

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

**Artisan Technology Group**
(217) 352-9330 | sales@artisantg.com | artisantg.com

**Full-service, independent repair center**
with experienced engineers and technicians on staff.

**We buy your excess, underutilized, and idle equipment**
along with credit for buybacks and trade-ins.

**Custom engineering**
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

**Expert team | Trust guarantee | 100% satisfaction**

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

**Visit our website - *Click HERE***

ResearchGate

See discussions, stats, and author profiles for this publication at:
https://www.researchgate.net/publication/227138291

# Pascal external functions for Strawberry Tree's "Analog Connection Workbench" on the Macintosh

**ARTICLE** *in* BEHAVIOR RESEARCH METHODS · DECEMBER 1994

Impact Factor: 2.93 · DOI: 10.3758/BF03204666

CITATIONS
2

READS
36

**3 AUTHORS**, INCLUDING:



Jonathan Vaughan
Hamilton College
**49** PUBLICATIONS **1,385** CITATIONS
SEE PROFILE



David A. Rosenbaum
Pennsylvania State University
**158** PUBLICATIONS **5,305** CITATIONS
SEE PROFILE

Available from: Jonathan Vaughan
Retrieved on: 22 January 2016

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*Behavior Research Methods, Instruments, & Computers*
*1994, 26 (4),461-466*

# Program Abstracts/Algorithms

## Pascal external functions for Strawberry Tree's "Analog Connection Workbench" on the Macintosh

MICHAEL M. BUSHE
*University of Massachusetts, Amherst, Massachusetts*

JONATHAN VAUGHAN
*Hamilton College, Clinton, New York*

and

DAVID A. ROSENBAUM
*University of Massachusetts, Amherst, Massachusetts*

*Strawberry Tree Incorporated is one of several manufacturers of input–output boards for the Macintosh computer. The Analog Connection Workbench program (Strawberry Tree Inc.) provides an icon-based programming system for these input–output boards that has most of the functions that are necessary to implement behavioral experiments. However, experimental psychologists may frequently encounter needs that the Workbench program cannot easily meet. Because the Workbench is programmed in C, whereas many psychologists are more expert in programming in Pascal, we describe, in tutorial fashion, how the user may define new external functions in Pascal on the Macintosh and integrate these functions into the icon-based Workbench programming system.*

Analog Connection Workbench is an icon-based programming system distributed by Strawberry Tree Inc. (San Rafael, CA) for use with the Strawberry Tree analog and digital input–output (I/O) boards.[1] We have found these NuBus I/O boards and the Workbench software package to be both powerful and relatively affordable. The best features of the Workbench software are its ease of use and its flexibility, although some users have expressed concerns about its speed in comparison with other manufacturers' products (which are, in turn, generally somewhat more expensive).

Workbench uses icons that can be placed on a worksheet using standard click-and-drag procedures. The user links the icons with "wires" to describe the desired flow of information. Workbench allows one to present stimuli and gather analog and digital data while performing other functions, such as on-line calculations (including derivatives and integration), logging data to the hard disk, and graphing data, all of which may be controlled by built-in logical operators.

Although the software is flexible, it has limitations. Despite the many functions already provided in the standard software package, other specialized needs are likely to arise, especially in experimental research. Examples of such needs, which the icons provided in Workbench do not directly permit, are (1) the display of visual stimuli, (2) the generation of complex or harmonic sounds (which may have been previously generated and stored in 'snd ' resources, which contain the information necessary to play a desired sound) and (3) certain mathematical calculations, such as grouping numbers or excluding certain values, and mathematical functions that depend on past events such as reaction times. There are other behavioral research needs that Workbench does permit, but these may take too much processing time or be cumbersome to implement, such as the generation of arbitrary sound waveforms (which requires several icons in the standard Workbench) or the presentation of speech that has been captured with the Macintosh built-in microphone (which may be more efficiently implemented using 'snd ' resources).

In order to accommodate unpredictable user needs, Workbench allows for the addition of new icons to its repertoire by adding *external functions* to one of the standard programming icons, the calculation icon. External functions provide a way for Workbench to incorporate a procedure or function written in C or another language into the Workbench worksheet. The external function may define new windows, display graphics, produce sound, or do any other type of operation that a program can perform.

The purpose of this article is to explain how to write and use an external function in the Pascal language—for example, the calculation of average reaction times over a block of trials. Since the Workbench manuals provide information only for users of the C language, this information will be useful to users of other languages, especially Pascal.

### External Functions in Workbench

An operation that is not directly available in Workbench can be incorporated by writing a code resource for it. When the code resource has been installed in Workbench, it appears in the list of functions that are available when the Calculation icon has been opened. Each external function (like all standard functions in the calculation list) may have two inputs, an $X$ input and a $Y$ input, and one output. In addition, each function icon also incorporates three internal constants ($a$, $b$, and $c$)

This work was supported by Grants BNS-8710933 and BNS-908665 from the National Science Foundation. We appreciate the helpful comments of Virginia Vaughan. Address correspondence to J. Vaughan, Department of Psychology, Hamilton College, Clinton, NY 13323 (e-mail: jvaughan@hamilton.edu). None of the authors have any financial interest in the software described here.

 Copyright 1994 Psychonomic Society, Inc.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

for modulating its operations, whose values can be set by opening the icon on the worksheet. Even though each icon is limited to two inputs and one output, more complex calculations may be accomplished by appropriately linking several icons. For example, to calculate $(i + j) / (j + k)$ would require three standard calculation functions: one for computing $i + j$, a second for computing $j + k$, and a third for dividing the numerator by the denominator. Nevertheless, for complex calculations it may be computationally more efficient to write an external function to perform the operation using only one icon.

**Writing an External Function**

This section is a quick introduction to the method for writing an external function and is intended to supplement the Workbench manual (Strawberry Tree Inc., 1989). The Workbench manual explains how to incorporate external functions in general and provides sample code in the C language (but not in other languages). Since many psychologists are more familiar with the Pascal language, the code necessary for using Pascal external functions will be described.

When a program is running, the Workbench program cycles at a high rate (as fast as 1 cycle/msec, depending on computational load and user preference) through the icons that are active on the worksheet. Each icon corresponds to an underlying built-in (standard) or external (user-programmed) function. Each calculation icon, whether built-in or external, is linked to a record containing the icon's variables. These variables hold (among other data) the values of the *X* and *Y* inputs, the *a*, *b*, and *c* constants, and a pointer to where the function's output should be stored. When an icon's external function is to be executed, the function is called with a single parameter, a pointer to the variable record that Workbench maintains for the icon. (The structure of this record in C may be found on pp. 7-4 and 7-5 of the *Analog Connection Workbench 3.0 Manual*, Strawberry Tree Inc., 1989.) If the same function is used in more than one place in a Workbench program, a separate variable record is maintained for each copy of the function, so each copy can have different inputs and outputs, and the constants of each copy may take on separate values.

In programming the code resource for an external function, it is most convenient to place the variable record type definition in a separate file, so that all external function programs can use the same definition. The next step is to write the code for the external function itself. The external function code must be written as a single procedure (which may, nevertheless, contain subprocedures). The procedure must be called *main*, and it takes the pointer to its variable record as its single argument. (The pointer is a static parameter and not a variable, or "var," parameter.)

The final step is to compile the code into a named code resource and install it. See the manuals for the appropriate programming language to find out how to build a code resource (e.g., *Think Pascal User Manual*, Symantec Corp., 1991, pp. 163–171). Once the code resource has been created, use ResEdit or another resource editor to copy the code resource and paste it to the list of code resources that already exists in the file "WBextern" in the "External Functions" folder of the Workbench system. The new external function will then appear in the list of functions in the calculation icon on the worksheet.

**Details of the Definitions File**

The Workbench manual describes the declaration for the variable types of the definition file for the C language. Whatever the source language, the number of bytes allocated for each variable in the record type declaration must exactly match the number of bytes in each of the fields in the Workbench variable types. In the C version of the definition, a number of variables are of the type *short*, which is 16 bits and corresponds to Pascal's *integer* type. C's *unsigned byte* type corresponds to Pascal's type *char*.

All communication between the Workbench program and the external function is governed by the Workbench variable record. The Pascal version of the variable record is shown in Figure 1. The record, of type WBParam, has three fields. The first field of WBparam is the *message* variable, an integer that the Workbench program uses to signal to the external function a desired operation. (For instance, the function will be initialized when a new worksheet is opened, it will perform the predefined calculation whenever it is the icon's turn in the cycle of the worksheet, or it will be closed because the user has quit the program. These operations will be further explained below.) Object-oriented programmers may wish to think of these operations as the "methods" of the function.

The second field of WBparam is wbp_in, which is a record of type wbpin. The wbp_in field holds all the input variables and constants (*X*, *Y*, *a*, *b*, and *c*), among other data that may be required by external functions. The third field of WBparam is wbp_out, which is a record of type wbpout. The first variable in wbpout is a pointer, *result*, to where the output of the function is to be stored.

**The message field.** The WBparam record contains the integer message variable, WBPtr^.WBparam.message. The Workbench manual contains detailed explanations of the messages that can be communicated by this variable. Whenever the function is executed, the message field informs the program as to what aspect of its function to execute (Open, DoCalc, or Close). Each external function is called first when a Worksheet is opened and then once during each of the Workbench cycles, whereupon it branches to perform the appropriate function as indicated by the message variable. *Open* calls for initializing any necessary internal variables or data structures, allocating any needed memory for userData (using the Memory Manager), installing icon-specific menus, and any similar once-only functions. Do_Calc calls for performing the appropriate computations relating the output to the input. *Close* calls for performing any necessary housekeeping chores (e.g., memory deallocation,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
Bushe defs.p                                          Page 1
Monday, December 7, 1992                              3:28:01 PM

unit WBPascalDefs;
interface

 type
  wbpin = record
    nodeRef: integer;(*integer is 16 bit = C's 'short'*)
    refCon: Longint;  (*Longinteger is 32 bit = C's 'long'*)
    userData: Handle;(* a handle is 32-bit *)
    quickTime: Longint;
    lastCycle: Longint;
    doWBonce: ptr;  {32-bit}
    eventPtr: ptr;
    menuItem: integer;
    itemNum: integer;
    WBVref: integer;
    extRF: integer;
    hasX: boolean; {Pascal boolean is stored as 8-bit char}
    hasY: boolean;
    X: double;      {doubles are 64 bits}
    Y: double;
    a, b, c: double;
   end;
  wbpout = record
    result: ^double;
    errFlag: integer;
    runFlag: integer;
   end;
  WBparam = record
    message: integer;
    wbp_in: wbpin;
    wbp_out: wbpout;
   end;
  WBPtrType = ^WBParam;

implementation
end.
```

**Figure 1. The Pascal type declarations file for the Workshop variable record of an external Workbench function.**

menu removal, equipment shutdown) prior to terminating the program.

**The input field.** The wbp_in field of WBPtr contains a number of variables, some of which are used in the present example.[2]

The *X* and *Y* variables (each a double-precision real) represent the values of the two inputs that are wired to the calculation icon on the worksheet. The external function will use one or both of these inputs when performing its calculations. In the present example, the *Y* input provides a parameter for the function's calculation, whereas the *X* input serves as a signal to indicate which calculation is to be performed. The *a*, *b*, and *c* constants (also double-precision real) are local to the external function, and they may be entered or changed by opening the icon on the worksheet. (Users should note that if the external function changes the value of one of these constants, it will automatically be restored to the value that was set in the icon when the function is called in the next Workbench cycle.)

The other extensively used variable in the wbp_in field, userData, is a handle that the user can exploit to refer to any data the function must retain from one Workbench cycle to the next. The following is an example of how an external function maintains such data when calculating the interval between two responses. When opened, the function defines and allocates memory in the userData field for a variable, LastTime, which is set to the current time (the *X* input) when each response occurs. On any program cycle when a response has occurred, the current time is subtracted from LastTime to compute the time between the two responses.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LastTime is then set to the current time so it can be used to calculate the interval to the next response.

**The output field.** The *result* field of WBPtr is a pointer, WBPtr^.wbpout. result^, to a double-precision result variable. Assigning the result of the final calculation of the external function to the result variable passes the result to the output "wire" of the external function's calculation icon on the worksheet.

### Example of an External Function

AveReactionTime is a code resource, written as a Pascal unit, that implements an external function to calculate the average reaction time of a series of responses. Negative reaction times (which should not occur in this case) and reaction times greater than 3.0 sec (which might indicate subject inattention) are excluded from the calculation. The Workbench application is the driver of the unit.

A Workbench external function in Pascal must always declare the data structures, either explicitly or in the interface section of a definitions file (as in Figure 1). Other files may be required, according to one's needs, as with any other project. For example, sound production will require the appropriate sound manager interface files to be added to the project. Each Workbench external function must be compiled as a separate unit with a single procedure named *main*. The only parameter passed to *main* is WBPtr, a pointer to the variable record that Workbench maintains for this calculation icon. WBPtr is not a variable ("var") parameter. The procedure *main* will declare any types, variables, and procedures that are local to *main*. For example, the calculation part of the AveReactionTime function is called every time a response occurs, with one of its input arguments being the reaction time of the trial. Therefore, variables are defined to accumulate reaction times and to count the number of trials so that average reaction time can be computed. Storage for variables whose values must be retained from one Workbench cycle to another must be allocated by the Memory Manager. In the type section of AveReactionTime is RecData, a record type that defines the data allocated to the userData handle of the wbpin record (see "The Input Field" section). The userData handle is then used to store data that can be retrieved the next time the function is called. Following the record declaration are two types that are necessary when assigning the userData field to point to the record. The type RecHandle is used to typecast the userData handle when the user's variables are referenced.

The main body of the external function is divided into operations to be performed when the worksheet is opened (e.g., initializing variables and setting up handles), to be performed in each worksheet cycle (the central computation of the function), and to be performed when the worksheet is closed (e.g., deallocating memory for function-specific data structures).

The first section of the main body of the AveReactionTime function is the *open* section. The *open* function (to be distinguished from opening an icon by double-clicking on it) will be called when the worksheet containing an external function is first opened, or when the external function has been selected from the list of functions in the calculation icon and pasted onto the worksheet. Specifically, when the *open* message (WBPtr^. message = 1) is received by the AveReactionTime function, it dynamically allocates memory for the RecData record. Memory for RecData is set aside and the appropriate handle (in this case, WBPtr.wbp_in.userData) is set to point to RecData. The handle is immediately locked so that the data cannot be lost between cycles by memory management functions. The two variables of the record are initialized to 0. Notice that each reference to these variables must be typecast as a RecHandle (RecHandle(WBPtr^.wbp_in.userData)^^...) because the type of userData is not specified in the type definition file.

The Do_Calc message (WBPtr^.message = 4) is received when it is the external function's turn in the Workbench icon cycle. In this example, the icon for the AveReactionTime function has two inputs. The *X* input is treated as a trivalent switch, which can take on the values −1, 0, and +1; the *Y* input is driven by the output of a timer icon, which reports time (in seconds) to the nearest millisecond.

On most cycles after the first, the value of *X* when the function is called is zero (no action is required or taken). If *X* is equal to 1 (indicating that a response has occurred), then the value of the *Y* input (the timer, indicating response latency) is checked to see if it falls within the valid range. Only reaction times between 0.0 and 3.0 sec are considered valid, so other times are ignored. If *Y* is between 0.0 and 3.0, RecData is accessed through the userData handle. The NumOfAdditions variable is incremented by 1 and the TimerTotal is incremented by *Y*. These two variables keep track of the number of responses and the cumulative reaction time, respectively. The result field is set to the timer total divided by the number of additions, via WBPtr^wbp_out.result^. Thus, the result contains a running average of the *Y* input.

The third possible value of the *X* variable, −1, gives the user a way to reset the running average in the middle of an experimental session, by manually changing the *X* value to −1. On the worksheet, the value of the *X* input of an icon may be changed by unhooking the *X* input wire, opening the icon, and typing in a new value of *X*. The change takes effect when the icon is closed.[3]

The last operation is *Close*. When the *Close* message is received (WBPtr^.message = 2), the handle to RecData is disposed of so that it is not left floating in memory when the function is deleted or the worksheet is closed. (Failure to deallocate memory that a function has allocated may cause later memory problems.)

### Making the Function Into a Code Resource

When the external function has compiled properly, the next step is to build the code resource (using RSRCRuntime.Lib in place of the standard Runtime.Lib library) and load it into Workbench by using a resource editor such as ResEdit. The code resource

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
unit AveReactionTime;
interface
 uses
  WBPascalDefs;
 procedure main (WBPtr: WBPtrType);

implementation

 procedure main (WBPtr: WBPtrType);
  type
   RecData = record
     TimerTotal: Double;
     NumOfAdditions: integer;
    end;
   RecDataPtr = ^RecData;
   RecHandle = ^RecDataPtr;
 begin

  if WBPtr^.message = 1 then {Open message: Allocate memory for total and N, and initialize}
   begin
    WBPtr^.wbp_in.userData := NewHandle(SizeOf(RecData));
    HLock(WBPtr^.wbp_in.userData);
    RecHandle(WBPtr^.wbp_in.userData)^^.TimerTotal := 0;
    RecHandle(WBPtr^.wbp_in.userData)^^.NumOfAdditions := 0;
   end; {if message = 1 }

  if WBPtr^.message = 2 then {Close message; houseclean}
   begin
    DisposHandle(WBPtr^.wbp_in.userData);
   end;                          (*if message = 2*)

  if WBPtr^.message = 4 then {Do_Calc message: compute average RT}
   begin
    If WBPtr^.wbp_in.X = 1 then {add current RT to running average}
     begin
      If (WBPtr^.wbp_in.Y > 0.0) and (WBPtr^.wbp_in.Y < 3.0) then
       begin
        RecHandle(WBPtr^.wbp_in.userData)^^.TimerTotal :=
RecHandle(WBPtr^.wbp_in.userData)^^.TimerTotal + WBPtr^.wbp_in.Y;
        RecHandle(WBPtr^.wbp_in.userData)^^.NumOfAdditions :=
RecHandle(WBPtr^.wbp_in.userData)^^.NumOfAdditions + 1;
        WBPtr^.wbp_out.result^ := RecHandle(WBPtr^.wbp_in.userData)^^.TimerTotal /
RecHandle(WBPtr^.wbp_in.userData)^^.NumOfAdditions;
       end;(*if*)
     end;(*if  X=1*)
    If WBPtr^.wbp_in.X < -1 then {Zero the running average}
     begin
      RecHandle(WBPtr^.wbp_in.userData)^^.TimerTotal := 0;
      RecHandle(WBPtr^.wbp_in.userData)^^.NumOfAdditions := 0;
     end;
   end;(*if message = 4*)

 end;                          (*Main*)

end.
```

**Figure 2. Code for external function AveReactionTime, a code resource for a user-defined computational icon for the Strawberry Tree Analog Connection Workbench.**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

is of type "WBex" for functions that do not need a 68881 coprocessor, and of type "WBem" for those that do require the coprocessor. The Purgeable, System Heap, and Preload attributes should be on. Constraints on the resource ID number for the external function are explained in the Workbench manual. Because the same ID number may not be used for more than one external function, the resource ID numbers of the existing functions in the "WBextern" file should be examined with a resource editor. Compile the code resource onto the disk, then copy the code resource and paste it into the WBex section of the "WBextern" file. The name of the external function will now appear in the calculation icon in the Workbench worksheet when it is opened.

For debugging purposes, an external function can be tested independently of the Workbench system. It is often time-consuming to repeatedly build the code resource, copy it into the "WBextern" file and then run Workbench while debugging. In addition, some Workbench functions do not run properly under Multifinder, making the resource compilation and installation process even longer due to the need to shift from one application to another. In order to save development time, a program can be tested within the programming language application by linking the external function unit with a main program that simulates Workbench by passing a WBPtr parameter to the main procedure to manipulate the *X*, *Y*, *a*, *b*, and *c* variables for testing.

## REFERENCES

STRAWBERRY TREE INCORPORATED (1989). *Analog connection workbench 3.0 manual*. Sunnyvale, CA: Author.

SYMANTEC CORPORATION (1991). *Think Pascal user manual*. Cupertino, CA: Author.

## NOTES

1. For the example described here, we used Analog Connection Workshop Version 3.0.6 for the Mac II running under System 6. The Analog Connection Workshop has recently been introduced in an IBM-compatible version, which we have not tested.

2. In the Pascal version, booleans must be used in place of C's unsigned chars (both are 8-bit unsigned bytes). Workbench only puts in "true" or "false" in these variables, so there is no need to worry about any out-of-range errors (e.g., the compiler getting a char when it is looking for a "true" or "false"). Notice too that the pointers to unused variables may be declared as type pointer even though in the Workbench manual (chap. 7, p. 4) they are declared as pointers to a particular type. Leaving these declarations as generic pointers poses no problem if the external function does not use these fields. To use them, they must be declared or typecast as pointers to particular types, as in the case of the result field in the wbpout field. Ordinarily, user functions should have reason to reference only the *X*, *Y*, *a*, *b*, *c*, message, userData, and results fields of the WBparam record.

3. Another way to manually affect the operation of an external function icon is by providing it with a menu. We did not utilize the menu feature of Workbench, but the Workbench program provides facilities to implement menu items and link them to external functions.

(Manuscript received December 10, 1992;
revision accepted for publication May 16, 1994.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

**Artisan Technology Group** is an independent supplier of quality pre-owned equipment

## Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

## We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

## Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP