BNL 40011
INFORMAL REPORT

BNL--40011

DE87 013298

Interactive Design of Accelerators

IDA

Mark Q. Barton

Research Supported by the
OFFICE OF BASIC ENERGY SCIENCES
U.S. Department of Energy
Washington, DC

NATIONAL SYNCHROTRON LIGHT SOURCE
BROOKHAVEN NATIONAL LABORATORY
Associated Universities, Inc.

Under Contract No. DE-AC-76CH00016

Interactive Design of Accelerators

IDA

Mark Q. Barton

Abstract

IDA is a beam transport line calculation program which runs interactively on an IBM PC computer.[1] It can be used for a large fraction of the usual calculations done for beam transport systems or periods of accelerators or storage rings. Because of the interactive screen editor nature of the data input, this program permits one to rather quickly arrive at general properties of a beam line or an accelerator period.

Background

The original version of this program was written in BASIC by Max Cornacchia for an Apple II series computer.[2] The author made a number of additions and modifications in this original program; then decided it would be interesting to recode the program in PASCAL. This was done in Apple PASCAL which is an Apple implementation of the UCSD PASCAL. In making this transition, the author was impressed with the convenience of the "command line" format of the commands used by UCSD PASCAL. This format was incorporated into the beam line code. The result is a system in which nearly all which one needs to know to select various options of the system are indicated on screen with little need to turn to a manual or help pages. Later, the program was copied into a suitable format for TURBO PASCAL.[3] TURBO PASCAL will run on an Apple with Z80 card; the same code runs without significant modification on an IBM PC. A large number of improvements have also been added to the TURBO PASCAL version. The author has only tried the code described below on standard PC, XT, and AT computers. At least 256k memory is required. The author has not tried the code on any so-called compatibles but would not expect difficulty if the machine is truly compatible. The graphics package uses a Hercules graphics card. If a different graphics board is present, the program will probably crash if the plotting option is requested. The graphics option may possibly be reinitialized for other graphics cards; the method of doing this is indicated in the appendix. The program with nearly all features described below runs on the author's Apple IIe with Z80 card. Because Apples and other CP/M implementations vary so much, the author is not prepared to distribute or support this code. The source codes are all available on the disc supplied. The CP/M user may try at his own risk to compile and use this code on his own system. The appendix indicates some of the configuration dependent features which may need to be changed.

One final comment about the configuration; this code was developed on an Apple with a 64k memory limitation. It was, therefore, written in a main program and four major chains; a TURBO PASCAL overlay structure. It might be possible to avoid this chain structure in the IBM PC. On the other hand, there are some programming conveniences in keeping the chains. The slight delay in waiting to load the various chains is not a significant annoyance to the user, particularly if one works from a hard disc. This chain structure might, however, be a major problem in converting the code to some other PASCAL implementation.

---

1. IBM PC refers to a line of computers manufactured by the International Business Machines Corporation.
2. Apple computers refer to machines constructed by the Apple Computer Corporation.
3. TURBO PASCAL is a software product developed by Borland International, Scotts Valley, California.

How to use IDA

I will now give you a tutorial on how to use IDA (and write these instructions in the first and second person as that seems such a user friendly thing to do).

If you have a hard disc, create a directory with any name you choose and copy all of the IDA disc into this directory. If you do not have a hard disc, boot your system and put the IDA disc in A. Working within the selected directory, or using A as your working disc, type IDA to start the program.

After the program is loaded and some variables are initialized, you will see at the top of the screen a Command Line which lists the available options. One letter of each command is emphasized by being capitalized and followed by an open parenthesis ,(. You can select the option indicated by typing in that letter, either upper or lower case, and <u>without</u> carriage return. As the various chains or options load in, you will see different command lines. At nearly every step in the use of this system, you can simply look at the top of the screen and see the options available, each of which is available with a single indicated keystroke.

The input is not buffered so you cannot type ahead. Wait for each screen to complete before typing a new command. Let us quickly look at the initial options available. The command line is:

E(dit, **G**(et, **S**(ave, **L**(ist, **T**(ype, **M**(ult, **C**(ycle, **P**(lot, **Q**(uit

I first identify what each of these options is with a few comments about what it does and how to use it. Some of the items require a more lengthy explanation which I will return to later.

E(dit is the screen editor used to enter new beam line elements, change parameters, etc. This will require a more detailed discussion.

F(it is a two parameter fitting routine which I will describe after some of the calculation modes and formats are clear.

G(et is a program to get the data from a previously developed (and saved) beamline. Perhaps, for the purpose of this tutorial, you should now exercise this option. Type g (or G), wait for the disc operation to load the correct chain, and you will see a prompt asking for the name of the beam line. Each beam line has a name which may be supplied or changed during the edit operation. That name is used for future "Get" references and is included in the label for all printouts. Type VUV. This is a beamline that I have prepared; it is actually a superperiod of the NSLS VUV storage ring. Unfortunately, I have not been able to figure out how to get IDA to give you a display of the directory for the Get operation so that you know what data files are available on the disc. Furthermore, if you ask for a non-existent file, IDA crashes! If you intend to use IDA to work on existing beam lines, then do the DOS operation, DIR *.DAT, before entering IDA and make a note of the beamlines available. If you are following this tutorial, you will note that after the disc quits whirring, your screen will contain a list of beam line elements and their parameters with the beamline name VUV.

S(ave is the obvious complement of Get. The beamline is saved on the disc in your working directory. You do not have to type in any argument for the save operation. The beamlines are saved in the format beamlinename.DAT, e.g. VUV.DAT. In using IDA you never see the .DAT suffix; it is useful for DOS operations. Note that the Get and Save routines handle files in the same working directory as the IDA system. I have not found this to be inconvenient. If the files are needed in other directories, copy them with DOS commands after quitting IDA. A S(ave operation destroys any previous saved file of the same name. If you need any kind of serialized sequence, you will have to explicitly change the filenames in the editor. The files, by the way, are structured records containing mixtures of ASCII and binary number items. A DOS operation like TYPE VUV.DAT will generate gibberish. Also, there is a TURBO PASCAL version which uses the 8087 coprocessor. IDA can be compiled and run with either system but data files generated with one cannot be read by the other because different word lengths are used for floating point numbers.

L(ist is an operation which causes the beam line elements and their parameters to be printed on your printer. If you have followed my suggestion and done a Get VUV, you see on your screen a copy of the same information that will be printed . L (or l) will cause this screen to be printed assuming you have a standard printer on the standard printer channel. One slight difference; if your beam line has more than 17 elements (IDA allows 49), they do not all show on the screen. They are all printed, however, when the List command is exercised.

T(ype functions gives the focusing functions for the beamline. Type T now. IDA prompts you for P (printer) or S (screen). Type S. The formats of the results are rather obvious and well labelled but a few comments are necessary. The focusing parameters refer to the beginning of the elements (unlike SYNCH, for example, where the values refer to the ends of elements). There is an additional line at the end which refers to the values at the end of the last element. Phase shifts are in radians rather than fractions of 2π as used by SYNCH and MAD. There is some summary information at the bottom of the page including the total phase shifts (redundant because these numbers are at the bottom of the respective phase shift columns), total beam line length, and total bending angle (in radians).

M(ultiply is the operation of multiplying all of the matrices together to generate the focusing functions shown by the T operation. Try it now. When this option is requested, you will get a screen which indicates the initial values of alpha and beta in each plane and the initial dispersion function and its derivative. You will notice a command line which gives the options:

C(hange, **R**(eturn, **G**(o

The R(eturn option permits you to go back to the main screen if you decide you do not really wish to do the computations at this time. The change option permits you to change the initial conditions. Use the up-down arrow keys to move the cursor to the parameter you wish to change; then hit C. The screen will prompt you for the new value. Type in any

floating point number followed by carriage return. Note that whereas IDA does not require a carriage return for single character commands, it does require return for data input. This is true throughout the system. Do not set an initial beta to a zero or negative number. There is no test for this unphysical condition; the program simply crashes! When you have finished setting the initial conditions to the desired values, hit G and IDA will proceed (after a somewhat facetious message) to do the computations. If you have followed this tutorial so far, try it. Move the cursor with the arrows to the horizontal beta, hit c, type in 1.0 then hit g. When the computations are done, you will see the original command line again. Type T, wait for the Print or Screen prompt, type S and you will see the new emittance parameters. Notice that the horizontal functions are no longer the same at the end of the lines as at the beginning as they would be if the beam line were a period of an accelerator. Clearly this multiply routine is what is used when IDA is used for beam transport lines, insertions, etc. It can also be used for accelerator cells if experimentation is needed to find stability regions.

C(ycle is the normal system used when the beam line is considered to be a period of an accelerator or storage ring. Cycle multiplies the two-by-two matrices together to find the phase shifts in each plane. The initial alphas, betas, dispersion, and derivative of the dispersion are computed from the product matrices in the standard manner. IDA then automatically enters the M(ultiply routine with these initial values and computes the functions all the way down the beam line. If you have been following this tutorial, try it. Type C (you get two facetious messages this time). Then type T, type S, and you will see the horizontal beta functions set back to the machine period values.

Now that you have seen the type of things IDA can do, I will teach you to use the editor. This is the feature of IDA that is particularly user friendly and makes the system so easy to use for rapid development of new systems. When you request E(dit, you see a new command line:

A(dd,**C**(hange,**D**(elete,**I**(nsert,**S**(creen,**U**(p,**N**(ame,**L**(ine,**L**(eave,**E**(dit,e**X**(punge

The editor is a screen editor; you can use the arrow keys to move the cursor around on the screen to the various elements and their data fields. Move the cursor to the gradient of Q1. Hit c (for change). You will see the prompt to type in a new value of that parameter. Type in a floating point number followed by return. If you type in something that is not a legitimate floating point number, IDA will beep at you and insist you try again. IDA, unfortunately, cannot tell if your number makes any sense, physically. Type D (for delete). Q1 disappears completely. Type I (for insert) and you will see the cursor line go blank with an asterisk * prompt at the left. (The element blanked out is not lost; when you are done with the insert, it and all succeeding elements reappear a line lower). Type Qx followed by carriage return (to reinsert a quadrupole). You will then be prompted for the value of the length; type in a floating point number followed by return. Similarly, you type in the gradient in response to the prompt. Thus, the basic concepts of changing an element's parameters, inserting new elements, or removing an element are quite obvious and simple.

A few special notes. When inserting a new element, the first character must be a capital letter and must be D (for drift space), Q (for quadrupole), M (for bending magnet), or S (for sextupole). The element name can be up to six characters long. After the first character, other symbols such as digits or lower case letters can be used. The lower case letters are distinct, i.e. Qa is different than QA. As noted before, although each command throughout IDA requires only a single keystroke, the input to each data field in the editor must be terminated with a carriage return. The units used by IDA are units of length for element lengths and radius of curvature in bending magnets. Gradients are in units of inverse length squared (gradient divided by $H\rho$). Sextupole strength is also in units of inverse length squared since the number (as used in the chromaticity calculations) is the product of the usual sextupole strength times the length, i.e. thin lens model. The normalization and sign is the same as that used in SYNCH and MAD. The sign convention for gradients in bending magnets and quadrupoles is that positive sign corresponds to horizontal focusing. For a bending magnet, use a hand calculator to preset the radius of curvature to be consistent with the path length and the desired bending angle. After entering the length, gradient, and radius of curvature, you also have to enter the entrance and exit angles. The conventions here are that these angles are zero for sector magnets and are in radians for non-normal cases with the usual sign conventions. For example, for rectangular magnets, each angle is just half the bending angle of the magnet.

If there are several elements in a beamline with the same name, you will notice that if you change a parameter of one of them, that same parameter will change for all elements of that name. You may change the name of an element but use this option with caution. If you try to change the name of an element and type in an illegal name (not starting with M,Q,D, or S) or if you try to change the type (e.g. you try to change a magnet to a drift space) the program will either crash or ignore your request. Sorry about that. Also, if you set the name to that of an element that is already in the line, IDA will not automatically reset its parameters. Thus changing the name is dangerous. One case where the option is useful; you have a beam line with several identical elements and you decide later that you wish to make them distinct. In this case, use the change option on the name. When in doubt, delete followed by insert is always a safe way to rename an element.

Also note that if you insert an element whose name already exists in the beamline, IDA knows its parameters and does not prompt you for them.

As you move the cursor around, if you have more than 17 elements in the line, you cannot see all elements of the line at the same time. The screen will scroll as needed so that you can access all elements. The scrolling is a little slow. The S(creen and U(p commands move several lines at a time to improve this situation. Scrolling is only available while in edit mode.

N(ame permits you to name or rename a beamline. You will notice the prompt characters on the screen where the beamline name indicator usually appears. Type in the name you wish (followed by return).

I assume that you are still following this tutorial. Type X (for eX(punge). The parameters and the name of the vuv beam line disappear. You are now in a position to create a new beam line. Let us make up a simple FODO

lattice. Type N and enter FODO. (Remember all data fields must be terminated with return.) Type A for A(dd. This is the command for adding new elements to the end of a line. You will note that the command line at the top of the screen is now a mnemonic reminder of the legal beam line element types. Type in Qf. In response to prompt, type in 1.0 for length, and 0.1 for the gradient. Type in Ds for a short drift. Give it a length of 0.2.

Similarly enter Mag, length of 3.0, radius of 11.4592 (chosen to make six periods form a complete ring), gradient of 0, entrance and exit angles of 0. Continue to get:

Qf
Ds
Mag
Ds
Mag
Ds
Qd (length of 1 and gradient of -.1),
Ds
Mag
Ds
Mag
Ds

Notice, again, that as elements are named whose properties are known, you are not prompted to repeat parameters; in fact, they are not even shown on the screen. Also, if while adding elements you type in illegal names, (in this case there is no crash), the element shows on the screen but is actually ignored as you will see when you finish adding elements. Do not try to correct typing errors during the A(dd operation. When you are finished with the entries, Type in an element name starting with capital E (for End of beam line additions). Your screen will now display all of the parameters and omit all of your illegal entries. Use C(hange, D(elete, or I(nsert to fix any typing errors which you made during the A(dd operation. When you believe you have the FODO line parameters you wish, type L for L(eave and you will be back to exercise the other options of IDA. For example, cycle this line and use T to look at its properties. Before leaving the discussion of the editor, one other option should be mentioned. If the beamline is to be a period of an accelerator and the period has planes of mirror symmetry, IDA recognizes a reflection operator. Simply type in the half period, then end by typing in an element beginning with R instead of E. IDA takes care of the rest. This feature should not be used for beam transport cases; the relevant matrices for the reflection element are computed during the cycle operation. There may only be one REFL element and it must be the last element of the line. IDA does not check these points; it will simply do unpredictable things. Also note that the editor computes the matrix elements for individual elements but you must explicitely do an operation like M(ultiply or C(ycle to generate correct beamline properties after any editing changes.

If you have followed this tutorial so far, you have a screen which looks like:

Parameters for FODO
Number of Elements = 12 Line Length = 15.200000

	NAME	LENGTH	GRAD	RHO	ENTRANCE ANGLE	EXIT ANGLE
1	Qf	1.0000	0.100000			
2	Ds	0.2000				
3	Mag	3.0000	0.000000	11.4592	0.0000	0.0000
4	Ds	0.2000				
5	Mag	3.0000	0.000000	11.4592	0.0000	0.0000
6	Ds	0.2000				
7	Qd	1.0000	-0.100000			
8	Ds	0.2000				
9	Mag	3.0000	0.000000	11.4592	0.0000	0.0000
10	Ds	0.2000				
11	Mag	3.0000	0.000000	11.4592	0.0000	0.0000
12	Ds	0.2000				

When you run T, you should see, at the bottom of the screen, something like:

Line length = 15.2000 Bending Angle = 1.0472
Horizontal Phase = 1.4207 Vertical Phase = 0.7428

Let us use the two parameter fitting routine, F(it, to adjust the two quadrupole strengths so that our new FODO lattice has a phase shift of about 70 degrees or 1.22 radians per period. Before you type F to do this, let me describe to you what the fitting program is going to do. It is going to cycle the line, then one at a time, change each quadrupole gradient by a small increment (which you can specify) and cycle again. In this manner, it determines the sensitivity to the gradients, solves a simple two by two linear equation set and interpolates or extrapolates to the correct gradients. This is a very primitive system and it can easily go awry with horrible crashes, etc. Thus, I recommend that before you try it, save the line. Then a crash does not imply another session with the editor. Type S. When it is done, Type F. You should see a screen which looks like:

I(terate, C(hange variables, R(edo determinant, L(eave fit

CHANGE:

element parameter = number in increment of number
element parameter = number in increment of number

TO OBTAIN:

property = number at position number
property = number at position number

The underlined items are all things you may wish to adjust or change. Again, we have a screen editor type format; you will see that you can move around to each

of the underlined items with the arrow keys. Go to the first element on the top line. Type C for C(hange, then enter Qf(with return). Move one item to the right with the arrow. Type C. You will get a message:

Type code for parameter to vary

L(ength, **R**(adius, **G**(radient, **E**(ntrance angle, **X**(it angle

Typing the emphasized letter (without return) will select that parameter to vary and fill in its current value. Usually it is the gradient that we wish to vary so type G. There is nothing in IDA to prevent you from typing in nonsense. For example, if you enter the Entrance angle for a drift space, IDA will simply crash later in the iteration! You may change the incremental steps used for calculating the sensitivities; for our tutorial case the default number 0.001 will be acceptable so leave it and move the cursor to the next line. Select Qd and Gradient here also. Go to the next line and set the cursor to the left. Type C for C(hange. You should see a message:

****Enter code for desired result****

(1)Alphah,(2)Betah,(3)Psih,(4)Alphav,(5)Betav,(6)Psiv,(7)X dispersion,(8)X-prime

In this case, select a number to indicate the desired property that you wish the fitting program to try to set. Be sure to type an integer 1 through 8. IDA crashes on any other input. Type 3 (with return). Move the cursor one motion to the right. Type C. Enter the floating point number 1.22 which is our desired phase shift. Move to the right. This is the number of the point in the beam line where you want the result to be set. This is the same number as shown for each element on the usual screen or typed listing of the beam line. For the results to be at the end of the beam line, the usual case, it should be one larger than the number of elements. If you cannot remember how many elements in the line, just type C, then enter a very large integer. IDA will clip it to the end number and indicate (END). Similarly, set up the last line for Psiv at the end of the line. The entries in this dialogue are recorded in the main data block of IDA. Thus if you now type in L for L(eave, you leave the fitting routine without actually doing any of the calculations but if you come back to the fitting routine with F, you will find the parameters and variables as you left them. In fact, the results of this dialogue are included in the disc save and get files so that if you wish to play with this beam line later, you may not need to do these changes again.

Once the screen looks like what you want, type I for I(terate. IDA will ask you if you wish to Cycle or Multiply. Note that the format of this fitting routine is such that it can be used in transport systems to find focal point values, etc. In that case, we would type M here. For our tutorial, we want to select C. The calculation now proceeds. As indicated above, four cycle operations are required but it really doesn't take very long. Occasionally, one of the small increment steps will take you out of the stability range; IDA proceeds to generate absolute nonsense and set your parameters way out in left field. A good reason to have saved the line before F(it. Usually F(it works without such disasters. After an iteration, F(it gives you a little printout so you can see how well it is working. In our case, it is going to take several iterations. Just keep punching I. You will notice that for each iteration, it only does one cycle operation. What IDA is doing is to use the initial sensitivities and just redoing the extrapolation. If the initial

conditions were a long way from the desired results, this convergence may be painfully slow. Then instead of typing I type R. It now repeats the entire set of four cycle operations. As I have mentioned, this fitting routine is very primitive and if it seems to be divergent or generates nonsense, it is best to go to some other hand calculations to get the beam line about like you want it. My impression of fancier beam line programs is that they all have this problem but do not tell you about it other than (too late) a cryptic message of floating point error God knows where!

When the fitting has converged to sufficient accuracy or for any other reason you may exit F(it by typing L for L(eave.

If the IO chain is active, i.e, you have just done a G(et, S(ave, L(ist, or T(ype operation, you will notice that there is an additional item, I(ntegrals, in the command line. This option should only be exercised on lines which are essentially complete. It is a routine which outputs a number of interesting summary integrals. It is oriented about electron machines and most of the data items are only relevant to such machines. I assume you have our FODO lattice all finished. If not Get VUV so that you can see what is involved in the following discussion. If you do not see I(ntegrals in the command line, Type T, when prompted Type S, exit with return, and you should see I(ntegral. Type I. Again, you will get a screen with variables which can be set. Use the cursor arrow keys and the C(hange operation to set the energy, the rf volts, and the harmonic number. Since this program is primarily designed for electron machines, you will need to tell it some astronomically large number for the rf volts if you are thinking of a Tev proton machine! The program, later on, tries to figure the stable phase angle for the assumed electrons to compensate for the radiation loss and crashes if you do not supply enough. Sorry about that. There is a line where you can tell IDA which sextupoles to vary and what chromaticities you want. Since we didn't put any sextupoles in our FODO lattice, we will skip this exercise. When the other parameters are set type G for G(o ahead. The computations take awhile. Note that the emphasis is on electron storage rings[4] and only a few of the numbers are relevant to proton or ion machines. (Horizontal and vertical tunes, momentum compaction factor, and chromaticities). Also note that whereas in any of the computations up to this point IDA really does not care what units of length are used, the summary results of the Integrals page assume that lengths are in meters. After the computations are finished, you will see a command line of:

A(djust chromaticity, P(rint, L(eave.

P(rint gives you a hard copy of the page before you. If you have filled in legitimate sextupole names and desired chromaticities before the computations, A(djust will set the sextupoles. Since we have not done so, hitting A now would cause a crash. Notice an interesting lattice problem. Our simple FODO has positive chromaticity without sextupoles! Yes it is true. IDA's numbers agree with MAD and SYNCH for this lattice. The reason is apparently that this rather strange lattice is getting lots of its focusing from the bending. Go back to the editor, make the radius of curvature ten times larger so that there are 60 periods instead of 6, use F(it to get to the same phase shift per cell, and try the chromaticity again. This is now a more usual looking high energy machine lattice and the chromaticity is quite negative. Another exercise for the reader:

4. The radiation integrals were calculated using the formalism of Helm et.al., IEEE Trans. on Nuc. Sci. - 20 #3 (1973) pp. 900-901.

Go to the editor, put a sextupole near Qd and another near Qf. Run the Cycle operation again. Then go to the integrals page, specify the sextupoles with the cursor and change options. Similarly select the wanted chromaticities as 0 in both planes, then, after the computation, let IDA adjust the sextupoles.

Some comments about the chromaticity calculation are in order. I have used the formulation given by Bassetti[5] and have verified, analytically, these derivations. These are tedious to program, however, and there are many places where the program can go astray, particularly at the ends of dipoles, in cases in which the curvature focusing nearly cancels the gradient (n-1), cases in which the gradient is near zero, etc. In IDA indeterminate cases are solved by expanding indeterminate forms as polynomials where such action seems to be needed. The answers appear to be reasonable in cases that have been examined. All of the various terms in Bassetti's formulae have been identified in the code and each checked with hand calculations in at least two lattices. Most numbers calculated by IDA agree well with results from SYNCH, MAD, and PATRICIA. This cannot be said for the chromaticity. For starters, MAD, SYNCH, and PATRICIA do not agree with each other so I do not know what to compare IDA with! The table below indicates the nature of this problem. The two lattices indicated are the VUV lattice which we used in the early part of our tutorial (but with different tunes) and BEN, a lattice for a compact x-ray lithography ring.

	VUV-ξ_x	VUV-ξ_y	Ben ξ_x	Ben ξ_y
PATRICIA	-3.47	-6.04	0.18	-2.53
SYNCH	-4.15	-5.43	-1.46	-1.47
MAD	-4.08	-5.45	-1.61	-2.51
IDA	-4.19	-4.88	-1.61	-2.65

Both of these lattices are rather testy for the chromaticity calculation. The VUV has a large (45°) bending angle in each magnet and sizeable (22.5°) edge focusing. Ben has two 180° dipoles and has significant gradient focusing in the dipoles. The chromaticity variations between programs are much less when the usual high energy machine lattices are considered.

Two other items are on the main command line. If you have a Hercules card installed, P(lot should work and generate rather obvious plots. The screen shows some tabular information and plots of B, xp, and the phase shifts. In the tabular information, the λc value is only guaranteed if there is only one type of bending magnet present. Finally, Q(uit has the obvious function of leaving IDA and returning to DOS.

I wish to acknowledge the original contribution of Max Cornacchia in generating the BASIC code for the first version of this program. In the present version, Gaetano Vignola and Jim Murphy provided useful tips on computing the radiation integrals and the chromaticity. Chas Archie of IBM offered useful suggestions about the graphics.

5. A Simplified Derivation of Chromaticity Formulae, Mario Bassetti, LEP Note 504, CERN, Geneva, Switzerland, June 1984 (unpublished).

APPENDIX

Auxiliary Programs using the IDA data blocks

The data blocks of IDA are structured in a manner that makes it simple for the PASCAL programmer to write other programs using the IDA data. In the first part of this appendix, I describe three such programs. Later, I describe in more detail the data formats for other possible applications.

SYNCH

To use this program, be sure you have saved the line; then type Q to leave IDA. Type SYNCH. You will see the same prompt that IDA uses to ask for a beamline name during the G(et operation. Type in your beamline name. Then you will see on your screen a SYNCH input deck for this data set. Use the DOS operation DIR and look at your directory. You will find a file beamlinename.SYN. Transfer this file to your favorite mainframe and use it as input to SYNCH.

Notice some peculiarities of the SYNCH deck that I have created with this system. It generates a sequence of SYNCH operations appropriate to electron machines. You may wish to change these using your mainframe editor. All letters used in element names have been converted to upper case. There could thus be elements which were distinct in IDA which will now be trouble for this SYNCH conversion. All magnetic parameters are set to scale to a single Brho value shown in the second line of the SYNCH deck; the value is derived from the energy entered in IDA on the I(ntegrals page.

This program has been tried on a number of lattices and appears to work well.

MAD

The concept of this program is similar to that of SYNCH described above. It generates a beamlinename.MAD file. There is no energy variable this time. The same comments about upper and lower case of beamline elements applies here. This program does one thing with sextupoles which the user may not wish. If the sextupole has zero length, then the strength is divided by the length to get the K2 as defined by MAD and a crash results. I avoid this by looking for zero length and using a Multipole lens with appropriate K2L for this case.

PATRICIA

This program follows the same format as the two above. It generates a file called beamlinename.PAT. A rather arbitrary set of PATRICIA switches are set; no tracking is included. I suggest that this file be tried and, if everything looks all right, use your mainframe editor to activate the switches you want for tracking. Note that PATRICIA crashes if there is not an explicit SF and SD in the lattice to use to adjust the chromaticity. This program is limited to about 16-18 non-drift-space elements; the uncertainty depends on whether or not the beam line begins or ends with drifts. The limitation occurs because I did not expend the patience and time to make the PATRICIA format beam line extend over more than one line.

To write other auxiliary programs:

The data for IDA are conveniently arranged for the PASCAL programmer to write other auxiliary programs. Look for example, at the first few lines of SYNCH.PAS. You will see the include file directives:

```
{$I dblk}
{$I fetch}
```

then the first executable statements:

```
  begin
    fetch;
```

dblk is a file giving all of the variable declarations and allocations used by IDA. fetch is a procedure (used by the G(et operation in IDA) which loads the information from beamline.DAT into the data block described by dblk. The programmer is free to proceed from this point to whatever he chooses. The listing of dblk.pas follows:

```
{DBLK.PAS FILE}
{This is data block for all of lattice system}
{Do not complile; it gets brought in through include statements}
   TYPE MATRIX=ARRAY[1..3,1..3] OF REAL;
        LINE=ARRAY[1..50] OF MATRIX;
        PARAM=ARRAY[1..50] OF REAL;
        INDEX=ARRAY[1..50] OF INTEGER;
        ELNAME=ARRAY[1..50] OF STRING[6];
        TWOMATRIX=ARRAY[1..2,1..2] OF REAL;
        LINEV=ARRAY[1..50] OF TWOMATRIX;
        KEYTEST=SET OF CHAR;
        Intpair=Array[1..2] of integer;
        Realpair=Array[1..2] of real;
        Charpair=Array[1..2] of char;
        CLASS=(HEADER,Supdata,MATDATA,LAST);
        MESS=RECORD
        CASE KIND: CLASS OF HEADER:
        (NAMELN:STRING[8];NOOFPART:INTEGER;LENGTH,XBA,XPH,XPV:REAL);
        Supdata:(rs1,rs2:integer;renergy,rharm,rrfv,rwantx,rwanty:real;
        rinx,rres,rloc:intpair;rdelx,rwant:realpair;rpx:charpair);
               MATDATA:(PARTNM:STRING[6];RLENGTH,RRHO,RGRAD,
                      ANG1,ANG2,SAH,SBH,SPSH,SAV,SBV,SPSV,SX,SXP:REAL;
                      THRE3:MATRIX;TWO2:TWOMATRIX);
               LAST:(RAH,RBH,RPSH,RAV,RBV,RPSV,RX,RXP:REAL) END;
   VAR  EL:LINE;
        EY:LINEV;
        LE,CURV,PSIH,PSIV,ANGENT,ANGEXIT,K,AH,BH,XP,XS,AV,BV:PARAM;
        CH,CV,TE:TWOMATRIX;
        N:ELNAME;
        SETOFOPS,LEGAL:KEYTEST;
        LINE1,LINE2,COLNO,LINENO,I,AN,
            s1,s2:INTEGER;
        RHO,BA,PH,PV,RC,RS,VC,DT,AA,BB,TT,KX,LT,SKX,AR,X1,X2,
            energy,harm,rfv,wantx,wanty:REAL;
        RTEST:BOOLEAN;
        FID:String[12];
        Inx,res,loc:Intpair;
        delx,want,Z,Parval:Realpair;
        px:Charpair;
        LINENAME,ENAME:STRING[8];
        SELECT,CHS:CHAR;
        GLASS:CLASS;
        MASS:MESS;
        LDATA:FILE OF MESS;
        HARDCPY:TEXT;
        active:file;
```

Many of the variables in this list are for temporary use by IDA. I will identify the ones most likely to be of interest to the programmer. The horizontal motion is described by a sequence of 3x3 matrices (to include dp/p in the usual manner). EL is the array of 3x3 matrices for this motion. For example EL[1,2][13] is the element in the first row and second column of the 13th matrix. Similary EY is an array of 2x2 matrices to describe the vertical motion. All of the beam element parameters, focusing functions, etc. are single dimensional arrays, i.e. user defined type PARAM. These are (see third line of variable declaration):

LE	length
CURV	curvature, i.e. reciprocal of radius of curvature
PSIH	accumulated horizontal phase shift
PSIV	vertical phase shift
ANGENT	entrance angle for bending magnets
ANGEXIT	exit angle for bending magnets
K	gradient
AH	horizontal alpha
BH	horizontal beta
XP	dispersion
XS	derivative of XP
AV	vertical alpha
BV	vertical beta

The names of elements are contained in the array N of six character long strings. The integer I contains the number of elements in the beamline. The name of the line is in the 8 character string LINENAME. Some summary values are the real numbers:

BA	total bending angle
PH	total horizontal phase shift
PV	total vertical phase shift
LT	beamline length
energy	energy
rfv	radio frequency volts
harm	harmonic number

The get and save files also include the parameters selected in the two parameter fitting dialogue and the sextupole selections for the chromaticity fitting. These are probably not of interest to other programs. Other numbers in the dblk list are temporary in IDA and are not included in the get and save files.

If you wish for some reason to recompile the IDA system, use the following sequences:

1. IDA.COM. Copy the TURBO system into your working directory or onto your working disc. Type W to select the work file and enter IDA. Type O to set compiler options. Select C for COM file. Type O to set memory allocation and enter C00 (paragraphs). Type D to allocate data space and enter 400. These space allocations are required for the plotting package. With these allocations, the IDA system should fit into a 256k machine. Type Q to leave compiler options menu and type C. If the compilation is done with TURBO-87 for use of the 8087 math coprocessor, then set the memory allocation to D00 and the data space to 500. I'm not sure that the resultant code will fit into a 256k machine

2. IO.CHN. Type W to select new workfile and enter IO. Type O to change compiler options. Select H for chain file, then Q to leave options. Type C.

3. EDIT.CHN Follow same procedure as IO but with EDIT selected after equesting a new work file.

4. LATTICE.CHN Follow the same procedure as the two previous chains.

5. PLOT.CHN This chain requires considerably more work. The graphics chain uses procedures defined in the Borland Turbo Graphix Toolbox.[6] If you have a hard disc, copy all of the IDA PAS files, the TURBO system, and the Toolbox system into a temporary directory. Run TGINST to install the toolbox routines for your system. The list of graphics cards supported and the details of this procedure are described in the Toolbox manual. Then run the TURBO system and compile the PLOT.CHN chain exactly as described above for the IO chain. You will only need the following files in your operating directory to use the PLOT chain:
PLOT.CHN
4X6.FON
8X8.FON
14X9.FON
ERROR.MSG.
You may copy these into the working directory and then clean out and remove the temporary directory used during compilation. Working with two floppies, you would put all of the IDA system and the TURBO system in drive A. Put the toolbox disc in B. Run B:TGINST. Using the TURBO editor put b: in front of the file names typedef, graphix, kernal, windows, polygon, and axis in the include directives in the first few lines of the PLOT.PAS file on disc A. Then compile PLOT.PAS into PLOT.CHN. Copy 4X6.FON, 8X8.FON, and ERROR.MSG from B to A and you are ready to go.

The PASCAL files on the disc are:

IDA	Contains initialization steps
LATTICE	Contains computations multiply, cycle, and two parameter fitting. Compile as a chain file.
IO	Contains all input-output operations. Compile as a chain file.
EDIT	The beam line editor. Compile as a chain file
PLOT	Plotting package. Compile as a chain file.
ELEMENTS	Computes matrix elements from device parameters. Do not compile. This file gets pulled in as an include file as needed.
FITTEST	Two parameter fitting routines. An include file for LATTICE.
PRELIM	Some mathematical and service routines. An include file for all chains
DBLK	The main data block. An include file for all chains
GRAB	A simple routine which fetches one character from the keyboard without waiting for carriage return. Lower case letters are converted to upper case. Included in all chains
FETCH	A procedure that fetches a beamline from disc. Used by the G(et operation. An include file in IO chain
INTEGRAL	Computation of summary integrals. It is an include file in IO chain.

6. The TURBO GRAPHIX TOOLBOX is a product of Borland International.

If you are compiling this system on other configurations, you may need to change some (or all) of the following:
GRAB.PAS. Rewrite for your system. Be sure the arrow keys codes are transmitted. IDA uses 12(up), 20(down), 17(right), and 15(left). If these are not the codes coming up through your GRAB routine, either put a translation into GRAB or search through the source codes for places where IDA is in screen edit mode and replace the code numbers there.

PRELIM.PAS A character with code 219 is used as the reverse print blank for the prompt for floating point input in the function infloat. You may need to do something different here. This usage also occurs in the procedure initit in FITTEST and in procedure inname of INTEGRAL.

If your system cannot use the TURBO GRAPHIX TOOLBOX, you should either eliminate the P(lot option in the case statement at the end of each chain or create a dummy PLOT.CHN which calls back LATTICE.CHN to avoid crashes when one accidently hits P.

You may also need to change the characters used to print form-feed in the List and Type procedures in the IO chain.