Colorado Springs Visit report

Andy Hamilton

February 18, 1988

1 Andy's D705 Workshop Visit report

To :-

Anthony Scott-Hodgetts
Peter Cavill
S/W Group: Tony Debling, Tony King, Graeme Tozer, Russell Wayman

Central Apps: Jamie Packer, Laurie Pegrum, Alan Pinder

USA: Philip Mattos, Steve Burns and others.

And everyone else that wants it.

1.1 Overview of this Report

This report outlines the visit to Colorado Springs by Andy Hamilton of Central Applications Group, Bristol, for the purposes of giving a 2 day workshop to American FAE's on the D705 Standalone Toolset, with particular emphasis on its use with Alien Language Compilers.

The Workshop was held on Thursday 11th and Friday 12th February 1988, with about 15 attendees.

The report will discuss the travel arrangements, preparation arrangements, course content, and finally conclude with a list of points that our American friends wish to be brought to the powers that be.

All technical issues are in the second section.

1.2 The preparation

The entire course, including structure, examples, and training material, was put together in just over a week. Many thanks to ASH's team, especially Tony King, for their assistance and information.

There was insufficient time to prepare training material to professional "customer-quality" standards. The assistance with Mac slides offered by

Sara is greatly appreciated. Ta muchly Saz!

As a means of rehearsing the course, Central Apps kindly agreed to sit through a trial run of the Material, on Monday 8th Feb. As a direct consequence of this, the course policy was restructured in such a way as to not give full information on any one topic in one go (so each topic would be revisited in greater detail). This avoided making too many forward references.

Time was so short that no S/W duplicates were taken, and the training foils were completed on the plane.

It was fortunate that I'd spoken with ASH before leaving; otherwise I'd not have taken any American money with me. As it is, I still don't know how much a "dime" is worth, except that I've brought five of them back home with me.

1.3 The Flight

Ooo-err!! The flight to Colorado was very long and tiring, especially since it was snowed-in at St Louis, Missouri, for eight hours. This meant I didn't get to the Clarion Hotel until 5 am, and I had to start the course that morning at 8 o'clock. Just time for a quick bath...

In fact, if one were to do this sort of thing again, one would arrange to be there a day early to recover from any such travel problems.

And in every case, my Psion Organizer tripped the airport metal detectors!

My discussion with Laurie before leaving led me to arrange for an Export Licence to cover the software I would be taking out of the country. In all, it had a commercial value of about 4400. This paperwork satisfied the American Customs, but I had hassle at the British Customs at Gatwick on the return leg. They were insistent that I should have filled out a "return goods" form on the way out of Britain, as they argued that anyone could have typed up my Inmos Export Licence.

1.4 The Course

The Americans wanted to know how to use the Standalone toolset with our Alien Language compilers. This really required a personal visit from somebody at Bristol as there is no documentation to my knowledge describing some of the "supporting" processes required with alien programs in different circumstances.

The Alien-languages covered were V1.3 C, V1.2 Pascal, and V1.1 FOR-

TRAN. All the alien compilers used were written by 3L. The toolset used was at September 1987 one - not the one I had *planned* to use.

All examples were integrated closely with the material covered, and handouts for reference and for the practicals were supplied for all attendees. I had insufficient time to professionally laser-print these items. I didn't ask the FAE's to write much alien code, but more to modify pre-supplied examples to incrementally implement more complex features.

The course was heavily practical-based, beginning with the trivial "hello world" program in each alien language, culminating in an extravaganza of multiple aliens running on a transputer network, on the second day.

Indeed, some FAE's must have been so inspired by my incantations that they proceeded to try and write a distributed Mandelbrot routine in each alien language and have them each updating their own quarter of the screen simultaneously. Armed with my entire set of alien documentation they embarked upon this adventure, and unfortunately were never seen again. Even more unfortunately, neither was my documentation...

I was able to offer many words of advice to get people and their practicals working, although in some cases I had no answers. In fact, I have brought back with me several disks of American problems, which mysteriously failed to compile and / or run, to be investigated.

Many questions from the audience were concerned with implementation detail of our software, and which I was unable to fully answer at the time. Consequently, I have a list of issues and points which I intend to pursue quickly so I can Catnip everyone resent with the answers.

Unfortunately, the audience were hostile to the usability of the product with alien languages, and this was not dampened any by the version incompatibility problems we had with the AFSERVER. I would go so far as to say they were singularly unimpressed with the toolset and its use with alien compilers, although it is undeniably powerful.

The main problems arose because some of my disk media, containing the toolset utilities that I planned to use was found to be unreadable in Colorado. As a consequence, we had to do the course with a toolset that was incompatible with the alien languages software. The ramifications of this are discussed in the technical section.

However, perhaps our difficulties be viewed as a blessing in disguise as this is what customers themselves may face if they do not use EXACTLY compatible products. Since each FAE had to devise a method of overcoming the difficulties, they will all be in a stronger position to assist the customers with their difficulties - if they ever want anything more to do with it.

1.5 Things I could do better next time

1. Bring readable S/W, ie, use low density media, written on a low density drive, with duplicates, and ensure not to X-ray anything before the course. This part was worked out with Philip.

2. Speak louder and more authoritatively. One must learn to control the rabble!

1.6 Conclusion

In conclusion, the course fulfilled and indeed exceeded its objectives.

On the one hand, the Americans now know how to use our existing products to allow multiple-alien transputer networks to be constructed. This will enable them to offer their customers the most well-balanced first-hand advice and support.

On the other hand, I have brought back to Bristol some questions and issues which the Americans raised during the workshop. These issues provide valuable feedback to the design processes employed here at Inmos, and can surely only lead to us producing better, more coherent, products. Once I have gathered some answers to these questions, I can further assist the Americans in their job of customer assistance in this complex area.

It is my firm belief that our engineers in other countries need this type of course without delay if they are to be able to maintain the high customer credibility and respect that we demand at Inmos.

2 Suggestions and Questions from the Americans

This is the technical bit. In detail. Indeed. INMOS.

2.1 General Information

1. Philip wants me to work with Alan Pinder and Laurie to make the course into a proper course, suitable for fee-paying customers.
2. I need the Catnip summaries every month.
3. The USA are putting a high emphasis on micro products now, embarking on a big training programme for transputers and DSP.
4. The Inmos Product Registration form (Part 72 TRN 139 00) has an incorrect zip code for the Colorado Springs postal address. There is an erroneous 9 in the zip: it should read CO 80935 rather than CO 980935. I have a sample from Philip for perusal.

2.2 Software Issues Requiring Action

Items in this section are perceived as requiring action of some sort. The first few items are presented in a prioritized manner, most urgent items first. *Suggestions* are included where possible; generally beginning with the words "We *must*..."!

I intend to notify the American FAE's and anyone else who wants to know, the comments I receive on these actionable items.

1. Somebody *must* go through the usability of our multi-alien stuff. Customers could not cope with the present arrangement, as it took Inmos people long enough to get things working. If we had a friendly front-end to all our existing utilities, to lead the user through the generation of a harness, specific to his requirements, I believe we could transform the appearance of our software beyond recognition.

   Such a front-end would be ideal for those people who, perhaps, don't want to know anything about occam. These people could simply lift their alien programs, include a few channel communication items (which could be partially automated too) and a harness could be automatically constructed for each application. Such a front end would greatly simplify the compilation (of occam and alien source), harness complexities, and the linking and configuring of the system.

I believe such a front-end could be written in less than a month to automatically generate an occam harness and even supervise the alien and occam compilations and linking. This would be an addition to our existing utilities, rather than a replacement for any of them. None of our current software would have to be changed for such a user-interface to be written.

Indeed, as recently as 18th Feb, our UK marketing and FAE's asked Central applications if such a program were possible to write in a month or so to encourage S/W houses of the ease of porting their code to transputers without having to know anything about occam. Even the stages of incorporating our run-time channel communications into customer's existing S/W could be partially automated.

2. This is REALLY important. We *must* have ONE AFserver that works with everything! The course got so seriously convoluted because the AFserver needed by all the toolset utilities (which was an earlier version, about Sept 87) was totally incompatible with the AFserver required by all the alien-language compilers. The incompatibility arose because one AFserver used BYTE tags and the other used INTeger tags, and manifested itself every time as a protocol violation.

   We currently have two version 1.3 AFservers that are *different*!

   To make matters worse, our EXE utilities supplied with toolset and alien languages, explicitly reference by *drive, name, and directory* the AFserver program. This was particularly irritating with the toolset linker, called LINKT.EXE.

   Imagine the scenario: Using a September 87 toolset and the newest alien language compilers, we had to use the alien AFserver to compile alien code (implicit in the alien EXE file used to call the B4 compiler), the toolset AFserver and linker to compile the occam harness and link the alien and occam code together. This became so ANNOYING, to ensure the correct AFserver and linker were used at each stage, requiring very careful set-up of DOS paths, development directories, and AFserver names. Some FAE's took to dynamically renaming the AFserver so the embedded references to it within our DOS EXE programs would use the correct one at the correct time. We had *real* problems in Colorado.

   In summary then, we need one AFserver that works with all our utilities, as it should be possible for the AFserver to detect whether to expect BYTE or INT protocol tags, and lock into that mode of operation. It is not adequate to have option flags at the AFserver command line, since some of our .EXE programs explicitly reference the AFserver by name and parameters. Having one multi-protocol AFserver will alleviate so much hassle.

Urgent decision needed here.

3. While on this subject, the toolset linker, LINKT.EXE, which is different from and incompatible with the alien language linker programs, is called exactly the *same* as each alien language linker. At least all our alien language linkers (and loaders) are the same now, but unless one is bf very careful with the DOS path names used, then almost certainly the DOS paths established to allow use of the alien compilers will cause the wrong linker to be run.

   In summary, we must either use and supply ONE linker for use with toolset and alien programs (currently the toolset linker is different and more powerful than the ones used by the alien language linkers), OR we must give the toolset linker LINKT.EXE a *different name* from the alien language linkers (to prevent DOS PATHs getting in the way).

4. A lot of our software is **drive-dependent**, for drive C:. This covers not just the toolset utilities, but also the alien language compiler packages. This is very very BAD, as most FAE's use drives D: or E:. One can just imagine customers doing the same too, and using logical drive names.

   The C compiler looks for the included header file `chanio.h` on C: drive. This is BAD. Again we have *drive dependent* utilities, if the C compiler thinks the "standard place" is on drive C:. This totally precludes the use of the "standard place" notation with logical drive names, or multiple / partitioned hard disks.

   Nothing must *assume* drive specificity.

   Can't we follow the practice of many other products, like Turbo C, and use files of set-up information which describe where the 'standard place" is, and all the user's own default flags and settings etc?

5. Investigation is required to find out what components of the new Function Compiler toolset fail to operate correctly with the FORTRAN and C run-time libraries. Exhibited behaviour anomalies include the C function `scanf` to hang up, and the FORTRAN output routines `WRITE` to cause garbled output. No doubt these are just surface effects of a more serious problem. But the fact is that there is some incompatibility between the Function Toolset utilities and out most recent alien languages, and we must get it fixed before widespread replication for customer sale.

6. We should document the supporting processes required for use with alien programs. For example, we should have written examples of the `stub.filer`, and the channel 0 purger, with multiple transputers and channel connections. This should be supplied with the toolset product.

As it is, I am planning on writing a Technote very soon to specifically describe what has to be done to get alien programs working with the toolset, which tied in nicely with my course in Colorado Springs.

7. The file formats used by our utilities should be published. This is especially important since we have so many of them and they are always being changed. I recently ran into difficulties when my toolset configurer objected to "incorrect link format". This was due to a change in .LSC file formats between the new Function Compiler Toolset and the previous release. It is, of course, not documented from the customer's standpoint.

   If one were to be in a situation of having a linked SC file, without the sources for it, and one wished to reconfigure it (perhaps as part of a hardware upgrade to Sun's from a PC host), then one is stuck unless one knows how to remove system dependent additives (like loaders etc) and re-process the system at the binary file level. Imagine how much time this could save in final development 'adjustments", and also in changing a transputer software base between hosts.

   This ties in with another "customer awareness" issue. I heard recently that we have of the order of eleven different loader bootstrap programs, and that bootable files written on one system *cannot* be simply AF-serve'd onto another system. This is crazy. How can we give customers an easy hardware upgrade path (eg from PC to SUN) if all the bootables have to be redone from scratch? If we had a published file format, and utilities to strip loaders etc off binary (.BIN, .LSC, .B4, and BT files), then this would significantly enhance the Inmos computability strategy. We could promote it as a toolset *feature*, rather than keeping quiet about the whole issue. The marketing team would love it!

8. We need some sort of debug facilities with our alien systems. Is this in the pipeline?

9. We need a mechanism to automatically unflatten TDS folds into flat files for use with toolset, preferably one that does not destroy all the comments in the top fold crease line. The utility should also make appropriate SC files into #SC files, with prompting to the user for a file name to use for each #SC. The bones of such a device already exists in the form of the toolset FDEP utility, which walks (briskly!) through toolset occam code and looks for #SC files and library usage (to make file dependency information since the order of compilation and linking of toolset occam source is critical). This could probably be hacked to traverse a TDS-format file and search for similar things and generate the correct set of #SC and #INCLUDE files. How about it

then? Customers will want this as a development upgrade path from their TDS commitments.

10. We need the capability to run our alien compilers with the AFserver's `-:e` option *safely.* Currently, all our alien compilers may inadvertently set the transputer error flag during normal operation (what sort of normal operation?). Surely, as with occam, being able to halt on error is critical to containing a rogue process. Surely we should have a compile-time option to exclude the generation of code that could set the error flag, at the recognised expense of performance.

11. If any of our alien language programs exceed the amount of stack space allocated to them, they fail unpredictably. Not very good for a "secure" system. So, can't a run-time routine be linked in that checks for this and can give appropriate warning to the program in danger of failing?

12. The FAE's would prefer to see *all* manuals bound into one, rather than having a Delivery Manual, a User Manual, and a Reference Manual, for each alien language. This also applies to the toolset documentation.

13. The FAE's would prefer for the occam compiler to be able to produce more than one error at each go. Currently, it terminates at each error it finds. Is this possible, perhaps as an option flag supplied to the compiler, to be able to go through all the source code and list all possible errors (on the proviso that some errors may be caused entirely by preceding ones)?

14. FAE's were concerned that the occam compiler will die if there are any TAB characters in the occam source file. They felt it was safe to assume the default of 8 spaces for one tab character, as most editors output tabs just for the hell of it. If it was suitably prominent that the compiler used 8 spaces to one TAB (or perhaps a compiler option could specify an alternative sizing - very flexible), then this would allow toolset usage with every editor on the market. Remember that a "big feature" of the toolset is that it allows the user to use a development environment with which they are familiar - must we also add the constraint that TABS are taboo?

15. The FAE's believe that it would be useful for an alien-language program to be able to know what channel it booted from. This could perhaps be provided by a run-time library utility. Such a facility would be useful to allow a program to communicate with the thing that booted it, (eg like TDS does with the host), in situations where a program may be booted from arbitrary links depending upon its purpose. So,

is there a need for an 'establish boot link" utility, and could one be provided easily?

16. It ought to be possible to use the 'output.or.fail.t" set of utilities with alien languages. The run-time library utilities could provide each of their four channel communication primitives with the two types of failure detection and correction associated with resilient systems (as available from occam) - that of abort communication on timeout, and abort communication on signal from another process. It is recognized that such capability could probably be provided at the occam harness level by having those parts of an alien, which are to communicate with hard links, wrapped-up by an occam process to provide the failure recovery needed (is even *this* possible with toolset and aliens?).

17. The C compiler seems to print trace guff on the screen when given a path name when it compiles something. The screen display observed is perhaps that of the T-code instructions being generated by the compiler.

    In fact, Glenn Hill and I have seen this behaviour with the FORTRAN compiler in connection with the Spice project, and observed a factor of about fifty times reduction in speed of compiler output.

    Something should be done about this.

18. The C compiler behaves awkwardly In some situations: For example, I am informed that according to Kernighan & Ritchie C (the standard which our compiler follows), with arrays `&buff` is the same as buff as a parameter. Yet using this form with our run-time library routine `_inmess` results in a compiler error.

    As another example, if any `#include` files do not terminate with a CR/LF, then the C compiler is unable to return to the original (calling) file - it gets stuck in the satellite included file. I have an example of this on my disk for the S/W people.

    I also have an example of the C compiler behaving differently to the Turbo C compiler in a casting operation. Our compiler appears to cast differently.

    Putting a C while loop with no actions (ie just a semicolon after it) to cause a live-lock causes utterly strange behaviour - it causes the entire program to die with no output at all. This ties in with the item discussing a problem whereby "if something *would* go wrong, then it can go wrong temporally before it happens anyway".

19. Under some conditions, an alien program is called with an (undocumented) support process called `stub.filer`, which exhibits the behaviour of an AFserver. It is used primarily by programs that expect

to be sitting on the AFserver (although only *one* can unless one uses a special multiplexer process). Now, if a transputer program sits on the server, it is called with a support process called `screen.handler`. However, the parameters required by these support processes are different (one is a subset of the other), but I was asked it it would be possible to make them "plug-compatible", with a view to making things clearer for the user.

Perhaps this could be combined with the "channel 0 purger" which is required with every instantiation of the `stub.filer` (except with Pascal?), and which is done implicitly within the `screen.handler` process anyway.

20. *All* our utilities should display a message on the screen to indicate what files they are writing. For example, the FDEP does (as it writes the .LNK and .DEP files), and this is useful. But the C compiler does *nothing* visible if it compiles successfully. It would be reassuring to have it display a message to indicate the writing of file `cprog.bin`, for example. The occam compiler is similarly quiet. Although, these utilities do have "information" switches, in the C and occam cases they don't give filename creation information *anyway.* Surely the output files being written should be default information?

21. Regarding our alien run-time libraries, if more than one alien requires the same routine from a library, then the linker will link in multiple copies of the routine. I am asked if this means that our run-time libraries are not re-entrant, and that they must use absolute addressing rather than relative addressing. Surely, although our run-time libraries require some *static* data, they could have separate workspaces which would allow only one copy of one routine to be used. Please explain this.

22. Inmos should supply the source of the occam and alien run-time libraries. This would allow users to specifically modify some of our routines for their own requirements. For example, I was told of customers who, in order to maximize performance, had to conserve every byte of code to get the program to fit on-chip. This required them to put together their own run-time libraries containing only essential routines for the program. Since Inmos must have the sources for all out run-time libraries somewhere, surely it would be possible to archive these and supply them with the toolset and alien products. Now that the component binaries of each run-time library module are supplied, this is a good step in the right direction.

23. My version of the multiplexer which allows several programs to have simultaneous access to the server does not work correctly. It fails

to terminate, possibly because it does not fully implement the most recent AFserver protocol. I need a new one to experiment with, and since it is a fairly sizeable beast, it should be supplied to customers with the toolset as standard software.

24. T2 support is wanted in USA. I pointed out that the smallest C program still needs about 40k to run it in, although I couldn't explain quite *why* it takes so much RAM. So, they still want T2 support, and presumably that would cover M2 support.

25. Our toolset comes supplied with an occam file dependency utility, used to ensure correct compilation and linking sequence (of occam code only). It can be post-processed into a DOS batch file for automatic compilation using the toolset TOBAT utility (it's drive dependent though) which is good for compiling everything in one go, but wasteful having made small occam source changes. Alternatively, one can use the toolset TOMAKE utility, which makes output suitable for use with a conditional recompilation tool, such as MAKE (which we don't supply). Although one assumes that the toolset user will wish to use his own development environment, and should therefore own a make-like utility, I wonder if it wouldn't be simple and inexpensive to include such a utility with the toolset as a complete package to allow for those who don't.

26. The toolset FDEP utility produces a .LNK file for use with the toolset linker. Unfortunately, it has to be hacked into a different form for use with alien programs. Recognising that FDEP doesn't know about the alien program names, it still requires layout changes and brackets and colons and fiddling to get the linker to use the LNK file properly. Also, the old linkers do not assume any file name extensions (eg bin) but the Function Toolset Linker one does - does FDEP correspond OK?

2.3 Questions about our Software

Items in this section *probably* do not require any action or decisions.

I intend to notify the American FAE's, and anyone else who wants to know, the answers I receive to these questions.

1. In some cases, our alien programs appear to execute in an ultra-slow "debug" mode, outputting about two characters per second to the screen. I have seen this behaviour before when static variable space had been exceeded, yet this happened in Colorado for no apparent

reason. Under what conditions will this slow-output phenomenon occur? And what is causing it? I have seen previously a message like "C startup Rev 009" in connection with insufficient workspace.

2. To enable alien programs to communicate with other alien programs, one uses the occam predefines LOAD.INPUT.CHANNEL and LOAD.OUTPUT.CHANNEL at the occam harness level, to perform the inter-connections. Generally, one would define a CHAN OF ANY to sit between each alien for one direction of communication. However, since the channel passing mechanism involves passing a vector of integer *pointers* to channels, can one simply assign the channel pointer integers to the appropriate channel pointer in the other alien? This appeared to work OK, but is it *safe*? It certainly simplified the harness a little.

3. Can one use the B008 with toolset? More generally, can one use any run-time configurable crossbar switch with toolset? I assume that the protocol used for configuring the cross-bar link switch is suitably documented to allow an occam harness to set-up the necessary connections, although this would have to be done *prior* to attempting to export code to the network. This almost implies a tow stage network boot ; first dynamically 'configure" the cross-bar switch controlling network layout, then boot the network with code to execute. Comments on feasibility and possibility of this please.

4. Our toolset utilities are generally booted and run with the `-:e` flag on the AFserver. However, on some occasions, the FDEP utility set the error flag during it's normal execution on some source occam harness files. This happened consistently. If the utility was booted *without* the error-test flag, then the utility ran correctly and concluded successfully with the LNK and DEP output files. Explain this.

5. The toolset TOBAT utility, which runs on the host PC rather than the transputer board, was observed to CRASH when running. Why? One enterprising attendee hacked the source code (which we supply with the utility - good move Inmos ; we need more of this) and used the transputer C compiler to successfully execute the utility on a transputer (having removed the drive dependency). Now that's what I call neat thinking!

6. Philip seemed to know of a Perihelion C compiler and debugger which is to be available mid year ("Mid what year", asked the FAE's). I knew nothing about it and so couldn't Comment. Info please.

7. Do all our alien programs execute at low priority? Is there an option to run anything at high priority? Could this option be easily provided,

and passed defined at the occam harness level? Or would a PRI PAR do the same job without requirement for implementation alterations?

8. On the subject of `stub.filer`, I was asked for some examples of situations where the `stub.filer` would be used with alien programs. For example, it is used with an alien program that does not talk to the AFserver, and yet has been linked with the full run-time libraries. For an alien not to talk to the server surely implies the use of channel i/o communications exclusively, and so therefore couldn't the alien be linked with the Standalone run-time library? Some examples please.

9. How does `screen.handler` know when to terminate with a Pascal program? Apparently the Pascal run-time libraries do not send a terminating character on channel zero like C and FORTRAN do.

10. Please explain the selective loading of the run-time libraries. At what sort of granularity is it selective? A 3 line C program with one `printf` statement results in bootable file of about 40k. This is truly fantastic. What is happening In there? What is linked in to take all that space? The Inmos distributed flight simulator takes only double that.

    The FAE's were not impressed with the sizes of the bootable files. In fact, on some of my examples, linking aliens with the stand-alone run-time libraries saved only hundreds of bytes, on bootable files of size 60k. Surely the standalone libraries are supposed to save space In situations where only channel i/o is used.

11. I need more information on the flag parameter used to call the alien programs with. In particular, bit 2 is supposed to determine whether the alien's run-time library terminates the filer or not, or something. I am asked what the libraries do to indicate they are shutting down. What *exactly* does the whole flag parameter do?

12. Do our toolset and alien utilities return correct "exit" information to the host operating system to allow proper use of batch files to abort subsequent operations if, say, an alien or occam compilation fails?

13. Can we send Pascal and C records / structures using the channel i/o routines for arbitrary length messages? I said we could send this in Pascal due to the `UNIV CHAR` part of the message routines, but one would have to *know* how many bytes to send since Pascal presumably won't tell you how much space a record takes. Is any of this true?

14. Behavioural question: If a harness is set-up in such a way that *would* cause an alien program to not terminate or not communicate properly, then it can cause even the parts that *should* work to fail without any actions at all. In other words, those parts temporally preceding

a bomb-out appear to be affected too, frequently resulting in total inactivity of the program.

Presumably this is a result of the implementation of our run-time channel communication routines (buffers being accidentally over written etc by a rogue communication?), but an explanation as to the cause of this is required, and how to spot it's possible occurrence too.

I have an example of this on my disk for the S/W people if they want it, which appears to be caused (in this case) by putting a C while loop with no actions (causing livelock). But *why* is the rest of the system prevented form operating?

2.4 Andy's observations on America

Lets end with some observations of the Colorado Trip.

1. TWA has no row 13 on their aircraft seating. But American Airlines does.
2. American toilets are *very* low. American baths generally have concealed inner-bath plugs, operated by levers. And the bath tap is a single wall mounted pipe which is controlled using a knob with two degrees of freedom, resulting in aerated bubbly water. Hmm.
3. Most cars are big and dirty. Lots of four-wheel drive trucks. Lots of gravel roads. Cars are cheap and so's the petrol (80 cents a gallon). Roads are not kerbed. No sprint returns on car instruments.
4. Blue skies, lots of sun, no clouds, yet it's cold unless in direct sunlight.
5. Everyone has ice with everything. Except American tea which is terrible.
6. Buildings are wooden, with underground windows, huge basements. Garage doors are electric and remote controlled. Mosquito netting is used on opening windows. Rooms are generously proportioned.
7. Shop prices have no tax shown - this is added cunningly when you come to pay for anything.