ON-LINE PROGRAMMING

R. Yaari, T. Gilead, B. Reuter, E.E. Ronat and R. Yaffe,
Weizmann Institute, Rehovot, Israel

I. Introduction:

Our control program ZOO was written for a PDP-9 computer. The coding was done by using most of DEC's utilities, like the Text Editor for creating the source files of subroutines, the Assembler MacroA-9 (a restricted version of Macro-9) the debugging aid D.D.T., the relocatable linking loader of the keyboard Monitor System (KBM-9), and the UPDATE for building a library of subroutines in binary mode.

The fact that all subroutines were stored on DEC tapes, enables us to modify easily any section of the program, without affecting the performance of the entire control program.

The program occupies:	4K	words of instruction
	2K	" for buffers
	1K	" common area, working space
	1K	" KBM resident (reserved for future developments)

It should be emphasized that the KBM was used only in the preexecution phases of the program. Once execution started the on-line program was completely stand-alone. Thus we wrote our own special purpose and compact handlers for DEC tape, Magnetic tape and teletype. Furthermore, during the execution phase the resident monitor area (1K) could be overlaid by storage space, (e.g. buffers), and thus no serious loss of storage resulted from use of the monitor system.

In designing our program we have tried to learn from all examples, to make it as simple as possible from all points of view, namely:

(a) No sophistication in the management, no Executives which control queue of tasks, software priorities etc. We have just used a simple sequential set of operations, out of which simultaneous jobs are handled via API handlers.

(b) Simple, straightforward I/O organization, 3 record types for the 3 different data structure. No circular buffering, we use double buffers instead.

(c) Since we have not used external assembler, all subroutines are kept in a file organization on DEC tape, and the program is loaded from a dump area of a DEC tape.

All these could, to some extent, slow the operation, but we feel confident that is only a small fraction of the "real things" (like fiducials and CPTS).

II. Program Structure:

The program consists of 6 major blocks:

(1) MAIN - Initializes S.R. activities, sets API trap addresses, readies I/O devices, checks various reference marks, calibrates fiducials, gets date, time, operator information, roll, tape, etc.

(2) LION - Controls all measurements. Calls upon XYLAM to perform 3 views measurements. ID information is read from DEC, film is moved to picture and stage is brought approximately to 1st fiducial.

(2) XYLAM - Measures one vertex at a time in the following sequence:

(a) 4 fiducials are measured by a push button, after stage was driven automatically to the expected position.

(b) Stage moves to vertex, vertex scan starts after manual adjustment, future vertex calculation is done, while the Q channel fills one buffer, the other is being written on tape, in addition the data is being displayed on the scope at the same time and various checks are being made.

(c) Crutch points measurements on all tracks, part of which are driven as flagged cpts, namely stage is driven to the expected position.

(4) Device handlers - Closed subroutines to handle the various devices of the S.R., including their API service sections.

(5) PILIM - A collection of special functions, which could be entered by hitting (ALT) key and typing a letter. These functions can

perform remeasurements, handle rejects, sign-in, sign-off, device checkout, fiducial calibration, data display, magnetic tape operation, etc.

In other words it can "Interrupt" the regular flow of program and perform some function and resume its flow afterwards.

(6) Utilities - Arithmetic functions, formating routines, etc.

I/O Structure:

(a) Input from DEC tape, 40 words per event: ID, VERTEX, CODES, when a new ID is to be read, it is already found in buffer, and the program initiates another read.

(b) Output on a 7-track, IBM compatible magnetic tape. Up to few months ago we were using 556 bpi without any trouble. We have switched to 800 bpi in order to have only 1 magnetic tape per roll, and we tolerate up to 3-4 parity errors per roll.

We have 3 types of records:

ID - 24	words,	scan information
Q - 524	"	512 data (128 hits) + 12 Header
FVC - 56	"	Fiducials, Vertex, Crutch points

Each record has a type signature, which identifies it, together with its length. The records are written as independent physical records, and this makes it easy to read it, without any need for deblocking, unpacking, etc. On the other side, it occupies more space on tapes, and after 2 years of operation the number of tapes is quite large. This is solved by an extra operation in the long chain of analysis, namely 5-6 tapes are being stored on one 1600 bpi, 9 track tape, after all the data was checked.

IV. Timing of Operations:

The following table shows how much time is consumed at each phase of measurement. All numbers are given for 3 views: (each view is approximately 1/3 of the time).

1)	Film movement, view switching, event verification	14 sec.
2)	4 Fiducials measurement	14 sec.
3)	Vertex drive, adjustment and scan	19 sec.
4)	Crutch points taken on all tracks (4 prong)	28 sec.
		75 sec.

We are confident that we can easily improve this situation by taking minimum cpts (30% better) and by using semiautomatic fiducial measurement, so that we could reach a time of about 45 sec/event.

V. Diagnostic Routines and External Utilities:

We have a local version of RAVEN, called OREV. It is an adaptation of LRL's routines, though vastly altered and expanded to accommodate the various changes in the Q-channel logic.

We have introduced some extra features which provide us with more useful functions:

Controlling the ramping (periscipe movements), collecting data on a magnetic tape, dumping or printing interesting parts of a buffer, and also the possibility to magnify selected areas according to radius range.

Our library of utilities includes some packages which can perform data transfer from the following devices:

Magnetic tape to DEC and vice versa

DISK to magnetic tape " " "

OFF-LINE PROGRAMMING

I. POOH:

We have started with an old version of LRL POOH (FILTER+MATCH) which ran on a 360 IBM, and was adapted to our local GOLEM. Since then it was modified quite extensively along 2 major lines, using an IBM 370/165 computer just recently installed.

(1) Our local data structure and the different machine parameters have dictated natural line of modifications. More debug aids, reject handling, more switches for skipping, selecting various types of events. Our calibration library file resides on a direct access device and is being updated by our CALB program. By using Fortran instruction one gets access to the proper deck of calibration by reading a directory record which points to the proper location on disk. (See Define File instruction in IBM's FORTRAN IV MANUAL).

(2) Improvements along the developments made by LRL in their NU-POOH. We have introduced the following items:

(a) Averaging 40 points per track into 12

(b) Better tracking out and in.

(c) Data checking, like monotony, bad points.

(d) Better matching routines.

We intend to introduce very soon the following features:

CLEANUP routines for dropping points from PH average calculation, when it belongs to 2 tracks simultaneously.

We have not to-date used the REDO feature because of various reasons (these events often failed in kinematics). But we intend to reactivate it soon. (A 5% gain in POOH success rate is expected).

Long chopping and negative cpts handling will be introduced, once the extra hardware is implemented.

Our program occupies 250K bytes under FORTRAN H (OPT=2) compiler, including buffers and plotting routines. It runs with 95% cpu utilization,

and process 30 events/minute. Most of the code is in Fortran, with very little assembler code (mainly for non standard I/O).

2 seconds per event is an overall average for our data, but obviously for a 10 track/view event, it will run up to 4-5 seconds, spending most of the time in MATCH. Let us look at the various POOH rejects chosen from a sample of 5000 events.

POOH FAILURES:	
NO BEAM TRK WAS FOUND	.1%
PARITY ERROR (ON INPUT)	.5%
TOO FEW TRACKS	1.0%
NO MATCH	8.6%
TRACK FAILED VOLUME	.15%
NO EVENT TYPE IN DICTIONARY	.2%
	10.6%

II. TVGP-SQUAW (GEOMETRY & KINEMATICS)

We are using a version which is very close to LRL's SIOUXC. 12% of our events fail in these sections of the system. 5% of them are due to non beam events (we are using strict criteria in this particular experiment), and 3% are events with unacceptable χ^2 . . The rest 4% are mainly from Geometry failure for various reasons like: Bad points scatter, wrong curvuture, etc. For some time we had troubles with handling short tracks, and stopping tracks. We have solved it completely by introducing LRL's TYPE 4 tracks (kinematical variables are azimuth, slope and momenta). Some of the unacceptable χ^2 were solved by introducing Dalitz Pair handling. The event type is reduced once a pair is detected. These 2 programmes are running independently in core.

180K is occupied by TVGP and 240K by SQUAW. We have attached to SQUAW a SQHIST section which gives us histograms of various quantities, like χ^2, beam distributions, pulls, and also a summary of failures and successes. These 2 programmes are 99% cpu bound and their corresponding rates are 120 events/minute for TVGP and 80 events/minute for SQUAW.

To sum it up, we have for the 3 big programmes the following timings:

POOH	30	events/minute	2 sec/event
TVGP	120	"	1/2 "
SQUAW	80	"	3/4 "
		total	$3\frac{1}{4}$