Using the ABEL Tools of PAC-Designer with Power Manager Devices

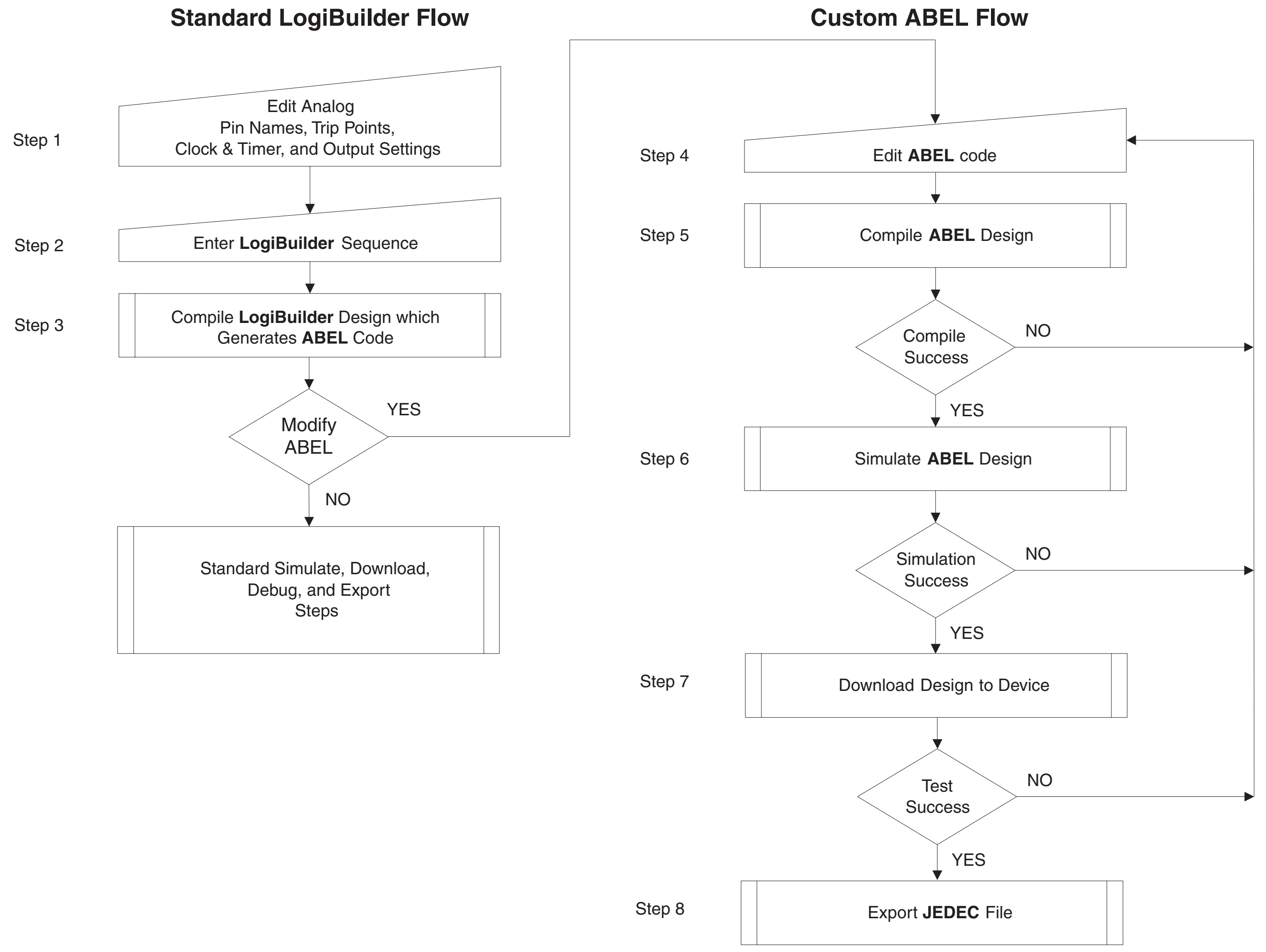
May 2003 Application Note AN6052

Introduction

Lattice Semiconductor's PAC-Designer® software provides the LogiBuilder™ interface for the development of sequential and logic designs for Power Manager devices. While the LogiBuilder interface is very capable of addressing a wide variety of monitoring and sequencing applications, occasionally a design will require extended functionality. For example implementing a "watch dog" timer or combining multiple functions within a state of the state-machine to increase the efficiency of the embedded CPLD can be realized by using the ABEL tools.

PAC-Designer generates ABEL code from the LogiBuilder design because ABEL is a mature language that supports optimal fitting of logic into CPLDs that contain only a few macrocells. VHDL on the other hand is a language that is more suited to system designs utilizing hundreds or thousands of macrocells. Fortunately, PAC-Designer has the built-in capability to support user-modified ABEL compilation and simulation. In this application note, we will cover the ABEL-flow process, a review of the timers, and demonstrate some features of the Power Manager devices with examples. Additional information on the ABEL language is available in other manuals (see the reference list).

Figure 1. ABEL Process Flow Chart


Standard LogiBuilder Flow
Custom ABEL Flow
Step 1
Edit Analog Pin Names, Trip Points, Clock & Timer, and Output Settings
Step 2
Enter LogiBuilder Sequence
Step 3
Compile LogiBuilder Design which Generates ABEL Code
Modify ABEL
YES
NO
Standard Simulate, Download, Debug, and Export Steps
Step 4
Edit ABEL code
Step 5
Compile ABEL Design
Compile Success
NO
YES
Step 6
Simulate ABEL Design
Simulation Success
NO
YES
Step 7
Download Design to Device
Test Success
NO
YES
Step 8
Export JEDEC File

The ABEL Flow

While the specific menus and tools within PAC-Designer may vary based upon the device or version of the software, the following design flow applies to any ABEL-based Power Manager design. In Figure 1 we see the major steps of the process required to enter a design, modify the ABEL code, simulate the design, and program a device. Each step is discussed in detail below.

Step 1: Edit Analog Settings

From the main schematic page, enter the names of the input signals to be monitored and set the trip points using the **Analog Input Setting** dialog box. Also name the output signals and configure the HVOUT settings (if applicable). Use the **Clock & Timer** dialog box to configure the timers and the clock settings. As a side note, all of these settings are stored in the analog bits.

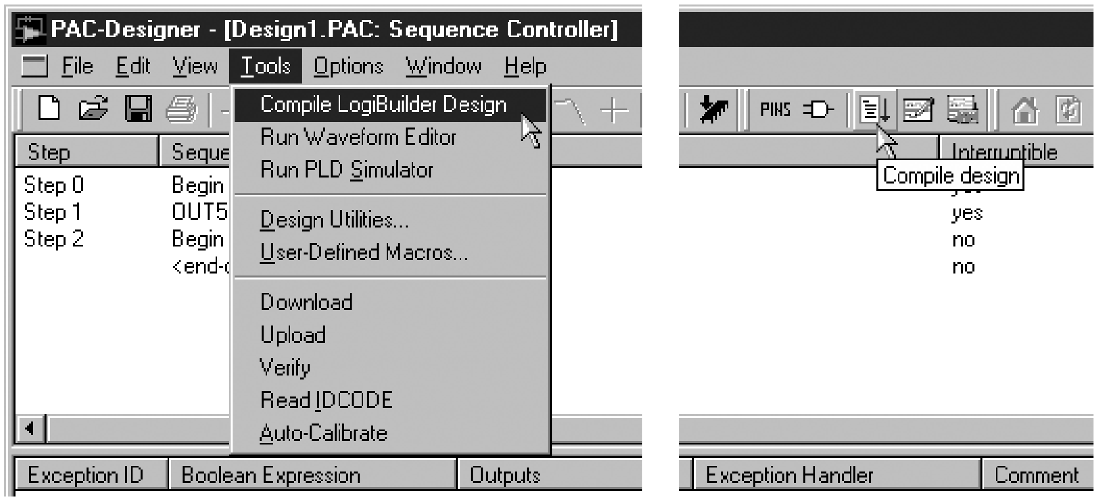
Step 2: Enter LogiBuilder Sequence

Double-click on the sequence controller block to enter a LogiBuilder sequence. Some form of LogiBuilder sequence has to be entered before PAC-Designer will successfully generate and compile an ABEL file. The LogiBuilder sequence can be as complex as a complete design that only needs minor changes to the ABEL code, or as simple as a single-output statement. In addition, the **PINS** window should be opened to configure the pins, because this information will be carried forward into the ABEL source code at compile time.

Step 3: Initial Compile

Use the LogiBuilder menu item **Tools > Compile LogiBuilder Design** or the icon on the toolbar (both are shown in Figure 2) to generate the ABEL code, compile, and fit the sequence. It is during this step in the process that the ABEL source file is written into the PLD files subdirectory (example: Design1_PLDFiles\Design1.abl). Up to this point, the process is the same as for any other Power Manager design.

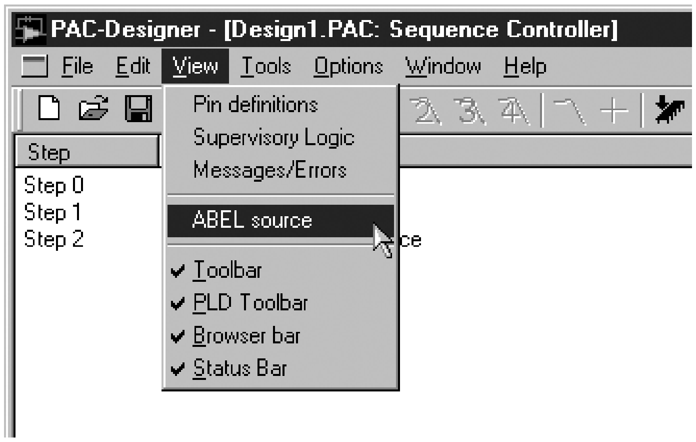
Figure 2. Either Menu or Compile Tool Can be Used to Generate the ABEL Source Code



Step 4: Edit ABEL

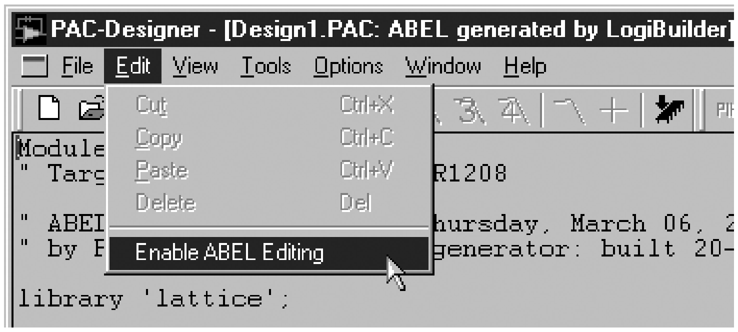
Use the menu item **View > ABEL Source** (see Figure 3) to open a new window that displays the ABEL source code. This window defaults to a grayed-out state to provide a read-only view of the ABEL source. The user can use the menu item **Edit > Enable ABEL Editing** (see Figure 4) to remove the gray and switch the mode in PAC-Designer from LogiBuilder-based to ABEL-based. This is indicated by the fact that the LogiBuilder and **PINS** windows are now grayed-out and modifications to them are not allowed. This protects users from over-writing and losing their customized ABEL code from an accidental compile from LogiBuilder.

Figure 3. View Menu, ABEL SOURCE Allows the User to See the ABEL Code



The ABEL editing window supports simple type and edit features including standard keystrokes (**CTRL+X, CTRL+C, and CTRL+V**) as well as a drop-down edit menu to support cut, copy, and paste. Once the edits are made, the ABEL code can be saved, compiled or simulated, and downloaded.

Figure 4. Editing Must be Enabled to Modify the ABEL Code

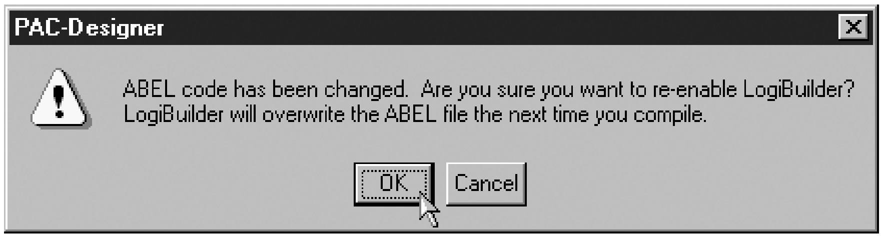

PAC-Designer - [Design1.PAC: ABEL generated by LogiBuilder]
File Edit View Tools Options Window Help
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Enable ABEL Editing

```
Module
" Tarc                R1208

" ABEI                hursday, March 06, 2
" by F                generator: built 20-

library 'lattice';
```

The user is free to modify almost all of the ABEL source code. However, certain sections should not be modified or compilation or fitting errors may result. The sections that are restricted from being edited are (in order of appearance): the **Module PSCProgram** name, the include statement **library 'lattice'**, the numerical value of the **Pin Numbers**, the following signals, **CLK_IN**, **TMR_clk**, **PLD_clk**, **RESET**, and the **END** keyword.

If the user would like to return to the LogiBuilder mode of PAC-Designer, the menu item **View > Enable ABEL Editing** has to be un-checked. PAC-Designer provides a safety net with the dialog box shown in Figure 5, because returning to LogiBuilder mode and recompiling will overwrite the ABEL source code, which may or may not be the intent of the user. If the ABEL source needs to be regenerated from the LogiBuilder sequence, click **OK**, otherwise **Cancel**.

Figure 5. PAC-Designer Warning when Returning to LogiBuilder Mode

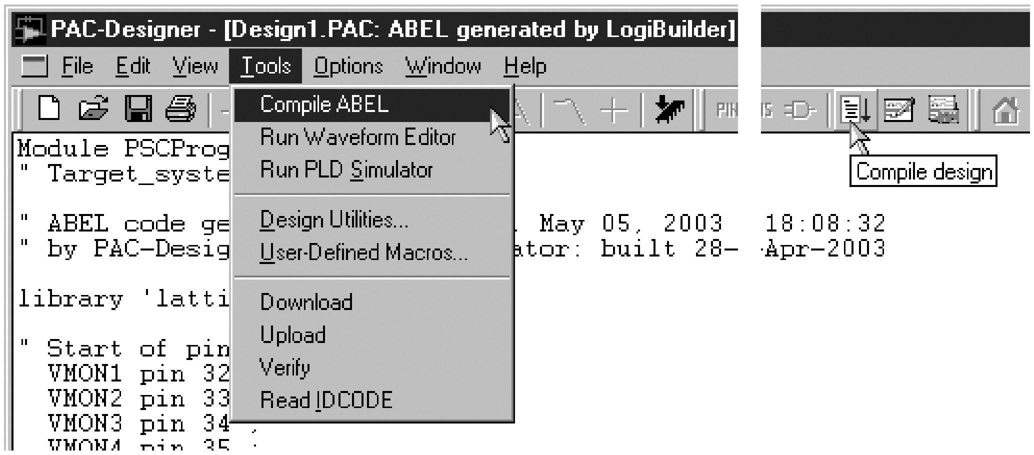


ABEL code has been changed. Are you sure you want to re-enable LogiBuilder?
LogiBuilder will overwrite the ABEL file the next time you compile.

Step 5: Compiling the ABEL Code

While an experienced ABEL designer may wish to simulate the design at this point, we recommend that the user compile the design first. This is because the fitting issues and error reporting are better handled by the compile flow. To compile the ABEL design, use the menu item **Tools > Compile ABEL** or the icon on the toolbar, both are shown in Figure 6. After the compile is successful, one can move on to the simulate or download step of the design process.

Figure 6. Compile ABEL Code with Either the Menu or Tool



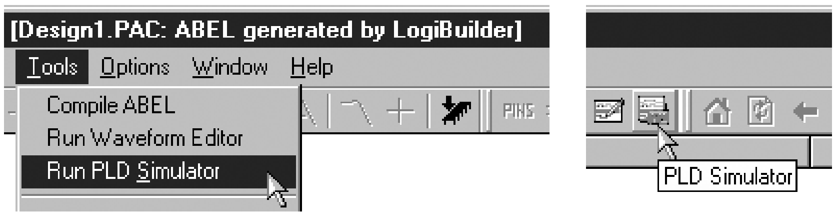
```
Module PSCProg
" Target_syste

" ABEL code ge          May 05, 2003   18:08:32
" by PAC-Desig        ator: built 28- Apr-2003

library 'latti

" Start of pin
  VMON1 pin 32
  VMON2 pin 33
  VMON3 pin 34 ;
  VMON4 pin 35 ;
```

Step 6: Simulating the Design

To simulate the modified ABEL, use the menu item **Tools > Run PLD Simulator** or the icon on the toolbar (both are shown in Figure 7). Either the default or other stimulus files should be edited before the simulation is started.

Note that the menu item provides a browser dialog to select both the ABEL source and stimulus file. This supports multiple design and stimulus files. Once changed, the selections remain in effect for all subsequent simulations or compilations during this session of PAC-Designer.

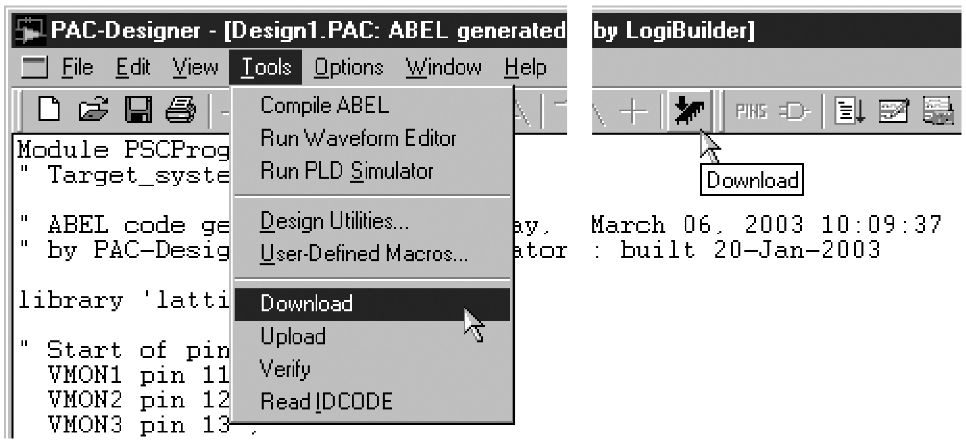
Figure 7. Simulate the ABEL Code with Either the Menu Item or Tool



Step 7: Downloading the Design to the Device

Use either the menu item **Tool > Download** or the icon from the toolbar (both are shown in Figure 8) to download the complete, both analog settings and PLD, design into the device. Test the design on the bench and make adjustments to the design as needed.

Figure 8. Download Menu or Tool are Used to Program Device with the Total Design

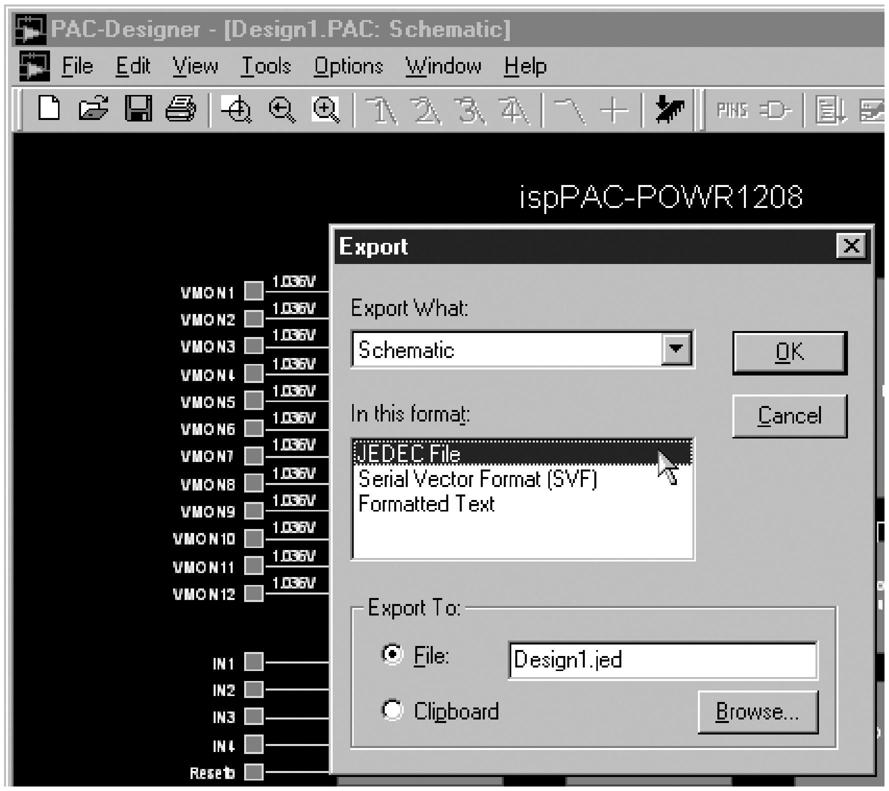


```
Module PSCProg
" Target_syste
" ABEL code ge                    ay,   March 06, 2003 10:09:37
" by PAC-Desig                    ator  : built 20-Jan-2003
library 'latti
" Start of pin
  VMON1 pin 11
  VMON2 pin 12
  VMON3 pin 13 ;
```

Step 8: Exporting JEDEC File

Return to the top level schematic window, and use the menu item **File > Export** to export the complete design into a JEDEC file (see Figure 9). Note that the **Export What** list box provides a selection of either **Schematic** or **Sequencer**. However, either selection will export the same JEDEC file and that is the complete design which includes both analog and PLD bits. The selection of **Schematic** or **Sequencer** will only make a difference in the **Text** export.

Figure 9. Export Complete Design to a JEDEC File



Export What: Schematic

In this format: JEDEC File

Export To: File: Design1.jed

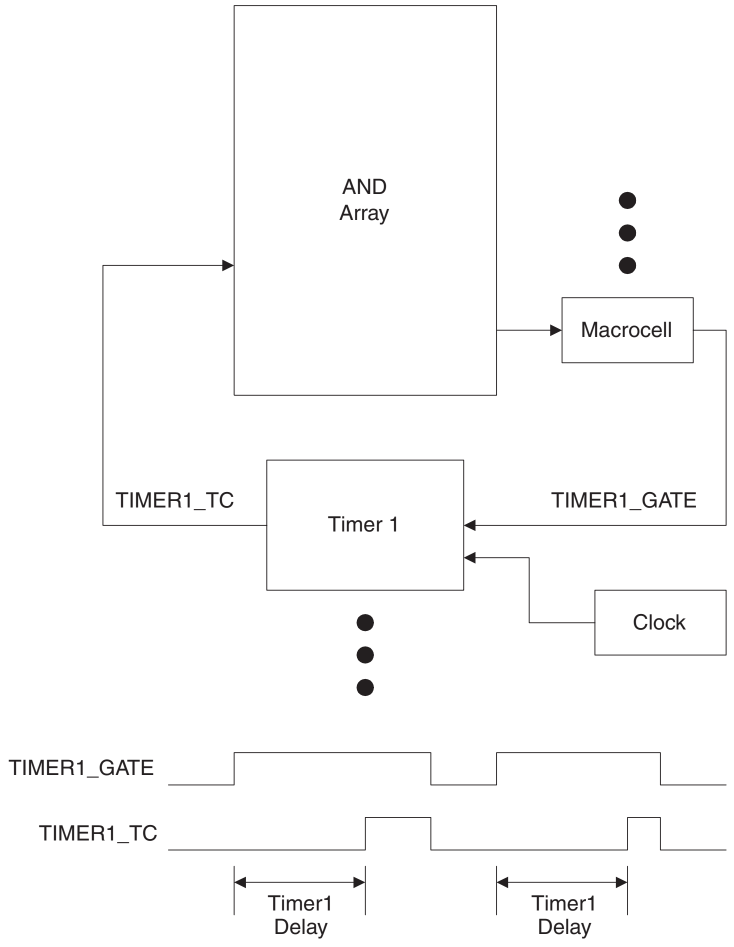
Timer Details

In the design examples that follow (in the next sections) the timer hardware will be controlled using ABEL. Therefore, a brief review of the timers is in order. Please refer to Figure 10 during the following discussion of both the timer hardware and functionality.

Each timer has a single input gate signal (TIMERx_GATE) that enables the timer when HIGH and resets it when LOW. The gate signal is the output of a macrocell and is thus controlled by the logic within the AND array. Each timer has a terminal-count output signal (TIMERx_TC) that is an input to the AND array. This signal goes HIGH when the timer has counted the programmed number of clock pulses and is reset LOW when the gate signal is LOW.

In Figure 10, the clock block represents either the internal clock or external clock source. It also contains the timer prescalar, which is configured using the **Clock & Timer Settings** dialog box. The same dialog box is also used to set the timer values. These settings are stored in the analog bits of the design. The initial compile of the LogiBuilder sequence places a copy of these settings within the macros in the ABEL file to support simulation.

Figure 10. Timer Block Diagram and Waveforms



Listing 1 shows a portion of the ABEL code that is generated by PAC-Designer. This section contains the settings from the **Clock & Timer Settings** dialog box, and are in the code to support simulation only. The first two lines are comments and the last two are macro instantiations. The macros are defined in the “lattice” library that is included at the top of the ABEL file. Neither macro has any effect on the compiler, fitter or exported JEDEC file because the clock and timers are configured with the analog bits.

Listing 1. Timer macros are for Simulation Only

```
“ PRESCALER Declaration with attributes:
“ XLAT_PRESCALER(TMR_clk, PLD_clk, clk_250K, RST, TMR_factor, PLD_factor)
XLAT_PRESCALER(TMR_clk, PLD_clk, CLK_IN, RESET, 4, 1);
XLAT_STIMER(TIMER1_TC, TIMER1_GATE, TMR_clk, RESET, 1);
```

Both macros can be edited to support simulation of different pre-scalar and timer settings either as a function of the design or to reduce simulation time. In the XLAT_PRESCALER macro, the last two parameters reflect the timer and PLD clock divider values. Note the timer divider value of 4 represents the 32.00 us to 4.096 ms selection from the **Timeout Range** list-box. Likewise, the value of 1 for the last parameter in the XLAT_STIMER macro represents the 32.00 us selection from the **Timeout1:** list-box. Table 1 lists both timeout values (in milliseconds) and valid values for each of the macros based upon the internal oscillator operating at 250kHz.

Table 1. Timeout Values in Milliseconds for Valid Macro Settings

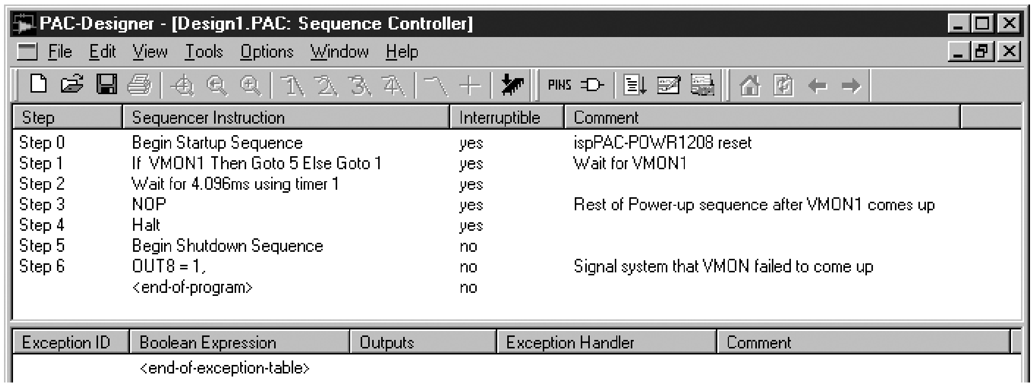
XLAT_STIMER Value	XLAT_PRESCALER TMR_factor							
	4	8	16	32	64	128	256	512
1	0.032	0.064	0.128	0.256	0.512	1.024	2.048	4.096
3	0.064	0.128	0.256	0.512	1.024	2.048	4.096	8.192
7	0.128	0.256	0.512	1.024	2.048	4.096	8.192	16.384
15	0.256	0.512	1.024	2.048	4.096	8.192	16.384	32.768
31	0.512	1.024	2.048	4.096	8.192	16.384	32.768	65.536
63	1.024	2.048	4.096	8.192	16.384	32.768	65.536	131.072
127	2.048	4.096	8.192	16.384	32.768	65.536	131.072	262.144
255	4.096	8.192	16.384	32.768	65.536	131.072	262.144	524.288

“Wait for VMON OR TIMER” Example

In this example, the system needs to wait for a primary supply (VMON1) to power up before proceeding with the sequence. If this primary supply fails to power-up in 4 milliseconds, a warning signal (OUT8) is set to indicate the situation. This example is a form of watchdog or dead-man timer. For this example, let us assume that LogiBuilder does not support such timer configurations. In this example however, the dead-man timer function is easily added and states are combined by making a few simple changes to the ABEL code.

For this example start a new design and enter the core of the sequence into the Logibuilder window, as shown in Figure 11. By including the **Wait for Timer** instruction in step 2, PAC-Designer not only generates an ABEL template, but it will also generate the corresponding code to support simulation of the timer. However, examination of the ABEL code generated in Listing 2 reveals that the gate to timer 1 is not active until step 2.

Figure 11. LogiBuilder Sequence to “Wait for Timer OR VMON”


PAC-Designer - [Design1.PAC: Sequence Controller]
File Edit View Tools Options Window Help

Step	Sequencer Instruction	Interruptible	Comment
Step 0	Begin Startup Sequence	yes	ispPAC-POWR1208 reset
Step 1	If VMON1 Then Goto 5 Else Goto 1	yes	Wait for VMON1
Step 2	Wait for 4.096ms using timer 1	yes	
Step 3	NOP	yes	Rest of Power-up sequence after VMON1 comes up
Step 4	Halt	yes	
Step 5	Begin Shutdown Sequence	no	
Step 6	OUT8 = 1,	no	Signal system that VMON failed to come up
	<end-of-program>	no	

Exception ID	Boolean Expression	Outputs	Exception Handler	Comment
	<end-of-exception-table>			

Thus, changes to the ABEL code are required to get the timer started before the **If Then Else** instruction in step 1. Additional changes are also required to combine states and implement the desired logic. The resulting code is shown in Listing 3. A brief description of the edits is provided between the two listings.

Listing 2. PAC-Designer Generated ABEL State-Machine "Wait for VMON OR Timer

```
state_diagram [STATE_FF3, STATE_FF2, STATE_FF1, STATE_FF0]
State 0:
    Goto 1;
State 1:
     TIMER1_GATE = 0 ;  " stop/clear timer 1
    If  VMON1 Then 5
    Else 1;
State 2:
" Wait for timer 1:
     TIMER1_GATE = 1 ;  " enable timer 1
    If TIMER1_TC Then 3 With TIMER1_GATE=0
    Else 2 With TIMER1_GATE=1;
State 3:
    Goto 4;
State 4:
    Goto 4;
State 5:
" Begin Shutdown sequence.
" All Exceptions branch here, but exceptions are disabled.
    Goto 6;
State 6:
    OUT8.J = 1 ; " set OUT8
    OUT8.K = 0 ;
    Goto 7;
State 7:
    Goto 7;

END
```

The first change to make is to start Timer 1 in state 0, by moving the line **`TIMER1_GATE = 1 ; "enable timer 1`** from state 2 to state 0. Next, add a **`With`** statement after the VMON1 in state 1 to reset the timer.

Merging states 1 and 2 using an **`Else`** statement combines the VMON1 test with the timer 1 test. The **`If-Then-Else`** construct in the third line of state 1 is entered by hand and branches to state 6 to indicate the timer timed out. To the last line in state 1, add the **`With`** statement to keep the timer active.

In state 2, enter the **`Goto`** statement to advance the sequencer. Similarly, this state could have been deleted or used for additional logic as a result of combining states.

Listing 3. Modified ABEL State-Machine (Edits in Bold), "Wait for VMON OR Timer"

```
state_diagram [STATE_FF3, STATE_FF2, STATE_FF1, STATE_FF0]

State 0:
    TIMER1_GATE = 1; “enable Timer 1
    Goto 1;
State 1:
    “ Wait for timer 1 terminal count or VMON1 to come up
    If  VMON1 Then 2 With TIMER1_GATE=0 “ VMON1 came up OK clear timer
    Else If TIMER1_TC Then 6 With TIMER1_GATE=0 “ Timer 1 timed out
    Else 1 With TIMER1_GATE=1; “ loop until timeout or VMON1

State 2:
   Goto 3;

State 3:
   Goto 4;
State 4:
   Goto 4;

State 5:
“ Begin Shutdown sequence.
“ All Exceptions branch here, but exceptions are disabled.
   Goto 6;
State 6:
    OUT8.J = 1 ; “ set OUT8 Indicate to system that VMON1 did not come up
    OUT8.K = 0 ;
    Goto 7;
State 7:
   Goto 7;

END
```

"Consecutive Timers" Example

This example shows both how a timer may be reused in consecutive states of a state machine and how to combine the **Wait for VMON** and the **Wait for Timer** instructions. This technique can be used to reduce the number of states or steps in order to minimize the PLD resources used by the design. Listing 4 contains a portion of the ABEL source code that illustrates the consecutive timers and combined instructions.

The timer gate in this example is driven by the output of a three-input registered AND gate and is thus a function of VMON1 and VMON2 and NOT timer 1 terminal count. Thus the timer will not start until both VMON1 and VMON2 are above their respective set points. When the timer completes counting, the terminal count signal is true for 1 PLD_clk cycle. At this point, the state machine moves from state 0 to state 1 and the timer is reset. The last line in state 0 maintains the timer gate logic and loops state 0 upon itself. State 2 is essentially a cut-and-paste of state 1, using different VMONs.

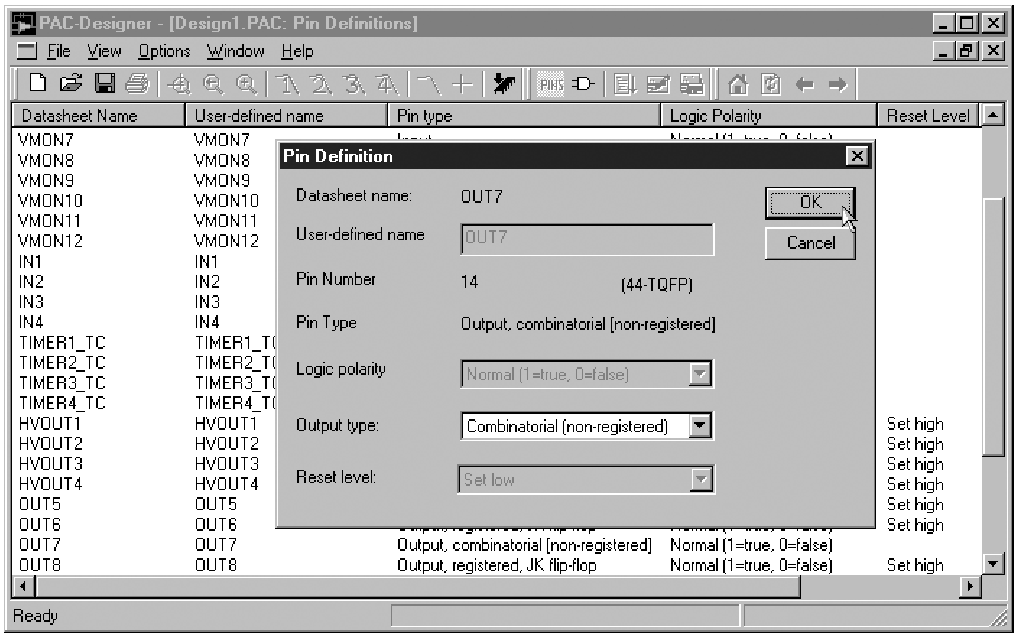
Listing 4. Snippet of ABEL Source Code for "Consecutive Timers" Example

```
state_diagram [STATE_FF3, STATE_FF2, STATE_FF1, STATE_FF0]
State 0:
    TIMER1_GATE = VMON1 & VMON2 & !TIMER1_TC;
    If TIMER1_TC Then 1
    Else 0 With TIMER1_GATE = VMON1 & Vmon2 & !TIMER1_TC;
State 1:
    OUT7.J = 1 ; " set OUT7
    OUT7.K = 0 ;
    TIMER1_GATE = VMON3 & VMON4 & !TIMER1_TC;
    If TIMER1_TC Then 2
    Else 1 With TIMER1_GATE = VMON3 & Vmon4 & !TIMER1_TC;

END
```

Combinatorial Example

In this example, a four-analog-input NOR gate is implemented using open-drain output OUT7 as the output and VMON1 - VMON4 as the inputs. This example can be tested on the evaluation board by setting the clock to external and do not connect an external clock source (turn off the clock). Then jumper the VMON inputs to the supply. The LED on OUT7 should turn on when any of the inputs are active.

Figure 12. The PINS Window is Configured Before Generating the ABEL Source


PAC-Designer - [Design1.PAC: Pin Definitions]
File View Options Window Help
PINS
Ready

Datasheet Name	User-defined name	Pin type	Logic Polarity	Reset Level
VMON7	VMON7	Input	Normal (1=true, 0=false)	
VMON8	VMON8			
VMON9	VMON9			
VMON10	VMON10			
VMON11	VMON11			
VMON12	VMON12			
IN1	IN1			
IN2	IN2			
IN3	IN3			
IN4	IN4			
TIMER1_TC	TIMER1_T			
TIMER2_TC	TIMER2_T			
TIMER3_TC	TIMER3_T			
TIMER4_TC	TIMER4_T			
HVOUT1	HVOUT1			Set high
HVOUT2	HVOUT2			Set high
HVOUT3	HVOUT3			Set high
HVOUT4	HVOUT4			Set high
OUT5	OUT5			Set high
OUT6	OUT6			Set high
OUT7	OUT7	Output, combinatorial [non-registered]	Normal (1=true, 0=false)	
OUT8	OUT8	Output, registered, JK flip-flop	Normal (1=true, 0=false)	Set high

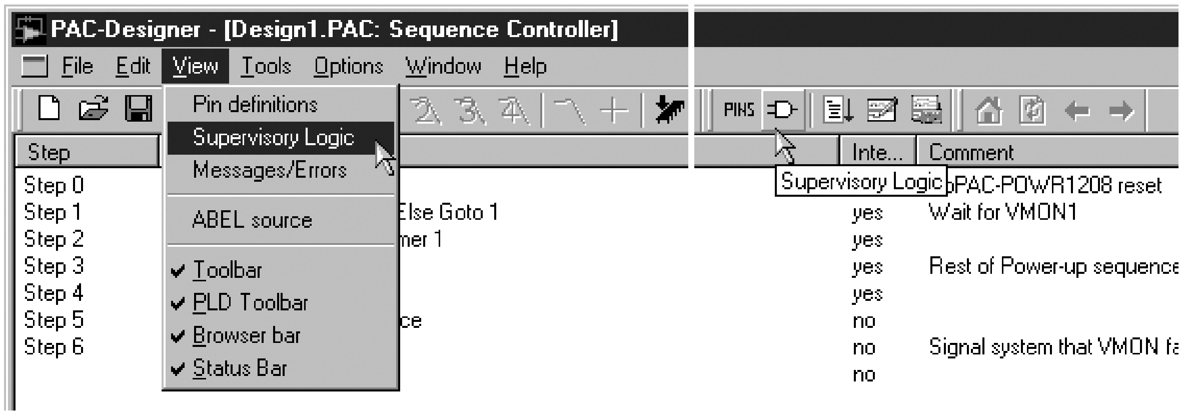
Pin Definition

Datasheet name:	OUT7
User-defined name	OUT7
Pin Number	14 (44-TQFP)
Pin Type	Output, combinatorial [non-registered]
Logic polarity	Normal (1=true, 0=false)
Output type:	Combinatorial (non-registered)
Reset level:	Set low

OK Cancel

For this example, the **Supervisory Logic** window and the **PINS** window will be used to generate the ABEL code as well as discussing direct ABEL edits to implement this function. Use the **PINS** window to set the output type for the OUT7 pin to 'Combinatorial' as shown in Figure 12. Then open the **Supervisory Logic** window using either the menu item **View > Supervisory Logic** or the icon on the toolbar (both are shown in Figure 13).

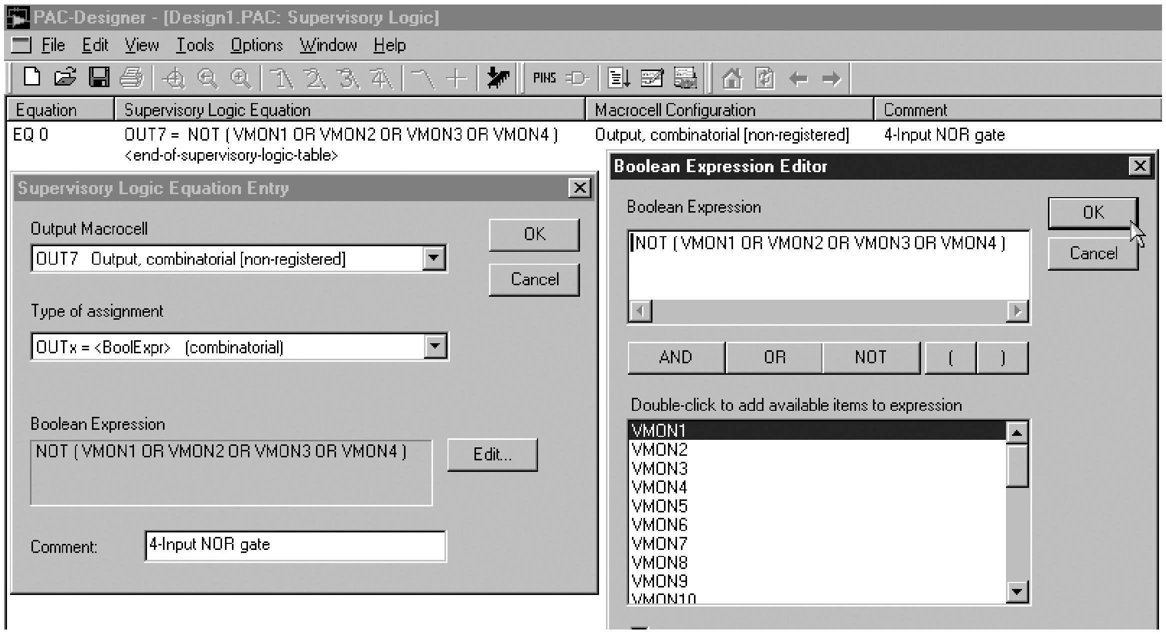
Figure 13. Open the Supervisory Logic Window with Either the Menu or Tool Bar



Step		Inte...	Comment
Step 0			PAC-POWR1208 reset
Step 1	Else Goto 1	yes	Wait for VMON1
Step 2	ner 1	yes	
Step 3		yes	Rest of Power-up sequence
Step 4		yes	
Step 5	ce	no	
Step 6		no	Signal system that VMON fa
		no	

Double-click on the **<end-of-supervisory-logic-table>** marker to insert an equation, and then double-click on the blank equation to bring up the **Supervisory Logic Equation Entry** dialog box (shown in Figure 14). This dialog box is then used to select both the output and enter the Boolean expression that defines the NOR function. The Supervisory Logic Window allows editing multi-input combinatorial or registered equations. However, the Pin-Type must first be set in the **PINS** window. Listing 5 shows relevant portions of the resulting ABEL source that is generated from a LogiBuilder compile of this example.

Figure 14. Supervisory Logic Equation Entry


PAC-Designer - [Design1.PAC: Supervisory Logic]
File Edit View Tools Options Window Help
PINS
Supervisory Logic Equation Entry
OK
Cancel
Edit...
Boolean Expression Editor
OK
Cancel
AND
OR
NOT
(
)
Double-click to add available items to expression

Equation	Supervisory Logic Equation	Macrocell Configuration	Comment
EQ 0	OUT7 = NOT (VMON1 OR VMON2 OR VMON3 OR VMON4)	Output, combinatorial [non-registered]	4-Input NOR gate
	<end-of-supervisory-logic-table>		

Output Macrocell: OUT7 Output, combinatorial [non-registered]

Type of assignment: OUTx = <BoolExpr> (combinatorial)

Boolean Expression: NOT (VMON1 OR VMON2 OR VMON3 OR VMON4)

Comment: 4-Input NOR gate

Boolean Expression: NOT (VMON1 OR VMON2 OR VMON3 OR VMON4)

VMON1
VMON2
VMON3
VMON4
VMON5
VMON6
VMON7
VMON8
VMON9
VMON10

Listing 5. Supervisory Logic Generated ABEL Source Code "Combinatorial"

```
OUT5 pin 12  istype 'reg_JK';
OUT6 pin 13  istype 'reg_JK';
OUT7 pin 14  istype 'com';
OUT8 pin 15  istype 'reg_JK';

" Start of reset assignments
LATTICE PROPERTY 'PRESET OUT8';
" end of reset assignments

equations
" Start of clock assignments
  OUT8.clk = PLD_clk;

" Start of supervisory logic table equations
    OUT7 =  ! ( VMON1 # VMON2 # VMON3 # VMON4 );
 " End of supervisory logic table equations

state_diagram [STATE_FF3, STATE_FF2, STATE_FF1, STATE_FF0]

END
```

Combinatorial logic can also be added to or modified in the ABEL source code. If the output pin type was not set in the **PINS** window before the ABEL source was generated, a few simple edits can be used to implement the source code in Listing 5.

First, the pin type can be changed from registered-JK (**reg_JK**) to combinatorial (**com**) by editing the text between the single quotes after the keyword **istype**. Next, if a clock assignment (**`OUTx.clk = PLD_clk;`**) exists for the subject pin, delete the line or comment it out. Finally, enter the equation after the **equations** keyword but before the **state_diagram** keyword, keeping in mind the symbols for NOT(!), AND(&), and OR(#).

Reset Assignment Example

One last topic to be covered is the reset assignments. After the ABEL editing is enabled, the **PINS** window is grayed out, because changes made in it would not be reflected in the ABEL source code. Thus to either change the power-up state of an output or define it, ABEL source code has to be edited or added. The proper location for the reset assignments is after the pin definitions and before the **`equations`** keyword (see Listing 5). The keywords **`LATTICE PROPERTY`** are used to identify a reset assignment in conjunction with either the **`RESET`** or **`PRESET`** keyword and the pin-name. In Listing 6, OUT5 will be reset at power-up while OUT6 will be preset.

Listing 6. LATTICE PROPERTY Presets and Resets

```
LATTICE PROPERTY 'RESET OUT5'; " OUT5 will be LOW after Power On reset
LATTICE PROPERTY 'PRESET OUT6'; " OUT6 will be HIGH after Power On reset

END
```

Summary

In this application note, we have seen how easy it is to both generate and edit ABEL code to extend the capabilities of Lattice's Power Manager devices beyond those provided when using only the LogiBuilder interface of PAC-Designer software. The important menu items of PAC-Designer to edit, compile, and simulate the ABEL source have been highlighted. A review of the timer hardware and functionality has been provided along with examples that illustrate additional modes of timer usage. It was also shown that both state machine logic and combinatorial logic can be supported in a custom ABEL file. For additional information regarding Lattice's Power Manager Devices or the ABEL language, please consult the references below.

Related Literature

- ispPAC®-POWR1208 Data Sheet
- ispPAC-POWR604 Data Sheet
- Application Note AN6044, *Simulating Power Supply Sequencers with the ispPAC-POWR1208 and PAC-Designer LogiBuilder*
- *ABEL-HDL Reference Manual Version 8.0,* Lattice Semiconductor Corporation.
- *ABEL Design Manual Version 8.0,* Lattice Semiconductor Corporation.
- *Digital Design Using ABEL,* David Pellerin and Michael Holley, 1994, Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632, ISBN 0-13-605874-4
- Lattice Semiconductor Corporation Digital Design Tools Help Files

Technical Support Assistance

Hotline:	1-800-LATTICE (Domestic)
	1-408-826-6002 (International)
e-mail:	ispPACs@latticesemi.com
Internet:	www.latticesemi.com