UCRL-9099

UNIVERSITY OF CALIFORNIA
Lawrence Radiation Laboratory
Berkeley, California

Contract No. W-7405-eng-48

REFERENCE MANUAL FOR KICK IBM PROGRAM

Arthur H. Rosenfeld, Editor

May 1961

DISCLAIMER

REFERENCE MANUAL FOR KICK IBM PROGRAM

Arthur H. Rosenfeld, Editor

Lawrence Radiation Laboratory
University of California
Berkeley, California

May 1961

ABSTRACT

This reference manual describes the IBM 704 program called Kick, by which complete bubble chamber events are kinematically analyzed. Kick's input data is the output from the Pang program, which uses raw track measurements to spatially reconstruct the tracks, and fits appropriate curves to them. The reader who requires a more introductory exposition of the general ideas is directed to UCRL-9098 which describes an earlier version of Kick in less detail.

MBANK for Kick 5 Only

MBANK+	(constitutes first record of binary output)	
0(1)*	Run I. D. [Copied from DATE (in TEMPAR)]	Full Wd Integer
1(2)	STM--Now used for KKT FailingTrack No. Contains 9999 if event not found in ZTABLE	Int D
2 & 3	Date Franckensteined	YR/MO/DAbbbb
4 & 5	Date Panged	YR/MO/DAbbbb
6(7)	Measurement No. (1 on first remeasurement)	Int D
7(8)	Franckenstein Operator Number	Int D
8(9)	Experiment number	Int D
9(10)	Event type	Int D
10 & 11	Serial number (R=Roll, F=Frame, T=Beam Track)	RRRbbbFFFbTT
12(13)	PUC "Pick-Up Command" The character P from PANGMC+6	Int D
13	Free	
14(15)	GPM "Good-Print Mark" Counts number of successful fits	Int D
15(16)	χ^2; 500 signifies HRJ; but for missing-mass "fit", "χ^2" is χ^2(Un3P)	Fl.
16(17)	DAMN (Sign bit determines sign used in OC ambiguity)	Sign Bit
17(18)	Report counter (Counts reports on a given beam track)	Int D
18(19)-24(25)	HWdD through J; Floating handy words	Fl.
25(26)	"Fuzzy KKT's". Reserved for $\Sigma_{fuzzy}(KKT)^2$†	Fl.
26(27)	LPM-Nonzero for last Print of an Event	
27(28)	CPM - "Check-Point Mark", also called Fit I. D.	Int D
28(29)	Parameter ID	Int D
29(30)	WALL. Max. proj. range of zero-length charged track (cm)	Fl.
30(31)	MMASS - missing-mass μ. Negative means $E_{out}-E_{in} < 0$	Fl.
31(32)	DMMASS - $\delta\mu$ Negative means $\mu^2 < 0$	Fl.
32(33)	ZCCTR1 = Z(Control of DAMN sign) Counter	Int D
33(34)	Ord	Int A
34(35)	Print type 0=Param, 1=P, 2=V, 3=MM, 4=HRJ, 5=FRJ, 77777=end of tape	Int D
35(36)	Length of second record of output	Int D
36(37)	Tracks in second record of output	Int D
37(38)	Reject type	Int D
38-40	ZVERT calling sequence	Coded
41(42)	Free	
42(43)	Free	
43(44)	Free	
44(45)	Free	
45(46)-59(60)	HWd1 through 15; floating handy words	Fl.
60(61)-74(75)	IHWd1 through 15; more integer handy words	Int D
SNYDER+	(Starts second binary record of output. Has additional GUTS fit information. Listed on assembly between GUTS and Kick. Described in Glossary of both write-ups and in missing-mass section.) Particularly useful locations are:	
0(1)	KLOCP (P is number of particles fitted by GUTS)	Int D
1(2)	KLOCL (L is number of constraints used by GUTS)	Int D
2(3)	KLOCI (I is number of variables)	Int D
3(4)	KLOCLL (Flag for special classes, two vertex or WALL)	Int D
4(5)		Int D
5(6)	GCONTR (Step Counter)	Int D
6-12	CUT1 through CUT (Cutstep Counters - see RJCT2 in Memo. 86)	Int D
46(47)	MPRIME [true value of χ^2, never touched by Reject routines or χ^2(Un4) for MM]	Fl.
49(50)	KDAMN (actual sign of DAMN as used by GUTS).	Int D

*Parentheses indicate MBANK word numbers (FORTRAN Convention) on output tape.

†Not yet in program.

ZTn Banks, Kick 5
(See also ZTn glossary)

Kick part, i.e. processing part
Set up by ZSET: overwritten by ZVERT and ZSWIM

ZTn+	Name	Units	Format
0(8)*	$\phi(0<\phi<2\pi)$ or flags; Wall=1 Poison=7	radians	Fl. IntA IntA
1(9)*	$s=\tan\lambda$		Fl.
2(10)*	k	$(\text{Mev/c})^{-1}$	Fl.
3(3)*	Status (+/-) =(fit/unfit)		Int D
4(4)*	Crn =0 $\overline{\delta\phi\delta\phi}$ =1 $\overline{\delta\phi\delta\phi}$ =3		Int D
5	$\overline{\delta\phi\delta\phi}$ $\overline{\delta\phi\delta\phi}$		Fl.
6	$\overline{\delta\phi\delta s}$ $\overline{\delta\phi\delta s}$		Fl.
7	$\overline{\delta\phi\delta k}$ $\overline{\delta s\delta\phi}$		Fl.
8	$\overline{\delta s\delta\phi}$ $\overline{\delta s\delta s}$		Fl.
9	$\overline{\delta s\delta s}$		Fl.
10	$\overline{\delta s\delta k}$		Fl.
11	$\overline{\delta k\delta\phi}$		Fl.
12	$\overline{\delta k\delta s}$		Fl.
13	$\overline{\delta k\delta k}$		Fl.
14(5)*	Endwd (+/-)=angles set up at (beg/end)		Int D

Error - Matrix; GARBAGE → ↓; GARBAGE ↑

Pang appendage

ZTn+	Name	Units	Format
15(6)*	Lsz; L=1, s=2, z=3, B=4		Int D
16(7)*	Ftn		IntD

From Pang ↓

ZTn+	Name	Units	Format
17(1)*	Track No.		IntD
18(2)*	Mass (and charge)	Mev	Fl.
19	L	cm	Fl.
20	δL	cm	Fl.
21	k [For 2pt {1.0}]	$(\text{Mev/c})^{-1}$	Fl.
22	δk [Set to {10.0}]	$(\text{Mev/c})^{-1}$	Fl.
23	δp via arc If neutral set 10.0	(Mev/c)	Fl.
24(11)	kPTest or kKTest		Fl.
25	$\overline{\delta\phi\delta s}$		Fl.
26	$\overline{\delta\phi\delta k}$		Fl.

From Pang ↑

ZTn+	Name	Units	Format
27	|k| via arc (Neutrals set to 1.0)	Mev/c	Fl.
28	δk via arc (Neutral set to 10.0)	Mev/c	Fl.
29	ϕ_b(beginning azimuth) $0<\phi<2\pi$	radians	Fl.
30	ϕ_e (end)	radians	Fl.
31	$\delta\phi$(beg. and end)	radians	
32	s_b(beg. slope)		Fl.
33	s_e (end slope)		Fl.
34	δs(both ends)		Fl.
35	Coulomb ratio		Fl.
36	$\overline{\delta k_{arc}\delta s}$ (Neutral set to 0)	$(\text{Mev/c})^{-1}$	Fl.
37	Track code (Contains LSZ, FTN information)		Coded
38	Points measured view P		IntD
39	Points measured view Q		IntD
40	p_b(For 2p set to 1.0)	Mev/c	Fl.
41	p_e(For 2p set to 1.0) or overshoot in mm	Mev/c mm	Fl.
42	p arc(if neutral set 1.0)	Mev/c	Fl.
43	$\delta p/p$ set to 10.0 for 2-point tracks		Fl.
44	Charge code, +1, 0, -1		IntD
45	X_b } beg. } Spatial coordinates	cm	Fl.
46	Y_b } beg. } Spatial coordinates	cm	Fl.
47	Z_b } beg. } Spatial coordinates	cm	Fl.
48	X_e } end } Spatial coordinates	cm	Fl.
49	Y_e } end } Spatial coordinates	cm	Fl.
50	Z_e } end } Spatial coordinates	cm	Fl.
51	σ_{xy} } residuals	cm	Fl.
52	σ_z } residuals	cm	Fl.
53	View P(integer in add) View Q(integer in dec.)		Coded
54	Pang test		Fl.

*Numbers in parentheses are the track word members (in FORTRAN convention) in a vertex report.

UCRL-9099
July 21, 1960

Contents

INTRODUCTION

General

This manual describes the Alvarez Group kinematics program Kick 5.6 for the IBM 704 as it runs in Berkeley.* For an assembly more recent than Kick 5.6, changes are indicated on the listing by a gang-punched comment (e. g. FEB 61) starting at column 68. A 709 version, Kick 6, is about to be released and will be described in a later edition of this manual.

It is assumed that the reader is familiar with the following:

1. A general description of the Franckenstein-Pang-Kick-Examin System. See A. H. Rosenfeld, Digital Computer Analysis of Data from Hydrogen Bubble Chambers at Berkeley, Alvarez Group Memo 123 (UCID-1047), 1959. Refer also to A. H. Rosenfeld, "Computer Programs and Uses," Paper 6A.9 in Proceedings of the 1960 International Conference on Instrumentation for High-Energy Physics (Lawrence Radiation Laboratory, University of California, Berkeley, November 1960).
2. A. H. Rosenfeld and J. N. Snyder, Digital Computer Analysis of Data from Bubble Chambers. IV. The Kinematic Analysis of Complete Events, UCRL-9098, February 16, 1960 (submitted to the Review of Scientific Instruments). Although this report describes an obsolete version of Kick, it is useful because of its didactic presentation.
3. IBM 704 and 709 manuals (at least the Command Structure sections), the Symbolic Assembly Program (SAP), and the local version of SAP, called 4AP and 9AP. (See the 4AP and 9AP descriptions in Section IV of the present report, entitled Notes on Auxiliary Processing Routines.)

In addition, we will refer to:

For Pang:

1. W. E. Humphrey, A Description of the Pang Program, Alvarez Group Memo 111 (UCID-1043), September 18, 1959.

---

*The owner of this copy should make sure that his name and room number (or mailing address) are on the Editor's list. He will then receive new or modified pages describing important changes in Kick. This manual itself will always need "debugging", since some of its pages will always be new. Therefore please inform the Editor of mistakes and suggestions for improvements.

2. W. E. Humphrey, A Description of the Pang Program, Supplement No. 1: Input and Output, Alvarez Group Memo 115 (UCID-1044), October 24, 1959.

For GUTS:

1. J. Peter Berge, GUTS, Alvarez Group Memo 86 (UCID-1251), January 18, 1960.

2. F. Safier and C. Hansen, IBM Program GUTS--Detailed Description and Flow, UCRL-9309, September 9, 1960.

3. J. P. Berge, F. T. Solmitz, and H. D. Taft, Digital Computer Analysis of Data from Bubble Chambers, III. The Kinematical Analysis of Interaction Vertices, UCRL-9097, March 15, 1960, to be published in Review of Scientific Instruments.

This book is written as a reference manual; therefore the sequence in which topics appear may not be the easiest for straight-through reading. We suggest for the initial reading that one peruse first the summaries of the input format, coordinate system, etc. in this Introduction section. Next, read the first page of the Banks section which is the combined index for the section plus the list of the banks and their functions. Then read the MBANK and Track Bank lists and their accompanying glossaries. Next, read the over-all flow diagram (first page of Section III, MAIN PROCESSING ROUTINES and the description of Output in this Introduction section. Finally, read simultaneously for each main processing routine (a) Section II. Calling Sequence and Summaries of Kick Processing Routines and (b) Sections III. Notes on the Main Processing Routines. In addition, as their use is indicated, the reader can read Notes on the Auxiliary Processing Routines, with their flow diagrams (and perhaps an assembly listing).

GUTS and Kick were written independently. The definitions in Kick were headed with a Z to avoid the possibility of choosing a GUTS definition. Now that Kick and GUTS are assembled together, the Z has been dropped on some new Kick routines, but it is still convenient to remember that often Z means a part of Kick. The heading K (as in KMASS) was used in GUTS to designate banks that are to be filled by Kick as it sets up for GUTS.

The first page of the Kick Assembly Listing contains a list of the sequence in which subroutines and banks have been ORGed. The second page lists the general-purpose subroutines available in a subroutine deck called SUBPAK 1.3, which starts at $14{,}000_{10}$.

Acknowledgments

By now practically everybody in the Alvarez Group has made some contribution to Kick and (or) to this manual; it seems pointless here to thank about 20 people. However, visitors from other laboratories who have helped with the program or with this manual deserve special acknowledgment--notably J. N. Snyder from Illinois, R. L. Glasser from NRL, Rudolph Bock and Ross Macleod from CERN, and Josh Kopp from Brookhaven.

This work was done under the auspices of the U. S. Atomic Energy Commission.

A. H. Rosenfeld
J. P. Berge
C. Tate

Kick Program Tapes and 4AP Assemblies

General

A complete running version of Kick contains three separate assemblies, as follows:

1. Subroutine package (Sub pac) — go onto Kick master tape
2. Pure Kick
3. Kick event types — Event-type deck

Notice that a given assembly of Kick is used with as many different event-type assemblies as there are independent experiments in progress. Therefore Sub pac and pure Kick are kept separately as a "Kick master tape."

After a complete Kick program has been assembled and tested, it may be necessary to make corrections or changes in pure Kick to insure that it is running properly. These corrections or changes are made in the form of CML cards* and become a part of pure Kick (they are kept at the back of the Kick deck). Similarly CML's on the event types must be kept behind each event-type deck.

Program Tapes

Event-type assemblies are normally assembled many times in order to add new event types, etc. Thus it is found to be more convenient to make a separate Kick master tape, containing no event types, and use this tape to make future program tapes for experiments, than to use a complete set of binary cards when making new Kick experiment tapes. However, when only one set of event types is to be used with a pure Kick assembly, Parts A and B below can be combined by omitting steps A-2, B-1, and B-2.

A. Kick master tape (Sub-pac and Pure Kick)

1. Load into core the following cards; stacked in the sequence specified:
   (a) Sub-pac
   (b) Pure Kick
   (c) Any CML's for Kick master tape (as soon as Kick is read in, it will read its CML's).
2. Load into core the one-card program SSCTCS, which dumps core to tape, to make a self-loading Kick master tape. SSCTCS is described at the back of the subroutines section.

*"Card Mixed Loader" (see "Utility Routines").

B. Kick experiment tape

1. Load Kick master tape into core.

2. Press reset, since the Kick program, after being read into core, will select card reader looking for CML cards.

3. Load event-type deck along with CML cards; if any, into core.

4. Load into core SSCTCS to make a self-loading program for a particular experiment.

4AP Assemblies (4AP write-up is in utility-routine section)

The following is intended to give a brief outline of the procedure to be used when modifying an assembly of Kick. By following the steps listed below, either a pure Kick or an Event-Type assembly may be made.

A. General Instructions

1. When making a pure Kick assembly, it will be necessary to obtain the collated tape (4AP output, tape 3) corresponding to the Kick listing which is to be modified.

2. Corrections or modification in CML form will, in general, have been made on the old assembly. Obtain a list of these changes and make sure they are converted to ALT cards for permanent assembly via 4AP.

3. Additional modifications or corrections that are to be made should be written according to the format described in the 4AP write-up (see subroutine section).

4. In the case of an event-type assembly, it is quite feasible to assemble an entire set of event types (original event-type assembly). However, after the event types have been assembled once, it would be more convenient to make future assemblies using the collated tape from the original event-type assembly.

5. Before an event-type assembly can be made, the symbol table which is punched when assembling pure Kick must be obtained.

6. A special assembly intended to modify or add a routine in Kick can also be assembled from the Kick symbol table. If such an assembly is desired, the same procedure that is used when making an event-type assembly should be followed.

B. Card for Altering Kick 4AP Assemblies

1. The "alteration deck", which is used to modify previous assemblies of Kick, will be defined to include all of the following cards:

(a) Remark card (REM); defining the assembly. (Any remarks that might appear at the of the previous assembly should be updated with an ALT card.)

(b) Alteration cards (ALT); used to insert or remove commands. These ALT cards must be in numerical order.

(c) A final card (FIN); defining the end of the alterations.

2. When event types are assembled, the pure Kick Symbol Table must be preceded by the psuedo-op (RST), and followed by a blank card. The blank card indicates that the alterations are on cards and not on tape. Thus the word "Symbol-Table deck" will refer to the following cards:

(a) RST

(b) Symbol table

(c) Blank card

3. The card ordering for the various types of Kick assemblies will be listed below. The event-type assembly using a collated tape (mentioned in Section (c) below) may be necessary because either (1) a New Kick assembly forces use of a new symbol table or (2) ALT's in event-type processing are desired, or both. [A comment on (c)-2 below: The RST from the previous event-type assembly will appear on the collated tape and will be executed, which will cause difficulties if not removed by an ALT.]

(a) Pure Kick assembly using collated tape
    (1) 4AP call card
    (2) Alteration deck

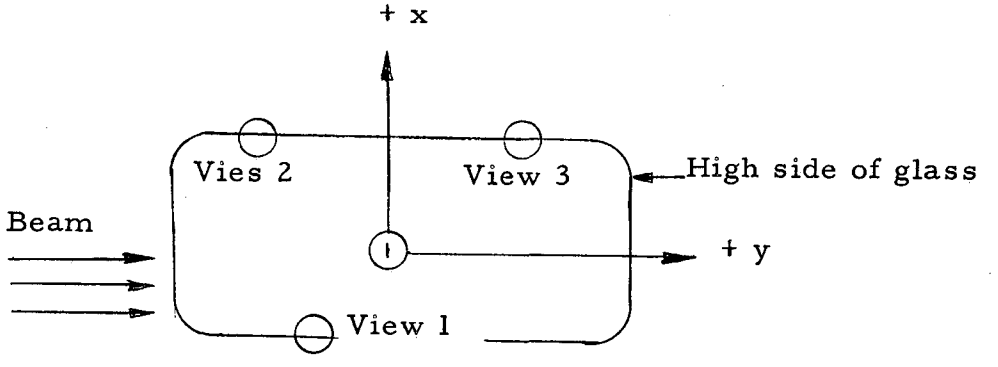
(b) Event-type assembly (original)
    (1) 4AP call card
    (2) Symbol-Table deck
    (3) Alteration deck

(c) Event-type assembly using collated tape
    (1) 4AP call card
    (2) ALT card removing old RST
    (3) Symbol-Table deck
    (4) Alteration deck.

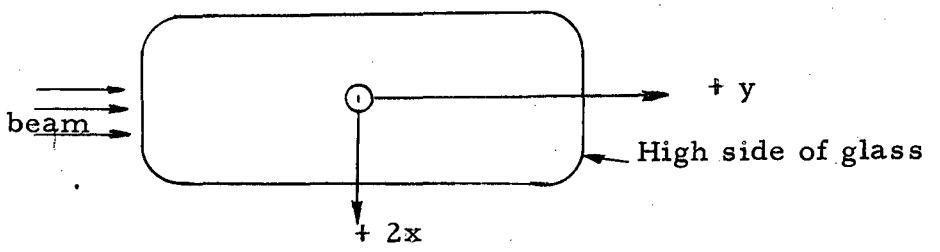
Coordinate System and Variables

The coordinate system used by Pang (and hence by Kick) is based on a right-handed coordinate system for the 15-in. chamber, and a left-handed system for the 72-in. chamber. In both chambers, the Z axis is perpendicular to the bottom of the chamber (not the tilted glass), i. e., parallel to the nominal magnetic-field axis. The origin is at the bottom of the chamber (top of the coat hangers) and in the center of the square defined by the camera lenses. The x coordinate increases toward the high side of the front glass (which is the side on which views 2 and 3 are located in the 72-in. chamber and on which 1 and 3 are located in the 15-in. chamber). The polar angles are defined so that λ = [90 deg - (angle between z axis and track direction)], where λ is the latitude, and the azimuth, ϕ is the angle measured in the x-y plane from the x axis (the y axis has ϕ = 90 deg). See Memo 111, p. 12.

The inversion in the case of the 72-in. chamber arises from a mirror between the glass plate and the film, i. e. at the bubble chamber itself. There is no inversion between the optical systems of the measuring projector and the scanning table. Hence, Pang-Kick-Examin print-outs agree with the scanning table, but <u>not</u> with real life. In summary, the 72-in. chamber is really like this:



Because of the inversion, the coordinate system, as seen on the scanning table is right-handed. The top view is:



Users of the 72 inch chamber must remember to change the sign of any vector cross-product obtained from a printout in order to make it realistic.

The variables used to specify tracks are "azimuth", $s (= \tan \lambda)$ and "curvature" $[k = 1/(p \cos \lambda)$, where p is the momentum in Mev/c]. These were chosen since it was hoped that, from the method of measurement and track reconstruction used, these would be independent and have an approximately Gaussian distribution, or at least more nearly so than any other available choice of variables. In fact, we know that while these variables at the middle are approximately independent, the curvature and azimuth at one end of a track are highly correlated, but we ignore this fact now.

Identification Numbers

In a system as complex as the present one, it is frequently important to know the history of a given event. Several identification numbers are output by Kick to provide this history. Changes both in the number and location of these identification words will occur in later versions of Kick, but those for Kick 5.6 are listed below.

Run ID (Old Kick ID) in MBANK + 0 and in DATE

This is the date the event was processed by Kick. It is loaded by a CML* card into DATE (in TEMPAR) as an integer. The format should be two digits each for year, month, and day, followed by four digits for such things as shot number; for example, 6104140002 would mean shot No. 2 of April 14, 1961. If this number is not encountered at the start of a run an HPR is encountered.

Parameter ID in MBANK + 28 and in PARID

This is an identification word for the parameters and might contain the date the parameters were decided on. Since this number is increased whenever PARCH cards are read, the format should be the same as for Run ID with four digits free to the right.

Date Measured in MBANK + 2 and MBANK + 3

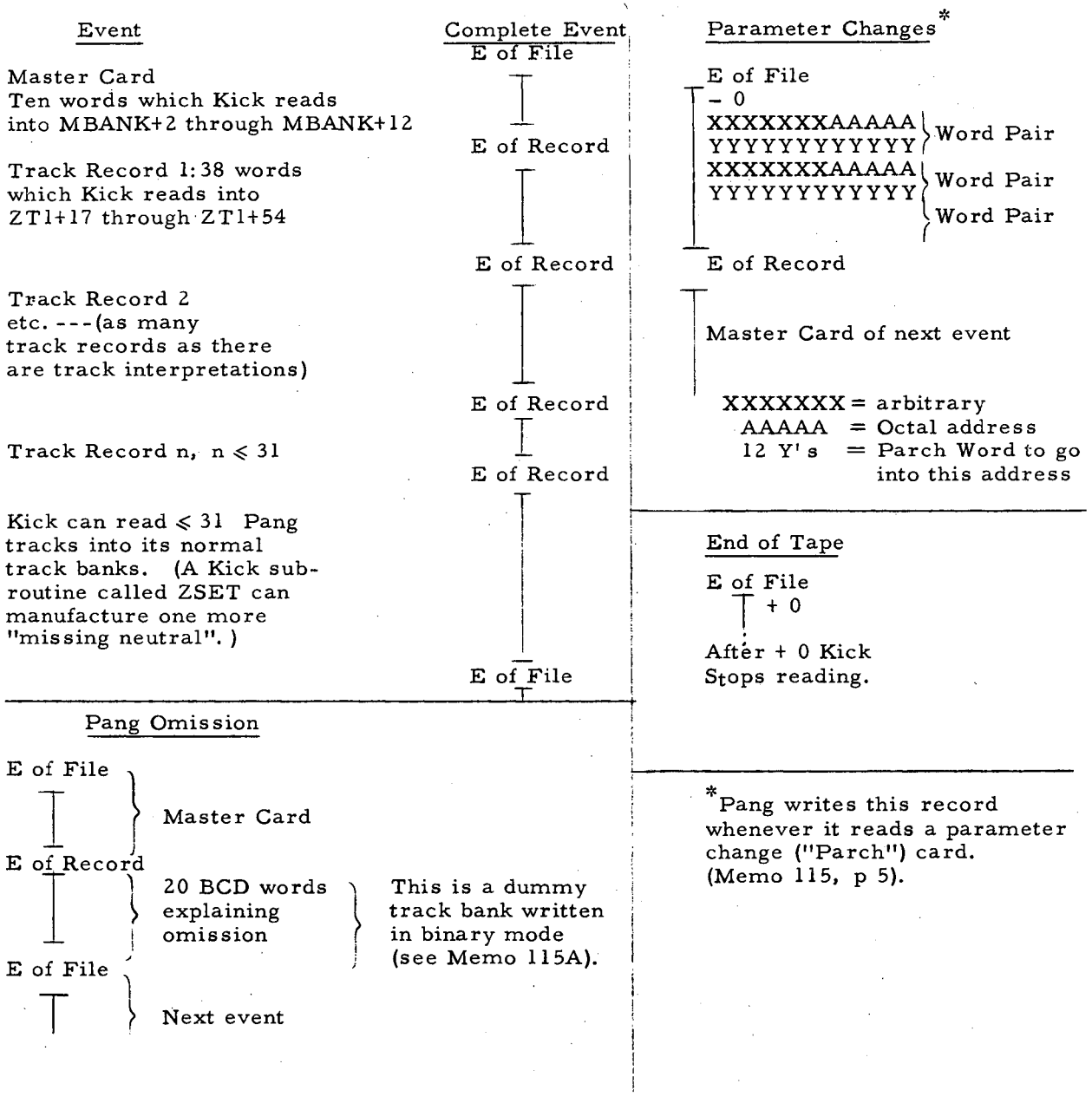
Date Panged in MBANK + 4 and MBANK + 5

*CML = "Card Mixed Loader" (See "Subroutines" Section).

Input

Pang Binary Output (Kick Input) Tape Formats - (Tape 3):

(For more details see W. E. Humphrey's Memorandum 115)



*Pang writes this record whenever it reads a parameter change ("Parch") card. (Memo 115, p 5).

k-Pang Test (KPT) and k-Kick Test (KKT)

Both before and after a fit, tests are made on the consistency of curvature and length, and the results stored in ZTn + 24. These tests are most often useful in discovering a wrong-mass hypothesis. When performed on the Pang data, they are referred to as KPT; later the tests on fitted data are written over them and called KKT.

A comment on notation: in Kick we use L for the measured length of a track and R for its range, as calculated from k (and M).

k-Pang Test

This is calculated in ZSET. ZVERT then rejects any hypothesis that fails by "three" standard deviations "three" is actually the parameter KPTLVL in PARAK) to satisfy the following requirement:

For a stopping track (i.e. $\ell sz = 2$): R = L

For a leaving track (i.e. $\ell sz = 1$): $R \geq L$

For a beam-averaged track: P(beam) = P(Pang).

If expressed in terms of length, however, the distribution would be skew; so we use the measured (Pang) curvature at the middle of the track k_m^P. We compare this measured curvature with the theoretical curvature for a given mass hypothesis, k(L/2), i.e. k computed assuming a mass and the residual range L/2. Specifically, we form, and store in ZTn + 24 the following k-Pang Test functions:

Stopping tracks $$-\left|\frac{k(L/2)-|k_m^P|}{\delta k_m^P}\right|$$ should be > (-"Three")

Leaving tracks: $L < R$ so $k(L/2) > k_m^P \Rightarrow$ should be > (-"Three")

$$KPT \equiv + \frac{k(L/2) - |k_m^P|}{\delta\, k_m^P}$$

Beam-averaged track:

$$KPT \equiv -\left|\frac{k^{beam} - |k_m^P|}{\sqrt{(\delta k^{beam})^2 + (\delta k_m^P)^2}}\right|$$ + analogues in ϕ and s.

should be > (-"Three").

By k^{beam} we specifically mean k_{MT}^{beam}, where the beam has been swum to the middle of the track (see Aux: BMAVG).

k-Kick Test

This can be applied only to charged, "leaving", outgoing tracks because of the following considerations:

(a) If a track was fitted as stopping, then of course any discrepancy between k and k(L) is reflected in χ^2. Thus we arbitrarily set KKT = 0.

(b) Incoming tracks must be excluded from KKT, since L is a length back upstream and has nothing to do with R(k).

But for charged, leaving tracks, ZVERT calculates

$$KKT = \frac{k(L) - k}{\delta(same)} ,$$

which is different from KPT, however, in that it should be subjected to two different tests. If any KKT is negative, ZVERT exits with the MQ negative, and with the track number of the questionable track in STM (MBANK+1). If, in addition, KKT is less than -"3" (we again use KPTLVL), the fit should be rejected. We do not give an ordinary HRJ, since we feel that the event-type writer is in the habit of thinking of the track banks having been untouched by HRJ. We therefore give the largest HRJ number possible, namely HRJ 62, and store the number of the KKT-rejected track in STM. (For further details see PRINT).

Note that we permit an ordinary "successful" print, with a deceptively low χ^2 value, for the "fuzzy" case where KKT is negative but greater than "-3". If the physicist wants to constrain the fit to have low KKT, he must call for a refit with "L" changed to "S". This is not done automatically, since the fit is consistent with the k information (via curvature). We leave it up to the physicist to decide whether he wants to use the length information. We do intend, however to set the MQ < 0 in ZVERT, and perhaps we shall eventually put in MBANK+25 the sum of the squares of all the "fuzzy" KKT's, so that Examin can add them to χ^2 for the vertex or the event if it wants to do so for some tests. This will make it possible to check on these fuzzy KKT's without having to look at many different track banks.

Missing Mass, μ

In the Kick-Examin system there are two routines that calculate a mass $\mu = \sqrt{w^2 - p^2}$ and $\delta\mu$:

(1) the "Missing Mass" entry to ZVERT

(2) the Effective Mass Fortran routine in Examin called MDM.

The ZVERT MM routine calculates the unbalance of four-momentum (UnPx, UnPy, UnPz, UnW) and then the mass μ and $\delta\mu$ of the missing "particle". In forming $\delta\mu$, GUTS assumes that there are no intertrack error correlations. That is, it assumes that although some of the tracks that enter into Un4 may be fitted tracks, the vertex at which Un4 is calculated has not been previously fitted.

The Effective-Mass routine MDM is written to handle the opposite case; it assumes that the vertex has been fitted and calculates $\delta\mu$ by using the GUTS 3n-by-3n error matrix. In summary, it will calculate the effective mass of any fitted diparticle (it will be extended soon to include triparticles).

Missing Mass Entry to ZVERT

The procedure of a 4c GUTS fit can be stopped after the first iteration, and, from the unbalance in four-momentum $\vec{P}, E \equiv -(F_1, F_2, F_3, F_4)$ and $\overline{\delta F_i \delta F_j}$, the following quantities calculated (see Memo 86, Sect V, pp. 13 and 14) and stored in:

(1) MBANK+30 and 31

The missing mass $\mu^2 \equiv E^2 - P^2$, and $\delta\mu^2$. [μ carries the sign of the energy balance (+ for $E_{in} > E_{out}$); $\delta\mu$ carries the sign of μ^2.]

(2) GMFCN

For no unbalance in three-momentum $\chi^2 \equiv \chi^2(\text{Un3})$ (see notes below).

(3) MPRIME

For no unbalance in four momentum $\chi^2 \equiv \chi^2(\text{Un4})$ (see notes below).

(4) GFFCN+0, 1, 2, 3 (SNYDER+27-30 in the Second Record)

P_x, P_y, P_z, W of missing particle

(5) GFFCN+4, 5, 6, 7 (SNYDER+31-34)

δP_x, δP_y, δP_z, δW

(6) UnP & DUnP(SNYDER+51, 52)

$P = \sqrt{(P_x^2 + P_y^2 + P_z^2)}^{1/2}$ and δP.

Notice that in the present GUTS, only μ and $\delta\mu$ are stored where they will not be overwritten at the first step; to see the rest one must call for a PRINT.* On return from an MMass GUTS entry, ZVERT of course copies GMFCN into MBANK+15 as usual.

Quantities Associated with Missing Mass

Consider a vertex which would be 4c if no particle escaped. If one then assumes a particle of mass m_π escaped, one loses three momentum measurements, and so is left with a 1c situation. The following notation is convenient: call $\mu^2 \equiv T$, $\delta\mu^2 \equiv \delta T$, and $m_\pi^2 \equiv T_0$.

Then our postulate can be tested with

$$\chi^2 \equiv \left[\frac{T - T_0}{\delta T}\right]^2 = \left[\frac{\mu^2 - m_\pi^2}{\delta\mu^2}\right]^2 . \qquad (1)$$

*In the next version of GUTS, it is suggested that μ, $\delta\mu$, $\chi^2(\text{Un3})$ and $\chi^2(\text{Un4})$ be stored automatically after the first iteration of any 4c fit. Then, for MMFLAG = 0, we get these four quantities and a fit. For MMFLAG $\neq$ 0, we get these four quantities and no further iterations.

Now we have $\delta\mu^2 = 2\mu\,\delta\mu$. Then for $\mu \approx m_\pi$, it is convenient to write

$$\chi^2 = \left[\frac{(\mu + m_\pi)(\mu - m_\pi)}{2\mu\,\delta\mu}\right]^2 \approx \left[\frac{\mu - m_\pi}{\delta\mu}\right]^2 . \qquad (2)$$

Notice further that $\chi^2 = \left(\frac{T-0}{\delta T}\right)^2$ does not test the hypothesis that no particle escaped [for this use χ^2(Un3) or χ^2(Un4)] but only that it was a massless particle (γ or ν), which is rare for strong interactions. Even if one should want to calculate $\chi^2(T_0=0)$, it is not given by Eq. (2) but by

$$\chi^2 = \left[\frac{\mu^2}{2\mu\,\delta\mu}\right]^2 - \frac{1}{4}\left(\frac{\mu}{\delta\mu}\right)^2 . \qquad (3)$$

More Detailed Discussion of χ^2(Un3) and χ^2(Un4)

If the missing momenta components P_i were uncorrelated, one would define a χ^2 for the hypothesis that no momentum is missing by $\chi^2_{UnP} = \sum_{i=1}^{4}\left(\frac{P_i}{\delta P_i}\right)^2$. To take into account that $\overline{\delta P_i \delta P_j} \neq 0$, one uses instead the reciprocal matrix $H_3^{-1} = (\overline{\delta P_i \delta P_j})^{-1}$, where $\overline{\delta P_i \delta P_j}$ is the 3-by-3 momentum-error matrix. Then we have

$$\chi^2_{Un3} = \sum_{1}^{4} P_i (H^{-1})_{ij} P_j ,$$

or, in GUTS notation for the missing-mass calculation (Memo 86, p. V-13),

$$\chi^2_{Un3} = \sum_{i}^{4} F_i (H^{-1}) F_j .$$

Notice that χ^2_{Un3} depends on three-momentum only, and does not explicitly depend on mass assignments, which may be wrong or ambiguous. For the case that no neutral particle escaped, χ^2_{Un3} should be distributed like χ^2 for three constraints ($\overline{\chi^2} = 3$). There is, of course, some implicit dependence on mass assignments because Kick has to assume a velocity (and therefore, a mass) to swim from the middle of a track to the vertex end.

One can also form a more specific χ^2_4 which tests all four components of the missing-momentum four vector:

$$\chi_4^2 = \sum_2^4 F_\lambda (H^{-1}) \lambda_\mu F_\mu \equiv \text{Sig Un4P.}$$

Thus an event with small χ^2_{Un3} and large χ^2_4 is evidence that either

(1) no neutral escaped but mass assignments are incorrect, or possibly

(2) neutrals did escape, but carried little total momentum.

GUTS puts χ^2_{Un3} into GMFCN, and χ^2_4 into MPRIME; then ZVERT copies GMFCN into MBANK.

In Assembly 4, the event-type writer will have to store GMFCN in some Handy Word. If he then calls for an immediate print, he can also see SigUn4P in MPRIME as well as UNPEE and DUNPEE, and P_x, P_y, P_z, E, δP_x, etc. which appear in the MBANK appendage as GFFCN through GFFCN+7.

Output

The binary output of Kick appears on Tape 4. Each report produces one record of MBANK followed by a second record for all the print types except File Reject. If there is to be a second record, this information is contained in MBANK.

First Record

MBANK is always 76 words long. It consists of one heading word (actually LLTB), which is required but ignored by Fortran, plus 75 words of MBANK:

MB + 26	set $\neq 0$ on last print of an event	Integers stored in the decrement
MB + 34	is PRINT TYPE, see next paragraph.	
MB + 35	is length of second record.	
MB + 36	gives number of track banks, P, in second record.	
MB + 36	gives REJECT TYPE, if any.	
MB + 38 + 39 + 40	complete ZVERT calling sequence for last entrance to ZVERT.	

Print Type

MB + 34 gives the print type:

(-) Bad Print
(+) Good Print

(0)	Parameter Print	All followed by second records
(1)	Pang Print	
(2)	Vertex Print	
(3)	Missing-Mass Print	
(4)	Hypothesis Reject	
(5)	File Reject	
$(77777)_8$	End of this tape	No second record.

Second Record

If present, it is again headed with a code word (DMASK) followed by the numbers of words specified in MBANK+35:

Print Type	MB+35 must be set to:	This report written by:
0, Parameters	150[PARAK(50) + TEMPAR(50) + EVPAR(50)]	PPRINT
1, Pang	55P[P tracks, set up at beginning]	PPRINT
2, Vertex	$68+15P+(3P)^2$[$P \leq 7$ tracks fitted by GUTS]	PRINT
3, MMass	68[Missing-Mass quantities; see "Missing Mass"]	PRINT

Print Type	MB+35 must be set to:	This report written by:
4, HRJ	68[Hypothesis Reject]	PRINT
5, FRJ	0[File Reject]	PPRINT
77777_8	0[Special record to signify end of tape]	either routine.

Note that the first two sorts of reports are written automatically via a routine called PPRINT. The Parameter report occurs automatically on return from CML (thus is seen every time the program is loaded or alterations are read by CML); the Pang Report is written automatically by ZREAD when it starts a new event. The next three print types are written by PRINT when the command TSX PRINT, 4 is written in an event type subprogram:

The 68 words heading the PRINT second record are the sum of:

(1) The 52-word SNYDER bank in GUTS, containing fit parameters such as the number of constraints, number of steps, χ^2, etc. They are listed in the GUTS glossary at the end of the Banks-and-Glossary Section.

(2) Sixteen words of HLM (ignore these, or see Memo 86 for more details).

The Vertex Tracks in the PRINT second record consist of 15 words for each track that took part in the fit, listed in the same order that they were specified in the ZVERT calling sequence. Further details of these words can be found in the Track Bank Map and Glossary. The 15 words are:

Word	from ZTn+	Name
1	17	Track Number
2	18	Mass and Charge Code (Mev)
3	3	Status
4	4	Crn (Constraint Reduction Number)
5	14	Endwd
6	15	Lsz
7	16	Ftn
8	0	ϕ
9	1	s
10	2	k
11	24	KKT
12-14	GYFCN	Normalized steps (see Memo 86)
15	Code	DMASK = 0 77777 0 00000

The Variance Matrix in the PRINT second record is the 3P-by-3P variance matrix on the variables (ϕ, s, k) for the P tracks is written out. This matrix is stored in the order

$$\phi_1 s_1 k_1, \phi_2, s_2, k_2, \phi_3, s_3, k_3 \ldots\ldots,$$

where the subscript applies to the track order in the ZVERT calling sequence. Thus, we have the symmetric matrix:

	ϕ_1	s_1	k_1	ϕ_2	s_2	$k_2 \ldots$	k_P
ϕ_1	$\overline{\delta\phi_1\delta\phi_1}$	$\overline{\delta\phi_1\delta s_1}$	$\overline{\delta\phi_1\delta k_1}$	$\overline{\delta\phi_1\delta\phi_2}$			
s_1	$\overline{\delta s_1\delta\phi_1}$	$\overline{\delta s_1\delta s_1}$	$\overline{\delta s_1\delta k_1}$	$\overline{\delta s_1\delta\phi_2}$			
k_1	$\overline{\delta k_1\delta\phi_1}$	$\overline{\delta k_1\delta s_1}$	$\overline{\delta k_1\delta k_1}$	$\overline{\delta k_1\delta\phi_2}$			
ϕ_2	$\overline{\delta\phi_2\delta\phi_1}$	$\overline{\delta\phi_2\delta s_1}$	$\overline{\delta\phi_2\delta k_1}$	$\overline{\delta\phi_2\delta\phi_2}$	$\overline{\delta\phi_2\delta s_2}$		
k_P							$\overline{\delta k_P\delta k_P}$

Physical Length of Output

Pang Report (assume 10 track banks) 2 ft
Vertex Report (assume four track banks) 1 ft
HRJ 1/2 ft

A typical event may then occupy 6 ft of tape, or 1/400 of a 2400 ft roll. (Mnemonic: at a tape reading speed of 75 in./sec, EXAMIN can read an event in about 1 sec.)

Print Types 6 through 10

Print C and Examin have been modified to accept a new print type from Kick. It can be given any one of the print-type numbers 6 through 10, and it must be the last report of an event. It consists of two records: the usual 76 words of MBANK as first record, and a second record of less than 1000 words. A zero-length second record is acceptable.

In Examin, the first record goes into a list called FINAL and the second record is left in COMWS.

The report can be produced by the event-type programmer by means of the Kick routine called PPRNT. Before calling PPRNT, load AC and MQ as follows:

AC: HTR 0, 0, T MQ: HTR 0, 0, R,

where T is the print type number, and R is the reject type. The calling sequence is

α TSX PPRNT, 4,

α + 1 HTR FWA+L, 0, L (First Word Address, and Length of second record),

α + 2 Return.

Parameter Storage

There are logically several different types of parameters in Kick, (true constants are of course in the constant banks and are not dealt with in this note). Some of these parameters change only seldomly, such as masses of the particles, levels of significance at which tests internal to processing routines are set, etc. These parameters belong in a relatively permanent parameter-storage bank, which we will call PARAK, (parameters of Kick). PARAK starts at location 00100_8 and is a part of the "pure Kick" deck. The particle masses are actually dealt with in such a special fashion that a special name MASTBL has been given to the part of PARAK containing the masses.

This a 30-word bank defined by a BES order. The neutrals are the last seven entries in the bank. Particle masses can be called by the order CLA MASTBL, 1 where IR1 has the code for the mass. This code is as follows:

Code	Particle	Code	Particle	Code	Particle
1	γ or ν	11	e^+	21	e^-
2	π^0	12	μ^+	22	μ^-
3	K^0 or $\bar{K}^0$	13	π^+	23	π^-
4	N or $\bar{N}$	14	K^+	24	K^-
5	Λ^0 or $\bar{\Lambda}^0$	15	P	25	$\bar{P}$
6	Σ^0	16	Σ^+	26	$\bar{\Sigma}^+$
7	Ξ^0	17	$\bar{\Sigma}^-$	27	Σ^-
8	Λ^0 + N	18	$\bar{\Xi}^+$	28	Ξ^-
9	Zero(OCT 0)	19	D	29	Zero(-) (OCT 3)
10	Zero(+) (OCT 1)	20	T	30	He^3

Note that particles and antiparticles are duplicated if charged, since the last two bits contain the particle charge code (see ZTG-2 for explanation of this code).

TEMPAR: Some other parameters are used in processing routines in a consistent manner but with values that change regularly from experiment to experiment (such things are direction of magnetic field, target nucleus, beam momentum, etc.). These parameters are stored in a temporary

parameter bank known as TEMPAR . TEMPAR is defined behind PARAK in the pure Kick deck to define the symbols, but the values are assigned in the event-type assembly. Some of the most important entries must be copied into MBANK before output of each event.

EVPAR: Still other parameters are not used in a general way but are used by the event types for internal use (e. g., the value of χ^2 at which a given hypothesis will be rejected). From experiment to experiment, these parameters will change not only their values, but also their functions. This bank is known as EVPAR. The values in EVPAR and TEMPAR are assembled into the event-type deck.

For the immediate future we will continue to print all parameters--PARAK, TEMPAR, and EVPAR--both immediately on entering ZKICK (after CML) and any time a parameter is changed via PARCH cards or by Chapter-and-Verse (see Pang). Eventually PARAK need be printed only on first entry but will in any case be consulted less frequently because of the stability of its entries.

Rejects

In addition to troubles that are so serious as to stop the IBM 704, there are two distinct types of rejects:

1. File Rejects*

The Read Routines ZKICK and ZREAD make many format checks on the Pang Input. Failure of any of these tests leads to a File Reject (see below) which involves a "Type 5" trouble print on Tape 4, and an on-line print, after which the whole event (i.e., one file of input data) is rejected and the next file is read. In order to minimize output on events that may lead to FRJ, all tests that may lead to FRJ's must be written ahead of the Pang report in ZREAD. There are currently neither File nor Hypothesis rejects in ZSET.

2. Hypothesis Rejects

If ZVERT detects evidence against a single hypothesis, or if GUTS fails to get a fit to a vertex,† a "Type 4" or trouble print is made on Tape 4, χ^2 is set to 100, and control is returned to ZVERTX, the exit of ZVERT.

File Rejects are accomplished with the open subroutine FRJ which follows ZKICK. Specifically we write TSX FRJ, 4, n; where n is the type of reject (n will appear in MB + 37). Note that FRJ is not a closed subroutine; Index 4 is used only to pick up n, control goes to ZKICK 8 (to clear PUB and return to ZKICK 2 to read the next file).

Hypothesis Rejects are not handled by writing TSX HRJ, 4, n. This is partly historical, in that GUTS was written before HRJ. The scheme used is to write in the error return: TRA RJCTn. The list of RJCT's follows ZVERT in the assembly. Rejects 1, 2, and 3 are seldom used; after these the list runs

```
RJCT4 TSX HRJ, 4, 4  CHI. SQ. large on first step
RJCT5 TSX HRJ, 4, 5  too many steps.
```

Again, Index 4 is used only to pick up n; control goes to the ZVERT return in the event-type subroutine. In summary, HRJ is an open subroutine used by

---

*Called Event Rejects in the Rev. Sci. Instr. article.

†There may be many reasons for this "failure!"; see RJCT list - R - 4ff. Some, like $\chi^2 > 500$ (Reject 4) are evidence that the input data are inconsistent with the hypothesis; others, like "no such GUTS class" (RJCT 24) are clearly inadequacies of the program.

ZVERT to bypass a GUTS fit. GUTS accomplishes its own bypass by the sequence RJCT HRJ.

We want in the future to distinguish between rejects from Kick proper (using HRJ) and those from control subroutines which vary from experiment to experiment. We shall write a new entry into HRJ called ETRJ (Event-Type Reject), which will write a new print type.

List of File Rejects

Reject Type --Report type 5. (In Kick 4 these are 10^8-type Rejects.)

1. Event needed to pick up previous data and found none.
2. Pang omitted this event.
3. Too many track records in event or a track record has too few words.
4. Pang gave a negative word in ZTn + 54.
5. Had tape-reading error on some event.
6. Attempted to store when all pick-up banks were full.
7. ZSET8 found no "mate" in any ZTn for ZPUBn.
8. (Kick 4 only) Some neutral track was called stopping.
9. (Old 7-track Kick) p imaginary on Oc fit in event types 10 through 19.
10. Got L < R on initial swim of K^- beam.
11. (Old 7 track) Got p imaginary in phony GUTS - Vertex II.
12. (Old 7 track) Got p imaginary on a decay.
13. (Old 7 track) Got p imaginary on Vertex I of Type 52.
14. (Old 7 track) Got p imaginary in 20-21-22 Vertex II.
15. Free
16. Experiment number fails to check
17. + non-Pang data.
18. - non-Pang data.
19. ZPUB $\neq$ 0, but file has neither FTN nor PUC (nonconnected Event type buried in middle of connected Event types.)

55. Reject Report forced by manual TRA 377608 to get by program stop.

+63. (Kick 5 only) ZVERT was told to process a numberless track (In Kick 4 this is HRJ63, rather than FRJ 63).

Hypothesis Rejects

See the RJCT list following ZVERT, and, for more details, the next few pages (taken from Memo 86).

A special comment should be made about RJCT 20 (Missing mass entered with LC$\neq$4. In Kick 4, no Report is given). RJCT 20 merely sets the Missing Mass equal to 9999.99, sets GMFCN equal to MPRIME = 100, and returns to the ZVERT Exit routine.

List of Hypothesis Rejects

RJCT1: This is not used in the present version of GUTS.

RJCT2: Too many cutsteps. If on some step after the first, one of five nonanalytic constraints is violated, that step is rejected and the size of the step that gave the violation is reduced by 1/2 for a new try. If a total of 30 cutsteps is taken, GUTS checks the size of χ^2. For $\chi^2 > 25.0$, the event is rejected. For $\chi^2 < 25.0$, the fitting process is continued. If at any time there are more than 21 cutsteps between successful steps, the event is rejected. If more than 500 total cutsteps are taken, the event is rejected. Cutsteps come in the following five brands. These cutsteps are counted separately and occur in KICK PRINT.

GCUT1 Some k became < 0.

GCUT2 The incident track undershot KLOCD.

GCUT3 In 3C, P_ℓ^2 or $E_\ell < 0$.

GCUT4 In 2C, k_ℓ or $k_m < 0$.

GCUT5 The constraints have not yet been satisfied, and they failed to improve on the previous step.

GCUT is the sum of all these cutsteps.

RJCT3: $H^\nu_{\lambda\mu}$ was singular.

RJCT4: $\chi^2 > 500$ on the first step.

RJCT5: Too many iterations. If GUTS takes 15 steps without obtaining a fit, χ^2 is checked. For $\chi^2 > 25.0$, the fit is rejected. For $\chi^2 < 25.0$, up to 250 steps are allowed.

RJCT6: $\left| \frac{\chi^2_A - \chi^2_B}{\chi^2_B} \right| >$ CHECK. CHECK is set at the present time to 0.01.

The actual ratio is printed to KICK PRINT as DELTAM.

RJCT7: Some GYFCN $= \frac{\chi^\nu_i - \chi^m_i}{\{(EH^{-1}E^T)_{ij}\}^{1/2}}$ was imaginary. (A fit has been obtained.)

RJCT8: FNORM was imaginary.

RJCT9: Not in present GUTS (in earlier assemblies this meant that DELM was imaginary).

RJCT10: In 2C on first step, k_ℓ or k_m were < 0.

RJCT11: Not used in present GUTS.

RJCT12: G^{-1}_{ij} is singular.

RJCT13: A divide check occurred, showing that some quantity computed by GUTS was zero. The five-digit octal address of the command where the divide check occurred is found at RJCT in KICK PRINT.

RJCT14: Zero C, ZCCTR1 = 0, the curvature as specified by DAMN became negative.

RJCT15: P_μ was imaginary in zero C.

RJCT16: Not used in the present GUTS.

RJCT17: Not used in the present GUTS.

RJCT18: In the missing-mass calculation, $\overline{\delta M^2}$ is less than 0. Here M has been found and put into MMASS; $(|\delta M^2|)^{1/2}$ has been put into DMMASS.

RJCT19: Not used in the present GUTS.

RJCT20: The missing-mass calculation was entered with KLOCL $\neq$ 04.

RJCT21: Some input k equals 0 when it shouldn't.

RJCT22: Invalid constraint class (directly from ZVERT).

RJCT23: LC = 0 when forbidden (ZVERT).

RJCT25: ZCCTR1 $\neq$ 0 and both signs of DAMN lead to $k_\mu < 0$.
In zero C, for ZCCTR1 = 0, the discriminant is assigned the algebraic sign of DAMN. For ZCCTR1 $\neq$ 0, the sign of k_μ is checked. For $k_\mu < 0$, the other sign of DAMN is tried. If both $k_\mu < 0$, the event is rejected.

RJCT26: KPT failure (from ZREAD).

RJCT27: Zero C, KLOCD $\neq$ 0, incident particle undershot KLOCD.

RJCT30: Poisoned Track (from ZREAD).

RJCT62: KKT failure (from ZVERT after return from GUTS).

There are really two kinds of reject. The first types can happen frequently. These rejects are:

Reject 2, 4, 5, 14, 15, 20, 25, 27.

The remaining rejects should happen RARELY. These can happen only through rounding errors or misprogramming:

Reject 3, 6, 7, 8, (9), 10, 12, 13, 18, 21.

If any examples of these occur, please notify P. Berge.

Running Instructions

1. Hang Tape: (1) called Production Program Tape
   (3) Pang
   (4) a blank full tape, appropriately labeled
   (9) ditto. (Note: all tapes except 9 to be saved; 9 to be printed.)

2. Ready the Card Reader. It is necessary to prepare a card called the Shot card or Date card. It has the following format:

```
LO 23172    I   60   04   20   0001   Y
               ↑    ↑    ↑     ↑
             year month day  shot
```

All of the control cards in front of the shot card must be loaded every time the program is loaded.

3. Clear and Load Tape. Kick will select the card reader, read the control cards, if any, and the shot card.

4. Press Start. Kick should start processing and will fill Tape 4 in about an hour unless the input tape is so short that it comes to the end of that first.

5. To End Kick

   a. If time is up:

      1. Depress Sense Switch 2. The program will stop in less than a minute at the first permissible stopping place. The MQ should be full of bits. The SR will have HPR (Ord).

      2. Write down on the log sheet Ord, and the contents of AC. (This is the last event processed, so we can run again before everything is printed.)

   b. In case of machine trouble or severe program trouble and if for some reason Sense Switch 2 cannot be used as above, transfer manually to a program called "Crash" at 77770. This will end tape 4 and rewind it.

6. Kick Stops

   The following stops may occur while running Kick:

HTR 77772 HTR 77773	CML error return, check CML cards and try again. If it again stops, PHONE.
HTR 00077	Tape 3 gave check while reading parameters; restart program.
HTR 00777	Experiment number on event fails to check; check labels on tapes 1 and 3. If they appear correct, push start.

HTR 77701	Free
HTR 77702	Non-Pang data, use sense switch No. 2 to force end of Kick.
HPR 50005	CML advance use to searching tape No. 3, program found wrong serial number; push start.
HPR 55555	CML advance use to searching tape No. 3, program found correct serial; push start to begin production.
HTR 14245	Stop to enter or exit CML; push start.
HTK 70707	You are through. Successfully Kicked and Barbed all the way to the end of an input tape.

Trap Kick

A note on automatic running features used in KICK.

The Berkeley campus IBM-704 is equipped with the floating-point trap feature, permitting automatic adjustment of numbers resulting in exponent spills. If the result of a multiplication, for example, is too small to be expressed with an eight-bit binary-biased exponent, the machine can be instructed to "trap" this multiplication, and a routine can be devised to take corrective action (in this case, substitute zero for the answer). [See the commands EFM/LFM (enter/leave floating-trap mode in the IBM 704 manual)].

This feature does not affect the operation with regard to attempted division by zero. This can be detected satisfactorily only by a divide-check-test instruction, which allows one to take appropriate action.

Our assembly routine (4AP, see "Subroutines") has been modified so that it assembles FDH commands as FDP commands, and hence all FDH commands are eliminated from the binary program, though not eliminated from the listings.

Furthermore, nearly all of the program halts (HTR and HPR commands) have been removed from the program and replaced with two commands:

```
ETM      (Enter Trapping Mode... not Floating Trap Mode)
TXL   A, 0, B
```

where A = address of next command to be executed

B = address of eight words of BCD information which will be printed on-line.

Since register 0 is specified, it will be lower than B, so we actually have an unconditional (always trapped) unindexable transfer. This sequence is used, for example, to print out the event number and a comment to the operator in case an event is found on the input tape whose experiment number is incorrect.

Since trapped instructions must be interpreted anyway, it is no extra work to detect transfers to location 00001 that are not trapped. If this happens, the program assumes that serious trouble has developed. A comment on the on-line printer requests a Tape 8 from the operator. On this tape is written in binary, the entire content of memory. The Kick program is then reloaded from Tape 1, and proceeds to calculate from the next data on Tape 3. This means that one may use this instruction to produce

what amounts to a file reject:

```
TSX  1,4
```

The reason for setting IR4 is that the "trap" routine can then identify the location where the trouble occurred. Since the entire memory content is preserved on Tape 8, one can later dump this tape and reconstruct the trouble, if desired.

Wall

Consider a small-angle scatter in which the recoil porton is $\lesssim$ 1/2-mm-long and cannot be seen. The proton is measured as a "zerolength(Z)" track, and ZVERT sets "WALL" to a parameter, currently 0.05 cm.

The treatment of WALL is different for 1c and 0c "FITS". The 1c processing is described in Memo 86; if the fitted Z track has $R_{xy} > \text{WALL}$, GUTS automatically introduces the constraint $R_{xy} = \text{WALL}$ and refits by using the 2'c constraint class.

With "FIT" now destined to replace GUTS, we are reluctant to write a 1'c class to impose WALL on a 0c calculation which violates WALL. Instead GUTS defines χ^2_{WALL}. On Return from 0c, χ^2_{WALL} (in DELTAM) is 0 for $R_{xy} < \text{WALL}$; otherwise it is set to $[(k_{0c} - k_{WALL})/\delta k]^2$.

In Kick 6, 2'c is being removed. Instead, if WALL was to have been imposed, and if $P^{fitted} > P^{WALL}$, the k-Test function

$$k\text{-Test}_{WALL} = \frac{k^{fitted} - k^{WALL}}{\delta k^{fitted}}$$

is computed and stored in DELTAM. . This is then treated as the KKT by ZVERT. We wrote 2'c because of obvious advantages, but are now aware of the following disadvantages:

(a) For the 72-in. chamber, R_{xy} should be a much more complicated projection anyway;

(b) Sometimes a Z track is invisible because it <u>is</u> long ($R_{xy} > \text{WALL}$) but goes along another track;

(c) Consider the following, which happens occasionally: R_{xy} is set to 0.05 cm. A track whose R_{xy} is really 0.03 cm is calculated by 0c to be, say, 0.05 ± 0.03 cm, and the momentum parallel to the Z track is not well constrained. WALL is then invoked, and this momentum is recalculated by 2'c. Its central value is probably improved, but its calculated error becomes far too small. R_{0c} can now never get to 0.04 cm, and we have seen events then flunk when these unrealistic errors are propagated to other vertices.

CALLING SEQUENCES FOR PROCESSING SUBROUTINES AND SOME NOTES ON EVENT-TYPE WRITING

Below is a check-list of facts which should be borne in mind when writing a new event-type program, and a summary of the subroutine calling sequences.

ZTABLE and EXTABLE

These tables contain lists of event-type routines that are available in a given assembly. They must be filled in by the event-type assembly with the event-type in the address field and a negative branching address in the decrement field. Thus for event-type 14 we have the entry:

```
ORG ZTABLE
PZE 14, 0-ZET14
```

where ZET14 is the symbolic location at which processing starts for event-type 14.

EXTBL has the same format as ZTABLE but differs in that it can only be entered once for each event, whereas, if the "-S→L", charge-exchange feature is invoked by ZEND, control will return a second time to ZTABLE. Further details on the use of these tables are to be found in the ZREAD section of this manual.

Check that no Pang track is missing

It is advisable to include, at the very beginning of a 72-inch event-type program, a test which checks that the correct number of track-banks have been read in from Pang. When ZREAD transfers control to the event-type program, the number of track-banks read in from Pang is contained in index-register 2, which should be compared with the required number. If these numbers disagree, give FRJ63.

TEMPAR

The event-type programmer is responsible for the correct setting of the parameters in TEMPAR. The most important of these in Kick 5 are the following:

PBEAM	Beam momentum as floating point number in Mev/c.
DPBEAM	Error in PBEAM; also floating point and Mev/c
GAPL	Mean gap length between bubbles. This is used in EXTENDing zero prongs. Floating point in cm.
DGAPL	Error in GAPL
EXP.	Experiment number
DATE	Kick I. D.

BMAVG

A description of the beam-averaging routine is contained in the Auxiliary Processing Section. The point to be noted here is that, if the direction of the beam has $\phi \approx 90^{o}$, the instruction FAD ZT1+49 at ZSET YY must be overwritten with FSB ZT1+49 by the event-type assembly. The beam averaging routine in ZSET is coded specifically for a system in which the beam direction is approximately along $\phi = 270^{o}$.

ZSET

Summary of Function

ZSET extracts measurements (beginning or end, etc.) first from the ZTn and then from the PickUp Banks, and sets up the Kick part of each numbered ZTn for further processing.

Entrance:

a	TSX ZSET, 4, N_0
a + 1	OCT code word
a + 2	(Return)

Here N_0 is the number specifying mass of missing neutral. For $N_0 = 0$, ZT31 is not set up. Otherwise it is set up as an LSZ = 3 neutral at the end specified by bit 31 in the octal code word a + 1 with a mass specified by N_0. (See list in MASTBL in the assembly.)

a + 1: Sign bit 0 means beam average, 1 means don't beam average Bits n, for $1 \leqslant n \leqslant 35$ specify end at which ZTn is to be set up: 0 for end, 1 for beginning. Note that the neutral in ZT31 must be set up at the correct end. Note further that it is permissible to save thinking when writing a + 1 by writing more "1" bits than there are tracks, since ZSET does not set up numberless tracks. For example, a typical case with ZT1 incoming and beam-averaged and all other ZTn outgoing is most easily written

OCT 177777777777 BITS 001 111 ....., etc.

A vertex with missing neutral ZT31 incoming (and obviously not beam-averaged) would call for

OCT 777777000000 BITS 111 111 ... 000.

ZSET2B

This is a closed subroutine of ZSET to set up a single track. It saves all index registers. Before entering ZSET2B, set Endword of that track to indicate the setup wanted at the beginning (+) or end (-) of the track, and set CRN = 0. Address -ZTn must be in IR2.

Calling Sequence:

a	TSX ZSET2B, 4
a + 1	Return

UCRL-9099
CS-4
September 1960

ZVERT

Summary of Function

ZVERT takes Kick (set up) track banks and enters GUTS to fit a hypothesis. Upon return from GUTS, ZVERT does the following:

(a) If a fit has been obtained, ZVERT puts χ^2 into MBANK+15, transfers fitted measurements and variances from GUTS into track banks, sorts χ^2 against a standard set of χ^2 test levels (1%) and returns to event type. AC is (+) for "good"fit and (-) for "bad"fit. MQ is (-) if any KKT was (-) but not < - "4".

(b) For hypothesis reject, REJECT type (an integer in the address) is put into PRINTR, 500.0 is put into MBANK+15, the AC is set (-), and return is made to the event type. No information in the track banks is changed from the entry conditions, EXCEPT:

(1) If $\delta k/k$ exceeds "DKLVL" (~ 1), CRN will have been set to 1,

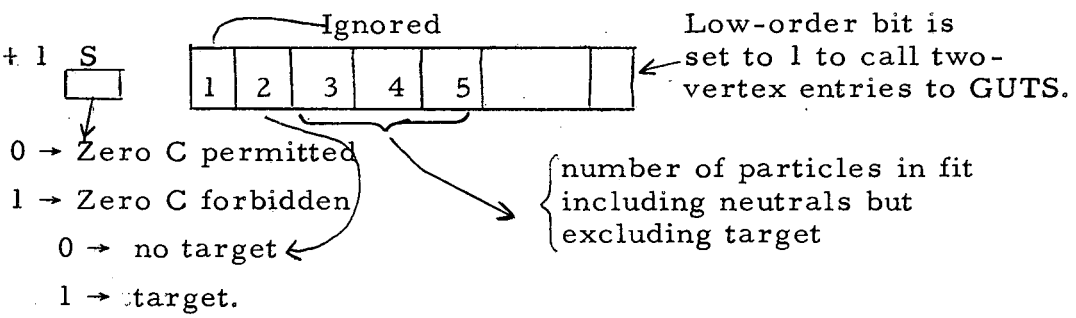
(2) If any KKT is less than - "4", leading to HRJ 63 after track banks have been changed.

Entrance

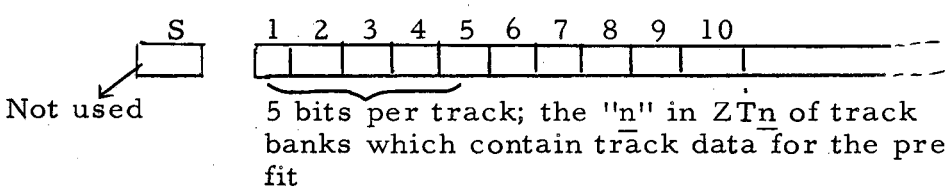
a	TSX ZVERT, 4, X
a + 1	OCT code word of vertex specifications
a + 2	OCT code word enumerating participating ZTn
a + 3	(Return, with AC and MQ as specified above)

X is normally zero; X = 16 specifies Missing-Mass calculation.

Code word a + 1



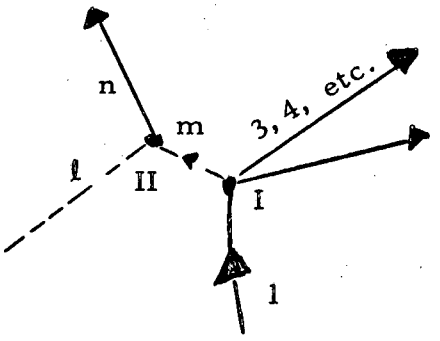
Code word a + 2



Special Requirements for Two-Vertex GUTS Entrances

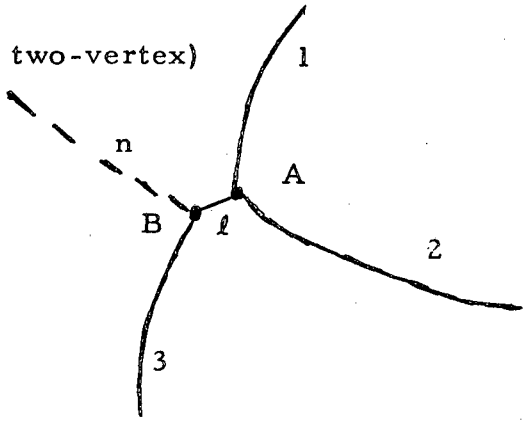
Special GUTS Entrances

2CZL2V (Two-constraint, zero-length, two-vertex):



Particles must be entered in the ZVERT calling sequence in the order 1, 2, 3, n, ℓ, m (where m is necessarily assumed leaving vertex I). For any 2V fit, the low-order bit of word a + 1 of the ZVERT calling sequence must be 1.

4C2V: (Four-constraint, two-vertex)



Specify particles in the order 1, 2, 3, ℓ, m. The following extra information must be filled into locations DD, PP, and TT (in ZVERT) for Kick 5.6 only.

Target at vertex B in TT

Number of completely measured particles in PP

Length of particle (if charged) in DD, otherwise zero in DD

Further, the sign of DD is (+) for ℓ incident to B.

The low-order bit of word a + 1 of the ZVERT calling sequence must again be 1.

Notes and Cautions

ZVERT uses the sign of ENDWD to find the incident track (-). Specifying > 1 incident leads to HRJ 28. Thus, the neutral track, ZT31, must be set up at the right end, and all tracks must be swum between vertices.

If the incident track is stated to be stopping, its mass is added to the target, and only the outbound tracks participate in the GUTS fit. After the fit, ZVERT sets the angles on this track to the measured end angles, k → 10, variances to 100.0, and covariances to 0. This treatment applies even to the special two-vertex GUTS classes.

PVERT

This is the alternative entry to ZVERT discussed at the end of UCRL-9098. If the locations PX, PY, PZ, and ENERGY in GUTS storage are filled before entry to ZVERT, their contents will be added to the constraint equations during the fit. The sign convention is (+) for an outgoing track. Thus suppose you want to force Track 2, outgoing from a proton elastic scatter, to have a momentum of 100 Mev/c in the y direction: Place in PX, PY, PZ, ENERGY the values 0, 100, 0, $(938^2+100^2)^{1/2}$, respectively, specify only Track 1 (incoming) and Track 3, and TSX PVERT. If you wish instead to force the incoming track to have 100 Mev/c, calculate the components of the four-momentum and change the sign of all components before planting them. In this case, of course, do not specify the bank on the incoming track; specify Tracks 2 and 3 outgoing.

ZSWIM

Summary of Function

ZSWIM transforms the contents of a set-up track bank to apply at the other end of the track.

Entrance

```
a       TSX ZSWIM, 4
a + 1   HTR 0, 0, -ZTn
a + 2   return.
```

(For neutrals, all that happens is that the sign of ENDWD is changed.)

VPRINT

(Vertex-Print, can also be called PRINT: SYN card is at back of VPRINT.)

Summary of Function

VPRINT reports Vertex fits and Hypothesis Rejects (prints types 2, 3, and 4) in accordance with specifications described under "Output". VPRINT decodes clues left by ZVERT and plants in MBANK the print type, length of second record, etc.:

Entrance

```
a       TSX PRINT, 4, X
a + 1   return.
```

Entry with the AC (+) gives good print, (-) gives bad print. (If a "good" report is made, GPM (=Good Print Mark) is increased by unity.) PRINT saves and restores the indices and AC condition. For X = 0, CPM (Check-Point Mark) is unchanged; for X ≠ 0, CPM is set equal to X.

PPRINT (Pang Print)

Summary of Function

A tiny report routine used automatically by:

ZKICK for the parameter report,
ZREAD for the Pang report, and
FRJ for File Rejects.

Uses COMMON+0, +10, +11.

Entrance

For Print Type T, Reject Type R: load AC and MQ as follows:

```
          AC: HTR 0, 0, T      MQ: HTR 0, 0, R
a         TSX PPRINT,4
a + 1     HTR FWA+L, 0, L      (First Word Address, and Length
                                  of second record).
a + 2     Return.
```

ZEND

(Not a closed routine, just TRA ZEND. Saves no index registers.)

ZEND normally returns control to ZKICK2 to read the next event; alternatively failures with a negative S track may be channeled down the charge-exchange branch and reprocessed via ZREAD6.

Special care must be taken with a connected scatter in which more than one hypothesis is tried, as each set-up over-writes the fitted quantities in the ZTn banks. For such an event it is necessary to store the results of each fit (except the last one) and then to restore them to the appropriate ZTn banks immediately before the transfer to ZEND.

For example, consider an ambiguous π^-p, K^-p scatter in which ZT1 and ZT2 contain the π^- and K^- respectively. The easiest procedure is to use QVRT. First try the K^- hypothesis designating ZT2 as the track to be saved automatically in the Fitted Banks.* Then use SAP coding to decide which hypothesis to accept, transfer the K^- results back to ZT2 if necessary, and poison any unacceptable ZTn banks. ZEND will then transfer ZT1 and ZT2 to Pick up Banks provided that at least one fit was good. If no hypothesis succeeded and it is desired to pass on Pang data, do a set-up and set GPM $\neq$ 0.

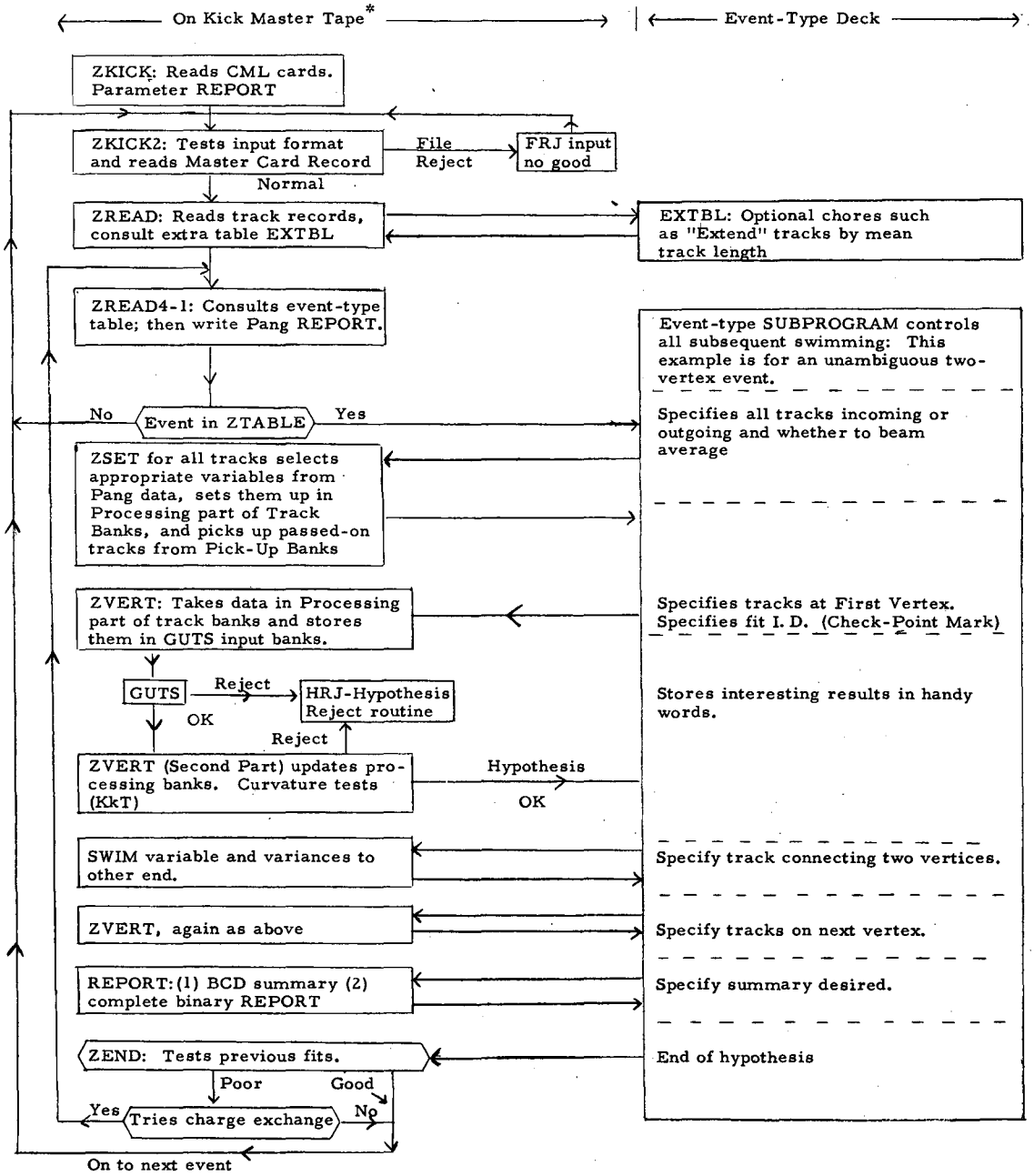
If the event-type program does not use QVRT, the K^- results in the above example may be stored in a PUB while the next fit is being done.

---

*The π^- results need not be stored as this will be the final fit. Note also that QVRT never stores ZT1 in a fitted bank; if ZT1 is the first track bank designated in a QVRT entry it is the second track that is saved. This restriction would be relevant to the case of a π^+p scatter in which two hypotheses must be tried but the connected track is in ZT1 for both. This difficulty can be overcome by transferring ZT1 to ZT11, say, for the first fit, and having QVRT save ZT11 in a fitted bank during the second set-up and fit.

MAIN PROCESSING ROUTINES

Summary of Kick Flow



*Kick Master Tape includes Subpak and pure Kick. See "Assemblies."

Notes on ZKICK

General Description

ZKICK has two main functions: it is the starting routine for KICK which sets up the 704 to receive the first event, and it is the routine which begins the reading of each new event from the input tape. (See the first flow diagram labelled "Summary of Kick Flow," and the ZKICK flow diagram.)

On the initial entry ZKICK reads in, from cards, any required program or parameter changes for the current run. On each "new event" entry the first record for each event is read. This is the Master Record, and contains a variety of identification numbers and marker flags relevant to the event. It is checked for format to identify the chamber to which the data belongs, and some master record data is decoded and stored; storage space is cleared before control is passed to ZREAD which reads in the remaining records of the event which contain the track data.

ZKICK responds to sense switch settings which can either temporarily halt the processing after an event or force an end to the KICK run. In this latter case the (binary) output tape is rewound, and it can then be read by any of the programs Examin, PrintB, or Print C to produce a BCD tape.

ZKICK also keeps count--the ordinal count--of the number of files (i.e., events) read during a run, and it detects various marks on the input tape (e.g. the end-code-word on Tape 3). In addition, there is an error entry to ZKICK which causes the binary program tape to be retroacted into core for a fresh start.

The following remarks are to be read in conjunction with the flow diagram; the marks in brackets on the flow diagram refer to the numbered remarks.

Detailed Description

The various entries and salient points in ZKICK are:

ZKICK - Initial entry point; flow goes from ZKICK to ZKICK 2 once per program load.

ZKICK 2 Entered each time a new event is to be processed. May be entered via ZKICK 8 which clears Pick-Up Bank, or directly.

ZKICK4-1 An end-of-input-tape routine which is used to rewind all tapes and terminate Kick processing.

ZKICK 5 Begin reading new event off input tape; commence format tests on first word of file.

ZKICK 6 - Master record read in.

ZKICK 7 - Return to read in next event when pickup banks are to be cleared.

1. SQERR - RPERR are two error entries (Imaginary square root called for and range-momentum table overshoot, which cause the program tape to be reloaded.)

2. WIMPY - WILLIE are two entries to be short sub-routine which perform the same functions as performed by pressing "Load Tape" and "Load Cards", respectively.

3. Comments - some versions of Kick contain a variety of HPR, HTR stops. Wherever COMMENT appears, the stop at this point may have been replaced by an on-line printed comment. For fuller details see sections entitled "Trap," "Stops," "Rejects."

The Comments invoked during ZKICK are, together with the stop instruction which they may have replaced:

1.	Square-root error	HTR WIMPY
2.	Range-momentum error	HTR WIMPY
3.	No Kick ID number	HTR ZKICK
4.	SS1 Stop	HPR 0
5.	SS2 Stop	HPR 0
6.	+ve more Pang garbage (ZKICK cannot recognize input at all)	HPR =77701
7.	-ve more Pang garbage (ZKICK cannot recognize input at all)	HPR =77702
8.	RTT test fail at PARCH card pad (PARCH cards are PARameter CHange cards, see INPUT - 1 -.)	HTR WIMPY
9.	Error return from CML pad return	HTR=7777$_z$

4. TAPES used by ZKICK are described under "Tapes." In summary:
   Tape 1. Self-loading binary Kick program tape
   Tape 3. Binary input data tape, previously output by PANG
   Tape 4. Binary output tape.

5. CML cards-cards read by Card Mixed Loader routine (see Subroutines Section) which allows numbers, instructions or BCD to be read to specified locations. The convention adopted here is that a Y-termination on a CML Card causes return to CML, while W-, Z-, or EOF (i.e., no cards in

hopper) terminations cause return to main routine. The first operation of ZKICK is to read a set of such cards to fill in any new values of constants, and any program modifications required, together with the serial numbers of any events which are to be searched for (see ZHUNT feature of ZREAD routine.) It is required that a date must be read in with the CMLs; this is

6. KICK IDENTIFICATION number, which is stored in location DATE (follows DCONS) see also Section called "Identification Numbers." DATE is set at 99 in the program, and if an ID # has been read, it can be detected by finding DATE $\neq$ 99.
7. Scale Factor-A triple entry range-momentum time table is stored with the sub-routines in SUBPAC (see section on Assemblies). For hydrogen and deuterium the shape of the range/momentum curve is assumed the same, but the absolute values may be changed by means of a scale factor read in from a CML card to temporary location RSCALE. ZKICK then plants it in a permanent location, called SCALE, in the subroutine deck.
8. Parameter Report-So that the binary output tape may be read by a FORTRAN program, the output format used by ZKICK is standardized. Output is in units of "Reports" made by the subroutines PRINT, PPRINT or VPRINT, and the output of an event may consist of several reports. Each report contains one or two records; the first always consists of the 75-word MBANK (certain words in MBANK specify the type of report being made, whether there is a second record in the report, and if so what is in it etc. (for details see section on Output.) MBANK+26 $\neq$ 0 means that this is the last report of a given event; ZKICK makes an initial report via PPRINT of the contents of the MBANK and the Parameter Bank, (PARAK), making an end-of-event mark to indicate that no more associated information follows.
9. Serial number- This is stored in MBANK + 11, but before reading in a new event, ZKICK stores the serial # of the last completed event in location GINOUT (part of GUTS storage) so that it can be examined later by ZREAD, which will be interested in whether the serial number has changed or not.

10. Connected events-if an event has been measured in two parts, the possibility of a sense switch test stopping the processing between the two parts must be avoided; a test is made to see if the contents of the first word of the first Pick Up Bank is zero or not. If the event is connected, the carried-over information is stored in these banks so this word will $\neq 0$, and the sense switch tests are skipped.
11. Sense Switches ("no effect" position defined as up)
    SS1 -down stops program after current event to allow debugger to take stock. Continue processing by pressing start. Just before the stop, routine ZWAIT makes a visual display (see below) and an on-line comment is printed.
    SS2-down, the Kick processing is brought to an end after the completion of the current event. An on-line comment is made and the output tape rewound.
12. Visual Display-Subroutine ZWAIT is usually followed by a stop so its calling sequence is:

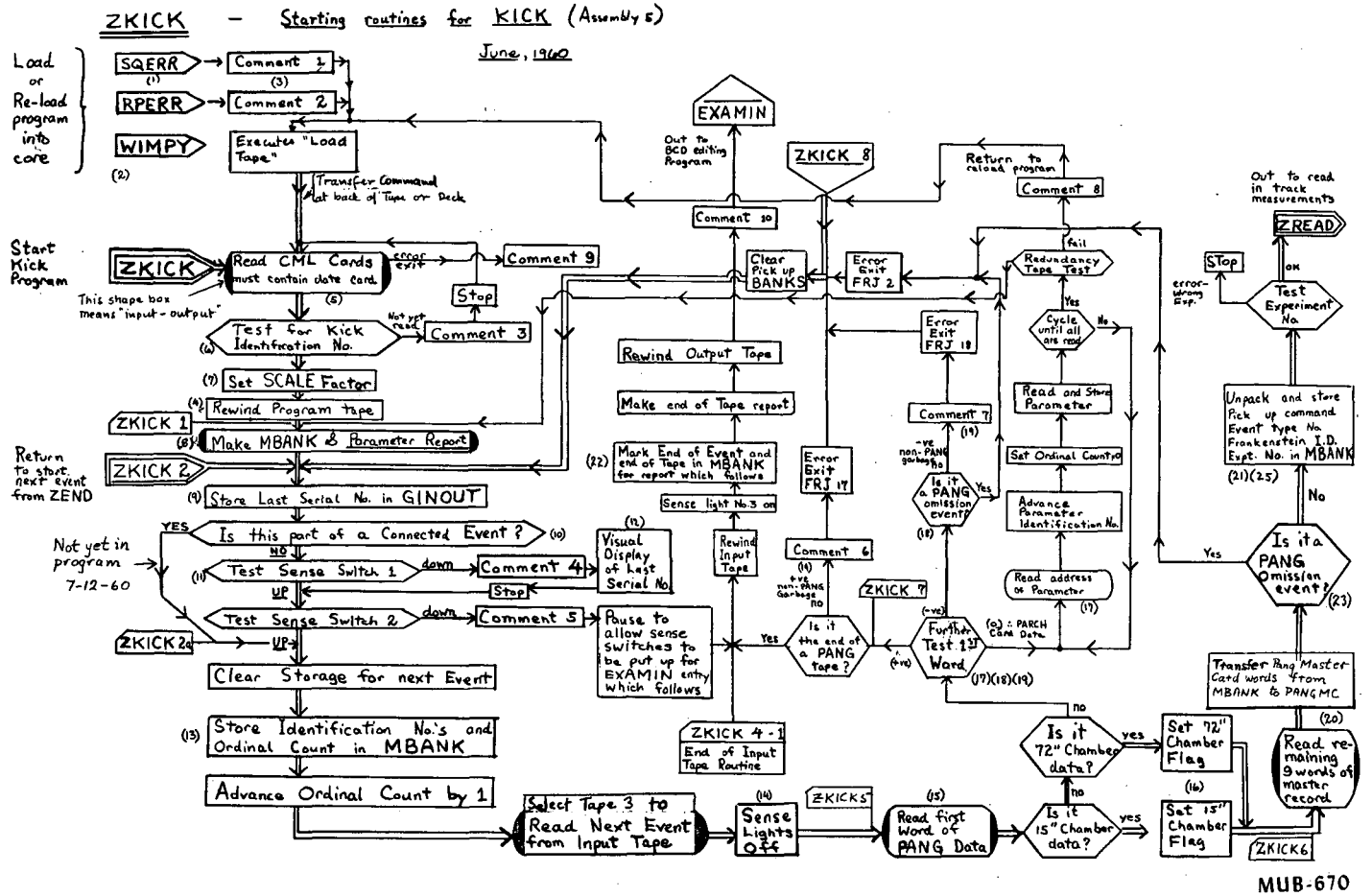
a	TSX ZWAIT, 4	(exit)
a+1	HPR 0	(return)

    The ordinal counter (see below) is loaded into the address part of (a + 1) by ZWAIT, and the serial number of the last completed event is copied from GINOUT to the AC. When the machine stops as the HPR in (a + 1) is obeyed, the operator has a visual display of these numbers.
13. Ordinal count-This is an integer which counts the number of files (i.e. number of events) read during the current Kick run; it is stored in the address of the location ZORD HPR 0 in the assembly. Just before a new file is read, it is advanced by one and copied to the decrement part of MBANK + 33. It is reset to zero if a PARCH card is read (see below).
14. Sense Lights-Whenever an end-of-tape mark is recognized, the corresponding sense light is turned on; in this respect forcing an end to the processing by SS2 is equivalent to an end-of-tape mark on tape 3.
15. Pang Data Format-The first word of the Master Record of an event is read to MBANK + 2. The word is tested for a Pang identification number, made up of BCD words, whose format determines whether 15-or 72-in. chamber data follows. For the 15 in., the format is ~~xxbxx~~ where b indicates a BCD blank 60_8, and x indicates any BCD character. The word is masked with BLK15A OCT 770000 770000 to select the first and third

BCD words, and this is tested by subtracting BLK 15B OCT 60 00 00 60 00 00 to see if they are blanks. For the 72-in., the format is bxbxxb and a similar test is made to identify it. If it is not a Pang ID number, the possibilities in 17, 18, 19 below are tested. These cases are summarized in the Input section.

16. Chamber flag-If the identification word corresponds to the 15-in. (72-in.) chamber, the sign of location ZKICKD is made +(-) as a marker.
17. PARCH cards-If MBANK + 2 = -0, then PARCH card data follows. These are Parameter Change Cards incorporated into the Frankenstein deck are copied to the Pang output tape. They are read by Kick thus allowing parameters to be changed during a Kick run. The new parameters are read into their specified addresses until an EOR is found. The parameter identification number, PARID, is advanced by one for each change of parameter, and the ordinal count is reset to zero.
18. Pang omission-If MBANK + 2=-1, this means Pang has omitted to process this event for some reason (e. g. errors of measurement). If this is found, an error exit FRJ2 is made and the next event read in.
19. Non-Pang input-If neither chamber has PARCH card data identified, then the record contains rubbish, classified positive or negative depending on the sign of the first word. Plus or minus nonzero in MBANK+2 gives error exits FRJ 17 or FRJ 18 respectively.
20. Master record-The 1st word of the master record from Pang has been read to MBANK+2 at ZKICK5+1. The remaining 9 words are read in by a copy loop at ZKICK6 into MBANK+3 to MBANK+11 using IR1 (previously set at ZKICK5) to count the entries. This completes the reading of input done by ZKICK.
21. Pick-Up Command-A marker (one odd octal digit) is used in the third BCD character of MBANK+8 to indicate if an event is connected to the previous measured one. This octal digit is unpacked and stored in MBANK+12 ready for examination by ZREAD3.
22. End-of-tape report-Just before rewinding tape 4, the subroutine PRINT makes a report of one record only; this is MBANK with 0 77777 0 00000 in MBANK+26 to indicate end of event and end of tape.

23. Pang omission test - Just before exit to ZREAD, which tests last BCD of MBANK + 6, is an addition made so that Kick will be ready for some preferred changes in Pang output when these are ready. For Kick 5 this test can be ignored, as Pang omission events will be found earlier in ZKICK by -1 in the 1st word of event record.
24. ZKICK - Starts at 10616_8, destroys contents on entry of all index registers, and makes the following error exits: FRJ2, FRJ17, FRJ18.
25. Other words - Other words stored in MBANK at this stage are event type, Franckenstein ID which are transferred to the decrement (for later use by Fortran programs as integers.)
    The experiment number is examined for format, after testing ZKICKD, the chamber flag. For details see MBANK glossary.
26. ZNUTS - Error routine, reads input tape to end-of-file, and exits to error FRJ 63. Called in whenever tape 3 has too short (or too many) records for an event. Press start to try next event.



MUB-670

UCRL-9099
ZKICK-7
June 1960

Notes on ZREAD

The basic function of ZREAD is to read, prepare, and check the data for the computational parts of Kick, and to make the Pang Report (of the unfitted Pang Input).

Reading Output of Pang

ZKICK has read the Master Record of the event in hand. The track records, one for each mass assignment of each track, must now be read into the track banks. The nth bank is denoted ZTn. Pang output is loaded into ZTn+17 to ZTn+54. The total number of track banks, denoted by NTB, at present equals 31. ZT31, however, is reserved for missing neutrals, so that up to 30 track records can be read in from Pang.

Before reading, location COMMON is set to zero for a purpose mentioned below, and IR2 is zeroed to count track records.

ZREAD +2. IR4 counts the words in each track record and is tested for exit from the copy loop that reads the data. An end-of-record mark on tape 3 causes a different jump out of the loop. This jump means a too-short record (because the end of a record of proper length will be detected by the word count) and causes a transfer to the error routine ZNUTS in ZKICK.

At the end of a proper record, the Pang test word ZTn+54 is tested. If Pang has found inconsistent curvatures in the two views of the track, this word will be negative, and COMMON in turn is set negative by ZREAD. Next, the Future Track Number and LSZ information is unpacked and stored in ZTn+16 and ZTn+15, respectively. The details of unpacking depend on whether the data come from the 15-or 72-in. chamber. Integer data, such as the track number, are shifted from the address to the decrement bits (with a view to making the Kick format compatible to FORTRAN format), and certain correlation terms on the track error matrix are zeroed.

If the track is not "poisoned", all angle data are also converted to radians.* Finally, the track count in IR2 is tested. If there are more than NTB-1 records, ZNUTS is invoked. If not, ZREAD starts another track (ZREAD+2).

---

*Poisoning is a method used by event-type routines to transmit information resolving ambiguous mass interpretations of pickup tracks. If successful fits cannot be obtained using certain mass assignments, the corresponding track banks are marked with the "Poison" flag so that they will not be tried in fitting the main event to which they are connect. As an example, a track might be *a priori* either a pion or proton, but is identified as a pion by kinematical analysis of a second scatter. Then the proton track bank is "poisoned."

An end-of-file marks the end of the event and causes a jump out of the loop to ZREAD 3.

ZREAD 3. A tape redundancy test is made to look for reading errors. (A tape error leads to File Reject 5.) Then the Pang test results in COMMON are tested. If any track was rejected, COMMON will have been set negative. At present, the negative sign is ignored. It will lead later to a rejection of the event when certain adjustments are made in Pang.

The serial number of the last event processed (in GINOUT) is examined. If it matches that of the new event, there may be pickup data. If it does not match, however, the pickup banks are cleared (because we have a new event) before continuing.

Pickup Test. The Pickup Command PUC in the Master Record is now tested. If absence of a low order bit indicates no pickup, control jumps to ZHUNT (SYN ZR3A). Otherwise, the track numbers in each of the pickup banks in turn are compared with those in the data banks. If a match is found, the mass-and charge words are also compared. If they match, i. e. if any two tracks "mate" so that a track can later be picked up, all is well, and control goes to ZHUNT. If no "mate" can be found, File Reject 1 is invoked.

Notice that the data in the PUBs is only checked, not picked up. The actual pickup is not done till the end of ZSET, but the check is made earlier to avoid a Pang report and other wasted time.

ZHUNT. The quantity SERHNT is examined. If it is zero, control jumps to ZR3C, and processing begins. If it is nonzero, it is the serial number of the next event to be processed. It is compared to the serial number of the event in hand. If SERHNT is smaller, then the events are out of order on the tape or we have somehow gone too far along the Pang tape. An on-line comment and halt are made. On restart, control returns to ZKICK 2 to read another event.

If SERHNT is greater, control goes to ZRD10, which looks to see if any tracks in the event in hand are to be picked up (FTN$\neq$0) and, if so, have present TN of 1. Those that do are set up at the end. Control then goes to ZEND to transfer pickup data to the PUB's, then to Kick 2 to read another event, continuing the search.

If SERHNT matches, control is sent to the CML card-reading routine. CML will read any data that should be loaded for this event, followed by either the serial number of the next event to be hunted, or STOP to end

processing. After exit from CML, an on-line comment is printed, and the selected event is processed starting at ZR3C.

If ZHUNT is not to be used, it is not necessary to zero SERHNT with a CML card. ZKICK sets it to zero before beginning the run.

CML cards for ZHUNT use the "W" return feature of CML, and are prepared as follows:

(a) Start punching in column 1 (Let NNNNNN equal a serial number).

(b) First event, first card: W1+NNNNNN

(c) Any special data or instuructions for processing this event. Terminate each card with Y. See the CML writeup under "Subroutines".

(d) Next card: W1+ followed by serial number of next event to be hunted, or

(e) by 000000 (6 zeros), or by nothing, if it is desired to process all events following the last hunted event, or

(f) by STOP if it is desired to stop after processing the last hunted event.

(g) The general rule is to stack first a "W1" card, then data or instructions for that event on "Y" cards, and so on, ending with W1+STOP, W1+000000, or W1.

Example: To start a run at event 567890, the CML cards for ZHUNT (at the end of the CML deck) would read:

```
W1+567890          (Hunt to here)
{W1+000000}        (Continue processing).
```

Event-type Lookup and Pang Report (Starting at ZR3C).

Next we consider two routing tables, ZTABLE and an extra table of the same format, EXTBL. They appear in the pure Kick assembly after the track-bank reservation as follows:

```
        HTR 100          (Length of table, for use in search loops)
ZTABLE  BSS 100          (Define and reserve space)
        HTR 25
EXTBL   BSS 25.
```

The tables are filled in by the event-type assembly with the event type in the address and a negative branching address in the decrement. Thus for event-type 14, we have the entry

```
        ORG   ZTABLE
        PZE   14, 0, -ZET14.
```

EXTBL and then ZTABLE are entered once during ZREAD. If the event fails later, however, and the "-S = > L" charge-exchange feature is invoked, control will return again to ZTABLE, whereas EXTBL is never entered a second time.

Every event that is to be fitted must be defined in ZTABLE so that control can go to the event-type subprogram (which is located in our example at ZET14). EXTBL need not be used, but is available for some extra operations.

EXTBL has so far governed useful operations such as:

(a) In some runs it has allowed selection of events within a fiducial volume of the chamber. The selection depends on event type, because some types contain several vertices, and the proper one must be tested.

(b) Tracks with zero-prong endings should be extended one-half mean bubble spacing before their end-points are used to compute the direction of neutral tracks. (If the ZTABLE controlled the extending routine, a track might get extended several times because the ZTABLE can be entered more than once per event, e. g. by the "-(S) Return" from ZEND to ZREAD 6.)

(c) The ZSET (a + 1) calling-sequence word at ZREAD9+2 preceding the Pang report may be changed to get beam averaging of a track before giving the Pang report. Normally this word is restored by RESTOR in ZR3C, which is where EXTBL routines should return. If a+1 has been changed, however, EXTBL must return to ZREAD4-1.

Normally the event is defined in ZTABLE; the logic is straightforward, and the reader should first read the flow diagram to the end (reading these comments in the following pages, starting at Flow for Events Defined in ZTABLE, but ignoring ZRD10).

Next we must consider two "abnormal" cases. In both the event is not defined in ZTABLE, but in

(a) Case 1 we do not want a Pang Report, and

(b) Case 2 we do want a Pang Report. This is the way ZREAD runs if not overwritten.

In both cases flow goes through ZREAD8.

An example of Case 1 might be as follows: We have just modified some subprogram, say type 14 (ZET14), and wish to process all type 14's, but nothing else, on an input tape. However there is the problem of connected events; some of the type 14's may have been measured with one track missing

because that track was connected to a scatter that was supposed to be fitted first. Therefore, nothing else involves some exceptions.

The most complete treatment is to define also in ZTABLE all elastic scatters to which type 14 might be connected. For example, in one experiment type 14 happens to be a Kp scatter and therefore can be connected to either another type 14 or to type 15, which is pp elastic scattering. For simplicity, let us assume that it cannot be related to anything else. If these two types are defined in ZTABLE, then we need truly process nothing else. Thus, we can simply overwrite the present command in ZREAD8 with TRA ZKICK8 to clear the pick-up banks, and continue looking for a defined event type. However it may be that some type 14's are also connected to a complicated event type which is not even written yet, so that the easiest thing to do is to pass on the Pang track data on all undefined events with tracks having future track numbers. This is accomplished by a short loop called ZRD10, which is described at the end of this section. ZRD10 sets up these tracks at the right end and then transfers to ZEND.

The Pang report may be bypassed by depressing sense switch 6. However we are trying to get Kick to run without having to tell the operators to set sense switches, and the sense-switch tests will eventually be removed. Hence we recommend bypassing the Pang report with CML cards. To simulate sense switch 6 down, merely write NOP over the TRA *+2 that follows the command SWT 6.

If sense switch 6 or overwriting is not used, ZREAD will give a Pang report on undefined events. This might be useful on a run in which we are processing most events types anyway and want a record of everything that is on the Pang output tape.

Flow for Events Defined in ZTABLE(Starting at ZREAD5-1)

If the type is in the list, IR4 is set to the proper event-type subprogram address (in our example, to ZET14). The table entry sign indicates whether the bias χ^2 values are to be set to zero (-) or to the standard values (+). ZREAD 7. The bias χ^2 numbers are used to determine whether additional hypotheses are to be tried on certain event-types even if the first hypothesis succeeds. For example, in the 15-in. Experiment 1, event-type 30 (KSF) is to be tested as a stopping K^- (KS). If GUTS finds a χ^2 exceeding the bias level, interaction in flight (KF) is to be tested also. On the other hand, event-type 40 (KS+F) uses the same event-type routine but is always tested

for both KS and KF interaction by setting the bias values to zero. In summary, the sign of the event-type listing word in ZTABLE determines whether the working biases are to be overwritten with zero (using ZREAD7) or left at their present values in ZCHBM (ZCHi-squared Bias Main).

ZREAD9. Before control can go to the proper ZET address in IR4, IR4 is saved, and a Pang report is invoked. Then control goes finally to the ZET subprogram.

The Stopping-Negative Return

ZREAD 6. Ordinarily, the branch through IR4 at the end of section ZREAD5 ends the operation of ZREAD. If, however, routine ZEND later finds that no hypothesis succeeded, it looks to see if any tracks were negative and were called "stopping" when measured (i. e. were -S). If it finds a -S track, it changes it to a leaving track (one that did not stop) on the chance that it charge-exchanged, then redoes the event by returning to ZREAD at point ZREAD 6. ZREAD 6 clears various working storages, then returns control to normal processing sequence at ZREAD 4-1.

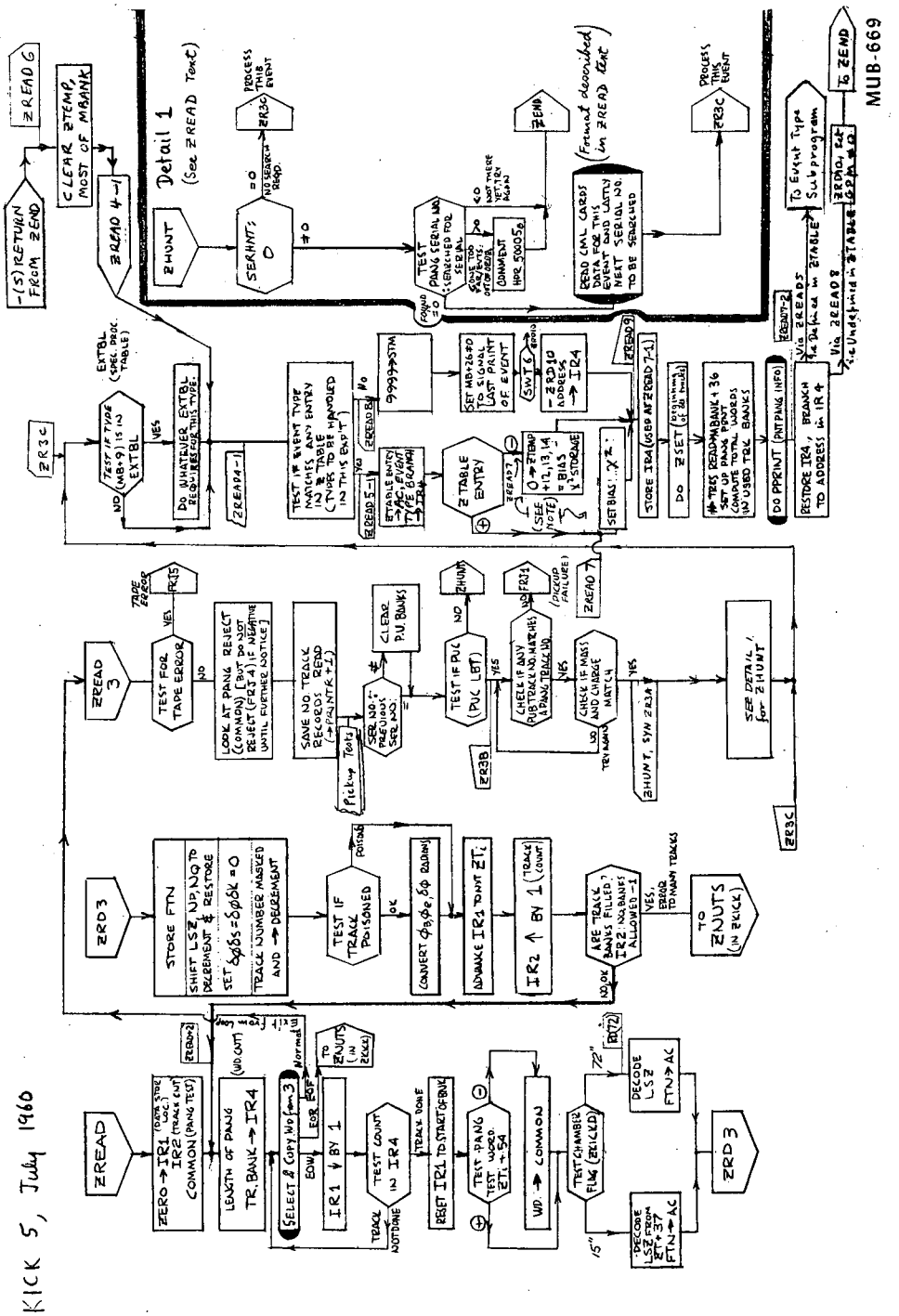
ZRD10. Special Set Up for Undefined Events, with Tracks Having NonZero FTN's.

This routine is straightforward and is not included on the flow diagram. It first sets the Good Print Mark (GPM $\neq$ 0) so that ZEND will be fooled into storing any track with a FTN. (The reader unfamiliar with the function of GPM should read the first paragraph of the notes on ZEND.)

In the routine we assume that tracks with nonzero FTN's should be set up as follows:

Present track number	Set up at	Comment
1	end	ZRD10 must re-set up.
all others	beginning	Already set up properly.

Note that if a track numbering convention should ever be chosen such that tracks other than 1 are incoming, you cannot rely on ZRD10 but will have to write some more general logic which tests the coordinates of the ends of all tracks and determines which ones are attached to a vertex.



KICK 5, July 1960

Notes on ZSET

General Description

ZSET is a closed subroutine of Kick, designed to set up the Kick portion of every track bank (at that end of the track specified in the calling sequence for ZSET) by using some of the Pang data from the Pang region of the track banks. These data consist of direction at both ends of the track, (ϕ_b S_b, ϕ_e S_e; b = beginning, e = end) and the curvature at the middle (k_m). Since most further Kick processing routines operate on information stored in the Kick region of the track banks, ZSET must be called very early in each event type. If the event is of sufficient complexity, ZSET may be invoked several times in the event type. (Each time measured instead of fitted data must be used.)

ZSET's job is to set up tracks with either "beginning" variables (if the track is outbound) or "end" variables (if inbound). We prefer, however, to think of the dynamics and state in the calling sequences whether the track is "In" or "Out". Specifically: "Out" tells ZSET to set up all variables at the beginning. But "In" does not mean set up all variables at the end, since GUTS will use the D-hold feature; instead angles are set up at the end, and k at the middle. ZSET calls on the subroutine KSWIM (see ZSWIM notes) to transform the measured curvature k_m to the beginning value k_b if required.

In addition ZSET does the following things that are not strictly connected with setting up measurements:

(I) If the calling sequence specifies a beam track, the momentum measurements of the track in ZT1 are averaged with a beam momentum stored in the program as a parameter. The resulting average is put into the Kick region of ZT1. If the track stored in ZT1 has a future track number, it is not beam-averaged. This allows connecting upstream scatters as connected events without complicating the event type.

(II) The last track bank (ZT31) can be set up as a completely missing track, with a neutral-particle's mass as specified in the calling sequence.

(III) After setting up all tracks, ZSET collects all information passed on from previous events for the current event from the Pick-up banks.

(IV) ZSET computes KPT ("K-Pang-Test"), one of three simple test functions on the internal consistency of the curvature information for each track, and stores this function for later reference (see note on K-Pang-Test).

(V) If something is radically wrong with the format of the event, ZSET may reject the event.

The general form of ZSET is shown in the first flow diagram; the routine may be divided into three separate sections. The "Preliminary Part" saves the indexes, a small loop cycles to clear the Kick parts of all track banks, and sets the values of two control words STATUS and ENDWD in each track bank. These words are used to indicate, respectively, the stage that the processing of the given track-bank data has reached and the way in which the variables have been set up. (For details see Track Bank glossary, and notes 2, 3 below.) Finally this part does the beam averaging on the beam track if required by the ZSET calling sequence.

The "Set-up Part" contains one large loop which sets up the variables in each track according to the ENDWD in the track bank, and manufactures the K-Pang test quantity for each track and stores it in the track bank for future use (by ZVERT for instance).

The "Pick-Up Part" examines the pick-up bank, and if there is any track previously measured which is to be connected into the present event, the necessary transfer of data to the track bank is made.

More detailed notes on ZSET follow. The flow diagram is divided into the three parts described above; numbers in brackets on the flow diagrams refer to detailed notes in the text.

Detailed Description*

(1) ZSET entrance and exit conditions. Notice the structure of the following type B 704 commands as translated by the assembly program (the octal address corresponding to ADDress is $xxxxx_8$):

TSX ADD, 4 = 007400 4 $xxxxx_8$

TSX ADD, 4, 8m+n = 0 074mn 4 $xxxxx_8$.

There are 6 bits (mn_8) available in most of the Kick subroutines calling sequences. In ZSET we use these two extra octal digits to control the setup of the last track bank (missing neutral).

CALLING SEQUENCES:

α : TSX ZSET, 4, N_0

α + 1 : OCT code word, referred to as "α+1" below and in the flow diagram

α + 2 : normal return.

ZSET preserves all indices; it uses all Common presently available (25 words).

*Numbers in parentheses refer to identical numbers in the flow chart.

α : N_0 is the code number specifying mass to put into ZT31 for the missing neutral;

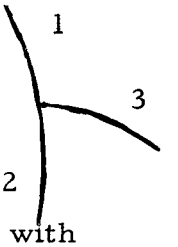
$N_0 = 0$ means we are not interested in ZT31, set its track number to zero;

$N_0 = 1$ means mass = 0 (γ or ν); 2 means π^0, 3 = K^0, 4 = n, 5 = Λ^0, 6 = Σ^0, 7 = Ξ^0 [all from MASTBL-(N_0 - 1)]

$\alpha + 1$: A(1/0) bit means set up as (outbound/inbound) and the nth bit refers to track bank n. The first track bank requires an extra (preceding) bit (the sign bit), for which 0 means beam average, and 1 means don't average.

All measured tracks are set up.

Example: Consider the sketched scatter: If track 1 is to be beam averaged, we have $\alpha + 1$ = .0011....; if it is not to be beam averaged, we have $\alpha + 1$ = 1011.....



$\alpha + 2$: Normal Return. Provided nothing important was wrong with the format of the event, ZSET returns control to ($\alpha + 2$) of the main program (i. e. event-type subroutine) with all track banks and ENDWD's set up (see bank map).

(2) STATUS : in ZTn + 3

Sign (+/-) means (fitted/unfitted). Magnitude set = 1 in ZSET.

(3) ENDWD : in ZTn + 14

Sign (+/-) means angle of a track set up at (beginning/end). The magnitude is the distance from this point that k is set up, in units of L/2.

(4) This small inner loop uses IR1 to select the word to be cleared, stepping it on by one each time; the count through these words is made by initially loading IR4 with the number to be cleared then loading this number into IR1; IR1 then contains the address of the 14th word of the current bank on exit from the loop (the address of ENDWD). The main outer loop counting through the track banks is controlled by IR1, and the loop is left when IR1 contains the address of ENDWD in the last bank.

(5) Somewhere early in ZSET (it happens to be here) we set the track number to zero in the last track bank to ensure that the "set-up" part (ZSET2) finally exits to the "pick-up", ZSET 8, by the test in ZSET2+1.

(6) The address of the first word of the track being examined is contained in IR2. The set-up loop is cycled until a zero track number is found in a track bank. At each pass through the set-up loop IR2 is advanced by the

number of words in a track bank (ZSET 2A, ZSET 2A+1). The actual setting up is done, one track at a time, by a closed subroutine starting in ZSET 2B.

(7) If a missing neutral is specified in the α word of the calling sequence of ZSET, the required mass is selected from MASTBL (in PARAK bank). The track is given a constraint reduction number of 3, an LSZ number of 3, a future track of 0, and a track number of 31 to identify it as a missing neutral.

(8) The closed subroutine ZSET 2B may also be entered from ZSWIM.

(9) In word 15 of each track bank is LSZ. Its magnitude is 1 for a leaving track, 2 for a stopping track, and 3 for a zero-length or missing-neutral track. The test is made to find if it's a Z track. If so the manufacture of the KPT word is bypassed.

(10) If it's an L or S track, the set-up is begun by storing the "angle" terms of the error matrix. This is stored by rows in words 5 to 13 of the track bank. $ZTN+5 = \delta\phi^2$, $ZTn+8 = \delta\phi\ \delta s = ZTn+6$ (symmetric matrix), and $ZTn+9 = \delta s^2$.

(11) It may be necessary to swim k later in ZSET, and so in preparation we store S_b now in readiness for a possible exit to KSWIM. The correct S_b or S_e according to ENDWD will not be stored until the end, after ZSET4.

(12) We assume the most likely case, that it's an L, and make up KPT word on this basis. RINTOP is a range-to-momentum conversion subroutine which is entered with the mass in the AC, and in the MQ the distance (half of track length) in this case, as we wish to calculate the momentum at the midpoint). Return from RINTOP leaves the momentum P(L/2) in the AC. This is converted to

$$k\left(\frac{L}{2}\right) = \frac{\sqrt{1 + S^2}}{P\ (1/2)}$$

and the KPT word

$$\frac{k\left(\frac{L}{2}\right) - \left|k_m\right|}{\delta k}$$

is stored in ZTn+24. RINTOP error exit goes to RPERR in ZKICK, and square-root routine error goes to RSERR in ZKICK.

(13) The last two bits of the mass word ZTn+18 are put in the first two bits of MQ. This makes the sign of MQ positive for positive charge, negative for negative charge, since the mass-word charge code is 00 for neutral, 01 for positive and 11 for negative for these two bits. The magnitude of the Pang curvature, k_m, is stored in ZTn+2, and its sign is made the same as (sign of charge times sign of Pang k_m). Thus if these two signs are the same we have $+|k_m|$ in ZTn+2; if they differ we have $-|k_m|$.

(14) Continuing the error matrix started before (Note 10) we have $ZTn+13 = \delta k^2$, $ZTn+7 = \delta k \delta\phi = ZTn+11$.

(15) We next want to see whether there are meaningful k measurements on the track. If not, we must increase the constraint reduction number to 1; and reduce the dimensions of the variance matrix to 2 by 2. There can be two reasons for unmeaningful k: (1) the track was a two-point track, (2) the fractional uncertainty dk/k, was larger than about unity. We know from the theory of least squares that if one variable is badly determined, the variance matrix tends to become singular; we find empirically that this trouble sets in at $dk/k \sim 1$.

Our next version of GUTS (to be called FIT) will probably solve for the badly known variables last (see Memo 187, by F. T. Solmitz); for the moment we simply throw away k information if dk/k > DKLVL.

We intend to put in better logic; namely to introduce two dk/k levels. For dk/k > "10", we may throw away information immediately; otherwise, we may try for a fit. Only if GUTS gives a reject for singular matrices will we re-try after throwing away dk/k > "1".

The test of dk/k is currently both in ZSET and ZVERT; the reason for putting it two places will probably continue as long as we have a version of GUTS that must be protected against large errors, even if it is a two-vertex version. The argument is as follows:

It must be in ZVERT, because even with a two-vertex GUTS, we shall often want to fit two vertices in sequence, one at a time, and the first one may give a large dk/k.

On the other hand, Peter Berge has found it useful at least in the K^-+p low-energy experiment, to have the constraint reduction numbers already assigned as far as possible on entry to ZVERT. This is an experiment where most of the two-vertex fits have been programmed to use the special 2V GUTS classes, so that most of the ZVERT entries are directly from ZSET.

It is convenient to know whether or not the GUTS class can be used before entry to ZVERT.

We have already mentioned that when there is no meaningful k information CRN is set to 1 and the error matrix reduced to 2 by 2 from 3 by 3. In addition, we set δk^2 in ZTn+13 = 10^{10}, and Endwd to 0.

(16) ENDWD negative means angles set up at end, so it's an incoming track. For this case, GUTS will automatically use the curvature upstream, k_m, and so it is unnecessary to SWIM k.

(17) For a zero-length track, CRN is set equal to 3 and ENDWD to 0, and the mass word is tested to see if the particle is charged. If so, and its mass is less than a Λ or more than a deuteron (i. e. it's not a very short-lived Σ or Ξ) the Wall condition is imposed.

When setting up a zero length, charged "stable" particle, we wish to set this flag (Integer+1 in ZTn+0; after a fit this will be occupied by ϕ, a floating number) so that ZVERT may invoke WALL (see Memo 86; Only one zero-length track per vertex is possible). We realize that the definition of "stable" is arbitrary. Currently all particles are called "stable" except Σ and Ξ (we had in mind low-energy particles when making this decision -- In terms of probability of traversing more than 1/2 mm it may be that even these particles should be considered "stable" for high energy experiments.)

(18) For S tracks, the momentum calculated from arc length by Pang is used. The magnitude of ENDWD is adjusted to show that k is set up at beginning. The error matrix is completed as before (14) except using the terms for k-via-arc rather than k_m. The KPT quantity is replaced by its negative modulus as required for an S track.

(19) At ZSET 4 the S and L branches rejoin, and (s_b, ϕ_b) and (S_e, ϕ_e) are stored in (ZTn+1, ZTn) according to the sign of ENDWD.

(20) KSWIM is a subroutine of ZSWIM, and wasn't designed to be used with ZSET. KSWIM assumes that the sign of ENDWD shows the place you want to swim away from. Hence if we wish to swim upstream, we must impress KSWIM that the track is already set up at the end, so we must make ENDWD negative before entry.

(21) The sign of the curvature before swimming is preserved and added to the magnitude of the swim curvature in ZTn+2. ENDWD is now set equal to zero to show k is set up at the beginning.

(22) Now that k has been swum, it is necessary to recalculate the terms involving k in the error matrix. ZSWIM contains machinery for doing this in a general way, but for this particular case it is simpler to do it explicitly. The following new correlations and errors are computed:

$$\frac{\partial k_2}{\partial k_1} \overline{\delta\phi_1\, \delta k_1} + \frac{\partial k_2}{\partial S_1} \overline{\delta\phi_1\, \delta S_1} \rightarrow \overline{\delta\phi\, \delta k}$$

$$\frac{\partial k_2}{\partial k_1} \overline{\delta S_1\, \delta k_1} + \frac{\partial k_2}{\partial S_1} \overline{\delta S_1\, \delta S_1} \rightarrow \overline{\delta S\, \delta k}$$

$$\left(\frac{\partial k_2}{\partial k_1}\right)^2 \overline{\delta k_1\, \delta k_1} + 2\, \frac{\partial k_2}{\partial S_1} \cdot \frac{\partial k_2}{\partial k_1} \overline{\delta S_1\, \delta k_1} + \left(\frac{\partial k_2}{\partial k_1}\right)^2 \overline{\delta k_1\, \delta k_1} \rightarrow \overline{\delta k\, \delta k}$$

(23) The pickup command (PUC) stored in MBANK+12 is tested to determine whether the event is connected or not. If connected, it is known that at least one track from the previous event should be found in the pickup banks. These tracks have been placed there by ZEND in the previous event starting from ZPUB1 and filling up in sequence as many as necessary of the nine available banks. ZSET8 looks at each ZPUBn+17 (the track number) and ZPUB+18 (the mass value) in turn starting from n=1, and finds the "mating" track bank with the corresponding track number and mass value. It then transfers the 55-word track from ZPUB into the track banks. No allowance is made in ZSET for pick-up failure because this is dealt with in ZREAD.

Map for Kick 5.* If a track has a future track number other than zero, ZEND transfers this track from the track banks to the pickup banks in the following modified format. The numbers given in square brackets are examples.

*Notice that in Kick 4 the PUBs are one word longer than the ZTN's, because the old FTN was temporarily stored in ZPBn+0. This leads to the annoying complication that the wth word of ZTn is associated with the (w+1)st word of ZPUBn.

In ZTn+		Kick 5 In ZPUBn+		Kick 4* ZPUBn+	
0	ϕ	ϕ swum		0	Old FTN [5]
1	S	S swum		1	(i. e.
2	k	k swum		2	displaced
3	Status [+1]	Doubled and set neg[-2]		3	one)
4	crn	Unmodified		4	
5	Error matrix	Swum		5	
.				6	
.				7	
13				8	
14	Endwd			9	
15	LSZ	Unmod.		10	
16	FTN [5]	Cleared [0]		11	
17	Present TN [3]	Old FTN [5]		.	
18	All Pang data passed along			.	
.				.	
.				.	
.				.	
54				55	

*See note on preceding page.

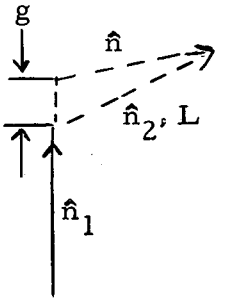
UCRL-9099
Processing: ZSET-9
(EXTEND-1) April 1960
Jon Peter Berge, Janice Button, and
Hovan Zarian

EXTEND*

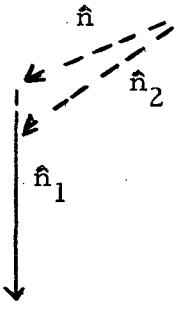
This closed subroutine is useful for events including a zero-prong and a neutral. Its purpose is to "extend" the zero-prong (since the average interaction occurs beyond the last bubble) and thus modify the measurements on the neutral track. The present routine is designed to treat tracks in the format resulting from the use of the Kick subroutine ZSET; the numbers that are altered are those in the Kick, rather than the Pang, part of the track banks.

There are three cases in which this routine EXTEND is applicable, as indicated by the sketches on the left. (The case in which two particles are inbound is ignored.) We have $\hat{n} = (\hat{n}_2 - \epsilon\, \hat{n}_1)/d$, where $\epsilon = (g/L$, $d = (1 + \epsilon^2 - 2\epsilon\, \hat{n}_1 \cdot \hat{n}_2)^{1/2}$, L is the length of the neutral, and g is the mean gap length, which equals the extension length. The quantity ϵ is given a positive value in Cases 1 and 2, and a negative value in Case 3. Subscript 1 refers to the charged track and subscript 2 to the neutral.

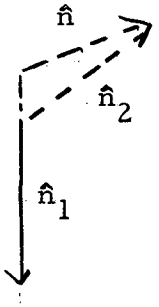
Case 1: NORMAL
(one in, two out)



Case 2: (1 out, 2 in)



Case 3; (1 out, 2 out)



The present calling sequence for EXTEND is:

a	TSX EXTEND, 4
a + 1	PZE -ZTn, T, -ZTm
a + 2	return

with T = 0 for Cases 1 and 2 and T = 7 for Case 3. The track-bank address of the charged zero prong to be extended is ZTn, while that of the neutral track to be modified is ZTm.

If the CRN of either track is 3, no action is taken and the input quantities are unchanged. If the CRN of track m is 0, they are also unchanged. Otherwise, the setup measurements in track bank m are modified in accordance with the procedure described below.

*The contributions of Frank Solmitz to the development of this program are very much appreciated.

In terms of Kick variables, we have

$$\hat{n}_1 = [1/(1+s_1^2)^{1/2}]\,[\cos\phi_1,\ \sin\phi_1,\ s_1] = [n_{1x},\ n_{1y},\ n_{1z}],$$

and n_2 has the same form. (The trigonometric factors within the brackets are associated respectively with the x, y, and z components of n.) Thus in terms of these same variables, the modified neutral direction is

$$\hat{n} = 1/dR_1R_2[R_1\cos\phi_2 - \epsilon R_2\cos\phi_1,\ R_1\sin\phi_2 - \epsilon R_2\sin\phi_1,\ R_1 s_2 - \epsilon R_2 s_1]$$

$$= 1/dR_1R_2\,(A,\ B,\ C),$$

where

$$R_1 = \sqrt{1+s_1^2}$$

$$A = R_1\cos\phi_2 - \epsilon R_2\cos\phi_1$$

$$B = R_1\sin\phi_2 - \epsilon R_2\sin\phi_1$$

$$C = R_1 s_2 - \epsilon R_2 s_1.$$

Then we have

$$\phi = \tan^{-1}(B/A)$$

$$s = C/(A^2+B^2)^{1/2}$$

with A, B, and C as defined above and with $0 \leq \phi \leq \pi$ for $B>0$ and with $\pi \leq \phi \leq 2\pi$ for $B<0$.

To compute the variance matrix of ϕ and s for the modified track m, we define the following:

$$Y_1 = \phi,\ Y_2 = s.$$

$$x_i = \phi_1,\ s_1,\ \phi_2,\ s_2,\ \epsilon$$

$$Y_i = Y_i(\phi_1,\ s_1,\ \phi_2,\ s_2,\ \epsilon),$$

$$\overline{\delta Y_i \delta Y_j} = A_{ik} G^{-1}_{k\ell}\ A^T_{\ell ji}$$

with $A_{ij} = \left(\frac{\partial Y_i}{\partial x_j}\right)_i$ and $G^{-1}_{ij} = \overline{\delta x_i \delta x_j}$.

The new variables Y_i and the new variance matrix $\overline{\delta Y_i \delta Y_j}$ are stored in the appropriate places in the Kick region of track bank m and return is made to the program.

Note that by this procedure particle-particle correlation terms have been introduced into the "normal" variance matrix for the tracks of the event, for there are now such terms as $\delta\phi\delta\phi_1 \neq 0$. That is, all terms of $\delta Y_i \delta Y_j$, the error matrix for the modified neutral track, are calculated, but no $\delta Y \delta x$ terms. GUTS does not accept these in its input.

To obtain $\overline{\delta Y_i \delta Y_j}$, we evaluate the following quantities with the appropriate gap length (GAPL), g, and its error (DGAPL), dg, for the film sample of interest. So far we have set DGAPL = GAPL. Thus we have:

$$\epsilon = (\pm)\, g/L$$

$$\overline{\delta\epsilon^2} = \epsilon^2 \left[\left(\frac{\delta g}{g}\right)^2 + \left(\frac{\delta L}{L}\right)^2 \right]$$

$$G_{ij}^{-1} = \begin{bmatrix} \overline{\delta\phi_1\delta\phi_1} & \overline{\delta\phi_1\delta S_1} & 0 & 0 & 0 \\ \overline{\delta S_1\delta\phi_1} & \overline{\delta S_1\delta S_1} & 0 & 0 & 0 \\ 0 & 0 & \overline{\delta\phi_2\delta\phi_2} & \overline{\delta\phi_2\delta S_2} & 0 \\ 0 & 0 & \overline{\delta S_2\delta\phi_2} & \overline{\delta S_2\delta S_2} & 0 \\ 0 & 0 & 0 & 0 & \overline{\delta\epsilon^2} \end{bmatrix}$$

$$A_{ij} = \frac{\partial Y_i}{\partial x_j} = \frac{1}{A^2+B^2} \left[A \frac{\partial B}{\partial x_j} - B \frac{\partial A}{\partial x_j} \right];$$

$$A_{2j} = \frac{\partial S}{\partial x_j} = \frac{1}{(A^2+B^2)^{1/2}} \frac{\partial C}{\partial x_j} - \frac{C}{(A^2+B^2)^{3/2}} \left(A \frac{\partial A}{\partial x_j} + B \frac{\partial B}{\partial x_j} \right)$$

$\frac{\partial A}{\partial \phi_1} = \epsilon R_2 \sin\phi_1$	$\frac{\partial B}{\partial \phi_1} = -\epsilon R_2 \cos\phi_1$	$\frac{\partial C}{\partial \phi_1} = 0$
$\frac{\partial A}{\partial S_1} = \frac{S_1}{R_1} \cos\phi_2$	$\frac{\partial B}{\partial S_1} = \frac{S_1}{R_1} \sin\phi_2$	$\frac{\partial C}{\partial S_1} = \frac{S_1 S_2}{R_1} - \epsilon R_2$
$\frac{\partial A}{\partial \phi_2} = -R_1 \sin\phi_2$	$\frac{\partial B}{\partial \phi_2} = R_1 \cos\phi_2$	$\frac{\partial C}{\partial \phi_2} = 0$
$\frac{\partial A}{\partial S_2} = -\frac{\epsilon S_2}{R_2} \cos\phi_1$	$\frac{\partial B}{\partial S_2} = -\frac{\epsilon S_2}{R_2} \sin\phi_1$	$\frac{\partial C}{\partial S_2} = -\epsilon \frac{S_1 S_2}{R_2} + R_1$
$\frac{\partial A}{\partial \epsilon} = -R_2 \cos\phi_1$	$\frac{\partial B}{\partial \epsilon} = -R_2 \sin\phi_1$	$\frac{\partial C}{\partial \epsilon} = -R_2 S_1$.

The following Fortran program was used to check the calculations performed in EXTEND. In its present form the Fortran program assumes the a + 1 word of calling sequence to be PZE -ZT1, 0, -ZT2. Furthermore it is not in the form of a Fortran subroutine.

This program can be made equivalent to the present SAP version of EXTEND by the addition of the following Fortran III statements:

a. A subroutine statement, SUBROUTINE EXTEND.

b. A common statement, and dimension statements serving to designate all the Kick constants as Fortran common variables.

c. Several logical statements in Fortran III that pick up the a + 1 word of the calling sequence, calculate from the address and decrement which two trackbanks are specified (counting the trackbanks backward with respect to the order in memory), and plant these two numbers in MEX 1 and MEX 2, respectively.

d. A test on the tag of the a + 1 word of the calling sequence as described on page ZSET 19 Extend-1 should be introduced. It also should be in Fortran III statements.

e. A RETURN statement should be added as the last statement.

Fortran program for checking EXTEND calculations

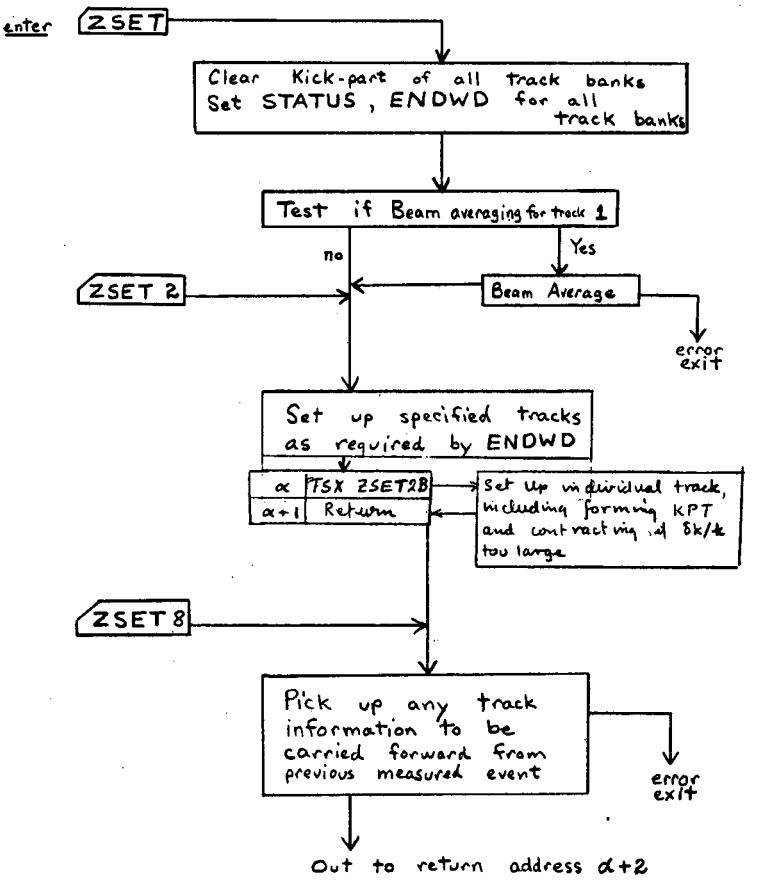
```
      DIMENSION EM(5, 3), EN(5, 5), BECPHI(5), VECS(5), EXTZTN(55, 15)
      MEX1=15
      MEX2=14
      GAPL=0.05
      DGAPL=0.05
      R1=SQRTF(1. +EXTZTN(54, MEX1)*EXTZTN(54, MEX1))
      R2=SQRTF(1. +EXTZTN(54, MEX2)*EXTZTN(54, MEX2))
      SINFI1=SINF(EXTZTN(55, MEX1))
      SINFI2=SINF(EXTZTN(55, MEX2))
      COSFI1=COSF(EXTZTN(55, MEX1))
      COSFI2=COSF(EXTZTN(55, MEX2))
      EPS=GAPL/EXTZTN(36, MEX2)
      EXTA=R1*COSFI2-EPS*R2*COSFI1
      EXTB=R1*SINFI2-EPS*R2*SINFI1
      EXTC=R1*EXTZTN(54, MEX2)-EPS*R2*EXTZTN(54, MEX1)
      EXTPHI=ATANF(EXTB/EXTA)
      EXTS=EXTC/(SQRTF(EXTA*EXTA+EXTB*EXTB))
      EN1DN2=(COSFI1*COSFI2+SINFI1*SINFI2+
     1    EXTZTN(54, MEX1)*EXTZTN(54, MEX2))/(R1*R2)
      EXTD=SQRTF(1. +EPS*EPS-2. *EPS*EN1DN2)
      EXDELL=EXTZTN(35, MEX2)
      DLSQEP=(DGAPL*DGAPL+EPS*EPS*EXDELL*EXDELL)/
     1    (EXTZTN(36, MEX2)*EXTZTN(36, MEX2))
      EM(1, 1)=EPS*R2*SINFI1
      EM(1, 2)=-EPS*R2*COSFI1
      EM(1, 3)=0.0
      EM(2, 1)=(EXTZTN(54, MEX1)*COSFI2)/R1
      EM(2, 2)=(EXTZTN(54, MEX1)*SINFI2)/R1
      EM(2, 3)=(EXTZTN(54, MEX1)*EXTZTN(54, MEX2)-EPS*R1*R2)/R1
      EM(3, 1)=-R1*SINFI2
      EM(3, 2)=R1*COSFI2
      EM(3, 3)=0.0
      EM(4, 1)=-(EPS*EXTZTN(54, MEX2)*COSFI1)/R2
      EM(4, 2)=-(EPS*EXTZTN(54, MEX2)*SINPHI1)/R2
      EM(4, 3)=-(EPS*EXTZTN(54, MEX1)*EXTZTN(54, MEX2)-R1*R2)/R2
      EM(5, 1)=-R2*COSFI1
      EM(5, 2)=-R2*SINFI1
      EM(5, 3)=-R2*EXTZTN(54, MEX1)
      DO 10 I=1, 5
      DO 10 J=1, 5
   10 EN(I, J)=0.0
      EN(1, 1)=EXTZTN(50, MEX1)
      EN(1, 2)=EXTZTN(49, MEX1)
      EN(2, 1) =EN(1, 2)
      KRN=EXTZTN(51, MEX1)
      IF (KRN-1) 20, 15, 20
   15 EN(2, 2)=EXTZTN(47, MEX1)
      GO TO 35
   20 IF (KRN-0) 30, 25, 30
   25 EN(2, 2)=EXTZTN(46, MEX1)
   35 EN(3, 3)=EXTZTN(50, MEX2)
```

```
      EN(3, 4)=EXTZTN(49, MEX2)
      EN(4, 3)=EN(3, 4)
      EN(4, 4)=EXTZTN(47, MEX2)
      EN(5, 5)=DLSQEP
      EMODAB=DQRTF(EXTA*EXTA+EXTB*EXTB)
      DO 40 I=1, 5
      VECPHI(I)=(EXTA*EM(I, 2)-EXTB*EM(I, 1))/(EMODAB*EMODAB)
   40 VECS(I)=(EM(I, 3)-EXTC*(EXTA*EM(I, 1)+EXTB*EM(I, 2))/(EMODAB*EMODAB))
     1    /EMODAB
      DFIDFI=0.0
      DFIDS=0.0
      DSDS=0.0
      DO 45 I = 1, 5
      DO 45 J = 1, 5
      DFIDFI=DFIDFI+VECPHI(I)*EN(I, J)*VECPHI(J)
      DFIDS=DFIDS+VECPHI(I)*EN(I, J)*VECS(J)
   45 DSDS=DSDS+VECS(I)*EN(I, J)*VECS(J)
      GO TO 30
   30 STOP
      END (0, 1, 0, 0, 1)
```

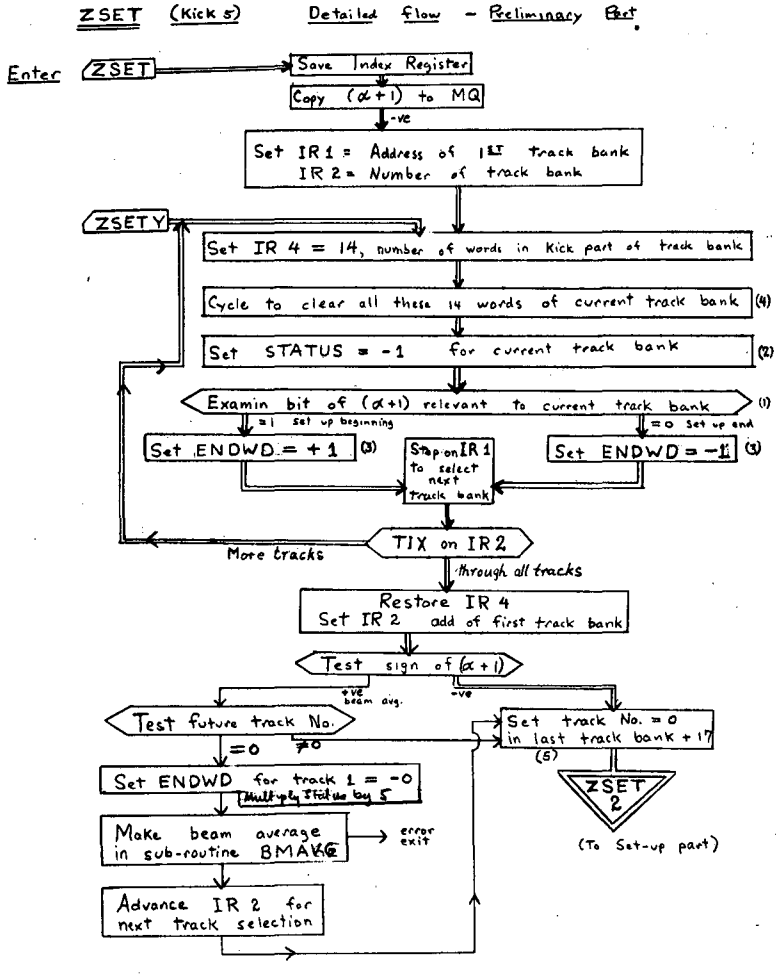
June 14, 1960

ZSET (Kick 5)

General Flow

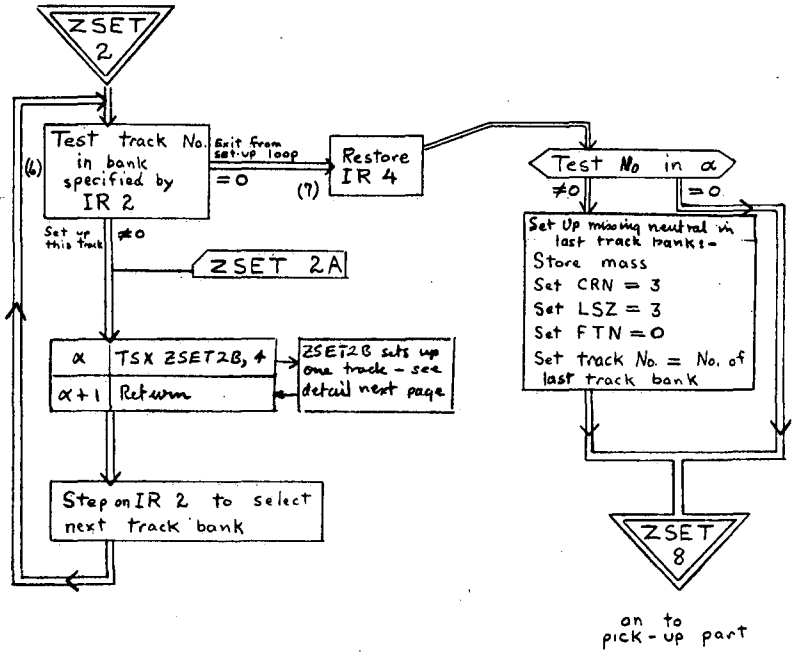


MU-23676


ZSET (Kick 5) Detailed flow - Preliminary Sheet
Enter
ZSET
Save Index Register
Copy (α+1) to MQ
-ve
Set IR 1 = Address of 1ST track bank
IR 2 = Number of track bank
ZSETY
Set IR 4 = 14, number of words in Kick part of track bank
Cycle to clear all these 14 words of current track bank (4)
Set STATUS = -1 for current track bank (2)
Examin bit of (α+1) relevant to current track bank (1)
=1 set up beginning
=0 set up end
Set ENDWD = +1 (3)
Skip on IR 1 to select next track bank
Set ENDWD = -1 (3)
TIX on IR 2
More tracks
through all tracks
Restore IR 4
Set IR 2 add of first track bank
Test sign of (α+1)
+ve beam avg.
-ve
Test future track No.
=0
≠0
Set ENDWD for track 1 = -0
Multiply Fixture by 5
Make beam average in sub-routine BMAVG
error exit
Advance IR 2 for next track selection
Set track No. = 0 in last track bank + 17
(5)
ZSET 2
(To Set-up part)

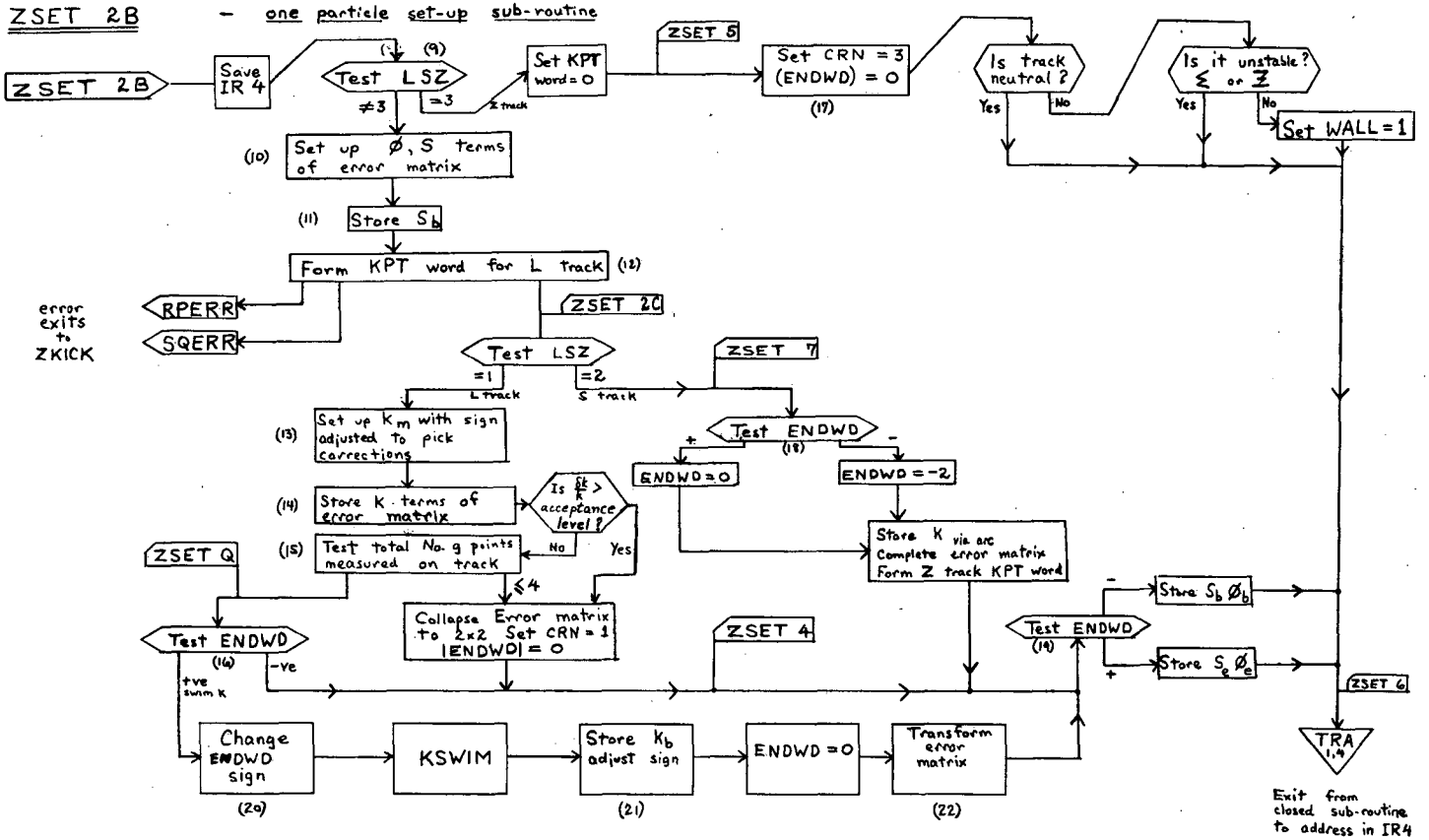
MU-23675

ZSET Set-up part



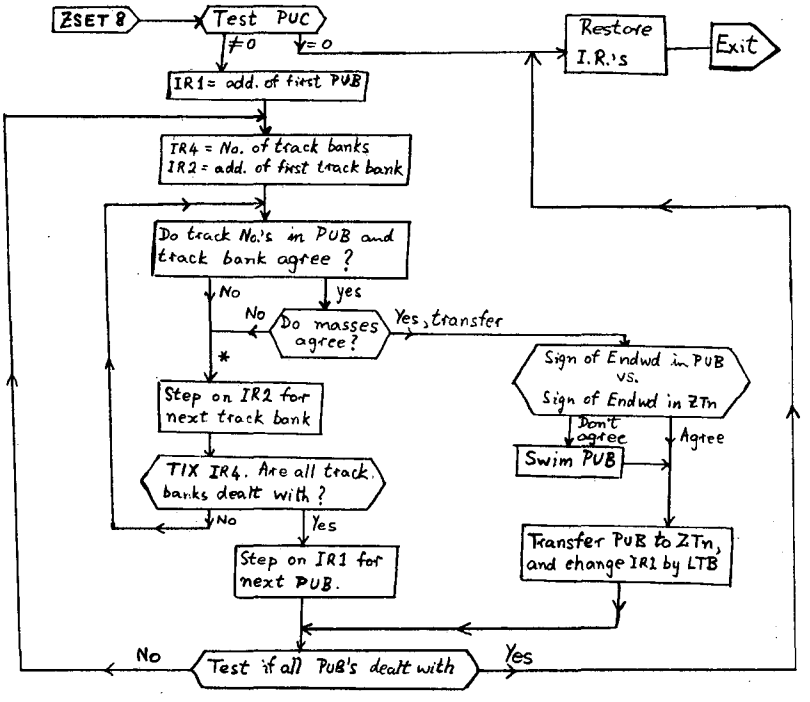
MU-23674

ZSET Set-up part continued



MUB-671

ZSET Pick-up part (23,24) page 5.



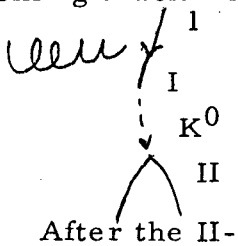
* Pick-up failures never reach ZSET as they result in FRJ1 in ZREAD.

MU-23670

Notes on ZVERT

On the assumption that the subroutine ZSET has been called on at least once to operate on the information contained in the track banks, ZVERT, as directed by two code words, (see Calling Sequence Section) extracts from the track banks, and places in GUTS storage locations the information necessary for GUTS to execute the fit. In this sense ZVERT may properly be termed a calling sequence routine. During the process of extraction and deposition, various tests are made on the physical and logical properties of the participating tracks. (These tests further delineate the precise manner in which GUTS will function.) The results of these tests sometimes manifest themselves as "hypothesis rejects."

Two entrances to ZVERT are possible; one is called ZVERT, the other PVERT. PVERT is used least frequently and its function is more easily described by an example. Consider an event of the type shown where the incoming track 1 has no measured k. Vertex I cannot be fitted first. The following procedure will give the "best" values for track variables and the correct value of chi squared for a multi-vertex event, but not the correct errors.



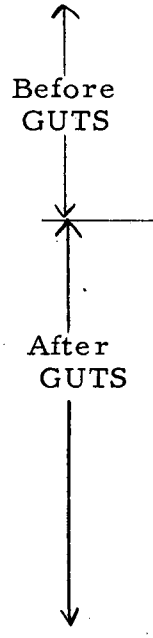
After the II-I fit, the K^0 track connecting the two vertices has been completely fitted and its vector momentum, P (connecting), can be calculated. As has been empirically shown, essentially the same P (connecting) would be obtained were it possible to do the I-II sequence of fits. If the II fit is repeated in such a way as to force the connecting track to have the correct value P (connecting), then the other tracks at the vertex will also have their correct values.

To effect this procedure, it is necessary before entry to PVERT to store the vector P (connecting) in ZVERT locations PX, PY, PZ, E. GUTS will then recognize the situation by setting $\delta\phi(\text{conn.}) = \delta\tan\lambda(\text{conn.}) = \delta k(\text{conn.}) = 0$ and carrying out the fit, subject to the constraint of a fixed P (connecting). The ZVERT entrance differs from PVERT only in that it zeroes PX, PY, PZ, E.

ZVERT reaches its climax, so to speak, when it enters GUTS; on return from GUTS, ZVERT takes the fitted quantities out of GUTS storage, and places them in the track banks. Other features include a rearrangement of GINOUT (the GUTS 3P-by-3P error matrix-see Memorandum 86) to make the order of each particle's elements correspond to the particle's order given in the

ZVERT calling sequence. This will simplify data handling in EXAMIN. Also, an equivalent to K-Pang test, called K-Kick is computed.

To facilitate a close scrutiny of the coding, a brief description of each of the major loops will be given in the following order (the mnemonic on the left is the symbolic first word address of the loop):



ZV2:	Track analysis
ZVERT2:	GUTS ordering
ZV5:	Constraint class determination
ZV5A:	Transfer of information to GUTS storage
	Enter GUTS
ZV9+4:	Expansion of GUTS output to account for stopping incident particle
ZV6:	Storage of fitted quantities into track banks
ZV10A:	Reordering of GUTS output to correspond to ZVERT order
ZV11:	Consistency check on fitted curvature of leaving tracks
ZVERTX:	Exit

Particle Lists Required by GUTS

Before GUTS is entered, ZVERT must provide four lists comprising the known information on the $P \leq 7$ tracks to be fitted at the current vertex. These lists are at locations:

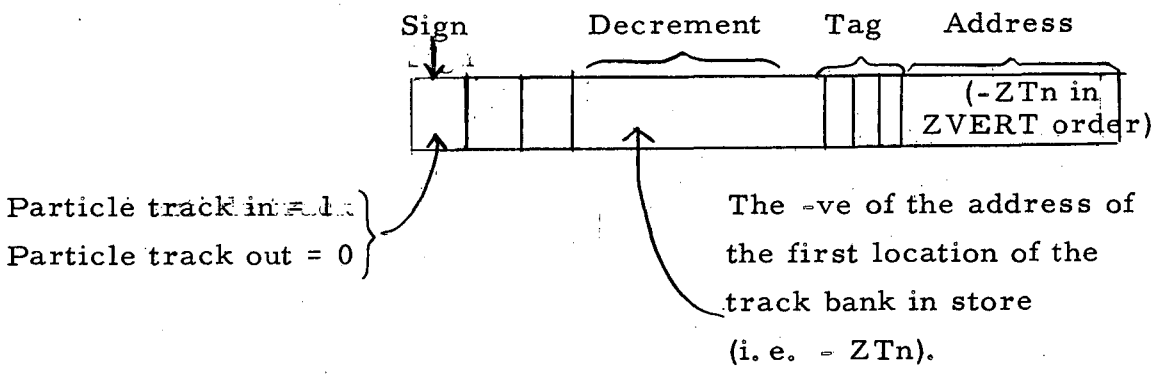
1. ZTEMP + 1 (onwards): The ZTn address and the in/out condition of each particle are stored as described in the table on page ZVERT-6
2. KMASS → (onwards): The masses of each of the particles
3. XMEAS → (onwards): The measured variables. Up to three variables for each particle.
4. GINA → (onwards): The error matrixes for each particle.

The ordering of the information for each particle is the same for all four lists. Namely, particle-track information is ordered such that those "best" particles with the lower CRN come first. Tracks which have the same CRN are ordered amongst themselves in the same sequence given in the code word (a + 2) of the ZVERT calling sequence.

ZTEMP list

This list contains P words, one for each particle, and the storage runs from ZTEMP + (8-P) through ZTEMP + 7. These words are called the "particle words".

The format of each particle word is shown in the accompanying sketch.



KMASS list. There are P entries in this list running from KMASS to KMASS + (P - 1), each entry of which contains the mass of the relevant particle. Also if a production vertex is being fitted, KMASS + 7 contains the target mass.

Location IRK in store is used as a pseudo index register in indexing through this list.

XMEAS list. There are P entries in this list each of which contain zero, two, or three words corresponding to the known information on each track. This information is packed so that no gaps are left in the list. If all information is known on P tracks at a vertex, the list runs from XMEAS to XMEAS + (3P + 1). The maximum length of the list is 21 words.

Location IRX in store is used as a pseudo index register in indexing through this list.

GINA list. There are P nine-word entries in this list which runs from GINA to GINA + (9P-1). Each block of nine words contains the appropriate error matrix stored by rows and starting from the first location in this block. The maximum length of this list is 63 words (7×9 words).

Location IRG in store is used as a pseudo index register in indexing through this list.

The ZTEMP list is first filled in by reordering the tracks denoted by the code word $\alpha + 2$ according to the specification given above (i. e. increasing CRN). A loop is then initiated which is circuited P times, once for each particle. Each time around, one entry is made (about this particle) in each of the lists KMASS, XMEAS, and GINA while the pseudo index registers IRK, IRX, and IRG are advanced.

ZT2

The address of the first word location of the track bank of the track in hand is constructed and stored in the order in which it occurs in the a + 2 code word. The storage takes place backwards from common +20 with the "first" particle in the lowest machine location (see example list on next page).

If the WALL flag has been set in ZSET, a program parameter to be used as a constraint under certain conditions is placed in WALL. The constraint reduction number (CRN) is tested, and if zero, the curvature is checked for goodness. If the curvature is found to be bad, the error matrix is collapsed from a three by three to a two by two, and the sign of Endwd is retained, but its magnitude is set equal to zero.

In case the track is associated with a stopping, incident particle, a flag is set ZVWS + 4, the target mass is incremented by the particles mass, and the number of particles to participate in the fit is decreased by one.

For nonstopping incident tracks, if the Endwd is equal to one or two, DHold is set equal to L/2 or L, respectively.

ZVERT2

Using a number called the constraint-comparison digit (CCD), we determine the total number of constraints pertaining to the hypothesis. With the CCD initially set to zero, a loop is executed which compares the CRN of each particle with the CCD. When the two are equal, the particle's address is stored in the decrement of a ZTEMP location. At the beginning of the loop, the CCD is incremented by one so that eventually the particle list is exhausted and the corresponding addresses are ordered monotonically (GUTS order) in the ZTEMP block according to increasing CRN values. As in the ZVERT ordering of particles in COMMON +20, the storage takes place backwards from ZTEMP +8 with the lowest CRN particle in the lowest machine location.

ZV2, ZVERT2 and ZV5 Loops

The operation of these loops is illustrated by an example and the table on the next page. Assume that the ZVERT calling sequence specified three ZTn:

ZT1 incoming and stopping,	CRN irrelevant
ZT2 outgoing,	CRN = 1
ZT3 outgoing,	CRN = 0.

First, the ZV2 loop fills the decrements of the three locations ahead of COMMON+20, 1 in ZVERT order. Next the ZVERT2 loop fills the decrements of the three locations ahead of ZTEMP+8, 1 in GUTS order, i.e. the best track (ZT3) first, followed by ZT2, and the incident S track last in ZTEMP+7. Finally, the short ZV5 loop stores the decrements of COMMON+20, 4 into the addresses of ZTEMP+8, 4.

	Decrement	Tag	Address		Decrement	Tag	Address
COMMON+13	(ZVERT			ZTEMP+1			(ZVERT
14	order)			2	(GUTS		order)
15				3	order)		
16				4			
17	-(-ZT1)			5	+(-ZT3)	0	(-ZT1)
18	+(-ZT2)			6	+(-ZT2)	0	(-ZT2)
19	+(-ZT3)			7	-(-ZT1)	0	(-ZT3)
					Incident		S in ZTEMP+7
	by ZV2				by ZVERT2		by ZV5

Note that ZTEMP+0 is not used.

ZV5+5

This straightforward section of coding deals with the question of whether the constraint class is a "2CZL2V" or "4C2V". These two-vertex constraint classes are called for by event-type subprograms by placing a bit in the lower-order bit of a + 1 in the ZVERT calling sequence (see section on "special requirements of two-vertex fits").

ZV5A*

At this point the program is prepared to store the appropriate quantities in GUTS storage to obtain a fit. Before the variables of any given track are transferred, its KPT and ϕ-locations are examined for a negative KPT and 7_8 ("Poison"), respectively. If neither of these tests gives rise to a reject, the three variables of each track are stored forward starting from XMEAS. The particle error matrices are stored similarly in GINA. Note that the ordering of the particles is that of GUTS as plucked out of the ZTEMP block of ZTn addresses.

*The few words after the TIX to ZV5A are to Reject the 1c case for which three CRN=1 tracks have been set up. That is, for the 1c case there must have been a CRN=3 track just transferred.

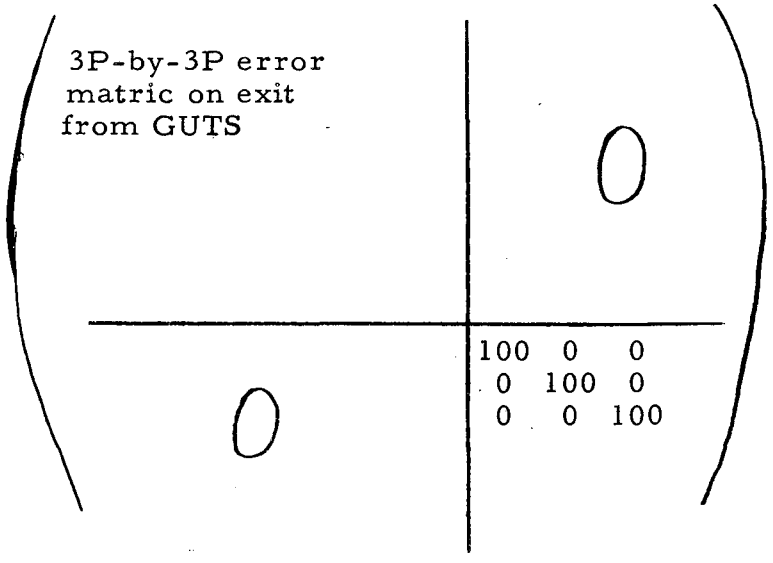
ZV9+2

Immediately after return from GUTS, the GYFCN block is rearranged from the GUTS order to the order given in the ZVERT calling sequence word $\alpha + 2$.

A test is then made to see whether an incident stopping particle was involved in the fit. If so, the following operations are performed:

1. Meaningful values of ϕ, s, and k for the incident S-track are stored in the GUTS bank, XVERT, in preparation for the subsequent transfer of fitted data from XVERT to the ZTn banks. The angles ϕ and s are given their Pang values (at the beginning of the track, as set up by ZSET) and the curvature k is set equal to 10.0 as a code.

2. The total error matrix resulting from a GUTS fit is located in GINOUT et seq., and its dimensions are 3P by 3P, where P is the number of particles participating in the fit. It will be recalled, however, that, if there is an incident stopping particle, P represents a number that is one less than the number of particles physically participating. Therefore, GINOUT is expanded by three rows and three columns to include a special 3-by-3 matrix for the incident S track. This matrix has variances equal to 100 and covariances equal to zero, as shown in the following diagram. The matrix is stored sequentially by rows in the machine's memory.



Thus, three diagonal elements of 100 are inserted. The lower right one in the diagram is located at GINOUT + $(3P+3)^2$-1; the next one up is 3P+3+1 earlier, etc.

ZV6

At this point the fitted variables are transferred to the track banks. The CRN's are set equal to zero; the magnitude of Endwd is also set equal to zero; and the sign of the status word is made positive.

ZV10A

The elements of GINOUT are rearranged according to the particle order given in the a+2 code word of the ZVERT calling sequence.

ZV11

Here a loop starts to form and test k-Kick-Test kKT for each track. As mentioned in "k-Tests", kKT makes sense only for charged, "L", tracks outbound from a vertex. Otherwise kKT is set at zero.

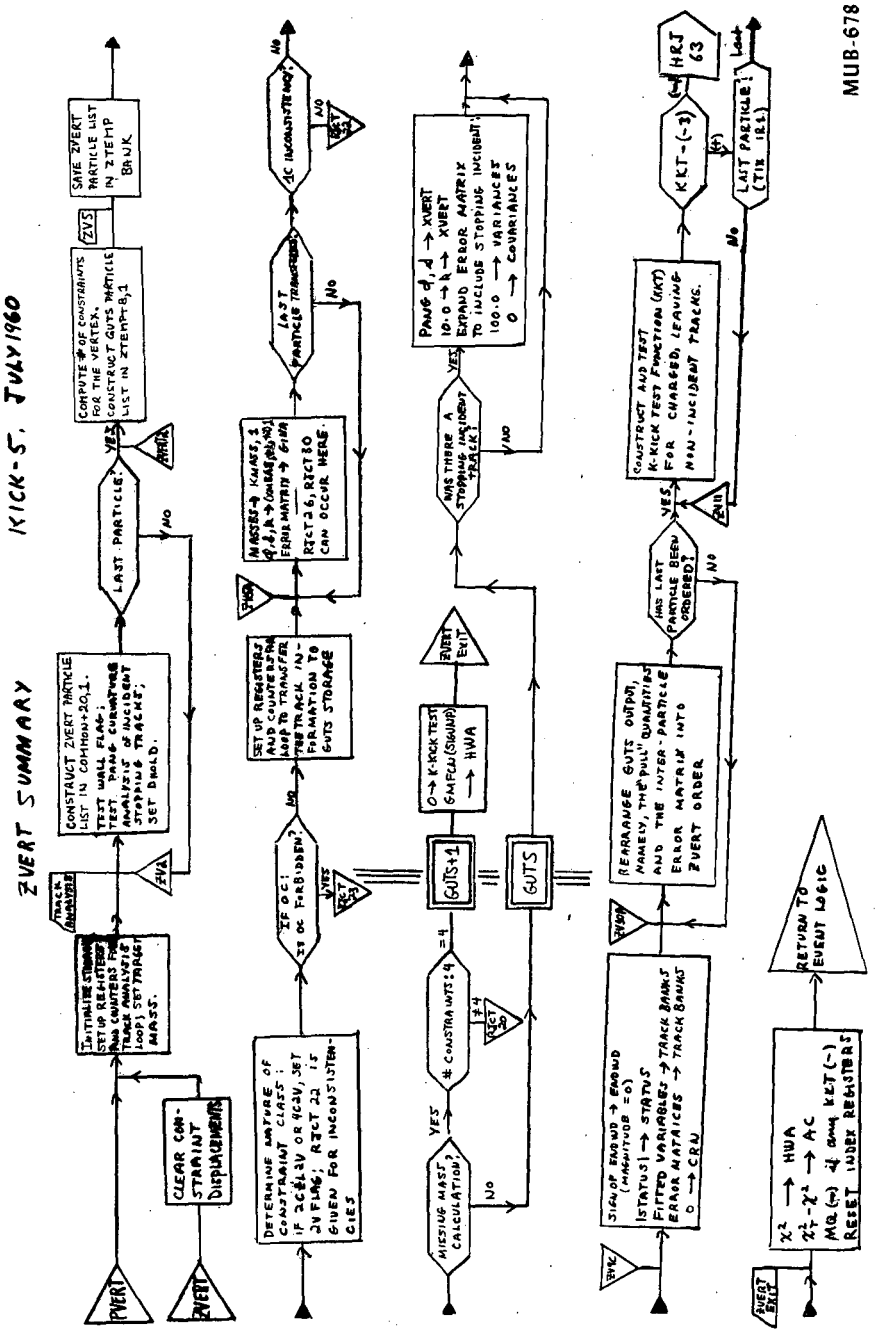
The result is stored over k-Pang Test in ZTn+24 and also (for historical reasons) in COMMON+8, 1. If kKT is negative for any track, (1) that value of kKT is stored in COMMON, which on exit from ZVERT is loaded into the MQ (i.e., the MQ will be negative for any negative kKT; (2) the number of that track is stored in STM.

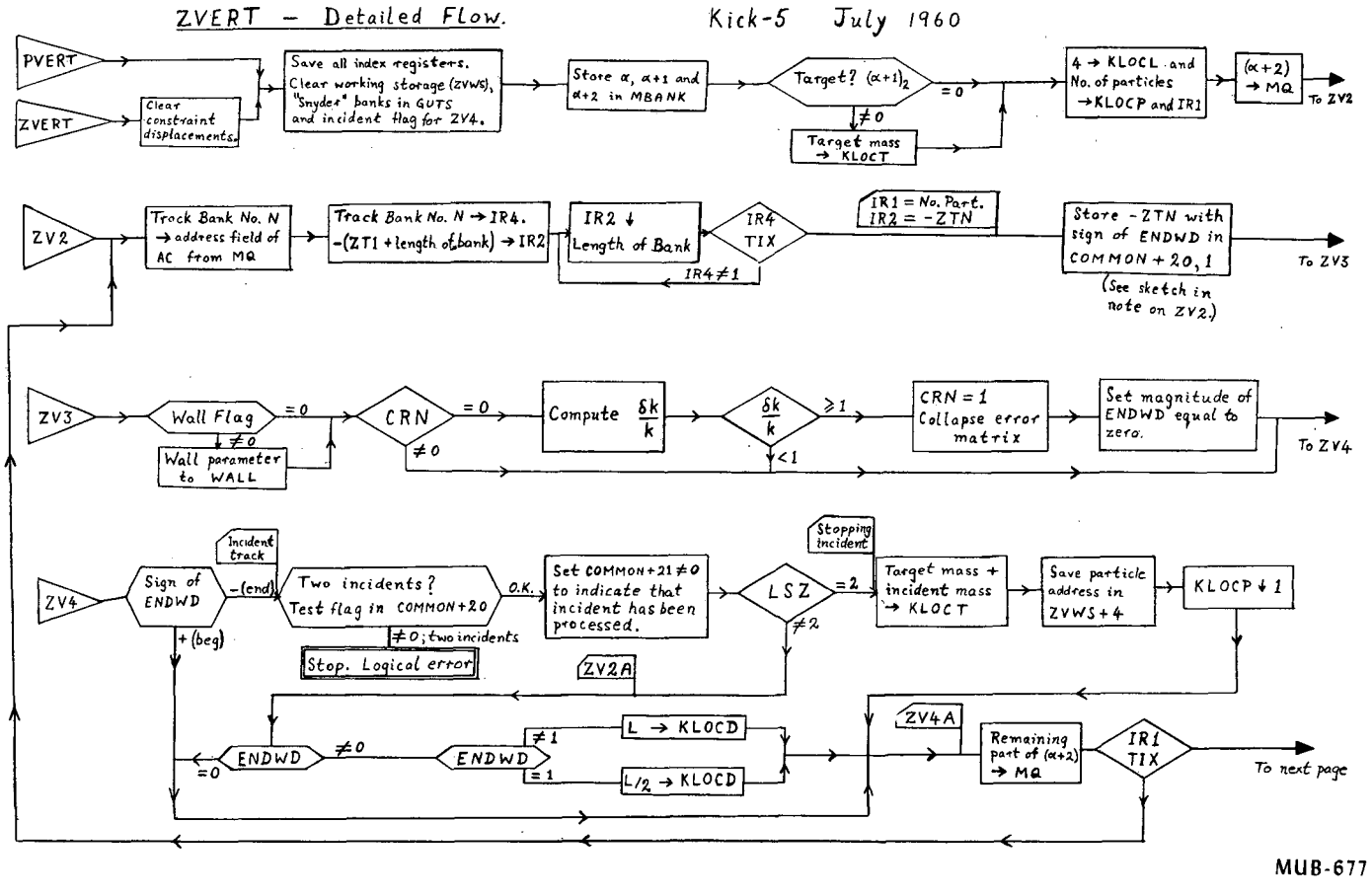
If any value of kKT is less than -3, the loop is abandoned, and we transfer to HRJ 62.

Sometime we shall perhaps add the following feature to take account of the kKT's in the fuzzy region between -3 and 0; namely, to put the sum of the squares of these kKT's in MBANK+25 so that later programs can add them to χ^2 if desired for certain tests.

ZVERT X

This is the exit of ZVERT. The chi squared is stored in Handy word A (HWA): the difference between the former and the 1% confidence limit is put into the AC and the MQ is loaded with COMMON+18 (see ZV11). Finally, the index registers are reset to the values they had upon entry to ZVERT and transfer back to the event type is made.

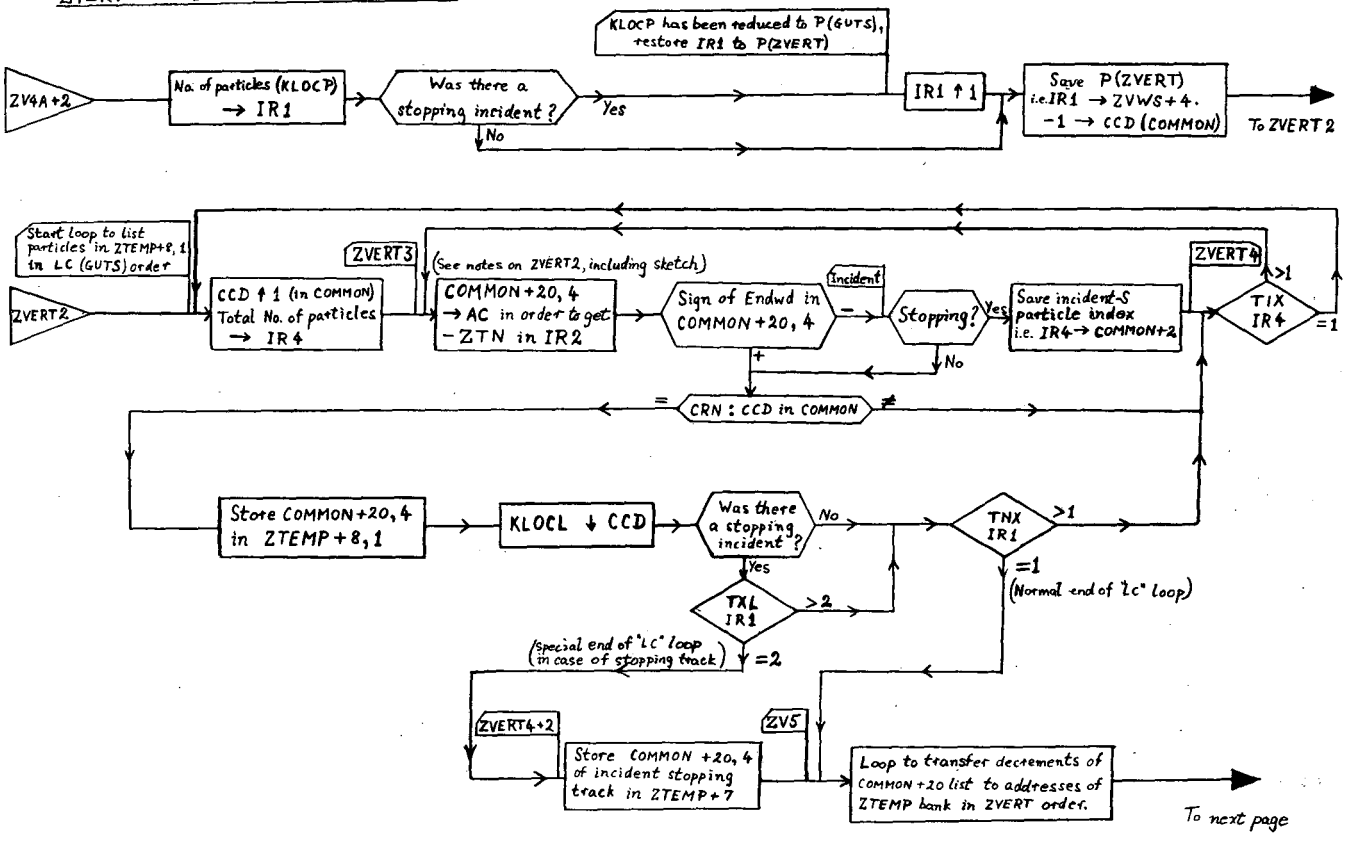

ZVERT SUMMARY
KICK-S, JULY 1960
ZVERT
CONSTRUCT ZVERT PARTICLE LIST IN COMMON+20,1.
TEST WALL FLAG;
TEST PANS CURVATURE
ANALYSIS OF INCIDENT
STOPPING TRACKS;
SET BOLD.
LAST PARTICLE?
YES
NO
COMPUTE # OF CONSTRAINTS FOR THE VERTEX.
CONSTRUCT GUTS PARTICLE LIST IN ZTEMP+8,1
SAVE ZVERT PARTICLE LIST IN ZTEMP BANK
CLEAR CONSTRAINT DISPLACEMENTS
DETERMINE NATURE OF CONSTRAINT CLASS: IF 2C&3V OR 4CV, SET IN FLAG; REJT 22 IF GIVEN FOR INCONSISTENCIES
IF OC: 3V OR FORBIDDEN?
SET UP REGISTERS AND COUNTERS FOR LOOP TO TRANSFER THE TRACK INFORMATION TO GUTS STORAGE
MASSES→ K+M+S, 1 q,p,λ→(CURRENT q,p,λ) ERROR MATRIX → G+M
REJT 26, REJT 80 CAN OCCUR HERE.
LAST PARTICLE TRANSFERRED?
NO
2C INCONSISTENT MOM?
MISSING MASS CALCULATION?
YES
NO
CONSTRAINTS = 4?
GUTS+1
GUTS
0 → K-KICK TEST SAMPLE (SWUMP) → HWA
ZVERT EXIT
PANS q, d → XVERT
10.0 → h → XVERT
EXPAND ERROR MATRIX TO INCLUDE STOPPING INCIDENT
100.0 → VARIANCES
0 → COVARIANCES
WAS THERE A STOPPING INCIDENT TRACK?
REARRANGE GUTS OUTPUT, NAMELY, THE "PULL" QUANTITIES AND THE INTER-PARTICLE ERROR MATRIX INTO ZVERT ORDER
HAS LAST PARTICLE BEEN ORDERED?
CONSTRUCT AND TEST K-KICK TEST FUNCTION (KKT) FOR CHARGED, LEAVING NON-INCIDENT TRACKS.
KKT-(-1)
HRJ 63
LAST PARTICLE? (TIK 181)
Last
SIG OF ERROR → ERROR (MAGNITUDE = 0)
ISTATUS(I) → STATUS
FITTED VARIABLES → TRACK BANKS
ERROR MATRICES → TRACK BANKS
0 → CRU
RETURN TO EVENT LOGIC
χ² → HWA
χ²-χ² → AC
MQ(-) IF ANY KET(-)
RESET INDEX REGISTERS
MUB-678



UCRL-9099
Processing: ZVERT-10
June 1960

ZVERT - Detailed Flow (continued)

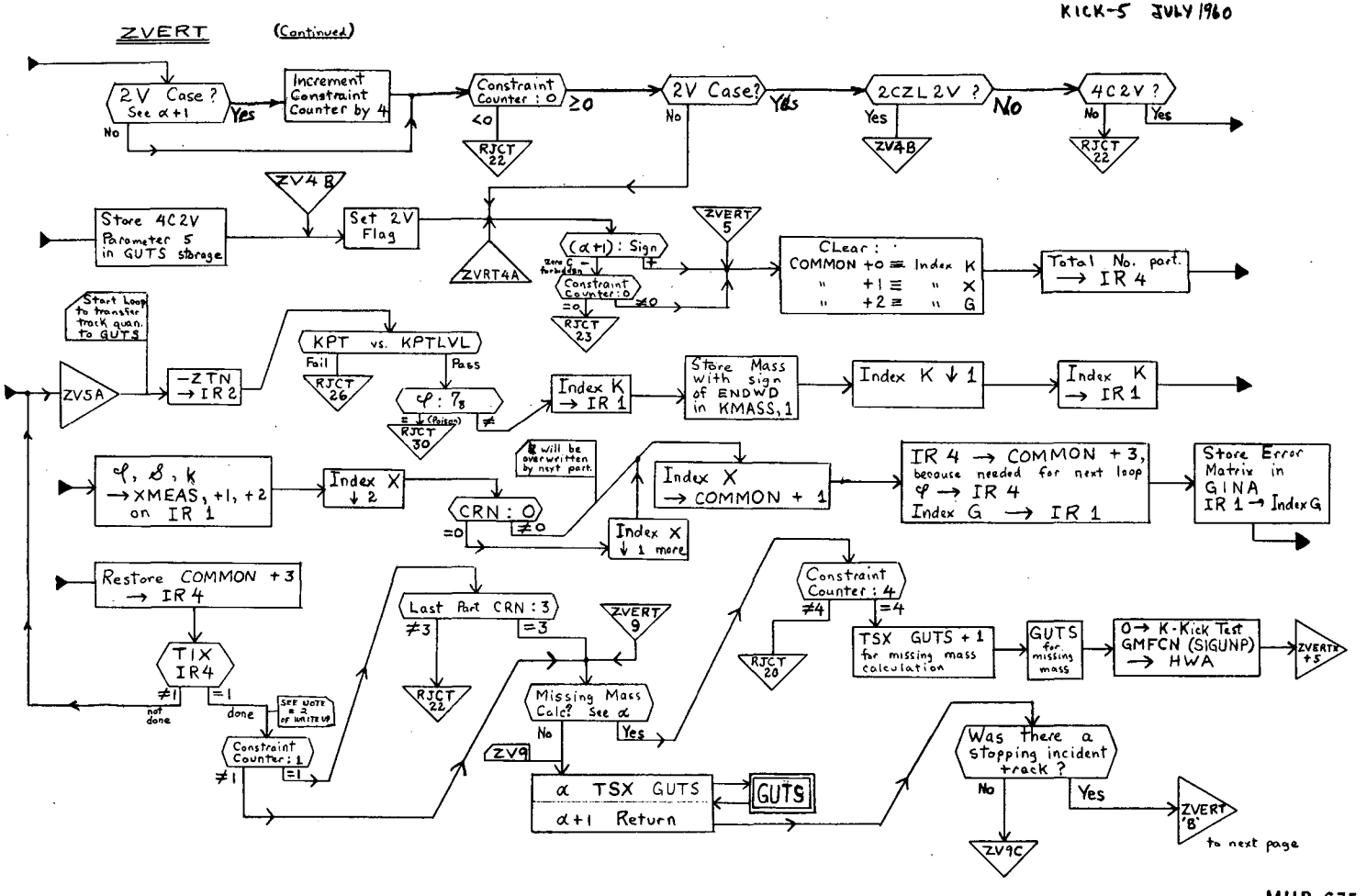
Kick-5 July 1960



MUB-676

UCRL-9099
Processing: ZVERT-11
June 1960

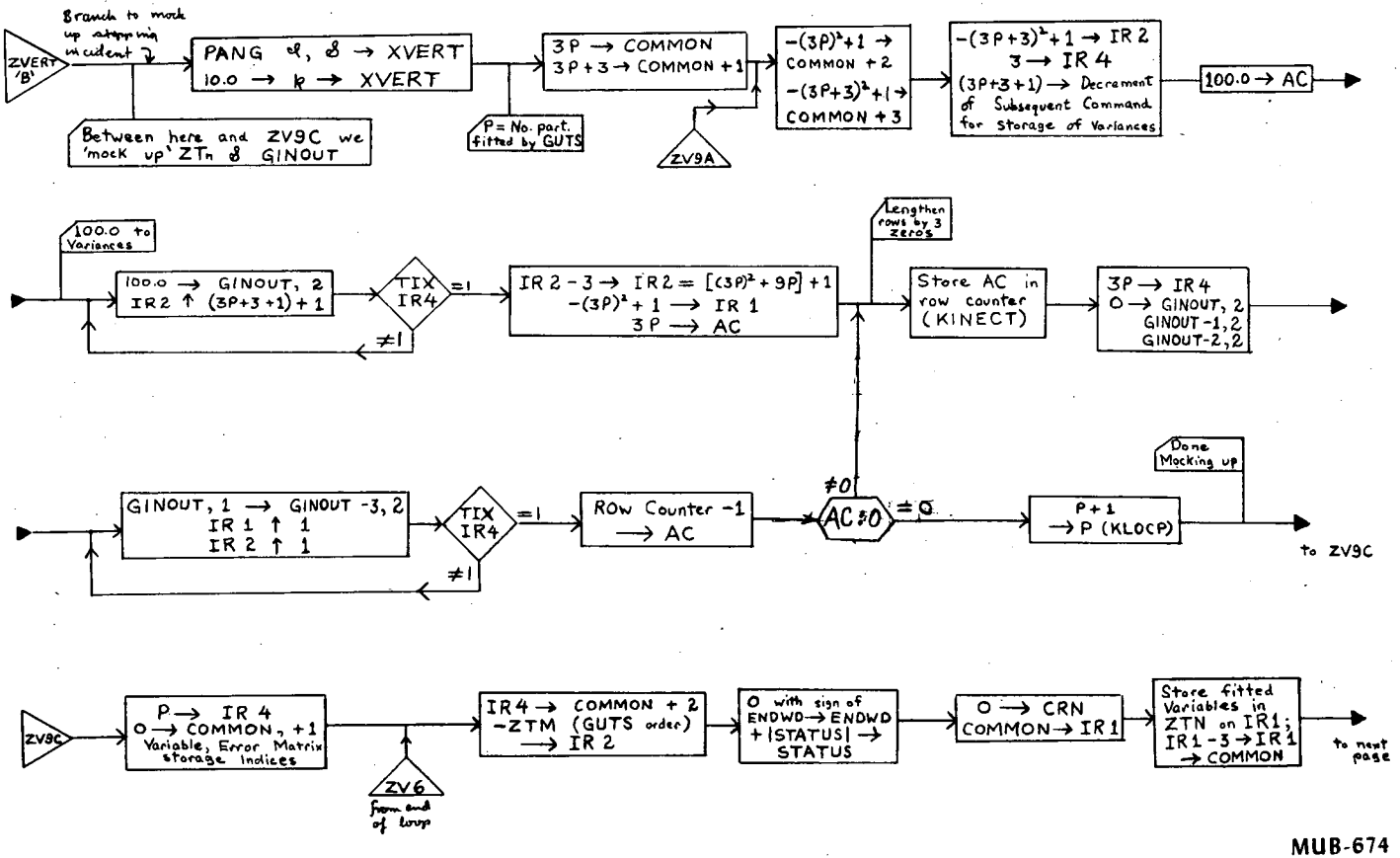
KICK-5 JULY 1960



MUB-675

KICK-5 JULY 1960

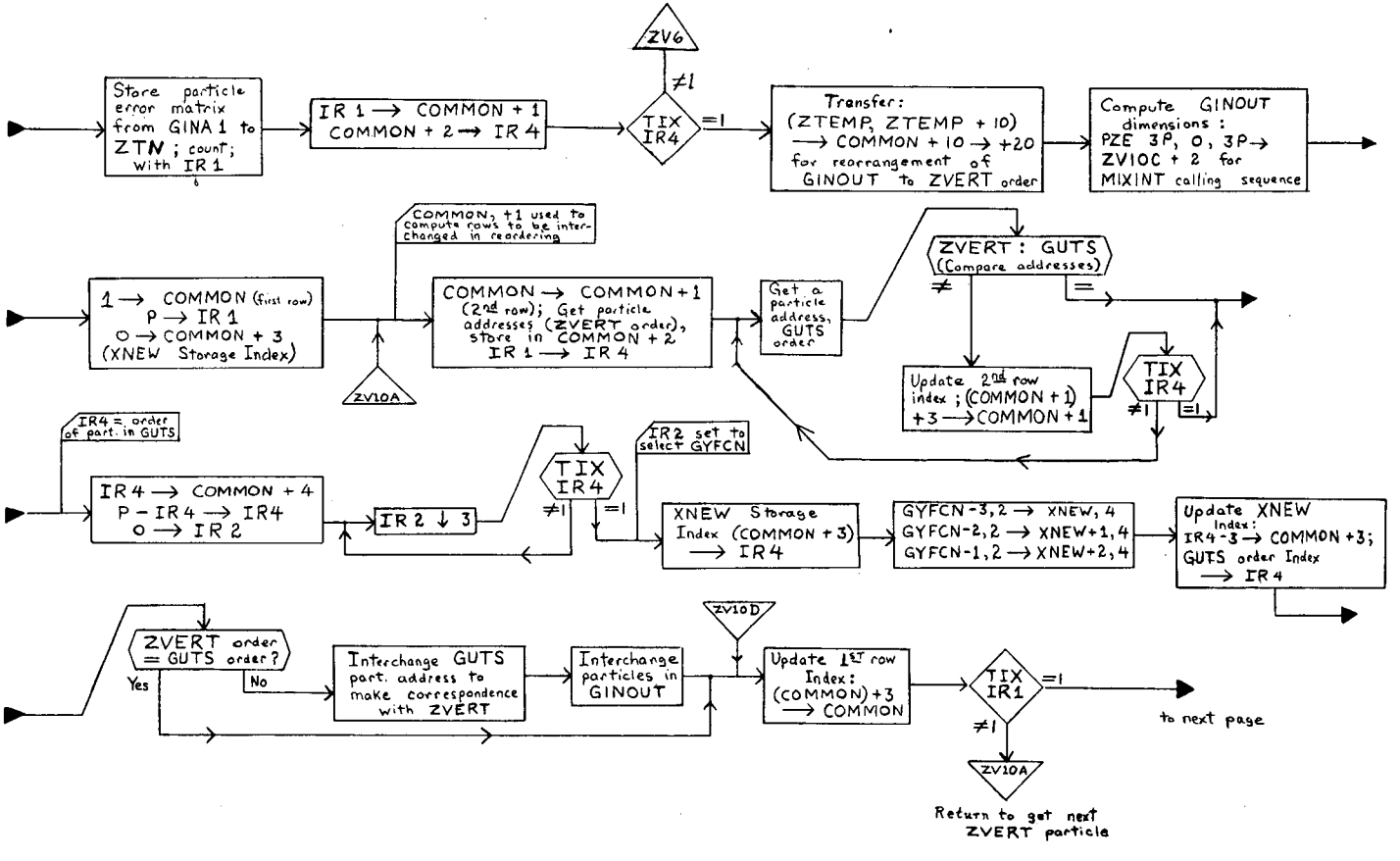
ZVERT 'B' return from GUTS



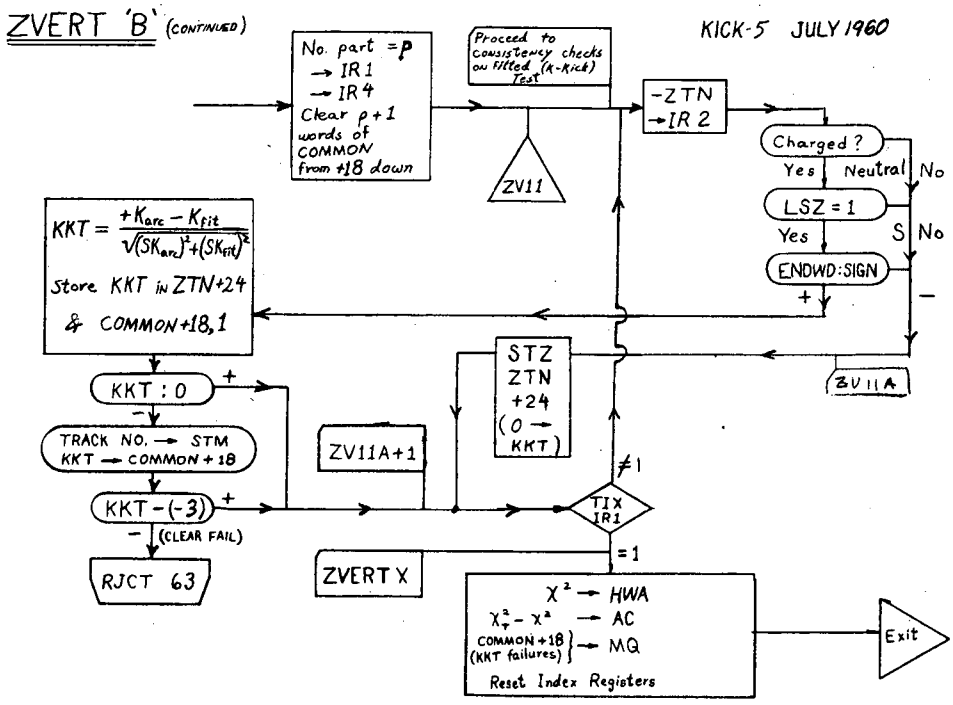
MUB-674

KICK-5 JULY 1960

ZVERT 'B' (continued)



MUB-673


ZVERT 'B' (CONTINUED)
KICK-5 JULY 1960
No. part = P
→ IR 1
→ IR 4
Clear p+1 words of COMMON from +18 down
Proceed to consistency checks on fitted (x-kick) Test
ZV11
-ZTN
→IR 2
Charged ?
Yes
Neutral
No
LSZ = 1
Yes
S
No
ENDWD:SIGN
+
−
KKT = +K_arc − K_fit / √((SK_arc)² + (SK_fit)²)
Store KKT in ZTN+24
& COMMON+18,1
ZV11A
STZ ZTN +24 (0 → KKT)
KKT : 0
+
−
TRACK NO. → STM
KKT → COMMON + 18
ZV11A+1
≠1
TIX IR1
KKT - (-3)
+
−
(CLEAR FAIL)
RJCT 63
ZVERT X
=1
χ² → HWA
χ²₊ - χ² → AC
COMMON+18 (KKT failures) → MQ
Reset Index Registers
Exit

MU-23669

Notes on ZSWIM

General Discussion

ZSWIM is the routine used to transfer the information on a track from one end (where it has participated in a fit) to the other (where it will, in general, be used to get another fit). This "swimming" of information naturally divides into two subtopics, swimming of curvatures and swimming of angles; these are separately discussed below. ZSWIM swims the angles directly but calls on a separate subroutine KSWIM to swim the curvature (using REROUT, a GUTS subroutine).

Swimming of "Angles" ϕ and S

Normal fitted tracks. If Kick had easy access to the magnetic-field components $H_z(x, y, z)$ and to the horizontal component normal to the track $H_{xy}(x, y, z)$, then for relatively short tracks it could swim "angles" ϕ_1 and S_1 from an initial vertex (1) to get ϕ_2, S_2 at a new vertex (2). For example consider a flat track ($\lambda = 0$) and suppose we swim ϕ downstream. Then we have $\phi_2 - \phi_1 \equiv \Delta\phi = \int_0^L (d\phi/dL)dL$, where $d\phi/dL$ is given by

$$(\pm)_c P_c = 0.3 H_z \frac{dL}{d\phi} \text{ (downstream)} \qquad \text{(1 Flat)}$$

with H_z in kgauss, L in cm, Pc in Mev, and with the particle charge indicated by $(\pm)_c$. It is easy to generalize (1-Flat) to dipping tracks if we resolve momentum p and displacement L into components parallel and perpendicular to H_z. Get rid of the parallel components by a Lorentz transformation parallel to H_z. 1-Flat then applies to the perpendicular components $p_c \cos\lambda$ and $L \cos\lambda$, giving:

$$(\pm)_c P_c \cos\lambda = 0.3 H_z \frac{d(L\cos\lambda)}{d\phi} \text{ (downstream).} \qquad (1)$$

If we denote the fitted k by k^F (always > 0), then we have $P_c \cos\lambda = 1/k^F$. Therefore Eq. (1) can be written

$$\frac{d\phi}{d(L\cos\lambda)} = (\pm)_c \, 0.3 H_z \, k^F \text{ (downstream)} \qquad (2)$$

and

$$\Delta\phi = \int_0^{L\cos\lambda} \frac{d\phi}{d\,L'\cos\lambda} \, d(L'\cos\lambda) = (\pm)_c \int_0^L k^F (0.3 H_z) \cos\lambda \, dL'. \qquad (3)$$

To perform this integration, one must take into account the variation along the track of k because of ionization loss and H_z. Luckily, Pang has already done this and passed along $\phi_{beg.}^{Pang}$ and ϕ_{end}^{Pang}. Thus we can form

$$\Delta\phi^{Pang} = \phi_e - \phi_b = \int k^P \cos\lambda \, (0.3 \, H_z) dL \text{ (downstream).} \qquad (4)$$

In this Pang case, Pang has assigned to k^P the observed sign of $\Delta\phi^{Pang}$. Then, to an excellent approximation the value of $\Delta\phi$ given by Eq. (3) is

$$\Delta\phi = (\pm)_c \left[\frac{k^F}{k^P}\right]_{beg.} \Delta\phi^{Pang} \text{ (downstream)} \qquad (5)$$

The reason for choosing "beginning" is discussed later below Eq. (8). We chose to regroup Eq. (5) and write

$$\Delta\phi = (\pm)_c \, k_b^F \left(\frac{\Delta\phi}{k_b}\right)^{Pang} \text{ (downstream).} \qquad (6)$$

Equation (3) shows that the last factor $(\Delta\phi/k_b)^{Pang}$ is $\int (k^F/k^F_{beg.})(0.3 \, H_z) dL$, so writing $\bar{H}_z$ as some effective average, we have

$$\left(\frac{\Delta\phi}{k_b}\right)^{Pang} = 0.3 \, \bar{H}_z \, L, \text{ (downstream),} \qquad (7)$$

which is independent of the particle's charge and the way it curved in the bubble chamber. Actually, for programming convenience, since we have stored (in ZTn+2) the absolute value k and not $(\pm)_c k$, the $(\pm)_c$ factor is associated with $(\Delta\phi/k_b)^P$ to make $(\pm)_c(\Delta\phi/k_b)^P$ which is in JACOB3 and used several times. Notice that $(\pm)_c$ and k^P will usually have the same sign, so that $\Delta\phi$ will usually have the same sign as $\Delta\phi^{Pang}$. But if the track coulomb-scattered so much that the sign of $\Delta\phi^{Pang}$ (and therefore k^{Pang}) is wrong, this is automatically taken care of by the known sign of $(\pm)_c$.

The above example was for a downstream swim. Now $\Delta\phi^P$ is $\phi_e - \phi_b$, not $\phi_2 - \phi_1$, so to take into account the direction of swim we recall that:

Endwrd +ve means angles are set up at the beginning, so must swim down;

Endwrd -ve means angles are set up at the end; so must swim up.

Hence in all formulas, the reservation "Downstream" can be replaced by a factor $(\pm)_{Endwd}$; and (6) becomes in general

$$\phi_2^F = \phi_1^F - (\pm)_{Endwd} \, k_b^F \left[(\pm)_c \left(\frac{\Delta\phi}{k^P}\right)^{Pang}\right]. \qquad (8)$$

A comment is in order on the choice of "beginning" for (k^F/k^P) in Eq. (5). Perhaps (k^F/k^P) middle might be more accurate, but after a fit, we have k^F at the ends, not the middle. It is easier to make k^P_{beg} from P^{Pang}_{beg} at ZTn + 40 than to swim k^F by L/2. The range-momentum relation is nonlinear and "beginning" variables are less skewed, in general, than "end" variables.

Dips are swum by the analogue to Eq. (8).

$$\lambda^F_2 = \left\{ \lambda_1^F \ (\pm)_{Endwd} \ k^F_b \left[(\pm)_c \left(\frac{\Delta\lambda}{k_b} \right)^{Pang} \right] \right\} \tag{9}$$

$$s^F_2 = \tan \lambda^F_2 = \tan \left\{ \quad \text{Eq. (9)} \quad \right\} \tag{10}$$

Fitted two-point charged tracks. For two-point tracks, Pang gives $\phi_b = \phi_e$, $s_b = s_e$. The swim logic could be simplified if Pang arbitrarily assigned say $k = 10^{-4}$ (p = 10 Bev/c) and calculated $\phi_b - \phi_e$, $s_b - s_e$, but this has not been done.

In ZSWIM2P, therefore, $dL/d\phi$ is actually calculated according to Eq. (2), getting H_z from HZ in TEMPAR (HZ = 0 in pure Kick, Event-Type assembly responsible for sign and value). Thus we have

$$\Delta\phi = (\pm)_{Endwd} \ (\pm)_c \ k_B^F \ (0.3 \ H_z) \ L \cos\lambda. \tag{11}$$

For a 100 Mev/c track, at 10 kgauss with L = 1 cm, we have $|\Delta\phi| = 10^{-2}(0.3 \times 10)1 = 0.03$ radian = 2 deg, which seemed to make Eq. (11) big enough to include. However $\Delta\lambda$ is much smaller than $\Delta\phi$, so ZSWIM2P sets

$$\Delta s = 0 \ . \tag{12}$$

Unfitted tracks. To swim these, we merely set up again at the other end.

This concludes the discussion of the swimming of angles, but the flow chart for ZSWIM will be much more intelligible if the reader looks next at the following paragraphs on KSWIM and Error Transformation in ZSWIM, and then at the flow chart for KSWIM before finally reading the flow chart for ZSWIM.

Swimming of Curvature

This is done via a closed subroutine called KSWIM, principally designed as a subroutine of ZSWIM. Its function is to compute k (hence its name) at one end of a track, given some input k in track banks. It is also used by ZSET directly to swim the curvature from the middle of the track to the desired end.

To understand the operation of KSWIM, first let us discuss the magnitude of Endwd, whose sign has already been discussed under "Swimming of Angles," i.e. +/- for "Angles" ϕ and λ set up at the beginning/end. If tracks were always set up with k at the same place as the angles (as was once nearly true in earlier versions of Kick) or if tracks were always set up with k at the middle of the track (as we would now like to do if it did not involve work, i.e. reprogramming GUTS), then this sign information would be sufficient. In fact, in our normal procedure if the "angles" of a track are set up at the beginning (Endwd +) then k is also set up there (except internally to ZSET). However if the direction is set up at the end, k may be set up either at the middle or the beginning (depending on whether the track is set up from Pang data or received from a fit at a previous vertex) and will later be set at the end of the track if a fit is obtained. Since we would like to be able to tell, from the track bank alone, where the track has been set up, we have had to add additional bits to Endwd to tell us this. This means that some of the information in Endwd is redundant if one assumes that Status, which tells whether the track has ever been fitted, plus STM and CPM (in MBANK), which tells the result of the last fit, are available. However we may at some time wish for multivertex events to merge tracks from different vertices (particularly in Examin) in which case the information from MBANK would not be available.

The arbitrary convention adopted for the magnitude of Endwd is to use it to give the distance D, in units of L/2 between the end where the angles are set up, and the place where k is set up. Thus we define

|Endwd| = 0 means k is stated at same end of track as angles, D=0,
= 1 means k is stated at middle of track, D=L/2,
= 2 means k is stated at opposite end of track from angles, $D=2\frac{L}{2}$.

In KSWIM the curvature is swum by the distance D from wherever it is stated on input to the opposite end of the track from that indicated by the Endwd. (Clearly ZSWIM must not change the sign of Endwd until after KSWIM has operated. Moreover ZSET2B which uses KSWIM to get from the middle to the end of a track must artificially change the sign of Endwd before entry and restore it later to achieve a swim in the right direction.) In normal Kick processing only the swim upstream is actually used, since with the automatic use of the D feature of GUTS, we do not swim curvatures downstream. However we visualized a physicist wanting to see a printout of a track swum down to its end, so KSWIM will swim downstream. This means that ZSWIM must bypass KSWIM for Endwd +.

REROUT, the subroutine of GUTS that does the actual swimming is described in Memorandum 192. It uses a lookup Table described under RINTOP (R into P) in the Subroutine Section. The REROUT error return occurs when, for a downstream swim (which we seldom use), the range computed from the curvature is less than the distance you attempt to swim. KSWIM in this case merely sets the AC negative as a signal [Note that in Assembly 4, k was set to $1(\text{Mev/c})^{-1}$ and $(\delta k)^2$ to $100\ (\text{Mev/c})^{-2}$]. This of course will not occur for the normal procedure of not swimming k downstream.

KSWIM does not alter any of the contents of the track banks, neither curvatures nor Endwd.* Any routine using it must take care of storing the swum k, modifying the error matrix, and changing the Endwd. In the notation explained in detail in the ERRORS section of SWIM, the quantities $A_{33}(\partial k_2/\partial k_1)$ and $A_{32}(\partial k_2/\partial s_1)$ are computed in KSWIM (by REROUT) and stored in REWS and REWS + 1.

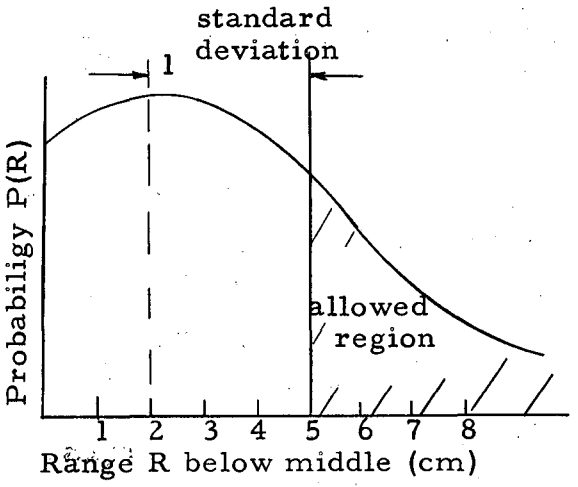
"Overstopped" tracks

A difficulty arises in the error propagation when an "overstopped" track is swum upstream. As an example and definition of such an "overstopped" track, consider a recoil proton track of length L = 10 cm which

*In assembly 4, KSWIM stored swum k in track banks, but did not change errors. KSWIM always swam tracks at distance L (so that if necessary any user would have to change L to $\frac{L}{2}$ in ZTn + 19 to accomplish a halfway swim).

leaves the chamber just before it stops, and is measured "leaving". At its middle, its momentum P(L/2) is 220 ± 40 Mev/c. Then 1/6 of such tracks would Coulomb scatter by one standard deviation or more in such a direction as to give $P \lesssim 180$ Mev/c and a calculated range beyond the middle of $\lesssim 2$ cm (whereas we know L/2 equals 5 cm).

The truncated a priori probability distribution for the range (calculated from k_{middle} and k_{middle}) is shown in the sketch; the allowed region is clearly non-Gaussian.



How should Kick describe the equivalent truncated k- distribution? Some users may want to replace it with a Gaussian centered at k(L/2 + E) with some appropriate δk, but at Berkeley we have chosen to stick with the measured variables until after the fit, so as to postpone the problem of working with skewed distributions. After the fit, k-Kick-Test will check R-L [actually $k(L)-k^F$] and for $R < L$ will give reject HRJ62. This warns the physicist that only a part of the fitted probability distribution $P(R^F)$ is consistent with L, just as only part of P(R-measured) was consistent before the fit.

There still remains the question of how ZSET should swim upstream by L/2 with δk, $\partial k_2/\partial s$ etc. so as to set up a track at the beginning:-First note how δk is swum for nonoverstopped tracks (see Memorandum 86, Sect IV). We assume $p \propto R^a$ [$a = 0.27$], so we have $k = 1/ps \propto 1/R^a s$. Then, ignoring for a moment variations in the slope s, we have at the beginning

$$\frac{\delta k_2}{k_2} = -a \frac{\delta R}{R_2} ,$$

and at the end

$$\frac{\delta k_1}{k_1} = -a \frac{\delta R}{R_1} .$$

Notice that δR is the same quantity in both equations; divide them:

$$\frac{\partial k_2}{\partial k_1} \equiv A_{33} = \frac{k_2}{R_2} \cdot \frac{R_1}{k_1} \quad \delta k_1 = \frac{p_1 R_1}{p_2 R_2} = \left(\frac{R_1}{R_2}\right) 1+a = 1.27 \quad \text{(understopped)} \tag{13}$$

Suppose we were to apply Eq(13) to our overstopped proton. If we were to believe k^{Pang}, ignoring the fact that the track is overstopped, then we have $R_1 = 2$ cm, $R_2 = 2 + \frac{L}{2} = 7$ cm, and $R_1/R_2 = 2/7$. On the other hand, we know $R_1 \leq 1/2$, and we have $R_1/R_2 \geq 1/2$. Thus, if we use the coefficient $A_{33} = (2/7)^{1.27}$ as computed by REROUT, we will have δk_2 smaller than it should be by at least $[(2/7)/(1/2) = 4/7)]^{1.27}$. Now we don't want to work with nonGaussian errors, but we do want the width of the Gaussian to be as reasonable as possible. Hence, in KSWIMB+4, we replace A_{33} as given by Eq. (13) by

$$A_{33} = \left(\frac{1}{2}\right)^{1.27} = 0.414 \quad \text{(overstopped)}. \tag{14}$$

A similar correction must be made in $\partial k_2/\partial s \equiv A_{32}$. As derived in Memo 86, page IV-6, this is normally given by

$$\frac{\partial k_2}{\partial s} \equiv A_{32} = -\frac{D}{R_2} \frac{s}{k_2 P_2^2} \quad \text{(normal)} \tag{15}$$

For an overstopped track, we can take k_2 from k via arc as given by Pang. Writing $P_2^2 = 1+s^2/k_2^2$ and $D/R_2 = 1/2$, we have

$$A_{32} = -\frac{1}{2} \frac{s \cdot k(\text{arc})}{1+s^2} \quad \text{(overstopped)}. \tag{16}$$

Errors

The errors are swum by modifying the variance matrix from the fit. Before modification this is $(G^{-1})_{ij} = \overline{\delta X_i \delta X_j}$. It becomes

$$(G^{-1})^{\text{swum}}_{i\ell} = \sum_{jk} A_{ij} G^{-1}_{jk} A^{T}_{k\ell}$$

where

$$A_{ij} = \frac{\partial X_{i2}}{\partial X_{j1}} = \begin{pmatrix} \frac{\partial \phi_2}{\partial \phi_1} & \frac{\partial \phi_2}{\partial s_1} & \frac{\partial \phi_2}{\partial k_1} \\ 0 & \frac{\partial s_2}{\partial s_1} & \frac{\partial s_2}{\partial k_1} \\ 0 & \frac{\partial k_2}{\partial s_1} & \frac{\partial k_2}{\partial k_1} \end{pmatrix}$$

Note that in the GUTS write-ups $\frac{\partial k_2}{\partial s_1}$ is referred to as B, and $\frac{\partial k_2}{\partial k_1}$ is referred to as A.

The elements on the first row come from differentiating (8).

$$\phi_2 = \phi_1 + k_b^F \left[\frac{\phi_2 - \phi_1}{(\pm)_c k_b} \right]^{\text{Pang}} \equiv \phi_1 + k_b^F \, [\text{const.}], \qquad (8)$$

where b stands for beginning. Thus if we were to swim downstream (which is rare) k_b is known, then we have

$$A_{11} = \frac{\partial \phi_2}{\partial \phi_1} = 1, \; A_{12} = 0, \; A_{13} = \frac{\partial \phi_2}{\partial k_1} = \frac{\partial \phi_2}{\partial k_b} = \left[\frac{\phi_2 - \phi_1}{(\pm)_c k_b} \right]^{\text{Pang}}$$

(17-down)

In general we swim up, so we have

$$\frac{\partial \phi_2}{\partial k_1} = \frac{\partial \phi_2}{\partial k_b} \frac{\partial k_B}{\partial k_1} = \frac{\partial \phi_2}{\partial k_b} \frac{\partial k_2}{\partial k_1} = \frac{\partial \phi_2}{\partial k_b} A_{33},$$

and

$$\frac{\partial \phi_2}{\partial s_1} = \frac{\partial \phi_2}{\partial k_b} \frac{\partial k_b}{\partial s_1} = \frac{\partial \phi_2}{\partial k_b} \frac{\partial k_2}{\partial s_1} = \frac{\partial \phi_2}{\partial k_b} A_{32},$$

and therefore

$$A_{13} = [A_{13}(\text{down})] A_{33} \text{ and } A_{12} = [A_{13}(\text{down})] A_{32}. \quad (17\text{-up})$$

For two-point tracks (as already mentioned) Eq. (8) goes over into Eq. (11) so we have

$$A_{13}(\text{down}) = (\pm)_{\text{Endwd}} (\pm)_c (0.3 H_z) Ls \quad (17\text{-2P})$$

The elements of the second row come from differentiating Eq. (9):

$$s_2 = \tan \{\lambda_2\} = \tan \{\lambda_1 + k_b [\text{const}]\}$$

$$ds_2 = \sec^2 \lambda_2 (d\lambda_1 + [\text{const}]\, dk_b)$$

$$= \sec^2 \lambda_2 \left(\frac{ds_1}{\sec^2 \lambda_1} \right) + [\text{const}]\, dk_b \quad (9)$$

Writing $\sec^2 \lambda = 1 + s^2$, we have

$$A_{22} = \frac{\partial s_2}{\partial s_1} = \frac{1 + s_2^2}{1 + s_1^2} \text{ and } A_{23} = \frac{\partial s_2}{\partial k_1} = (1 + s_2^2) \left[\frac{\lambda_2 - \lambda_1}{(\pm)_c k_b} \right]^{\text{Pang}}. \quad (18)$$

For two-point tracks we set the constant in Eq. (9) equal to zero, so we have

$$A_{23} = 0 \quad (18\text{-2P})$$

The elements in the third row were discussed in KSWIM and are evaluated by REROUT, internal to KSWIM. On return from KSWIM, A_{33} is in REWS, A_{32} in REWS+1.

The matrix multiplication $(\underset{\sim}{G}^{-1})_2 = \underset{\sim}{A} (\underset{\sim}{G}^{-1})_1 \underset{\sim}{A}^T$ is performed using MMATRIX (see subroutines), where $\underset{\sim}{A}$ is loaded into a block starting at JACOB1, $\underset{\sim}{A}^T$ is loaded into a block starting at JACOB2, and $\underset{\sim}{G}^{-1}$ is already in a block starting at ZTn+5. First $\underset{\sim}{A} \times \underset{\sim}{G}^{-1}$ is formed and stored at JACOB3; C(JACOB3) $\times \underset{\sim}{A}^T$ is then formed and stored at ZTn+5.

In the matrix operations above, we assume that G^{-1} is a 3-by-3 matrix, i. e. that CRN = 0. If this is not the case, ZSWIM treats the track as if it were unfitted, and swims by setting it up at the other end. Event-type programmers should notice that for CRN $\neq$ 0 the above procedure is inappropriate. Such cases have not arisen yet.*

Note for the Future

The preceding paragraphs have described what Kick now does on swimming curves and angles. In this memo, suggestions as to possible future alterations in the swimming of angles will be discussed. (We know of no significant mistakes in our swimming of curvatures from one end of the track to the other.)

It will be convenient to begin at the beginning, i. e. with Pang. Pang fits tracks by making a least-squares adjustment of curvature and direction of the track to the measured film coordinates. For the case in which errors on the individual points are independent and Gaussian distributed (as we assume for true measurement errors), this gives us the "best" values of the output quantities. However, when the errors on the individual points are correlated (as is true for errors caused by multiple Coulomb scattering and possibly turbulence) this is no longer true. At present Pang throws away some of the information relevant to the determination of directions at the ends. The "best" answers could be obtained from a much more complicated least-squares fit using not only the full correlated error matrix but several extra parameters. The complexity of doing this correct sort of fitting is such as to make it completely impractical. A compromise could be arrived at by using the present calculation for the curvature and a different calculation for the angle at each end in which the weight of the measured points decreases as you get further from the point at which the direction is required, or even using only the nearest segment of the track to make the fit. Possibly Pang will in the future provide some such answer.

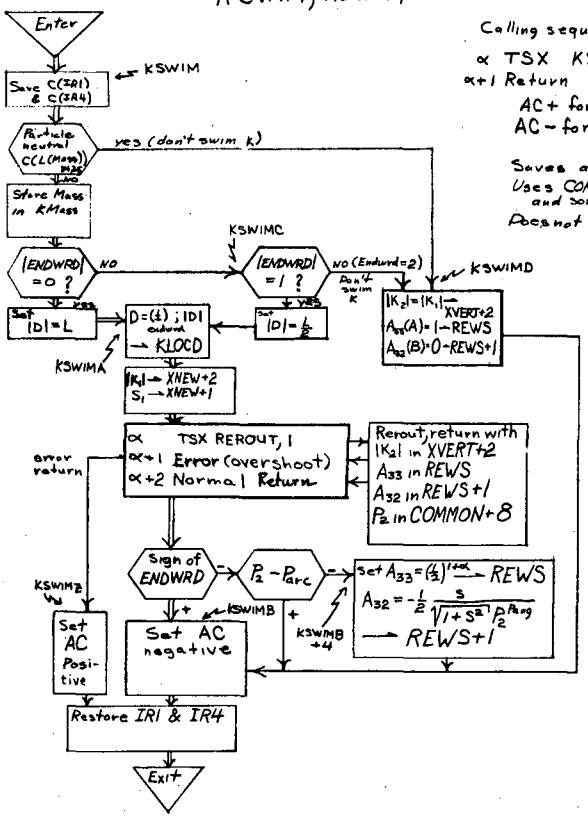
---

*The only rare example of such a case that we have thought of so far is as follows: Consider a I, II sequence of vertex fits in which we desire to swim back to vertex I, for example to make the I, II, II', I' validity check. Suppose fit I is successful, but the connecting track badly determined, so that ZVERT during the II fit increases its CRN; and then the II fit fails.

If after this double trouble one invokes SWIM, it will simply set the connected track up again; whereas actually clever programming could have it resurrect the real 3-by-3 error matrix and swim properly.

Even without this information, Kick could, and at some future date may, do better than it now does for long tracks. Here the cumulative effect of Coulomb scattering will greatly reduce the correlation of angles at the two ends. We would probably be better off to keep the Pang angles for sufficiently long tracks rather than swimming. We have not done this yet, primarily because the error matrix would be somewhat messy to compute, since after the fit there are correlations between various quantities, which are no longer correct if we revert to Pang angles. If it is correct to use the original Pang angles, one should probably zero all correlations, but this is not certain since k, for instance, depends on s through the swim procedure, if not otherwise. More thought is needed on this point.

KSWIM, Assembly 5 July 14, 1960



Calling sequence -ZTn in IR2

α TSX KSWIM,4

α+1 Return

AC+ for normal swim

AC− for REROUT error return

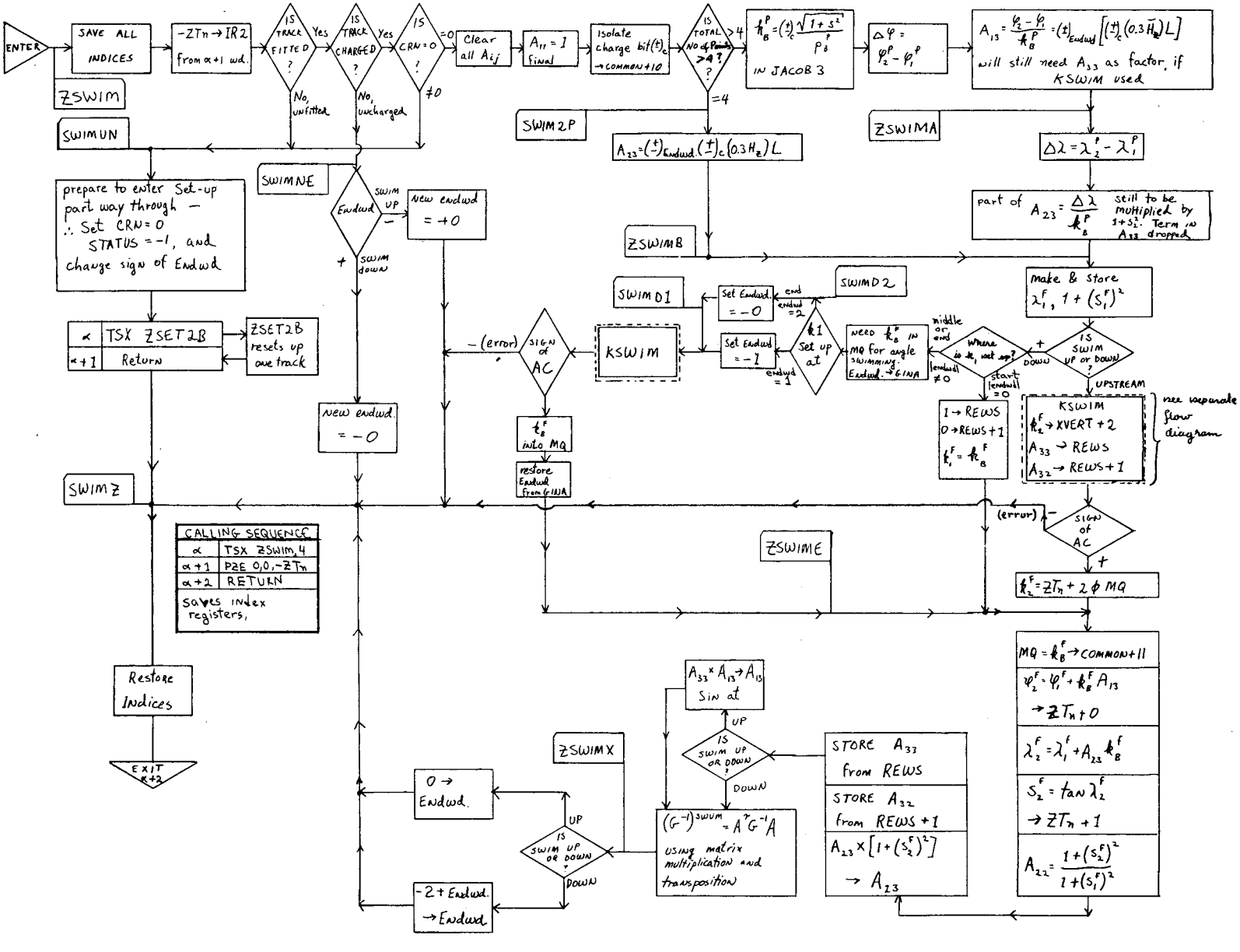
Saves all index registers

Uses COMMON to COMMON+9 and some GUTS storage

Does not alter track bank.

MU−23672

ZSWIM for Assembly 5, Sept 5, 1960 Calling sequence, See lower left of page



5 Sept. 1960 - DRW.

UCRL-9099
Processing:SWIM-13
July 1960

Notes on ZEND

A. General Description

Each Event Subprogram ends with TRA ZEND, which then does quite different things depending on whether or not there has been a successful fit to a hypothesis as determined by the Good Print Mark (GPM).

The history of GPM (in MBANK) is as follows: At exit from ZVERT, the AC is (+/-) according to whether the fit was (successful/unsuccessful). Entry into the report Routine PRINT with AC + then will cause a "good" print (sign of MB+34 positive), with the GPM advanced by unity. If one vertex of a two-vertex hypothesis succeeds but the second fails, GPM would misleadingly be nonzero; it is up to the event-type subprogram writer to clear GPM before the last vertex of a multivertex hypothesis.

The main functions of ZEND are:

(a) For an event in which no successful fit was made, a search is made for -ve stopping particles. If one is found, the LSZ word in ZTn+15 of the first one discovered is changed to indicate a leaving particle, and the event is recalculated (i. e. a charge exchange is tried on the track).

(b) For successful events a search is made for non-zero Future Track Numbers, and if no track is to be saved for the next event (all Ftn's = 0), control is returned to ZKICK8 to clear the pick-up banks and read in the next event.

(c) If a track is to be saved for the next event, the variables are swum (by ZSWIM) to the other end of the track (if it is to be connected up to another event, the variables are needed at the end away from the vertex which has been fitted). Then the contents of the track bank are transferred to a pick-up bank. When all tracks to be picked up have been transferred, exit is made to ZKICK2 to read the next event.

B. Detailed Notes

These notes should be used with the flow diagrams at the end of this section. The numbers in brackets on the flow diagram compare to numbered items below:

(1) Two flags in common + 17 and common + 18 are zeroed here. Nonzero words are stored later in:

(a) common + 17 if the Pick-Up-Command was even, i. e. if a pick-up was made earlier

(b) common + 18 if any number of tracks have a nonzero Ftn.

(2) Good Print Mark in MBANK + 14 is advanced by 1 (from 0) each time a "good" print is made after a vertex fit (i. e. entry to PRINT with AC +) If we have GPM = 0, no good fit has been found. One possible explanation is that a negatively charged particle has been called stopping (LSZ word set = 2), whereas it really underwent charge exchange and so left the chamber (in which case, of course, k from curvature should have been used rather than k from arc length.)

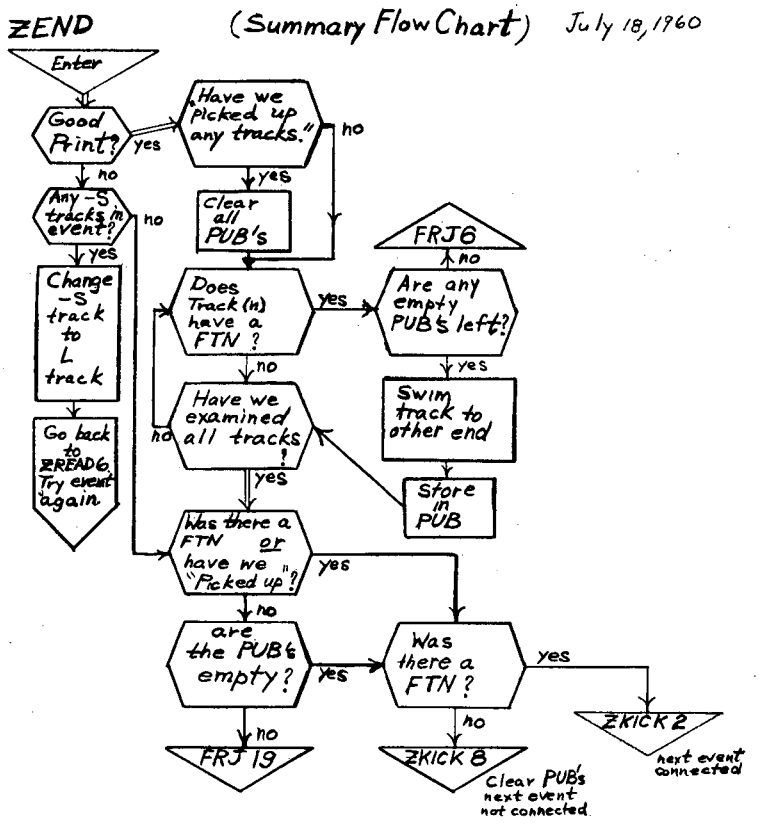
(3) A loop looks for -ve S tracks and changes them to L's, followed by exit to ZREAD6 to reprocess the event under this new assumption. If a track has already been picked up (|STATUS| = 2 in ZTn + 3) from a previous measurement, this change is not carried out, for two reasons:

(a) If it is done exactly as for normal tracks (change -S to L in Pang Appendage to ZTn, but not in Pick-Up Banks) we simply get a loop, since ZSET overwrites Pang Appendage in ZTn.

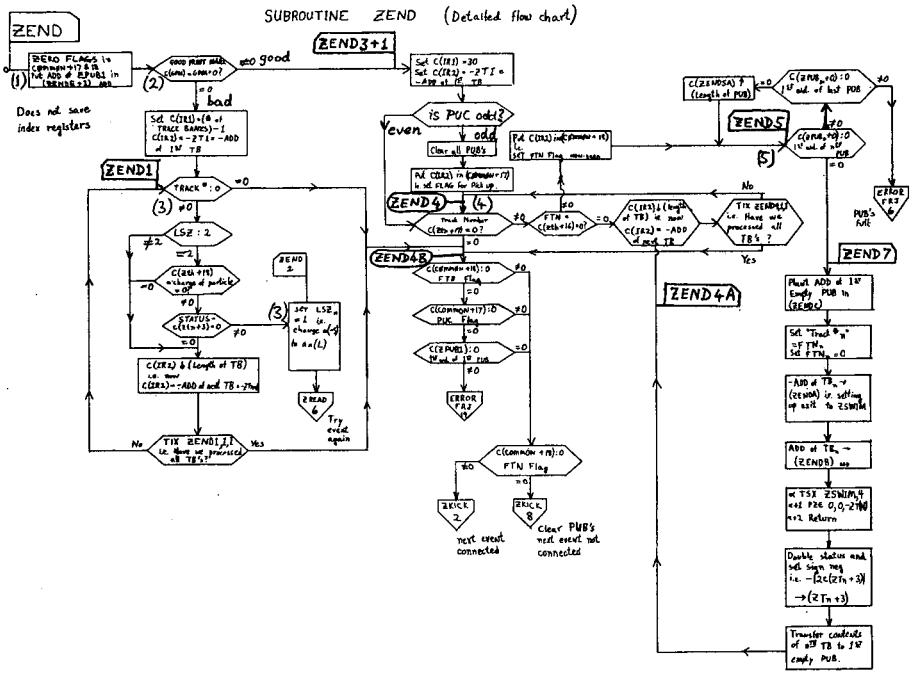
(b) Charge exchange of a connected event is likely to have been already caught and fixed when it was processed.

(4) Loop at ZEND 4 examines track numbers in successive tracks. On finding the first numberless track, it exists from the loop. If a numbered track has a nonzero Ftn, it is transferred to a PUB (ZEND5) and then control returns to the loop again searching for more tracks with nonzero Ftn's. When all track banks have been examined, control goes to a check-before-exit routine starting at ZEND4B. Before leaving ZEND, a test is made which could result in FRJ 19, namely that if there are tracks stored in the PUBs, then either (1) there must have been a pickup, or (2) some track must have a nonzero Ftn. If this test is passed, then we leave ZEND via ZKICK8 (which clears the PUBs if there is no Ftn) or ZKICK2 (if there is an Ftn).

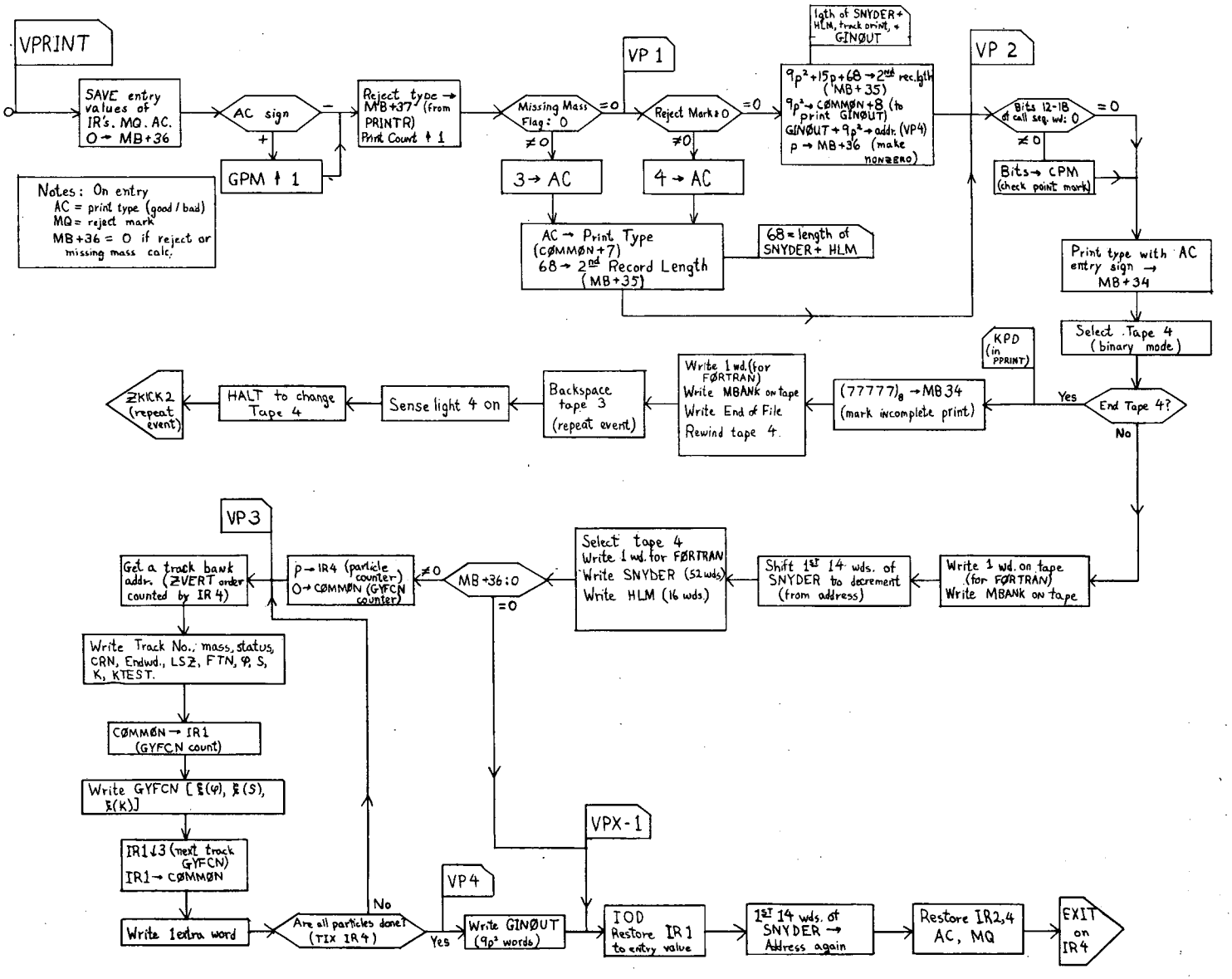
(5) The test at ZEND5 checks that all PUB's are not full (if they are full, FRJ6 results; see list of rejects). If the PUB's are not full, the variables are swum to the other end of the track and then transferred to a PUB.


ZEND
(Summary Flow Chart)
July 18, 1960
Enter
Good Print?
yes
no
Have we "picked up" any tracks."
no
yes
Any -S tracks in event?
no
yes
Clear all PUB's
Change -S track to L track
Go back to ZREAD6. Try event again
FRJ6
no
Does Track (n) have a FTN ?
yes
Are any empty PUB's left?
no
yes
Have we examined all tracks !
no
yes
Swim track to other end
Store in PUB
Was there a FTN or have we "Picked up"?
yes
no
are the PUB's empty?
yes
Was there a FTN ?
yes
no
FRJ 19
ZKICK 8
Clear PUB's next event not connected
ZKICK 2
next event connected

MU-23671


SUBROUTINE ZEND (Detailed flow chart)
ZEND
ZERO FLAGS in COMMON+17 & TB
Does not save index registers
GOOD PART WORD
good
bad
ZEND3+1
Set C(IR1) = 20
Set C(IR2) = -2TI = -ADD of 1ST TB
is PUC odd?
even
odd
Clear all PUB's
Put C(IR2) in COMMON+17
to set FLAG for Pick up
ZEND1
TRACK # : 0
LSZ : 2
C(IR2) & (Length of TB) is now C(IR2) = -ADD of next TB
STATUS = C(TB+3) = 0
Set LSZ = 1 to change at (4) to a=(L)
ZEND 2
READ 6
Try event again
TIX ZEND1,1 i.e. Have we processed all TB's?
No
Yes
ZEND4
Track Number C(IR2+7) = 0?
FTN = C(TB+14) = 0?
C(IR2) & (length of TB) is now C(IR2) = -ADD of next TB
TIX ZEND4,1 i.e. Have we processed all TB's ?
No
Yes
ZEND4B
C(COMMON+16): 0 FTB Flag
C(COMMON+17): 0 PUC Flag
C(IPUB): 0 1st wd of 1st PUB
ERROR PRT 5
C(COMMON+16): 0 FTN Flag
2KICK 2
next event connected
2KICK 8
Clear PUB's next event not connected
ZEND4A
C(ZEXPDSA) & (Length of PUB)
C(ZPUB,+0): 0 1st wd of last PUB
C(ZPUB,+0): 0 1st wd of 1st PUB
ZEND5
ERROR PRT 6
PUB's Full
ZEND7
Place ADD of 1st Empty PUB in (ZEND)
Set "Track # n" = FTN, Set FTN, = 0
-ADD of TB, → (ZEND4) in setting up exit to ZSWIM
ADD of TB, → (ZEND8) etc
TSX ZSWIM,4 exit P26 0,0,-2TN+2 Return
Double station and set sign neg i.e. -(2c(ZTn+3)) → (ZTn+3)
Transfer contents of all TB to 1st empty PUB.

MU-24029


VPRINT
SAVE entry values of IR's. MQ. AC. 0 → MB+36
AC sign
+
−
GPM + 1
Notes: On entry
AC = print type (good / bad)
MQ = reject mark
MB+36 = 0 if reject or missing mass calc.
Reject type → MB+37 (from PRINTR)
Print Count + 1
Missing Mass Flag: 0
≠0
=0
VP 1
Reject Mark ≠ 0
=0
≠0
3 → AC
4 → AC
AC → Print Type (CØMMØN+7)
68 → 2nd Record Length (MB+35)
68 = length of SNYDER + HLM
lgth of SNYDER + HLM, track print, + GINØUT
9p² + 15p + 68 → 2nd rec. lgth (MB+35)
9p² → CØMMØN+8 (to print GINØUT)
GINØUT + 9p² → addr. (VP4)
p → MB+36 (make NONZERO)
VP 2
Bits 12-18 of call seq. wd: 0
=0
≠0
Bits → CPM (check point mark)
Print type with AC entry sign → MB+34
Select Tape 4 (binary mode)
End Tape 4?
Yes
No
KPD (in PPRINT)
(77777)₈ → MB 34 (mark incomplete print)
Write 1 wd. (for FØRTRAN) Write MBANK on tape Write End of File Rewind tape 4.
Backspace tape 3 (repeat event)
Sense light 4 on
HALT to change Tape 4
≠KICK 2 (repeat event)
Write 1 wd. on tape (for FØRTRAN) Write MBANK on tape
Shift 1st 14 wds. of SNYDER to decrement (from address)
Select tape 4 Write 1 wd. for FØRTRAN Write SNYDER (52 wds) Write HLM (16 wds)
MB+36: 0
≠0
=0
p → IR4 (particle counter) 0 → CØMMØN (GYFCN counter)
VP3
Get a track bank addr. (≠VERT order counted by IR 4)
Write Track No., mass, status, CRN, Endwd., LS≠, FTN, φ, S, K, KTEST.
CØMMØN → IR1 (GYFCN count)
Write GYFCN [Σ(φ), Σ(S), Σ(K)]
IR1 ↓3 (next track GYFCN) IR1 → CØMMØN
Write 1 extra word
Are all particles done? (TIX IR4)
No
Yes
VP4
Write GINØUT (9p² words)
VPX-1
IOD
Restore IR1 to entry value
1st 14 wds. of SNYDER → Address again
Restore IR2,4 AC, MQ
EXIT on IR4

dw - 3 Aug. 1960

UCRL-9099
Processing:QVRT-1
April 10, 1961
S. P. Almeida and P. G. Burke

Notes on QVRT

Introduction

This memorandum describes the operation and use of a multi-vertex processing routine in Kick called "QVRT", which automatically fits, swims and stores fitted track banks.

The octal code words associated with each entry to QVRT are designed primarily to allow them to be constructed automatically by some "master routine" yet to be written. The input to the master routine should be the minimum required to specify uniquely the particular processing that a physicist requires for an event.

This memo is written with the following two purposes in mind. First, QVRT can now perform automatically several of the chores associated with current event-type writing, and it may be convenient for people who have lengthy event-type routines to write, to learn this calling sequence. Second, we would like to have suggestions from people concerning what they think is a minimum input for the master routine and also suggestions regarding modifications to the present version of the QVRT routine in order to increase its usefulness.

Part A describes QVRT itself and Part B illustrates its use in programming event-types 30 and 40.

Part A

General description of QVRT. The following is a list of operations now performed by QVRT:

(a) A test on the K-Pang test word of every track on the first entry to QVRT (using KPTST)

(b) Two missing-track set ups (in ZT30 and ZT31) to allow the GUTS entry 2VZL2C (using ZSET)

(c) Up to seven extends (using EXTEND)

(d) Either a normal vertex fit or a missing-mass calculation plus a normal vertex fit

(e) Labelling up to and including 70 vertex fits with their corresponding check-point marks.

(f) Storing all chi-squres (good and bad) into LIST AA according to their CPM. Reject numbers for their correspondingly bad or no vertex fits are stored into LIST B with reference to CPM.

(g) Printing-out the chi-square and reject-number LISTS AA and B respectively, at the end of each event type.

(h) Swimming and storing in fitted banks the tracks required at the next vertex.

(i) Bringing back from "fitted banks" all tracks previously fitted at the other end of the other vertex and required for the fit at the current vertex. If any of the previous vertices linked to the current vertex by a track stored in the fitted banks failed the fit, then the current vertex fails.

(j) Automatically carrying out a vertex print when the fit is good.

(k) A coplanarity test on the Pang data at a three-track vertex using the COPLAN routine. The results are stored in LIST C.

(l) Examination of each vertex for tracks with LSZ=3. Detection of such a track causes the vertex to be skipped. If the missing track connects also to another vertex, the marker 222.00 is stored in LISTB; if it does not, 333.00 is stored in LISTAA.

QVRT Calling Sequence:

```
a       TSX  QVRT, 4, X
a + 1   OCT  XXXXXX XX XX XX
                   a    b  c  a
a + 2   OCT  XX XX XX XX XX XX
a + 3   OCT  XX XXXXXX XX X X
                d      e     f  g h
a + 4   OCT  XXXXXXXXXXX X
                     i       j
a + 5   RETURN
```

The calling-sequence code words are defined as follows:

a — X = 0: no missing-mass calculation, only a normal fit
X = 16: missing-mass calculation with print and also normal fit

a + 1 — a: eight octal bits corresponding to the first ZVERT octal code word
b: first missing-mass set up in ZT31
c: second missing mass set up in ZT30. The numbers b and c refer to appropriate locations in the mass list.

a + 2 — second octal code word of ZVERT

a + 3 — d: check point mark (CPM) ranging from 1 to 63. The fitted tracks are stored away in the fitted track banks (AETS bank) according to these CPM. The value zero is not to be used.

a + 3 — e: three 6-bit words indicating that a transfer from the fitted track banks (AETS) to the ZTN banks is required. The 6 bits give the CPM number of the previous vertex where the stored track was fitted. If any of the linked vertices failed, the current vertex is automatically failed.

a + 3 — f: 01 if coplanarity test on Pang data is desired; 00 otherwise.
g: one bit, where 0 indicates no set up to be done;
1 indicates set up all tracks. The ZSET octal code word has already been stored in the appropriate location of QVRT;
2 indicates set up all tracks using the (a + 4) word as the ZSET octal code word.

a + 3 — h: one bit, where 0 indicates print if a good fit was obtained;
1 indicates print always (use one only on the last fit).

a + 4 — i: Eleven octal bits of the ZSET octal code word
j: One bit, gives the number of EXTEND's to be done, ranging from 0 to 7

A note on EXTEND. At present, the EXTEND octal code words must precede the first TSX QVRT, 4 entry. (namely; (PZE-ZTN, T, -ZTN) refer to EXTEND subroutine for its interpretation) Thus the appropriate EXTENDS can be carried out at any QVRT entry where required.

In later assemblies, EXTEND may operate on the Pang part of the track banks rather than the Kick part, thereby eliminating its need in the first entry of the QVRT calling sequence.

The fitted track banks. In 63 storage-bank locations, each 20 words long, QVRT stores one track from each fit. The banks are referred to as AETS banks. The ZTN number (address) is stored in the first location of an AETSN track bank, and the first 19 locations of ZTN appropriately swum are stored into the following 19 locations of the AETSN bank. Since there is only room to store one track in the "fitted banks" from each vertex, the convention is made that the first track referred to in the octal code word (a + 2) is stored, unless this happens to be ZT1. In this latter case the second track referred to in the (a + 2) code word is stored. The programmer must insure that he orders the tracks in his (a + 2) code word so that tracks to be used later at another vertex occur first.

In the QVRT program, the banks are defined accordingly:

AETS 1	BSS	20
AETS 63	BSS	20.

When the fitted tracks are needed for a particular vertex fit, they can be transferred back to their original ZTN bank by inserting the appropriate CPM bits into the (a + 3) QVRT code word (position e, provides for three such transfers).

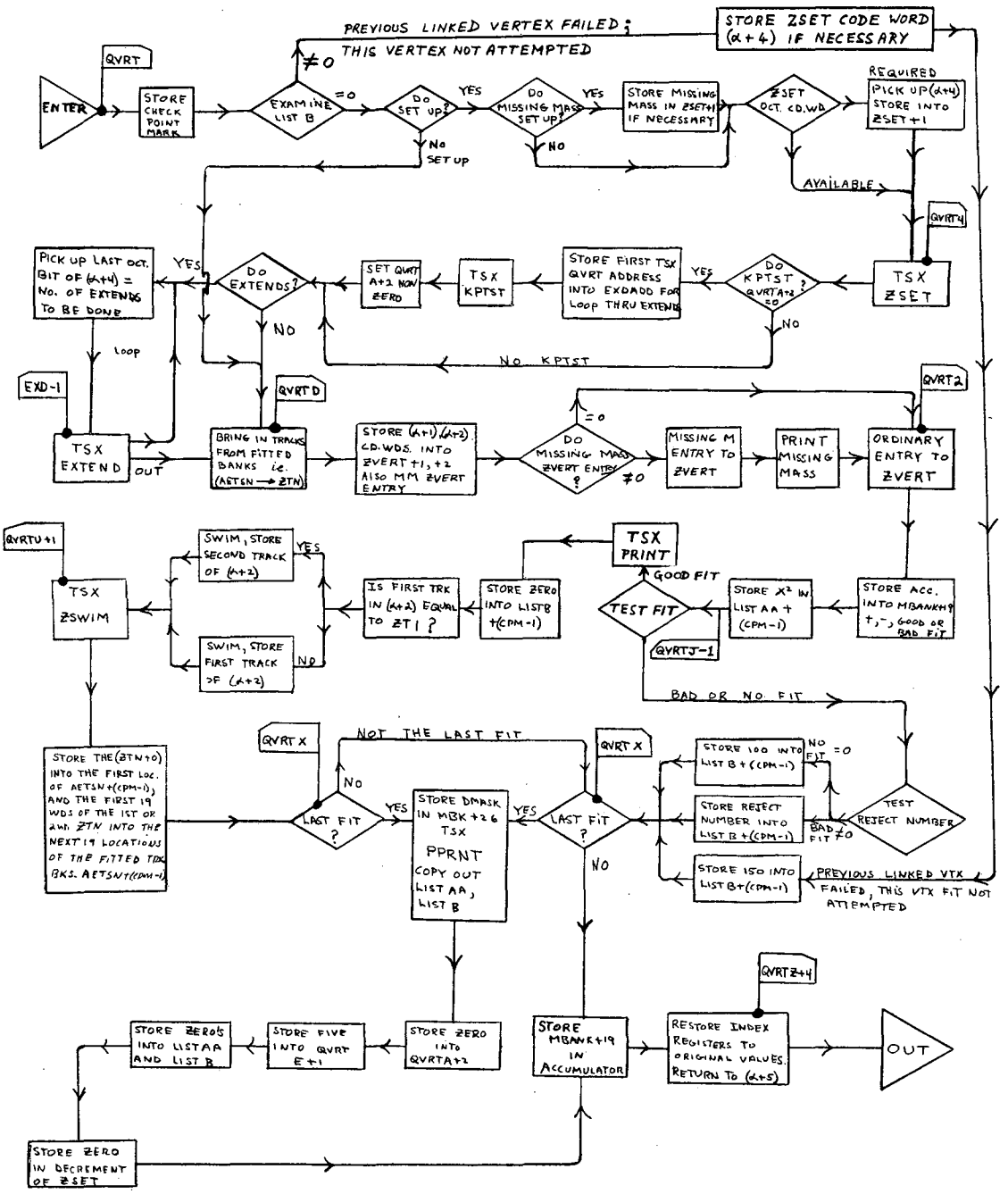
LIST's AA, B and C. LISTAA and LISTB are each 70 words long and LISTC is 10 words long. All three are printed by the Fortran Print C program in one block which has 15 rows and 10 columns and is called Report Type 6.

LISTAA occupies the first 7 rows of this block and the nth word (counting along rows) contains either the χ^2 value or the missing track marker 333.00 for CPMn or else it may be zero.

LISTB starts at the first word of row 8, the nth word belonging to the CPMn as for LISTAA. The word is zero for a good fit, the reject number if the fit was rejected, 100 for a bad fit, 150 if no fit was attempted owing to failure of the previous linked vertex, or 222.00 if the track which connects this vertex to a subsequent one has LSZ=3.

The last ten-word row of the block comprises LISTC. For each coplanarity test LISTC contains a pair of words: the triple scalar product $T = (\vec{a} \times \vec{b}) \cdot \vec{c}$ and its error δT which are defined in this manual under COPLAN.

FLOW DIAGRAM FOR QVRT 2.1

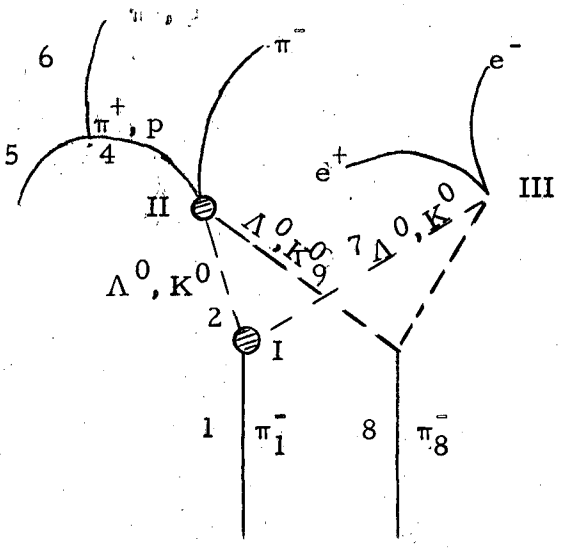


MUB-672

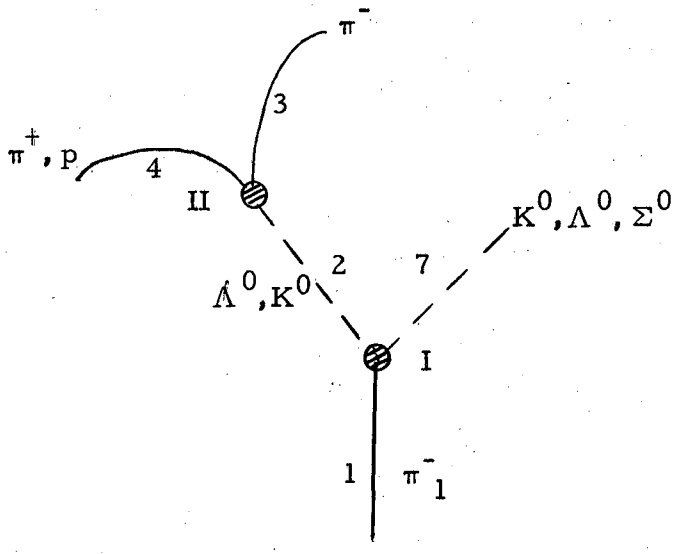
Part B (Examples)

As an example of the use of QVRT, we have written two event-type programs to be included here--event-type 30 (a zero-prong single-vee type) and event-type 40 (a zero-prong, double-vee type). The new type-30 and -40 programs have been reduced in size by a factor of the order 2 to 3 in comparison to the old 30 and 40 programs. Programming event types with QVRT amounts for the most part to filling in the octal code bits of its calling sequence.

Event-type-30 reactions



Sketch A



Sketch B

We shall consider event-type 30 as drawn in sketch B. This event type includes the following reactions:

$$\pi^- + p \rightarrow \Lambda^0 + K^0$$

$$\Lambda^0 \rightarrow P + \pi^- \quad \text{or} \quad \Lambda^0 \rightarrow N^+ + \pi^0$$

$$K^0 \rightarrow \pi^+ + \pi^- \quad \text{or} \quad K^0 \rightarrow \pi^0 + \pi^0$$

and

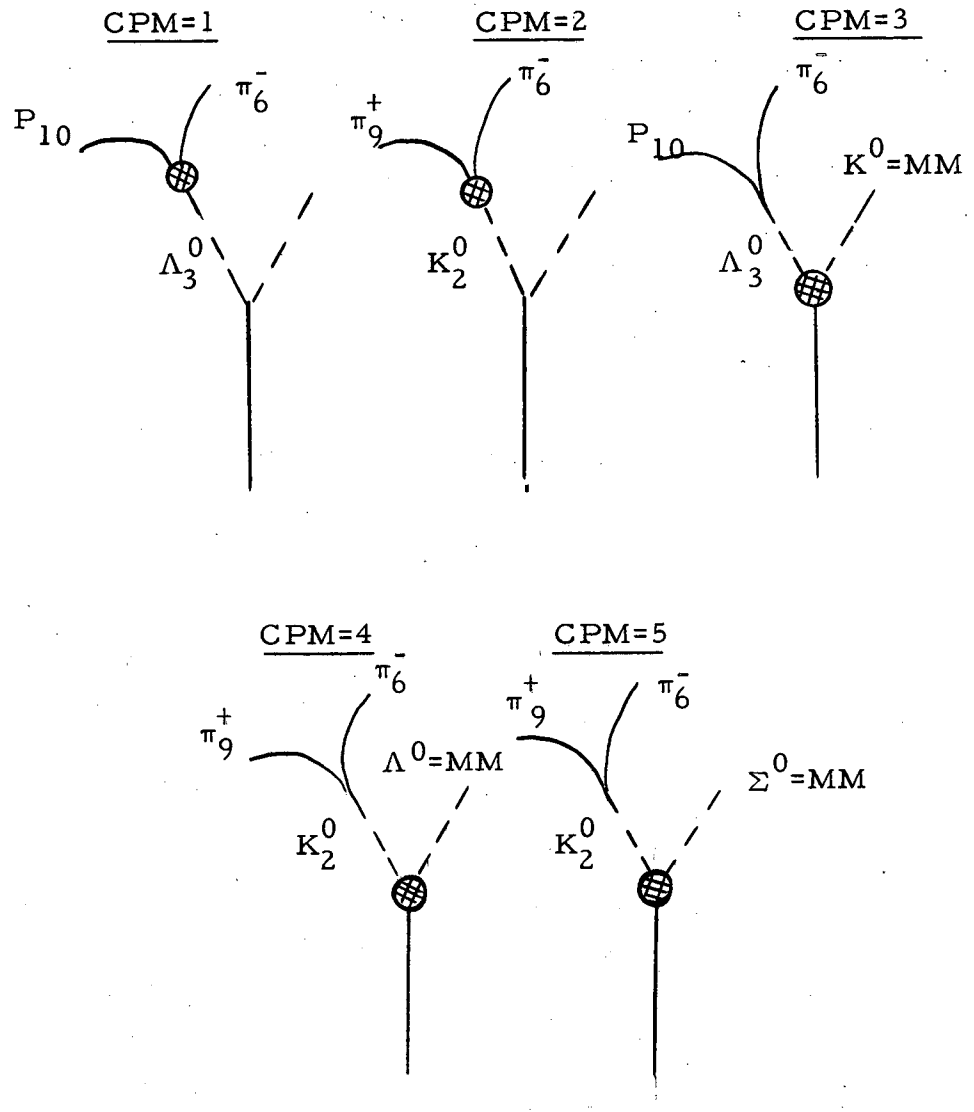
$$\pi^- + \rightarrow \Sigma^0 + K^0$$

$$\Sigma^0 \rightarrow P\Lambda^0 + \gamma$$

$$\Lambda^0 \rightarrow P + \pi^- \qquad K^0 \rightarrow \pi^0 + \pi^0$$

$$K^0 \rightarrow \pi^+ + \pi^- \qquad \Lambda^0 \rightarrow N + \pi^0.$$

Event type-30 reaction sketches according to Check-Point-Mark vertex fits

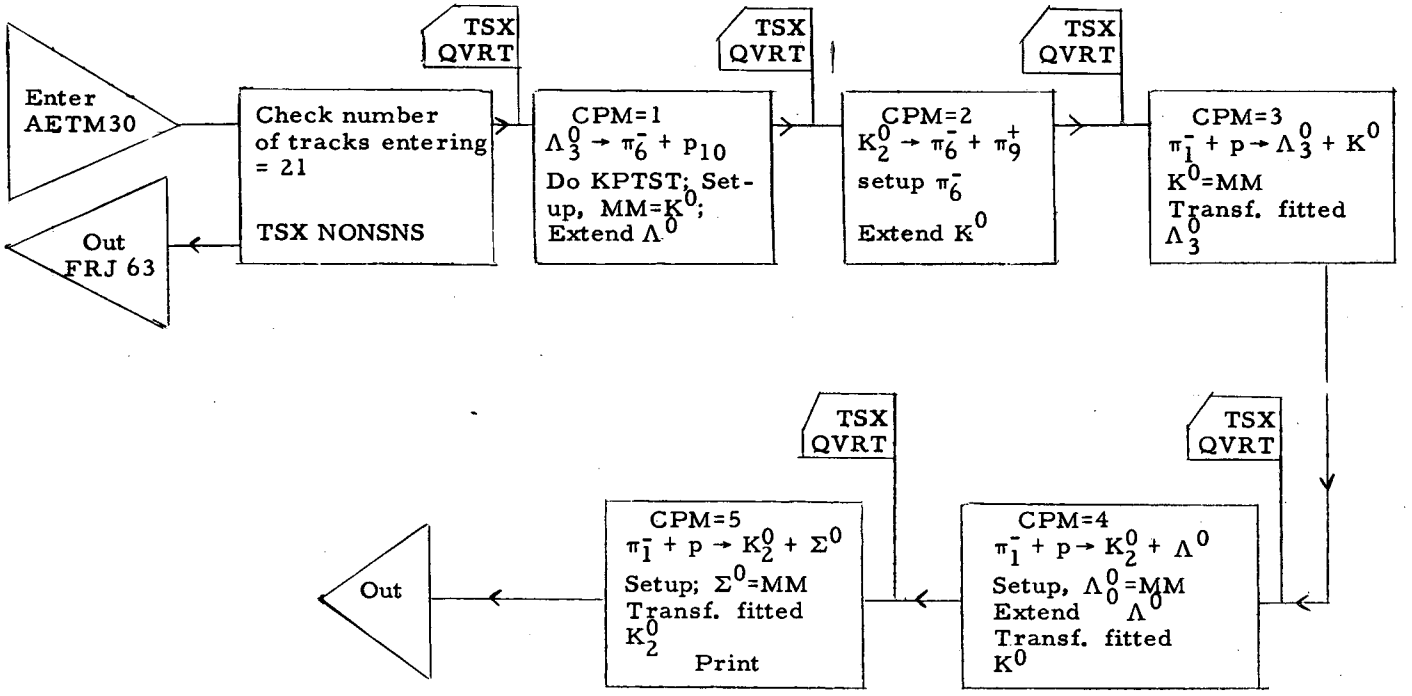


AETM30 track-bank allocations (ZTN BKS in Kick)

Physical Track No.	1	2		3			4				5		6	
ZTN	1	2	3	4	5	6	7	8	9	10	11	12	13	14
TRACK	π^-	K^0	Λ^0	e^-	μ^-	π^-	e^+	μ^+	π^+	P	π^+	P	π^+	P

Physical Track No.	7		8	9		10		20		
ZTN	15	16	17	18	19	20	21	22		31
TRACK	K^0	Λ^0	π^-	K^0	Λ^0	K^0	Λ^0	γ	← Not from Pang	MM

Flow diagram for event-type 30 using QVRT 2.1



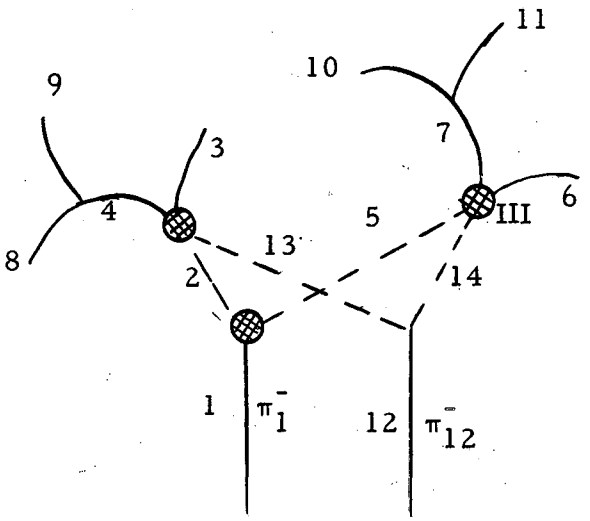
Remarks on associated event-type 30 (AETM30) program. The program first checks via a closed subroutine (NONSNS) to see if the right number of tracks has been read in by Pang. The vertices are then fitted according to their corresponding check-point marks (CPM) (see sketches), where the subscript numbers refer to their ZTn number. To find the physical track number see sketch A, page 7. For interpretation of the QVRT code words, refer to Part A of this memorandum in the QVRT calling sequence.

A note of caution. On the last fit, the (a + 3) code-word octal bit 12 (extreme right-hand bit) must be a one to obtain all the chi squares and reject numbers printed out in a table.

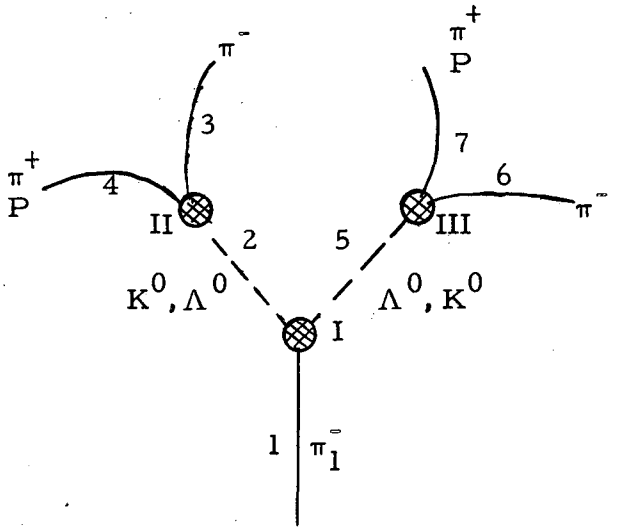
Event-type program 30 for QVRT 2.1

```
AETM30   CLA   AET 304                 DEC 21
         TSX   NONSNS, 4
         TRA   *+3
         PZE   -ZT1, 0, -ZT2           K^0 EXTEND word
         PZE   -ZT1, 0, -ZT3           Λ^0 EXTEND word
         TSX   QVRT, 4
         OCT   43 00 00 03 01 00       MM1=3=K^0; MM1=γ → ZT30
         OCT   03 14 50 00 00 00       Λ_3^0 → π_6^- + P_10
CPM1     OCT   01 00 00 00 00 20       Set up;
         OCT   03 76 00 00 00 22       Set-up word; two EXTEND's
         TSX   QVRT, 4
         OCT   43 00 00 03 01 00
         OCT   02 14 44 00 00 00       K_2^0 → π_6^- + π_9^+
CPM2     OCT   02 00 00 00 00 10       Set up;
         OCT   00 00 00 00 00 02       Two EXTEND's
         TSX   QVRT, 4
         OCT   53 00 00 00 00 00
         OCT   01 07 74 00 00 00       π_1^- + p → Λ_3^0 + K^0 = MM3
CPM3     OCT   03 01 00 00 00 00       Transfer Λ^0
         OCT   00 00 00 00 00 00
         TSX   QVRT, 4
         OCT   53 00 00 05 01 00       MM5 = Λ^0, γ = MM1 → ZT30
         OCT   01 05 74 00 00 00       π_1^- + p → K_2^0 + Λ^0 = MM5
CPM4     OCT   04 02 00 00 00 10       Transfer K^0; Set up
         OCT   00 00 00 00 00 02       Two EXTENDS
         TSX   QVRT, 4
         OCT   53 00 00 06 01 00       MM6 = Σ^0, γ = MM1 → ZT30
         OCT   01 05 74 00 00 00       π_1^- + p → K_2^0 + Σ^0 = MM6
CPM5     OCT   05 02 00 00 00 11       Transfer K^0; Set up; Print
         OCT   00 00 00 00 00 00
         TRA   ZEND
```

Event-type-40 reactions



Sketch A



Sketch B

In event-type 40 we shall consider only the reaction indicated in sketch B. The leptonic decay modes of the Λ^0, K^0 are treated in our event type called (AETLEP) and shall be discussed in a later memorandum.

The sketch-B event includes the following reactions:

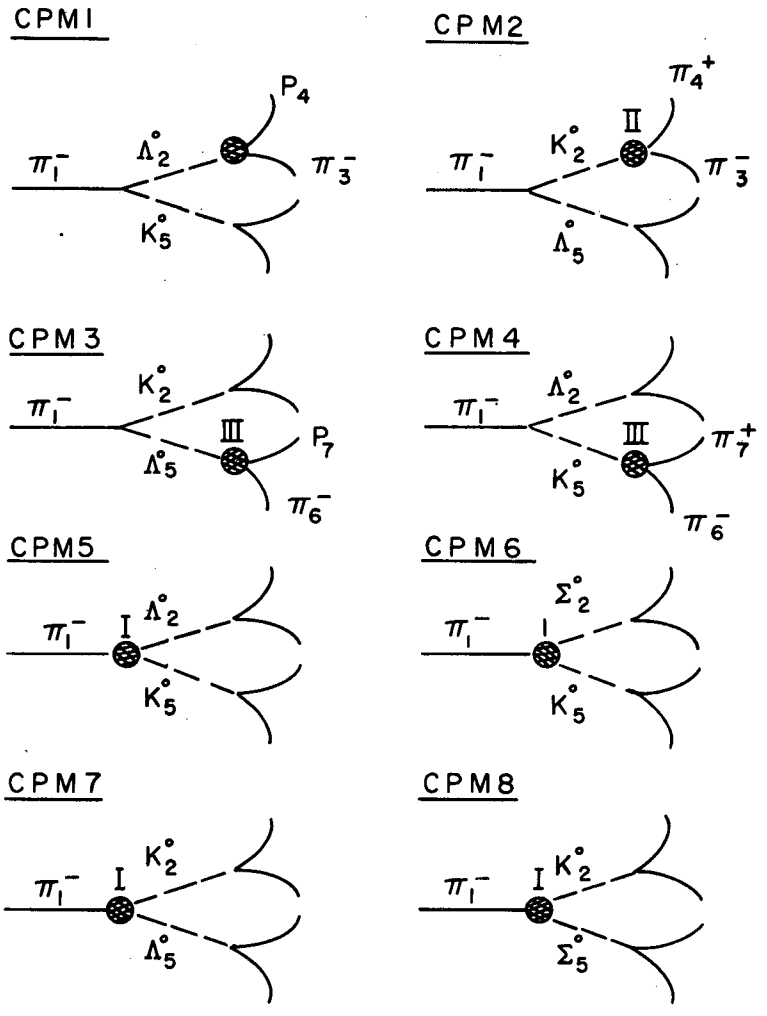
$$\pi_1^- + P \rightarrow \Lambda_2^0 + K_5^0 \qquad \pi_1^- + P \rightarrow K_2^0 + \Lambda_5^0$$

$$\Lambda_2^0 \rightarrow \pi_3^- + P_9 \qquad K_2^0 \rightarrow \pi_3^- + \pi_8^+$$

$$\Lambda_5^0 \rightarrow \pi_6^- + P_7 \qquad K_5^0 \rightarrow \pi_6^- + \pi_7^+$$

$$(\pi_1^- + P \rightarrow \Sigma^0 + K_5^0 \text{ or } K_2^0) \qquad (\pi^- + p \rightarrow K_2^0 + \Lambda_5^0),$$

where subscripts refer to the physical track number as in sketch B.



MU-24181

Vertex fits in event-type 40 sketched according to CPM.

AETM40 track-bank allocations (ZTN banks in Kick)

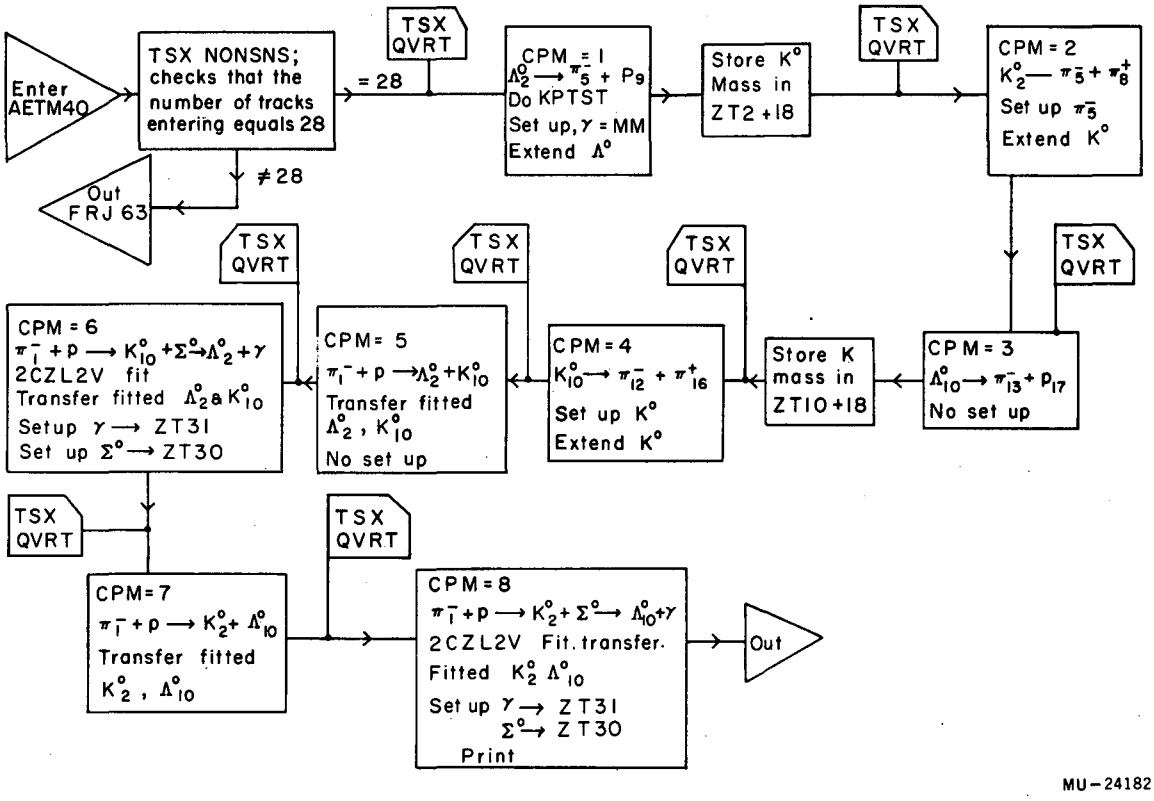
Physical Track No.	1	2	3			4				5	6			7			
ZTN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Track	π^-	Λ^0	e^-	μ^-	π^-	e^+	μ^+	π^+	P	Λ^0	e^-	μ^-	π^-	e^+	μ^+	π^+	P

↑ Store in K^0 at appropriate time

↑ Store in K^0 at appropriate time

Physical Track No.	8		9		10		11		12	13	14			
ZTN	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Track	π^+	P	π^+	P	π^+	P	π^+	P	π^-	Λ^0	Λ^0		MM	MM

Remarks on associated event-type 40 program (AETM40). The number of tracks entered by Pang is first check via a closed subroutine called NONSNS; then [refer to CPM sketches on p. (12)] eight possible vertex fits are done and labeled with respect to their CPM's. The subscripts on the tracks designate their ZTN number. Note that between fits CPM = 1, 2 and CPM = 3, 4, the K^0 mass has been stored into ZTn banks 2 and 10 respectively, because of the change in vertex hypothesis. (Pang originally read in ZT2, and ZT10 as lambda tracks). In doing both CPM6 and CPM8, a two-vertex fit (2CZL2V) is performed. In such a case, the ZVERT(a+1) code word must have a one in its twelfth octal bit position (extreme right); refer to the section on ZVERT for details.



Flow diagram for event-type 40 using QVRT 2.1.

Event-type program 40 for QVRT 2.1

```
AETM40   CLA   AET 404                  DEC 28
         TSX   NONSNS, 4
         TRA   *+3
         PZE   -ZT1, 0, -ZT2             Λ2 or K0
         PZE   -ZT1, 0, -ZT10            Λ10 or K0 10
         TSX   QVRT, 4
         OCT   43 00 00 01 01 00         Gamma = MM1, Gamma → ZT30
         OCT   02 12 44 00 00 00         Λ2 → π-5 + P9
CPM1     OCT   01 00 00 00 00 20         Set up
         OCT   07 75 77 77 60 22         Set up word; two extends
         CLA   MASTBL-3                  K0 mass
         STO   ZT2 + 18
         TSX   QVRT, 4
         OCT   43,00 00 01 01 00         Gamma = MM1, MM2
         OCT   02 12 40 00 00 00         K0 2 → π-5 + π+8
CPM2     OCT   02 00 00 00 00 10         Set up
         OCT   00 00 00 00 00 02         Two extends
         TSX   QVRT, 4
         OCT   43 00 00 00 00 00
         OCT   12 33 04 00 00 00         Λ10 → π-13 + P17
CPM3     OCT   03 00 00 00 00 00
         OCT   00 00 00 00 00 00
         CLA   MASTBL-3                  K0 Mass
         STO   ZT10 + 18
         TSX   QVRT, 4
         OCT   43 00 00 01 01 00         Gamma = MM1, MM2
         OCT   12 33 00 00 00 00         K0 10 → π-12 + π+16
CPM4     OCT   04 00 00 00 00 10         Set up
         OCT   00 00 00 00 00 02         Two extends
         TSX   QVRT, 4
         OCT   53 00 00 00 00 00
         OCT   01 04 50 00 00 00         π-1 + P → Λ0 2 + K0 10
CPM5     OCT   05 01 04 00 00 00         Transfer Λ0 2  K0 10
         OCT   00 00 00 00 00 00
```

Event-type program 40 for QVRT 2.1 (continued)

```
        TSX   QVRT,4
        OCT   55 00 00 01 06 01      Gamma = MM1, Σ0 = MM6 → ZT30
        OCT   01 24 13 77 40 00      π1- + p → K10 0 + Σ0 + γ
CPM6    OCT   06 01 04 00 00 10      Transfer  Λ2 0; K10 0; set up
        OCT   00 00 00 00 00 00
        TSX   QVRT,4
        OCT   53 00 00 00 00 00      Gamma = MM1, Σ0 = MM6 → ZT30
        OCT   01 04 50 00 00 00      π1- + p → K2 0 + Λ10 0
CPM7    OCT   07 02 03 00 00 10      Transfer K2 0; Λ10 0; set up
        TSX   QVRT,4
        OCT   55 00 00 01 06 01      Gamma = MM1, Σ0 = MM2 → ZT30
CPM8    OCT   01 04 53 77 40 00      π1-, K2 0, Σ0, Λ10 0, γ
        OCT   10 02 03 00 00 11      Transfer K2 0, Λ10 0; set up
        OCT   00 00 00 00 00 00      Print
        TRA   ZEND
```

UCRL-9099
Aux. Processing-COPLAN-1
December 1960
G. Kalbfleisch

AUXILIARY PROCESSING ROUTINES

COPLAN: A Closed Subroutine to Test Coplanarity

Introduction

COPLAN evaluates a triple product for the evaluation of the coplanarity of three tracks at a vertex. This triple product is

$$T = \hat{a} \cdot (\hat{b} \times \hat{c}) ,$$

where $\hat{a}$, $\hat{b}$, and $\hat{c}$ are unit vectors along the three tracks. The error δT is also evaluated. Special returns are made for CRN = 3.

Formulae

We have

$$T = \hat{a} \cdot (\hat{b} \times \hat{c}) = a_x b_y c_z - a_x b_z c_y + a_y b_z c_x$$

$$- a_y b_x c_z + a_z b_x c_y - a_z b_y c_x ,$$

where

$$m_x = \frac{\cos \phi_m}{\sqrt{1+s_m^2}} , \qquad m_y = \frac{\sin \phi_m}{\sqrt{1+s_m^2}} , \quad \text{and} \quad m_z = \frac{s_m}{\sqrt{1+s_m^2}} ,$$

with m = a, b, or c. To evaluate δT, we use the six derivatives

$$\frac{\partial T}{\partial \phi} , \quad \frac{\partial T}{\partial s_1} , \quad \frac{\partial T}{\partial \phi_2} , \quad \frac{\partial T}{\partial s_2} , \quad \frac{\partial T}{\partial \phi_3} , \quad \text{and} \quad \frac{\partial T}{\partial s_3} .$$

Thus we have (tracks are now 1, 2, and 3)

$$T = \frac{1}{\sqrt{1+s_1^2}\ \sqrt{1+s_2^2}\ \sqrt{1+s_3^2}} \left\{ \begin{array}{l} \cos\phi_1 \sin\phi_2 s_3 - \cos\phi_1 s_2 \sin\phi_3 \\ + \sin\phi_1 s_2 \cos\phi_3 - \sin\phi_1 \cos\phi_2 s_3 \\ + s_1 \cos\phi_2 \sin\phi_3 - s_1 \sin\phi_2 \cos\phi_3 \end{array} \right\} .$$

Letting $AA = \sqrt{1+s_1^2}\ \sqrt{1+s_2^2}\ \sqrt{1+s_3^2}$, we can also write

$$\frac{\partial T}{\partial \phi_1} = \frac{1}{AA}\left\{-\sin\phi_1(\sin\phi_2\, s_3 - s_2 \sin\phi_3) + \cos\phi_1(s_2\cos\phi_3 - \cos\phi_2\, s_3)\right\},$$

$$\frac{\partial T}{\partial \phi_2} = \frac{1}{AA}\left\{\cos\phi_2(\cos\phi_1\, s_3 - s_1\cos\phi_3) - \sin\phi_2(s_1\sin\phi_3 - \sin\phi_1\, s_3)\right\},$$

$$\frac{\partial T}{\partial \phi_3} = \frac{1}{AA}\left\{\cos\phi_3\,(s_1\cos\phi_2 - \cos\phi_1\, s_2) - \sin\phi_3(\sin\phi_1\, s_2 - s_1\sin\phi_2)\right\},$$

and

$$\frac{\partial T}{\partial s_i} = \frac{B_i}{AA} - T\,\frac{s_i}{1+s_i^2},$$

where B_i is defined by

$$T \equiv \left(\sum_j s_j\, B_j\right)/AA.$$

For example; we have

$$\frac{\partial T}{\partial s_1} = \frac{1}{AA(1+s_1^2)}\left\{\sin(\phi_3-\phi_2) - s_1 s_2 \sin(\phi_1-\phi_3) - s_1 s_3 \sin(\phi_2-\phi_1)\right\}.$$

Thus we may write

$$\delta T = \sqrt{B\, M\, \tilde{B}},$$

where

$$B = \left(\frac{\partial T}{\partial \phi_1}, \frac{\partial T}{\partial s_1}, \frac{\partial T}{\partial \phi_2}, \frac{\partial T}{\partial s_2}, \frac{\partial T}{\partial \phi_3}, \frac{\partial T}{\partial s_3}\right)$$

and

$$M = \begin{pmatrix} X_1 & 0 & 0 \\ 0 & X_2 & 0 \\ 0 & 0 & X_3 \end{pmatrix} \quad \text{(a 6-by-6 matrix)}$$

$$X_i = \begin{pmatrix} \overline{\delta\phi_i\, \delta\phi_i} & \overline{\delta\phi_i\, \delta s_i} \\ \overline{\delta s_i\, \delta\phi_i} & \overline{\delta s_i\, \delta s_i} \end{pmatrix}$$

and

$$0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Use

COPLAN works on set-up part of track banks. Therefore if necessary first TSX ZSET, or do after a fit.

Calling Sequence

	Calling sequence
a	TSX COPLAN, 4
a + 1	HTR 0, 0, -ZTa
a + 2	HTR 0, 0, -ZTb
a + 3	HTR 0, 0, -ZTc
a + 4	RETURN

a, b, and c are the three track bank numbers to be used.

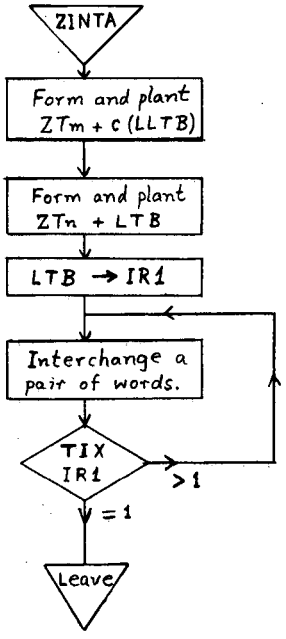
Upon exit from COPLAN the AC is positive if the tracks are coplanar to within three standard deviations, and negative if not. That is, the AC contains - $|T/\delta T| + 3.0$. T and δT are left in COMMON and COMMON + 1, respectively. For CRN = 3, AC is made positive, T is set equal to zero, and δT equal to 9.0.

ZINT - Track Bank Interchange

Calling Sequence:

```
TSX  ZINT, 4
HTR  ZTn, 0, ZTm
```

where track banks m and n are to be interchanged.



This routine is also defined as ZINTA by means of the pseudo-operation SYN:

```
ZINT  SYN  ZINTA
```

MU-23673

BMAVG

If the initial bit of the ZSET octal code word is zero, the beam-averaging routine forms a weighted average of the Pang curvature with the curvature derived from PBEAM. In Kick 5 this routine starts at ZSET YY-4, and PBEAM must be given at the chamber origin (y = 0). PBEAM is converted to a curvature at y = 0, and the GUTS routine REROUT is used to swim this curvature back to the middle of the beam track. The resulting curvature, k^{Beam}, is then substituted into the formula

$$k^{B.A.} = \frac{\dfrac{k^{Pang}}{(\delta k^{Pang})^2} + \dfrac{k^{Beam}}{(\delta k^{Beam})^2}}{\left(\dfrac{1}{\delta k^{Pang}}\right)^2 + \left(\dfrac{1}{\delta k^{Beam}}\right)^2}$$

and $k^{B.A.}$ is placed in ZT1+2.

BANKS

Table of Contents of Storage Banks and Glossary Section

MBANK map: "Main Bank" contains Pang master card plus some Kick quantities pertaining to the whole event, including some Handy Words.

ZTn map ZTn is the first word of the n-th track bank ($1 \leqslant n \leqslant 31$)

ZPUBn Pick-Up Bank n ($1 \leqslant n \leqslant 9$). The PUB's are Kick's only way of passing information from one event to another later (connected) event.

ZTEMP "Temporary Event Bank" is an extension of Main Bank which differs mainly in that it is not put on output Tape 4.

PARAK "Parameters of Kick" as opposed to constants, are written on Tape 4 whenever the program is loaded or "Parch Cards" are passed on to Kick via Pang. Parameters may be identified by the Parak ID "PARID" (in TEMPAR and in MBANK) which appears on every print. See Special "Parameters" comment for distinctions between this bank and the ones that follow.

MASTBL "Mass Bank" consists of 30 words of PARAK. See comments under "Parameters."

TEMPAR* (See "Parameter") Contains values assembled along with the Event-Type deck (See Flow-1-)

EVPAR* "Event Parameters" which differ from experiment to experiment; also assembled along with event-type deck.

CONS "Constants" (Integers): Part of GUTS

FCONS "Floating Cons" (Integers): Part of GUTS

} Berge asks that no additions be made to these.

DCONS (and some definitions following that) are used by Kick, DCONS are integer in decrement.

MAPS for most of the above appear in this section -- for those that do not, consult the assembly.

GLOSSARIES for MBANK and ZTn are at the back of this section, (paginated as MBG- and ZTG-).

---

*In event-type deck.

PANGMC

(This page should also be considered as an appendix to Alvarez-Group Memorandum 115, page 18.)

Characters under the column heading "FORMAT" are BCD; b stands for Blank, and the "variable" characters (used for actual data) are underlined. "IntAdd" denotes Integer in Address.

This is a Bank in which the master-card information is saved in the format in which it comes from the Pang binary output tape. Next the formats of some of the words in MBANK+2 through MBANK+11 are changed so that there will be no 72"-vs.-15" format ambiguities, etc. Of course, the final format is on page MBANK-2.

PANGMC+		72" Format	15" Format
0	Date Franckensteined	bYbM0b	bDAbM0
1	Date Franckensteined	DAb704	bYb704
2	Date Panged	YbM0bD	DAbM0b
3	Date Panged	AbPANG	b60bPA
4	Measurement No. (Normally zero; set 1 on first remeasurement. When Pang omits an event it changes 0P to 0M).	. . bMb0P	. bM6T0
5	Franckenstein Operator's number	IntAdd	IntAdd
6	Pickup Command (PUC) and Experiment No. (Tens and Units)	ObPTUE	ObPUbE
7	Event Type No.	IntAdd	IntAdd
8	Beam Track No.	bNO.TR	bNO. bT
9	Serial No. of Event (Roll and Frame)	RRRFFF	RRRFFF

Main Bank for Kick 5

Notes

1. The list below is MBANK format for Kick 5 just before entry to VPRINT. This is the form it is assumed to have by all Kick Processing routines. However at entry to PPRINT it is convenient to have the serial number and several ID numbers and dates in words that will later be overwritten for and by the processing routines. These numbers are listed on a separate page, PANGMC. MBank is cleared in ZKICK2, and there is an auxilliary clear if the -S return is made via ZREAD6.
2. All integers are in the decrement "Int D" (Fortran convention).
3. The characters in the format column are BCD (B stands for blank). The "variable" characters (used for data) are underlined.
4. Names written all in capitals have SAP definitions; others do not. For example, GPM is defined, HwdD is not.
5. All 75 words of MBANK are written on binary tape by both PPRINT and VPRINT, but EXAMIN will not necessarily print all of it.
6. For further comments see the MBANK glossary.

MBANK for Kick 5 Only

MBANK+	(constitutes first record of binary output)	
0(1)*	Run I. D. [Copied from DATE (in TEMPAR)]	Full Wd Integer
1(2)	STM--Now used for KKT FailingTrack No. Contains 9999 if event not found in ZTABLE	Int D
2 & 3	Date Franckensteined	YR/MO/DAbbbb
4 & 5	Date Panged	YR/MO/DAbbbb
6(7)	Measurement No. (1 on first remeasurement)	Int D
7(8)	Franckenstein Operator Number	Int D
8(9)	Experiment number	Int D
9(10)	Event type	Int D
10 & 11	Serial number (R=Roll, F=Frame, T=Beam Track)	RRRbbbFFFbTT
12(13)	PUC "Pick-Up Command" The character P from PANGMC+6	Int D
13	Free	
14(15)	GPM "Good-Print Mark" Counts number of successful fits	Int D
15(16)	χ^2; 500 signifies HRJ; but for missing-mass "fit", "χ^2" is χ^2(Un3P)	Fl.
16(17)	DAMN (Sign bit determines sign used in OC ambiguity)	Sign Bit
17(18)	Report counter (Counts reports on a given beam track)	Int D
18(19)-24(25)	HWdD through J; Floating handy words	Fl.
25(26)	"Fuzzy KKT's". Reserved for $\Sigma_{fuzzy}(KKT)^2$†	Fl.
26(27)	LPM-Nonzero for last Print of an Event	
27(28)	CPM - "Check-Point Mark", also called Fit I. D.	Int D
28(29)	Parameter ID	Int D
29(30)	WALL. Max. proj. range of zero-length charged track (cm)	Fl.
30(31)	MMASS - missing-mass μ. Negative means $E_{out}-E_{in} < 0$	Fl.
31(32)	DMMASS - $\delta\mu$ Negative means $\mu^2 < 0$	Fl.
32(33)	ZCCTR1 = Z(Control of DAMN sign) Counter	Int D
33(34)	Ord	Int A
34(35)	Print type 0=Param, 1=P, 2=V, 3=MM, 4=HRJ, 5=FRJ, 77777=end of tape	Int D
35(36)	Length of second record of output	Int D
36(37)	Tracks in second record of output	Int D
37(38)	Reject type	Int D
38-40	ZVERT calling sequence	Coded
41(42)	Free	
42(43)	Free	
43(44)	Free	
44(45)	Free	
45(46)-59(60)	HWd1 through 15; floating handy words	Fl.
60(61)-74(75)	IHWd1 through 15; more integer handy words	Int D
SNYDER+	(Starts second binary record of output. Has additional GUTS fit information. Listed on assembly between GUTS and Kick. Described in Glossary of both write-ups and in missing-mass section.) Particularly useful locations are:	
0(1)	KLOCP (P is number of particles fitted by GUTS)	Int D
1(2)	KLOCL ($\bar{L}$ is number of constraints used by GUTS)	Int D
2(3)	KLOCI ($\bar{I}$ is number of variables)	Int D
3(4)	KLOCLL (Flag for special classes, two vertex or WALL)	Int D
4(5)		Int D
5(6)	GCONTR (Step Counter)	Int D
6-12	CUT1 through CUT (Cutstep Counters - see RJCT2 in Memo. 86)	Int D
46(47)	MPRIME [true value of χ^2, never touched by Reject routines or χ^2(Un4) for MM]	Fl.
49(50)	KDAMN (actual sign of DAMN as used by GUTS).	Int D

*Parentheses indicate MBANK word numbers (FORTRAN Convention) on output tape.

†Not yet in program.

MBANK Glossary

MBANK+

0 Kick Identification No. is currently used for data of Kick run. Write year, month, day, with two digits for each, e.g. 31 December, 1960 is the decimal integer 601231. For further details, see "Identification Numbers" section.

1 STM (Secondary Type Mark) is no longer used for this purpose in Kick 5, but the name remains. Carries KKT information, or reasons why event was not processed:

Print Type	MBANK + 1	Comment
(4) Reject 63	Some track number	Track failed KKT by > "4"
(2) Vertex Print	Some track number	Failed KKT by < "4"
(1) Pang Print	9999	Event not listed in ZTABLE

See section on K Tests for more details on KKT = "K-Kick Test". Kick 4 users should instead refer to the small Appendix following this section, since the significance of STM has been completely changed in Kick 5.

2 to 7 Dates measured and Panged, measurement No., and Franckenstein operator No. are not used logically in Kick but are passed on for later BCD printouts by EXAMIN.

12 PUC (Pickup Command) is a digit set up during measurement of the second or third of a series of connected events. At the end of ZSET, after all the Pang tracks have been set up, only the LOB (lowest-order bit of this word) is tested. If it is unity, the entire contents of the Pickup bank are transferred to the appropriate track banks. Some experiments have used the higher-order bits of PUC to store other information.

All the following words except Report Counter (+17), Parameter ID (+28), and Ord, are cleared when ZREAD6 is re-entered from ZEND as a result of a (-S) track being changed to leaving.

MBANK+

13 Free in Assembly 5. [In Assembly 4:DHOLD. The Distance (a positive number in centimeters) above the vertex that the incoming track curvature can be specified for GUTS. DHOLD is often zero, but will become less often so. DHOLD is really a property of Track 1, but we did not want to make ZT1 different from that of the other tracks, so we put it in MBANK.]

14 GPM(Good Print Mard) is an integer in the decrement which is advanced by one each time that a good print is made. ZEND regards a zero GPM as a serious symptom and tries the -S test. For details of how GPM is set, see first paragraph of notes on ZEND

16 DAMN: See Zero-C Special-Case Supplement to Memo 86, p. 25. Related to sign ambiguity in Zero-C fitting problem.

17 Report Counter, is increased by one for each report and of course cleared when a new "beam event" is read in, i.e., it continues to add up for each report, even connected events, until the beam track No. or serial No. changes.

18 to 24 Handy words can be filled in by event types. Other floating-point handy words are MBANK+60-74. Integer handywords are MBANK+45-59.

25 "Fuzzy" KKT's. Reserved for $\Sigma_{fuzzy}(KKT)^2$.

27 CPM (Check Point Mark) is set at various stages by the event type, just before entering ZVERT, to identify which ZVERT entry has led to the current data. It should really be called Fit I. D.

28 Parameter ID No. Contains the contents of PARID (the parameter ID number) which is advanced automatically as each Parch Card (Parameter Change Card) word-pair is read. PARID contains, for example, a date and perhaps a serial number. Format depends on the individual user.

29 WALL. (See Memo 86, p. V-8) and Wall section.

30 MMASS (Missing Mass) See section "Missing Mass" MBANK+30 and 31 are filled by ZVERT only if the missing-mass calculation has been performed. If MM is called for but a variable is missing, 9999.99 is stored here. A negative sign on the missing mass μ means that the energy balance (E_{out}-E_{in}) was negative; a negative $\delta\mu$ means that μ^2 was negative.

31 DMMASS: δ (MMASS), where δ is the "uncertainty"). It carries the sign of $\mu^2 = E^2 - p^2$

32 ZCCTR1: "Z (Control of Damn sign) CounTeR," needed in O-C case. See GUTS Memorandum 86, p. 34. If ZCCTR1 is nonzero, GUTS is permitted to change sign of DAMN in search for positive momentum.

33 Ord: Ordinal number of event being processed. It is raised on reading a new file.

34 to 37 Output identification words. See section on "Output"

38 to 40 Words a to a + 2 of ZVERT calling sequence. May be needed by EXAMIN.

45 to 59 Integer handywords

60 to 74 More floating handywords.

Appendix on STM in Kick 4

MBANK+1 STM: "Secondary-Type Mark" is the essential word needed to understand a printout. If no reject has occurred on a fit, STM shows which event-type subroutine has just called for a "report" (i.e. written the results of a fit on Tape 4.) Thus consider Type 32 (Sigma production "KSF"; i.e. we want to try both the KS hypothesis, SUB122, that the K stopped, and the KF hypothesis, SUB22, that the K is in flight.) There will be a number of reports (each can have a separate checkpoint Mark, see MBANK+27) with STM = 0000000122 and a number with STM = 0000000022. If an event is unambiguous, STM may be zero. The convention $n \times 10^4$ added to STM by REPORT is used to indicate the results of chi-squared testing. For example 0×10^4 indicates a good fit, 1, "poor", and 2, "uncertified" (i.e. results of a 0-constraint fit or no fit at all).

STM = 2999 shows that the Missing-Mass calculation has just been done.

STM is also used to indicate the two types of rejects, in which case the two lowest-order digits, instead of indicating the secondary type (sometime soon we hope to also save the secondary type) are used to indicate the type of reject (see "Rejects"). Specifically:

10^6 + n indicates a hypothesis-reject type-n via subroutine HRJ

10^8 + n indicates file-reject type-n via subroutine FRJ

6 or 7, $\times 10^9$ indicates a "forced" print.

A print can be "forced" by one of two very short operator-controlled programs located at the very end of the core:

TRA 77760_8 will "print and proceed" and set STM = 6000020000;

77770_8 will "print and end" and set STM = 7000020000.

If 9000 is added to STM, it means that the event type was not found in the ZTABLE, so was given the standard set-up, report, and end-test treatment described in ZREAD. Notice that this standard treatment sets up ZT1 at the beginning, and all others at the end.

Kick part, i.e. processing part
Set up by ZSET: overwritten by ZVERT and ZSWIM

	ZTn+	Name	Units	Format
	0(8)*	$\phi(0 < \phi < 2\pi)$ or flags; Wall=1 Poison = 7	radians	Fl. IntA IntA
	1(9)*	$s = \tan\lambda$		Fl.
	2(10)*	k	$(Mev/c)^{-1}$	Fl.
	3(3)*	Status (+/-) =(fit/unfit)		Int D
	4(4)*	Crn =0 =1 =3		Int D
Error - Matrix	5	$\overline{\delta\phi\delta\phi}$ $\overline{\delta\phi\delta\phi}$		Fl.
	6	$\overline{\delta\phi\delta s}$ $\overline{\delta\phi\delta s}$		Fl.
	7	$\overline{\delta\phi\delta k}$ $\overline{\delta s\delta\phi}$		Fl.
	8	$\overline{\delta s\delta\phi}$ $\overline{\delta s\delta s}$		Fl.
	9	$\overline{\delta s\delta s}$ GARBAGE →	GARBAGE	Fl.
	10	$\overline{\delta s\delta k}$		Fl.
	11	$\overline{\delta k\delta\phi}$		Fl.
	12	$\overline{\delta k\delta s}$		Fl.
	13	$\overline{\delta k\delta k}$ ← GARBAGE		Fl.
	14(5)*	Endwd (+/-)=angles set up at (beg/end)		Int D
Pang appendage	15(6)*	Lsz; L=1, s=2, z=3, B=4		Int D
	16(7)*	Ftn		IntD
From Pang	17(1)*	Track No.		IntD
	18(2)*	Mass (and charge)	Mev	Fl.
	19	L	cm	Fl.
	20	δL	cm	Fl.
	21	k [For 2pt {1.0}	$(Mev/c)^{-1}$	Fl.
	22	δk [Set to {10.0]	$(Mev/c)^{-1}$	Fl.
	23	δp via arc If neutral set 10.0	(Mev/c)	Fl.
	24(11)	kPTest or kKTest		Fl.
	25	$\overline{\delta\phi\delta s}$		Fl.
	26	$\overline{\delta\phi\delta k}$		Fl.

	ZTn+	Name	Units	Format
From Pang	27	|k| via arc (Neutrals set to 1.0)	Mev/c	Fl.
	28	δk via arc (Neutral set to 10.0)	Mev/c	Fl.
	29	ϕ_b(beginning azimuth) $0 < \phi < 2\pi$	radians	Fl.
	30	ϕ_e (end)	radians	Fl.
	31	δφ(beg. and end)	radians	
	32	s_b(beg. slope)		Fl.
	33	s_e (end slope)		Fl.
	34	δs(both ends)		Fl.
	35	Coulomb ratio		Fl.
	36	$\overline{\delta k_{arc}\delta s}$ (Neutral set to 0)	$(Mev/c)^{-1}$	Fl.
	37	Track code (Contains LSZ, FTN information)		Coded
	38	Points measured view P		IntD
	39	Points measured view Q		IntD
	40	p_b(For 2p set to 1.0)	Mev/c	Fl.
	41	p_e(For 2p set to 1.0) or overshoot in mm	Mev/c mm	Fl.
	42	p arc(if neutral set 1.0)	Mev/c	Fl.
	43	δp/p set to 10.0 for 2-point tracks		Fl.
	44	Charge code, +1, 0, -1		IntD
	45	X_b } beg. } Spatial coordinates	cm	Fl.
	46	Y_b } beg. } Spatial coordinates	cm	Fl.
	47	Z_b } beg. } Spatial coordinates	cm	Fl.
	48	X_e } end } Spatial coordinates	cm	Fl.
	49	Y_e } end } Spatial coordinates	cm	Fl.
	50	Z_e } end } Spatial coordinates	cm	Fl.
	51	σ_{xy} } residuals	cm	Fl.
	52	σ_z } residuals	cm	Fl.
	53	View P(integer in add) View Q(integer in dec.)		Coded
	54	Pang test		Fl.

*Numbers in parentheses are the track word members (in FORTRAN convention) in a vertex report.

UCRL-9099
Banks: ZT-1

Track-Bank Glossary

Track Banks consist of three parts, all of which are cleared by ZKICK. Track Bank 31 (ZT31) is used for a missing neutral track if present. There are also nine Pickup Banks, ZPUB1 to ZPUB9, to which the contents of a bank (to be stored for the following event) are transferred. The format of all ZTn and ZPUBn banks is the same.

Pang Part: ZTn+17 to +54. This part of the bank is changed only by the data-loading routine ZREAD except for one word, K-PANG-Test in ZTn+24. It would have been put in the Pang Appendage part if there had been room. ZREAD makes various trivial changes, such as converting degrees to radians and shifting integer data into the decrement field. ZREAD does more in the case of tracks with only two measured points (Two-Point Tracks, = 2Pt.) and neutral tracks. It plants harmless dummy values of k, δk, and p in appropriate words of the Pang part, for the convenience of the Fortran Print Routine.

Pang Appendage: ZTn+15 and +16. LSZ (Leaving-Stopping-Zero length) and FTN (Future Track Number) words are unpacked from the Pang data in ZTN+37 by ZREAD. They are accessed by Kick from the Pang Appendage part of the bank. LSZ may be changed from S to L by ZEND. If so, ZSET will later change k-Pang-Test (kPT) appropriately.

Processing Part: ZTn+0 to +14. ZSET initially sets this part up with Pang data appropriate to that end of the track specified in the ZSET calling sequence. It is frequenctly overwritten by passed-on data, fitted data, swum data, etc., thus containing the most "up-to-date" information on the track. Although "Processing Part" is the better name, this is also referred to in this UCRL as the "Kick Part".

Detailed Notes on the Ztn Map

Processing Part

ZTn+0 ϕ (track azimuth). For a description of the coordinate system, see CS1. ZTn+0 serves a triple purpose: in addition to its normal use for a nonzero length track, it is used for the Wall Flag (+1 in address). This is set up by ZSET for a charged, stable, zero length track. When ZVERT sees Wall Flag, it sets up WALL in MBANK+29 (see "Wall Flag" in ZSET comments). The third use is the Poison Flag (+7 in address). This can be planted by the event-type logic of a previous connected event. It signals, via the picked-up track, that it has resolved an ambiguity in favor of a particular interpretation of the track by causing rejection by ZVERT of the others.

1 s ($\tan \lambda$ = "slope")

2 $k = \pm \left| 1/p \cos \lambda \right|$. The sign convention is that of GUTS, i.e., the product of the sign of k (Pang) $\times$ sign of charge (from mass word ZTn+18). Since + denotes "Good" and - denotes "Bad," the curvature disagrees with the sign of charge. On completion of a fit the sign is positive. (For the point on the track at which k applies, see Endword, below.)

3 Status. The sign +/- means Fitted/Unfitted. Status is set -1 by ZSET. After a fit, ZVERT sets its sign +. If placed in Pickup Banks by ZEND, Status is doubled and set negative as if it were for an unfitted track,* so is normally -2 when a track is picked up. ZSET8, the pickup part, will then pull in this -2 along with the rest of the track into ZTn. This permits one to distinguish pickups from virgins and thus avoid complications (loops). EXTEND sets STATUS (of modified tracks only) to -3. The Beam-Averaging Routine BMAVG sets status -5. Thus after fitting a beam-averaged track will have status +5.

*It seemed to us that for the event type in hand, the track plays the role of a measurement. Thus in computing χ^2 we want to treat the uncertainties like the uncertainties from Pang data. The subroutine CHISQ will do this if Status is (-). Other logically similar cases may crop up, so it seemed best to set Status (-).

ZTn+4 Crn(Constraint Reduction Number). This is the number of unavailable variables for this track, initially the number of unmeasured variables. Zero means all three variables are present; 1 means no momentum; 3 means zero length, or missing neutral for ZT31. After a fit, ZVERT sets Crn equal to zero and supplies three variables and the fitted 3-by-3 error matrix.

5 to 13 Error Matrix. The error matrix has different rank for Crn = 0, 1, and 3. $\overline{\delta x_i \delta x_j}$ means the corresponding element of the covariance matrix, which we sometimes call the error matrix. It is always assumed that interparticle correlations are zero.

14 Endword.* The sign of endword indicates the end at which "angles" are set up. The magnitude indicates the distance from this point (in units of L/2) that k is set up. The following table will amplify this further. (See ZSWIM for fuller discussion.)

Endwd	"Angles" set up at	k set up at
+0	beginning	beginning
+1	beginning	middle
+2	beginning	end
-0	end	end
-1	end	middle
-2	end	beginning

Pang Appendage

ZTn+15 Lsz (Leaving-Stopping-Zerolength) We now have L = 1, S = 2, and Z = 3. We shall soon add B (Beam) = 4 and M(Missing) = 5. The charge-exchange branch of ZEND changes -S to L in ZTn+15 but not in ZTn+37.

16 Ftn (Future Track Number) This is the number the track will have in a future connected event. It is cleared by ZEND.

---

*Until Assembly 5, 1960, Endwd was +/- as above, but was always 1 in magnitude except for the track to which DHOLD was to apply, for which it was -0.

Pang Part (see Memorandum 115, p. 18ff, and later appendices).

17 Track Number. All track banks which contain information relevant to the event in hand have nonzero track numbers. All other banks have track number zero. ZEND writes the Ftn over the old Track Number before storing a track for pickup.

18 Mass Code. The last two bits of the mass are used by both Pang and Kick to define sign of charge. None of the masses we use are known to the 1 part in 10^8 by which the mass is changed by adding a bit in the next-to-last binary place of the floating-point number. Any track which is charged has a 1 in the last bit. The next-to-last bit is the sign bit of the charge. Thus 00 = neutral, 01 = + charge, 11 = - charge. The values agreed on for all masses are found in MASTBL.

19 L (Length). In Kick we use L to signify the measured length, as opposed to a calculated range R.

21, 22 k and δk apply to the middle of the track. The sign is that of measured curvature. For two-point tracks, Pang writes 10^4 for both, and ZREAD overwrites these with k = 1.0, δk = 10.0.

24 k-Pang Test/k-Kick Test. Consistency checks to compare the range appropriate to curvature (k) with the measured length of the track. See "k-Tests" section for the details. Tests are made before and after fitting.

27 k_{arc} (k calculated for a particle of range L and mass M. This is used for the k-tests and for stopping tracks.

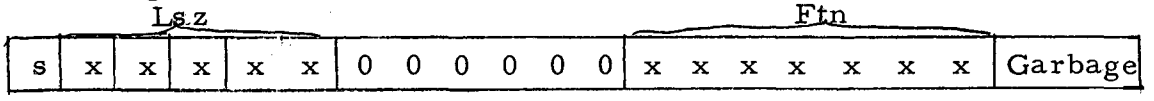
31 $\delta\phi$ is assumed to have the same value at both ends of a track.

34 δs is assumed to have the same value at both ends of a track.

35 Coulomb ratio ($\delta p_{coulomb}/\delta P_{measurement}$). To date this ratio is not used in any processing subroutine. Note that δp is calculated from externally assigned measuring errors -- as are all errors used by Kick -- not from internal consistency of the measured points with the Pang polynomial fit.

37 Track code. This is different for the 72-in. Pang and the 15-in. Pang.

72-in. Pang. The Lsz and Ftn numbers are packed by Pang into the left half of this word as $-[2^{-5}(\text{Lsz}) + 2^{-17}(\text{Ftn})]$; the bit configuration thus is:



Lsz and Ftn are unpacked in ZREAD and stored in the Pang Appendage. The data in the track code come from Franckenstein as the "pre-track word," but the Franckenstein-Pang convention for Lsz is smaller by one than the Kick convention; i.e., Pang: L.s.z. = 0, 1, 2 and Kick: L.s.z. = 1, 2, 3. ZREAD adds the 1 as it decodes.

15-in. Pang. The Lsz and Ftn numbers are again packed by Pang into the left half of this word, but the format is different; namely

$$-2^{-17} \times 10^{4}\ \underbrace{(\text{Lsz})}_{L=1,\ s=2,\ z=3.} + 10^{3}\ (\text{Ftn } \underline{\text{units}}) + 10^{2}\ (\text{Ftn } \underline{\text{tens}}).$$

Lsz and Ftn are unpacked in ZREAD and stored in the Pang Appendage. The data in the track code come from Franckenstein, as the "after-track word." It is planned eventually to abolish the 15-in. format and put everything on Pang output into 72-in. format.

40 P_b. This is the momentum from curvature, at the beginning; the sign is that of the measured curvature.

41 If Pang fits the track as "overstopped" (i.e. for $R(k_m) < L/2$), P_e is replaced by the "overshoot" R-L/2 in millimeters.

44 λ_b (in degrees) is read in from Pang, and is overwritten by the charge-code.

45 through 50 See "Coordinate System Comment CS1."

51 and 52 Root-mean-square deviation of measured track points from fitted track in horizontal and vertical planes.

54 Pang test on $[p_{\text{view P}}/p_{\text{view Q}}]$ described in Memo 115, p. 22 and appendices. If this word is negative, indicating a large discrepancy for some track, ZREAD will reject the whole event (FRJ4, 4). This is not to be confused with k-Pang-Test in ZTn+24.

Glossary of Terms Used in GUTS

This glossary is intended to contain all locations frequently used by by the program, whether or not they are referred to in Memorandum 86, UCRL-9099, or the GUTS flow diagrams. Whenever these GUTS symbols have synonyms in the SNYDER bank this is indicated in parenthesis. SNYDER in its own order is listed elsewhere in this manual (UCRL-9099).

ADR	Block containing some addresses used by program.
ALFA	Range, momentum exponent, set 0.283. See page V-9, Memo 86.
ALPHA	(SNYDER+39→42) A four-word block containing a_λ, the Lagrangian multipliers. See page III-1, ff. of Memorandum 86.
ALPHA1	(SNYDER+35→38) A four-word block containing the values of a_λ after the first step.
BLFCN	A four-word block containing b_λ. See page III-2 ff. of Memo 86.
CHECK	Acceptance level for the fractional differences between χ^2_A and χ^2_B. See page III-5 of Memorandum 86.
CONS	Block containing fixed-point constants of the program.
CORREL	A 90-word block containing $\delta x_\alpha^* \delta y_\beta^*$ matrix after a fit. See page III-7 of Memorandum 86.
DAMN	Sign of Zero C discriminant, set outside GUTS. See page V-10 of Memorandum 86.
DELM	(SNYDER+48) Test function $\overline{\delta M^\nu}$. See Section III-b of Memorandum 86.
DELMOK	Level for $\overline{\delta M^\nu}$ acceptance.
DELTAM	(SNYDER+47) After a step, contains $(\overline{\delta M^{\nu-1}})$. After a cut-step type 5, contains the $\sqrt{\overline{F^{\nu 2}}}$ that was unacceptable. After a cutstep type 6, contains the $(\overline{\delta M^\nu})$ that was unacceptable. After a RJCT6, contains the quantity $\lvert(\chi^2_A - \chi^2_B)/\chi^2_A\rvert$ that was unacceptable.
DKUDXI	A 21-word block containing Zero C k_μ derivatives. See Memorandum 86, page V-17 ff.
DMMASS	Error on missing-mass in missing-mass case. See page V-13 of Memorandum 86.

DUNPEE	(SNYDER+51) Error on missing momentum in missing-mass case. See page V-13 of Memorandum 86.
EIL	An 84-word block containing $E^{\nu}_{i\lambda}$. See page III-2 of Memorandum 86.
ELI	An 84-word block containing $(E^{\nu})^{T}_{\lambda i}$ matrix.
ERROR	A 36-word block containing $\overline{\delta y^{*}_{\alpha}\ \delta y^{*}_{\beta}}$ matrix after a fit. See page III-7 of Memo 86.
FCONS	Block containing floating-point constants of the program.
FIL	An 84-word block containing $F^{\nu}_{i\lambda}$. See page III-1 of Memo 86.
FILOLD	Storage for $F^{\nu}_{i\lambda}$ while $F^{\nu+1}_{i\lambda}$ is being made.
FITCT	Fit counter. This is zero until first fit is obtained.
FNORM	(SNYDER+45) Normalized constraint functions, $\sqrt{\overline{F^{\nu 2}}}$.
FOK	Level for constraint-function acceptance.
GCONTR	(SNYDER+5) Step counter.
GCONTR+1→6	(SNYDER+6→11) CUTSTEP counters (equivalent to GCUT1→GCUT6).
GDELX	A 21-word block containing $(x_i^{\nu+1} - x_i^{\nu})$ after each step.
GFFCN	(SNYDER+27→30) A four-word block containing constraint functions (F_{λ}). See page III-1 of Memorandum 86.
GFFCN1	(SNYDER+23→26) A four-word block containing the values of F_{λ} after the first step.
GFSQRT	(SNYDER+44) Contains the value of $\sqrt{\overline{(F^{\nu-1})^2}}$.
GINA	A 63-word block containing input error matrix G^{-1}_{ij}. See page III-6 of Memorandum 86.
GINA1	A 63-word block containing central 3-by-3 units of output error matrix G^{*-1}_{ij}. See page III-6 of Memorandum 86.
GINOUT	A 441-word block containing full 3P-by-3P output error matrix. See Section IV-b of Memorandum 86.
GLOCA	A seven-word block containing dimensions of output error matrix.
GLOCP	Number of completely measured particles.
GMFCN	χ^2_A. See pages III-1 ff., III-4, of Memorandum 86.
GMFCN+1	Zero-C pass counter (equivalent to ZCCTR).
GMFCN+2	Counter of number of Cutsteps since last good step (equivalent to LCT).

GUTIR1	A three-word block used to store index registers on entrance to GUTS.
GYFCN	A 21-word block containing normalized step functions after a fit. See page III-7 of Memorandum 86.
HELI	A 126-word block containing $(HE^T)^{\nu}_{\lambda i}$. See page III-3 of Memo 86.
HLM	A 16-word block containing $H^{\nu}_{\lambda\mu}$. See page III-3 of Memo 86.
HLM1	A 16-word block containing $H^{-1}_{\lambda\mu}$ for this step.
HLM2	A 16-word block containing $H^{-1}_{\lambda\mu}$ for last step.
KDAMN	(SNYDER+49) Sign of Zero C discriminant inside GUTS. See page V-10 of Memo 86.
KINECT	Block of counters used in loops throughout program.
KLOCD	(SNYDER+43) Distance specified from a vertex for the incident-particle's curvature. See section IV-d of Memo 86.
KLOCI	(SNYDER+2) Number of measured variables.
KLOCL	(SNYDER+1) Number of constraints.
KLOCLL	(SNYDER+3) Flag for anomalous constraint classes (1'C, 2'C, 2CZL2V, 3C bypass, 4C2V).
KLOCM	Number of missing variables.
KLOCP	(SNYDER) Number of particles.
KLOCPP	Number of particles at vertex A in two-vertex classes.
KLOCQ	(SNYDER+4) Flag for 3C bypass (zero if not 3C bypass).
KLOCT	(SNYDER+21) Target mass.
KLOCTT	(SNYDER+22) Target mass at second vertex in a 2V constraint class.
KMASS	(SNYDER+14→20) A seven word block containing masses of particles.
KMSQ	A seven word block containing squares of masses.
KNSTEP	Test level for too many steps.
KNSTEP+1	Test level for too many cutsteps.
LCAFCN	A seven word block containing $E_s k_s$ during calculation of $F_{i\lambda}$, F_{λ}, Y_a, and V_{aj}.
LCBFCN	A seven word block containing $P_s^2/E_s k_s$ during calculation of $F_{i\lambda}$, F_{λ}, Y_a, and V_{aj}.

LCDMTX	A 90-word block containing V_{aj}.
LCKCOS	A seven word block containing x momenta of each track.
LCKEGY	A seven word block containing energies of each particle.
LCKSIN	A seven word block containing y momenta of each track
LCKTAN	A seven word block containing z momenta of each track.
LMAX	The z-axis range of a "zero-length" particle in 2' C case = 6 × WALL. See page V-8 f. of Memorandum 86.
MMASS	Missing mass in missing-mass case. See page V-12 of Memorandum 86.
MMAX	Test level for χ^2 size at end of first step.
MMFLAG	Flag for missing-mass case. Zero if standard entrance. See page V-10 of Memorandum 86.
MPRIME	(SNYDER+46) χ^2_B. See page III-4 of Memorandum 86.
PSEGY	(SNYDER+34) Partial sum of energies ϵ. See page V-1 of Memorandum 86.
PSPX	(SNYDER+31) Partial sum of x momenta π_x. See page V-1 of Memorandum 86.
PSPY	(SNYDER+32) Partial sum of y momenta π_y. See page V-1 of Memorandum 86.
PSPZ	(SNYDER+33) Partial sum of z momenta π_z. See page V-1 of Memorandum 86.
PX	A four-word block containing original momentum-energy translation 4 vector.
REWS	Block containing transformation derivatives from range-momentum routine. See Section IV-d of Memorandum 86.
RIPS+5	This location contains a 10^{10}.
RJCT	(SNYDER+13) If a reject 13 has occurred in GUTS, this contains the location at which it occurred.
UNPEE	(SNYDER+50) Missing momentum in missing-mass case. See page V-12 of Memorandum 86.
WALL	The x-y-plane range of a zero-length particle in the 2' C case. See page V-8 f. of Memorandum 86.
XMEAS	A 21-word block containing measured input variables. See page III-1 of Memorandum 86.
XNEW	A 21-word block containing variables used for step.
XOLD	Storage for x_i^{ν} while $x_i^{\nu+1}$ is being made.

XVERT	A 21-word block containing variables specified at vertex for each step.
ZCCTR	Zero-C pass counter.
ZCKU	A two-word block containing k_u in the zero-C case. See page V-14 of Memorandum 86.
ZDISC	Unsigned discriminant in ZC K_u calculation. See page V-17 of Memorandum 86.
2VFLAG	Flag for two vertex classes, set outside GUTS.

UTILITY ROUTINES AND 4AP

Subpac

The Subroutines are gathered into a separate Subroutine Package Assembly called Subpac, which contains the following routines (and has its own COMMON):

CML	:	Card Mixed Loader
GUTRAN	:	Block Transfer
RINTOP	:	"R into P"
PINTOR	:	"P into R"
RINTOT	:	"R into Time"
MMATRX	:	Multiply Matrix
MTXTP	:	Matrix Transpose
MEQ1	:	Matrix Equation Solution (SAP GM - MEQ1)
SIN, COS, ATAN, SQRT		
MIXINT	:	Matrix Interchange of Rows or Columns or Both
OUT	:	General Purpose Output Routine (NY OUT 3)
Lines 9 or 12	}	Special Purpose Output Routine (for use in conjunction with OUT) to report 9 or 12 floating words per line.

Also defined in Subpac are ten words of RIPS (Range-Momentum Storage). Subpac begins at Location 14000_{10}; to change the beginning of Subpac to the location Y, simply change the first card of the assembly to SCALE SYN Y.

MMATRX

```
a       TSX MMATRX, 4
a + 1   PZE N, 0, M
a + 2   PZE L, 0, AA
a + 3   PZE BB, 0, CC
a + 4     return
```

Matrix multiplication, nonstandard. products are accumulated double precision, then truncated.

89 words
Saves indices
Uses COMMON → COMMON+2

Multiplies the N×M matrix stored row-wise beginning at AA by the M×L matrix stored row-wise beginning at BB and stores the N×L product matrix row-wise beginning at CC.

MTXTP

```
a       TSX MTXTP, 4
a + 1   PZE AA, 0, BB
a + 2   PZE M, 0, N
a + 3     return
```

Matrix transposition, nonstandard

33 words, saves indices, uses COMMON→COMMON+4

Transposes the M×N matrix stored row-wise beginning at AA onto the region headed by BB.

GUTRAN — Block transfer routine, Nonstandard
19 words long

a	TSX GUTRAN, 4	Uses no common
a + 1	PZE A, 0, N	Saves indices
a + 2	PZE B	
a + 3	return	Transfers the N-word block headed by A to the block headed by B.

SQRT: Square root, SAP UASQR4

ATAN: arc tangent: SAP; NA 33.1

SIN / COS } sin + cos. SAP; UAS + C1

RIP — Does (range → momentum), (momentum → range), and (range → time), saves indices. Uses no COMMON.

a	TSX(*), 4 with |Mass| in AC in floating pt, argument in MQ in floating pt.
a + 1	Error return: argument < 0
a + 2	Error return: argument > largest table entry
a + 3	Normal return

where (*) indicates:

		Argument		Function
RINTŌP:	converts	R	into	P
PINTŌR:		P		R
RINTŌT:		R		T

Exit is with function in AC, and in RIPS + 2

Also $\frac{\Delta f}{\Delta x} = \left(\frac{f_{i+1} - f_i}{x_{i+1} - x_i} \right)$ in RIPS + 3

For M = 0, RIP sets the function and its derivative to 10^{20} and gives a normal return.

RIP now does a table look-up in a 103-element table, finds

$$\begin{cases} X_{i-1} \leqslant X \leqslant X_i \\ F(x) = f_{i-1} + (x - x_{i-1}) \left(\frac{f_i - f_{i-1}}{x_i - x_{i-1}} \right) \end{cases},$$

and returns.

The table of moderation times is just a list of the moderation time $\left(\frac{T}{M}\right)_{\eta = \frac{f}{M}}$ down to $\left(\frac{T}{M}\right)_{\eta = 0.047}$, not to $\left(\frac{T}{M}\right)_{\eta = 0}$.

The η vs ξ table extends from $\eta = 0$ to $\eta = 5.0$.

The range-momentum table is identical with the "R vs P" (actually R/M vs P/M) table in Pang. It was constructed by R. R. Ross, using as a calibration point the μ^+ range $R_\mu = 1.116$ cm from $\pi^+ \rightarrow \mu^+ + e^+$, which corresponds to an $\eta(=P/M)$ of 0.2820 for the μ.

For more dense hydrogen, (or for deuterium) increase SCALE [part of Rips Storage in Subpac, SCALE = $R_\mu(H)/R_\mu(X)$] from 1.000 to the appropriate value. Thus in deuterium the μ^+ range is shorter than 1.116 cm by 6.9% and SCALE is to be set equal to 1.069 (which is actually stored in DSCALE in Parak)

The R/M vs P/M table for $P/M \leqslant 5.0$ was calculated by integrating the standard $-dP/dx$ formula from some very low value of P/M up to the desired P/M. The density was varied to give the proper range for the μ^+ from the decay of a π^+ at rest ($\eta_\mu = P/M \equiv 0.2820$). The ionization potential used was 17.5 ev. at $\eta = 1$. It is estimated that the table is accurate to 0.2% (error due to knowledge of I = 17.5); at $\eta = 5$ it is estimated to be accurate to 0.5%.

The main uncertainty in use of the table comes from knowledge of the range of the μ. The R/M entries should be scaled from 1.116 cm to the value measured in the hydrogen or deuterium chamber being used, simply by changing SCALE from 1.000.

The table also contains an entry at the end for $5 \leqslant \eta \leqslant 10^{10}$. This entry is based on a constant energy loss calculated using density-effect corrections and using only delta-ray energies up to 25 Mev. Energy loss to electrons of energy greater than 25 Mev must be added to the energy loss given by the table. The value of dP/dx in the region $5 \leqslant \eta \leqslant 8$ may be in error by as much as 10%. Since the value of dP/dx at large η has never been measured, to our knowledge, we are placing our faith in an article by E. Fermi [Phys. Rev. 57, 485 (1940)] on the density effect.

MEQ1 (SHARE Program GM-MEQ1).

Matrix equation solution.

```
a      TSX MEQ1, 4
a + 1  PZE A, 0, B
a + 2  PZE n, 0, 1*
a + 3  Error return: A singular
a + 4  Normal return
```

Solves matrix equation $A\vec{X} = B$ for $\vec{X}$ given A, B. Saves indices. A is $(n \times n)$, B is $(n \times 1)$.

*If in a + 2, "1" is set equal to zero, the matrix A is just inverted, and B = I does not have to be constructed by the programmer. The answer, $A^{-1}B$, is stored row-wise on top of B. Both A and B are destroyed by MEQ. Further, MEQ assumes a space of n beyond the end of each of A and B for working storage.

Error return → A was singular or ill-conditioned.

SQRT

Square-root subroutine. Saves indices. Uses COMMON → COMMON + 3

```
a      TSX SQRT, 4          (argument in AC)
a + 1  Error return:        Argument was negative: √|Arg| in AC
a + 2  Normal return:       Answer is in AC.
```

SIN - COS

Sin. or Cos. subroutine. Saves indices. Uses COMMON → COMMON + 1.

```
a      TSX SIN, 4 or TSX COS, 4  (argument in AC)
a + 1  Return. Answer in AC.
```

ATAN

Arctangent routine. Saves indices. Uses no COMMON.

```
a      TSX ATAN, 4                (argument in AC)
a + 1  Return. Answer, π/2 ≤ answers ≤ π/2 in AC.
```

MIXINT [Matrix row (or column) interchange]

```
a      TSX MIXINT, 4, X
a + 1  HTR AA
a + 2  HTR R1, 0, R2
a + 3  Return.
```

AA is the first word address of matrix, R1 is the row (or column) index of one row (column), and R2 is the row (or column) index of the second row (column). For x = 0, rows only are interchanged; for $0 < X < 16$, columns only are interchanged; for $X > 16$ both rows and columns interchange.

NY OUT 3. General-Purpose Output Program

Purpose. To set up and print one line (72 or 120 columns) or to output a complete line to a specified tape, or both. Any desired format may be used and conversions from floating binary to fixed decimal, floating binary to floating decimal, or fixed binary to fixed decimal are made as indicated. An echo check is made when printing. Locations of words to be output may be indexed if desired.

Restrictions:

1. The SAP table of operations will be modified to include the eight pseudo operations. Until this is completed, the alternative op codes must be used.
2. No tape checking is done.
3. This program requires the SHARE II board.
4. NYOUT contains internal definition (see assembly); in Subpac it is headed with "B" (for Bark)

Method. For floating-to-fixed conversions, fractional and integral parts are converted separately as integers. A polynomial approximation method is used in the floating-to-floating conversions. All output words are converted to binary coded decimal form and are stored according to the print-wheel position specified. Conversion to card image is made just before printing.

This program is a modified version of GL OUT 2 written by Mr. E. R. Clark of the Lockheed Aircraft Corporation. GL OUT 2 incorporates methods and ideas from the General Electric Package 2 program.

Usage: Calling Sequence

a	TSX OUT, 4 or TSX LINE9 (see LINE9 section of Output)
a + 1	Error Return
a + 2	Pseudo-operations specifying type of conversion desired.
a + 3	"
.	"
.	"

One word of calling sequence is needed for each word to be output in the line (except for Hollerith information, where one word of calling sequence initiates printing of up to 120 columns.) The calling sequence is completed by giving pseudo-operations to specify the type of output desired.

Pseudo-operations that may be used

Op.	A, T, D	Op. mnemonic significance	Optional Operations Code
Indexable			
FTF	k, T, $1000D_1$ + PP	Floating to Floating	SIX
FTX	k, T, $1000D_2$ + PP	Floating to Fixed	SVN
ITI	k, T, PP	Integer to Integer	FOR
HTH	k, T, 1000N + PP	Hollerith to Hollerith	PTH
XTX	k, T, $1000D_2$ + PP	Fixed to Fixed	PTW
Non-indexable			
BIN	BP	Binary Point	PZE
TPW	TN,,INSTR.	Tape Write	PON
PRT	INSTR1,,INSTR2	Print	FVE

In these pseudo-operations, PP specifies the rightmost print-wheel position which will be used for this calling sequence instruction, and consecutive print-wheel positions from right to left will be used as needed. Characters using print positions less than 1 will be lost. Specifying a print position greater than 120 will cause an error return. It is the coder's responsibility to avoid unintentional overlapping of fields. If fields should overlap, a later calling-sequence word will take precedence over any earlier word of this calling sequence.

In any of the calling-sequence words that may be indexed, T may have the value 0, 1, or 2, and the effective address, L, will be k minus the contents of the corresponding index register. Output may be to both printer and tape. In this case the TPW instruction must precede the PRT instruction. Off-line output should be printed under program control. Specifications for output pseudocommands are given in the following paragraphs.

FTF. The operation FTF will convert the word in location L from floating binary to floating decimal. The answer is rounded to D_1 significant digits where $1 \leqslant D_1 \leqslant 8$. The format of the answer will be

-YY - XXXXXXX.

Positive signs are indicated by blanks, and lead zeros in the characteristic YY do not print. If the mantissa is zero, no characteristic will print.

FTX. The operation FTX will convert the word in location L from floating binary to fixed decimal rounded to D_2 decimal places. D_2 should not exceed 8. If D_2 equals zero, a rounded integer will be entered without a decimal point. If the number is negative, a minus sign will print immediately to the left of the leftmost character. No lead zeros to the left of the decimal point will print. If the number is zero, it will be printed with D_2 zeros to the right of the point. If the absolute value of the number exceeds 34, 359, 738, 367 it will be printed in floating-decimal form as described above.

ITI. The operation ITI will convert the word in location L to a decimal integer and print it without a decimal point. If the word is negative, a minus sign will print immediately to the left of the high-order digit.

HTH. The HTH operation is used in printing headings or words of alphanumeric information. The N words of information starting in location L will be set up for printing across from left to right so that the rightmost character will fall in the print position specified. N should not exceed 20. Normally, the calling sequence for a full line of heading will be:

```
        TSX OUT, 4
        Error return
        HTH L,, 20120
        TPW TN
(and/or) PRT)
        Normal return
```

The words from L through L + 19 will print from left to right across the sheet.

XTX. The operation XTX must be followed by the operation code BIN (XTX not followed by BIN, or BIN not preceded by XTX will give an error return*). BP gives the binary point of the word in location L of the preceding XTX operation. This word is converted to fixed decimal rounded to D_2 decimal places and printed with a format similar to that for the FTX operation. D_2 should not exceed 8. A BP value outside the range 0 through 35 gives an error return.

---

*For a modification of the use of BIN permitting OUT to write octal, see OUT -5.

TPW. The operation TPW signifies that the entire line which has been set up is to be output on the tape unit specified by TN. The decrement part of this instruction may contain any of the following decimal numbers which will control off-line printing.

Number	Significance
16	Suppress space
48	Single space
0	Double space
1-9 (at Berkeley 1 means Page Restore)	Skip to channels 1-9
33-41	Short skip to channels 1-9

When output is to tape, print position 1 should not be used for printing data.

PRT. The PRT command initiates the setting up of the card image (s) for the output line. The address part of this instruction may contain any of the following decimal numbers which will control on-line printing.

	Number	Significance
INSTR1:	241	Programmed overflow causing skip to channel 1
	242	Skip to channel 2
	244	Double space
	245	Suppress spacing

INSTR2 is the decrement part of the PRT instruction; it may contain the decimal number 243 which will provide an extra space after printing. An attempt to print a non-Hollerith character will cause a blank to be inserted in place of the character.

Usage: Error Codes. The following codes are left in the AC on an error return. A is the address of the calling sequence word in error. In the case of XTX followed by BIN, A will be the address of BIN.

-0:	Echo-check error - the line just printed contains the error.
+A, 0, 1:	XTX not followed by BIN, or BIN not preceded by XTX.
+A, 0, 2:	BP outside range 0-35
+A, 0, 3:	PP greater than 120.

Coding information

1. Space required: 408 locations, of which the last 51 are erasable.
2. Timing: Printing will take up to three cycles per line depending on number, length, and type of words being converted for printing.

The following table gives results of timing test with tape output only.

Type of conversion	Words/record	Millisecond/ record	Milliseconds/ word
Floating-floating	6	107	18
Floating-fixed	8	96	12
Fixed-fixed	8	93	12

3. Card decks available:

UA SAP symbolic deck labelled OUT3 0000-0361
Relocatable binary deck labelled NY OUT3 01-18

LINE 9 and LINE 12

Barbara Levine has added these routines to the version of OUT in Subpac. They are a simplified calling sequence for writing 9 or 12 floating words per line. Their calling sequences are punched on cards, and may be read in the assembly of OUT, near the end.

NYOUT 3 will now output octal. The command BIN is used for this. If this command immediately follows XTX as discussed in the NYOUT 3 writeup, it performs as specified. Otherwise, with the modification the command works as follows:

BIN k, T, PP.

This command will convert the word in location k to a 36-bit unsigned octal integer with the rightmost print-wheel position at PP. The address may be tagged by a Tag T = 0, 1, or 2 (leading zeros will be suppressed).

UCRL-9099
Utility: SSCTCS-1
July 18, 1960
William E. Humphrey

Selfloading Routine to Transfer Core Storage to Tape in Selfloading Form (SSCTCS)

Purpose

To provide a means of transferring memory to tape in a form which can be easily loaded back into memory (Written for IBM 704).

Requirements

a. First 24 locations in memory are used by the single binary card containing the routine.

b. Last location in memory is filled in by the routine with a checksum of all previous locations.

c. Control is tranferred to next-to-last location in memory when the tape is successfully loaded into memory.

d. Tape unit 1 is used to write the memory on tape or to read the tape into memory.

Writing Memory on Tape

a. Load 1 card selfloading routine "SSCTCS" with a blank tape on unit 1.

b. The routine will calculate a checksum, copy memory onto tape unit 1, and stop at location 15_8 with HTR 15_8.

c. The memory is unchanged except as mentioned in (2).

Reading Tape into Memory

a. Push load tape with tape, prepared as described in (3), loaded on unit 1.

b. The tape will read itself in, compute a checksum, check the checksum and RTT indicator, and transfer control to the next-to-last location in memory.

c. If either the checksum or RTT indicator do not check, the routine stops at 15_8 with HTR 15_8.

d. The memory is as in (3c) when the tape was produced.

Size of Memory

The SSCTCS routine accommodates itself to any size memory, but the size of the memory should be the same when reading the tape as when the tape was written. Also, special versions of SSCTCS have been written to transfer less than a full memory onto tape.

UCRL-9099
CML-1
May 1, 1958

Dinsmore Mixed Loader (CML)

Coded by Robert Dinsmore, October 8, 1956
Modified by Leota Barr, May 1, 1958

Purpose

1. To convert information on cards to BCD form, then translate each numeric part into a binary number and store or use it, as defined by its preceding CML code letter.
2. Or, to skip the card-reading part of the coding, and translate the BCD information located in core memory.

Restrictions

1. The converted binary number before the decimal point must be less than 2^{36} or overflow will occur. Also, the binary number after the decimal point must not have a bit smaller than 2^{-35}. By using a sign (+ or -) to define the first bit as 0 or 1, a binary number equal to 2^{36} can be entered into the machine (i. e., an octal instruction). Hence, exclusive of the sign, the maximum size of a decimal number is 10 decimal digits, and of the octal number, 11 octal digits.
2. A numeric value must be contained completely on one card, because after each card is read, all translating is completed then a return to the calling sequence is executed.

Method

1. The card format is variable, with all nonnumeric characters acting as sentinels defining the numeric values. The information on a card is translated to BCD, then scanned character by character and each numeric value is converted and used according to the CML nonnumeric character which precedes it. All blanks are ignored.
2. For $(a)_{12-17} \neq 0$, BCD words in core memory are processed as described above (see calling sequence).

Description of New Features

1. CML and ML have been combined.
2. S, W, Y, and Z code letters have been added.
3. L(0) does not have to be given initially.
4. Ten decimal digits, on either side of the decimal point, can be entered without an overflow error.

Calling Sequence

LOC	OPN	ADD	TAG	DEC*
a	TSX	CML	4	No. of BCD words*
a + 1	HTR	RL†	ϕ	FWL of BCD*
a + 2	Error return			
a + 3	Illegitimate-character return			
a + 4	Normal return at the end of each record (card or LW of BCD)			
a + 5	W, Z, and end of file return			
a + 6	Y return			

*If a card is to be read, the decrement parts of a and a + 1 are not filled in. In fact, a card is read only if $C(a)_{12-17}$ equals zero, for otherwise, CML will convert and translate BCD words stored in memory starting with the BCD word at FWL (the first word location of the BCD words).

†RL is a "reference location" for storing the converted binary word in core memory.

Error Returns

(a + 2) Overflow, underflow return. Overflow and underflow are indicated by all ones or all zeros in the accumulator, with the sign the same as the sign of the word causing the error return. When (a + 2) is set equal to TRA 1, B, and this instruction is executed, CML will store the contents of the accumulator and return to its normal execution.

(a + 3) Illegitimate-character return. All characters other than the CML code letters are called illegitimate and will cause CML to place in index register B, the complement of the numeric value that follows the illegitimate character and then to return to the a + 3 exit.

The CML Sentinels

Storage locations in core memory for the numeric input can be defined and varied at anytime by giving a combination of the CML sentinels, RL, L, and K. RL is given in the calling sequence and remains fixed. (Notice that it can be set equal to zero.) Ln can be given anywhere in the input and when interpreted causes n, an integer, to be added to RL to form a new FWL for the numbers that follow. Also, the increment, Δ_k, is set equal to +1. Kn also can be given anywhere in the input, and the integer, n, is used as the storage-location increment, Δ_k, until either another Kn or Ln is given.

The numeric values to be stored which are given immediately preceding and following the Kn are placed n memory locations apart. Hence, the numbers, n_i, i = 0, 1, . . . , m are stored in memory locations, $\bar{Y} = RL + n_L + \sum_{k=1}^{i} \Delta_k$, with n_L the integer associated with L. Notice that it is not necessary to give an Ln or Kn in the input.

Code Letters which Define the Input.

An: n is stored in $C(\bar{Y})_{21-35}$, where $\bar{Y}$ is the current input storage location.

Dn: n is stored in $C(\bar{Y})_{3-17}$.

Cn: n storage locations are cleared Δ_k locations apart, starting with $\bar{Y}$.

On: n is interpreted as an octal number. "O" must be given after any other letter code defining n, since all letter codes except O, +, and - preset CML to read numeric values as decimals. n will be placed in the least-significant positions in its storage location. Also, the O must be given before the sign of n. All numeric values not preceded by O are interpreted as decimal numbers.

±mSn: m is stored in n storage locations Δ_k locations apart, starting with Y.

In: n is converted into a fixed octal integer with zeros on the left occupying positions not defined by n. When "I" is not given, n is converted to a floating octal number.

±mEn: n is a decimal exponent of m. The numeric value $m \times 10^n$ is converted to a floating octal number, then stored.

±n: n is read as an integer and converted to a floating octal number.

.n: n less than one is converted to a floating octal number.

±mPn: and ±mQn: n is a binary power of m, which is converted to a fixed octal number with P indicating binary scaling from the right, and Q, binary scaling from the left.

Numeric values in a sequence can be separated by any of the CML sentinels except the point (.), which serves as a separator only when the sentinel preceding it is another point, (i.e., $+.n_1.n_2.\ldots.n_m$) or one of the code letters A, D, C, S, O or K (i.e., $An_1.n_2Dn_3.n_4\ldots$). These letters refer only to integers.

The "-" sign following A, D, C, L or K will complement (2^{15}), its associated n. Either the 11, 12, or the 8/3, 8/4 punches may be used to represent the signs.

Exits from CML

Z: All registers are reset and a transfer to the end of file exit (a + 5) is executed.

Y: Same as Z except a transfer to (a + 6) is made.

Wn: Same as Z except the complement of n is loaded into index register B.

Comments

CML can be used to insert instructions if the first binary bit is written as a + or - (i.e., 0-300000 104222 or 0-070000 102221).

Problem headings, which must be changed with each problem variation, can be placed following the exit letter-code on the last card of the input. After returning to the calling sequence, the contents of the last card in BCD can be found in common+13 through common+24.

Space Required

437_{10} words of coding (704)
25_{10} words of common (704)
49_{10} words of common (709)
454_{10} words of coding (709)
506_{10} words of coding (709) CML deck and CXS3 decks

Example

Let (a + 1) = 000000 000700 (RL), and let the data card contain +1+1. 1. 1+1+. 1. 1E1K2D011Z. Then the binary numbers stored in core memory will be:

(700) = 201400 000000
(701) = 201431 463146
(702) = 175631 463146
(703) = 201400 000000
(704) = 175631 463146
(705) = 200777 777777
(707) = 000011 000000.

Preliminary Write-Up on 4AP

4AP is a 704 version of LC9AP9. It will assemble SAP format cards as well as the extended 709 format cards. This write-up will be extended and probably modified later.

Operator Instructions

Tape assignment.

1. 4AP
2. Input tape
3. Collated tape (save)
4. Intermediary tape -- supposedly used only on batch assemblies; presently selected on some nonbatch assemblies.
5. BCD correction tape
9. Listing tape (have printed)

Sense-switch assignments

6. Up--No corrections
   Down--Corrections to be read according to S. S. 1.
5. Up--Prepare off-line listing
   Down--Suppress off-line listing
4. Up--Print definite errors on-line
   Down--Suppress on-line error print
3. Up--Suppress on-line listing
   Down--List on-line
2. Not used
1. Up--Input from BCD tape
   Down-- Input from card reader

Note that if S. S. 6 is up, S. S. 1 refers to input mode of main program; if S. S. 6 is down, the main code is assumed to come from tape 2 and S. S. 1 determines the source of corrections.

The Alter Feature

The assembly program assigns a unique number to each card of the input. This assignment is retained in the compressed BCD tape 3. These numbers are presently on the far left side of the listing, they may be moved in the future. If the collated tape, 3, from one assembly is saved and used as input for the next assembly, then corrections may be made by referring to these alter numbers. The ALT pseudo-op is used to do this.

Example: If the correction deck is

```
ALT 59
CLA X
ALT 572, 1096
ALT A105, B200
STO Y
```

then the CLA card will be inserted after card No. 59, cards 572 to 1096, inclusive, will be deleted, cards A105 to B200 inclusive will be deleted and the STO card will be inserted. Notice that the alter number is a four-digit number and if more than 9999 cards are in an assembly the first character will become alphabetic.

Differences from SAP

1. There is no library routine provision presently. It will be added shortly.

2. The REP pseudo-op will not work.

3. The SKP pseudo-op presently results in a page restore regardless of the content of the address field. This will be changed to conform to SAP with the exception that a blank address field will result in a page restore.

Additional Features

1. Operation codes may be from three to six characters long.

2. The variable field must be separated from the op code by at least one blank and begin no later than column 16.

3. 709 codes may be assembled.

4. A fourth field in the variable field is interpreted as an octal number and inserted in the word starting from the left.

5. A number or symbol in the address field of an ABS card will be interpreted, truncated modulo 8 and punched as the first three bits of 9L on all cards it controls.