Datasheet: DlgMake.s2s, DlgDesign.s2s

Introduction

Writing the code to generate dialogs in a Signal script line by line can become repetitive and rather tedious. However, you can speed up the process by using one of the above scripts to generate dialog code for you. In both cases, you select the next item to add to your dialog by clicking on a toolbar button. Then you are prompted to enter relevant information, such as the lower and upper limits of the acceptable range of numbers, before proceeding with the next item. When you have finished adding dialog items, the script will generate the code to create your dialog automatically. You can then copy and paste it into your own script.

The *DlgMake* script works in all versions of Signal for Windows. You should find it in the Signal scripts folder on your hard disk. It is suitable for creating 'simple' dialogs. By simple, we mean that the dialog consists of a list of items one below the other, with automatic justification of the prompts and selection boxes and with OK and Cancel buttons in the bottom row. There is a limit of 20 dialog items with an argument in DlgShow().

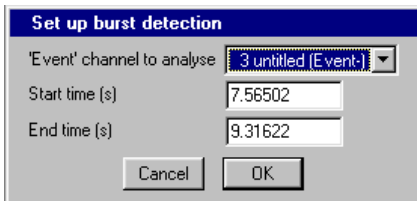
The *DlgDesign* script requires Signal version 3 or higher. It provides full control of the x- and y-coordinates of prompts and selection boxes. This allows you to create larger, more elaborate dialogs than can be produced using the *DlgMake* script. You can include up to 40 dialog items of each type, - reals, integers, checkboxes etc. and up to 200 text prompts.

Using the *DlgDesign* script, you start with an empty dialog box on the screen and add items to it by clicking on the appropriate toolbar button and entering the necessary parameters in a dialog. Then, you can move the new dialog item into the desired position interactively, by pressing the arrow keys on the cursor pad You can also edit or clear previously added dialog components and save the layout of the dialog for future use.

DlgMake.s2s

User guide

The following description refers to the script released with Signal version 3. Previous versions work in a similar way but lack some of the features mentioned below. We will use the script to create the 'Set up burst detection' dialog shown.

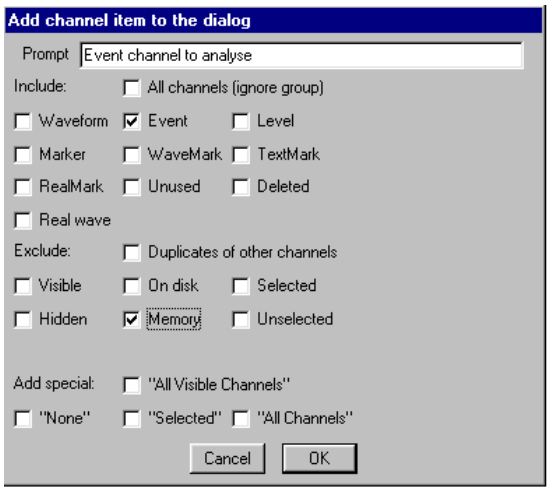


Run the *DlgMake* script using the *Load and run* option on the Signal script menu. Unwanted screen furniture such as the Signal toolbar and Status bar are hidden and an introductory message is displayed. When you close the Message box by clicking on OK, the script toolbar appears and the log window opens

If the script was used previously, the log window will contain a list of the contents of the last dialog that was created. To restart from scratch, click on the Clear button followed by Yes go ahead in the following *'Are you sure'* query. This clears all items except the OK and Cancel buttons.

Add Channel Item

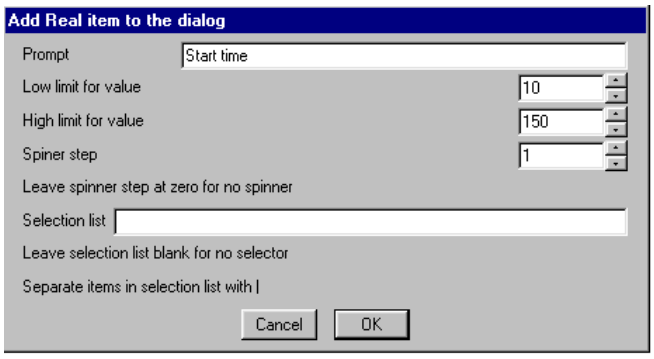
To create the example dialog shown above, click on the Channel button to add the first item. A dialog box opens for you to enter the prompt and you can select which combination of channel types to include in the list by checking the appropriate boxes. Press OK when ready. The new item will be added to the list in the log view.



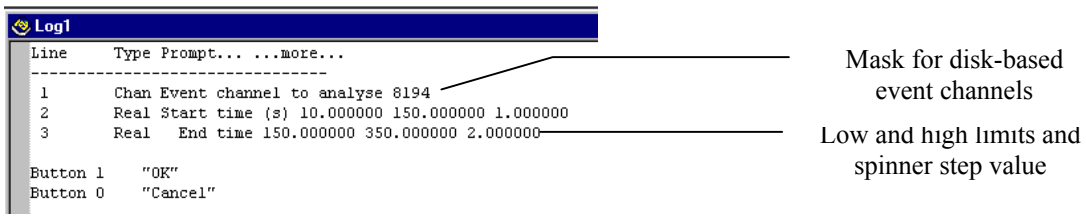
Add Real Item

Next, click on the Real button to add the 'Start time' item. A dialog opens for you to select the item number (and y-position) of the next item. The default value is the next line of the dialog. You are not allowed to set an item number higher than the default value. If you change it to a lower value, then the previously defined dialog items are automatically renumbered. Now press OK to continue.

In the next dialog, you enter the prompt for the real item and the upper and lower limits of the acceptable range of numbers. You can also enter a step size for a spinner or, define a list of values to appear in the Real selection box. Press OK when ready. Then click on the Real button again to add the 'End time' item.

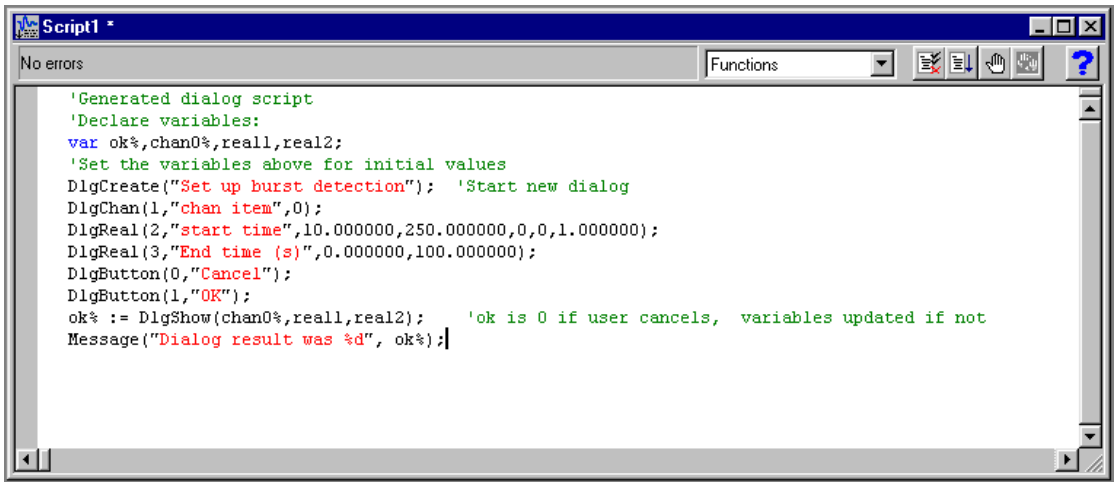


When you have finished the log view should look like this:



Now click on Output. A box will open for you to enter a title for the dialog. When you click on OK to close this box, a script holding the code required to generate your dialog is displayed. You must click on Quit to close the *DlgMake* script first, in order to run the new script and display the dialog. You can save this script in the usual way, or copy and paste it into the body of your own script. You will need to edit the dialog code to replace the variable names used in the DlgShow() function with those appropriate for your own script.

Script generated by DlgMake.s2s

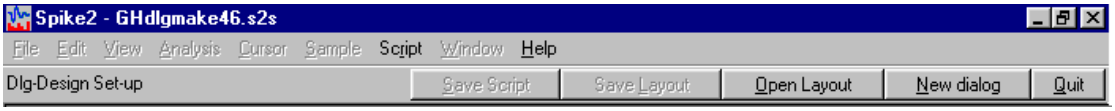


DlgDesign.s2s

User guide

Some advance planning is advisable before you start to create your dialog. Make a sketch of the dialog you want to create and annotate each item with all the relevant details, such as item type, -string, real, integer etc., the low and high limits of number ranges, the variable names to use for each item etc. This will reduce the time needed to edit the script later.

When you run the script, a toolbar with 5 buttons appears.



Quit closes the script and Open Layout gives you access to previously saved dialogs. However, to start from scratch, click on New Dialog. This opens the Set-up dialog.

Here, you enter the title for the new dialog and the name for the Dialog function. The script places most of the code for generating your dialog in a separate function with this name. If you plan to use more than one dialog created with *DlgDesign* in your final script, make sure that you give the dialog functions different names. . You can edit the title of the dialog and the name of the dialog function later, if necessary, by pressing the hotkey T (Title) when the main toolbar is showing.

Main toolbar

When you close the Set Up dialog by clicking on OK, a new toolbar with 15 buttons appears and the new dialog is shown in the top left-hand corner of the screen. Initially, this dialog will be empty apart from the Cancel and OK buttons.

Unlike normal script toolbars, clicking the buttons on this one has no effect. This is because the dialog under construction is open and it takes precedence over the toolbar. Thus, the toolbar buttons serve only as a guide to the currently available options. You *must* use keyboard hotkeys to drive this part of the script. Some of the hotkeys are shown underlined in the toolbar button labels.

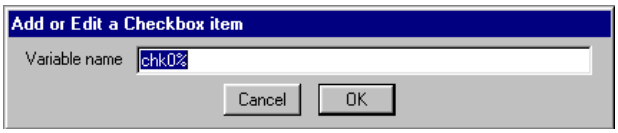
The hotkeys corresponding to the dialog positioning options: Down, Up, Left and Right are the arrow keys on the cursor pad. The shortcut keys for changing the size of the dialog box, Taller, Shorter, Narrow, Widen are the function keys F5 to F8. The Shift key toggles between coarse and fine adjustment of the position or size of the current dialog item. The adjustment defaults to the coarse setting initially. After moving an item to approximately the desired position, press Shift and then use the hotkeys to make fine adjustments.

As a first step, move the dialog to the desired screen position using the cursor arrow keys and set the dialog size using the function keys. The coordinates of the dialog will be displayed on the toolbar. The x and y values represent the position of the top left-hand corner of the dialog as a percentage of screen size. Dialog width and height are shown in dialog units, that is, in terms of the number of characters that would fit.

Add a New Item to the dialog

Next, click on A to add an item to the dialog. The Select Item toolbar will be displayed and the dialog is hidden. This toolbar has 11 buttons that can be operated in the usual way, that is, either by clicking on them or by pressing the appropriate hotkey.

Clicking on Back (shortcut *Enter*) returns you to the previous toolbar and re-displays the dialog. The remaining buttons allow you to select the type of item to add to the dialog. Clicking on one of these buttons opens a dialog for you to enter all the information needed to create that item.



For example, if you select CheckK, the only information required is the integer variable name to use for storing the state of the check box. The default name is chk followed by a serial number. When entering your own variable name, you don't need to add "%" as the final character. This will be added automatically if it is missing. Press Cancel to return to the Select Item toolbar or OK to add the check box to the dialog.

When you press OK, the dialog under construction is re-displayed with the new item, a checkbox appears near the OK and Cancel buttons. You can move it into the required position using the arrow keys on the cursor pad. Press the shift key to toggle between coarse and fine adjustments. The next step is to add a prompt for the checkbox. All prompts in dialogs created with the *DlgDesign* script are separate text items so that you can adjust their position independently of the selection box to which they refer. So, press A to add a new item followed by T for a text item. Next, enter the required prompt in the dialog box that opens and click on OK or press *Enter*. Once again, the new item appears at the bottom of the dialog and can be moved into position using the arrow keys.

Adding and positioning other types of dialog item works in the same way except that the dialogs for entering data relevant to the item vary. For example, those items that require an argument in the DlgShow() function prompt you to supply a name for that variable, Real and Integer items call for you to specify the upper and lower limits of the allowed range of numbers and the increment for the spinner control. Button items require a button label and a name for the associated function and so on. As well as moving items into position, you can change the width of selection boxes and the width and height of group boxes using the function keys, F5 to F8. The default values for width of a selection box, the upper and lower limits for numbers and the increment for spinner controls are based on those used in the previously added dialog item. You can add up to 40 dialog items of each type, up to 20 buttons and up to 200 text prompts. If you should reach the upper limit for a given type, the associated toolbar button will be disabled.

Modify OK and Cancel buttons

All dialogs must have a button that, when pressed, closes the dialog box and updates the values of all the variables in the DlgShow() list. By default, this button is labeled OK. However, if you wish to change the button label, - to Go or Close for example, or if you want to move it from its default position, then select the OK option (press K). Enter a new button label in the dialog box, if required,

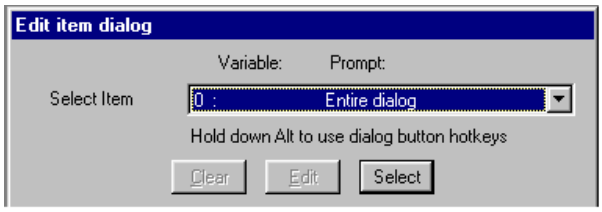
and then press the Select button. This will make the OK button the current item so that it can be moved using the cursor keys. If you move the button to the left edge of the dialog it will automatically jump back to its default position.

Not all dialogs need a Cancel button. To hide it or move it from its default position, select the Cancel option (press C). Check the box in the dialog followed by OK to hide the button. Alternatively, just press OK to make the Cancel button the current item so that you can reposition it using the cursor arrow keys.

Edit the dialog

You can make alterations to existing items in the dialog under construction using the Edit option (press E). In the dialog that opens, you can select the item to work on from a numbered drop-down list.

This list includes the dialog box itself and all the components added so far, identified either by variable name or a text label. After selecting the item to work on, click one of the dialog buttons or hold down Alt and press the underlined shortcut key. Choose Select (*Enter*) to make the chosen item current so that it can be moved using the arrow keys or press Clear to remove the selected item from the dialog altogether. Click on Edit to modify the parameters of the selected item. A dialog will open, showing the current values as defaults, so that you can update them.



Discard the dialog

You can press on D to discard the current dialog whenever the main toolbar is visible. There is an *'Are you sure'* query in case you pressed this button inadvertently. If you proceed to discard the current dialog, you can use the Set-up toolbar buttons to Quit, or start again by selecting New Dialog or Open Layout.

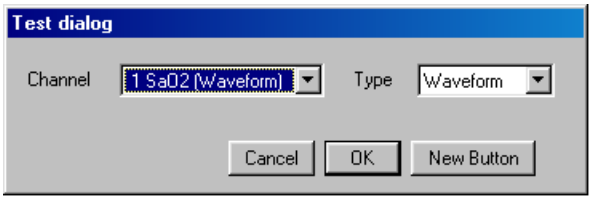
Generate the script code

When you are satisfied with the appearance of your dialog, press O to select the Output option. The dialog closes and is replaced by a script window containing the code required to generate your dialog. The Set-Up toolbar reappears. If you wish, you can click on the Save script button to store the file to disk. You can also save the layout of the dialog as a binary file by clicking on Save Layout. The advantage of saving the layout is that you can quickly re-create the dialog in future by re-loading the binary file using the Open Layout button. You can then modify the dialog by adding more items or modifying existing ones. It is usually much quicker to update a dialog by editing a previous version rather than starting afresh.

Next, press Quit to close the *DlgDesign* script. The script for your dialog will remain open on the desktop. The next step is to compile and run this script. If it was not previously saved and the *'Save modified scripts before they run'* option in the Signal Preferences was set, then a *Save As* dialog will open to give you a further opportunity to save the script. Otherwise, the script will run immediately and the dialog that it creates will be displayed.

Example dialog and corresponding script

A very simple dialog and the corresponding script are shown below. Clearly, the code is somewhat longer and more complex than would be required if you were creating the dialog in the usual way. This is the price you pay for the relative ease of composing dialogs with the *DlgDesign* script. Fortunately, it isn't necessary to understand the underlying code in great detail. Simply, adhere to the following guidelines for building the dialog into your script.




```
'global variables
     var okk%,ng%[30],nt%[200];                                          1
'    user-defined global variables:
     var chan0%,dlistl%,dummy;

'Code for inserting a dialog-----------------------------------------------
     okk%:=MyDialog();
'--------------------------------------------------------------------------  2

'Function to generate the dialog
     func MyDialog();
     ArrConst(ng%[],0);  ArrConst(nt%[],0);'clear previous group and button item numbers
     var bh%,vh%,nn%,chk%[30],int%[30],real[30],chan%[30],dlist%[30],dstr$[30],xval[30];
     vh%:=View();'remember current view
     bh%:=FileOpen("Dlg.dat",9,1);
     ' copy user-defined variables to the dialog
     BWrite(chan0%,dlistl%);                                             3
     Bseek(0);
     BRead(chan%[:1],dlist%[:1]);
     Bseek(0);
     View(vh%);
     var i%:=1,okk%,dum,noftype%[10],dttl$:="Test dialog",Cancel$:="Cancel";
     noftype%[3]:=1;noftype%[4]:=1;noftype%[8]:=2;noftype%[9]:=1;
     var x[14],y[14],lo[14],hi[14],wi[14],list$[14],sp[14],OK$:="OK";

     x[0]:=10.1; y[0]:=12.0; wi[0]:=60.0; hi[0]:= 4.0;
     x[4]:=13.0; y[4]:= 1.5; wi[4]:=20.0; hi[4]:= 1.0;'      item 4: chan0%
     x[5]:=44.0; y[5]:= 1.5; wi[5]:=13.0; lo[5]:= 3.0; hi[5]:= 2.5;list$[5]:="Waveform |Marker| etc.";'item 5: dlistl%
     x[9]:= 2.0; y[9]:= 1.5; list$[9]:="Channel";
     x[10]:=36.0; y[10]:= 1.5; list$[10]:="Type";
     x[11]:=42.0; y[11]:= 3.5; lo[11]:= 2.0;list$[11]:="New Button";
     x[12]:=33.0; y[12]:= 3.5;'OK button
     x[13]:=23.0; y[13]:= 3.5;'Cancel button

     DlgCreate(dttl$,x[0],y[0],wi[0],hi[0]);
'    DlgAllow(allow%);'optional allow/idle/change functions
     i%+=3;
     for i%:=i% to i%+noftype%[3]-1 do'channels
          DlgChan(i%,wi[i%],lo[i%],x[i%],y[i%]);
     next;
     for i%:=i% to i%+noftype%[4]-1 do'list items
          DlgList(i%,wi[i%],list$[i%],lo[i%],x[i%],y[i%]);
     next;
     i%+=3;
     nn%:=0;
     for i%:=i% to i%+noftype%[8]-1 do'Text items
          nt%[nn%]:=DlgText(list$[i%],x[i%],y[i%]);
          nn%+=1;
     next;
     DlgButton(lo[i%],list$[i%],newbtn%,x[i%],y[i%]);
     i%+=1;
     DlgButton(1,OK$,0,x[i%],y[i%]);'OK button
     i%+=1;
     DlgButton(0,Cancel$,0,x[i%],y[i%]);'Cancel' button
     okk%:=DlgShow(dum,dum,dum,chan%[:1],dlist%[:1]);
     if okk% then
          View(bh%);
          BWrite(chan%[:1],dlist%[:1]);
          Bseek(0);
          ' update user-defined variables
          BRead(chan0%,dlistl%);
     endif;
     View(bh%);
     FileClose();
     View(vh%);'back to the original view
     return okk%;
     end;
'--------------------------------------------------------------------------
     func newbtn%();
     'Your code here                                                     4
     return 1;
     end;
```

Embedding the dialog code in your script

The dialog code generated by *DlgDesign* consists of 4 sections demarcated with dashed lines. Section 1 contains global variables. You need to cut and paste the first row of variables in at the beginning of any script containing one or more dialogs created with the *DlgDesign* script. Do not use these variables in your own script code. The remaining variables are the user-defined variables that will be modified by the dialog. They must be declared as global variables. Paste them into the global variables section of your script unless you have already declared them.

Section 2 contains the call to the function that actually generates the dialog together with comments indicating the names of the variables that will be set by the dialog. The comments are omitted from the example to save space. Cut and paste the function call into your script at the point where you want the dialog to appear. Like DlgShow(), this function will display the dialog and return 0 if the Cancel button was pressed and 1 if OK was pressed.

Section 3 contains the function that makes the dialog. Paste this section into the procedures and functions section of your script, -that is, anywhere after the halt instruction of the main program.

Section 4 holds the functions that will be executed if a user-defined dialog button is pressed. Simply insert the code to execute when the button is pressed into the body of each function and paste them into your script along with all the other functions and procedures.

Call-back functions

In Signal version 3 you can allow the user to access the drop-down menus and to drag cursors while a dialog is open. You can also use call-back functions to access the contents of an open dialog. For example, you can read from or write to a dialog selection box using DlgValue(), or show and hide a dialog item using DlgVisible(). Full instructions for using call-back functions are contained in the Signal *Help* index. You can make use of these facilities in dialogs created with *DlgDesign*.

In order to allow access to Signal menus while your dialog is open, simply reinstate the DlgAllow() function in section 3 of the dialog code by deleting the apostrophe. Replace allow% with the mask for the particular combination of menus that you want to access; see the *Help* menu under Interact() for details. If you require a dialog Idle() routine and/or a dialog Change() function, then enter the names of the functions here.

In order to use functions such as DlgValue(), DlgVisible() and DlgEnable(), in Idle, Change or Button functions, you need to know the item number(s) of the dialog elements you want to act on. You can get this information by inspecting the code of the function that generates the dialog.

The region of interest lies between the last var statement and the DlgCreate() function. Each line begins with x[and contains information related to a single dialog item, that is x and y coordinates (x[], y[]), low and high limits (lo[], hi[]) width of the item (wi[]) and text component, if any (list$[]). The first line beginning x[0] refers to the position and size of the dialog itself. The item number and variable name are shown in the comment at the end of each line. Text prompts, group boxes and user defined buttons are listed after the items with an argument in DlgShow(). You will be able to recognize them by the text in the list$[] variable. The item numbers for group boxes are stored in the global array ng%[]. The item number of the first group box in the list is ng%[0], the second is nt%[1] and so on. Similarly the item number for the first text item is ng%[0].

The definitions of additional buttons come next. They should be easily recognizable from the button labels at the right of the line The button number of user defined buttons is stored in the variable lo[]. If the Cancel or OK buttons in the dialog are non-standard, the final lines beginning x[...] define their characteristics. The item number of a button for use in call-back functions is *minus n* where n is the button number, (e.g. –2).

Finally, if you have defined spinner controls for any dialog items, the increment values are listed in the array sp[]. The array indices indicate the item number of the dialog elements to which they refer.

Modify the layout of a dialog

You can make minor modifications to the layout of your dialog by editing the list of parameters above the DlgCreate() statement. For example, if you decided to increase the width of the dialog box in the example script, you could change the value of wi[0] from 60 to 65. Alternatively, if you wanted to move New button to the left of Cancel, you could change x[] from 42 to 10. However, should you contemplate more extensive changes, it is much easier modify a previously stored layout. Run the *DlgDesign* script, click on the Open Layout button and load the layout file for the dialog, This is a binary file with the *.dat*. extension. You can then add more items and move, edit or clear existing ones to create the new dialog. Remember to save the layout when you are finished so that if further changes are needed later, you can build on the existing dialog rather than starting again from scratch!