IN-35
40109
p-25

NASA
Technical
Memorandum

NASA TM-103552

MULTIPLEXING READOUT CHANNELS IN PROPORTIONAL COUNTERS

By James Caristi

Space Science Laboratory
Science and Engineering Directorate

August 1991

(NASA-TM-103552) MULTIPLEXING READOUT CHANNELS IN PROPORTIONAL COUNTERS (NASA) 25 p CSCL 14B

N91-31606

Unclas
G3/35 0040109

National Aeronautics and
Space Administration

George C. Marshall Space Flight Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Technical Memorandum

4. TITLE AND SUBTITLE

Multiplexing Readout Channels in Proportional Counters

5. FUNDING NUMBERS

6. AUTHOR(S)

James Caristi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

George C. Marshall Space Flight Center
Marshall Space Flight Center, AL 35812

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-103552

11. SUPPLEMENTARY NOTES

Prepared for Space Science Laboratory, Science and Engineering Directorate while author was in residence at MSFC during the summer 1991 NASA/University Joint Venture (JOVE) Program on leave from the Department of Mathematics and Computer Science, Valparaiso University, Valparaiso, IN 46383 (B. Ramsey/ES65/Science Advisor)

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified--Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Proportional counters are important instruments used in sensing "hard" x rays. This document describes the possibility of doubling the number of readout channels in the detector without increasing the electronics needed to amplify channel signals. This suggests that it should be possible, conversely, to reduce the number of amplifiers, thereby reducing the weight and energy budget of the instrument. Various numerical multiplexing schemes are analyzed, and a computer program is presented that can reconstruct multiplexed channel outputs with very good accuracy.

14. SUBJECT TERMS

Hard X-Ray Measurement and Imaging, Multiwire Proportional Counters

15. NUMBER OF PAGES

33

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

TABLE OF CONTENTS

TECHNICAL MEMORANDUM

MULTIPLEXING READOUT CHANNELS IN PROPORTIONAL COUNTERS

PROBLEM STATEMENT

Proportional counters for x-ray astronomy under development at MSFC can use improved cathode planes constructed through a photolithographic process. Consequently, there is the opportunity to double the number of readout channels without incurring significant cost. The signal from each channel must still be amplified before it can be processed. The problem is that there is no convenient way to make a corresponding increase in the number of amplifiers used in the instrument. This report describes and illustrates an idea for multiplexing pairs of channels into amplifiers, which allows the determination, with high degree of confidence, of the channel numbers and amplitudes that caused the signal. The instrument under consideration will have 140 readout channels and 70 amplifiers in each of the x and y directions.

DIFFICULTIES WITH MULTIPLEXING SCHEMES

No multiplexing scheme can be completely successful. For example, suppose that an x-ray event interacting with the gas in the detector causes two distinct clusters of channels to produce signals, where one cluster consists of four adjacent channels (labeled using X's below) and the other consists of two adjacent channels (labeled using Y). Each wire must be multiplexed with another wire in the array. The following situation can therefore occur. One of the wires in the four-channel cluster is multiplexed as shown:

```
XXXX          A
|             |
|_____________|
```

The channel after A, B, must be multiplexed somewhere. If it is multiplexed with one of the wires from the two-channel cluster,

```
XXXX          AB        Y
               |        |
               |________|
```

it may be impossible to determine whether a photoelectron caused a two channel cluster AB or the legitimate two-channel cluster containing Y. Any multiplexing scheme is subject to this problem involving the creation of spurious two-channel clusters.

Besides the occurrence of spurious two-channel clusters, multiplexing schemes are subject to two additional problems. A legitimate cluster can be enlarged when a channel adjacent to an endpoint of the cluster is multiplexed with a channel that is part of another cluster. Also, the amplitude of a channel that is part of a cluster can be increased if the channel is multiplexed with a channel that is part of another cluster. Both problems lead to error in the determination of the position of the event. In summary, any multiplexing scheme must face three problems:

1. Spurious two-channel clusters
2. Cluster enlargement
3. Amplitude increase

ASSUMPTIONS AND REQUIREMENTS

Before describing the proposed multiplexing scheme, assumptions and requirements are given, as well as the general procedure for "decoding" the amplifier outputs. It is assumed that a legitimate x-ray event induces signals in two clusters of channels, where each cluster has width between two and five (although some clusters may be larger than five channels). We assume that if there were a continuum of channels, a cluster would appear as a Gaussian curve. In the detector within a single time slice (microseconds), one, two, or (rarely) three clusters of channels may produce signals. It is important to identify those situations where two clusters are activated.

The general idea in reconstructing the actual channel clusters and amplitudes from the amplifier outputs involves first determining all the possible channels that might have been activated. Any channels that are "isolated" can be eliminated from further consideration ("isolated" means that neither adjacent channel was possibly activated by the event). For this idea to work, it is necessary that adjacent channels NOT be multiplexed with adjacent channels. Furthermore, channels that are multiplexed together must be separated by at least five channels. Otherwise, even a single cluster cannot be unambiguously reconstructed, as the following example shows:

AXXXXB

If A and B are both possible active channels that are multiplexed together, it is impossible to determine whether a five-channel cluster starts at A or ends at B.

NUMERICALLY BASED MULTIPLEXING SCHEMES

There are many approaches to obtaining a multiplexing scheme that will be good enough to allow the reconstruction of at least single clusters. The simplest idea is to multiplex channels in the first half of the array with channels in the second half of the array according to some numerical pattern. In this way, it is easy to guarantee separation of multiplexed pairs. With 140 channels in the array numbered from 0 to 139, we can consider the following scheme:

Given channel I for I between 0 and 69, multiplex channel I
with channel k*I (mod 70) + 70.

If k is chosen to be any integer relatively prime to 70, then the set of multiples of k (mod 70) will cover all of the integers between 0 and 69 (since k is invertible in the integers mod 70 if and only if k is relatively prime to 70). Note also that in this multiplexing scheme with k relatively prime to 70, given channel I for I between 70 and 139, the channel multiplexed with I is given by I*(1/k) (mod 70). For example, if k = 9, 1/k = 39 (mod 70), so the channel multiplexed with channel 88 is 88*39 = 2 (mod 70). Good choices for k are discussed below.

The quality of the choice of k can be assessed using two measures. The first measure is the maximum number of consecutive channels in half of the array, which, if activated, can still be unambiguously reconstructed. This maximum unambiguously differentiable length (MUDL) is the number of "consecutive multiples" of k or 1/k (mod 70) that can be written in the integers mod 70 without obtaining any consecutive integers. For example, if k = 9, the progression of multiples

0,9,18,27,36,45,54,63,2,11,20,...

does not possess any consecutive integers until the 39th multiple, 1, is obtained. Notice, however, that for the multiples of 1/9 = 39 (mod 70), the progression proceeds

0,39,8,47,16,55,24,63,32,1,...

Thus for k = 9, in channels 0 through 69 MUDL is 39, while in channels 70 through 139 MUDL is 9. The MUDL is not simply k or 1/k (mod 70). For example, if k = 41, then k = 1/k (mod 70) and it turns out that MUDL is 29.

The other measure of quality for k involves determining the maximum number of consecutive channels spanning the center of the array, which, if activated, can still be unambiguously reconstructed. This central unambiguously differentiable length (CUDL) is simply

min {b-a: (a,b) is a multiplexed pair and a < b}.

For example, when k = 9 CUDL is 14 because (63,77) is a multiplexed pair. For k = 41, CUDL is 10 since 65 and 75 are multiplexed together. Use of the value computed for CUDL should perhaps be tempered by the determination of how many multiplexed pairs have the same CUDL and the next highest minimum separation. For example, though CUDL is 10 for k = 41, only one multiplexed pair is separated by 10. The next highest separation is 20. In contrast, for k = 29, CUDL is 14 with three pairs (58,72), (63,77), and (68,82) having the same separation, and the next highest separation is 28.

In the schemes with k chosen to be 9 or 29 or 41, it is easily verified that any single cluster of width less than 9 can be unambiguously reconstructed from amplifier outputs. The best choice of k to deal with pairs of clusters depends on characteristics of the detector's operation. For example, if most clouds land near the middle of the detector, it would be best to use k = 9 and arrange to have channels 0 through 69 centered around the middle of the detector (since MUDL in that region is 39). If instead, clouds land in all portions of the detector with equal likelihood, it would be better to choose k to be either 29 or 41, because in each case k = 1/k (mod 70) and MUDL is 29. Then depending on the expected separation of clouds, a choice could be made based on CUDL.

TECHNIQUES FOR RECONSTRUCTING ACTUAL CHANNELS

With any of the numerically based methods described above, the reconstruction of actual channels and amplitudes involved in the event requires more than eliminating isolated channels. If we assume that triple events involving two-channel clusters are rare, it is possible to eliminate all occurrences of spurious two-channel clusters. The problem of cluster enlargement can be dealt with using a kind of smoothing.

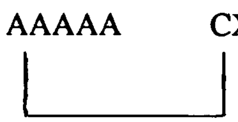
ELIMINATING SPURIOUS TWO-CHANNEL CLUSTERS

The program called RECON.FOR in the appendix contains two subroutines used to eliminate spurious two-channel clusters. ELIM2A is invoked when a two-channel cluster is detected with two larger clusters. If both channels in the two-channel cluster are found to be multiplexed with channels that are part of the larger clusters, the two-channel cluster is eliminated. The reasoning is that triple events are rare to begin with, and the probability is remote that a triple event would occur with a two-channel cluster having both channels multiplexed into larger clusters. In fact, using actual data from a prior flight, we can obtain an empirical probability of 0.00518 for triple events. We can combine this with the probability that a channel will be multiplexed into one of two, four-channel clusters (8/138). Thus we obtain an estimate less than 0.00002 for the probability that a triple event has taken place and both channels of a two-channel cluster are multiplexed into the larger clusters. (This estimate should be reduced by the currently unknown conditional probability that one of the clusters of the triple event has width 2.)

The subroutine ELIM2B is invoked when there are at least three clusters, at least two having width 2, and at least one having larger width. To handle this situation, we have the following result:

If a channel in a two-channel cluster is multiplexed under a numerical scheme as described above with $k = 41$ or $k = 29$ or $k = 9$, and its multiplexed pair is part of a cluster of width 3, 4, or 5, then the multiplexed pair of its neighboring channel cannot be adjacent to any of the other channels of the larger cluster or their multiplexed pairs.

This situation is illustrated as follows:



The claim is that X cannot cause a spurious two-channel cluster with the multiplexed pairs of the larger cluster. So, if there are spurious two-channel clusters, then any legitimate two-channel cluster would be multiplexed into only spurious clusters. Thus when invoked, ELIM2B eliminates all two-channel clusters containing a channel that is multiplexed into a larger cluster.

The result described above can be proved by considering cases. One case is presented here; the others are similar. Suppose $C > 69$ is part of a two-channel cluster and is multiplexed (using $k = 41$) with A, a channel that is part of a five-channel cluster. Then $A = 41C \pmod{70}$, and the channels multiplexed with $C + 1$ and $C - 1$ are (mod 70) $41(C + 1)$ and $41(C - 1)$, respectively. Clearly, neither of these can be equal to $A + i$ or $A - i$ for $i = 1, 2, 3,$ or 4. Also, the channels multiplexed with $C + 1$ and $C - 1$ are less than 70, while channels multiplexed with $A + i$ and $A - i$ are greater than 70 (unless one cluster contains channel 70). Hence the multiplexed pairs of the channels in the two given clusters cannot be near each other to form a spurious two-channel cluster. (The case where one cluster contains channel 70 is verified by exhausting all the possibilities.)

ALLEVIATING CLUSTER ENLARGEMENT

To deal with the problem of cluster enlargement, subroutine ZAPEND identifies endpoints of clusters that have larger amplitudes than their neighbor channels. When a large amplitude endpoint is discovered, subroutine EXTRAPOLATE is called to determine what the expected endpoint value should be. This predicted value is computed by calculating the center of the cluster omitting the endpoint, looking at the amplitudes on the opposite end of the cluster, and using a linearly interpolated value. Exploiting the symmetry of the data in this way leads to a highly computationally efficient routine as well as excellent accuracy. The extrapolation subroutine was tested separately on a large set of clusters by comparing the actual cluster centers with the centers obtained using EXTRAPOLATE on those same clusters with an endpoint removed. The absolute difference between the predicted and actual centers of clusters had a mean of 4% of a channel width, and a standard deviation approximately equal to the mean. The maximum absolute difference was 46% of a channel width. A histogram showing the distribution of the absolute differences is given in the appendix. Over 99% of the predictions are within 20% of a channel width of the actual center. Further testing results are given below.

AMPLITUDE AMPLIFICATION

The problem of amplitude amplification was not directly addressed. If the amplitude of an endpoint of a cluster turns out to be increased sufficiently, subroutine ZAPEND will adjust it appropriately. Errors in endpoint amplitudes have the largest significance in the computation of the event position. An amplitude increase caused by multiplexing in a central channel of a cluster obviously has negligible influence on the computation of the center of the cluster. But errors in multiplexed channels that are neither central nor on the end of a cluster lead to some error in the computed position of the center. Future work may involve assessing the magnitude of this type of error empirically, and perhaps designing a broader form of Gaussian smoothing to reduce the error.

RESULTS

The multiplexing scheme using k = 41 was tested using data derived from a previous balloon flight and randomly generated data. Single events were always correctly reconstructed, so they are not included in the results below. In addition, the program also correctly reconstructed the event state (single, double, or triple event) in all cases.

The detector in the previous balloon flight used 70 readout channels. The data from that flight were adjusted slightly to make it appear as if they came from a detector with 140 channels. This was done by doubling the beginning channel number of each cluster and building new clusters with the same widths as the old ones, but starting at the doubled locations.

Testing was done by using existing MSFC analysis software to determine the centers of the events. Centers were first obtained for the actual data and placed in file CENTERSA. Then the actual data were supplied to a program that created file CODED consisting of the multiplexed amplifier outputs (file formats are given in the appendix). The reconstruction program, using only file CODED, built a file, PREDRAW, that contained the predicted channel numbers and their amplitudes. The analysis software was then run on file PREDRAW to obtain file CENTERSP. Finally summary statistics and a histogram were generated from the differences between CENTERSA and CENTERSP. One channel width on the 140 channel detector is 2 mm.

Using the 54 double events in the file from the previous balloon flight, 216 computations of cluster centers were made in each of CENTERSA and CENTERSP. The results are given below:

Mean absolute difference = 0.024 mm
Standard deviation = 0.126 mm
Maximum absolute difference = 1.28 mm
Number of overestimates = 12
Number of underestimates = 11
Number of exactly (*) correct estimates = 193

(*) to the limit of single precision arithmetic

The file of simulated data contained 934 double events scattered uniformly on the detector. This gave rise to 3736 computations of cluster centers in each of the files CENTERSA and CENTERSP. The results are as follows:

Mean absolute difference = 0.026 mm
Standard deviation = 0.103 mm
Maximum absolute difference = 1.28 mm
Number of overestimates = 179
Number of underestimates = 231
Number of exactly (*) correct estimates = 3326

A histogram shows these differences to be quite concentrated in the first 0.1 mm range:

Bin Range (mm)	Number of Observations
0 - 0.1	3456
0.1 - 0.2	82
0.2 - 0.3	91
0.3 - 0.4	30
0.4 - 0.5	24
0.5 - 0.6	19
0.6 - 0.7	21
0.7 - 0.8	4
0.8 - 0.9	0
0.9 - 1.0	5
1.0 -	4

Evidently multiplexing readout channels will not lead to significant error in the computation of x-ray event locations.

APPENDIX A

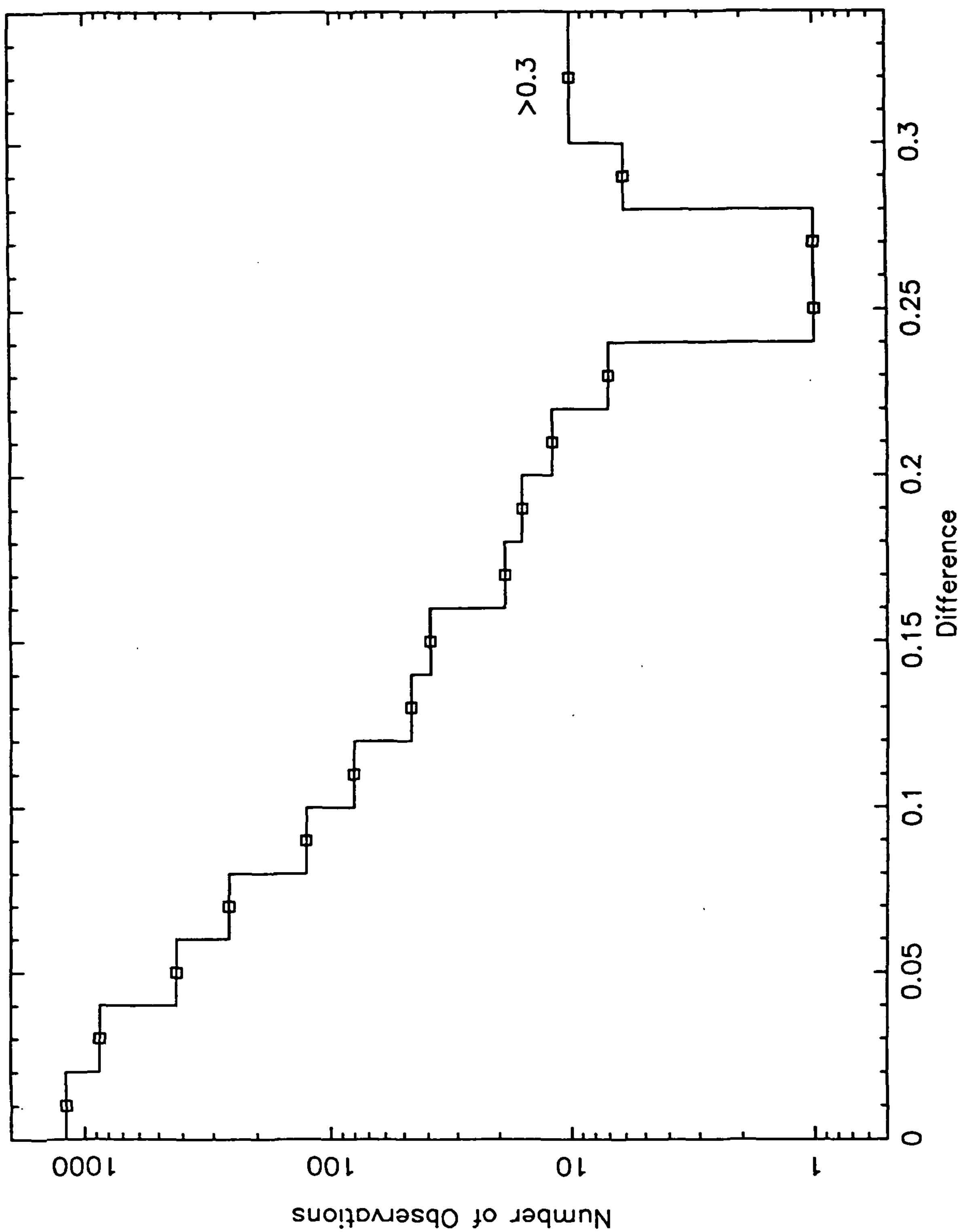


Figure A.1. Distribution of absolute differences between predicted and actual centers.

PRECEDING PAGE BLANK NOT FILMED

APPENDIX B

File Formats

RAW and PREDRAW

Each event is associated with five records:

(1) number of x channels number of y channels
(2) x channel numbers active (range 1-140)
(3) x amplitudes corresponding to active channels
(4) y channel numbers active (range 1-140)
(5) y amplitudes corresponding to active channels

CODED

Each event is associated with one record containing:

number of x amplifiers active
x amplifier numbers active
x amplitudes corresponding to active amplifiers
number of y amplifiers active
y amplifier numbers active
y amplitudes corresponding to active amplifiers

APPENDIX C

RECON.FOR

```
$DECLARE
      PROGRAM RECON
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER PX,PY,XOUT,YOUT,NX,NY,PSUBX,PSUBY,NXCLUMPS,NYCLUMPS
      INTEGER XSTART,YSTART,XWIDTH,YWIDTH,XSUM,YSUM,I,J,N
      DIMENSION PX(0:69),PY(0:69),XOUT(0:MAXCHAN-1),YOUT(0:MAXCHAN-1)
      DIMENSION PSUBX(50),PSUBY(50)
C
C  Reconstruct events in raw form from events passing through the
C  preamplifiers and coded in file CODED.  The program proceeds by
C  assigning the observed amplitude from a preamplifier to both of
C  the channels multiplexed to it.  Then all isolated channels are
C  eliminated.  An attempt is made to determine whether a single
C  event took place.  If this cannot be established, subroutine
C  FIND2 is invoked, which is structured like an expert system.
C  It contains rules which deal with clump enlargement and spurious
C  length 2 clumps.
C
      OPEN(32,FILE='CODED',STATUS='OLD')
      OPEN(33,FILE='PREDRAW',STATUS='UNKNOWN')
C  Obtain preamplifier levels for each record in CODED file
C  Beginning of main loop
C  First initialize all preamps to 0 since not all are in CODED
 10   DO 20 I=0,69
        PX(I)=0
        PY(I)=0
 20   CONTINUE
      DO 30 I=0,MAXCHAN-1
        XOUT(I)=0
        YOUT(I)=0
 30   CONTINUE
C
      READ(32,*,END=999)NX,(PSUBX(I),I=1,NX),(PX(PSUBX(I)),I=1,NX),
     +                  NY,(PSUBY(I),I=1,NY),(PY(PSUBY(I)),I=1,NY)
C
C  Convert preamplifier outputs to all channels theoretically possible
      DO 40 I=0,69
        IF (PX(I).GT.0) THEN
          XOUT(I)=PX(I)
          N=MOD(41*I,70)+70
          XOUT(N)=PX(I)
        ENDIF
        IF (PY(I).GT.0) THEN
          YOUT(I)=PY(I)
          N=MOD(41*I,70)+70
          YOUT(N)=PY(I)
        ENDIF
 40   CONTINUE
C
```

```
C  Eliminate islands
       IF ((XOUT(0).GT.0).AND.(XOUT(1).LE.0)) XOUT(0)=-XOUT(0)
       IF ((YOUT(0).GT.0).AND.(YOUT(1).LE.0)) YOUT(0)=-YOUT(0)
       IF ((XOUT(MAXCHAN-1).GT.0).AND.(XOUT(MAXCHAN-2).LE.0))
     +          XOUT(MAXCHAN-1)=-XOUT(MAXCHAN-1)
       IF ((YOUT(MAXCHAN-1).GT.0).AND.(YOUT(MAXCHAN-2).LE.0))
     +          YOUT(MAXCHAN-1)=-YOUT(MAXCHAN-1)
       DO 50 I=1,MAXCHAN-2
         IF ((XOUT(I).GT.0).AND.(XOUT(I-1).LE.0).AND.
     +       (XOUT(I+1).LE.0))  XOUT(I)=-XOUT(I)
         IF ((YOUT(I).GT.0).AND.(YOUT(I-1).LE.0).AND.
     +       (YOUT(I+1).LE.0))  YOUT(I)=-YOUT(I)
 50    CONTINUE
C
C  Find the number of active channel clumps in each direction
        CALL COUNTCLUMPS(XOUT,NXCLUMPS)
        CALL COUNTCLUMPS(YOUT,NYCLUMPS)
C
        IF ((NXCLUMPS.GT.1).OR.(NYCLUMPS.GT.1)) THEN
          CALL FIND2(XOUT,YOUT,NXCLUMPS,NYCLUMPS)
        ELSE
C    In this case, the number of clumps in each direction is 1
C    CLUMPINFO returns start channel numbers, widths, and amplitude
C    sums for the clump found
          CALL CLUMPINFO(XOUT,0,XSTART,XWIDTH,XSUM)
          CALL CLUMPINFO(YOUT,0,YSTART,YWIDTH,YSUM)
          IF ( (XWIDTH.NE.YWIDTH).OR.(XSUM.NE.YSUM).OR.
     +         (XWIDTH.GT.5) ) THEN
            CALL FIND2(XOUT,YOUT,NXCLUMPS,NYCLUMPS)
          ELSE
            WRITE(33,'(2I3)')XWIDTH,YWIDTH
C  Write out x channels triggered in raw format
            WRITE(33,81)(XSTART+J,J=1,XWIDTH)
C  Write out x amplitudes
            WRITE(33,81)(XOUT(XSTART+J),J=0,XWIDTH-1)
C  Write out y channels triggered in raw format
            WRITE(33,81)(YSTART+J,J=1,YWIDTH)
C  Write out y amplitudes
            WRITE(33,81)(YOUT(YSTART+J),J=0,YWIDTH-1)
          ENDIF
        ENDIF
C  End of main loop
       GO TO 10
 81    FORMAT(100I4)
 999   STOP
       END
C
C
       SUBROUTINE FIND2(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS)
C  Top level processor for case where double events are suspected
       INTEGER MAXCHAN
       PARAMETER(MAXCHAN=140)
       INTEGER XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,I,INIT
       INTEGER WX(10),WY(10),XSTART(10),YSTART(10),XSUM(10),YSUM(10)
       LOGICAL ELIM2A,ELIM2B
       DIMENSION XCHAN(0:MAXCHAN-1),YCHAN(0:MAXCHAN-1)
       DO 1 I=1,10
         XSUM(I)=0
         YSUM(I)=0
 1     CONTINUE
```

```
C  Accumulate information about the clumps
      INIT=0
      DO 10 I=1,NXCLUMPS
         CALL CLUMPINFO(XCHAN,INIT,XSTART(I),WX(I),XSUM(I))
         INIT=XSTART(I)+WX(I)
 10   CONTINUE
      INIT=0
      DO 20 I=1,NYCLUMPS
         CALL CLUMPINFO(YCHAN,INIT,YSTART(I),WY(I),YSUM(I))
         INIT=YSTART(I)+WY(I)
 20   CONTINUE
C  Eliminate spurious endpoints of clumps with width GT 2
      CALL ZAPEND(NXCLUMPS,XCHAN,XSTART,WX,XSUM)
      CALL ZAPEND(NYCLUMPS,YCHAN,YSTART,WY,YSUM)
C  Sort the clump info pointers by clump width
 30   IF (NXCLUMPS.GT.1) CALL SORT(NXCLUMPS,WX,XSTART,XSUM)
      IF (NYCLUMPS.GT.1) CALL SORT(NYCLUMPS,WY,YSTART,YSUM)
C  The following code operates like an expert system; each condition
C  of the case structure represents a known situation, and a low
C  confidence catch-all condition occurs at the end
      IF ((NXCLUMPS.EQ.2).AND.(NYCLUMPS.EQ.2).AND.(WX(1).LT.WX(2))
     +    .AND.(WX(1).EQ.WY(1)).AND.(WX(2).EQ.WY(2))) THEN
C  Case where there are clearly 2 events of different widths
         CALL WRITER(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,XSTART,YSTART,WX,WY)
      ELSEIF ((NXCLUMPS.EQ.2).AND.(NYCLUMPS.EQ.1).AND.
     +        (WX(1).EQ.WX(2)).AND.(WX(1).EQ.WY(1))) THEN
C  Case where events line up in the y dimension, but widths equal
         CALL WRITER(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,XSTART,YSTART,WX,WY)
      ELSEIF ((NXCLUMPS.EQ.1).AND.(NYCLUMPS.EQ.2).AND.
     +        (WY(1).EQ.WY(2)).AND.(WX(1).EQ.WY(1))) THEN
C  Case where events line up in the x dimension, but widths equal
         CALL WRITER(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,XSTART,YSTART,WX,WY)
      ELSEIF ( (NXCLUMPS.GT.2).AND.(WX(NXCLUMPS-1).GT.2).AND.
     +         (WX(1).EQ.2).AND.
     +         (ELIM2A(XCHAN,NXCLUMPS,WX,XSTART,XSUM)) ) THEN
C  Case where a width 2 clump is multiplexed with other clumps of
C  larger width.  ELIM2A checks the duals, and if successful, reduces
C  the number of clumps, resets the width, start, and sum pointers,
C  and returns .TRUE.
         GO TO 30
      ELSEIF ( (NYCLUMPS.GT.2).AND.(WY(NYCLUMPS-1).GT.2).AND.
     +         (WY(1).EQ.2).AND.
     +         (ELIM2A(YCHAN,NYCLUMPS,WY,YSTART,YSUM)) ) THEN
         GO TO 30
C
      ELSEIF ( (NXCLUMPS.GT.2).AND.(WX(2).EQ.2).AND.
     +         (WX(NXCLUMPS).GT.2).AND.
     +         (ELIM2B(XCHAN,NXCLUMPS,WX,XSTART,XSUM)) ) THEN
C  Case where a width 2 clump is multiplexed with other width 2
C  clumps.  ELIM2B looks for width 2 clumps multiplexed with a larger
C  clump and eliminates them.  If any clumps are eliminated, ELIM2B
C  returns .TRUE.
         GO TO 30
      ELSEIF ( (NYCLUMPS.GT.2).AND.(WY(2).EQ.2).AND.
     +         (ELIM2B(YCHAN,NYCLUMPS,WY,YSTART,YSUM)) ) THEN
         GO TO 30
      ELSE
         CALL WRITER(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,XSTART,YSTART,WX,WY)
      ENDIF
      RETURN
      END
```

```
      LOGICAL FUNCTION ELIM2A(CHAN,NCLUMPS,WIDTH,START,SUM)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER CHAN,WIDTH,START,SUM,NCLUMPS,DUAL,I
      DIMENSION CHAN(0:MAXCHAN-1),WIDTH(10),START(10),SUM(10)
C  When there are more than 2 clumps in one dimension, determine
C  whether one of the width 2 clumps is spurious by seeing if
C  the duals of its members are part of other clumps
C
      ELIM2A=.FALSE.
      IF ( (WIDTH(1).EQ.2).AND.
     +        (CHAN(DUAL(START(1))).EQ.CHAN(START(1))).AND.
     +        (CHAN(DUAL(START(1)+1)).EQ.CHAN(START(1)+1)) )THEN
         CHAN(START(1))=-CHAN(START(1))
         CHAN(START(1)+1)=-CHAN(START(1)+1)
         NCLUMPS=NCLUMPS-1
         DO 100 I=1,NCLUMPS
            WIDTH(I)=WIDTH(I+1)
            START(I)=START(I+1)
            SUM(I)=SUM(I+1)
 100     CONTINUE
         ELIM2A=.TRUE.
        ENDIF
      RETURN
      END
C
      LOGICAL FUNCTION ELIM2B(CHAN,NCLUMPS,WIDTH,START,SUM)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER CHAN,WIDTH,START,SUM,NCLUMPS,DUAL,CLNO,I,J,K
      DIMENSION CHAN(0:MAXCHAN-1),WIDTH(10),START(10),SUM(10)
      LOGICAL MEMBER
C  If a width 2 clump is found which is multiplexed with a larger
C  clump, it is eliminated.  This function is only invoked when
C  there are at least 2 width 2 clumps
        ELIM2B=.FALSE.
        CLNO=1
 10   IF ( (WIDTH(CLNO).EQ.2).AND.(CLNO.LT.NCLUMPS) ) THEN
         I=DUAL(START(CLNO))
         J=DUAL(START(CLNO)+1)
C       See if I or J is multiplexed into the larger clump
         IF ( (MEMBER(I,START(NCLUMPS),WIDTH(NCLUMPS))).OR.
     +        (MEMBER(J,START(NCLUMPS),WIDTH(NCLUMPS))) ) THEN
C         Eliminate the current clump
C         Do not increment CLNO in this case.
          CHAN(START(CLNO))=-CHAN(START(CLNO))
          CHAN(START(CLNO)+1)=-CHAN(START(CLNO)+1)
C           Move width, start, and sum pointers down
          DO 30 K=CLNO,NCLUMPS-1
            WIDTH(K)=WIDTH(K+1)
            START(K)=START(K+1)
            SUM(K)=SUM(K+1)
 30       CONTINUE
          NCLUMPS=NCLUMPS-1
          ELIM2B=.TRUE.
         ELSE
         CLNO=CLNO+1
         ENDIF
         GO TO 10
      ENDIF
      RETURN
      END
```

```
      LOGICAL FUNCTION MEMBER(CHNO,START,WIDTH)
      INTEGER CHNO,START,WIDTH
C  Determine whether CHNO is part of the clump of length WIDTH
C  beginning at channel number START
      IF ( (CHNO.GE.START).AND.(CHNO.LT.START+WIDTH) ) THEN
         MEMBER=.TRUE.
      ELSE
         MEMBER=.FALSE.
      ENDIF
      RETURN
      END
C
      SUBROUTINE WRITER(XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,XSTART,YSTART,WX,WY)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER XCHAN,YCHAN,NXCLUMPS,NYCLUMPS,TOTXCH,TOTYCH
      INTEGER WX(10),WY(10),XSTART(10),YSTART(10),I,J
      DIMENSION XCHAN(0:MAXCHAN-1),YCHAN(0:MAXCHAN-1)
      TOTXCH=0
      TOTYCH=0
      DO 10 I=1,NXCLUMPS
         TOTXCH=TOTXCH+WX(I)
 10   CONTINUE
      DO 20 I=1,NYCLUMPS
         TOTYCH=TOTYCH+WY(I)
 20   CONTINUE
      WRITE(33,'(2I3)')TOTXCH,TOTYCH
C  Write out x channels triggered in raw format
      WRITE(33,81)((XSTART(I)+J,J=1,WX(I)),I=1,NXCLUMPS)
C  Write out x amplitudes
      WRITE(33,81)((XCHAN(XSTART(I)+J),J=0,WX(I)-1),I=1,NXCLUMPS)
C  Write out y channels triggered in raw format
      WRITE(33,81)((YSTART(I)+J,J=1,WY(I)),I=1,NYCLUMPS)
C  Write out y amplitudes
      WRITE(33,81)((YCHAN(YSTART(I)+J),J=0,WY(I)-1),I=1,NYCLUMPS)
      RETURN
 81   FORMAT(100I4)
      END
C
      SUBROUTINE CLUMPINFO(CHAN,INIT,START,WIDTH,SUM)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER CHAN,INIT,START,WIDTH,SUM,I
      DIMENSION CHAN(0:MAXCHAN-1)
C  Find consecutive active channels and their amplitude sum
C  starting search at position INIT
C
C  Find the first non-zero amplitude channel
      START=INIT
 5    IF (CHAN(START).LE.0) THEN
         START=START+1
         IF (START.GE.MAXCHAN) THEN
            WIDTH=0
            SUM=0
            RETURN
         ELSE
            GO TO 5
         ENDIF
      ENDIF
```

```
C  Find total width and accumulate sum
        SUM=CHAN(START)
        WIDTH=1
        I=START+1
 10     IF (CHAN(I).GT.0) THEN
           WIDTH=WIDTH+1
           SUM=SUM+CHAN(I)
           I=I+1
           IF (I.GT.MAXCHAN-1) RETURN
           GO TO 10
        ENDIF
        RETURN
        END
C
      SUBROUTINE SORT(NCLUMPS,WIDTH,START,SUM)
      INTEGER NCLUMPS,WIDTH(10),START(10),SUM(10),I,J
C  Sort the clump width, start, and sum pointers by width
C  Bubble sort is appropriate here since NCLUMPS is 5 or less
      DO 10 I=1,NCLUMPS-1
         DO 20 J=1,NCLUMPS-I
            IF (WIDTH(J).GT.WIDTH(J+1)) THEN
               CALL SWAP(WIDTH(J),WIDTH(J+1))
               CALL SWAP(START(J),START(J+1))
               CALL SWAP(SUM(J),SUM(J+1))
            ENDIF
 20      CONTINUE
 10   CONTINUE
      RETURN
      END
C
      SUBROUTINE SWAP(I,J)
        INTEGER I,J,TEMP
      TEMP=I
      I=J
      J=TEMP
      RETURN
      END
C
        SUBROUTINE COUNTCLUMPS(CHAN,NCLUMPS)
        INTEGER MAXCHAN
        PARAMETER(MAXCHAN=140)
        INTEGER CHAN,NCLUMPS,I,START,WIDTH,SUM
        DIMENSION CHAN(0:MAXCHAN-1)
        NCLUMPS=0
        I=0
 10     CALL CLUMPINFO(CHAN,I,START,WIDTH,SUM)
        IF (WIDTH.EQ.0) RETURN
        NCLUMPS=NCLUMPS+1
        I=START+WIDTH
        IF (I.LT.MAXCHAN-1) GO TO 10
        RETURN
        END
C
      INTEGER FUNCTION DUAL(N)
      INTEGER N
      IF (N.LT.70) THEN
         DUAL=MOD(41*N,70)+70
      ELSE
         DUAL=MOD(41*N,70)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ZAPEND(NCLUMPS,CHAN,START,WIDTH,SUM)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER I,NCLUMPS,ENDPT,CHAN,START,WIDTH,SUM,DUAL,GSTART,WSUB
      INTEGER PREDICT
      DIMENSION CHAN(0:MAXCHAN-1),WIDTH(10),START(10),SUM(10)
      DO 25 I=1,NCLUMPS
        IF ( (WIDTH(I).GT.2).AND.
     +       (CHAN(START(I)).GT.CHAN(START(I)+1)).AND.
     +       (CHAN(DUAL(START(I))).GT.0) ) THEN
C         Extract the Gaussian shaped clump from a possibly large clump
C           looking from the left side of the clump
          CALL GETSUBCLUMP(CHAN,START(I),WIDTH(I),GSTART,WSUB,1)
C         Extrapolate the correct value for CHAN(START(I))
          CALL EXTRAPOLATE(CHAN,WSUB,START(I)+1,START(I),PREDICT)
C         Adjust the amplitudes of the endpoint and its dual
          SUM(I)=SUM(I)-CHAN(START(I))+PREDICT
          CHAN(START(I))=PREDICT
          CHAN(DUAL(START(I)))=CHAN(DUAL(START(I)))-PREDICT
C         Readjust width and start pointers if necessary
          IF (PREDICT.EQ.0) THEN
            WIDTH(I)=WIDTH(I)-1
            START(I)=START(I)+1
          ENDIF
        ENDIF
        ENDPT=START(I)+WIDTH(I)-1
        IF ( (WIDTH(I).GT.2).AND.
     +       (CHAN(ENDPT).GT.CHAN(ENDPT-1)).AND.
     +       (CHAN(DUAL(ENDPT)).GT.0) ) THEN
C         Extract the Gaussian shaped clump from a possibly large clump
C           looking from the right side of the clump
          CALL GETSUBCLUMP(CHAN,START(I),WIDTH(I),GSTART,WSUB,-1)
C         Extrapolate the correct value for CHAN(ENDPT)
          CALL EXTRAPOLATE(CHAN,WSUB,GSTART,ENDPT,PREDICT)
C         Adjust the amplitudes of the endpoint and its dual
          SUM(I)=SUM(I)-CHAN(ENDPT)+PREDICT
          CHAN(ENDPT)=PREDICT
          CHAN(DUAL(ENDPT))=CHAN(DUAL(ENDPT))-PREDICT
C         Readjust width pointer if necessary
          IF (PREDICT.EQ.0) WIDTH(I)=WIDTH(I)-1
        ENDIF
 25   CONTINUE
      RETURN
      END
C
      SUBROUTINE GETSUBCLUMP(CHAN,START,WIDTH,GSTART,WSUB,INCR)
      INTEGER MAXCHAN
      PARAMETER(MAXCHAN=140)
      INTEGER CHAN,START,WIDTH,GSTART,WSUB,INCR,POS
      DIMENSION CHAN(0:MAXCHAN-1)
C  Find the Gaussian subclump starting from the left if INCR=1 or
C  from the right if INCR=-1
      WSUB=1
C  Set starting position for Gaussian clump search; endpoint is bad
      IF (INCR.EQ.1) THEN
        POS=START+1
      ELSE
        POS=START+WIDTH-2
      ENDIF
```

```
C  Find other end of Gaussian subclump by climbing one hill until
C  the next hill is detected or end of given clump
C    First uphill
 10   IF (CHAN(POS).LE.CHAN(POS+INCR)) THEN
        WSUB=WSUB+1
        POS=POS+INCR
        IF ( (POS.GT.START+1).AND.(POS.LT.START+WIDTH-2).AND.
     +       (CHAN(POS).GT.0) ) GO TO 10
      ENDIF
C    Then downhill
 20   IF ( (POS.GT.START).AND.(POS.LT.START+WIDTH-1).AND.
     +     (CHAN(POS).GE.CHAN(POS+INCR)) )  THEN
        WSUB=WSUB+1
        POS=POS+INCR
        GO TO 20
      ENDIF
C    Set starting pointer for the Gaussian sub clump
      IF (INCR.EQ.1) THEN
        GSTART=START
      ELSE
        GSTART=START+WIDTH-WSUB-1
      ENDIF
      RETURN
      END
C
      SUBROUTINE EXTRAPOLATE(CHAN,WIDTH,XSTART,XSEARCH,PREDICT)
      INTEGER MAXCHAN,OPP1,OPP2,Y1,Y2
      REAL C,XOPP,CENTER
      PARAMETER(MAXCHAN=140)
      INTEGER CHAN,WIDTH,XSTART,XSEARCH,PREDICT
      DIMENSION CHAN(0:MAXCHAN-1)
C  Find the center C of the distribution
      C=CENTER(CHAN,WIDTH,XSTART)
C  Compute the two x values closest to the opposite from XSEARCH
C  These formulas are correct whether XSEARCH is to the left or the
C  right of C.
      OPP1=2*C-XSEARCH
      OPP2=OPP1+1
C  Get the opposite value from XSEARCH relative to C as a real number
      XOPP=2*C-XSEARCH
C  Set PREDICT to the linear interpolated value
      IF ( (OPP1.LT.XSTART).OR.(OPP1.GE.XSTART+WIDTH) ) THEN
        Y1=0
      ELSE
        Y1=CHAN(OPP1)
      ENDIF
      IF ( (OPP2.GE.XSTART+WIDTH).OR.(OPP2.LT.XSTART) ) THEN
        Y2=0
      ELSE
        Y2=CHAN(OPP2)
      ENDIF
      PREDICT=(1.0*Y2-Y1)/(1.0*OPP2-OPP1)*(XOPP-OPP1)+Y1
      RETURN
      END
```

```
      REAL FUNCTION CENTER(CHAN,WIDTH,START)
      INTEGER MAXCHAN,I,WIDTH,START,CHAN
      REAL YSUM,XDOTY
      PARAMETER(MAXCHAN=140)
      DIMENSION CHAN(0:MAXCHAN-1)
C  Determine the center of the clump starting at START of size WIDTH
      YSUM=0
      XDOTY=0
      DO 10 I=START,START+WIDTH-1
        YSUM=YSUM+CHAN(I)
        XDOTY=XDOTY+I*CHAN(I)
 10   CONTINUE
      IF (YSUM.EQ.0) THEN
        WRITE(*,*)'*** ERROR -- YSUM = 0'
        CENTER=9999
        RETURN
      ENDIF
      CENTER=XDOTY/YSUM
      RETURN
      END
```

APPROVAL

MULTIPLEXING READOUT CHANNELS IN PROPORTIONAL COUNTERS

By James Caristi

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

E. Tandberg-Hanssen

E. TANDBERG-HANSSEN
Director
Space Science Laboratory

☆ U.S. GOVERNMENT PRINTING OFFICE 1991–531-081/40136