100      GO TO 201
13200  C STOP IF DATA READ FROM KEYBOARD.
13300  210    IF (K.EQ.5) STOP
13400  C IF DATA IS READ FROM A FILE THEN SEE IF THERE IS MORE
13500  C TO READ AND IF SO THEN BEGIN READING IT AND JUMP TO 53.
13600  C IF END OF FILE THEN CLOSE THE FILE AND STOP.
13700      NF=1
13800      NCH=0
13900  220    READ (K,60,END=300) N(NF),IFREQ(NF)
14000      IF (N(NF).LE.0) GO TO 220
14100      IFLN=IFLN+1
14200      GO TO 53
14300  300    CLOSE (UNIT=10,FILE=FILE)
14400      IF (IODEV.EQ.11) CLOSE (UNIT=11,FILE=FILOUT)
14500      STOP
14600      END
14700  C
14800  C
14900      SUBROUTINE SETUP
15000  C
15100      COMMON NF,N,IFREQ,NGCF,IDIV,IODEV,NSOL,NCH,MGCF,LCM,IVAL
,ITOTL,
15200      1 NMAX(2000)
15300      DIMENSION N(18),IFREQ(18),IDIV(2000),IVAL(2000),MGCF(18,
18)
15400  C FIND GCF OF FREQUENCIES.
15500      NGCF=IFREQ(1)
15600      DO 10 I=2,NF
15700  10      NGCF=KGCF(NGCF,IFREQ(I))
15800  C FIND SUM OF (N(I)*IFREQ(I)) AND LCM OF FREQUENCIES.
15900      ITOTL=0
16000      LCM=NGCF
16100      DO 20 I=1,NF
16200      LCM=LCM*IFREQ(I)/KGCF(LCM,IFREQ(I))
16300  20      ITOTL=ITOTL+N(I)*IFREQ(I)
16400      K=MOD(ITOTL,LCM)
16500      IF (K.NE.0) ITOTL=ITOTL+LCM-K
16600  30      K=1
16700      L=ITOTL/IFREQ(1)
16800      J=N(1)
16900      DO 50 I=1,NCH
17000      J=J-1
17100      IF (J.GE.0) GO TO 40
17200      K=K+1
17300      L=ITOTL/IFREQ(K)
17400      J=N(K)-1
17500  40      IVAL(I)=K
17600      NMAX(I)=L-J
17700  50      IDIV(I)=L
17800      DO 60 I=1,NF
17900      II=ITOTL/IFREQ(I)
18000      DO 60 J=1,NF

```

```

18100      JJ=ITOTL/IFREQ(J)
18200      MGCF(I,J)=KGCF(II,JJ)
18300      IF (MGCF(I,J).NE.1) GO TO 60
18400      ITOTL=ITOTL+LCM
18500      GO TO 30
18600  60    CONTINUE
18700      RETURN
18800      END
18900  C
19000  C
19100      FUNCTION KGCF(I,J)
19200  C
19300  C  THIS FUNCTION ROUTINE WILL COMPUTE THE GREATEST COMMON FACTOR
19400  C  OF TWO INTEGERS.  I AND J ARE NOT CHANGED.
19500  C
19600      K=I
19700      KGCF=J
19800  10    CONTINUE
19900      IF (K.EQ.0) RETURN
20000      L=KGCF-(KGCF/K)*K
20100      KGCF=K
20200      K=L
20300      GO TO 10
20400      END
20500  C
20600  C
20700      SUBROUTINE SOLVE
20800  C
20900      COMMON NF,N,IFREQ,NGCF,IDIV,IODEV,NSOL,NCH,MGCF,LCM,IVAL
,ITOTL,
21000      1 NMAX(2000)
21100      DIMENSION N(18),IFREQ(18),IDIV(2000),IVAL(2000),MGCF(18,
18)
21200      DIMENSION KST(2000)
21300      COMMON/TIME/ MTOT,MPROB,MSOLN
21400      COMMON/NEW/ NS1,NS2
21500      KST(1)=1
21600      NK=1
21700  10    NK=NK+1
21800      M=1
21900      IF (IDIV(NK).EQ.IDIV(NK-1)) M=KST(NK-1)
22000  20    M=M+1
22100  25    IF (M.GT.NMAX(NK)) GO TO 100
22200  30    DO 40 I=1,NK-1
22300      IF (M.EQ.KST(I)) GO TO 20
22400  40    CONTINUE
22500      KST(NK)=M
22600      N1=IVAL(NK)
22700      DO 50 I=1,NK-1
22800      N2=IVAL(I)
22900      KD=IABS(M-KST(I))
23000      KG=MGCF(N1,N2)
23100      IF (MOD(KD,KG).EQ.0) GO TO 110
23200  50    CONTINUE
23300      IF (NK.NE.NCH) GO TO 10
23400      IRS=MSOLN
23500      CALL JIFFY (MSOLN,III)
23600      NSOL=NSOL+1
23700      IF (NSOL.LT.NS1) GO TO 110
23800      WRITE (IODEV,55) NSOL
23900  55    FORMAT (//,3X,'SOLUTION NUMBER ',I5,/)
24000      N2=0

```

```

24100 60      N1=N2+1
24200      N2=N2+10
24300      IF (N2.GT.NCH) N2=NCH
24400      WRITE (IODEV,65) (KST(I),I=N1,N2)
24500 65      FORMAT (/,' FIRST LOC:',10I6)
24600      WRITE (IODEV,70) (IDIV(I),I=N1,N2)
24700 70      FORMAT (' REPT RATE:',10I6)
24800      IF (N2.LT.NCH) GO TO 60
24900      N1=ITOTL/NGCF
25000      WRITE (IODEV,75) N1,ITOTL
25100 75      FORMAT (/,' SCAN TABLE LENGTH',I10,15X,'CLOCK RATE',I10)
25200      CALL JIFFY(MSOLN,III)
25300      RS=FLOAT(MSOLN-IRS)/60.
25400      RP=FLOAT(MSOLN-MPROB)/60.
25500      RT=FLOAT(MSOLN-MTOT)/60.
25600      WRITE (IODEV,80) RT,RP,RS
25700 80      FORMAT (' CPU TIMES (TOTAL,PROBLEM,SOLUTION) ',3F10.2,/)
25800      IF (NSOL.LT.NS2) GO TO 110
25900      RETURN
26000 100     NK=NK-1
26100      IF (NK.EQ.1) RETURN
26200 110     M=KST(NK)+1
26300      GO TO 25
26400      END

```

APPENDIX II
SAMPLE RUNS OF ZAP2

The following is a session on the DEC-10 where ZAP2 is used to solve Problem 1 which was stored in a file.

.RU ZAP2

TYPE INPUT FILNAM.EXT> PROB1.DAT

TYPE OUTPUT FILNAM.EXT> PROB1.OUT

SOLUTIONS DESIRED? > 1

FILE 1
STOP

END OF EXECUTION
CPU TIME: 0.18 ELAPSED TIME: 15.42
EXIT

.TYPE PROB1.OUT

FILE 1

INPUT: 5 FREQUENCIES, 49 CHANNELS

NUMBER:	11	5	12	14	7
FREQUENCY:	300	200	60	50	10

SOLUTION NUMBER 1

1,	10,	0	
5,	2,	2	
1,	3,	3	
1,	10,	10	
1,	12,	12	
1,	2,	1	
1,	1,	5,	5
1,	3,	2	
1,	1,	4,	2
	1,	5,	5
	1,	5,	5

FREQUENCIES USED: 5 CLOCK RATE -- 6000

NUMBER:	11	5	15	14	10
FREQUENCY:	300	200	60	50	10

55 CHANNELS, 15 WHEELS, 99 MEMORY WORDS
1 WHEEL, 602 WORDS -- 16.4% (RATIO)

CONDENSED SCAN TABLE				SCAN TABLE LENGTH --				600		
CHANNEL ID:	1	2	3	4	5	6	7	8	9	10
FREQUENCY:	10	10	10	10	10	10	10	50	50	50
FIRST LOC:	90	210	330	450	570	120	240	8	18	28
REP RATE:	600	600	600	600	600	600	600	120	120	120
CHANNEL ID:	11	12	13	14	15	16	17	18	19	20
FREQUENCY:	50	50	50	50	50	50	50	50	50	50
FIRST LOC:	38	48	58	68	78	88	98	108	118	30
REP RATE:	120	120	120	120	120	120	120	120	120	120
CHANNEL ID:	21	22	23	24	25	26	27	28	29	30
FREQUENCY:	50	60	60	60	60	60	60	60	60	60
FIRST LOC:	60	7	17	27	37	47	57	67	77	87
REP RATE:	120	100	100	100	100	100	100	100	100	100
CHANNEL ID:	31	32	33	34	35	36	37	38	39	40
FREQUENCY:	60	60	60	200	200	200	200	200	300	300
FIRST LOC:	97	19	39	6	16	26	10	20	1	11
REP RATE:	100	100	100	30	30	30	30	30	20	20
CHANNEL ID:	41	42	43	44	45	46	47	48	49	
FREQUENCY:	300	300	300	300	300	300	300	300	300	
FIRST LOC:	2	12	3	13	4	14	5	15	9	
REP RATE:	20	20	20	20	20	20	20	20	20	
CPU TIMES (TOTAL, PROBLEM, SOLUTION)					0.18	0.10	0.10			

The following is a session with the DEC-10 where ZAP2 is used to solve Problem 2. This problem is typed in from the terminal.

.RU ZAP2

TYPE INPUT FILNAM.EXT>

TYPE OUTPUT FILNAM.EXT>

SOLUTIONS DESIRED? > 1

TYPE DATA IN AS CHANNEL ID, FREQUENCY IN 2I FORMAT

1	50
2	50
3	50
4	50
5	50
6	50
7	50
8	50
9	50
10	50
11	50
12	50
13	50
14	50
15	50
16	50
17	50
18	50
19	50
20	50
21	50
22	50
23	50
24	50
25	50
26	50
27	50
28	50
29	50
30	50
31	50
32	50
33	50
34	50
35	50
36	50
37	15
38	15
39	15
40	15
41	15
42	15
43	20
44	20
45	20

46 20
47 30
48 30
49 30

FILE 1

INPUT: 4 FREQUENCIES, 49 CHANNELS

NUMBER: 36 3 4 6
FREQUENCY: 50 30 20 15

SOLUTION NUMBER 1

1, 7, 0
6, 6, 6
1, 5, 0
1, 2, 2
1, 3, 3
1, 4, 4
1, 2, 1
1, 1, 2, 2
1, 3, 3

FREQUENCIES USED: 4 CLOCK RATE -- 2100

NUMBER: 36 3 6 6
FREQUENCY: 50 30 20 15

51 CHANNELS, 14 WHEELS, 92 MEMORY WORDS
1 WHEEL, 422 WORDS -- 21.8% (RATIO)

CONDENSED SCAN TABLE SCAN TABLE LENGTH -- 420

CHANNEL ID:	1	2	3	4	5	6	7	8	9	10
FREQUENCY:	50	50	50	50	50	50	50	50	50	50
FIRST LOC:	1	8	15	22	29	36	2	9	16	23
REP RATE:	42	42	42	42	42	42	42	42	42	42
CHANNEL ID:	11	12	13	14	15	16	17	18	19	20
FREQUENCY:	50	50	50	50	50	50	50	50	50	50
FIRST LOC:	30	37	3	10	17	24	31	38	4	11
REP RATE:	42	42	42	42	42	42	42	42	42	42
CHANNEL ID:	21	22	23	24	25	26	27	28	29	30
FREQUENCY:	50	50	50	50	50	50	50	50	50	50
FIRST LOC:	18	25	32	39	5	12	19	26	33	40
REP RATE:	42	42	42	42	42	42	42	42	42	42
CHANNEL ID:	31	32	33	34	35	36	37	38	39	40
FREQUENCY:	50	50	50	50	50	50	15	15	15	15
FIRST LOC:	6	13	20	27	34	41	21	56	91	126
REP RATE:	42	42	42	42	42	42	140	140	140	140

CHANNEL ID:	41	42	43	44	45	46	47	48	49
FREQUENCY:	15	15	20	20	20	20	30	30	30
FIRST LOC:	63	133	14	49	84	35	7	42	28
REP RATE:	140	140	105	105	105	105	70	70	70

CPU TIMES (TOTAL,PROBLEM,SOLUTION)	0.22	0.13	0.13
------------------------------------	------	------	------

STOP

END OF EXECUTION

CPU TIME: 0.25 ELAPSED TIME: 3:12.35

EXIT

Here the routine TABLE is used to produce a scan table for
 Problem 2. FOR22.DAT was produced by ZAP2 when solving Problem 2.

.RU TABLE

TYPE INPUT FILNAM.EXT> FOR22.DAT

TYPE OUTPUT FILNAM.EXT> PROB2.SCN

STOP

END OF EXECUTION

CPU TIME: 0.21 ELAPSED TIME: 42.95

EXIT

.TYPE PROB2.SCN

1	7	13	19	25	31	47	2	8	14
20	26	32	43	3	9	15	21	27	33
37	4	10	16	22	28	34	49	5	11
17	23	29	35	46	6	12	18	24	30
36	48	1	7	13	19	25	31	44	2
8	14	20	26	32	38	3	9	15	21
27	33	41	4	10	16	22	28	34	0
5	11	17	23	29	35	47	6	12	18
24	30	36	45	1	7	13	19	25	31
39	2	8	14	20	26	32	49	3	9
15	21	27	33	0	4	10	16	22	28
34	48	5	11	17	23	29	35	43	6
12	18	24	30	36	40	1	7	13	19
25	31	42	2	8	14	20	26	32	46
3	9	15	21	27	33	47	4	10	16
22	28	34	44	5	11	17	23	29	35
37	6	12	18	24	30	36	49	1	7
13	19	25	31	0	2	8	14	20	26
32	48	3	9	15	21	27	33	45	4
10	16	22	28	34	38	5	11	17	23
29	35	41	6	12	18	24	30	36	0
1	7	13	19	25	31	47	2	8	14
20	26	32	43	3	9	15	21	27	33
39	4	10	16	22	28	34	49	5	11
17	23	29	35	46	6	12	18	24	30
36	48	1	7	13	19	25	31	44	2
8	14	20	26	32	40	3	9	15	21
27	33	42	4	10	16	22	28	34	0
5	11	17	23	29	35	47	6	12	18
24	30	36	45	1	7	13	19	25	31
37	2	8	14	20	26	32	49	3	9
15	21	27	33	0	4	10	16	22	28
34	48	5	11	17	23	29	35	43	6
12	18	24	30	36	38	1	7	13	19
25	31	41	2	8	14	20	26	32	46
3	9	15	21	27	33	47	4	10	16
22	28	34	44	5	11	17	23	29	35
39	6	12	18	24	30	36	49	1	7
13	19	25	31	0	2	8	14	20	26
32	48	3	9	15	21	27	33	45	4
10	16	22	28	34	40	5	11	17	23
29	35	42	6	12	18	24	30	36	0

The following is a session on the DEC-10 where ZAP2 is used to solve Example 1 and TABLE is used to produce a scan table.

.RU ZAP2

TYPE INPUT FILNAM.EXT>

TYPE OUTPUT FILNAM.EXT>

SOLUTIONS DESIRED? > 1

TYPE DATA IN AS CHANNEL ID, FREQUENCY IN 2I FORMAT

1 30
2 15
3 15
4 15
5 10
6 10
7 10
8 10
9 5
10 5
11 5
12 5

FILE 1

INPUT: 4 FREQUENCIES, 12 CHANNELS

NUMBER:	1	3	4	4
FREQUENCY:	30	15	10	5

SOLUTION NUMBER 1

1,	5,	1	
1,	2,	2	
1,	3,	3	
1,	2,	1	
1,	1,	3,	3
1,	3,	2	
1,	1,	2,	2

FREQUENCIES USED: 4 CLOCK RATE -- 150

NUMBER:	1	3	5	5
FREQUENCY:	30	15	10	5

14 CHANNELS, 7 WHEELS, 34 MEMORY WORDS
1 WHEEL, 32 WORDS -- 106.3% (RATIO)

CONDENSED SCAN TABLE SCAN TABLE LENGTH -- 30

CHANNEL ID:	1	2	3	4	5	6	7	8	9	10
FREQUENCY:	30	15	15	15	10	10	10	10	5	5
FIRST LOC:	1	2	7	4	3	8	13	5	9	19
REP RATE:	5	10	10	10	15	15	15	15	30	30

CHANNEL ID: 11 12
 FREQUENCY: 5 5
 FIRST LOC: 29 15
 REP RATE: 30 30

CPU TIMES (TOTAL, PROBLEM, SOLUTION) 0.22 0.15 0.15

STOP

END OF EXECUTION

CPU TIME: 0.25 ELAPSED TIME: 1:53.92

EXIT

.RU TABLE

TYPE INPUT FILNAM.EXT> FOR22.DAT

TYPE OUTPUT FILNAM.EXT> EXAMP1.TBL

STOP

END OF EXECUTION

CPU TIME: 0.10 ELAPSED TIME: 1:7.50

EXIT

.TY EXAMP1.TBL

1	2	5	4	8	1	3	6	9	0
1	2	7	4	12	1	3	5	10	8
1	2	6	4	0	1	3	7	11	0

The following is a session on the DEC-10 where ZAP2 is used to solve Example 2 and TABLE is used to produce a scan table.

.RU ZAP2

TYPE INPUT FILNAM.EXT>

TYPE OUTPUT FILNAM.EXT>

SOLUTIONS DESIRED? > 1

TYPE DATA IN AS CHANNEL ID, FREQUENCY IN 2I FORMAT
1 3
2 2

FILE 1

INPUT: 2 FREQUENCIES, 2 CHANNELS

NUMBER: 1 1
FREQUENCY: 3 2

SOLUTION NUMBER 1

1, 2, 0
1, 2, 2
1, 3, 3

FREQUENCIES USED: 2 CLOCK RATE -- 12

NUMBER: 2 3
FREQUENCY: 3 2

5 CHANNELS, 3 WHEELS, 13 MEMORY WORDS
1 WHEEL, 14 WORDS -- 92.9% (RATIO)

CONDENSED SCAN TABLE SCAN TABLE LENGTH -- 12

CHANNEL ID: 1 2
FREQUENCY: 3 2
FIRST LOC: 1 2
REP RATE: 4 6

CPU TIMES (TOTAL, PROBLEM, SOLUTION) 0.13 0.08 0.08

STOP

END OF EXECUTION

CPU TIME: 0.16 ELAPSED TIME: 39.15

EXIT

.RU TABLE

TYPE INPUT FILNAM.EXT> FOR22.DAT

```
TYPE OUTPUT FILNAM.EXT> EXAMP3.TBL
```

STOP

END OF EXECUTION

CPU TIME: 0.11 ELAPSED TIME: 16.87

EXIT

.TYPE EXAMP3.TBL

1	2	0	0	1	0	0	2	1	0
0	0								

0 0

APPENDIX III

FILES USED IN TESTING THE ALGORITHMS

The following is a listing of the files contained in BIGFIL.

00051,500	00020,10	00028 10
00015,250	00013,25	00014 50
00009,200	00010,50	00012 60
00025,25	00005,60	00005 200
00149,50		00011 300
00813,20		
00483,10	00020,10	00007 330
00019,1000	00020,25	00028 110
	00010,50	00002 385
	00005,60	00011 70
		00001 2310
00001,500		
00003,250		
00004,200	00006,10	00007 10
00025,25	00004,15	00014 50
00009,50	00015,20	00012 60
00013,20	00005,30	00005 200
00083,10	00003,60	00011 300
	00002,75	
		00006 330
00002,500		00028 110
00005,250	00002,5	00002 385
00005,200	00002,6	00011 70
00001,25		00001 2310
00001,50		
00003,20	00006,5	00004,50
00083,10	00005,6	00016,10
		00020,15
		00012,20
00020,100	00007,10	
00050,10	00014,50	
00035,50	00012,60	00028,10
00020,25	00005,200	00014,50
00100,20	00011,300	00012,60
		00005,200
		00011,300
00020,100	00012,60	
00055,10	00009,50	
00035,50		00020,10
00020,25		00020,25
00100,20	00011,60	00010,50
	00010,50	00005,60
00020,100		
00050,10	00015,60	00500,10
00036,50	00012,50	00400,60
00020,25		00004,1000
00100,20		
	00004 10	
	00006 15	00020,10000
00050,10	00004 20	00015,5000
00035,50	00003 30	00021,1000
00020,25		00025,500
00100,20	00004 50	00100,25
	00016 10	00500,10
	00020 15	
	00012 20	

00020,100	00008,10	00001,30000
00035,50	00002,50	00005,10000
00050,10	00002,60	00015,5000
00020,25	00001,300	00025,2500
00100,20		00050,1000
	00015,20	00050,100
	00016,25	00019,50
00015,5	00001,700	00800,20
00020,10		00025,10
00004,20	00020 1000	00325,5
00003,25	00052 500	
00002,35	00016 250	
	00010 200	00140,5
	00150 50	00028,25
00035,50	00026 25	00035,20
00019,25	00813 20	00070,10
00099,20	00483 10	00020,35
00049,10		
00005,10	00007,10000	
	00015,5000	00195,5
	00025,2500	00020,10
00036,50	00050,1000	00004,20
00006,15	00050,100	00003,25
00004,20	00019,50	00002,35
00003,30	00800,20	
	00100,60	
	00025,10	00021,5
00160,5	00325,5	00030,10
00028,25		00005,20
00035,20		00012,25
00060,10	00009,10000	00017,35
00020,35	00015,5000	
	00025,2500	
00160,5	00050,1000	00500,10
00028,25	00050,100	00400,60
00070,20	00019,50	00004,1000
00075,10	00800,20	
	00025,10	
00083,10	00325,5	00500,10
00003,20		00004,1000
00001,25		00400,60
00001,50	00008,10000	
00005,200	00015,5000	
00005,250	00025,2500	00250,20
00002,500	00050,1000	00100,10
	00050,100	00035,50
00015 5	00019,50	00050,25
00083,10	00800,20	00020,100
00003,20	00025,10	
00001,25	00325,5	
00001,50		
00005,200		
00005,250		
00002,500		

00035,50	00021,5	00033 250
00050,25	00020,10	00012 138
00100,10	00005,20	00080 5
00020,100	00016,25	00019 16
00250,20	00017,35	
	00015,5	
00060,10	00028,10	
00035,50	00007,20	
00020,25	00025,25	
00100,20	00008,35	
00300,100		
	00016,5	
00006,10	00020,10	
00004,15	00021,20	
00015,20	00007,25	
00005,30	00015,35	
00003,60		
00002,75	00015,5	
	00025,25	
00020,1000	00028,10	
00052,500	00007,20	
	00008,35	
00016,250	00035,50	
00010,200	00019,25	
00150,50	00099,20	
00016,25	00049,10	
00813,20	00021,5	
00483,10		
	00075,1000	
00020,10	00001,500	
00013,25	00003,250	
00010,50	00004,200	
00005,60	00009,50	
	00025,25	
00152,5	00013,20	
00083,10	00083,10	
00013,20	00157,5	
00026,25		
00010,50	00002 1000	
00079,1000	00006 100	
	00005 50	
00001,9600	00013 35	
00006,2400	00010 25	
00003,600	00040 20	
00004,150	01000 10	
00005,400		
00003,100	00002 1000	
00004,25	00006 100	
	00005 50	
	00010 25	
	00040 20	
	01000 10	

The following is a listing of miscellaneous files used in testing the algorithms.

33 250	1,50	1 50
12 138	10,3	1 30
80 5	5,20	3 20
19 16	100,4	2 25
	20,1	5 6
	2,300	6 10
		4 5
1,50	11,1000	
10,3	9,600	1 5
11,20	15,300	1 3
2,300	21,100	8 2
	77,50	
1,50	23,60	1 5
10,3	111,20	2 3
6,20	123,10	6 2
25,4		
2,300		
1,50		1 5
10,3	11,1000	3 3
5,20	9,600	4 2
25,4	17,300	
20,1	28,100	1 5
2,300	77,50	4 3
	23,60	2 2
	111,20	
	123,10	2 5
		1 3
		4 2
	51,50000	
	15,25000	2 5
	09,20000	2 3
	25,2500	3 2
	149,5000	
	813,2000	2 5
	483,1000	3 3
	19,100000	2 2
		2 5
	2,2	4 3
	3,3	1 2
	5,5	
	7,7	1 7
	11,11	1 3
		12 2
	2,5	1 7
	1,10	2 3
	1,25	10 2
	1 10	1 7
	1 6	3 3
	4 3	8 2
	3 4	
	10 2	

1 7
4 3
6 2

1 7
5 3
4 2

1 7
6 3
2 2

2 7
1 3
6 2

2 7
2 3
5 2

2 7
3 3
4 2

2 7
4 3
3 2

2 7
5 3
2 2

2 7
6 3
1 2

1 7
1 5
24 2

1 7
2 5
20 2

1 7
3 5
16 2

1 7
4 5
12 2

1 7
5 5
8 2

1 7
6 5
4 2

2 7
1 5
18 2

2 7
2 5
15 2

2 7
3 5
12 2

2 7
4 5
9 2

2 7
5 5
6 2

2 7
6 5
3 2

3 7
1 5
12 2

3 7
2 5
10 2

3 7
3 5
8 2

3 7
4 5
6 2

3 7
5 5
4 2

3 7
6 5
2 2

4 7
1 5
6 2

4 7
2 5
5 2

4 7
3 5
4 2

4 7
4 5
3 2

4 7
5 5
2 2

4 7
6 5
1 2

1 7
2 5
24 3

1 7
4 5
20 3

1 7
6 5
16 3

1 7
8 5
12 3

1 7
10 5
8 3

1 7
12 5
4 3

2 7
2 5
18 3

2 7
4 5
15 3

2 7
6 5
12 3

2 7
8 5
9 3

2 7
10 5
6 3

2 7
12 5
3 3

3 7
2 5
12 3

3 7
4 5
10 3

3 7
6 5
8 3

3 7
8 5
6 3

3 7
10 5
4 3

3 7
12 5
2 3

4 7
2 5
6 3

4 7
4 5
5 3

4 7
6 5
4 3

4 7
8 5
3 3

4 7
10 5
2 3

4 7
12 5
1 3

2 7
1 5
24 3

2 7
2 5
20 3

2 7
3 5
16 3

2 7
4 5
12 3

2 7
5 5
8 3

2 7
6 5
4 3

4 7
1 5
18 3

4 7
2 5
15 3

4 7
3 5
12 3

4 7
4 5
9 3

4 7
5 5
6 3

4 7
6 5
3 3

6 7
1 5
12 3

6 7
2 5
10 3

6 7
3 5
8 3

6 7
4 5
6 3

6 7
5 5
4 3

6 7
6 5
2 3

8 7
1 5
6 3

8 7
2 5
5 3

8 7
3 5
4 3

8 7
4 5
3 3

8 7
5 5
2 3

8 7
6 5
1 3

2 5
3 3
2 2

1 100
 100 1
 3 60
 2 40

1 100
 100 1
 30 2
 20 3
 1 60
 1 40
 8 5

30 20
 30 10
 5 50
 10 25
 5 100

2 6
 1 4
 2 3
 7 2

7 12
 9 9
 10 8

7 12
 9 9
 10 8
 1 4

VITA

Richard Odell Eason was born in Anniston, Alabama on July 18, 1956. He attended Woodland Elementary School and Jefferson Junior High School in Oak Ridge, Tennessee, before graduating from Oak Ridge High School in June 1974. The following September he entered The University of Tennessee, Knoxville. In 1975, he was recipient of the Allen medal given each year to the outstanding freshman mathematics student at the University, and in 1976 and 1977, he was a Fred M. Roddy Scholarship recipient. In March, 1978, he graduated with highest honors with a Bachelor of Science degree in Electrical Engineering.

After accepting a research assistantship and teaching assistantship at The University of Tennessee, Knoxville, he began his study toward a Master of Engineering degree in Electrical Engineering, which was awarded in June 1980.

The author is a member of Phi Kappa Phi, Tau Beta Pi, Eta Kappa Nu and IEEE.