TREETIPS

A Publication of Treehouse Software Inc.

Issue: 1 | Sewickley, Pennsylvania | October 1, 1987

Treehouse Software Begins Newsletter

Most of you are familiar with Treehouse Software, and its commitment to making the ADABAS and NATURAL environment more efficient and effective. Treehouse Software produces TRIM® (a performance monitor) and provides consulting, classes and general support for ADABAS and NATURAL. In an effort to expand on this support, Treehouse Software is starting a newsletter. The purpose of this newsletter will be to share information that can help others in their everyday environment.

The newsletter will have the following items:

NATURAL version 1.2 article(s)
Since many of you may not have version 2.0 for quite some time, it is important that your current environment perform as well as possible. Furthermore, we hope 1.2 articles will make it clear what is effective coding, and identify problem areas that can be corrected before the version 2.0 conversion. Any problems, questions, or general information you would like to share with others on 1.2 can be done through this newsletter.

NATURAL version 2.0 article(s)
Obviously, this is the hottest topic around these days. Everyone's talking about it. Some people have it. Some people have installed it. Most of you are anxious to know what the advantages and disadvantages are. We would like you to be able to share with others tips, quirks, problems, etc. All of us have different 'creative' environments and if there are better ways of doing things, or problems to be found, somebody will find them. For the most part, regional and local users groups are the place to share information, but we'd really like to see a more 'worldwide' effort. That's one of the reasons that we are starting this newsletter.

General System Article(s)
This first newsletter will cover CLOG problems and solutions as well as a general explanation of what is going on when logging occurs.

Generally, we would like to have database (especially ADABAS version 5), TP, operating system, security, performance, auditing and similar type articles. They do not have to be long, in fact short tips are just as helpful. Furthermore, if you don't want to write it up, call us, and if it's something that will benefit everyone, we'll write it up from your experience.

Letters to the Editor
This is your area to talk about Treehouse Software, SAG or items of general interest. Please feel free to send them along. We will respond to all letters, if not in this newsletter, then with a personal letter answering your questions or statements. For the most part, we will try to answer all letters in this paper, but sometimes time and space may not permit. We do hope that you will have things to contribute.

Current Breezes
This is written by Treehouse's President and TRIM creator, George Szakach. George has plenty of plans, ideas and general future goals. This is where he will sit down and tell you about what's coming in the future, as well as any other items of interest he may want to share. This not only gives George a chance to tell you his priorities, it also gives him a chance to make you appreciate where Treehouse and TRIM are going, and when they will be there.

Calendar of Events
This is for everybody. If you have an event that you want to share with us, please tell us, and we'll pass it on. This is a good place to share local and regional information with others. You may be interested in attending other regional and local meetings. If your event is covering topics that are of interest, we would be willing to publish an agenda.

Sewickley in the News
Not only is Sewickley the hometown of Treehouse and its executives, it's also a great place to live. Some of you will be traveling to garden spots around the world, and we'd like to share this gem with you.

Table of Contents

(Continued on Page 14)

Letters to the Editor

We Look Forward To Hearing From You.

Current Breezes

by George

Let me take this opportunity to thank our 200 customers who have helped to make Treehouse Software a success. As we enter our sixth year, our increasing company size, international marketing, and scope of products and services necessitate keeping you informed. A newsletter is one method of doing this.

Our marketing and support will be stronger with our recent employee additions. This will free the developers to better meet your expectations on improvements to TRIM and TRUST, and to forge ahead with a few new products. By the next newsletter, we hope to have one or two new product developments announced. We have to be careful about announcing new products before completion (competitors you know). However, in response to your change enhancement requests, we are prepared to mention a few new improvements in TRIM version 3.2.

PROTECTION LOG - AUDITING IMPROVEMENTS:

TRIM processes one or more PLOGs in one pass to produce multiple reports. Files to be audited can have selected fields extracted from the PLOG's compressed data for display or output. Users have asked for a TRIM process to identify the changed fields. This is a big improvement over sight comparison of pages of displays of decompressed PLOG fields. The next logical extension (version 4.0?) is to INCLUDE/EXCLUDE records based upon field values in the PLOG. (For example, include only records with the salary field > 20,000.)

USER-EXIT-1 and USER-EXIT-4 IMPROVEMENTS:

Dynamic control and security through User-Exit-1 will be enhanced by allowing for pre-set parameters to Disallow commands, Lock files, and do password checking or substitution. There will be no need to 'communicate' to the user-exit, as the user-exit will load its own parameters on the first call. Furthermore, the user-exit will load different parameters at different times. For example, you may Disallow S2s on certain files during prime time, while allowing all S2s at other times, without DBA intervention.

User-Exit-4 will operate similarly. LOG/NOLOG parameters, PRESUM settings, and Real-time Monitor options for tallying, sorting, displaying, etc., can be different for different time periods. There is no need to keep certain Real-time Monitor statistics when the DBA has gone home. So, you will probably set up four different sets of parameters; for active prime time, night time, early morning slack usage time, and for weekends. User-Exit-4 knows what time it is and will act accordingly.

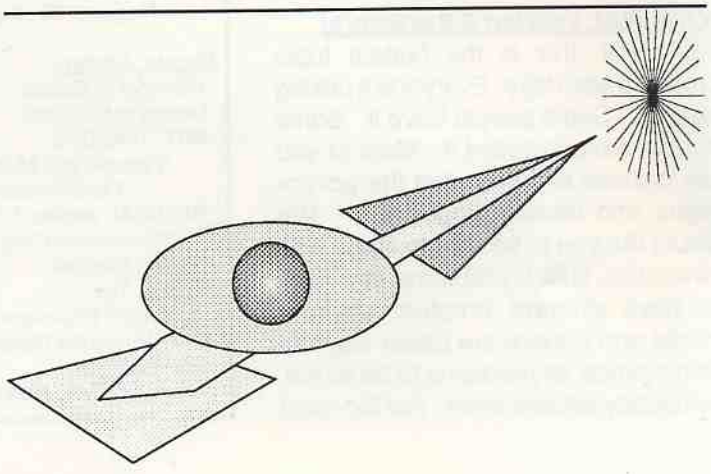
HISTORICAL FACILITY:

TRIM version 3.2 can load our User-Exit-4 summarized statistics (PRESUM) onto an ADABAS file for reporting hourly, daily, monthly, and yearly historical statistics. Trend analysis can easily be observed for ADABAS calls, Durations, I/Os, and CPU times for selected Files, Jobs, Users, Programs, etc. An effective method of combining historical records and purging old data insures the efficiency of this ADABAS/NATURAL based historical facility.

INTERNATIONAL MARKETING:

We are now represented internationally in South America, Germany, France, Netherlands, Austria, Scandanavia, Japan, and South Africa. For more information please contact our Sewickley office.

We hope you enjoy the mix of technical, marketing, and "light" topics we present in this newsletter. You can see from this newsletter that our manuals, documentation and class notes will take on a new Mac look. Be sure to pass this newsletter around or call for extra copies. If we don't have your name and address, drop us a note or call us. We'll add you to our mailing list. Once again I want to thank you for your support over the years, and I look forward to many new opportunities to share our products and expertise with you.



NATURAL TIPS: Periodic and Multiple Value Field Processing

Multiple Value Field (MU) and Periodic Group (PE) processing has been a consistent problem in ADABAS shops that program in NATURAL. Although not technically difficult, MU and PE processing can be confusing, and the documentation available is unclear at best. Many shops consistently use GET SAMEs to access multiple occurrences of MUs and PEs. This is not surprising since GET SAME is very easy to code and its inefficiencies haven't been well-published. As time goes on, and program problems and efficiencies can be identified using a performance monitor such as TRIM , less and less of this type of inefficient coding will proliferate. For those of you processing MUs and PEs with GET SAMEs, you may have found that to read a record and load 30 occurrences of an MU or PE to working storage would take more than 30 calls to the data base instead of the desired 1. For this reason, and because some shops have problems with indexing fields, it seems timely to discuss these areas. Although NATURAL version 2 does away with OBTAINs and MOVE INDEXED in Structured Mode, OBTAIN will still be used in report mode. MOVE INDEXED will be changed to MOVEs, so this concept is one that should be examined. For many shops, NATURAL version 2 is not in the immediate future, but inefficient coding practices are an everyday fact of life. It is for these people that this article is written.

MULTIPLE VALUE FIELDS (MUs):

A Multiple Value Field (MU) is a special field with values repeated multiple times. In other words, it contains more than one occurrence of a field. This information usually consists of a list or string of related data items which is stored under one field name. For example, take the record CLASS-DATA. This record could be defined:

```
G     1     AA    CLASS-DATA
      2     AB    CLASS-NAME          (A10)
      2     AC    CLASS-NUMBER        (N5)
      2     AD    CLASS-ROOM          (A5)
M     2     AE    CLASS-ATTENDEES     (A30)
```

If this record were written out, it might look like this:

CLASS-NAME	CLASS-NUMBER	CLASS-ROOM	CLASS ATTENDEES
CS C1974	78732	A4750	Smith, Robert Jones, Mary Doe, John Williams, Mike

From this example, you will see that there are 4 people (4 occurrences) in CLASS-ATTENDEES. Any MU cannot hold more than 191 occurrences. If an occurrence of CLASS-ATTENDEES is deleted (ie: in the case where CLASS-ATTENDEES is a null suppressed field (which most MU fields should be), an occurrence is updated to a null value), all occurrences would be moved up. In this case, the number of occurrences would decrease from 4 to 3.

ACCESSING AND OBTAINING MUs:

To reference MUs, they must be indexed. NATURAL understands a fixed number as the index. Therefore, it would be possible to code:

```
MOVE CLASS-ATTENDEES (1)   TO #CLASS-ATTENDEES1
MOVE CLASS-ATTENDEES (2)   TO #CLASS-ATTENDEES2
    etc.
```

for as many occurrences as you may have. Now this type of coding may seem easy for 1 or 2 members of a MU, but when there are 100 or so, it would be time and space consuming. Therefore, array processing is desired. In true array processing, you have the ability to specify a variable index, and have the proper occurrence be accessed. Unfortunately, true array processing is not available in NATURAL version 1.2. However, there is a method of table handling. To process an MU, you must first read all occurrences that will be accessed. This should be done using OBTAIN, not GET or GET SAME. Therefore, if there is never going to be more than 30 class attendees, you could code the following:

```
OBTAIN CLASS-ATTENDEES (1-30)
```

This statement creates a table of 30 occurrences of CLASS-ATTENDEES. A MOVE INDEXED operation could be used to access them. You must be careful in determining whether or not you will be indexing a file field or a user-defined field. When moving a user-defined field you would code:

```
MOVE INDEXED #TABLE-VALUE1 <#INDEX>
    TO #HOLD-TABLE-VALUE
```

When moving an MU file field that has been obtained, you would code:

```
MOVE INDEXED CLASS-ATTENDEES (1) <#INDEX>
    TO #HOLD-ATTENDEE
```

The (1) indicates that it will start at the first occurrence in the table. This is frequently confusing and frustrating to new programmers or programmers that do not frequently work with MUs. It would be possible to start at the 2nd or 10th or whatever number of the MU by specifying that number in parenthesis, but usually that is not a desirable option.

Therefore, instead of moves using individual hard-coded indices, the following could be coded:

(Continued on Page 4)

NATURAL TIPS: PE & MU Field Processing

(Continued from Page 3)

```
MOVE INDEXED CLASS-ATTENDEES (1) <#INDEX>
    TO #CLASS-ATTENDEE1 <#INDEX>
```

A loop containing this statement would move the corresponding class-attendees to the fields in the program (ie: map display fields). However, remember that #CLASS-ATTENDEE1 through #CLASS-ATTENDEE30 must be contiguous. In other words, all these fields must be defined in working storage together. Like:

```
RESET       #CLASS-ATTENDEE1 (A30)
            #CLASS-ATTENDEE2(A30)
            .
            .
            .
            #CLASS-ATTENDEE29(A30)
            #CLASS-ATTENDEE30(A30)
```

or slightly safer might be:

```
RESET       #CLASS-ATTENDEES1 (A240)
            #CLASS-ATTENDEES2 (A240)
            #CLASS-ATTENDEES3 (A240)
            #CLASS-ATTENDEES4 (A180)
REDEFINE    #CLASS-ATTENDEES1
                  (##CLASS-ATTENDEE1 (A30)
                   ##CLASS-ATTENDEE2 (A30)
                   .....          )
            etc.
```

The latter statement is more likely to assure that the space will remain contiguous, but it would be up to the programmer's discretion.

Having obtained the necessary members of the MU, they could also be updated using the obtained group. For example, if the fields had been modified or you wanted to update a record from information on a screen, you could do the following:

```
0200   GET IN FILE CLASS-INFORMATION *ISN (0050)
0210     OBTAIN CLASS-ATTENDEES (1-30)
0220     FOR #I 1 THRU 30
0230           MOVE INDEXED ##CLASS-ATTENDEE1 <#I>
0240                TO CLASS-ATTENDEE (1) <#I>
0250     CLOSE LOOP (0220)
0260     UPDATE SAME (0200)
0270   END TRANSACTION
```

The above statement would allow the data in the ##CLASS-ATTENDEE table to be moved to the file fields, and updated. It is easy to see how much time and space can be saved when you do this type of programming. Using the GET allows the record to be reread so that it did not have to be placed on hold during the screen processing, thereby preventing a possible time out error (NAT 3009).

Please note that if you do a GET on the file, you are rereading the record. If you are certain that only one person through only one application could be updating this record, then it would be acceptable to perform the GET without checking the record. However, it is often the case that someone else could be updating the record. To prevent any data errors, you should save the original record information and verify that no changes have occurred since the record was last accessed before updating the record with the changed information. If changes have been made, you may decide to prompt the user with the new information, or show them the changed information and request further action.

In accessing MU fields, C* is available. C* is a standard system function that allows you to find the number of occurrences in a field. The format is C*field-name, where field-name is the name of the MU. In the example used, C*CLASS-ATTENDEES would give us 4. This could allow us to add the next occurrence of the MU using the following code:

```
MOVE C*CLASS-ATTENDEES   TO #INDEX (N3)
ADD 1                    TO #CURRENT-NUMBER
MOVE #NEW-NAME           TO CLASS-ATTENDEES (#INDEX)
UPDATE SAME
```

C* is also useful as a control variable in a processing loop. In the above example, we assumed that no more than 30 occurrences were on the file. However, MOVE INDEXED is not cheap, and if we only have one or 2 occurrences on the file, we really don't want to process 30. Therefore, you could code:

```
OBTAIN CLASS-ATTENDEES (1-30)
FOR #I 1 THRU C*CLASS-ATTENDEES
```

This would only access the occurrences that were found. You will need to consider what will happen if no occurrences were found (in this case, it would not enter the FOR loop). As well as what to do if you exceed 30 occurrences. Most people feel sure their data is good, but it never hurts to check its validity. There are three things that you may want to do if you exceed 30. The first case would be to ignore the rest. In that case you could code the following after the above lines:

```
IF #I > 30
   ESCAPE
```

This will prevent accessing those fields. This is rarely the desired approach. In fact, in most cases, you will want to issue an error of some sort. In this case, you could code:

```
IF #I > 30
   PERFORM INDEX-ERROR
```

and subroutine INDEX-ERROR may print 'More occurrences on file' or tell them to call the analyst. The final case, and the most common is where you want to handle the additional occurrences. In this instance, you must remember that you have only obtained 30 occurrences. You cannot conditionally obtain more occurrences, so you will have to

(Continued on Page 5)

NATURAL TIPS: PE & MU Field Processing

(Continued from Page 4)

read the record again to access the additional occurrences. In this case you would code the following:

```
0300      IF #I > 30
0310        DO
0320        GET SAME (0010)
0330          MOVE CLASS-ATTENDEES (0320/#I) TO #FIELD
0340        DOEND  /*(0310)
```

In this example, you are rereading the record to get the additional fields. This is not efficient, but it saves having to OBTAIN more records than you normally process. If 99% of all your records have 30 or fewer students, or you only expect this to happen in an unusual circumstance, then the above logic would be better than always obtaining all the occurrences possible. As always, the more the programmer knows the data, the more efficient the program will be.

To DISPLAY MU fields, there are various options available. For example:

DISPLAY CLASS-ATTENDEES

will DISPLAY only the first occurrence of the MU (unless your shop has specified more as the default). It is the same as saying:

DISPLAY CLASS-ATTENDEES (1)

So, a specific number can be used to specify the occurrence of the field you would like to DISPLAY. To DISPLAY a range you could code:

DISPLAY (ES=ON) CLASS-ATTENDEES (1-30)

In the above example, up to 30 CLASS-ATTENDEES could be displayed, but the ES=ON (Empty Line Suppression) causes only the occurrences with values in them to be displayed.

PERIODIC GROUP PROCESSING:

Periodic Groups are very similar to MUs. Instead of having a single field repeated, a PE can have 1 or more fields repeated. PEs cannot have more than 99 occurrences. PE occurrences do not move up if a middle occurrence is deleted. So for fields where occurrence is important such as monthly totals, PEs with a single field would be used instead of MUs. (MUs that are not null suppressed would also work but this is not recommended.)

In the example that we used above, you may have decided that class name isn't enough information for each student. In fact, class grade and major might also be desired. Therefore, the above file description would be changed to:

```
G     1     AA    CLASS-DATA
      2     AB    CLASS-NAME          (A10)
      2     AC    CLASS-NUMBER        (N5)
      2     AD    CLASS-ROOM          (A5)
P     1     BA    CLASS-INFORMATION
      2     BB    CLASS-ATTENDEES     (A30)
      2     BC    CLASS-GRADE         (A2)
      2     BD    MAJOR               (A5)
```

Displaying PEs is much the same as MUs. However, you can display the group or individual fields. For example:

DISPLAY CLASS-NAME CLASS-INFORMATION (1-4)

would result in:

	CLASS-INFORMATION		
CLASS-NAME	CLASS ATTENDEES	CLASS-GRADE	MAJOR
CSC1974	Smith, Robert	A	CS
	Jones, Mary	B	CS
	Doe, John	F	ENG
	Williams, Mike	C	CS

Alternatively, each field could be specified:

```
DISPLAY CLASS-NAME                CLASS-ATTENDEES (1-4)
        CLASS-GRADE (1-4)         MAJOR (1-4)
```

and the same information would be displayed without the CLASS-INFORMATION header.

PEs also differ in the way you can obtain them. Like the DISPLAY, you can OBTAIN each field, or the group. Usually the group is preferred. However, occasionally you want to redefine a field in the group, or do not want to access all the fields. In this case, you would only OBTAIN the fields you desire. To OBTAIN the group you would code:

OBTAIN CLASS-INFORMATION (1-30)

or if you only wanted CLASS-NAME:

OBTAIN CLASS-NAME (1-30)

You cannot move the group level to a field defined like the group. It is possible to do this in many other languages. So PE processing may seem cumbersome to people not used to this process. Instead you have to move each member of the PE individually. This requires a slightly different storage structure. There are two common methods. One is to define the entire group, repeated in storage. This is the most common method for non-NATURAL programmers because it seems intuitively logical. In this case, you would:

```
RESET    #HOLD-CLASS-INFORMATION  (A37)
REDEFINE #HOLD-CLASS-INFORMATION
                    (##HC-NAME (A30)
                     ##HC-GRADE (A2)
                     ##HC-MAJOR (A5))
```

(Continued on Page 6)

FEATURE: Treehouse Software Attends Regional Meetings

Over the past 18 months, Treehouse Software has had the opportunity to attend several regional meetings. We try to do something special at each regional meeting, not just try to sell you our products and services. In 1986, we presented tutorials on NATURAL performance topics at many regional meetings. In 1987, we have been sponsoring presentations on NATURAL version 2. These have been well received. For example, the attendance at the recent regional meeting in Washington DC was more than double their normal attendance.

Other regional meetings we have attended were in:

Orlando	Indianapolis
New Orleans	Austin
Philadelphia	Rochester
Boston	Denver
Seattle	Santa Barbara
Sun Valley	Columbia
Montreal	Calgary
Sacramento	St. Louis
Chicago	

Furthermore, we played a significant part in the University/ SAG Business and Interest Group (BIG) International Conference held in Austin last November. We demonstrated TRIM on an Apple MacIntosh, linked to a nearby ADABAS site. TRIM ran uninterrupted for several days, monitoring six databases that received 15 million calls per day. It was fascinating watching a MAC Mouse chase a huge mainframe database.

If we have not visited your area recently and you wish to hear more about TRIM, or some other ADABAS/NATURAL topic, just let us know as well as your regional representative.

NATURAL TIPS: PE & MU Field Processing

(Continued from Page 5)

```
RESET     #CLASS-INFO1 (A222) #CLASS-INFO2 (A222)
          #CLASS-INFO3 (A222) #CLASS-INFO4 (A222)
          #CLASS-INFO5 (A222)
REDEFINE  #CLASS-INFO1 (##CLASS-INFO (A37))
```

Using the above structure, you would code after the OBTAIN:

```
FOR #I 1 THRU C*CLASS-INFORMATION
  MOVE INDEXED CLASS-NAME (1) <#I>    TO ##HC-NAME
  MOVE INDEXED CLASS-GRADE (1) <#I>   TO ##HC-GRADE
  MOVE INDEXED MAJOR (1) <#I>         TO ##HC-MAJOR
  MOVE INDEXED #HOLD-CLASS-INFORMATION
                                      TO ##CLASS-INFO <#I>
CLOSE LOOP
```

As you can see this is a bit more cumbersome than MU processing, but not really any more difficult. Note that C* uses the group name of the field. Also note that the field #HOLD-CLASS-INFORMATION is not really needed for processing. In fact, even though a screen or report may be in the order of the PE, you will always be specifying the individual fields. A slightly preferred way of defining your working storage and indexing would be:

```
RESET     #CLASS-NAMES1 (A240)   #CLASS-NAMES2 (A240)
          #CLASS-NAMES3 (A240)   #CLASS-NAMES4 (A180)
*
          #CLASS-GRADES (A60)
          #CLASS-MAJOR  (A150)
*
REDEFINE #CLASS-NAMES1           (##CLASS-NAME1 (A30)
          ##CLASS-NAME2 (A30)    ##CLASS-NAME3 (A30)
          ##CLASS-NAME4 (A30)    ##CLASS-NAME5 (A30)
          ##CLASS-NAME6 (A30)    ##CLASS-NAME7 (A30)
          ##CLASS-NAME7 (A30)    ##CLASS-NAME8 (A30))
REDEFINE #CLASS-NAMES2           (.....                )
REDEFINE #CLASS-GRADES           (##CLASS-GRADE1 (A2)
                                      .....           )
REDEFINE #CLASS-MAJORS           (##CLASS-MAJOR1 (A5)
                                      .....           )
```

With this structure, you could change the code to be:

```
FOR #I 1 THRU C*CLASS-INFORMATION
  MOVE INDEXED CLASS-NAME (1) <#I>   TO ##CLASS-NAME1 <#I>
  MOVE INDEXED CLASS-GRADE (1) <#I>  TO ##CLASS-GRADE1 <#I>
  MOVE INDEXED MAJOR (1) <#I>        TO ##CLASS-MAJOR1 <#I>
CLOSE LOOP
```

Obviously both will work, but the latter is slightly preferred. Of course you will want to do all the error-checking and additional occurrence checks that you did for the MU fields.

NATURAL v2 TIPS: Conversion Considerations

Many of you are anticipating the conversion to NATURAL version 2. A few of you are excited and happy about conversion, others are dreading the task involved. Hopefully, for those of you dreading the task, this article might make you more excited. For those of you ready to jump in, perhaps this article will prepare you for possible pitfalls.

As most of you know by now, there are two modes in NATURAL version 2, Structured Mode and Report Mode. Report Mode is what we currently call NATURAL with a few extra capabilities. Structured Mode, in many ways looks like Report Mode, and certainly the verbs are the same. Some of the statement syntax have changed producing a more structured looking code. Furthermore, all loops in Structured Mode must be explicitly closed, data must be defined in a DEFINE DATA section of the program and database access is done through local data User-views instead of DDMs. This User-view 'feature', for many people, is the biggest drawback to Structured Mode. Since OBTAIN and MOVE INDEXED are gone in Structured Mode, the method of handling MUs and PEs must be done by defining them as arrays within the User-view, and then coding normal array access. At the time of this writing, large MUs and PEs were causing problems with conversion to Structured Mode.

So, one of the considerations that you should be making is whether you want to make the easier conversion to Report Mode or the more significant conversion to Structured Mode. Currently, both modes are supported, and both modes have all the new capabilities. The most common question asked is what are the benefits of Structured Mode over Report Mode, and in some cases, why even consider Structured Mode. This is difficult. Obviously, Structured Mode programming looks cleaner. Comparing the two:

Report Mode:

```
IF #FIELDA > #FIELDB
   DO
   WRITE 'Field A is larger'
   RESET #FIELDA #FIELDB
   DOEND
ELSE
   DO
   WRITE 'Field B is larger'
   ESCAPE
   DOEND
```

Structured Mode:

```
IF #FIELDA > #FIELDB
   WRITE 'Field A is larger'
   RESET #FIELDA #FIELDB
ELSE
   WRITE 'Field B is larger'
   ESCAPE BOTTOM
END-IF
```

Not very many people liked the DO DOEND processing, and most felt uncomfortable with the spacing/indention. Using the END-IF permits clean, structured looking code. Furthermore, a requirement of getting rid of many peoples' nemesis MOVE INDEXED and OBTAIN might make Structured Mode attractive. More importantly, it seems very unlikely that SAG would develop a language that won't eventually be preferred, and Report Mode ultimately may be discouraged or not supported.

So what's the conclusion? Well, the best of all worlds might be a Report Mode conversion of all existing programs, with all new or major changes to programs being done in Structured Mode. In fact, it is expected that a majority of people will choose this approach. At some later date, you may decide you prefer one mode over the other, and make all your programs be in that mode to provide consistency and uniformity. But now, it is really too early to make this decision, and until the full capabilities and drawbacks have been determined, it would be best to feel comfortable with both modes.
Since NATURAL version 1.2 object modules are totally compatible and will run with NATURAL version 2 programs, some people may decide to keep 1.2 running for changes and modifications until a proper conversion to version 2 can be done. In this case, minimal advantages from NATURAL version 2 will be seen. In particular, most object code enhancements would not be seen, but data transfer enhancements would.

When you are ready to start conversion, you will have to take into account the new "objects" that are now available. You will have programs and maps as you've always had, but now you also can have external Data Areas (Local, Global and Parameter), Subprograms (CALLNAT programs and COPYCODE), external Subroutines and HELP routines. There are also maps that are larger than 23 by 79 characters and can be scrolled up, down and sideways. The more up-front planning done in regard to naming conventions and use, the more painless conversion will be. Treehouse Software has a list of these modules, and suggested standard names. If you are interested in receiving this document please contact us.

Conversion of programs from version 1.2 to Report Mode version 2 will not be too difficult. Global variables will need to be converted to the new architecture. SAG has created a relatively painless method of global data conversion. Since you are not working with Structured Mode, you will want to specify GLOBALS SM=OFF if that is not the default. Once in your application, enter the data area editor (by typing 'E G' or 'EDIT GLOBALS'), then type 'CREATE GLOBALS *', this will cause all the global variables defined in any catalogued programs in that application to be defined in a global data area that you can STOW with any name other than COMMON. (Note: Treehouse Software recommended standards for Global Data areas is XXXDGxxx where XXX is an area prefix and xxx is free form.) The globals that you have created will be in the order they were found in your application. If you used Treehouse Software standards of having all your globals defined in one module, the order would be the same. If however, globals are defined in various programs, they will be created in the way they were found while scanning alphabetically through the catalogued programs (object modules) in that application. You will be able to change the order of the globals in the global editor. Be sure to do this before you catalog any programs that will use them.

(Continued on Page 8)

NATv2 TIPS: Conversion Considerations

(Continued from Page 7)

Once you have done this, most programs will catalog by just coding:

```
DEFINE DATA GLOBAL USING XXXDGxxx
END-DEFINE
```

in all the programs that use the globals you have created, and removing any format/length reference on all global variables. Treehouse Software understands that SAG has a program that will do this conversion that is being tried at various sites. If you would like to try it, it should be available (at this time) from your SAG representative for the asking. If you have created standard, clean code in reasonably sized modules with no 'tricky' programming techniques, you can probably feel confident that conversion will be simple. However, it is more likely that a few techniques that you have used in your shop may cause problems.

DATA AREAS:

Data Areas will seem foreign to you at first, but the use of externally defined data will allow uniformity in your applications where consistent field naming conventions and format definitions are desired. There are 3 types of Data Areas: LOCAL, GLOBAL and PARAMETER. LOCAL Data Areas contain all fields that will be referenced by a single program. It is equivalent to all non-global fields that are RESET in a program in NATURAL version 1.2. LOCAL fields are initialized at the beginning of every program. GLOBAL Data Areas contain fields that are shared by more than one program. GLOBAL fields retain their values when another program is fetched or executed. PARAMETER Data Areas are used only in programs that will be executed from a CALLNAT statement. The PARAMETER Data Area contains the fields that are passed from the calling program. No other fields from the calling program are available in the CALLNAT program.

Some shops recommend all new applications with more than one program create a central external LOCAL Data Area. For those of you familiar with the PREDICT Standard File concept, this could be considered an extension of that concept. Each application, or in large applications, each functional part of the application could have a LOCAL data area defined. This would allow each program to have:

```
DEFINE DATA LOCAL USING XXXLDxxx
END-DEFINE
```

instead of having to RESET all the individual fields in every program.

User-views could be defined in LOCAL, GLOBAL or PARAMETER Data Areas with other fields. Some shops are recommending that file User-views remain separate entities that could be named in an understandable manner.

MAP CONSIDERATIONS:

The mapping utility has changed drastically. NATURAL version 1.2's free form full screen editor was liked by everyone that used it. Non-programmers could make their own screens and modifications easily. The new mapping facility is a bit more difficult. Many of its features are not intuitively obvious, and take a bit of getting used to. You also start out in a split screen, seeing only a portion of the full screen. Although you can go to the full screen, most people find this a bit disconcerting. Furthermore, fields can't be pushed around the screen as they could before. If you didn't like the alignment, you could just move the field over with the insert key. Now you have to use a move command. Some of us had gotten spoiled with the map facility.

Now that we've discussed the negative features, let's talk about the positive ones, and there are many. Once you get used to the line and field commands on the map editor, you will find it to be more powerful than you would have dreamed possible. NATURAL version 2 has full array processing, pulling down fields from split screen User-views, programs, subroutines, data areas etc., full 3270 IBM color terminal support, complete edit masking capabilities and the ability to change field attributes dynamically. These are just a few of the new capabilities available. There are in fact so many new capabilities, that it is easy to become overwhelmed. For this reason, we'll mention a few features that we recommend starting with, and save the rest when you feel more comfortable with the mapping facility.

When creating maps you should always use fields that were defined in a LOCAL, GLOBAL or PARAMETER data area. Use the split screen technique to 'pull in' the necessary data area, and 'pull down' the fields onto the map using the mapping commands.

When inputting a map, do not use field names, just input the map name, and the field names. If they have been derived from a data area referenced in the program, they will be available to the map. It is very rare that you will need to use the field names on the map input, but if you do, they will not be in the order they appear on the map. Instead they will be in field name alphabetic order. This can be confusing.

Use system variables when needed. In particular, *DATE, *TIME, *INIT-USER, *INIT-TID etc. These variables can be defined on the map and edited in a suitable manner, and you will not have to define or move any values to them in your programs.

Make use of the REINPUT verb with your maps. REINPUT is much more powerful than ever before, and you'll appreciate having it. In particular, you can say MARK *field-name where field-name is a field on your map, and the cursor will be placed in that field. No more fancy algorithms or changing numbers when you add or delete fields. Furthermore, the MARK option also gives you the capability of dynamically

(Continued on Page 9)

NATv2 TIPS: Conversion Considerations

(Continued from Page 8)

changing the field attribute (ie: PROTECTED from UNPROTECTED etc). You will have to refer to the manual for restrictions on what changes to attributes may be done. You can REINPUT with TEXT like you always did or with HELP. That is, a HELP routine could be a window that might be used by multiple programs.

Consider how you would like to have HELP available to the end-user. There is screen help, and field help. If you define a screen help in your profile, and no field help is available, it will get the screen help. Therefore, it is recommended that you have some sort of standard screen help defined.

Use skeleton maps as much as possible. NATURAL version 2 only sends changed data to and from the terminal when the data is changed (version 1.2 sends the entire map). This is a vast improvement over NATURAL version 1.2. So now it's critical to make as much data on the screen be the same. Date, time, titles etc., may visually look like they are in the same place. However, being 1 character off will force all the data to be retransmitted.

Consider using SAG's standard PF key formatted maps. This will save some coding as well as make it easier to take advantage of the new PF key capabilities like assigning names and automatically creating the PF Key lines at the bottom of your program. The new format looks like:

```
12:02:32          Title of the Application Here          10/10/87
USER-ID              Screen Title Here

        Code          Description
        ____          ___________________________
          A           Add a Record
          D           Delete a Record
          .           Terminate
        ____          ___________________________

    Code:        Data:

Direct Command:

Enter-PF1---PF2---PF3---PF4---PF5---PF5---PF7---PF8---PF9---PF10---
     HELP        QUIT  ADD  DELE
```

You will note the last 2 lines are for PF Keys. These lines could be generated by SAG's mapping facility, and it would not have to be coded on the map. Furthermore, by using the SET KEY command, you can assign names to the keys (up to four letters). The SET KEY command is more powerful now, and in combination with setting the parameter KD=ON (key definition equal on, which causes the PF Keys to be displayed at bottom of screen), much redundant coding could be saved.

Consider if you want to use file fields on the map. If you do, you will benefit from having PREDICT rules that can be automatically incorporated into the map when the file field is specified. This is an advantage in that it makes audits uniform and treatment consistent. However, it may be confusing for new or beginning programmers. Furthermore, having layers of rules could be difficult to handle.

Windowing is becoming the standard for help. The window facility is very useful, and it assists in error handling and choice selections. Windows will take some getting used to, but their value far outweighs the time involved in learning how to use them.

ITEMS OF CONCERN:

Perform conversions on an application by application basis. Note any problems or difficulties. You should be sure to report problems to SAG, and we also would appreciate hearing them. Start with your least critical systems and test the results.

Programs and maps must be the same version. In other words, version 2 programs do not work with version 1.2 maps and vice versa. So if you convert a program you must convert all the maps it invokes.

Format specification of (N2.) with a period on the end will not be accepted. This must be changed to (N2.0). Labels are allowed in NATURAL version 2, and they are identified with a name ending in a period. Fields ending in periods will be confused with labels.

FIND FIRST and FIND UNIQUE are not available in Structured Mode, although they remain in Report Mode.

When using DEFINE SUBROUTINE, the word SUBROUTINE was optional. Now it is required.

For efficiency, arithmetic operations should be done on like packed fields. ('Like' being same length and format.) In this case, executable machine code is generated.

CONCLUSION:

We think that most of you will be really impressed with NATURAL version 2. The efficiency, power and new capabilities make it well worth making the conversion. We feel with all of us working together, we can make this conversion quick and easy. For those of you currently converting to NATURAL version 2, we would appreciate hearing about any problems you may have encountered, or your general feelings about NATURAL version 2. Everyone's interested in making the transition as easy and painless as possible, and working together is the best way to do that.

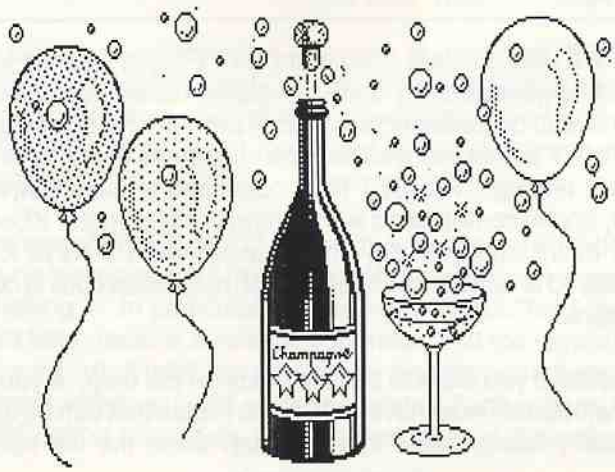
Treehouse Software at SAG Conference in Miami

All of us are looking forward to the next SAG conference in Miami on November 1st through the 5th. Treehouse Software will be there to meet with everyone, learn what we can, share what we can and tell you all about our new products and future plans. We will be having our own little conference in our hospitality suite at the Fontainebleau. The room will be open from 1PM until 6PM Sunday through Tuesday. So please come up, unwind, have some refreshments, and talk about TRIM, NATURAL, ADABAS, etc. We encourage our guests to bring examples of their experiences to share with us and others.

On Sunday we will be giving talks about TRIM enhancements, internals, user experiences, etc. We will have on hand guest speakers, and various material about new products and soon-to-come enhancements. If you cannot make it on Sunday, please arrange to meet with us on Monday or Tuesday afternoon.

MacNews

Some of you may be wondering about how we made this newsletter (Being computer people and all). Well, we went out and purchased a Macintosh SE, an Apple Laser Printer, Aldus' PageMaker and MacPaint. We have used MacPics by MAGNUM for some of the art work. We're still in the learning phases, but we're quite pleased with the result. We plan on expanding to a scanning device, and working with color mock-ups in later issues. As our expertise and software improves, so will our newsletter. We think it's nice to show that computers can be your friend, and save you time and money with a little up-front work. We are celebrating its success.



System Tips: CLOG Discussion

There is often a lot of confusion about command logging (CLOG) and how it works. This article is an effort to give you a better understanding of what occurs.

When ADABAS completes each command, it attempts to write information about this command onto a command log (CLOG). This is done by an ADABAS call to its ADALOG module (not to be confused with the NATURAL ADALOG feature). ADALOG handles command logging and protection logging for both single and dual logging procedures. ADALOG makes calls to the ADAIOR module to actually do its outputting. ADALOG is the same for various operating systems. ADAIOR functions at the operating system level and is therefore different for each operating system ADABAS runs under (ie: OS, DOS, etc). This article concentrates on dual disk command logging involving blocking of command log records.

For command logging to take place, the JCL for the ADABAS CLOG must indicate a non-dummy dataset. If the dataset is not there, or dummied out, ADALOG will not call ADAIOR to output the CLOG records. Furthermore, the ADARUN parameters must state LOGGING=YES. This will cause the 'basic part' of each command log record to be output. This part is 60 bytes long. There are further logging parameters:

LOGCB=YES	(Control Block)
LOGFB=YES	(Format Buffer)
LOGRB=YES	(Record Buffer)
LOGSB=YES	(Search Buffer)
LOGVB=YES	(Value Buffer)
LOGIB=YES	(ISN Buffer)
LOGIO=YES	(I/O List)

The basic part of the log record contains information which is known or calculated about the command, such as jobname, userid, wall-clock time to complete the command (duration), number of Associator, Data and Work I/Os (physical), the thread in which the command was run, etc. It also contains a few important pieces of information from the control block, such as command code, file, and response code. It also contains indicators about the absence or presence of the various additional logging parts (control block and buffers). Therefore, the basic part contains some of the necessary information for obtaining resource utilization statistics and performance information. But realistically, the entire control block is needed.

LOGCB controls logging of the last 64 bytes of the control block (the first 16 bytes are included in the basic part). The control block contains important performance and resource data such as the length of the format, record, etc., buffers, necessary for calculating a reasonable CPU time estimate. So, most installations turn on logging and additionally log the control block. This means log records will be 124 bytes each.

Buffers

When the format, record, etc., buffers are logged, they are each appended to the record, in order, following the control block. Each buffer is preceded by a 2-byte inclusive length. For example, if the format buffer is 37 bytes, the length field will contain the value 39 (hex 27). In addition, a bit in the front of the basic part of the log record is turned on to indicate the presence of the format buffer. If the control block is logged, the length of the log record is 124 plus 39 equals 163. This length is contained in the first 2 bytes of this standard-form variable blocked record. Each of the buffers are appended in this fashion. Usually the log record length will remain reasonable as long as the record buffer is not logged. Record buffers can be large. The log records are blocked when they are written out.

With ADABAS, you get an all or nothing situation with the buffers. If you want to log record buffers to capture data for some specific commands or response codes, you will have to log the record buffers for all commands issued by all users. There are operator commands allowing some flexibility (you can turn logging off for a while). But with TRIM in place, users typically turn ADABAS logging on fully and allow TRIM to trim things.

With ADABAS you can issue a Delete command with a record buffer or an RC command with a format buffer, etc. You can also state a format buffer length which is larger than necessary, as long as you end the format buffer properly with a period. ADABAS is intelligent enough not to deal with buffers that have no meaning for the particular command. Furthermore, ADALOG does a command check to determine which buffers would be logical to log. In the above case it would not be logical to log a record buffer for the Delete command. So, it will not be logged. ADABAS also intelligently deals with the format buffer and search buffer up to the ending period or the length of the buffer, whichever occurs first. ADABAS has full knowledge of the necessary record buffer and value buffer lengths based upon what it has seen in the format and search buffers respectively. If the format buffer is stated as "BA,15,BB,10." and the RB length is stated as 2,000, only 25 bytes of record buffer are manipulated by ADABAS and returned to the calling program. However, ADALOG logging will append the entire buffer for the length as stated in the FBL, RBL, SBL, and VBL fields in the control block.

User-exit-4

ADABAS has always provided for User-exit-4 for the specific

(Continued on Page 12)

System Tips: CLOG Discussion

(Continued from Page 11)

purpose of determining whether individual log records should actually be logged. It is documented that User-exit-4 will receive control with a pointer to the log record, and that the return from the user exit should indicate in register 15 whether to log (zero) or not log (non-zero) the record.

Over the years, users have developed User-exit-4 code to do CPU time calculations, write SMF data, write data to their own defined datasets, and log certain records. These user-exits are usually inflexible. If it is desired to write other data or to log other records, the user-exit must be modified, assembled, tested, and reinstalled with the database.

TRIM contains a very flexible User-exit-4, which can do many different calculations and outputs. For this article, we need to mention that TRIM User-exit-4 can strip off the various buffers and/or turn off logging of the entire record based upon parameters which have been fully validated and passed to this user-exit - dynamically - without bringing the database down and up. This allows for turning ADABAS logging on for the absolute worst case - log everything all the time. Then with TRIM, the logging can be cut to a bare minimum - all the way to nothing if desired. It is becoming the standard that command logging is done in a selective fashion using TRIM. Commands having many I/Os, bad responses, security violations, and other special conditions cause a small amount of detail logging during the day. To have the necessary data available for total resource utilization statistics, performance tuning, charge-back, etc., TRIM contains a User-exit-4 summarization capability (PRESUM). PRESUM records are output every hour if desired. We will talk more about PRESUM records below. Let's get back to what happens with detail command logging.

Log Records, Blocks, Problems

One log record may look like this:

RL	BP	CB	FB	RB	SB	VB

Another log record may look like this:

RL	BP	CB

These records are blocked, up to the block size of the DASD log device, typically 3K. The block may look like this:

BL	REC	REC	REC	REC	REC	——

After ADALOG builds the record, User-exit-4 is passed R1 containing the address of the log record within the log block. We have noticed that User-exit-4 can not only prevent a log record from being logged, but may optionally reduce the length of the record, such as by removing the buffer(s). Upon return to ADALOG, the accumulated block length will be similarly reduced.

User-exit-4 may optionally leave the record length the same, or make it smaller, while placing different data in the log record. This is actually what happens in the TRIM "Crunching" feature, described below. User-exit-4 may not, however, make the log record larger. ADALOG cannot adjust for this - primarily because it has the log record already within the log block. It cannot handle a log record growing beyond the block boundary.

Assume LOGGING=YES, and LOGCB, FB, RB, SB, and VB are all =YES. When ADALOG builds a log record, it first takes a look at the current log block. For example, assume the log contains records A, B, and C already, with 200 bytes available at the end of the block.

BL	A	B	C	—200 bytes—

To build log record D, ADALOG first calculates its record size. The basic part plus the control block would fit nicely in the block. However, a 100-byte FB is to be logged (LOGFB=YES). The record will no longer fit in the block, which will cause ADALOG to write the current block, A through C. ADALOG then builds a new block starting with record D.

BL	D

Record D, like all others, is presented to User-exit-4. Suppose User-exit-4 keeps the record but decides to remove the FB (log the FB only when job=xxxxx or response code=nnn). Now the block looks like:

BL	D

You can see that record D could actually have fit in the A through C block. This doesn't seem so critical, yet. Next ADALOG works on log record E. Suppose it calculates that the buffers would extend beyond the block. Then, it will write a very short block containing only record D. Shouldn't ADALOG present the log record to User-exit-4 first? Maybe User-exit-4 will remove some large buffers and the record could fit in the block. But, ADALOG cannot present the log record because it cannot build it (in the block). With TRIM's adjustments it would fit. Now consider that record buffers, especially, are usually large. And, there are many of them because they are used on many commands and you have told ADABAS LOGRB=YES (for everybody). TRIM will get rid of the record buffers, or many of them, but you can see the potential scenario. The log area on disk will become full with many short blocks, fill up faster, cause more dual area switching, and write more log tapes than it really should. While Treehouse Software advertises that TRIM cuts out log buffers and log records and reduces logging data, there are situations where more tapes are used up in this reduction!

(Continued on Page 13)

System Tips: CLOG Discussion

(Continued from Page 12)

Solution

What's the solution? Optimally, ADALOG should not build a log record with all its buffers directly in the log block. The work should not be done first, only to find it was not necessary to do. ADALOG should just build a basic part, and pass the addresses of this basic part and the CB, FB, RB, SB, and VB to User-exit-4. Forget the IB and IO list. They are useless. Then, User-exit-4 could do the work of building the log record only if it is instructed to do so for this command, file, job, response code, etc. This would be more efficient in CPU cycles and use the DASD space more efficiently. Less log tapes would result.

What do you think happens if the entire block is empty and ready to receive a log record for a command with a 4K record buffer? I think you can guess.

Since ADALOG has been inefficient all these years, it is unlikely to change soon. It needs to be replaced with TRMLOG. This is in our plans. TRMLOG will be a very efficient replacement for ADALOG because it will not perform significant work until it has checked with User-exit-4 to see if the log record and buffers should be output to the CLOG. Its blocking mechanism will be much more logical and efficient compared to ADALOG.

PRESUM

To capture a complete picture of resources used during an ADABAS session, to analyze trends, and for charge-back purposes, it is necessary to either log all CLOG detail data (basic part and control block) or to summarize appropriate statistics in User-exit-4. With TRIM, both methods are effective, although the standard is becoming *PRESUM with minimal selected detail logging* . ADABAS runs for days at most sites without filling one dual CLOG area.

Statistics for ADABAS count of calls, durations, I/Os, and CPU time are captured on every command and tallied for the current hour. Six different PRESUM categories are currently supported: thread, command code, file number, jobname, userid, and NATURAL program. At the end of each hour, the PRESUM statistics are written out to the CLOG where they reside along with any detail log records which may have been written. This precludes post processing of these redefined CLOGs with anything but TRIM. But it makes TRIM efficient and effective.

PRESUM records are each 36 bytes in length, 83 records to a 3K block. Only a few blocks are written each hour to cover all the resource statistics you'll need.

Subsequent batch TRIM-runs against CLOGs can print various detail and summary reports from detail or PRESUM data. PRESUM summaries often execute 100 or more times faster compared to similar reports from detail data.

By interspersing PRESUM data with normal CLOG records, PRESUMing requires no additional datasets or JCL beyond what is normally used to run ADABAS. ADABAS facilities for single/dual logging, disk/tape, OS/DOS/etc., are all used.

Crunching

Examination of normal CLOG detail records (basic part and control block) reveals much wasted space. Little can be done to compress command codes, userids, etc., but 2-byte file number and response codes, 8-byte password and cipher codes (usually blank), and durations which hardly ever use a full word can all be more efficiently stored on the CLOG.

We considered an ADABAS-like compression algorithm to save some log space. Clearly better would be a command-code-based record (L3s need certain fields, S1s need others, RCs need hardly anything). We ultimately decided upon a 32-bit 'bit map' indicating the presence or absence of 32 associated fields. We call this *crunching* the data. For example, the response code 0-255 (1 byte) has its presence indicated using 1 bit. Zero means the response code is not present (its value is zero). One means the response code is present, further on in the record, as 1 byte. So, the response code is either 1 bit or 9 bits. 99.9% of the time it is 1 bit. The response code length averages 1.001 bits when it is crunched. You can work up the figures for other normally blank or zero fields such as passwords, cipher codes, user area, etc. We determined that the average crunched record is usually about 40 bytes, one third of its original size. This savings eventually results in one third of the log tapes being used. Uncrunching the log records in subsequent batch processing is not time consuming. TRIM's batch processing can even convert the crunched logs back to original uncrunched format in case you want to process with special programs you may have.

***** Calendar of Events *****

October

1	Distribution of TREETIPS

November

1-5	SAG Conference
1-3	TREEHOUSE General Information Session with Refreshments at Miami SAG Conference
15	Deadline for January TREETIPS Feature Articles

December

1	Deadline for January TREETIPS Letters and Short Articles
???	Public ADABAS Internals Class Taught by Treehouse Software
???	TRIM version 3.2 Release, with Performance Guide

January

1	Distribution of January TREETIPS
1	New Treehouse Product Annoucement

February

15	Deadline for TREETIPS Feature Articles

March

15	Deadline for TREETIPS Letters and Short Articles

April

1	Distribution of April TREETIPS
1	New Treehouse Product Annoucement

Feature: Treehouse Begins Newsletter

(Continued from Page 1)

Entertainment

For sheer enjoyment we give you a page or two. All of you must keep a sense of humor to do what you do. Everyone experiences days when there really isn't much to laugh about. Share with us what makes you laugh too. We all need a good chuckle.

Features

This will depend on what's going on. We'll be doing surveys on software, hardware, operating systems, education, etc. We'll collect the information, summarize it, share it with you and keep it on file for future reference when doing our own corporation planning and responding to questions from our customers or potential customers. The surveys will help us make sure that we are giving you what you want or need. This time we are telling you about our newsletter. Next time, we'll talk about the SAG conference in Miami and more.

Where Are They Now?

Ever wonder what happened to some of the old-timers, SAG stalwarts of the past, former User-Group officers, etc? Sometimes we wonder too. We plan to look them up, interview them and let you know. If you have a specific person in mind, let us know. If they're willing, we'll write them up.

Future

We're a company always looking and planning for the future. In this respect, we have some definite ideas about what we would like to see happen. Competition is always good, and writing good software or helping people with ADABAS and NATURAL is what we feel should be the directions of everyone who uses it. We would like to have a consultants' corner. Other newsletters don't give the consultants, educators, and vendors a forum for announcements of their products and services. If you are a "consultant" or associated with ADABAS and NATURAL in any way, feel free to contribute articles and other information for inclusion in our newsletter.

We also plan to give your TRIM change and enhancement requests some attention in future newsletter issues. Maybe we'll list them all with some of our own plans and let you prioritize them. We are always interested in your comments. If you desire a new product or would like to see a better or cheaper version of an existing one, we're interested in your input.

The Staff

Let me introduce you to the staff. George Szakach, as you all know, is the President of Treehouse Software and TRIM creator. Rich Jacobson is George's assistant and wears many hats. Terri McCrum is the Corporation's right and left hand and many of you already have talked to her. Dwight Beadle, Chuck Starkey and Susan Pryor teach, consult, and develop. Susan is the editor of this newsletter. Ruth Rainey is our newest member who will help you with TRIM questions or problems. We also have several consultants who assist us when we need expertise in a specific area. These people provide technical assistance for classes, questions and general consulting.

We want you to know that this is your newsletter. We are committed to you. This can be as good as you want it to be. Please feel free to write articles and share information with us.

Sewickley in the News

The Borough of Sewickley is located 13 miles northwest of Pittsburgh, Pennsylvania on the Ohio River and 6 miles from the Greater Pittsburgh Airport. It was originally named "Seweekly" or "sweet water" by an indian tribe, the Asswikales. In the 19th century, travelers in wagons and stage coaches stopped in the area en route to their westward-bound dreams. Over the years, the area attracted riverboat captains and their families as well as some of Pittsburgh's industrial barons who built homes here to escape the "Smokey City". Today, Sewickely retains the charming village atmosphere with lovely homes, a strong civic pride and community involvement. It is a village where you can still hear the church bells ring on Sunday mornings. Sewickley could be one of the factors contributing to Pittsburgh's new title: "America's Most Livable City".

The surrounding wooded suburbs of Sewickley Heights, Edgeworth and Osborne are established, affluent neighborhoods. Some wealthy families have lived in the area for generations and the sylvan Sewickley Heights was once a summer refuge for Pittsburgh industrialists. Sewickley has its own lively shopping area. Current population is estimated to be 14,600. Houses range from $50,000 to more than $1,000,000. It is in the Quaker Valley school district with an enrollment of 1,406 students, with a 18:1 pupil-teacher ratio. Located here also is the Sewickley Academy, a highly respected private school, well known throughout the tri-state area.

We think Sewickley is a great place to visit or to live, and we plan on telling you more about it with each issue.

Entertainment

Where Are They Now?

CONTEST

The futuristic drawing on page 2 was produced on the MAC by Dwight Beadle of our staff using Cricket Draw. Some publications hold a contest to "Name the Publication". We did that in-house and TREETIPS won. We've decided to hold a contest for our readers to provide a suitable quote or name for the drawing on page 2. Some Treehouse suggestions are "Advancing the future through better products for today", "Take off with Treehouse Software" and "Ode to a Fried Egg". Use your imagination. Send us an entry. The winning entry will be published in the next issue of TREETIPS and the winner will receive a nice Pittsburgh-oriented prize. Entries must be submitted by December 1, 1987.

Who would you like to see featured here?

Some of you are familiar with 'SHOE', but for those of you that aren't, we'd like to tell you why it was chosen as the comic strip for this newsletter. Shoe is the head of a small publishing company TREETOPS which happens to be in a tree. Of course most of the characters are birds which does make it a bit different than (or maybe similar) to some of the firms we know and love. Anyway, they have everyday dealings with computers, publishing, and just getting through the day. We think most of you can identify with Shoe.

Reprinted by permission: Tribune Media Services

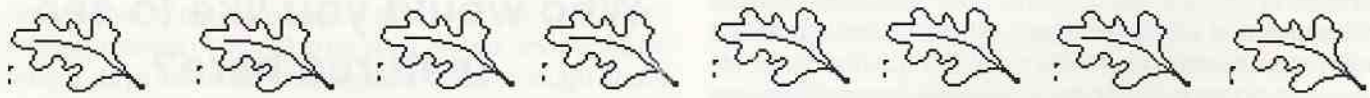
Products and Services Include:

TRIM®- an ADABAS/NATURAL performance monitor
TRUST - an online ADABAS utility submission system
Consulting on ADABAS, NATURAL and associated products
Classes:

- NATURAL Workshop
- Advanced NATURAL Workshop
- NATURAL (End User)
- NATURAL v2 Introduction
- NATURAL v2 Workshop
- Advanced NATURAL v2
- ADABAS Basic Concepts and Facilities
- ADABAS Direct Calls
- ADABAS Performance and Tuning
- Advanced ADABAS Topics
- COM-PLETE
- Customized Classes

ADABAS, NATURAL, COM-PLETE and PREDICT are products of Software AG

For further information call (412) 741-1677 or write to:
Treehouse Software, Inc.
441 Broad Street
Sewickley, PA 15143



Treehouse Software, Inc.
441 Broad Street
Sewickley, PA 15143