CO-OP REPORT

by

Amalaye Oyake
Rensselaer Polytechnic Institute

Coop Report: Working at the Jet Propulsion Laboratory

By Amalaye Oyake, ECSE
Rensselaer Polytechnic Institute, Troy NY

This is a summary of my co-op experience in 1994. I am a third year student of the Rensselaer Polytechnic Institute in Troy NY, My major is Electrical and Computer Systems Engineering, and my co-op experience with the Jet Propulsion Labs (JPL), was my first co-op tour,

January 10th marked my first day as an employee of JPL. Over the last eleven months I have had the opportunity to be involved in various jobs, I have also had the opportunity to undergo training. This report will describe my basic tasks, major jobs, opportunities for learning, and other significant events that occurred during my co-op tour.

Basic tasks cm the job:

My primary area of work is with the Measurement Technology Center group, of Section 351. This group provides the JPL community with solutions in-data acquisition, instrument control, and data analysis. The Manager of the MTC is Dr. Ed Baroth, and I worked on projects that were tasked out.

The first task I was given, was to review of a paper describing the programming paradigms used by LabVIEW 3.0, a data acquisition toed developed by National Instruments. I had to describe what object oriented features LabVIEW possessed, and the ways it is not object oriented. With the assistance of Mr. Chris Hartsough, I was able to complete this review. During the same week, I was able to assist Clyde Sydnor, in his reports on low pass filter designs. I assisted Mr. Sydnor in using Frame Maker, to create his reports, as well giving in input in optimizing the filter design.

Other basic tasks I was involved with in the MTC, was the challenge of learning new software tools on demand, giving Friday morning demos to customers searching for data acquisition solutions, and managing the resources of a Macintosh Quadra AV.

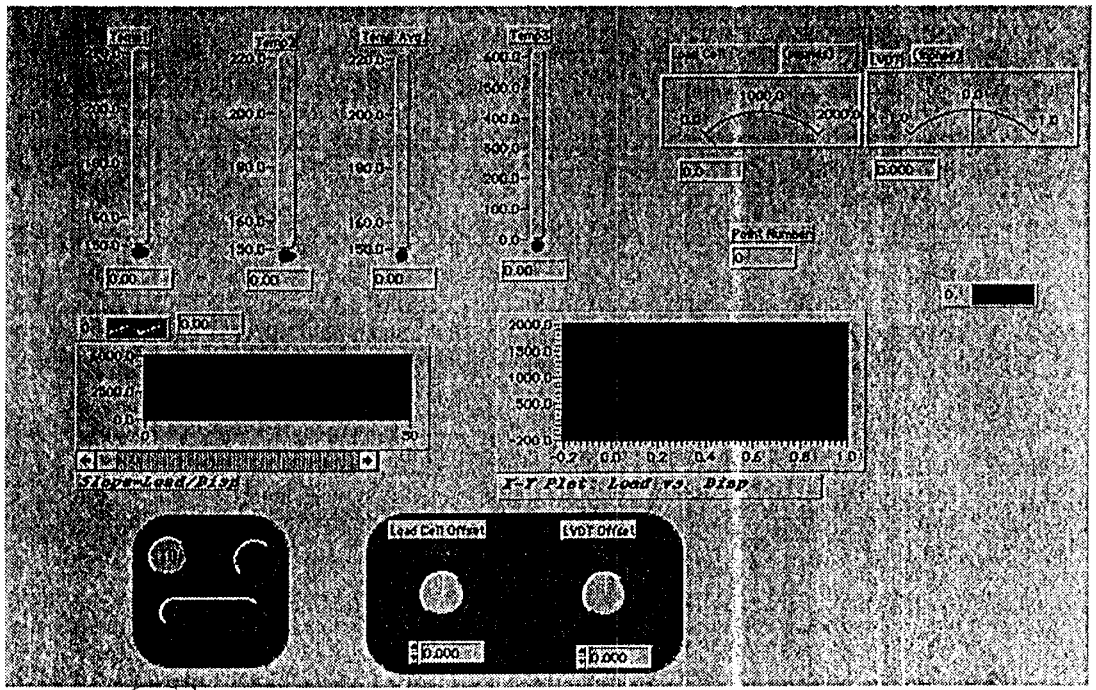
Major jobs during my co-op tour:

During the year, I worked on some major engineering projects. These jobs included; the end to end test of the NASA Scatterometer, Performing a static Test on the Cassini RTG, Evaluating Prograph CPX, Evaluating, various simulation packages, Demonstrations for Ms. Denise Pateros, Dr. Charles Elachi, and a demo for the National Alliance of Black School Educators. In support of the MTC, I have written several papers; these include "Porting LabVIEW Code Between Platforms", "Visual Programming with Prograph CPX", as well as creating a report for Mike Shumate's, structural analysis work.

A brief outline of these projects is as follows;

a. The NASA Scatterometer project: I assisted Mr. Phil Yates on the NASA Scatteromter project, an instrument that will monitor sea conditions from the ADEOS satellite. The test involved was an end to end test, which involved simulating the actual return signal. I was able to gain some experience using wave guides, function generators, and microwave generators. Working cm this job taught me important safety tips, while working with microwaves.

2 b. The Cassini RTG Static Test: On this project, I performed a static test on the Cassini Radioactive Thermal Generators, under the supervision of Mr. Doug Clark, with the assistance of Consuela Hargrove. In this experiment a mass model of the RTG was subject to various physical constraints, such as heating and tension, and readings were taken along the mass model of the various temperatures at different points, as well as the stresses using strain gauges. My task was to design a virtual instrument, to record all relevant data, creating files and real time charts and graphs of the incoming data. Fig 1. shows the front panel of the program. The Front panel represents the user interface between the hardware and the operator. The core code was written in about three days using National Instruments LabVIEW 3.01, on a 486 Pc.



; 1. The fron panel

font ?

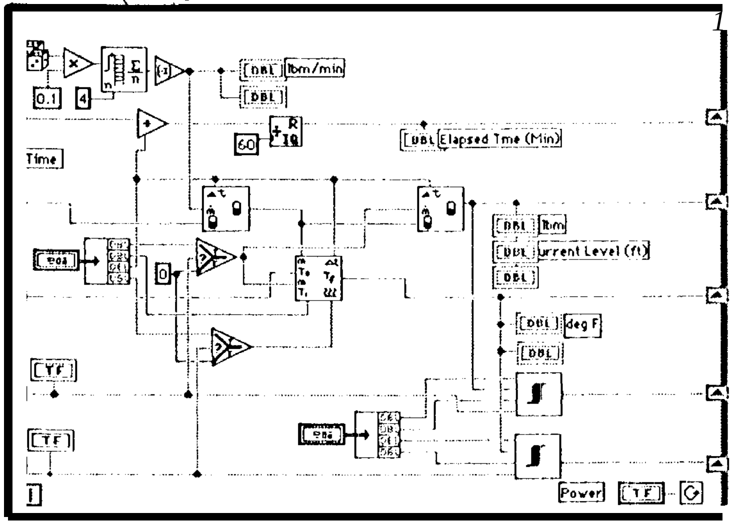
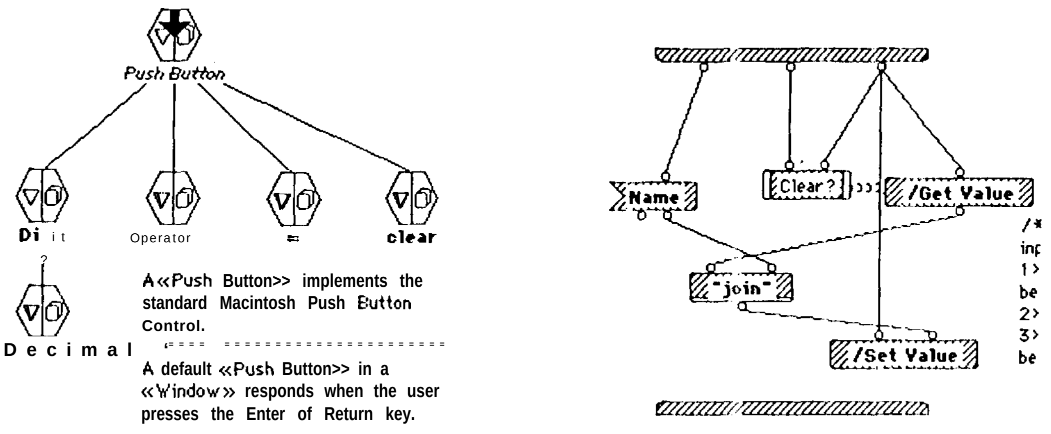


fig 2. sample LabVIEW 3.x code

3 c. Product Evaluations: An ongoing task that I performed through out the year was the evaluation of various simulation packages and new development packages. Among these were Prograph CPX, BuildSim, and G2. Prograph was a product that was actually purchased by the Measurement Technology Center. While evaluating Prograph CPX, I wrote small applications, demonstra ting its capability in the area of Data-acquisition. Prograph uses a visual syntax, thus there is textual code, except in the naming of the various visual objects. Fig 3. shows the visual code, Fig 4 shows the calculator implemented by this visual code.



A«Push Button» implements the standard Macintosh Push Button Control.

A default «Push Button» in a «Window» responds when the user presses the Enter of Return key.

fig 3. the visual code of Prograph CPX

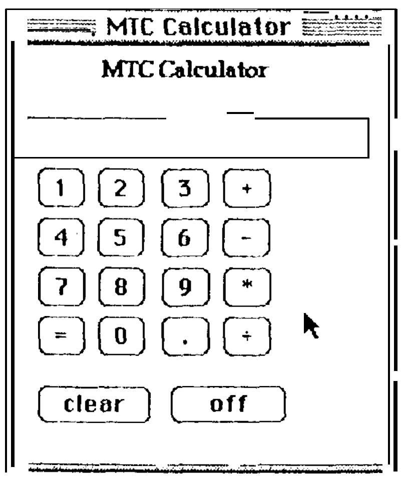


fig **4. the** *calcula [or application*

d. Official Demonstrations: At various times representatives from different Sections or Divisions on the Lab, representatives from industry, representatives from specific interest groups, and specialized organizations, visit the MTC. The visual programming tools and the relationship to the object oriented paradigm are areas usually discussed. Other demonstrations include the integration of software and hardware instrumentation. I have had the opportunity to give demonstrations to Dr. Charles Elachi, Director of the Space and Earth Science Programs Directorate; The National Alliance of Black School Educators (NASBE); Denise Pateros, Associate Dean of Undergraduate Admissions, Rensselaer Polytechnic Institute.

4 Opportunities for learning:

Training opportunities have been available within my section. While on co-op I have learnt how to use several tools. The most important however have been LabVIEW 3.x and Prograph CPX. From Oct.17 to Oct. 21, I attended a Fundamentals of Prograph CPX class, in Redwood City California. I have also attended numerous National Instruments seminars at JPL.

Other Significant Events:

During my co-op tour I had the chance to visit the following aerospace facilities; Vandenberg AFB, Edward's AFB, The Palomar Observatory, The Space Craft Assembly Facility at JPL, The Goldstone Antenna Range as well as other on lab scientific sites, such as the Micro-devices lab.

REPORT FOR

MIKE SHUMATE

STATIC LOADING TEST

JUNE 15th 1994

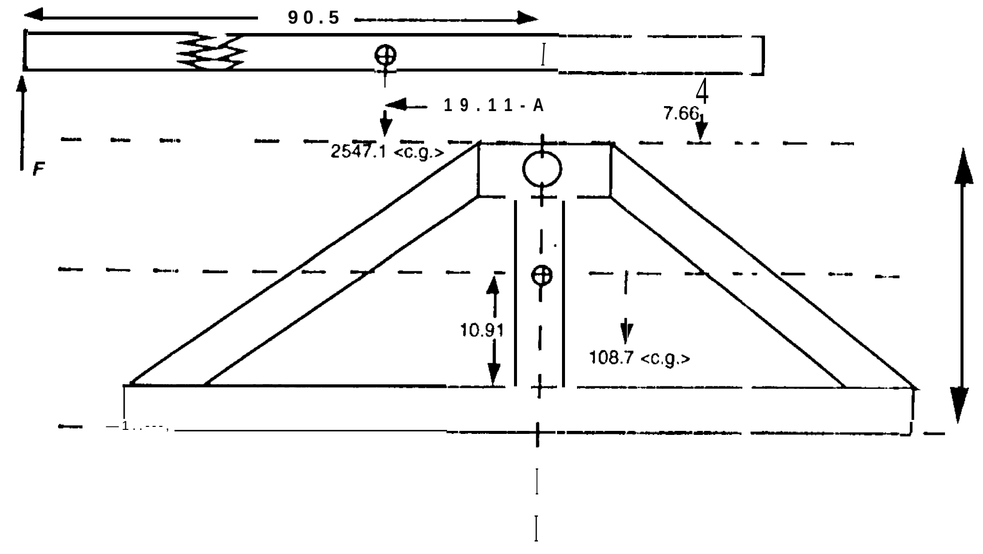
MTC 3510

*ref: Amalaye **Oyake** - **MS 12.5-177***

Add affiliation

5

Static Loading of LIDAR on Rails

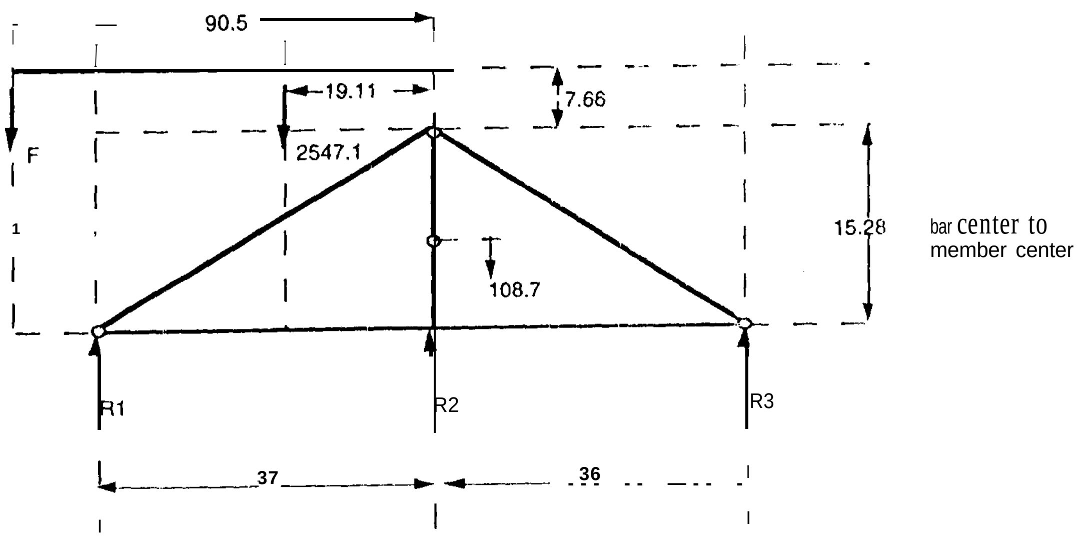


$$F = \frac{2547.1(19.11)}{90.5} = 537.8$$

Total floor loading = 2547.1- 537.8+108.7= 2118.0

Loading per inch on rails $= \frac{2118.0}{73} = 29.0$ *lb/in*

Dynamic Loading on Tabs and Floor Attatchements



F(90.5) = 9(2547 .1)(7.66)-2547.1(19.11) ‘

F = 1402

Total effect on Load at O:

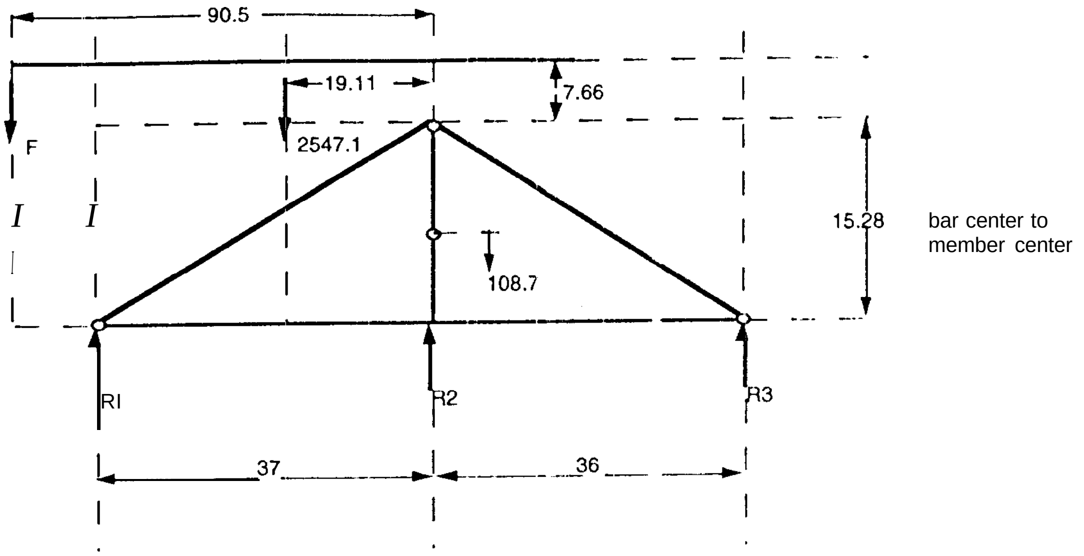
Horizontal = 9(2547.1)$\theta_2 \approx$ 22924

Vertical = 2547.1 + 1402 + 108.7 $\approx$ 4058

From symmetry, we need to only look at one bay

assume $H_1 = H_2 = H_3 = 11462/3 =$ 3821

Let R2 = F_2 be a redundant Force, consider vertical loading only.



$.381F_3 + F_2 + .391F_1 = 2029$

$.924F_3 + = .920F_1$

$F_4 = .924F_3$

$F_5 = .920F_1$

$F_1 = 2634 - 1.3F_2$

$F_3 = 2622 -- 1.29F_2$

$F_4 = 2423 - 1.19F_2$

$F_5 = 2423 - 1.19F_2$

$R_1 = 1001 - .442F_2$ $R_1 = 1001 - .442J'_2$

$R_1 = 1029 - .508F_2$ $R_1 = 1029 - .508F_2$

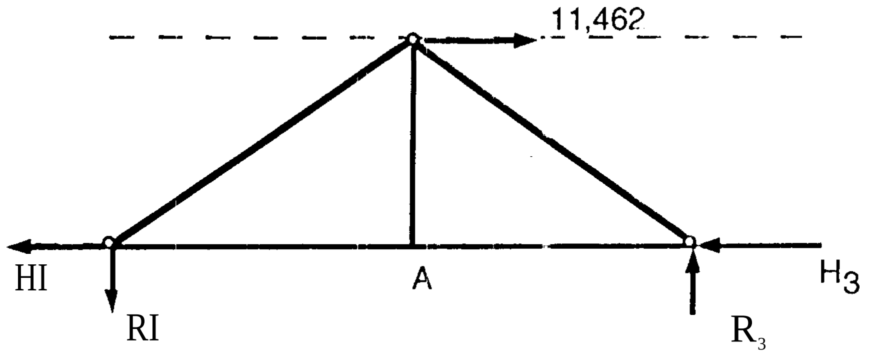
$$TAEU = [(2622 + 3F_2)(39.1) + (2622 - 1.29F_2)(40.3) + F_2^2(15.28) + (21/23 - 1.19F_2)^2(73)]$$

$$\frac{\partial u}{\partial F_2} = 0; \quad = 1.3(39.1)(2634 - 1.3F_2) - 1.29(2622 - 1.291',)(40.03 + 15.28F_2) - 1.19(2923 - 1.19F_2)(73) = 0$$

$$479,769 = 251.8F_2;\ F_6 = 1909 = R_2$$

$$R_1 = 61.8$$

$$R_3 = 58.2$$



Assume pivot about A;

$R_3' = R_1' = R; R(73) = 11{,}462(1\ 8.53)^{*}; R = 2909$

* pivot about pin.

at pin:

position	F_y	Fz
1	3821	2848
2	3821	-1909
3	3821	-2967

Pin leading on Tab.
Pins are loaded in double shear.

$\therefore \quad F_{s\,\max} = [3.821^2 \text{-t } 2.967^2]^{\%} \bullet 10^3 = 4838\text{lb}$

$F_{S\,\max} = \frac{\pi a \omega q_o}{v}$; a = pin radius = $3/16$, ω = Tab width = $3/8$

q_o = max. bearing stress (sine dist.)

$\therefore q_o$ = 43.8 hsp' (less than bearing allowable, so o.k.)

Porting Labview Code Between Platforms

By

Amalaye Oyake, Member Technical Staff"
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California

Abstract

The Measurement Technology Center, provides solutions to various scientific and engineering tasks that are encountered at JPL. The need to be able to quickly produce software applications for data acquisition, data analysis and visualization, has been an important focus. Using Labview as a tool to tackle such problems has been very successful. However, an important problem has presented itself in trying to provide our services quickly and easily to the JPL community. This problem has been the various kinds of computer platforms that exist and support Labview, and how to port Labview between these platforms. Currently Labview runs on the following platforms:

- Mac 11 and Quadra series
- Windows 3.x series of computers (386 or greater)
- The Sun Spare series
- The HP 9000 Series 700 (HP 745i)
- Windows NT

A problem has been the difficulty migrating from Labview 2.x to Labview 3.x. This has become important, because for different reasons a given Labview application will need to run on a 486 PC running Windows as well as a Mac Quadra.

What this paper looks at is a case study: Porting the DEMO BOX application from Labview 2.2 on a Mac to Labview 3.1 b on a HP 745i

Introduction

Performing this task required the following steps:

1) Loading and saving the code under Labview 3.1 (Mac)
2) Rewriting the 6b Serial Port Init sub-vi.
3) Ftp'ing the code to the HP 745i
4) Loading and re-saving all the code in Labview 3.1 b on the HP 745i
5) Reloading all sub-vi's
6) Rewriting the string-to-boolean array sub-vi
7) Running the code

Loading and saving the 2.2 code under Labview 3.1 (Mac):

This step was necessary, so as to preserve the core Labview code. There is no guarantee that Labview 2.2 code can be loaded into Labview 3.x versions on a system other than a Mac. Thus, without modifying the files or the code, the demo box application was loaded and re-saved under Labview 3.1 using "save as", to protect the original files.

Rewriting the 6b Serial Port Init sub-vi:

Labview 2.2 handles serial port initialization differently than Labview 3.1. Thus, the serial port initialization code on Labview 3.1 had to be rewritten to account for this. This one major hurdle when porting any 2.x. application to a Labview 3.x. The code for port initializations is now standardized across all platforms in Labview 3.x.

Ftp'ing the code to the HP 745i:

Once all the code has been saved under Labview 3.1 on the Macintosh, a means of transferring it to the HP 745i must be found. The best way to this is by using ftp. NCSA Telnet 2.5 is recommended for doing it from a Mac, or save the code to a PC formatted floppy (using Apple File Exchange), and ftp it from a. PC based network for example:

```
PC-486> ftp amaretto.jpl.nasa.gov
220 amaretto FTP server (Version 1.7.193.3 Thu Jul 2218:32:22 GMT 1993) ready.
Name  (amaretto:aoyake)
331 Password required for aoyake.
Pass word:
2.?0 User aoyake logged in.
ftp> binary
200 Type set to I
ftp>
```

(figure 2. example ftp commands)

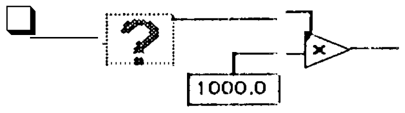
Note! always use binary transfer

Loading and re-saving all the code in Labview 3.1b on the HP 745i:

The code must then be reloaded into Labview 3.1 b on the HP745i. Labview will start to look for all the associated sub-vi's. Labview must be told to ignore the search, since it will it look for the sub-vi's in the locations they existed on before the port, which will not exist on the new machine. This problem is usually caused by the file naming conventions of each independent operating system; ftp or any file transfer method will result in some reformatting or truncation of the file name when moving between the Mac, DOS, or UNIX systems
Broken arrows may mean, missing sub-vi's within the sub-vi, which is not a problem since these can be manually reloaded also.

Reloading all sub-vi's:

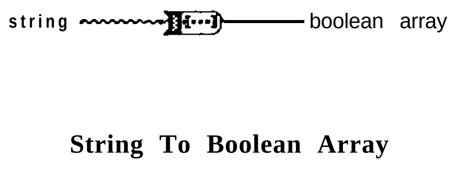
Once the code has been saved under Labview 3.1 b, all the sub-vi's must be reloaded into respective positions. A missing sub-vi will appear as a question mark, within a frame. Using the corresponding mouse clicks, it must be replaced with itself, using the "replace" option. Replacing a sub-vi by itself involves finding it's file, since during an ftp process the file's name might have changed slightly, and it would not be in a directory where Labview would think it is. This process must be repeated for all missing sub-vi's.

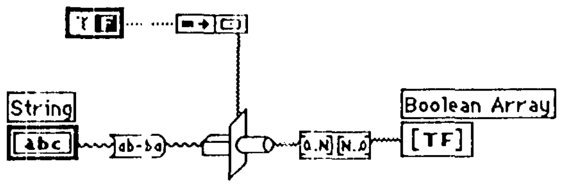


(figure 3. a missing sub-vi)

Rewriting the string-to-boolean array sub-vi:

"String to boolean array" does not exist in Labview 3.x on any platform. However, it does exist in Labview 2.x, and is an integral part of the demo box application. It had to be rewritten (see diagram below)





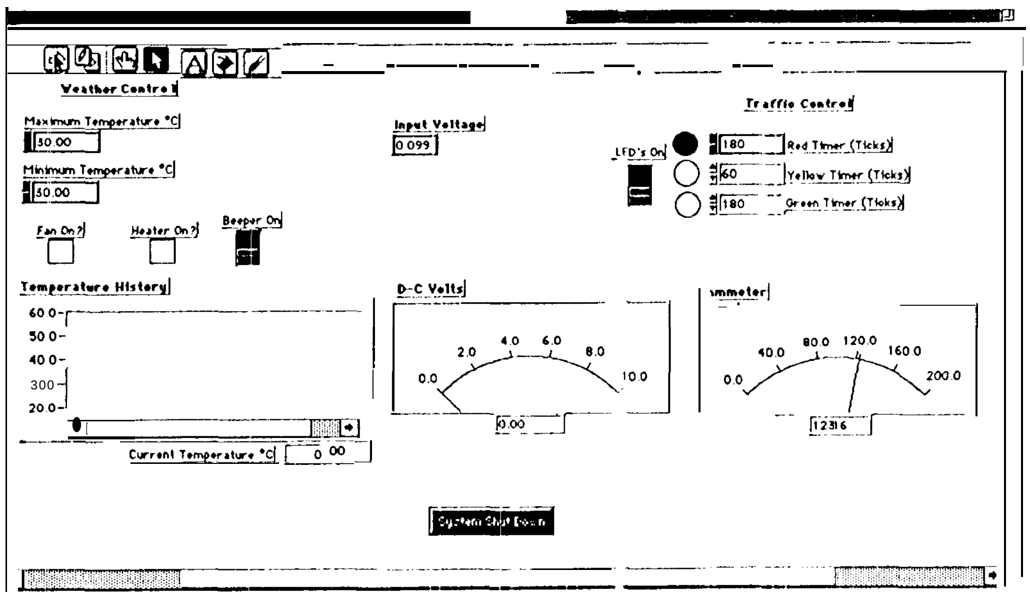
(figure 4, the function ancl the sub-vi "string to boolean array")

Running the code:

Once the code has been completely ported and there are no longer any broken arrows, the application must be tested. Sometimes an application will not run due to the fact that sometimes a given hardware platform will impose it's behavior on the operation of software code. For example, to assume that serial ports behave the same on the HP 745i as the ports on a IBM compatible system is a mistake. Although it is a safe assumption, it could have easily not been the case. In the case of the demo box, the only two hurdles were rewriting the "serial port init" and the "string to boolean array"

Conclusion

When porting Labview from different platforms, obey the guidelines listed above, and always be wary of functions omitted from Labview 2.x. It is a fairly simple task to port the applications, however, it does take time and patience, and a fair knowledge of the Operating Systems you are moving across. The demo box application behaved very well while being ported from the Mac Quadra 950 to the HP 745i. There was not any difference in front panel colors, and all the user objects maintained their respective places on the screen. National instruments maintains a stringent programming format, consistent across all platforms running Labview. This makes it a very well-behaved system if one wishes to develop similar scientific or data-acquisition applications for different platforms.



(figure 1. the demo box application)

Notes on the Demo Box;

The demo box is a demonstration of a control and data acquisition system. Labview controls a "traffic light" setup, and a heater, and also reads temperature, resistance and voltage data.

It was originally designed on Labview 2.2 on a Mac IIfx, and for the purpose of this report, was ported to a HP 745i.

A completely visual Object Oriented Programming (OOP) language
An editor/interpreter with a built in debugger
A GUI builder
An Integrated database engine
Support SQL databases
A 680x0 compiler
Support for integrating C libraries into Prograph applications

Introduction

The Prograph language is a very good implementation of several modern paradigms in computer science such as data flow, OOP properties like inheritance (only single heritance is supported) and polymorphism. It@ à provides a pictorial programming format, textual syntax is absent, text appearing only in comments, and naming methods and classes. Reusability of code is part of programming in this language, which is simply a matter of cutting and pasting the code. The editor/interpreter, coupled with the compiler provide a very friendly programming environment, It allows you simultaneously, create and edit your application, thus removing any subsequent bugs that may appear in an application.

The Measurement Technology (enter (MTC) is concerned with providing the JPL engineering and scientific research community with tools and solutions in data acquisition, instrument monitoring and control, as well as data analysis and display. It neccessary to have the capability to provide these solutions as quickly as the demand requires. Traditional software solutions in these areas usually involve languages such as BASIC, C or C++, anti require the programmer to design everything from the user interface to the essential data acquisition code.

To solve complex problems and to meet the demands of engineers and scientists, it has been demonstrated over time that visual programming tools can provide a quick and easy way to achieve the objectives of a project. These applications developed have met the requirements of the respective clients. For further reference see MTC Tasks, Ott 14th 1994.

In the MTC, the visual programming tools of choice have been Labview 2.x and 3.x versions by National Instruments, and HP-VEE by Hewlett Packard. As seen by the completed tasks list most of the jobs have involved mostly Labview, although there have been a few HP-VEE jobs.

The need to diversify, and add other visual programming tools is one of the areas being looked at by the MTC. Prograph was presented as a possible tool that could be used within the MTC. It is more general purpose that Labview, however it does allow additions to it's capabilities by the user, third parties, via C and Pascla code conversion tools.

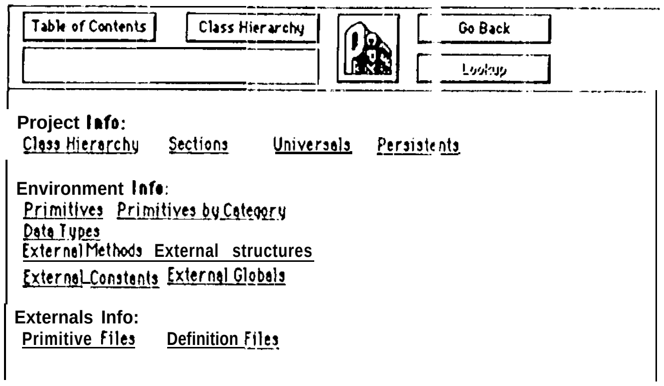
Pascal

David Prue of Acme Mining and Software inc. MD, has developed code that allows one to perform data acquisition from National Instruments Boards, and he provides a set of primitives implementing the basic NI-DAQ funct ions. Along with these external primitives, Prograph Int. itself provides a robust set of primitives that support most any data operation (please see figures 1. and 2.). Thus it is possible to perform data acquisition with Prograph, the level of complexity *however* has yet to be explored.

Dan Shafer, author of several computer related books, and an authority in Object Oriented Programming, hosts a class in Redwood City, training users in Prograph. Topics covered include;

- Introduction to Prograph CPX, and what it is,
- The "Hello World" example in CPX
- The Editor/Interpreter and the application building process
- C.S. and how 00P is implemented in Prograph
- GUI Developement and Documents
- Tips and techniques of debugging
- Primitives, External Primitives, Controls ancl Operations
- Accessing Macintosh OS resources
- Compiling the case study application

The class is over a four day period and exposes the student to other users from various background' and is very hands on. In the time spent there one is able to find out specific solutions to some of the problems of DAQ using Prograph.
At the end of the class there is an experiment with a case study application (see figures 3. and 4.). Mr. Shafer is also the author of the only third party text book on Prograph, "The Power of CPX" D. Shafer 1994.

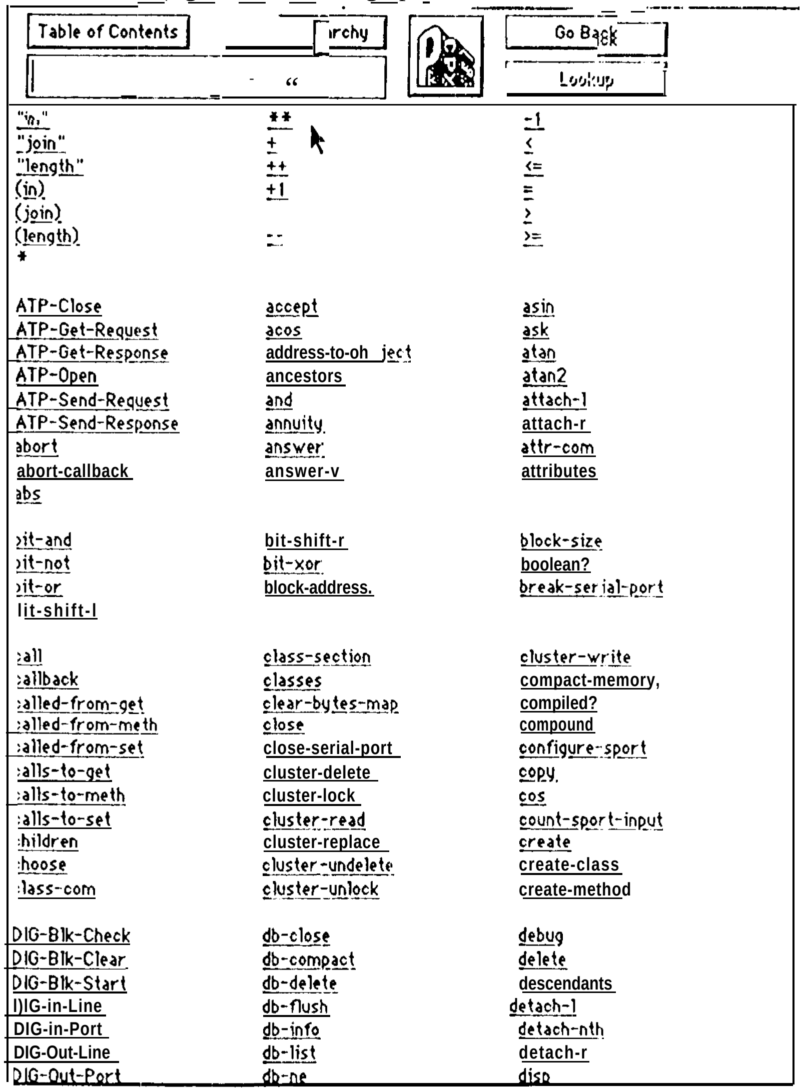


Project Info:
Class Hierarchy Sections Universals Persistents

Environment Info:
Primitives Primitives by Category
Data Types
External Methods External structures
External Constants External Globals

Externals Info:
Primitive Files Definition Files

(figure 1. the Prograph CPX info menu)

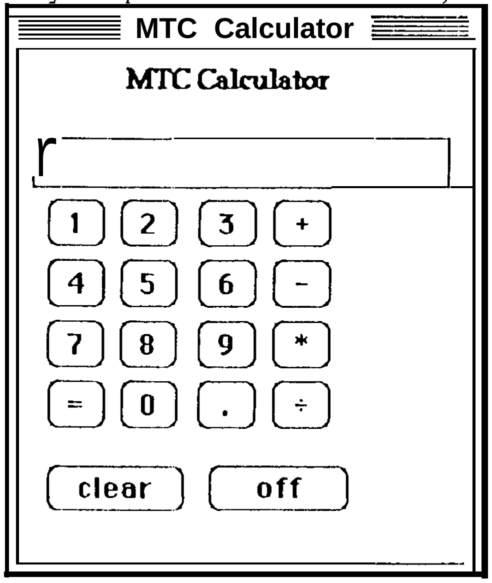


"in"	**	-1
"join"	+	<
"length"	++	<=
(in)	+1	=
(join)		>
(length)	--	>=
*		
ATP-Close	accept	asin
ATP-Get-Request	acos	ask
ATP-Get-Response	address-to-oh ject	atan
ATP-Open	ancestors	atan2
ATP-Send-Request	and	attach-l
ATP-Send-Response	annuity	attach-r
abort	answer	attr-com
abort-callback	answer-v	attributes
abs		
›it-and	bit-shift-r	block-size
›it-not	bit-xor	boolean?
›it-or	block-address.	break-ser ial-port
lit-shift-l		
:all	class-section	cluster-write
:allback	classes	compact-memory,
:alled-from-get	clear-bytes-map	compiled?
:alled-from-meth	close	compound
:alled-from-set	close-serial-port	configure-sport
:alls-to-get	cluster-delete	copy
:alls-to-meth	cluster-lock	cos
:alls-to-set	cluster-read	count-sport-input
:hildren	cluster-replace	create
:hoose	cluster-undelete	create-class
:lass-com	cluster-unlock	create-method
DIG-Blk-Check	db-close	debug
DIG-Blk-Clear	db-compact	delete
DIG-Blk-Start	db-delete	descendants
I)IG-in-Line	db-flush	detach-l
DIG-in-Port	db-info	detach-nth
DIG-Out-Line	db-list	detach-r
DIG-Out-Port	db-ne	disp

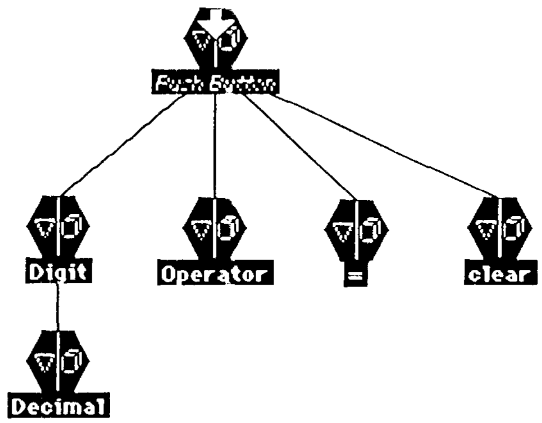
(figure 2. primitives list of Prograph CPX)

Conclusion

Creating a GUI and the supporting applications is very easy with Prograph CPX. Building a GUI simply involves putting the objects available on the palette on your Edit Application window. Writing code, though not quite as simple as building the GUI or writing code in Labview, is also fairly easy to accomplish. The difficulties in Programming in Prograph CPX, lie in understanding the ABC Starter Project (Prograph's equivalent of stdio.h), the Apple toolbox, and becoming familiar with the order and dataflow principles. Some knowledge of OOP principles helps tremendously, for example, knowing what a Class is, concepts such as Member functions, olymorphism and Inheritance, a well as Instantiation.



(figure 3. MTC Calculator)



(figure 3. Calculator code showing inheritance)

When one does become familiar with Prograph, it is a powerful tool. Applications like BuildSim (Tangent Systems) or MTC Calculator (case study - MTC, see figure 3.), were made using Prograph. Therefore due to the support, it is possible to build a data acquisition application using Prograph CPX. If there is any C code involved it can be converted to a Prograph primitive automatically via the CTool or manually using Xprims. The resulting application will be stand alone program, and can be delivered to the customer.

CO-OP Presentation:
Visual Programming and it's Applications
By
Dr. Ed Baroth
Amalaye Oyake
3510, MTC Group

What is Visual Programming?

There are several definitions:
(a) Visual Programming (VP) refers to any system that allows the user to specify a program in two-(or more) -dimensional fashion. [...] conventional textual languages are not considered two dimensional since the compilers or interpreters process them as long, one-dimensional streams. [Myers90a]

(b) A Visual Language manipulates visual information or supports visual interaction, or allows programming with visual expressions, The latter is taken to be the definition of a visual programming language. Visual programming languages may be. further classified according to the type and extent of visual expression used, into icon-based languages, form-based languages and diagram languages. Visual programming environments provide graphical or iconic elements which can be manipulated by the user in an interactive way according to some specific spatial grammar for program construction. [Golin90b]

The need to meet deadlines and complete projects has berm a concern for programmers involved; notably projects involving scientific research, Visual Programming, is quickly becoming the preferred way of meeting project requirements. Visual programming uses the visual objects to create fully functional programs. These programming languages, tools, or application modellers attempt to achieve tbe programming power of languages such as C or C++.

Non Visual Programming Languages

Despite their names Visual Basic and the entire line of Microsoft Visual (tin) family (Visual C++ and the like), are not visual programming languages. These languages use a GUI builder, layered on top a textual language, hence tier name.

Commercial Languages Available Today

Prograph	Prograph Int'l	800-927-4847
Serius Developer		
PhonePro	Cypress Research	408-752-2700
Iconicode		
LabVIEW	National Instruments	800433-3488
Design/CPN	Meta Software	617-576-6920
SystemSpecs	Ivy Team, Bern Switz	
HP-VEE	Hewlett Packard	800-452-4844

Two Languages will be looked at today:

1 National Instruments LabVIEW

2. Prograph International's Prograph CPX

LabVIEW is a graphical programming system designed for data acquisition and control, data analysis, and data presentation. It can also perform the equivalent operations of textual programming language, e.g. file searches, printing operations etc.

Prograph is a Visual Object-Oriented, Dataflow Model, Dynamic Programming and Debugging Environment. It uses object oriented principles such as classes, inheritance etc., [o create fully functional computer programs.

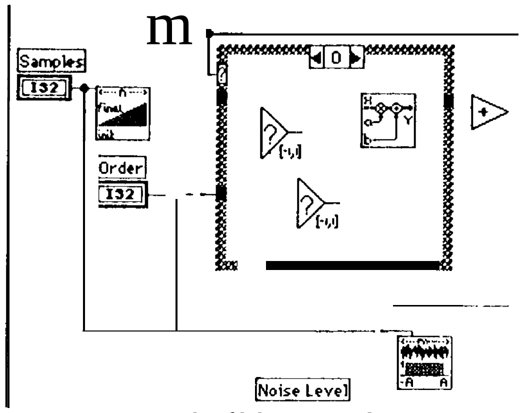
Points To Note

Both labVIEW and Prograph can import C code.

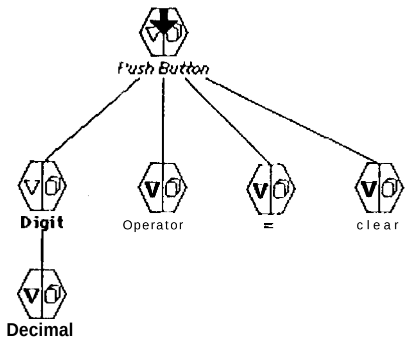
LabVIEW places more emphasis on data flow while Prograph is an object oriented product.

· Using visual programming languages makes it much easier to port between computer platforms

Both products are demonstrated at the MTC (125 -B32) every Friday, between 9:OOam and 12:00pm



an example of lab VIEW code



an example of Prograph code