SMT9650
USER GUIDE

Note:

If this copy is no longer in use, return to sender.

APPROVAL PAGE

Name	Signature	Date

AUTHOR/S

Name	Signature	Date
Peter Robertson		4 Dec 2006

DOCUMENT HISTORY

Date	Initials	Revision	Description of change
6 Dec 2006	PSR	0.1	Initial Draft
22 Dec 2006	PSR	1.0	Various corrections and extra pictures added
26 Dec 2006	PSR	2.0	Updated and extra demo description added
29 Dec 2006	PSR	3.0	Revised extra demos

TABLE OF CONTENTS

1 INTRODUCTION

The SMT9650 is a software package that allows Labview to be used with code executing on hardware made up from combinations of DSP and FPGA modules. The code is generated from Simulink models using **PARS**, a development tool that uses **3L Diamond** to create multiprocessor applications.

2 PREREQUISITES

You need the following software to have been installed before you can use the SMT9650:

- Labview (version 8.0 or later)
- Matlab & Simulink
- PARS
- 3L Diamond DSP & 3L Diamond FPGA (V3.1.4 or later)

This document assumes that you are familiar with these software components.

3 OVERVIEW

An application built with the SMT9650 comprises two parts: a Labview front-end running on a PC and a DSP application[1] running in attached hardware.

The Labview front-end uses custom Virtual Instruments (VIs) to load the DSP application and communicate with it. The nature of the communication between the PC and the attached hardware is invisible to the user; the various combinations permitted by Sundance hardware (host comport or USB connection, for example) are managed by the underlying systems software.

[1]To keep the text simple, this document will use the term "DSP application" to mean an application suitable for a multiprocessor system comprising both DSP and FPGA components.

4 CONSTRAINTS

Labview and Simulink have different models of computation, and two constraints are necessary in order to allow them to cooperate.

1.1 SIZE CONSTRAINT

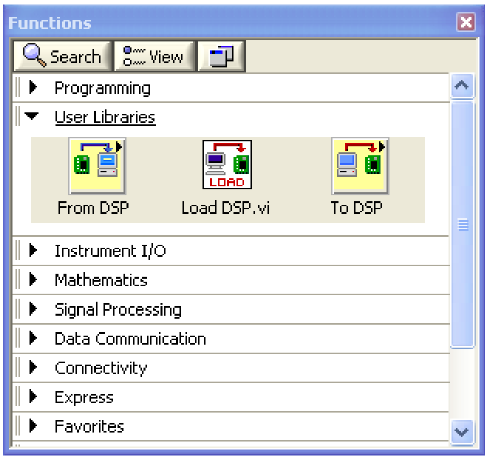
The size of arrays handled by Labview VIs is determined by the source of the data; for the input instruments this will be Simulink (from the DSPs) and for the output instrument it will be Labview. When running a system entirely within either the Simulink or Labview environments, matching of data sizes is mostly done for you. In the hybrid environment of the SMT9650, you must make sure that array sizes match the requirements of the data destination manually.

1.2 ORDERING CONSTRAINT

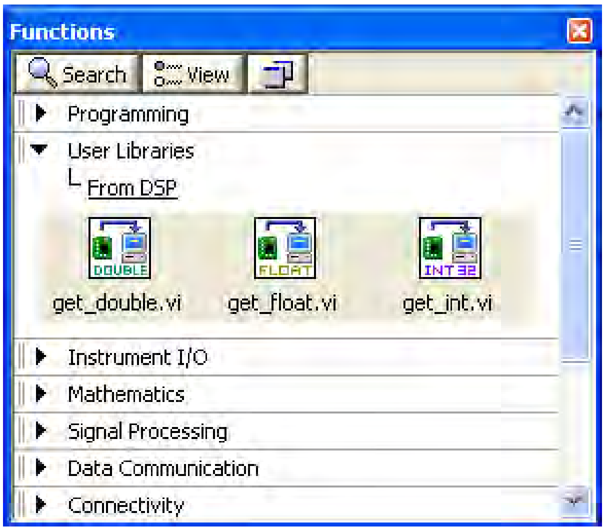
The code generated by Simulink imposes a strict ordering constraint on the application; all input sources will be read before all output sinks are written. This means that the loop in Labview communicating with the DSP system must ensure that all the SMT9650 output VIs are executed before all of the input VIs. The simplest way to achieve this is to use a flat sequence and make sure that all your output VIs appear in one frame and all the input VIs appear in the next.

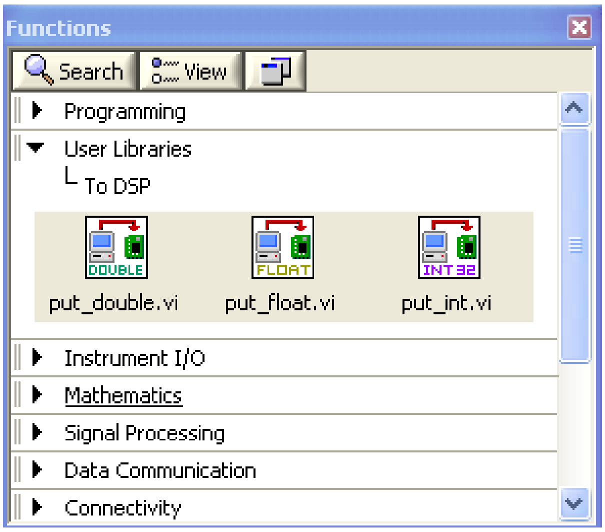
5 LABVIEW INTERFACE

The Labview interface is presented as seven virtual instruments: one to load the Sundance hardware, three to get data from the DSP application, and three to send data to the DSP application. These instruments appear as **User Libraries** in the Labview **Functions** palette. You can display this palette by going to the **Block Diagram** window and selecting **`View/Functions Palette`**.



`From DSP` is a folder with the three input instruments and **`To DSP`** is a folder with the three output instruments. **`LOAD DSP.vi`** is the instrument used to send an application to the Sundance hardware.





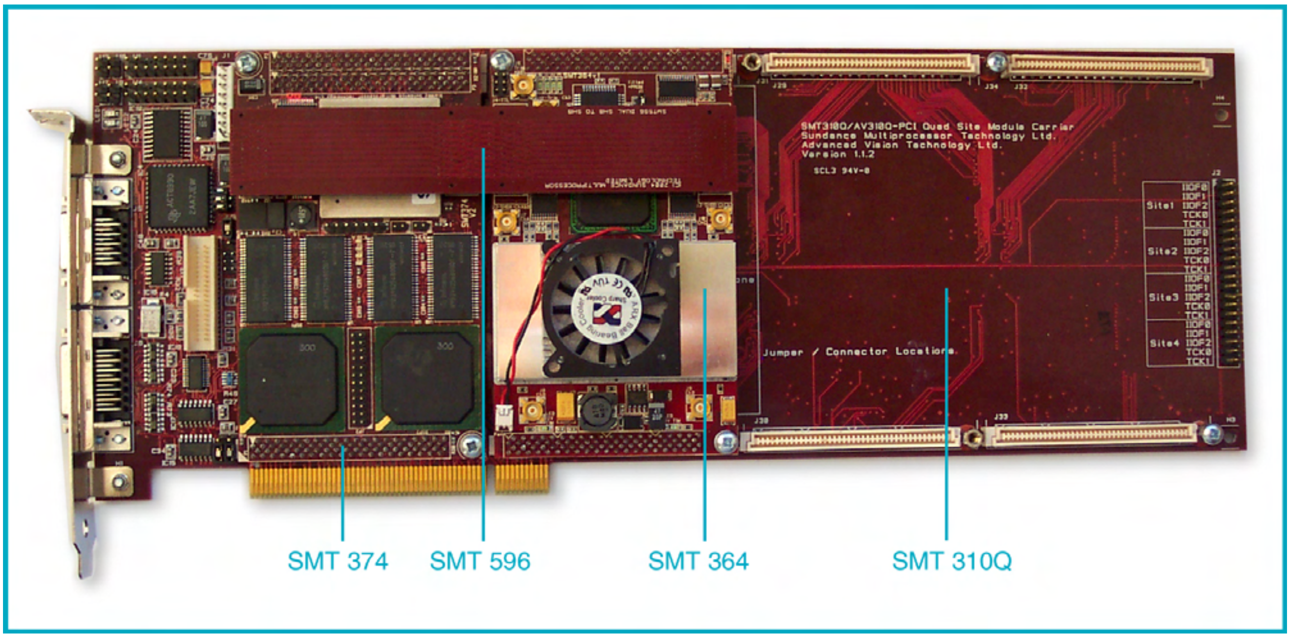
The "**`get_...`**" instruments receive data from the DSP application in the form of arrays of values of the selected type; the size of array is determined by the DSP. For example, **`get_int.vi`** returns an array of 32-bit integers.

The "**`put_...`**" instruments send arrays of the selected type to the DSP application. For example, **`put_float.vi`** sends arrays of 32-bit floating point values. The size of array transmitted must match the expectation of the receiving DSP code.

6 DEMONSTRATION 1

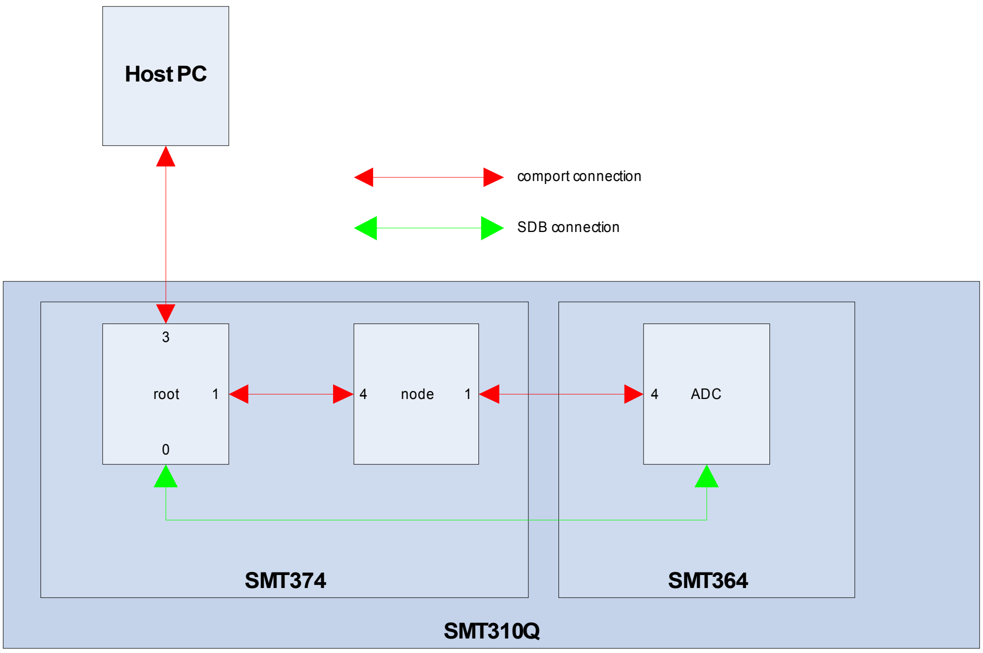
This demonstration shows how a simple DSP application generated using PARS can be combined with a front-end developed under Labview. TIt can be found in the `\Demos\Demo1\` sub-folder of the SMT9650 installation folder; this will be `C:\SundanceDSP\SMT9650\Demos\Demo1\` by default.

The hardware needed for this demonstration is pictured below.

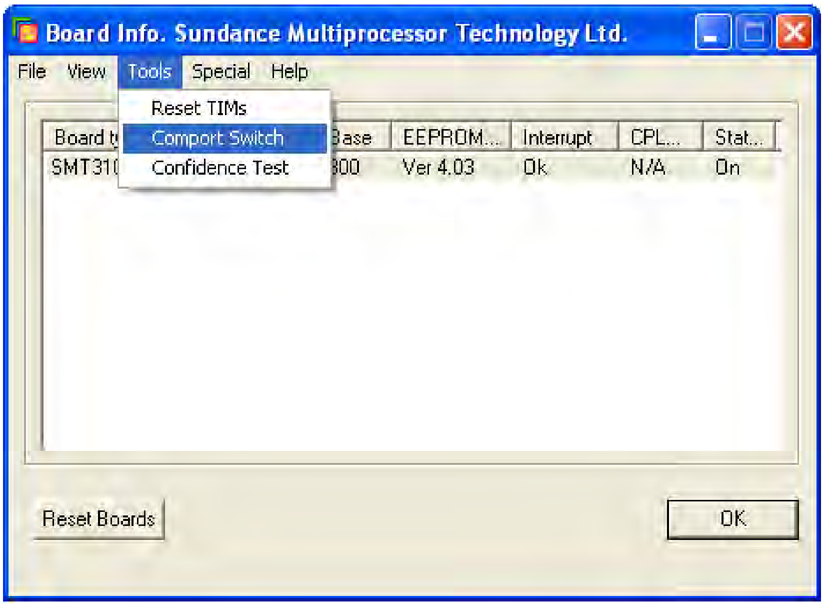


There are two Sundance TIMs mounted on an SMT310Q carrier board: an SMT374 dual C6713 DSP module (root & node) and an SMT364 4-channel ADC module. Data are carried between these modules over comports and SDBs. The comport connections are built into the carrier, and an SMT596 SHB connector is used to join the SDBs of the two modules.

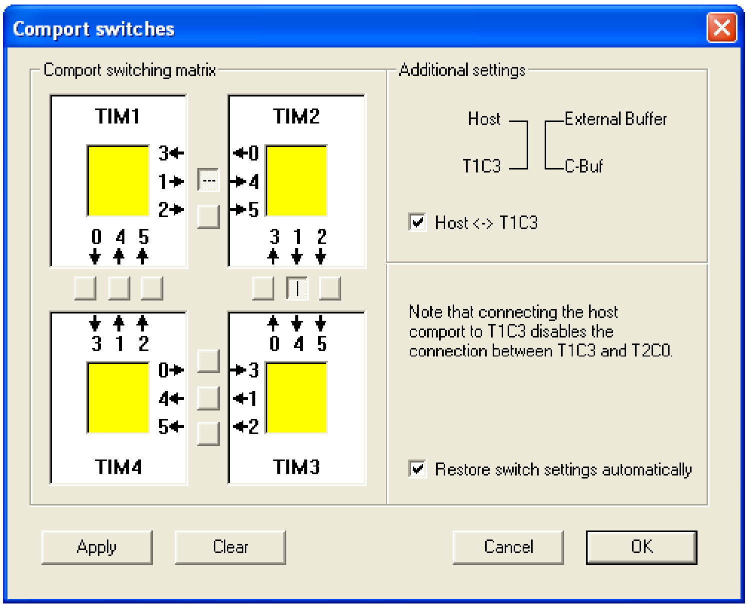
The logical connections are shown in the following diagram:



The on-board comport connection between the STM374 and the SMT364 (CP:1 to CP:4) is enabled using the Sundance **BoardInfo** utility:



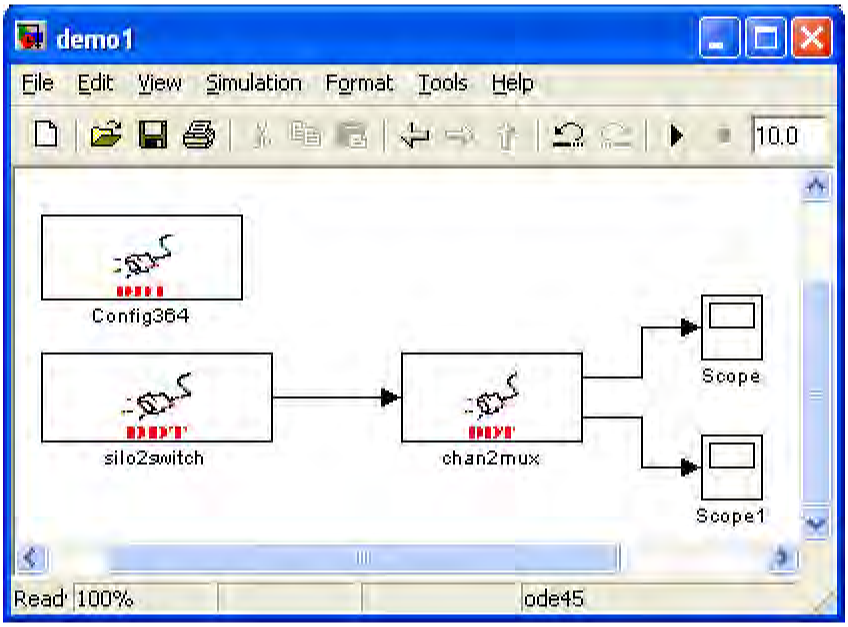
Select **Tools** and then **Comport Switch**. This will bring up the following display:



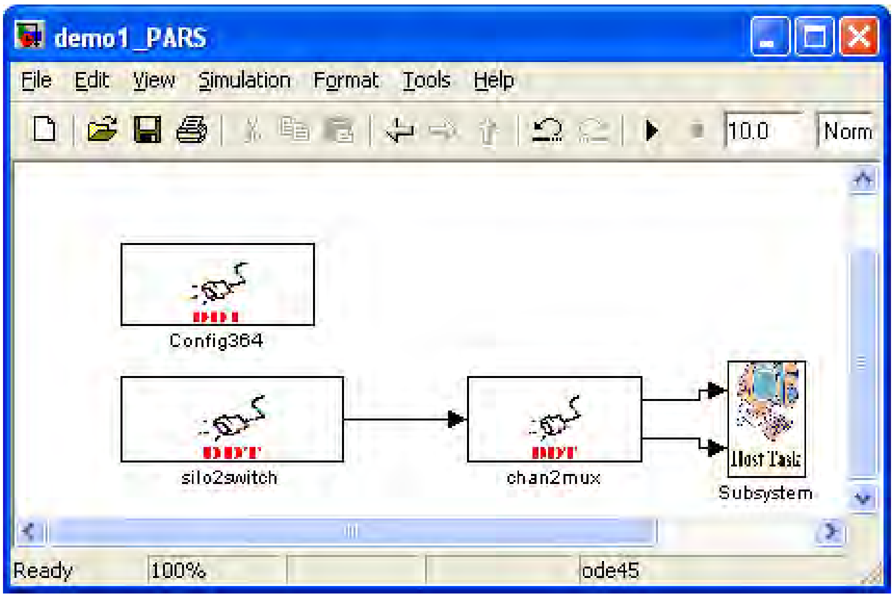
Click the square between TIM1 comport 1 and TIM2 comport 4 to make it display as **...** as in the picture above, if necessary. This enables the required comport connection. Click **OK** and then **OK** again.

The demonstration can be tailored to run on different combinations of hardware modules and carrier boards by changing parameters in the model, as described in the PARS documentation.

1.3 SIMULINK MODEL



1.4 PARS MODEL



The demonstration is an **SMT364** ADC TIM communicating with an **SMT374** TIM running three user tasks on two DSPs. **`Config364`** runs on the node DSP and starts the ADC running as a counter. The remaining tasks run on the root DSP: **`Silo2switch`** gathers the ADC data and passes them to **`chan2mux`** which splits out two channels of data. These two channels are then passed to the **`Host Task`** which uses **Labview** to display the data.

1.5 THE PRE-BUILT TASKS

The demonstration uses three PARS pre-built (PB) tasks that have been developed to provide access to the SMT364 ADC: config364, silo2switch, and chan2mux.

1.5.1. Config364

This PB task configures an SMT364 ADC board. For more information about the SMT364, please refer to its user manual. This PB task is a wrapper around the Config364 tasks shipped with the SMT6045. For more information about how the task works and its parameters, please refer to the SMT6045 documentation. This PB tasks takes the following parameters:

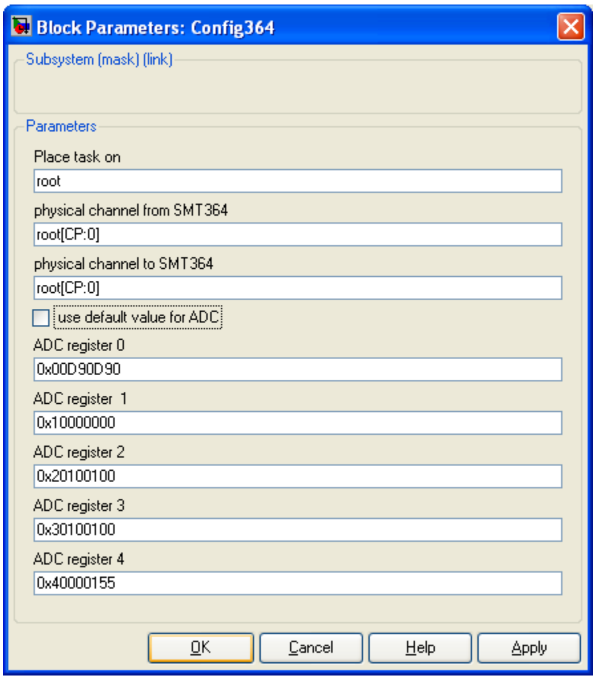


Figure 1: Parameters window for Config364 task.

- **Place task on**: specify which processor this task should be placed on.
- **Physical channel from SMT364**. Define the physical channel connecting the SMT364 to the processing board.
- **Physical channel to SMT364**. Define the physical channel connecting the SMT364 to the processing board. Normally this is the same as the previous parameter.
- **Use default values for ADC**. If selected, any parameters entered for ADC registers are ignored and the default values will be used.
- **ADC registers** 0 to 4: values for ADC registers. Please refer to the SMT364 User Guide for an explanation of these registers.

1.5.2. silo2switch

This is a wrapper for a task with the same name from the SMT6045. It is a task that reads data from a physical channel and buffers them and allows a slow data sink to work with an SMT364 ADC.

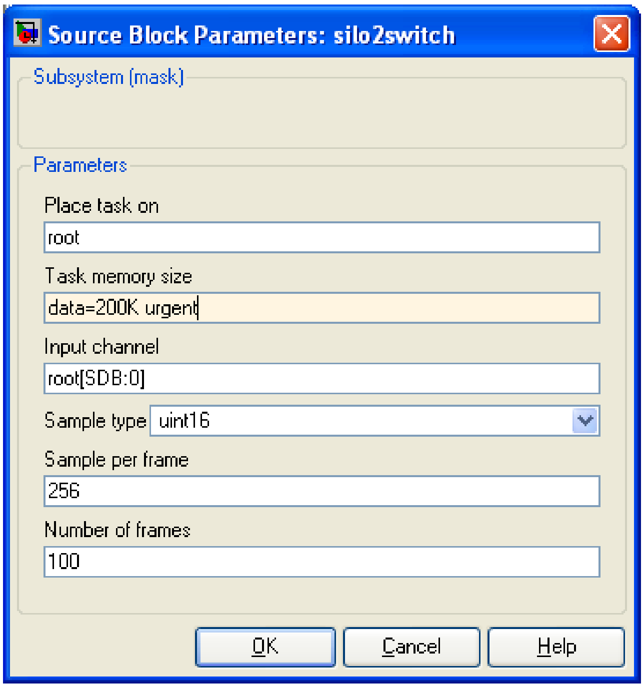


Figure 2: Parameter window for silo2switch task

- **Place task on**: specify which processor this task should be placed on.
- **Task memory size**: the amount of memory that will be assigned to this task for stack and heap. Since this task needs a lot of memory to buffer input data, you need to make sure that enough memory is assigned to this task.
- **Input channel:** Physical channel that connects to the ADC board.
- **Sample type**: this depends on the ADC settings and could be 16-bit integer or 16-bit unsigned integer.
- **Sample per frame**: the number of samples in one frame.
- **Number of Frames**: the number of frames that the task allocates for buffering.

1.5.3. chan2mux

This is a wrapper for a task with the same name from the SMT6045. This task multiplexes input data into several outputs.

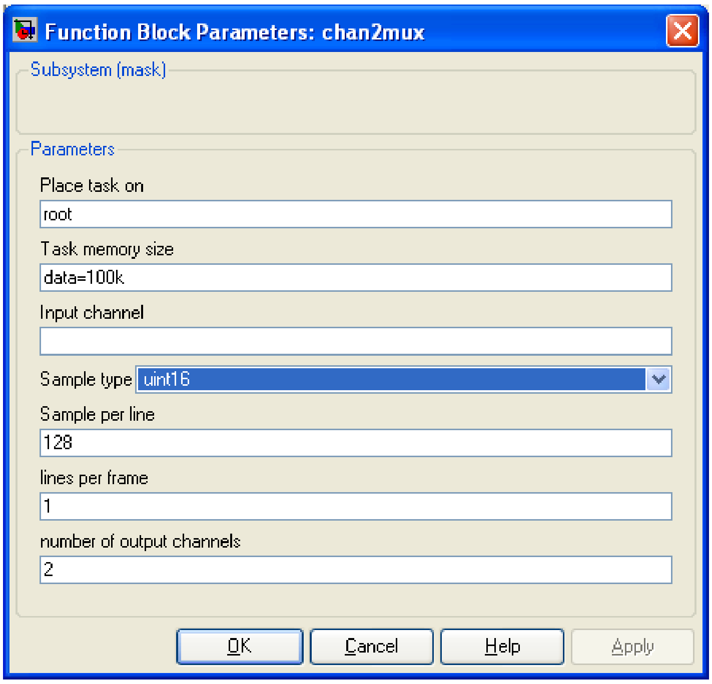


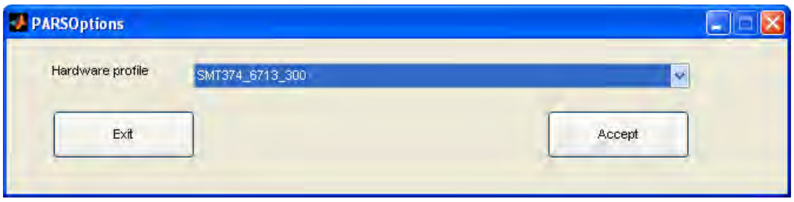
Figure 3: Parameter window for chan2mux task.

- **Place task on**: specify the processor on which this task should be placed.
- **Task memory size**: the amount of memory that will be assigned for this task's stack and heap. Since this task allocates a lot of memory to buffer input data, you need to make sure that enough memory is assigned.
- **Input channel**: If a value is entered in this field, PARS will use it as an input for task (the Simulink model is changed and the input will disappear). If no value is entered in this field, then a virtual channel is assumed for input.
- **Sample per line**: the number of samples in each line.
- **Lines per frame**: the number of lines in each frame.
- **Number of output channels**: the number of output channels into which the input will be divided.

1.6 BUILDING THE DEMONSTRATION

You should start by reading the PARS documentation to understand the build process.

1. Open Matlab and give the command **`PARSOptions`** at the **>>** prompt, then use the resulting dialogue to select and then **`Accept`** your DSP hardware. In this example it will be **`SMT374_6713_300`**.



2. Give the command **`PARS`** at the **>>** prompt.

3. In the **`PARS`** window, click on **>>** and browse to the demonstration model **`demo1.mdl`**.

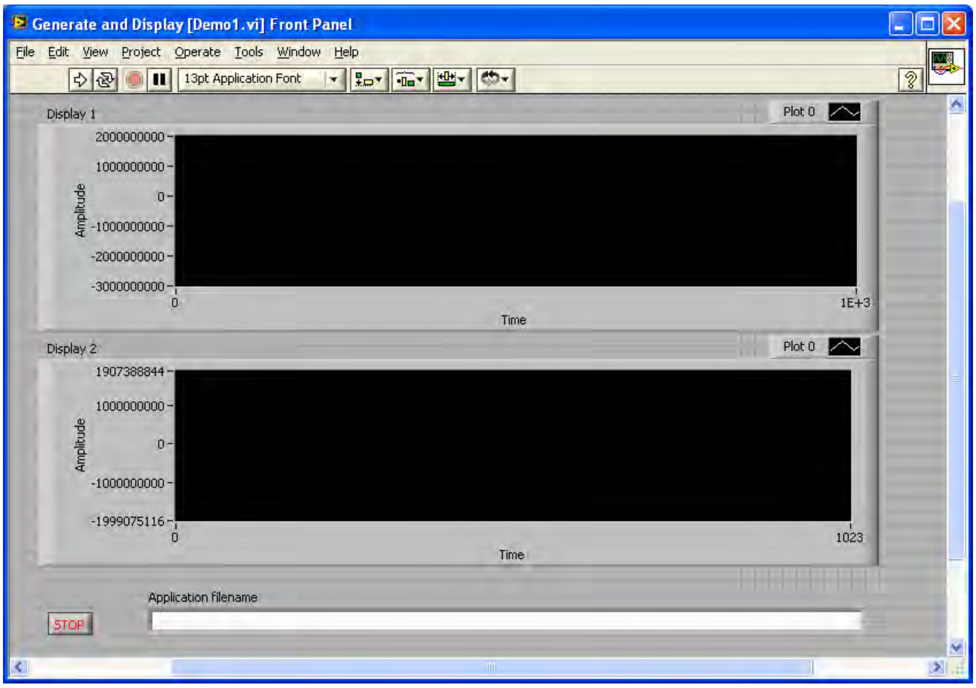
4. Select the file, click **`Open`**, and then click the **`PARS`** command **`Open model`**. The should open a **`demo1_PARS`** window.

5. Click Generate in the **`PARS`** window. This should start the build process which will continue with activity in several windows. Eventually it should finish and report "**System generation successfully finished**". Click **`OK`** to dismiss the report.

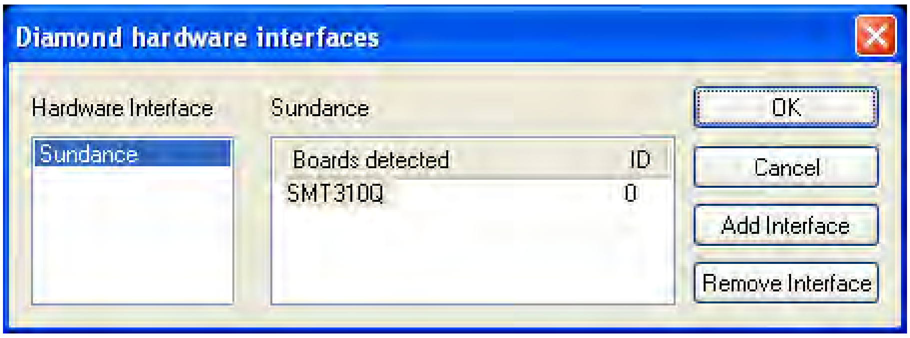
6. The file **`demo1_PARS.app`** should have been created in a new sub folder, **`demo1_PARS`**, of the folder containing the original model.

1.7 RUNNING THE DEMONSTRATION

Start Labview and select **`File/Open`**, then browse to the folder containing the demonstration VI, **`Demo1.vi`**. Double click on **`Demo1.vi`** to open it. This should bring up the following window:

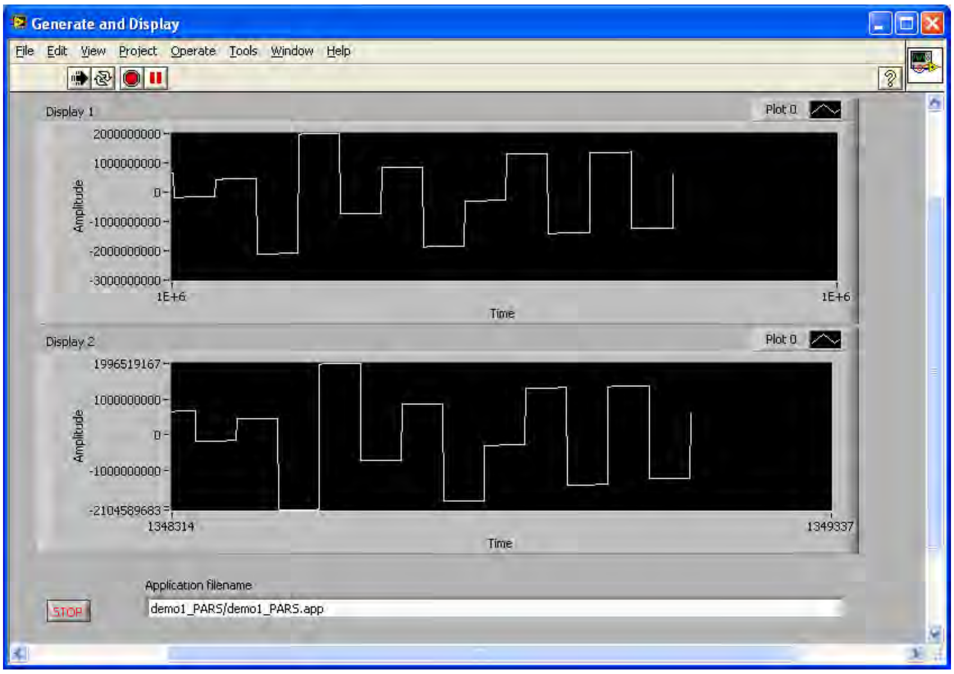


Type in the name of the PARS application you wish to execute. The filename you supply here can be relative to the directory containing the VI itself. For this demonstration you can use **demo1_PARS/demo1_PARS.app**. Now click the run button, the arrow below and to the left of **View**. This will start the demonstration running and bring up the following dialogue:



PARS has been used to generate a Diamond application that contains all the code for the DSP part of the demonstration, and Labview is attempting to load that application. To do this, it needs to be told which DSP hardware to use. In this case there is only one recognised Sundance carrier board in the system, so click **OK** to select it.

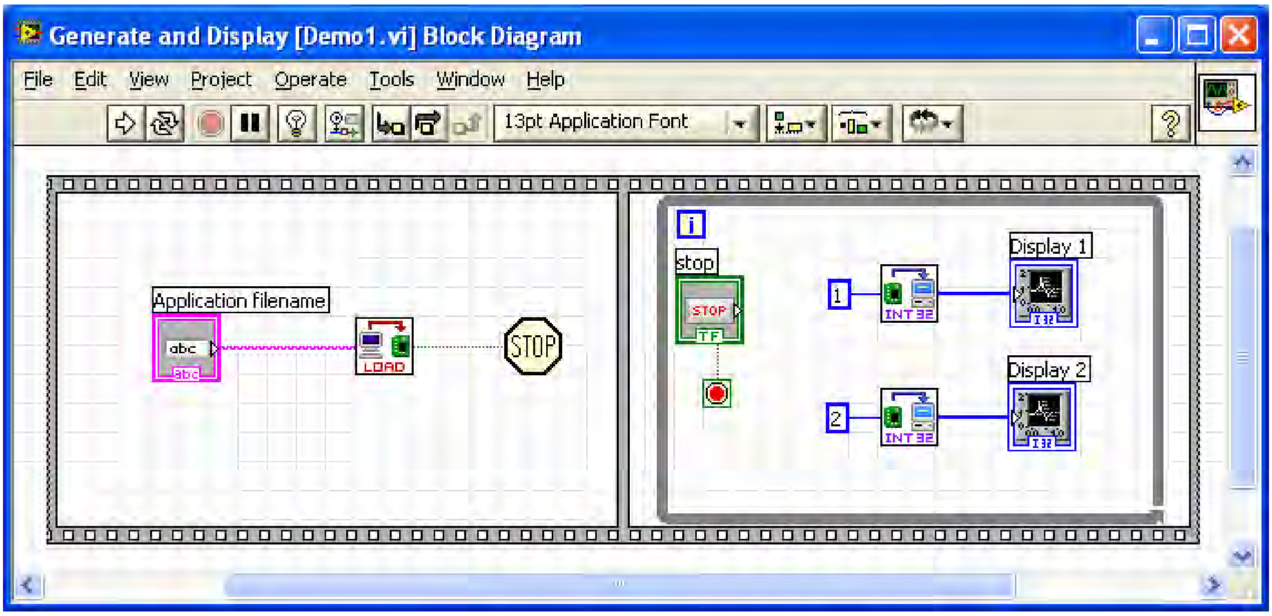
The demonstration should now load and start to execute:



You can stop the demonstration by pressing the **STOP** button.

1.8 INTERNAL DETAILS

You can investigate the internal details of the demonstration by going to the front panel **Window** menu and opening the block diagram (**Show Block Diagram**):



You will see that the diagram is made up from two frames which will execute from left to right. The left frame is responsible for loading the application into the DSP. It uses the **LOAD** instrument which is given the name of the application. The only output from **LOAD** is a boolean signal that is true should an error occur while sending the application to the Sundance hardware; this will cause the demonstration to stop.

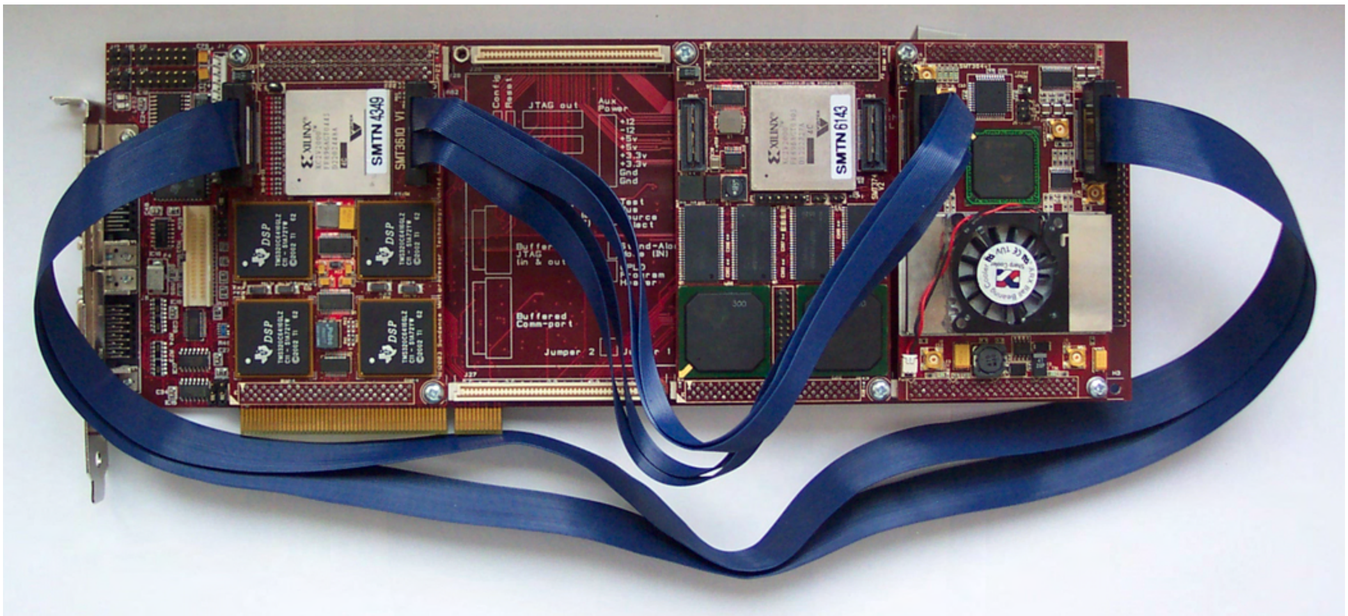
The right frame is the body of the demonstration. It is a loop that continually requests two streams of data from the DSPs and displays them. Each stream is made up of arrays of integers brought in through the **INT32** instruments. Each of these instruments is given an integer that selects the number of the DSP's *output port* that has been specified in PARS as presenting the data. In this example the DSPs use only two output ports, 1 and 2. The output from the **INT32** instruments will be arrays of integers. The frame also implements the **STOP** button.

7 DEMONSTRATION 2

This demonstration can be found in the `\Demos\Demo2\` sub-folder of the SMT9650 installation folder; this will be `C:\SundanceDSP\SMT9650\Demos\Demo2\` by default. It takes four channels of data from the SMT364 and passes them through SDBs to an SMT361Q; an SMT374 is used to configure the SMT364. The data values are scaled by different constant amounts and then displayed.

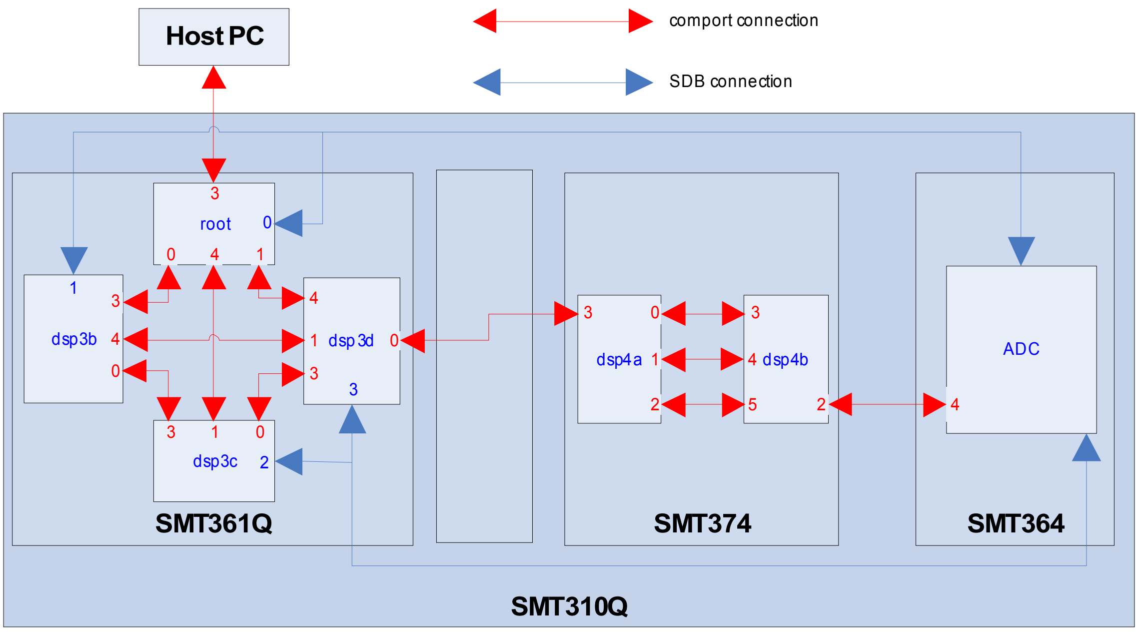
There are two variants of this demonstration: one using an SMT310Q carrier and one using an SMT148.

The SMT310Q hardware for this demonstration is shown in the picture below:

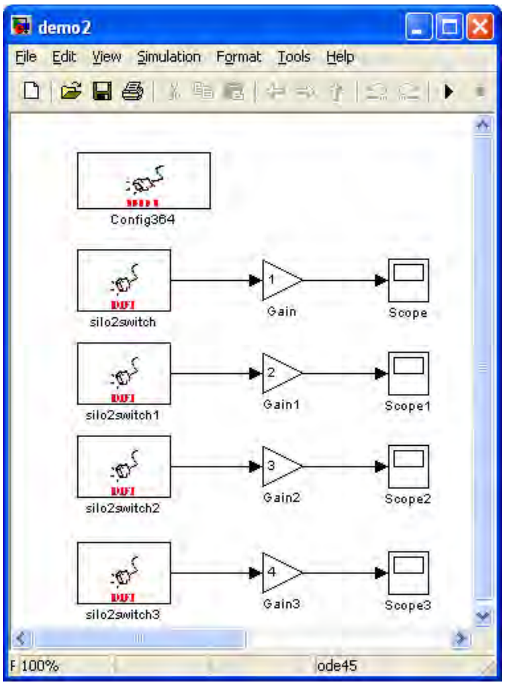


There are three Sundance TIMs mounted on an SMT310Q carrier board: a 4-processor SMT361Q, an SMT374 dual C6713 DSP module, and an SMT364 4-channel ADC module; slot two is unused. Data are carried between these modules over comports and SDBs. The comport connections are either built into the modules or are realised using FMS cables on the back of the carrier. An SMT596 SHB connector is used to join the SDBs using the SHB connectors on the SMT374 and the SMT3634; individual SHB cables can also be used.

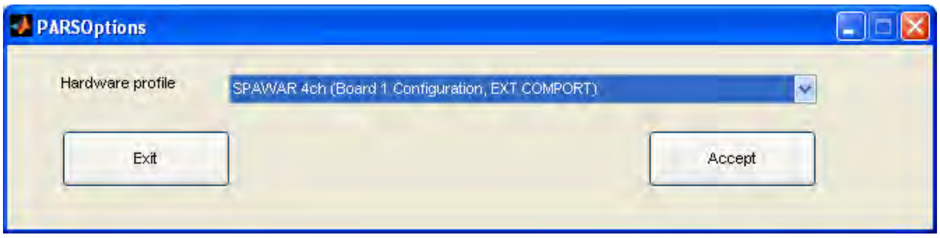
The logical connections are shown in the following diagram:



The application in this demonstration configures the SMT364 to generate four streams of 16-bit data on the four SDB outputs of the ADC. These streams are passed to the simple tasks on each of the four SMT361Q processors to be multiplied by different factors. Finally, the results are sent to Labview to be displayed.

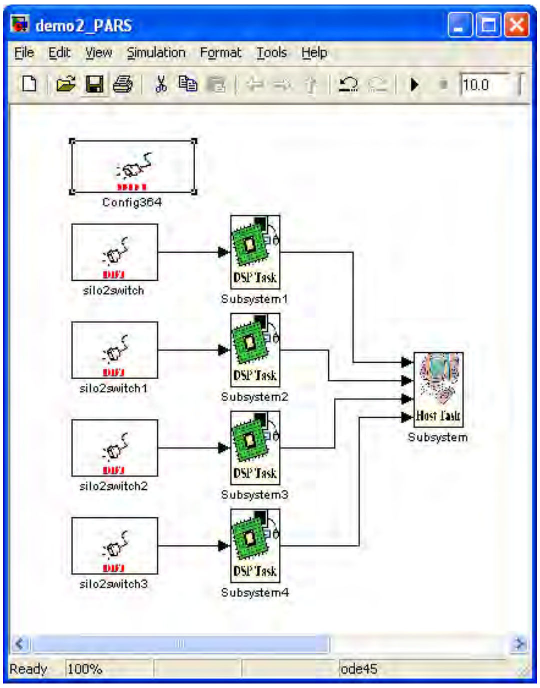


For the SMT310Q variant, PARSOptions should be set to **`SPAWAR 4ch (Board 1 Configuration, EXT COMPORT)`**.

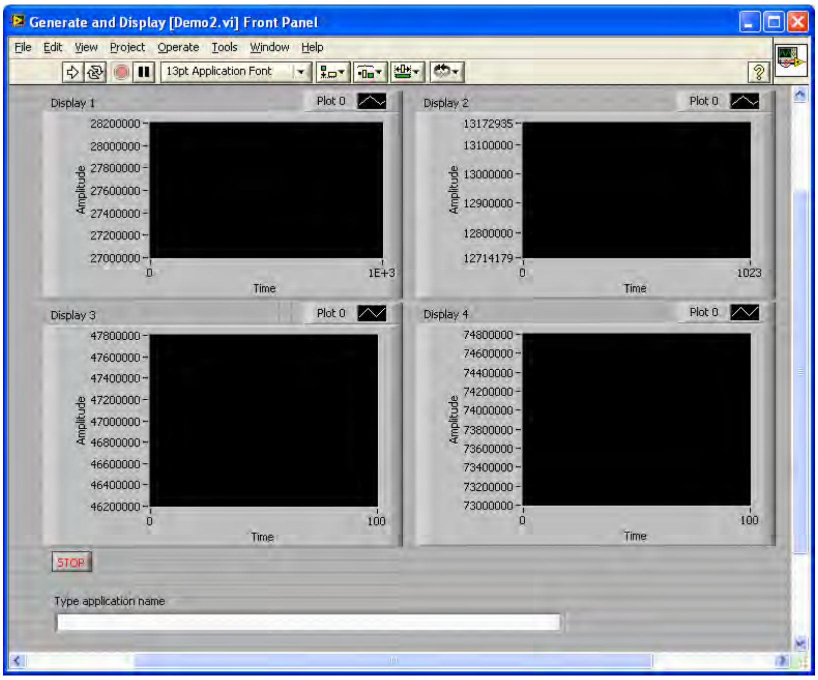


The files for this demonstration can be found in **`demos/demo2/SMT310Q`**.

The Simulink model above is converted into the PARS model below, with the **silo2switch** tasks placed on each processor of the SMT361Q, along with their corresponding "Gain" tasks. The configuration of the pre-built tasks **silo2switch** and **Config364** have been described for the previous demonstration.

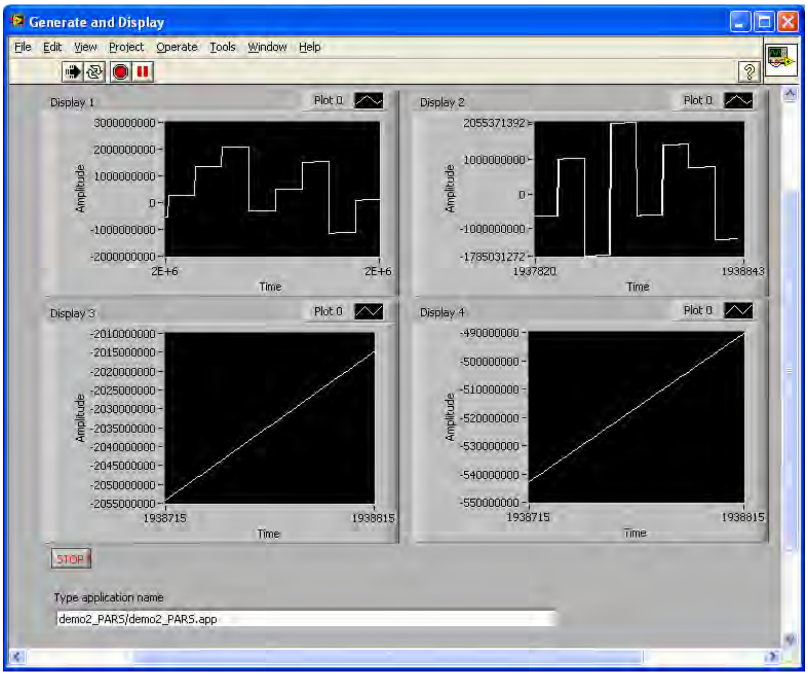


The PC side of the application is the following Labview VI; it can be loaded by double-clicking the file **Demo2/SMT310Q/Demo2.vi** in the demonstration folder.

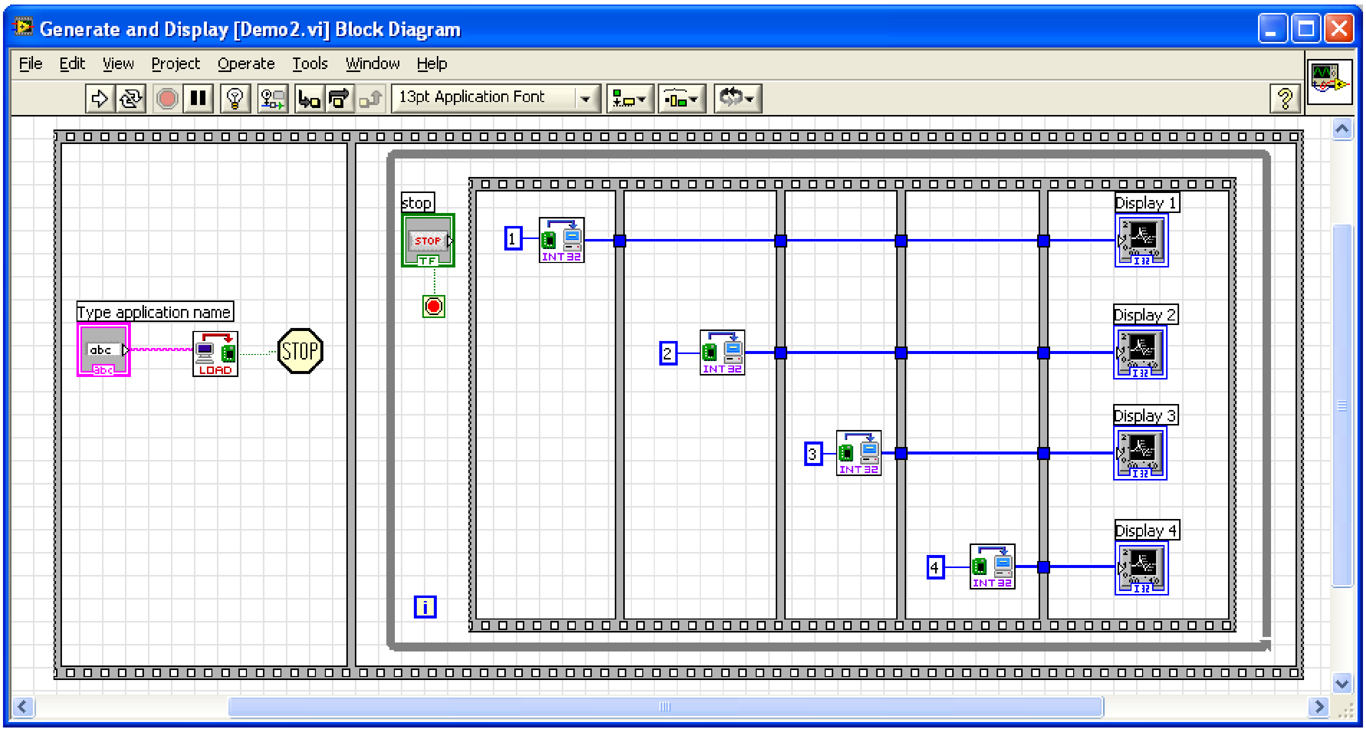


You can type in the name of the demonstration application file (**`demo2_PARS/demo2_PARS.app`**) and then execute the VI by pressing the right-arrow below View. File names are be relative to the folder containing the VI.

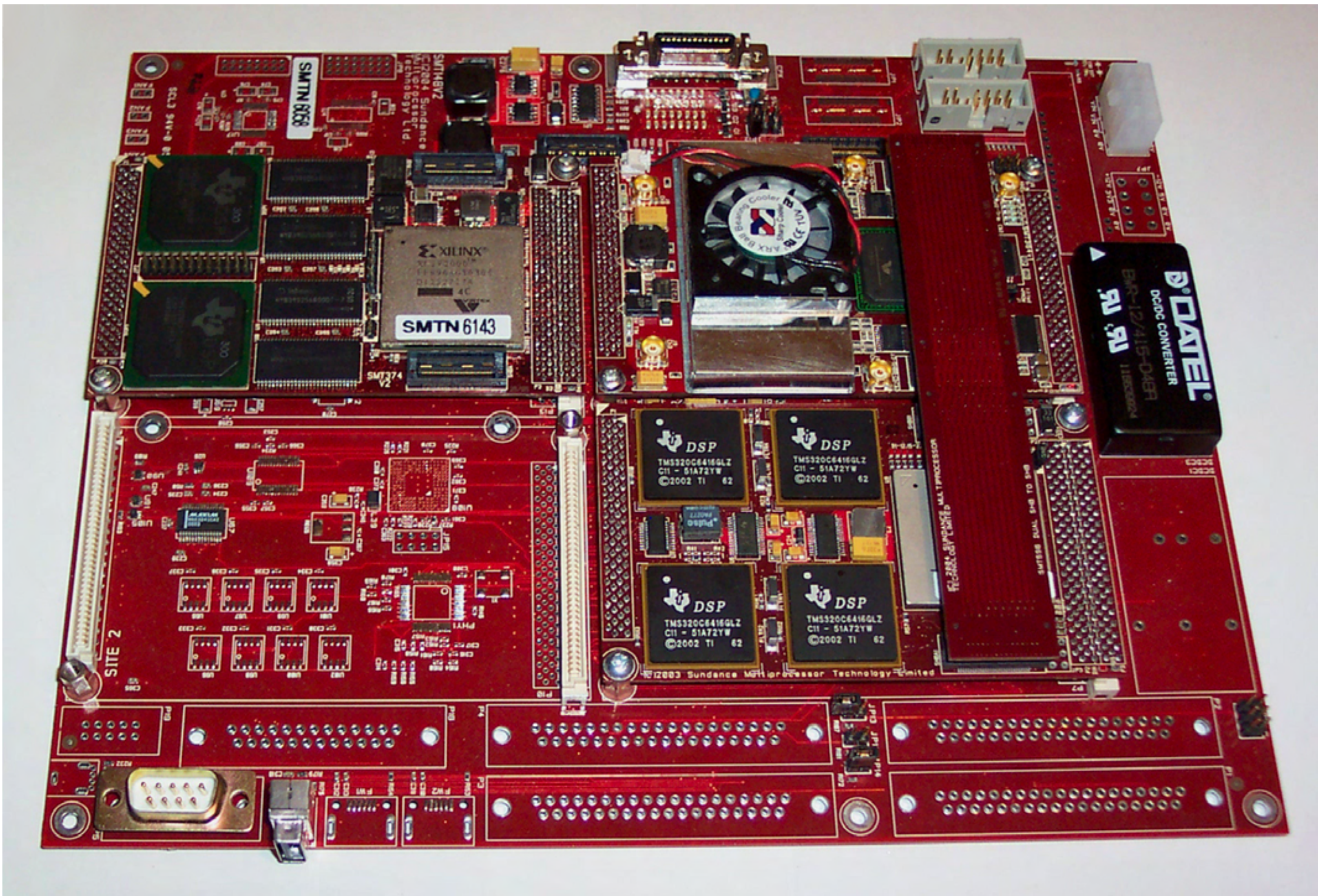
When the application runs, the following output will be displayed:



The application is loaded into the DSP system and starts to run. The four channels of integer data are read by four INT32 VIs. Note that Simulink will generate these streams of data and expect them to be read in the strict sequence that is echoed by the placement of the VIs into sequential frames within the main loop. The received data are then sent to be displayed.



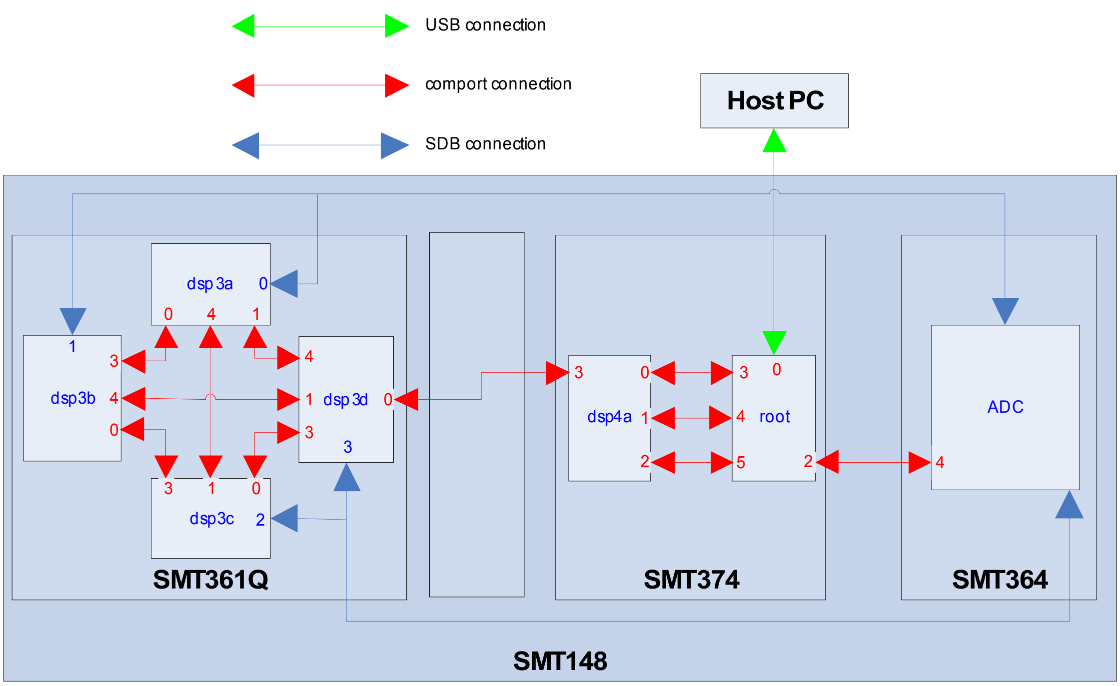
There is a variant of the demonstration that uses an SMT148 carrier.



This can be selected by setting PARSOptions as follows:

The files for this demonstration can be found in **`demos/demo2/SMT148`**.

The configuration is similar to the SMT310Q configuration with the exception that the root processor is now the second DSP on the SMT374, and this communicates with the host using a USB connection. This change in host communication is handled for you automatically.

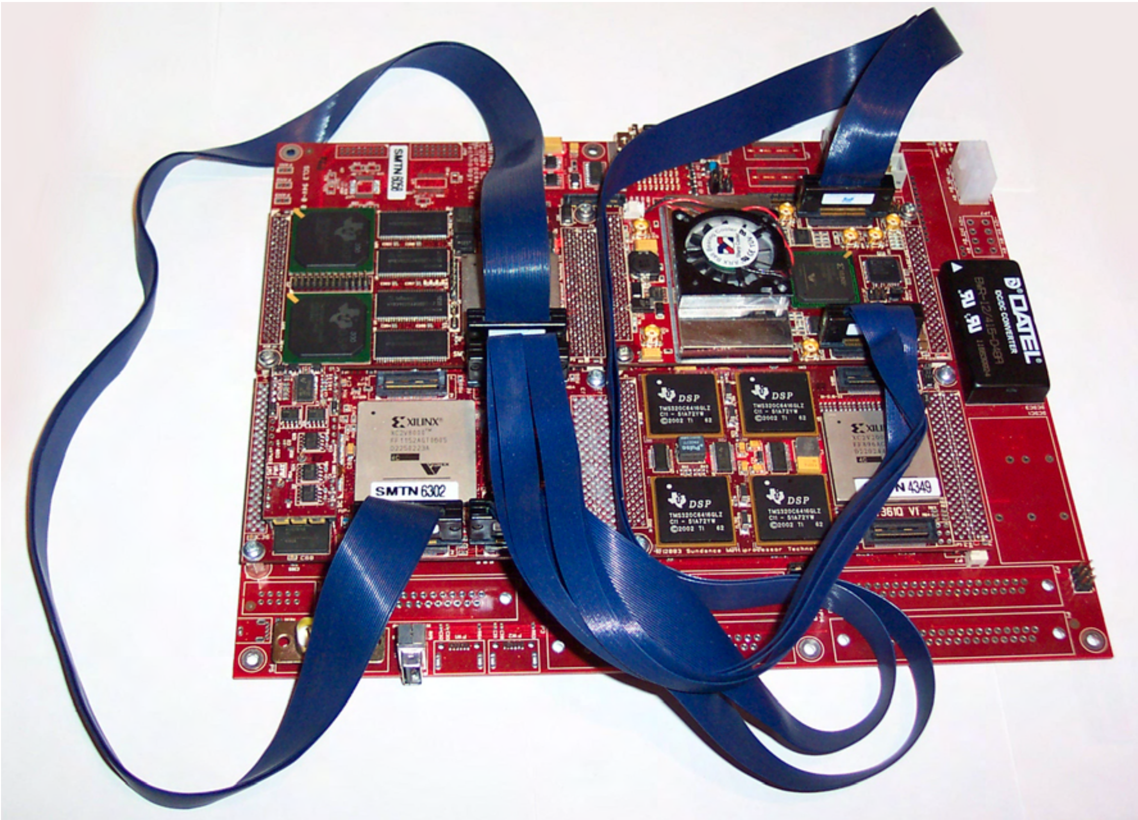


The application can be executed by double-clicking **`demo2/SMT148/demo2.vi`**.

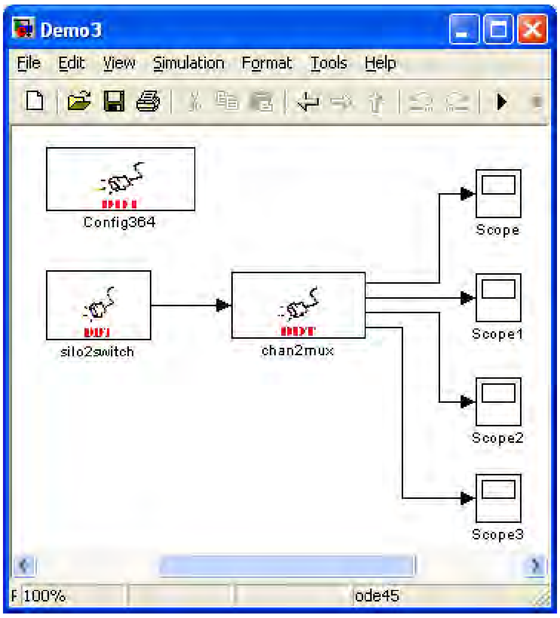
8 DEMONSTRATION 3

This demonstration can be found in the `\Demos\Demo3\` sub-folder of the SMT9650 installation folder; this will be `C:\SundanceDSP\SMT9650\Demos\Demo3\` by default. It takes four channels of 16-bit data from the SMT364 and passes them through an SMT398 which multiplexes them into a single output stream. This stream is passed to an SMT374 which demultiplexes it and displays the resulting four streams.

The SMT148 hardware for this demonstration is shown in the picture below[2]:

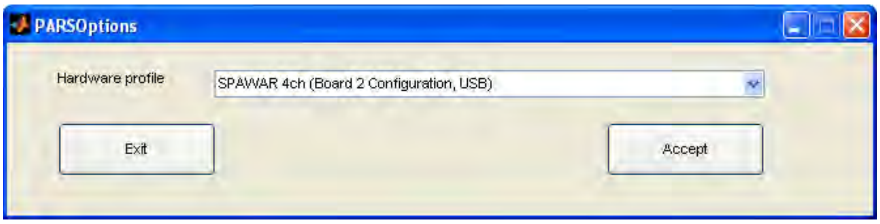


The Simulink model for this demonstration is as follows:



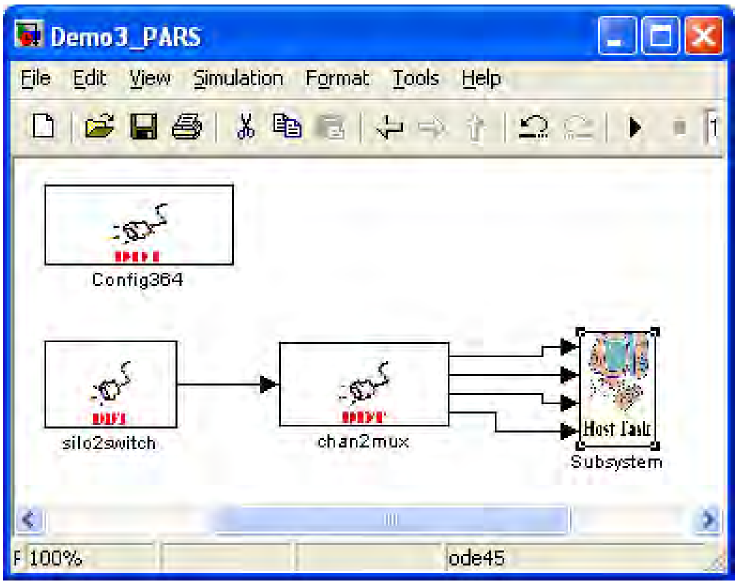
[2] The SMT361Q is not used in this demonstration.

The hardware profile for this demonstration is:

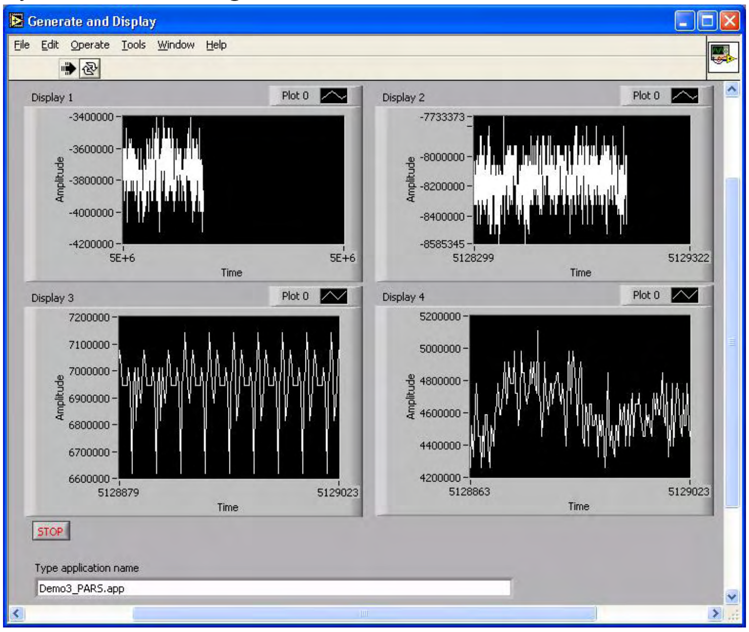


Note that this profile will reconfigure the SMT398 with firmware from the file **`fpga20.rfd`**. This file is present in the **`demo3`** folder, but a copy needs to exist in the **`demo3_PARS`** folder.

Once this has been set, PARS can create the following model:



The application can be executed by double-clicking **`demo3/demo3.vi`**.



9 ACRONYMS AND ABBREVIATIONS

ADC	Analog to Digital Converter
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
PARS	Parallel Application from Rapid Simulation
SDB	Sundance Digital Bus
TIM	Texas Instruments Module
USB	Universal Serial Bus
VI	LabVIEW Virtual Instrument

10 KNOWN ISSUES

Users should be aware of the following issues:

1. Labview can get into a strange state when an application fails or is stopped. This appears in two forms:

   a) When attempting to re-run an application or run a different application, Labview terminates.

   b) When an application hangs or fails, Labview starts to close but ends up in a minimal state that can only be cleared by using Task Manager to kill Labview.

2. When launching a VI by means of an executable, the VI starts to execute before the user can type an application name. Cancelling the selection of the hardware results in a notification of failure to load the application. The application name can now be given and the VI started manually.

3. The pre-built tasks must be allocated enough memory. Failure to do so will result in the application hanging. Details of the required memory can be found in **`doc/*.txt`** in the smt6045 installation directory.

4. The current release does not support logging.

5. Host output port numbers are allocated sequentially from top to bottom of the Host task, starting at 1. Host input port numbers are allocated in the same manner.

11 COPYRIGHT

Matlab and *Simulink* are trademarks of The MathWorks.

LabVIEW is a trade mark of National Instruments.

12 LICENSE AGREEMENT

END-USER LICENSE AGREEMENT for

SMT9650

Sundance Digital Signal Processing Inc.

Referred to hereafter as (SDSP)

IMPORTANT - READ CAREFULLY: This END-USER LICENSE AGREEMENT

(Or "AGREEMENT") is a Legal Agreement between you (either an individual or entity) and SDSP.

The "LICENSED MATERIALS" subject to this Agreement include the enclosed software programs and documentation and any "on-line" or electronic documentation associated with the software programs. The "LICENSED MATERIALS" (collectively referred to as the "SOFTWARE") may include certain SDSP proprietary software programs that. By installing, copying or otherwise using the "LICENSED MATERIALS", you agree to abide by the following provisions.

You assume responsibility for the selection of the SOFTWARE to achieve your intended results, and for the installation, use and results obtained from the SOFTWARE.

1. SOFTWARE LICENSE - The Licensed Materials are protected by copyright laws, international copyright treaties, and trade secret laws, as well as other intellectual property laws and treaties. The Licensed Materials are licensed, not sold to you, and can only be used in accordance with the terms of this Agreement. SDSP retains title and ownership of the Licensed Materials, including all intellectual property rights.

   a) Restrictions - This license is for a single-user host computer. You may not install the Licensed Materials on a network server or otherwise use the Licensed Materials on more than one host computer at the same time. Additionally, if this package contains multiple versions of the Licensed Materials, you may only use one version of the Licensed Materials on a single host computer. You may either make one copy of the Licensed Materials for archival purposes or copy the Licensed Materials to another medium and keep the original Licensed Materials for archival purposes. Other than as expressly set forth in Section 1(b) below, you may not otherwise copy or reproduce the

Licensed Materials. In no event may you use two copies or versions of the Licensed Materials on more than one host computer at the same time.

b) Copy, modify, and merge - You may copy the SOFTWARE into any machine-readable or printed form for backup purposes in support of your use of the SOFTWARE on the single-user machine. You may not modify the SOFTWARE. You may merge it into another program for your use on the single-user machine. Any portion of this SOFTWARE merged into another program will continue to be subject to the terms and conditions of this Agreement. You must reproduce and include the copyright notice (s) on any copy, modification, or portion merged into another program. The license entitles the user to 20 runtime licenses. To obtain additional runtime licenses please contact SDSP.

c) Termination - This license is effective until terminated. Without prejudice to any other rights, SDSP may terminate your right to use the Licensed Materials and any applications generated using the Licensed Materials under this Agreement if you fail to comply with the terms of this Agreement. In such event, you must destroy all copies of the SOFTWARE and all of its component parts.

2. INTELLECTUAL PROPERTY RIGHTS. The Licensed Materials contain copyrighted material, trade secrets and other proprietary information. In order to protect the Licensed Materials, and except as specifically permitted by statute, you may not decompile, reverse engineer, disassemble or otherwise translate the object code versions of the software programs included in the Licensed Materials to human-perceivable form. If you are a corporation you agree you will use your best efforts to prevent your employees and contractors from decompiling, reverse engineering, disassembling, modifying or translating the Licensed Materials. In no event may you alter, remove or destroy any copyright notice included in the Licensed Materials. SDSP reserves all rights not specifically granted under this Agreement.

3. APPLICABILITY. This license only applies to the version of the SOFTWARE for which it was issued. It does not automatically entitle the licensee to software advancements and functionality to be offered in future versions of the software. The licensee can obtain future versions of the SOFTWARE by prior agreement with SDSP.

4. LIMITATIONS. SDSP MAKES NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (NOT SDSP) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION.

5. IN NO EVENT WILL SDSP BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOSS OF BUSINESS INFORMATION, BUSINESS INTERRUPTION, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH SOFTWARE, EVEN IF SDSP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

6. YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND SDSP WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

7. EXPORT CONTROL. The re-export of United States origin software and documentation is subject to the Export Administration Act of 1969 as amended. Compliance with such regulations is your responsibility.

8. U.S. GOVERNMENT RESTRICTED RIGHTS. This computer software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (Oct. 1988) or if provided under a contract or subcontract with NASA or a civilian agency of the Government, to the restrictions set forth is such contract or subcontract.

9. Should you have any questions concerning this Agreement, or if you desire to contact SDSP for any reason, please contact SDSP.

13 INSTALLATION

Before installing the SMT9650:

- Make sure that all the required software components have been installed and configured properly;
- Close any instances of Matlab.

To install the SMT9650, run **`SMT9650_setup_vX_X.exe`** from the installation CD, where ***`X_X`*** denotes the SMT9650 version number. For example, use **`SMT9650_setup_v4_0.exe`** to install SMT9650 version 4.0.

You should see the following window.

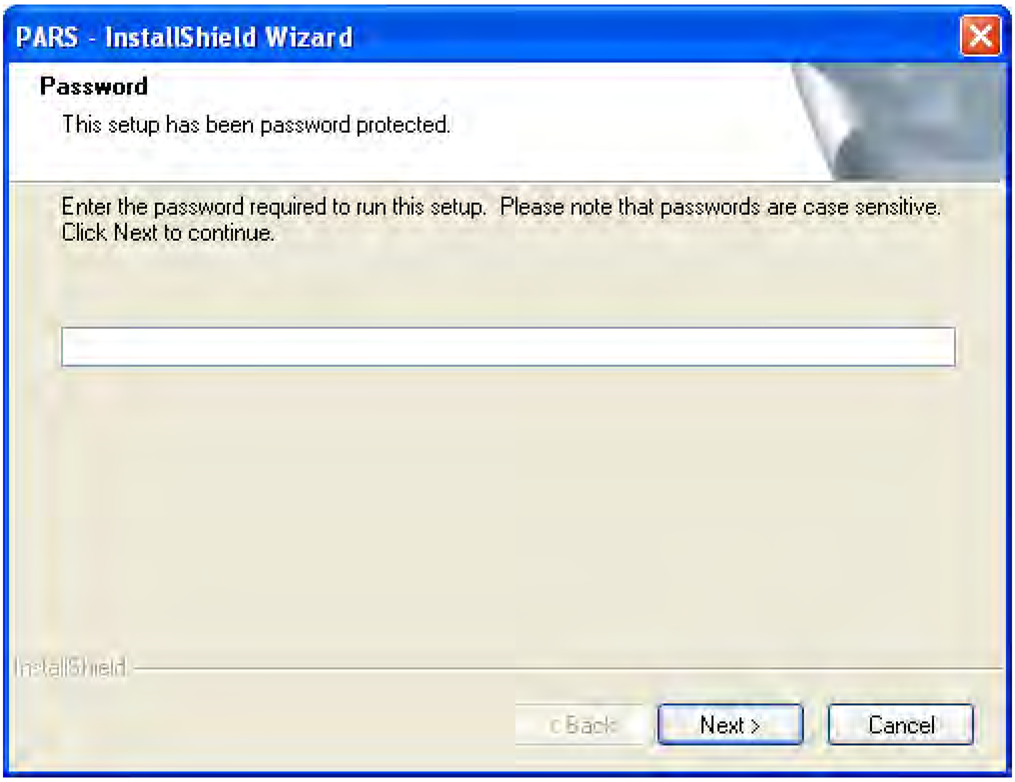


Figure 4: Setup step 1, Password page.

Enter your password and press next. Please contact your reseller if you have not been given a password. Press **Next>** to continue to the welcome page, and then press **Next>** again.

Figure 5: Setup step 2, welcome page.

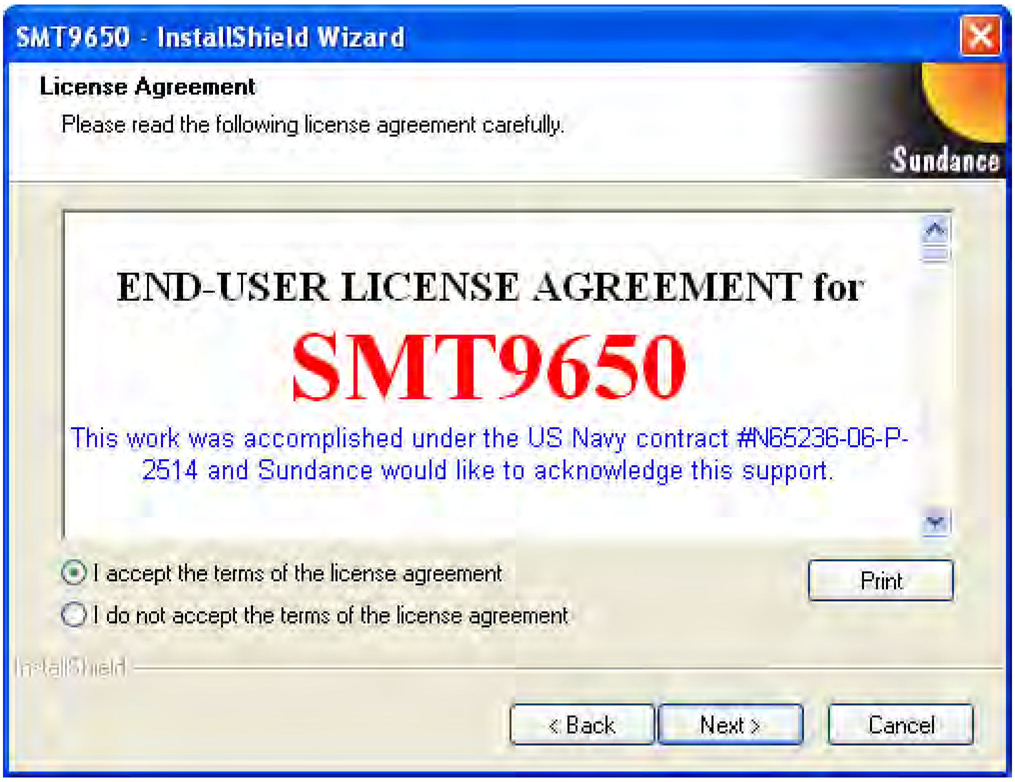


Figure 6: Setup step 3, accepting the end user license agreement.

Please read the End User License Agreement (EULA) carefully. You can find the text of this agreement in the previous section of this User Guide. If you agree with the terms of the license, select the accept option and press **Next>**. If you do not agree with the terms and conditions, **Cancel** the installation and contact your reseller.

Figure 7: Setup step 4, user name and company name.

Enter your user name and company name, then press **Next>**.

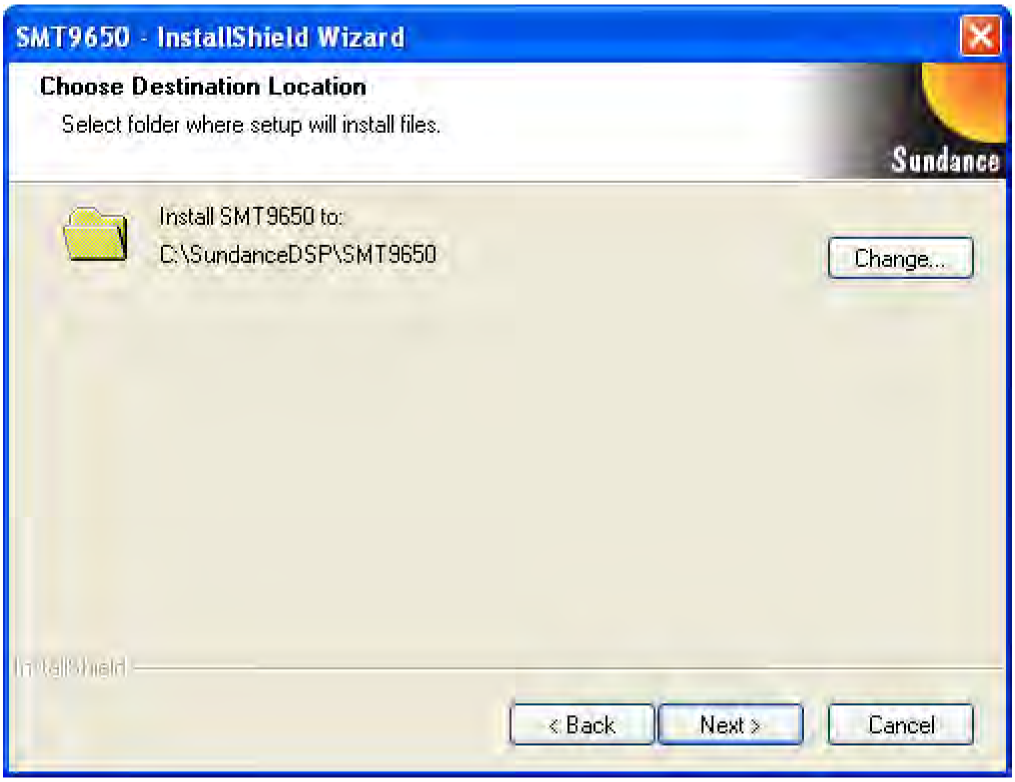


Figure 8: Setup step 5, selecting the installation directory.

The SMT9650 will usually be installed into the folder `C:\SundanceDSP\SMT9650\`.

If you want to use a different folder, select change and browse to the correct location.

Click **Next>** when you have chosen the installation folder.

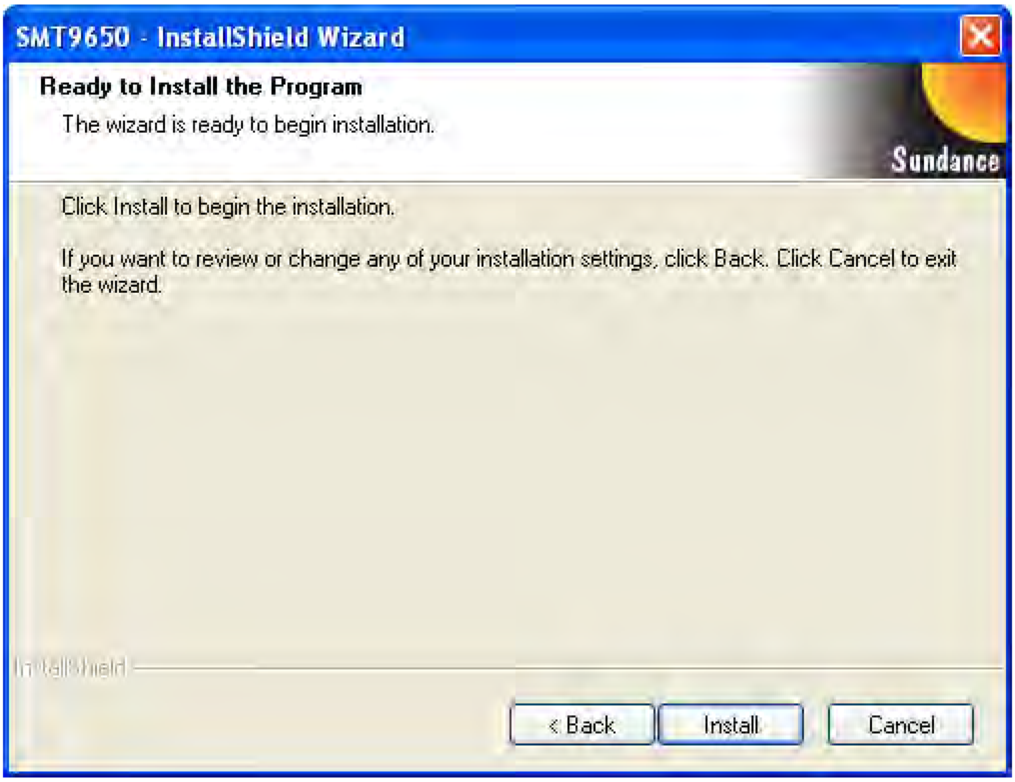


Figure 9: Setup step 6, checking installation setting.

Press **<Back** if you want to change any settings before starting the installation process by clicking **Install**.

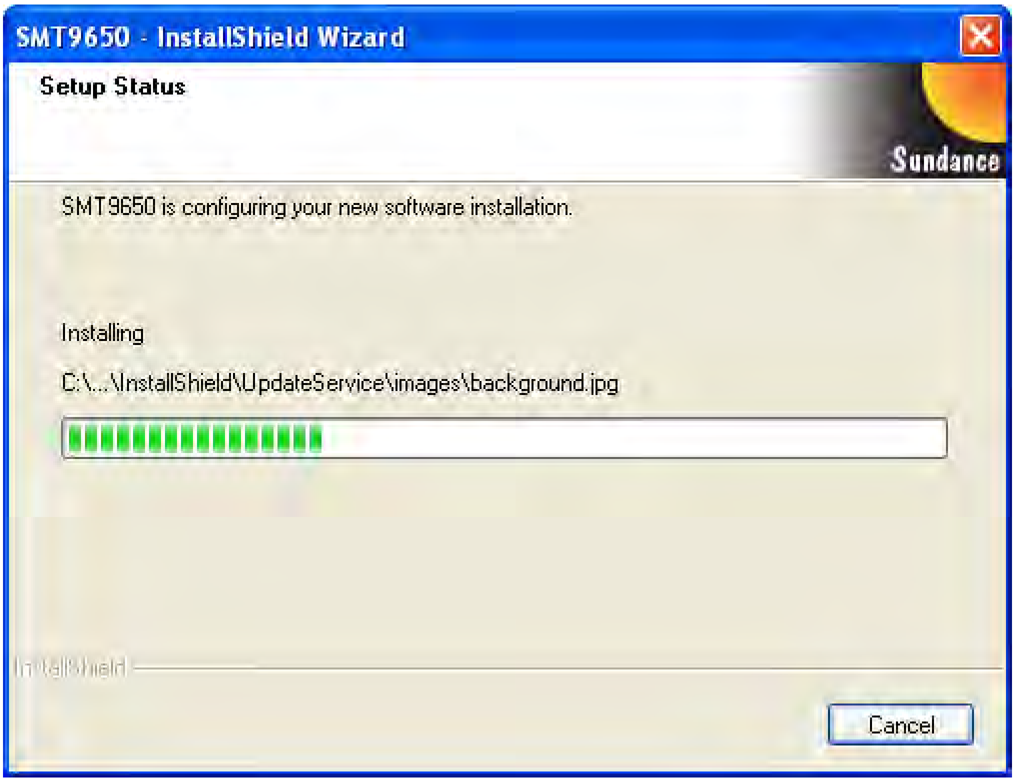


Figure 10: Setup step 7, Installation in progress.

Do not turn your computer off until the “Installshield Wizard Complete” message has appeared.

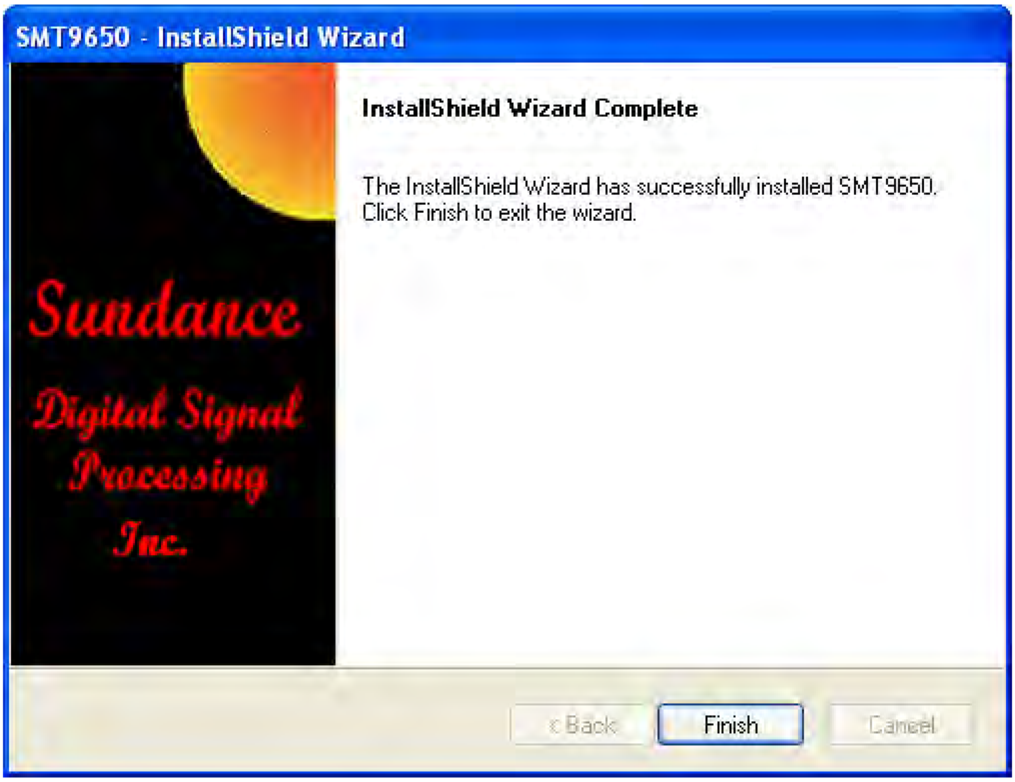


Figure 11: Setup step 8, Installation completed window.

This window shows that the installation of SMT9650 has finished.

14 ACKNOWLEDGEMENT

This work was accomplished under the US Navy contract #N65236-06-P-2514 and Sundance would like to acknowledge this support.