Technical notes on using Analog Devices DSPs, processors and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Managing the Core PLL on SHARC® Processors

Contributed by Jeyanthi Jegadeesan *Rev 4 – November 18, 2010*

Introduction

This EE-Note describes how to program the core phase-locked loop (PLL) on ADSP-2126x, ADSP-2136x, ADSP-2137x, ADSP-2146x, ADSP-2147x and ADSP-2148x SHARC® processors. This document also provides example code for programming the PLL and discusses possible problems that can occur due to incorrect programming. Lastly, it provides debug techniques for correct PLL handling.

C callable library functions for programming the PLL on all processors are also provided with this EE-Note. This library function can be added to the C run-time library and called from VisualDSP++® project files.

Programming the PLL on SHARC Processors

ADSP-2126x, ADSP-2136x, ADSP-2137x, ADSP-2146x, ADSP-2147x and ADSP-2148x SHARC processors use a PLL to provide clocks that switch at higher frequencies than the clock source (`CLKIN`). The PLL derives the clock for the processor's peripherals in addition to the processor's core and internal memory.

The processor's `CLK_CFG1-0` pins select the core clock (`CCLK`) to `CLKIN` ratio during power-up in hardware. For the ADSP-2147x and ADSP-2148x processors, ratios of 32:1, 16:1, and 8:1 can be selected using the `CLK_CFG` pins during reset. For the ADSP-2136x, ADSP-2137x, and ADSP-2146x processors, ratios of 32:1, 16:1, and 6:1 can be selected using the `CLK_CFG` pins during reset. For the ADSP-2126x processors, ratios of 16:1, 8:1, and 3:1 can be selected using the `CLK_CFG` pins during reset.

The power management control register (`PMCTL`) allows you to program the PLL dynamically in software. The `PMCTL` can be used to select `CCLK`-to-`CLKIN` ratios that are not supported by hardware pins. It can also provide power savings in applications that have time periods during which the full instruction rate is not needed. The `CCLK` rate can be decreased during periods of less-intensive processing.

The PLL block diagram in Figure 1 shows an input divider, multiplier, and a divider for deriving the `CCLK`. Figure 1 shows the PLL block diagram for the ADSP-21367, ADSP-21368, ADSP-21369 (hereafter referred to as ADSP-21368 processors) and ADSP-2137x processors. The PLL block diagram for the ADSP-21362, ADSP-21363, ADSP-21364, ADSP-21365, ADSP-21366 (hereafter referred to as ADSP-21362 processors), and ADSP-2126x processors is functionally the same with the following differences:

Copyright 2006-2010, Analog Devices, Inc. All rights reserved. Analog Devices assumes no responsibility for customer product design or the use or application of customers' products or for any infringements of patents or rights of others which may result from Analog Devices assistance. All trademarks and logos are property of their respective holders. Information furnished by Analog Devices applications and development tools engineers is believed to be accurate and reliable, however no responsibility is assumed by Analog Devices regarding technical accuracy and topicality of the content provided in Analog Devices Engineer-to-Engineer Notes.

- The SDRAM clock (SDCLK) shown in Figure 1 does not apply to ADSP-2126x and ADSP-21362 processors.
- The peripheral clock (PCLK) for the ADSP-2126x processors is the same as the CCLK. For all ADSP-2136x, ADSP-2137x, and ADSP-214xx processors the PCLK is half the CCLK.
- The CCLK of ADSP-2126x processors can be a maximum of 200 MHz, and the CCLK of ADSP-21362 processors can be a maximum of 333 MHz. The CCLK of ADSP-21368 processors can be a maximum of 400 MHz. The CCLK of ADSP-2137x processors can be a maximum of 266 MHz. The CCLK of ADSP-2146x processors can be a maximum of 450 MHz, while the CCLK of ADSP-2147x and ADSP-2148x processors can be a maximum of 266 MHz and 400 MHz, respectively.
- For ADSP-2126x processors, the ratios selected by the CLK_CFG pins are different and the divider values selected by the software is 2, 4, 8 or 16.

The following bits in the PMCTL register are used for programming the PLL:

- PLLMx – These bits select the multiplier value (0 to 64).
- PLLDx – These bits select the divider value (1, 2, 4, or 8 for ADSP-2136x and ADSP-2137x processors, 2, 4, 8 or 16 for ADSP-2126x processors).
- INDIV – This bit selects the input divider, which divides the input clock by 2.
- SDCKR – These bits select the CCLK-to-SDCLK ratio (2.0:1, 2.5:1, 3.0:1, 3.5:1, or 4.0:1). These bits apply only to ADSP-21368 and ADSP-2137x processors. These bits are reserved for the ADSP-2126x and ADSP-21362 processors.
- DIVEN – This bit enables the divisor.

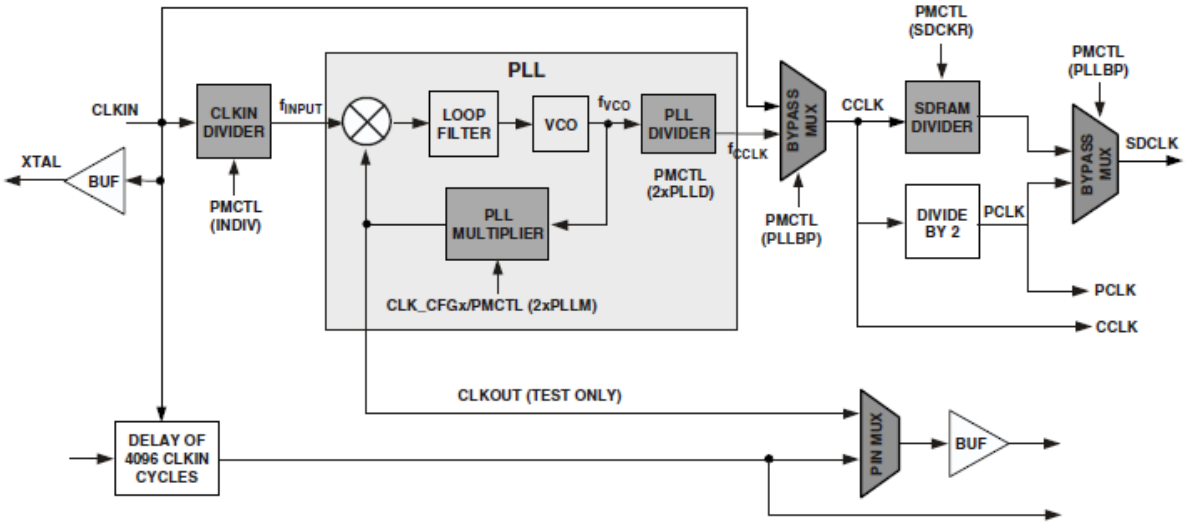

Figure 1. PLL block diagram for ADSP-21368 and ADSP-2137x processors

The Figure 2 shows the PLL block diagram for the ADSP-2146x processors. This block diagram includes the clock dividers for the DDR2 and link port clocks, both of which are only available on the ADSP-2146x processors.

The following bits in the PMCTL register are used for programming the PLL:

- PLLMx – These bits select the multiplier value (0 to 64).
- PLLDx – These bits select the divider value (2, 4, 8 or 16 for ADSP-2146x processors).
- INDIV – This bit selects the input divider, which divides the input clock by 2.
- DDR2CKR – These bits select the CCLK-to-DDR2CLK ratio (2:1, 3:1, or 4:1). These bits apply only to ADSP-2146x processors.
- LCLKR – These bits select the CCLK-to-LCLK ratio (2.0:1, 2.5:1, 3.0:1, or 4.0:1). These bits apply only to ADSP-2146x processors.
- DIVEN – This bit enables the divisor.

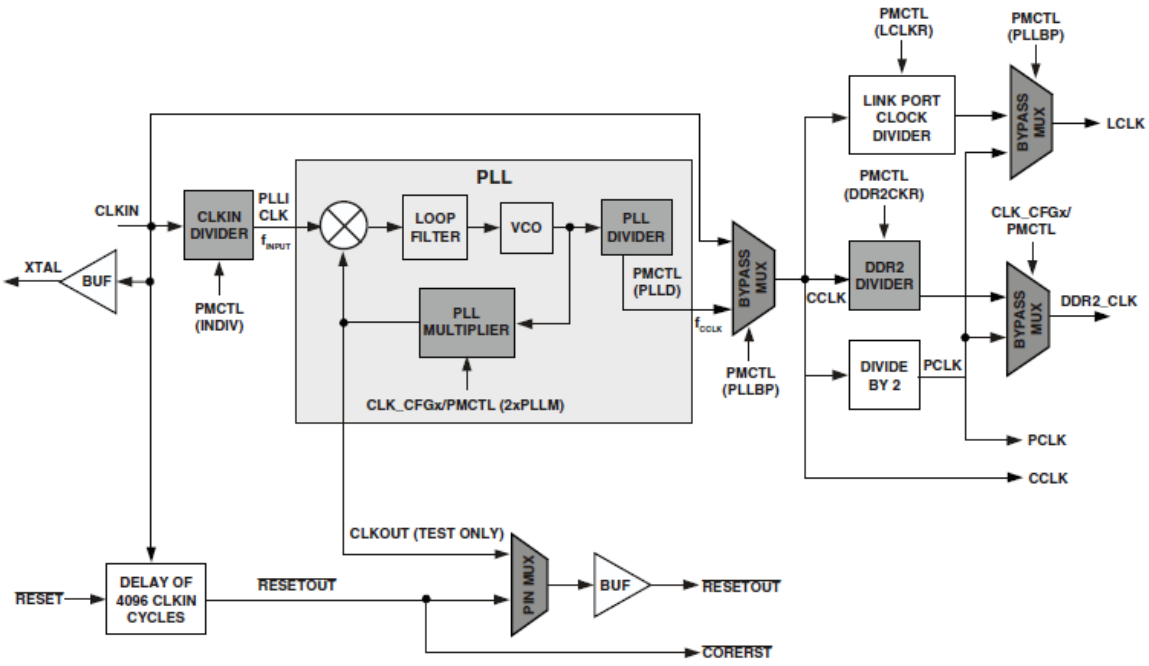

Figure 2. PLL block diagram for ADSP-2146x processors

The Figure 3 shows the PLL block diagram for the ADSP-2147x and ADSP-2148x processors. The following bits in the PMCTL register are used for programming the PLL:

- PLLMx – These bits select the multiplier value (0 to 64).
- PLLDx – These bits select the divider value (2, 4, 8 or 16).
- INDIV – This bit selects the input divider, which divides the input clock by 2.
- SDCKR – These bits select the CCLK-to-SDCLK ratio (2:1, 2.5:1, 3:1, 3.5:1 or 4:1).
- DIVEN – This bit enables the divisor.

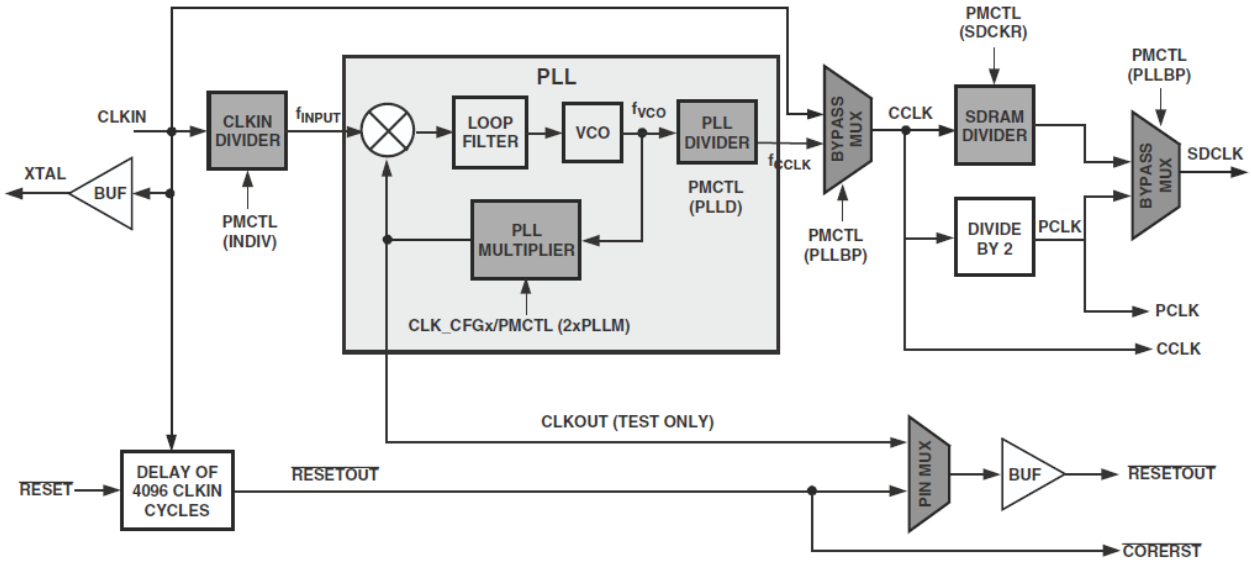

Figure 3. PLL block diagram for ADSP-2147x and ADSP-2148x processors

Steps for Programming the PLL

The input divider, multiplier, and/or divider are used to select the CCLK ratio.

- When using only the multiplier to program the PLL, after setting the multiplier values, the application should place the PLL in bypass mode and wait for the PLL to lock in the new rate. In bypass mode, the processor core runs at the CLKIN frequency. The PLL requires 4096 CCLK cycles (in this case, the CLKIN cycles) to lock at the programmed frequency. Once the PLL is locked, it should be taken out of bypass mode.
- When using only the divider to program the PLL, the divisor value can be set using the PLLDx bits, and the divisor can be enabled using the DIVEN bit in the same instruction. Wait a minimum of 16 CCLK cycles for the new divider values to take effect.

Use either of the following two approaches to program the PLL using both the multiplier and divider:

Approach #1

1. Set the PLL multiplier and divisor value and enable the divider by setting (=1) the DIVEN bit.
2. After one CCLK cycle, place the PLL in bypass mode by setting (=1) the bypass (PLLBP) bit. Ensure that the DIVEN bit is cleared before writing to the PMCTL register.
3. Wait a minimum of 4096 CCLK cycles in bypass mode until the PLL locks.
4. Take the PLL out of bypass mode by clearing (=0) the bypass bit (PLLBP). Ensure that the DIVEN bit is cleared before writing to the PMCTL register.
5. Wait a minimum of 16 CCLK cycles for the new divider values to take effect.

Approach #2

1. Set the PLL multiplier and divisor values and place the PLL in bypass mode by setting (=1) the bypass (PLLBP) bit.
2. Wait a minimum of 4096 CCLK cycles in bypass mode until the PLL locks.
3. Take the PLL out of bypass mode by clearing (=0) the bypass (PLLBP) bit.
4. Wait for one CCLK cycle.
5. Enable the divider by setting (=1) the DIVEN bit.
6. Wait a minimum of 16 CCLK cycles for the new divider values to take effect.

Considerations

Be aware of the following considerations while programming the PMCTL register:

- When the PLL is programmed using a multiplier and a divider, *do not* enable the DIVEN and PLLBP bits during the same CCLK cycle. There should be a delay of at least one CCLK cycle between programming these bits.
- In the user application, select the PLL multiplier value according to the following conditions:
  - The product of CLKIN and PLLM must never exceed half of the maximum VCO (f_{VCO}) frequency if the Input divider is not enabled (INDIV = 0).
  - The product of CLKIN and PLLM must never exceed the maximum VCO (f_{VCO}) frequency if the input divider is enabled (INDIV = 1).

  This limitation is due that the VCO frequency programmed must never exceed the maximum VCO (f_{VCO}) frequency for a specific processor. Please refer the datasheet of the specific processor for the minimum and maximum VCO frequency values. The VCO frequency is calculated as follows:

  $f_{VCO} = 2 * PLLM * f_{INPUT}$

  Equation 1. VCO frequency calculation

  The core clock is calculated as follows for the ADSP-2126x, ADSP-2136x, and ADSP-2137x processors.

  $f_{CCLK} = (2 * PLLM * f_{INPUT}) / (2 * PLLN)$

  Equation 2. CCLK frequency calculation for ADSP-2126x, ADSP-2136x, and ADSP-2137x processors

  The core clock is calculated as follows for the ADSP-214xx processors.

  $f_{CCLK} = (2 * PLLM * f_{INPUT}) / (PLLN)$

  Equation 3. CCLK frequency calculation for ADSP-214xx processors

where:

f_{VCO} = VCO frequency

f_{CCLK} = CCLK frequency

PLLM = Multiplier value programmed

PLLN = Divider value programmed

f_{INPUT} = Input frequency to the PLL

f_{INPUT} = CLKIN when the input divider is disabled, or CLKIN/2 when the input divider is enabled

- Clear the DIVEN bit during the write to the PMCTL register while placing the PLL in bypass mode (by setting the PLLBP bit to 1) or bringing it out of bypass mode (by clearing the PLLBP bit).
- Set the DIVEN bit while setting any of the core clock to peripheral clock ratio. For example, the CCLK-to-SDCLK ratio, CCLK-to-DDR2CLK ratio, or CCLK-to-LCLK ratio bits in the PMCTL register. After changing the ratios wait for at least 16 CCLK cycles for the new divider values to take effect.

Example Code

The assembly listings in this section demonstrate various ways of programming the PLL using the multiplier, divider, or both. Listing 1 shows an example of PLL programming using the divider.

```
InitPLL:

//Set and enable PLL Divider for 8
    ustat1 = dm(PMCTL);
    bit set ustat1 PLLD8|DIVEN;
    dm(PMCTL) = ustat1;

//Wait for 16 CCLK cycles at least for the new divider values to take effect
    lcntr = 16, do wait_loop until lce;
    wait_loop: nop;
    rts;
```

Listing 1. PLL programming using divider only

Listing 2 shows an example of programming the PLL using only the multiplier.

```
InitPLL:
//Set the multiplier value for 8 and
//place the PLL in bypass mode
    ustat1 = dm(PMCTL);
    bit set ustat1 PLLM8|PLLBP;
    dm(PMCTL) = ustat1;

//Wait for 4096 cycles min for the PLL
//to lock at the programmed rate
    r0 = 5000;
    lcntr = r0, do delay until lce;
    delay: nop;
```

```
//Take the PLL out of bypass mode
    ustat1 = dm(PMCTL);
    bit clr ustat1 PLLBP;
    dm(PMCTL) = ustat1;

    rts;
```

Listing 2. PLL programming using multiplier only

Listing 3 and Listing 4 show examples of programming the PLL using both the multiplier and divider.

```
InitPLL:

//Set the multiplier value for 8, //divider value of 2 and enable the
//divider
    ustat1 = dm(PMCTL);
    bit set ustat1 PLLD2|DIVEN| PLLM8;
    dm(PMCTL) = ustat1;

//Place the PLL in bypass mode and   // disable the divider
    bit set ustat1 PLLBP;
    bit clr ustat1 DIVEN;
    dm(PMCTL) = ustat1;

//Wait for 4096 cycles min for the PLL
//to lock at the programmed rate
    r0 = 5000;
    lcntr = r0, do delay until lce;
    delay: nop;

//Take the PLL out of bypass mode
// Reading the PMCTL register value will return the DIVEN bit value as zero
    ustat1 = dm(PMCTL);
    bit clr ustat1 PLLBP;
    dm(PMCTL) = ustat1;

//Wait for 16 CCLK cycles at least for the new divider values to take effect
    lcntr = 16, do wait_loop until lce;
    wait_loop: nop;
    rts;
```

Listing 3. PLL programming using multiplier and divider (approach #1)

```
InitPLL:

//Set the multiplier value for 8, divider value of 2 and
//place the PLL in bypass mode
    ustat1 = dm(PMCTL);
    bit set ustat1 PLLM8|PLLBP|PLLD2;
    dm(PMCTL) = ustat1;

//Wait for 4096 cycles min for the PLL to lock at the programmed rate
    r0 = 5000;
    lcntr = r0, do delay until lce;
```

```
    delay: nop;

//Take the PLL out of bypass mode
    ustat1 = dm(PMCTL);
    bit clr ustat1 PLLBP;
    dm(PMCTL) = ustat1;

//Enable the divider
    bit set ustat1 DIVEN;
    dm(PMCTL) = ustat1;
//Wait for 16 CCLK cycles at least for the new divider values to take effect
    lcntr = 16, do wait_loop until lce;
    wait_loop: nop;
    rts;
```

Listing 4. PLL programming using multiplier and divider (approach #2)

Listing 5 shows an example of incorrect programming of the PLL:

```
InitPLL:
//Set the multiplier value for 8, divider value of 2 and enable the divider
    ustat1 = dm(PMCTL);
    bit set ustat1 PLLD2|DIVEN| PLLM8;
    dm(PMCTL) = ustat1;


//Place the PLL in bypass mode and the divider is not disabled - this
//can cause the problem
    bit set ustat1 PLLBP;
    dm(PMCTL) = ustat1;

//Wait for 4096 cycles min for the PLL to lock at the programmed rate

    r0 = 5000;
    lcntr = r0, do delay until lce;
    delay: nop;

//Take the PLL out of bypass mode
    ustat1 = dm(PMCTL);
    bit clr ustat1 PLLBP;
    dm(PMCTL) = ustat1;
//Wait for 16 CCLK cycles at least for the new divider values to take effect
    lcntr = 16, do wait_loop until lce;
    wait_loop: nop;
    rts;
```

Listing 5. Incorrect PLL programming example

Effects of Incorrect PLL Programming

The PLL is used to derive the core clock and the clock for the peripherals. If the PLL is not programmed correctly, the PLL does not lock to the desired frequency. This can cause the core (as well as the peripherals) to behave unpredictably.

Debug Help for PLL Handling

The PLL is the heart of the SHARC processors and requires robust hardware and software handling. Whenever the user is faced with problems based on PLL changes the following can be checked out in the system:

- Is the power-up specification met according to the data sheet (valid core and IO voltage, input clock, reset signal)?
- Are the `CLK_CFG` pins static settings (may change only during reset)?
- Did the core wait at least 5000 `CLKIN` cycles in a loop after entering bypass?
- If required to change multiplier settings, was PLL entered in bypass mode?
- Is the `RESETOUT` pin asserted (indicating that the 4096 cycles are elapsed)?
- Did the core wait at least 15 `CCLK` cycles in a loop after changing any of the divider value?

initPLL Library Function

The `initPLL_2126x`,`initPLL_21362`,`initPLL_21368`,`initPLL_2137x`, `initPLL_2146x`, `initPLL_2147x`, and `initPLL_2148x` functions are the assembly library functions implemented for programming the PLL on third and fourth generation SHARC processors. These functions can be added to the VisualDSP++ C run-time library. After adding to the C run-time library, these functions can be called from any project file.

How to Use

The `initPLL_2126x`, `initPLL_21362`, `initPLL_21368`, `initPLL_2137x`, `initPLL_2147x` and `initPLL_2148x` functions accept the following four parameters:

- PLL multiplier value – (0 to 64)
- PLL divisor value – (0, 1, 2, or 3). For ADSP-2136x and ADSP-2137x processors these values correspond to the divider values of 1, 2, 4 or 8 respectively. For ADSP-2126x, ADSP-2147x, and ADSP-2148x processors these values correspond to the divider values of 2, 4, 8 or 16 respectively.
- `SDCLK` ratio – `CCLK`-to-`SDCLK` ratio (0, 1, 2, 3, or 4). The SDCLK ratio is not used on ADSP-2126x and ADSP-21362 processors. For ADSP-2136x, ADSP-2137x, ADSP-2147x, and ADSP-2148x processors these values correspond to the SDCLK ratio values of 2, 2.5, 3, 3.5 or 4 respectively.
- Input divider – Binary value. When true, enables the input divider.

The `initPLL_2146x` function accepts the following five parameters:

- PLL multiplier value – (0 to 64)
- PLL divisor value – (0, 1, 2, or 3). These values correspond to the divider values of 2, 4, 8 or 16 respectively.
- `DDR2CLK` ratio – `CCLK`-to-`DDR2CLK` ratio (0, 2, or 4). These values correspond to the DDR2CLK ratio values of 2, 3, or 4 respectively.
- Input divider – Binary value. When true, enables the input divider.
- `LCLK` ratio - `CCLK`-to-`LCLK` ratio (0, 1, 2, or 3). These values correspond to the LCLK ratio values of 2, 2.5, 3, or 4 respectively.

When a negative value is passed as the parameter for any of the above values, the corresponding parameter is not set.

The prototypes of the library functions for initializing the PLL are defined as follows:

```
void initPLL_2126x(int multi, int div, int sdckr, boolean indiv);
void initPLL_21362(int multi, int div, int sdckr, boolean indiv);
void initPLL_21368(int multi, int div, int sdckr, boolean indiv);
void initPLL_2137x(int multi, int div, int sdckr, boolean indiv);
void initPLL_2146x(int multi, int div, int ddckr, boolean indiv, int lckr);
void initPLL_2146x _anomaly_1500020_Workaround (int multi, int div, int ddckr,
boolean indiv, int lckr);
void initPLL_2147x(int multi, int div, int sdckr, boolean indiv);
void initPLL_2147x_anomaly_1500020_Workaround (int multi, int div, int sdckr,
boolean indiv);
void initPLL_2148x(int multi, int div, int sdckr, boolean indiv);
void initPLL_2148x_anomaly_1500020_Workaround (int multi, int div, int sdckr,
boolean indiv);
```

This function can be called from a C function as follows:

```
#include <initPLL.h>
initPLL_21368(8,4,3,false);
```

Adding initPLL to the C Run-Time Library

The VisualDSP++ archiver (`elfar`) combines object files with the existing library functions. The object file generated for the `initPLL_2126x`, `initPLL_21362`, `initPLL_21368`, `initPLL_2137x`, `initPLL_2146x`, `initPLL_2147x and initPLL_2148x` functions can be added to the existing C run-time library using the `elfar` utility. The `elfar` utility can be called from the command prompt. This utility is available under:

`C:\Program Files\Analog Devices\ VisualDSP x.x`

The following section describes how to add the `initPLL_2146x` function to the ADSP-2146x C run-time library (see Figure 4) The same sequences also apply to ADSP-2126x, ADSP-21362, ADSP-21368, ADSP-2137x, ADSP-2147x and ADSP-2148x processors. The associated `ZIP` file contains `initPLL` functions for each processor.

For ADSP-2146x, ADSP-2147x and ADSP-2148x processors, the C run-time library files (`libc.dlb, libc_nwc.dlb`) are available under:

```
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib\21469_rev_any
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\ lib\ 21479_rev_0.0
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib\21479_rev_any
```

To create the `initPLL_2146x.doj` file for the ADSP-2146x C run-time library:

- Copy the `initPLL_2146x.asm` file for ADSP-2146x processors to the following folder:

```
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib\src\libc_src
```

- Copy the `initPLL.h` file for ADSP-2146x processors to the following folder:

```
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\include
```

- In a command prompt window, change the path to the following folder:

```
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib\src\libc_src
```

- Run the following command to create the `initPLL_2146x.doj` file under the library folder:

```
C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib
"C:\Program Files\Analog Devices\VisualDSP 5.0\easm21k.exe" –proc ADSP-21469 –o "C:\Program Files\Analog Devices\VisualDSP 5.0\214xx\lib\initPLL_2146x.doj" initPLL_2146x.asm
```

To add the `initPLL_2146x.doj` file to the ADSP-2146x C run-time library:

- In a command prompt window, change the path to the following folder (from which the `initPLL_2146x.doj` file and the library files are available):

```
C:\Program Files\Analog Devices\ VisualDSP 5.0\214xx\lib
```

- Run the following command to add the `initPLL_2146x.doj` function to the library for the first time:

```
"C:\Program Files\Analog Devices\VisualDSP 5.0\elfar.exe" –a libc.dlb initPLL_2146x.doj
"C:\Program Files\Analog Devices\ VisualDSP 5.0\elfar.exe" –a libc_nwc.dlb initPLL_2146x.doj
```

- You may run the following command to update the `initPLL_2146x.doj` function to the library:

```
"C:\Program Files\Analog Devices\VisualDSP 5.0\elfar.exe" –r libc.dlb initPLL_2146x.doj
"C:\Program Files\Analog Devices\ VisualDSP 5.0\elfar.exe" –r libc_nwc.dlb initPLL_2146x.doj
```

This command will replace the existing .obj file with the updated one.

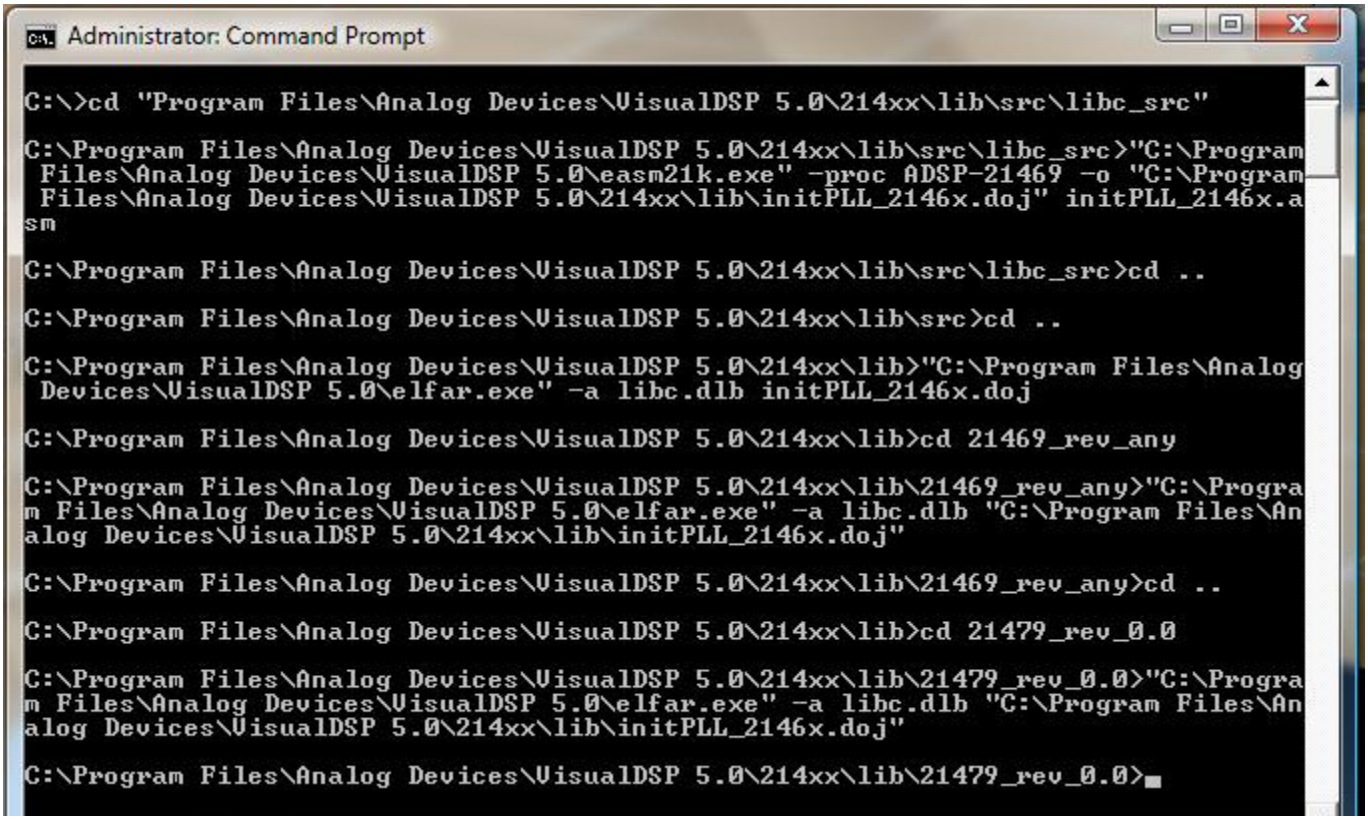

Figure 4. Commands for adding the initPLL_2146x to the C runtime library

The `initPLL` library functions should also be added to all the C run time library files for a specific processor.

For ADSP-2126x processors, the C Runtime library `libc26x.dlb` file is available under the following paths:

`C:\Program Files\Analog Devices\VisualDSP 5.0\212xx\lib`

`C:\Program Files\Analog Devices\VisualDSP 5.0\212xx\lib\2126x_any`

`C:\Program Files\Analog Devices\VisualDSP 5.0\ 212xx\ lib\ 2126x_rev_0.0`

For ADSP-2136x and ADSP-2137x processors, the C Runtime library `libc36x.dlb` and `libc37x.dlb` files are available under the following paths:

`C:\Program Files\Analog Devices\VisualDSP 5.0\213xx\lib`

`C:\Program Files\Analog Devices\VisualDSP 5.0\213xx\lib\2136x_any`

`C:\Program Files\Analog Devices\VisualDSP 5.0\ 213xx\ lib\ 2136x_rev_0.0`

The C Runtime library file `libc36x.dlb` is also available under the following path:

`C:\Program Files\Analog Devices\VisualDSP 5.0\ 213xx\ lib\ 2136x_LX3_rev_0.1`

`C:\Program Files\Analog Devices\VisualDSP 5.0\ 213xx\ lib\ 21369_rev_any`

Alternately, the Batch files available with the source code can also be used for adding the initPLL library function to the corresponding runtime library. Unzip the initPLL assembly and header files, batch files into the local machine. Use the below steps to add the initPLL_2146x library function to the 214xx runtime library files. The batch file takes the path of the folder where the initPLL header and assembly files are copied as the first parameter and it takes the path where the VDSP++ tool is installed as the next parameter.

- Unzip the example code available to the below path:

```
D:\EE290V04
```

- In command prompt change the path to the below one:

```
cd D:\EE290V04\ Batch_Files
```

- Run the below command:

```
2146x.bat D:\EE290V04\214xx C:\Program Files\Analog Devices\VisualDSP 5.0
```

Summary

This EE-Note demonstrates how to program the PLL on SHARC processors. It describes the `initPLL` library functions and also provides the procedure for integrating them with the existing VisualDSP++ C run-time library.

The `initPLL` function source code and object files for the ADSP-2126x, ADSP-2136x, ADSP-2137x, ADSP-2146x, ADSP-2147x, and ADSP-2148x processors are provided in the associated `ZIP` file.

References

[1] *ADSP-2126x SHARC Processor Peripherals Manual.* Rev 3.0, December 2005. Analog Devices, Inc.

[2] *ADSP-2136x SHARC Processor Hardware Reference for the ADSP-21362/3/4/5/6 Processors.* Rev 2.0, June 2009. Analog Devices, Inc.

[3] *ADSP-21368 SHARC Processor Hardware Reference.* Rev 1.0, September 2006. Analog Devices, Inc.

[4] *ADSP-214xx SHARC Processor Hardware Reference.* Rev 0.3, July 2010. Analog Devices, Inc

[5] *VisualDSP++ 5.0 Linker and Utilities Manual*. Rev 3.3, September 2009. Analog Devices, Inc.

[6] *ADSP-21261/ADSP-21262/ADSP-21266 SHARC Embedded Processors datasheet.* Rev F, August 2009. Analog Devices, Inc.

[7] *ADSP-21362/ADSP-21363/ADSP-21364/ADSP-21365/ADSP-21366 SHARC Processors datasheet.* Rev F, April 2010. Analog Devices, Inc.

[8] *ADSP-21367/ADSP-21368/ADSP-21369 SHARC Processors datasheet.* Rev E, July 2009. Analog Devices, Inc.

[9] *ADSP-21371/ADSP-21375 SHARC Processors datasheet.* Rev C, September 2009. Analog Devices, Inc.

[10] *ADSP-21469 SHARC Processor datasheet.* Rev 0, June 2010. Analog Devices, Inc.

[11] *ADSP-21478/ADSP-21479 SHARC Processor datasheet.* Rev PrB, April 2010. Analog Devices, Inc.

[12] *ADSP-21483/ADSP-21486/ADSP-21487/ ADSP-21488/ADSP-21489 SHARC Processor datasheet. Rev PrA, April 2010. Analog Devices, Inc.*

Document History

Revision	Description
Rev 4 – November 18, 2010 *by Jeyanthi Jegadeesan*	Added information on the ADSP-2147x and ADSP-2148x processors. Furthermore, tested the code under VisualDSP++ 5.0 Update 8.
Rev 3 – December 24, 2009 *by Jeyanthi Jegadeesan*	Changed the document title from "Managing the Core PLL on Third-Generation SHARC Processors" to "Managing the Core PLL on SHARC Processors" to include information on the ADSP-2146x processors. Furthermore, tested the code under VisualDSP++ 5.0 Update 7.
Rev 2 – May 16, 2007 *by Jeyanthi Jegadeesan*	Generalized the EE-Note for ADSP-2126x, ADSP-2136x and ADSP-2137x processors, and updated document title accordingly.
Rev 1 – June 23, 2006 *by Jeyanthi Jegadeesan*	Initial revision.