

## Introduction

This application note describes the operational characteristics of the phase-locked loops (PLLs) that are located in the embedded stripe of Excalibur™ embedded processor PLDs, and presents various design requirements that affect selection of the PLL control parameters.



To make best use of this application note, you should be familiar with the architecture of the Excalibur devices, which is detailed in the *Excalibur Devices Hardware Reference Manual*.



Refer to “[Revision History](#)” on page 13 to see the changes made for this version of the document.



EXCALIBUR™

In Excalibur devices, two dedicated PLLs in the embedded stripe provide clock-multiplier circuits for synthesizing the embedded stripe clock frequencies:

- PLL1 provides the clock for the embedded processor bus (CLK\_AHB1) and the peripheral bus (CLK\_AHB2)
- PLL2 provides the clocks (CLK\_SDRAMx2 and CLK\_SDRAMx1) for the SDRAM controller



A wide variety of clocking values can be created from a single clock input by setting the registers that govern the stripe PLLs.

CLK\_AHB1 and CLK\_AHB2 are one half and one fourth, respectively, of the PLL1 output clock; CLK\_SDRAMx2 is equal to, and CLK\_SDRAMx1 is one-half of, the PLL2 output clock. The CLK\_AHB1 clock operates at a maximum of 200 MHz, which limits the maximum-allowed frequency of PLL1 to 400 MHz. The CLK\_SDRAMx2 clock operates at a maximum of 266 MHz, which limits the maximum-allowed frequency of PLL2 to 266 MHz.



This document discusses the measured operational characteristics and control of PLL1—the measured operational characteristics and control for PLL2 are similar.

In the recommended tool flow, the Excalibur MegaWizard® Plug-In determines the PLL control parameters based on the input clock frequency, `CLK_REF`, and the desired frequency of operation for the embedded processor bus, AHB1. The wizard calculates the nearest-possible desired frequency using a fixed algorithm. In typical applications, the output of the wizard is sufficient, but in some instances a user might wish to control the PLLs or use other settings specific to the application.


 This application note assumes that you are using the Quartus® II software version 2.1 or higher.

Figure 1 shows the circuit for synthesizing the `CLK_AHB1` and `CLK_AHB2` clocks. You can decide whether `CLK_AHB1` and `CLK_AHB2` originate from the `CLK_REF` signal (i.e., bypass mode) or from the output of the voltage control oscillator (VCO). The output of the phase comparator controls the VCO. The phase comparator generates an error signal based on the `CLK_REF` input and the feedback signal from the VCO. The parameters  $N$ ,  $M$ , and  $K$  perform the clock multiplication. The lock detection circuit monitors the phase error between the `CLK_REF`/ $N$  signal and the feedback clock. The signals are considered to be locked when the margin of error is within the lock window.

Figure 1. PLL 1 Circuit

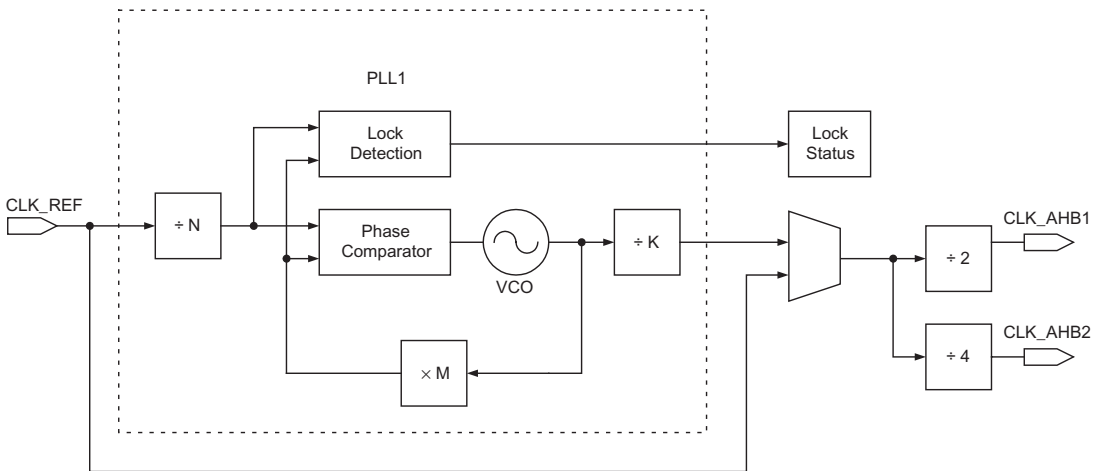
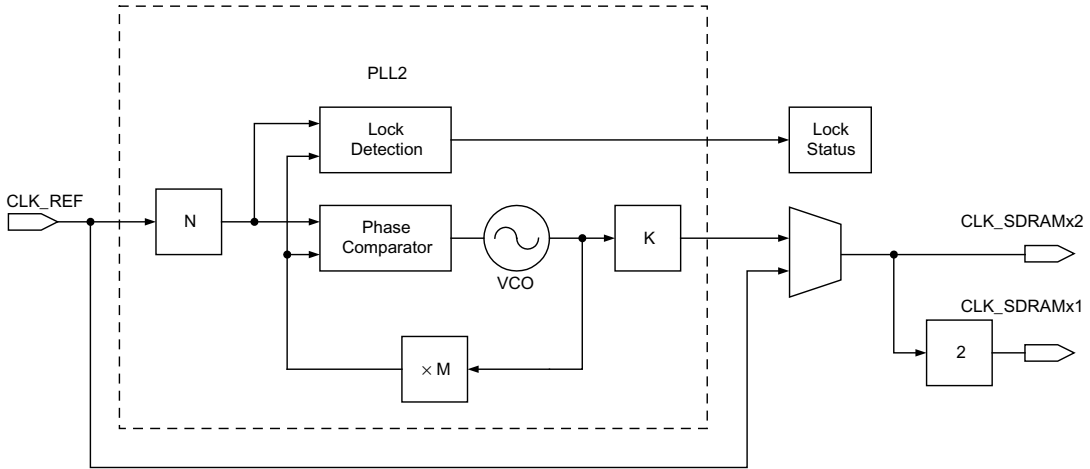


Figure 2 shows the circuit for synthesizing the CLK\_SDRAMx2 and CLK\_SDRAMx1 clocks. You can decide whether CLK\_SDRAMx2 and CLK\_SDRAMx1 originate from CLK\_REF (i.e., bypass mode) or from the output of the voltage control oscillator (VCO).

Figure 2. PLL 2 Circuit



$N$ ,  $M$ , and  $K$  control clock boost synthesis for the PLL through the embedded stripe registers: CLK\_PLL $_y$ \_NCNT, CLK\_PLL $_y$ \_MCNT, and CLK\_PLL $_y$ \_KCNT, where  $y$  identifies the PLL number, either 1 or 2.

The following equations are used to program the PLL:

$$F_{VCO} = CLK\_REF \times M / N$$

$$PLL_{out} = F_{VCO} / K$$

where the device parameter settings given in Table 1 apply:

Device	$N$	$M$	$K$	CLK_REF (MHz)	$F_{VCO}$ (MHz)
EPXA10	1 to 15	1 to 15	1 to 7	10 to 66	160 to 600
EPXA4	1 to 255	1 to 255	1 to 255	1 to 66	160 to 600
EPXA1	1 to 255	1 to 255	1 to 255	1 to 66	160 to 600

The following equation is used to derive the processor, peripheral, and SDRAM controller clocks from the PLL outputs:

$$\text{CLK\_AHB1} = \text{PLL1}_{\text{out}}/2$$

$$\text{CLK\_AHB2} = \text{PLL1}_{\text{out}}/4$$

$$\text{CLK\_SDRAMx2} = \text{PLL2}_{\text{out}}$$

$$\text{CLK\_SDRAMx1} = \text{PLL2}_{\text{out}}/2$$

Where CLK\_AHB1 and CLK\_AHB2 have the frequency range shown in [Table 2](#), depending on the speed grade of the device:

**Table 2. Clock Frequency Versus Speed Grade**

Speed Grade	CLK_AHB1 (MHz)	CLK_AHB2 (MHz)	CLK_SDRAMx2 (MHz)	CLK_SDRAMx1 (MHz)
-1	200	100	266	133
-2	166	83	200	100
-3	133	66	166	83

## Setting up the Stripe PLLs with the Quartus II Software

On power-up, the PLLs are set to the bypass mode. CLK\_AHB1 and CLK\_AHB2 are clocked from the CLK\_REF signal at one half and one fourth, respectively. Early in the boot-from-flash process, the embedded processor programs the PLL control registers with the user settings supplied by the Quartus II software. The boot code waits for the L1 bit in the CLK\_STATUS register to be set, indicating that the output of the PLL is locked to the input CLK\_REF setting. After achieving lock, the boot code clears the BP1 bit in the CLK\_DERIVE register and the CLK\_AHB1 and CLK\_AHB2 signals operate from the output of the VCO.



“PLL Control Register Values” on page 8 describes the PLL registers and their contents. The registers can be configured under embedded processor control or through the configuration bit stream of the PLD. This document only describes the procedures for setting the registers under embedded processor control.

The Quartus II software uses an algorithm to determine the lock window settings and values for  $N$ ,  $M$ , and  $K$ , which must be properly set to achieve lock:

$$\text{Maximum Lock Window Setting} \leq 20\% \times (\text{input clock period}) \times (N \text{ counter setting})$$

Therefore:

- If  $\text{CLK\_REF}/N \leq 20 \text{ MHz}$ , the maximum lock window setting is 10 ns
- If  $\text{CLK\_REF}/N > 20 \text{ MHz}$ , the maximum lock window setting is 2.5 ns

The wizard determines values for  $N$ ,  $M$ , and  $K$  that meet the following objectives:

- Achieve the closest frequency to the desired target frequency without exceeding the target
- Set the VCO to the highest-possible frequency range, to minimize the clock jitter; this typically results in  $N$  being 1 and  $K$  being 2



$K$  is only set to 2 or 4 for PLL2. Odd values of  $K$  do not generate the 50:50 duty cycle clock required for the SDRAM controller in DDR mode.

For example, if you configure the stripe PLL for a reference clock of 50 MHz, a desired AHB1 clock of 166 MHz, and an SDRAM clock of 133 MHz, the Quartus II software creates a system build descriptor (.sbd) file that includes the parameter settings given below:

```
parameters clocks
{
    input_clock_frequency = 50000000;
    desired_ahb1_clock_frequency = 166000000;
    desired_sdram_clock_frequency = 133000000;
    pll_ahb_bypass = false;
    pll_ahb_m = 13;
    pll_ahb_n = 1;
    pll_ahb_k = 2;
    ahb1_divide = 2;
    ahb2_divide = 4;
    pll_sdram_bypass = true;
    pll_sdram_m = 10;
    pll_sdram_n = 1;
    pll_sdram_k = 2;
}
```

You can check the memory initialization file, **memory.regs**, for the actual values programmed into the PLL registers. This file is created by Quartus II after compiling the design, with the values for the PLL settings shown below:

```
3/30000040000;           # CLK_PLL_AHB_NCNT
4/30400010706;           # CLK_PLL_AHB_MCNT
5/30800020101;           # CLK_PLL_AHB_KCNT
6/30C00001055;           # CLK_PLL_AHB_CTRL
7/31000040000;           # CLK_PLL_SDRAM_NCNT
8/31400020505;           # CLK_PLL_SDRAM_MCNT
9/31800020101;           # CLK_PLL_SDRAM_KCNT
A/31C00001055;           # CLK_PLL_SDRAM_CTRL
B/32000000010;           # CLK_DERIVE
```

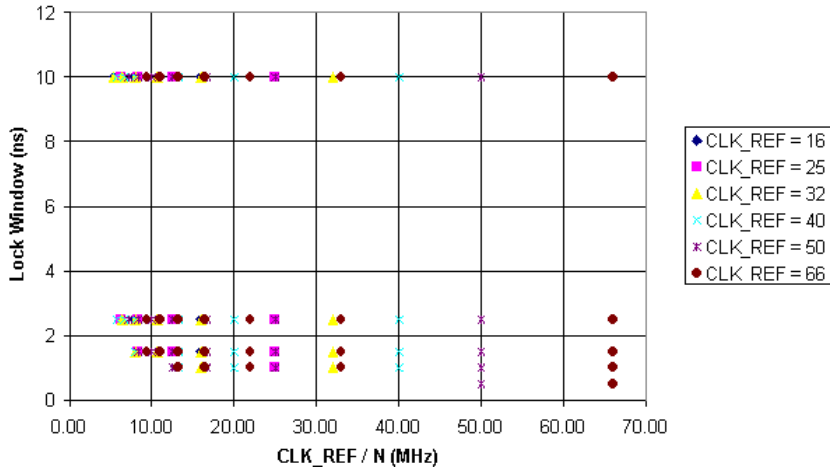


In the PLL settings above, the first 12 bits following the / indicate the register offset address, with the remaining 32 bits denoting the register value. Data is presented in hexadecimal format.

## Operating the Stripe PLLs under User Control

For some applications, you might want to take control of the operation of the embedded stripe PLLs by including PLL programming instructions in the software code that boots after the PLD has been configured. The PLL programming instructions insert values for  $N$ ,  $M$ ,  $K$  and the lock window into the PLL control registers. However, the PLLs must be put into bypass mode prior to changing the values for  $M$ ,  $N$ , and  $K$ , which you do by setting the BP1 bit in CLK\_DERIVE.

All the factors,  $N$ ,  $M$ ,  $K$  and the lock window must be properly set to achieve lock. [Figure 3](#) shows combinations of the input frequency to the phase comparator and  $CLK\_REF/N$  that achieve a stable lock for the specified lock window setting.

**Figure 3. Lock Window Values in Relation to the Phase Comparator Input**

The guidelines for selecting the lock window settings are as follows:

- If  $\text{CLK\_REF} / N \leq 10$  MHz, the minimum lock window setting is 10 ns
- If  $\text{CLK\_REF} / N > 10$  MHz, the minimum lock window setting is 2.5 ns
- If  $\text{CLK\_REF} / N > 15$  MHz, the minimum lock window setting is 1.5 ns
- If  $\text{CLK\_REF} / N > 20$  MHz, the minimum lock window setting is 1 ns

After selecting the lock window settings, you select values for  $M$  and  $K$ .  $M$  determines the VCO output frequency. The output frequency must be  $2 \times K$  times the desired frequency for  $\text{CLK\_AHB1}$ , and within the range of the VCO; i.e.,

$$160 \text{ MHz} \leq (\text{CLK\_AHB1} \times 2 \times K) \leq 600 \text{ MHz}$$

For example, in a design requiring a  $\text{CLK\_REF}$  signal at 50 MHz and a desired  $\text{CLK\_AHB1}$  speed of 150 MHz, there are 3 possible  $N$  values. [Table 3 on page 8](#) shows the  $M$ ,  $K$ , and lock window values that produce the closest output frequency to the desired  $\text{CLK\_AHB1}$  frequency. As listed, only two selections are possible.

**Table 3. *M*, *K*, and Lock Window Settings for EPXA10**

N	CLK_REF/N	Lock Window (Minimum, ns)	M	K	CLK_AHB1 (MHz)
1	50	2.5	6	1	150
2	25	2.5	12	1	150
3	12.5	10	15	1	94(1)

**Note:**

(1) Does not meet design specification.

## PLL Control Register Values

This section describes the registers necessary to program and check the lock status information on the embedded stripe PLL (PLL1). Four registers are used to set *N*, *M*, *K*, and the lock window:

- CLK\_PLL1\_NCNT
- CLK\_PLL1\_MCNT
- CLK\_PLL1\_KCNT
- CLK\_PLL1\_CTRL

### CLK\_PLL1\_xCNT

The three registers CLK\_PLL1\_xCNT, where *x* represents *N*, *M*, or *K*, set the multiplication and division factors for PLL1. To ensure data integrity and maximum PLL stability, the registers setting the *N*, *M*, and *K* values use an encoding scheme.

The encoded values for *N*, *M*, and *K* are shown in [Table 4](#).



Refer to the *Excalibur Devices Hardware Reference Manual* for details on how to calculate the encoded values.

**Table 4. Encoded Values for *N*, *M*, & *K* (Part 1 of 3)**

N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded
1	40000H	52	21A1AH	103	13433H	154	24D4DH	205	16766H
2	20101H	53	11B1AH	104	23434H	155	14E4DH	206	26767H
3	10201H	54	21B1BH	105	13534H	156	24E4EH	207	16867H
4	20202H	55	11C1BH	106	23535H	157	14F4EH	208	26868H
5	10302H	56	21C1CH	107	13635H	158	24F4FH	209	16968H
6	20303H	57	11D1CH	108	23636H	159	1504FH	210	26969H
7	10403H	58	21D1DH	109	13736H	160	25050H	211	16A69H
8	20404H	59	11E1DH	110	23737H	161	15150H	212	26A6AH
9	10504H	60	21E1EH	111	13837H	162	25151H	213	16B6AH

**Table 4. Encoded Values for N, M, & K (Part 2 of 3)**

<b>N, M, K</b>	<b>Encoded</b>	<b>N, M, K</b>	<b>Encoded</b>	<b>N, M, K</b>	<b>Encoded</b>	<b>N, M, K</b>	<b>Encoded</b>	<b>N, M, K</b>	<b>Encoded</b>
10	20505H	61	11F1EH	112	23838H	163	15251H	214	26B6BH
11	10605H	62	21F1FH	113	13938H	164	25252H	215	16C6BH
12	20606H	63	1201FH	114	23939H	165	15352H	216	26C6CH
13	10706H	64	22020H	115	13A39H	166	25353H	217	16D6CH
14	20707H	65	12120H	116	23A3AH	167	15453H	218	26D6DH
15	10807H	66	22121H	117	13B3AH	168	25454H	219	16E6DH
16	20808H	67	12221H	118	23B3BH	169	15554H	220	26E6EH
17	10908H	68	22222H	119	13C3BH	170	25555H	221	16F6EH
18	20909H	69	12322H	120	23C3CH	171	15655H	222	26F6FH
19	10A09H	70	22323H	121	13D3CH	172	25656H	223	1706FH
20	20A0AH	71	12423H	122	23D3DH	173	15756H	224	27070H
21	10B0AH	72	22424H	123	13E3DH	174	25757H	225	17170H
22	20B0BH	73	12524H	124	23E3EH	175	15857H	226	27171H
23	10C0BH	74	22525H	125	13F3EH	176	25858H	227	17271H
24	20C0CH	75	12625H	126	23F3FH	177	15958H	228	27272H
25	10D0CH	76	22626H	127	1403FH	178	25959H	229	17372H
26	20D0DH	77	12726H	128	24040H	179	15A59H	230	27373H
27	10E0DH	78	22727H	129	14140H	180	25A5AH	231	17473H
28	20E0EH	79	12827H	130	24141H	181	15B5AH	232	27474H
29	10F0EH	80	22828H	131	14241H	182	25B5BH	233	17574H
30	20F0FH	81	12928H	132	24242H	183	15C5BH	234	27575H
31	1100FH	82	22929H	133	14342H	184	25C5CH	235	17675H
32	21010H	83	12A29H	134	24343H	185	15D5CH	236	27676H
33	11110H	84	22A2AH	135	14443H	186	25D5DH	237	17776H
34	21111H	85	12B2AH	136	24444H	187	15E5DH	238	27777H
35	11211H	86	22B2BH	137	14544H	188	25E5EH	239	17877H
36	21212H	87	12C2BH	138	24545H	189	15F5EH	240	27878H
37	11312H	88	22C2CH	139	14645H	190	25F5FH	241	17978H
38	21313H	89	12D2CH	140	24646H	191	1605FH	242	27979H
39	11413H	90	22D2DH	141	14746H	192	26060H	243	17A79H
40	21414H	91	12E2DH	142	24747H	193	16160H	244	27A7AH
41	11514H	92	22E2EH	143	14847H	194	26161H	245	17B7AH
42	21515H	93	12F2EH	144	24848H	195	16261H	246	27B7BH
43	11615H	94	22F2FH	145	14948H	196	26262H	247	17C7BH
44	21616H	95	1302FH	146	24949H	197	16362H	248	27C7CH
45	11716H	96	23030H	147	14A49H	198	26363H	249	17D7CH
46	21717H	97	13130H	148	24A4AH	199	16463H	250	27D7DH

**Table 4. Encoded Values for N, M, & K (Part 3 of 3)**

N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded	N, M, K	Encoded
47	11817H	98	23131H	149	14B4AH	200	26464H	251	17E7DH
48	21818H	99	13231H	150	24B4BH	201	16564H	252	27E7EH
49	11918H	100	23232H	151	14C4BH	202	26565H	253	17F7EH
50	21919H	101	13332H	152	24C4CH	203	16665H	254	27F7FH
51	11A19H	102	23333H	153	14D4CH	204	26666H	255	1807FH

## CLK\_PLL1\_CTRL

This register enables PLL1 and sets the lock window. Its reset value is 0x1A04, which disables PLL1 and sets a narrow lock window of 0.1 ns. Before the PLL can lock, these bits must be updated with the proper lock window value.

Table 5 lists the lock window settings and the corresponding register values which enable PLL1.

**Table 5. Register Values for the Lock Window Settings**

Lock Window Setting (ns)	Value
10	1065H
2.5	1055H
1.5	1045H
1.0	1035H
0.5	1025H

## CLK\_DERIVE

The CLK\_DERIVE register controls whether the PLLs are bypassed or enabled. Table 6 lists valid settings for CLK\_DERIVE.

**Table 6. Valid Clock Derive Values**

PLL1 (AHB)	PLL2 (SDRAM)	Value
Not Bypassed	Not Bypassed	0010H
Bypassed	Not Bypassed	1010H
Not Bypassed	Bypassed	2010H
Bypassed	Bypassed	3010H

## CLK\_STATUS

CLK\_STATUS is a read-only register that indicates the current state of the PLLs. Table 7 lists valid settings for CLK\_STATUS and their significance.

Bit	Field	Description
0	L1	0 – PLL1 not locked 1 – PLL1 locked
1	L2	0 – PLL2 not locked 1 – PLL2 locked
2	C1	0 – PLL1 did not change lock 1 – PLL1 lock changed
3	C2	0 – PLL2 did not change lock 1 – PLL2 lock changed
4	P1	0 – PLL1 bypassed 1 – PLL1 not bypassed
5	P2	0 – PLL2 bypassed 1 – PLL2 not bypassed
31:6	Reserved	Read as 0

## Conclusion

With the Excalibur embedded processor solutions, you can control the embedded stripe clocking of the Excalibur devices. There are two possible design flows for controlling the the AHB1 and AHB2 clocks in the embedded stripe: you can use the PLL circuit controls defined in the Quartus II software, which use simplified and conservative methods of setting the PLL. Alternatively, you can set the control parameters manually.

## Example Code

The following code exemplifies reprogramming the PLL1 parameters to make the AHB1 clock run at 150 MHz. This example assumes that PLL1 was initially running the AHB1 clock at a different frequency, as set by the Quartus II software. The programming sequence is as follows:

1. Put PLL1 into bypass mode by setting the BP1 bit in the CLK\_DERIVE register.
2. Set the  $N$ ,  $M$ , and  $K$  parameters using the CLK\_PLL1\_xCNT registers.
3. Set the lock window parameter using the CLK\_PLL1\_CTRL register.
4. Take PLL1 out of bypass mode by clearing the BP1 bit in the CLK\_DERIVE register.

5. Poll the L1 lock bit, the C1 lock changed status bit, and the P1 not-bypassed bit in the CLK\_STATUS register.
6. Clear the C1 lock-changed status bit in the CLK\_STATUS register.



EXC\_CLOCK\_CTRL00\_BASE is defined in the header file (**stripe.h**) produced by the wizard.

```
#include "stripe.h"

#define CLOCK_CTRL00_BASE EXC_CLOCK_CTRL00_BASE
#define CLOCK_CTRL00_TYPE (volatile unsigned int *)

#define CLK_PLL1_NCNT(base_addr) (CLOCK_CTRL00_TYPE (base_addr ))
#define CLK_PLL1_MCNT(base_addr) (CLOCK_CTRL00_TYPE (base_addr + 0x4 ))
#define CLK_PLL1_KCNT(base_addr) (CLOCK_CTRL00_TYPE (base_addr + 0x8 ))

#define CLK_PLL1_CTRL(base_addr) (CLOCK_CTRL00_TYPE (base_addr + 0xC ))
#define CLK_PLL1_CTRL_P_MSK 0x0001
#define CLK_PLL1_CTRL_RLOCK_MSK 0x0070

#define CLK_DERIVE(base_addr) (CLOCK_CTRL00_TYPE (base_addr + 0x20 ))
#define CLK_DERIVE_BP1_MSK 0x1000
#define CLK_DERIVE_BP2_MSK 0x2000

#define CLK_STATUS(base_addr) (CLOCK_CTRL00_TYPE (base_addr + 0x24 ))
#define CLK_STATUS_L1_MSK 0x0001
#define CLK_STATUS_L2_MSK 0x0002
#define CLK_STATUS_C1_MSK 0x0004
#define CLK_STATUS_C2_MSK 0x0008
#define CLK_STATUS_P1_MSK 0x0010
#define CLK_STATUS_P2_MSK 0x0020

// Reprogram PLL1 to have AHB1 run at 150 MHz. It is assumed that PLL1
// was initially running AHB1 at a different frequency set by Quartus II.
int main(void)
{
    unsigned int status_mask;
    // Set PLL1 to bypass mode by setting BP1 bit in CLK_DERIVE register
    *CLK_DERIVE(CLOCK_CTRL00_BASE) |= CLK_DERIVE_BP1_MSK;

    // Set N parameter to a 2
    *CLK_PLL1_NCNT(CLOCK_CTRL00_BASE) = 0x20101;

    // Set M parameter to a 12
    *CLK_PLL1_MCNT(CLOCK_CTRL00_BASE) = 0x20606;
```

```

// Set K parameter to a 1
*CLK_PLL1_KCNT(CLOCK_CTRL00_BASE) = 0x40000;

// Clear previous lock window setting
*CLK_PLL1_CTRL(CLOCK_CTRL00_BASE) &= ~CLK_PLL1_CTRL_RLOCK_MSK;

// Set lock window setting to 2.5 ns
*CLK_PLL1_CTRL(CLOCK_CTRL00_BASE) |= 0x0050;

// Set PLL1 to not bypass by clearing BP1 bit in CLK_DERIVE register
*CLK_DERIVE(CLOCK_CTRL00_BASE) &= ~CLK_DERIVE_BP1_MSK;

// Poll that PLL1 locked, lock changed, and is not bypassed
status_mask = CLK_STATUS_L1_MSK | CLK_STATUS_C1_MSK | CLK_STATUS_P1_MSK;
while ( (*CLK_STATUS(CLOCK_CTRL00_BASE) & status_mask) != status_mask );

// Clear lock changed status of PLL1 by setting C1 bit in CLK_STATUS register
*CLK_STATUS(CLOCK_CTRL00_BASE) |= CLK_STATUS_C1_MSK;
}

```

## Revision History

Table 8 shows the document revision history.

<b>Table 8. Revision History</b>	
<b>Date</b>	<b>Description</b>
October 2001	First publication
January 2002	Modification to the PLL programming sequence
July 2002	Modifications to accommodate revised device timings
July 2002	Document re-release



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
<http://www.altera.com>  
**Applications Hotline:**  
(800) 800-EPLD  
**Literature Services:**  
[lit\\_req@altera.com](mailto:lit_req@altera.com)

Copyright © 2002 Altera Corporation. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. All rights reserved.

